



National Technical University of Athens
School of Electrical and Computer Engineering
Division of Signals, Control and Robotics
Computer Vision, Speech Communication
and Signal Processing Group

A study of a Hierarchical Multi-Attentive Framework for Real-Time Single Object 6-D Pose Tracking

DIPLOMA THESIS

Isidoros Marougkas

Supervisor: Petros Maragos
Prof. NTUA

Athens, January 2021



Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών
και Μηχανικών Υπολογιστών
Τομέας Σημάτων, Ελέγχου, Ρομποτικής
Εργαστήριο Όρασης Υπολογιστών,
Επεξεργασίας Σημάτων και Φυσικής Γλώσσας

Μια Μελέτη Ιεραρχικής Δομής Πολλαπλής
Παράλληλης Οπτικής Προσοχής για RGB-D
Παρακολούθηση Πόζας Αντικειμένων σε 6 Βαθμούς
Ελευθερίας, σε πραγματικό χρόνο.

Διπλωματική Εργασία

Ισίδωρος Μαρούγκας

Επιβλέπων : Πέτρος Μαραγκός
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 12η Ιανουαρίου 2021.

.....
Πέτρος Μαραγκός
Καθηγητής Ε.Μ.Π.

.....
Κωσταντίνος Τζαρέστας
Αναπλ.Καθηγητής Ε.Μ.Π.

.....
Χαράλαμπος Ψυλλάκης
Λέκτορας Ε.Μ.Π.

Αθήνα, Ιανουάριος 2021

.....
Isidoros Marougkas
Electrical and Computer Engineer
©2021 - All rights reserved.

This work is copyright and may not be reproduced, stored nor distributed in whole or in part for commercial purposes. Permission is hereby granted to reproduce, store and distribute this work for non-profit, educational and research purposes, provided that the source is acknowledged and the present copyright message is retained. Enquiries regarding use for profit should be directed to the author. The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of the National Technical University of Athens.

Abstract

In this thesis, we present a novel multi-attentional convolutional architecture to tackle the problem of real-time RGB-D 6D object pose tracking of single, known objects. Such a problem poses multiple challenges originating both from the objects' nature and their interaction with their environment, which previous approaches have failed to fully address thus far. The proposed framework encapsulates methods for background clutter and occlusion handling by integrating multiple parallel soft spatial attention modules into a multitask Convolutional Neural Network (CNN) architecture. Moreover, we consider the special geometrical properties of both the object's 3D model and the pose space, and we use a more sophisticated approach for data augmentation for training. The provided experimental results confirm the effectiveness of the proposed multi-attentional architecture, as it improves the State-of-the-Art (SoA) tracking performance by an average score of 34.03% for translation and 40.01% for rotation, when testing on the dataset presented in [83], the most complete dataset designed, up to date, for the problem of RGB-D object tracking. Our algorithm is trained solely in simulation and transferred to real-world scenarios in "zero-shot" fashion. Following, we examine multiple alternative extensions in explicitly modeling both long and short term temporal dynamics in our architectural design. Finally, we expose possible future research directions and practical applications of this problem. This work was published as a Long Paper [82] in the European Conference of Computer Vision (ECCV) 2020 Workshops.

Keywords: Object Pose Tracking, RGB-D, Real-Time, Synthetic Data, Multiple Visual Attention modules, Geodesic Distance, Rotation Representation, Symmetries, Multi-Task, Convolutional Neural Networks, Temporal Dynamics

Περίληψη

Στην παρούσα Διπλωματική Εργασία, παρουσιάζουμε μια καινοτόμα πολυ-επικεντρωμένη συνελικτική αρχιτεκτονική για να αντιμετωπίσουμε το πρόβλημα της παρακολούθησης της 6-διάστατης πόζας μοναδικών, γνωστών αντικειμένων, βασισμένοι σε εικόνες RGB-D, σε πραγματικό χρόνο. Τέτοια προβλήματα δημιουργούν πολλαπλές προκλήσεις ως προς την αντιμετώπισή τους, οι οποίες εξαρτούν τη φύση τους τόσο από μορφή του αντικειμένου, όσο και στις αλληλεπιδράσεις του με το περιβάλλον του, προκλήσεις που προηγούμενες προσεγγίσεις έχουν, προς ώρας, αποτύχει να αντιμετωπίσουν πλήρως. Το προτεινόμενο πλαίσιο ενθυλακώνει μεθόδους για χαώδες παρασκήνιο και διαχείριση επικαλύψεων ενσωματώνοντας πολλαπλές παράλληλες μαλακές χωρικές μονάδες επικέντρωσης της οπτικής προσοχής σε μια αρχιτεκτονική πολυστοχοθετικού Συνελικτικού Νευρωνικού Δικτύου (ΣΝΔ). Επιπλέον, λαμβάνουμε υπόψη τις ειδικές γεωμετρικές ιδιότητες τόσο του 3-διάστατου μοντέλου του αντικειμένου, όσο και του χώρου των ποζών, και χρησιμοποιούμε μια πιο εκλεπτυσμένη προσέγγιση για την επάυξη δεδομένων εκπαίδευσης. Τα πειραματικά αποτελέσματα που παρέχονται επιβεβαιώνουν την αποτελεσματικότητα της προτεινόμενης πολυ-επικεντρωμένης αρχιτεκτονικής, καθώς αυτή βελτιώνει την, μέχρι τώρα, πλέον σύγχρονη απόδοση παρακολούθησης κατά μια μέση διαβάθμιση του 34.03% για μετατόπιση και 40.01% για περιστροφή, όταν η μέθοδος μας δοκιμάζεται στο σύνολο δεδομένων που παρουσιάζεται στη δουλεία των Garon et al. [83], το πιο πλήρως σχεδιασμένο σύνολο δεδομένων, μέχρι τώρα, που αφορά την παρακολούθηση αντικειμένων με τη χρήση RGB-D εικόνων. Ο αλγόριθμός μας εκπαιδεύεται μόνο σε συνθήκες προσομοίωσης και μεταφέρεται για να ελεγχθεί η αποδοτικότητα του σε σενάρια του πραγματικού κόσμου με ‘μηδενική’ εξειδίκευση σε αυτά. Εν συνεχεία, εξετάζουμε πολλαπλές εναλλακτικές επεκτάσεις μοντελοποιώντας με συγκεκριμένο τρόπο τόσο τις μεγάλες όσο και τις μικρού μήκους χρονικές δυναμικές στον αρχιτεκτονικό μας σχεδιασμό. Τέλος, εκθέτουμε δυνατές μελλοντικές κατευθύνσεις έρευνας και και πρακτικές εφαρμογές του προβλήματος. Η εργασία αυτή δημοσιεύτηκε [82] στα Σεμινάρια του Ευρωπαϊκού Συνεδρίου Όρασης Υπολογιστών 2020.

Λέξεις Κλειδιά: Παρακολούθηση Πόζας Αντικειμένων, RGB-D, Πραγματικός Χρόνος, Συνθετικά Δεδομένα, Πολλαπλή Οπτική Προσοχή, Γεωδεδιγμένη Απόσταση, Αναπαράσταση Περιστροφών, Συμμετρίες, Πολυστοχοθεσία, Συνελικτικά Νευρωνικά Δίκτυα, Χρονικές Δυναμικές

Ευχαριστίες

Θα ήθελα, κατα πρώτον, να ευχαριστήσω τον Καθηγητή κ. Πέτρο Μαραγκό για την ευκαιρία που μου έδωσε να αποτελέσω μέρος του εργαστηρίου Όρασης Υπολογιστών, Επικοινωνίας Λόγου και Επεξεργασίας Σημάτων (CVSP), και της επιστημονικής του ομάδας. Η εμπειρία, η πληρότητα και η μεθοδικότητα με την οποία πραγματοποιεί τις διαλέξεις του, καθώς και η στοχευμένη οργάνωση των μαθημάτων του αποτέλεσαν καταλυτικό παράγοντα τόσο στην ανάπτυξη των επιστημονικών μου ενδιαφερόντων όσο και στην αντιληπτική μου επάρκεια για τον χειρισμό ενός τομέα που εξελίσσεται καθημερινά. Από την πρώτη στιγμή της γνωριμίας μας μέχρι και σήμερα υπήρξε ένας πραγματικός μέντορας ο οποίος ενθαρρύνει την ατομική πρωτοβουλία, ανταμοίβει την δομημένη προσπάθεια και είναι πάντοτε διαθέσιμος να ακούσει και να επανατροφοδοτήσει δημιουργικές ιδέες.

Επίσης, θα ήθελα να ευχαριστήσω τους άλλους δύο επιβλέπωντες καθηγητές της εργασίας αυτής, τους κ.κ. Ψυλλάκη Χαράλαμπο και Τζαφέστα Κωνσταντίνο. Ο ζήλος και το μεράκι με το οποίο διδάσκουν ο καθένας τους το αντικείμενο του αποτέλεσαν για μένα ζώσα πηγή έμπνευσης στα φοιτητικά μου χρόνια, καθόρισαν αποφασιστικά τα επιστημονικά μου ενδιαφέροντα, και θέρειψαν την ηθονοσοτροπία μου.

Η διατριβή αυτή δεν θα μπορούσε να ολοκληρωθεί χωρίς τη συμβολή του Δρ. Πέτρου Κούτρα τον οποίο και θα ήθελα, εγχαρδώς να ευχαριστήσω. Πάντοτε ενθαρρυντικός, δεν λυπήθηκε ποτέ ούτε χρόνο ούτε κόπο, αλλά ανέκαθεν επεδείκνυε γνήσιο ενδιαφέρον κι επιμέλεια τόσο στο επιτυχές αποτέλεσμα αυτής της εργασίας, όσο και στην δική μου, προσωπική ανάπτυξη. Δίπλα του θεωρώ πως εξελίχτηκα σημαντικά σαν ερευνητής και πως έχουμε χτίσει μια σχέση βασισμένη στην ειλικρίνεια και την αμοιβαία εκτίμηση.

Θα ήθελα επίσης να ευχαριστήσω τον Υποψήφιο Διδάκτορα κ. Νίκο Κάρδαρη, ο οποίος επίσης επένδυσε σημαντικό μέρος του προσωπικού του χρόνου σε εμένα. Οι συζητήσεις μαζί του υπήρξαν το εφελτήριο αυτής της διπλωματικής εργασίας και η οργανωτικότητα και η οξύμεια των συμβουλών του συνέβαλλαν καθοριστικά στην πραγμάτωση της.

Θα ήθελα να ευχαριστήσω και τον Δρ. Γιώργο Ρετσινά και την Δρ. Γεωργία Χαλβατζάκη, οι οποίοι με έκαναν να δω τις επί μέρους προκλήσεις από πολλαπλές οπτικές γωνίες και μου παρείχαν, πέρα από ένα φιλικό κλίμα, καθοριστικές συμβουλές για την επιτυχία τόσο πειραματικού όσο και του συγγραφικού μέρους του εν λόγω πονήματος.

Ειδική μνεία θα ήθελα να κάνω στην Βασιλική Τασσοπούλου και τον Στάθη Γαλανάκη με τους οποίους οι ατελείωτες ώρες συζητήσεων ενεπλούτισαν την οπτική μου και τις ερευνητικές μου ιδέες.

Φυσικά, θα ήθελα να ευχαριστήσω και όλα τα υπόλοιπα μέλη του Εργαστηρίου για το φιλικό κλίμα, το αυθεντικό ενδιαφέρον και την αμέριστη αποδοχή τους.

Στο σημείο αυτό θα ήθελα να αναφερθώ σε μια ομάδα ανθρώπων με τους οποίους η γνωριμία θεωρώ πως έχει αλλάξει τη ζωή μου για τα καλά. Την σχέση μου με τον Μάριο, τον Κοσμά, την Ίρια, τη Μελίνα, τη Φωτεινή, τη Μαρία, το Σωτήρη, τον Παναγιώτη Μ., την Ισιδώρα, την Δανάη, το Θανάση, τη Μυρτώ, τον Παναγιώτη Π., τη Μαρίνα, τη Λίλιαν, παρά τις πολυπαραγοντικές και τυχαίες της καταβολές, την αξιολογώ ως τον υψηλότερης αξίας απότοκο των φοιτητικών μου χρόνων. Γιατί πέραν από συμφοιτητές, η καθημερινή διαλεκτική με τους οποίους έκανε τις σπουδές μου μια δημιουργική κι ευχάριστη ασχολία, πρόκειται για πραγματικούς φίλους που είναι εκεί και για τα εύκολα και για τα δύσκολα.

Αλλά δεν θα ήθελα σε καμία περίπτωση να λησμονήσω να ευχαριστήσω τον Κυριάκο, τον Γιώργο, τον Σπύρο, τον Μιχάλη, τη Ναυσικά και τη Σωτηρία για την επιμονή με την οποία κατόρθωσαν να διατηρήσουν τη φιλία μας μετά το πέρας του Λυκείου και το γεωγραφικό μας διαχωρισμό, αλλά και το άοκνο ενδιαφέρον τους για την πορεία των σπουδών μου και την προσωπική μου εξέλιξη.

Πάνω από όλα, όμως, ευχαριστώ του γονείς μου, Χρήστο και Ολυμπία, και τον αδερφό μου, Φίλιππο, για την αδιασάλευτη υποστήριξη, κατανόηση και συμπαράσταση με την οποία με ενίσχυσαν καθόλη τη διάρκεια των σπουδών και την εκπόνηση της διπλωματικής μου εργασίας.

Ισιδώρος Μαρούγκας,
Ιανουάριος 2021

Acknowledgements

First of all, I would like to thank Professor Petros Marangos for giving me the opportunity to be part of the Computer Vision Laboratory, Speech Communication and Signal Processing Laboratory (CVSP), and his scientific team. The ingenuity, completeness and methodicalness with which he gives his lectures, as well as the targeted organization of his courses, have been a catalyst for both the development of my scientific interests and my perceptual competence to handle a field that evolves on a daily basis. From the first moment of our acquaintance until today he has been a real mentor who encourages individual initiative, rewards structured effort and is always available to listen and feed back creative ideas.

I would also like to thank the other two supervisors of this work, Mr.Charalambos Psyllakis and Mr.Constantinos Tzafestas. The zeal and passion with which each of them teaches his subject became a living source of inspiration for me in my student years, determined my scientific interests decisively, and undermined my moral cast of mind.

This dissertation could not have been completed without the contribution of Dr.Petros Koutras, whom I would like to thank from the bottom of my heart. Always encouraging, he never skimmed time or effort, but always showed genuine interest and diligence both in the successful outcome of this work and in my own personal development. Next to him, I believe that I have been developed significantly as a researcher and that we have built a relationship based on honesty and mutual respect.

I would also like to thank the PhD Candidate Mr.Nikos Kardaris, who also invested a significant part of his time in me. Discussions with him have been the starting point for this diploma thesis and the organization and acumen of his advice have contributed significantly to its realization.

I would also like to thank Dr. George Retsinas and Dr. Georgia Chalvatzaki, who made me see the individual challenges from multiple perspectives and provided me, in addition to a friendly atmosphere, with decisive advice on the success of both the experimental and the writing part of this work.

I would like to make a special mention of Vasiliki Tassopoulou and Stathis Galanakis, with whom the endless hours of discussions enriched my perspective and my research ideas.

Of course, I would also like to thank all the other members of the Laboratory for their friendly atmosphere, genuine interest and undivided acceptance.

At this point I would like to refer to a group of people with whom I think my acquaintance has changed my life for the better. My relationship with Marios, Kosmas, Iria, Melina, Fotini, Maria, Sotiris, Panagiotis M., Isidora, Danae, Thanasis, Myrto, Panagiotis P., Marina, Lilian, despite its multifactorial and random origins, I consider it to be the highest prize of my student years. Because apart from fellow students, the daily dialectic with which made my studies a creative and enjoyable pastime, these are real friends who are there for both at fun fun and difficult times.

I would never want to forget to thank Kyriakos, Giorgos, Spyros, Michalis, Nafsika and Sotiria for the persistence with which they managed to maintain our friendship after high school albeit our geographical division, but also their genuine interest in the course of my studies.

Above all, however, I thank my parents, Christos and Olympia, and my brother, Filippos, for the unwavering support and understanding with which I have been strengthened throughout my studies and the completion of my dissertation.

Isidoros Marougkas,
January 2021

Η Διπλωματική Εργασία αυτή είναι αφιερωμένη στην μητέρα μου Ολυμπία, τον πατέρα μου Χρήστο, και τον αδερφό μου Φίλιππο, που διαχρονικά επιδεικνύουν αγάπη, εμπιστοσύνη και ανιδιοτελή στήριξη στις αποφάσεις μου.

In memory of the late Dr. Andreas Hilboll, the kind-hearted, joyful and aspiring young researcher who lost the battle with COVID-19 in 25/03/2020 in Heraklion, Crete.

Κεφάλαιο 1

Εκτεταμένη Περίληψη

Στην παρούσα Διπλωματική Εργασία, παρουσιάζουμε μια καινοτόμα πολυ-επικεντρωμένη συνελικτική αρχιτεκτονική για να αντιμετωπίσουμε το πρόβλημα της παρακολούθησης της 6-διάστατης πόζας μοναδικών, γνωστών αντικειμένων, βασισμένοι σε εικόνες RGB-D, σε πραγματικό χρόνο. Τέτοια προβλήματα δημιουργούν πολλαπλές προκλήσεις ως προς την αντιμετώπισή τους, οι οποίες εξαρτούν τη φύση τους τόσο από μορφή του αντικειμένου, όσο και στις αλληλεπιδράσεις του με το περιβάλλον του, προκλήσεις που προηγούμενες προσεγγίσεις έχουν, προς ώρας, αποτύχει να αντιμετωπίσουν πλήρως. Το προτεινόμενο πλαίσιο ενθυλακώνει μεθόδους για χαώδες παρασκήνιο και διαχείριση επικαλύψεων ενσωματώνοντας πολλαπλές παράλληλες μαλακές χωρικές μονάδες επικέντρωσης της οπτικής προσοχής σε μια αρχιτεκτονική πολυστοχοθετικού Συνελικτικού Νευρωνικού Δικτύου (ΣΝΔ). Επιπλέον, λαμβάνουμε υπόψη τις ειδικές γεωμετρικές ιδιότητες τόσο του 3-διάστατου μοντέλου του αντικειμένου, όσο και του χώρου των ποζών, και χρησιμοποιούμε μια πιο εκλεπτυσμένη προσέγγιση για την επάυξηση δεδομένων εκπαίδευσης. Τα πειραματικά αποτελέσματα που παρέχονται επιβεβαιώνουν την αποτελεσματικότητα της προτεινόμενης πολυ-επικεντρωμένης αρχιτεκτονικής, καθώς αυτή βελτιώνει την, μέχρι τώρα, πλέον σύγχρονη απόδοση παρακολούθησης κατά μια μέση διαβάθμιση του 34.03% για μετατόπιση και 40.01% για περιστροφή, όταν η μέθοδος μας δοκιμάζεται στο σύνολο δεδομένων που παρουσιάζεται στη δουλειά των Garon et al. [83], το πιο πλήρως σχεδιασμένο σύνολο δεδομένων, μέχρι τώρα, που αφορά την παρακολούθηση αντικειμένων με τη χρήση RGB-D εικόνων. Ο αλγόριθμός μας εκπαιδεύεται μόνο σε συνθήκες προσομοίωσης και μεταφέρεται για να ελεγχθεί η αποδοτικότητα του σε σενάρια του πραγματικού κόσμου με 'μηδενική' εξειδίκευση σε αυτά. Εν συνεχεία, εξετάζουμε πολλαπλές εναλλακτικές επεκτάσεις μοντελοποιώντας με συγκεκριμένο τρόπο τόσο τις μεγάλες όσο και τις μικρού μήκους χρονικές δυναμικές στον αρχιτεκτονικό μας σχεδιασμό. Τέλος, εκθέτουμε δυνατές μελλοντικές κατευθύνσεις έρευνας και και πρακτικές εφαρμογές του προβλήματος. Η εργασία αυτή δημοσιεύτηκε [82] στα Σεμινάρια του Ευρωπαϊκού Συνεδρίου Όρασης Υπολογιστών 2020.

Η εύρωστη, ακριβής και ταχεία εκτίμηση και παρακολούθηση πόζας των 3-διάστατων μετατοπίσεων και περιστροφών των αντικειμένων έχει αποτελέσει ζήτημα εμβρυθούς έρευνας ανα τα χρόνια. Οι εφαρμογές αυτού του προβλήματος εκτίμησης μπορούν να βρεθούν στην Ρομποτική, την Αυτόματη Περιήγηση, την Επαυξημένη Πραγματικότητα κλπ. Παρόλο που η κοινότητα της Όρασης Υπολογιστών έχει επισταμμένως μελετήσει το πρόβλημα της εκτίμησης και παρακολούθησης πόζας των αντικειμένων για δεκαετίες, η πρόσφατη

διάχυση των φτηνών και αξιόπιστων αισθητήρων εικόνων τύπου RGB και Βάθους, σαν το Kinect, μαζί με τις εξελίξεις στην Βαθιά Μάθηση (BM) και, ειδικότερα, τη χρήση των Συνελκτικτικών Νευρωνικών Δικτύων (ΣΝΔ) ως τους πιο σύγχρονους εκφραστής χαρακτηριστικών εικόνας, έχουν οδηγήσει σε μια νέα εποχή την έρευνα και την επανεξέταση αρκετών γνωστών προβλημάτων, με το γενικότερο σκοπό της γενίκευσης πολλαπλών στόχων. Τα ΣΝΔ έχουν επιτύχει ριζοσπαστικά αποτελέσματα σε 2-διάστατα προβλήματα όπως η Ταξινόμηση Αντικειμένων, η Ανίχνευση Αντικειμένων και η Κατάτμηση. Γι αυτό η ερευνητική κοινότητα βρίσκει όλο και πιο δελεαστική τη χρήση τους στους πιο προκλητικούς 3-διάστατους στόχους.

Οι εγγενείς προκλήσεις της Εκτίμησης Πόζας των Αντικειμένων από RGB-D ρεύματα περιλαμβάνουν χαώδες παρασκήνιο, επικαλύψεις (στατικές, από άλλα αντικείμενα που παρίστανται στη σκηνή, και δυναμικές, εξαιτίας πολλαπλών αλληλεπιδράσεων με τον ανθρώπινο χρήστη), ποικιλότητες φωτισμού, θόρυβο αισθητήρων, θόλωμα εικόνας (εξαιτίας ταχείας κίνησης του αντικειμένου) και αλλαγές στην εμφάνιση καθώς η πόζα του αντικειμένου μεταβάλλεται. Επιπλέον, πρέπει να συλλογιστούμε τις ασάφειες πόζας, που αποτελούν άμεση συνέπεια της γεωμετρίας του αντικειμένου, την πρόκληση της ορθής παραμετροποίησης των περιστροφών και τις αναπόφευκτες δυσκολίες που αντιμετωπίζει κάθε απόπειρα να εξαχθεί πληροφορία για την 3-διάστατη γεωμετρία της σκηνής από 2-διάστατες προβολές της στο πεδίο της εικόνας.

Φορμαλισμός του Προβλήματος

Το πρόβλημά μας συνίσταται στην εκτίμηση της πόζας του αντικειμένου P , η οποία συνήθως περιγράφεται ως ένας συμπαγής 3-διάστατος μετασχηματισμός ως προς ένα σταθερό πλαίσιο αναφοράς, για παράδειγμα ένα στοιχείο της Ειδικής Ευκλείδειας Ομάδας Lie στις 3 διαστάσεις, την $(\text{EEO}L3)$. Η πόζα μπορεί να αποδομηθεί σε 2 συνιστώντα στοιχεία της: έναν πίνακα περιστροφών R , ο οποίος αποτελεί στοιχείο της Ειδικής Ορθογώνιας Ομάδας Lie στις 3 διαστάσεις $(\text{EEO}L(3))$ κι ένα διάνυσμα μετατόπισης $\mathbf{t} \in \mathbb{R}^3$. Παρολαυτά, ο Brégier [11] πρότείνει έναν ευρύτερο ορισμό της πόζας του αντικειμένου, η οποία μπορεί να θεωρηθεί ως μια οικογένεια συμπαγών μετασχηματισμών, ώστε να συμπεριληφθεί στη μοντελοποίησή της η ασάφεια που προκαλείται από δυνατές περιστροφικές συμμετρίες, που συμβολίζονται $G \in (\text{EEO}L(3))$. Αξιοποιούμε αυτόν τον επαυξημένο μαθηματικό ορισμό για να εισάγουμε μια χαλάρωση του ορισμού του χώρου πόζων \mathcal{C} :

$$\mathcal{C} = \left\{ P \mid P = \left[\begin{array}{c|c} R \cdot G & \mathbf{t} \\ \hline \mathbf{0}^T & 1 \end{array} \right], \mathbf{t} \in \mathbb{R}^3, R \in \text{SO}(3), G \in \text{SO}(3) \right\}. \quad (1.1)$$

Επί παραδείγματι, όπως αναφέρεται στο [11], η περιγραφή της πόζας ενός αντικειμένου με σφαιρική συμμετρία απαιτεί μόνον 3 από τις συνολικά 12 παραμέτρους: εκείνες που αντιστοιχούν στην μετατόπιση $t_{x,y,z}$, καθώς ο G μπορεί να είναι οποιοδήποτε στοιχείο του $\text{EEO}L(3)$ και το αποτύπωμα του σχήματος του αντικειμένου στην εικόνα βάθους να διατηρείται το ίδιο.

Περιγραφή Βασικής Προσέγγισης

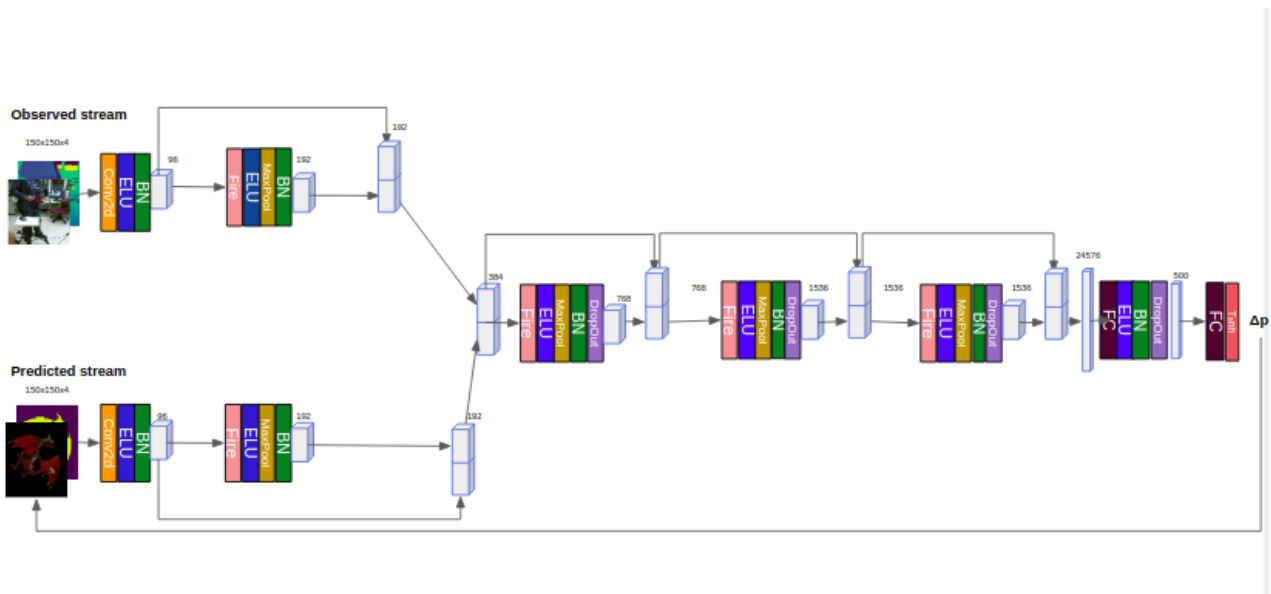


Figure 1.1: Οπτική περιγραφή της βασικής αρχιτεκτονικής ΣΝΔ των Garon et al.[83] για την οπτική παρακολούθηση της πόζας μοναδικών, γνωστών αντικειμένων στο χρόνο.

Περιγραφή της Αρχιτεκτονικής

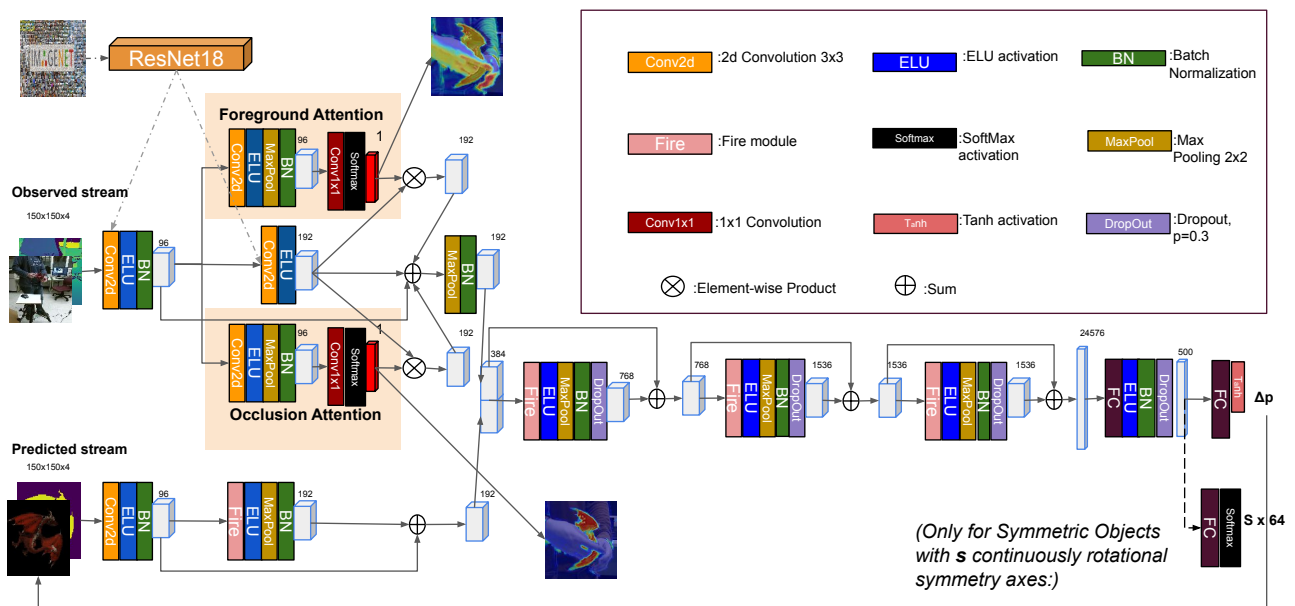


Figure 1.2: Οπτική περιγραφή της προτεινόμενης αρχιτεκτονικής ΣΝΔ για την οπτική παρακολούθηση της πόζας μοναδικών, γνωστών αντικειμένων στο χρόνο.

Η προτεινόμενη αρχιτεκτονική απεικονίζεται στην εικόνα 1.2. Οι είσοδοι του ΣΝΔ μας είναι ένα ζευγάρι RGB-D εικόνων μεγέθους 150×150 : $I(t)$, $\hat{I}(t)$ (με το $I(t)$ να είναι το ‘Παρατηρούμενο’ και το $\hat{I}(t)$ να είναι το ‘Προβλεπόμενο’ ζεύγος εικόνων) και παλινδρομούμε μια αναπαράσταση πόζας εξόδου $\Delta \mathbf{p} \in \mathbb{R}^9$, με 3 παραμέτρους για μετατόπιση ($\hat{t}_{x,y,z} \in [-1, 1]$) και 6 για περιστροφή.

Τα πρώτα 2 επίπεδα του ‘Παρατηρούμενου’ αυλακίου αρχικοποιούνται με τα βάρη του

ResNet18[37], το οποίο έχει πρότερα εκπαιδευτεί στο σύνολο δεδομένων ImageNet[68], μια βάση δεδομένων για ταξινόμηση εικόνων που αριθμεί πάνω από ένα εκατομμύριο δείγματα και αποτελεί διεθνές σημείο αναφοράς στην ερευνητική κοινότητα για την ευρύτητα της, προκειμένου να αμβλύνουμε το κενό μεταξύ του συνθετικού συνόλου δεδομένων εκπαίδευσης και της βάσης δεδομένων ελέγχου, η οποία έχει εξαχθεί από τον πραγματικό κόσμο, σύμφωνα με τις υποδείξεις των Hintenko et al. [42]. Εν αντιθέσει με τις υποδείξεις τους, εμείς βρίσκουμε ως προτιμότερο αντί να ‘παγώσουμε’ τα βάρη των 2 πρώτων νευρωνικών επιπέδων που αναθέτουμε στα αντίστοιχα δικά μας, να τα αφήσουμε ελεύθερα να εκπαιδευτούν περαιτέρω επί του συνθετικού μας συνόλου δεδομένων, κατά την διαδικασία εκπαίδευσης του ΣΝΔ. Ο λόγος που επικαλούμαστε για την απόφασή μας αυτή είναι το γεγονός πως ο απώτερος σκοπός της εργασίας μας είναι να παρακολουθήσουμε την πόζα μοναδικών και a-priori γνωστών αντικειμένων στα οποία κι εκπαιδεύεται το δίκτυό μας κατ’ αποκλειστικότητα κι όχι η γενίκευση της απόδοσης του σε νέα, με τα οποία δεν έχει έρθει ποτέ σε επαφή. Έτσι, συνειδητά υπερταυριάζουμε τα βάρη του δικτύου στα συγκεκριμένα διακριτά χαρακτηριστικά εμφάνισης και σχήματος του εκάστου αντικειμένου κι αυτό βοηθά τον παρακολουθητή μας να επικεντρωθεί μόνο στην αναγνώριση των χαρακτηριστικών διαφοράς μεταξύ των δύο RGB-D εικόνων που παρέχουν τη διαφορά πόζας μεταξύ τους. Στην έξοδο του δεύτερου επιπέδου του καναλιού ‘Παρατήρησης’, εφαρμόζουμε μια Μονάδα Χωρική Μαλακής Προσοχής με στόχο να εξάγει τα χαρακτηριστικά παρασκηνίου από τα βαθιά χαρακτηριστικά της εικόνας. Ομοίως πράττουμε και με μια δεύτερη Μονάδα Μαλακής Χωρικής Οπτικής Προσοχής η οποία αφορά και μαθαίνει στον χειρισμό των επικαλύψεων του αντικειμένου. Τα αποτελέσματά του, μαζί με τις αντίστοιχες εξόδους των χαρτών χαρακτηριστικών του δεύτερου επιπέδου αθροίζονται για να παράγουν τον τελικό χάρτη χαρακτηριστικών. Επιπλέον, στο άθροισμα αυτό πρέπει να συνυπολογίσουμε και την προσθήκη του χάρτη χαρακτηριστικών που προέρχεται από το πρώτο συνελικτικό επίπεδο, διαμέσου μια διαφορικής σύνδεσης [37]. Ως επόμενο βήμα, ενώνουμε τους χάρτες χαρακτηριστικών των δύο καναλιών τοποθετώντας τους τον ένα πλάι στον άλλον και περνάμε το συνισταμένο τους αποτέλεσμα διαμέσου τριών επομένων ακολουθιακών μονάδων ‘Fire’, όλων διασυνδεδεμένων, το καθένα με το προηγούμενό του, μέσω διαφορικών συνδέσεων [37].

Διαχείριση Παρασκηνίου κι Επικαλύψεων

Μετά το πρώτο ‘Παρατηρήσιμο’ επίπεδο ‘Fire’, το μοντέλο μας δημιουργεί έναν Χάρτη Βαρών Χωρικής Οπτικής Προσοχής χρησιμοποιώντας ένα συνελικτικό επίπεδο αφιερωμένο στο χειρισμό επικαλύψεων και, αντίστοιχα, ένα δεύτερο παράλληλο, όμοιο, στην εξαγωγή παρασκηνίου, αντιστοίχως. Αυτό ακολουθείται από ένα συνελικτικό επίπεδο με φίλτρα με πηλύνες μεγέθους 1×1 το οποίο στοχεύει να συμπίεσει τον χάρτη χαρακτηριστικών στον έν λόγω μονοκάναλο Χάρτη Βαρών (αφού φρντίσουμε για την κανονικοποίηση του με τη χρήση μιας συνάρτησης Softmax προκειμένου να του προσδώσουμε ιδιότητες χωρικών πιθανοτήτων). Ο στόχος μας είναι να εκμαιεύσουμε μαλακές μάσκες κατάτμησης παρασκηνίου κι επικαλύψεων, μέσω της εκπαίδευσης του μονάδων αυτών του δικτύου στην μίμηση σκληρών, δυαδικών μασκών (τις οποίες διατηρούμε από το στάδιο επάυξης δεδομένων, επαυξάνοντας τον αντικειμενοκεντρική εικόνα με ένα τυχαίο δείγμα παρασκηνίου κι επικαλυπτή) ούτως ώστε να έχουμε τις εκτιμήσεις των μασκών αυτών κατά την

πραγματική λειτουργία δοκιμής του παρακολουθητή μας. Για το σκοπό αυτό, προσθέτουμε στην συνολική συνάρτηση απώλειας, με βάση την οποία εκπαιδεύεται το δίκτυό μας, και δύο επιπλέον συναρτήσεις Δυαδικής Δι-Εντροπίας. Επιχειρηματοποιούμε προς όφελος αυτής της σχεδιαστικής επιλογής μας, της χρήσης, δηλαδή, δύο παραλλήλων μονάδων Οπτικής Προσοχής, λέγοντας πως, κατόπιν ενδελεχούς πειραματισμού, βρήκαμε πως το να αναθέτεις έναν ξεκάθαρο και δομημένο στόχο προς εκμάθηση σε κάθε τέτοια μονάδα λειτουργεί ευεργετικά για την απόδοση και των δύο, αντί να βασιστείς στην ελπίδα μία και μοναδική μονάδα Οπτικής Προσοχής να επιτελέσει επιτυχώς και τα δύο καθήκοντα και να επιλύσει και τις δύο προκλήσεις ολιστικά. Για λεπτομέρειες παραπέμπουμε στην Αναλογιστική μας μελέτη του Κεφαλαίου 7.

$$Att_{Foregr} = Softmax^{(2D)}(Conv_{1 \times 1}(ELU(Conv2D^{(Foregr)}(X_{Obs}^{(1)})))) \quad (1.2)$$

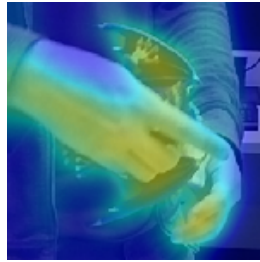
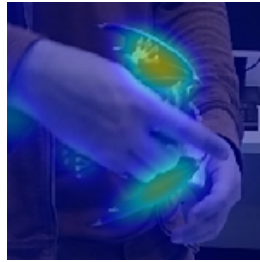


Figure 1.3: Ένας Χάρτης Βαρών Χωρικής Προσοχής αφιερωμένος στην Εξαγωγή Προσκήνιου. Τα τονισμένα εικονοστοιχεία είναι εκείνα που αφορούν τα κινούμενα μέρη της σχήνης.

$$Att_{Occl} = Softmax^{(2D)}(Conv_{1 \times 1}(ELU(Conv2D^{(Occl)}(X_{Obs}^{(1)})))) \quad (1.3)$$



Σχήμα 1.4: Ένας Χάρτης Βαρών Χωρικής Προσοχής αφιερωμένος στο Χειρισμό Επικαλύψεων. Τα τονισμένα εικονοστοιχεία είναι εκείνα που αντιστοιχούν μόνο στο αντικείμενο ενδιαφέροντος.

Συνολική Συνατηση Απώλειας και Αναπαράσταση Περιστροφών

Από μια καθαρά μαθηματική οπτική, η άμεση παλινδρόμηση των παραμέτρων πόζας, που επιτελούν οι Garon et al.[83], χρησιμοποιώντας μια Ευκλείδεια μετρική της απώλειας του δικτύου είναι ξεκάθαρα υποβέλτιστη, χρησιμοποιώντας την παρακάτω παρατήρηση: ενώ η μία συνιστώσα της πόζας του αντικειμένου, η 3-διάστατη μετατόπισή του, ανήκει στον Ευκλείδειο χώρο, ούτως ειπείν αύξηση της αλγεβρικής τιμής της συνεπάγεται ένα γεωμετρικό αντίκτυπο αύξησης, αναλογικά, και της γεωμετρικής της απεικόνισης, κάτι τέτοιο δεν συμβαίνει για την έταιρη συνιστώσα: αυτή των 3-διάστατων περιστροφών. Και τούτο διότι οι 3-διάστατες περιστροφές κείνται επί της μη-γραμμικής πολλαπλότητας του $EOOL(3)$. Γι αυτό γεννάται άμεσα η απαίτηση της μοντελοποίησης της συνιστώσας των

περιστροφών. σε ότι αφορά τη συνάρτηση Απώλειας, χρησιμοποιώντας μια Γεωδεσική μετρική απόστασης (για λεπτομέρειες αποταθείτε στον Τομέα 2.11) στο $EOOL(3)$, δηλαδή χρησιμοποιώντας μια ελάχιστη μετρική μήκους που ορίζεται επί της πολλαπλότητας αυτής, και μετρημένη σε ακτίνια. Ένα Γεωδεσικό μονοπάτι (που συνεπάγεται φυσικά και την ομόνυμη απόσταση) είναι το μονοπάτι ελαχίστου μήκους που συνδέει 2 στοιχεία του $EOOL(3)$: $\Delta\hat{R}, \Delta R$. Με σκοπό την ελαχιστοποίηση των λαθών περιστροφής λόγω ελλειπούς μοντεολοποίησης αυτής της συνιστώσας και των ασάφειών που κάτι τέτοιο επιφέρει, επιλέγουμε να εντάξουμε στο συνολικό μας πλαίσιο την 6-διάστατη αναπαράσταση περιστροφών που πρώτοι εισήγαγαν οι Zhou et al. στο [171]:

$$\begin{aligned}\Delta\mathbf{R}_x &= N(\Delta\mathbf{r}_x) \\ \Delta\mathbf{R}_y &= N[\Delta\mathbf{r}_y - (\Delta\mathbf{R}_x^T \cdot \mathbf{r}_y) \cdot \Delta\mathbf{R}_x] \\ \Delta\mathbf{R}_z &= \Delta\mathbf{R}_x \times \Delta\mathbf{R}_y\end{aligned}\tag{1.4}$$

όπου $\Delta\mathbf{R}_{x/y/z} \in \mathbb{R}^3$, $\mathcal{N}(\cdot) = \frac{(\cdot)}{\|(\cdot)\|}$ είναι η συνάρτηση κανονικοποίησης. Επιπροσθέτως, όπως έχει ήδη συζητηθεί από τον Brégier στο [11], κάθε 3-διάστατη γωνία περιστροφής έχει ένα διαφορετικό οπτικό αποτύπωμα σε σχέση με τον αντίστοιχο άξονα. Έτσι, πολλαπλασιάζουμε και τους δύο πίνακες στροφής με ένα (σχεδόν) διαγώνιο Αδρανεϊακό Τένσορα Λ , υπολογισμένο επί της βεβαρυμένης επιφανείας του αντικειμένου ενδιαφέροντος και σε συνάρτηση με το κέντρο μάζας του, ούτως ώστε να αναθέσει μια διαφορετική στάθμιση σε κάθε περιστροφική συνιστώσα. Σημειώνουμε. εδώ, πως από τη στιγμή που επιθυμούμε αυτό το γινόμενο πινάκων να συνεχίζει να κείται επί του $EOOL(3)$, εφαρμόζουμε μια κανονικοποίηση Gramm-Schmidt στον Αδρανεϊακό Τένσορα Λ , προτού αυτός πολλαπλασιαστεί εκ δεξιών με τον κάθε πίνακα περιστροφής. Τελικώς, σταθμίζουμε τόσο την συνάρτηση απώλειας μετατόπισης όσο και την αντίστοιχη περιστροφική με τη χρήση ενός ζεύγους εκπαιδευόμενων βαρών $\mathbf{v} = [v_1, v_2]^T$. Τα βάρη αυτά εκπαιδεύονται μαζί με το υπόλοιπο σύνολο βαρών του ΣΝΔ δια μέσου ενός Αλγορίθμου βελτιστοποίησης παραμέτρων Πλεόν Απότομης Κατάβασης, μια ιδέα που έχει πρωτοπραθεί από τον Kendall στην δημοσίευσή του [59].

Χειρισμός Συμμετρικών Αντικειμένων

Στην ειδική περίπτωση των συμμετρικών αντικειμένων, αποδομούμε τις ασάφειες που υπεισέρχονται στην εκτίμησης πόζας λόγω της ιδιότητας αυτής (δηλαδή της συμμετρίας) από τον κορμό της εκτίμησης περιστροφής διαμέσου παλινδρόμησης. Παλινδρομούμε μια ξεχωριστή τριάδα γωνιών Euler που αφορούν παραμέτρους συμμετρίας $\hat{\mathbf{g}} \in \mathbb{R}^3$ που μετατρέπεται σε έναν εκτιμώμενο πίνακα περιστροφών \hat{G} , ο οποίος πολλαπλασιάζεται, από δεξιά, με τον $\Delta\hat{R}$ πριν σταθμιστεί υπό τις παραμέτρους του Αδρανεϊακού Τένσορα $\Lambda_{(G.S.)}$. Χρησιμοποιούμε ένα κυλινδρικό μοντέλο από ένα κουτί με μπισκότα ως παράδειγμα της συμμετρικής περίπτωσης αντικειμένων, το σχήμα του οποίου μας προικοδοτεί με έναν, μοναδικό άξονα περιστροφική συμμετρίας. Συμπερασματικά, εκτιμούμε μια μοναδική παράμετρο συμμετρίας, αυτή του αντικειμενοπαγούς άξονα z. Προτού αυτή η παράμετρος μετατραπεί σε πλήρη πίνακα περιστροφών, η παράμετρος αυτή περνάει διαμέσου μιας συνάρτησης αντιστρόφου εφαπτομένης και πολλαπλασιάζεται με π , με στόχο να περιορίσει τις τιμές παλινδρόμησης.

Ως αποτέλεσμα, η συνολική συνάρτηση απώλειας της παρακολούθησης πόζας ορίζεται ως:

$$L_{Track}(\Delta\hat{\mathbb{P}}, \Delta\mathbb{P}) = e^{(-v_1)} \cdot MSE[(\Delta\hat{\mathbf{t}}, \Delta\mathbf{t})] + v_1 + v_2 + e^{(-v_2)} \cdot \arccos\left(\frac{\text{tr}((\Delta\hat{R} \cdot \hat{G} \cdot \Lambda_{(G.S.)})^T \cdot (\Delta R \cdot \Lambda_{(G.S.)})) - 1}{2}\right) \quad (1.5)$$

Κάνοντας χρήση ενός παρόμοιου, εξωτερικού αυτή τη φορά, πολυστοχοθετικού εκπαιδευόμενου σχήματος βαρών ($\mathbf{s} = [s_1, s_2, s_3]^T$), όπως στην περίπτωση της εξίσωσης(6.36), συνδυάζουμε τον κύριο στόχο εκμάθησης, την χρονική παρακολούθηση πόζας, με τις δύο βοηθητικές συναρτήσεις απώλειας: την διαχείριση χαώδους παρασκηνίου και των επικαλύψεων.

$$Loss = e^{(-s_1)} \cdot L_{Track} + e^{(-s_2)} \cdot L_{Unoccl} + e^{(-s_3)} \cdot L_{Foregr} + s_1 + s_2 + s_3 \quad (1.6)$$

Η ελαχιστοποίηση της συνολικής συνάρτησης απώλειας παρακολούθησης αντικειμένου ως προς το πίνακα συμμετριών \hat{G}^* (δείτε τη δημοσίευση του Brégier [11]) δεν επιβάλλει με συγκεκριμένο τρόπο έναν περιορισμό καθολικής λύσης, κάτι που μας δίνει την ελευθερία να επιλέξουμε το πως το δίκτυο θα μάθει τις συνεχόμενα περιστροφικές παραμέτρους συμμετρίας. Η επιλογή του φάσματος αυτού εκτείνεται από το να εκπαιδεύσει κανείς μια έκδοση που απαιτούμε μοναδική λύση για την παράμετρο, την οποία, φυσικά, εκπαιδεύουμε εν παραλλήλω με τις υπόλοιπες παραμέτρους του δικτύου, κατά τη φάση της εκπαίδευσης και την κρατάμε παγωμένη κατά την φάση του εμπρόσθιου περάσματος χρήσης του παρακολουθητή, μέχρι την προσθήκη ενός εξτρά σταδίου παλινδρόμησης από πλήρως συνδεδεμένους νευρώνες πάνω στο πρώτο γραμμικό επίπεδο του παρακολουθητή. Το επίπεδο αυτό θα είναι υπεύθυνο για την παλινδρόμηση των παραμέτρων συνεχούς περιστροφικής συμμετρίας, αποδίδοντας μιας διαφορετική τιμή ανά ζεύγος ποζών, με σκοπό να παρέχουμε τον βέλτιστο δυνατό πίνακα συμμετριών. Μια ενδιάμεση προσέγγιση αποτελεί η ενδιάμεση εκμάθηση μιας συστάδας από τέτοιες παραμέτρους, συνεχώς περιστροφικής συμμετρίας (ανα περιστροφική συνιστώσα) και έπειτα, ή η χρήση του μέσου όρου των παγωμένων, συγκεκριμένων αυτών παραμέτρων ή η on-line επιλογή της πλέον καταλλήλου παραμέτρου, αναλόγως του ζεύγους ποζών που έχουμε στην είσοδο. Μια επιλογή που θα γίνει με τη χρήση ενός επιπέδου κατηγοριοποίησης επί του πρώτου πλήρως συνδεδεμένου επιπέδου του Νευρωνικού παρακολουθητή μας. Η τελευταία αυτή προσέγγιση είναι και εκείνη που τελικά προτείνεται ως βέλτιστη μεταξύ των τριών. Με μια αξιοπρόσεκτη παρατήρηση: το επίπεδο κατηγοριοποίησης πρέπει να έχει ένα ευρύ πεδίο επιλογών για παραμέτρους συνεχούς περιστροφικής συμμετρίας, ώστε να αξιοποιήσει πλήρως την επίδρασή τους. Για το σκοπό αυτό, επιβάλλουμε στις παραμέτρους της κάθε συστάδας κάθε συνιστώσας συνεχούς περιστροφικής συμμετρίας να είναι όσο το δυνατόν πιο διεσπαρμένες στο χώρο τους γίνεται, άρα και πιο ομοιόμορφες. Αυτό είναι κάτι που πρέπει να ενσωματωθεί στη Συνάρτηση Απώλειας που ελαχιστοποιείται κατά το στάδιο εκπαίδευσης το Συνελικτικού Δικτύου.

Ως αποτέλεσμα, για αντικείμενα με συνεχώς περιστροφική συμμετρία η συνάρτηση

απωλείας γίνεται:

$$Loss^{(Symm)} = Loss + e^{(-s_4)} \left(\frac{1}{B} \sum_{b=1}^B \frac{1}{\xi_b} \right) + s_4, \quad \text{με} \quad (1.7)$$

$$\xi_b = \frac{1}{B_2(B_2 - 1)} \sum_{j=1}^{B_2} \sum_{k \neq j} d_{Rot}^{(Geod)}(\hat{G}_k, \hat{G}_j), \quad (1.8)$$

Ο επιπλέον όρος που προτίθεται στην πολυστοχοθετική συνάρτηση απώλειας είναι μια ποινή που εκπαιδεύει αντιθετικά το(τα) επίπεδο(α) ταξινόμησης που επιλέγει(-ουν) την(τις) κατάλληλη(-ες) παράμετρο(-ους) συνεχώς περιστροφικής συμμετρίας σε κάθε χρονικό βήμα χρήσης του παρακολουθητή. Αυτό που κάνει είναι να ενθαρρύνει τις γεωδαισικές περιστροφικές αποστάσεις μεταξύ όλων των ζευγών παραμέτρων στη συστάδα να είναι οι μέγιστες δυνατές και, γι αυτό το λόγο, τελικώς να συγκλίνουν σε μια κατανομή που να είναι όσο το δυνατόν πιο ομοιόμορφη. Στην εργασία αυτή εκπαιδεύουμε $B_2=64$ τέτοιες παραμέτρους για κάθε συνιστώσα συνεχούς περιστροφικής συμμετρίας. όλες οι παράμετροι συμμετρίας αρχικοποιούνται μέσω τυχαίας δειγματοληψίας από μια ομοιόμορφη κατανομή $U[-180^\circ, 180^\circ]$.

Διακριτές Περιστροφικές Συμμετρίες:

Όπως έχουμε προαναφέρει, υπάρχει μια δεύτερη περίπτωση συμμετρίας: αυτή των **Διακριτών/Εξαρτόμενων από τη γωνία θέασης συμμετριων αντικειμένων**, παραθέτουμε την ακόλουθη πορεία σκέψης που οδηγεί στον κάτωθι ευριστικό αλγόριθμο.

Ο προσεκτικός αναγνώστης δικαιολογημένα θα διατύπωνε την επόμενη ένσταση: Γιατί χρειαζόμαστε να λύσουμε με το χέρι αυτή τη διαφορά που προκαλείται από την οικογένεια διακριτών συμμετριών? Δεν αρκεί η ύπαρξη του βρόχου ανατροφοδότησης για να αποδοθεί μια ισχυρή προτέρα πληροφορία η οποία θα την οδηγήσει να επιλέξει τον πλέον κατάλληλο προσανατολισμό? Εξάλλου, οι περιστροφικές διαφορές μεταξύ των δύο εκτιμήσεων είναι καταπληκτικές και θα μπορούσαν να γίνουν ένα από τα πρώτα πράγματα κάποιος θα ανέμενε ο αλγόριθμος να διορθώσει.

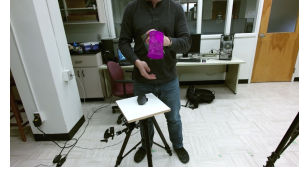
Αυτή η ανησυχία είναι δικαιολογημένη, πράγματι. Παρολαυτά, η πραγματικότητα αντιτίθεται στην πορεία σκέψης αυτής που θα μπορούσε ο ενδεχόμενος αναγνώστης να διατυπώσει, όπως μπορούμε να παρατηρήσουμε, ιδίως όμμασι, στο επόμενο (αριστερό) πλαίσιο. Από τη μία, βλέπουμε την λανθασμένη εκτίμηση πόζας που παρέχει το ίδιο το ΣΝΔ, χωρίς χειριστική διόρθωση. Σε ένα, συγκεκριμένο, πλαίσιο η εκτίμηση γυρίζει από τις 180 μοίρες και προσθέτει ένα δυσανάλογο λάθος από την μετρήσιμη περιστροφική μετρική που αποτελεί τίποτα περισσότερο από μια πλάνη. Στην πραγματικότητα, εξαιτίας του συνόλου της υπομονής την οποία έχουμε εμφυσήσει στον αλγόριθμό μας ήδη, πριν αποφασίσουμε την επαναρχικοποίησή του, το λάθος είναι πιθανό να συσσωρευτεί με ταχύτητα και να προσθέσει ένα λάθος που αντιστοιχεί στις εκτιμήσεις των επόμενων πλαισίων. Γι αυτό το λόγο, ακολουθούμε τον επόμενο, πολύ απλό αλγόριθμο.



Σχήμα 1.5 Η Περιστροφική μας Εκτίμηση για το 126ο πλαίσιο για το σενάριο ‘Δύσκολης Αλληλεπίδραση’.



Σχήμα 1.6 Η Περιστροφική μας Εκτίμηση για το 127ο πλαίσιο του σεναρίου ‘Δύσκολης Αλληλεπίδρασης’ χωρίς τον προτεινόμενο αλγόριθμο.



Σχήμα 1.7 Η Περιστροφική μας Εκτίμηση για το 126ο πλαίσιο του σεναρίου ‘Δύσκολης Αλληλεπίδρασης’.



Σχήμα 1.8 Η Περιστροφική μας Εκτίμηση για το 127ο πλαίσιο του σεναρίου ‘Δύσκολης Αλληλεπίδρασης’ με χρήση του προτεινόμενου αλγορίθμου.

Ο αλγόριθμος αυτός είναι αποκλειστικά βασισμένος στην παρατήρησή μας ότι οι εκτιμήσεις ποζών του τωρινού πλαισίου είναι κοντά σε εκείνη του προηγούμενου, ή τουλάχιστον, πολύ κοντινότερη από το λάθος που εισάγεται για μια λανθασμένη εκτίμηση διακριτής συμμετρίας. Έτσι, διαχωρίζουμε τις 360 μοίρες του χώρου των περιστροφικών συνιστωσών, για κάθε συνιστώσα, κατά έναν αριθμό από διακριτά εισαγόμενες συμμετρίες του αντικειμένου ενδιαφέροντος. Σε κάθε βήμα, αφού το ΣΝΔ εξάγει τις προβλέψεις του, επιλέγουμε το κατά πόσον η περιστροφική απόσταση μεταξύ της παρούσας και της επόμενης περιστροφικής εκτίμησης $d_A^o(\hat{r}_i(t), r_i(t-1))$:

$$d_A^o(\hat{r}_i(t), \hat{r}_i(t-1)) = 180^\circ - (180^\circ - |\hat{r}_i - r_i^{(GT)}|) \% 360^\circ, \quad i = a, b, c. \quad (1.9)$$

υπερβαίνει αυτό το πηλίκο, μείον το περιθώριο ασφαλείας. Αν το υπερβεί, αναθέτουμε ως παροντική εκτίμηση περιστοφής, την εκτίμηση του προηγούμενου πλαισίου, την οποία έχουμε φυλάξει από το προηγούμενο βήμα. Επί παραδείγματι, για το μοντέλο του Κουτιού με τα Μπισκότα, έχουμε δύο διακριτές συμμετρίες (στα δύο καπάκια του Κουτιού) και θέτουμε το κατώφλι να είναι $th = 80^\circ$.

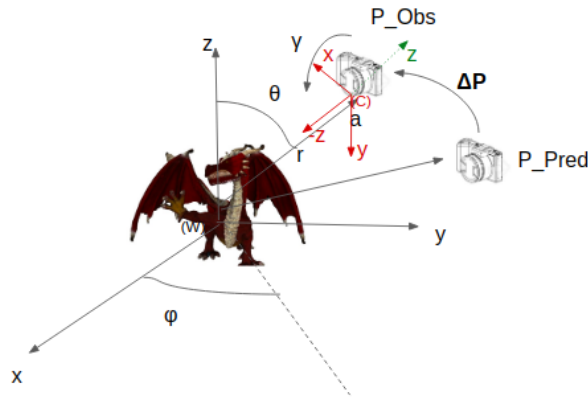
Algorithm 1: Discrete/Viewpoint-induced Rotational Symmetries discrepancy resolution

```

for every Rotation estimation  $\hat{R}(t)$  do
     $\hat{r}_a(t), \hat{r}_b(t), \hat{r}_c(t) = \text{mat2eulerXYZ}(\hat{R}(t))$  (in degrees)
    for every euler angle  $r_i, i=a,b,c$ : do
        if  $d_A^o(\hat{r}_i(t), \hat{r}_i(t-1)) \geq \left[ \frac{360^\circ}{N_{DiscrSymm}} - th \right]$  then
             $\hat{r}_i(t-1) \rightarrow \hat{r}'_i(t)$ 
             $\hat{R}(t) = \text{eulerXYZ2mat}(\hat{r}'_a(t), \hat{r}'_b(t), \hat{r}'_c(t))$ 
        else
             $\hat{R}(t)$  is a valid estimation
        end
    end
end

```

Δημιουργία Συνθετικού Συνόλου Δεδομένων Εκπαίδευσης



Σχήμα 1.9: Μια οπτική σύνοψη της μεθόδου δημιουργίας συνθετικών δεδομένων της εργασίας [29].

Δημιουργούμε ένα σύνολο δεδομένων εκπαίδευσης της τάξης των 20000 ζευγών RGB-D πλαισίων εικόνας: I_t, \hat{I}_t . Ακολουθούμε, κατά κύριο λόγο τη διαδικασία που επιδεικνύεται στη δημοσίευση των Garon et al. [29],[83] για τη συνθετική γέννηση των δεδομένων εκπαίδευσης η οποία περιλαμβάνει την απεικόνιση των ζευγών πόζας του αντικειμένου σε αντίστοιχα ζεύγη RGB-D πλαισίων εικόνας. Η διαδικασία αυτής της αποτύπωσης των 3-διάστατων γραφικών στο 2-διάστατο επίπεδο της εικόνας περιγράφεται λεπτομερώς, τόσο στη γενικότερη φιλοσοφία της, όσο και σε ό,τι αφορά τα συνιστώμενα μέρη, στο Κεφάλαιο 5. Το αντικείμενο ενδιαφέροντος τοποθετείται στην αρχή των αξόνων του συστήματος αναφοράς του 3-διάστατου κόσμου, (W), και δειγματολητούμε την πόζα μιας εικονικής κάμερας απεικόνισης, (c), η οποία και απεικονίζει τη σκηνή.

Οι Garon et al. [29], [83] δειγματολητούσαν την ‘Παρατηρήσιμη’ πόζα του ζεύγους με τη χρήση μιας πιθανοτικής δειγματοληψίας : $\theta \sim U(-\pi, \pi), x \sim U(0, 1), \phi = \arccos 2x - 1, r \sim U(0.4m, 1.5m)$, όπου τα θ, ϕ, r ήταν η τετμημένη η τεταγμένη και η ακτίνα της πόζας της εικονικής κάμερας, η οποία εκείτο εντός του σφαιρικού χώρου. Για τη γωνία κύλησης γ της κάμερας η δειγματοληψία ήταν $\gamma \sim U(-180^\circ, 180^\circ)$. Η ‘Προβλεπόμενη’ πόζα P_{Pred} της εικονικής κάμερας που απεικόνιζε το αντικείμενο στο πλαίσιο εικόνας αποκτείτο εφαρμόζοντας έναν ομοιόμορφο μετασχηματισμό (για την ακρίβεια τον αντίστροφό του) στην ‘Παρατηρούμενη’ πόζα P_{Obs} . Ο μετασχηματισμός αυτός, T , αποτελείτο από μια τυχαία μετατόπιση $\delta t_{x,y,z} \sim \mathcal{N}(0, \Delta t)$ και μια περιστροφή $\delta r_{a,b,c} \sim (0, \Delta r)$, π.χ. με επισημείωση Γωνίας-Άξονα και με μετατροπή της γωνιακής συνιστώσας στις αντίστοιχες παραμέτρους Euler.

$$\Delta P^{(c)} = T^{-1} = P_{Pred} P_{Obs}^{-1} \quad (1.10)$$

Γι αυτό και η αποσύνθεση του αντιστρόφου αυτού του μετασχηματισμού σε 6 παρα-

μέτρους, $\delta \mathbf{p} = \begin{bmatrix} t_x \\ t_y \\ t_z \\ r_a \\ r_b \\ r_c \end{bmatrix}$, αποτελεί την ετικέτα εκμάθησης της παλινδρόμησης πόζας.

Παραολαυτά, στην εργασία αυτή, εμείς τροποποιούμε αυτή τη συνταγή δημιουργίας των συνθετικών δεδομένων χρησιμοποιώντας μια πιο εύρωστη παραλλαγή της που βασίζεται στην ντετερμινιστική δειγματοληψία των παραμέτρων.

Συγκεκριμένα, η στρατηγική μας αποτελείται από έναν συνδυασμό δειγματοληψίας επί της επιφανείας της μοναδιαίας σφαίρας, η οποία ονομάζεται **‘Προσέγγιση της Χρυσής Σπείρας’**, ενός δεκαπλάσιου υπερσυνόλου ποζών σε σχέση με αυτές που τελικά χρησιμοποιούμε και της ανεπίβλεπτης συσταδοποίησης με χρήση **K-Μέσων**.

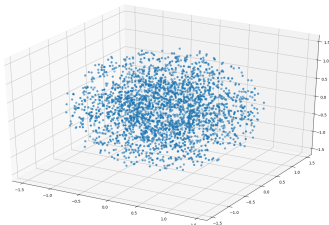
Επιγραμματικά,

$$\begin{aligned} \text{indices} &= \text{arange}(0, N_{\text{points}}) \\ \phi &= \text{acos}\left(2\frac{\text{indices} + 0.5}{N_{\text{points}}} - 1\right) \\ \theta &= \pi(1 + \sqrt{5})\frac{\text{indices} + 0.5}{N_{\text{points}}} \\ r &= \sqrt{\frac{\text{indices} + 0.5}{N_{\text{points}}}} \end{aligned} \quad (1.11)$$

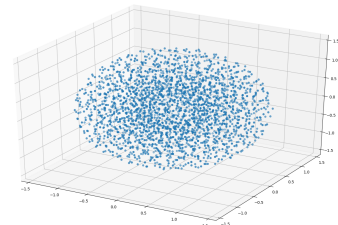
(Η δειγματοληψία της γωνίας κύλησης διατηρείται.)

Στην πράξη, αυτό που καταφέρνουμε, είναι να χωρίσουμε την μοναδιαία σφαίρα πιο ομοιόμορφα με έναν ντετερμινιστικό τρόπο και να αποφύγουμε την δημιουργία συστάδων που δημιουργεί η τυχαιότητα επιλογής. Αυτό μετράται πειραματικά, καθώς η μετρική επιτυχίας μας που εισάγουμε, η **μέγιστη απόσταση γειτονικών σημείων** είναι η ελάχιστη δυνατή, κατά μέσο όρο, μετά από δέκα επαναλήψεις, μεταξύ των παραλλαγών που δοκιμάζουμε (μαζί με αυτή, την αρχική, των Garon et al.[29]).

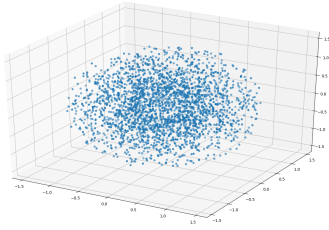
Γραφικά, μπορούμε να δούμε παρακάτω την ομοιομορφία δειγματοληψίας του συνόλου των μεθόδων που δοκιμάστηκαν.



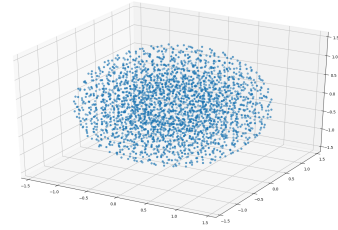
Σχήμα 1.10: (α) Τυχαία δειγματοληψία 20.000 σημείων. Η μέγιστη απόσταση μεταξύ των δύο κοντινότερων γειτόνων είναι: 0.256532 m.



Σχήμα 1.11: (β) Τυχαία Δειγματοληψία 200.000 σημείων και συσταδοποίησης με χρήση του Αλγορίθμου K-Μέσων σε 20.000 σημεία-κεντροειδή. Η μέγιστη απόσταση μεταξύ 2 κοντινότερων γειτόνων είναι: 0.222318 m.



Σχήμα 1.12: (γ) Ντετερμινιστική δειγματοληψία 20.000 σημείων. Η μέγιστη απόσταση μεταξύ των 2 κοντινότερων γειτόνων είναι: 0.295056 m.



Σχήμα 1.13: (δ) Ντετερμινιστική δειγματοληψία 20.000 σημείων και συσταδοποίηση με τον Αλγόριθμο K-Μέσων στα 20.000 σημεία κεντροειδή. Η μέγιστη απόσταση μεταξύ των 2 κοντινότερων γειτόνων είναι: 0.214753 m.

Σχήμα 1.14: Σύγκριση της μέγιστης δυνατής Ευκλιδειακής απόστασης κοντινότερων γειτόνων από τυχαία/ντετερμινιστική δειγματοληψία 3D σημείων σε μια σφαίρα χωρίς/με χρήση ενός ενδιάμεσου σημείου συσταδοποίησης KMέσων ενός μεγαλύτερου υπερσυνόλου.

Από τη στιγμή που έχουμε περιγράψει τη διαδικασία δημιουργίας του συνθετικού ζεύγους ποζών P_{Obs} , P_{Pred} , γεννούμε το ‘Προβλεπόμενο’ και το ‘Παρατηρούμενο’ πλαίσιο RGB-D εικόνας, τοποθετώντας την εικονική κάμερα στην P_{Pred} και στην P_{Obs} , αντίστοιχα και φωτίζουμε τη σκηνή καταλλήλως.

Επάυξηση Δεδομένων

Ακολουθώντας το δίδαγμα των Garon et al. [29], για το δίκτυο της εικόνας 1.2, δημιουργούμε συνθετικά ζεύγη RGB-D πλαισίων εικόνας $I(t), \hat{I}(t)$ και τροποποιούμε τη διαδικασία επαύξησης του Garon, [29], [83] όπως φαίνεται παρακάτω: Καταρχάς, ενσωματώνουμε την εικόνα αντικειμένου με μια τυχαία εικόνα παρασκηνίου, δειγματοληπτημένη από ένα υποσύνολο της βάσης SUN3D [163]. Μιμούμαστε την διαδικασία του Garon [29], [83] σε ότι αφορά την αποτύπωση του 3-διάστατου μοντέλου χεριού-επικαλυπτή και την ενσωμάτωσή του στο πλαίσιο εικόνας του αντικειμένου με μια πιθανότητα 60%. Μια παραλλαγή που προσθέτουμε στην διαδικασία του Garon, είναι η πλήρης επικάλυψη του αντικειμένου από το μοντέλο επικαλυπτή, με πιθανότητα 15% του υποσυνόλου των επικαλυπτόμενων αντικειμένων. Ας σημειωθεί πως και η δυαδική μάσκα παρασκηνίου και η αντιστοιχη δυαδική μάσκα ανεπικάλυπτων χωρικών εικονοστοιχείων διατηρούνται στο τέλος της διαδικασίας επαύξησης. Για αυτό, μπορούμε να τις χρησιμοποιήσουμε ως σήματα-ετικέτες αληθείας κατάκτησης για εξαγωγή χαώδους παρασκηνίου και χειρισμό επικαλύψεων στις βοηθητικές συναρτήσεις απώλειας για να επιβλέψουν τους αντίστοιχους χωρικούς μαλακούς χάρτες οπτικής προσοχής. Προσθέτουμε στο ‘Παρατηρήσιμο’ ζεύγος πλαισίων εικόνας $I(t)$: (i) Γκαουσιανό χρωματικό θόρυβο RGB, (ii) θόρυβο HSV, (iii) θάμπωση (για να προσομοιώσει ταχεία κίνηση του αντικειμένου), (iv) υποδειγματοληψία βάθους και (v) πιθανοτική κατάρριψη ενός από τις πηγές πκηροφορίας, όλες με τις ίδιες παραμέτρους με την εργασία του Garon [83]. Με μια πιθανότητα του 50%, αλλάζουμε την αντίθεση εικόνας, χρησιμοποιώντας παραμέτρους $\alpha \sim U(0, 3)$, $\beta \sim U(-50, 50)$ (όπου $U(\cdot)$ είναι μια ομοιόμορφη κατανομή) και η διόρθωση της παραμέτρου γ $\gamma \sim U(0, 2)$ με πιθανότητα 50%, για να βοηθήσουμε την γενίκευση σε περιπτώσεις διαφορετικών διαφορών φωτισμού μεταξύ των εικόνων που γεννώνται από τους αισθητήρες αντίληψης και των διορθωτικών εικόνων ανάδρασης. Αντί της μοντελοποίησης του θορύβου που προστίθεται

στην ‘Παρατηρούμενη’ πηγή πληροφορίας βάθους με μια ad-hoc Γκαουσιανή κατανομή, όπως στην δημοσίευση του Garon [83], σκεφτόμαστε τις συγκεκριμένες ιδιότητες του θορύβου του αισθητήρα Kinect [91] και μοντελοποιούμε τον θόρυβο αυτό μέσω μιας 3-διάστατης Γκαουσιανής κατανομής θορύβου (η οποία εξαρτάται τόσο από την ίδια την εικόνα βάθους, όσο και από την γνωστή πόζα του αντικειμένου εκπαίδευσης), που χρησιμοποιείται για να προσομοιώσουμε το κενό συνθετικής φύσης και πραγματικότητα που έχουν εκ γενετής οι εικόνες εκπαίδευσης και δοκιμής. Συγκεκριμένα,

Αξονικός Θόρυβος:

$$\sigma_{NA}(z, \theta_y) = 0.0012 + 0.0019 \cdot (z - 0.4)^2 + \frac{0.0001}{\sqrt{z}} \cdot \frac{\theta_y^2}{(\pi/2 - \theta_y)^2} \quad (1.12)$$

Βλέπουμε την αξονική συνιστώσα του θορύβου να εξαρτάται βαρέως από το z . Η σχέση της, όμως, με την γωνία θ_y είναι πιο περίπλοκη:

- για $\theta_y \in [0^\circ, 60^\circ]$ παραμένει προσεγγιστικά ίδια και
- για $\theta_y \in [60^\circ, 90^\circ]$ ο τετραγωνικός όρος κυριαρχεί στην εξίσωση και η Αξονική Συνιστώσα Θορύβου αυξάνει ακολουθώντας την ίδια τάση.

Παραλληλος Θόρυβος

$$\sigma_{NL}(\theta_y) = 0.8 + \frac{0.035 \cdot \theta_y}{\frac{\pi}{2} - \theta_y}, \quad (1.13)$$

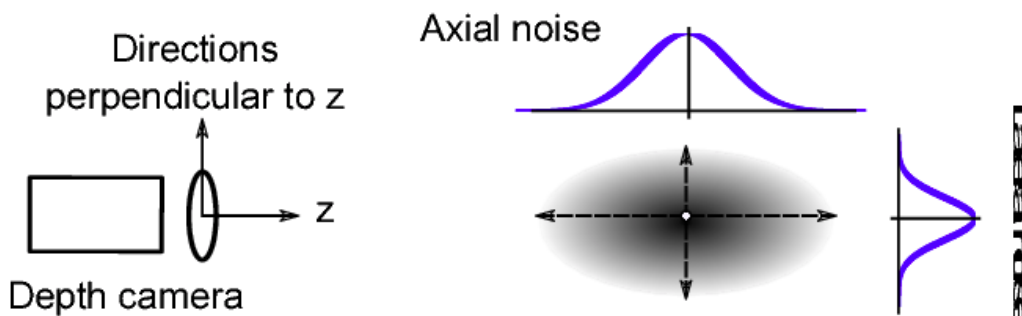
σε μονάδες εικονοστοιχείων (pixels([pxls])).

$$\begin{aligned} \sigma_{N_L^x}(z, \theta_y) &= \sigma_{NL}[\text{pxls.}] \cdot z \frac{c_x}{f_x} \\ \sigma_{N_L^y}(z, \theta_y) &= \sigma_{NL}[\text{pxls.}] \cdot z \frac{c_y}{f_y}, \end{aligned} \quad (1.14)$$

σε μέτρα (μ).

$c_{x/y}$ είναι οι συνιστώσες του κέντρου της κάμερας και $f_{x/y}$ τα αντίστοιχα εστιακά βάθη.

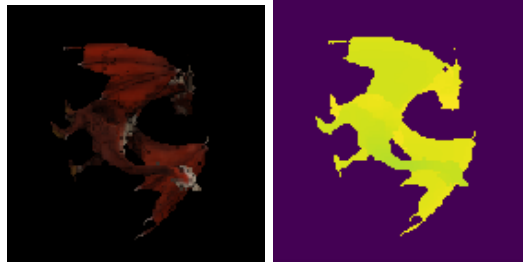
Οι Παράλληλες Συνιστώσες, από την άλλη μεριά, εξαρτώνται ελαφρώς από το βάθος z , ειδικά καθώς αυτό αυξάνει.



Σχήμα 1.15: Μια 3-διάστατη Γκαουσιανή Κατανομή την οποία χρησιμοποιούμε για να μοντελοποιήσουμε τον θόρυβο του Kinect, η οποία αποτελεί γινόμενο δύο Παράλληλων και μιας Αξονικής συνιστώσας θορύβου ως προς τον κύριο άξονα της κάμερας[91].

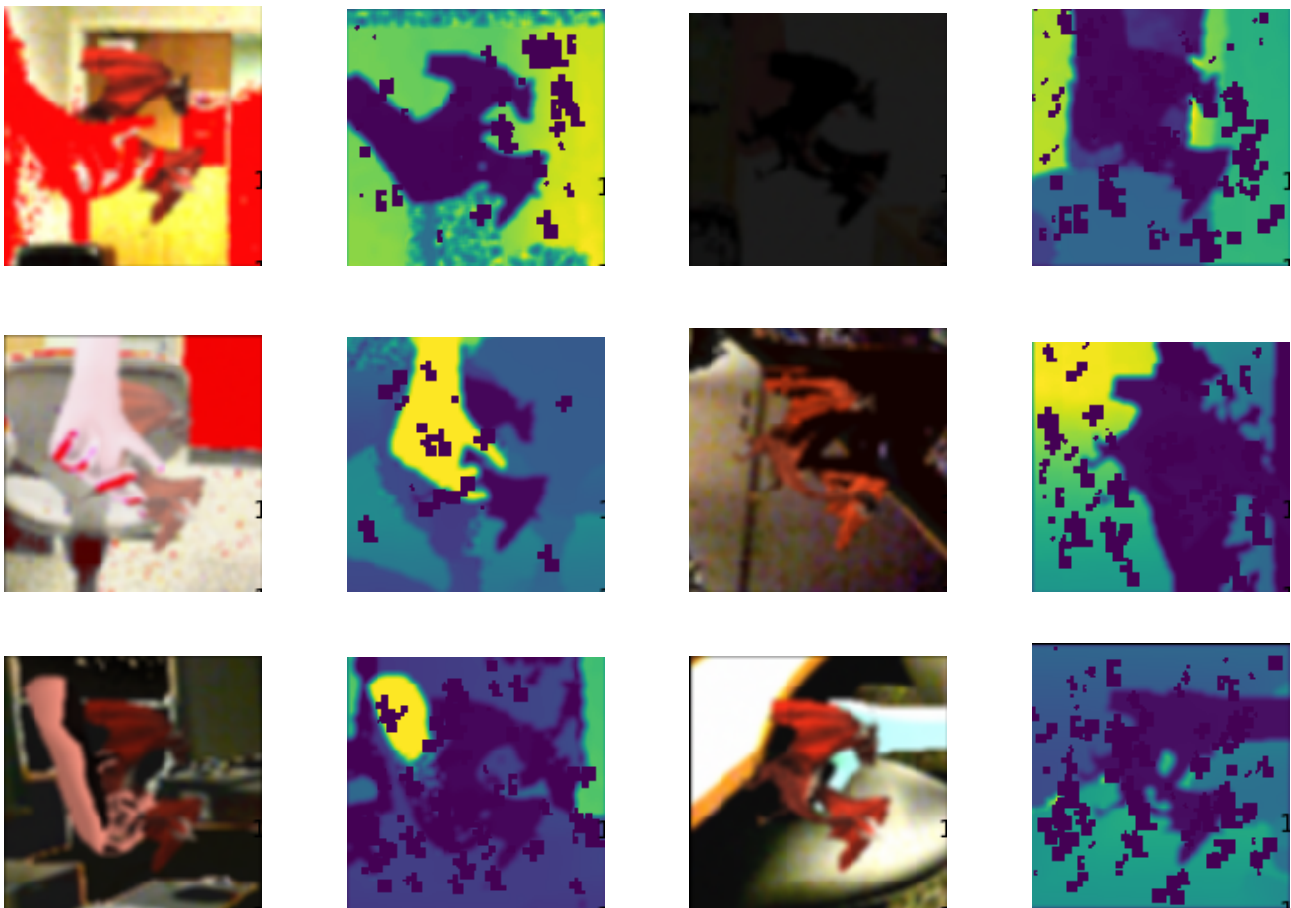
Η τελική 3-διάστατη κατανομή του θορύβου του Kinect είναι γινόμενο και των τριών, όπως έχει προαναφερθεί (Εικόνα 1.15):

$$\mathcal{N}(0, \sigma_{Kinect3D}) = \mathcal{N}(0, \sigma_{N_A}) * \mathcal{N}(0, \sigma_{N_L^{(x)}}) * \mathcal{N}(0, \sigma_{N_L^{(y)}}) \quad (1.15)$$



Σχήμα 1.16: Ένα ενδεικτικό τυχαίο Παρατηρούμενο ζεύγος RGB-D ζεύγος εικόνων στο οποίο επιδεικνύουμε τυχόντες συνδυασμούς της συνολικής μας στρατηγικής.

Υπενθυμίζουμε πως το Προβλεπόμενο RGB-D πλαίσιο εικόνας δεν απαιτεί επαύξηση, καθώς τυπωνεται συνθετικά και στην εκπαίδευση και στην αξιολόγητη του παρακολουθητή.



Σχήμα 1.17: Πλήρη τυχαία παραδείγματα επαύξησης της ίδιας Παρατηρούμενης RGB (αριστερα) και -D (δεξιά) εικόνας.

Λεπτομέρειες Υπολοποίησης

Χρησιμοποιούμε Εκθετικές Γραμμικές Μονάδες (EGM) για συναρτήσεις ενεργοποίησης, ορίζουμε την μικροσυστάδα μεγέθους 128 δειγμάτων, εφαρμόσουμε Αποκοπή με πιθανότητα 0.3, έναν βελτιστοποιητή Adam ο οποίος με μια διορθωμένη παρακμή βαρών

[78] κατά έναν παράγοντα $1e^{-05}$, ρυθμό μάθησης $1e^{-03}$ και ρύθμιση του βετιστοποιητή με θερμές επανενάρξεις με ανόπτηση συνημιτόνων κάθε 10 εποχές. Όλα τα βάρη του δικτύου (εκτός αυτών που μεταφέρονται από το ResNet18 [37]) αρχικοποιούνται από μια στρατηγικής ομοιόμορφης κατανομής Kaiming He [38]. Δημιουργείται το πρόβλημα πως αν το δίκτυο εκπαιδευτεί, μετά την αρχικοποίηση των βαρών του, απευθείας με τη χρήση της συνολικής συνάρτησης απωλείας που περιλαμβάνει τη Γεωδεσική απόσταση περιστροφών, το δίκτυο θα υποεκπαιδευτεί. Αυτό συμβαίνει εξαιτίας του γεγονότος πως, αν συλλογιστούμε τις τοπικές μερικές παραγώγους της Γεωδεσικής απόστασης περιστροφών

$$\begin{aligned}
\bullet \quad \frac{\partial L_{Rot}(R_{GT}, \hat{R})}{\partial \hat{R}} &= \frac{1}{B} \frac{\partial \sum_{b=1}^B \frac{\arccos(\text{tr}(R_{(\mathbf{b}),GT}^T \cdot \hat{R}_{(\mathbf{b})}) - 1)}{2}}{\partial \hat{R}_{(\mathbf{b})}} = \frac{1}{B} \sum_{b=1}^B \frac{1}{2} \frac{\partial [\arccos(\text{tr}(R_{(\mathbf{b}),GT}^T \cdot \hat{R}_{(\mathbf{b})}) - 1)]}{\partial \hat{R}_{(\mathbf{b})}} \\
&= \left[-\frac{1}{B} \sum_{b=1}^B \frac{\partial (\text{tr}(R_{(\mathbf{b}),GT}^T \cdot \hat{R}_{(\mathbf{b})}) - 1)}{\partial \hat{R}_{(\mathbf{b})}} \frac{1}{2\sqrt{1 - (\text{tr}(R_{(\mathbf{b}),GT}^T \cdot \hat{R}_{(\mathbf{b})}) - 1)^2}} \right] \\
&= \left[-\frac{1}{B} \sum_{b=1}^B \frac{R_{(\mathbf{b}),GT}}{2\sqrt{2\text{tr}(R_{(\mathbf{b}),GT}^T \cdot \hat{R}_{(\mathbf{b})}) - \text{tr}^2(R_{(\mathbf{b}),GT}^T \cdot \hat{R}_{(\mathbf{b})})}} \right] \quad (1.16)
\end{aligned}$$

$$\begin{aligned}
\bullet \quad \frac{\partial L_{Rot}(R_{GT}, \hat{R})}{\partial R_{GT}} &= \frac{1}{B} \frac{\partial \sum_{b=1}^B \frac{\arccos(\text{tr}(R_{(\mathbf{b}),GT}^T \cdot \hat{R}_{(\mathbf{b})}) - 1)}{2}}{\partial R_{(\mathbf{b}),GT}} = \frac{1}{B} \sum_{b=1}^B \frac{1}{2} \frac{\partial [\arccos(\text{tr}(R_{(\mathbf{b}),GT}^T \cdot \hat{R}_{(\mathbf{b})}) - 1)]}{\partial R_{(\mathbf{b}),GT}} \\
&= \left[-\frac{1}{B} \sum_{b=1}^B \frac{\partial (\text{tr}(R_{(\mathbf{b}),GT}^T \cdot \hat{R}_{(\mathbf{b})}) - 1)}{\partial R_{(\mathbf{b}),GT}} \frac{1}{2\sqrt{1 - (\text{tr}(R_{(\mathbf{b}),GT}^T \cdot \hat{R}_{(\mathbf{b})}) - 1)^2}} \right] \\
&= \left[-\frac{1}{B} \sum_{b=1}^B \frac{\hat{R}_{(\mathbf{b})}}{2\sqrt{2\text{tr}(R_{(\mathbf{b}),GT}^T \cdot \hat{R}_{(\mathbf{b})}) - \text{tr}^2(R_{(\mathbf{b}),GT}^T \cdot \hat{R}_{(\mathbf{b})})}} \right], \quad (1.17)
\end{aligned}$$

θα αντιληφθούμε πως, από μια μαθηματική αντίληψη, εξηγούμε πως το να ελαχιστοποιήσουμε την Γεωδεσική Απόσταση Περιστροφών από την αρχή είναι η αιτία παγίδευσης των βαρών του δικτύου στα τοπικά ελάχιστα : κατα την αρχικοποίηση των βαρών του δικτύου, όταν το γινόμενο πινάκων $R_{(\mathbf{b}),GT}^T \cdot \hat{R}_{(\mathbf{b})}$ αποκλείει σημαντικά από τον μοναδιαίο πίνακα \mathbb{I}_3 , οι παράγωγοί της έχουν έναν ισχυρό παράγοντα παρονομαστή (καθώς τα ίχνη παρουσιάζουν τότε υψηλότερες τιμές) αυτό τις κάνει όλο και πιο ανίσχυρες καθώς πσιωδιαβαίνουμε, μέσω του αλγορίθμου εκπαίδευσης, στα προηγούμενα επίπεδα. Από την άλλη, μια ευμενής αρχικοποίηση των βαρών του δικτύου μετά από μια ελαχιστοποίησης μιας κυρτής, αρχικής συνάρτησης απωλείας, παράσχει κατάλληλα, υποβέλτιστα, ‘θερμότερα’ βάρη τα οποία αποτελούν ένα καλύτερο αρχικό σημείο βελτιστοποίησης της Γεωδεσικής Απόστασης, οπότε και το ανωτέρο πρόβλημα αμβλύνεται και έτσι οι τοπικές

παράγωγοι έουν τη δυνατότητα να πισωδιαβούν ευκολότερα και, έτσι, να βελτιστοποιήσουν, κατά το δυνατόν, την εκτίμηση περιστροφών. Η αρχική αυτή κυρτή συνάρτηση απωλείας που επιλέγεται είναι η Λογαριθμική Αντιστροφή του Συνημιτόνου, η ποία και ελαχιστοποιείται για 25, αρχικές, εποχές. Τότε, εκπαιδεύουμε το δίκτυο, με την ολική, Γεωδেসική, μέχρις συγκίσεως του. Ο μέσος χρόνος εκπαίδευσης είναι 12 ώρες σε μια μοναδική Κάρτα Γραφικών τύπου GeForce 1080 Ti.

Βάση Δεδομένων Ελέγχου και Μετρικές

Ελέγχουμε την αποτελεσματικότητα της μεθόδου μας στο σενάριο των ‘Δύσκολων Ελεύθερων Αλληλεπιδράσεων’ το οποίο περιλαμβάνεται στη βάση δεδομένων που πρωτοσυστήθηκε από την ομάδα του Garon στην δημοσίευση [83]. Η βάση αυτή θεωρείται, μέχρις ώρας η πληρεστερη βάση περιβάλλοντος παραγματικού κόσμου και το εν λόγω σενάριο το πολυπλοκότερο και δυσχερεστερο. Περιλαμβάνει ελεύθερες 3-διάστατες κινήσεις (τόσο μετατοπίσεις, όσο και περιστροφές) του αντικειμένου, σε συνδυασμό με αυθόρμητες αλληλεπιδράσεις κι επικαλύψεις από τα χέρια του χρήστη. Το συμπέρασμά μας είναι πως εαν η μέθοδος που προτείνουμε παρέχει ανώτερη απόδοση σε αυτό το πλέον προκλητικό σενάριο, θα παρουσιάσει παρόμοια συμπεριφορά και θα έχει τουλάχιστον το ίδιο καλές επιδόσεις (ή και καλύτερες, συγκριτικά) σε κάθε άλλο σενάριο που περιέχεται στη βάση. Ακολουθώντας τη μεθοδολογία των Garon et al. [83], αρχικοποιούμε τον εκπαιδευμένο παρακολουθητή μας ανά 15 πλαίσια εικόνας και χρησιμοποιούμε τις ίδιες μετρικές αξιολόγησης που θεμελιώνονται στην ως άνω εργασία :

- Το Λάθος Μετατόπισης ορίζεται απλά με τη χρήση της L2 νόρμας μεταξύ των 2 διανυσμάτων μετατόπισης: του εκτιμώμενου και του παραγματικού.

$$\delta_t(\Delta\hat{t}, \Delta t_{GT}) = \|\Delta\hat{t} - \Delta t_{GT}\|_2 \quad (1.18)$$

- Το Λάθος Περιστροφής μεταξύ του εκτιμώμενου και του παραγματικού πίνακα 3-διάστατων περιστροφών είναι:

$$\delta_R(\Delta\hat{R}, \Delta R_{GT}) = \arccos\left(\frac{\text{tr}(\Delta\hat{R}^T \cdot \Delta R_{GT}) - 1}{2}\right), \quad (1.19)$$

όπου η σημειολογία $\text{tr}(\cdot)$ δηλώνει το ίχνος πίνακα.

Λόγω των περιορισμένων υπολογιστικών πόρων που διαθέτουμε, παράγουμε μόνο 20000 δεδομένα εκπαίδευσης, η ποικιλότητα των οποίων φροντίζουμε να καλύπτει επαρκώς το χώρο ποζών, τόσο για την εξασφάλιση της επιτυχίας της συγκριτικής μας μελέτης όσο και την εγκυρότητα της συνολικής πειραματικής μας διάταξης.

Συγκριτική Μελέτη

- Ο Παρακολουθητής βασίζεται κυρίως στη συνιστώσα του 3-διάστατου σχήματος του αντικειμένου, δηλαδή τις εικόνες Βάθους, για να εκτιμήσει την πόζα του. Μάλιστα, η πηγή αυτή πληροφορίας αρκεί για εκτιμήσεις σχετικά υψηλής ευκρίνειας. Παρολαυτά, η συνιστώσα της Εμφάνισης (εικόνες RGB) συνεισφέρουν σε καίρια σημεία στην παραγωγή ακριβέστερων εκτιμήσεων πόζας.

- Οι Διαφορικές διεπίπεδες συνδέσεις αντί για τις συνέσεις Πυκνού Δικτύου (συνορευόμενης τοποθέτησης) και η ορθότερη αρχικοποίηση του δικτύου με τη χρήση των τεχνικών των Xavier[31] και He[38] είναι οι δύο παράγοντες που προσφέρουν την υψηλότερη μείωση λάθους του παρακολουθητή. Παρολαυτά, δεν αρκούν για ακριβείς εκτιμήσεις περιστροφών.
- Κάθε βήμα σταδιακής λεπτομερέστερης μοντελοποίησης της συνιστώσας περιστροφών βελτιώνει την απόδοση του αμέσως προηγούμενου του, βελτιώνοντας την ακρίβεια των εκτιμήσεων από την αρχή μέχρι το τέλος τουλάχιστον 60%.
- Η χρήση της Χωρικής Οπτικής Προσοχής, σε κάθε της μορφή, βελτιώνει την εκτίμηση της Πόζας του Αντικειμένου, με μικρό υπολογιστικό κόστος. Θεωρητικά, μόνο η μονάδα Χειρισμού των Επικαλύψεων θα αρκούσε, καθώς η Εξαγωγή Προσκήνιου είναι υποστόχος της ανωτέρω, αλλά τα πρακτικά μας πειράματα αναδεικνύουν ως καλύτερη τον Ιεραρχικό συνδυασμό των δύο και ως βέλτιστη την Παράλληλη συνύπαρξή τους.
- Η βέλτιστη τακτική ‘ζεστάματος’ των βαρών του δικτύου πριν το ραφινάρισμά τους με την Γεωδесική συνάρτηση απώλειας είναι εκείνη όπου οι παράμετροι πόζας αρχικά φράσσονται και ελαχιστοποιούν την συνάρτηση Λογαριθμικού Υπερβολικού Συνημιτόνου.
- Οι διάφορες συνιστώσες μας λειτουργούν προσθετικά και συμπληρωματικά κι επιφέρουν εντυπωσιακά αποτελέσματα μειώνοντας την ολίσθηση πόζας, προσδίδοντας καλύτερη κατανόηση του οπτικού αποτυπώματος των περιστροφών και μειώνοντας τα ολοσχερή σφάλματα.

Πειραματικά αποτελέσματα

Ακολουθώντας την αναλογιστική μας μελέτη, προχωράμε στην συνένωση των παραλλήλων δομών Οπτικής Προσοχής, με την Γεωδесική Συνάρτηση Απώλειας Περιστροφών της 1.5, μαζί με τα υπολείποντα στοιχεία του Κεφαλαίου 6. Αξιολογούμε τη μέθοδο πάνω σε δύο αντικείμενα του συνόλου δεδομένων των Garon et al. [83]: ένα ασύμμετρο μοντέλο δράκου με πλούσια υφή και περίτεχνο, περίπλοκο σχήμα και ένα συμμετρικό κυλινδρικό μοντέλο κουτιού από μπισκότα, το οποίο έχει χαμηλές πιστότητας και ποιότητας υφή και απλό 3-διάστατο σχήμα.

Παρουσιάζουμε μια εκτενή αξιολόγηση της μεθόδου μας σε 2 αντικείμενα του συνόλου δεδομένων που παρέχεται από τους Garon et al.[83]: ένα μη-συμμετρικό με πλούσια υφή και περιπεπλεγμένο σχήμα (το μοντέλο του Δράκου) και ένα ακόμα, συμμετρικό με φτωχή-ελλειπή υφή και απλοϊκό 3-διάστατο σχήμα (το μοντέλο του Κουτιου με τα Μπισκότα). Παρολαυτά, ο παρακολουθητής, ας δοκιμάζεται όχι μόνο στα δύο αυτά αντικείμενα, αλλά και σε τρία επιπλέον μοντέλα αντικείμενων: τον ‘Σκύλο’, το ‘Σύμπλεγμα από Lego’ και το ‘Ποτηστήρι’, το καθένα με τα δικά του ιδιαίτερα χαρακτηριστικά. Το ευρύ αυτό φάσμα από χαρακτηριστικά είναι αυτό το οποίο κάνει τη διαδικασία αξιολογησής μας αξιόπιστη και τα αποτελέσματα μας ικανά να γενικευτούν και σε άλλα αντικείμενα.

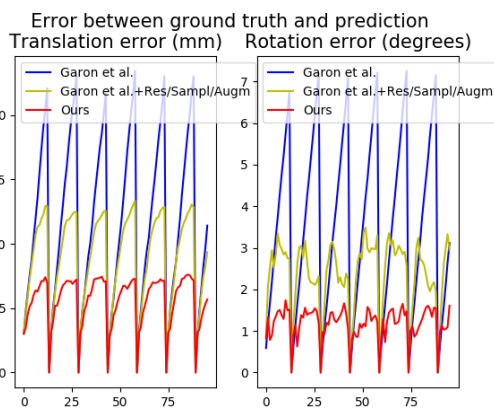


Αντικείμενο	Χαρακτηριστικά					
	Μέγεθος	Συμμετρία	Σχήμα	Υφή	Ξεχωριστά Μέρη	
‘Δράκος’	Μέτριο	Όχι	Περιπλεγμένο	Πλούσια	Ναι	
‘Κουτί με Μπισκότα’	Μέτριο	Περιστροφoαναλαστική	Απλό	Φτωχή και Επαναλαμβανόμενη	Όχι	
‘Σχάλος’	Μέτριο	Όχι	Περιπλεγμένο	Σχεδόν Καθόλου	Ναι	
‘Σύμπλεγμα από Lego’	Μικρό	Όχι	Περιπλεγμένο	Πλούσια και Επαναλαμβανόμενη	Όχι	
‘Ποτιστήρι’	Μεγάλο	Όχι	Απλό	Φτωχή	Ναι	

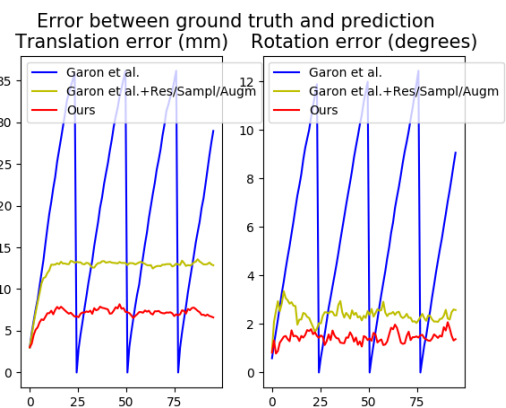
Πίνακας 1.1: Χαρακτηριστικά από πέντε αντικείμενα στα οποία δοκιμάζουμε την προσέγγισή μας. Το γεγονός πως δεν υπάρχουν δυο πανομοιότυπα αντικείμενα επαληθεύει τις ικανότητες γενίκευσης του παρακολούθητή μας.

Το Μοντέλο Δράκου: η ασύμμετρη περίπτωση

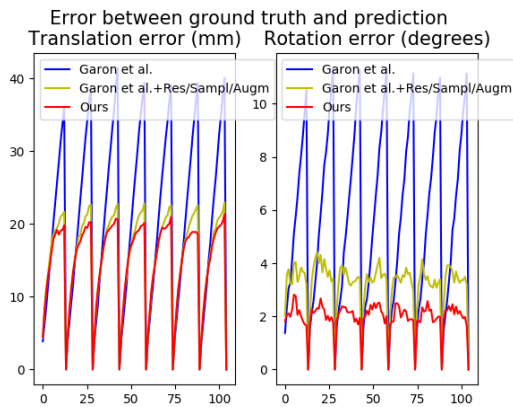
Η προσέγγισή μας μειώνει τα μέσα λάθη κατά περίπου 40.5 % για μετατοπίσεις και 57.7 % για περιστροφές ως προς τη βασικά αρχιτεκτονική τω Garon et al. [83]. Όταν το αντικείμενο δεν επικαλύπτεται, ο παρακολούθητής επικεντρώνει κυρίως στο 3-διάστατο κέντρο του αντικειμένου, ουσιαστικά κατανοώντας, με αυτό τον τρόπο, ότι αυτό είναι το κύριο 3-διάστατο σημείο ενδιαφέροντος παρακολούθησης πόζας. Όταν το χέρι του χρήστη επικαλύπτει μέρη του δράκου, η οπτική προσοχή μεταφέρεται στα μέρη ενδιαφέροντος του σώματος του αντικειμένου, τα οποία ξεχωρίζουν μες από τη λαβή του χρήστη και τον κύριο κορμό του αντικειμένου, όπως επι παραδείγματι, ο λαιμός, τα φτερά και η ουρά. Η αποτελεσματικότητα της μεθόδου μας επιδεικνύεται από το γεγονός ότι, ενώ οι Garon et al. [83] διατηρούν την παρακολούθηση μόνο της 3-διάστατης θέσης του αντικειμένου-δράκου υπό ακραίες συνθήκες, η βελτιωμένη μας εκδοχή ξεπερνά την ιδιότητα αυτή για τις 3-διάστατες περιστροφές, επίσης. Παρόλο που η δική μας μέθοδος είναι πιο φορτική υπολογιστικά, η ταχύτητα του ΣΝΔ (40 πλαίσια εικόνας / δευτερόλεπτο) ακόμα κείται εντός των ορίων της απόδοσης πραγματικού χρόνου που τίθεται από τους Garon et al. [83].



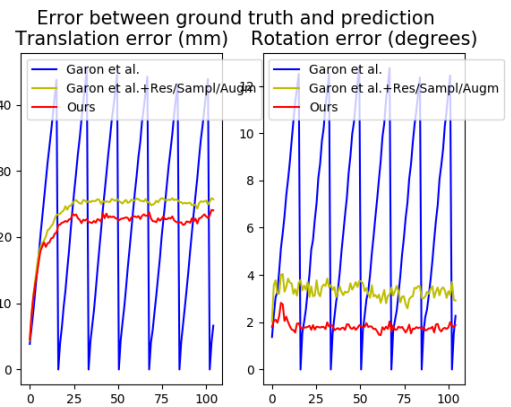
Σχήμα 1.18 Αποτύπωση μετρικών 3-διάστατου Μεταφορικού και Περιστροφικού λάθους για το σενάριο ‘Κοντινής Σταθερότητας’ του 3-διάστατου μοντέλου του ‘Δράκου’, στην περίπτωση που ο παρακολούθητής επαναρχικοποιείται κάθε 15 εικόνες.



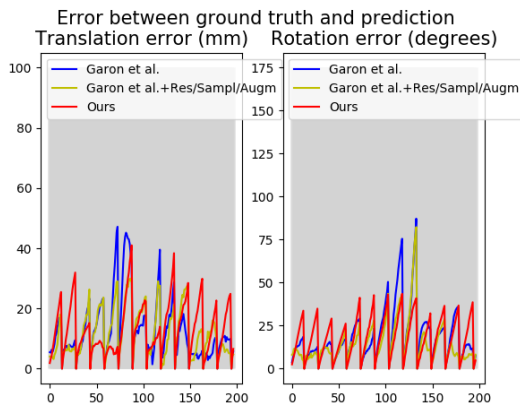
Σχήμα 1.19 Αποτύπωση μετρικών 3-διάστατου Μεταφορικού και Περιστροφικού λάθους για το σενάριο ‘Κοντινής Σταθερότητας’ του 3-διάστατου μοντέλου του ‘Δράκου’, στην περίπτωση που ο παρακολούθητής επαναρχικοποιείται κάθε φορά που αποτυγχάνει.



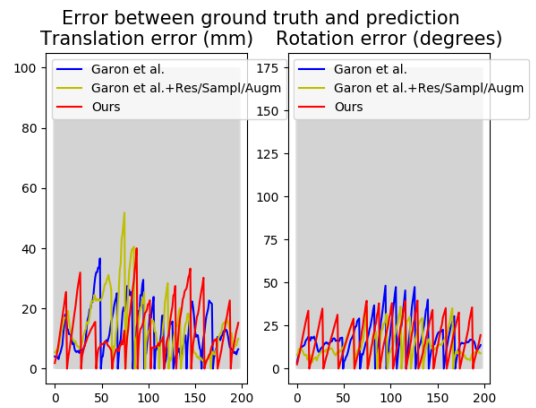
Σχήμα 1.20 Αποτύπωση μετρικών 3-διάστατου Μεταφορικού και Περιστροφικού λάθους για το σενάριο 'Μακρινής Σταθερότητας' του 3-διάστατου μοντέλου του 'Δράκου', στην περίπτωση που ο παρακολουθητής επαναρχικοποιείται κάθε 15 εικόνες.



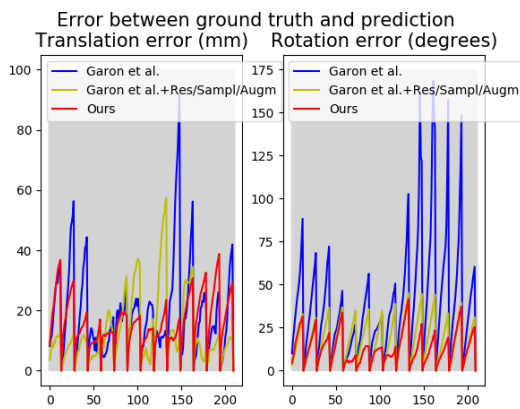
Σχήμα 1.21 Αποτύπωση μετρικών 3-διάστατου Μεταφορικού και Περιστροφικού λάθους για το σενάριο 'Μακρινής Σταθερότητας' του 3-διάστατου μοντέλου του 'Δράκου', στην περίπτωση που ο παρακολουθητής επαναρχικοποιείται κάθε φορά που αποτυγχάνει.



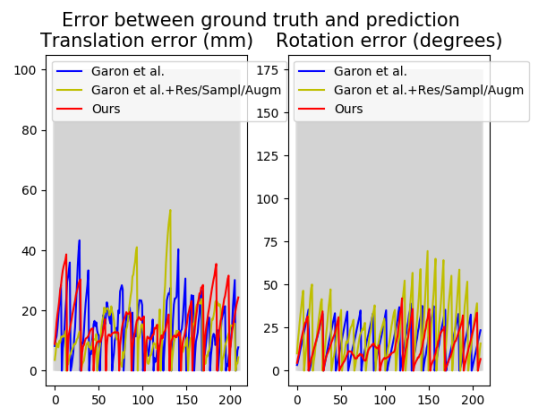
Σχήμα 1.22 Αποτύπωση μετρικών 3-διάστατου Μεταφορικού και Περιστροφικού λάθους για το σενάριο '75% Κάθετης Επικάλυψης' του 3-διάστατου μοντέλου του 'Δράκου', στην περίπτωση που ο παρακολουθητής επαναρχικοποιείται κάθε 15 εικόνες.



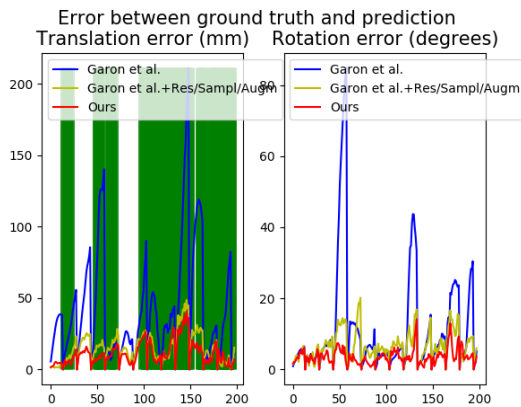
Σχήμα 1.23 Αποτύπωση μετρικών 3-διάστατου Μεταφορικού και Περιστροφικού λάθους για το σενάριο '75% Κάθετης Επικάλυψης' του 3-διάστατου μοντέλου του 'Δράκου', στην περίπτωση που ο παρακολουθητής επαναρχικοποιείται κάθε φορά που αποτυγχάνει.



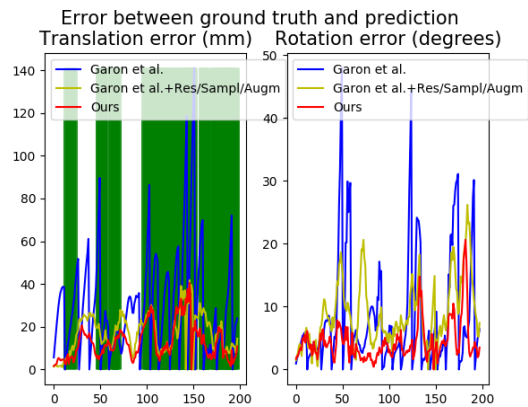
Σχήμα 1.24 Αποτύπωση μετρικών 3-διάστατου Μεταφορικού και Περιστροφικού λάθους για το σενάριο '75% Οριζόντιας Επικάλυψης' του 3-διάστατου μοντέλου του 'Δράκου', στην περίπτωση που ο παρακολουθητής επαναρχικοποιείται κάθε 15 εικόνες.



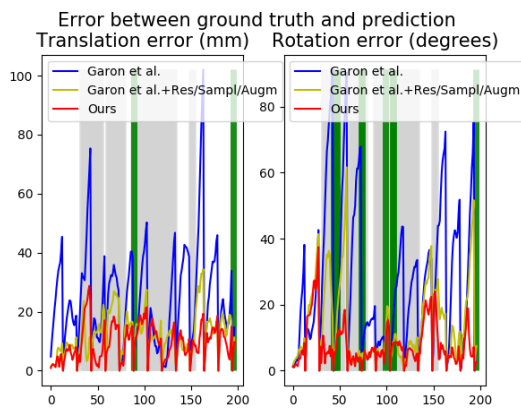
Σχήμα 1.25 Αποτύπωση μετρικών 3-διάστατου Μεταφορικού και Περιστροφικού λάθους για το σενάριο '75% Οριζόντιας Επικάλυψης' του 3-διάστατου μοντέλου του 'Δράκου', στην περίπτωση που ο παρακολουθητής επαναρχικοποιείται κάθε φορά που αποτυγχάνει.



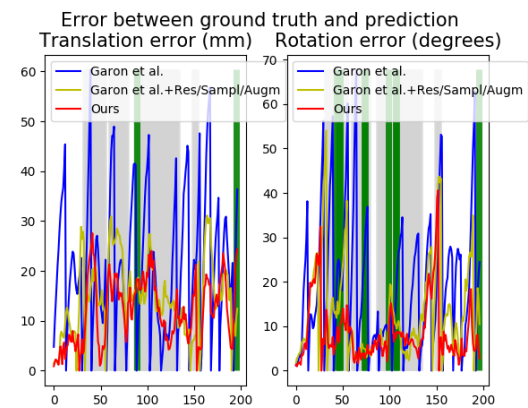
Σχήμα 1.26 Αποτύπωση μετρικών 3-διάστατου Μεταφορικού και Περιστροφικού λάθους για το σενάριο 'Αποκλειστικά Μετατόπισης' του 3-διάστατου μοντέλου του 'Δράκου', στην περίπτωση που ο παρακολουθητής επαναρχικοποιείται κάθε 15 εικόνες.



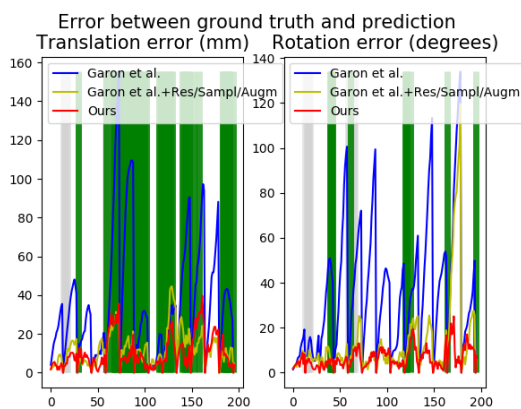
Σχήμα 1.27 Αποτύπωση μετρικών 3-διάστατου Μεταφορικού και Περιστροφικού λάθους για το σενάριο 'Αποκλειστικά Μετατόπισης' του 3-διάστατου μοντέλου του 'Δράκου', στην περίπτωση που ο παρακολουθητής επαναρχικοποιείται κάθε φορά που αποτυγχάνει.



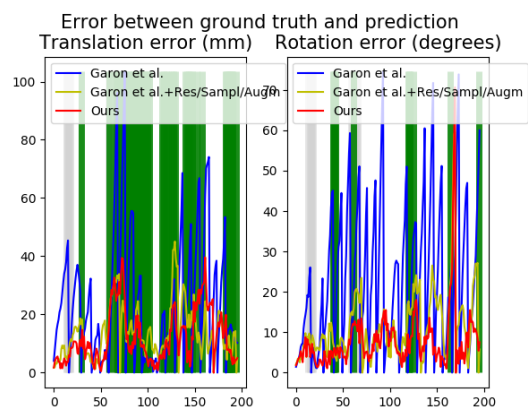
Σχήμα 1.28 Αποτύπωση μετρικών 3-διάστατου Μεταφορικού και Περιστροφικού λάθους για το σενάριο 'Αποκλειστικά Περιστροφών' του 3-διάστατου μοντέλου του 'Δράκου', στην περίπτωση που ο παρακολουθητής επαναρχικοποιείται κάθε 15 εικόνες.



Σχήμα 1.29 Αποτύπωση μετρικών 3-διάστατου Μεταφορικού και Περιστροφικού λάθους για το σενάριο 'Αποκλειστικά Περιστροφών' του 3-διάστατου μοντέλου του 'Δράκου', στην περίπτωση που ο παρακολουθητής επαναρχικοποιείται κάθε φορά που αποτυγχάνει.

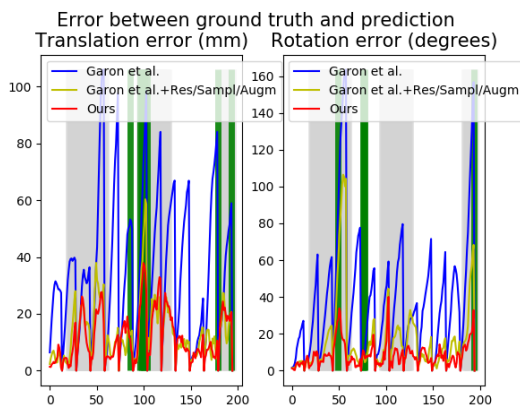


Σχήμα 1.30 Αποτύπωση μετρικών 3-διάστατου Μεταφορικού και Περιστροφικού λάθους για το σενάριο 'Πλήρους Αλληλεπίδρασης' του 3-διάστατου μοντέλου του 'Δράκου', στην περίπτωση που ο παρακολουθητής επαναρχικοποιείται κάθε 15 εικόνες.

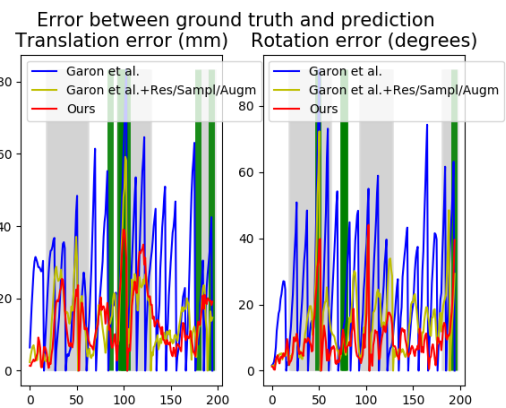


Σχήμα 1.31 Αποτύπωση μετρικών 3-διάστατου Μεταφορικού και Περιστροφικού λάθους για το σενάριο 'Πλήρους Αλληλεπίδρασης' του 3-διάστατου μοντέλου του 'Δράκου', στην περίπτωση που ο παρακολουθητής επαναρχικοποιείται κάθε φορά που αποτυγχάνει.

Αποτύπωση Λαθών (αριστερά:επαναρχικοποίηση κάθε 15 εικόνες) και (δεξιά:επαναρχικοποίηση κάθε φορά που ο παρακολουθητής αποτυγχάνει):

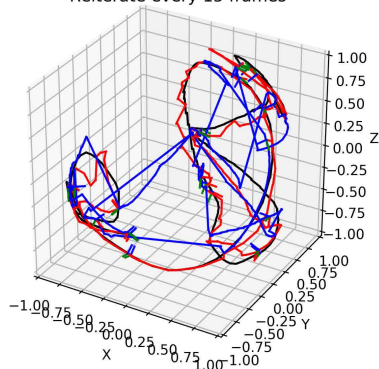


Σχήμα 1.32 Αποτύπωση μετρικών 3-διάστατου Μεταφορικού και Περιστροφικού λάθους για το σενάριο 'Δύσκολης Αλληλεπίδρασης' του 3-διάστατου μοντέλου του 'Δράκου', στην περίπτωση που ο παρακολουθητής επαναρχικοποιείται κάθε 15 εικόνες.

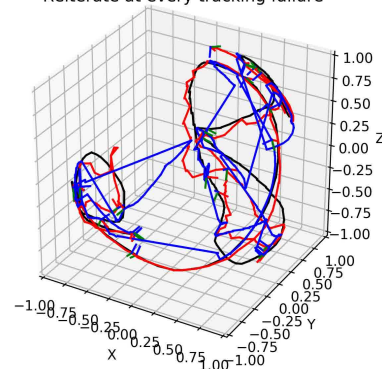


Σχήμα 1.33 Αποτύπωση μετρικών 3-διάστατου Μεταφορικού και Περιστροφικού λάθους για το σενάριο 'Δύσκολης Αλληλεπίδρασης' του 3-διάστατου μοντέλου του 'Δράκου', στην περίπτωση που ο παρακολουθητής επαναρχικοποιείται κάθε φορά που ο παρακολουθητής αποτυγχάνει.

6D Temporal Pose Tracking Visualization
Dragon --- Hard Interaction scenario
Reiterate every 15 frames



6D Temporal Pose Tracking Visualization
Dragon --- Hard Interaction scenario
Reiterate at every tracking failure

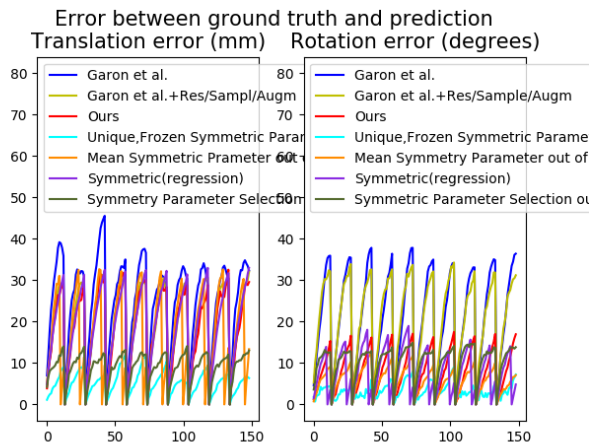


Σχήμα 1.34: Οι 6-Δ Τροχίες Πόζας του μοντέλου του 'Δράκου', στο χρόνο, για το σενάριο των 'Δύσκολων Αλληλεπιδράσεων'. Είναι προφανές, και στις 2 περιπτώσεις, και από την σκοπιά αυτή, πως η προτεινόμενη προσέγγισή μας, αυτή ξεπερνά σε απόδοση εκείνη των Garon et al. [83] κατά πολύ, μιας και παράγει μικρότερα λάθη και λιγότερες αποτυχίες. Όλες οι μονάδες μέτρησης μήκους είναι σε χιλ. και οι περιστροφές σε μοίρες. Οι μαύρη τροχιά είναι η πραγματική, η μπλε τροχιά είναι η βασική τροχιά σύγκρισης της μεθόδου των Garon et al.[83] και η κόκκινη είναι η δική μας.

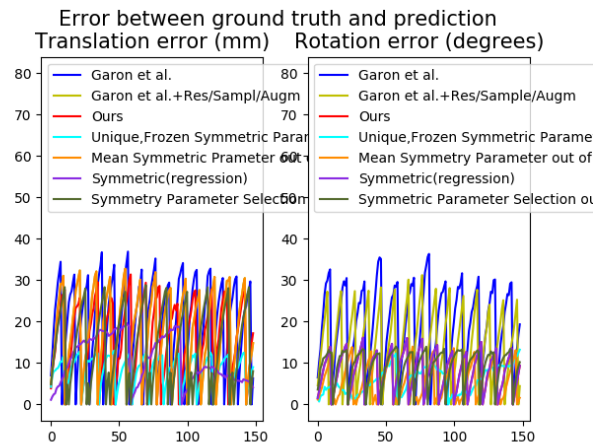
Το Μοντέλο του Κουτιού με τα Μπισκότα: η συμμετρική περίπτωση

Για την ειδική περίπτωση της περιστροφoανακλαστικής συμμετρίας ενός κυλινδρικού σχηματος, εκθέτουμε, επίσης, τα αποτελέσματά μας χωρίς και με ένα επιπλέον λογαριασμό του άξονα συμμετρίας του αντικειμένου, ως έναν εξτρά παράγοντα απροσδιοριστίας της εκτίμησης περιστροφής, όταν ορίζουμε την συνάρτηση περιστροφικής απώλειας. Βελτιώνουμε την προσέγγιση των Garon et al.[83] κατά 33.5% και 31%, κατά μέσο όρο, αντίστοιχα για τις μετατοπίσεις και τις περιστροφές, αν δεν λάβουμε υπόψην μας αυτή την αβεβαιότητα και κατά 66% και 50%, αντιστοίχως, αν λάβουμε υπόψην μας τον άξονα περιστροφικής συμμετρίας του αντικειμένου.

Αποτύπωση Λαθών (αριστερά:επαναρχικοποίηση κάθε 15 εικόνες) και (δεξιά:επαναρχικοποίηση κάθε φορά που ο παρακολουθητής αποτυγχάνει):

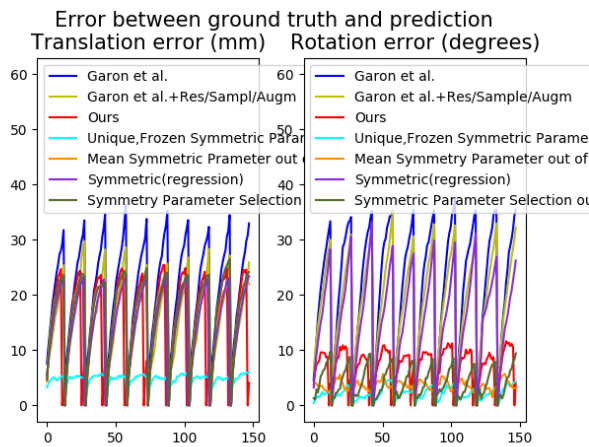


Σχήμα 1.35 Αποτύπωση Μετρικών 3-διάστατου Λάθους Μετατόπισης και Περιστροφής, για το σενάριο ‘Κοντινής Σταθερότητας’ του 3-διάστατου μοντέλου του ‘Κουτιού με τα Μπισκότα’, στην περίπτωση που ο παρακολουθητής επαναρχικοποιείται κάθε 15 εικόνες.

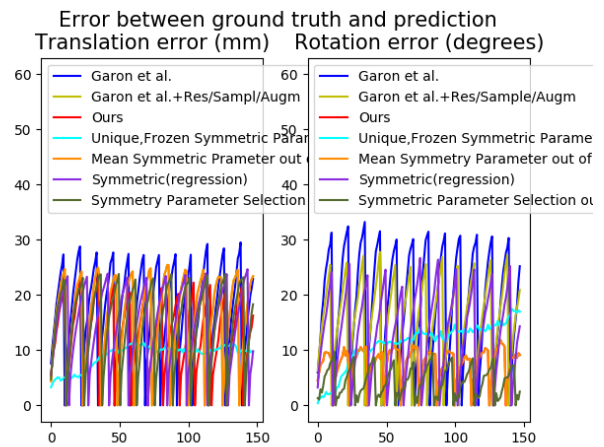


Σχήμα 1.36 Αποτύπωση Μετρικών 3-διάστατου Λάθους Μετατόπισης και Περιστροφής, για το σενάριο ‘Κοντινής Σταθερότητας’ του 3-διάστατου μοντέλου του ‘Κουτιού με τα Μπισκότα’, στην περίπτωση που ο παρακολουθητής επαναρχικοποιείται κάθε φορά που ο παρακολουθητής αποτυγχάνει.

Αποτύπωση Λαθών (αριστερά:επαναρχικοποίηση κάθε 15 εικόνες) και (δεξιά:επαναρχικοποίηση κάθε φορά που ο παρακολουθητής αποτυγχάνει):

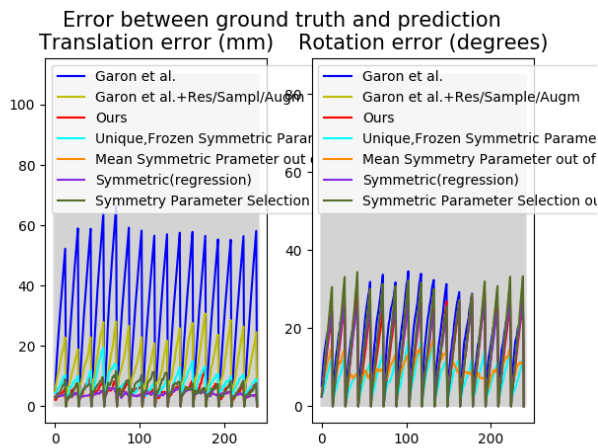


Σχήμα 1.37 Αποτύπωση Μετρικών 3-διάστατου Λάθους Μετατόπισης και Περιστροφής, για το σενάριο ‘Μακρινής Σταθερότητας’ του 3-διάστατου μοντέλου του ‘Κουτιού με τα Μπισκότα’, στην περίπτωση που ο παρακολουθητής επαναρχικοποιείται κάθε 15 εικόνες.

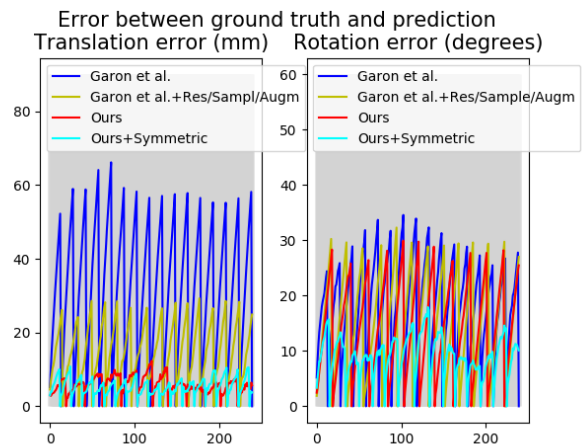


Σχήμα 1.38 Αποτύπωση Μετρικών 3-διάστατου Λάθους Μετατόπισης και Περιστροφής, για το σενάριο ‘Μακρινής Σταθερότητας’ του 3-διάστατου μοντέλου του ‘Κουτιού με τα Μπισκότα’, στην περίπτωση που ο παρακολουθητής επαναρχικοποιείται κάθε φορά που αποτυγχάνει.

Αποτύπωση Λαθών (αριστερά:επαναρχικοποίηση κάθε 15 εικόνες) και (δεξιά:επαναρχικοποίηση κάθε φορά που ο παρακολουθητής αποτυγχάνει):

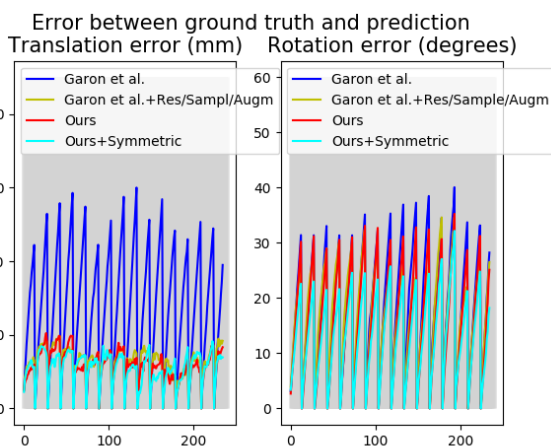


Σχήμα 1.39 Αποτύπωση Μετρικών 3-διάστατου Λάθους Μετατόπισης και Περιστροφής, για το σενάριο ‘Μακρινής Σταθερότητας’ του 3-διάστατου μοντέλου του ‘Κουτιού με τα Μπισκότα’, στην περίπτωση που ο παρακολουθητής επαναρχικοποιείται κάθε 15 εικόνες.

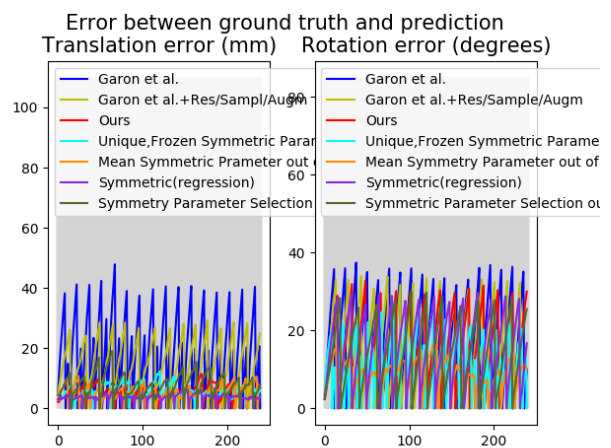


Σχήμα 1.40 Αποτύπωση Μετρικών 3-διάστατου Λάθους Μετατόπισης και Περιστροφής, για το σενάριο ‘Μακρινής Σταθερότητας’ του 3-διάστατου μοντέλου του ‘Κουτιού με τα Μπισκότα’, στην περίπτωση που ο παρακολουθητής επαναρχικοποιείται κάθε φορά που αποτυγχάνει.

Αποτύπωση Λαθών (αριστερά:επαναρχικοποίηση κάθε 15 εικόνες) και (δεξιά:επαναρχικοποίηση κάθε φορά που ο παρακολουθητής αποτυγχάνει):

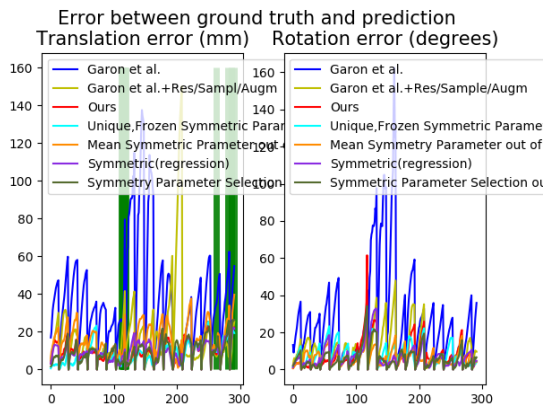


Σχήμα 1.41 Αποτύπωση Μετρικών 3-διάστατου Λάθους Μετατόπισης και Περιστροφής, για το σενάριο ‘Μακρινής Σταθερότητας’ του 3-διάστατου μοντέλου του ‘Κουτιού με τα Μπισκότα’, στην περίπτωση που ο παρακολουθητής επαναρχικοποιείται κάθε 15 εικόνες.

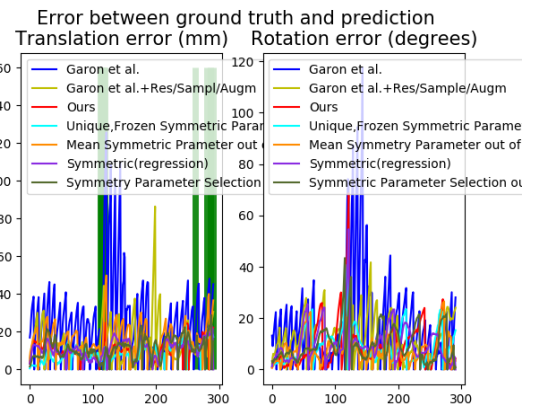


Σχήμα 1.42 Αποτύπωση Μετρικών 3-διάστατου Λάθους Μετατόπισης και Περιστροφής, για το σενάριο ‘Μακρινής Σταθερότητας’ του 3-διάστατου μοντέλου του ‘Κουτιού με τα Μπισκότα’, στην περίπτωση που ο παρακολουθητής επαναρχικοποιείται κάθε φορά που αποτυγχάνει.

Αποτύπωση Λαθών (αριστερά:επαναρχικοποίηση κάθε 15 εικόνες) και (δεξιά:επαναρχικοποίηση κάθε φορά που ο παρακολουθητής αποτυγχάνει):

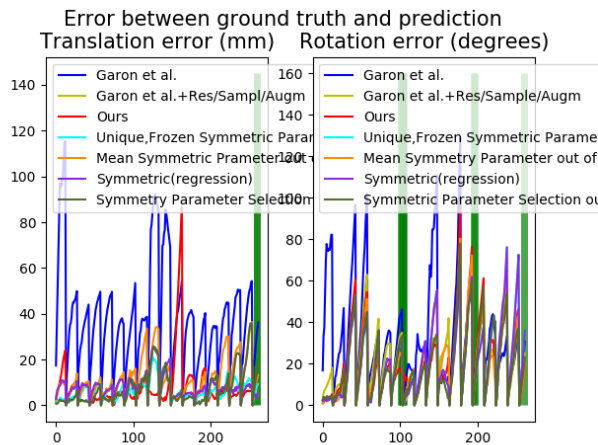


Σχήμα 1.43 Αποτύπωση Μετρικών 3-διάστατου Λάθους Μετατόπισης και Περιστροφής, για το σενάριο ‘Μόνο Μετατόπισης’ του 3-διάστατου μοντέλου του ‘Κουτιού με τα Μπισκότα’, στην περίπτωση που ο παρακολουθητής επαναρχικοποιείται κάθε 15 εικόνες.

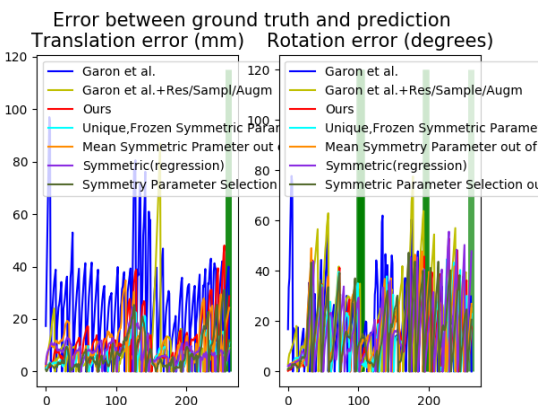


Σχήμα 1.44 Αποτύπωση Μετρικών 3-διάστατου Λάθους Μετατόπισης και Περιστροφής, για το σενάριο ‘Μόνο Μετατόπισης’ του 3-διάστατου μοντέλου του ‘Κουτιού με τα Μπισκότα’, στην περίπτωση που ο παρακολουθητής επαναρχικοποιείται κάθε φορά που ο παρακολουθητής αποτυγχάνει.

Αποτύπωση Λαθών (αριστερά:επαναρχικοποίηση κάθε 15 εικόνες) και (δεξιά:επαναρχικοποίηση κάθε φορά που ο παρακολουθητής αποτυγχάνει):

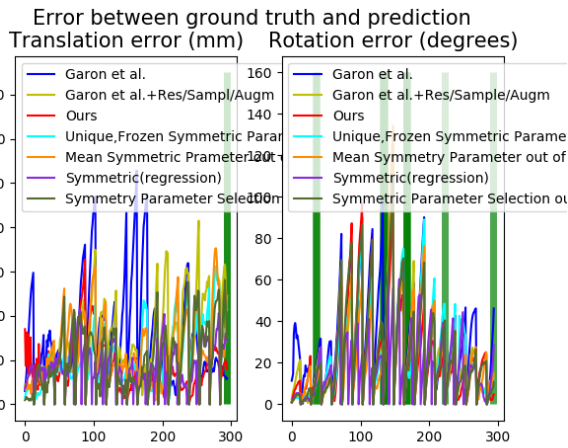


Σχήμα 1.45 Αποτύπωση Μετρικών 3-διάστατου Λάθους Μετατόπισης και Περιστροφής, για το σενάριο ‘Μόνο Περιστροφών’ του 3-διάστατου μοντέλου του ‘Κουτιού με τα Μπισκότα’, στην περίπτωση που ο παρακολουθητής επαναρχικοποιείται κάθε 15 εικόνες.

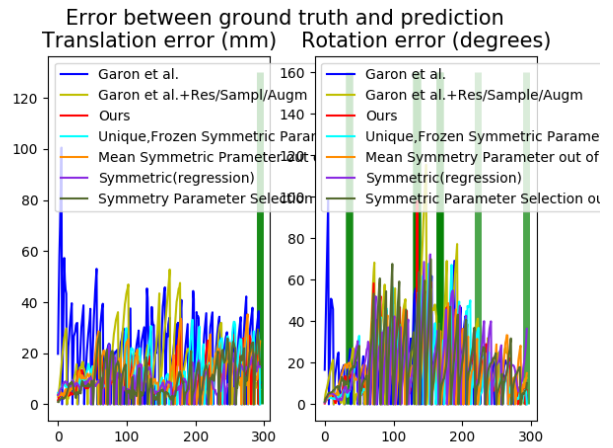


Σχήμα 1.46 Αποτύπωση Μετρικών 3-διάστατου Λάθους Μετατόπισης και Περιστροφής, για το σενάριο ‘Μόνο Περιστροφών’ του 3-διάστατου μοντέλου του ‘Κουτιού με τα Μπισκότα’, στην περίπτωση που ο παρακολουθητής επαναρχικοποιείται κάθε φορά που ο παρακολουθητής αποτυγχάνει.

Αποτύπωση Λαθών (αριστερά:επαναρχικοποίηση κάθε 15 εικόνες) και (δεξιά:επαναρχικοποίηση κάθε φορά που ο παρακολουθητής αποτυγχάνει):

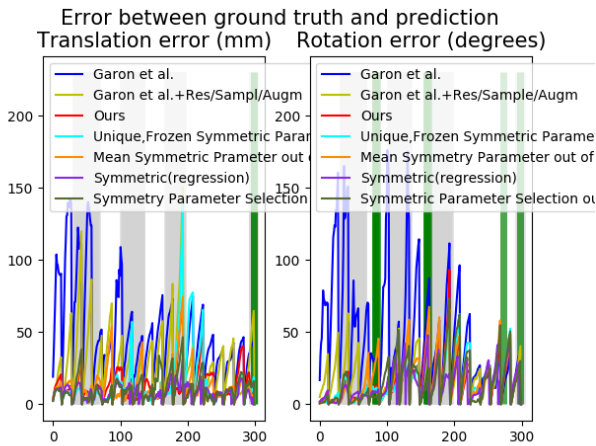


Σχήμα 1.47 Αποτύπωση Μετρικών 3-διάστατου Λάθους Μετατόπισης και Περιστροφής, για το σενάριο ‘Πλήρους Αλληλεπίδρασης’ του 3-διάστατου μοντέλου του ‘Κουτιού με τα Μπισκότα’, στην περίπτωση που ο παρακολουθητής επαναρχικοποιείται κάθε 15 εικόνες.

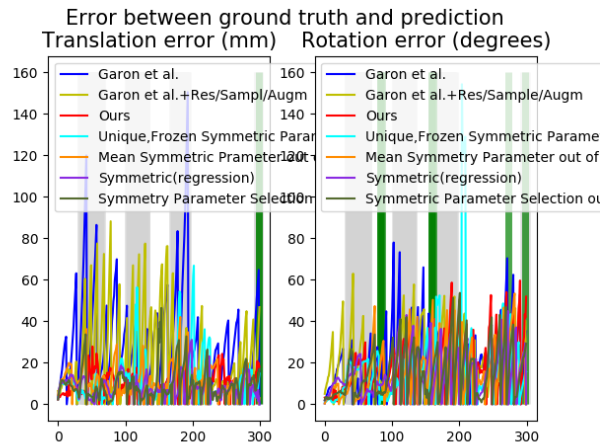


Σχήμα 1.48 Αποτύπωση Μετρικών 3-διάστατου Λάθους Μετατόπισης και Περιστροφής, για το σενάριο ‘Πλήρους Αλληλεπίδρασης’ του 3-διάστατου μοντέλου του ‘Κουτιού με τα Μπισκότα’, στην περίπτωση που ο παρακολουθητής επαναρχικοποιείται κάθε φορά που ο παρακολουθητής αποτυγχάνει.

Αποτύπωση Λαθών (αριστερά:επαναρχικοποίηση κάθε 15 εικόνες) και (δεξιά:επαναρχικοποίηση κάθε φορά που ο παρακολουθητής αποτυγχάνει):

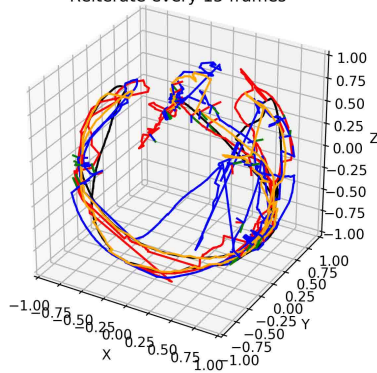


Σχήμα 1.49 Αποτύπωση Μετρικών 3-διάστατου Λάθους Μετατόπισης και Περιστροφής, για το σενάριο ‘Δύσκολη Αλληλεπίδρασης’ του 3-διάστατου μοντέλου του ‘Κουτιού με τα Μπισκότα’, στην περίπτωση που ο παρακολουθητής επαναρχικοποιείται κάθε 15 εικόνες.

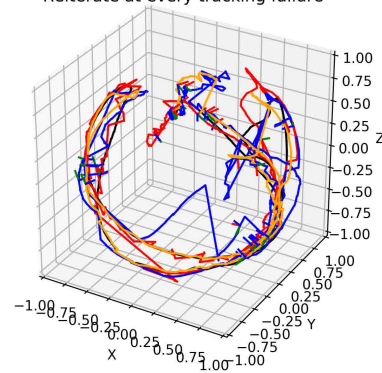


Σχήμα 1.50 Αποτύπωση Μετρικών 3-διάστατου Λάθους Μετατόπισης και Περιστροφής, για το σενάριο ‘Δύσκολη Αλληλεπίδρασης’ του 3-διάστατου μοντέλου του ‘Κουτιού με τα Μπισκότα’, στην περίπτωση που ο παρακολουθητής επαναρχικοποιείται κάθε φορά που ο παρακολουθητής αποτυγχάνει.

6D Temporal Pose Tracking Visualization
Cookie Jar --- Hard Interaction scenario
Reiterate every 15 frames



6D Temporal Pose Tracking Visualization
Cookie Jar --- Hard Interaction scenario
Reiterate at every tracking failure

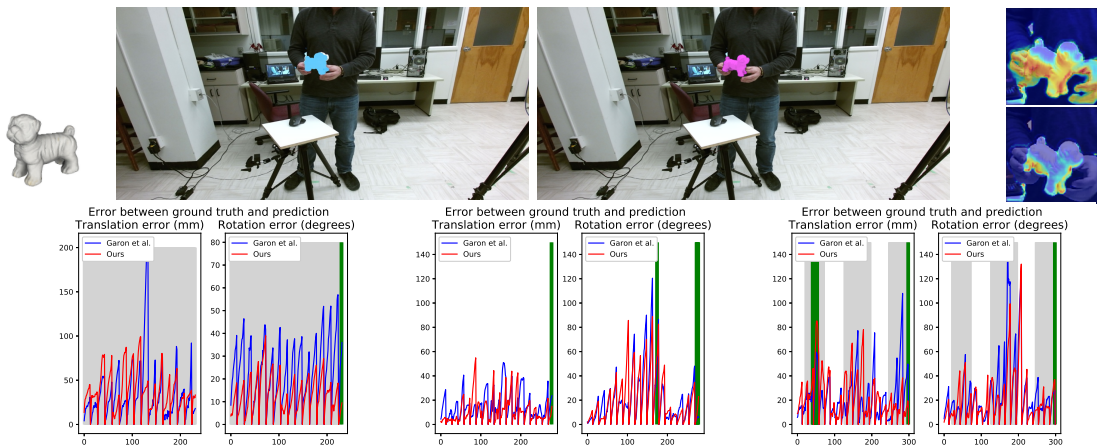


Σχήμα 1.51: Οι 6-Δ Τροχιές Πόζας του μοντέλου του ‘Κουτιού με τα Μπισκότα’, στο χρόνο, για το σενάριο των ‘Δύσκολων Αλληλεπιδράσεων’. Είναι προφανές, και στις 2 περιπτώσεις, και από την σκοπιά αυτή, πως η προτεινόμενη προσέγγισή μας, αυτή ξεπερνά σε απόδοση εκείνη των Garon et al. [83] κατά πολύ, μιας και παράγει μικρότερα λάθη και λιγότερες αποτυχίες. Όλες οι μονάδες μέτρησης μήκους είναι σε χιλ. και οι περιστροφές σε μοίρες. Οι **μαύρη** τροχιά είναι η πραγματική, η **μπλε** τροχιά είναι η βασική τροχιά σύγκρισης της μεθόδου των Garon et al.[83], η **κόκκινη** είναι η δική μας και, προαιρετικά, η **πορτοκαλί** είναι για τα αντικείμενα με συνεχώς περιστροφική συμμετρία, όταν αυτή λαμβάνεται υπόψη στη σχεδιάσή μας.

1.1 Αξιολόγηση των αλγορίθμων μας σε άλλα αντικείμενα: ο ‘Σκύλος’, το ‘Σύμπλεγμα των Lego’, το ‘Ποτιστήρι’

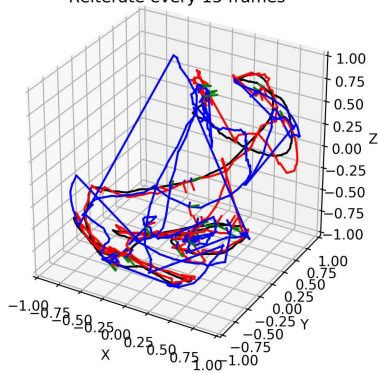
Στη συνέχεια, προκειμένου να καταστήσουμε δεδομένη την αποδοτικότητα του παρακολουθητή μας και να επιδείξουμε πλήρως τις ικανότητες γενίκευσης, αναφέρουμε τα αποτελέσματα του σε 3 επιπλέον αντικείμενα : το ‘Σκύλο’, το ‘Σύμπλεγμα των Lego’, το ‘Ποτιστήρι’ του συνόλου δεδομένων που παρέχουν οι Garon et al.[83]. Στις επόμενες εικόνες, κάποιος θα μπορούσε να παρατηρήσει τις εκτιμώμενες πόζες που παρέχονται από το κατεστημένο παρακολουθητή και τον δικό μας, ενσωματωμένο σε ένα ‘Παρατηρούμενο’ RGB πλαίσιο εικόνας το καθένα. Επιπλέον, επιδεικνύουμε το αντίστοιχο ζεύγος Οπτικής Προσοχής Παρασκηνίου και Χειρισμού Επικαλύψεων, όπως επίσης και των τριών **τυχαίως επιλεγόμενων** απεικονήσεων λαθών (με μια επαναρχικοποίηση κάθε 15 εικόνες). Τελικώς, παρουσιάζουμε τα 3-διάστατα λάθη Μετατόπισης και Περιστροφής, όπως επίσης και τις συνολικές περιπτώσεις ολοσχερούς αποτυχίας του παρακολουθητή, για τα πιο δύσκολα σενάρια του συνόλου δεδομένων: τα δύο σενάρια για ‘75% Επι κάλυψη’ και όλα τα σενάρια ‘Αλληλεπίδρασης’.

1.1.1 Μοντέλο Σκύλου

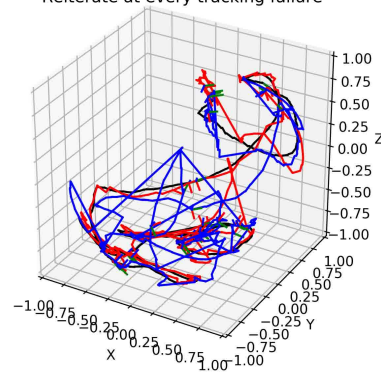


Σχήμα 1.52: Σύγκριση μεταξύ της ως τώρα βέλτιστης μεθόδου, των Garon et al.[83] (γαλάζιο) και της δικής μας (ροζ) για το ‘Σκύλο’ σε 3 σενάρια: ‘75% Κάθετη Επικάλυψη’, ‘Μόνον Περιστροφές’ και ‘Δύσκολη Αλληλεπίδραση’.

6D Temporal Pose Tracking Visualization
Dog --- Hard Interaction scenario
Reiterate every 15 frames

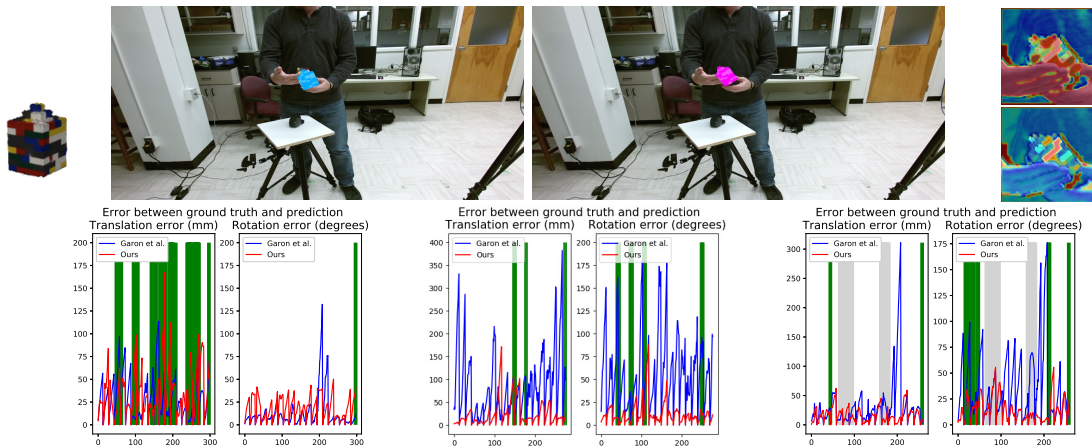


6D Temporal Pose Tracking Visualization
Dog --- Hard Interaction scenario
Reiterate at every tracking failure



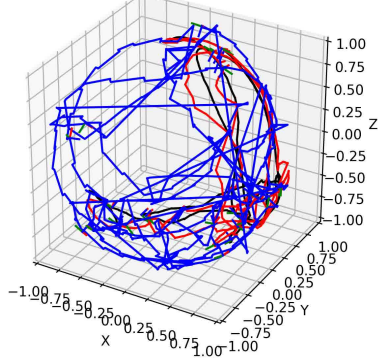
Σχήμα 1.53: Οι 6-Δ Τροχιές Πόζας του μοντέλου του ‘Σκύλου’, στο χρόνο, για το σενάριο των ‘Δύσκολων Αλληλεπιδράσεων’. Είναι προφανές, και στις 2 περιπτώσεις, και από την σκοπιά αυτή, πως η προτεινόμενη προσέγγισή μας, αυτή ξεπερνά σε απόδοση εκείνη των Garon et al. [83] κατά πολύ, μιας και παράγει μικρότερα λάθη και λιγότερες αποτυχίες. Όλες οι μονάδες μέτρησης μήκους είναι σε χιλ. και οι περιστροφές σε μοίρες. Οι **μαύρη** τροχιά είναι η πραγματική, η **μπλε** τροχιά είναι η βασική τροχιά σύγκρισης της μεθόδου των Garon et al.[83], και η **κόκκινη** είναι η δική μας.

1.1.2 Μοντέλο ‘Lego’

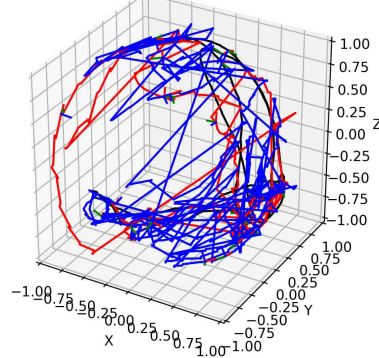


Σχήμα 1.54: Σύγκριση μεταξύ της εως τώρα βέλτιστης μεθόδου, των Garon et al.[83] (γαλάζιο) και της δικής μας (ροζ) για το ‘Σύμπλεγμα από Lego’ σε 3 σενάρια: ‘Μόνον Μετατοπίσεις’, ‘Πλήρης’ και ‘Δύσκολη Αλληλεπίδραση’.

6D Temporal Pose Tracking Visualization
Lego --- Hard Interaction scenario
Reiterate every 15 frames

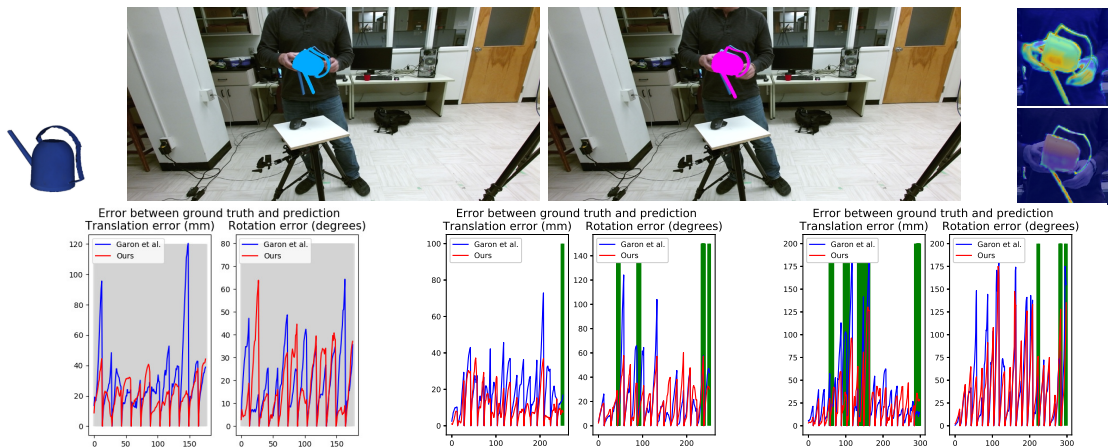


6D Temporal Pose Tracking Visualization
Lego --- Hard Interaction scenario
Reiterate at every tracking failure



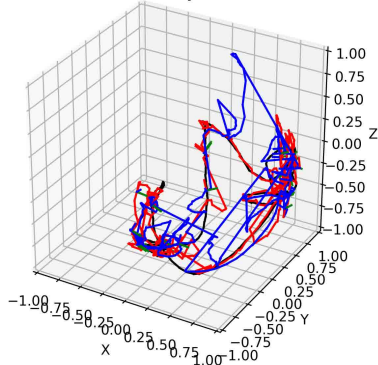
Σχήμα 1.55: Οι 6-Δ Τροχιές Πόζας του μοντέλου του ‘Block από Legos’, στο χρόνο, για το σενάριο των ‘Δύσκολων Αλληλεπιδράσεων’. Είναι προφανές, και στις 2 περιπτώσεις, και από την σκοπιά αυτή, πως η προτεινόμενη προσέγγισή μας, αυτή ξεπερνά σε απόδοση εκείνη των Garon et al. [83] κατά πολύ, μιας και παράγει μικρότερα λάθη και λιγότερες αποτυχίες. Όλες οι μονάδες μέτρησης μήκους είναι σε χιλ. και οι περιστροφές σε μοίρες. Οι **μαύρη** τροχιά είναι η πραγματική, η **μπλε** τροχιά είναι η βασική τροχιά σύγκρισης της μεθόδου των Garon et al.[83] και η **κόκκινη** είναι η δική μας.

1.1.3 Μοντέλο Ποτιστηριού

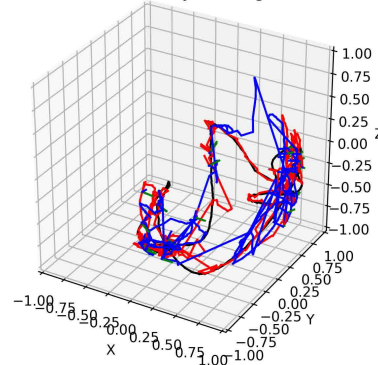


Σχήμα 1.56: Σύγκριση μεταξύ της έως τώρα βέλτιστης μεθόδου, των Garon et al.[83] (γαλάζιο) και της δικής μας (ροζ) για το 'Ποτιστήρι' σε 3 σενάρια: '75% Οριζόντια Επικάλυψη', 'Μόνον Περιστροφές' και 'Πλήρης Αλληλεπίδραση'.

6D Temporal Pose Tracking Visualization
Watering Can --- Hard Interaction scenario
Reiterate every 15 frames



6D Temporal Pose Tracking Visualization
Watering Can --- Hard Interaction scenario
Reiterate at every tracking failure



Σχήμα 1.57: Οι 6-Δ Τροχιές Πόζας του μοντέλου του 'Ποτιστηριού', στο χρόνο, για το σενάριο των 'Δύσκολων Αλληλεπιδράσεων'. Είναι προφανές, και στις 2 περιπτώσεις, και από την σκοπιά αυτή, πως η προτεινόμενη προσέγγισή μας, αυτή ξεπερνά σε απόδοση εκείνη των Garon et al. [83] κατά πολύ, μιας και παράγει μικρότερα λάθη και λιγότερες αποτυχίες. Όλες οι μονάδες μέτρησης μήκους είναι σε χιλ. και οι περιστροφές σε μοίρες. Οι **μαύρη** τροχιά είναι η πραγματική, η **μπλε** τροχιά είναι η βασική τροχιά σύγκρισης της μεθόδου των Garon et al.[83], η **κόκκινη** είναι η δική μας και, προαιρετικά, η **πορτοκαλί** είναι για τα αντικείμενα με συνεχώς περιστροφική συμμετρία, όταν αυτή λαμβάνεται υπόψη στη σχεδίασή μας.

Ποσοτικά αποτελέσματα για τα υπόλοιπα τρία αντικείμενα μελέτης:

Προσέγγιση	75% Οριζόντια Επικάλυψη			75% Κάθετη Επικάλυψη		
	Λάθος Μετατόπισης(χιλ.)	Λάθος Περιστροφής(μοίρες)	Αποτυχίες	Λάθος Μετατόπισης(χιλ.)	Λάθος Περιστροφής(μοίρες)	Αποτυχίες
Garon et al.[83]('Σκύλος')	37.96 ± 23.39	47.94 ± 31.55	21	32.84 ± 34.07	22.44 ± 13.60	21
Δικό μας ('Σκύλος')	24.43 ± 18.92	17.24 ± 12.41	25	36.53 ± 22.39	12.67 ± 7.95	20
Garon et al.[83]('Σύμπλεγμα από Lego')	68.25 ± 46.97	40.04 ± 47.37	28	40.04 ± 47.37	35.30 ± 31.32	20
Δικό μας ('Σύμπλεγμα από Lego')	72.04 ± 34.10	18.41 ± 13.84	28	12.92 ± 5.73	12.92 ± 9.02	20
Garon et al.[83]('Ποτιστήρι')	21.59 ± 11.32	23.99 ± 16.95	14	32.76 ± 24.12	26.74 ± 19.05	18
Δικό μας ('Ποτιστήρι')	20.71 ± 10.24	17.00 ± 18.99	13	17.66 ± 17.95	13.46 ± 10.43	12

Προσέγγιση	Αλληλεπίδραση Μετατόπισης			Περιστροφική Αλληλεπίδραση		
	Λάθος Μετατόπισης(χιλ.)	Περιστροφικό Λάθος(μοίρες)	Αποτυχίες	Λάθος Μετατόπισης(χιλ.)	Περιστροφικό Λάθος(μοίρες)	Αποτυχίες
Garon et al. [83] ('Σκύλος')	58.87 ± 71.86	16.42 ± 13.51	20	11.16 ± 10.28	20.00 ± 21.31	17
Δικό μας ('Σκύλος')	21.64 ± 22.78	9.27 ± 8.03	14	10.68 ± 7.53	20.07 ± 19.29	17
Garon et al. [83] ('Σύμπλεγμα από Lego')	27.90 ± 23.53	11.89 ± 18.50	29	16.42 ± 10.90	17.83 ± 15.90	32
Δικό μας ('Σύμπλεγμα από Lego')	22.66 ± 24.58	9.08 ± 7.60	12	10.13 ± 6.79	7.22 ± 4.55	4
Garon et al. [83] ('Ποτιστήρι')	24.95 ± 42.91	13.26 ± 11.34	16	13.14 ± 8.99	22.19 ± 25.93	15
Δικό μας ('Ποτιστήρι')	24.30 ± 21.51	8.79 ± 6.35	16	12.22 ± 9.46	18.66 ± 15.51	15

Προσέγγιση	Πλήρης Αλληλεπίδραση			Δύσκολη Αλληλεπίδραση		
	Λάθος Μετατόπισης(χιλ.)	Λάθος Περιστροφών(μοίρες)	Αποτυχίες	Λάθος Μετατόπισης(χιλ.)	Λάθος Περιστροφών(μοίρες)	Αποτυχίες
Γαρον et al. [83] ('Σκύλος')	37.73 ± 42.32	20.77 ± 19.66	23	23.95 ± 38.86	24.38 ± 26.39	20
Δικός μας ('Σκύλος')	24.88 ± 35.85	28.52 ± 25.38	20	19.32 ± 15.97	19.72 ± 20.17	19
Γαρον et al. [83] ('Lego')	30.96 ± 31.44	22.10 ± 20.20	20	30.71 ± 42.62	36.38 ± 34.99	20
Δικός μας ('Lego')	23.58 ± 27.73	11.80 ± 12.28	13	16.47 ± 12.95	14.29 ± 11.68	11
Γαρον et al. [83] ('Ποτιστήρι')	33.76 ± 37.62	40.16 ± 35.90	26	28.31 ± 19.49	23.04 ± 24.27	28
Δικός μας ('Ποτιστήρι')	19.82 ± 19.98	28.76 ± 30.27	26	18.03 ± 14.99	19.57 ± 17.47	23

Πίνακας 1.2: 3-διάστατα λάθη Μετατόπισης και Περιστροφών και συνολικές αποτυχίες παρακολούθησης σε έξι διαφορετικά σενάρια για τα τρία τελικά αντικείμενα στα οποία δοκιμάζουμε τον αλγόριθμό μας.

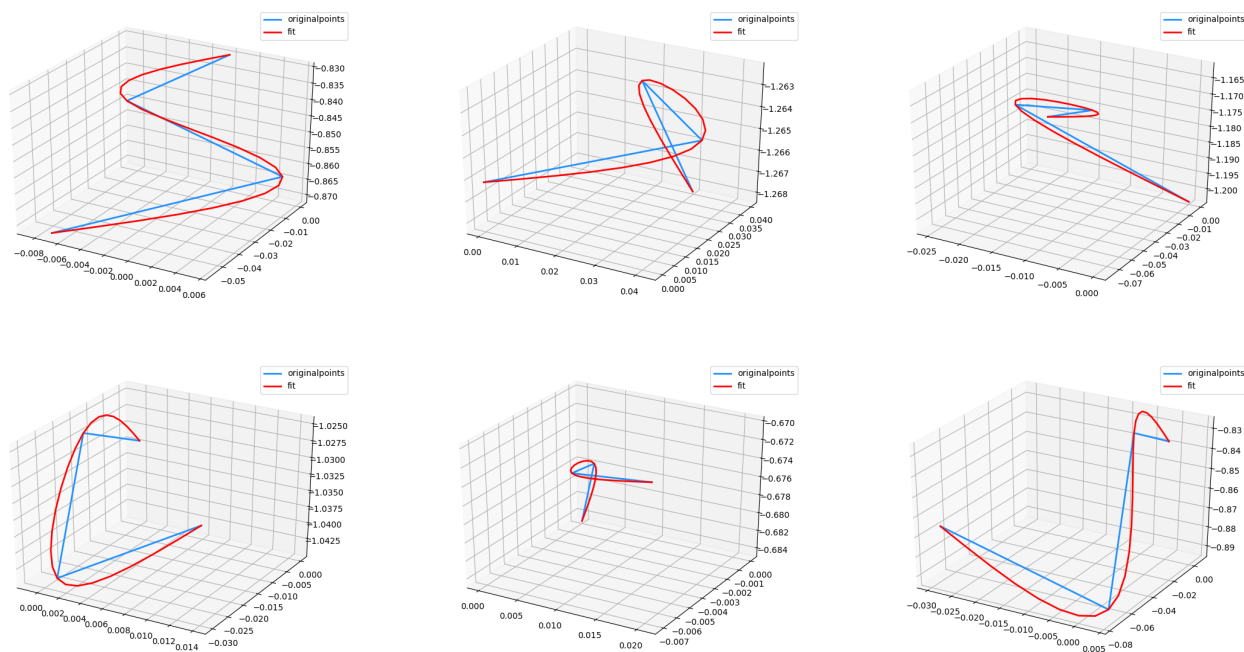
Όπως έχουμε δει, η ακρίβεια της προσέγγισής μας ξεπερνάει αυτή της εως τώρα βέλτιστης των Garon et al. [83], σε άπαντα τα αντικείμενα και σχεδόν σε κάθε σενάριο ανά αντικείμενο, ειδικά για τις συνιστώσες της 3-διάστατης περιστροφής (οι οποίες εν γένει θεωρούνται και αυτές που επιφέρουν τις υψηλότερες προκλήσεις). Σύμφωνα με τον Πίνακα 1.2, και οι δύο μετρικές λάθους που αναφέρονται είναι γενικά χαμηλότερες (και σε όρους μέσης τιμής και τυπικής απόκλισης) και ο παρακολουθητής μας αποτυγχάνει το ίδιο, ή κατα κύριο λόγο, λιγότερο συχνά. Η προσέγγισή μας παρουσιάζει υψηλότερα λάθη στις περιπτώσεις ταχείας κίνησης του αντικειμένου, σε λιγότερες περιπτώσεις από την δουλειά των Garon et al. [83] και χειρίζεται τόσο τις στατικές όσο και τις δυναμικές επικαλύψεις, υψηλού ποσοστού, πιο επιτυχημένα. Η αποδοτικότητα της μεθόδου μας επαληθεύεται από το γεγονός ότι, όχι μόνο καταφέρνει να κρατήσει εύρωστη την εκτίμηση της 3-διάστατης θέσης του αντικειμένου υπο το βάρος ισχυρών επικαλύψεων, αλλά επεκτείνει την ιδιότητα αυτή και στην έταιρη συνιστώσα, των 3-διάστατων περιστροφών στο χρόνο.

Υπό μια συνολική θεώρηση, αυταπόδεικτα, το αντικείμενο που επωφελείται τα πλείστα από τις μεθοδολογικές μας βελτιώσεις είναι ο 'Δράκος'. Αυτό δε θα πρέπει να μας εκπλήσσει, καθώς είναι το αντικείμενο με την πλέον περίπλοκη γεωμετρία, την πλουσιότερη υφή και τα πλέον χαρακτηριστικά επεκταταμένα μέρη σώματος (πχ. ουρά, φτερά κλπ) τα οποία εκφεύγουν της λαβής του χρήστη. Έτσι, και η γεωμετρική μοντελοποίηση της Συνάρτησης Απωλείας 3-διάστατων Περιστροφών μέσω της Γεωδετικής Απόστασης και οι παράλληλες μονάδες Οπτικής Προσοχής βρίσκουν την βέλτιστη εφαρμογή τους στην περίπτωση αυτή. Όταν το χέρι του χρήστη επικαλύπτει μέρος του 'Δράκου', η Οπτική Προσοχή στρέφεται στα μέρη του σώματός του εκείνα τα οποία παρουσιάζουν ενδιαφέρον για τον παρακολουθητή, ξαθώς βρίσκονται εκτός της λαβής του. Τέτοια είναι ο λαιμός, η ουρά και τα δύο φτερά του 'Δράκου'. Για το συμμετρικό μοντέλο του 'Κουτιού με τα Μπισκότα', οι διαφορές μεταξύ της μεθόδου μας και της βασικής μεθόδου με την οποία συγκρινόμαστε είναι χαμηλότερες. Η επίδραση των δύο μονάδων Οπτικής Προσοχής είναι λιγότερο εμφανής εδώ και αφού το μοντέλο είναι απλούστερης φύσης, με συμμετρικό σχήμα και φτωχή κι επαναλαμβανόμενη υφή, είναι επόμενο να έχουν και ισχνότερο αποτέλεσμα. Γιατί τα διακριτά χαρακτηριστικά του 'Δράκου' αντικαθίστανται από ασάφειες οι οποίες

με τη σειρά τους αρνούνται στις δύο συζυγείς μονάδες την δυνατότητα να αναγνωρίσουν εύκολα την πόζα του αντικειμένου. Μαζί με το ‘Lego’, αποτελούν τα δύο αντικείμενα που επωφελούνται τα μέγιστα από τον ευριστικό αλγόριθμο χειρισμού των ανακλαστικών συμμετριών, καθώς έτσι αποφεύγονται απότομα υψηλά λάθη που ο αλγόριθμος του παρακολουθητή θα προωθούσε και στις μελλοντικές του προβλέψεις. Παρόλο που το ΣΝΔ συγκεντρώνει την προσοχή του κυρίως στο σχήμα του αντικειμένου για να κατανοήσει την αλλαγή της πόζας του από εικόνα σε εικόνα, η εμφάνιση αποδεικνύεται πειραματικά να παίζει δραστικό ρόλο στην εκλέπτυνση των εκτιμήσεων του παρακολουθητή, μιας και τα λάθη των μοντέλων του ‘Σκύλου’ και του ‘Ποτιστηριού’, των δύο φτωχότερων σε υφή μοντέλων δηλαδή, ελαττώνονται ελαφρύτερα μετά από τις βελτιώσεις μας. Ο χάρτης βαρων για την Εξαγωγή Προσκήνιου βοηθά να ξεδιαλυθούν το οπτικά χαρακτηριστικά του μοντέλου του ‘Σκύλου’ από το παρασκήνιο, στο οποίο κυριαρχεί ένα τραπέζι ίδιου χρώματος και ίδιας πενίας υφής, στα σενάρια των ‘Επικαλύψεων κατά 75%’. Από την άλλη πλευρά βέβαια, για το ‘Ποτιστήρι’ παρατηρείται πως παρουσιάζει τις μεγαλύτερες τους ασάφειες όταν το στόμιο του αντικρίζει ευθεία τον αισθητήρα RGB-D και έτσι κρύβεται από την 2-διάστατη προβολή του μοντέλου στο πεδίο της εικόνας, λόγω της αλλαγής αυτής της γωνίας θέασης. Τότε η επίδραση και των δύο μονάδων Οπτικής Προσοχής, αλλά και της Γεωμετρικού μοντελοποίησης της εκμάθησης Περιστροφικών συνιστωσών απαλείφεται πρακτικά, καθώς εν τις πράγματι εισάγεται ένα τρίτο είδος μεταβλητής περιστροφικής συμμετρίας, εξαρτώμενης από τη γωνία θέασης και το ποσοστό απόκρυψης των χαρακτηριστικών του μοντέλου αντί για την εν γένει γεωμετρία του.

Διερεύνηση Μελλοντικών Επεκτάσεων: Εκμετάλλευση Δυναμικών Μακράς Διάρκειας μέσω Εκμάθησης Συνεχών 3-διάστατων Τροχιών

• Παρεμβολή 3-διάστατων Συνεχών Μετατοπίσεων



Σχήμα 1.58: Παραδείγματα 3-διάστατης συνεχούς τροχιάς θέσης. Η μπλε γραμμή παραβάλλει γραμμικά μεταξύ τεσσάρων διακριτών σημείων (σε μέτρα) και η κόκκινη γραμμή αναπαριστά τις λείες τροχιές που περνούν και από τα 4 αυτά σημεία, χρησιμοποιώντας κυβικές σπλίνες.

• Παρεμβολή 3-διάστατων Συνεχών Περιστροφών

Σε ότι αφορά τις περιστροφές, η σχεδιαστική απόφαση που παίρνουμε είναι, λόγω της επαναληπτικής, μη Ευκλείδειας φύσης τους, να ακολουθούν την Γεωδετική τροχιά μεταξύ μιας αρχικής και μιας τελικής διάταξης για ένα χρονικό διάστημα των 25 πλαισίων εικόνας.

Αν συλλογιστούμε μια ΕΕΟ L (3) και μας δοθεί ένα ζεύγος ποζών P_1, P_2 που ζουν σε αυτήν:

$$P_1 = \begin{bmatrix} R_1 & \mathbf{t}_1 \\ \mathbf{0}^T & 1 \end{bmatrix} \text{ και } P_2 = \begin{bmatrix} R_2 & \mathbf{t}_2 \\ \mathbf{0}^T & 1 \end{bmatrix}, \text{ η κίνηση του αντικειμένου που ξεκινά από τη}$$

μία και καταλήγει στην άλλη, $P(t) = \begin{bmatrix} R(t) & \mathbf{t}(t) \\ \mathbf{0}^T & 1 \end{bmatrix}$ συνεχίζει να κινείται στην ίδια πολλαπλότητα.

Αν θεωρήσουμε το χρονοεξαρτώμενο διάνυσμα ταχύτητας επί της τροχιάς αυτής :

$$\mathbf{V}(t) = \begin{bmatrix} \omega \\ v \end{bmatrix}, \text{ που το μέτρο του είναι:}$$

$$V(t) = \frac{dP(t)}{dt}. \quad (1.20)$$

Τότε η τροχιά της συντομότερου (γεωδετικού) μονοπατιού είναι αυτή που ελαχιστοποιεί την μετρική Riemman:

$$J = \int_{t_0}^T \langle \mathbf{V}(t), \mathbf{V}(t) \rangle dt = \int_{t_0}^T \mathbf{V}^T(t) \mathbb{W} \mathbf{V}(t) dt, \quad (1.21)$$

που έχει προταθεί ήδη από την εργασία των Park et al. [96], όπου $t_0 = 0, T=24$ και ο πίνακας \mathbb{W} είναι ένας διαγώνιος πίνακας της μορφής :

$$\mathbb{W} = \begin{bmatrix} \alpha \mathbb{I} & 0 \\ 0 & \beta \mathbb{I} \end{bmatrix}. \quad (1.22)$$

με α, β τις θετικές πραγματικές τιμές που κλιμακώνουν τις ταχύτητες μετατόπισης και γωνίας. Για τις παραμέτρους αυτές επιλέγουμε αυθαίρετα $\beta = 1$ and $\alpha = \max_{i=1}^3 \{\lambda_{\Lambda}^{(i)}\}$, όπου Λ ο Αδρανειακός Τένσορας του αντικειμένου. Σύμφωνα με την εργασία των Zefran et al. [168], η επιλογή αυτή δεν παίζει ρόλο στην ελαχιστοποίηση του συναρτησιακού, καθώς αυτή γίνεται από η λύση του συστήματος διαφορικών εξισώσεων :

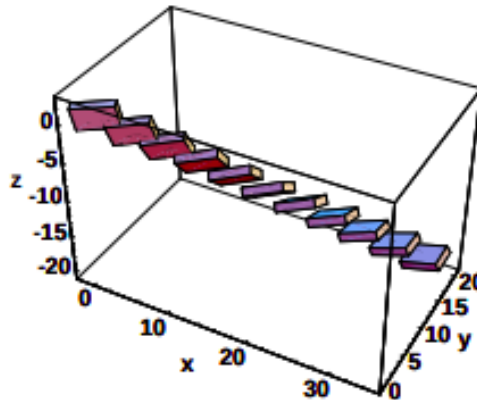
$$\begin{aligned} \frac{d\omega}{dt} &= 0 \\ \frac{dv}{dt} &= -\omega \times v, \end{aligned} \quad (1.23)$$

Εμάς μας αφορά μόνο το περιστροφικό τμήμα, οπότε η παρεμβολή της συνιστώσας αυτής επί του γεωδεσικού μονοπατιού, δηλαδή του μονοπατιού ελαχίστου μήκους, της ΕΟΟ L (3) δίνεται από την εξίσωση :

$$R(t) = R_1 e^{\Omega_0 t}, \quad (1.24)$$

όπου λογίζουμε ως $\Omega_0 = \log(R_1^T R_2)$ και το $\log(\cdot)$ αναφέρεται στον λογαριθμικό πίνακα (λεπτομερής αναφορά τον Τομέα 2.11).

Στο επόμενο γράφημα παρατίθεται η γραφική αναπαράσταση της γεωδαιτικής περιστροφικής κίνησης της πόζας ενός αντικειμένου :



Σχήμα 1.59: Μια γραφική αναπαράσταση της περιστροφικής γεωδαιτικής 3-διάστατης τροχιάς που ακολουθεί ένα αντικείμενο για να μετασχηματίσει την πόζα του από μία αρχική κατάσταση (πάνω αριστερή γωνία) σε μια αντίστοιχη τελική κατάσταση (κάτω δεξιά).[168]

Διερεύνηση Μελλοντικών Επεκτάσεων: Εκμετάλλευση Δυναμικών Μικρής Διάρκειας μέσω Ένωσης των Ρευμάτων Εμφάνισης-Βάθους με επιπλέον Ρεύματα Οπτικής Ροής



Σχήμα 1.60: Η 'Παρατηρούμενη' συνθετικής πυκνή εικόνα Οπτικής Ροής παραγόμενη περνώντας δύο συνεχόμενες 'Παρατηρούμενες' RGB εικόνες, τις οποίες γεννούμε από κόνες, τις οποίες γεννούμε από τις συνεχούς τροχιές της προηγούμενου κεφαλαίου, μες από ένα παγωμένο και προεκπαιδευμένο FlowNet2.

Σχήμα 1.61: Η 'Προβλεπόμενη' συνθετικής πυκνή εικόνα Οπτικής Ροής παραγόμενη περνώντας δύο συνεχόμενες 'Προβλεπόμενες' RGB εικόνες, τις οποίες γεννούμε από τις συνεχούς τροχιές της προηγούμενου κεφαλαίου, μες από ένα παγωμένο και προεκπαιδευμένο FlowNet2.

Σχήμα 1.62: Η συνθετική πυκνή εικόνα Οπτικής Ροής Επαυξημένης Πραγματικότητας (EAP), παραγόμενη από το πέρασμα μιας 'Παρατηρούμενης' και μιας 'Προβλεπόμενης' RGB εικόνας, διαμέσου ενός προεκπαιδευμένου και παγωμένου FlowNet2.

Μια δεύτερη σκέψη, με την οποία πειραματιζόμαστε, στην Διπλωματική αυτή, με σκοπό την επέκταση της παραπάνω προσέγγισης, είναι να δοκιμάσουμε να μοντελοποιήσουμε με τρόπο συγκεκριμένο τις χρονικές δυναμικές μικρού εύρους με στόχο να αντιμετωπιστεί η ολίσθηση πόζας. Με αυτό στο μυαλό, αντιμετωπίζουμε την επιπλέον πηγή πληροφορίας που προβάλλει την 3-διάστατη διαφορά πόζας στο 2-διάστατο επίπεδο της εικόνας με έναν τρόπο άμεσο: τις εικόνες (Πυκνής) Οπτικής Ροής. Ακριβώς όπως με τις εικόνες RGB

και Βάθους, η πληροφορία Οπτικής Ροής έχει τόσο τα δικά της πλεονεκτήματα όσο και μειονεκτήματα:

Πλεονεκτήματα:

- Παρέχει πληροφορία κίνησης μικρής διάρκειας η οποία δύναται να κάνει τις προβλέψεις μας ακριβέστερες.

Μειονεκτήματα:

- Η παραγωγή της αυξάνει την χρονική πολυπλοκότητα του παρακολουθητή
- Περιέχει κυρίως πληροφορία που αφορά περισσότερο την 2-διάστατη μετατόπιση του αντικειμένου και λιγότερο τις άλλες συνιστώσες της ποζας του αντικειμένου.

Στοχεύουμε να βρούμε μια βέλτιστη αρχιτεκτονική σχεδίαση που να χρησιμοποιεί τα πλεονεκτήματα με στόχο να ενισχύσει τις δυνατότητες διάκρισης μεταξύ των ποζών τις οποίες προσφέρουν οι άλλες δυο πηγές πληροφορίας, ενώ ταυτόχρονα θέλουμε να βρούμε έναν τρόπο να αποτρέψουμε τα μειονεκτηματά της. Βασισμένοι στη δομή του συνθετικού μας συνόλου δεδομένων, πειραματιζόμαστε, σε ύστερα κεφάλαια της εργασίας αυτής, με τέσσερα ίδη πληροφορίας οπτικής ροής: Την ‘Παρατηρούμενη’ Οπτική Ροή ($OF_{Obs}(t)$), την ‘Προβλεπόμενη’ Οπτική Ροή ($OF_{Pred}(t)$), την Ροή Επαυξημένης Πραγματικότητας (AF(t)), αλλά και την αντίστροφη της: (InvAF(t)). Οι τέσσερις αυτές ποικιλίες αποκτώνται μέσω ενός πρόσθιου περάσματος από ένα παγωμένο, προεκπαιδευμένο “FlowNet2”.

- **Σύντηξη Πληροφορίας Οπτικής Εμφάνισης, Βαθους και Οπτικής Ροής**
- **Μονάδες Οπτικής Προσοχής για Χειρισμό Επικαλύψεων και Παρασκηνίου, καθοδηγούμενες από εκδόσεις Οπτικής Ροής.**
- **Τύλιξη Χάρτη Χαρακτηριστικών με τη χρήση Δικτύων Χωρικού Μετασχηματισμού**
- **Ιεραρχική Διαδοχή Χαρτών Οπτικής Προσοχής βασιζόμενων σε πληροφορία Ροής Επαυξημένης Πραγματικότητας**

Παρόλο που η χρήση της Οπτικής Ροής ως έξτρα πηγή πληροφορίας είναι μια φυσική επέκταση της μεθόδου μας από διαισθητικής απόψεως, αλλά και παρόλα τα ευεργετικά αποτελέσματα που έχει επιφέρει η χρήση της σε προηγούμενες

Σύνοψη

Στην εργασία αυτή, προτείνεται μια αρχιτεκτονική ενός ΣΝΔ για γρήγορη και ακριβή χρονική παρακολούθηση πόζας μοναδικών και γνωστών a-priori αντικειμένων. Πραγματοποιούμε μια συγκεκριμένη τμηματική σχεδίαση της διαχείρισης χαώδους παρασκηνίου και επικαλύψεων και αντιμετωπίζουμε μαθηματικά τις γεωμετρικές ιδιότητες τόσο του χώρου

ποζών όσο και του 3-διάστατου μοντέλου του αντικειμένου κατά την εκπαίδευση του ΣΝΔ. Αποτελεσματικά, μειώνουμε τα πλέον σύγχρονα λάθη πόζας κατά μια μέση διαφορά 34.03% για τις μετατόπισεις και 40.01% για τις περιστροφές και αποκτούμε μια διαισθητική κατανόηση του τεχνητού μας μηχανισμού παρακολούθησης μέσω οπτικοποίησης του μηχανισμού οπτικής προσοχής. Επιπλέον, πειραματιζόμαστε με διατάξεις επέκτασης της αρχικής αυτής ιδέας μοντελοποιώντας με συγκεκριμένες τμηματικές μονάδες τις μακρόπνοες και βραχύπνοες χρονικές δυναμικές 3-διάστατης κίνησης και καταγραφουμε τα συμπεράσματα και τις προτάσεις μας. Στο μέλλον, στοχεύουμε να επεκτείνουμε τη μέθοδο αυτή στην περίπτωση των άγνωστων αντικειμένων και να εκμαιεύσουμε την πληροφορία βάθους μόνο με τη χρήση εικόνων εμφάνισης.

Μελλοντικές Επεκτάσεις

Είναι φανερό πως για την πραγματοποίηση και υλοποίηση του αλγορίθμου που παρουσιάζεται στην εργασία αυτή έχει γίνει μια σειρά παραδοχών, καθώς ο παρακολουθητής μας βελτιώνει αισθητά μια ειδική, κατα τα άλλα, περίπτωση του γενικού πορβλήματος Οπτικής Αλληλεπίδρασης Ανθρώπου-Αντικειμένου. Έτσι, είναι δυνατόν να επεκταθεί με πιθανούς συνδυασμούς των κάτωθι κατευθύνσεων:

- Απελευθέρωση από την προτέρα γνώση του 3-διάστατου μοντέλου του αντικειμένου
- Παρακολούθηση Πόζας με βάση μόνο το RGB-μέρος της εισόδου (Διύληση Πληροφορίας Βάθους)
- Επεξεργασία των χαρακτηριστικών με ΕΟΟ(3)-ισοδύναμο τρόπο
- Γεφύρωση της αναντιστοιχίας μοντελοποίησης 2-διάστατης πηγής πληροφορίας και 3-διάστατου στόχου
- Παρακολούθηση Πολλαπλών Αντικειμένων και Καθοδήγηση Συναρμολόγησης με χρήση τεχνικών Ενισχυμένης Μάθησης
- Αυτο-Επιβλεπόμενη Χρονικής Οπτική Παρακολούθηση Πόζας Αντικειμένου με ενδιάμεση χρήση μονάδων Διαφορίσιμης Αποτύπωσης και Ανακατασκευής
- Μη-Επιβλεπόμενη Διάταξη του Χώρου Ποζών ως ενδιάμεσο στάδιο διευκόλυνσης της Νευρωνικής Παλινδρόμησης
- Οπτική Παρακολούθηση Πόζας Αρθρωτών, Παραμορφώσιμων και Πολυμερών αντικειμένων
- Μετριάσμος Εγωκεντρικής Κίνησης Κινούμενης Κάμερας ως προτέρα γνώση για Οπτική Παρακολούθηση Πόζας
- Η Παρακολούθηση της Πόζας ως προτέρα γνώση για Ρομποτική Αρπαγή

"Ever tried.
Ever failed.
No matter.
Try again.
Fail again.
Fail better."

Samuel Beckett, "Worstward Ho"

Contents

1	Εκτεταμένη Περίληψη	6
1.1	Αξιολόγηση των αλγορίθμων μας σε άλλα αντικείμενα: ο ‘Σκύλος’, το ‘Σύμπλεγμα των Lego’, το ‘Ποτιστήρι’	31
1.1.1	Μοντέλο Σκύλου	32
1.1.2	Μοντέλο ‘Lego’	33
1.1.3	Μοντέλο Ποτιστηριού	34
2	Introduction	41
2.1	Computer Vision	41
2.2	Computer Graphics	41
2.3	Machine Learning	42
2.4	Deep Learning	43
2.5	Visual Object Tracking	43
2.6	Applications of 6D Object Pose Tracking	45
2.7	Performance Requirements	48
2.8	Proposed Approach - Motivation for Deep Learning Employment	48
2.9	Challenges of the problem	48
2.10	The full Camera Projection Matrix	50
2.10.1	<i>Dissecting the Camera Projection Matrix: The Pinhole Camera Calibration Matrix/ Intrinsic Parameters</i>	50
2.10.2	<i>Dissecting the full Camera Projection Matrix: Definition of the problem of 6D Object Pose Estimation and Tracking - Pose Representation / Extrinsic Parameters</i>	51
2.11	3D Rotation Representations	55
2.12	Introduction to the approach	60
2.12.1	A discussion about the nature of the information sources	60
2.12.2	Datasets	61
2.12.3	Inspection of the dataset	63
2.13	Thesis Structure	64
3	Background	65
3.1	Mathematical Background	65
3.1.1	Matrix Trace	65
3.1.2	Frobenius norm	65
3.1.3	Gramm-Schmidt Orthonormalization	66
3.1.4	Cosine Similarity	66
3.1.5	SVD	66
3.2	Computer Vision Elements	67
3.2.1	HSV Color Space	67
3.2.2	SSIM Simiarity Metric	68
3.2.3	Optical Flow	69
3.2.4	k-Means Clustering algorithm	72

3.3	Deep Learning	73
3.3.1	Artificial Neuron Model - Rosenblatt's Perceptron	73
3.3.2	Multi-Layer Perceptron	74
3.3.3	BackPropagation Algorithm	75
3.3.4	Activation Functions	76
3.3.5	Convolutional Neural Networks	79
3.3.6	Fire modules	82
3.3.7	Batch Normalization	82
3.3.8	Pooling Layers	83
3.3.9	Dropout	84
3.3.10	Weight Initialization	84
3.3.11	Residual Connections	85
3.3.12	Dense Connections	86
3.3.13	Deconvolutional Layers	87
3.3.14	Transfer Learning	88
3.3.15	Visual Attention	89
3.3.16	Gumbel Distribution - Gumbel Softmax Trick	90
3.3.17	Loss Functions	90
3.3.18	Rotation Anisotropy Weighting using Inertial Tensor	102
3.3.19	The effect of Rotation Representations on Neural Networks	104
3.3.20	Multi-Task Loss functions	110
3.3.21	Optimization Algorithms	113
3.3.22	Scheduler schemes	118
4	Related Work	121
4.1	Single-frame Pose Detection	121
4.2	Camera Pose Estimation	135
4.3	Pose Matching/Refinement - Tracking	135
5	Data Generation	141
5.1	3D Graphics Rendering	141
5.2	Training Data Nature Tradeoff	150
5.3	Data Generation Process	151
6	Method	157
6.1	Overview	157
6.2	Preprocessing steps	157
6.3	Domain Randomization	158
6.3.1	Data Augmentation	159
6.4	Baseline Approach	165
6.5	Proposed Architecture	166
6.6	Input Normalization	166
6.6.1	Pretrained Resnet Weights - Initialization	170
6.6.2	Residual Connections	171
6.6.3	Soft Spatial Attention for Background Extraction	171
6.6.4	Soft Spatial Attention for Occlusion Handling	172
6.6.5	Linear Layers	172
6.6.6	Output Transformation	172
6.6.7	Multitask Loss function	172
6.6.8	Symmetric Object Handling	173
6.6.9	Initialization Loss	176
6.7	Implementation Details	176
6.8	Testing Metrics	177

7	Ablation Study	180
7.1	State-of-the-Art Architecture - Baseline Results	181
7.2	The necessity of the Feedback-Predicted Stream	182
7.3	Translation-Rotation range exploration	182
7.4	Insufficiencies of Unimodal architectures (only RGB/Depth-based streams)	183
7.5	Weight Initialization Algorithm comparison	185
7.6	Data Augmentation improvements	186
7.7	Frozen vs Unfrozen Transfer Learning ResNet layers	187
7.8	Activation function choice	188
7.9	Losses	189
7.10	Dense vs Residual connections	190
7.11	Stream Fusion choices	192
7.12	Clutter and Occlusion Handling Attention	195
7.12.1	Clutter Handling	195
7.12.2	Occlusion Handling	197
7.13	Hierarchical vs Parallel Clutter/Occlusion Attentions	201
7.14	Rotation Loss choices	204
8	Evaluation	207
8.1	Evaluation of modifying the baseline architecture to our proposed approach	207
8.2	Weighting the Multi-Task Loss	210
8.3	Weight warm-up alternatives	211
8.4	Object cases	212
8.4.1	The Dragon model: the asymmetric case	212
8.4.2	3D Object Movement	223
8.4.3	the Cookie Jar model: the symmetric case	234
8.5	Evaluation of our algorithms on other objects: Dog, Lego, Watering Can	279
8.5.1	Dog model	279
8.5.2	Lego model	280
8.5.3	Watering Can model	281
9	Ongoing Work	283
9.1	Unaddressed challenges	283
9.2	Continuous 3D Trajectories Learning	284
9.2.1	3D Translation continuous interpolation	285
9.2.2	3D Rotation continuous interpolation	289
9.2.3	Recurrent Units	290
9.2.4	Temporal Convolutional Networks	296
9.2.5	Recurrent variations	297
9.3	Real-time Small Drifts counter Distilled Optical Flow fusion	305
9.4	RGB-D + Optical Flow Fusion	309
9.4.1	Optical Flow guided Foreground and Occlusion handling Attention modules	311
9.4.2	Feature map Warping using Spatial Transformer Networks	315
9.4.3	Hierarchical Augmented Reality Flow Attentions	320
10	Future Work	325
10.1	Model-agnostic Single Object Tracking	325
10.2	Bridging the 2D (source) - 3D (task) modeling mismatch	325
10.2.1	(Normalized) Object Coordinate (NOCs) Regression	326
10.2.2	Umeyama Algorithm	326
10.2.3	Tensor Field Networks	328
10.3	Depth Information Distillation	329

10.4 Reinforcement Learning-based Multiple Object Tracking and Assembly Guidance	331
10.5 Self-Supervised Object Pose 6D Tracking	333
10.6 Pose Tracking of Deformable, Articulated and Compositional Object	335
10.7 Ego-Motion Mitigation of a Moving Camera Sensor as a Pose Tracking prior	336
10.8 Pose Tracking as a prior for Robotic Grasping	337

11 Conclusion 338

List of Figures

1.1	Οπτική περιγραφή της βασικής αρχιτεκτονικής ΣΝΔ των Garon et al.[83] για την οπτική παρακολούθηση της πόζας μοναδικών, γνωστών αντικειμενων στο χρόνο.	8
1.2	Οπτική περιγραφή της προτεινόμενης αρχιτεκτονικής ΣΝΔ για την οπτική παρακολούθηση της πόζας μοναδικών, γνωστών αντικειμενων στο χρόνο.	8
1.3	Ένας Χάρτης Βαρών Χωρικής Προσοχής αφιερωμένος στην Εξαγωγή Προσκηνίου. Τα τονισμένα εικονοστοιχεία είναι εκείνα που αφορούν τα κινούμενα μέρη της σκηνής.	10
1.4	Ένας Χάρτης Βαρών Χωρικής Προσοχής αφιερωμένος στο Χειρισμό Επικαλύψεων. Τα τονισμένα εικονοστοιχεία είναι εκείνα που αντιστοιχούν μόνο στο αντικείμενο ενδιαφέροντος.	10
1.5	Η Περιστροφική μας Εκτίμηση για το 126ο πλαίσιο για το σενάριο ‘Δύσκολης Αλληλεπίδραση’.	14
1.6	Η Περιστροφικής μας Εκτίμησης για το 127ο πλαίσιο του σεναρίου ‘Δύσκολης Αλληλεπίδρασης’ χωρίς τον προτεινόμενο αλγόριθμο	14
1.7	Η Περιστροφικής μας Εκτίμηση για το 126ο πλαίσιο του σεναρίου ‘Δύσκολης Αλληλεπίδρασης’.	14
1.8	Η Περιστροφικής μας Εκτίμηση για το 127ο πλαίσιο του σεναρίου ‘Δύσκολης Αλληλεπίδρασης’ με χρήση του προτεινόμενου αλγόριθμου	14
1.9	Μια οπτική σύνοψη της μεθόδου δημιουργίας συνθετικών δεδομένων της εργασίας [29].	15
1.10	(α) Τυχαία δειγματοληψία 20.000 σημείων. Η μέγιστη απόσταση μεταξύ των δύο κοντινότερων γειτόνων είναι: 0.256532 m.	16
1.11	(β) Τυχαία Δειγματοληψία 200.000 σημείων και συσταδοποίησης με χρήση του Αλγορίθμου K-Μέσων σε 20.000 σημεία-κεντροειδή. Η μέγιστη απόσταση μεταξύ 2 κοντινότερων γειτόνων είναι: 0.222318 m.	16
1.12	(γ) Ντετερμινιστική δειγματοληψία 20.000 σημείων. Η μέγιστη απόσταση μεταξύ των 2 κοντινότερων γειτόνων είναι: 0.295056 m.	17
1.13	(δ) Ντετερμινιστική δειγματοληψία 20.000 σημείων και συσταδοποίηση με τον Αλγόριθμο K-Μέσων στα 20.000 σημεία-κεεντροειδή. Η μέγιστη απόσταση μεταξύ των 2 κοντινότερων γειτόνων είναι: 0.214753 m.	17
1.14	Σύγκριση της μέγιστης δυνατής Ευκλιδειας απόστασης κοντινότερων γειτόνων από τυχαία/ντετερμινιστική δειγματοληψία 3Δ σημείων σε μια σφαίρα χωρίς/με χρήση ενός ενδιάμεσου σημείου συσταδοποίησης KMέσων ενός μεγαλύτερου υπερσυνόλου.	17
1.15	Μια 3-διάστατη Γκαουσιανή Κατανομή την οποία χρησιμοποιούμε για να μοντελοποιήσουμε τον θόρυβο του Kinect, η οποία αποτελεί γινόμενο δύο Παράλληλων και μιας Αξονικής συνιστώσας θορύβου ως προς τον κύριο άξονα της κάμερας[91].	18
1.16	Ένα ενδεικτικό τυχαίο Παρατηρούμενο ζεύγος RGB-D ζεύγος εικόνων στο οποίο επιδεικνύουμε τυχόντες συνδυασμούς της συνολικής μας στρατηγικής.	19
1.17	Πλήρη τυχαία παραδείγματα επαύξησης της ίδιας Παρατηρούμενης RGB (αριστερά) και -D (δεξιά) εικόνας.	19
1.18	Αποτύπωση μετρικών 3-διάστατου Μεταφορικού και Περιστροφικού λάθους για το σενάριο ‘Κοντινής Σταθερότητας’ του 3-διάστατου μοντέλου του ‘Δράκου’, στην περίπτωση που ο παρακολουθητής επαναρχικοποιείται κάθε 15 εικόνες.	23
1.19	Αποτύπωση μετρικών 3-διάστατου Μεταφορικού και Περιστροφικού λάθους για το σενάριο ‘Κοντινής Σταθερότητας’ του 3-διάστατου μοντέλου του ‘Δράκου’, στην περίπτωση που ο παρακολουθητής επαναρχικοποιείται κάθε φορά που αποτυγχάνει.	23

1.20	Αποτύπωση μετρικών 3-διάστατου Μεταφορικού και Περιστροφικού λάθους για το σενάριο ‘Μακρινής Σταθερότητας’ του 3-διάστατου μοντέλου του ‘Δράκου’, στην περίπτωση που ο παρακολουθητής επαναρχικοποιείται κάθε 15 εικόνες.	24
1.21	Αποτύπωση μετρικών 3-διάστατου Μεταφορικού και Περιστροφικού λάθους για το σενάριο ‘Μακρινής Σταθερότητας’ του 3-διάστατου μοντέλου του ‘Δράκου’, στην περίπτωση που ο παρακολουθητής επαναρχικοποιείται κάθε φορά που αποτυγχάνει.	24
1.22	Αποτύπωση μετρικών 3-διάστατου Μεταφορικού και Περιστροφικού λάθους για το σενάριο ‘75% Κάθετης Επικάλυψης’ του 3-διάστατου μοντέλου του ‘Δράκου’, στην περίπτωση που ο παρακολουθητής επαναρχικοποιείται κάθε 15 εικόνες.	24
1.23	Αποτύπωση μετρικών 3-διάστατου Μεταφορικού και Περιστροφικού λάθους για το σενάριο ‘75% Κάθετης Επικάλυψης’ του 3-διάστατου μοντέλου του ‘Δράκου’, στην περίπτωση που ο παρακολουθητής επαναρχικοποιείται κάθε φορά που αποτυγχάνει.	24
1.24	Αποτύπωση μετρικών 3-διάστατου Μεταφορικού και Περιστροφικού λάθους για το σενάριο ‘75% Οριζόντιας Επικάλυψης’ του 3-διάστατου μοντέλου του ‘Δράκου’, στην περίπτωση που ο παρακολουθητής επαναρχικοποιείται κάθε 15 εικόνες.	24
1.25	Αποτύπωση μετρικών 3-διάστατου Μεταφορικού και Περιστροφικού λάθους για το σενάριο ‘75% Οριζόντιας Επικάλυψης’ του 3-διάστατου μοντέλου του ‘Δράκου’, στην περίπτωση που ο παρακολουθητής επαναρχικοποιείται κάθε φορά που αποτυγχάνει.	24
1.26	Αποτύπωση μετρικών 3-διάστατου Μεταφορικού και Περιστροφικού λάθους για το σενάριο ‘Αποκλειστικά Μετατόπισης’ του 3-διάστατου μοντέλου του ‘Δράκου’, στην περίπτωση που ο παρακολουθητής επαναρχικοποιείται κάθε 15 εικόνες.	25
1.27	Αποτύπωση μετρικών 3-διάστατου Μεταφορικού και Περιστροφικού λάθους για το σενάριο ‘Αποκλειστικά Μετατόπισης’ του 3-διάστατου μοντέλου του ‘Δράκου’, στην περίπτωση που ο παρακολουθητής επαναρχικοποιείται κάθε φορά που αποτυγχάνει.	25
1.28	Αποτύπωση μετρικών 3-διάστατου Μεταφορικού και Περιστροφικού λάθους για το σενάριο ‘Αποκλειστικά Περιστροφών’ του 3-διάστατου μοντέλου του ‘Δράκου’, στην περίπτωση που ο παρακολουθητής επαναρχικοποιείται κάθε 15 εικόνες.	25
1.29	Αποτύπωση μετρικών 3-διάστατου Μεταφορικού και Περιστροφικού λάθους για το σενάριο ‘Αποκλειστικά Περιστροφών’ του 3-διάστατου μοντέλου του ‘Δράκου’, στην περίπτωση που ο παρακολουθητής επαναρχικοποιείται κάθε φορά που αποτυγχάνει.	25
1.30	Αποτύπωση μετρικών 3-διάστατου Μεταφορικού και Περιστροφικού λάθους για το σενάριο ‘Πλήρους Αλληλεπίδρασης’ του 3-διάστατου μοντέλου του ‘Δράκου’, στην περίπτωση που ο παρακολουθητής επαναρχικοποιείται κάθε 15 εικόνες.	25
1.31	Αποτύπωση μετρικών 3-διάστατου Μεταφορικού και Περιστροφικού λάθους για το σενάριο ‘Πλήρους Αλληλεπίδρασης’ του 3-διάστατου μοντέλου του ‘Δράκου’, στην περίπτωση που ο παρακολουθητής επαναρχικοποιείται κάθε φορά που αποτυγχάνει.	25
1.32	Αποτύπωση μετρικών 3-διάστατου Μεταφορικού και Περιστροφικού λάθους για το σενάριο ‘Δύσκολης Αλληλεπίδρασης’ του 3-διάστατου μοντέλου του ‘Δράκου’, στην περίπτωση που ο παρακολουθητής επαναρχικοποιείται κάθε 15 εικόνες.	26
1.33	Αποτύπωση μετρικών 3-διάστατου Μεταφορικού και Περιστροφικού λάθους για το σενάριο ‘Δύσκολης Αλληλεπίδρασης’ του 3-διάστατου μοντέλου του ‘Δράκου’, στην περίπτωση που ο παρακολουθητής επαναρχικοποιείται κάθε φορά που ο παρακολουθητής αποτυγχάνει. . .	26
1.34	Οι 6-Δ Τροχιές Πόζας του μοντέλου του ‘Δράκου’, στο χρόνο, για το σενάριο των ‘Δύσκολων Αλληλεπιδράσεων’. Είναι προφανές, και στις 2 περιπτώσεις, και από την σκοπιά αυτή, πως η προτεινόμενη προσέγγισή μας, αυτή ξεπερνά σε απόδοση εκείνη των Garon et al. [83] κατά πολύ, μιας και παράγει μικρότερα λάθη και λιγότερες αποτυχίες. Όλες οι μονάδες μέτρησις μήκους είναι σε χιλ. και οι περιστροφές σε μοίρες. Οι μαύρη τροχιά είναι η πραγματική, η μπλε τροχιά είναι η βασική τροχιά σύγκρισης της μεθόδου των Garon et al. [83] και η κόκκινη είναι η δική μας.	26
1.35	Αποτύπωση Μετρικών 3-διάστατου Λάθους Μετατόπισης και Περιστροφής, για το σενάριο ‘Κοντινής Σταθερότητας’ του 3-διάστατου μοντέλου του ‘Κουτιού με τα Μπισκότα’, στην περίπτωση που ο παρακολουθητής επαναρχικοποιείται κάθε 15 εικόνες.	27

1.51	Οι 6-Δ Τροχιές Πόζας του μοντέλου του ‘Κουτιού με τα Μπισκότα’, στο χρόνο, για το σενάριο των ‘Δύσκολων Αλληλεπιδράσεων’. Είναι προφανές, και στις 2 περιπτώσεις, και από την σκοπιά αυτή, πως η προτεινόμενη προσέγγισή μας, αυτή ξεπερνά σε απόδοση εκείνη των Garon et al. [83] κατά πολύ, μιας και παράγει μικρότερα λάθη και λιγότερες αποτυχίες. Όλες οι μονάδες μέτρησις μήκους είναι σε χιλ. και οι περιστροφές σε μοίρες. Οι μαύρη τροχιά είναι η πραγματική, η μπλε τροχιά είναι η βασική τροχιά σύγκρισης της μεθόδου των Garon et al.[83], η κόκκινη είναι η δική μας και, προαιρετικά, η πορτοκαλί είναι για τα αντικείμενα με συνεχώς περιστροφική συμμετρία, όταν αυτή λαμβάνεται υπόψη στη σχεδιάσή μας.	31
1.52	Σύγκριση μεταξύ της εως τώρα βέλτιστης μεθόδου, των Garon et al.[83] (γαλάζιο) και της δικής μας (ροζ) για το ‘Σκύλο’ σε 3 σενάρια: ‘75% Κάθετη Επικάλυψη’, ‘Μόνον Περιστροφές’ και ‘Δύσκολη Αλληλεπίδραση’.	32
1.53	Οι 6-Δ Τροχιές Πόζας του μοντέλου του ‘Σκύλου’, στο χρόνο, για το σενάριο των ‘Δύσκολων Αλληλεπιδράσεων’. Είναι προφανές, και στις 2 περιπτώσεις, και από την σκοπιά αυτή, πως η προτεινόμενη προσέγγισή μας, αυτή ξεπερνά σε απόδοση εκείνη των Garon et al. [83] κατά πολύ, μιας και παράγει μικρότερα λάθη και λιγότερες αποτυχίες. Όλες οι μονάδες μέτρησις μήκους είναι σε χιλ. και οι περιστροφές σε μοίρες. Οι μαύρη τροχιά είναι η πραγματική, η μπλε τροχιά είναι η βασική τροχιά σύγκρισης της μεθόδου των Garon et al.[83], και η κόκκινη είναι η δική μας.	32
1.54	Σύγκριση μεταξύ της εως τώρα βέλτιστης μεθόδου, των Garon et al.[83] (γαλάζιο) και της δικής μας (ροζ) για το ‘Σύμπλεγμα από Lego’ σε 3 σενάρια: ‘Μόνον Μετατοπίσεις’, ‘Πλήρης’ και ‘Δύσκολη Αλληλεπίδραση’.	33
1.55	Οι 6-Δ Τροχιές Πόζας του μοντέλου του ‘Block από Legos’, στο χρόνο, για το σενάριο των ‘Δύσκολων Αλληλεπιδράσεων’. Είναι προφανές, και στις 2 περιπτώσεις, και από την σκοπιά αυτή, πως η προτεινόμενη προσέγγισή μας, αυτή ξεπερνά σε απόδοση εκείνη των Garon et al. [83] κατά πολύ, μιας και παράγει μικρότερα λάθη και λιγότερες αποτυχίες. Όλες οι μονάδες μέτρησις μήκους είναι σε χιλ. και οι περιστροφές σε μοίρες. Οι μαύρη τροχιά είναι η πραγματική, η μπλε τροχιά είναι η βασική τροχιά σύγκρισης της μεθόδου των Garon et al.[83] και η κόκκινη είναι η δική μας.	33
1.56	Σύγκριση μεταξύ της εως τώρα βέλτιστης μεθόδου, των Garon et al.[83] (γαλάζιο) και της δικής μας (ροζ) για το ‘Ποτιστήρι’ σε 3 σενάρια: ‘75% Οριζόντια Επικάλυψη’, ‘Μόνον Περιστροφές’ και ‘Πλήρης Αλληλεπίδραση’.	34
1.57	Οι 6-Δ Τροχιές Πόζας του μοντέλου του ‘Ποτιστηριού’, στο χρόνο, για το σενάριο των ‘Δύσκολων Αλληλεπιδράσεων’. Είναι προφανές, και στις 2 περιπτώσεις, και από την σκοπιά αυτή, πως η προτεινόμενη προσέγγισή μας, αυτή ξεπερνά σε απόδοση εκείνη των Garon et al. [83] κατά πολύ, μιας και παράγει μικρότερα λάθη και λιγότερες αποτυχίες. Όλες οι μονάδες μέτρησις μήκους είναι σε χιλ. και οι περιστροφές σε μοίρες. Οι μαύρη τροχιά είναι η πραγματική, η μπλε τροχιά είναι η βασική τροχιά σύγκρισης της μεθόδου των Garon et al.[83], η κόκκινη είναι η δική μας και, προαιρετικά, η πορτοκαλί είναι για τα αντικείμενα με συνεχώς περιστροφική συμμετρία, όταν αυτή λαμβάνεται υπόψη στη σχεδιάσή μας.	34
1.58	Παραδείγματα 3-διάστατης συνεχούς τροχιάς θέσης. Η μπλε γραμμή παραβάλλει γραμμικά μεταξύ τεσσάρων διακριτών σημείων (σε μέτρα) και η κόκκινη γραμμή αναπαριστά τις λείες τροχιές που περνούν και από τα 4 αυτά σημεία, χρησιμοποιώντας κυβικές σπλίνες.	36
1.59	Μια γραφική αναπαράσταση της περιστροφικής γεωδαισικής 3-διάστατης τροχιάς που ακολουθεί ένα αντικείμενο για να μετασχηματίσει την πόζα του από μία αρχική κατάσταση (πάνω αριστερή γωνία) σε μια αντίστοιχη τελική κατάσταση (κάτω δεξιά).[168]	38
1.60	Η ‘Πρατηρούμενη’ συνθετικής πυκνή εικόνα Οπτικής Ροής παραγόμενη περνώντας δύο συνεχόμενες ‘Παρατηρούμενες’ RGB εικόνες, τις οποίες γεννούμε από μια απο τις συνεχού τροχιές της προηγούμενου κεφαλαίου, μες από ένα παγωμένο και προεκπαιδευμένο FlowNet2.	38
1.61	Η ‘Προβλεπόμενη’ συνθετικής πυκνή εικόνα Οπτικής Ροής παραγόμενη περνώντας δύο συνεχόμενες ‘Προβλεπόμενες’ RGB εικόνες, τις οποίες γεννούμε από μια απο τις συνεχού τροχιές της προηγούμενου κεφαλαίου, μες από ένα παγωμένο και προεκπαιδευμένο FlowNet2.	38

1.62	Η συνθετική πυκνή εικόνα Οπτικής Ροής Επαυξημένης Πραγματικότητας (EAP), παραγόμενη από το πέρασμα μιας ‘Παρατηρούμενης’ και μιας ‘ Προβλεπόμενης’ RGB εικόνας, διαμέσου ενός προεκπαιδευμένου και παγωμένου FlowNet2.	38
2.1	The result of a famous 2D Object Tracking Algorithm called “GOTURN” in the soccer example.[41]	43
2.2	The second and most recent version of the Microsoft Kinect RGB-D sensor.	44
2.3	Per-frame Object 6D Pose Detection based on Learning to Regress Object Coordinates. [8]	44
2.4	Real-time Model-based Tracking up to 100 6D Object Poses at the same time with a model-based Energy minimization framework leveraging RGB-D image and Optical Flow information. [100]	45
2.5	Real-time Model-based Single Object 6D Pose Tracking using an online Energy minimization scheme, only with RGB image information source. [134]	45
2.6	An interactive Object Assembly framework, occasioned by the Baby-Robot project realized by the CVSP Lab Team, that aids children with special abilities to assemble constructions made of Lego blocks.[34]	45
2.7	Microsoft Hololens	46
2.8	A training Ping-Pong table, launched by a German startup company that interactively analyzes player’s hit history and suggests the next move for each of them.	46
2.9	A mobile application recently launched by IKEA. The user’s smartphone is headed towards objects present in the scene, to capture their full pose. Then, pieces of IKEA furniture are rendered on these objects, at the exact same pose, to provide a preview of the customer’s potential market options.	47
2.10	A Robotic manipulation example.	47
2.11	The augmented visual environment of a contemporary Autonomous Driven car tested today.	47
2.12	The full image registration diagram. The object coordinates are first transformed by the application of the Pose homogenous matrix from the Object to the Camera coordinate frame (consisted of a sequence of transforms $(O) \rightarrow (W)$, $(W) \rightarrow (C)$) and, sequentially, each 3D point w.r.t. (C) is projected using the properties of the Camera parameter matrix K to the image plane filling the suitable pixel values.	50
2.13	A Riemannian Manifold. [86]	52
2.14	A Riemannian Exponential Map [23].	52
2.15	The Earth is a famous case of a manifold in \mathbb{R}^3 , where we can define a “1-1” relationship between its curved structure and a map plane (defined by the map scale), but only locally.[86]	52
2.16	The Table1 from [11] that classifies rotation symmetry categories.	55
2.17	Visualization of the transformations between all standard ways of 3D rotation representation.	55
2.18	The Roll-Yaw-Pitch angle representation w.r.t the center mass of a rigid object.[156]	56
2.19	Normal situation: 3 independent Gimbals.[156]	57
2.20	Gimbal lock: 2 out of 3 Gimbals are aligned - 1 DOF lost.[156]	57
2.21	The Axis-Angle 3D rotation convention.[154]	57
2.22	The 3D Unit Quaternia rotation space.[159]	59
2.23	A Typical RGB frame for the synthetic dataset of Choi and Christensen [17].	61
2.24	The dataset introduced in the first work of Garon et al. [29]. Objects are static on a 2D turntable and their pose labels are obtained with the use of fiducial markers. This dataset includes occlusion scenarios, where the object is covered horizontally or vertically by a white plane by a percentage starting from 10% and gradually increasing to 40%.	61
2.25	The testing scenarios presented in [128]. This test dataset contains a wide variety of tracking challenges as it starts from simple single object cases and it slowly escalates to adding clutter by other objects, dynamic occlusions by the user’s hands, multiple instances of the same objects and lighting variation.	62
2.26	A scene of the LINEMOD [112] pose detection dataset.	62

2.27	A scene of the LINEMOD [112] pose detection dataset, under the presence of severe occlusions between the various objects in the scene.	62
2.28	Sample RGB images from sequences belonging to the domain adaptation benchmark of [55].	62
2.29	The dataset presented by Garon et al. in [83] is the one we work on in this thesis. Here, we present the collection of 3D object models contained in it.	63
2.30	Examples of the “Free hard interaction” scenario of the dataset of [83] on 2 separate objects: the clock and the dragon ones.	64
3.1	Graphical description of the Gram-Schmidt process for \mathbf{v}_2 . It is comprised by one component parallel to \mathbf{e}_1 (that we subtract), and another one perpendicular to it (that we keep normalized).	66
3.2	A Graphical representation of the SVD decomposition of a given matrix M . [160]	67
3.3	The RGB and HSV color spaces. [157]	67
3.4	Dense Optical Flow examples given by the FlowNet2 architecture in the synthetic ‘Sintel’ video dataset. [52]	69
3.5	The displacement of every pixel in the top left (multi colored) image is the vector from the center of the square to that particular pixel, as indicated by the image on the right. This means at the middle of the image i.e. the white portion, indicates no optical flow, in the blue quadrant, it indicates flow to the left and to the top, the greater the shade of blue the greater the magnitude of the the vector. This applies to the other quadrants as well where the more intense the color the greater the magnitude of flow to that point. [52]	70
3.6	An overview of the FlowNet Simple (FlowNetS) (top) and the FlowNet Correlation (FlowNetC) (bottom) architectural components of FlowNet2. [30]	70
3.7	Explaining the function of the refining submodule. [30]	71
3.8	Overview of the overall FlowNet2.0 architecture [52].	71
3.9	A k-Means clustering example. [158]	72
3.10	The biological neuron model that inspired McCulloch and Pitts Cookie Jar’s Artificial Neuron. [85]	73
3.11	The Artificial Neuron model. [114]	74
3.12	A Deep Neural Network (or else Multi Layer Perceptron) with an input, an output and 3 hidden layers [19].	74
3.13	A graphical representation of the Backpropagation error pass in a Neural architecture. [66]	75
3.14	An abbreviated visualization of the Gradient Clipping algorithm [32].	76
3.15	A pivot table of Activation function choices, their diagram and derivatives [153].	78
3.16	The Swish activation function [107].	78
3.17	A single Convolutional filter activation map as it is colvolved with an input image [67] .	79
3.18	The “Waldo” pattern-template.	80
3.19	A “Find Waldo” terrain, from a popular game for children. A “Waldo detection” hypothetical CNN would process each location, in its filter size dimension, in the same manner and would locate the “Waldo” pattern-termplate wherever it may be in the picture. Due to its translational equivariance, it would locate it in its exact spot (instead of, for example, appointing it to the center of the image no matter its initial position, something that a “translational invariant” approach would have done, instead).	80
3.20	In case of a Fully Connected layer applied to an image, even for a small resolution of 200×200 , matching a pixel to every neuron results in an architecture with billions of parameters. [66]	81
3.21	For this reason, we substitute Fully Connected layers with Local ones that cover a specific region of the image. [66]	81
3.22	One step further, instead for interconnected pixel-neuron connections we just use a kernel of few learnable weights. [66]	81
3.23	Finally, different kernels per region are dropped out and the same filters transit on the image to transfuse translational equivariance. [66]	81

3.24	The evolution of Neural Networks from just a final component that processes handcrafted features, that are constructed in an unsupervised, unified and problem-agnostic way, to a full end-to-end data driven feature extraction and decision making tool that hierarchically extracts progressively complex features as combinations of the previous ones.[66]	81
3.25	The Fire module of [51]. At first, the input feature map is transformed to a lower dimensional one (via a 1×1 Convolution) to reduce the required parameters followed by an expansion layer that processes the squeezed Feature map with a stack of 1×1 and 3×3 Convolutions.	82
3.26	A sequence of applying a convolutional and a pooling layer to the feature maps of the previous layer.[66]	83
3.27	A graphical example of the application of a Max Pooling filter to a depth slice spatial neighbourhood.	84
3.28	Neural Network neuron Dropout, with a probability p , during model training. [121]	84
3.29	Landscape comparison between Residual and Dense connecting.[72]	85
3.30	A graphical representation of a residual block of the “ResNet” architectural design.[37]	86
3.31	A single layer Residual connection that adds an Identity mapping to the layer’s Feature map.[37]	86
3.32	An overview of a “DenseNet” architectural module.[49]	86
3.33	An overview of the “DenseNet” architecture. [49]	87
3.34	[72]	87
3.35	A Deconvolutional (or Transpose Convolutional) layer operation on a Feature map channel.[120]	87
3.36	Initializing and freezing of the two first layers of the lower CNN architecture with those of the CNN architecture, above, trained on dataset A to execute task A. Then, the rest of CNN B’s layer weights are trained on dataset B to execute task B, which is our task at hand[95]	88
3.37	Visual imprint of the Spatial Attentions module’s effect on images, in another task: Visual Question answering. The model enhances its distinctive abilities by softly focusing on the image attributes that are of interest of the particular question.	89
3.38	Soft vs Hard Visual Attention tradeoff.[165]	90
3.39	A convex and a non convex Loss function. While the former has a Global minimum that we can find, the latter may have many local ones.[1]	91
3.40	The Global and Local minima of a multivariate Loss function.	91
3.41	[1]	91
3.42	[1]	92
3.43	MSE vs MAE Loss.[1]	92
3.44	Graphical representation of the LogCosh regression loss function. It balances the tradeoff between MSE and MAE.	93
3.45	The curve of Barron Loss for different choices or parameters α, c that simulate a family of well-known regression Loss functions and its derivatives. [4]	93
3.46	A portrait of William of Ockham (c.1287–1347).	95
3.47		96
3.48	Three dimensional translations consist an element of the Euclidean space.	96
3.49	Visualization of the Chordal Geodesic Rotation Distance between two rotations R_1, R_2 [86]	97
3.50	Visualization of the Riemannian Geodesic Rotation Distance between two rotations R_1, R_2 . [86]	97
3.51	Visualization of different paths on a Manifold-like terrain. The left is not a Geodesic, but the right one is.	99
3.52	The 2D map-projection of the airplane’s 3D trajectories (both the Euclidean and the Geodesic ones).	102
3.53	The Geodesic (shortest) distance path, on the globe, which is created by the intersection of the globe with the greatest plane passing from the 2 points of the globe and at a direction that follows the globe’s parallels.	102

3.54	Usual metrics would consider the distances between the two poses in cases (a) and in (b) equal – as in both cases the two poses are linked by a rotation of 15 deg around the center of mass of the object. Our distance will account for the object geometry and discriminates between these two configurations.[11]	103
3.55	A graphical interpretation of Continuous Rotation Representations in the NN context, as presented in [171].	104
3.56	A simple 2D example which motivates the definition of [171] for Continuous Representation for 3D Roations.	105
3.57	A visual comparison of the various rotation representations. The three columns showcase three different curves $\in \text{SO}(3)$:the “X,Y,Z” rotations of the correponding axes. Each curve is mapped $\in \text{SO}(3)$ to each of the rotation representations in different rows and then then it is projected in 2D via PCA. The hue is used to visualize the rotation angle $\in \text{SO}(3)$ around each of the three canonical axes “X,Y,Z”. We understand that a representation is continuous if that projection is homeomorphic to a circle and has the same direction of color change.[171]	110
3.58	A Gradient-Descent-based Loss function optimization scheme.	113
3.59	(a) SGD without momentum. (b) SGD with momentum [116]	115
3.60	The Learning Rate magnitude tradeoff: when the learning rate is too high the optimization process with heavily pertrubate and diverge from the function minimum. On the opposite casem where it is too low, the training process will be slow and the optimizer scheme will likely be trapped into local minima that it will not have the power to escape from.	118
3.61	The learning rate scheduling policy of reducing its value by an order of magnitude after a number “patience” epochs where the validation loss does not decrease anymore. In that scenario, a single model is kept (that with the lowest validation loss) and at the latter training stages the algorithm heads deeper and deeper to a set (local) minimum with ever-decreasing step.	119
3.62	Left: Illustration of SGD optimization with a typical learning rate scheduling scheme that decreases the gradient descent step when the loss does not decrease anymore. Right: The model undergoes several learning rate annealing cycles, escaping this way from a sequence of local minima.[79]	119
3.63	Cosine Annealing learning rate schedule with “Warm Restarts” with train cycles increasing each time.[79]	120
4.1	An example of the Hintetoisser approach of sliding a 2D-window in an image and compare the convolutional features produced by it with the convolutional features that correspond to a template [112].	122
4.2	Illustration of the voting scheme of [56]. They densely sample the scene to extract scale-invariant RGB-D patches. These are fed into a network to regress features for a subsequent kNN search in a codebook of pre-computed synthetic local object patches. The retrieved neighbors then cast 6D votes if their feature distance is smaller than a threshold τ . [56]	123
4.3	PVNet: Overview of the keypoint localization: (a) An image of the Occlusion LINEMOD dataset. (b) The architecture of PVNet. (c) Pixel-wise unit vectors pointing to the object keypoints. (d) Semantic labels. (e) Hypotheses of the keypoint locations generated by voting. The hypotheses with higher voting scores are brighter. (f) Probability distributions of the keypoint locations estimated from hypotheses. [102]	123
4.4	PVNet3D:The Feature Extraction module extracts the per-point feature from an RGBD image. They are fed into the modules: M_K , M_C and M_S to predict the translation offsets to keypoints, center point and semantic labels of each point, respectively. A clustering algorithm is then applied to distinguish different instances with the same semantic label and points on the same instance vote fortheir target keypoints. Finally, a least-square fitting algorithm is applied to the predicted keypoints to estimate 6DoF pose parameters.[40]	123
4.5	The P’nP Pose Estimation algorithm [71]	125

4.6	<p>Top left: RGB-D Test image (upper-half depth image and lower-half RGB image). The estimated 6D pose of the query object (camera) is illustrated with a blue bounding box, and the respective ground truth with a green bounding box. Top right: Visualization of the algorithms search for the optimal pose, where the inset is a zoom of the centre area. The algorithm optimizes the energy function in a RANSAC-like fashion over a large, continuous 6D pose space. The 6D poses, projected to the image plane, which are visited by the algorithm are color coded: red poses are disregarded in a very fast geometry check; blue poses are evaluated using our energy function during intermediate, fast sampling; green poses are subject to the most expensive energy refinement step. Bottom, from left to right: (a) Probability map for the query object, (b) predicted 3D object coordinates from a single tree mapped to the RGB cube, (c) corresponding ground truth 3D object coordinates, (d) overlay of the 3D model in blue onto the test image (rendered according to the estimated pose) [8].</p>	126
4.7	<p>An overview of the approach proposed by Krull et al. (it is similar to the one previously proposed by Brachman et al.). A feature extractor (either a CNN or a Random Forest structure) is used to map RGB-D pairs to 3D Object Coordinates, while keeping their correspondences in the 2D image plane. These correspondences are then fed to P'n'P module with RANSAC hypothesis evaluation [69].</p>	126
4.8	<p>Three-dimensional descriptors of [161] for several objects under many different views computed by this method on RGB-D data. Top-left: The training views of different objects are mapped to well-separated descriptors, and the views of the same object are mapped to descriptors that capture the geometry of the corresponding poses, even in this low dimensional space. Top-right: New images are mapped to locations corresponding to the object and 3D poses, even in the presence of clutter. Bottom: Test RGB-D views and the RGB-D data corresponding to the closest template descriptor. [161]</p>	127
4.9	<p>Given an input depth image patch x_i, we create corresponding triplets (x_i, x_j, x_k) and pairs (x_i, x_j) to optimize our model on both manifold embedding, creating robust feature descriptors, and pose regression. Obtaining either a direct pose estimate or using the resulting feature descriptor for nearest neighbor search in the descriptor database. [12]</p>	128
4.10	<p>(left) The overview of the method of Balntas et al. A triplet network learns embeddings suitable for both object recognition and exact pose retrieval (right) Illustration of the proposed pose loss. Intuitively, the method aims to train a CNN in such a way that the distance in the resulting D-dimensional embedding space is analogous to the pose differences. [3]</p>	128
4.11	<p>Architecture of PoseCNN for 6D object pose estimation [162]</p>	129
4.12	<p>Schematic overview of the SSD-style network prediction. We feed our network with a 299×299 RGB image and produce six feature maps at different scales from the input image using branches from InceptionV4. Each map is then convolved with trained prediction kernels of shape $(4 + C + V + R)$ to determine object class, 2D bounding box as well as scores for possible viewpoints and in-plane rotations that are parsed to build 6D pose hypotheses. Thereby, C denotes the number of object classes, V the number of viewpoints and R the number of in-plane rotation classes. The other 4 values are utilized to refine the corners of the discrete bounding boxes to tightly fit the detected object. [58]</p>	129
4.13	<p>Overview of YOLO6D [130]: (a) The proposed CNN architecture. (b) An example input image with four objects. (c) The $S \times S$ grid showing cells responsible for detecting the four objects. (d) Each cell predicts 2D locations of the corners of the projected 3D bounding boxes in the image. (e) The 3D output tensor from our network, which represents for each cell a vector consisting of the 2D corner locations, the class probabilities and a confidence value associated with the prediction. [130]</p>	130
4.14	<p>Training process for the architecture of [126]; a) reconstruction target batch of uniformly sampled $SO(3)$ object views; b) geometric and color augmented input; c) reconstruction after 30000 iterations. [126]</p>	130

4.15	Top: creating a codebook from the encodings of discrete synthetic object views; Bottom: object detection and 3D orientation estimation using the nearest neighbor(s) with highest cosine similarity from the codebook of [126]	131
4.16	Illustration of our modular, 3-stage pipeline for both RGB and RGB-D input images of the architecture of [48]	131
4.17	Pipeline description of DPOD [167]: Given an input RGB image, the correspondence block, featuring an encoder-decoder neural network, regresses the object ID mask and the correspondence map. The latter one provides us with explicit 2D-3D correspondences, whereas the ID mask estimates which correspondences should be taken for each detected object. The respective 6D poses are then efficiently computed by the pose block based on PnP+RANSAC[[71],[25]].[167]	132
4.18	Refinement module of DPOD [167]: The network predicts a refined pose given an initial pose proposal. Crops of the real image and the rendering are fed into two parallel branches. The difference of the computed feature tensors is used to estimate the refined pose.[167]	132
4.19	An overview of the architecture of Pix2Pose object coordinate regression pipeline.[98] .	133
4.20	Overview of the stacked hourglass architecture for predicting the semantic keypoints. Here, two hourglass modules are stacked together. The symmetric nature of the design allows for bottom-up processing (from high to low resolution) in the first half of the module, and top-down processing (from low to high resolution) in the second half. Intermediate supervision is applied after the first module. The heatmap responses of the second module represent the final output of the network that is used for keypoint localization [101]	133
4.21	Pipeline of the approach of Pavlakos et al. Given a single RGB image of an object (a), they localized a set of class-specific keypoints using a CNN with the stacked hourglass design of fig4.20. The output of this step is a set of heatmaps for each keypoint, which are combined for visualization in (b), sometimes leading to false detections. In (c), green dots represent the detected keypoints and the corresponding blue dots (connected with an arrow) the groundtruth locations. For robustness against such localization errors, they solved a fitting problem to enforce global consistency of the keypoints, where the response of the heatmaps is used as a measure of certainty for each keypoint. The optimization recovers the full 6-DoF pose of the object present in the image (d).[101]	133
4.22	System overview. The color and depth images together with a segmentation mask of the target object are used to create a point cloud. The segment is randomly downsampled, and the estimated translation of the down-sampled segment is removed. The normalized segment is fed into a network for rotation prediction, using an Axis-Angle representation.[28]	134
4.23	Overview of the PointNet architecture.[104]	134
4.24	Overview of the proposed pipeline. (a) Given an RGB image, we use CNN to detect the bounding box of the target object and the object label which is used as one-hot feature for PointPoseNet. (b) Given the point clouds in the target region, we use proposed PointPoseNet to do 3D segmentation and vector prediction. (c) Top: 3D mask for target object; bottom: Point-wise unit vectors pointing to the keypoint. (d) 3D keypoints hypotheses generated from the unit vectors. (e) Final pose after hypotheses selection. (f) Legend of this figure. The number is the output channel of the corresponding layer.[35]	134
4.25	Overview of the pipeline. Instance masks are obtained at first through a semantic segmentation network (SegNet). With each obtained mask, the pose estimation network extracts a point cloud from depth image and crops a tight image window from color image as inputs. Color features and geometric features are densely concatenated, from which rotation and translation of the object are predicted in two separate branches.[133]	135
4.26	A graphical representation of the Iterative Closest Points iterations. [119]	135
4.27	Overview of the multi-part assembly framework of Hadfield et al. [34].	137
4.28	Overview of the total architectural design of the mixed Deep Particle Filtering approach of [21].	137
4.29	Overview of the DeepIM architecture of [73].	139

4.30	Details about the iterative training of every image-pose pair of the DeepIM architecture[73].	139
4.31	The DeepTAM [169]tracking network uses an Encoder-Decoder type architecture with direct connections between the encoding and decoding part. The decoder is used by two tasks, which are Optical Flow prediction and the generation of pose hypotheses. The Optical Flow prediction is a small stack of two convolution layers and is only active during training to stimulate the generation of motion features. The pose hypotheses generation part is a stack of downsampling convolution layers followed by a fully connected layer, which then medskips into N=64 fully connected branches sharing parameters to estimate the $\delta\xi_i$. Along with the current camera image I_C we provide a virtual keyframe (I_V, D_V) as input for the network, which is rendered using the active keyframe (I_K, D_K) and the current pose estimate TV . [169]	139
4.32	Overview of the tracking networks and the incremental pose estimation. We apply a coarse-to-fine approach to efficiently estimate the current camera pose. We train three tracking networks each specialized for a distinct resolution level corresponding to the input image dimensions (80×60) , (160×120) and (320×240) . Each network computes a pose estimate δT_i with respect to a guess TV_i . The guess TV_0 is the camera pose from the previously tracked frame. Each of the tracking networks uses the latest pose guess to generate a virtual keyframe at the respective resolution level and thereby indirectly tracking the camera with respect to the original keyframe (I_K, D_K) . The final pose estimate T_C is computed as the product of all incremental pose updates δT_i . [169]	140
5.1	The 2D Screen Coordinates: The origin is located at the top-left corner, with x-axis pointing left and y-axis pointing down, in contrary to the popular Cartesian Coordinated System. [18]	141
5.2	Raster Scan: The display updates it contents from line to line top-to-bottom and left-to-right by accessing the pixel color values saved at the frame buffer which is in front at every given moment. [18]	142
5.3	3D Graphics Rendering Pipeline: The output of one stage is fed into the next. A vertex has attribute (x,y,z) postion, (R,G,B) or (R,G,B,A) color, (n_x, n_y, n_z) vertex normal and texture. A primitive is made up of one or more vertices.The rasterizer raster-scans each primitive to produce a set of grid-aligned fragments, by interpolating the vertices.[18]	142
5.4	The hierarchy of transformation from 3D sparse verices to 2D pixels [18]	143
5.5	The sequence of Homogenous Transformations that convert the initial 3D vertex representation w.r.t. the Object Coordinate Frame to a pixel grid on the Screen Space.[18]	143
5.6	The sequence of the 4 Transformations	144
5.7	The Transformation process of one or more object(s) from their corresponding Coordinate Frame to the global World one. [18]	144
5.8	The camera is placed at the position described by the Eye vector (w.r.t. (W)) and snaps a picture of the objects in the scene at the direction given by the LookAT transformation process. [18]	144
5.9	The View Frustum of the Perspective Projection, specified by: fovy,aspect ratio, z_{near} and z_{far} , its Clip Volume Space and the final Screen Space, where the object points are transformed.	146
5.10	Orthographic projection: Camera positioned infinitely far away at $z=\infty$ and the object analogies are conserved.	147
5.11	The Vertex Rasterization process [18]	147
5.12	Graphical representation of (from left to right): Magnification with Bilinear Interpolation, Magnification with Nearest Point Sampling and Minmapping. [18]	148
5.13	The most prominent types of lighting in 3D Graphics(from left to right): Combined, Diffuse (only) and Specular (only) lighting. [18]	149
5.14	Decomposition of lighting interactions. [18]	150
5.15	A 3D CAD Dragon model of [83] at a specific pose.	150
5.16	An overview of the synthetic generation method presented in [29].	151

5.17	The results of the 2D uniform deterministic distribution for 100 and 1000 points, correspondingly.	153
5.18	The proposed 3D sphere sampling approach for $N_{points} = 100$ and $N_{points} = 1000$, correspondingly.	154
5.19	Graphical representation of a differential volumetric domain in spherical coordinates.	155
5.20	(a) Randomly sampling 20,000 points. The max distance of 2 nearest neighbors is: 0.256532 m.	155
5.21	(b) Randomly Sampling 200,0000 points and the clustering with KMeans to 20,000 points-centroids. The max distance of 2 nearest neighbors is: 0.222318 m.	155
5.22	(c) Deterministically sampling 20,000 points. The max distance of 2 nearest neighbors is: 0.295056 m.	156
5.23	(d) Deterministically Sampling 200,0000 points and the clustering with KMeans to 20,000 points-centroids. The max distance of 2 nearest neighbors is: 0.214753 m.	156
5.24	Comparing the maximum nearest neighbor Euclidean distance of randomly/deterministically sampling 3D points in a sphere without/with an intermediate point clustering KMean step of a bigger superset.	156
6.1	Conceptual illustrations of Domain Randomization.[152]	158
6.2	RGB image.	159
6.3	Depth image.	159
6.4	Occluded RGB image.	159
6.5	Occluded Depth image.	159
6.6	RGB image with background.	159
6.7	Depth image with background.	159
6.8	Occluded RGB image with background.	159
6.9	Occluded Depth image with background.	159
6.10	RGB image with random illumination.	160
6.11	RGB image with HSV noise.	160
6.12	Blurred RGB image.	160
6.13	Blurred Depth image.	160
6.14	RGB image with random Gaussian RGB noise.	160
6.15	Depth image with random Gaussian Depth noise.	160
6.16	Depth image with the Kinect noise of [83].	160
6.17	Depth image with random holes	160
6.18	Graphical representations of the data augmentation options presented in this section.	160
6.19	A RGB-D pair in the SUN3D [163] dataset. Such pairs are blended behind the object of interest for background augmentation.[163]	160
6.20	A 2D Gaussian blur kernel that is convolved with RGB-D frame pairs to blur them [29].	162
6.21	The noisy Depth image using the Kinect noise model that we describe in this Section (left) and the noise distribution image (right). We observe that the object’s edges are noised by the two lateral components and the axial one noises its core using a noise distribution that varies with the object’s depth.	162
6.22	The Gaussian random noise added to augment Depth images in [[29],[83]].	162
6.23	The experimental setup that was used in [91] for deriving the Lateral and Axial kinect noise distributions.	163
6.24	A 3D Gaussian Distribution that we use to model the Kinect noise, which is the product of 2 Lateral and 1 Axial noise components w.r.t. the camera’s principle axis[91].	164
6.25	An indicative random Observed RGB-D frame pair that we showcase random augmentation combinations of our overall strategy.	164
6.26	Full random augmentation examples of the same Observed RGB(left)-D(right) frame.	165
6.27	Overview of the SoA CNN architecture of Garon et al. [83], for object pose tracking.	165
6.28	Overview of the proposed CNN architecture for object pose tracking.	166
6.29	Standarization of the data distributions of the CNN inputs (both streams). The new mean converges to zero and the new standard deviation converges to 1.[105]	166

6.30	The “ImageNet” classification dataset,first presented in [42]. It is considered one of the toughest challenges for image-based classification and it contains 1 Million training images. As a result, weights of NN architectures pretrained on it constitute a very reliable initialization attempt for architectures later finetuned on other tasks. That is because the Imagenet dataset possesses great appearance and categorical variability, giving the current architecture a robust prior understanding of the world.	170
6.31	A Spatial Attention weight map dedicated to Foreground extraction.The highlighted pixels are the ones of the moving parts of the scene.	171
6.32	A Spatial Attention weight map dedicated to Occlusion handling. The highlighted pixels are the ones that correspond only to the object of interest.	172
6.33	All possible rotational uncertainties induced by all possible symmetrical scenarios.[27]	174
6.34	Our Rotational Estimation for the 126th frame of the “Hard Interaction Scenario”.	175
6.35	Our Rotational Estimation for the 127th frame of the “Hard Interaction Scenario” without the proposed algorithm	175
6.36	Our Rotational Estimation for the 126th frame of the “Hard Interaction Scenario”.	175
6.37	Our Rotational Estimation for the 127th frame of the “Hard Interaction Scenario” with the proposed algorithm	175
7.1	An overview of the State-of-the-Art CNN architecture proposed in [83].	181
7.2	An overview of the State-of-the-Art CNN architecture proposed in [83] without the pose rendering-based Feedback from the previous estimation, both unavailable in the training and in the inference mode of the tracker.	182
7.3	An Overview of a modification of the architecture of [83], with the use only of the 3D shape modality, as described by the the “Predicted” and “Observed” Depth maps.	184
7.4	An Overview of a modification of the architecture of [83], with the use only of the appearance modality, as described by the the “Predicted” and “Observed” RGB images.	184
7.5	An overview of the architecture of [83], where the first two “Observed” layers are initialized with the corresponding weights of a ResNet18 pretrained on ImageNet.	187
7.6	A modified version of the Network architecture proposed in [83] with the “Dense” cross-layer connections [49] replaced by Residual [37] connections.	190
7.7	LogCosh train loss progress for the architecture of Garon et al.[83], with random weight initialization.	191
7.8	LogCosh train (in red) and validation (in blue) loss progress for the architecture of Garon et al.[83], with random weight initialization.	191
7.9	LogCosh train loss progress for the architecture of Garon et al.[83], with random weight initialization and Residual inter-layer connections.	191
7.10	LogCosh train (in red) and validation (in blue) loss progress for the architecture of Garon et al.[83], with random weight initialization and Residual inter-layer connections.	191
7.11	LogCosh train loss progress for the architecture of Garon et al.[83],with Residual inter-layer connctions and “K.He-Xavier” proper combination weight initialization.	191
7.12	LogCosh train (in red) and validation (in blue) loss progress for the architecture of Garon et al.[83],with Residual inter-layer connctions and “K.He-Xavier” proper combination weight initialization.	191
7.13	An alternative version of the architecture of Garon et al.[83], with the “Observed”-“Predicted” feature fusion been executed by subtraction of the corresponding feature maps. A similar attempt has been recorder in the DeepIM [73] architecture.	192
7.14	An alternative version of the architecture of Garon et al.[83], with the “Observed”-“Predicted” feature fusion been executed by taking the Cosine Similarity (Section 3.1.4) between the corresponding feature maps. Then, the resulted constraint 1D weight map is applied to the “Observed” feature map. Lastly, the outcome is enhanced by a Residual connection from the “Observed” feature map.	192

7.15	An alternative version of the architecture of Garon et al.[83], with the “Observed”-“Predicted” feature fusion been executed by applying an attentional weight map, extracted immediately from the “Predicted” stream, to the feature map of the “Observed” stream.	192
7.16	An alternative version of the architecture of Garon et al.[83], with the “Observed”-“Predicted” feature fusion been executed in “Late” fashion, i.e. at the output feature maps of the last convolutional layer.	193
7.17	An overview of a modified version of the SoA CNN architecture of Garon et al.[83], where a Foreground pixel Attention Weight map is applied to the “Observed” feature map of the second corresponding layer, for handling Background Clutter.	195
7.18	The Spatial Attention weight map of the unsupervised module. Its accuracy is remarkable, if you consider the fact of its complete lack of supervision. It’s qualitative differences are, indeed, existent, but minor. Mostly, this map presents brighter attention leakage from the object features to the user’s hand, in comparison to its supervised counterparts. Although this is not forbidden, its profoundly impedes the tracker from reaching optimal tracking accuracy and robust generalization to difficult pose configurations.	196
7.19	Here, in the Spatial Attention map that corresponds to the second approach (Binary Cross Entropy with steady inter-loss multi-task weighting), we start to see the benevolent effects of binary mask supervision. Note that this extra supervision is of no extra computational cost, as these masks are to be computed, anyway, during the augmentation stage. The only difference of this approach is that we keep them as a guiding signal, so, if someone would want to be fully diligent, we could only speak about extra memory, instead of computational cost (insignificant nonetheless). The effectiveness of our approach starts to show from the fact that peaks, here, are sharper and the attention leakages are significantly less, since the whole module’s focus is concentrated on the feature of the object of interest.	196
7.20	The optimal Spatial Attention map weights’ learning strategy: the two losses, the main tracking one and the auxiliary BCE for Attention supervision are inter-weighted via a learnable multi-task scheme. It is clear that peaks here are the most consistently concentrated on the object’s surface, they are the sharpest and give clear hints of the object’s keypoints that are the most interesting for tracking. This success is the aftermath of leaving the weights of different losses to be free to scale up or down in order to balance out the learning difficulties of loss hardness discrepancy.	196
7.21	An overview of a modified version of the SoA CNN architecture of [83], where an unoccluded pixel Attention Weight map is applied to the “Observed” feature map of the second corresponding layer, for handling occlusions.	197
7.22	A visualized Spatial Attention weight map for Occlusion Handling overlaid on an “Observed” RGB frame. We see, here, that the Occlusion Handling task is considered more difficult for the tracker’s convolutional module to learn, w.r.t. its foreground extraction counterpart (see above). The need for supervision is more important here and will, inevitably improve the accuracy of the prediction of unoccluded pixels that are important for proper estimating of the object’s pose.	198
7.23	Clearly, the incorporation of a uniform binary unoccluded pixel mask ameliorates the sharpness of the attentional regions. The dragon’s wings emerge as its two parts that are more descriptive of its pose (in this layout, at least). Both the main, tracking loss and the auxiliary BCE one were inter-weighted with steady and equal weights (simple addition) during training.	198
7.24	Again, we can ratify the effectiveness the adaptive learnable multi-task loss weighting scheme has not only in the numerical tracking accuracy but in its visual reasoning as well. Both wings remain the most important feature for the tracker to focus in order to accomplish its goal. However, deeper learning of the auxiliary task has generated an implicit keypoint hierarchy, where attention has shifted to one wing in a more bold manner than the other.	198

7.25	A visual demonstration of the Occlusion Handling Spatial Attention weight map that is learned by substituting the convolutional unit in the relevant module with a ConvLSTM one. Attention leakage is slightly increased and distributional peaks have been mitigated, two factors that justify the slightly diminished accuracy of this alternative method. It seems that LSTM (even this Convolutional variant) justify their reputation of being hard to train due to the combination of sequential and parallel weight adaptation during the BPTT algorithm backward run.	198
7.26	Although Gumbel Softmax, whose use in the Occlusion Handling Attentional module we witness in this figure, seems like a methodological modification with theoretical charms, (since it allows for increased degrees of freedom in shaping the attention distribution and balancing out attention sharpness), in practice, it is proven suboptimal as simultaneous learning of the hardness level of the attentional distribution is proven an extra task that loads the backpropagation algorithm with extra, unnecessary performance burdens. As a result, nor the tracker’s accuracy nor its robustness see any spike of statistical significance and this alternation can be considered extraneous.	198
7.27	An overview of the modification of the SoA CNN architecture of [83], where we employ an hierarchical Foreground Extraction/Occlusion handling Soft Spatial Attention scheme to highlight the desired regions of interest in the feature maps’s spatial dimensions. . .	202
7.28	An overview of the modification of the SoA CNN architecture of [83], where we employ a parallel Foreground Extraction/Occlusion handling Soft Spatial Attention scheme to highlight the desired regions of interest in the feature maps’s spatial dimensions.	202
7.29	The Foreground/Clutter handling Spatial Attention weight map of the Hierarchical method, overlaid on an “Observed” RGB frame. Although the module has assigned no focus at all to the static background (i.e. the visual attributes of the room), most of the attentional distributional peaks have been gathered towards the human user instead of the object of interest. We blame this learning inefficiency to the short-term vanishing gradient effect that inevitably occurs as the gradient backward stream passes multiple softmax activations in the row. As a result, although occlusion detection is of quite high reliability the hierarchical implementation of one weight map over the other would result to a pair of sharp peaks that lie inside the object visual region (i.e. on the dragon’s wings) and a third one that stays on the user’s hand, which results in confusing the tracker and, thus, reducing its performance.	203
7.30	The Occlusion handling Spatial Attention weight map of the Hierarchical method, overlaid on an “Observed” RGB frame. Since this map has been sufficiently learned and due to his topographical location in the architecture (i.e. a layer after the foreground attention module), we can verify that our vanishing gradient arguments stands as it is.	203
7.31	The Foreground/Clutter handling Spatial Attention weight map of the Parallel method, overlaid on an “Observed” RGB frame. This time, the Foreground extraction task is parallel to the Occlusion handling one, which exonerates gradients of the extra burden to pass two heavily nonlinear activation function in the row, before being connected again with their main convolutional stream counterparts.	203
7.32	The Occlusion handling Spatial Attention weight map of the Parallel method, overlaid on an “Observed” RGB frame. Maybe surprisingly, the fact that the two tasks are learned in parallel has given the extra freedom to the Occlusion handling module to weight the most important keypoint in a more explicit manner and, thus, create a lucid region of unoccluded pixel visual interest, since the other regions are left for its siamese Foreground extraction module to highlight.	203
8.1	The SoA architectural design of Garon et al. [83]	208
8.2	The SoA architectural design of Garon et al. [83], with the “Observed” stream initialized by the weights of a pretrained ResNet18.	208
8.3	The SoA architectural design of Garon et al. [83], with the “Observed” stream initialized by the weights of a pretrained ResNet18 and with residual inter-layer connection. . . .	209

8.4	Overview of the architecture of the proposed approach. The previous layout is enhanced by the two parallel attention modules and the 9D pose parameter regression, suitable for the use of Geodesic rotation loss.	209
8.5	(a) Demonstration of an “Observed” RGB frame of the “Stability near” scenario with a rendered predicted pose of the “Dragon” model, generated by the baseline tracker of Garon et al.[83].	213
8.6	(b) Demonstration of an “Observed” RGB frame of the “Stability near” scenario with a rendered predicted pose of the “Dragon” model, generated by the baseline tracker of Garon et al.[83] with Residual inter-layer connections and augmentation/initialization/sampling strategic improvements.	213
8.7	(c) Demonstration of an “Observed” RGB frame of the “Stability near” scenario with a rendered predicted pose of the “Dragon” model, generated by the proposed tracker.	213
8.8	(a) Demonstration of the Foreground Extraction Attention map provided by the proposed tracker in the “Stability near” scenario.	215
8.9	(b) Demonstration of the Occlusion Handling Attention map provided by the proposed tracker in the “Stability near” scenario.	215
8.10	3D Translational and Rotational error metrics’ plot, for the “Stability near” scenario of the Dragon 3D CAD model, in the case of tracker re-iteration every 15 frames.	216
8.11	3D Translational and Rotational error metrics’ plot, for the “Stability near” scenario of the Dragon 3D CAD model, in the case of tracker re-iteration only after a failure.	216
8.12	(a) Demonstration of an “Observed” RGB frame of the “Stability far” scenario with a rendered predicted pose of the “Dragon” model, generated by the baseline tracker of Garon et al.[83].	217
8.13	(b) Demonstration of an “Observed” RGB frame of the “Stability far” scenario with a rendered predicted pose of the “Dragon” model, generated by the baseline tracker of Garon et al.[83], with Residual inter-layer connections and augmentation/ initialization/sampling strategic improvements.	217
8.14	(c) Demonstration of an “Observed” RGB frame of the “Stability far” scenario with a rendered predicted pose of the model generated by the proposed tracker.	217
8.15	(a) Demonstration of the Foreground Extraction Attention map provided by the proposed tracker in the “Stability far” scenario.	217
8.16	(b) Demonstration of the Occlusion Handling Attention map provided by the proposed tracker in the “Stability far” scenario.	217
8.17	3D Translational and Rotational error metrics’ plot, for the “Stability far” scenario of the Dragon 3D CAD model, in the case of tracker re-iteration every 15 frames.	218
8.18	3D Translational and Rotational error metrics’ plot, for the “Stability far” scenario of the Dragon 3D CAD model, in the case of tracker re-iteration only after a failure.	218
8.19	(a) Demonstration of an “Observed” RGB frame of the “75% Vertical Occlusion” scenario with a rendered predicted pose of the model generated by the baseline tracker of Garon et al.[83]	219
8.20	(b) Demonstration of an “Observed” RGB frame of the “75% Vertical Occlusion” scenario with a render predicted pose of the model generated by the baseline tracker of Garon et al.[77] with Residual inter-layer connections and augmentation/ initialization/sampling strategic improvements.	219
8.21	(c) Demonstration of an “Observed” RGB frame of the “75% vertical occlusion scenario” scenario with a render predicted pose of the model generated by the proposed tracker.	219
8.22	(a) Demonstration of the Foreground Extraction Attention map provided by the proposed tracker in the “75% Vertical Occlusion” scenario.	220
8.23	(b) Demonstration of the Occlusion Handling Attention map provided by the proposed tracker in the “75% Vertical Occlusion” scenario.	220
8.24	3D Translational and Rotational error metrics’ plot, for the “75 % Horizontal Occlusion” scenario of the Dragon 3D CAD model, in the case of tracker re-iteration every 15 frames.	221

8.25	3D Translational and Rotational error metrics' plot, for the "75% Horizontal Occlusion" scenario of the Dragon 3D CAD model, in the case of tracker re-iteration only after a failure.	221
8.26	(a) Demonstration of an "Observed" RGB frame of the "75% Vertical Occlusion" scenario with a render predicted pose of the model generated by the baseline tracker of Garon et al.[83].	222
8.27	(b) Demonstration of an "Observed" RGB frame of the "75% Horizontal Occlusion" scenario with a render predicted pose of the model generated by the baseline tracker of Garon et al.[83] with Residual inter-layer connections and augmentation/ initialization/sampling strategic improvements.	222
8.28	(c) Demonstration of an "Observed" RGB frame of the "75% Horizontal Occlusion" scenario with a render predicted pose of the model generated by the proposed tracker.	222
8.29	(a) Demonstration of the Foreground Extraction Attention map provided by the proposed tracker in the "75% Horizontal Occlusion" scenario.	222
8.30	(b) Demonstration of the Occlusion Handling Attention map provided by the proposed tracker in the "75% Horizontal Occlusion" scenario.	222
8.31	3D Translational and Rotational error metrics' plot, for the "75% Horizontal Occlusion" scenario of the Dragon 3D CAD model, in the case of tracker re-iteration every 15 frames.	223
8.32	3D Translational and Rotational error metrics' plot, for the "75% Horizontal Occlusion" scenario of the Dragon 3D CAD model, in the case of tracker re-iteration only after a failure.	223
8.33	(a) Demonstration of an "Observed" RGB frame of the "Translation Only Interaction" scenario with a render predicted pose of the model generated by the baseline tracker of Garon et al.[83].	224
8.34	(b) Demonstration of an "Observed" RGB frame of the "Translation Only Interaction" scenario with a render predicted pose of the model generated by the baseline tracker of Garon et al.[83] with Residual inter-layer connections and augmentation/ initialization/sampling strategic improvements.	224
8.35	(c) Demonstration of an "Observed" RGB frame of the "Translation Only Interaction" scenario with a render predicted pose of the model generated by the proposed tracker.	224
8.36	(a) Demonstration of the Foreground Extraction Attention map provided by the proposed tracker in the "Translation Only Interaction" scenario.	224
8.37	(b) Demonstration of the Occlusion Handling Attention map provided by the proposed tracker in the "Translation Only Interaction" scenario.	224
8.38	3D Translational and Rotational error metrics' plot, for the "Translation Only Interaction" scenario of the Dragon 3D CAD model, in the case of tracker re-iteration every 15 frames.	225
8.39	3D Translational and Rotational error metrics' plot, for the "Translation Only Interaction" scenario of the Dragon 3D CAD model, in the case of tracker re-iteration only after a failure.	225
8.40	(a) Demonstration of an "Observed" RGB frame of the "Rotation Only Interaction" scenario with a render predicted pose of the model generated by the baseline tracker of Garon et al.[83].	226
8.41	(b) Demonstration of an "Observed" RGB frame of the "Rotation Only Interaction" scenario with a render predicted pose of the model generated by the baseline tracker of Garon et al.[83] with Residual inter-layer connections and augmentation/ initialization/sampling strategic improvements.	226
8.42	(c) Demonstration of an "Observed" RGB frame of the "Rotation Only Interaction" scenario with a render predicted pose of the model generated by the proposed tracker.	226
8.43	(a) Demonstration of the Foreground Extraction Attention map provided by the proposed tracker in the "Rotation Only Interaction" scenario.	227
8.44	(b) Demonstration of the Occlusion Handling Attention map provided by the proposed tracker in the "Rotation Only Interaction" scenario.	227

8.45	3D Translational and Rotational error metrics' plot, for the "Rotation Only Interaction" scenario of the Dragon 3D CAD model, in the case of tracker re-iteration every 15 frames.	228
8.46	3D Translational and Rotational error metrics' plot, for the "Rotation Only Interaction" scenario of the Dragon 3D CAD model, in the case of tracker re-iteration only after a failure.	228
8.47	(a) Demonstration of an "Observed" RGB frame of the "Full Interaction" scenario with a render predicted pose of the model generated by the baseline tracker of Garon et al.[83].	230
8.48	(b) Demonstration of an "Observed" RGB frame of the "Full Interaction" scenario with a render predicted pose of the model generated by the baseline tracker of Garon et al.[83] with Residual inter-layer connections and augmentation/ initialization/sampling strategic improvements.	230
8.49	(c) Demonstration of an "Observed" RGB frame of the "Full Interaction" scenario with a render predicted pose of the model generated by the proposed my tracker.	230
8.50	(a) Demonstration of the Foreground Extraction Attention map provided by the proposed tracker in the "Full Interaction" scenario.	230
8.51	(b) Demonstration of the Occlusion Handling Attention map provided by the proposed tracker in the "Full Interaction" scenario.	230
8.52	3D Translational and Rotational error metrics' plot, for the "Full Interaction" scenario of the Dragon 3D CAD model, in the case of tracker re-iteration every 15 frames.	231
8.53	3D Translational and Rotational error metrics' plot, for the "Full Interaction" scenario of the Dragon 3D CAD model, in the case of tracker re-iteration only after a failure.	231
8.54	(a) Demonstration of an "Observed" RGB frame of the "Hard Interaction" scenario with a render predicted pose of the model generated by the baseline tracker of Garon et al.[83].	232
8.55	(b) Demonstration of an "Observed" RGB frame of the "Hard Interaction" scenario with a render predicted pose of the model generated by the baseline tracker of Garon et al.[83] with Residual inter-layer connections and augmentation/ initialization/sampling strategic improvements.	232
8.56	(c) Demonstration of an "Observed" RGB frame of the "Hard Interaction" scenario with a render predicted pose of the model generated by the proposed tracker.	232
8.57	(a) Demonstration of the Foreground Extraction Attention map provided by the proposed tracker in the "Hard Interaction" scenario.	232
8.58	(b) Demonstration of the Occlusion Handling Attention map provided by the proposed tracker in the "Hard Interaction" scenario.	232
8.59	3D Translational and Rotational error metrics' plot, for the "Hard Interaction" scenario of the Dragon 3D CAD model, in the case of tracker re-iteration every 15 frames.	234
8.60	3D Translational and Rotational error metrics' plot, for the "Hard Interaction" scenario of the Dragon 3D CAD model, in the case of tracker re-iteration only after a failure.	234
8.61	The 6D Pose Trajectories of the "Dragon" object model, in time, for the "Hard Interaction" scenario. It is evident, in both cases, from this perspective as well, that our proposed approach outperforms the SoA of Garon et al.[83] by far, as it produces smaller errors and fewer failures. All length units are in mm and all rotations in degrees.	234
8.62	(a) Demonstration of an "Observed" RGB frame of the "Stability near" scenario with a rendered predicted pose of the "Cookie Jar" model generated by the baseline tracker of Garon et al.[83].	239
8.63	(b) Demonstration of an "Observed" RGB frame of the "Stability near" scenario with a rendered predicted pose of the "Cookie Jar" model generated by the baseline tracker of Garon et al.[83] with Residual inter-layer connections and sampling/initialization/augmentation improvements.	239
8.64	(c) Demonstration of an "Observed" RGB frame of the "Stability near" scenario with a rendered predicted pose of the "Cookie Jar" model generated by proposed tracker of this work.	239

8.65	(d) Demonstration of an “Observed” RGB frame of the “Stability near” scenario with a rendered predicted pose of the “Cookie Jar” model generated by proposed tracker, with the special case of unique rotational symmetry parameter, that remains frozen after training, of this work.	239
8.66	(e) Demonstration of an “Observed” RGB frame of the “Stability near” scenario with a rendered predicted pose of the “Cookie Jar” model generated by proposed tracker, with a learnable batch of rotational parameters, of this work.	239
8.67	(f) Demonstration of the Occlusion Handling Attention map provided by the proposed tracker, with a rotational parameter regression, in the “Stability near” scenario and tested on the “Cookie Jar” 3D model.	239
8.68	(g) Demonstration of the Occlusion Handling Attention map provided by the proposed tracker, with a per-pose pair optimal selection of the symmetry parameter, out of a nearly Uniform batch of Continuously Rotational Symmetry parameters, in the “Stability near” scenario and tested on the “Cookie Jar” 3D model.	239
8.69	(a) Demonstration of the Foreground Extraction Attention map of the “Cookie Jar” 3D model, provided by the proposed tracker in the “Stability near” scenario, without accounting for symmetries.	240
8.70	(b) Demonstration of the Occlusion Handling Attention map of the “Cookie Jar” 3D model, provided by the proposed tracker in the “Stability near” scenario, without accounting for symmetries.	240
8.71	(c) Demonstration of the Foreground Extraction Attention map of the “Cookie Jar” 3D model, provided by the proposed tracker, with a unique, frozen, learnable symmetrical rotational parameter, in the “Stability near” scenario.	240
8.72	(d) Demonstration of the Occlusion Handling Attention map of the “Cookie Jar” 3D model, provided by the proposed tracker, with a unique, frozen, learnable symmetrical rotational parameter, in the “Stability near” scenario.	240
8.73	(e) Demonstration of the Foreground Extraction Attention map of the “Cookie Jar” 3D model, provided by the proposed tracker, with a learnable batch of symmetrical rotational parameters, in the “Stability near” scenario.	240
8.74	(f) Demonstration of the Occlusion Handling Attention map of the “Cookie Jar” 3D model, provided by the proposed tracker, with the mean of a batch of frozen, learnable symmetrical rotational parameters, in the “Stability near” scenario.	240
8.75	(g) Demonstration of the Foreground Extraction Attention map of the “Cookie Jar” 3D model, provided by the proposed tracker, with a per-pose regressed learnable symmetrical rotational parameter, in the “Stability near” scenario.	240
8.76	(h) Demonstration of the Occlusion Handling Attention map of the “Cookie Jar” 3D model, provided by the proposed tracker, with a per-pose regressed learnable symmetrical rotational parameter, in the “Stability near” scenario.	240
8.77	(i) Demonstration of the Foreground Extraction Attention map of the “Cookie Jar” 3D model, provided by the proposed tracker, with a per-pose pair optimal selection of the symmetry parameter, out of a nearly Uniform batch of Continuously Rotational Symmetry parameters, in the “Stability near” scenario.	240
8.78	(j) Demonstration of the Occlusion Handling Attention map of the “Cookie Jar” 3D model, provided by the proposed tracker, with a per-pose pair optimal selection of the symmetry parameter, out of a nearly Uniform batch of Continuously Rotational Symmetry parameters, in the “Stability near” scenario.	240
8.79	3D Translational and Rotational error metrics’ plot, for the “Stability near” scenario of the Cookie Jar 3D CAD model, in the case of tracker re-iteration every 15 frames. . . .	242
8.80	3D Translational and Rotational error metrics’ plot, for the “Stability near” scenario of the Cookie Jar 3D CAD model, in the case of tracker re-iteration only after a failure. . .	242
8.81	(a) Demonstration of an “Observed” RGB frame of the “Stability far” scenario with a rendered predicted pose of the “Cookie Jar” model generated by the baseline tracker of Garon et al.[83].	244

8.82	(b) Demonstration of an “Observed” RGB frame of the “Stability far” scenario with a rendered predicted pose of the “Cookie Jar” model generated by the baseline tracker of Garon et al.[83] with Residual inter-layer connections and sampling/ initialization/augmentation improvements.	244
8.83	(c) Demonstration of an “Observed” RGB frame of the “Stability far” scenario with a rendered predicted pose of the “Cookie Jar” model generated by proposed tracker of this work.	244
8.84	(d) Demonstration of an “Observed” RGB frame of the “Stability far” scenario with a rendered predicted pose of the “Cookie Jar” model generated by proposed tracker, with the special case of unique rotational symmetry parameter, that remains frozen after training, of this work.	244
8.85	(e) Demonstration of an “Observed” RGB frame of the “Stability far” scenario with a rendered predicted pose of the “Cookie Jar” model generated by proposed tracker, with a learnable batch of rotational parameters, of this work.	244
8.86	(f) of an “Observed” RGB frame of the “Stability far” scenario with a rendered predicted pose of the “Cookie Jar” model generated by proposed tracker, with a per-pose rotational parameter regression.	244
8.87	(g) Demonstration of an “Observed” RGB frame of the “Stability far” scenario with a rendered predicted pose of the “Cookie Jar” model generated by proposed tracker, with a per-pose pair optimal selection out of a nearly uniform batch of continuously rotational symmetry parameters.	244
8.88	(a) Demonstration of the Foreground Extraction Attention map of the “Cookie Jar” model, provided by the proposed tracker in the “Stability far” scenario	245
8.89	(b) Demonstration of the Occlusion Handling Attention map of the “Cookie Jar” model, provided by the proposed tracker in the “Stability far” scenario.	245
8.90	(c) Demonstration of the Foreground Extraction Attention map provided by the proposed tracker, with a unique,frozen, rotational symmetry parameter, in the “Stability far” scenario	245
8.91	(d) Demonstration of the Occlusion Handling Attention map of the “Cookie Jar” model, provided by the proposed tracker, with a unique, frozen, learnable rotational symmetry parameter in the “Stability far” scenario.	245
8.92	(e) Demonstration of the Foreground Extraction Attention map provided by the proposed tracker, with a batch of frozen, learnable rotational symmetry parameters, in the “Stability far” scenario	245
8.93	(f) Demonstration of the Occlusion Handling Attention map of the “Cookie Jar” model, provided by the proposed tracker, with the mean of a batch of frozen learnable in the “Stability far” scenario.	245
8.94	(g) Demonstration of the Foreground Extraction Attention map provided by the proposed tracker, with per-pose regression of a learnable rotational symmetry parameter, in the “Stability far” scenario	245
8.95	(h) Demonstration of the Occlusion Handling Attention map of the “Cookie Jar” model, provided by the proposed tracker, with a per-pose regressed learnable rotational symmetry parameter, in the “Stability far” scenario.	245
8.96	(i) Demonstration of the Foreground Extraction Attention map provided by the proposed tracker, with a per-pose pair optimal selection out of a nearly uniform batch of continuously rotational symmetry parameters, in the “Stability far” scenario	245
8.97	(j) Demonstration of the Occlusion Handling Attention map of the “Cookie Jar” model, provided by the proposed tracker, with a per-pose pair optimal selection out of a nearly uniform batch of continuously rotational symmetry parameters, in the “Stability far” scenario.	245
8.98	Special case of rotational symmetry parameter estimation: 3D Translational and Rotational error metrics’ plot, for the “Stability far” scenario of the Cookie Jar 3D CAD model, in the case of tracker re-iteration every 15 frames.	247

8.99	Special case of rotational symmetry parameter estimation: 3D Translational and Rotational error metrics' plot, for the "Stability far" scenario of the Cookie Jar 3D CAD model, in the case of tracker re-iteration every 15 frames.	247
8.100	(a) Demonstration of an "Observed" RGB frame of the "75% Horizontal Occlusion" scenario with a rendered predicted pose of the "Cookie Jar" model generated by the baseline tracker of Garon et al.[83].	249
8.101	(b) Demonstration of an "Observed" RGB frame of the "75% Horizontal Occlusion" scenario with a rendered predicted pose of the "Cookie Jar" model generated by the baseline tracker of Garon et al.[83] with Residual inter-layer connections and sampling/initialization/augmentation improvements.	249
8.102	(c) Demonstration of an "Observed" RGB frame of the "75% Horizontal Occlusion" scenario with a rendered predicted pose of the "Cookie Jar" model generated by proposed tracker of this work.	249
8.103	(d) Demonstration of an "Observed" RGB frame of the "75% Horizontal Occlusion" scenario with a rendered predicted pose of the "Cookie Jar" model generated by proposed tracker, with the special case of unique rotational symmetry parameter, that remains frozen after training, of this work.	249
8.104	(e) Demonstration of an "Observed" RGB frame of the "75% Horizontal Occlusion" scenario with a rendered predicted pose of the "Cookie Jar" model generated by proposed tracker, with a learnable batch of rotational parameters, of this work.	249
8.105	(f) Demonstration of the Occlusion Handling Attention map provided by the proposed tracker, with a continuous rotational parameter regression, in the "75% Horizontal Occlusion" scenario and tested on the "Cookiejar" 3D model.	249
8.106	(g) Demonstration of the Occlusion Handling Attention map provided by the proposed tracker, with a per-pose pair optimal selection out of a nearly uniform batch of continuously rotational symmetry parameters, in the "75% Horizontal Occlusion" scenario and tested on the "Cookiejar" 3D model.	249
8.107	(a) Demonstration of the Foreground Extraction Attention map of the "Cookie Jar" model, provided by the proposed tracker in the "75% Horizontal Occlusion" scenario	250
8.108	(b) Demonstration of the Occlusion Handling Attention map of the "Cookie Jar" model, provided by the proposed tracker in the "75% Horizontal Occlusion" scenario.	250
8.109	(c) Demonstration of the Foreground Extraction Attention map of the "Cookie Jar" model, provided by the proposed tracker, with a unique,frozen learnable rotational symmetry paramater, in the "75% Horizontal Occlusion" scenario	250
8.110	(d) Demonstration of the Occlusion Handling Attention map of the "Cookie Jar" model, provided by the proposed tracker, with a unique,frozen learnable rotational symmetry parameter, in the "75% Horizontal Occlusion" scenario.	250
8.111	(e) Demonstration of the Foreground Extraction Attention map of the "Cookie Jar" model, provided by the proposed tracker, with the mean of a batch of frozen, learnable continuous rotational symmetry parameters, in the "75% Horizontal Occlusion" scenario	250
8.112	(f) Demonstration of the Occlusion Handling Attention map of the "Cookie Jar" model, provided by the proposed tracker, with the mean of a batch of frozen learnable rotational symmetry parameters, in the "75% Horizontal Occlusion" scenario.	250
8.113	(g) Demonstration of the Foreground Extraction Attention map of the "Cookie Jar" model, provided by the proposed tracker, with the per-pose regressed learnable continuous rotational symmetry parameter, in the "75% Horizontal Occlusion" scenario	250
8.114	(h) Demonstration of the Occlusion Handling Attention map of the "Cookie Jar" model, provided by the proposed tracker, with a per-pose regressed learnable rotational symmetry parameter, in the "75% Horizontal Occlusion" scenario.	250
8.115	(i) Demonstration of the Foreground Extraction Attention map of the "Cookie Jar" model, provided by the proposed tracker, with the per-pose pair optimal selection out of a nearly uniform batch of continuously rotational symmetry parameters, in the "75% Horizontal Occlusion" scenario	250

8.116	(j) Demonstration of the Occlusion Handling Attention map of the “Cookie Jar” model, provided by the proposed tracker, with a per-pose pair optimal selection out of a nearly uniform batch of continuously rotational symmetry parameters, in the “75% Horizontal Occlusion” scenario.	250
8.117	Special case of rotational symmetry parameter estimation: 3D Translational and Rotational error metrics’ plot, for the “75% Horizontal Occlusion” scenario of the Cookie Jar 3D CAD model, in the case of tracker re-iteration only after a failure.	252
8.118	Special case of rotational symmetry parameter estimation: 3D Translational and Rotational error metrics’ plot, for the “75% Horizontal Occlusion” scenario of the Cookie Jar 3D CAD model, in the case of tracker re-iteration only after a failure.	252
8.119	(a) Demonstration of an “Observed” RGB frame of the “75% Vertical Occlusion” scenario with a rendered predicted pose of the “Cookie Jar” model generated by the baseline tracker of Garon et al.[83].	254
8.120	(b) Demonstration of an “Observed” RGB frame of the “75% Vertical Occlusion” scenario with a rendered predicted pose of the “Cookie Jar” model generated by the baseline tracker of Garon et al.[83] with Residual inter-layer connections and sampling/ initialization/augmentation improvements.	254
8.121	(c) Demonstration of an “Observed” RGB frame of the “75% Vertical Occlusion” scenario with a rendered predicted pose of the “Cookie Jar” model generated by proposed tracker of this work.	254
8.122	(d) Demonstration of an “Observed” RGB frame of the “75% Vertical Occlusion” scenario with a rendered predicted pose of the “Cookie Jar” model generated by proposed tracker, with the special case of unique rotational symmetry parameter, that remains frozen after training, of this work.	254
8.123	(e) Demonstration of an “Observed” RGB frame of the “75% Vertical Occlusion” scenario with a rendered predicted pose of the “Cookie Jar” model generated by proposed tracker, with a learnable batch of rotational parameters, of this work.	254
8.124	(f) Demonstration of the Occlusion Handling Attention map provided by the proposed tracker, with a rotational parameter regression, in the “75% Vertical Occlusion” scenario and tested on the “Cookiejar” 3D model.	254
8.125	(g) Demonstration of the Occlusion Handling Attention map provided by the proposed tracker, with a rotational parameter regression, in the “75% Vertical Occlusion” scenario and tested on the “Cookiejar” 3D model.	254
8.126	(a) Demonstration of the Foreground Extraction Attention map of the “Cookie Jar” model, provided by the proposed tracker in the “75% Vertical Occlusion” scenario.	255
8.127	(b) Demonstration of the Occlusion Handling Attention map of the “Cookie Jar” model, provided by the proposed tracker in the “75% Vertical Occlusion” scenario.	255
8.128	(c) Demonstration of the Foreground Extraction Attention map of the “Cookie Jar” model, provided by the proposed tracker, with a unique,frozen learnable continuous rotational symmetry parameter, in the “75% Vertical Occlusion” scenario.	255
8.129	(d) Demonstration of the Occlusion Handling Attention map of the “Cookie Jar” model, provided by the proposed tracker, with a unique,frozen learnable rotational symmetry parameter, in the “75% Vertical Occlusion” scenario.	255
8.130	(e) Demonstration of the Foreground Extraction Attention map of the “Cookie Jar” model provided by the proposed tracker, with the mean of a batch of frozen, learnable continuous rotational symmetry parameters, in the “75% Vertical Occlusion” scenario.	255
8.131	(f) Demonstration of the Occlusion Handling Attention map of the “Cookie Jar” model, provided by the proposed tracker, with the mean of a batch of frozen,learnable, rotational symmetry parameters, in the “75% Vertical Occlusion” scenario.	255
8.132	(g) Demonstration of the Foreground Extraction Attention map of the “Cookie Jar” model, provided by the proposed tracker, with a per-pose regressed learnable continuous rotational symmetry parameter, in the “75% Vertical Occlusion” scenario.	255

8.133	(h) Demonstration of the Occlusion Handling Attention map of the “Cookie Jar” model, provided by the proposed tracker, with a per-pose regressed learnable rotational symmetry parameter, in the “75% Vertical Occlusion” scenario.	255
8.134	(g) Demonstration of the Foreground Extraction Attention map of the “Cookie Jar” model, provided by the proposed tracker, with a per-pose regressed learnable continuous rotational symmetry parameter, in the “75% Vertical Occlusion” scenario.	255
8.135	(h) Demonstration of the Occlusion Handling Attention map of the “Cookie Jar” model, provided by the proposed tracker, with a per-pose regressed learnable rotational symmetry parameter, in the “75% Vertical Occlusion” scenario.	255
8.136	Special case of rotational symmetry parameter estimation: 3D Translational and Rotational error metrics’ plot, for the “Translation Only Interaction” scenario of the Cookie Jar 3D CAD model, in the case of tracker re-iteration only after a failure. . . .	257
8.137	Special case of rotational symmetry parameter estimation: 3D Translational and Rotational error metrics’ plot, for the “Translation Only Interaction” scenario of the Cookie Jar 3D CAD model, in the case of tracker re-iteration only after a failure. . . .	257
8.138	(a) Demonstration of an “Observed” RGB frame of the “Translation Only Interaction” scenario with a rendered predicted pose of the “Cookie Jar” model generated by the baseline tracker of Garon et al.[83].	259
8.139	(b) Demonstration of an “Observed” RGB frame of the “Translation Only Interaction” scenario with a rendered predicted pose of the “Cookie Jar” model generated by the baseline tracker of Garon et al.[83] with Residual inter-layer connections and sampling/initialization/augmentation improvements.	259
8.140	(c) Demonstration of an “Observed” RGB frame of the "Translation Only Interaction" scenario with a rendered predicted pose of the “Cookie Jar” model generated by proposed tracker of this work.	259
8.141	(d) Demonstration of an “Observed” RGB frame of the “Translation Only Interaction” scenario with a rendered predicted pose of the “Cookie Jar” model generated by proposed tracker, with the special case of unique rotational symmetry parameter, that remains frozen after training, of this work.	259
8.142	(e) Demonstration of an “Observed” RGB frame of the "Translation Only Interaction" scenario with a rendered predicted pose of the “Cookie Jar” model generated by proposed tracker, with a learnable batch of rotational parameters, of this work.	259
8.143	(h) Demonstration of the Occlusion Handling Attention map provided by the proposed tracker, with a rotational parameter regression, in the “Translation Only Interaction” scenario and tested on the “Cookiejar” 3D model.	259
8.144	(g) Demonstration of the Occlusion Handling Attention map provided by the proposed tracker, with a per-pose pair optimal selection out of a nearly uniform batch of continuously rotational symmetry parameters, in the “Translation Only Interaction” scenario and tested on the “Cookiejar” 3D model.	259
8.145	(a) Demonstration of the Occlusion Handling Attention map of the “Cookie Jar” model, provided by the proposed tracker, in the “Translation Only Interaction” scenario.	260
8.146	(b) Demonstration of the Occlusion Handling Attention map of the “Cookie Jar” model, provided by the proposed tracker in the “Translation Only Interaction” scenario.	260
8.147	(c) Demonstration of the Occlusion Handling Attention map of the “Cookie Jar” model, provided by the proposed tracker, with a unique, frozen learnable continuous rotational symmetry parameters, in the “Translation Only Interaction” scenario.	260
8.148	(d) Demonstration of the Occlusion Handling Attention map of the “Cookie Jar” model, provided by the proposed tracker, with a unique, frozen learnable rotational symmetry parameter, in the “Translation Only Interaction” scenario.	260
8.149	(e) Demonstration of the Occlusion Handling Attention map of the “Cookie Jar” model, provided by the proposed tracker, with the mean of a batch of frozen, learnable learnable continuous rotational symmetry parameter, in the “Translation Only Interaction” scenario.	260

8.150	(f) Demonstration of the Occlusion Handling Attention map of the “Cookie Jar” model, provided by the proposed tracker, with the mean of a batch of frozen, learnable rotational symmetry parameters, in the “Translation Only Interaction” scenario.	260
8.151	(g) Demonstration of the Occlusion Handling Attention map of the “Cookie Jar” model, provided by the proposed tracker, with a per-pose regressed learnable rotational symmetry parameter, in the “Translation Only Interaction” scenario.	260
8.152	(h) Demonstration of the Occlusion Handling Attention map of the “Cookie Jar” model, provided by the proposed tracker, with a per-pose regressed learnable rotational symmetry parameter, in the “Translation Only Interaction” scenario.	260
8.153	(i) Demonstration of the Occlusion Handling Attention map of the “Cookie Jar” model, provided by the proposed tracker, with a per-pose regressed learnable rotational symmetry parameter, in the “Translation Only Interaction” scenario.	260
8.154	(j) Demonstration of the Occlusion Handling Attention map of the “Cookie Jar” model, provided by the proposed tracker, with a per-pose regressed learnable rotational symmetry parameter, in the “Translation Only Interaction” scenario.	260
8.155	Special case of rotational symmetry parameter estimation: 3D Translational and Rotational error metrics’ plot, for the “Rotation Only Interaction” scenario of the Cookie Jar 3D CAD model, in the case of tracker re-iteration only after a failure.	262
8.156	Special case of rotational symmetry parameter estimation: 3D Translational and Rotational error metrics’ plot, for the “Translation Only Interaction” scenario of the Cookie Jar 3D CAD model, in the case of tracker re-iteration only after a failure.	262
8.157	(a) Demonstration of an “Observed” RGB frame of the “Rotation Only Interaction” scenario with a rendered predicted pose of the “Cookie Jar” model generated by the baseline tracker of Garon et al.[83].	264
8.158	(b) Demonstration of an “Observed” RGB frame of the “Rotation Only Interaction” scenario with a rendered predicted pose of the “Cookie Jar” model generated by the baseline tracker of Garon et al.[83] with Residual inter-layer connections and sampling/initialization/augmentation improvements.	264
8.159	(c) Demonstration of an “Observed” RGB frame of the "Rotation Only Interaction" scenario with a rendered predicted pose of the “Cookie Jar” model generated by proposed tracker of this work.	264
8.160	(d) Demonstration of an “Observed” RGB frame of the “Rotation Only Interaction” scenario with a rendered predicted pose of the “Cookie Jar” model generated by proposed tracker, with the special case of unique rotational symmetry parameter, that remains frozen after training, of this work.	264
8.161	(e) Demonstration of an “Observed” RGB frame of the "Rotation Only Interaction" scenario with a rendered predicted pose of the “Cookie Jar” model generated by proposed tracker, with a learnable batch of rotational parameters, of this work.	264
8.162	(f) Demonstration of the Occlusion Handling Attention map provided by the proposed tracker, with a rotational parameter regression, in the “Rotation Only Interaction” scenario and tested on the “Cookiejar” 3D model.	264
8.163	(g) Demonstration of the Occlusion Handling Attention map provided by the proposed tracker, with a rotational parameter regression, in the “Rotation Only Interaction” scenario and tested on the “Cookiejar” 3D model.	264
8.164	(a) Demonstration of the Occlusion Handling Attention map of the “Cookie Jar” model, provided by the proposed tracker in the “Rotation Only Interaction” scenario.	265
8.165	(b) Demonstration of the Occlusion Handling Attention map of the “Cookie Jar” model, provided by the proposed tracker in the “Rotation Only” scenario.	265
8.166	(c) Demonstration of the Occlusion Handling Attention map of the “Cookie Jar” model, provided by the proposed tracker, with a unique, frozen learnable continuous rotational symmetry parameter, in the “Rotation Only Interaction” scenario.	265
8.167	(d) Demonstration of the Occlusion Handling Attention map of the “Cookie Jar” model, provided by the proposed tracker, with a unique, frozen learnable rotational symmetry parameter, in the “Rotation Only” scenario.	265

8.168	(e) Demonstration of the Occlusion Handling Attention map of the “Cookie Jar” model, provided by the proposed tracker, with the mean of a batch of frozen, learnable continuous rotational symmetry parameter, in the “Rotation Only Interaction” scenario.	265
8.169	(f) Demonstration of the Occlusion Handling Attention map of the “Cookie Jar” model, provided by the proposed tracker, with the mean of a batch of frozen, learnable rotational symmetry parameters, in the “Translation Only” scenario.	265
8.170	(g) Demonstration of the Occlusion Handling Attention map of the “Cookie Jar” model, provided by the proposed tracker, with a per-pose regressed learnable rotational symmetry parameter, in the “Rotation Only Interaction” scenario.	265
8.171	(h) Demonstration of the Occlusion Handling Attention map of the “Cookie Jar” model, provided by the proposed tracker, with a per-pose pair regressed learnable rotational symmetry parameter, in the “Translation Only” scenario.	265
8.172	(i) Demonstration of the Occlusion Handling Attention map of the “Cookie Jar” model, provided by the proposed tracker, with a per-pose regressed learnable rotational symmetry parameter, in the “Rotation Only Interaction” scenario.	265
8.173	(j) Demonstration of the Occlusion Handling Attention map of the “Cookie Jar” model, provided by the proposed tracker, with a per-pose pair regressed learnable rotational symmetry parameter, in the “Translation Only” scenario.	265
8.174	Special case of rotational symmetry parameter estimation: 3D Translational and Rotational error metrics’ plot, for the “Rotation Only Interaction” scenario of the Cookie Jar 3D CAD model, in the case of tracker re-iteration only after a failure.	267
8.175	Special case of rotational symmetry parameter estimation: 3D Translational and Rotational error metrics’ plot, for the “Rotation Only Interaction” scenario of the Cookie Jar 3D CAD model, in the case of tracker re-iteration only after a failure.	267
8.176	(a) Demonstration of an “Observed” RGB frame of the “Full Interaction” scenario with a rendered predicted pose of the “Cookie Jar” model generated by the baseline tracker of Garon et al.[83].	269
8.177	(b) Demonstration of an “Observed” RGB frame of the “Full Interaction” scenario with a rendered predicted pose of the “Cookie Jar” model generated by the baseline tracker of Garon et al.[83] with Residual inter-layer connections and sampling/ initialization/augmentation improvements.	269
8.178	(c) Demonstration of an “Observed” RGB frame of the "Full Interaction" scenario with a rendered predicted pose of the “Cookie Jar” model generated by proposed tracker of this work.	269
8.179	(d) Demonstration of an “Observed” RGB frame of the “Full Interaction” scenario with a rendered predicted pose of the “Cookie Jar” model generated by proposed tracker, with the special case of unique rotational symmetry parameter, that remains frozen after training, of this work.	269
8.180	(e) Demonstration of an “Observed” RGB frame of the "Full Interaction" scenario with a rendered predicted pose of the “Cookie Jar” model generated by proposed tracker, with a learnable batch of rotational parameters, of this work.	269
8.181	(f) Demonstration of the Occlusion Handling Attention map provided by the proposed tracker, with a rotational parameter regression, in the “Full Interaction” scenario and tested on the “Cookiejar” 3D model.	269
8.182	(g) Demonstration of the Occlusion Handling Attention map provided by the proposed tracker, with per-pose pair optimal selection out of a nearly uniform batch of continuously rotational symmetry parameters, in the “Full Interaction” scenario and tested on the “Cookiejar” 3D model.	269
8.183	(a) Demonstration of the Occlusion Handling Attention map of the “Cookie Jar” model, provided by the proposed tracker in the “Full Interaction” scenario.	270
8.184	(b) Demonstration of the Occlusion Handling Attention map of the “Cookie Jar” model, provided by the proposed tracker in the “Full Interaction” scenario.	270

8.185(c)	Demonstration of the Occlusion Handling Attention map of the “Cookie Jar” model, provided by the proposed tracker, with a unique,frozen learnable continuous rotational symmetry parameter, in the “Full Interaction” scenario.	270
8.186(d)	Demonstration of the Occlusion Handling Attention map of the “Cookie Jar” model, provided by the proposed tracker, with a unique,frozen learnable rotational symmetry parameter, in the “Full Interaction” scenario.	270
8.187(e)	Demonstration of the Occlusion Handling Attention map of the “Cookie Jar” model, provided by the proposed tracker, with the mean of a batch of frozen learnable continuous rotational symmetry parameter, in the “Full Interaction” scenario.	270
8.188(f)	Demonstration of the Occlusion Handling Attention map of the “Cookie Jar” model, provided by the proposed tracker, with the mean of a batch of frozen, learnable rotational symmetry parameters, in the “Rotation Only” scenario.	270
8.189(g)	Demonstration of the Occlusion Handling Attention map of the “Cookie Jar” model, provided by the proposed tracker, with a per-pose regressed learnable continuous rotational symmetry parameter, in the “Full Interaction” scenario.	270
8.190(h)	Demonstration of the Occlusion Handling Attention map of the “Cookie Jar” model, provided by the proposed tracker, with a per-pose pair larnable rotational symmetry parameter, in the “Full Interaction” scenario.	270
8.191(i)	Demonstration of the Occlusion Handling Attention map of the “Cookie Jar” model, provided by the proposed tracker, with a per-pose pair optimal selection out of a nearly uniform batch of continuously rotational symmetry parameters, in the “Full Interaction” scenario.	270
8.192(j)	Demonstration of the Occlusion Handling Attention map of the proposed model, provided by the proposed tracker, with a per-pose pair optimal selection out of a nearly uniform batch of continuously rotational symmetry parameters, in the “Full Interaction” scenario.	270
8.193	Special case of rotational symmetry parameter estimation: 3D Translational and Rotational error metrics’ plot, for the “Full Interaction” scenario of the 3D CAD model, in the case of tracker re-iteration only after a failure.	272
8.194	Special case of rotational symmetry parameter estimation: 3D Translational and Rotational error metrics’ plot, for the “Full Interaction” scenario of the 3D CAD model, in the case of tracker re-iteration only after a failure.	272
8.195(a)	Demonstration of an “Observed” RGB frame of the “Hard Interaction” scenario with a rendered predicted pose of the tracking model generated by the baseline tracker of Garon et al.[83].	272
8.196(b)	Demonstration of an “Observed” RGB frame of the “Hard Interaction” scenario with a rendered predicted pose of the tracking model generated by the baseline tracker of Garon et al.[83] with Residual inter-layer connections and sampling/ initialization/augmentation improvements.	272
8.197(c)	Demonstration of an “Observed” RGB frame of the “Hard Interaction” scenario with a rendered predicted pose of the " " model generated by proposed tracker of this work.	272
8.198(d)	Demonstration of an “Observed” RGB frame of the “Hard Interaction” scenario with a rendered predicted pose of the tracking model generated by proposed tracker, with the special case of unique rotational symmetry parameter, that remains frozen after training, of this work.	274
8.199(e)	Demonstration of an “Observed” RGB frame of the "Hard Interaction” scenario with a rendered predicted pose of the " " model generated by proposed tracker, with a learnable batch of rotational parameters, of this work.	274
8.200(f)	Demonstration of the Occlusion Handling Attention map provided by the proposed tracker, with a rotational parameter regression, in the “Hard Interaction” scenario and tested on the " " 3D model.	274

8.201	(g) Demonstration of the Occlusion Handling Attention map provided by the proposed tracker, with a per-pose pair optimal selection out of a nearly uniform batch of continuously rotational symmetry parameters, in the “Hard Interaction” scenario and tested on the “” 3D model.	274
8.202	(a) Demonstration of the Foreground Extraction Attention map of the tracking model, provided by the proposed tracker in the “Hard Interaction” scenario.	274
8.203	(b) Demonstration of the Occlusion Handling Attention map of the tracking model, provided by the proposed tracker in the “Hard Interaction” scenario.	274
8.204	(c) Demonstration of the Occlusion Handling Attention map of the tracking model, provided by the proposed tracker, with a unique, frozen learnable continuous rotational symmetry parameter, in the “Hard Interaction” scenario.	274
8.205	(d) Demonstration of the Occlusion Handling Attention map of the tracking model, provided by the proposed tracker, with a unique, frozen learnable rotational symmetry parameter, in the “Hard Interaction” scenario.	274
8.206	(e) Demonstration of the Occlusion Handling Attention map of the tracking model, provided by the proposed tracker, with the mean of a batch of frozen learnable, continuous rotational symmetry parameters, in the “Hard Interaction” scenario.	275
8.207	(f) Demonstration of the Occlusion Handling Attention map of the tracking model, provided by the proposed tracker, with the mean of a batch of frozen learnable rotational symmetry parameters, in the “Hard Interaction” scenario.	275
8.208	(g) Demonstration of the Occlusion Handling Attention map of the tracking model, provided by the proposed tracker, with a per-pose regressed learnable rotational symmetry parameter, in the “Hard Interaction” scenario.	275
8.209	(h) Demonstration of the Occlusion Handling Attention map of the tracking model, provided by the proposed tracker, with a per-pose pair regressed learnable rotational symmetry parameter, in the “Hard Interaction” scenario.	275
8.210	(i) Demonstration of the Occlusion Handling Attention map of the tracking model, provided by the proposed tracker, with a per-pose pair optimal selection out of a nearly uniform batch of continuously rotational symmetry parameters, in the “Hard Interaction” scenario.	275
8.211	(j) Demonstration of the Occlusion Handling Attention map of the tracking model, provided by the proposed tracker, with a per-pose pair optimal selection out of a nearly uniform batch of continuously rotational symmetry parameters, in the “Hard Interaction” scenario.	275
8.212	Special case of rotational symmetry parameter estimation: 3D Translational and Rotational error metrics’ plot, for the “Hard Interaction” scenario of the 3D CAD model, in the case of tracker re-iteration only after a failure.	278
8.213	Special case of rotational symmetry parameter estimation: 3D Translational and Rotational error metrics’ plot, for the “Hard Interaction” scenario of the 3D CAD model, in the case of tracker re-iteration only after a failure.	278
8.214	The 6D Pose Trajectories of the “Cookie Jar” object model, in time, for the “Hard Interaction” scenario. It is evident, in both cases, from this perspective as well, that our proposed approach outperforms the SoA of Garon et al.[83] by far, as it produces smaller errors and fewer failures. Also, the effectiveness of our symmetries’ handling is ratified. All length units are in mm and all rotations in degrees.	278
8.215	Comparison of the SoA[83] (<i>light blue</i>) and our (<i>pink</i>) approaches for the “Dog” in 3 scenarios: “75% Vertical Occlusion”, “Rotation Only” and “Hard Interaction”.	279
8.216	The 6D Pose Trajectories of the “Dog” object model, in time, for the “Hard Interaction” scenario. It is evident, in both cases, from this perspective as well, that our proposed approach outperforms the SoA of Garon et al.[83] by far, as it produces smaller errors and fewer failures. All length units are in mm and all rotations in degrees.	280
8.217	Comparison of the SoA[83] (<i>light blue</i>) and our (<i>pink</i>) approaches for the “Lego” in 3 scenarios: “Translation Only”, “Full” and “Hard Interaction”.	280

8.218	The 6D Pose Trajectories of the “Lego block” object model, in time, for the “Hard Interaction” scenario. It is evident, in both cases, from this perspective as well, that our proposed approach outperforms the SoA of Garon et al.[83] by far, as it produces smaller errors and fewer failures. All length units are in mm and all rotations in degrees.	280
8.219	Comparison of the SoA [83] (<i>light blue</i>) and our (<i>pink</i>) approaches for the “Watering Can” in 3 scenarios: “75% Horizontal Occlusion”, “Rotation Only” and “Full Interaction”.	281
8.220	The 6D Pose Trajectories of the “Watering Can” object model, in time, for the “Hard Interaction” scenario. It is evident, in both cases, from this perspective as well, that our proposed approach outperforms the SoA of Garon et al.[83] by far, as it produces smaller errors and fewer failures. All length units are in mm and all rotations in degrees.	281
9.1	A graphical representation of two neighbouring segments of a cubic spline that interpolates a function $f(x)$. [93]	285
9.2	Examples of 3D continuous positional trajectories. The blue line linearly interpolates between the 4 discrete points (in meters) and the red one represents the smooth trajectories that pass all 4 of them, interpolated using the cubic spline method.	288
9.3	A graphical representation of the rotational geodesic 3D trajectory that an object follows to transform from one initial configuration (upper left corner) to a corresponding final one (lower right corner)[168]	290
9.4	The looped module in a standard RNN [44] contains a single Fully Connected layer[33]	290
9.5	Unrolling the Recurrent Neural Networks [33]	291
9.6	Internal Structure of LSTM Layer[44]	291
9.7	The LSTM’s cell state highway.[33]	292
9.8	The Forget-Gate f of LSTM. In this step, the LSTM module decides what information will be thrown away from the cell state. This is implemented by a sigmoid layer which looks at the h_{t-1} and the x_t and assigns a number of 0 or 1 on each of the units of C_{t-1} . [94]	292
9.9	The Input-Gate i of LSTM. In this step, the LSTM decides what new information will be stored. This has two parts. First, a sigmoid layer called the “input gate layer” decides which values we’ll update. Next, a tanh layer creates a vector of new candidate values, \hat{C}_t , that could be added to the state. In the next step, we’ll combine these two to create an update to the state.[94]	293
9.10	In order to update the old cell state, c_{t-1} , into the new cell state c_t , we multiply the old state by f_t , forgetting the things we decided to forget earlier. Then we add $i_t \odot c_t$. [94]	293
9.11	The Output Gate o of LSTM. In this step, the LSTM updates the C_{t-1} value with the new one C_t and its synthesized by how much information will be abandoned and what new information will be stored in the cell. [94]	293
9.12	A comparison between a LSTM module (on the left) and a Gated Recurrent Unit (GRU) (on the right) functionalities.[92]	294
9.13	A ConvLSTM cell.[164]	295
9.14	A dilated causal convolution with dilation factors $d = 1, 2, 4$ and filter size $k = 3$. [143]	296
9.15	Overview of an alteration of our CNN architecture where the first Linear layer is replaced by a recurrent type (RNN/LSTM/GRU) one with the same number of neurons.	298
9.16	Overview of an alteration of our CNN architecture where the last Fire layer is replaced by a FireLSTM one with the same number of filters.	298
9.17	Overview of an alteration of our CNN architecture where the first Linear layer is replaced by a sequence of 3 1d Convolutional layers.	299
9.18	Hard parameter sharing for multi-task learning in deep neural networks. [115]	301
9.19	Overview of an alteration of the baseline [83] CNN architecture where the first Linear layer is replaced by a recurrent type (RNN/LSTM/GRU) one with the same number of neurons.	302
9.20	Overview of an alteration of the baseline [83] CNN architecture where the last Fire layer is replaced by a FireLSTM one with the same number of filters.	303

9.21	Overview of an alteration of our CNN architecture where the last Fire layer is replaced by a FireLSTM one with the same number of filters.	303
9.22	The computational graph of the “Differential Gaussian Process Motion Planner2” (dGMPM2) where ϕ_F are user defined planning parameters that are fixed and ϕ_L are learned planning parameters [6].	303
9.23	The “VIBE” Human Pose Estimation framework. [64].	304
9.24	Coarse overview of the GAN framework for realistic 6D Pose Trajectory planning we are exploring for an extension of this work, in order to resolve the realism problem.	304
9.25	The “Observed” synthetic dense Optical Flow image produced by passing two consecutive “Observed” RGB frames, from one of the continuous trajectories of the previous Section, via the pretrained FlowNet2.	306
9.26	The “Predicted” synthetic dense Optical Flow image produced by passing two consecutive “Predicted” RGB frames, from one of the continuous trajectories of the previous Section, via the pretrained FlowNet2.	306
9.27	The synthetic dense “Augmented Reality” Optical Flow image (AF) [100] produced by passing an “Observed”-“Predicted” RGB frame pair via the pretrained FlowNet2.	306
9.28	The synthetic dense Inverse “Augmented Reality” Optical Flow image (InvAF) [100] produced by passing an “Predicted”-“Observed” RGB frame pair via the pretrained FlowNet2.	306
9.29	Overview of an alternative version of the SoA architecture of Garon et al. [83] with a single AF input.	306
9.30	Overview of an alternative version of the SoA architecture of Garon et al. [83] with two streams with Observed Optical Flow and AF inputs, correspondingly.	307
9.31	Overview of an alternative version of the SoA architecture of Garon et al. [83] with two streams with “Observed” Optical Flow and “Predicted” Optical Flow inputs, correspondingly.	307
9.32	Overview of an alternative version of the SoA architecture of Garon et al. [83] with three concatenated input sources: “Observed” Optical Flow, “Predicted” Optical Flow and AF, correspondingly.	307
9.33	Overview of an alternative version of the SoA architecture of Garon et al. [83] with three streams, with “Observed” Optical Flow, “Predicted” Optical Flow and AF inputs, correspondingly.	308
9.34	Overview of a variation of the SoA architecture of Garon et al. [83] with concatenated RGB-D-Flow types of input.	309
9.35	Overview of a variation of the SoA architecture of Garon et al. [83] with Early (concatenation-based) Fusion of the RGB-D and Flow-based feature streams.	310
9.36	Overview of a variation of the SoA architecture of Garon et al. [83] with Late (concatenation-based) Fusion of the RGB-D and Flow streams.	310
9.37	Overview of our approach of Chapter 6 where the Foreground Attention module is guided by the “Observed” Optical Flow stream.	312
9.38	Overview of our approach of Chapter 6 where the Occlusion Handling Attention module is guided by the Backward AF stream, following the idea of the paper of Wang et al.[150].	314
9.39	Overview of a modified version of our approach of Chapter 6 where the Augmented Reality Flow (AF) is warped at the feature level via a Spatial Transformer Network module.	316
9.40	Bilinear pixel interpolation for completing Optical Flow-based image warping.[155]	317
9.41	The diagram of a Spatial Transformer module[53].	318
9.42	An example of the effect a STN module has on an example image of the MNIST dataset. The image source is quantized into a discrete grid and then transformed by an Affine 2D Transformation to a target image that lies on a regular grid.[53]	318
9.43	Overview of the Hierarchical Multi-Layer AF-based Spatial Attention fusion modification the approach of Chapter 6.	321
9.44	Overview of our approach of Chapter 6 with a Multi-Layer Hierarchical Distilled AF-based Attention to every layer concatenated RGB-D-based feature map.	323

10.1	The (Normalized) Object Coordinate Space (NOCS) is a 3D space contained within a unit cube. For a given object category, canonically oriented instances are normalized to lie within the NOCS. Each (x, y, z) position in the NOCS is visualized as an RGB color tuple. [149]	326
10.2	An iteration of the Umeyama algorithm with the use of 3D points of interest.[140] . . .	326
10.4	A visualization of the 3D Spherical Harmonic component of the SE(3)-equivariant filters of Tensor Field Networks.[132].	328
10.3	An example of a Tensor Field Network presented in [132].	328
10.5	Overview of our future exploration of a modified version of the approach of Chapter 6 where we will reconstruct 3D Object Coordinates, we will process them with a pair of TFNs and we will, then, fuse them at a late stage.	329
10.6	Pixel-Adaptive Convolution.PAC modifies a standard convolution on an input by modifying the spatially invariant filter W with an adapting kernel K. The adapting kernel is constructed using either predefined or learned features f .denotes element-wise multiplication of matrices followed by a summation.Only one output channel is shown for the illustration.[123]	330
10.7	Overview of an exploration of a future variation of our approach of Chapter 6, which is based only on RGB inputs and the Depth information is distilled with the use of Pixel-Adaptive Convolutions [123] and its stream is available only during training. . . .	331
10.8	6D localization of a Varying number of instances of a Varying number of objects in a single RGB-D image, the number of instances is known. A list of instances to localize provided with the image. [45]	331
10.9	LatentFusion: An end-to-end reconstruction and rendering pipeline. This pipeline is used to perform pose estimation on unseen objects using single gradient updates in a render-and-compare fashion[97].	333
10.10	Multi-modal distributions estimated by a Learned Comparison Histogram approach of Okoron et al.(2020). These distributions are generated for the “tuna can”, “bowl” and “sugar box” object models using PoseCNN[162] featurizations. The estimator captures multiple possible viewpoint for the tuna can, while still placing most of the probability density on the correct mode. In the case of unambiguous poses, like the “sugar box”, it is still capable of producing tight uni-modal distributions.(Okorn et al.(2020)).	334
10.11	Obtained using direct pose regression[13]	334
10.12	Obtained using our multi-task learning framework.[13]	334
10.13	In [13], Bui et al., by using a multi-task manifold learning framework, are able to improve feature descriptors learned for object pose estimation. Depicted here is the feature visualization using left: PCA and right: t-SNE for five objects of the LINEMOD dataset. Pose instances are disentangled in th pose space. If such an ordered input is given to a pose regressor, less samples will be needed for an accurate and consistent prediction not only of individual poses, but of trajcetories, maybe.[13]	334
10.14	Indicative example cases of deformable, articulated and compositional objects whose pose needs to be tracked in future applications.[74]	335
10.15	A visual demonstration of the work of Hsiao et al.[139] that recognizes, accurately estimates and counters sensor’s ego-motion before proceeding to geomterical visual feature reasoning for the moving objects’ of interest.	336
10.16	Overview of an example of using 6D Pose Estimation as a prior for successful household object grasping. Grasp hypotheses are generated, based on the Object Pose estimations, and the, they are verified or discarded at a later, cascaded level [137].	337

List of Tables

1.1	Χαρακτηριστικά από πέντε αντικείμενα στα οποία δοκιμάζουμε την προσέγγισή μας. Το γεγονός πως δεν υπάρχουν δυο πανομοιότυπα αντικείμενα επαληθεύει τις ικανότητες γενίκευσης του παρακολουθητή μας.	23
1.2	3-διάστατα λάθη Μετατόπισης και Περιστροφών και συνολικές αποτυχίες παρακολούθησης σε έξι διαφορετικά σενάρια για τα τρα τελικά αντικείμενα στα οποία δοκιμάζουμε τον αλγόριθμό μας.	35
7.1	The mean and the standard deviation of both the Translation and Rotation errors of the baseline tracker of [83], when it is trained on the full dataset (200,000 samples) stated in [83] and only on 10%, just for comparison reasons in the Ablation Study.	181
7.2	The Normalised Pose Errors and the Inverses of the Coefficient of Variation of both the Translation and Rotation errors of the baseline tracker of [83], when it is trained on the full dataset (200,000 samples) stated in [83] and only on 10%, just for comparison reason in the Ablation Study.	181
7.3	The mean and the standard deviation of both the Translation and Rotation errors of the baseline tracker of [83], trained only on 10% of its overall dataset and of a modified version of it without an available pose feedback, in order to indicate the values that this feedback brings to stabilizing the tracker’s predictions.	182
7.4	The means and the standard deviations of the Normalised Pose Errors of both the Translation and Rotation errors of the baseline tracker of [83], trained only on 10% of its overall dataset and of a modified version of it without an available pose feedback, in order to indicate the values that this feedback brings to stabilizing the tracker’s predictions.	182
7.5	The mean and the standard deviations of both the Translation and Rotation errors of the baseline tracker of [83], trained only on 10% of its overall dataset, with the samples of this dataset taken from a distribution with ranges $30mm/15^\circ$ and the modified versions where the samples are taken from mean-centered Gaussian distributions with Big,Medium and Small ranges.	183
7.6	The means and the standard deviations of the Normalised Pose Errors of both the Translation and Rotation errors of the baseline tracker of [83], trained only on 10% of its overall dataset, with the samples of this dataset taken from a distribution with ranges $30mm/15^\circ$ and the modified versions where the samples are taken from mean-centered Gaussian distributions with Big,Medium and Small ranges.	183
7.7	The mean and the standard deviations both of the Translational and Rotational errors of the architecture of [83], trained only on 10% of the overall dataset, and the two modifications of it, each using only one of the two input modalities:RGB and Depth.	184
7.8	The Normalised Pose Errors both of the Translational and Rotational errors of the architecture of [83],trained only on 10% of the overall dataset, and the two modifications of it, each using only one of the two input modalities:RGB and Depth.	184
7.9	The means and the standard deviations both of the Translational and Rotational errors of the architecture of [83], trained only on 10% of the overall dataset, and its strategic modifications, where the (Uniformly) random weight initialization is replaced by Xavier Uniform (Section 3.3.10) and Kaiming He (Section 3.3.10) Uniform initialization schemes, correspondingly.	185

7.10	The means and the standard deviations of the Normalised Pose Errors both of the Translational and Rotational errors of the architecture of [83], trained only on 10% of the overall dataset, and its strategic modifications, where the (Uniformly) random weight initialization is replaced by Xavier Uniform (Section 3.3.10) and Kaiming He (Section 3.3.10) Uniform initialization schemes, correspondingly.	185
7.11	The mean and the standard deviations both of the Translational and Rotational errors of the architecture of [83], trained only on 10% of the overall dataset, and its modifications where the Data Augmentation policy proposed in [83] is further sophisticated to include the cases of Full occlusion, illumination variation and a realistic modelling of a 3D Kinect depth sensor noise.	186
7.12	The means and the standard deviations of the Normalised Pose Errors of the architecture of [83], trained only on 10% of the overall dataset, and its modifications where the Data Augmentation policy proposed in [83] is further sophisticated to include the cases of Full occlusion, illumination variation and a realistic modeling of a 3D Kinect depth sensor noise.	186
7.13	The mean and the standard deviations both of the Translational and Rotational errors of the architecture of [83], trained only on 10% of the overall dataset, and its modifications where the first two “Observed” of both the two “Observed” and “Predicted” Convolutional layers are initialized using the corresponding weights of a ResNet18[37], pretrained on ImageNet [42].	188
7.14	The mean and the standard deviations of the Normalized Pose Errors of the architecture of [83], trained only on 10% of the overall dataset, and its modifications where the first two “Observed” of both the two “Observed” and “Predicted” Convolutional layers are initialized using the corresponding weights of a ResNet18[37], pretrained on ImageNet [42]	188
7.15	A comparison between the mean and standard deviations of the translational and rotation errors among different selections of activation function. The experiments concern the main coprus of the architecture (meaning the exclusion of branches and output layers.) Promising candidates tested here are: ReLU, Leaky ReLU, PReLU, Tanh, Simgoid, Swift, SELU.	188
7.16	A comparison between the mean and standard deviations of the Normalized Pose Errors among different selections of activation function. The experiments concern the main coprus of the architecture (meaning the exclusion of branches and output layers.) Promising candidates tested here are: ReLU, Leaky ReLU, PReLU, Tanh, Simgoid, Swift, SELU.	189
7.17	The mean and the standard deviations both of the Translational and Rotational errors of the architecture of [83], trained only on 10% of the overall dataset, and its modifications where the MSE parameter loss is subtituted by a L1/LogCosh/Adaptive Barron loss alternative.	189
7.18	The mean and the standard deviations of the Normalised Pose Error measurements for the architecture of [83], trained only on 10% of the overall dataset, and its modifications where the MSE parameter loss is subtituted by a L1/LogCosh/Adaptive Barron loss alternative.	189
7.19	The 3D Translational and Rotational error metrics comparison between the architecture of [83] and its variation with the use of Residual connections.	190
7.20	The 3D Normalised Pose Error metric comparison between the architecture of [83] and its variation with the use of Residual connections.	190
7.21	The 3D Translational and Rotational error metrics comparison between the architecture of [83] and its modified versions employing different Stream Fusion strategies.	194
7.22	The Normalised Posed Error metric comparison between the architecture of [83] and its modified versions with different Stream Fusion strategies.	194
7.23	The 3D Translational and Rotational error metrics comparison between the architecture of [83] and its modified versions employing the various Foreground pixel Attention schemes described in this section.	196
7.24	The 3D Normalised Pose Error metric comparison between the architecture of [83] and its modified versions employing the various Foreground pixel Attention schemes described in this section.	196

7.25	The 3D Translational and Rotational error metrics comparison between the architecture of [83] and its modified versions employing the various Unoccluded pixel Attention schemes described in this section.	200
7.26	The 3D Normalised Pose Error metrics comparison between the architecture of [83] and its modified versions employing the various Unoccluded pixel Attention schemes described in this section.	200
7.27	The 3D Translational and Rotational error metrics comparison between the architecture of [83] and its modified versions where we employ a lone Occlusion handling Attention module and two different pairing options of it with a corresponding Foreground extraction one: one hierarchical and one in parallel, as described in this section.	203
7.28	The 3D Normalised Pose Error metrics comparison between the architecture of [83] and its modified versions where we employ a lone Occlusion handling Attention module and two different pairing options of it with a corresponding Foreground extraction one: one hierarchical and one in parallel, as described in this section.	204
7.29	Comparison of the 3D rotational error metric as the proposed rotational loss scheme is gradually developed by adding one improvement after the other. Note that, in the first row, we include an experimentation on the sole 3D translation estimation, in order to understand which of the two components is easier to train to and what is the effect of coupling them to the accuracy of each one, independently.	205
7.30	Comparison of the Normalized Pose Error metric as the proposed rotational loss scheme is gradually developed by adding one improvement after the other. Note that, in the first row, we include an experimentation on the sole 3D translation estimation, in order to understand which of the two components is easier to train to and what is the effect of coupling them to the accuracy of each one, independently.	205
8.1	Characteristics of the five objects we test our approach on. The fact that there are no two identical items validates the generalization capabilities of our tracker.	207
8.2	Quantitative overview of the impact of the gradual inclusion of every consecutive design amelioration in the proposed neural architecture.	210
8.3	Comparison of the 3D Translational and Rotational errors when evaluating different multi-task weighting schemes that leverage the main tracking and the auxiliary attention losses.	210
8.4	Comparison of the Normalized Pose Errors (N.P.E.) when evaluating different multi-task weighting schemes that leverage the main tracking and the auxiliary attention losses.	211
8.5	The 3D Translational and Rotational error metrics' comparison between the various weight warm-up schemes that alter the baseline architecture of Garon et al. [83]	211
8.6	The Normalized Pose Error (N.P.E.) metrics' comparison between the various weight warm-up schemes that alter the baseline architecture of Garon et al. [83]	211
8.7	3D Translational and Rotational Pose Errors and Fail count of the "Dragon" model, for the "Stability near" for the baseline approach of Garon et al. [83], without/with sampling/initialization/augmentation improvements we propose and the approach of Chapter 6.	214
8.8	3D Translational and Rotational Pose Errors and Fail count of the "Dragon" model, for the "Stability far" for the baseline approach of Garon et al. [83], without/with sampling/initialization/augmentation improvements we propose and the approach of Chapter 6.	216
8.9	3D Translational and Rotational Pose Errors and Fail count of the "Dragon" model, for the "75% Vertical Occlusion" scenario for the baseline approach of Garon et al. [83], without/with sampling/initialization/augmentation improvements we propose and the approach of Chapter 6.	219
8.10	3D Translational and Rotational Pose Errors and Fail count of the "Dragon" model, for the "75% Horizontal Occlusion" scenario for the baseline approach of Garon et al. [83], without/with sampling/initialization/augmentation improvements we propose and the approach of Chapter 6.	221

8.11	3D Translational and Rotational Pose Errors and Fail count of the “Dragon” model, for the “Translation Only Interaction” scenario for the baseline approach of Garon et al. [83], without/with sampling/initialization/augmentation improvements we propose and the approach of Chapter 6.	224
8.12	3D Translational and Rotational Pose Errors and Fail count of the “Dragon” model, for the “Rotation Only Interaction” scenario for the baseline approach of Garon et al. [83], without/with sampling/initialization/augmentation improvements we propose and the approach of Chapter 6.	226
8.13	3D Translational and Rotational Pose Errors and Fail count of the “Dragon” model, for the “Full Interaction” scenario for the baseline approach of Garon et al. [83], without/with sampling/initialization/augmentation improvements we propose and the approach of Chapter 6.	229
8.14	3D Translational and Rotational Pose Errors and Fail count of the “Dragon” model, for the “Hard Interaction” scenario for the baseline approach of Garon et al. [83], without/with sampling/initialization/augmentation improvements we propose and the approach of Chapter 6.	232
8.15	A random subset of the overall batch of $B_2 = 64$ learnable rotational symmetry parameters.	236
8.16	3D Translational and Rotational Pose Errors and Fail count of the “Cookie Jar” model, for the “Stability near” scenario, for the baseline approach of Garon et al. [83], without/with sampling/ initialization/augmentation improvements we propose and the approach of Chapter 6. and its symmetry handling alternatives.	238
8.17	3D Translational and Rotational Pose Errors and Fail count of the “Cookie Jar” model, for the “Stability far” for the baseline approach of Garon et al. [83], without/with sampling/initialization/augmentation improvements we propose and the approach of Chapter 6 and its symmetry handling variations.	243
8.18	3D Translational and Rotational Pose Errors and Fail count of the “Cookie Jar” model, for the “75% Horizontal Occlusion” scenario for the baseline approach of Garon et al. [83], without/with sampling/initialization/augmentation improvements we propose and the approach of Chapter 6.	248
8.19	3D Translational and Rotational Pose Errors and Fail count of the “Cookie Jar” model, for the “75% Vertical Occlusion” scenario for the baseline approach of Garon et al. [83], without/with sampling/initialization/augmentation improvements we propose and the approach of Chapter 6 and its symmetry handling variations.	253
8.20	3D Translational and Rotational Pose Errors and Fail count of the “Cookie Jar” model, for the "Translation Only Interaction” scenario for the baseline approach of Garon et al. [83], without/with sampling/initialization/augmentation improvements we propose and the approach of Chapter 6 and its symmetric handling variations.	258
8.21	3D Translational and Rotational Pose Errors and Fail count of the “Cookie Jar” model, for the “Rotation Only Interaction” scenario for the baseline approach of Garon et al. [83], without/with sampling/initialization/augmentation improvements we propose and the approach of Chapter 6.	263
8.22	Angular distance between the predicted and ground truth z-rotational components (in degrees) of the “Cookie Jar” model, for the “Rotation-Only Interaction” scenario for the baseline approach of Garon et al. [83], the approach of Chapter 6 and its symmetry handling variations.	267
8.23	3D Translational and Rotational Pose Errors and Fail count of the “Cookie Jar” model, for the “Full Interaction” scenario for the baseline approach of Garon et al. [83], without/with sampling/ initialization/augmentation improvements we propose and the approach of Chapter 6 and its symmetry handling variations.	268
8.24	3D Translational and Rotational Pose Errors and Fail count of the tracking model, for the “Hard Interaction” scenario for the baseline approach of Garon et al. [83], without/with sampling/ initialization/augmentation improvements we propose and the approach of Chapter 6 and its symmetry handling variations.	273

8.25	3D Translational and Rotational errors and overall tracking failures in six different scenarios for the final three employed objects.	282
9.1	The 3D Translational and Rotational error metrics comparison for the architecture of [83], when this CNN design is trained on the Discretely/Continuously sampled training dataset, shuffled in a Discrete/Continuous manner and without/with a-priori blending the same occluder 3D continuous trajectory for every batch iteration.	299
9.2	The Normalized Pose Error metrics comparison for the architecture of [83], when this CNN design is trained on the Discretely/Continuously sampled training dataset, shuffled in a Discrete/Continuous manner and without/with a-priori blending the same occluder 3D continuous trajectory for every batch iteration.	300
9.3	The 3D Translational and Rotational error metrics comparison between the architecture of [83] and its variations with the use of single-layered recurrent modules (RNN / LSTM / GRU / FireLSTM / TCNs) in place of the first Linear/last Fire layer.	302
9.4	The 3D Translational and Rotational error metrics comparison between the architecture of [83] and its variations with the use of single-layered recurrent modules (RNN / LSTM / GRU / FireLSTM / TCNs) in place of the first Linear/last Fire layer.	302
9.5	The Translational and Rotational Error comparison of the variations of the SoA architecture of Garon et al. [83], with different Optical Flow inputs, instead of RGB-D ones.	308
9.6	The Normalized Pose Error comparison of the variations of the SoA architecture of Garon et al. [83], with different Optical Flow inputs, instead of RGB-D ones.	308
9.7	The Translational and Rotation Error metric comparison of the SoA RGB-D architecture of Garon et al. [83] with its variations of Fusing the Flow-based information at different layers of the Network.	310
9.8	The Normalized Pose Error metric comparison of the SoA RGB-D architecture of Garon et al. [83] with its variations of Fusing the Flow-based information at different layers of the Network.	311
9.9	The Translational and Rotation Error metric comparison between our approach of Chapter 6 and its variations where the Clutter Handling Spatial Attention module is guided by an “Observed” Optical Flow input type.	313
9.10	The Normalized Pose Error metric comparison between our approach of Chapter 6 and its variations where the Clutter Handling Spatial Attention module is guided by an “Observed” Optical Flow input type.	313
9.11	The Translational and Rotation Error metric comparison between our approach of Chapter 6 and its variations where each Spatial Attention module is guided by a Backward Augmented Reality Flow (AF) input type.	315
9.12	The Normalized Pose Error metric comparison between our approach of Chapter 6 and its variations where each Spatial Attention module is guided by a Backward Augmented Reality Flow (AF) input type.	315
9.13	The Translational and Rotation Error metric comparison between our approach of Chapter 6 and its variation where Spatial Transformer AF-based feature warping is used for early fusion between the RGB-D and Flow modalities.	320
9.14	The Normalized Pose Error metric comparison between our approach of Chapter 6 and its variation where Spatial Transformer AF-based feature warping is used for early fusion between the RGB-D and Flow modalities.	320
9.15	3D Translational and Rotational Error metric comparison between the baseline architecture of Garon et al. [83], our improved approach of Chapter 6 and its modified version with an incorporated Multi-Layer Hierarchical AF-based Spatial Attention fusion scheme.	321
9.16	Normalized Pose Error metric comparison between the baseline architecture of Garon et al. [83], our improved approach of Chapter 6 and its modified version with an incorporated Multi-Layer Hierarchical AF-based Spatial Attention fusion scheme.v	321

9.17	3D Translational and Rotational Error metric comparison between the baseline architecture of Garon et al. [83], our improved approach of Chapter 6 and its modified version with an incorporated Distilled Multi-Layer Hierarchical AF-based Spatial Attention fusion scheme.	323
9.18	Normalized Pose Error metric comparison between the baseline architecture of Garon et al. [83], our improved approach of Chapter 6 and its modified version with an incorporated Distilled Multi-Layer Hierarchical AF-based Spatial Attention fusion scheme.v	323

Chapter 2

Introduction

"They were given the choice of becoming kings or the kings' messengers. As is the way with children, they all wanted to be messengers. That is why there are only messengers, racing through the world and, since there are no kings, calling out to each other the messages that have now become meaningless. They would gladly put an end to their miserable life, but they do not dare to do so because of their oath of loyalty."

Franz Kafka, "The Blue Octavo Notebooks"

This Diploma thesis belongs in the intersection of the scientific fields of Computer Vision, Computer Graphics and Machine Learning.

2.1 Computer Vision

Computer Vision is the scientific domain that incorporates extracting, processing and analyzing attributes hidden in digital images or a sequence of them, called video, intending to acquire an understanding of a high-level concept and produce numerical or symbolic information, e.g. in the form of decisions. In order to disentangle symbolic information from image data, it uses models constructed with the aid of geometry, physics, statistics, and learning theory. It is intensely motivated by the function both of the human and the animals' visual systems as it consists an engineering effort to mimic such mechanisms in the form of automated algorithms trying to reach -or even surpass- human-level accuracy. The full operation of the field can be further divided into sub-tasks that living beings achieve in a unified way such as instance classification, scene understanding, pose estimation, action recognition, text comprehension and translation, among others.

2.2 Computer Graphics

Computer graphics is the discipline of generating images with the aid of computers. Using them, we may display art and image data effectively and meaningfully to the consumer. Today, computer graphics is a core technology in digital photography, film, video games, cell phone and computer displays, and many specialized applications. They are tightly linked with Computer Vision as they have the ability to feed its algorithms with simulated data, transform statistical distributions across domains and demonstrate its advances.

2.3 Machine Learning

Machine Learning (ML) is defined as the sub-domain of Artificial Intelligence that uses algorithms and computational statistics to learn from data. Such approaches are particularly useful in problems that are too perceptually complex to be modelled with extensive lists of rules, or for problems that are too computationally hard to be solved analytically and thus an approximation of a solution based on patterns extracted from large amounts of data is satisfactory enough. Machine Learning algorithms can be categorized based on the level of annotation existed during the training process, with the algorithms classified as follows:

- **Supervised Learning:** Supervised learning algorithms teach the model to produce outputs derived from a specific distribution using data with known labels as examples and calculating the error between the predicted and the ground-truth results. The two best known applications of them are **classification** and **regression** problems. In the former, there is the need to correctly categorize unlabeled test data in a class belonging to a predefined set. So, it is profound that the target distribution is discrete. To this end, the model is firstly trained on the same task on a well-balanced training set with elements derived from the same distribution as the test ones. On the other hand, the latter is described as the effort to determine a relationship among many different variables, with the target distribution here being continuous. The computer program is asked to predict a numerical value given some input. Mathematically, the learning algorithm is asked to output a function $f: \mathbb{R}^n \rightarrow \mathbb{R}^m$. The only difference between the two problems comprises in the output type.
- **Semi-Supervised:** Semi-supervised Learning techniques mix small quantities of labeled data with a large amount of unlabeled ones. They are of practical value because data annotation for a learning problem often requires human agents with domain expertise, many work hours of a populated staff or the realization of a physical experiment with the cost associated with the labeling process resulting in the acquisition to be uneconomical. On the other hand, acquisition of unlabeled data is relatively inexpensive. With these small quantities of data with ground truth labels available, we have a disproportionally positive impact on the algorithms' performance guiding the update process for the rest of the samples as well.
- **Unsupervised Learning:** Unsupervised Learning discovers patterns in a dataset without pre-existing labels. Such algorithms try to optimize a criterion constructed by a model's outputs with the intention to provide self-organization in the data structure. Famous examples of this sort of techniques are Clustering, Dimensionality Reduction and Probability Density Estimation. The first one segments data with shared functional dependencies by identifying commonalities in their attributes and reacting based on the presence or absence of such commonalities in each new entry. The second one transforms high-level data representations to feature spaces of lower dimension for quantization or visualization purposes, while the last aims to infer an a-priori data probability distribution $\mathbb{P}_{\mathbb{X}}(x)$.
- **Reinforcement Learning:** Reinforcement Learning comprises of algorithms that do not teach the model using labels on a per-sample base, but instead guide an agent to optimize a more general, complex objective over many steps along one or more dimensions, by optimizing its parameters. Reinforcing the learning procedure means providing a suitable reward for the model when it takes a right decision during interaction with its environment and a corresponding punishment in the opposite case.

2.4 Deep Learning

Deep Learning is a bit like smoking,
you know that it's wrong but you do it
anyway because you want to look cool.

Carl Henrik Ek

Machine Learning algorithms are trained to recognize patterns in the chaos of data complexity when they are provided with a sufficient representation of raw inputs. Such a description of the input samples' properties is genuinely hard as they are subject to a high degree of variation (e.g. for digital images due to illumination change during days and nights, or for voice data because of the voice pitch difference between different users). So, Deep Learning, a subcategory of the Machine Learning domain that is based on Neural Networks, attempts to bridge the representation learning gap by providing complex features that are expressed in terms of other, simpler representations and thus, they are robust to input variations. For example, a Deep Neural Network describes an image containing a car as a hierarchy of its attributes, starting from edges and corners found in the image, in the first layers, and combining them in the following ones to create higher-level features corresponding to wheels, lights and the car skeleton.

2.5 Visual Object Tracking

One of the primitive tasks that Computer Vision research has dealt with is figuring out the location of one or multiple objects present in a frame during a video sequence. People spend most of their time interacting with objects such as tools, toys, pieces of clothing or accessories and thus, it rises as crucial for a Computer Vision algorithm to understand where they are placed in an image. A common example is that of a football match, where the importance of the players' poses or actions comes secondary for the viewer in comparison to where the ball lies at every given moment.



Figure 2.1: The result of a famous 2D Object Tracking Algorithm called “GOTURN” in the soccer example.[41]

2D Object Tracking

The most common approach Computer Vision uses to achieve so is the estimation of a bounding box surrounding the object, which includes the x, y location of its center: c_x, c_y and its size, with the use of the width w and height h of the bounding box boundaries, as seen in fig.2.1.

The staple algorithm of the region, for many years, has been the Lucas-Kanade-Tomasi(KLT) Tracker [124] which uses the current video frame features of interest and the sparse Optical Flow [47] (see Section 3.2.3) estimation from the corresponding features of the previous frame to estimate the location and size of the bounding box. Yet, many years have passed since its development with the recent success of Convolutional Neural Networks (see Section 3.3.5) giving rise to more effective and computationally efficient frameworks such as YOLO [110], SSD [77], Siamese Networks [65] or GOTURN [41] that track the necessary parameters using only the RGB current (and/or previous) frame.

6D Object Tracking

Nevertheless, extracting the 2D bounding box information is rather incomplete, as, firstly, the bounding box size, implying its scale, is just a crude approximation of the object's position in the axis perpendicular to the image plane, and, secondly, it is invariant to the object's orientation. So, usually scene understanding tasks require the more complete description of the full 3D object's Pose: both its position in the Camera Space and its orientation. Simultaneously, the development of cheap and reliable depth sensors like the Microsoft Kinect, presented in fig.2.2, has provided the extra information source of the object's 3D shape, expressed in a Depth map, that greatly facilitates the transition from the 2D Image Space to the 3D Camera coordinated one.



Figure 2.2: The second and most recent version of the Microsoft Kinect RGB-D sensor.

Thus, the whole research area has seen a hot trending the few previous years with every method surpassing the State-of-the-Art, every one or two publications down the way. Some ontensive examples, are shown in the following figures with the researchers proposing a wide variety of algorithmic techniques ranging from per frame Pose Detection (fig.2.3) to Temporal Tracking (fig.2.4), from online Energy minimization methods (fig.2.5) to learning based ones (fig.2.3), with (fig.2.4) or without (fig.2.5) the need for prior Optical Flow estimation and for single (fig.2.3) or multiple (fig.2.4) objects of interest.

Similarly to the 2D bounding box estimation case, the 3D pose is recognised by the 6 (c_x, c_y, c_z, h, w, d (for depth)) parameters of the 3D bounding box, but, more frequently, by the 12 pose parameters of a Rigid Transformation from the Object to the Camera Coordinate Frame (see Section 2.10.2).



Figure 2.3: Per-frame Object 6D Pose Detection based on Learning to Regress Object Coordinates. [8]

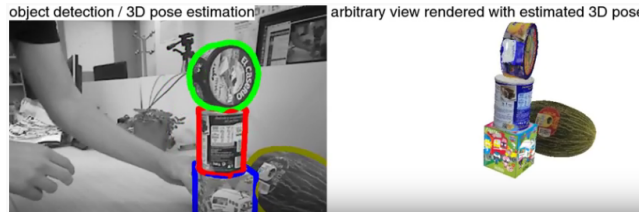


Figure 2.4: Real-time Model-based Tracking up to 100 6D Object Poses at the same time with a model-based Energy minimization framework leveraging RGB-D image and Optical Flow information. [100]

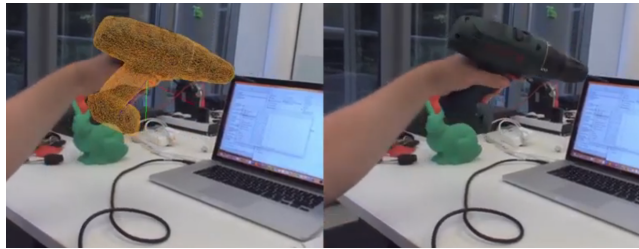


Figure 2.5: Real-time Model-based Single Object 6D Pose Tracking using an online Energy minimization scheme, only with RGB image information source. [134]

2.6 Applications of 6D Object Pose Tracking

In the research domain, Hadfield et al.[34] used a 6D Pose Tracker as the principal component of an Assembly Guidance framework for educational purposes of children with special abilities that interact with a Teacher-robot (fig.2.6).

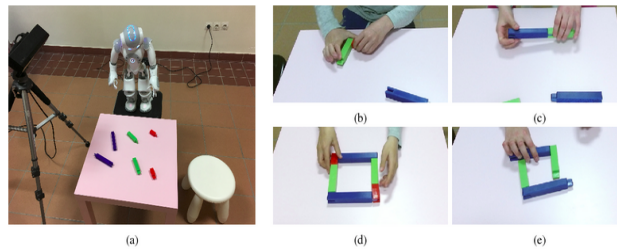


Figure 2.6: An interactive Object Assembly framework, occasioned by the Baby-Robot project realized by the CVSP Lab Team, that aids children with special abilities to assemble constructions made of Lego blocks.[34]

However, tracking the 6-D Object Pose through time is not only a matter of scientific research, but of great market value as well, with a wide variety of commercial technologies utilizing the pose information as the mid-level of a broader application, with the most prominent fields being: Augmented Reality, Virtual Assistants, Robotic Manipulation and Autonomous Driving.

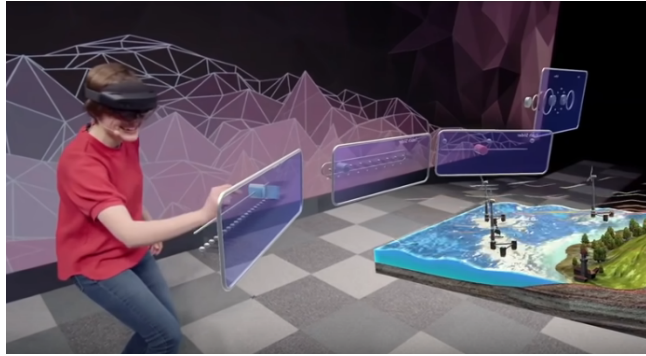


Figure 2.7: Microsoft Hololens

Augmented Reality (AR) applications have seen an abrupt popularity in the last 5 years promising to change our every-day encounters. Recently, Microsoft released the newest version of “Hololens”: a multi-sensory headset that provides advanced optical and holographic processing that blends seamlessly artificial scene parts with its real world environment, establishing Mixed (Augmented) Reality and providing the opportunity to the user to track and interact with objects not present in the physical world or with graphically altered versions of them. Hololens is equipped with a variety of (central and peripheral) RGB HD and Depth cameras, specialized speakers that simulate sound from anywhere in the room, several microphones, an ambient light sensor and a custom “Holographic Processing Unit” that they claim has more processing power than the average laptop, all of them aiming to offer a user experience directly derived from science fiction novels (see fig.2.7 from a recent live demonstration) and are expected to boost scientific progress in Multimodal Scene Understanding and Computer Graphics realism.

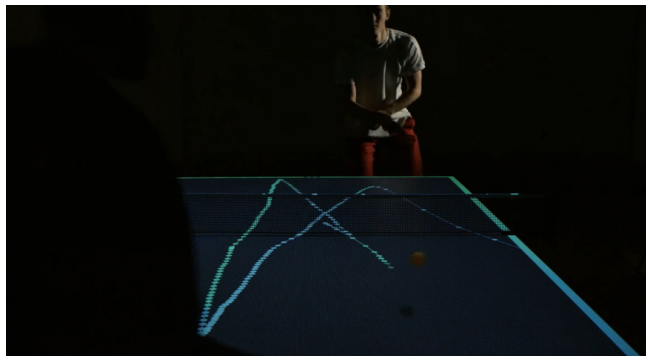


Figure 2.8: A training Ping-Pong table, launched by a German startup company that interactively analyzes player’s hit history and suggests the next move for each of them.

A second impressive commercial application of AR recently launched is that of an Interactive Training Ping-Pong Table (fig.2.8) that tracks and visualizes the ball’s 3D trajectory history for the set, records the opponent’s favourite and weak spots and suggests to each user an optimal target 2D region for their next hit.

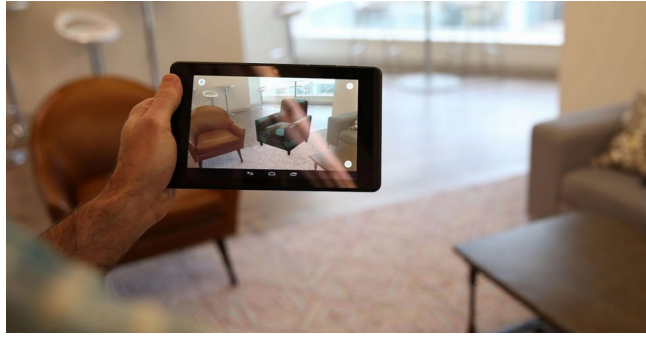


Figure 2.9: A mobile application recently launched by IKEA. The user's smartphone is headed towards objects present in the scene, to capture their full pose. Then, pieces of IKEA furniture are rendered on these objects, at the exact same pose, to provide a preview of the customer's potential market options.

Last but not least, in 2017, IKEA published an AR mobile application (see fig.2.9) that estimates the 6D pose of an object the user has his/her smart phone camera oriented to and superimposes the furniture of the user's choice at its place. In this way, the customer may study the esthetic impact of his/her future purchase and buy the furniture online without ever needing to visit the IKEA store or return the product.

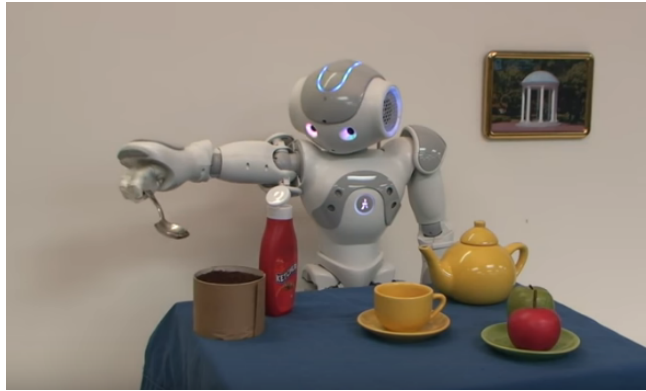


Figure 2.10: A Robotic manipulation example.

Furthermore, besides AR, improved Robotic manipulation is a natural by-product of accurate and robust 6D Object Tracking frameworks allowing control schemes to target not only steady, but also moving objects as well, for Grasping and further handling (fig.2.10).



Figure 2.11: The augmented visual environment of a contemporary Autonomous Driven car tested today.

Finally, the region of Object Pose Estimation and Tracking has the ambition to be part of the Autonomous Driven cars of the future (fig.2.11) as there are more than few real-life cases where the

driver has to be aware of the positions and movements of bigger and smaller objects present on the streets. Besides, it is sure that similar algorithms are employed to solve the dual problem of the Pose Estimation of the car camera, a fundamental component of the vehicle’s self localization.

2.7 Performance Requirements

The optimal 6D Pose Temporal Tracker is expected to satisfy a variety of different and frequently clashing criteria since producing accurate estimations at every 3D position and velocity change possible is necessary but far from sufficient. It also needs to be robust to sensor noise, estimation jitter and initial detector choice.

Furthermore, it is required to counter drift produced by small tracking errors that accumulate over time resulting into rare fails. From the time complexity perspective, it is required to produce the pose estimates in real-time close to or faster than 30 frames per second and, lastly, it would be ideal if it would be compatible with any device that would be installed in. That needs the tracker to be modular so that any component would be replaced properly in the future and rely only on RGB images as most low power portable devices (e.g. smartphones) do not offer the luxury of depth sensors like Kinect does. The approach proposed fulfills most of the above and we schedule to face the rest in the future in specific ways analyzed in the following sections.

2.8 Proposed Approach - Motivation for Deep Learning Employment

The careful reader would immediately imagine 6-DOF Object Pose Tracking as an online task, where learning of the object features, trajectory characteristics and interactions with its environment would be computed without any prior expectation at the arrival of each input frame. However, the exceptional results Machine Learning-based techniques have recently presented motivate us to transfer some of the computational burden of the tracking procedure to a learning backend which is not time-critical, leaving only the inference stage to be executed online. More explicitly, a training set is created with the object put into predefined positions of one frame or moving into specific trajectories of a video clip and a Neural Network’s parameters gain prior experience by being trained on it. Therefore, the success of the tracking online process, in terms of accuracy and robustness, depends only on the quantity and the quality (in terms of similarity with the real-life scenario and their intra-variability) of the training image samples and the way the Network components are modelled to face the challenging scenarios present in the training set. To this end, learning-based algorithms allow trackers to infer poses faster and, if trained properly, not to overfit on specific conditions present in the test scenarios.

2.9 Challenges of the problem

After a careful analysis of the relevant literature, we may observe that the task at hand faces the following challenges that should be taken into consideration when designing our training scheme:

- *2D Source -3D Target mismatches:* We are required to estimate the 3D Object Pose while being relied only on 2D image sources (RGB-D). Depth maps, when available, may give information about the 3D Object structure, but are not as accurate as direct, full Point Cloud/3D Mesh/Occupancy map representations, due to viewpoint, noise and truncation effects, so they are listed as a 2D source. This discrepancy may appear in the feature or in the output level:
 - *2D-3D Feature mismatch:* Classical 2D Convolutional Neural Networks may have shown an admirable success in 2D tasks, but offer only 2D translational equivariance w.r.t. the input and lack the full 6D equivariance needed for an 1-1 correspondence between the object pose depicted in the image source and the convolutional features produced by it.

- *2D-3D Output representation mismatch:*
The fact that the object's 3D shape structure leaves its mark in its depiction in the image level makes it challenging for the tracking algorithm to estimate the object's pose at its proper scale, as part of this information is distorted by the projection of the shape to the image plane.
- *Background Clutter - Color Noise:* The appearance component of the feature extraction procedure is highly sensitive to similarly-looking or grandstanding elements of its environment that may distract the model from the object's of interest details.
- *Occlusion Handling:* It can be separated in:
 - *Static Occlusion Handling:* when other static objects hide from the camera view some or the whole of the object of interest.
 - *Dynamic Occlusion Handling:* when the object is occluded during manipulation with a user or robot due to their hand(s)/joint(s') movements.
- *Motion blur:* Abrupt quick object movements or camera focus changes reduce the image quality making it more difficult to distinguish the appearance details necessary.
- *Appearance change due to pose variation:* As objects alter their pose in time, the amount and quality of visible points of interest varies, which causes fluctuation in the viewer's pose understanding.
- *Sensor noise modeling:* The probabilistic noise model, that we augment our samples with, must follow a distribution that should seem realistic to the viewer.
- *Illumination conditions:* Different illumination conditions between the train cases and the real-time inference one, as well, as those that change across time, due to lightning source alteration or reflections from the other scene surfaces, hold back the tracker's accuracy.
- *Pose Representation ambiguities:*
 - *Mathematical representation of pose components:*
The choice of the object's translational and rotational parameters dictates the existence of possible ambiguities and/or abundances in the pose representation and facilitates/hinders the tracker's capabilities either via an online or a learning-based perspective.
 - *Symmetries:*
As it will thoroughly explained in later sections, symmetries (either in the object's geometrical characteristics or in the way the camera captions it) may be the reason for an algorithm to output an estimation in which it has selected the erroneous between two or more equivalent representations of one of the pose components. It is a serious source of ambiguity as the errors, artificially induced in the pose estimate, are usually large, due to the, frequently, maximum distance of the aforementioned equivalent representations.
 - *Geometry/Shape-induced symmetries:*
This, first category of symmetries lie in the geometry of the object's 3D shape, are innate in its construction and neglecting their existence may cause, otherwise reasonable approaches, to fail completely or increase the pose error by large margins.
 - *Viewpoint-induced symmetries:*
This, second kind of symmetries is generated due to the image source selection (i.e. it is absent in 3D representations (e.g. point clouds etc.)), is external to the object's shape/appearance and has to do with possible dualities in the camera viewpoint distribution and/or occlusions of one of many identical substructures of the object.
- *Modeling Temporal Continuity:* Object Pose Tracking is an innate continuous task, w.r.t. the time component, so the demand of learning continuous pose trajectories rises naturally. Such approaches have shown positive results in 2D tracking applications, but transferring them to the 6D pose problem is a much more complicated procedure. That scepticism is common throughout

a variety of temporal Computer Vision problems who often do not choose to carry the burden of proper temporal modeling and compromise with per-frame options (either in online or learning-based proposed solutions).

In the implementation section, we will discuss how we explicitly handle each of these cases.

2.10 The full Camera Projection Matrix

The process of encapsulating an object present in a scene into an image requires the projection of its 3D points to the corresponding plane via a matrix multiplication. The Projection Matrix required can be dissected into the **Intrinsic Camera parameter matrix** that dictates how the object 3D points (considered w.r.t. the Camera Coordinate frame) will be projected to the 2D Image plane and its **Extrinsic Camera Parameter** counterpart i.e. its Pose.

$$\mathbf{x}^{(I)} = \mathbb{P}_{(O)}^{(I)} \mathbb{X}^{(O)} = K \mathbb{T}_{(O)}^{(C)} \mathbb{X}^{(O)}, \quad (2.1)$$

with homogenous pixel coordinates $\mathbf{x}^{(I)} = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \in \mathbb{R}^3$ and homogenous object points $\mathbb{X}^{(O)} = \begin{bmatrix} x_1^{(O)} \\ x_2^{(O)} \\ x_3^{(O)} \\ 1 \end{bmatrix} \in \mathbb{R}^4$.

The fig.2.12 gives a graphical representation of the full model. In our setting, K will be known and constant and our main goal is estimating $\mathbb{T}_{(O)}^{(C)}$ at every timestep t.

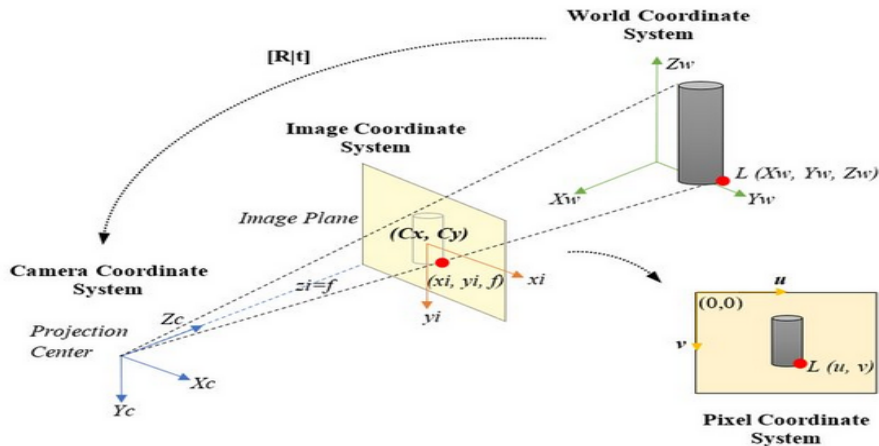


Figure 2.12: The full image registration diagram. The object coordinates are first transformed by the application of the Pose homogenous matrix from the Object to the Camera coordinate frame (consisted of a sequence of transforms $(O) \rightarrow (W)$, $(W) \rightarrow (C)$) and, sequentially, each 3D point w.r.t. (C) is projected using the properties of the Camera parameter matrix K to the image plane filling the suitable pixel values.

2.10.1 Dissecting the Camera Projection Matrix: The Pinhole Camera Calibration Matrix/ Intrinsic Parameters

Intrinsic parameters describe geometric properties of the camera. They form a 3x4 matrix K:

$$K = \begin{bmatrix} f_x & s & c_x & 0 \\ 0 & f_y & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \quad (2.2)$$

with each of them having a different meaning:

- **Focal lengths f_x, f_y :** The focal length is the distance between the pinhole and the film (a.k.a. image plane), one for each x,y-direction. In a true pinhole camera, both f_x and f_y have the same value f . Otherwise, different such values indicate non-rectangular pixel shapes.
- **Pinhole Offsets x_0, y_0 :** The camera’s “principal axis” is the line perpendicular to the image plane that passes through the pinhole. Its intersection with the image plane is referred to as the “principal point”. The “principal point offset” is the location of the principal point relative to the film’s origin.
- **Axis skew s :** Axis skew measures the shear distortion in the projected image.

2.10.2 *Dissecting the full Camera Projection Matrix: Definition of the problem of 6D Object Pose Estimation and Tracking - Pose Representation / Extrinsic Parameters*

Physically, the Rigid Object Pose is defined as the combination of position and orientation of the object w.r.t. a 3D reference frame. In literature, the problem of Pose Estimation is usually formalized mathematically as the attempt to predict a rigid transformation \mathbb{T} , from the Object Coordinate Frame (O) to the Camera Coordinate Frame (C) that lies in the Special Euclidean Group $SE(3)$. $SE(3)$ is made up of a rotation component, represented with a square matrix $R \in SO(3)$, and a translation vector $\mathbf{t} \in \mathbb{R}^3$. So, $SE(3) = SO(3) \times \mathbb{R}^3$, has the homogeneous representation:

$$SE(3) = \{ \mathbb{T} = \begin{bmatrix} R & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \in \mathbb{R}^{4 \times 4} \mid R \in SO(3), \mathbf{t} \in \mathbb{R}^3 \}, \quad (2.3)$$

$SE(3)$ has the structure both of a differentiable Riemannian manifold and an algebraic group, two properties that classify it as a Lie Group.

The set of rotations has also a group structure, known as the group of Special Orthogonal Matrices in 3D: $SO(3)$. It is a smooth, compact Lie group, equipped with the operations of the tangent $so(3)$ Lie Algebra and a closed manifold of $\mathbb{R}^{3 \times 3}$, (which means that the composition of two rotations is also a rotation).

$$SO(3) = \{ R \in \mathbb{R}^{3 \times 3} \mid RR^T = R^T R = \mathbb{I}_3, \det(R) = 1 \}. \quad (2.4)$$

$RR^T = R^T R = I$ means that the column vectors of R are mutually orthogonal and with unit norm, while $\det(R) = 1$ suggests a right-handed coordinate system.

A first look would lead us to think that an object’s pose is characterized by 12 parameters. However, only 6 of them are independent from each other, thus creating its irreducible 6-DOF representation. They can be physically interpreted as a corresponding object’s translation and rotation component $t_x, t_y, t_z, r_x, r_y, r_z$ along each of the x,y,z-axis. When the Pose $\mathbb{T}_{(O)}^{(C)}$ is applied to a 3-D point $\mathbb{X}^{(O)} = [x_1^{(O)}, x_2^{(O)}, x_3^{(O)}, 1]^T$ which belongs to the Object Pointset \mathcal{S} in the Homogenous Space \mathbb{R}^4 , that point is transformed to its corresponding $\mathbb{X}^{(C)} = [x_1^{(C)}, x_2^{(C)}, x_3^{(C)}, 1]^T$:

$$\mathbb{X}^{(C)} = \mathbb{T}_{(O)}^{(C)} \mathbb{X}^{(O)} = \begin{bmatrix} R & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{bmatrix} x_1^{(O)} \\ x_2^{(O)} \\ x_3^{(O)} \\ 1 \end{bmatrix} \quad (2.5)$$

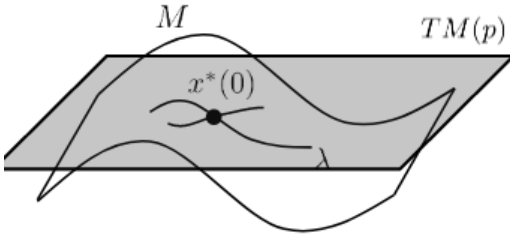


Figure 2.13: A Riemannian Manifold. [86]

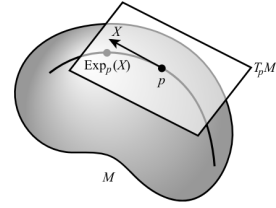


Figure 2.14: A Riemannian Exponential Map [23].

A **Manifold** of \mathbb{R}^n is a topological space that locally looks like an open subset of \mathbb{R}^n . Each point of an n-dimensional manifold has a neighbourhood that is homeomorphic to a Euclidean space of dimension n.

A **Riemannian** Manifold of \mathbb{R}^n is a real, smooth manifold M equipped with a positive-definite inner product g_p on the **Tangent Space** T_pM at each point p of M. The family g_p of inner products is called a **Riemannian Metric Tensor**. Physically, a (Riemannian) Metric Tensor renders the inner product of vectors that belong in the Tangent Space T_pM . When the basis of this space is orthonormal, the Metric Tensor is the Identity Matrix. However, when this is not the case, or/and the basis varies with time (for example in the case of a Riemannian Manifold, where the Tangent Space changes at every point of the Manifold), then g_p has different elements or/and varies in spacetime according to the parameters that are used to define T_pM . For example, if the manifold is described by a surface $g=g(u,v)$, where u,v, a parameter pair in T_pM , then $g_p = g_p(u, v)$.

The Tangent Space $T_I M$ is a linear vector space, which means that one could select its basis vectors and its inner product rule. One of the most common such inner products is the **Hilbert-Schmidt** inner product:

$$\langle A, B \rangle_I = \text{tr}(A^T B), \quad (2.6)$$

for every $A, B \in T_pM$. Then, T_pM is a normed space, equipped with the Hilbert-Schmidt norm:

$$\|A\|_{HS} = \sqrt{\text{tr}(A^* A)} = \sqrt{\text{tr}(A^T A)}. \quad (2.7)$$

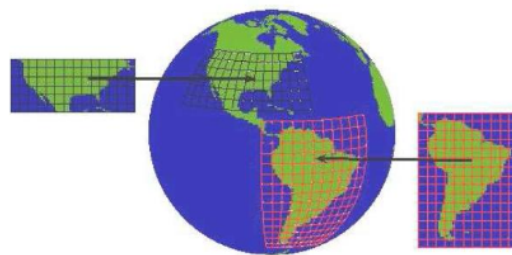


Figure 2.15: The Earth is a famous case of a manifold in \mathbb{R}^3 , where we can define a “1-1” relationship between its curved structure and a map plane (defined by the map scale), but only locally.[86]

As stated above, $SE(3)$ and $SO(3)$ are Lie groups. A Lie group is a set \mathcal{G} with an operation $*$ so that $(\mathcal{G}, *)$ has group properties [117]:

- Closure: $\forall a, b \in \mathcal{G} : a * b \in \mathcal{G}$
- Associativity: $\forall a, b, c \in \mathcal{G} : (a * b) * c = a * (b * c)$
- Identity: $\exists e \in \mathcal{G} \text{ s.t. } \forall a \in \mathcal{G} : e * a = a * e$
- Inverse element: $\forall a \in \mathcal{G} : \exists b \in \mathcal{G} \text{ s.t. } a * b = b * a = e$

and as a smooth manifold of \mathbb{R}^3 with smooth group multiplication μ and group inversion ι :

- $\mu: \mathcal{G} \times \mathcal{G} \longrightarrow \mathcal{G}, (a, b) \longrightarrow a * b$
- $\iota: \mathcal{G} \longrightarrow \mathcal{G}, \alpha \longrightarrow \alpha^{-1}$

Every Lie group has an associated Lie algebra, which is the tangent space around the identity element of the group.

Definition 2.3. A Lie algebra is a vector space \mathfrak{g} over some field \mathbf{F} together with an operation $[\cdot, \cdot]: \mathfrak{g} \times \mathfrak{g} \longrightarrow \mathfrak{g}$ that has the following properties [2]:

- **Bilinearity**

- $[ax + by, z] = a[x, z] + b[y, z]$
- $[z, ax + by] = a[z, x] + b[z, y]$

$\forall a, b \in F, x, y, z \in \mathfrak{g}$.

- **Skew symmetry**

- $[x, y] = -[y, x], \forall x, y \in \mathfrak{g}$

- **Jacobi identity**

- $[x, [y, z]] + [z, [x, y]] + [y, [z, x]] = 0, \forall x, y, z \in \mathfrak{g}$,

where $[\cdot, \cdot]$ is the **Lie Bracket operation**: $[A, B] = AB - BA$, which has its own properties:

- $[A, B] = -[B, A]$
- $[A, [B, C]] + [C, [A, B]] + [B, [C, A]] = 0$
- $[A, A] = 0$,

with A, B, C being Lie Group elements.

If a matrix A is an element of a Lie Algebra \mathfrak{g} :

$$e^A = \mathbb{I} + A + \frac{A^2}{2!} + \frac{A^3}{3!} + \dots \in \mathcal{G}. \quad (2.8)$$

This is called the **Riemannian Exponential mapping** $\mathfrak{g} \rightarrow \mathcal{G}$. The inverse operation maps \mathcal{G} back to \mathfrak{g} and is called the **Riemannian Logarithmic mapping**, as expressed by the logarithmic matrix of Lie Group elements (see in Sect.2.11).

With the Lie algebra, the tangent space at any element of its Lie group can be constructed via parallel transport [14]. This can be used to compare poses. For example, the Lie Algebra of the SO(3) Lie Group is

$$so(3) = \{\Omega | \Omega \in \mathbb{R}^{3 \times 3}, \Omega^T = -\Omega\} = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix}, \quad (2.9)$$

with $\omega = [\omega_x, \omega_y, \omega_z]^T$ and the Lie Algebra of SE(3) is

$$se(3) = \begin{bmatrix} \omega & \mathbf{v} \\ \mathbf{0}^T & 1 \end{bmatrix} \quad (2.10)$$

with $\omega = [\omega_1, \omega_2, \omega_3]^T$, $\mathbf{v} = [v_1, v_2, v_3]^T$

Using the Lie Algebra, we may define the following operations:

$\forall A, B \in M$, where M is a Lie Group, (and thus, a Riemannian Manifold), $\exists X \in T_A M$ that can be used to define the following:

- **Left Translation on the Manifold M:**

$$L_A B = AB \quad (2.11)$$

- **Right Translation on the Manifold M:**

$$R_A B = BA^{-1} \quad (2.12)$$

- **Left Tangent Mapping of X:**

$$(L_A)_* X = AX \quad (2.13)$$

- **Right Tangent Mapping of X:**

$$(R_A)_* X = XA \quad (2.14)$$

Those definitions are valid for every Lie Algebra, and thus for $\mathfrak{so}(3)$ and $\mathfrak{se}(3)$, of course.

Augmenting the definition of Object Pose: the Symmetry paradox

The definition above regarding the Object Pose has always been the status quo until recently. However, in 2017, Brégier et al. [11] stated that it is incomplete as it is geometrically unaware, neglecting the symmetry properties of the object w.r.t. its Principal Rotation Axes (see fig.2.16). As a matter of fact, proper symmetries are an attribute extremely common in man-made objects. Such symmetries make both the object shape and appearance invariant to certain transformations, which means that there may be a subset $\mathcal{M} \subset SE(3)$ of Transformations \mathbb{T}' s that may represent the same pose as a certain \mathbb{T} . As a result, the Pose Space must not be defined as a set of unique rigid transformations, but rather as a (super-)set of them. To make things clearer, it is more thorough to express the Pose as the “distinguishable static state” [11] of the object which consists of the Transformations \mathbb{T} 's that vary according to its set of Proper Symmetries. If we have a rigid Transformation $\mathbb{T}_{(O)}^{(C)}$, the set of Proper Symmetries of the object is the set of the combinations of $\mathbb{T}_{(O)}^{(C)}$ with another Rigid Transformation \mathbb{G} that would not result in a change of the Pose described by \mathbb{T} itself.

Thus, **Proper Symmetries** are defined as the set of Rigid Transformations that do not have an effect on the object's static state and more specifically, a set of rotations around its center (O).

Mathematically,

$$\mathcal{G} = \left\{ \begin{bmatrix} G_{Rot} & \mathbf{0} \\ \mathbf{0}^T & 1 \end{bmatrix} \in SE(3) \mid \mathbb{T}_{(O)}^{(C)(-1)} \cdot \mathbb{G}, G_{Rot} \in SO(3) \right\} \quad (2.15)$$

and the Pose Space is fully defined as:

$$\mathcal{P} = \{ \mathbb{T}_{(O)}^{(C)} \} = \{ \mathbb{T}_{(O)}^{(C)} \cdot \mathbb{G}, \mathbb{G} \in \mathcal{G} \} \quad (2.16)$$

and more concretely:

$$\mathcal{P} = \left\{ \begin{bmatrix} R \cdot G_{Rot} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \in \mathbb{R}^{4 \times 4} \mid R \in SO(3), G_{Rot} \in \mathcal{G}_{Rot} \subset SO(3), \mathbf{t} \in \mathbb{R}^3 \right\} \quad (2.17)$$

So, we would like to select the object pose to be an element of the Pose Space $\mathbb{T}^* \in \mathcal{P}$ where:

$$\mathbb{T}^* = \min_{G_{Rot}} \begin{bmatrix} R \cdot G_{Rot} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \quad (2.18)$$

Now, let's delve into the meaning of object symmetries. In general, proper symmetries can be divided into groups of Finite or Infinite symmetries, with the corresponding sub-categories of Revolutionary

Symmetry with/without rotoreflective invariance and Spherical/Finite but non-trivial symmetry, see Table 1 [11].

As the discussion above can be quite difficult to comprehend by the unfamiliar reader, we present specific examples where such symmetries are obvious and, as a result, the number of pose parameters needed can be reduced.

Table 1 Classification of the potential groups of proper symmetries for a 3D bounded physical object.






Infinite groups		Finite groups		
Revolution symmetry				
				
(a) Without rotoreflection invariance	(b) With rotoreflection invariance	(c) Spherical symmetry	(d) No proper symmetry	(e) Finite non trivial

Figure 2.16: The Table1 from [11] that classifies rotation symmetry categories.

The most striking example is that of objects with spherical proper symmetry like e.g. a perfect sphere (in terms of shape) that has uniform appearance on its surface (e.g. the same color at every surface point). As shown in Table 1, the pose of the sphere can be viewed as invariant w.r.t. every 3D rotation, which means that no matter which Symmetry Rotation Matrix G_{Rot} is applied to its Rotation Matrix R , the sphere has the same emblazonment both in an RGB image and a Depth map. So, we can naturally select a unique Pose Transformation from the set \mathcal{P} as the one with the minimal G_{Rot} . The invariance described above means that when we want to describe the Pose of a spherical object, the Rotation parameters are excessive and the representation can be reduced to the 3 translation parameters of \mathbf{t} .

A second enlightening example is that of an object with cylinder shape (embodied with Revolution symmetry with rotoreflective invariance). Rotoreflective invariance means, that such objects are not just symmetrical w.r.t. their symmetry axis but are also symmetrical to a plane perpendicular to their Principal Rotation Axis (in this case, the z-axis). Their rotation R can be right-multiplied with any $G_{Rot}(r_z)$ Rotational Symmetry Matrix $\in SO(3)$, without visible changes on the object's pose. Thus, the representation loses a degree of Freedom, giving us the chance to handle it as a manifold of \mathbb{R}^5 .

Last, but not least, naturally, objects with irregular shapes/appearances resulting in no Proper Symmetries, preserve the unique pose representation of the Transformation Matrix as their $G_{Rot} = \mathbb{I}_3$ and keep all their degrees of freedom.

2.11 3D Rotation Representations

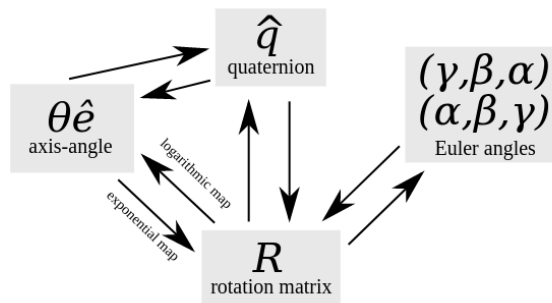


Figure 2.17: Visualization of the transformations between all standard ways of 3D rotation representation.

The object's orientation can be described in a variety of ways with the main ones being:

1. Rotation matrix
2. Euler Angles
3. Quaternia
4. Axis - Angle Representation

Let's briefly analyze their characteristics, as their choice significantly effects the performance of Pose Estimation algorithms:

1. **Rotation Matrix Representation:** The rigid body's orientation can be described by three unit vectors $\hat{\mathbf{x}}, \hat{\mathbf{y}}, \hat{\mathbf{z}}$ (one for each axis of the reference frame), which can be combined in the (3×3) matrix $R \in SO(3)$, by stacking them as columns.

$$R = \begin{bmatrix} \hat{\mathbf{x}} & \hat{\mathbf{y}} & \hat{\mathbf{z}} \end{bmatrix} = \begin{bmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \end{bmatrix} = \begin{bmatrix} \hat{\mathbf{x}}^T \hat{\mathbf{x}} & \hat{\mathbf{y}}^T \hat{\mathbf{x}} & \hat{\mathbf{z}}^T \hat{\mathbf{x}} \\ \hat{\mathbf{x}}^T \hat{\mathbf{y}} & \hat{\mathbf{y}}^T \hat{\mathbf{y}} & \hat{\mathbf{z}}^T \hat{\mathbf{y}} \\ \hat{\mathbf{x}}^T \hat{\mathbf{z}} & \hat{\mathbf{y}}^T \hat{\mathbf{z}} & \hat{\mathbf{z}}^T \hat{\mathbf{z}} \end{bmatrix} \quad (2.19)$$

Since, R is orthonormal: $R^{-1} = R^T$.

Nevertheless, rotation matrices are considered redundant representations of orientation as they require 9 parameters, combined with the 6 corresponding constraints of orthogonality, to describe only 3 possible degrees of freedom of rotation. The main side-effects of this are memory inefficiency and the hazard of numerical drift during floating point operations that may result in diverging from the orthogonality constraints in case of regression. So, a minimal representation of only 3 independent parameters needs to be achieved as long as the $SO(3)$ structure of R is preserved.

2. **Euler Angles:** Such a minimal representation is that of Euler Angles, where the rotation vector \mathbf{r} is described with a set of three angles: $\mathbf{r} = [\phi, \theta, \psi]^T$ where ϕ is roll angle, θ is yaw angle and ψ is the pitch angle. Each of the elements of this triplet can produce a rotation matrix that represents an elementary rotation with respect to its corresponding axis and we can combine such elementary rotation matrices multiplicatively in order to acquire a general 3D rotation. However, the order of doing so is not standard since a sequence of them is suitable only when it is guaranted that two successive rotations are not made about parallel axes. Thus, we have resulted into 27 different possible combinations. In this thesis, the **XYZ** order was used:

$$R(\mathbf{r}) = R(\psi)R(\theta)R(\phi) = \begin{bmatrix} \cos(\theta) \cos(\psi) & -\cos(\psi) \cos(\theta) & \sin(\theta) \\ \cos(\phi) \sin(\psi) + \cos(\psi) \sin(\phi) \sin(\theta) & \cos(\phi) \cos(\psi) - \sin(\phi) \sin(\theta) \sin(\psi) & -\cos(\theta) \sin(\phi) \\ \sin(\phi) \sin(\psi) - \cos(\phi) \cos(\psi) \sin(\theta) & \cos(\psi) \sin(\phi) + \cos(\phi) \sin(\theta) \sin(\psi) & \cos(\phi) \cos(\theta) \end{bmatrix} \quad (2.20)$$

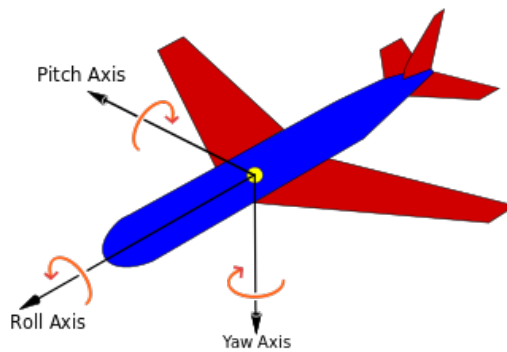


Figure 2.18: The Roll-Yaw-Pitch angle representation w.r.t the center mass of a rigid object.[156]

Euler angles are compact and intuitive (matching each rotation axis with a corresponding angle) and arithmetically stable. However, the fact that the rotation order counts causes extra ambiguity cases, the most known being that of “Gimbal Lock”.

The Problem of Gimbal Lock: It arises when 2 Euler rotation axes align, resulting in losing 1 DOF of the rotation representation. Graphically, that happens if the two outer axes (different according to the Euler order convention) in fig.2.20 become parallel, creating the image of fig.2.19.

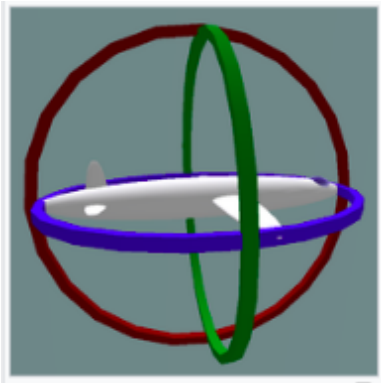


Figure 2.19: Normal situation: 3 independent Gimbals.[156]

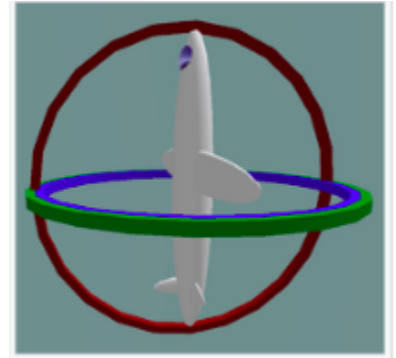


Figure 2.20: Gimbal lock: 2 out of 3 Gimbals are aligned - 1 DOF lost.[156]

3. Axis-Angle Representation: The “Euler Rotation Theorem” states that:

"In \mathbb{R}^3 , any displacement of a rigid body such that a point on the rigid body remains fixed, is equivalent to a single rotation about some axis that runs through the fixed point".

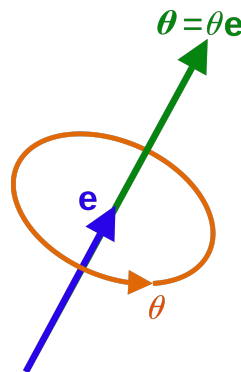


Figure 2.21: The Axis-Angle 3D rotation convention.[154]

So, we can transform every rotation sequence to a single rotation with respect to the unitary axial vector $\mathbf{e} = [e_1, e_2, e_3]^T$, with a start at the World Coordinate Center (W), by an angle amplitude θ . This is a non-minimal parametric representation of the rotation matrix $R(\theta, \mathbf{e})$, as it comprises of 4 parameters. To this end, we embed θ in the rotation vector, constructing the Euler vector:

$$\boldsymbol{\theta} = \theta \mathbf{e}, \quad (2.21)$$

which incorporates the full rotation information. Knowing the Euler vector, on the other hand, allows as to disentangle the rotation amplitude:

$$\theta = \|\boldsymbol{\theta}\|_2 \quad (2.22)$$

from the rotation direction:

$$\mathbf{e} = \frac{\boldsymbol{\theta}}{\|\boldsymbol{\theta}\|_2}. \quad (2.23)$$

The **Exponential Matrix** of a matrix $A \in \mathbb{R}^{n \times n}$ is:

$$e^A = \sum_{k=0}^{\infty} \frac{1}{k!} A^k \quad (2.24)$$

and its inverse, the **Logarithmic Matrix**:

$$\log(A) = \sum_{k=1}^{\infty} \frac{(-1)^{k+1}}{k} (A - \mathbb{I})^k \quad (2.25)$$

For the special case of Rotation matrices $R \in SO(3)$, we may express it using the **Rodriguez formula**, which is in fact an exponential mapping $\exp(\cdot) : so(3) \rightarrow SO(3)$. The Rodriguez formula:

$$R(\theta, \mathbf{e}) = e^{\theta \times} = \mathbb{I}_3 + \frac{\sin(\theta)}{\theta} E_{\times} + \frac{1 - \cos(\theta)}{\theta^2} E_{\times}^2, \quad (2.26)$$

where A_{\times} is the cross-product matrix for a corresponding vector \mathbf{a} :

$$\mathbf{a} \times \mathbf{b} = A_{\times} \mathbf{b}, \quad (2.27)$$

with:

$$A_{\times} = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix}. \quad (2.28)$$

The inverse mapping of the Exponential Matrix: $\exp(\cdot) : so(3) \rightarrow SO(3)$ is the *Logarithmic Matrix* : $\log(\cdot) : SO(3) \rightarrow so(3)$:

$$\log(R) = \begin{cases} 0, & \theta = 0 \\ \frac{\theta}{2\sin(\theta)} (R - R^{-1}), & \theta \in (0, \pi). \end{cases} \quad (2.29)$$

Unit Quaternia: Quaternia are the 4D expansion of complex numbers.

They are defined as follows:

$$\mathbf{q} = q_1 \hat{\mathbf{i}} + q_2 \hat{\mathbf{j}} + q_3 \hat{\mathbf{k}} + q_0 \quad (2.30)$$

with $\hat{\mathbf{i}}, \hat{\mathbf{j}}, \hat{\mathbf{k}}$ being hyperimaginary numbers that:

$$\hat{\mathbf{i}}^2 = \hat{\mathbf{j}}^2 = \hat{\mathbf{k}}^2 = \hat{\mathbf{i}}\hat{\mathbf{j}}\hat{\mathbf{k}} = -1 \quad (2.31)$$

and from there we can show:

$$-\hat{\mathbf{i}}\hat{\mathbf{j}} = \hat{\mathbf{j}}\hat{\mathbf{i}} = \hat{\mathbf{k}} \quad (2.32)$$

$$-\hat{\mathbf{j}}\hat{\mathbf{k}} = \hat{\mathbf{k}}\hat{\mathbf{j}} = \hat{\mathbf{i}} \quad (2.33)$$

$$-\hat{\mathbf{k}}\hat{\mathbf{i}} = \hat{\mathbf{i}}\hat{\mathbf{k}} = \hat{\mathbf{j}}. \quad (2.34)$$

Quaternia multiplication:

$$\begin{aligned} \mathbf{qp} &= (q_1 \hat{\mathbf{i}} + q_2 \hat{\mathbf{j}} + q_3 \hat{\mathbf{k}} + q_0)(p_1 \hat{\mathbf{i}} + p_2 \hat{\mathbf{j}} + p_3 \hat{\mathbf{k}} + p_0) = \\ & (q_1 p_0 - q_2 p_3 + q_3 p_2 + q_0 p_3) \hat{\mathbf{i}} + (q_1 p_3 + q_2 p_0 - q_3 p_1 + q_0 p_2) \hat{\mathbf{j}} + \\ & + (-q_1 p_2 + q_2 p_1 + q_3 p_0 + q_0 p_3) \hat{\mathbf{k}} + (-q_1 p_1 - q_2 p_2 - q_3 p_3 + q_0 p_0) = \end{aligned} \quad (2.35)$$

$$\begin{bmatrix} q_1 p_0 - q_2 p_3 + q_3 p_2 + q_0 p_3 \\ q_1 p_3 + q_2 p_0 - q_3 p_1 + q_0 p_2 \\ -q_1 p_2 + q_2 p_1 + q_3 p_0 + q_0 p_3 \\ -q_1 p_1 - q_2 p_2 - q_3 p_3 + q_0 p_0 \end{bmatrix}$$

which is not commutable:

$$\mathbf{q}^T \mathbf{p} = \mathbf{p}^T \mathbf{q}. \quad (2.36)$$

Geometrically, if \mathbf{q}, \mathbf{p} represent two separate 3D rotations, then the result of their multiplication gives the ultimate rotation if we execute first the one that is represented by \mathbf{p} and then the one of \mathbf{q} .

Quaternion conjugate:

$$\mathbf{q}^* = -q_1 \hat{\mathbf{i}} - q_2 \hat{\mathbf{j}} - q_3 \hat{\mathbf{k}} + q_0 \quad (2.37)$$

Quaternion norm:

$$\|\mathbf{q}\|_2 = \sqrt{\mathbf{q}^* \mathbf{q}} = \sqrt{\mathbf{q} \mathbf{q}^*} = \sqrt{q_1^2 + q_2^2 + q_3^2 + q_0^2} \quad (2.38)$$

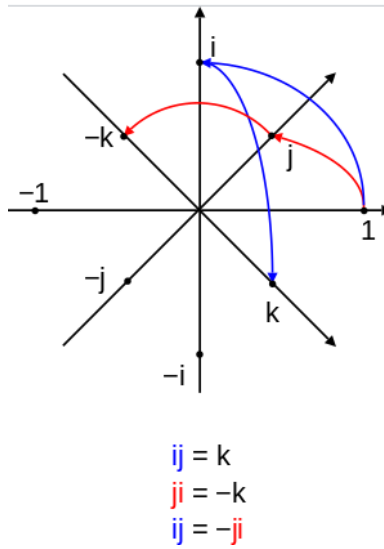


Figure 2.22: The 3D Unit Quaternia rotation space.[159]

Quaternia/Axis-Angle Representations connection

We can connect the two latter rotation representation as follows:

A vector \mathbf{e} can be written as:

$$\mathbf{e} = \begin{bmatrix} k_x \sin \frac{\theta}{2} \\ k_y \sin \frac{\theta}{2} \\ k_z \sin \frac{\theta}{2} \end{bmatrix} = \hat{\mathbf{k}} \sin \frac{\theta}{2}. \quad (2.39)$$

Then, we may consider the quaternion \mathbf{q} as a rotation around \mathbf{e} at an angle θ . For it to be unitary ($\|\mathbf{q}\| = 1$) we choose:

$$q_0 = \cos \frac{\theta}{2} \quad (2.40)$$

and

$$\|\hat{\mathbf{k}}\|_2 = 1. \quad (2.41)$$

Consequentially, the imaginary part of a quaternion represents the rotation axis direction and the real part half the cosine of the rotation angle magnitude.

Inverse Quaternion:

For every quaternion \mathbf{q} , there exists its inverse one, $-\mathbf{q}$:

$$-\mathbf{q} = -q_1\hat{\mathbf{i}} - q_2\hat{\mathbf{j}} - q_3\hat{\mathbf{k}} - q_0 \quad (2.42)$$

that represents the *same* rotation. This may seem anti-intuitive until you think that “if \mathbf{q} corresponds to the 3D rotation pair (\mathbf{e}, θ) , then according to the above relationship, $-\mathbf{q}$ should correspond to $(-\mathbf{e}, -\theta)$ which gives an opposite amount of rotation on the opposite direction, resulting in the original one.

On the contrary, if we want to express an inverse 3D rotation in a quaternion form we should take $-\mathbf{q}^*$, which implies a rotation around \mathbf{e} by $-\theta$.

Quaternion-based rotation representations have the advantage of avoiding ambiguities such as Gimbal lock and they are compact and arithmetically stable. On the other hand, when they are utilized in a learning algorithm, they cause ambiguities due to their ability to represent the same rotation in multiple ways, as explained above.

So, in summary:

- **Axis-Angle \rightarrow Quaternia:**

If we have a vector $\theta = \theta\mathbf{e}$, then we compute its corresponding quaternion:

$$\mathbf{q} = \frac{e_x \sin \frac{\theta}{2}}{\sqrt{e_x^2 + e_y^2 + e_z^2}}\hat{\mathbf{i}} + \frac{e_y \sin \frac{\theta}{2}}{\sqrt{e_x^2 + e_y^2 + e_z^2}}\hat{\mathbf{j}} + \frac{e_z \sin \frac{\theta}{2}}{\sqrt{e_x^2 + e_y^2 + e_z^2}}\hat{\mathbf{k}} + \cos \frac{\theta}{2} \quad (2.43)$$

- **Quaternia \rightarrow Axis-Angle:**

$$\theta = \begin{bmatrix} \frac{2q_1 \arccos q_0}{\sqrt{1-q_0^2}} \\ \frac{2q_2 \arccos q_0}{\sqrt{1-q_0^2}} \\ \frac{2q_3 \arccos q_0}{\sqrt{1-q_0^2}} \end{bmatrix} \quad (2.44)$$

2.12 Introduction to the approach

In this work, we are proposing a Multi-Attentional Convolutional framework for Real-time temporal 6D Single Object Pose Tracking. Our architecture is trained end-to-end using exclusively a synthetic dataset and achieving new, State-of-the-Art performance in the most demanding scenarios of the most complete and challenging test set for our problem up to date: the one introduced in [83].

Specifically, like [29], at every timestep, along with the “Observed” RGB-D pair I_t (from the video stream), we use as input to our network the “Predicted” one, \hat{I}_t , which is created by rendering the known 3D object model at the pose prediction of the previous timestep and we try to estimate the Pose Transform $\Delta\mathbb{T}_t$ between the two pairs, with a CNN architecture that takes care of each of the aforementioned challenges in a specific way. For generating our synthetic dataset, we follow the procedure introduced by [29] (we will discuss the details in a following Section).

2.12.1 A discussion about the nature of the information sources

- **RGB Modality:** The RGB modality encapsulates the appearance information of the scene. It is the one with the most intuitive content as it projects its high resolution, affordability and localization properties w.r.t. to the object’s presence in the scene as advantages. However, it is susceptible to color noise i.e. multiple clashing patterns present in the same scene that may confuse a Computer Vision algorithm, to blurring due to fast movement of the camera/target, to possible illumination variabilities and to lack of information about the object’s scale w.r.t. to the camera. This disadvantage makes its usage in 3D perception tasks extremely challenging (especial in the monocular case) as it inserts ambiguities concerning some of the task’s components (e.g. the object’s translation in the z_c dimension etc.).
- **Depth Modality:** The Depth Modality is used alongside the RGB one to solve the problem of scale ambiguity, although it is susceptible to poor resolution and limited depth and field of view.

It provides information about the 3D shape of the objects. However, it is considered extremely noisy, as every suspicion of specularity and/or transparency in the objects makes the light beams to reflect unpredictably on its surface. Additionally, applications that are mainly based on this modality cannot be employed in mobile devices, as they rarely have a depth sensor embedded in their hardware.

2.12.2 Datasets

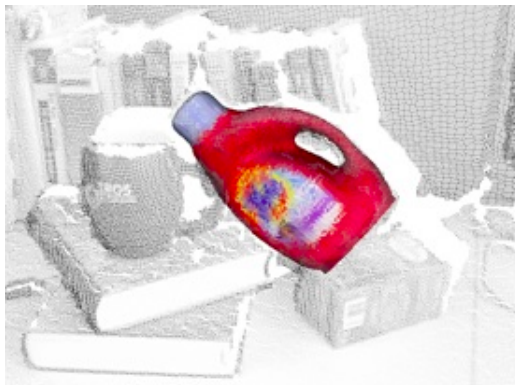


Figure 2.23: A Typical RGB frame for the synthetic dataset of Choi and Christensen [17].

Historically, the first dataset used to evaluate 6-DOF tracking algorithms was introduced in 2013 by Choi and Christensen [17] and consists of 4 short sequences of purely synthetic scenes. The scenes are made of unrealistic, texture-less backgrounds with a single colored object to track, resulting in noiseless RGBD images (see fig.2.23). The object is static and the camera rotates around it in wide motions, occasionally creating small occlusions (at most 20% of the object is occluded). While challenging at first, most recent Deep Learning-based methods, when evaluated on this dataset have reached a near-perfect accuracy of $0.1mm/0.07^\circ$ [128], which shows that the dataset has reached the end of its useful life.

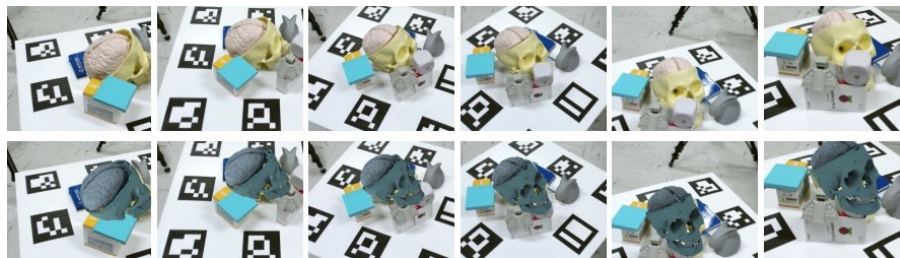


Figure 2.24: The dataset introduced in the first work of Garon et al. [29]. Objects are static on a 2D turntable and their pose labels are obtained with the use of fiducial markers. This dataset includes occlusion scenarios, where the object is covered horizontally or vertically by a white plane by a percentage starting from 10% and gradually increasing to 40%.

Another dataset, introduced by Garon and Lalonde [29], includes 12 sequences of real objects captured with real sensors. While a significant improvement over the synthetic dataset of [17], dealing with real data raises the issue of providing accurate ground truth pose of the object at all times. To obtain this ground truth information, their strategy (also adopted in 6-DOF detection datasets [[112], [55]]) is to use calibration boards with **fiducial markers**. While useful to accurately and easily determine an object pose, this has the unfortunate consequence of constraining the object to lie on a large planar surface (fig.2.24).

A special mention has to be made to the LineMOD [112] and the HomebrewedDB [55] pose detection datasets.

- **LINEMOD** and its evolution **OCCLUDED LINEMOD** contain 15, 1100+ frame video sequences of 15 various objects and ,each, for the evaluation of competing pose detectors. In both

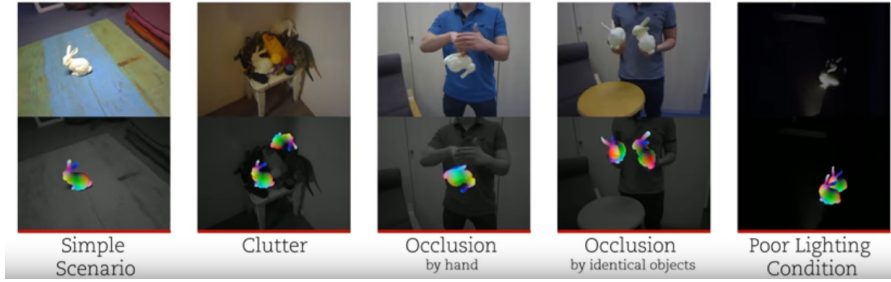


Figure 2.25: The testing scenarios presented in [128]. This test dataset contains a wide variety of tracking challenges as it starts from simple single object cases and it slowly escalates to adding clutter by other objects, dynamic occlusions by the user’s hands, multiple instances of the same objects and lighting variation.

sets the objects are standing on a turntable under the presence of heavy clutter and/or occlusions by other instances of the same object or entirely different models. It is considered the standard in the domain of static object pose detection for years.

- **HomebrewedDB** contains 33 object models (17 toy, 8 household and 8 industry-relevant objects) over 13 static scenes of various difficulty (with annotated 3D ground truth poses). The testing scenarios feature both single and multi-instance object scenes, both the presence and absence of heavily clutter and a wide variety of lighting sources and conditions. It also features a set of benchmarks to test various desired detector properties, particularly focusing on scalability with respect to the number of objects and resistance to changing light conditions, occlusions and clutter. It practically consists of an evolution of the LineMOD database.

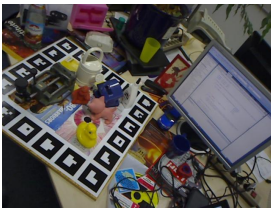


Figure 2.26: A scene of the LINEMOD [112] pose detection dataset.

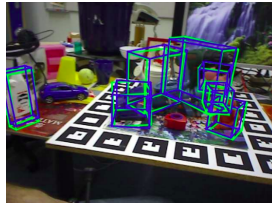


Figure 2.27: A scene of the LINEMOD [112] pose detection dataset, under the presence of severe occlusions between the various objects in the scene.



Figure 2.28: Sample RGB images from sequences belonging to the domain adaptation benchmark of [55].

The main problem with both those datasets is that they are designed with the problem of Object Pose Detection (especially in a potential robotic workspace) more in mind than that of Object Pose Temporal Tracking (in an alternative workspace that would suit both robotic and AR applications). As a result the object is constrained to be static and to lie on a 2D planar surface while the camera is moving slowly to estimate its pose under a series of different viewpoints. However, the camera speed is very low for the testing scenarios to be challenging for a temporal tracker that utilizes feedback information and the occlusions are static and remain in the same topology throughout the testing video sequence. On the contrary, temporal tracking requires testing the proposed approaches in videos with complex and abrupt motion patterns.

Lastly, we feel the need to honourably mention the YCB [15] and the FAT [136] object pose benchmarks, as well. We do not perform a more detailed analysis of their strengths and weaknesses, as they are more or less similar to the previous two in terms of design preception.

Speaking of complex motion patterns, an interesting case to consider is the testing scenarios presented in [128]. This test dataset contains a wide variety of tracking challenges as it starts from simple single object cases and it slowly escalates to adding clutter by other objects, dynamic occlusions by the user’s hands, multiple instances of the same objects and lighting variation. However, despite of its demonstrative powers, it has two distinctive disadvantages that discredit it from being an option to train our models on:

- The first and the most significant one is the absence of 3D pose annotations in it, which makes it unsuitable for Deep Neural Network training, but most importantly, it makes it unsuitable to test our models on, as it makes it harder to measure quantitative errors.
- The second problem is the absence of a variety of highly textured 3D object models that constraints our possible experiments and results.

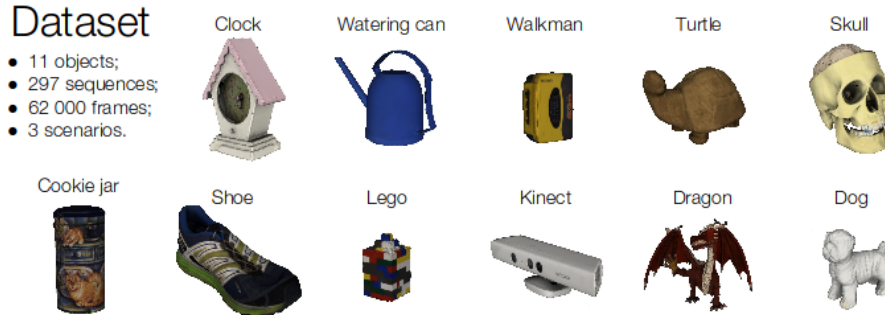


Figure 2.29: The dataset presented by Garon et al. in [83] is the one we work on in this thesis. Here, we present the collection of 3D object models contained in it.

Consequentially, the choice we make to test our tracker on is the dataset recently presented in [83], which is an evolution of [29] but this time without the constraint of the 2D plane. In general, a user is standing in front of a steady Kinect sensor and moves each of the objects of the dataset either by rotating a 2D planar surface it lies on or by moving in specific configurations in 3 dimensions. The ground truth object poses are recorded and kept to measure errors with the aid of Motion capture sensors.

2.12.3 Inspection of the dataset

The dataset we work on contains 11 scanned 3D objects with a variety in symmetry, appearance and texture, presented in fig.2.29. For our experimentation, we use the “Dragon”, the “Cookie Jar”, the “Dog”, the “Lego Block” and the “Watering Can” models, since it would be too time-consuming to evaluate our approach on all the objects. This model selection is representative of two extreme situations: the “Dragon” model is highly asymmetrical and textured, with parts that stand out of its main body, while the “Cookie Jar” model is textured more poorly and is described by a simpler and symmetric shape. Using both of them, we will be able to make a “value for computational cost” analysis of our approach and extract conclusions w.r.t. its generalization capabilities to different object types. The Deep Network is trained from scratch for every different object and we operate the same for every object.

Scenarios

The dataset of [83] contains 3 scenario categories:

- **The Stability Scenarios:** where the object is steady on a table on various poses. It is used for measuring the tracker’s jitter and robustness from frame to frame. This is important as it shows the tolerance of the tracker to its initialization and the margins of the Initial Pose Estimator we need to employ.
- **The Occlusion Scenarios:** where the object is again steady on a turning table and occluded vertically or horizontally by an increasing percentage by a white plane, hiding its attributes. It is used to measure the tracker’s robustness to occlusions.
- **The Interaction Scenarios:** where the object is moving in the 3D space by a human-user. It is the most interesting scenario category, the most novel one (as it is presented in no other dataset) and the one being closest to a Real-World experience scenario. It is divided in 4 categories:

Free interaction

The object is freely manipulated in 4 different ways: translation only, rotation only, rotation and translation and a hard sequence where multiple occlusions can occur.



Figure 2.30: Examples of the “Free hard interaction” scenario of the dataset of [83] on 2 separate objects: the clock and the dragon ones.

- *Interaction Translation*: where the object keeps mostly its rotation during its movement with light hand-occlusions
- *Interaction Rotation*: where the object keeps mostly its 3D position in the user’s hands and change its rotation with light hand occlusions.
- *Interaction Free Easy*: where we combine both the above scenarios
- *Interaction Free Hard*: where we again combine the Translation and Rotation movement, but this time with faster frame-by-frame movement (and as a result heavier motion blur) and severe occlusions by the user’s hand. It is, justly, considered the most difficult of them all and that is why **we design our approach having this scenario in mind, for the most part, as we expect that a Neural Architecture with the best results on it, will mostly have the best results when tested on all the other scenarios, as well.** We validate this opinion by testing our method on most of the scenarios present in the dataset at hand for the aforementioned object cases.

2.13 Thesis Structure

We organize our work as follows: in Chapter 3, we establish all the necessary background knowledge for every component of our approach. We suggest for this chapter to be used as a terminological glossary by the reader, in order to delve into each component of our approach. In Chapter 4, we examine previous approaches in the fields of Object/Camera Pose Estimation and Tracking. Later, in Chapter 5, we explain how we generate our synthetic training dataset and in Chapter 6 we present our completed approach. In Chapter 7, we discuss all the design choices that we made during the construction of our CNN architecture and in Chapters 8, 11, respectively, we evaluate and conclude our results. Finally, in Chapter 9, we discuss the extensions of our work we have already actively researched and in 10 we propose future research directions for our approach.

Chapter 3

Background

"Finally, from so little sleeping and so much reading, his brain dried up and he went completely out of his mind."

Miguel de Cervantes Saavedra, "Don Quixote"

3.1 Mathematical Background

3.1.1 Matrix Trace

Trace (often abbreviated to tr) of a square matrix A is defined to be the sum of elements on the main diagonal (from the upper left to the lower right) of A . The trace of a matrix is the sum of its (complex) eigenvalues, and it is invariant with respect to a change of basis. This characterization can be used to define the trace of a linear operator in general. The trace is only defined for a square matrix ($n \times n$).

Mathematically, let's say that we have a square matrix $A \in R^{n \times n}$:

$$\text{tr}(A) = \sum_{i=1}^n a_{ii} = a_{11} + a_{22} + \cdots + a_{nn} \quad (3.1)$$

Properties:

- Additive property: $\text{tr}(A + B) = \text{tr}(A) + \text{tr}(B)$
- Transpositivity: $\text{tr}(A^T) = \text{tr}(A)$
- Scalar Multiplication: $\text{tr}(cA) = c\text{tr}(A)$
- Cyclic Property: $\text{tr}(ABC) = \text{tr}(BCA) = \text{tr}(CAB) = \text{tr}(CBA)$

3.1.2 Frobenius norm

The Frobenius norm $\|A\|_F$ is the matrix norm of an $m \times n$ matrix A defined as the square root of the sum of the absolute squares of its elements:

$$\|A\|_F = \sqrt{\sum_{i=1}^m \sum_{j=1}^n |a_{ij}|^2}. \quad (3.2)$$

It is also equal to the square root of the matrix trace of AA^H , where A^H is the conjugate transpose, i.e.,

$$\|A\|_F = \sqrt{\text{tr}(AA^H)} \quad (3.3)$$

3.1.3 Gram-Schmidt Orthonormalization

Let's assume that we have 3 linearly independent vectors: $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3$. If we want to transform them into 3 orthonormal vectors $\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3$ we use the following process:

- \mathbf{e}_1 may have any direction possible, so it may follow the direction of \mathbf{v}_1 , but it is required to have unit norm, so:

$$\mathbf{e}_1 = \frac{\mathbf{v}_1}{\|\mathbf{v}_1\|_2} \quad (3.4)$$

- In order to obtain \mathbf{e}_2 , we wish to subtract the \mathbf{v}_2 component that is linearly dependent to \mathbf{e}_1 (see fig.3.1):

$$\mathbf{e}_2 = \mathbf{v}_2 - (\mathbf{e}_1^T \mathbf{v}_2) \mathbf{e}_1 \quad (3.5)$$

- Finally, acquiring \mathbf{e}_3 will be succeeded after we subtract the component of \mathbf{v}_3 that lies on the plane of $\mathbf{e}_1, \mathbf{e}_2$. In other words, \mathbf{e}_3 is the component of \mathbf{v}_3 that is perpendicular to the plane of $\mathbf{e}_1, \mathbf{e}_2$ and has unit norm:

$$\mathbf{e}_3 = \frac{\mathbf{v}_3 - (\mathbf{e}_1^T \mathbf{v}_3) \mathbf{e}_1 - (\mathbf{e}_2^T \mathbf{v}_3) \mathbf{e}_2}{\|\mathbf{v}_3 - (\mathbf{e}_1^T \mathbf{v}_3) \mathbf{e}_1 - (\mathbf{e}_2^T \mathbf{v}_3) \mathbf{e}_2\|_2} \quad (3.6)$$

This idea naturally generalizes to n vectors $\mathbf{v}_i, i=1, \dots, n$: we transform each one of them to their orthonormal equivalent \mathbf{e}_i by subtracting its components that are perpendicular to the previous $\mathbf{e}_j, j=1, \dots, i-1$ and dividing by their norm.

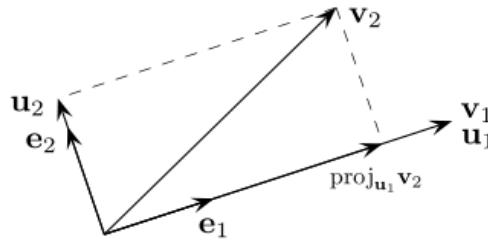


Figure 3.1: Graphical description of the Gram-Schmidt process for \mathbf{v}_2 . It is comprised by one component parallel to \mathbf{e}_1 (that we subtract), and another one perpendicular to it (that we keep normalized).

3.1.4 Cosine Similarity

Given two input attributes A and B (e.g. feature map outputs of a Convolutional Neural Network (see Section 3.3.5)), their **Cosine similarity (CS)**, is represented as follows:

$$CS = \frac{A \cdot B}{\|A\|_F \cdot \|B\|_F} = \frac{\sum_{c=1}^C \sum_{i=1}^H \sum_{j=1}^W A_c[i, j] B_c[i, j]}{\sqrt{\sum_{c=1}^C \sum_{i=1}^H \sum_{j=1}^W A_c^2[i, j]} \cdot \sqrt{\sum_{c=1}^C \sum_{i=1}^H \sum_{j=1}^W B_c^2[i, j]}} \quad (3.7)$$

The resulting similarity ranges from -1 meaning exactly opposite, to 1 meaning exactly the same, with 0 indicating orthogonality, while in-between values indicate intermediate similarity or dissimilarity. If we want to normalize the Cosine Similarity's values in $[0, 1]$ we transform it as follows: $CS' = \frac{CS+1}{2}$.

3.1.5 SVD

Let's say we have a matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$. It is useful to think of the symmetric matrices that are produced with the use of \mathbf{A} :

$$\mathbf{A} \mathbf{A}^H = \mathbf{U} \mathbf{\Sigma} \mathbf{\Sigma}^T \mathbf{U}^H = \sum_{i=1}^r \sigma_i^2 \mathbf{u}_i \mathbf{u}_i^H \quad (3.8)$$

$$\mathbf{A}^H \mathbf{A} = \mathbf{V} \mathbf{\Sigma}^T \mathbf{\Sigma} \mathbf{V}^H = \sum_{i=1}^r \sigma_i^2 \mathbf{v}_i \mathbf{v}_i^H, \quad (3.9)$$

which share the same eigenvalues (as they are Hermitian), which, are equal to their singular values squared:

$$\lambda_i(\mathbf{A}\mathbf{A}^H) = \lambda_i(\mathbf{A}^H\mathbf{A}) = \sigma_i^2(\mathbf{A}\mathbf{A}^H), \quad (3.10)$$

$i=1, \dots, r$, where r is the Matrix rank of \mathbf{A} .

Matrix Rank $\text{rank}(\mathbf{A})=r$: is the maximum number of linearly independent matrix columns and it equals with the maximum number of linearly independent matrix rows.

It is well known that $\mathbf{A}\mathbf{A}^H, \mathbf{A}^H\mathbf{A}$ are symmetric and positive definite and that the eigenvalues of symmetric matrices are real and their eigenvectors orthogonal:

- $\lambda_{\mathbf{A}\cdot\mathbf{A}^H}^{(i)} \in \mathbb{R}$
- $\mathbf{e}_i^T(\mathbf{A} \cdot \mathbf{A}^H) \cdot \mathbf{e}_j = \{0, i \neq j\}$

Then, we have the ability to decompose it in the following way:

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T = \sum_{i=1}^r \sigma_i \mathbf{u}_i^T \mathbf{v}_i \quad (3.11)$$

with $\mathbf{U} \in \mathbb{R}^{m \times m}$, $\mathbf{\Sigma} \in \mathbb{R}^{m \times n}$, $\mathbf{V} \in \mathbb{R}^{n \times n}$.

\mathbf{U} and \mathbf{V} are orthonormal ($\text{rows} \cdot \text{columns}^H = 0$, when they belong in $\mathbb{R}^{m,n}$ and $\text{rows} \cdot \text{columns}^H = 0$, when they belong in $\mathbb{C}^{m,n}$, $\mathbf{\Sigma}$ is a positive definite matrix $\mathbf{\Sigma} = \text{diag}(\sigma_i)$ and r is the Matrix Rank of \mathbf{A} .

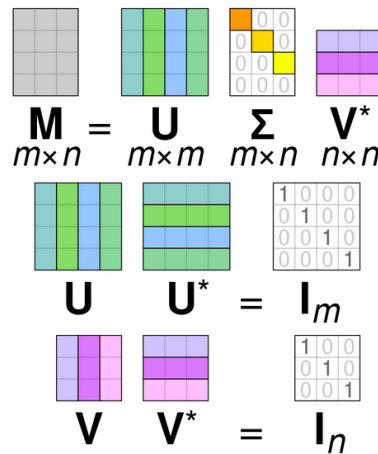


Figure 3.2: A Graphical representation of the SVD decomposition of a given matrix \mathbf{M} . [160]

3.2 Computer Vision Elements

3.2.1 HSV Color Space

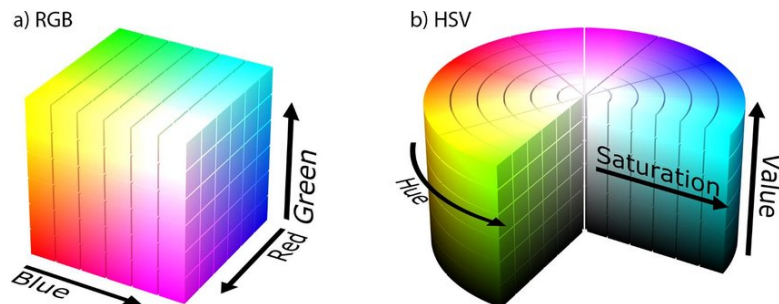


Figure 3.3: The RGB and HSV color spaces. [157]

The RGB (red, green, blue) color model is the most well-known way to mix and create colors. Unlike it, HSV is closer to how humans perceive color. It has three components: **hue**, **saturation** and **value**.

- **Hue:**

Hue is the color portion of the model, expressed as a number from 0 to 360 degrees:

- **Saturation:**

Saturation describes the amount of gray in a particular color, from 0 to 100 %. Reducing this component toward zero introduces more gray and produces a faded effect. Sometimes, saturation appears as a range from just [0,1], where 0 is gray, and 1 is a primary color.

- **Value:**

Value works in conjunction with saturation and describes the brightness or intensity of the color, from 0 – 100%, where 0 is completely black, and 100 is the brightest and reveals the most color.

RGB → HSV:

$$\begin{aligned}
 V &= \max\{R, G, B\} \\
 V_{min} &= \min\{R, G, B\} \\
 S &= \begin{cases} \frac{V-V_{min}}{V}, & \text{if } V \neq 0 \\ 0, & \text{if } V = 0 \\ \frac{G-B}{V-V_{min}}, & \text{if } V = R \\ 2 + \frac{B-R}{V-V_{min}}, & \text{if } V = G \\ 4 + \frac{R-G}{V-V_{min}}, & \text{if } V = B \end{cases} \\
 H &= \begin{cases} H' + 360^\circ, & \text{if } H' < 0 \\ H', & \text{if } H' > 0 \end{cases}
 \end{aligned} \tag{3.12}$$

HSV → RGB:

We have $0^\circ \leq H < 360^\circ$, $0 \leq S \leq 1$ and $0 \leq V \leq 1$:

$$\begin{aligned}
 C &= V \times S \\
 X &= C \cdot (1 - |\text{mod}(\frac{H}{60^\circ}, 2) - 1|) \\
 m &= V - C \\
 (R', G', B') &= \begin{cases} (C, X, 0), & \text{if } 0^\circ \leq H \leq 60^\circ \\ (X, C, 0), & \text{if } 60^\circ \leq H \leq 120^\circ \\ (0, C, X), & \text{if } 120^\circ \leq H \leq 180^\circ \\ (0, X, C), & \text{if } 180^\circ \leq H \leq 240^\circ \\ (X, 0, C), & \text{if } 240^\circ \leq H \leq 300^\circ \\ (C, 0, X), & \text{if } 300^\circ \leq H \leq 360^\circ \end{cases} \\
 \begin{bmatrix} R \\ G \\ B \end{bmatrix} &= \begin{bmatrix} (R' + m) \cdot 255 \\ (G' + m) \cdot 255 \\ (B' + m) \cdot 255 \end{bmatrix}
 \end{aligned} \tag{3.13}$$

3.2.2 SSIM Simiarity Metric

The **Structural Similarity (SSIM) index** is a method for measuring the visual (instead of the mathematical) similarity between two images. The SSIM index is calculated on various windows of an image. The measure between two windows x and y of common size $N \times N$ is:

$$SSIM(x, y) = \frac{(2\mu_x \cdot \mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \tag{3.14}$$

with:

- μ_x : the average of x
- μ_y : the average of y
- σ_x^2 : the variance of x
- σ_y^2 : the variance of y
- σ_{xy} : the covariance of x,y
- $c_1 = (k_1 \cdot L)^2$
- $c_2 = (k_2 \cdot L)^2$ (two variables to stabilize the division with weak denominator)
- L: the dynamic range of the pixel-values (typically this is $2 \cdot \frac{\text{bits}}{\text{pixel}} - 1$).
- $k_1 = 0.01$
- $k_2 = 0.03$

The SSIM formula is based on three comparison measurements between the samples of x and y: **luminance (l)**, **contrast (c)** and **structure (s)**.

The individual comparison functions are:

- $$l(x, y) = \left\{ \frac{2\mu_x\mu_y + c_1}{\mu_x^2 + \mu_y^2 + c_1} \right\} \quad (3.15)$$

- $$c(x, y) = \left\{ \frac{2\sigma_x\sigma_y + c_2}{\sigma_x^2 + \sigma_y^2 + c_2} \right\} \quad (3.16)$$

- $$s(x, y) = \left\{ \frac{\sigma_{xy} + c_3}{\sigma_x\sigma_y + c_3} \right\} \quad (3.17)$$

with, in addition to above definitions: $c_3 = \frac{c_2}{2}$

SSIM is, then, a weighted combination of those comparative measures:

$$SSIM(x, y) = \left[l(x, y)^\alpha \cdot c(x, y)^\beta \cdot s(x, y)^\gamma \right] \quad (3.18)$$

Setting the weights α, β, γ to 1, the formula can be reduced to the form shown at the top of this Subsection.

3.2.3 Optical Flow

Optical flow is used to estimate the 2D projection of the 3D velocities of all visible 3D points of interest. It is translated as the motion of the pixels of an image sequence (under the brightness constancy assumption) and provides a dense (point to point) pixel correspondance. The Optical flow is the principal component in solving the **correspondence problem**: i.e.the determination of where the pixels of an image at time t are in the image at time t+1.



Figure 3.4: Dense Optical Flow examples given by the FlowNet2 architecture in the synthetic 'Sintel' video dataset.[52]

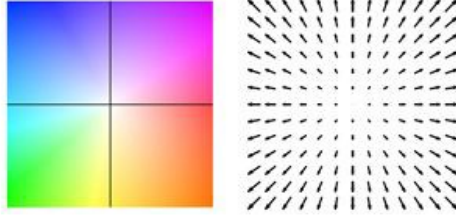


Figure 3.5: The displacement of every pixel in the top left (multi colored) image is the vector from the center of the square to that particular pixel, as indicated by the image on the right. This means at the middle of the image i.e. the white portion, indicates no optical flow, in the blue quadrant, it indicates flow to the left and to the top, the greater the shade of blue the greater the magnitude of the the vector. This applies to the other quadrants as well where the more intense the color the greater the magnitude of flow to that point.[52]

In this work, when an estimation of the Optical Flow is needed, we employ a variation of the popular Flownet [24] architecture: FlowNet2 [52]. Following, we briefly explain its main components:

The first one is **FlowNetS**. It has an Encoder-Decoder structure. This means data are spatially compressed in a contractive part of the network and then refined in an expanding part. Two sequentially adjacent input images are stacked together and fed to the network. The image pair is then processed and motion information is extracted.

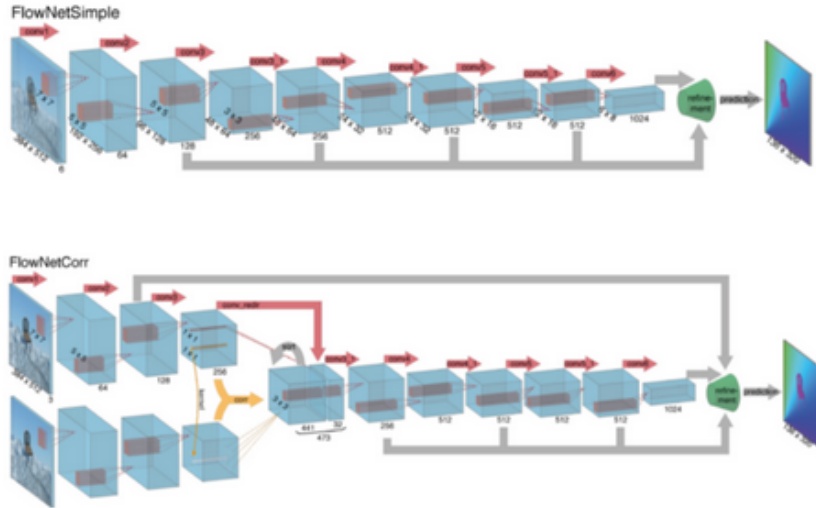


Figure 3.6: An overview of the FlowNet Simple (FlowNetS) (top) and the FlowNet Correlation (FlowNetC) (bottom) architectural components of FlowNet2.[30]

FlowNetC stands for Flownet Correlated. A visual summary can be found in fig3.6. Compared with FlowNetS, FlowNetCorr first produces representations of the two images separately, and then combines them together in the “Correlation layer”, and learn the higher representation together. The **Correlation layer** is used to perform multiplicative patch comparisons between two feature maps. More specifically, given two multi-channel feature maps \mathbf{f}_1 , \mathbf{f}_2 , with w , h , and c being their width, height and number of channels. The “Correlation” of two patches centered at \mathbf{x}_1 in the first map and \mathbf{x}_2 in the second map is then defined as:

$$c(\mathbf{x}_1, \mathbf{x}_2) = \sum_{o \in [-k, k] \times [-k, k]} \langle \mathbf{f}_1(\mathbf{x}_1 + o), \mathbf{f}_2(\mathbf{x}_2 + o) \rangle \quad (3.19)$$

where \mathbf{x}_1 and \mathbf{x}_2 are the center of the first map and the second map respectively, and the square space patch of size $K = 2k+1$. Afterwards, the feature map from \mathbf{f}_1 is concatenated with the output using a convolution layer.

However, after a series of convolution layers and pooling layers, resolution has been reduced. Thus, the authors of [52] have refined the coarse pooled representation by Deconvolutional layers (see Section 3.3.5), consisting of unpooling and upconvolution. After upconvolutioning the feature maps, the authors concatenate it with corresponding feature maps and an upsampled coarse flow prediction. Therefore, both of the two architectures employ convolutional refinements.

A refinement layer is as follows:

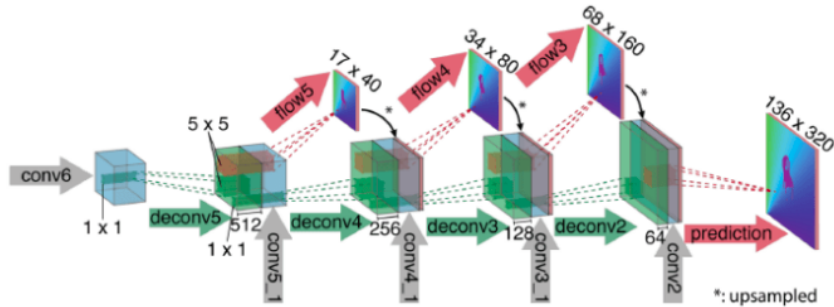


Figure 3.7: Explaining the function of the refining submodule.[30]

In this subsection of the network, it performs a series of Deconvolutions (see Section 3.3.5) in order to increase the image resolution which means it's the expansive part of the network i.e. the decoder. This is done by combining coarse feature maps (from an earlier part of the network) with the fine local information provided in lower level feature maps.

FlowNet2.0

Combining all the above ideas, the FlowNet 2.0 architecture is created where the visual representation is:

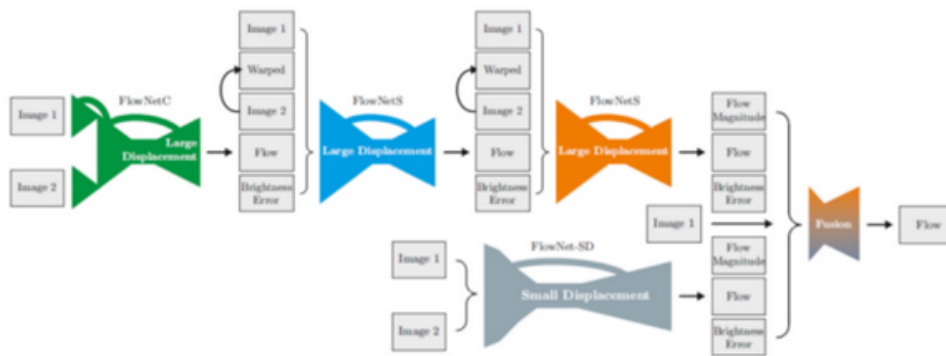


Figure 3.8: Overview of the overall FlowNet2.0 architecture [52].

From the above final FlowNet 2.0 architecture diagram we can see the very first part of the stack is the FlowNetC network, which takes in 2 images and is designed to detect large displacements. The resulting flow field is applied to the second image via warping (see below) and this, together with Image 1, is fed to the following FlowNetS network which calculates large Flow Displacements. The brightness error which is the difference between the warped image and the original image is also passed into FlowNetS. This combination of FlowNetC and FlowNetS is called **FlowNetCS**. This is then repeated with a second FlowNetS network which produces a flow field and flow magnitude. This combination is called **FlowNetCSS**, and is one possible variant of the FlowNet 2.0 network.

FlowNetCSS was introduced in the [52] and the main idea behind it is that optical flow estimations can be greatly improved by stacking the networks. A second stream of the network contains FlowNetSD which is fed with the original Image 1 and Image 2. The **FlowNetSD** network stands for FlowNet Small Displacements. In [52], it was found that despite stacking, the network was still unable to accurately

produce flow fields for small motions thus resulted in a lot of noise. Thus a variation of the FlowNetS was added to the network with some slight modifications done in order to capture smaller motions, these changes included changing the kernel size as well as adding convolutions during up convolutions.

The resulting flows from FlowNetCSS and FlowNetD are fused with the main branches flow in a fusion network to produce the final flow field.

Warping: Looking at the first FlowNetC network, it computes an optical flow. The optical flow produced by FlowNetC is “applied” to the second image to shift the image according to the optical flow field so as to try match Image 1. This new Image 2 is then fed to the following FlowNetS layer. This way the network in the stack can focus on remaining increment between Image 1 & Image 2.

Last but not least, here, we have to note that the FlowNet2 architecture was employed whenever a dense Optical Flow estimation was needed in this work, for the two following reasons:

1. Firstly, it provides one of the best accuracy-speed tradeoff available in the field and generalizes relatively well to multiple, different domains.
2. However, this is something that is also provided by more recent CNN variations trained on Optical Flow estimations. What was unique with FlowNet2 is that we aimed to use synthetic data in our approach (specifically, small patches with one object centered in the scene and various distractors (background elements/occluders) also present in the scene) and this version was the only one which (i.e. its weights) was available online and, most importantly it was pretrained on a similr-looking synthetic dataset to ours: the “FlyingThings3D” one, [24], which skyrocketed its performance w.r.t. other Optical Flow Estimators (both online and offline) we experimented with.

3.2.4 k-Means Clustering algorithm

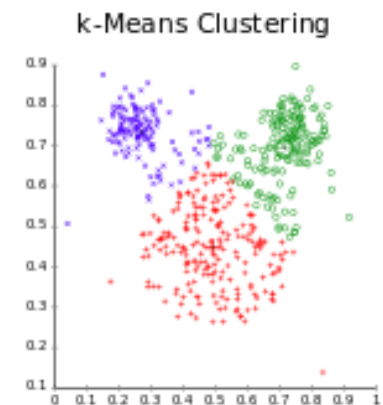


Figure 3.9: A k-Means clustering example.[158]

K-Means Clustering is an unsupervised learning algorithm that aims to partition N observations into K clusters, in which, each observation belongs to the cluster with the nearest mean, serving as a prototype of the cluster. The K-means Clustering algorithm is used to find groups which have not been explicitly labeled in the data and make better decisions. Once the algorithm has been run and the groups are defined, any new data can be easily assigned to the most relevant group.

It is an iterative algorithm that repeats the next two steps until a convergence criterion is satisfied:

- **Cluster assignment:** We assign each data point to the cluster which is closer, i.e. its centroid has the minimum (Euclidean) distance from that data point.
- **Move centroid step:** We calculate as the new centroid for each cluster, the average of the points contained in the cluster.

K is either chosen randomly or given specific initial starting points by the user.

3.3 Deep Learning

3.3.1 Artificial Neuron Model - Rosenblatt's Perceptron

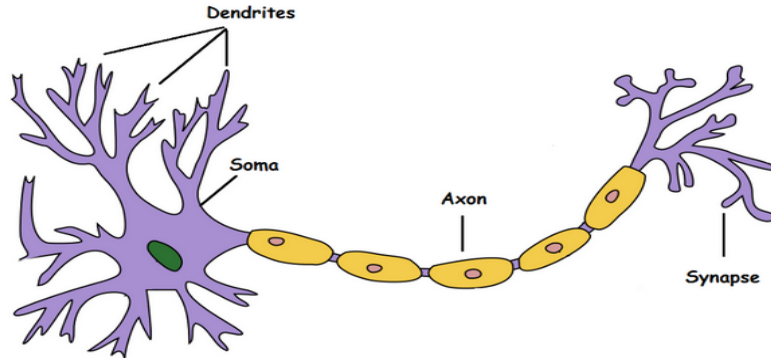


Figure 3.10: The biological neuron model that inspired McCulloch and Pitts Cookie Jar's Artificial Neuron.[85]

The most fundamental unit of Deep Neural Networks is the Artificial Neuron/Perceptron that was designed by Rosenblatt [114] and mimics the functionality of a biological neuron model that was created in 1943 by McCulloch and Pitts [85]. As it is shown in fig.3.10, a simplified model of a real-life neuron receives an input signal from the sense organs that interact with the environment, using its “Dendrites”, it processes it at the “Soma” and it transmits the output, through the “Axon” to the other neurons, using the connection between them, the “Synapses”. Practically, the neuron accumulates input information until a threshold is passed. When this happens, it lets it flood the “Axon” resulting it to be passed as input to the rest of the neurons of the nervous system.

The computational system that Rosenblat created is inspired by this procedure. It is divided by two parts: the first one, g , aggregates all the input signals from the surrounding neurons, each one multiplied by a corresponding weight, whose values is learned to be optimal with regards to the input data:

$$g(x_1, x_2, \dots, x_n) = \sum_{i=1}^n w_i x_i + w_0 x_0 = \mathbf{w}^T \mathbf{x} + w_0 x_0 \quad (3.20)$$

and the second one, f , takes the decision of letting the total information, g , to pass to the rest of the neurons according to its values and that is why f is called the Activation Function.

The activation function plays the integral role of ensuring the output is mapped between required values such as (0,1) or (-1,1) and it does so as follows: if g is smaller than a threshold, then the f 's output is 0 (equiv. -1), while if g exceeds it, it is 1 (equiv. 1), giving a non-linear form to f . Because of the fact that the linear combination of linear functions is another linear function, a non-linear activation function like that is essential in giving the Perceptron the ability to simulate functions that are more complex than first-order polynomials.

$$y = f(g) \quad (3.21)$$

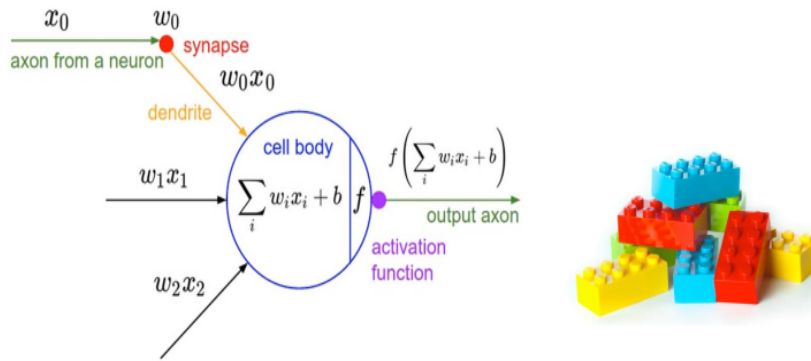


Figure 3.11: The Artificial Neuron model.[114]

Perceptrons are very useful for classifying data sets that are linearly separable. However, they encounter serious limitations with data sets that do not conform to this pattern.

3.3.2 Multi-Layer Perceptron

Such artificial neuron duplicates are used as fully interconnected building blocks (fig.3.11) to create a hierarchical architecture called MultiLayer Perceptrons or Neural Networks. They are consisted of an input, one or more hidden, and an output, layers with the information flowing from the former to the latter in a “forward” way. If the number of hidden layers exceeds 3, they are called Deep Neural Networks (DNNs). In simple words, MLPs break the restriction of linearly separable data.

Their structure as a hierarchy of fully connected non-linear functions of their input has shown to be extremely versatile in solving a wide variety of data-dependent problems, with the motivation for such approaches given by the ‘Universal approximation theorem’(Cybenko,1981 [19]):

“A Feed-Forward Network with a single hidden layer containing a finite number of neurons can approximate continuous functions on compact subsets of \mathbb{R}^n , under mild assumptions on the activation function.”

The idea above generalizes for functions on intersections of compact subsets of \mathbb{R}^n (2 hidden layers) and unions of intersections of compact subsets of \mathbb{R}^n (3 or more hidden layers), which makes the Neural Networks to be considered universal function approximators.

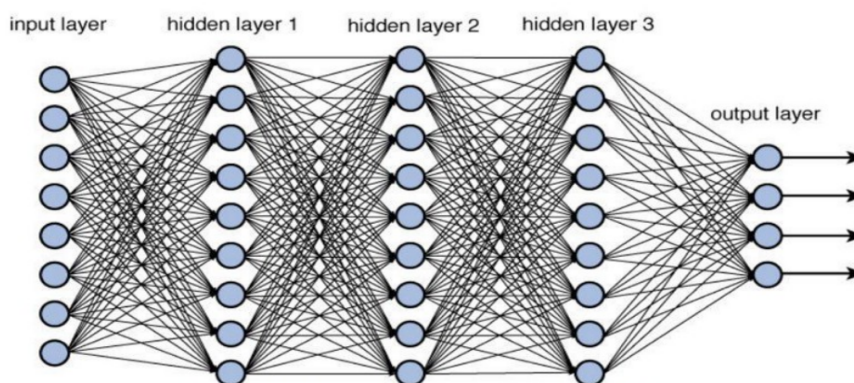


Figure 3.12: A Deep Neural Network (or else Multi Layer Perceptron) with an input, an output and 3 hidden layers[19].

So far, we have explained the forward pass of a DNN. Nevertheless, we have not yet discussed the most important part of a DNN algorithm: how the network’s weights and biases are updated. This is done with the aim of minimizing a cost function via a Gradient Descent-based optimization scheme (see Section 3.3.21) using the gradients w.r.t every Network weight to adjust how much these learnable parameters are changed. These gradients are calculated during a backward pass of the Network following the Error BackPropagation algorithm.

3.3.3 BackPropagation Algorithm

BackPropagation is a learning technique for neural networks (and every other Direct Acyclic Graph) that calculates the gradient of descent for weighting different variables. When an error occurs, the algorithm calculates the gradient of the error function, adjusted by the network's various weights. The gradient for the final layer of weights is calculated first, with the first layer's gradient of weights calculated last. Partial calculations of the gradient from one layer are reused to determine the gradient for the previous layer using the Leibniz's chain rule.

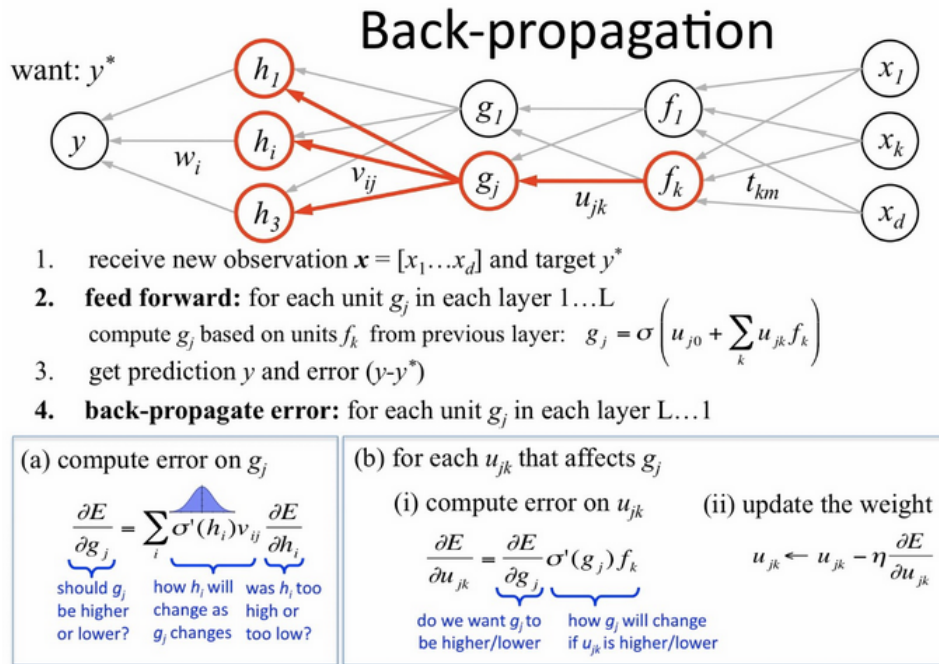


Figure 3.13: A graphical representation of the Backpropagation error pass in a Neural architecture.[66]

Training Terminology:

- **Epoch:** One epoch is completed when an entire dataset is passed forward and backward only once through the neural network.
- **Batch Size B:** Number of training examples to utilize in one iteration.
- **Batch or Iteration:** A training set of 1000 examples, with a batch size of 20 will take 50 iterations/batches to complete one epoch.

Exploding and Vanishing Gradient Problem: We know that, when we have more hidden layers, our model tends to perform better. When we perform back propagation, if the gradients become larger and larger or smaller and smaller, then the weights of the neurons in the earlier stages change too much or do not change at all. This problem is called **Exploding/Vanishing gradients problem**.

Gradient Clipping:

When gradients explode, they could become NaN because of the numerical overflow or we might see irregular oscillations in training cost when we plot the learning curve. A solution to fix this is to apply **gradient clipping**; which places a predefined threshold on the gradients to prevent it from getting too large. Please note that gradient clipping only alters only the length of the gradient and preserves its direction.

if $\|g\| > threshold$

$$g \leftarrow \frac{threshold \times g}{\|g\|}$$

where: g is the gradient and

$\|g\|$ is the norm of the gradient

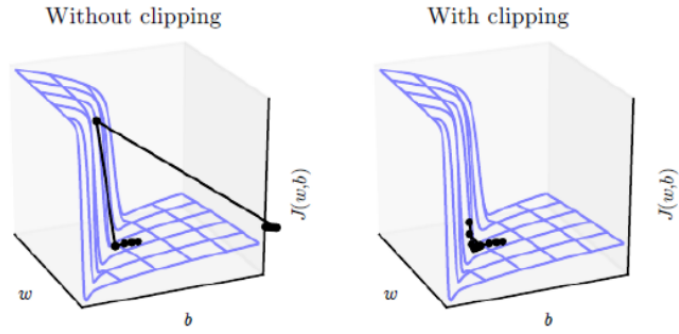


Figure 3.14: An abbreviated visualization of the Gradient Clipping algorithm[32].

3.3.4 Activation Functions

The first activation function that we saw in the neuron model is the Step function (fig.3.15) which serves as a binary classifier threshold classifier.

$$f(x) = \begin{cases} 0, & x < 0 \\ 1, & x \geq 0 \end{cases} \quad (3.22)$$

Sigmoid and Tanh function

However, it is not smooth, and thus incompatible with the Back Propagation Algorithm. So, our first thought is to substitute it with its smooth version: the **Sigmoid function** when we want the neuron activations to lie between $[0,1]$:

$$f(x) = \frac{1}{1 + e^{-x}} \quad (3.23)$$

or the **Tanh function** for $[-1,1]$:

$$f(x) = \frac{\sinh x}{\cosh x} = \frac{e^x - e^{-x}}{e^x + e^{-x}} = \frac{e^{2x} + 1}{e^{2x} - 1} = \frac{2}{1 + e^{-2x}} - 1. \quad (3.24)$$

Both their derivatives:

$$\frac{dSigmoid(x)}{dx} = \frac{e^{-x}}{1 + e^{-x}} \quad (3.25)$$

and

$$\frac{dTanh(x)}{dx} = \frac{4e^{-2x}}{(1 + e^{-2x})^2} \quad (3.26)$$

have a denominator which tends to 0 as their input increases to ∞ . As a result, the backward gradients are smaller at each consecutive layer. For example, for Sigmoid, it has been shown that every layer's gradients are only $\frac{1}{4}$ of the previous one's. This causes an extreme Vanishing gradient problem which must be dealt with. Moreover, calculating the exponentials at the corresponding denominators is a source of extreme computational burden that makes the training process rather slow.

Rectified Linear Unit (ReLU) function

To this end, **Rectified Linear Unit** has arisen as the go-to activation function:

$$f(x) = \begin{cases} 0, & x < 0 \\ x, & x \geq 0 \end{cases} \quad (3.27)$$

When a neuron's activation is positive, it passes as it is to the next, but when it is negative it gets dropped out. It is smooth, computationally simple and its derivative is 1 for a neuron that activated,

tackling the Vanishing gradient problem. However, due to the fact that negative neuron activations are cut out it gives rise to the **Dead neuron problem**: where, because of their inactivation, these neurons will not be trained at all if that pattern repeats itself in many minibatches.

Parametric ReLU (PReLU) function

In order to handle the Dead neuron problem, we may substitute ReLU with **Parametric ReLU (PReLU)**, which carries over all the ReLU's advantages but also allows negative neuron activations to pass to the next layer after being scaled down by a parameter a . Small negative values (zeroed out in ReLU) may still be relevant for capturing patterns underlying the data, whereas a gives the chance to tune large negative values contribution.

$$f(x) = \begin{cases} ax, & x < 0 \\ x, & x \geq 0 \end{cases} \quad (3.28)$$

Leaky ReLU function

A special case of PReLU for $a=0.01$ is called **Leaky ReLU**:

$$f(x) = \begin{cases} 0.01x, & x < 0 \\ x, & x \geq 0 \end{cases} \quad (3.29)$$

ELU

An alternative to the Leaky ReLU is **ELU**:

$$f(x) = \begin{cases} a(e^x - 1), & x < 0 \\ x, & x \geq 0 \end{cases} \quad (3.30)$$

It has all the advantages of LeakyReLU plus the fact that it saturates at big negative values (bigger than $-a$) preventing Exploding Gradients, while Leaky ReLU in its linear fashion does not accomplish completely.








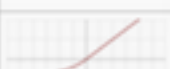

Base	Plot	Equation	Derivative
Identity		$f(x) = x$	$f'(x) = 1$
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x \neq 0 \\ ? & \text{for } x = 0 \end{cases}$
Logistic (a.k.a Soft step)		$f(x) = \frac{1}{1 + e^{-x}}$	$f'(x) = f(x)(1 - f(x))$
Tanh		$f(x) = \tanh(x) = \frac{2}{1 + e^{-2x}} - 1$	$f'(x) = 1 - f(x)^2$
ArcTan		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2 + 1}$
Rectified Linear Unit (ReLU)		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Parameteric Rectified Linear Unit (PReLU) [2]		$f(x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Exponential Linear Unit (ELU) [3]		$f(x) = \begin{cases} \alpha(e^x - 1) & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} f(x) + \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
SoftPlus		$f(x) = \log_e(1 + e^x)$	$f'(x) = \frac{1}{1 + e^{-x}}$

Figure 3.15: A pivot table of Activation function choices, their diagram and derivatives[153].

Swish function

Recently, a new activation function called **Swish** [107] was developed. It differs from the above as it is the only one which is not monotonous. This means that according to its β value it will be able keep small negative integrations to solve the Dead neuron problem (as PReLU and ELU do), zero-out the bigger ones to avoid Exploding gradients and have a non-decaying derivative in order to avoid Vanishing gradients. So, extra care should be given to its β parameter choice, as $\beta = 0$ results in a Linear function, while as $\beta \rightarrow \infty$, Swish converges to ReLU. Usually, β is chosen to be 1.

$$f(x) = 2x \cdot \text{Sigmoid}(\beta x) = \frac{2x}{1 + e^{-\beta x}} \quad (3.31)$$

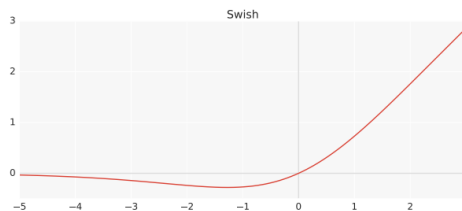


Figure 3.16: The **Swish** activation function[107].

Softmax function

Softmax activation function

$$f(x) = \frac{e^{x_i}}{\sum_{i=1}^K e^{x_i}}, \quad (3.32)$$

for $i=1, \dots, K$,

seems similar to the Sigmoid one at the first glance, as it limits its output values between 0 and 1. However, it has a further physical meaning that derives from the fact that if we add up all the activations of this kind of all the neurons of a layer, that sum is exactly 1. The aforementioned observation means that the Softmax function allows probabilistic interpretation of its input as it determines the probability of a neuron activation belonging to a particular class.

3.3.5 Convolutional Neural Networks

Fully Connected NN layers, when dealing with images present two distinct weaknesses:

- Due to the fact that every input, intermediate and final-layer neuron is interconnected with every unit of the previous layer, using Matrix Multiplication to formulate cross-layer interactions, even input images of the smallest resolutions make the total parameter number skyrocket and the training computational burden unfeasible.
- Each element of a layer Weight Matrix of a NN processes a single previous-layer unit and thus it is used only once per pass. A physical repercussion of this is that these weights handle every piece of information source (input or previous layer activation) in the same way, which makes their activations commutable. One step further, that deters them from preserving and propelling spatial properties of the images.

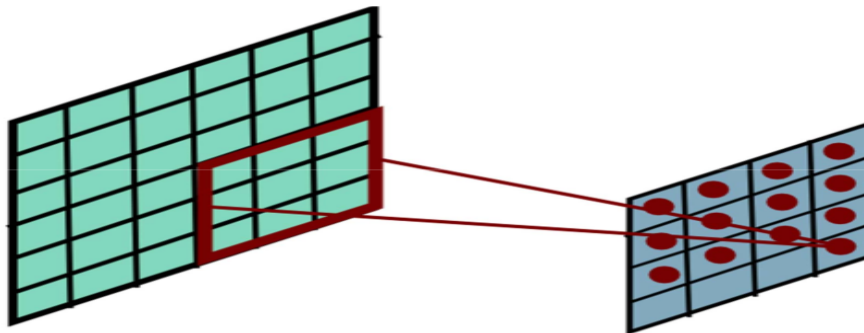


Figure 3.17: A single Convolutional filter activation map as it is convolved with an input image[67]

For the above reasons, local Convolutional kernels emerge as a more favorable building block choice. These filters only have a limited number of learnable parameters, smaller than the input size, and they are transposed throughout the layer input pixels by a spatial step called **stride**. Their aim is to detect patterns within their **Receptive field**, i.e. the kernel cardinality. This is called **Local connectivity with Parameter Sharing** and it connects every layer neurons to a subset of the activations of the previous layer.

Compared to Fully Connected NN weights, Convolutional kernels:

- **Maintain spatial structure** from layer to layer, i.e. closely located pixels have close mappings.
- **Share** the same **parameters** to a multiplicity of model functions, making them computationally, memory and statistically efficient. Consequentially, they process images in a locally targeted manner and extract visual characteristics whose range depends on the filter's receptive dimension.
- Are **translationally equivariant in 2D**. In simple words, a translation of the input image will result in a translation of the Convolutional layer activation by the same amount, detecting, this way,

specific visual features no matter their location in the image. In simple words, each convolutional filter responds in the same way to the same input pattern, wherever it may appear in the image. If the input pattern is laterally moved the output feature will be laterally moved by the same amount of translation.



Figure 3.18: Figure 3.19: A “Find Waldo” terrain, from a popular game for children. The “Waldo” pattern-template. A “Waldo detection” hypothetical CNN would process each location, in its filter size dimension, in the same manner and would locate the “Waldo” pattern-template wherever it may be in the picture. Due to its translational equivariance, it would locate it in its exact spot (instead of, for example, appointing it to the center of the image no matter its initial position, something that a “translational invariant” approach would have done, instead).

Convolutional filter activations are called **Activation maps**. Every convolutional layer is consisted of many such filters. By stacking their Activation maps along the depth dimension, we create a **Feature map** volume. That is the output of a Convolutional layer.

Mathematically, the operation described above is not exactly a convolution, as it is defined in Signal Processing, as the kernel is not flipped by 180° before applied to the input. The term is rather a convention because the kernel is run throughout the image in the same way as a convolution. The most preferable mathematical term seems to be **Cross Correlation**:

$$F_{(c)}^{(l)}[i, j] = \sum_{u=-w}^w \sum_{v=-h}^h k[u, v] F_{(c)}^{(l-1)}[i + u, j + v], \quad (3.33)$$

with k : the Convolutional kernel and F : the corresponding layer feature map.

In the following figures, the thought process of evolving from Fully Connected to Convolutional Filters is described graphically:

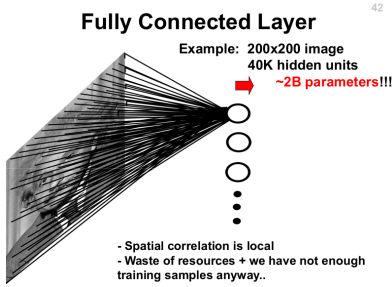


Figure 3.20: In case of a Fully Connected layer applied to an image, even for a small resolution of 200×200 , matching a pixel to every neuron results in an architecture with billions of parameters.[66]

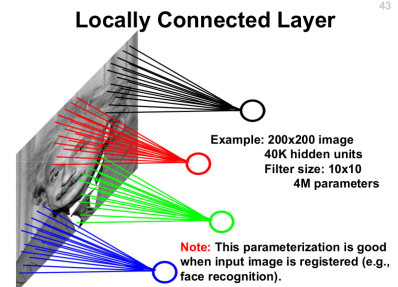


Figure 3.21: For this reason, we substitute Fully Connected layers with Local ones that cover a specific region of the image.[66]

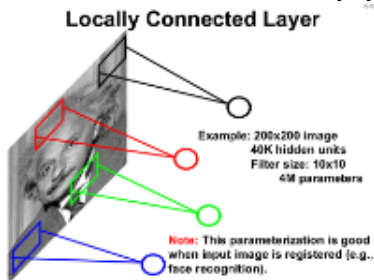


Figure 3.22: One step further, instead for interconnected pixel-neuron connections we just use a kernel of few learnable weights.[66]

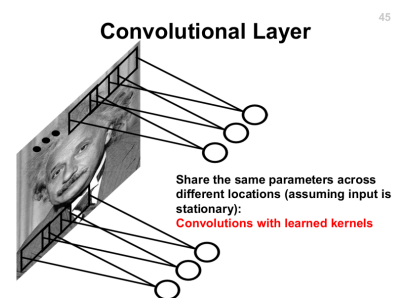


Figure 3.23: Finally, different kernels per region are dropped out and the same filters transit on the image to transfuse translational equivariance.[66]

Maintaining the spatial image properties from layer to layer gives the opportunity to a CNN to learn to identify simple, limited size patterns in the first layers such as edges, corners etc. and, climbing up the layer hierarchy, to learn to combine them to more complex representations. As a result, CNNs have the ability to replace the intermediate handcrafted image feature extractors (see fig.3.24) with a data-driven feature extraction architecture that is trained end-to-end to solve a specific problem and thus their features are the optimal for it. Usually, after their last Convolutional layer the final feature representation is suitably vectorized and fed to a Fully Connected NN that takes on the classification/regression task.

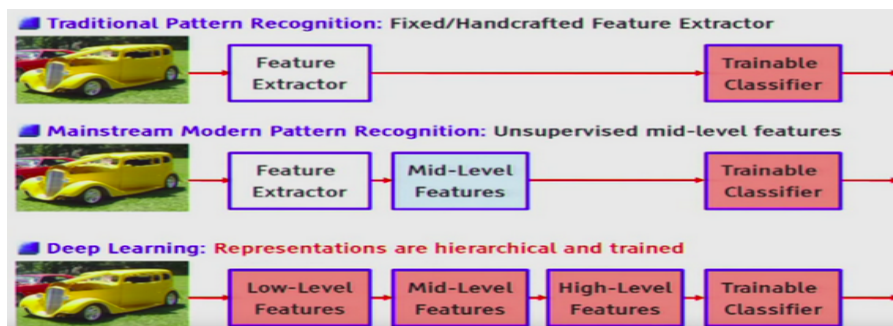


Figure 3.24: The evolution of Neural Networks from just a final component that processes handcrafted features, that are constructed in an unsupervised, unified and problem-agnostic way, to a full end-to-end data driven feature extraction and decision making tool that hierarchically extracts progressively complex features as combinations of the previous ones.[66]

1 × 1 Convolutions

A special case that should be discussed are Convolutional layers with filter size of 1. **1 × 1 Convolutions**, as they are called, function as a Fully connected layer of small dimension (as their weights consist a vector rather than a matrix) applied to the Feature map, in order to add extra non-linearity (a.k.a. representation power) to the network without increasing its parameters.

Like Pooling layers (see Section 3.3.8), that downsample the spatial dimensions of a Feature map, 1×1 Convolutions have the ability to shrink the channel dimension in order to save computations.

Overfitting-Underfitting:

- **Overfitting:** When the complexity of the model is too high, compared to the data that it is trying fit on, the model is said to **Overfit**. In other words, with increasing model complexity, the model tends to fit the Noise present in data (e.g. Outliers). The model learns training data too well and hence fails to generalize. Overfitting occurs for a model with High Variance and Low Bias.
- **Underfitting:** When the complexity of the model is too low for it to learn the training data that is given to as input, the model is said to **Underfit**. In other words, the excessively simple model fails to “learn” the intricate patterns and underlying trends of the given dataset. Underfitting occurs for a model with Low Variance and High Bias.

3.3.6 Fire modules

A **Fire module** (first introduced in [51]) is comprised of:

- a **squeeze convolution layer** (which has only 1×1 Convolutional filters) fed into
- an **expand layer** of 1×1 filters and
- an **expand layer** of 3×3 filters

that are concatenated to compose the output Feature map of the layer (see 3.25).

Transforming the feature representation to a lower dimensional space in the intermediate **squeeze layer** that functions as a bottleneck causes a reduction of the Convolutional layer parameters which makes it computationally more efficient without much of a performance dropout (as seen in [51]).

Since it is not clear how much we benefit from an amount of parameters perspective let’s give the following example: a 3×3 Convolutional layer with input feature dimension of 96 and output of 196 has $196 \times 96 \times 3 \times 3 = 139,968$ parameters. On the contrary if we substitute it with a Fire module of the following fashion: $96 \rightarrow 24 \rightarrow (96 + 96)$, the total amount of parameters that we will need is $(24 \times 96 \times 1 \times 1) + (96 \times 24 \times 1 \times 1) + (24 \times 96 \times 3 \times 3) = 2,304 + 2,304 + 20,736 = 25,344$ i.e. only 18% of the initial ones.

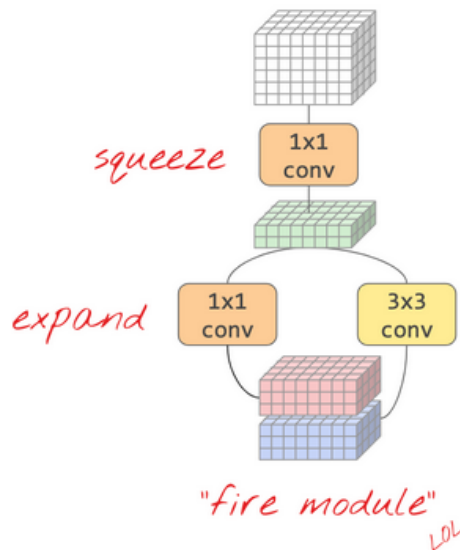


Figure 3.25: The **Fire module** of [51]. At first, the input feature map is transformed to a lower dimensional one (via a 1×1 Convolution) to reduce the required parameters followed by an expansion layer that processes the squeezed Feature map with a stack of 1×1 and 3×3 Convolutions.

3.3.7 Batch Normalization

If the activations of the neurons of the previous layer diverge, a potential **Exploding** or **Vanishing** gradient problem may occur during training. To avoid it, we wish to standardize the feature maps of

each layer in order to make them have the same scale. To this end, exactly as the famous Data Science practice of Standardization, we use Batch Normalization. First and foremost, we calculate the mean and standard deviation of the activations of an input mini-batch (see Subsection 3.3.21). Then we standardize the activations of the previous layer using them to scale them down to a Gaussian with mean equal to 0 and standard deviation equal to 1. However, we recognize that this may not be an accurate distribution for the feature maps description, so, we further multiply them by a learnable parameter γ and add a learnable parameter β . The method is called **Batch Normalization** because those parameters are optimized using Minibatch Gradient Descent.

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1\dots m}\}$;

Parameters to be learned: γ, β

Output: $\{y_i = \text{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m} \sum_{i=1}^m x_i \quad // \text{ mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m} \sum_{i=1}^m (x_i - \mu_{\mathcal{B}})^2 \quad // \text{ mini-batch variance}$$

$$\hat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \quad // \text{ normalize}$$

$$y_i \leftarrow \gamma \hat{x}_i + \beta \equiv \text{BN}_{\gamma,\beta}(x_i) \quad // \text{ scale and shift}$$

During training time, we save the **running average** (or Exponential Weighted) Average of the mean and standard deviations calculated across minibatches. At the end, they are frozen and used during the Network's inference.

Batch Normalization is a very effective way to stabilize the training process, as it makes it robust to the hyperparameter selection and makes the Network converge better in terms of speed and inference quality. In practice, it makes deeper layers more robust to the previous layer's activations perturbations and minimizes their Covariate shift. Furthermore, since the mean and standard deviation estimations are performed on minibatches they tend to be noisy as well as the trained γ, β . This means that Batch Normalization also slightly regularizes the Network's weights by adding noise augmentation during the training process.

3.3.8 Pooling Layers

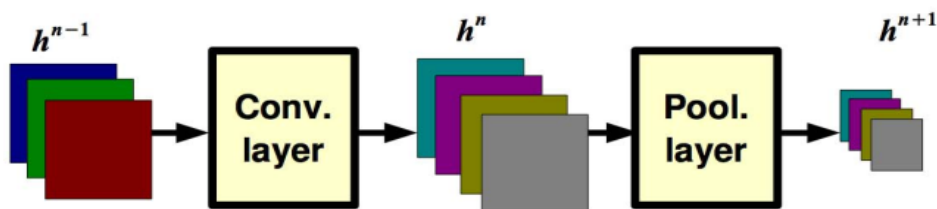


Figure 3.26: A sequence of applying a convolutional and a pooling layer to the feature maps of the previous layer.[66]

They perform spatial subsampling of the Feature map which is outputted from the current layer. In detail, most commonly, they convolve a 2×2 Pooling Filter with each individual depth slice of the Feature map and they apply an operation resulting in only one pixel per such window. That reduces spatial resolution and thus computations per layer, allowing CNN architectures to add more layers (at least as much as ever-reducing initial spatial resolution allows). They, also, insert translational and noise invariance in the equivalent neighbourhoods of their filters, reducing Overfitting. For the operation choice, average/max/min pooling have been tested among others with the Max Pooling being shown

empirically as the most robust. It is logical as it outputs the highest activation of the previous layer in the region under study.

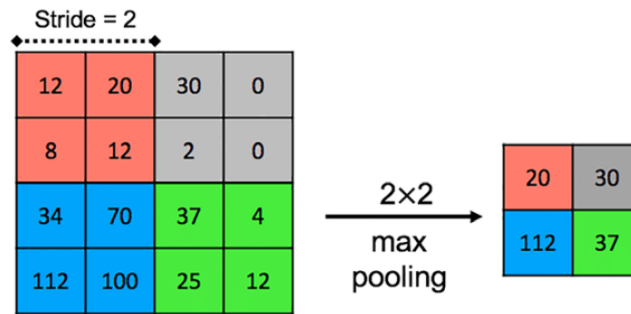


Figure 3.27: A graphical example of the application of a Max Pooling filter to a depth slice spatial neighbourhood.

3.3.9 Dropout

In order to avoid Overfitting, we assign a Bernoulli random variable with a probability p to each Network node. During each mini-batch (3.3.21) sampling of the training process, if that node is assigned 1 (following p), it is kept in the Network, otherwise, for the 0 case, it is temporarily rejected and that minibatch's training is performed without it. In practice, this creates a variety of different network architectures, with different neurons missing each time. Thus, **Dropout** functions as augmentation causing the CNN's weights to be more robust to perturbations of the previous layer's activation values (most commonly appeared when the Network is called to generalize to previously unseen conditions).

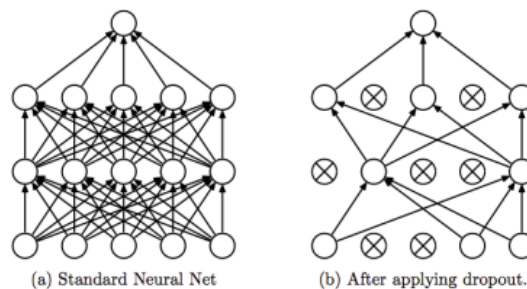


Figure 3.28: Neural Network neuron Dropout, with a probability p , during model training. [121]

3.3.10 Weight Initialization

Initializing the network with the right weights is very important if you want to ease the training procedure and, ultimately, reach a lower loss function minimum. We need to make sure that the weights are in a reasonable range before we start training the network to avoid either Vanishing or Exploding gradients phenomena that harm NNs' convergence. Our first thought would be random initialization from, let's say, a normalized Gaussian distribution. But, in this case, excluding vanishing gradients and exploding gradients is far from guaranteed. That is because, even after standardizing previous layers' outputs to a Gaussian Distribution with mean of 0 and standard deviation of 1 (e.g. with Batch Normalization (Section 3.3.7)), hierarchically passing from one layer to the other may cause the outcome of matrix multiplications/convolutions to skyrocket or vanish with similar results to the BackPropagation gradients, since no measure has been taken to avoid it as much as possible.

Xavier Initialization

For neurons with symmetric non-linear function (e.g. Tanh), we keep the idea of assigning the weights from a Gaussian distribution. Obviously, we would wish that this distribution would have zero mean and variance of 1 and, passing from a linear neuron to linear neuron, we want it to be the same to

ensure the gradient quality. A way to do this is, instead of sampling the weights from $\mathcal{N}(0, 1)$, to divide that variance by the number of total input plus output filters of the current layer. The former denominator term ensures that the forward layer outputs are constrained in a Gaussian distribution with standard deviation of 1 (assuming that current layers inputs $\sim \mathcal{N}(0, 1)$ after being passed through a Batch Normalization layer (Section 3.3.7)), while the latter takes care of the same thing for the Backpropagation gradients. So, the weight initialization scheme proposed by Xavier and Glorot [31] samples from a distribution with variance:

$$\text{Var}(W) = \frac{1}{N_{out}}, \quad (3.34)$$

He Initialization

Contrarily to symmetrical activation functions like Tanh, ReLU-like ones turn half of the Z-values (the negative ones) into zeros, effectively removing about half of the variance. So, we need to double the variance of the weights to compensate for it. After this observation K.He [38] proposed a weight initialization sampling from a Normal distribution with variance:

$$\text{Var}(W) = \frac{2}{N_{in} + N_{out}}. \quad (3.35)$$

3.3.11 Residual Connections

A logical initial assumption would be that deeper NN architectures would yield better performance. They may suffer from overfitting but they would represent more and more complex functions due to the increase of the overall parameters. However, experimentally, that has not shown to be the case with their performance saturating even in the training phase! Graphically, one would describe it with an image like fig.3.29([72]), where the multidimensional Loss function landscape appears to be noisy and full of abrupt local minima that trap the optimizer. Another intuitive explanation is that, besides the right hyperparameter selection of the rest of the architecture modules, BackPropagation gradients may not be vanishing completely, but their values are reduced during their journey from the latter to the former layers.

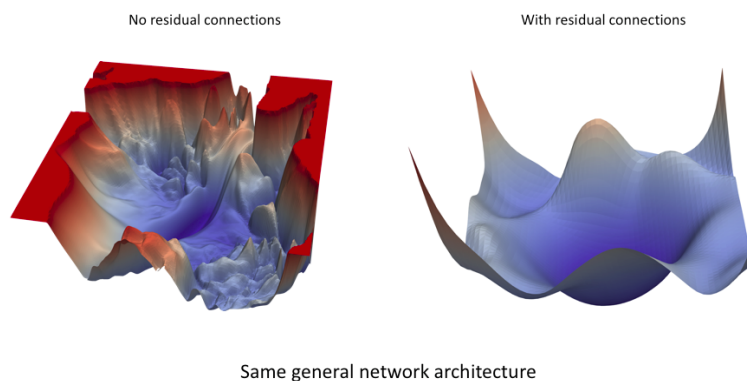


Figure 3.29: Landscape comparison between Residual and Dense connecting.[72]

With that in mind, **Residual connections** (first introduced in **ResNet** [37]) add an Identity mapping (x) to the corresponding layer activation ($F(x)$). That mapping forward propagates the input feature map and adds it to the output one:

$$H(x) = F(x) + x \quad (3.36)$$

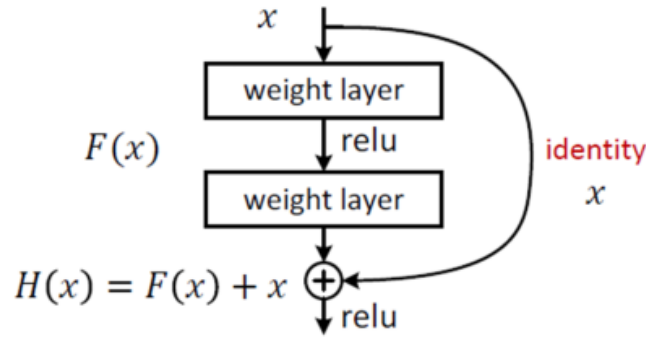


Figure 3.30: A graphical representation of a residual block of the “ResNet” architectural design.[37]

This has shown to be a catalytic idea that should be viewed by both the forward and backward perspective. In the former, the Identity map acts as an image baseline whose features are enhanced from the processed next layer output. This is why we pass from the noisy landscape on the left side of fig.3.29 to the smoother and more convex right one. We keep an intermediate baseline information to treat the layer-induced perturbation. The backward argument is that the Residual connection creates a gradient highway that bypasses the previous layers’ gradients intact and adds them with their reduced versions in an attempt to reduce the Vanishing gradient phenomenon. Indeed, this is the case and, therefore, Residual connections allow the existence of deeper and deeper NN architectures, as practice has shown [37].

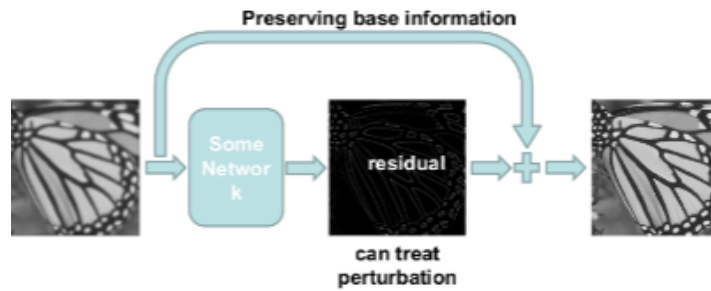


Figure 3.31: A single layer Residual connection that adds an Identity mapping to the layer’s Feature map.[37]

3.3.12 Dense Connections

Using the same argument, in DenseNet [49] (fig.3.32),

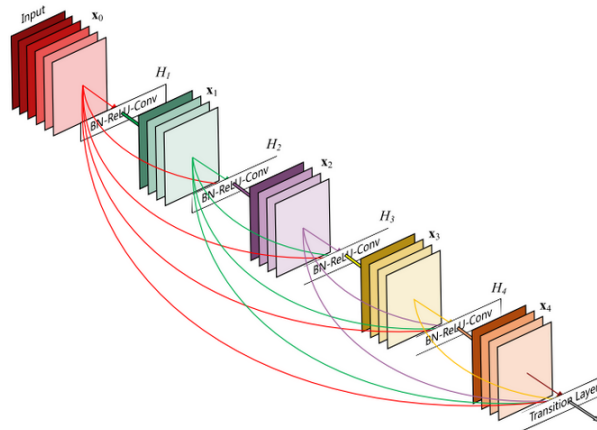


Figure 3.32: An overview of a “DenseNet” architectural module.[49]

the authors replaced Residual blocks with “**Dense Blocks**”. These are nothing more than substitutions of the addition function of ResNet with a depth-wise **Feature map concatenation**.

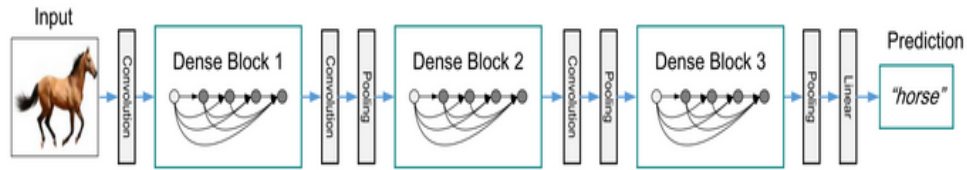


Figure 3.33: An overview of the “DenseNet” architecture. [49]

“Dense blocks” (fig.3.33) are defined between 2 consecutive Max Pooling operations for obvious spatial dimension preservation reasons. They have the drawback of being more memory and computationally hungry as concatenation increases the Feature map depth channel dimension and thus the Convolutional layer’s input weight dimension. At the same time, they are considered the state-of-the-art in the inductive task of classification in the ImageNet [68] dataset. An intuitive reason for this is that they smooth out the Loss function landscape even further helping the optimization algorithm of choice.

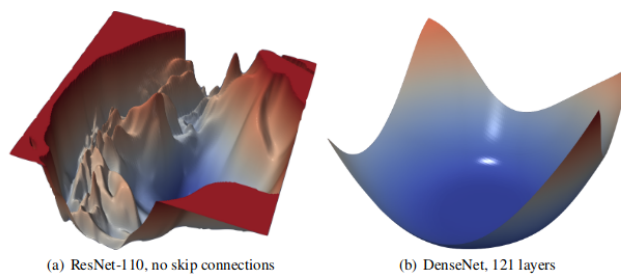


Figure 3.34: [72]

3.3.13 Deconvolutional Layers

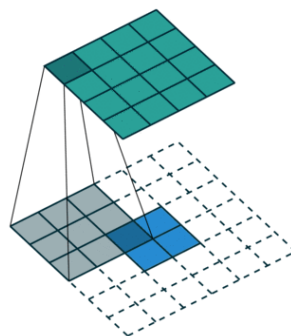


Figure 3.35: A Deconvolutional (or Transpose Convolutional) layer operation on a Feature map channel.[120]

The **Deconvolution** or **Transposed Convolution** operation forms the same connectivity as the normal convolution but in the opposite direction. It is usually used to conduct spatial upsampling, but instead of employing a predefined interpolation method, it models it with learnable weights.

Even though it is called the Transposed Convolution, it does not mean that we take some existing convolution weight matrix and use the transposed version. The need for transposed convolutions generally arises from the desire to use a transformation going in the opposite direction of a normal

convolution, i.e., from something that has the shape of the output of some convolution to something that has the shape of its input while maintaining a connectivity pattern that is compatible with said convolution.

That type of layers is predominantly used in Encoder-Decoder style of architectures, where images are transformed in a low-dimensional latent feature space by the Convolutional encoding part and then a stack of Deconvolutional layers are employed to reconstruct the original image or, more generally, to project the latent space to a higher-dimensional one.

3.3.14 Transfer Learning

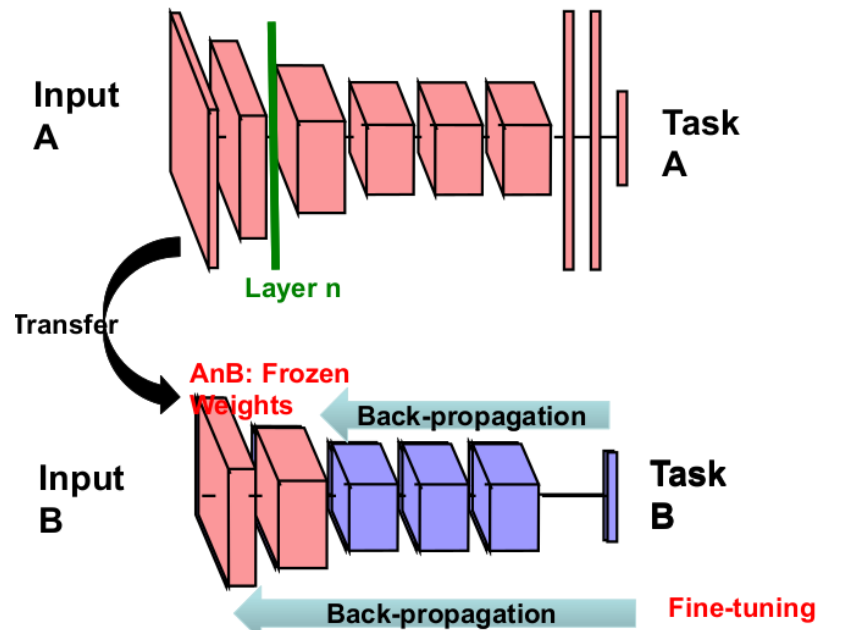


Figure 3.36: Initializing and freezing of the two first layers of the lower CNN architecture with those of the CNN architecture, above, trained on dataset A to execute task A. Then, the rest of CNN B's layer weights are trained on dataset B to execute task B, which is our task at hand[95]

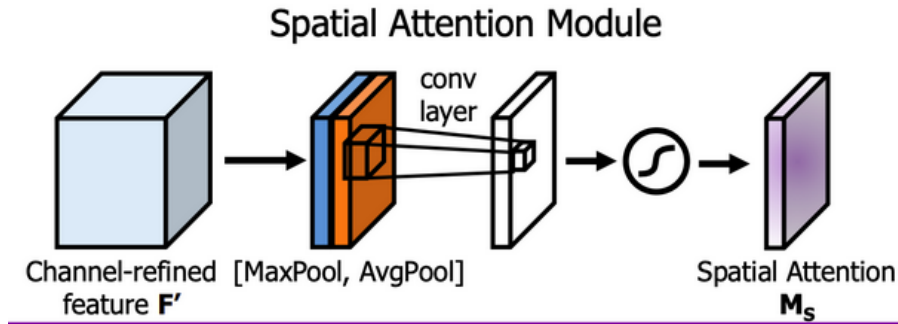
An advantage of NN architectures is that we may assign weights trained from another architecture to our own, given that the dimensions match. This is called **Transfer Learning** and gives us the chance to initialize our model with the weights of a pretrained one that has been trained on the same or another task, the same or another (usually big) dataset and exploit ad-hoc expensive trainings that have been saved in the past. Then, we have the choice of keeping all of those layers frozen and just use this NN module for inference or unfreeze as many of the final layers as we wish and finetune our architecture on our dataset and on the task at hand. The motivation for Transfer Learning is that, for computer vision tasks, the functionality of the first CNN layers is rather similar as they detect low-level features. Transfer Learning initialization tends to improve NN performance, both in terms of accuracy and robustness, as well as speeding up the training process.

3.3.15 Visual Attention



Figure 3.37: Visual imprint of the Spatial Attention module's effect on images, in another task: Visual Question answering. The model enhances its distinctive abilities by softly focusing on the image attributes that are of interest of the particular question.

Motivated by biology, we observe that human visual system does not handle every image region in the same way. Some parts of it are more important for the task at hand, while others are less so. So, we need a NN module that formulates that focus i.e. the **Visual Attention**. Basically, such a mechanism comprises of weighting of image (or Feature map, in deeper layers) pixel spatial locations by a weight of probability fashion (for **Soft Attention**) or of binary fashion (for **Hard Attention**). Here, we focus only on Soft Attention, which is a differentiable operation and can be trained using the BackPropagaion Algorithm along with the other NN modules.



A Soft Spatial Attention module consists of three steps:

- An arbitrary sequence of dedicated Convolutional layers for this task.
- A 1×1 Convolutional layer that outputs a single-channel Feature map. This map will be used as a **Weight map** $W = [w]_{i,j}$ of Soft Spatial Attention weights that will be applied to Feature maps of interest via element-wise multiplication.
- A Softmax operation (see. Section 3.3.4) that weighs every pixel of the Weight map between $[0,1]$ and satisfies the probability summation constraint.

So, every pixel $w[i,j]$, $i=1, \dots, H$, $j=1, \dots, W$ of the Weight map is of the form:

$$a[i, j] = \frac{e^{w[i,j]}}{\sum_{i'=1}^H \sum_{j'=1}^W e^{w[i',j']}} \quad (3.37)$$

In this way, we have created the Attention map $A = [a]_{i,j}$.

- The Attention map is applied to every channel of the Feature map of interest using a Hadamard product:

$$F^{(c)} = A \odot F^{(c)}, \quad (3.38)$$

$c=1, \dots, F$.

Visual attention is trained end-to-end as every other NN module without a dedicated loss function being necessary. Its presence may help its effectiveness though. That module helps the Network to

understand which features are more salient and need to be focused on. Thus, it aids in producing more enlightened decisions. Visualizing the Attention Weight map provides us explanations of the reasons the Network performs in a particular way and highlights its strengths and weaknesses.

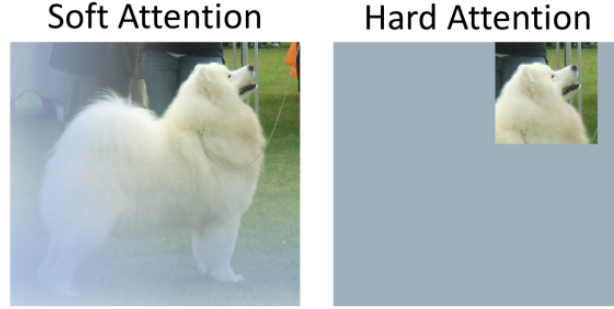


Figure 3.38: Soft vs Hard Visual Attention tradeoff.[165]

3.3.16 Gumbel Distribution - Gumbel Softmax Trick

Gumbel Softmax expands the Softmax idea using a temperature hyperparameter $T > 0$ to escape the exact Softmax probability formulation and control how much such a function may bridge between Softmax (soft-plus encoding) and Hardmax (1-hot encoding). Intuitively, this may be beneficial for an Attention module, especially when it simulates a Segmentation task in which a decision about a pixel belonging in one class or another must be made. Harder probability modeling biases values closer to 0 or to 1, possibly denoising the Attention weights of intermediate values.

Mathematically,

$$a_T[i, j] = \frac{e^{\frac{w[i, j]}{T}}}{\sum_{i'=1}^H \sum_{j'=1}^W e^{\frac{w[i', j']}{T}}} \quad (3.39)$$

for $i=1, \dots, H, j=1, \dots, W$.

- As $T \rightarrow 0$, $A_T = [a_T[i, j]]_{i, j}$ converges to a Dirac distribution (Hardmax) where the pixel attention weights tend to take values of only 0 or 1.
- As $T \rightarrow \infty$, $A_T = [a_T[i, j]]_{i, j}$ converges to a Uniform Distribution, where pixel attention weights tend to take similar values, without presenting specific peaks at high excitation regions.

3.3.17 Loss Functions

A NN's weights are learned to optimize an **Objective** or **Loss function** that measures the NN predictions in comparison to the expected outcome.

For a Regression problem, like the one we encounter in this thesis, the Loss function has the form:

$$L = \sum_{i=1}^B L_i(\hat{\mathbf{y}}_{\mathbf{n}} - \mathbf{y}_{\mathbf{n}}) = \sum_{i=1}^B L_i(\mathbf{f}(D_i; W), \mathbf{y}_{\mathbf{n}}) \rightarrow \min. \quad (3.40)$$

Specific subcases for L_i 's are discussed below.

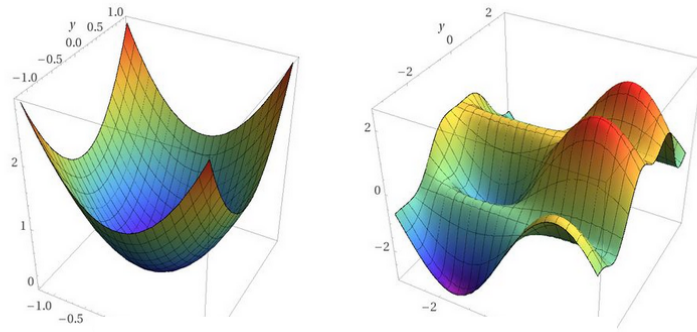


Figure 3.39: A convex and a non convex Loss function. While the former has a Global minimum that we can find, the latter may have many local ones.[1]

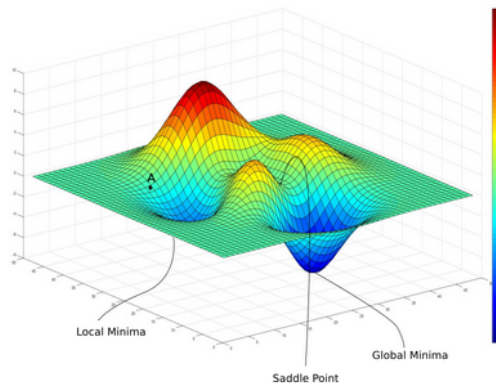


Figure 3.40: The Global and Local minima of a multivariate Loss function.

In order to handle the model overfitting problem, **Weight decay** is used:

$$L_{Overall} = L + \lambda \sum_k \sum_l W_{k,l}^2 \rightarrow \min, \tag{3.41}$$

where the regularization parameter λ is tuned to optimize the following tradeoff: during the training process the Network weights are constrained to be as small as possible, while at the same time, the main Loss function minimization increases their values (or at least some of them) to learn the training patterns.

Mean Square Error

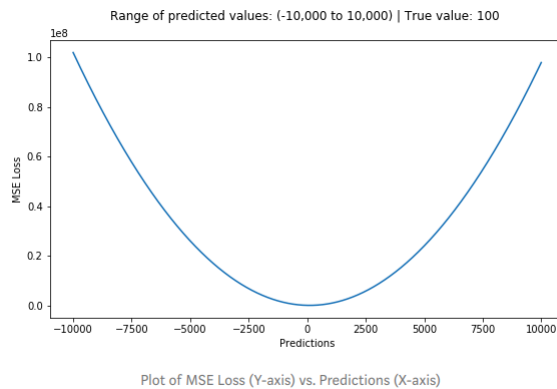


Figure 3.41: [1]

Mean Square Error (MSE) is the most commonly used regression loss function. MSE is the sum of squared distances between our target variable and predicted values.

$$L_{MSE}(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{B \cdot d} \sum_{i=1}^B \sum_{j=1}^d (y_b^{(j)} - \hat{y}_b^{(j)})^2 \quad (3.42)$$

Mean Absolute Error

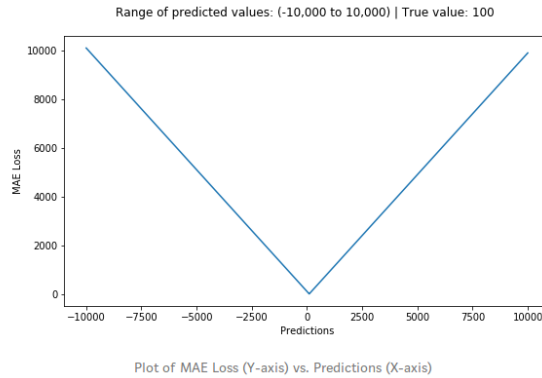


Figure 3.42: [1]

The **Mean Absolute Error (MAE)** is another loss function used for regression models. MAE is the sum of absolute differences between our target and predicted variables.

$$L_{MAE}(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{Bd} \sum_{i=1}^B \sum_{j=1}^d |y_b^{(j)} - \hat{y}_b^{(j)}| \quad (3.43)$$

MSE is more sensitive to outliers than MAE as it weighs them more, compared to samples inducing smaller error, due to its squared term. This will be done at the expense of other common examples. Mathematically, minimizing MSE minimizes the mean of all target values while minimizing MAE that prediction would be the median of all observations, which is more robust to outliers than mean. However, MAE's derivatives are discontinuous, making NN training harder to find the solution, compared to the smooth derivatives of MSE.

One big problem with using MAE for training of neural nets is its constantly large gradient, which can lead to missing minima at the end of training using gradient descent. For MSE, gradient decreases as the loss gets close to its minima, making it more precise.

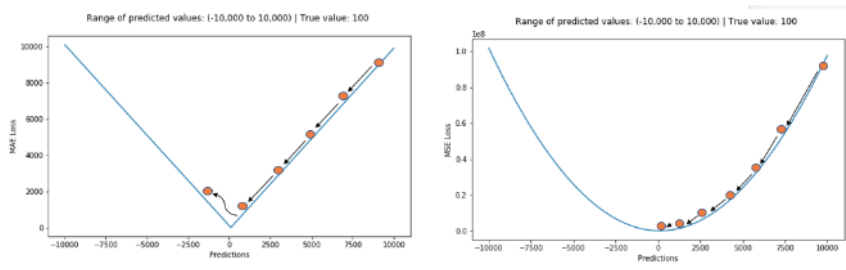


Figure 3.43: MSE vs MAE Loss.[1]

Log-cosh Loss

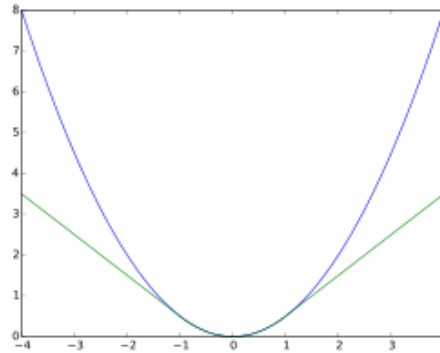


Figure 3.44: Graphical representation of the LogCosh regression loss function. It balances the tradeoff between MSE and MAE.

One solution to the tradeoff presented above is **Log-cosh Loss function**.

$$L_{Logcosh}(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{d \cdot B} \sum_{b=1}^B \sum_{j=1}^d \log(\cosh(y_j - \hat{y}_j)) \quad (3.44)$$

It's basically absolute error, which becomes quadratic when the error is small.

Robust Adaptive Loss Function (Barron Loss)

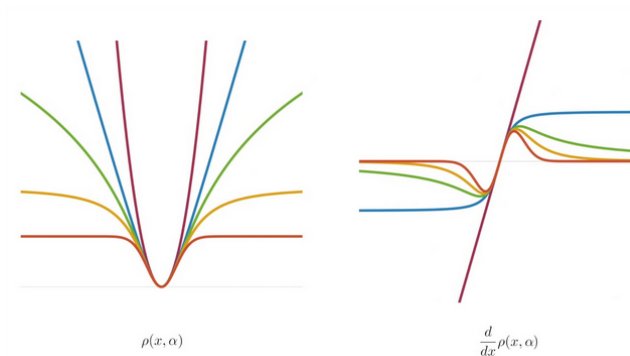


Figure 3.45: The curve of Barron Loss for different choices of parameters α, c that simulate a family of well-known regression Loss functions and its derivatives. [4]

This Loss function is an Robust Adaptive curve that consists the general case of some well-known Regression function for various values of a and changes its curve in-between them during training to determine its own robustness: e.g. for $a = 2$ it converges to the L2 loss, for $a = 0$ it is the Cauchy loss and for $a = \infty$ it is the Charbonnier loss. It was first presented in [4]. All the above robust losses belong in the family of:

$$\rho(x, a, c) = \frac{|a - 2|}{a} \left(\left(\frac{\left(\frac{x}{c}\right)^2}{|a - 2|} + 1 \right)^{\frac{a}{2}} - 1 \right) \quad (3.45)$$

It has the pros of being monotonic and smooth w.r.t. the inputs and its derivatives can be bounded.

Ideally, we would like to let the loss curve's shape parameter a to be an extra learnable parameter that we optimize through Gradient Descent in order to adapt to the inlier/outlier distributions:

$$(\mathbf{x}^*, a^*) = \arg \min_{\mathbf{x}, a} \sum_{i=1}^B \rho(e_i(\mathbf{x}), a, c), \quad (3.46)$$

where $e_i(\mathbf{x})$ the batch regression errors.

However, such an idea would allow the loss function's shape to overfit to the training dataset and zero-out the overall training loss, by severely constraining generalization capabilities. So, in order to fix this, we define the outlier Conditional Probability Distribution

$$P(\mathbf{x}|a, c) = \frac{1}{c\mathbb{Z}(a)} e^{-\rho(\mathbf{x}, a, c)}, \quad (3.47)$$

where \mathbb{Z} is the Riemann partition function:

$$\mathbb{Z}(a) = \int_{-\infty}^{\infty} e^{-\rho(\mathbf{x}, a, 1)} d\mathbf{x}. \quad (3.48)$$

Although this construct does not cover the exact definition of probability distributions as $\lim_{a \rightarrow \mp\infty} \mathbb{Z}(a)$, this discrepancy is solved by constraining the outlier field of definition in the finite range $[-T, T]$ for a big T value.

During training, what we need to minimize is the negative log-likelihood of this conditional probability :

$$-\log(\mathbb{P}(\mathbf{x}|a, c)) \quad (3.49)$$

which is bounded below by:

$$\rho(\mathbf{x}, a, c) + \log(c\mathbb{Z}(a)). \quad (3.50)$$

Optimizing the second term is not straightforward, but it can be proved that $\mathbb{Z}(a)$ can be successfully approximated by cubic splines (see Chapter 9), so we substitute it with this approximation during the optimization procedure (proof in [4]).

This notation forces this adaptive loss, theoretically, to avoid cheating as if it tries to overfit the training data, it is penalized by the second term to ensure a minimum generalization insurance. However, it is not straightforward that it will yield the optimal regression results, as it still has the freedom to reach the local optimum either by minimizing the overall loss w.r.t. to the NN weights or the shape parameters and which will prevail in every problem is a matter of experimentation.

Entropy

Entropy is a quantitative measure of uncertainty of a system modeled with a Probability Distribution \mathbb{P} .

We can compute the Entropy of a classification process in one of C total classes, using the formula below :

$$H(\mathbb{P}) = \sum_{c=1}^C \mathbb{P} \log\left(\frac{1}{\mathbb{P}}\right) = - \sum_{c=1}^C \mathbb{P} \log(\mathbb{P}), \quad (3.51)$$

where \mathbb{P} is the probability distribution of each class.

If the distribution \mathbb{P} is unknown, we may use it to approximate it with some other distribution, say, \mathbb{Q} .

Cross-Entropy

Cross Entropy is a way of measuring the “distance” between two probability distributions \mathbb{P} and \mathbb{Q} . If our model explains our dataset with \mathbb{Q} , which is an approximation of the ground truth \mathbb{P} distribution, then we want to minimize the Cross-Entropy between the two distributions as in order \mathbb{P} to converge to \mathbb{Q} :

$$L_{CE}(\mathbb{P}, \mathbb{Q}) = H(\mathbb{P}, \mathbb{Q}) = -\frac{1}{C \cdot B} \sum_{b=1}^B \sum_{c=1}^C \mathbb{P} \log\left(\frac{1}{\mathbb{Q}}\right) = -\frac{1}{C \cdot B} \sum_{c=1}^C \mathbb{P} \log(\mathbb{Q}) \quad (3.52)$$

Binary Cross Entropy:

If the ground truth distribution \mathbb{P} is binary then the distance is called **Binary Cross Entropy**. In this case $C=2$ and if we assign a probabilities \mathbb{P}, \mathbb{Q} for one class, we should assign $1 - \mathbb{P}, 1 - \mathbb{Q}$ for the other, accordingly. Then, the Binary Cross Entropy:

$$L_{BCE}(\mathbb{P}, \mathbb{Q}) = H(\mathbb{P}, \mathbb{Q}) = \frac{1}{2B} \sum_{b=1}^B (\mathbb{P} \log(\mathbb{Q}) + (1 - \mathbb{P}) \log(1 - \mathbb{Q})) \quad (3.53)$$

Intuitively, this formula converges \mathbb{P} to \mathbb{Q} using the following thought process:

Since we are in a binary scenario the ground truth probability must be either $\mathbb{Q}=0$ or $\mathbb{Q}=1$. In the first case, only the first term of formula is left, and the Binary Cross Entropy Loss function is minimized for $\mathbb{P} \rightarrow 0$, as well. Identically, if $\mathbb{Q} = 1$, then only the second term is left and the Binary Cross Entropy loss function is minimized for $\mathbb{P} \rightarrow 1$.

Multi-Class Cross Entropy

The above formulation may expand to the Multi-Class case. In such a scenario, every class's probabilities are considered to be independent from each other. So, for C classes, we sum up the corresponding Binary Cross Entropy assigning a probability \mathbb{P} for each of them and $1 - \mathbb{P}$ for the data samples belonging to all the rest $C-1$ classes. This approach is called **one-vs-all**.

Kullback-Leibler Divergence

Kullback-Leibler Divergence is the information gained when we move from a predicted distribution \mathbb{Q} to a ground truth distribution \mathbb{P} i.e. it is the difference between the Entropy of \mathbb{P} , $H(\mathbb{P})$, and the Cross Entropy between \mathbb{P} and \mathbb{Q} , $H(\mathbb{P}, \mathbb{Q})$:

$$H(\mathbb{Q}, \mathbb{P}) = H(\mathbb{P}) + D_{KL}(\mathbb{Q}||\mathbb{P}) \quad (3.54)$$

$$D_{KL}(\mathbb{Q}||\mathbb{P}) = H(\mathbb{Q}, \mathbb{P}) - H(\mathbb{P}) = \sum_{c=1}^C \mathbb{Q} [\log(\mathbb{Q}) - \log(\mathbb{P})] \quad (3.55)$$

This means that, the closer \mathbb{Q} gets to \mathbb{P} , the lower the divergence and, consequently, the cross-entropy, will be. Kullback-Leibler is a divergence and not a metric: rather a semi-metric as $D_{KL}(\mathbb{Q}||\mathbb{P}) \neq D_{KL}(\mathbb{P}||\mathbb{Q})$.

Geodesic Rotational Loss function

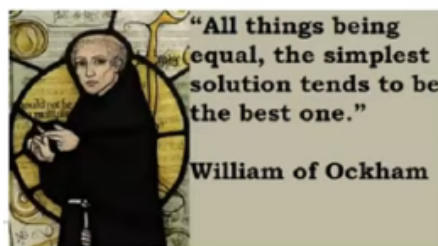


Figure 3.46: A portrait of William of Ockham (c.1287–1347).

The Philosophic Principle of Ockam's Razor:

Ockam's razor is a problem solving principle, attributed to the scholastic philosopher and theologian William of Ockham (c. 1287–1347), that states that:

“Entities should not be multiplied without necessity.”

and it simply paraphrased as:

“The simplest solution is most likely the right one.”

So, following this philosophical principle, in order to regress the rotational component of the Object’s Pose, the Loss function that we will be trying to minimize must be the minimal possible distance metric (between its Predicted and the Ground truth 3D rotations) in the 3D representation space of our choice (i.e. the length of the simplest path between them, in the $SO(3)$ space).

Euclidean/Hyperbolic distance on the parameter space:

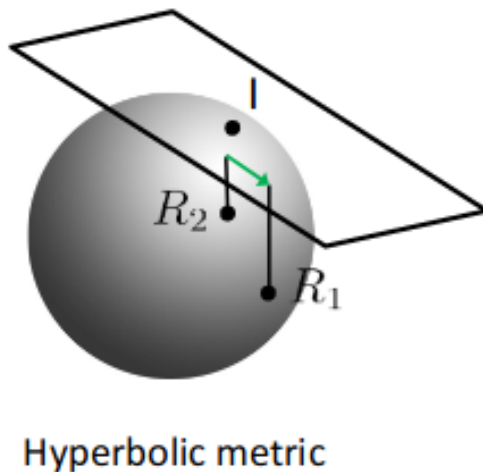


Figure 3.47

In literature, the most frequent such approach employs a Euclidean metric between the equivalent Rotations’ parameter representations. For example, for the Axis-Angle case, which maps the $SO(3)$ space to its tangent $so(3)$ Algebra through the logarithmic matrix mapping, the following **Euclidean/Hyperbolic Rotation metric** is used:

$$L(\hat{R}, R_{GT}) = d_H(\hat{R}, R_{GT}) = \left\| \log(\hat{R}) - \log(R_{GT}) \right\|_F. \quad (3.56)$$

Geometrically, as it is shown in fig.3.47, that the Loss is calculated on the tangent space of the $SO(3)$ Lie Group. Similarly, Euclidean distances are used between the corresponding Euler/Quaternia vectors.

Chordal Rotation Distance:

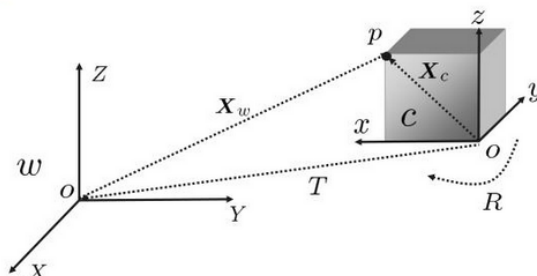


Figure 3.48: Three dimensional translations consist an element of the Euclidean space.

However, minimizing a Loss function based on the parametric rotation representation makes the Network prone to the corresponding ambiguities that may occur from such an uneducated regression, as no special care is taken in the Loss function computation to avoid them (e.g. Gimbal Locks for Euler parameters). It has different values and scale, according to the representation choice. So, it rises as a

better solution to transform the output parameters to the corresponding 3D Rotation matrices, that are free of such problems, and then take the **Chordal/Frobenius distance** between the two. Chordal distance is the Euclidean distance between the two corresponding elements of the Lie Group $SO(3)$.

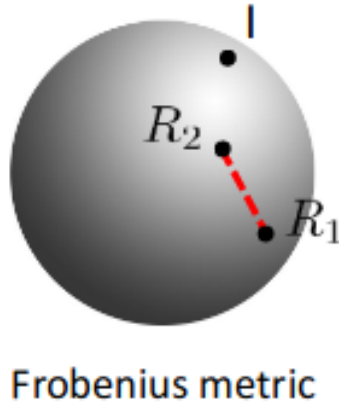


Figure 3.49: Visualization of the Chordal Geodesic Rotation Distance between two rotations R_1, R_2 [86]

$$L(\hat{R}, R_{GT}) = d_{Ch}(\hat{R}, R_{GT}) = \|\hat{R} - R_{GT}\|_F \quad (3.57)$$

Riemannian Geodesic Rotation Distance:

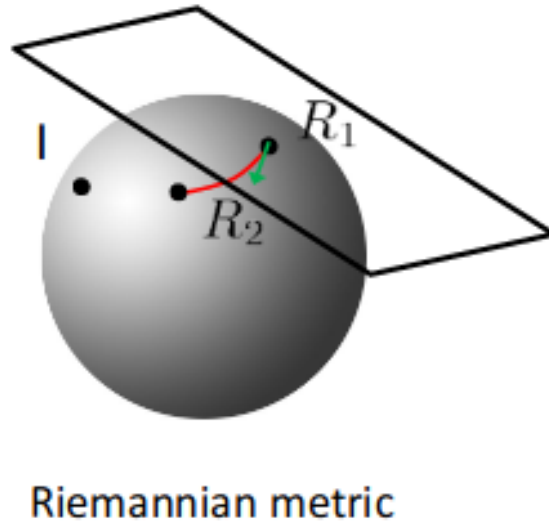


Figure 3.50: Visualization of the Riemannian Geodesic Rotation Distance between two rotations R_1, R_2 . [86]

Both metric choices above are of the Euclidean fashion, motivated by the way we measure translational distances, where Euclidean distance is the clear favourite. We must state, though, that such an approach has a profound weakness: while it suits to the 3D Euclidean translation space, it does not respect the geometry of the $SO(3)$ Lie Group. Entities belonging in the Euclidean space, have the property of monotonic, linear value progression. For example, when the object is translated along the z -axis, its t_z distance is only going to increase. However, that is not the case, obviously, for rotations, as a decrease in the Euclidean distance may indicate a circular behaviour of the parameters. For example, the rotation r_a will increase from 0 to 2π , but will later return to 0, again. That is because rotations do not belong in the linear Euclidean space, but rather in the non-linear $SO(3)$. As a result, straight lines (whose distance is the Euclidean one) are not geodesics (i.e. the straightest paths possible) on $SO(3)$. And according to the Ockam's Razor, it is this, straightest rotational path, that we need to find and

optimize our Network on.

We remind that rotation matrices R are elements of the Lie Group $SO(3)$, that consists a Riemannian Manifold in \mathbb{R}^3 and that is equipped with the Lie Algebra $so(3)$. This Lie Algebra is nothing more than the Tangent Space of the $SO(3)$ Manifold and, thus, all velocity vectors of the positions of the Manifold lie in that Tangent Space. As a result, one would select the basis of $so(3)$ to be a vector triplet consisting of 2 perpendicular velocity vectors at every element of $SO(3)$, along with the manifold normal vector, that could be found if we take their outer product and normalize it to unit length.

Furthermore, the Metric Tensor of $so(3)$ can be defined as:

$$\langle A, B \rangle_R = \langle L_{A^{-1}}A, L_{A^{-1}}B \rangle_{\mathbb{I}} = \langle R^{-1}A, R^{-1}B \rangle_{\mathbb{I}} = \text{tr}((R^{-1}A)^T R^{-1}B) = \text{tr}(A^T R^{-T} R^{-1}B). \quad (3.58)$$

A **Geodesic path** of a Riemannian Manifold is the straightest possible path available in the space of motion. Now, if we define a matrix curve $R(t) : [0, 1] \rightarrow SO(3)$. From all those curves, the Geodesics are:

$$R^*(t) = \arg \min_{R(t)} \{d(\hat{R}, R_{GT})\} \quad (3.59)$$

However, we can show that those Geodesics also minimize the velocity of an element moving on the Riemannian manifold $SO(3)$, and thus, its Kinetic energy measure:

$$E[R(t)] = \int_0^1 \langle \dot{R}(t), \dot{R}(t) \rangle_{R(t)} dt = \int_0^1 \text{tr}(\dot{R}^T(t) R^{-T}(t) R^{-1}(t) \dot{R}(t)) dt. \quad (3.60)$$

We will prove this property for the general case of an element of a Riemannian manifold, but only in 1D. Then, scaling up to multiple dimensions is straightforward. In this, general case, a curve of the Manifold is $\gamma(t) : [0, 1] \rightarrow M$. This curve has a known starting and an ending point: $\gamma(0) = \gamma_0, \gamma(1) = \gamma_f$. The differential length of the curve is:

$$d\mathcal{L} = \sqrt{d\gamma^2 + dt^2} = \sqrt{1 + \left(\frac{d\gamma}{dt}\right)^2} = \sqrt{1 + \dot{\gamma}^2} \quad (3.61)$$

and the total curve length is calculated by integrating this infinitesimal length across the entire curve domain:

$$\mathcal{L} = \int_0^1 d\mathcal{L} = \int_0^1 \sqrt{1 + \dot{\gamma}^2} dt, \quad (3.62)$$

In order to find the curve that minimizes this curve length we take the **Euler-Lagrange equation**:

$$\frac{d}{dt} \left(\frac{\partial \mathcal{L}}{\partial \dot{\gamma}} \right) + \frac{\partial \mathcal{L}}{\partial \gamma} = 0 \iff \frac{d}{dt} \frac{\dot{\gamma}}{\sqrt{1 + \dot{\gamma}^2}} = 0 \iff \frac{\dot{\gamma}}{\sqrt{1 + \dot{\gamma}^2}} = C \iff \dot{\gamma} = \sqrt{\frac{C^2}{1 - C^2}} \iff \gamma(t) = C_1 t + C_0, \quad (3.63)$$

which is the same solution that we would get if we minimized the total velocity energy measure E :

$$E[\gamma(t)] = \int_0^1 f(\gamma, \dot{\gamma}, t) dt = \int_0^1 \dot{\gamma}^2 dt, \quad (3.64)$$

as the occurred Euler-Lagrange equation this time:

$$\frac{d}{dt} \frac{\partial f}{\partial \dot{\gamma}} - \frac{\partial f}{\partial \gamma} = 0 \iff \frac{d}{dt} (2\dot{\gamma}) + 0 = 0 \iff \ddot{\gamma} = 0 \iff \gamma(t) = C_1 t + C_0. \quad (3.65)$$

It has become evident that, in order to estimate the Geodesic path along a Riemannian manifold, one should minimize the Energy function of its velocity, along the path itself. This, profoundly, happens in the case of the $SO(3)$ Lie Group, as well, for every rotation matrix $\gamma(t) = R(t)$:

$$\mathbb{R}^*(t) = \arg \min_{R(t)} E[R(t)] = \arg \min_{R(t)} \int_0^1 \text{tr}(\dot{R}^T(t)R^{-T}(t)R^{-1}(t)\dot{R}(t))dt. \quad (3.66)$$

This means that $R^*(t)$ is a crucial point of $E[R(t)]$ and according to Fermat's theorem:

$$\delta E = 0 \iff \delta \int_0^1 \text{tr}(\dot{R}^T(t)R^{-T}(t)R^{-1}(t)\dot{R}(t))dt = 0, \quad (3.67)$$

with

$$\delta R^{-1} = -R^{-1}\delta R R^{-1}. \quad (3.68)$$

Due to the linearity of both the integral and trace operators:

$$-2 \int_0^1 \text{tr}\left(\left(\frac{d}{dt}(\dot{R}^T R^{-T} R^{-1}) + R^{-1}\dot{R}\dot{R}^{-T}R^{-1}\right)\delta R\right)dt = 0. \quad (3.69)$$

The equation above is valid for every value of δR , which varies along the Geodesic path, and, thus, we result in the following **Geodesic equation**:

$$\frac{d}{dt}\left(\dot{R}^T R^{-T} R^{-1}\right) + R^{-1}\dot{R}\dot{R}^{-T}R^{-1} = 0 \iff \nabla_{R(t)}\dot{R}(t) = 0. \quad (3.70)$$

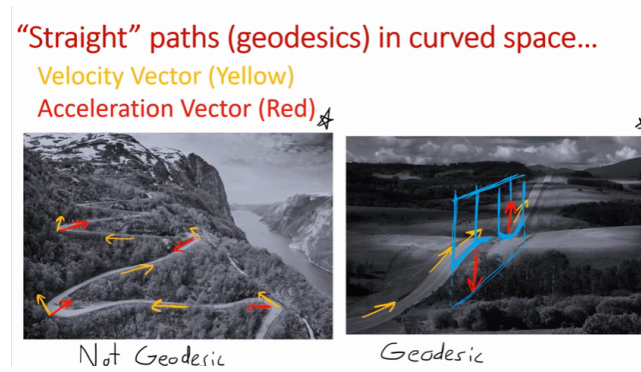


Figure 3.51: Visualization of different paths on a Manifold-like terrain. The left is not a Geodesic, but the right one is.

From a physical standpoint, one can notice that, according to this Geodesic equation, Geodesic paths have the property of the **component of their Acceleration that is tangent to the Manifold being zero**.

In order to solve this Geodesic differential equation, accompanied by its boundary conditions, we will try to guess a solution. We observe that if we select:

$$R(t) = R_{GT}e^{At}, \quad (3.71)$$

where A is an element of $\text{so}(3)$:

$$A = \log(R_{GT}^{-1}\hat{R}), \quad (3.72)$$

then, this selection consists of a solution of the Geodesic boundary condition problem:

More specifically, the temporal derivative of $R(t)$:

$$\dot{R}(t) = R_{GT}Ae^{tA} = R(t)A \iff A = R^{-1}(t)\dot{R}(t) \quad (3.73)$$

and if we substitute it in the Geodesic equation:

$$\frac{d}{dt}(A^T R^{-1}) + AA^T R^{-1} = -A^T R^{-1}\dot{R}R^{-1} + AA^T R^{-1} = [A, A^T]R^{-1} = 0. \quad (3.74)$$

Please note that nowhere is specified that this Geodesic path of $SO(3)$ is unique.

Now that we have found this Geodesic path, we can calculate its length, i.e. the length of the curve of $SO(3)$ that connects \hat{R}, R_{GT} , namely their distance:

$$\begin{aligned} \mathcal{L}[R^*(t)] &= \int_{R^*(t)} \sqrt{\langle \dot{R}(t), \dot{R}(t) \rangle_{R^*(t)}} dt = \int_0^1 \sqrt{\text{tr}(\dot{R}^T R^{-T} R^{-1} \dot{R})} dt = \int_0^1 \sqrt{\text{tr}(A^T A)} dt = \iff \\ &\iff \mathcal{L}[R^*(t)] = \|A\|_F = \left\| \log(\hat{R}^{-1} R_{GT}) \right\|_F. \end{aligned} \quad (3.75)$$

This Riemannian Rotational Distance is a metric, since it satisfies all three of the prominent metric properties:

1. $d_{Riem}(\hat{R}, R_{GT}) = \mathcal{L}[R^*(t)] \geq 0$
2. $d_{Riem}(\hat{R}, R_{GT}) = \mathcal{L}[R^*(t)] = 0 \iff \log(\hat{R}^T R_{GT}) = 0 \iff \hat{R} = R_{GT}$
3. $d_{Riem}(\hat{R}, R_{GT}) = \mathcal{L}[R^*(t)] = \left\| \log(\hat{R}^T R_{GT}) \right\|_F = \left\| -\log(R_{GT} \hat{R}^T) \right\|_F = d_{Riem}(R_{GT}, \hat{R})$
4. $d_{Riem}(\hat{R}, R_{GT}) \leq d_{Riem}(\hat{R}, R_{Int}) + d_{Riem}(R_{Int}, R_{GT}), \forall R_{Int} \in SO(3).$

Indeed, this Rotational distance metric yields a gradient descent path at the Geodesic trajectory direction, the simplest one, according to the Ockam's Razor's principle:

$$\begin{aligned} \nabla d_{Rot}(\hat{R}, R_{GT}) &= \frac{d}{dA} \|A\|_F = \frac{d}{dA} \sqrt{\text{tr}(AA^T)} = \frac{1}{2\sqrt{\text{tr}(AA^T)}} \frac{d}{dA} \text{tr}(AA^T) = \\ &= \frac{1}{2\sqrt{\text{tr}(AA^T)}} \cdot 2A = \frac{A}{\|A\|_F} = \mathcal{N}(A) = \hat{A} = \mathcal{N}(\log(\hat{R}^T R_{GT})) \end{aligned} \quad (3.76)$$

To conclude with, according to what has been stated above, $SO(3)$ is multiplicative, rather than an additive space, so we must set the Rotation Matrix difference as:

$$\Delta R = \hat{R}^T R_{GT} \in SO(3). \quad (3.77)$$

As a result, the **Geodesic Rotation metric on the Riemannian manifold $SO(3)$** , i.e. the distance of the minimal path that connects two of its elements: $\hat{R}, R_{GT} \in SO(3)$ is defined as:

$$L(\hat{R}, R_{GT}) = d_{Riem}(\hat{R}, R_{GT}) = \left\| \log(\hat{R}^T R_{GT}) \right\|_F, \quad (3.78)$$

graphically presented in fig.3.50.

The Rotation Matrix difference:

$$\Delta R = \hat{R}^T R_{GT} \in SO(3), \quad (3.79)$$

has an Axis-Angle representation of \mathbf{e}, θ :

$$\theta = \arccos\left(\frac{\text{tr}(\Delta R) - 1}{2}\right), \quad (3.80)$$

$$\mathbf{e} = \frac{1}{2\sin(\theta)} \begin{bmatrix} \Delta R_{32} - \Delta R_{23} \\ \Delta R_{13} - \Delta R_{31} \\ \Delta R_{21} - \Delta R_{12} \end{bmatrix} \quad (3.81)$$

which are combined as:

$$\Delta R = e^{\theta[\mathbf{e}]_x}, \quad (3.82)$$

If we use the Rodriguez formula (just for the proof):

$$\Delta R = \mathbb{I}_3 + \sin(\theta)[\mathbf{e}]_x + (1 - \cos(\theta))[\mathbf{e}]_x^2, \quad (3.83)$$

we can simplify the aforementioned distance as follows:

$$\begin{aligned}
d(\hat{R}, R_{GT}) &= \left\| \frac{\theta}{2 \sin(\theta)} (\hat{R} - R^{-1}) \right\|_F = \left\| \frac{\theta}{2 \sin(\theta)} (\hat{R} - R^T) \right\|_F = \\
&= \left\| \frac{\theta}{2 \sin(\theta)} (\mathbb{I}_3 + \sin(\theta)[\mathbf{e}]_{\times}) + (1 - \cos(\theta))[\mathbf{e}]_{\times}^2 - (\mathbb{I}_3 + \sin(-\theta)[\mathbf{e}]_{\times} + (1 - \cos(\theta))[\mathbf{e}]_{\times}^2) \right\|_F = \\
&= \left\| \frac{\theta}{2 \sin(\theta)} 2 \sin(\theta)[\mathbf{e}]_{\times} \right\|_F = \|\theta[\mathbf{e}]_{\times}\|_F = \theta \|\mathbf{e}\|_F = \\
&= \theta = \arccos\left(\frac{\text{tr}(\hat{R}^T \cdot R_{GT}) - 1}{2}\right),
\end{aligned} \tag{3.84}$$

by measuring the Predicted-Ground Truth Rotation distance as the 3D angle range between them.

In our implementation, we substitute it with the equivalent (eq. (3.84)), measured in rad, as it is mathematically simpler and computationally cheaper.

We need to state here, that, as it has been observed in [81], eq. (3.84) suffers from a severe local minima problem, asking for a suitable convex coarse-grained initialization to tighten up the Pose Search Space. Afterwards, the Geodesic Rotation Loss function is employed to achieve a lower fine-grained rotation error.

This experimental observation, made in many previous works, is theoretically supported if we explicitly calculate the loss's derivatives w.r.t. the two matrix components: R_{GT} and \hat{R} , just like the BackPropagation algorithm does:

•

$$\begin{aligned}
\frac{\partial L_{Rot}(R_{GT}, \hat{R})}{\partial \hat{R}} &= \frac{1}{B} \frac{\partial \sum_{b=1}^B \frac{\arccos(\text{tr}(R_{(\mathbf{b}),GT}^T \cdot \hat{R}_{(\mathbf{b})}) - 1)}{2}}{\partial \hat{R}_{(\mathbf{b})}} = \\
&= \frac{1}{B} \sum_{b=1}^B \frac{1}{2} \frac{\partial [\arccos(\text{tr}(R_{(\mathbf{b}),GT}^T \cdot \hat{R}_{(\mathbf{b})}) - 1)]}{\partial \hat{R}_{(\mathbf{b})}} = \\
&= -\frac{1}{B} \sum_{b=1}^B \frac{\partial (\text{tr}(R_{(\mathbf{b}),GT}^T \cdot \hat{R}_{(\mathbf{b})}) - 1)}{\partial \hat{R}_{(\mathbf{b})}} \frac{1}{2\sqrt{1 - (\text{tr}(R_{(\mathbf{b}),GT}^T \cdot \hat{R}_{(\mathbf{b})}) - 1)^2}} = \\
&= -\frac{1}{B} \sum_{b=1}^B \frac{R_{(\mathbf{b}),GT}}{2\sqrt{2\text{tr}(R_{(\mathbf{b}),GT}^T \cdot \hat{R}_{(\mathbf{b})}) - \text{tr}^2(R_{(\mathbf{b}),GT}^T \cdot \hat{R}_{(\mathbf{b})})}}
\end{aligned} \tag{3.85}$$

•

$$\begin{aligned}
\frac{\partial L_{Rot}(R_{GT}, \hat{R})}{\partial R_{GT}} &= \frac{1}{B} \frac{\partial \sum_{b=1}^B \frac{\arccos(\text{tr}(R_{(\mathbf{b}),GT}^T \cdot \hat{R}_{(\mathbf{b})}) - 1)}{2}}{\partial R_{(\mathbf{b}),GT}} = \\
&= \frac{1}{B} \sum_{b=1}^B \frac{1}{2} \frac{\partial [\arccos(\text{tr}(R_{(\mathbf{b}),GT}^T \cdot \hat{R}_{(\mathbf{b})}) - 1)]}{\partial R_{(\mathbf{b}),GT}} = \\
&= -\frac{1}{B} \sum_{b=1}^B \frac{\partial (\text{tr}(R_{(\mathbf{b}),GT}^T \cdot \hat{R}_{(\mathbf{b})}) - 1)}{\partial R_{(\mathbf{b}),GT}} \frac{1}{2\sqrt{1 - (\text{tr}(R_{(\mathbf{b}),GT}^T \cdot \hat{R}_{(\mathbf{b})}) - 1)^2}} = \\
&= -\frac{1}{B} \sum_{b=1}^B \frac{\hat{R}_{(\mathbf{b})}}{2\sqrt{2\text{tr}(R_{(\mathbf{b}),GT}^T \cdot \hat{R}_{(\mathbf{b})}) - \text{tr}^2(R_{(\mathbf{b}),GT}^T \cdot \hat{R}_{(\mathbf{b})})}}
\end{aligned} \tag{3.86}$$

So, from a mathematical standpoint, we explain why minimizing the Geodesic Rotation Loss from the beginning is the cause of trapping the Network's weights into a local minimum: during the Network's initialization, when the matrix product $R_{(\mathbf{b}),GT}^T \cdot \hat{R}_{(\mathbf{b})}$ heavily diverges from \mathbb{I}_3 , its derivatives have a strong denominator term (as the traces present bigger values) that makes them weaker as they further back-propagate to previous layers. On the other hand, a favorable prior initialization with a proper sub-optimal regression loss will give a "warmer" initial optimization point, where this problem is less severe and allows gradients to back-propagate more easily and, thus, refine rotational estimations.



Figure 3.52 The 2D map-projection of the airplane's 3D trajectories (both the Euclidean and the Geodesic ones).



Figure 3.53 The Geodesic (shortest) distance path, on the globe, which is created by the intersection of the globe with the greatest plane passing from the 2 points of the globe and at a direction that follows the globe's parallels.

Practical Example:

This is the reason for which air travels do not follow straight lines on the map (like suggested in fig.3.52), but slightly curved ones instead. That happens as they travel in 3D and they have to follow the globe's curvature. So, every time, the shortest path is the Geodesic one, called as **Great circle distance path**, in the globe special case, i.e. the intersection of the globe with the greatest plane passing from the 2 points of the globe and has a direction along its parallels (see fig.3.53).

For example, in order to travel from Constantinople to New York, the airplane will not travel in a straight line, like the 2D map projection of the curvature would suggest, but will pass from Greenland, first, instead.

3.3.18 Rotation Anisotropy Weighting using Intertial Tensor

If we focus only on the rotational component of the pose, we can make the following observation: the same 3D rotation angle θ (in Axis-Angle formulation) has a different visual imprint regarding each rotation axis. That notion will become much clearer after noticing fig.3.54.

When an object (in this case the "Eiffel tower") is rotated by the same angle (e.g. of 15°), the visual perception of it tends to differ according to the object's rotation axis. For example, rotation (a) seems to the viewer/sensor much smaller than rotation (b), despite the fact that they are of the same amplitude. As it is stated in [11], the same effect occurs when using Pose metrics that do not take into consideration this anisotropy and the object's geometry in general.

To this end, Brégier et al. [11] formulated the square root of the Covariance Matrix of the object's weighted surface, Λ , as a **Moment of Inertia Tensor** in order to assign a different weight to each rotational component. Just like a Moment of Inertia Tensor, Λ may be calculated with respect to any point in space, so its center of mass is selected for practical reasons.

$$\Lambda = \sqrt{\left(\frac{1}{S} \int_{\mathcal{O}} \mu(\mathbf{x}) \mathbf{x} \mathbf{x}^T ds\right)}, \quad (3.87)$$

where \mathcal{O} is the object's Triangle mesh 3D model, S is its surface area and $\mu(\mathbf{x})$ its mass distribution.

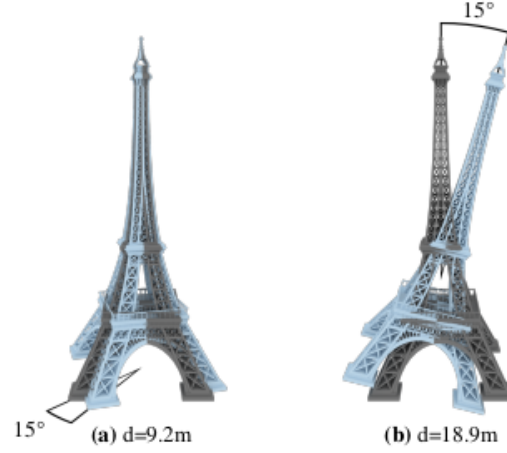


Figure 3.54: Usual metrics would consider the distances between the two poses in cases (a) and in (b) equal – as in both cases the two poses are linked by a rotation of 15 deg around the center of mass of the object. Our distance will account for the object geometry and discriminates between these two configurations.[11]

When the object is registered as a triangular mesh, we calculate Λ as follows:

Let's assume that we have an object 3D CAD model $\mathcal{O} = \cup_i \mathcal{T}(\mathbf{a}_i, \mathbf{b}_i, \mathbf{c}_i)$, which consists of triangular meshes where $\mathcal{T}(\mathbf{a}, \mathbf{b}, \mathbf{c})$ is a triangle defined by three vertices $\mathbf{a}, \mathbf{b}, \mathbf{c} \in \mathbb{R}^3$. The model's characteristics can be computed easily through the contributions of its triangles.

Let $\mathcal{T}(a, b, c)$ be a given triangle.

Its surface area can be computed with a cross product:

$$S_{\mathbf{a},\mathbf{b},\mathbf{c}} = \frac{\|(\mathbf{b} - \mathbf{a}) \times (\mathbf{c} - \mathbf{a})\|_2}{2}, \quad (3.88)$$

its center mass:

$$\mathbf{O}_{\mathbf{a},\mathbf{b},\mathbf{c}} = \frac{\mathbf{a} + \mathbf{b} + \mathbf{c}}{3} \quad (3.89)$$

and its uncentered covariance matrix via:

$$\sigma_{\mathbf{a},\mathbf{b},\mathbf{c}} = \frac{S_{\mathbf{a},\mathbf{b},\mathbf{c}}}{12} \left(9\mathbf{O}_{\mathbf{a},\mathbf{b},\mathbf{c}}\mathbf{O}_{\mathbf{a},\mathbf{b},\mathbf{c}}^T + \mathbf{a}\mathbf{a}^T + \mathbf{b}\mathbf{b}^T + \mathbf{c}\mathbf{c}^T \right). \quad (3.90)$$

From those results, we deduce the expression of the total surface area of the mesh:

$$S = \sum_i S_{\mathbf{a}_i,\mathbf{b}_i,\mathbf{c}_i} \quad (3.91)$$

and its center of mass:

$$\mathbf{O} = \sum_i S_{\mathbf{a}_i,\mathbf{b}_i,\mathbf{c}_i} \mathbf{O}_{\mathbf{a}_i,\mathbf{b}_i,\mathbf{c}_i}. \quad (3.92)$$

If that triangle mesh center of mass is chosen as origin of the object frame, then the **Inertia Tensor** is numerically approximated by the square root of its Covariance Matrix, normalized by its total surface area:

$$\Lambda = \sqrt{\frac{1}{S} \sum_i \sigma_{\mathbf{a}_i,\mathbf{b}_i,\mathbf{c}_i}} \quad (3.93)$$

3.3.19 The effect of Rotation Representations on Neural Networks

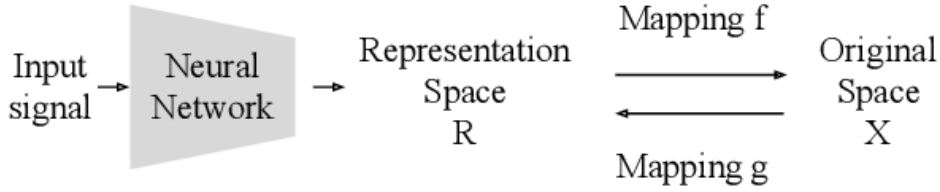


Figure 3.55: A graphical interpretation of Continuous Rotation Representations in the NN context, as presented in [171].

Let's call \mathbf{f} the mapping from the Representation space \mathcal{R} to the $SO(3)$ Lie Group and \mathbf{g} its inverse transformation.

Rotation Matrices

As mentioned before, the first thought that should be passing our minds is immediately regressing the 9 parameters of the Rotation Matrix \mathbf{R} . However, such an approach would set those parameters free of constraints with the predicted \mathbf{R} finally being out of the $SO(3)$ Lie Group. Even if we attempt to add its orthogonality constraints in the loss function as extra (weighted) regularization terms, the result would not be much more than a crude approximation of $SO(3)$, while an external fitting to the $SO(3)$ constraints does not guarantee the existence of suitable solutions. So, we understand that, as far as Machine Learning algorithms are concerned, a parameterization of the 3D Rotations is necessary for regression. Later, those parameters can be transformed suitably to a matrix \mathbf{R} with all the suitable properties which will be used in our Loss function in training and in the Pose Matrix for inference.

There are rotation scenarios when the NN returns highly inaccurate rotation estimations, that cannot be improved by lengthening the training duration or increasing the number of its learnable parameters. That happens because the Rotation parameterization must be convenient for the optimization process, which means that the mapping from the $SO(3)$ Lie Group to the parameter space $\mathbf{g}(\mathbf{R})$ must be **continuous** (see fig.3.55). When that is not the case, it becomes harder for our NN architecture to learn the correct mapping near the points of discontinuity.

Each of the parameterizations we have seen in Section 2.11 is equipped with an equivalent transformation to and from \mathbf{R} , with all of them having the disadvantage of the existence of discontinuities in the Euclidean topology.

Euler Angles

- Euler Angles $\rightarrow \mathbf{R}$:

$$\begin{aligned}
 R(\mathbf{r}) &= R(\psi)R(\theta)R(\phi) = \\
 &= \begin{bmatrix} \cos(\theta) \cos(\psi) & -\cos(\psi) \cos(\theta) & \sin(\theta) \\ \cos(\phi) \sin(\psi) + \cos(\psi) \sin(\phi) \sin(\theta) & \cos(\phi) \cos(\psi) - \sin(\phi) \sin(\theta) \sin(\psi) & -\cos(\theta) \sin(\phi) \\ \sin(\phi) \sin(\psi) - \cos(\phi) \cos(\psi) \sin(\theta) & \cos(\psi) \sin(\phi) + \cos(\phi) \sin(\theta) \sin(\psi) & \cos(\phi) \cos(\theta) \end{bmatrix}
 \end{aligned} \tag{3.94}$$

$$\begin{aligned}
 f_{E.A.}(\mathbf{r}) &= R(\mathbf{r}) = R(\psi)R(\theta)R(\phi) = \\
 &= \begin{bmatrix} \cos(\theta) \cos(\psi) & -\cos(\psi) \cos(\theta) & \sin(\theta) \\ \cos(\phi) \sin(\psi) + \cos(\psi) \sin(\phi) \sin \theta & \cos(\phi) \cos(\psi) - \sin(\phi) \sin(\theta) \sin(\psi) & -\cos(\theta) \sin(\phi) \\ \sin(\phi) \sin(\psi) - \cos(\phi) \cos(\psi) \sin(\theta) & \cos(\psi) \sin(\phi) + \cos(\phi) \sin(\theta) \sin(\psi) & \cos(\phi) \cos(\theta) \end{bmatrix}
 \end{aligned} \tag{3.95}$$

- $\mathbf{R} \rightarrow$ Euler Angles:

$$\mathbf{g}_{E.A.}(R) = \begin{bmatrix} \phi = \arctan 2\left(\frac{-r_{12}}{r_{11}}\right) \\ \theta = \arcsin(r_{13}) \\ \psi = \arctan 2\left(\frac{-r_{12}}{r_{13}}\right) \end{bmatrix} \quad (3.96)$$

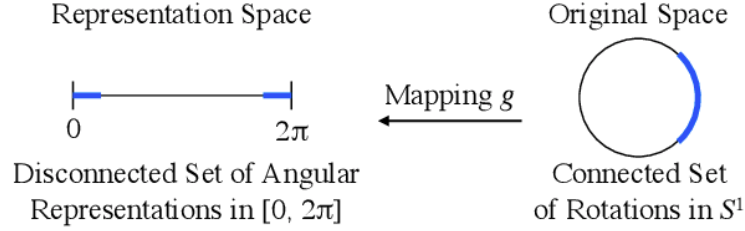


Figure 3.56: A simple 2D example which motivates the definition of [171] for Continuous Representation for 3D Rotations.

Proof of discontinuity:

When the predicted Rotation matrix R is close to the Identity matrix:

•

$$\lim_{R \rightarrow \mathbb{I}_3^+} \mathbf{g}_{E.A.}(R) = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

•

$$\lim_{R \rightarrow \mathbb{I}_3^-} \mathbf{g}_{E.A.}(R) = \begin{bmatrix} 0 \\ 2\pi \\ 0 \end{bmatrix}$$

which says that $\mathbf{g}_{E.A.}$ is discontinuous w.r.t. $SO(3)$ at its identity element, so it is discontinuous overall.

The mathematical definition of discontinuity of representation, presented right above, may not be clear from an intuitive perspective. In order to make it so, we give the following example:

Let's consider the subset of $SO(2)$ rotations and try to parameterize it.

The most profound way to do it is to model that planar rotation with an angle θ (different from the yaw θ angle), as in fig.3.56. Then, the Rotation matrix $R \in SO(2)$:

$$R = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \quad (3.97)$$

is mapped to the representation space $\mathcal{R} = [0, 2\pi]$ using $\mathbf{g}(R) = \theta \in [0, 2\pi]$. If we take the limits as R gets closer to \mathbb{I}_2 from left and right:

$$\lim_{R \rightarrow \mathbb{I}_2^+} \mathbf{g}(R) = 0, \quad (3.98)$$

while

$$\lim_{R \rightarrow \mathbb{I}_2^-} \mathbf{g}(R) = 2\pi \quad (3.99)$$

which means that \mathbf{g} is discontinuous at the identity rotation matrix, thus trying to map the connected set $SO(2)$ to a disconnected one: \mathcal{R} .

On the contrary, if we had chosen another representation: let's say the anti-intuitive

$$\mathbf{g}(R) = \begin{bmatrix} \cos \theta \\ \sin \theta \end{bmatrix}, \quad (3.100)$$

i.e. the first column of R, then we would be trying to map the connected set $SO(2)$ to another connected set: the unit circle S^1 . Mathematically,

$$\lim_{R \rightarrow \mathbb{I}_2^+} \mathbf{g}(R) = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad (3.101)$$

$$\lim_{R \rightarrow \mathbb{I}_2^-} \mathbf{g}(R) = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad (3.102)$$

and

$$\mathbf{g}(\mathbb{I}_2) = \begin{bmatrix} 0 \\ 1 \end{bmatrix}. \quad (3.103)$$

Unfortunately, in 3D there is not such a profound continuous rotation representation. As a result, after exposing the discontinuities in the other established representations, as well, we will formally define the Mapping Continuity and follow this thought process to define a Continuous mapping $\mathbf{g}_{Cont} : SO(3) \rightarrow \mathbb{R}$.

Quaternia

- **Quaternia $\rightarrow \mathbf{R}$:**

$$R(\mathbf{r}) = R \begin{pmatrix} x \\ y \\ z \\ w \end{pmatrix} = \begin{bmatrix} 1 - 2y^2 - 2z^2 & 2xy - 2zw & 2xz + 2yw \\ 2xy + 2zw & 1 - 2x^2 - 2z^2 & 2yz - 2xw \\ 2xz - 2yw & 2yz + 2xw & 1 - 2x^2 - 2y^2 \end{bmatrix} \quad (3.104)$$

- **$\mathbf{R} \rightarrow$ Quaternia:**

$$\mathbf{g}_q(R) = \begin{cases} \begin{bmatrix} r_{32} - r_{23} \\ r_{13} - r_{31} \\ r_{21} - r_{12} \\ \text{tr}(R) + 1 \end{bmatrix}, & \text{tr}(R) + 1 \neq 0 \\ \begin{bmatrix} \sqrt{r_{11} + 1} \\ c_2 \sqrt{r_{22} + 1} \\ c_3 \sqrt{r_{33} + 1} \\ 0 \end{bmatrix}, & \text{tr}(R) + 1 = 0 \end{cases}, \quad (3.105)$$

$$\text{with } c_i = \begin{cases} 1, & r_{i1} + r_{i2} > 0 \\ -1, & \text{otherwise} \end{cases}.$$

Proof of discontinuity:

Let's define as R_π the set of 3D Rotations by 180° :

$$R_\pi = \{R \in SO(3) | \text{tr}(R) = -1\} \quad (3.106)$$

For quaternia,

-

$$\lim_{R \rightarrow R_\pi} \mathbf{g}_q(R) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad (3.107)$$

-

$$\mathbf{g}_q(R_\pi) \neq \mathbf{0} \quad (3.108)$$

So, the quaternia representation is discontinuous.

The same holds for Unit Quaternia (i.e. for the mapping \mathbf{g}_{qu}) as one can be linearly transformed to the other, just by scaling. As a result the Unit Quaternia representation is discontinuous in the Euclidean Topology. Here, we have to note that it is the Euclidean Topology that we care about because all Neural Network units are continuous in it. The Unit Quaternia representation remains continuous in $SO(3)$, but this is not what we need for performance improvement.

Axis Angle Representation - Rodriguez Theorem

- **Axis-Angle \rightarrow R:**

$$R(\mathbf{r}) = \begin{bmatrix} e_x^2(1 - \cos(\theta)) + \cos(\theta) & e_x e_y(1 - \cos(\theta)) - e_z \sin(\theta) & e_x e_z(1 - \cos(\theta)) + e_y \sin(\theta) \\ e_x e_y(1 - \cos(\theta)) + e_z \sin(\theta) & e_y^2(1 - \cos(\theta)) + \cos(\theta) & e_y e_z(1 - \cos(\theta)) - e_x \sin(\theta) \\ e_x e_y(1 - \cos(\theta)) - e_y \sin(\theta) & e_x e_y(1 - \cos(\theta)) + e_x \sin(\theta) & e_z^2(1 - \cos(\theta)) + \cos(\theta) \end{bmatrix} \quad (3.109)$$

- **R \rightarrow Axis-Angle:**

$$\mathbf{g}_{A.A.}(R) = \begin{bmatrix} \theta = \arccos \frac{\text{tr}(R)-1}{2} \\ \mathbf{e} = \frac{1}{2 \sin \theta} \begin{bmatrix} r_{32} - r_{23} \\ r_{13} - r_{31} \\ r_{21} - r_{12} \end{bmatrix} \end{bmatrix} \quad (3.110)$$

Proof of discontinuity:

As we have already described the linear relationship between the Unit Quaternia and Axis-Angle representation, since the former is discontinuous in the Euclidean topology, the latter will also be discontinuous in it.

As a matter of fact, in [171], Zhou et al. prove that:

"As far as it concerns 3D Rotations, all parametric representations that belong in the Euclidean topology R^d , $d \leq 4$ are discontinuous."

As a result, if we regress the parameters of one of them we should be extra cautious about avoiding their discontinuities.

Continuous Rotation Representations

We need a clear mathematical definition for the mapping continuity property.

Following [171], we present the following definitions:

Definition1: A **Homeomorphism** is a continuous and bijective mapping equipped with a continuous inverse. As a result, two topological spaces $\mathcal{X}_1, \mathcal{X}_2$ are **Topologically Equivalent** if \exists a Homeomorphism \mathcal{H} such that: $\mathcal{X}_1 \xleftrightarrow{\mathcal{H}} \mathcal{X}_2$.

Definition2:

The pair of mappings (f,g) is a **Representation** if and only if $f(g(R))=R, \forall R \in SO(3)$, which means $g^{-1}(R) = f|_{g(R)}$, i.e. that g is a **Topological Embedding** $\in SO(3)$.

Definition3:

The 3D rotation representation (f,g) is a **Continuous Representation** if and only if g is Continuous, which means that since R is a connected set (as a Lie Group), then g(C) is a connected set, as well.

In [171], Zhou et al. proposed an alternative parametric representation of rotation matrices $R \in SO(3)$, described by 6 parameters.

- **Continuous Rotation Representation \rightarrow R:**

If $\mathbf{r} \in \mathbb{R}^6 = [\mathbf{r}_1, \mathbf{r}_2]$ the 6-D parameter vector, with $\mathbf{r}_1 \in \mathbb{R}^3, \mathbf{r}_2 \in \mathbb{R}^3$ then there are two equivalent f mappings that we may define to the $SO(3)$ rotation space.

The first is that of a minimal Gramm-Schmidt-like orthonormalization. The above two 3D parametric vectors that are outputs of our architecture cannot be used as 2 of the 3 columns of a rotation matrix immediately, as there is no guarantee that they are orthonormal. So, we may transform them to an equivalent 3D orthonormal vector basis that equips the rotation matrix R

with its 3 column vectors. In simple words, we only have to normalize the length of \mathbf{r}_1 , keep only the normalized component of \mathbf{r}_2 that is perpendicular to \mathbf{r}_1 and then create the third column of \mathbf{R} just by using the outer product of the 2 new orthonormal vectors:

$$\mathbf{r}_1^{(GS)} = \frac{\mathbf{r}_1}{\|\mathbf{r}_1\|_2} \quad (3.111)$$

$$\mathbf{r}_2^{(GS)} = \frac{\mathbf{r}_2 - (\mathbf{r}_1^{(GS)} \cdot \mathbf{r}_2) \cdot \mathbf{r}_1^{(GS)}}{\left\| \mathbf{r}_2 - (\mathbf{r}_1^{(GS)} \cdot \mathbf{r}_2) \mathbf{r}_1^{(GS)} \right\|_2} \quad (3.112)$$

$$\mathbf{r}_3^{(GS)} = \mathbf{r}_1^{(GS)} \times \mathbf{r}_2^{(GS)} \quad (3.113)$$

or we may choose another transformation: take the outer product between the two output vectors as the third columns and then the corresponding outer product of this exact column with the first (normalized) output parameter vector to create the second one, as follows:

$$\mathbf{r}_1^{(OP)} = \frac{\mathbf{r}_1}{\|\mathbf{r}_1\|_2} \quad (3.114)$$

$$\mathbf{r}_3^{(OP)} = \frac{\mathbf{r}_1^{(OP)} \times \mathbf{r}_2}{\left\| \mathbf{r}_1^{(OP)} \times \mathbf{r}_2 \right\|_2} \quad (3.115)$$

$$\mathbf{r}_2^{(OP)} = \mathbf{r}_3^{(OP)} \times \mathbf{r}_1^{(OP)} \quad (3.116)$$

where $\mathbf{r}_i^{(GS/OP)}$ are the elements that build the Rotation matrix \mathbf{R} :

$$\mathbf{R} = f_{Cont}^{(GS)}(\mathbf{r}) = [\mathbf{r}_1^{(GS)}, \mathbf{r}_2^{(GS)}, \mathbf{r}_3^{(GS)}] \quad (3.117)$$

or

$$\mathbf{R} = f_{Cont}^{(OP)}(\mathbf{r}) = [\mathbf{r}_1^{(OP)}, \mathbf{r}_2^{(OP)}, \mathbf{r}_3^{(OP)}]. \quad (3.118)$$

We note that we did not examine the possibility of a full Gram-Schmidt orthonormalization process for the creation of our orthonormal basis, as it adds computational cost without adding any new substantial properties.

- **$\mathbf{R} \rightarrow$ Continuous Rotation Representation:**

As for the \mathbf{g} mapping, [171] creates our desired 6D parametric representation \mathbf{r} , just by using the first two columns of a Rotation matrix \mathbf{R} .

$$\mathbf{g}_{Cont}(\mathbf{R}) = \mathbf{g}_{Cont}([\mathbf{R}_x, \mathbf{R}_y, \mathbf{R}_z]) = \begin{bmatrix} r_{11} \\ r_{21} \\ r_{31} \\ r_{12} \\ r_{22} \\ r_{32} \end{bmatrix} \quad (3.119)$$

We have already seen that 4D or lower representations present discontinuities and we have described a 6D one. In [171], the authors experimented with a projection of the aforementioned 6D representation to 5D, but with suboptimal results. That is the reason we did not take it into consideration for a parameterization choice.

Finally, we only need to explain why the 6D representation is continuous. According to the above definitions, we only need to show that \mathbf{g}_{Cont} is Continuous and bijective.

- **Proof of Continuity:** It is rather profound that since the 6 parameters of the representation vector \mathbf{r} are taken without any processing (linear nor non-linear) from a rotation matrix R , whenever that matrix changes its parameters by a small amount, the representation will change by the exact same amount.
- **Proof of Bijectivity:** This is not as trivial, although it is far from difficult. We will only present the proof for the Gramm-Schmidt like mapping \mathbf{f}_{GS} , but the proof for the other case is exactly the same.

We need to show that $\forall R \in SO(3): \mathbf{f}_{GS}(\mathbf{g}_{GS}(R)) = R$.

We have assumed that $\mathbf{g}_{GS}(R) = [\mathbf{r}_{1(R)}, \mathbf{r}_{2(R)}]$. So, according to the \mathbf{f}_{GS} :

•

$$\mathbf{r}'_1 = \frac{\mathbf{r}_{1(R)}}{\|\mathbf{r}_{1(R)}\|_2} = \mathbf{r}_{1(R)} \quad (3.120)$$

•

$$\mathbf{r}'_2 = \frac{\mathbf{r}_{2(R)} - (\mathbf{r}'_{1(R)} \cdot \mathbf{r}_{2(R)})\mathbf{r}'_{1(R)}}{\|\mathbf{r}_{2(R)} - (\mathbf{r}'_{1(R)} \cdot \mathbf{r}_{2(R)})\mathbf{r}'_{1(R)}\|_2} = \frac{\mathbf{r}_{2(R)}}{\|\mathbf{r}_{2(R)}\|_2} = \mathbf{r}_{2(R)} \quad (3.121)$$

•

$$\mathbf{r}'_3 = \mathbf{r}'_1 \times \mathbf{r}'_2 = \mathbf{r}_{1(R)} \times \mathbf{r}_{2(R)} = \mathbf{r}_{3(R)} \quad (3.122)$$

Visualization of the Representation Comparison:

In fig.3.57, we see the 2D projection of the various 3D rotation representation parameter choices. We can see that only representations of more than 4 dimensions are continuous, which graphically means that their projection is homeomorphic to a circle and follows the same color order. That last requirement is set in order to secure that the reconstructed Rotation matrices that will be created using one of those representations has $\det(R) = 1$ and not $\det(R) = -1$, as it would be the case without it, securing this way its place in the $SO(3)$ Lie Group.

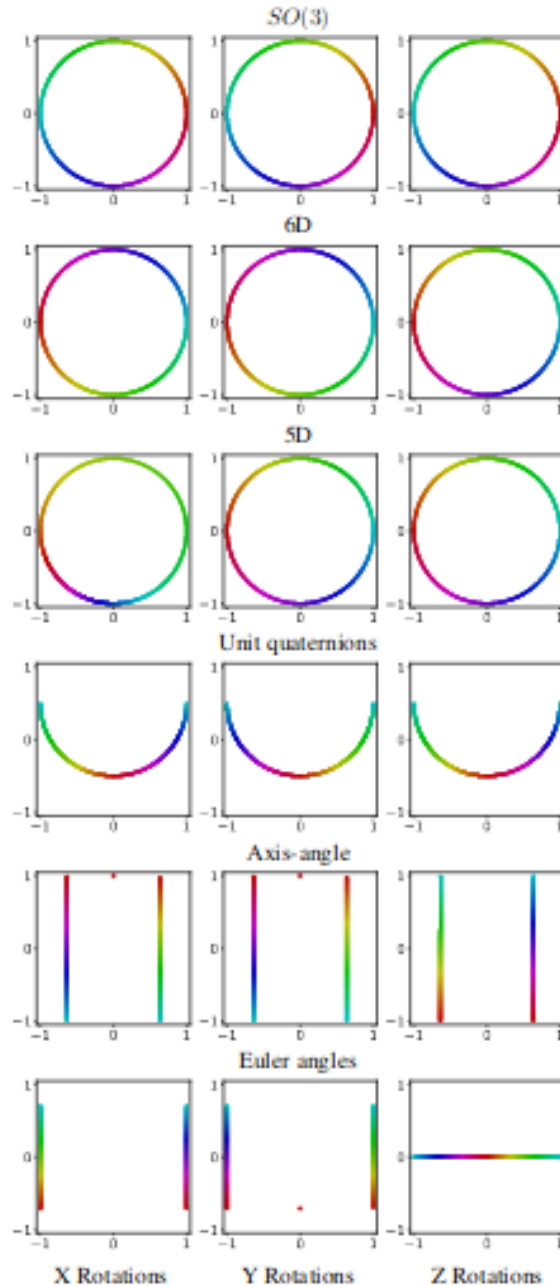


Figure 3.57: A visual comparison of the various rotation representations. The three columns showcase three different curves $\in SO(3)$: the “X,Y,Z” rotations of the corresponding axes. Each curve is mapped $\in SO(3)$ to each of the rotation representations in different rows and then it is projected in 2D via PCA. The hue is used to visualize the rotation angle $\in SO(3)$ around each of the three canonical axes “X,Y,Z”. We understand that a representation is continuous if that projection is homeomorphic to a circle and has the same direction of color change.^[171]

3.3.20 Multi-Task Loss functions

Weighting

There are cases where a learning scheme is benefited from trying to optimize multiple objectives at once. These objectives may come in two distinct cases: one main and several auxiliary ones or multiple principal objectives that give the network a more spherical understanding of the scene when they are handled at the same time. In both scenarios, we deal with them in the same way: we employ a Multi-Task Loss function that performs a weighted linear sum of the losses for each individual task:

$$Loss_{Overall} = \sum_{i=1}^M w_i L_i \quad (3.123)$$

Homoscedastic Uncertainty Weights

The weighting of multiple loss functions, as presented above, is far from a simple choice as the model performance is heavily depended on it. The optimal weighting scheme depends on the losses' scale (e.g. the different units of measurement when they have a physical meaning) and the amplitude of the model noise, namely, how much the loss diverges from its expected value. Direct setting of them is heavily anti-intuitive with the most frequent solution being an extensive manual tuning of these hyper-parameters, with the side-effect of multiplying the training time cost. So, we desire a way to learn them along with the rest of the Network parameters with a Gradient Descent-based optimization scheme.

In summary, each loss function is characterised by a level of uncertainty. If we choose a Bayesian modeling approach for this uncertainty, we can divide it in 2 different types, as stated in [60]:

1. **Model-related Uncertainty (Epistemic uncertainty):** The amount of information the model is unaware of when there are missing data points from the dataset distribution. Model uncertainty reduces as the dataset expands quantitatively and qualitatively.
2. **Data-related Uncertainty (Aleatoric uncertainty):** The amount of information the data type used cannot express, no matter the size of the dataset. It can further be separated in:
 - *Heteroscedastic Uncertainty:* When the missing information can be spotted in the inherent nature of the dataset (e.g. images of small resolution cannot depict object details with a desired accuracy, no matter how many there are in the dataset)
 - *Homoscedastic Uncertainty:* the one that depends totally on the task at hand and has nothing to do with the available dataset. It is this Uncertainty subcategory that we use to model our Multi-task Loss weights in a learnable way, following [60].

For a regression task, in the general case, our model outputs a vector function $\mathbf{f}^{\mathbb{W}}(\mathbf{x})$ to estimate a ground-truth vector \mathbf{y} . So, its Homoscedastic Uncertainty is modelled as a Gaussian Likelihood:

$$\Pr(\mathbf{y} | \mathbf{f}^{\mathbb{W}}(\mathbf{x})) \sim \mathcal{N}(\mathbf{y} | \mathbf{f}^{\mathbb{W}}(\mathbf{x}), \sigma^2) \quad (3.124)$$

where σ^2 models the observation noise.

Since we aim to handle multiple tasks at once and optimize the overall loss, the Likelihood of them all can be written as:

$$\begin{aligned} \Pr(\mathbf{y} | \mathbf{f}^{\mathbb{W}}(\mathbf{x})) &= \Pr(\mathbf{y}_1 | \mathbf{f}_1^{\mathbb{W}}(\mathbf{x})) \cdot \dots \cdot \Pr(\mathbf{y}_M | \mathbf{f}_M^{\mathbb{W}}(\mathbf{x})) = \\ &= \mathcal{N}(\mathbf{y}_1 | \mathbf{f}_1^{\mathbb{W}}(\mathbf{x}), \sigma_1^2) \cdot \dots \cdot \mathcal{N}(\mathbf{y}_M | \mathbf{f}_M^{\mathbb{W}}(\mathbf{x}), \sigma_M^2) \end{aligned} \quad (3.125)$$

Our final goal is to match $\mathbf{f}_i^{\mathbb{W}}(\mathbf{x})$ and \mathbf{y}_i as tightly as possible, which is achieved by maximizing this Likelihood. Specifically, we need to find the argument parameters (\mathbb{W}, σ) that maximize it. So, we can maximize the Log-Likelihood instead, since the arguments that do so are the same:

$$\begin{aligned} \arg \min_{\mathbb{W}, \sigma} \left\{ Loss_{Overall}(\mathbf{x}; \mathbb{W}, \sigma) \right\} &= \arg \min_{\mathbb{W}, \sigma} \left\{ -\log(\Pr(\mathbf{y} | \mathbf{f}^{\mathbb{W}}(\mathbf{x}))) \right\} = \arg \max_{\mathbb{W}, \sigma} \left\{ \log(\Pr(\mathbf{y} | \mathbf{f}^{\mathbb{W}}(\mathbf{x}))) \right\} = \\ &= \arg \max_{\mathbb{W}, \sigma} \left\{ -\frac{1}{2\sigma_1^2} \|\mathbf{y}_1 - \mathbf{f}_1^{\mathbb{W}}(\mathbf{x})\|_2^2 - \frac{\log(\sigma_1^2)}{2} - \dots - \frac{1}{2\sigma_M^2} \|\mathbf{y}_M - \mathbf{f}_M^{\mathbb{W}}(\mathbf{x})\|_2^2 - \frac{\log(\sigma_M^2)}{2} \right\}, \end{aligned} \quad (3.126)$$

where $\|\mathbf{y}_i - \mathbf{f}_i^{\mathbb{W}}(\mathbf{x})\|_2^2$ is the L_2 Loss (squared), the standard Regression Loss, but they can be substituted by any choice of Regression loss:

$$\begin{aligned} \arg \min_{\mathbb{W}, \sigma} \left\{ -\log(\Pr(\mathbf{y} | \mathbf{f}^{\mathbb{W}}(\mathbf{x}))) \right\} &= \\ = \arg \min_{\mathbb{W}, \sigma} \left\{ \frac{1}{2\sigma_1^2} L_1(\mathbf{f}_1^{\mathbb{W}}(\mathbf{x}), \mathbf{y}_1) + \log(\sigma_1) + \dots + \frac{1}{2\sigma_M^2} L_M(\mathbf{f}_M^{\mathbb{W}}(\mathbf{x}), \mathbf{y}_M) + \log(\sigma_M) \right\} \end{aligned} \quad (3.127)$$

For a Classification Task the case is rather similar, upgrading the expression above as global between the two types of problems. In this case, the Likelihood follows the Gibbs distribution as we take the Softmax (see Section 3.3.4) of the scaled model output $\mathbf{f}^{\mathbb{W}}(\mathbf{x})$):

$$\Pr(\mathbf{y} | \mathbf{f}^{\mathbb{W}}(\mathbf{x})) \sim \text{Softmax}\left(\frac{1}{\sigma^2} \mathbf{f}^{\mathbb{W}}(\mathbf{x})\right). \quad (3.128)$$

According to the Conditional Probability theory:

$$\Pr(\mathbf{y} | \mathbf{f}^{\mathbb{W}}(\mathbf{x}), \sigma) = \frac{\Pr(\mathbf{y}, \mathbf{f}^{\mathbb{W}}(\mathbf{x}) | \sigma)}{\Pr(\mathbf{f}^{\mathbb{W}}(\mathbf{x}) | \sigma)} \quad (3.129)$$

The Loss minimization with respect to the weights \mathbb{W} and the noise parameter σ follows the next thought process:

$$\begin{aligned} & \arg \min_{\mathbb{W}, \sigma} \left\{ \text{Loss}_{\text{Overall}}(\mathbf{x}; \mathbb{W}, \sigma) \right\} = \\ & = \arg \min_{\mathbb{W}, \sigma} \left\{ -\log \left(\Pr(\mathbf{y} | \mathbf{f}^{\mathbb{W}}(\mathbf{x}), \sigma) \right) \right\} = \\ & = \arg \min_{\mathbb{W}, \sigma} \left\{ -\log \left(\Pr(\mathbf{y} | \mathbf{f}^{\mathbb{W}}(\mathbf{x})) \Pr(\mathbf{f}^{\mathbb{W}}(\mathbf{x}) | \sigma) \right) \right\} = \\ & = \arg \min_{\mathbb{W}, \sigma} \left\{ -\log \left(\frac{\Pr(\mathbf{y}, \mathbf{f}^{\mathbb{W}}(\mathbf{x}) | \sigma)}{\Pr(\mathbf{f}^{\mathbb{W}}(\mathbf{x}) | \sigma)} \right) \right\} = \\ & = \arg \min_{\mathbb{W}, \sigma_1, \dots, \sigma_M} - \left\{ \sum_{i=1}^M \log \left(\Pr(y_i = c_i, \mathbf{f}_{c_i}^{\mathbb{W}}(\mathbf{x}) | \sigma_i) \right) + \sum_{i=1}^M \log \left(\Pr(\mathbf{f}_{c_i}^{\mathbb{W}}(\mathbf{x}) | \sigma_i) \right) \right\} = \\ & = \arg \min_{\mathbb{W}, \sigma_1, \dots, \sigma_M} \left\{ -\sum_{i=1}^M \log \left(\frac{\text{Softmax}(y_i, \mathbf{f}_i^{\mathbb{W}}(\mathbf{x}))}{\sigma_i} \right) + \sum_{i=1}^M \log \left(\Pr(\mathbf{f}_{c_i}^{\mathbb{W}}(\mathbf{x}) | \sigma_i) \right) \right\} = \\ & = \arg \min_{\mathbb{W}, \sigma_1, \dots, \sigma_M} \left\{ \sum_{i=1}^M \frac{L_i(\mathbf{x}; \mathbb{W}, \sigma_i)}{\sigma_i} + \sum_{i=1}^M \log \left(\Pr(\mathbf{f}_{c_i}^{\mathbb{W}}(\mathbf{x}) | \sigma_i) \right) \right\}. \end{aligned} \quad (3.130)$$

We can use the following approximating assumption to simplify our expression (which becomes an equation for $\sigma_i \rightarrow 1$):

$$\left(\sum_{c'} e^{\frac{1}{\sigma^2} \mathbf{f}_c^{\mathbb{W}}(\mathbf{x})} \right)^{\frac{1}{\sigma^2}} \rightarrow \frac{1}{\sigma^2} \sum_{c'} e^{\frac{1}{\sigma^2} \mathbf{f}_c^{\mathbb{W}}(\mathbf{x})}, \quad (3.131)$$

with the Classification expression becoming:

$$\begin{aligned} & \arg \min_{\mathbb{W}, \sigma} \left\{ \text{Loss}_{\text{Overall}}(\mathbf{x}; \mathbb{W}, \sigma) \right\} = \\ & \arg \min_{\mathbb{W}, \sigma_1, \dots, \sigma_M} \left\{ \sum_{i=1}^M \frac{L_i(\mathbf{x}; \mathbb{W}, \sigma_i)}{\sigma_i} + \sum_{i=1}^M \log \sigma_i^2 \right\} \end{aligned} \quad (3.132)$$

Of course, the mixed Regression-Classification case is a trivial combination of the two problem types.

As it is easily understood, $\frac{1}{2\sigma_i^2}$ are the weights w_i that we presented in the subsection above. They can be learned along with Network's parameters with a classical Gradient Descent-based optimization scheme. However, the careful reader will immediately come up with the following question: *How can it be assured that the optimization procedure will not try to learn σ_i to be big enough (in terms of norm), in order to make the overall Training Loss to converge to zero without having the obligation of*

properly training the Network's parameters \mathbb{W}_i in the meantime? Such a possibility would create abysmal overfitting.

To this end, the extra terms $\log \sigma_i^2$ are added. Their effect will be much clearer if we simplify the expression above:

(We get limited to only 2 Losses for the simplification but the results are automatically generalized to M of them)

In order to explain the effect of the $\log \sigma_i^2$ terms and, in the same time, avoid the numerical instabilities that may occur during the training procedure due to the presence of σ_i in the denominator, we set:

$$s = \log(\sigma^2), \quad (3.133)$$

which results in:

$$\frac{1}{\sigma^2} = e^{\log(\sigma^{-2})} = e^{-\log \sigma^2} = e^{-s} \quad (3.134)$$

So, the Overall Loss to minimize is simplified to the following formula:

$$Loss_{Overall} = e^{-s_1} L_1(\mathbf{f}_1^{\mathbb{W}}(\mathbf{x}), \mathbf{y}_1) + 2s_1 + e^{-s_2} L_2(\mathbf{f}_2^{\mathbb{W}}(\mathbf{x}), \mathbf{y}_2) + 2s_2 \quad (3.135)$$

Now, the following tradeoff is rather clear:

when the optimizer increases s_i to reduce the weight e^{-s_i} of one of the Losses L_i , it is added as an extra linear penalty to the Overall Loss, avoiding the case of perpetual monotonical increase of s_i 's which would zero-out the training cost and thus leaving the reduction of L_i 's as the primary way to minimize $Loss_{Overall}$, ensuring, this way, that the Network parameters are trained indeed.

3.3.21 Optimization Algorithms

Optimizer schemes

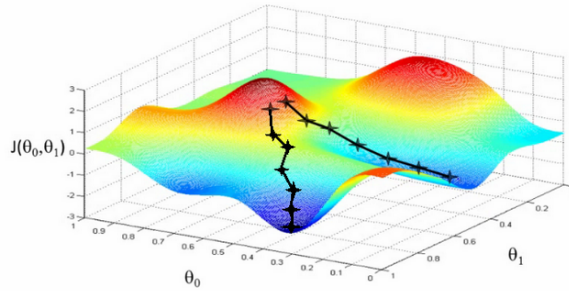


Figure 3.58: A Gradient-Descent-based Loss function optimization scheme.

Gradient descent is a way to minimize an objective function $J(\theta)$ parameterized by a model's parameters $\theta \in \mathbb{R}^m$ by updating the parameters in the opposite direction of the gradient of the objective function $\nabla_{\theta} J(\theta)$ w.r.t. to the parameters. The **learning rate** η determines the size of the steps we take to reach a (local) minimum. In other words, we follow the direction of the slope of the surface created by the objective function downhill until we reach a valley.

Batch Gradient Descent

Vanilla Gradient descent computes the gradient score of the objective function to the parameters θ of the entire training dataset.

$$\theta = \theta - \eta \cdot \nabla_{\theta} J(\theta) \quad (3.136)$$

The batch gradient descent algorithm needs to calculate the gradients of the whole train set. This is very slow and can evoke technical impediments such as memory insufficiency. Also, Batch gradient descent does not allow to train our model online.

Stochastic Gradient Descent

On the contrary, **Stochastic Gradient Descent (SGD)** performs a parameter update for each training example x^i and y^i :

$$\theta = \theta - \eta \cdot \nabla_{\theta} J(\theta; \mathbf{x}^i; \mathbf{y}^i) \quad (3.137)$$

Comparing both Batch Gradient Descent and Stochastic Gradient Descent algorithms, we conclude that the first implement abundant computations of the gradients. In large datasets it is possible that some examples will be similar. On the contrary, Stochastic Gradient Descent eschews this redundancy by updating the parameters per one training sample. Also, SGD can also be utilized in online learning.

On the one hand, while Batch Gradient Descent converges to the minimum of the basin the parameters are placed in, SGD's fluctuation enables it to jump to new and potentially better local minima. On the other hand, this ultimately complicates convergence to the exact minimum, as SGD will keep overshooting. However, it has been shown that **when we slowly decrease the learning rate**, SGD shows the same convergence behaviour as Batch Gradient Descent, almost certainly converging to a local or the global minimum for non-convex and convex optimization respectively.

Mini-Batch Gradient Descent

The **Mini-Batch Gradient Descent** Algorithm combines both BGD and SGD ideas. For one update, it takes into consideration only n samples from the training set.

$$\theta = \theta - \eta \cdot \nabla_{\theta} J(\theta; \mathbf{x}^{i,i+B}; \mathbf{y}^{i,i+B}) \quad (3.138)$$

- *For convex (Loss) functions*: the algorithm is proven to converge to their global minimum
- *For non-convex (Loss) functions*: the algorithm converges to a local minimum which, empirically, is close enough to the global one.

However Mini-Batch Gradient descent does not guarantee good convergence and some emerging challenges must be addressed:

- The gradients are calculated in batch mode and not just based on a single input. That is why the following tradeoff occurs:
 - **Big Batch size**:
 1. The gradient calculations for the elements of the mini-batch are executed in parallel increasing the training speed.
 2. As there is one gradient update performed for the whole minibatch, increasing it makes the individual bias count less in this update, making it less accurate.
 - **Small Batch size**:
 1. The Loss function, which is usually subjugated to a form of averaging, does not include a high information variability to guide the training process with its gradients with a robust enough way.
 2. It tends to run the Back Propagation algorithm more sequentially resulting in slower training.
- The selection, initialization and tuning of the suitable learning rate is a difficult task and may demand lots of experiments so as to deduce the appropriate one. Following, we discuss improvements to the standard Minibatch Gradient Descent algorithm, regarding the notion of learning rate, that improves the network's convergence:

Momentum

SGD has trouble navigating ravines, i.e. areas where the surface curves much more steeply in one dimension than in another, which are common around local optima. In these scenarios, SGD oscillates across the slopes of the ravine while only making hesitant progress along the bottom towards the local optimum, like in fig.3.59

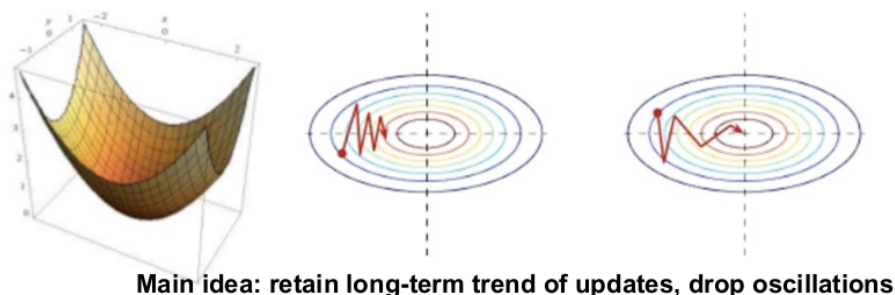


Figure 3.59: (a) SGD without momentum. (b) SGD with momentum [116]

Momentum is a method that helps in accelerating SGD in the relevant direction and dampens oscillations as can be seen in 3.59. It does this by adding a fraction γ of the update vector of the past time step to the current update vector:

$$\begin{aligned} \mathbf{v}_t &= \gamma \mathbf{v}_{t-1} - \eta \nabla_{\theta} (J(\theta)) \\ \theta' &= \theta - \mathbf{v}_{t-1} \end{aligned} \tag{3.139}$$

Essentially, when using momentum, we push a ball down a hill. The ball accumulates momentum as it rolls downhill, becoming faster and faster on the way (until it reaches its terminal velocity if there is air resistance, i.e. $\gamma < 1$). The same thing happens to our parameter updates: the momentum term increases for dimensions whose gradients point in the same directions and reduces updates for dimensions whose gradients change directions. As a result, we gain faster convergence and reduced oscillation.

Nesterov accelerated gradient

However, a ball that rolls down a hill, blindly following the slope, is highly unsatisfactory. We'd like to have a smarter ball, a ball that has a notion of where it is going so that it knows to slow down before the hill slopes up again.

Nesterov accelerated gradient (NAG) [7] is a way to give our momentum term this kind of prescience. We know that we will use our momentum term $\gamma \mathbf{v}_{t-1}$ to move the parameters θ . Computing $\theta - \gamma \mathbf{v}_{t-1}$ thus gives us an approximation of the next position of the parameters (the gradient is missing for the full update), a rough idea where our parameters are going to be. We can now effectively look ahead by calculating the gradient not w.r.t. to our current parameters θ but w.r.t. the approximate future position of our parameters:

$$\begin{aligned} \mathbf{v}_t &= \mathbf{v}_{t-1} - \eta \nabla_{\theta} J(\theta - \gamma \mathbf{v}_{t-1}) \\ \theta' &= \theta - \gamma \mathbf{v}_{t-1} \end{aligned} \tag{3.140}$$

Again, we set the momentum term γ to a value of around 0.9. While Momentum first computes the current gradient and then takes a big jump in the direction of the updated accumulated gradient, NAG first makes a big jump in the direction of the previous accumulated gradient, measures the gradient and then makes a correction, which results in the complete NAG update. This anticipatory update prevents us from going too fast and results in increased responsiveness.

Now that we are able to adapt our updates to the slope of our error function and speed up SGD in turn, we would also like to adapt our updates to each individual parameter to perform larger or smaller updates depending on their importance.

Adagrad

Adagrad is an algorithm for gradient-based optimization that does just this: It adapts the learning rate to the parameters, performing smaller updates (i.e. low learning rates) for parameters associated with frequently occurring features, and larger updates (i.e. high learning rates) for parameters associated with infrequent features.

Previously, we performed an update for all parameters θ at once, as every parameter $\theta^{(i)}$ used the same learning rate η . As Adagrad uses a different learning rate for every parameter $\theta_t^{(i)}$ at every time step t , we first show Adagrad's per-parameter update, which we then vectorize. For brevity, we use g_t to denote the gradient at time step t . $g_t^{(i)}$ is then the partial derivative of the objective function w.r.t. to the parameter $\theta_t^{(i)}$ at time step t :

$$\mathbf{g}_t^{(i)} = \nabla_{\theta} J(\theta_t^{(i)}) \quad (3.141)$$

The SGD update for every parameter $\theta_t^{(i)}$ at each time step t then becomes:

$$\theta_{t+1}^{(i)} = \theta_t^{(i)} - \eta_t \cdot g_t^{(i)} \quad (3.142)$$

In its update rule, Adagrad modifies the general learning rate η at each time step t for every parameter $\theta_t^{(i)}$ based on the past gradients that have been computed for $\theta_t^{(i)}$:

$$\theta_{t+1}^{(i)} = \theta_t^{(i)} - \frac{\eta}{\sqrt{G_{tt} + \epsilon}} \cdot \mathbf{g}_t^{(i)} \quad (3.143)$$

$G_t \in \mathbb{R}^{d \times d}$ here is a diagonal matrix where each diagonal element (i,i) is the sum of the squares of the gradients w.r.t. $\theta_t^{(i)}$ up to time step t , while ϵ is a smoothing term that avoids division by zero (usually on the order of $1e^{-8}$). Interestingly, without the square root operation, the algorithm performs much worse.

As G_t contains the sum of the squares of the past gradients w.r.t. to all parameters θ along its diagonal, we can now vectorize our implementation by performing a matrix-vector product between G_t and \mathbf{g}_t :

Adadelta

Adadelta is an extension of Adagrad that seeks to reduce its aggressive, monotonically decreasing learning rate. Instead of accumulating all past squared gradients, Adadelta restricts the window of accumulated past gradients to some fixed size w . Instead of inefficiently storing w previous squared gradients, the sum of gradients is recursively defined as a decaying average of all past squared gradients. The running average $E[g^2]_t$ at time step t then depends (as a fraction γ similarly to the Momentum term) only on the previous average and the current gradient:

$$E[g^2]_t = \gamma \cdot E[g^2]_{t-1} + (1 - \gamma) \cdot g_t^2. \quad (3.144)$$

We set γ to a similar value as the momentum term, around 0.9. For clarity, we now rewrite our vanilla SGD update in terms of the parameter update vector $\Delta\theta_t$:

$$\begin{aligned} \Delta\theta_t &= -\eta \cdot g_t^{(i)} \\ \theta_t &= \theta_t + \Delta\theta_t \end{aligned} \quad (3.145)$$

The parameter update vector of Adagrad that we derived previously thus takes the form:

$$\Delta\theta_t = -\frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} g_t \quad (3.146)$$

As the denominator is just the root mean squared (RMS) error criterion of the gradient, we can replace it with the criterion short-hand:

$$\Delta\theta_t = -\frac{\eta}{RMS[g]_t} g_t \quad (3.147)$$

The authors note that the units in this update (as well as in SGD, Momentum, or Adagrad) do not match, i.e. the update should have the same hypothetical units as the parameter. To realize this, they

first define another exponentially decaying average, this time not of squared gradients but of squared parameter updates:

$$E[\Delta\theta^2]_t = \gamma \cdot E[\Delta\theta^2]_{t-1} + (1 - \gamma) \cdot \Delta\theta_t^2 \quad (3.148)$$

The root mean squared error of parameter updates is thus:

$$RMS[\Delta\theta]_t = \sqrt{E[\Delta\theta^2]_t + \epsilon} \quad (3.149)$$

Since $RMS[\Delta\theta]_t$ is unknown, we approximate it with the RMS of parameter updates until the previous time step. Replacing the learning rate η in the previous update rule with $RMS[\Delta\theta]_{t-1}$ finally yields the Adadelta update rule: Since $RMS[\Delta\theta]_t$ is unknown, we approximate it with the RMS of parameter updates until the previous time step. Replacing the learning rate η in the previous update rule with $RMS[\Delta\theta]_{t-1}$ finally yields the Adadelta update rule:

$$\begin{aligned} \Delta\theta_t &= -\frac{RMS[\Delta\theta_{t-1}]}{RMS[\Delta g_t] \cdot g_t} \\ \theta_{t+1} &= \theta_t + \Delta\theta_t \end{aligned} \quad (3.150)$$

With Adadelta, we do not even need to set a default learning rate, as it has been eliminated from the update rule.

RMSProp

RMSprop is an adaptive learning rate method proposed by Geoff Hinton in [43].

RMSprop and Adadelta have both been developed independently around the same time stemming from the need to resolve Adagrad's radically diminishing learning rates. RMSprop in fact is identical to the first update vector of Adadelta that we derived above:

$$\begin{aligned} E[g^2]_t &= 0.9 \cdot E[g^2]_{t-1} + 0.1 \cdot g_t^2 \\ \theta_{t+1} &= \theta_t - \frac{\eta}{\sqrt{E[g_t^2] + \epsilon}} \end{aligned} \quad (3.151)$$

RMSprop as well divides the learning rate by an exponentially decaying average of squared gradients. Hinton et al. suggest γ to be set to 0.9, while a good default value for the learning rate η is 0.001.

Adaptive Moment Estimation (Adam)

Adam, [63], is a method that computes adaptive learning rate for each parameter. Adam keeps an exponentially decaying average of past gradients m_t , similar to momentum. Whereas momentum can be seen as a ball running down a slope, Adam behaves like a heavy ball with friction, which thus prefers flat minima in the error surface. We compute the decaying averages of past and past squared gradients m_t and v_t respectively as follows:

$$\begin{aligned} m_t &= \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t \\ v_t &= \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2 \end{aligned} \quad (3.152)$$

m_t and v_t are estimates of the first moment (the mean) and the second moment (the variance) of the gradients respectively. For shunning the bias of m_t and v_t towards zero, authors [63] introduced the unbiased first and second moments.

$$\begin{aligned} \widehat{m}_t &= \frac{m_t}{1 - \beta_1^t} \\ \widehat{v}_t &= \frac{v_t}{1 - \beta_2^t} \end{aligned} \quad (3.153)$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\widehat{v}_t + \epsilon}} \cdot \widehat{m}_t \quad (3.154)$$

The above equations are fixed by the following two recent observations:

- **Decoupling Weight Decay in Adam:**

In Vanilla SGD Weight Decay regularization is done with adding a scaled L2 regularization loss to the overall loss function, as explained in a previous section. As a result, its gradients would be subtracted by the updated weights to put a brake on their optimization scheme. In Adam, though, the moving gradient average and its squared calculations precede the weight update, with the result of subtracting older weights as well, along with the current one due to derivation. In [78], they understood that mismatch and proposed to overcome it by just add a weight subtraction term at the end of the weight update to release it from the intermediate consequences.

- **Amsgrad:**

An understanding of the Adam proof of convergence misconception led [116] to replace the denominator $E[g^2]_t$ with its max along all the previous optimization steps: $\max_t\{E[g^2]_t\}$ to guide Adam to better local minima.

3.3.22 Scheduler schemes

One of the key hyperparameters to set in order to train a NN is the learning rate for gradient descent. It scales the magnitude of our weight updates in order to minimize the network’s loss function.

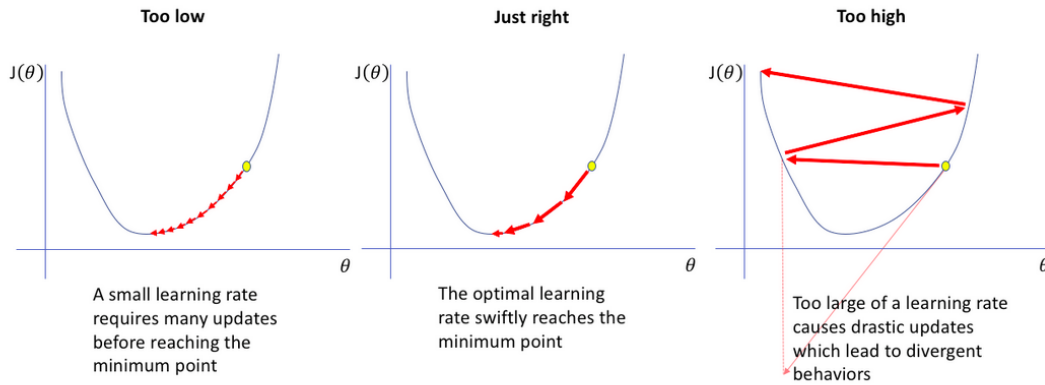


Figure 3.60: The Learning Rate magnitude tradeoff: when the learning rate is too high the optimization process with heavily perturbate and diverge from the function minimum. On the opposite casem where it is too low, the training process will be slow and the optimizer scheme will likely be trapped into local minima that it will not have the power to escape from.

Specifically, while we might think that the adaptivity of Adam’s learning rates might mimic learning rate annealing in Vanilla Gradient Descent, an explicit annealing schedule can still be beneficial: If we add SGD-style learning rate annealing to Adam, it converges faster and outperforms SGD and vanilla Adam, as shown in a variety of publications [[63],[116]].

Learning Rate reduction on Loss Plateaus

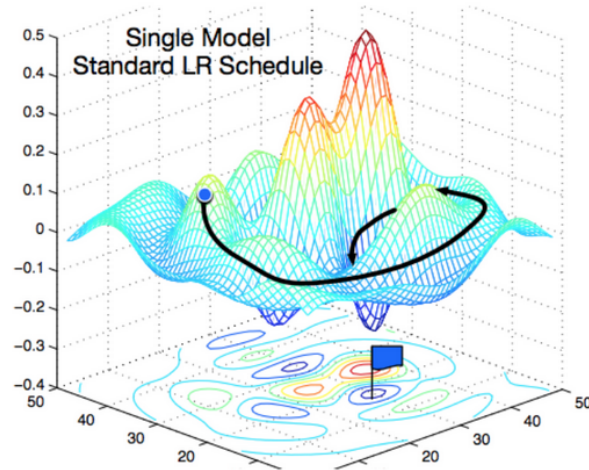


Figure 3.61: The learning rate scheduling policy of reducing its value by an order of magnitude after a number “patience” epochs where the validation loss does not decrease anymore. In that scenario, a single model is kept (that with the lowest validation loss) and at the latter training stages the algorithm heads deeper and deeper to a set (local) minimum with ever-decreasing step.

The first thought passing our mind for tuning the learning rate is to start by assigning a relatively high value to that gradient step variable (endorsing loss landscape **exploration** and give the algorithm the chance to escape shallow local minima) and reduce it by an order of magnitude whenever the loss function measured on the validation set **plateaus** for a specific number of epochs (enforcing this time **exploitation** of the minimum you are heading to). fig.3.61, graphically shows this learning rate scheduling policy.

Learning Rate Scheduler with Warm Restarts

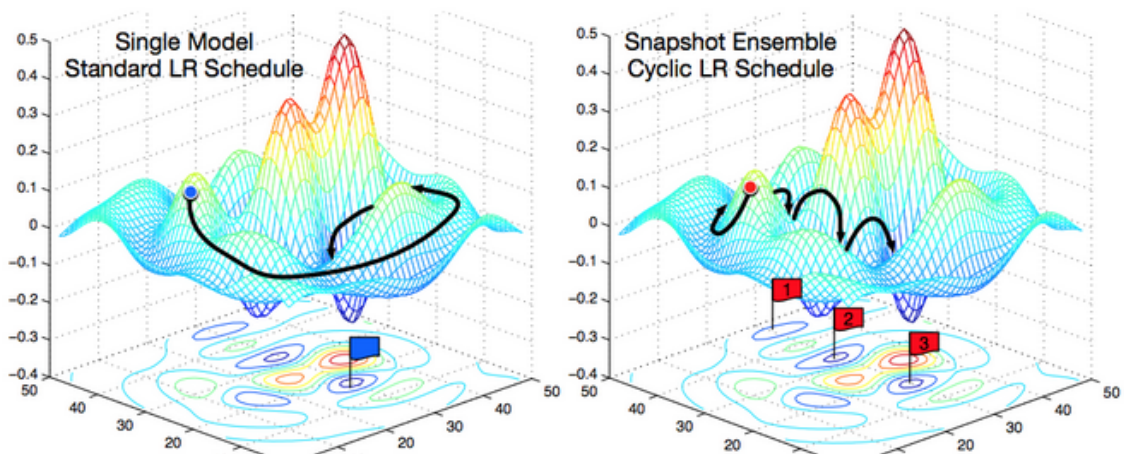


Figure 3.62: **Left:** Illustration of SGD optimization with a typical learning rate scheduling scheme that decreases the gradient descent step when the loss does not decrease anymore. **Right:** The model undergoes several learning rate annealing cycles, escaping this way from a sequence of local minima.[79]

Generally, between two equivalent good local minima, the flatter ones are preferred as they offer both an accurate and stable solution that generalizes better; that is, it has a higher ability to react to new data. In order to find a more stable local minimum or in order to generate enough gradients to overcome saddle points, after some epochs of decreasing it, we may choose to increase the learning rate and then let it decrease again, according to a predefined schedule, encouraging the model to “jump” from one local minimum to another if it is in a steep trough. This modification is called **Warm Restarts of the Learning Rate**. Importantly, the restart is warm as the optimization does not start from scratch

but from the parameters to which the model converged during the last step. The key factor is that the learning rate is increased/decreased with an aggressive cosine annealing schedule, which rapidly lowers/elevates the learning rate.

$$\eta_t = \eta_{min}^i + \frac{1}{2}(\eta_{max}^i + \eta_{min}^i)(1 + \cos(\frac{T_{curr}}{T_i}\pi)), \quad (3.155)$$

where η_{min}^i and η_{max}^i are ranges for the learning rate during the i -th run, T_{curr} indicates how many epochs passed since the last restart, and T_i specifies the epoch of the next restart. The high initial learning rate after a restart is used to essentially catapult the parameters out of the minimum to which they previously converged and to a different area of the loss surface. The aggressive annealing then enables the model to rapidly converge to a new and better solution.

Each of these ‘periods’ of ‘ups’ and ‘downs’ is known as a **cycle**, and in training our neural network, we can choose the number of cycles and the length of each cycle ourselves. In fact, the cycle length is increased after each cycle. Specifically, for Adam, we set $\eta_{min}^{(i)} = 0$ and $\eta_{max}^{(i)} = 1$, which yields:

$$\eta_t = \frac{1}{2}(1 + \cos(\frac{T_{curr}}{T_i}\pi)), \quad (3.156)$$

and multiply the initial epoch range by a factor of T_{mult} .

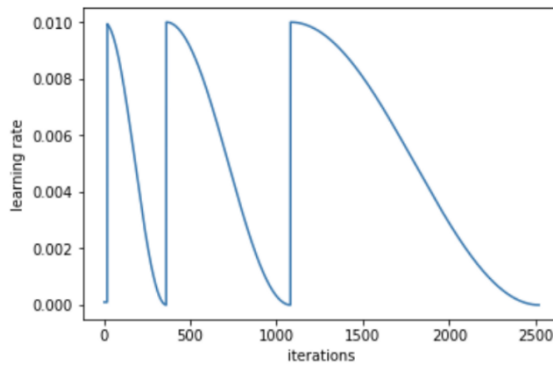


Figure 3.63: Cosine Annealing learning rate schedule with “Warm Restarts” with train cycles increasing each time.[79]

This seems to output a better, more accurate result as it allows us to find the minimum point in a stable region more precisely.

Chapter 4

Related Work

"And following its path, we took no
care
To rest, but climbed: he first, then I—
so far,
Through a round aperture I saw
appear
Some of the beautiful things that
Heaven bears,
Where we came forth, and once more
saw the stars."

Dante Alighieri, "Inferno"

In this section, we report previous attempts in the fields of 6D Object Pose Detection and Tracking (including the ones in the dual problem of Camera Pose Estimation).

4.1 Single-frame Pose Detection

This family of attempts processes every frame of the video separately without any prior feedback knowledge from the previous timestep.

They can be segregated in the following subcategories:

Global - (Feature) Template based:

They are occasioned by psycho-physiological observations and they render synthetic patches from different viewpoints that are distributed in a 3D sphere around the object. Using them as input, handcrafted (e.g. color/depth/surface normal-based histograms) or deep learning-based feature templates are extracted and saved in the database. When a new full test image is imported, image patches are extracted via sliding windows techniques and the patch in the database with the closest pose is selected using Nearest Neighbour searching.

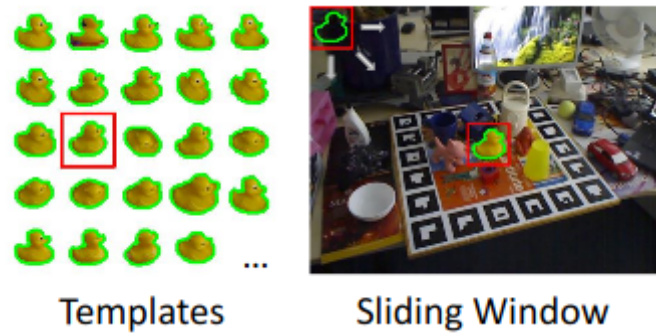


Figure 4.1: An example of the Hintetoisser approach of sliding a 2D-window in an image and compare the convolutional features produced by it with the convolutional features that correspond to a template [112].

Main advantages:

- Small processing times due to small spatial complexity
- They tend to work well for texture-less objects

Main disadvantages:

- Discrete-only pose prediction capability. By discretizing the pose space extra model noise is added to our estimations.
- Sensitivity to occlusions
- They fail to extract salient points.
- In [112], Hintetoisser et al. detected the poses of textureless objects by creating LINEMOD templates for their RGB-D images, with the templates being a combination of gradients and surface normal histograms (see fig.4.1). The poses estimated are far from perfect and therefore there is severe need for extra refinement. Moreover, noise, background clutter and occlusions are not handled in any way in this approach.

Local - Voting based

The object image patch is further divided into smaller patches of its parts from which corresponding features are learned. Then, during inference, voting schemes are employed between the pose hypotheses, every part patch proposes, in order to verify the final one.

Main advantages:

- Efficient occlusion handling as hypotheses of the occluded parts get rejected by the ones proposed by the rest of them.

Main disadvantages:

- They are not ideal for real-time applications as the search time complexity is multiplied.
- They do not take into account global object structure, so they do not, usually, generalize well outside of cases present in their training set.
- In [56] Kehl et al. divided the pose space into a icosaedron geometry from where local RGB-D patches of the various viewpoints were extracted. Using them as inputs to a Convolutional AutoEncoder, they built feature descriptors of the patches that they saved in a database. When a new test image arrives, it is itself divided in patches, for which the closest descriptor of the database is extracted via a random forest search. Then, a 6D voting scheme is utilized to have the final pose estimation result. This is used as a prior to a final iterative pose refiner. With this method they tried to bridge the gap between local and global approaches. The random forest searching process has the ability to combine local information in their leaves with a more general concept in their higher levels. It can also be scaled easily to multiple objects. However, in this case, the complexity of the random forest search increases linearly with the object number exceeding real-time constraints.

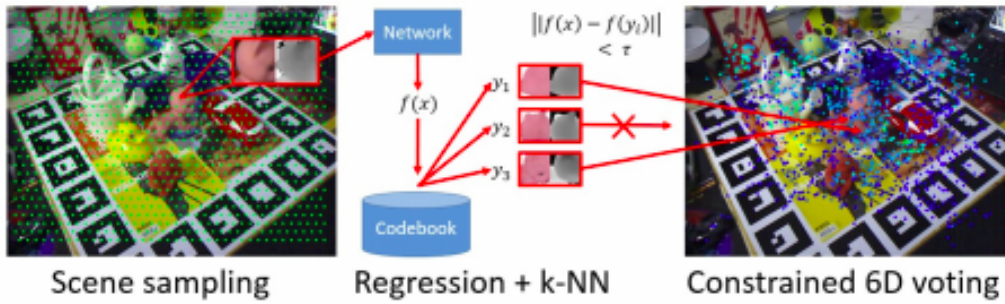


Figure 4.2: Illustration of the voting scheme of [56]. They densely sample the scene to extract scale-invariant RGB-D patches. These are fed into a network to regress features for a subsequent kNN search in a codebook of pre-computed synthetic local object patches. The retrieved neighbors then cast 6D votes if their feature distance is smaller than a threshold τ . [56]

- More recently, in PVNet [102] and PVNet3D [40] approaches performed per-pixel voting-based regression to match 3D object coordinates with predefined keypoints inside the object surface, effectively handling occlusions in this manner.

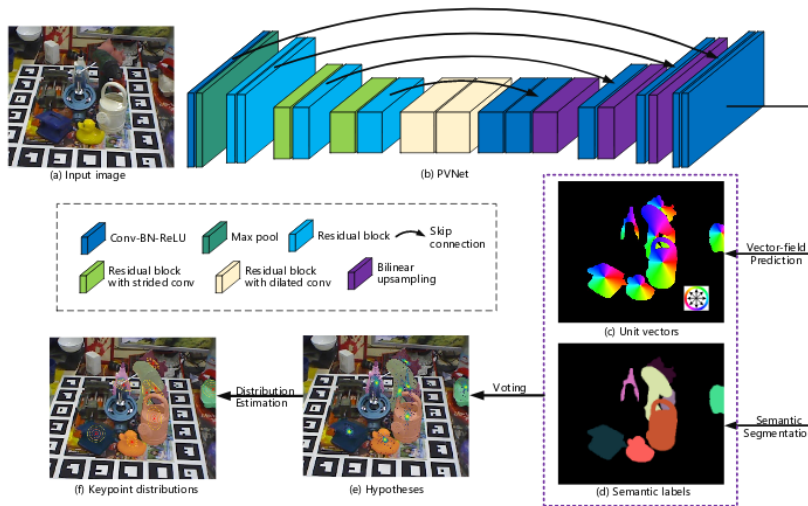


Figure 4.3: PVNet: Overview of the keypoint localization: (a) An image of the Occlusion LINEMOD dataset. (b) The architecture of PVNet. (c) Pixel-wise unit vectors pointing to the object keypoints. (d) Semantic labels. (e) Hypotheses of the keypoint locations generated by voting. The hypotheses with higher voting scores are brighter. (f) Probability distributions of the keypoint locations estimated from hypotheses. [102]

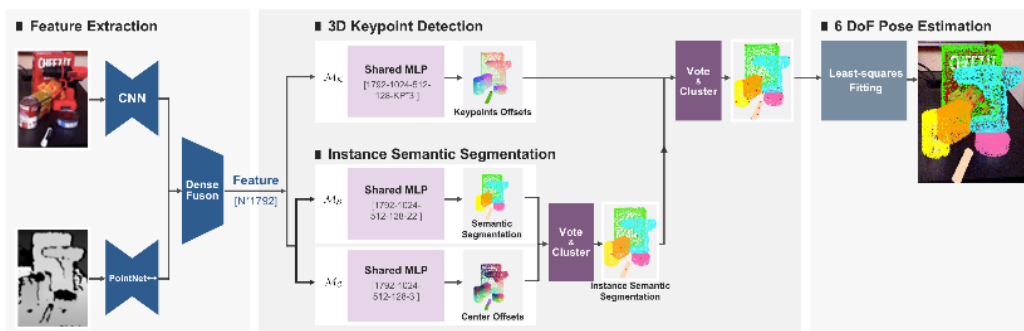


Figure 4.4: PVNet3D: The Feature Extraction module extracts the per-point feature from an RGBD image. They are fed into the modules: M_K , M_C and M_S to predict the translation offsets to keypoints, center point and semantic labels of each point, respectively. A clustering algorithm is then applied to distinguish different instances with the same semantic label and points on the same instance vote for their target keypoints. Finally, a least-square fitting algorithm is applied to the predicted keypoints to estimate 6DoF pose parameters. [40]

2D-3D Correspondence based using PnP(Perspective 'n' Projection)+RANSAC(RANdom SAmpled Consecus)

The P'n'P algorithm:

These methods, in their latter stage, they assume to have a set of known 3D Object Points(Coordinates), there equivalent 2D pixel projections and a calibrated camera (K is known) in order to estimate the rigid Transformation $T_{(c)}^{(O)}$ (for Camera Pose Estimation) or $T_{(O)}^{(C)}$ (for Object Pose Estimation). Using it we form the Projection Matrix \mathbb{P} :

$$\mathbb{P} = K \begin{bmatrix} R & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \quad (4.1)$$

(let's assume we are talking for an assymmetric object ($R_{sym} = \mathbb{I}_3$) without loss of generality.)

Then, the Projection equation is:

$$\mathbb{P}\mathbb{X} = \lambda\mathbf{x}, \quad (4.2)$$

where P is the 3x4 Projection Matrix, and $\mathbb{X} \in \mathbb{R}^4$, $\mathbf{x} \in \mathbb{R}^3$ the Homogenous 3D Object point and image pixel, accordingly.

Since, the 3D point is projected onto the 3D image plane:

$$[\mathbf{x}]_{\times}\mathbb{P}\mathbb{X} = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}_{\times} \begin{bmatrix} \mathbf{P}_1^T \\ \mathbf{P}_2^T \\ \mathbf{P}_3^T \end{bmatrix} \mathbb{X} = \mathbf{0}. \quad (4.3)$$

More analytically,

$$\begin{bmatrix} 0 & -1 & v \\ 1 & 0 & -u \\ -v & u & 0 \end{bmatrix} \begin{bmatrix} \mathbb{X}^T & \mathbf{O}_{1 \times 4} & \mathbf{O}_{1 \times 4} \\ \mathbf{O}_{1 \times 4} & \mathbb{X}^T & \mathbf{O}_{1 \times 4} \\ \mathbf{O}_{1 \times 4} & \mathbf{O}_{1 \times 4} & \mathbb{X}^T \end{bmatrix} \begin{bmatrix} \mathbf{P}_1^T \\ \mathbf{P}_2^T \\ \mathbf{P}_3^T \end{bmatrix} = \mathbf{0}_{3 \times 1} \quad (4.4)$$

The above equation applies for every 3D-2D point projection combination. Every 3D Object (Coordinate) Point corresponds to 2 pixel constraints (u,v), so in order for the above linear system to be well-defined, we need to know at least 6 3D Points, as they would bring $6 \times 2 = 12$ pixel constraints with them that will be enough for us to estimate the 12 parameters of \mathbb{P} .

If we stack the equations (4.4) for all 6 of them points, we will result in a homogenous linear system:

$$A_{18 \times 12} \mathbf{p}_{12 \times 1} = \mathbf{0}_{12}, \quad (4.5)$$

which is solved using an SVD decomposition of A (see Section 4.1):

$$SVD(A) = U\Sigma V^T, \quad (4.6)$$

with the required 12-d vector of Projection parameters \mathbf{p}^* being the last column of V^T .

Finally, since the Projection parameters are now known, one would assume that if we follow the Projection formula:

$$\mathbb{P} = K \begin{bmatrix} R & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \quad (4.7)$$

and by left-multiplying with the inverse of K:

$$K^{-1}\mathbb{P} = \begin{bmatrix} R & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \quad (4.8)$$

we could get the translation and rotation pose components just by equating columns. However, that would violate the requirement of R belonging in $SO(3)$. In order to ensure that this is the case, we calculate another SVD decomposition, this time of matrix R:

$$SVD(R) = U_R \Sigma_R V_R^T \quad (4.9)$$

and R is acquired as a product of the unitary matrices U, V^T (which is also unitary due to the closing property of $SO(3)$):

$$R^* = U \cdot V^T. \quad (4.10)$$

As for the translation:

$$\mathbf{t}^* = \begin{bmatrix} K^{-1} \frac{\mathbf{p}_x}{d_x} \\ K^{-1} \frac{\mathbf{p}_y}{d_y} \\ K^{-1} \frac{\mathbf{p}_z}{d_z} \end{bmatrix}, \quad (4.11)$$

where $\Sigma_R = \begin{bmatrix} d_x & 0 & 0 \\ 0 & d_y & 0 \\ 0 & 0 & d_z \end{bmatrix}$.

RANSAC:

Approaches that finally use the P'n'P algorithm to estimate the object pose are, on one hand, computationally efficient, but on the other one, they suffer from severe sensitivity to outliers coming from the previous point correspondence estimation stages. For this reason, they are enhanced with a RANSAC method. This means that the P'n'P process will be repeated many times for many sets of 3D-2D point correspondences giving a corresponding pose hypothesis for each one. However, since we are talking about rigid objects, the best hypothesis is unique, so we will verify one from the pool as the best, when its projection on the 2D image plane gives the largest set of inlier pixels.

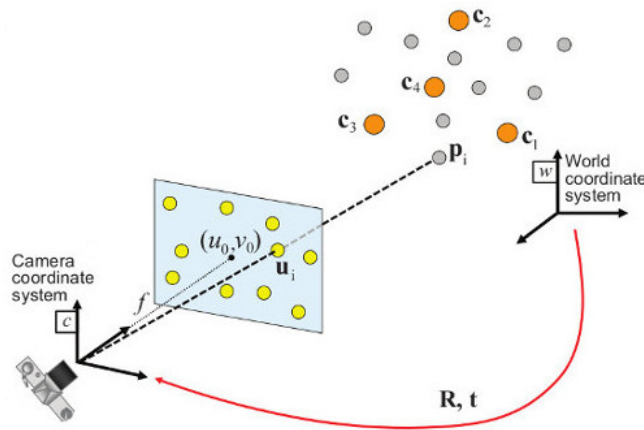


Figure 4.5: The P'n'P Pose Estimation algorithm [71]

In their former stages, such approaches on image-based application require a prior 2D-3D correspondence regression scheme that will predict 3D Object (Coordinate) Points on which the P'n'P algorithm is executed.

Main advantages:

- Robustness to occlusions as the Object Coordinates, they predict live in the 3D space, where the Object of interest and its occluder can be separated.
- Efficient pose computation as small amount of samples and in-sample 3D points are required to generate a pose hypothesis.

Main disadvantages:

- They only depend on the object shape, neglecting the image appearance information, which leaves them sensitive to symmetries.

- P'n'P variations cannot be used by themselves, as they require a prior estimation of the 3D Object Coordinates (when the data we deal with are images and not 3D representations (e.g. Point Clouds)). So, as explained above, they are extremely sensitive to prior errors and outliers harming the pose accuracy. Though, if RANSAC is employed to improve their generalization capabilities, the algorithm needs multiple samples in expense of real-time speed.
- When they try to reconstruct the object's 3D Coordinates they often execute Regression and Segmentation (from other objects/background/occluders) on the same step, which has been shown to undermine their performance.

Two examples where the P'n'P refinement is put on top of a prior coarse estimation:

- In [8], Brachmann et al. used Random Forests to Recognize the Object Class and Regress the 3D Object Coordinates necessary by minimizing an Energy function.

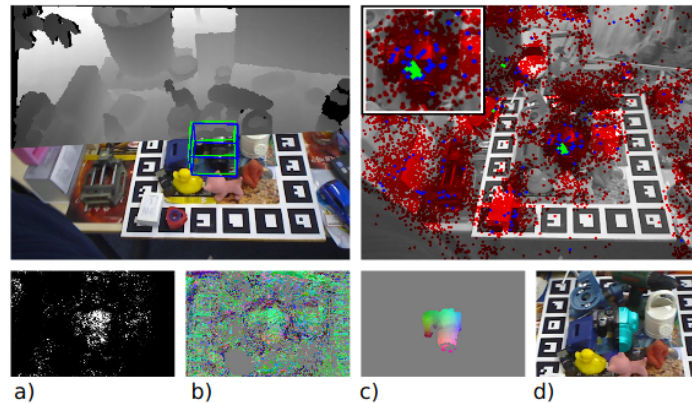


Figure 4.6: **Top left:** RGB-D Test image (upper-half depth image and lower-half RGB image). The estimated 6D pose of the query object (camera) is illustrated with a blue bounding box, and the respective ground truth with a green bounding box. **Top right:** Visualization of the algorithm's search for the optimal pose, where the inset is a zoom of the centre area. The algorithm optimizes the energy function in a RANSAC-like fashion over a large, continuous 6D pose space. The 6D poses, projected to the image plane, which are visited by the algorithm are color coded: red poses are disregarded in a very fast geometry check; blue poses are evaluated using our energy function during intermediate, fast sampling; green poses are subject to the most expensive energy refinement step. **Bottom, from left to right:** (a) Probability map for the query object, (b) predicted 3D object coordinates from a single tree mapped to the RGB cube, (c) corresponding ground truth 3D object coordinates, (d) overlay of the 3D model in blue onto the test image (rendered according to the estimated pose) [8].

- In [69], Krull et al. did the above regression using a CNN which also provided Object Probabilities and Depth estimation.

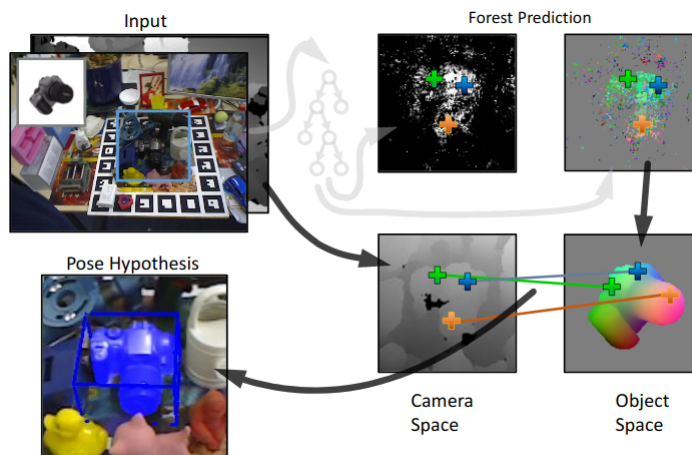


Figure 4.7: An overview of the approach proposed by Krull et al. (it is similar to the one previously proposed by Brachman et al.). A feature extractor (either a CNN or a Random Forest structure) is used to map RGB-D pairs to 3D Object Coordinates, while keeping their correspondences in the 2D image plane. These correspondences are then fed to P'n'P module with RANSAC hypothesis evaluation [69].

Deep Learning-based

Convolutional Neural Networks have shown to serve as very efficient and robust feature extractors. Because of the fact that their features are learned with the sole target of Pose Estimation, without intermediate handcrafted ones, they are expected to leverage both of the previous methods qualities.

There are two philosophies here that should be mentioned: the first one tries to organize the feature space in an underlying manifold-like manner explicitly, in order to resemble the structure of pose space. In order to achieve something like that, it minimized a **Contrastive** Loss function (either supervised or unsupervised) that is based on minimizing feature distances between “anchor” and “positive” samples and maximizing the corresponding metrics between the same “anchor” and “negative samples”. On the other hand, the second school of thought lets the Deep Network free of such constraints.

Contrastive/Metric learning-based

In such approaches, extra terms are added to the loss function that aim to produce similar feature representations for similar poses and dissimilar feature representations for dissimilar poses (and/or objects) in an explicit way. This way the network learns low-dimensional viewpoint descriptors and later, k-Nearest Neighbours are used to search for the optimal pose in this feature space.

Main advantages:

- They embed an holistic approach of the object structure.

Main disadvantages:

- Fails increase under clutter and occlusion conditions.
- In [161], Wolhart et al. minimized a Triplet Loss and Pair Loss combination with a CNN to extract feature descriptors for every object viewpoint present in the training dataset. In particular, the same objects in dissimilar poses would give small distances and different objects would result in big distance metrics. Those descriptors are then used for Nerest Neighbour(NN) seaching to estimate the poses of multiple objects. It has the advantage of $O(1)$ NN searching, even for multiple objects. Its main disadvantages are the fact that it does not take in-plane rotations into consideration, the pose space discretization, while Euclidean distances are employed to measure differences between poses, i.e. elements that lie in a non-Euclidean manifold.

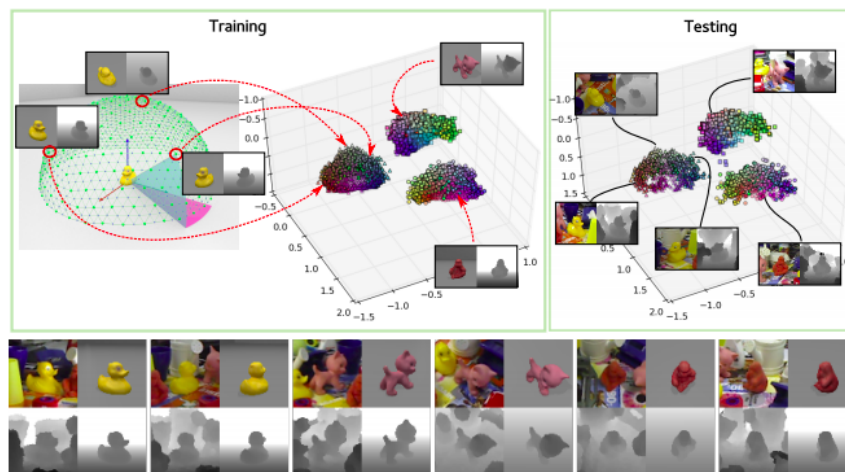


Figure 4.8: Three-dimensional descriptors of [161] for several objects under many different views computed by this method on RGB-D data. **Top-left:**The training views of different objects are mapped to well-separated descriptors, and the views of the same object are mapped to descriptors that capture the geometry of the corresponding poses, even in this low dimensional space. **Top-right:** New images are mapped to locations corresponding to the object and 3D poses, even in the presence of clutter. **Bottom:** Test RGB-D views and the RGB-D data corresponding to the closest template descriptor.[161]

- In [12], Bui et al. combined a similar manifold feature learning approach (for Depth maps only) with direct pose regression, by substituting the pair (discrete) loss with the latter, for more refined

estimations. In this way, they augmented the feature clustering capability of the network by boosting the following search’s accuracy. There, they also explicitly map images of different kind (real/synthetic) to the same object-pose configuration for domain adaptation and for a background variety. Their generalization performance is pretty good, they solve the pose ambiguity in an explicit way and they solve the problem of in-plane rotations by embedding their triplet loss with a dynamic margin depended on the negative sample of each triplet. On the contrary, their approach does not scale to new objects and it is too slow because of the employment of kNN searching.

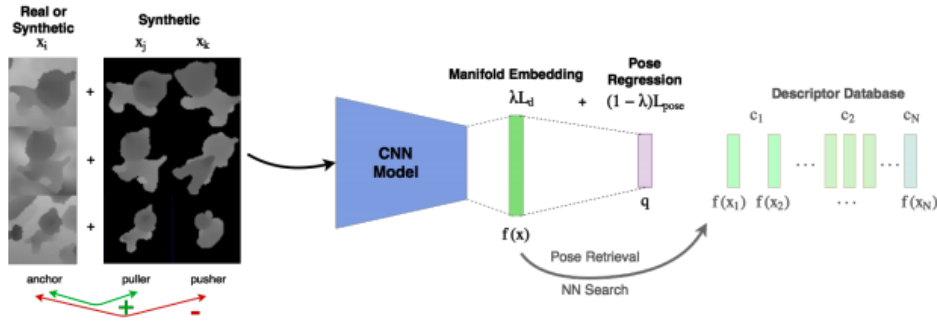


Figure 4.9: Given an input depth image patch x_i , we create corresponding triplets (x_i, x_j, x_k) and pairs (x_i, x_j) to optimize our model on both manifold embedding, creating robust feature descriptors, and pose regression. Obtaining either a direct pose estimate or using the resulting feature descriptor for nearest neighbor search in the descriptor database.[12]

- In [3], Balntas et al. repeated a similar combination of manifold learning and pose regression that enhances the distinctive capabilities between elements that are close in the pose space, but this time they employed a weighting strategy that reflects the degree of symmetry of each object. No neighbour-based searching was used there increasing the algorithm’s speed.

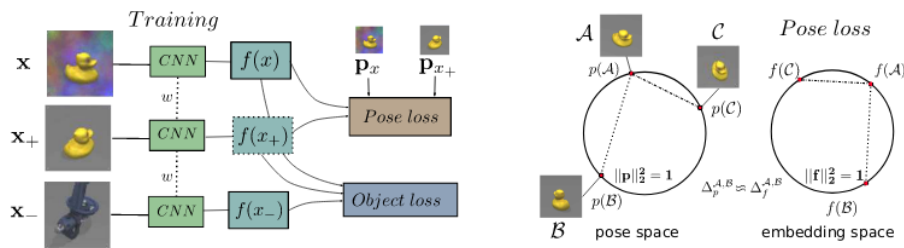


Figure 4.10: (left) The overview of the method of Balntas et al. A triplet network learns embeddings suitable for both object recognition and exact pose retrieval (right) Illustration of the proposed pose loss. Intuitively, the method aims to train a CNN in such a way that the distance in the resulting D-dimensional embedding space is analogous to the pose differences.[3]

Deep Neural Networks for immediate Regression/Classification:

Main advantages:

- They encode information in a hierarchical way, gradually fusing low-level features to derive high-level semantics.
- They tend to generalize better to samples outside of the training set and to noise, clutter and occlusion conditions.
- They are equivariant to change in appearance, scale and small translations.

Main disadvantages:

- They require big amounts of high-precision, annotated data which are often expensive to acquire.
- They are really slow to train.

- In PoseCNN [162], the authors constructed a single-stream CNN that firstly estimates binary object masks and then predicts the object class, translation and rotation separately. It has the pro of not needing to know a-priori the object numbers and identity/3D model, but also the con of being too deep and slow for a real-time application.

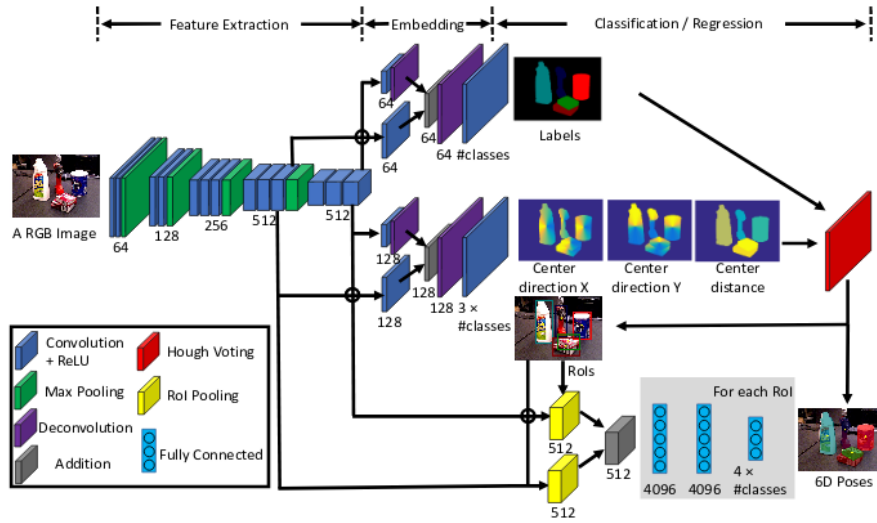


Figure 4.11: Architecture of PoseCNN for 6D object pose estimation[162]

- In SSD6D [58], they extended the SSD framework [77] for 2D Object Detection in 6D as follows. They sampled the scene (synthetic train image samples) in dense patches and then they used a pretrained InceptionNet [127] to produce multiple overlapping bounding box hypotheses. They performed 2D-to-3D unprojection to each of them and they used a neural network to perform discrete viewpoint classification for some known object cases. Finally, they refined the estimations produced with some ICP iterations (see sec.4.3). They succeeded in handling viewpoint-induced symmetries as they trained only on views of a hemisphere and they refined the, initially crude, rotation angles, but the refinement process of their approach seems to be necessary for it to function well, making the whole framework really slow (about 24ms/frame). If the refinement stage is neglected though, the discrete poses, in which the test samples are clustered, insert extra prediction noise, being crude estimations of the real ones.

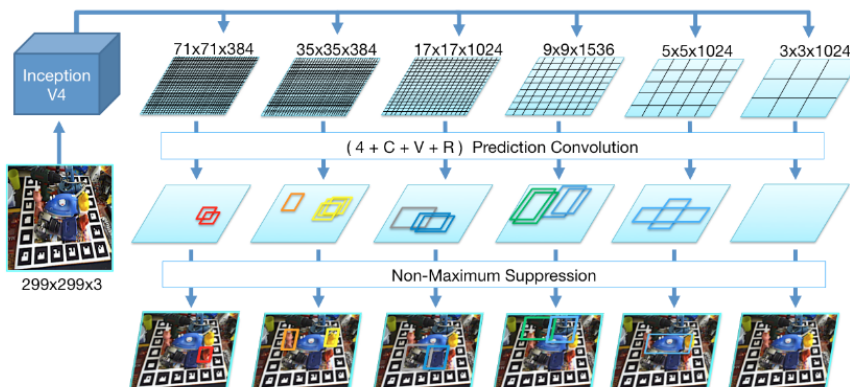


Figure 4.12: Schematic overview of the SSD-style network prediction. We feed our network with a 299×299 RGB image and produce six feature maps at different scales from the input image using branches from InceptionV4. Each map is then convolved with trained prediction kernels of shape $(4 + C + V + R)$ to determine object class, 2D bounding box as well as scores for possible viewpoints and in-plane rotations that are parsed to build 6D pose hypotheses. Thereby, C denotes the number of object classes, V the number of viewpoints and R the number of in-plane rotation classes. The other 4 values are utilized to refine the corners of the discrete bounding boxes to tightly fit the detected object. [58]

- In “BB8” [106], (trained on real data), Rad et al. cascaded a coarse to fine segmentation CNN architecture that returns 2D projections of 3D bounding boxes as 2D-3D correspondences. Those correspondences are the input of a following P’n’P algorithm with RANSAC pose verification. This approach is really slow, using a very deep state-of-the-art CNNs for Image Segmentation, does not fully extract the object background in the former layers and is sensitive to occlusions because it uses as inputs for the P’n’P algorithm the sparse correspondences of the bounding boxes instead of ones from the whole object surface.
- In YOLO6D [130], they extended the popular YOLO single-shot framework [110] for 2D Object Detection by allowing image grids to predict both the object’s instance and 6D pose. Their attempt is compact and efficient and it produces accurate poses without any refinement, as it does not discretize the pose space. On the other hand, it is sensitive to occlusions as it doesn’t handle them in any explicit way.

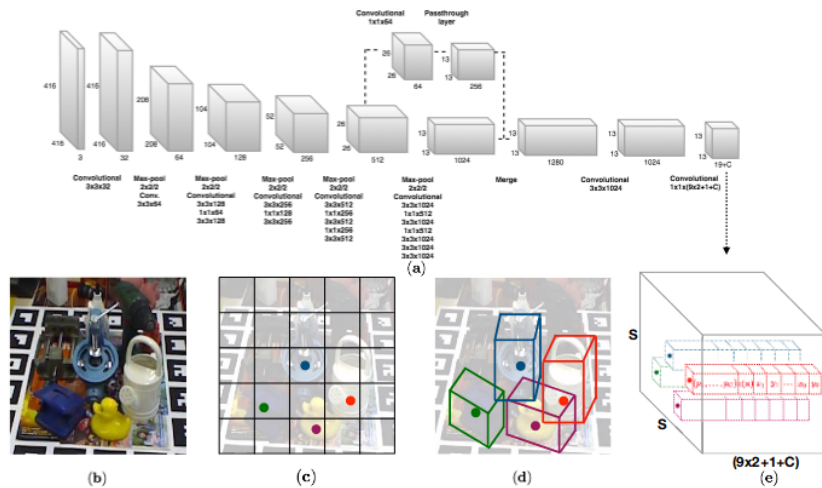


Figure 4.13: Overview of YOLO6D [130]: (a) The proposed CNN architecture. (b) An example input image with four objects. (c) The $S \times S$ grid showing cells responsible for detecting the four objects. (d) Each cell predicts 2D locations of the corners of the projected 3D bounding boxes in the image. (e) The 3D output tensor from our network, which represents for each cell a vector consisting of the 2D corner locations, the class probabilities and a confidence value associated with the prediction. [130]

- In [126], the authors use the 2D bounding box predictions from an SSD [77] as an input to an Augmented AutoEncoder that predicts 3D object rotations in an unsupervised way, using only synthetic data. However, it can not be used online using conventional hardware as it is too slow (200ms/frame).

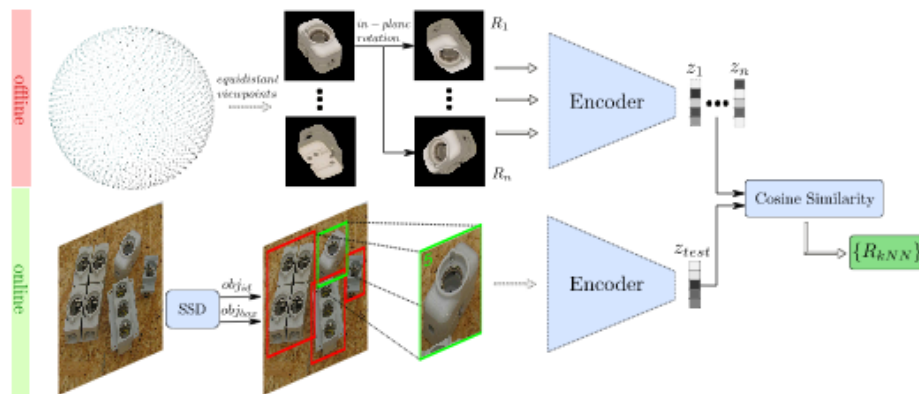


Figure 4.14: Training process for the architecture of [126]; a) reconstruction target batch of uniformly sampled $SO(3)$ object views; b) geometric and color augmented input; c) reconstruction after 30000 iterations. [126]

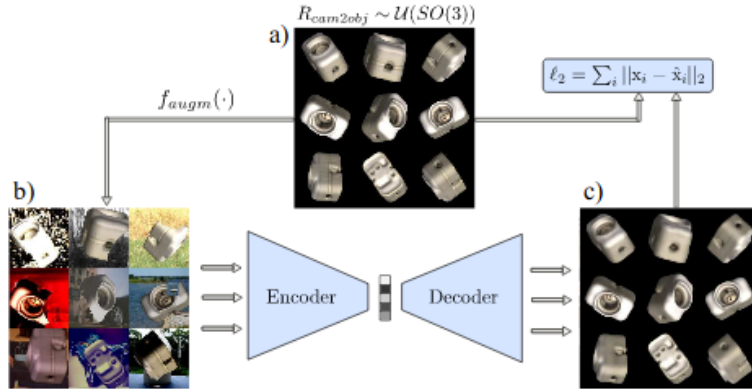


Figure 4.15: **Top:** creating a codebook from the encodings of discrete synthetic object views; **Bottom:** object detection and 3D orientation estimation using the nearest neighbor(s) with highest cosine similarity from the codebook of [126]

- Furthermore, iPose [48] is one of the closest attempts to ours. There, they segment a binary object mask with a pretrained MaskRCNN [39] to extract background clutter and occluders and they map 2D pixels to dense 3D Object Coordinates with an Encoder-Decoder (minimizing a Huber loss [50]), which, in turn, are used as input for a geometric optimization performer by a P’n’P algorithm. The main disadvantage of this approach is the need for too many layers used in MaskRCNN, a result of the requirement for a hard binary mask segmentation, which is not much more than an Auxiliary Task. In our approach, we will show that relaxing this constraint produces impressive results with very fewer parameters and by handling the above challenges in an explicit way. It should, also, been noted that the 3D Coordinates regression distills the appearance information, keeping only the shape one, which makes the approach sensitive to object symmetries.

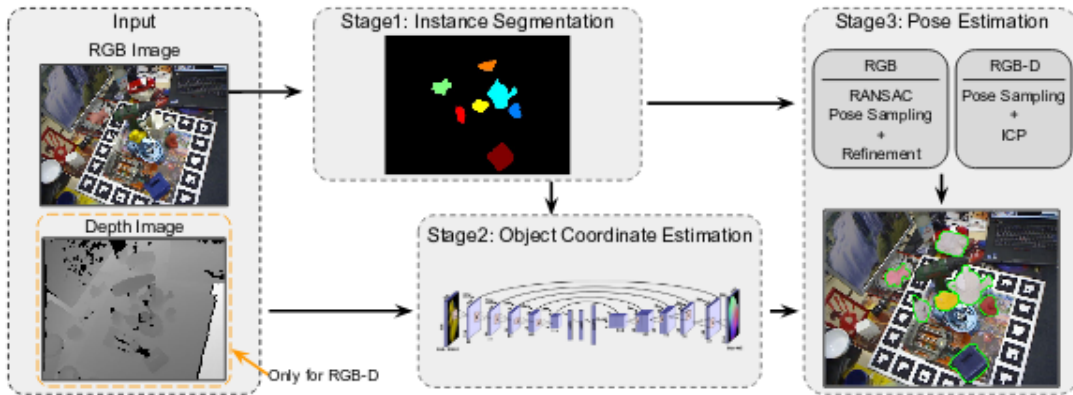


Figure 4.16: Illustration of our modular, 3-stage pipeline for both RGB and RGB-D input images of the architecture of [48]

- DPOD, [167], inputs a combination of real and synthetic RGB images to 12-layer ResNet [37] encoder, pretrained on ImageNet [42], [68], to segment a pixel-wise prediction of the object ID in a mask-level. Following, multi-class UV texture maps are estimated to extract the dense 2D-3D Normalized Object Correspondences [149] between 2D images and 3D CAD object models using Cross Entropy Losses. Finally, those correspondences are used for Pose Estimation via a P’n’P algorithm with RANSAC pose verification strategy. This estimation is then used as a prior estimation to a 2-stream Convolutional refinement network that outputs the final pose prediction (with the translation component prediction being first and used as extra information for the rotation one). This approach works very well for the difficult case of small objects, it is twice as fast (13ms/frame + 5ms for refinement) as SSD6D and 5 times as AAE [126]. By predicting the ID masks and the UV texture maps, the algorithm succeeds to minimize of the outliers for P’n’P as it ensures that 2D-3D correspondences almost always belong in the object surface. Furthermore,

the choice to regress the 2D maps instead of 3D Object Coordinates gives it the chance to use the more stable Cross Entropy loss, resulting in easier network convergence. However, it is far from real-time performance.

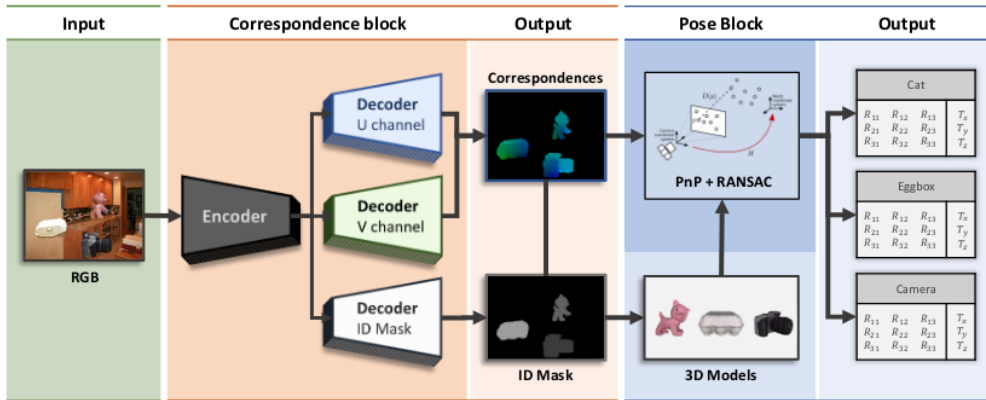


Figure 4.17: Pipeline description of DPOD [167]: Given an input RGB image, the correspondence block, featuring an encoder-decoder neural network, regresses the object ID mask and the correspondence map. The latter one provides us with explicit 2D-3D correspondences, whereas the ID mask estimates which correspondences should be taken for each detected object. The respective 6D poses are then efficiently computed by the pose block based on PnP+RANSAC[[71],[25]]. [167]

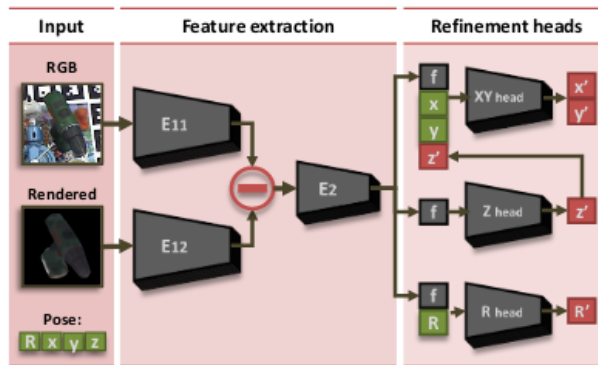


Figure 4.18: Refinement module of DPOD [167]: The network predicts a refined pose given an initial pose proposal. Crops of the real image and the rendering are fed into two parallel branches. The difference of the computed feature tensors is used to estimate the refined pose. [167]

- In Pix2Pose [98], Park et al. train an adversarially[32] guided Encoder-Decoder on real images, with textureless objects, to predict pixel-wise coordinates in a given image and then feed them to a P'n'P algorithm. They handle occlusions by predicting not only the unoccluded, but the occluded pixel region of the image, as well and they mitigate symmetry's effects by training the network with a transformer loss function that guides the prediction to the closest of the two symmetric poses, each time an ambiguity occurs. Although they report State-of-the-Art performance in a variety of datasets, their speed is far from real-time, as many deep NN components are employed in sequence.

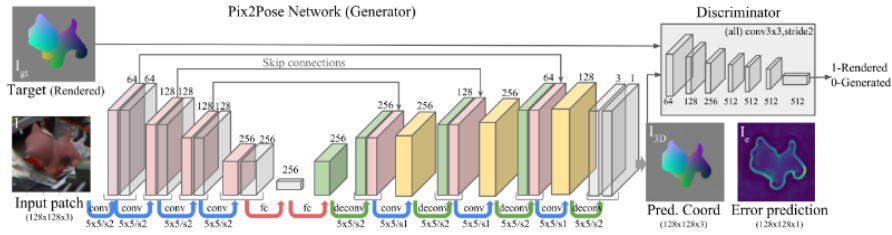


Figure 4.19: An overview of the architecture of Pix2Pose object coordinate regression pipeline.[98]

- In [101], Pavlakos et al. e. extracted semantic keypoints in a single RGB image, both with textured and textureless objects, using a CNN architecture and incorporated them in a deformable shape model in order to estimate the 6D object pose.

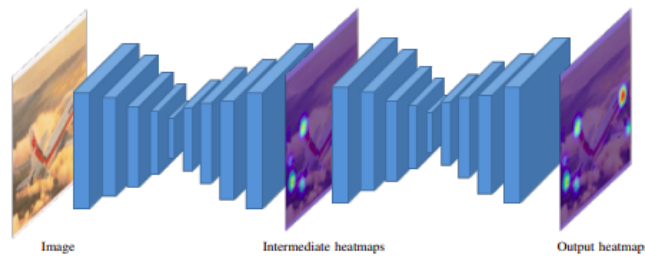


Figure 4.20: Overview of the stacked hourglass architecture for predicting the semantic keypoints. Here, two hourglass modules are stacked together. The symmetric nature of the design allows for bottom-up processing (from high to low resolution) in the first half of the module, and top-down processing (from low to high resolution) in the second half. Intermediate supervision is applied after the first module. The heatmap responses of the second module represent the final output of the network that is used for keypoint localization [101]

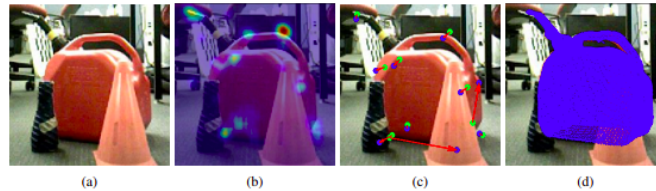


Figure 4.21: Pipeline of the approach of Pavlakos et al. Given a single RGB image of an object (a), they localized a set of class-specific keypoints using a CNN with the stacked hourglass design of fig4.20. The output of this step is a set of heatmaps for each keypoint, which are combined for visualization in (b), sometimes leading to false detections. In (c), green dots represent the detected keypoints and the corresponding blue dots (connected with an arrow) the groundtruth locations. For robustness against such localization errors, they solved a fitting problem to enforce global consistency of the keypoints, where the response of the heatmaps is used as a measure of certainty for each keypoint. The optimization recovers the full 6-DoF pose of the object present in the image (d).[101]

- In [28], Gao et al. used an RGB-D image pair, along with an object instance segmentation mask, to reconstruct point clouds of the target objects. Those coloured point clouds were, at first, normalized w.r.t. their 3D translation (which was considered to be known) and then were processed by a PointNet [104] neural architecture like the one in fig.4.23 to estimate the object's 3D Rotation. The authors claim that this method makes the pose estimation robust to intra-object occlusions.

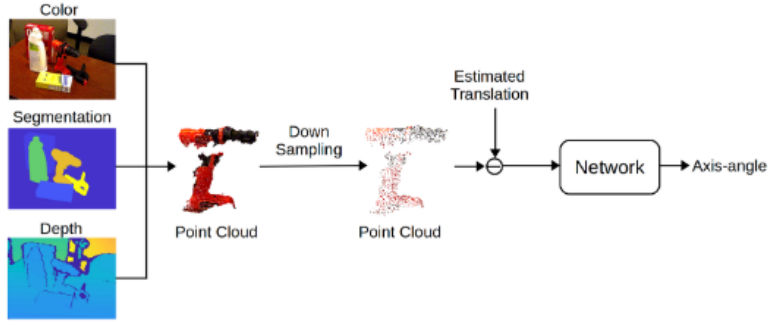


Figure 4.22: System overview. The color and depth images together with a segmentation mask of the target object are used to create a point cloud. The segment is randomly downsampled, and the estimated translation of the down-sampled segment is removed. The normalized segment is fed into a network for rotation prediction, using an Axis-Angle representation.[28]

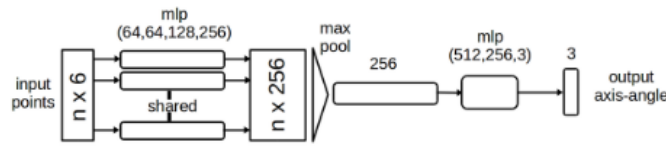


Figure 4.23: Overview of the PointNet architecture.[104]

- In PointPoseNet [35], Hagelskjaer et al. proposed a point cloud based CNN for 6D object pose estimation, which was trained to perform two tasks: 3D point cloud segmentation and unit vectors prediction. These unit vectors generated a bunch of pose hypotheses and a scoring mechanism was employed to choose the best of them.

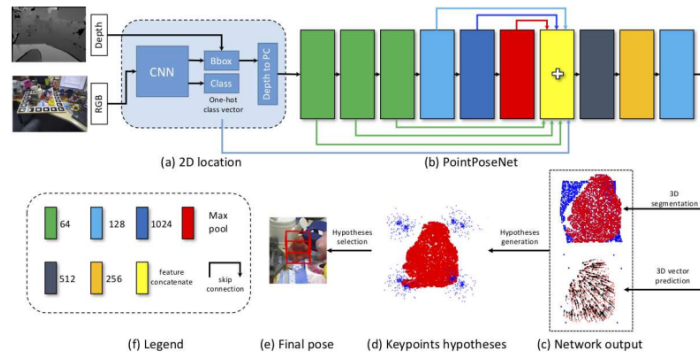


Figure 4.24: Overview of the proposed pipeline. (a) Given an RGB image, we use CNN to detect the bounding box of the target object and the object label which is used as one-hot feature for PointPoseNet. (b) Given the point clouds in the target region, we use proposed PointPoseNet to do 3D segmentation and vector prediction. (c) Top: 3D mask for target object; bottom: Point-wise unit vectors pointing to the keypoint. (d) 3D keypoints hypotheses generated from the unit vectors. (e) Final pose after hypotheses selection. (f) Legend of this figure. The number is the output channel of the corresponding layer.[35]

- In [133], Tian et al. disentangle the 3D translation and rotation estimations. For the latter, they try to resolve the local-optimum problem of rotation regression by uniformly sampling rotation anchors in $SO(3)$, and predict a constrained deviation from each anchor to the target, as well as uncertainty scores for selecting the best prediction. As far as it concerns the former, the object 3D position is detected by aggregating point-wise vectors pointing to the 3D center.

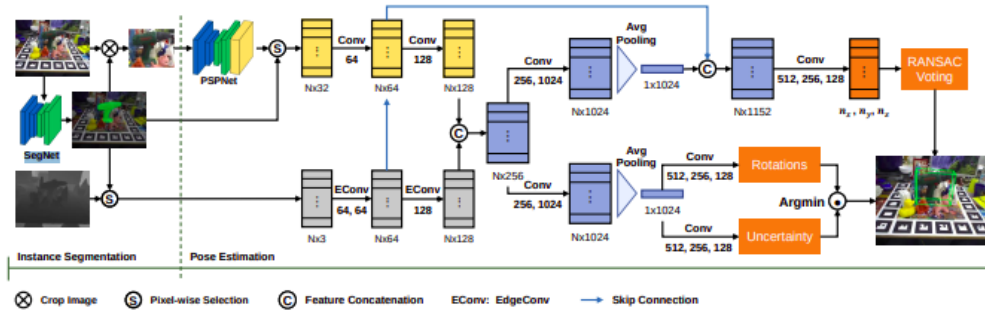


Figure 4.25: Overview of the pipeline. Instance masks are obtained at first through a semantic segmentation network (SegNet). With each obtained mask, the pose estimation network extracts a point cloud from depth image and crops a tight image window from color image as inputs. Color features and geometric features are densely concatenated, from which rotation and translation of the object are predicted in two separate branches.[133]

4.2 Camera Pose Estimation

It is the dual problem of Object Pose Estimation as instead of trying to extract $T_{(O)}^{(C)}$, we estimate $T_{(c)}^{(O)} = (T_{(O)}^{(C)})^{-1}$. So, we find essential to report the progress in this field as well, as it can automatically be reproduced in our scenario.

- Specifically, in PoseNet [61], Kendal et al. used a CNN on single images to regress the camera translation with an MSE loss and the rotation with a geodesic loss on the quaternion space.

4.3 Pose Matching/Refinement - Tracking

The third big category of Pose Estimation approaches that we need to study is tracking the pose in the time domain, using the prediction of the previous timestep as a prior to predict the current one or refine it on the loop domain, where an initial pose guess is improved iteratively. Methods of both can be interchanged as their only difference is that domain field, so let's only discuss the Pose Tracking problem. Here, instead of single-frame Pose Detection the information is utilized specifically via a feedback process, allowing to skip steps required without prior knowledge of the previous pose resulting in faster processing and real-time performance.

Pointset Registration

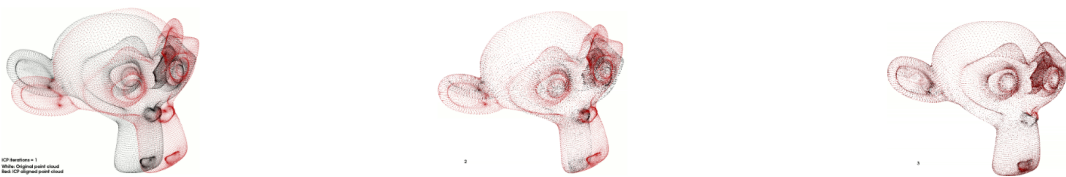


Figure 4.26: A graphical representation of the Iterative Closest Points iterations. [119]

In such approaches, the relative pose transformation is defined as the one that matches a source pointset (or Point Cloud) to a target one in an optimal way. The source is extracted from the current scene configuration while the target from its 3D (object or scene) model scanning at the previous timestep. The most famous such algorithm is named Iterative Closest Points [119] which is consisted of 3 particular steps that get iterated until a certain accuracy threshold is passed.

- Match each source point cloud element with the closest one of the target.
- Find the translation and rotation pair that minimizes the mean squared error between the above samples.
- Transform the source point cloud according to the Pose Transformation estimated in step 2.

Many alternatives of the ICP algorithm have been proposed throughout the years with their attempts for improvement focused mainly on the initial point sampling, the matching technique variations, the error metric and its minimization optimizer and some extra tricks to weigh points of different importance to increase robustness to outliers. Some well known ICP variations are Signed Distance Function [16], Coherent Point Drift [88], Robust Point Matching [5] and kernel correlation-based methods [138].

Main disadvantages:

- It is heavily depended on the initial estimation, which is hard to find, especially for fast moving objects.
- Increased sensitivity to occlusions, especially in high percentages. Then there is a shortage of points of the CAD 3D model to be matched. While some approaches have tried to handle this by using the estimate of the previous timestep(s) when the object is heavily occluded, this is only a half-measure as extra drift is added to the estimation and there is the hazard of matching point of the model with elements of the background.
- In the case of multiple object tracking, especially when they have similar shapes and their pointsets are intersected, there is a possibility of matching the points between different object models.
- As temporal motion of the object is not taken into account, pose estimations are not smooth through time resulting in abrupt changes from one timestep to the next.
- In the case of image-based scenarios, where the point cloud representation is not taken from a sensor monitoring the scene, but is estimated in a previous stage as 3D Object (Coordinate) Points, errors appeared in this prior step are transferred to the ICP-like estimation as well.

Particle Filtering - Recursive Bayesian Estimation

They consider the object pose as the hidden state of a Hidden Markov Model (HMM) and the image information as the observation state. They aim to estimate the optimal hidden state with regard to the current and the previous observations.

- In [129], they use RGB-D frames to calculate object shape and silhouette information and they estimate the pose using an Unscented Kalman Filter [147].
- Though, in most cases, due to the non-Gaussian formulation of the noise the Kalman Filter is substituted with a Particle Filter. In particular, in [34] Hadfield et al. used a colour detection-based-tracker to produce crude 2D position estimates on the image plane, and then he employed a Rao-Blackwellized Particle Filter to refine the estimates and infer the 6DoF poses. The tracking information was then used to detect which connections have been established and to determine the state of an assembly procedure (see fig.4.27).

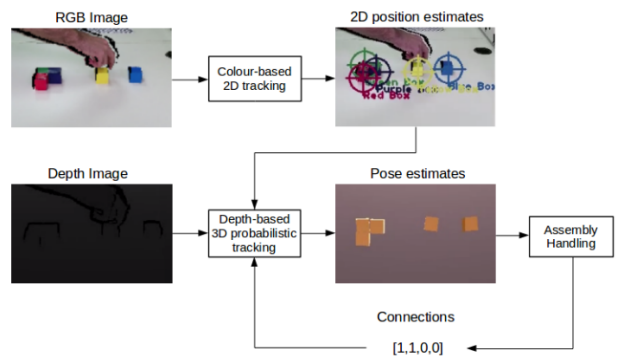
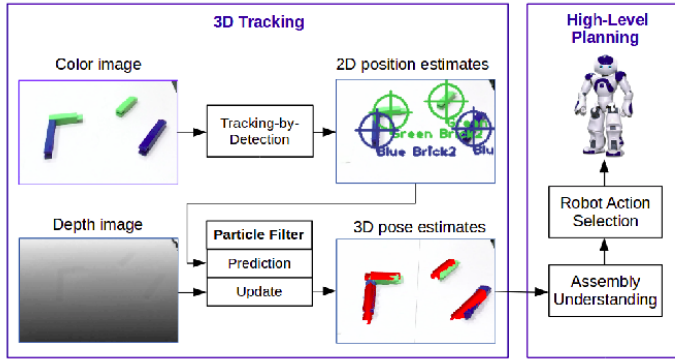


Figure 4.27: Overview of the multi-part assembly framework of Hadfield et al. [34].

- In [21], Deng et al. extended the approach of [126] by combining it with a Rao-Blackwellized Particle Filter as Hadfield et al. did in [34]. In particular, instead of sampling the whole 6D pose space with particles, they sampled an educated subset of the State Space and then solved analytically for the desired 6D pose, conditioned on the sampled subset. In a nutshell, they succeeded so by randomly sampling bounding boxes on the RGB 2D images to infer 3D translation possibilities (from the 2 spatial dimensions of each bounding box and its scale), create crops of each image based on the bounding box and finally searching for the closest saved rotated sample of [126]. Finally, the particles of the filter were weighted according to the orientation probabilities provided by the encoder of [126] and were prepared for the next sampling iteration.

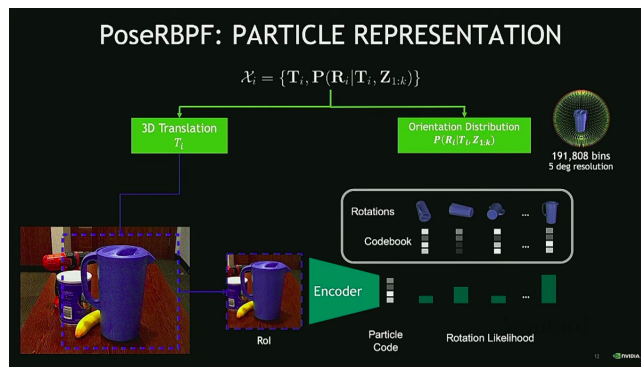


Figure 4.28: Overview of the total architectural design of the mixed Deep Particle Filtering approach of [21].

Online Energy Minimization

Online attempts have formulated the Pose Tracking problem as an Energy minimization one where the energy is depended on the distance between the predicted and the ground truth pose at any given moment and the goal is to find the predicted pose that will minimize it, as the scene evolves in real

time.

- In [118], tracking the object’s pose is based on the Newton’s 2nd law. Energy minimization includes two terms: one for measuring the difference between the predicted and the real image and a second one that calculates the total force applied on all objects.
- In [70], Kyriazis et al. a two-term energy function was employed as well. In this case, there was a predicted Depth image error metric along with a punishment term for intersections of the various objects present in the scene.

Optical Flow-based

With the name **Optical Flow** we characterize the visible 2D motion distribution between two images. It can be separated into sparse, where it is calculated only between specific points of interest that sufficiently describe the object movement, and dense optical flow, where it is calculated for all image pixels.

Main advantages:

- Algorithms that use optical flow exclusively or as a first stage of estimating the object’s pose are provided with an exact measurement of its motion through time (even projected on the 2D image plane), which may lead to higher accuracy (at least for the translation component).

Main disadvantages:

- Most high quality optical flow estimation algorithms are too slow for real-time tracking capabilities.
- This information source neglects to handle occlusions present in the scene or differences in its configuration described by its appearance.
- In [100], Pauwels combined the two latter approaches in a single object, real-time framework by constructing a multi-task energy function and optimizing it iteratively. In particular, he projected the approximated 3D motion equation onto the 2D pixel-image plane and he formulated the energy function as a sum of the mean squared point-to-plane distance between the current observed frame and the previously predicted one and the Optical and Augmented Reality Flow (AF) Mean Squared Errors.

Deep Learning-based

They are consisted of 2 Convolutional streams (one based on the Observed, by the sensor, frame and the other based on the ‘Predicted’ frame being rendered using the previous pose prediction.) They are typically compared in some way and their Difference Rigid Transformation is predicted as ΔT .

- In [54], Tan et al. developed decision trees that regress a subset of 3D Points in the Object Coordinate System only using the Depth maps. It robustly handles occlusions for the translation, but not for the rotation case, and rarely fails irrecoverably.
- In their later attempt, in [128], they tried to combine offline Machine Learning of the various viewpoints with the use of Random Forests that regress the Transformation parameters between the two frames in their leaves, employing a separate tree for each parameter-viewpoint combination and online Energy-based minimization trackers. The latter optimize a ICP-like 3D Point-2D pixel projection energy function. As for the the approach’s pros, it extracts global reasoning of the object pose based on the neighbouring leaves and is robust to abrupt movements, while the use of Random Forest inserts the hazard of overfitting to training samples.
- In [29] and [83] Garon et al. formulated the tracking problem exclusively as a learning one by generating synthetic RGB-D frame pairs from independent viewpoints, process them using a CNN stream for each and compare them by concatenating those streams. In a later section, we will further discuss the details and weaknesses of their work, as it is the main reference for our approach as we try to develop specific modifications that take explicit care of challenges they left untackled.

- In DeepIM [73], they initialised a similar CNN with the weights taken from a FlowNet [52] variation and

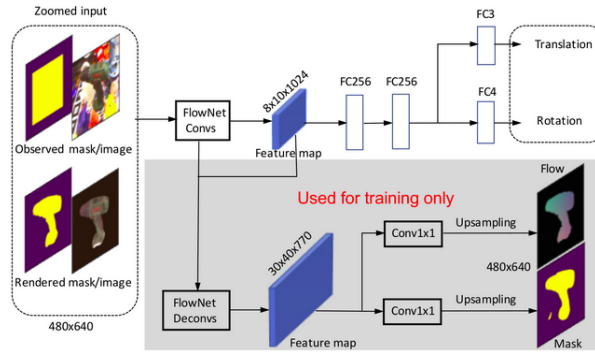


Figure 4.29: Overview of the DeepIM architecture of [73].

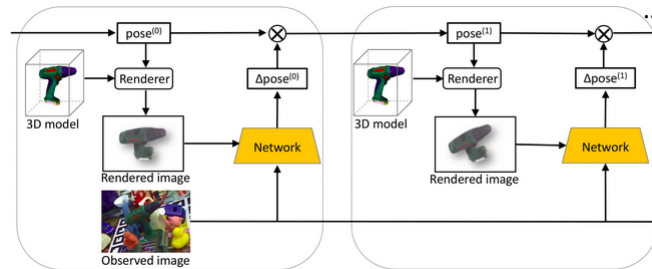


Figure 4.30: Details about the iterative training of every image-pose pair of the DeepIM architecture[73].

- In DeepTAM [169], they simultaneously handled the camera/object tracking problem along with the SLAM one. As for the tracking case, they differ from the above examples in the sense that they also use, only in training, an Optical-flow based term for regularization and that they encourage the production of multiple and as different as possible pose hypotheses that are bootstrapped in the final layer to get the final one.

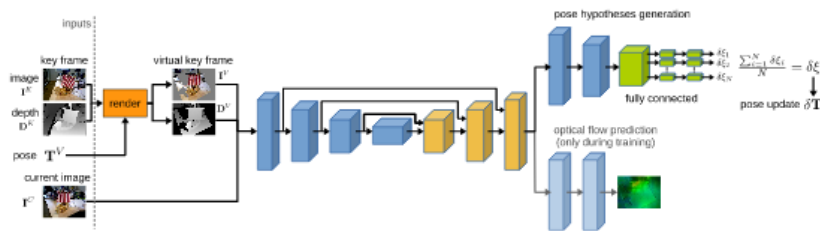


Figure 4.31: The DeepTAM [169] tracking network uses an Encoder-Decoder type architecture with direct connections between the encoding and decoding part. The decoder is used by two tasks, which are Optical Flow prediction and the generation of pose hypotheses. The Optical Flow prediction is a small stack of two convolution layers and is only active during training to stimulate the generation of motion features. The pose hypotheses generation part is a stack of downsampling convolution layers followed by a fully connected layer, which then medskips into $N=64$ fully connected branches sharing parameters to estimate the $\delta\xi_i$. Along with the current camera image I_C we provide a virtual keyframe (I_V, D_V) as input for the network, which is rendered using the active keyframe (I_K, D_K) and the current pose estimate T^V . [169]

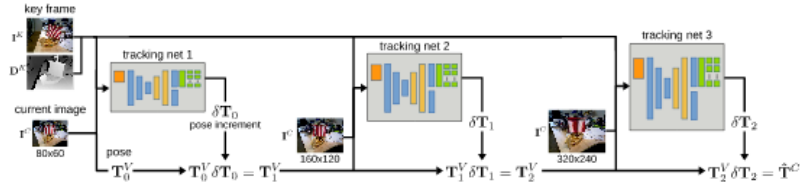


Figure 4.32: Overview of the tracking networks and the incremental pose estimation. We apply a coarse-to-fine approach to efficiently estimate the current camera pose. We train three tracking networks each specialized for a distinct resolution level corresponding to the input image dimensions (80×60), (160×120) and (320×240). Each network computes a pose estimate δT_i with respect to a guess T_i^V . The guess T_0^V is the camera pose from the previously tracked frame. Each of the tracking networks uses the latest pose guess to generate a virtual keyframe and thereby indirectly tracking the camera with respect to the original keyframe (I_K, D_K) . The final pose estimate T_C is computed as the product of all incremental pose updates δT_i . [169]

Chapter 5

Data Generation

It was the best of times, it was the worst of times, it was the age of wisdom, it was the age of foolishness, it was the epoch of belief, it was the epoch of incredulity, it was the season of Light, it was the season of Darkness, it was the spring of hope, it was the winter of despair, we had everything before us, we had nothing before us, we were all going direct to Heaven, we were all going direct the other way.

Charles Dickens, "A tale of two cities"

5.1 3D Graphics Rendering

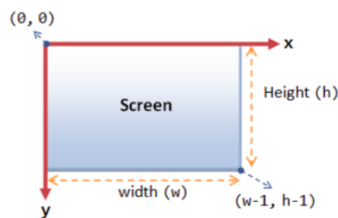


Figure 5.1: **The 2D Screen Coordinates:** The origin is located at the top-left corner, with x-axis pointing left and y-axis pointing down, in contrary to the popular Cartesian Coordinated System. [18]

All image samples used to train our neural architecture are synthetically constructed via a rendering process. **Rendering** means producing the image shown on the Display from a CAD 3D object model source. The rendering pipeline accepts a description of 3D objects in terms of vertices of primitives (e.g. triangle meshes) and outputs the color-value of the pixels on the display. Displays are raster-based: they are consisted of a 2D rectangular pixel grid. Color values are saved in the frame buffer, located in the Graphic Memory. From there, they are accessed from left to right and from top to bottom (“raster scan”) and put on the display, which is refreshed at a specific rate. In more details, in order to avoid the problem of “tearing”, which occurs when the frame buffer values are renewed at the same time they are shown on the screen, 2 identical buffers are employed (one in the front, whose values are displayed, and one in the back, ready to be updated at any time) and they are swapped asynchronously with the arrival of a suitable synchronization signal.

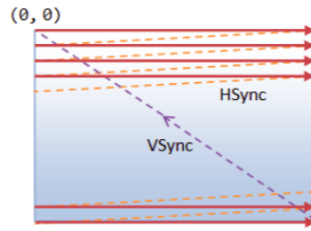


Figure 5.2: **Raster Scan:** The display updates its contents from line to line top-to-bottom and left-to-right by accessing the pixel color values saved at the frame buffer which is in front at every given moment. [18]

The 3D graphics rendering pipeline executes the conversion of the 3D primitives into pixel values and saves them in the frame buffer via a hierarchy of processing stages.

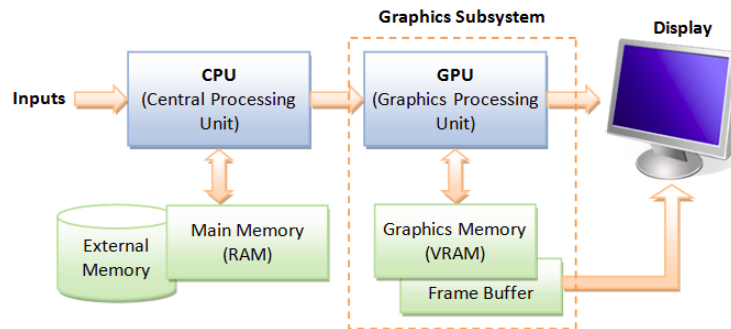


Figure 5.3: **3D Graphics Rendering Pipeline:** The output of one stage is fed into the next. A vertex has attribute (x,y,z) position, (R,G,B) or (R,G,B,A) color, (n_x, n_y, n_z) vertex normal and texture. A primitive is made up of one or more vertices. The rasterizer raster-scans each primitive to produce a set of grid-aligned fragments, by interpolating the vertices.[18]

It is comprised of the following main stages:

- **Vertex Processing:** Process and transform individual vertices.
- **Rasterization:** Convert each primitive (connected vertices) into a set of fragments. A fragment can be treated as a **candidate pixel** in 3D, which is aligned with the pixel grid, with attributes such as position, color, vertex-normals and texture.
- **Fragment Processing:** Process individual fragments.
- **Output Merging:** Combine the fragments of all primitives (in 3D space) into 2D color-pixel for the display.

A primitive is made up of one or more vertices. A vertex, in computer graphics, has these attributes:

- **Position in 3D space $V=(x, y, z)$,** normalized between $[-1.0,1.0]$. They are connected with the use of edges into more complex shapes such as triangles (for a triangle mesh representation).
- **Color:** expressed in RGB (Red-Green-Blue) or RGBA(Red-Green-Blue-Alpha) components. The component values are typically normalized to the range of 0.0 and 1.0 (or 8-bit unsigned integer between 0 and 255). Alpha $\in [-1.0,1.0]$ is used to specify the transparency level, with alpha of 0 for totally transparent and alpha of 1 for opaque.
- **Vertex-Normal $N = (n_x, n_y, n_z)$:** Like surface normal vectors, which are perpendicular to the surface, in computer graphics, there is the need to attach a normal vector to each vertex, known as vertex-normal. They are used to differentiate the front- and back-face, and lighting calculations. Each vertex normal is pointing outwards, indicating the outer surface (or front-face).
- **Texture $T=(u, v)$:** In order to make the object's appearance more realistic, we project each 3D object vertex to the corresponding 2D texture image coordinates (u, v) and we use them to wrap the object with.

Pixels refer to the positions on the 2D grid of the display, with each one of them assigned to an RGB color value (there is no alpha value for pixels).

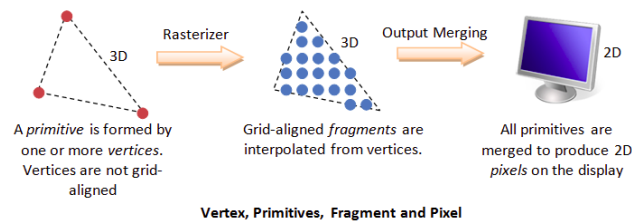


Figure 5.4: The hierarchy of transformation from 3D sparse vertices to 2D pixels [18]

However, what we have so far are 3D object vertices which may be sparse or dense in terms of space, while, we need to know the value of every pixel in order to print it on the screen. So, the intermediate values between the 2D projections of vertices are required, as well. The rasterizer of the graphics rendering pipeline produces the grid-aligned pixel values by taking each input primitive and perform raster-scan to produce a set of grid-aligned fragments enclosed within the primitive.

A fragment is 3-dimensional, with a (x, y, z) position and has the same attributes as a vertex. The (x, y) positions are aligned with the 2D pixel-grid. The z -value (not grid-aligned) denotes the relative depth of the various primitives and it is used to discard the parts of the scene components that are occluded by other elements.

In modern GPU, vertex processing and fragment processing are programmable. The programs are called vertex shader and fragment shader.

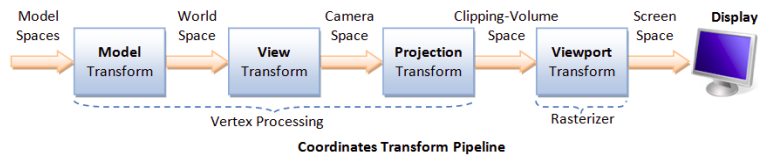


Figure 5.5: The sequence of Homogenous Transformations that convert the initial 3D vertex representation w.r.t. the Object Coordinate Frame to a pixel grid on the Screen Space.[18]

Let's examine every element of the pipeline closer:

The process of taking a 2D photograph of a 3D scene involves four transformations:

- At first, we need to place the various objects on the scene. The mathematic equivalent is the transformation of the vertices coordinates from the object-centric Coordinate frame to the World Coordinate Frame (Model Transformation).
- Secondly, we must place the camera at the proper position and orientation i.e transform the Camera Coordinate frame w.r.t. the World Coordinate frame (View transformation).
- Our third action is to select a camera lens (wide angle, normal or telescopic), adjust the focus length and zoom factor to set the camera's field of view. Mathematically, this means establishing the Projective Transformation that aligns our 3D scene elements on the 3d image plane. (Projection transformation).
- Print the photo on a selected area of the paper (Viewport transformation) - in rasterization stage.

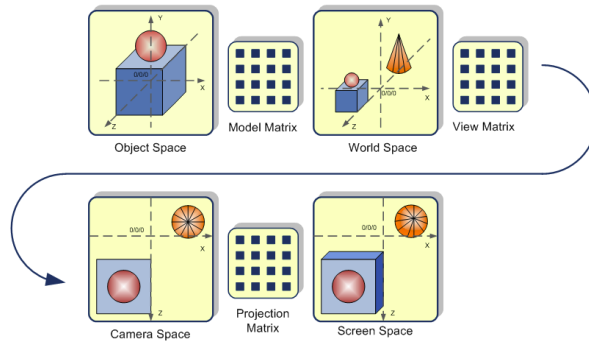


Figure 5.6: The sequence of the 4 Transformations

Let's delve into the details of every of the aforementioned transformations:

- Each object (or model or avatar) in a 3D scene is typically drawn in its own coordinate system, known as its model space (or local space, or object space). As we assemble the objects, we need to transform the vertices from their local spaces to the world space, which is common to all the objects. This is known as the world transform. The world transform consists of a series of scaling (scale the object to match the dimensions of the world), rotation (align the axes), and translation (move the origin).

Rotation and scaling belong to a class of transformation called linear transformation (by definition, a linear transformation preserves vector addition and scalar multiplication). Linear transform and translation form the so-called affine transformation. Under an affine transformation, a straight line remains a straight line and ratios of distances between points are preserved.

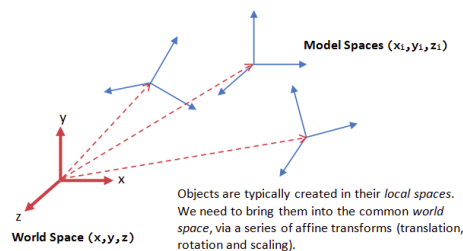


Figure 5.7: The Transformation process of one or more object(s) from their corresponding Coordinate Frame to the global World one. [18]

After the world transform, all the objects are assembled into the world space. We shall now place the camera to capture the view.

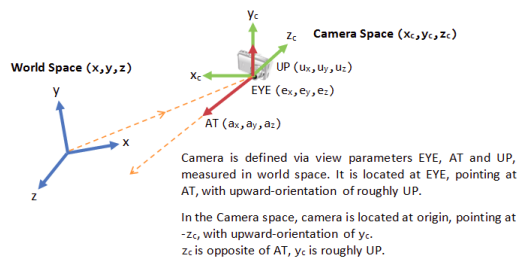


Figure 5.8: The camera is placed at the position described by the Eye vector (w.r.t. (W)) and snaps a picture of the objects in the scene at the direction given by the LookAT transformation process. [18]

- In 3D graphics, we position the camera onto the world space by specifying three view parameters: EYE, AT and UP, in world space.
 - The point EYE $\mathbf{E} = (e_x, e_y, e_z)$ defines the location of the camera.

- The vector AT $\mathbf{A} = (a_x, a_y, a_z)$ denotes the direction where the camera is aiming at, usually at the center of the world or an object.
- The vector UP $\mathbf{U} = (u_x, u_y, u_z)$ denotes the upward orientation of the camera roughly. UP is typically coincided with the y-axis of the world space. UP is roughly orthogonal to AT, but not necessary. As UP and AT define a plane, we can construct an orthogonal vector to AT in the camera space.

Notice that the 9 values actually produce 6 degrees of freedom to position and orientate the camera, i.e., 3 of them are not independent.

From EYE, AT and UP, we first form the coordinate (x_c, y_c, z_c) for the camera, relative to the world space. We fix z_c to be the opposite of AT, i.e., AT is pointing at the $-z_c$. We can obtain the direction of x_c by taking the cross-product of AT and UP. Finally, we get the direction of y_c by taking the cross-product of x_c and z_c . Take note that UP is roughly, but not necessarily, orthogonal to AT.

$$\mathbf{z}_c = \frac{\mathbf{E} - \mathbf{A}}{\|\mathbf{E} - \mathbf{A}\|_2} \quad (5.1)$$

$$\mathbf{x}_c = \frac{\mathbf{U} \times \mathbf{z}_c}{\|\mathbf{U} \times \mathbf{z}_c\|_2} \quad (5.2)$$

$$\mathbf{y}_c = \mathbf{z}_c \times \mathbf{x}_c \quad (5.3)$$

- Now, we transform the object coordinates from the World to the Camera Space, centered at EYE and characterized by the basis $(\mathbf{x}_c, \mathbf{y}_c, \mathbf{z}_c)$. To do so, we perform a Translation from $\mathbf{0}$ to the EYE and then a Rotation from $(\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3)$ to $(\mathbf{x}_c, \mathbf{y}_c, \mathbf{z}_c)$, i.e. we apply the **View Transform**:

$$M_{View} = \begin{bmatrix} x_c^1 & x_c^2 & x_c^3 & -e_x \\ y_c^1 & y_c^2 & y_c^3 & -e_y \\ z_c^1 & z_c^2 & z_c^3 & -e_z \end{bmatrix} \quad (5.4)$$

Once the camera is positioned and oriented, we need to decide what it can see (analogous to choosing the camera's field of view by adjusting the focus length and zoom factor), and how the objects are projected onto the screen. This is equivalent the selecting a **projection mode** (either **Perspective** or **Orthographic**) and specifying a suitable **viewing** or **clipping volume**. Object coordinates inside this volume will be projected on to the 2D image plane and the rest will be rejected out of the scene.

The two Projection types are divided according to:

- **Perspective Projection**: In this first type, we aim to present points that are closer to the Center of Projection (i.e. the intersection of the camera principal axis and the frustum) appear larger than objects further to the it that have the same size. To do so using the limited field of view of the camera, a truncated pyramid called **View Frustum** is established using four parameters: *fovy*, *aspect*, z_{near} and z_{far} .
 - * Fovy: specifies the total vertical angle of view deg.
 - * Aspect = $\frac{W}{H}$.
 - * z_{near} : the cut-off plane, perpendicular to the camera principal axis that is the nearest to the camera.
 - * z_{far} : the cut-off plane, perpendicular to the camera principal axis that is the furthest from the camera.

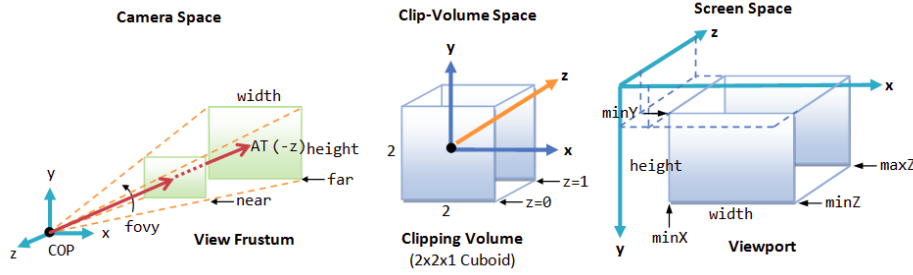


Figure 5.9: The View Frustum of the Perspective Projection, specified by: fovy, aspect ratio, z_{near} and z_{far} , its Clip Volume Space and the final Screen Space, where the object points are transformed.

If an object is outside this frustum it is clipped out of the final image entirely, while if it intersects it, the overlapping part is contained in it.

We apply the Perspective Projection Matrix to transform the view-frustum into a axis-aligned cuboid clipping-volume of $2 \times 2 \times 1$ centered on the near plane. The near plane has $z_{near}=0$, whereas the far plane has $z_{far} = -1$.

The Perspective Projection Matrix is:

$$M_{Perspective} = \begin{bmatrix} \frac{\cot(\frac{fovy}{2})}{AR} & 0 & 0 & 0 \\ 0 & \cot(\frac{fovy}{2}) & 0 & 0 \\ 0 & 0 & -\frac{z_{far}}{z_{far}-z_{near}} & -\frac{z_{near} \times z_{far}}{z_{far}-z_{near}} \\ 0 & 0 & -1 & 0 \end{bmatrix} \quad (5.5)$$

Note that:

- * Its last column is not that of a classical Homogenous Matrix: $[0, 0, 0, 1]^T$ to normalize the

enclosed Homogenous Object point's w-component and ensure that it is written as $\begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} = 1$.

- * The third row is negated (transforming the Right-Hand-Side coordinate system to a Left-Hand-Side one) to flip the z_{far} from -1 to 1 and ensure that the z-dimension of our frustum $\in [0, 1]$.

– **Orthographic Projection:** is a special case where the camera is placed very far away from the world. Its view volume is a parallelepiped, which means that vertices far away from the camera are presented in the same scale as those close to it.

The Orthographic Projection Matrix:

$$M_{Ortho} = \begin{bmatrix} \frac{2}{right-left} & 0 & 0 & -\frac{right+left}{right-left} \\ 0 & \frac{2}{top-bottom} & 0 & -\frac{top+bottom}{top-bottom} \\ 0 & 0 & -\frac{2}{z_{far}-z_{near}} & -\frac{z_{far}+z_{near}}{z_{far}-z_{near}} \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (5.6)$$

The rest of the Projection process remains the same as above. This is the type of projection that we utilized here.

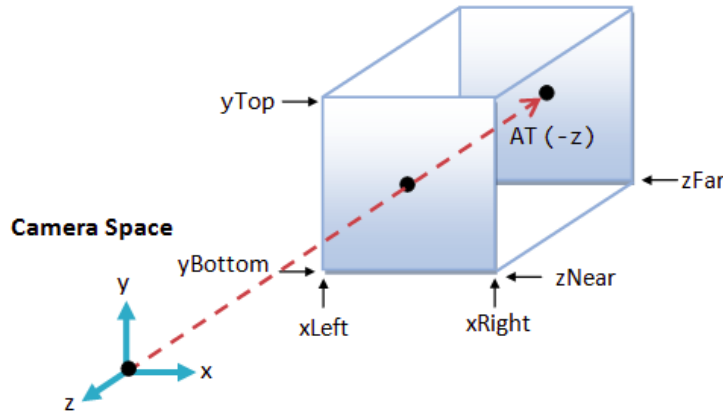


Figure 5.10: **Orthographic projection:** Camera positioned infinitely far away at $z=\infty$ and the object analogies are conserved.

Each vertex is transformed and positioned in the clipping-volume cuboid space, together with their vertex-normal. The x and y coordinates (in the range of -1 to $+1$) represent its position on the screen, and the z value (in the range of 0 to 1) represents its depth, i.e., how far away from the near plane.

- In the rasterization stage, each primitive is raster-scanned to obtain a set of fragments enclosed within it. Fragments can be treated as 3D pixels, which are aligned with the pixel-grid. They are interpolated from the vertices and have the same set of attributes as them.

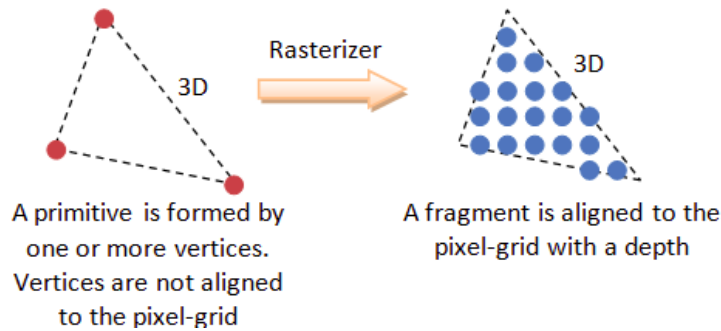


Figure 5.11: The Vertex Rasterization process [18]

The substages of rasterization include:

- *Viewport Transform:* A **Viewport** is a rectangular display area measured in pixels, with origin at the top-left corner. It defines the size and shape of the display area to map the projected scene captured by the camera onto the application window.

The viewport transform:

$$M_{\text{Viewport}} = \begin{bmatrix} \frac{w}{2} & 0 & 0 & \min X + \frac{w}{2} \\ 0 & -\frac{h}{2} & 0 & \min Y + \frac{h}{2} \\ 0 & 0 & \max Z - \min Z & \min Z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (5.7)$$

maps the clipping-volume ($2 \times 2 \times 1$ cuboid) to the 3D viewport.

- *Clipping:* All pixels outside of the clipping volume are cut off.
- *Perspective division:* The projection matrix sets things up so that after multiplying with the it, each coordinate's W will increase the further away the object is. So, we divide every one of the X, Y, Z components by W .
- *Back-face Culling:* It discards primitives which are not facing the camera based on the corresponding normal vector and the vector connecting the surface and the camera.

- After rasterization, each primitive corresponds to a fragment with depth, color, normal and texture coordinates that are interpolated from its vertices.

The fragment processing is consisted of:

1. *Texturing*:

Texturing



UV Image



A texture is typically a 2D image with coordinates (u, v) is typically normalized to $[0,1]$. In general, the resolution of the texture image is different from the displayed fragment (or pixel). If the resolution of the texture image is smaller, we need to perform so-called magnification to magnify the texture image to match the display. On the other hand, if the resolution of texture image is larger, we perform minification.

Magnification: The commonly used methods are:

- (a) *Nearest Point Filtering*: the texture color-value of the fragment is taken from the nearest texel. This filter leads to “blockiness” as many fragments are using the same texel.
- (b) *Bilinear Interpolation*: the texture color-value of the fragment is formed via bilinear interpolation of the four nearest texels. This yields smoother result.

Minification: Minification is needed if the resolution of the texture image is larger than the fragment. Again, you can use the “nearest-point sampling” or “bilinear interpolation” methods. However, these sampling methods often to the so-called “aliasing artefact”, due the low sampling frequency compared with the signal. For example, a far-away object in perspective projection will look strange due to its high signal frequency.

Minmapping: A better approach for performing minification is called minmapping (miniature maps), where lower resolutions of the texture image are created. For example, suppose the original image is 64×64 (Level 0), we can create lower resolution images at 32×32 , 16×16 , 8×8 , 4×4 , 2×2 , 1×1 . The highest resolution is referred to as level 0; the next is level 1; and so on. We can then use the nearest matched-resolution texture image; or perform linear interpolation between the two nearest matched-resolution texture images.

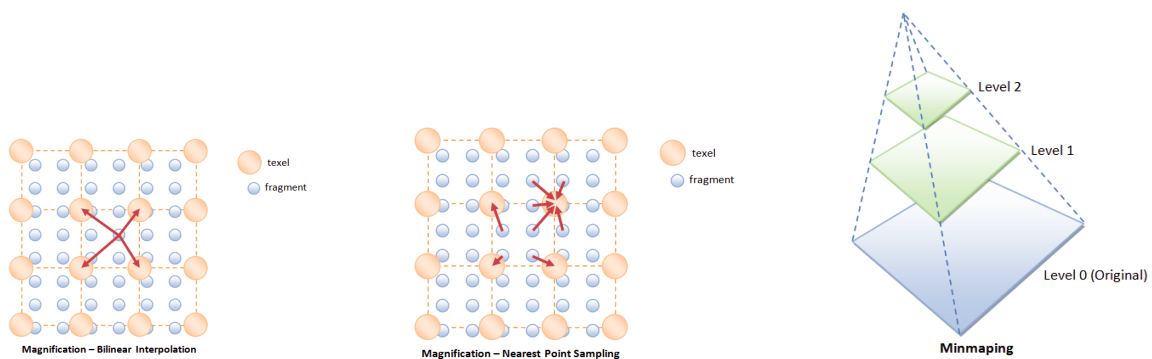


Figure 5.12: Graphical representation of (from left to right): Magnification with Bilinear Interpolation, Magnification with Nearest Point Sampling and Minmapping. [18]

2. *Depth buffer test*: The depth-buffer is used to remove hidden surfaces (surfaces blocked by other surfaces and cannot be seen from the camera). It is initialized to 1 (farthest) and the color-buffer initialized to the background color. Next, we follow the Painter's algorithm: for each fragment processed, if its z-value is smaller than the depth-buffer, its color and z-value are copied into the buffer. Otherwise, this fragment is discarded. At the end, we output the depth buffer as a synthetic Depth image, if we first normalize it between the minimal and maximal possible distance to the camera.
3. *Lighting*:

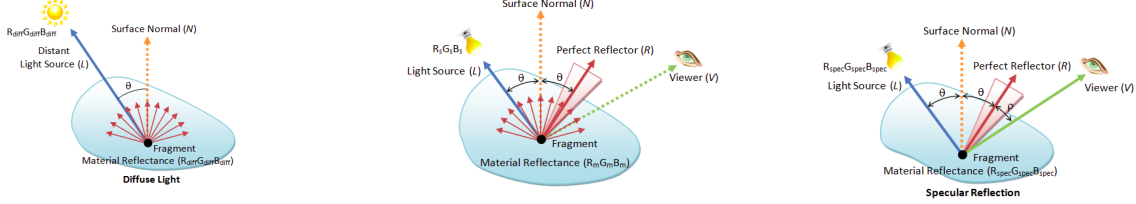


Figure 5.13: The most prominent types of lighting in 3D Graphics(from left to right): Combined, Diffuse (only) and Specular (only) lighting. [18]

Lighting refers to the handling of interactions between the light sources and the objects in the 3D scene. It plays a significant role in producing a realistic scene. Consider a fragment P on a surface, four vectors are used: the light source \mathbf{L} , the viewer \mathbf{v} , the fragment-normal \mathbf{N} , and the perfect reflector \mathbf{R} . The perfect reflector \mathbf{R} can be computed from the surface normal \mathbf{N} and the incidence light \mathbf{v} , according to Newton's law which states that the angle of incidence is equals to the angle of reflection.

- **Diffuse light**: models distant directional light source (such as the sun light). The reflected light is scattered equally in all directions. The strength of the incident light is $[\mathbf{L} \cdot \mathbf{N}]_+$. We use the max function to discard the negative number, i.e., the angle is more than 90° . Suppose the light source has color \mathbf{s}_{Diff} , and the fragment has diffusion reflectance of m_{Diff} , the resultant color \mathbf{c}_{Diff} is:

$$c_{Diff}^{(i)} = [\mathbf{LN}]_+ \times s_{Diff}^{(i)} \times m_{Diff}, \quad (5.8)$$

for $i = \{R, G, B\}$

- **Specular Lighting**: The reflected light is concentrated along the direction of perfect reflector vector \mathbf{r} . What a viewer sees depends on the angle between \mathbf{v}, \mathbf{r} . The resultant color due to specular reflection is given by:

$$c_{Spec}^{(i)} = [\mathbf{rv}]_+^{sh} \times s_{Spec}^{(i)} \times m_{Spec}, \quad (5.9)$$

for $i = \{R, G, B\}$, with sh being the shininess factor.

- **Ambient Light** is a constant amount of light applied to every point of the scene. The resultant color is:

$$c_{Amb}^{(i)} = S_{Amb}^{(i)} \cdot m_{Amb}, \quad (5.10)$$

for $i = \{R, G, B\}$

- **Emissive Lighting**: Some surfaces may emit light. The resulting color in this case is

$$c_{em}^{(i)} = m_{em}, \quad (5.11)$$

$i = R, G, B$. However, we handle no such objects in our work, so this lighting component is emitted in our analysis.

The resulting colour is the sum of all the light sources' contribution:

$$\mathbf{c}_{Final} = \mathbf{c}_{Diff} + \mathbf{c}_{Amb} + \mathbf{c}_{Spec}. \quad (5.12)$$

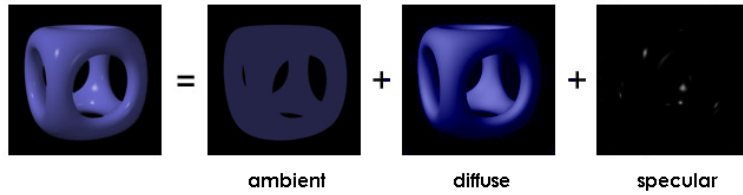


Figure 5.14: Decomposition of lighting interactions. [18]

5.2 Training Data Nature Tradeoff

Lastly, we need to discuss the use of a synthetic training dataset. It is logical to assume that datasets that contain real images and cover sufficiently the pose space are the go-to option as they offer quick convergence, realistic appearance and illumination conditions. However, acquiring such a dataset requires the time-consuming and painstaking procedure of 6D pose annotation, which cannot be taken for granted. Moreover, those kind of datasets tend to bias the tracker towards particular scales, motion and occlusion patterns harming its generalization ability in unseen environments. For all the above reasons, although a real dataset for training is provided in [83], we opt to produce synthetic training data. As a result, we are more confident about the generalization performance of our tracker than if we had trained it on the one that [29] provides and tested it on the other. Nevertheless, we recognize the domain gap issue that the use of synthetic images generates and we take particular care of it in our architecture design.



Figure 5.15: A 3D CAD Dragon model of [83] at a specific pose.

5.3 Data Generation Process

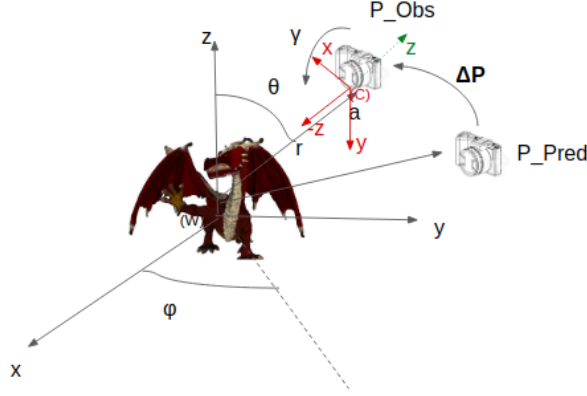


Figure 5.16: An overview of the synthetic generation method presented in [29].

We wish to create a training dataset of 20,000 RGB-D image pairs: I_t, \hat{I}_t . We follow the framework presented in [29],[83] to synthetically generate our training dataset by rendering each target object in pose pairs. When we say “rendering” from now on, we mean the whole pipeline presented above in this Section.

Since we are handling the single object case in this work, there is no need to place the object in an arbitrary 3D pose w.r.t the steady (W) coordinate origin and the camera pose in a second, independent, 3D pose (C) and then render the scene. On the contrary, the object is placed at the origin of the world reference frame. In [29], Garon et al. sampled the random camera pose in spherical coordinates (θ, ϕ) :

- $$\theta \sim U(-180^\circ, 180^\circ) \quad (5.13)$$

- $$\phi = \arccos(2x - 1), \quad (5.14)$$

where $x \sim U[0,1]$ ($U(a,b)$ indicates a uniform distribution in the $[a,b]$ interval).

A random roll angle was sampled uniformly as $\gamma \sim U(-180^\circ, 180^\circ)$ and the camera is displaced along the ray between the camera pose and the origin by a random amount $r \sim U(0.4m, 1.5m)$ to obtain the Observed Pose P_{Obs} . The Predicted camera Pose P_{Pred} was obtained by applying a random, 6-DOF rigid transformation starting from P_{Obs} . This transformation was acquired by sampling a random translation $t_{x,y,z} \sim \mathcal{N}(0, \Delta t)$ and rotation $r_{a,b,c} \sim (0, \Delta r)$ i.e. in Axis-angle notation and then by converting the rotation components to their corresponding Euler Angles. The inverse of this random transformation was applied to P_{Obs} to obtain P_{Pred} .

$$\Delta P^{(c)} = T^{-1} = P_{Pred} P_{Obs}^{-1} \quad (5.15)$$

Thus, the target label (displacement between the two poses) is the 6-vector $\delta \mathbf{p} = \begin{bmatrix} t_x \\ t_y \\ t_z \\ r_a \\ r_b \\ r_c \end{bmatrix}$ of con-

catenated translations and Euler angles representing the object transformation in the camera reference frame.

Let’s briefly discuss why Garon et al. [[29],[83]] have made this sampling choice. The first approach

that would immediately come to one's mind would be to sample the spherical coordinates θ, ϕ via 2 corresponding uniform distributions:

$$\begin{aligned}\theta &\sim U(-180^\circ, 180^\circ) \\ \phi &\sim U(-90^\circ, 90^\circ).\end{aligned}\tag{5.16}$$

However, this sampling results in an uneven distribution, with the density increasing as we get closer to the poles. We may visualize this by dividing the sphere surface into smaller areas of given ‘‘squares’’ of width $\Delta\theta$ and height $\Delta\phi$. We can clearly see that each such area varies with ϕ . Indeed, the ‘‘squares’’ become smaller and smaller as we approach the poles (see [108]). Therefore, Garon et al. have chosen the aforementioned strategy. However, we experimentally find as more beneficial to alter their sampling strategy. as follows:

•

$$\phi = \arccos(2x - 1),\tag{5.17}$$

but with x being deterministically, (instead of randomly) sampled:

$$x = \text{linspace}[0, 1, 20000] + 0.5\tag{5.18}$$

•

$$\theta = \pi(1 + \sqrt{5})x\tag{5.19}$$

This angle parameterization we use is called the **Golden Spiral approach** and it is based on the observation that uniform random sampling, used by Garon et al. in [[29],[83]], tends to create clusters compromising the requirement for even point distribution along the 3D sphere surface (see[108]). In a nutshell, we wish to solve the problem of uniform sampling 3D points on a sphere, which is far from trivial. The Golden Spiral approach samples 3D points in a 3D sphere as uniformly as possible in a deterministic way. Next, we present its thought process by starting from the 2D planar example (of the 2D disk) and extending to its 3D counterpart (the sphere):

The 2D case:

First, we give the method intuition by describing the 2D example. We start by taking the most irrational number possible: the golden ratio $\varphi = \frac{(1+\sqrt{5})}{2}$. If one follows the next algorithm to emit points:

Stand at the center, turn a golden ratio of whole turns, then emit another point in that direction

one naturally constructs a 2D spiral which, as you get to higher and higher numbers of points, nevertheless refuses to have well-defined ‘‘bars’’ that the points line up on.

As for the radial sampling, we want these to have even-area spacing around the 2D disk. That's the same as saying that in the limit of large number of points (N_{points}), we want a little region $R \in (r, r + dr)$, $\theta \in (\theta, \theta + d\theta)$ **to contain a number of points proportional to its area:**

$$dA = r dr d\theta.\tag{5.20}$$

Now, let's, temporarily, pretend that we are talking about random variables. This is the same as saying that the joint probability density for (R, Θ) is:

$$\mathbb{P}_{R,\Theta} = cr\tag{5.21}$$

for some constant c.

Normalization on the unit disk, then, forces:

$$\int_0^{2\pi} \int_0^1 \mathbb{P}_{R,\Theta} dr d\theta = 1 \iff \int_0^1 cr dr \int_0^{2\pi} d\theta = 1 \iff \frac{1}{2}r^2 \Big|_0^1 2c\pi = 1 \iff c\pi = 1 \iff c = \frac{1}{\pi}.\tag{5.22}$$

Now, we use a trick from the probability theory, known as **sampling the inverse Cumulative Distribution Function (CDF)**:

Suppose we wish to generate a random variable Z with a probability density $f(z)$, but all we have is a random variable $V \sim U[0, 1]$:

- We turn the density $f(z)$ into a CDF:

$$F(z) = \int_0^z f(z')dz, \tag{5.23}$$

which increases monotonically from 0 to 1.

- We calculate the CDF's inverse function: $F^{-1}(z)$.
- Then, $Z = F^{-1}(V)$ is distributed according to the target density $f(z)$.

Let's apply the algorithm above. The golden-ratio spiral trick spaces the points out in a nicely even pattern for θ so the integral of $f(z)$ over the unit circle leaves us with:

$$F(r) = \int_0^{2\pi} \int_0^r f(r, \theta)drd\theta = \int_0^r \frac{r'}{\pi}dr' = \frac{r^2}{2\pi}2\pi = r^2 \tag{5.24}$$

So, the inverse function is:

$$F^{-1}(v) = \sqrt{v} \tag{5.25}$$

Therefore, we could generate random points on the sphere in polar coordinates with

$$\begin{aligned} r &= \sqrt{x} \\ \theta &= 2\pi x, \end{aligned} \tag{5.26}$$

with $x \sim U[0, 1]$.

However, instead of randomly sampling this inverse function, we opt to sample it uniformly in a deterministic way. This has the advantage that points are spread out and in the limit of large N_{points} , the method will behave as if we had randomly sampled them. This combination is called the ‘Golden spiral’ trick. In a nutshell, we uniformly sample r to get equal-area spacing, and we use the spiral increment to avoid awful ‘bars’ of points in the output. Overall, mathematically,

$$\begin{aligned} indices &= \text{linspace}(0, N_{points}) + 0.5 \\ r &= \sqrt{\frac{indices}{N_{points}}} \\ \theta &= \pi(1 + \sqrt{5}) \cdot indices \end{aligned} \tag{5.27}$$

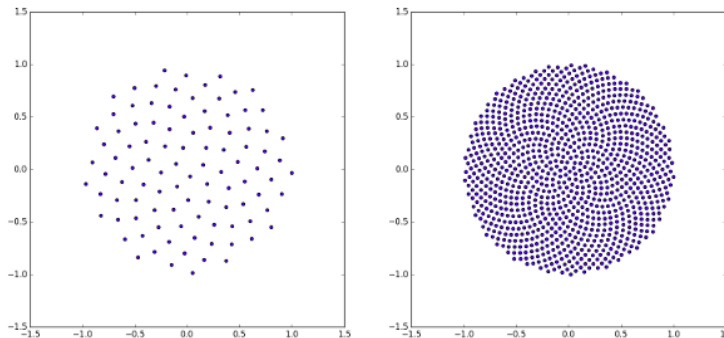


Figure 5.17: The results of the 2D uniform deterministic distribution for 100 and 1000 points, correspondingly.

The 3D case: Uniformly sampling 3D points on a sphere

We, basically, switch from polar to spherical coordinates, basically replacing the variable r with ϕ . We notate the spherical latitude as $\phi \in [0, \pi]$ and the spherical longitude as $\theta \in [0, 2\pi]$.

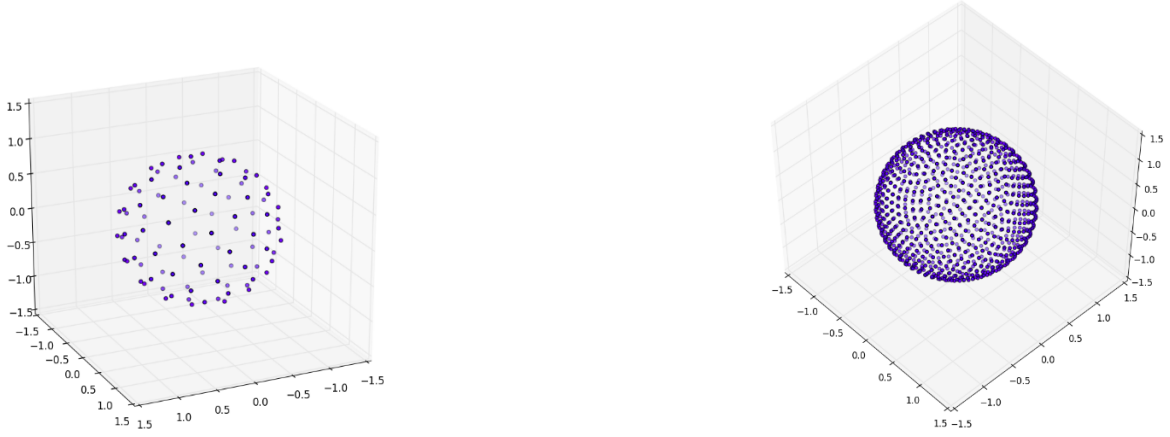


Figure 5.18: The proposed 3D sphere sampling approach for $N_{points} = 100$ and $N_{points} = 1000$, correspondingly.

Our area element now becomes:

$$dA = \sin \phi d\phi d\theta. \quad (5.28)$$

So, our joint density for uniform spacing, now, is:

$$\mathbb{P}_{\Phi, \Theta}(\phi, \theta) = c \cdot \sin \phi \quad (5.29)$$

We normalize it on the unit sphere and we get:

$$\begin{aligned} \int_0^{2\pi} \int_0^\pi \mathbb{P}_{\Phi, \Theta}(\phi, \theta) d\phi d\theta = 1 &\iff \int_0^{2\pi} \int_0^\pi c \cdot \sin \phi d\phi d\theta = 1 \iff \int_0^{2\pi} d\theta \int_0^\pi \sin \phi d\phi = 2\pi(-c \cdot \cos \phi|_0^\pi) = 1 \\ &\iff c = \frac{1}{4\pi} \end{aligned} \quad (5.30)$$

So, the joint Probability Density Function (PDF) becomes:

$$\mathbb{P}_{\Phi, \Theta}(\phi, \theta) = \frac{\sin \phi}{4\pi} \quad (5.31)$$

Integrating out θ , we find

$$f(\phi) = \mathbb{P}_{\Phi}(\phi) = \int_0^{2\pi} \mathbb{P}_{\Phi, \Theta}(\phi', \theta') d\theta' = \frac{2\pi \sin \phi}{4\pi} = \frac{\sin \phi}{2}, \quad (5.32)$$

thus:

$$F(\phi) = \int_0^\phi \mathbb{P}_{\Phi}(\phi') d\phi' = \int_0^\phi \frac{\sin \phi'}{2} d\phi' = \frac{1 - \cos \phi}{2}. \quad (5.33)$$

Inverting this, we can see that a uniform random variable would look like

$$F(\phi) = \frac{1 - \cos \phi}{2} = v \iff \cos \phi = -2v + 1 \iff \phi = \arccos(1 - 2v) \iff \phi = \arccos(2v - 1), \quad (5.34)$$

with $v \sim U(0, 1)$.

but we sample uniformly instead of randomly, so we instead use:

$$\phi_k = \arccos\left(1 - \frac{2(k + 0.5)}{N_{points}}\right) \quad (5.35)$$

where k is each current deterministic index of aranged numbers from 0 to N_{points} .

In total, we sample the 3D sphere surface:

$$\begin{aligned}
 \text{indices} &= \text{arange}(0, N_{\text{points}}) \\
 \phi &= \arccos\left(2\frac{\text{indices} + 0.5}{N_{\text{points}}} - 1\right) \\
 \theta &= \pi(1 + \sqrt{5})\frac{\text{indices} + 0.5}{N_{\text{points}}} \\
 r &= \sqrt{\frac{\text{indices} + 0.5}{N_{\text{points}}}}
 \end{aligned} \tag{5.36}$$

The roll angle γ is still uniformly sampled, like in [[29],[83]].

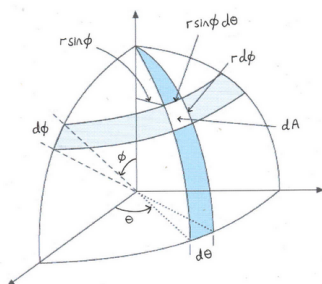


Figure 5.19: Graphical representation of a differential volumetric domain in spherical coordinates.

However, even with this deterministic sampling strategy, we have neither a theoretical nor an experimental guarantee that the sphere sampling is optimally spread in order to cover the pose space with the most even way possible. We, also, need to remember that the more uniform this sampling will be, the better generalization capability our tracker will have. Therefore, instead of sampling the exact number of N_{points} pair samples that we will train on, we sample 10 times as much (i.e. 200000) and then, we cluster them into N_{points} centroids via the KMeans algorithm (see Section 3.2.4).

We measure the effectiveness of this approach in constructing an evenly spaced spherical distribution in terms of the **maximum nearest neighbor Euclidean distance**. The best approach is the one, in which this metric gets its minimum value. We repeat the 3d sphere sampling 4 times in order to get all the possible combinations of random/deterministic point sampling, without/with the addition of the KMeans clustering improvement. We repeat each combination 10 times and we average out the results.

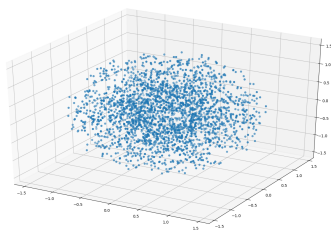


Figure 5.20: (a) Randomly sampling 20,000 points. The max distance of 2 nearest neighbors is: 0.256532 m.

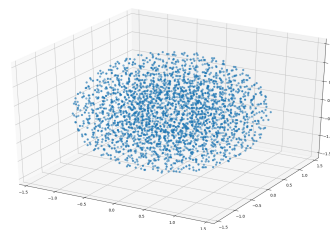


Figure 5.21: (b) Randomly Sampling 200,000 points and the clustering with KMeans to 20,000 points-centroids. The max distance of 2 nearest neighbors is: 0.222318 m.

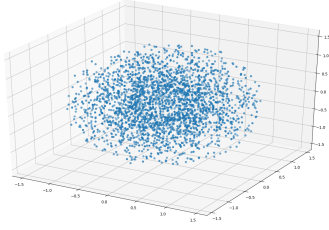


Figure 5.22: **(c)** Deterministically sampling 20,000 points. The max distance of 2 nearest neighbors is: 0.295056 m.

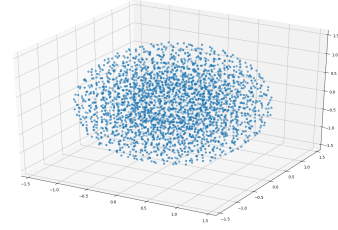


Figure 5.23: **(d)** Deterministically Sampling 200,000 points and the clustering with KMeans to 20,000 points-centroids. The max distance of 2 nearest neighbors is: 0.214753 m.

Figure 5.24: Comparing the maximum nearest neighbor Euclidean distance of randomly/deterministically sampling 3D points in a sphere without/with an intermediate point clustering KMean step of a bigger superset.

Based on the comparative analysis above, we conclude that the optimal way to sample the 20,000 3D points on the sphere as evenly as possible is to deterministically sample 200,000 points, first, and then cluster them to the, required, 20,000 centroids and save those centroids. For each of them, we uniformly sample a roll angle and we convert them to the corresponding Observed Poses. Then, for each Observed Pose, we sample a Pose Transformation, to obtain the equivalent Predicted pose and complete the sample pair.

When both the P_{Obs} , P_{Pred} poses are defined, the predicted image \hat{I}_t is obtained by rendering the object on its own by placing the virtual camera at P_{Pred} . The object is rendered using ambient occlusions, and lit with a combination of an ambient and a directional white light source of intensity 0.65 and 0.4 respectively. The directional light source points downwards with respect to the camera viewing direction. The observed image I_t is obtained by rendering the object by placing the virtual camera at P_{Obs} . Here, the light source direction is sampled uniformly on the sphere.

Chapter 6

Method

"The only people for me are the mad ones, the ones who are mad to live, mad to talk, mad to be saved, desirous of everything at the same time, the ones who never yawn or say a commonplace thing, but burn, burn, burn like fabulous yellow roman candles exploding like spiders across the stars."

Jack Kerouac, "On The Road"

6.1 Overview

Similarly to Garon et al. [[29],[83]], we train a CNN architecture on the task of Object Pose Tracking by randomly sampling batches of RGB-D frames of the rendered Observed and Predicted object poses, I_t, \hat{I}_t , correspondingly. With I_t , we notate each sample index. We wish to regress the object's pose transformation induced between the frame's visual difference, in terms of its 6D pose.

6.2 Preprocessing steps

It is possible to use the previous pose of the object to normalize its representation on the image plane with the following method:

We calculate the 3D Bounding Box corresponding to the $P_{pred}(t)$. By projecting it to the image plane, a 2D square bounding box is drawn around the object. We crop the RGB-D image pair to its boundaries and resize it with bilinear interpolation to the fixed 150x150 pixel size in order to normalize its scale. To account for large variations in pose changes, the bounding box is 15% larger than the object's size. This step brings the object to the same projection plane regardless of its pose w.r.t the camera. Then, we shift the depth map pixels to the object's center to make it invariant to its absolute depth.

6.3 Domain Randomization

Domain Randomization



If the model sees enough simulated variation, the real world may look like just the next simulator

With **Domain randomization (DR)**, we are able to create a variety of simulated environments with randomized properties and train a model that works across all of them. Likely this model can adapt to the real-world environment, as the real system is expected to be one sample in that rich distribution of training variations. In particular, the network is trained to work on various poses, occluder appearance/poses, backgrounds, textures, camera, noise distributions and lighting conditions. It is expected that, if it is successfully working in the simulation, its skills will transfer on the real world images since their statistics would roughly fall under the extremely wide distribution that was trained on. By making the training distribution extremely broad in terms of components such as texture, camera, and lighting, this method is able to ensure generalization to real world test environments by reducing the covariate shift.

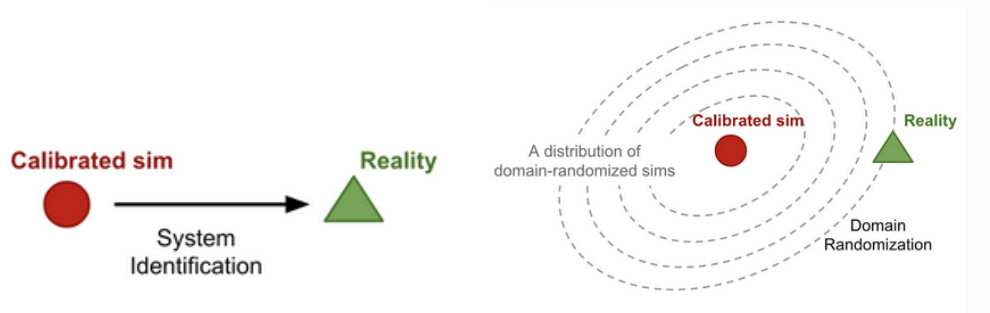


Figure 6.1: Conceptual illustrations of Domain Randomization.[152]

In general, this philosophy is practically implemented in the Generation step, but the special properties of our own, specific problem force us to share it both to the Data Generation/Sampling and Data Augmentation stages. In the former, what the pose pairs are the elements varied (remember than lighting conditions remain constant per sample), while in the latter, it is the case of all image-related transformation, and background/occluder combinations.

6.3.1 Data Augmentation



Willing to bridge the synthetic-reality domain gap, in order to enhance the Network’s robustness to a variety of situations, we simulate realistic capture conditions including a variety of backgrounds, occlusion patterns, noise, and lighting. We synthesize all the above in the Data Augmentation step.

We, mainly, follow the Data Augmentation process of [29] and [83], but we change it wherever we believe we may achieve a more meticulous augmentation either by adding extra scenarios that enrich the generalization capabilities of the dataset or by modeling noise distributions in a more realistic way. At every training iteration step, every batch element is processed with a probabilistic sequence of the following operations in order to create modified versions of the same training pairs.

Here, we have to note that Data Augmentation is meaningful only for the “Observed” RGB-D frame that, during the inference process will be derived from a real-time working Kinect sensor. The “Predicted” stream will be rendered using the feedback loop when fed the pose prediction of the previous timestep, so it will not face noisy or occluded samples.



Figure 6.2 RGB image.



Figure 6.3 Depth image.



Figure 6.4 Occluded RGB image.



Figure 6.5 Occluded Depth image.

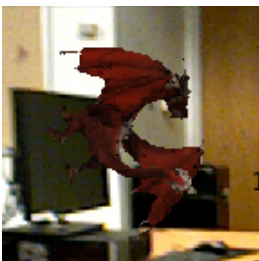


Figure 6.6 RGB image with background.

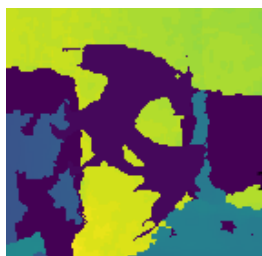


Figure 6.7 Depth image with background.

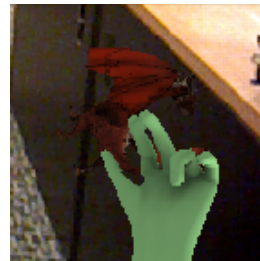


Figure 6.8 Occluded RGB image with background.



Figure 6.9 Occluded Depth image with background.



Figure 6.10 RGB image with random illumination.

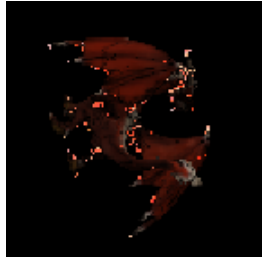


Figure 6.11 RGB image with HSV noise.



Figure 6.12 Blurred RGB image.



Figure 6.13 Blurred Depth image.

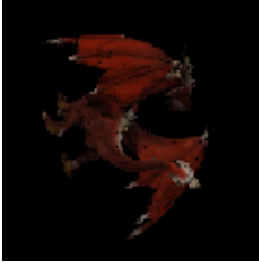


Figure 6.14 RGB image with random Gaussian RGB noise.

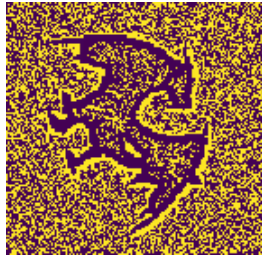


Figure 6.15 Depth image with random Gaussian Depth noise.



Figure 6.16 Depth image with the Kinect noise of [83].



Figure 6.17 Depth image with random holes

Figure 6.18: Graphical representations of the data augmentation options presented in this section.

- First of all, we create an equivalent synthetic training dataset compounded by object occluders. With a probability 60%, we blend the image of the object of interest at the “Observed” pose and a randomly posed RGB-D image of the occluder to create an occluded version of the object, using the depth channel as a z-buffer for depth comparisons. As an occluder, we use the textured 3D CAD model of a hand. In order to avoid overfitting to the hand’s texture, we randomly sample hue and luminosity values in the HSV color space (see Subsection 3.2.1), and convert that color triplet to RGB. During the blending procedure, we output a binary mask with 1’s in all pixels that the object is more closely to the camera than the hand, i.e. it is unoccluded and 0’s in the opposite case. Additionally to the procedure above, from the 60% of the train dataset that is (partially) occluded, we specify a 15% that is fully occluded as this is the case when the object is either completely hidden by the hand or put behind another bigger object in general. Naturally, in this case, the binary **Occlusion Mask** is full of 0’s.
- Secondly, just like [[29], [83]], with the intention to simulate small color shifts between the render and sensor, a random perturbation of is applied to all the channels of the HSV color space: $h \sim U(-0.07, 0.07)$, $s \sim U(-0.05, 0.05)$ and $v \sim U(-0.1, 0.1)$ (see Section 3.2.1) of the object, with a probability of 50%.



Figure 6.19: A RGB-D pair in the SUN3D [163] dataset. Such pairs are blended behind the object of interest for background augmentation.[163]

- Thirdly, the “Observed” image pair is composited in an RGB-D background pair to make the

tracker robust to different background conditions and color noise. Such background images are taken from the SUN3D [163] dataset, comprised of 415 RGBD sequences captured in 254 different places. Since, in this dataset, the camera movement is rather slow, there is a bunch of images in every video clip that have the same appearance. So, in order to cut them off, we start from the first one in every video clip and traverse it, measuring the SSIM (Section 3.2.2) image similarity metric with the current image. If that SSIM metric surpasses a 0.8 threshold, then the current RGB-D frame is counted as different and it is kept. All the others are thrown away.

We extend that augmentation process presented in [29], by allowing the background frame pair to be rotated by $0^\circ, 90^\circ, 180^\circ$ or 270° . When blending the ‘‘Observed’’ RGB-D frame pair with the equivalent Background one, we output a binary **‘‘Foreground Mask’’** with 1’s in the pixels that the object is in front of the background and 0’s in the rest. Both the ‘‘Occlusion Mask’’ and the ‘‘Foreground Mask’’ will be used as auxiliary ground truth signals for two Spatial Attention Convolutional modules, integrated into our architecture, in order to teach them to perform soft segmentation more efficiently, without any extra computational cost.

- Fourthly, we also noise our appearance frames in the RGB color space, as well. Specifically, the noise is generated by a gaussian distribution $\mathcal{N}(0, \sigma)$, where $\sigma \sim U(0, 2)$, and is added to all of the RGB channels of the image, for 95% of the dataset.
- Fifthly, we observe that in [[29],[83]] they have not taken care of different illumination conditions explicitly during data augmentation and we have already stated this one of the causes of dropping trackers’ performance. Specifically, illumination change can be conducted in a linear fashion:

$$g[i, j] = \alpha f[i, j] + \beta \tag{6.1}$$

that comes down to two distinct factors: α that controls contrast and β that controls the image brightness.

Increasing (/ decreasing) the brightness parameter value will add (/ subtract) a constant value to every pixel. Pixel values outside of the $[0, 255]$ range will be saturated (i.e. a pixel value higher (/ lesser) than 255 (/ 0) will be clamped to 255 (/ 0)).

On the other hand, contrast modifies how the levels spread. If $\alpha < 1$, the color levels will be compressed and the result will be an image with less contrast.

However, illumination changes may be execute non-linearly as well: this is succeeded via **Gamma Correction**. Gamma correction can be used to correct the brightness of an image by using a non linear transformation between the input values and the mapped output values:

$$g[i, j] = \left(\frac{f[i, j]}{255}\right)^\gamma \cdot 255. \tag{6.2}$$

Value works in conjunction with saturation and describes the brightness or intensity of the color, from 0-100%, where 0 is completely black, and 100 is the brightest and reveals the most color.

When $\gamma < 1$, the original dark regions will be brighter, whereas it will be the opposite with $\gamma > 1$. The gamma correction should tend to add less saturation effect as the mapping is non linear and there is no numerical saturation possible as in the previous method.

In our augmentation process, we uniformly sample $\alpha \sim U(0, 3)$ and $\beta \sim U(-50, 50)$ and $\gamma \sim U(0, 2)$ and we change the illumination of the RGB image using a linear or non-linear mapping with a probability of 50%.

- Sixthly, we follow [[29],[83]] in downsampling the synthetically-created depth stream frame using a 2×2 block Nearest Neighbour interpolation to simulate more realistic Kinect measurements.
- Seventhly, to account for rapid camera movement, 40% of the training images are blurred with a 3×3 mean filter. A kernel like (fig.6.20) is applied to all RGBD channels.

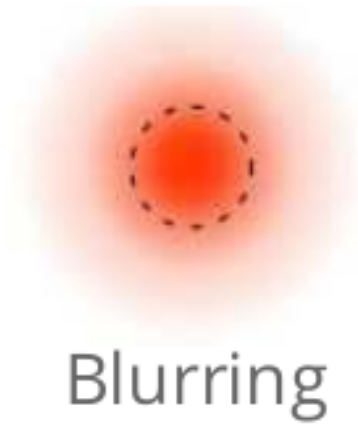


Figure 6.20: A 2D Gaussian blur kernel that is convolved with RGB-D frame pairs to blur them [29].

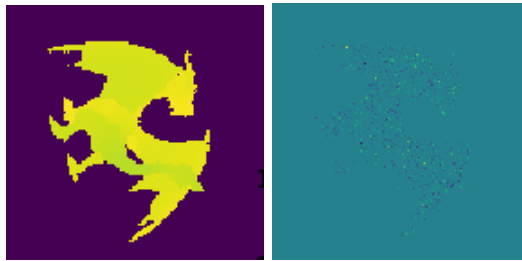
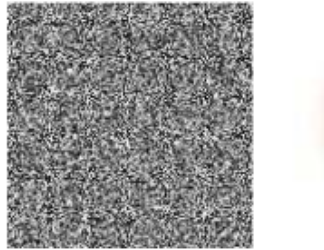


Figure 6.21: The noisy Depth image using the Kinect noise model that we describe in this Section (left) and the noise distribution image (right). We observe that the object’s edges are noised by the two lateral components and the axial one noises its core using a noise distribution that varies with the object’s depth.

- Eighthly, we improve the attempt of [[29],[83]] for simulating Kinect sensor noise. While they have added a noise derived from a simple Gaussian distribution, this is far from a realistic modeling of Kinect’s actual noise distribution, as none of the discriminative characteristics of Kinect’s emblazonment process are taken into account.



Random Noise

Figure 6.22: The Gaussian random noise added to augment Depth images in [[29],[83]].

Kinect sensor noise is in fact the difference between the raw Depth image that Kinect outputs and the Ground Truth one it should have outputed. In this approach, we adopt the modeling of Nguyen et al., in [91], where they have synthesized a 3D Gaussian Kinect Noise distribution $\mathcal{N}_{Kinect3D}$ from which they sample the noise to add to the synthetic image. This distribution is formed based on the observation that Kinect noise is divided into three spatial components: one for **Axial Noise** $\mathcal{N}(0, \sigma_{N_A})$, in the direction of the camera principal axis, which is responsible for the increased variance in reported depth as the distance between the sensor and observed surface increases and two **Lateral Noise** components: $\mathcal{N}(0, \sigma_{N_L^{(x)}})$ and $\mathcal{N}(0, \sigma_{N_L^{(y)}})$ which are to blame for making straight object boundaries appear as distorted zig-zag lines. Mathematically, they express the Point Spread Function (PSF) that is approximated by the absolute distances from the observed

edge pixels to a fitted straight edge the target. So, the standard deviation (STD) of Lateral noise has been measured as the STD of the edge distance distribution.

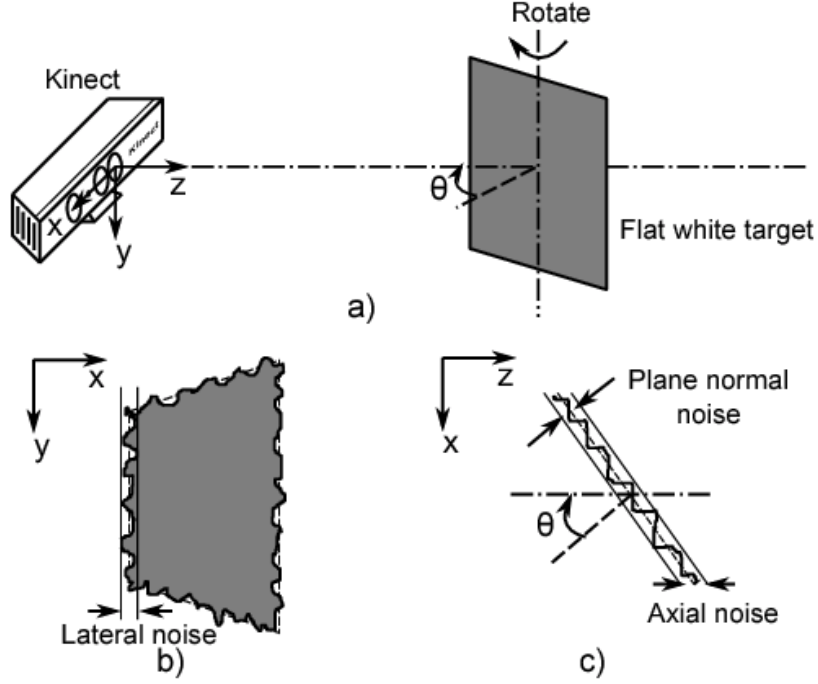


Figure 6.23: The experimental setup that was used in [91] for deriving the Lateral and Axial kinect noise distributions.

The above standard deviations are extracted through an experimental setup where a planar object is set in front of the camera at different combinations of depth and rotation angle of the object, that are taken from a predefined range, as it can be seen in fig.6.23. Different noisy Depth image samples are created with the application of the combinations above and the authors extracted mathematical models for the standard deviation of each of the distribution by a suitable curve fitting action. This experimental setup indicates that noise is not just a function of the distance between the object and the camera, but also of its rotation angle w.r.t. an axis perpendicular to the camera's principal axis, as follows:

Axial Noise:

$$\sigma_{NA}(z, \theta_y) = 0.0012 + 0.0019 \cdot (z - 0.4)^2 + \frac{0.0001}{\sqrt{z}} \cdot \frac{\theta_y^2}{(\pi/2 - \theta_y)^2} \quad (6.3)$$

We can see that the Axial component depends heavily on z . Its relationship with the angle θ_y is more complex:

- for $\theta_y \in [0^\circ, 60^\circ]$ it stays approximately the same and
- for $\theta_y \in [60^\circ, 90^\circ]$ the squared term dominates the equation and the Axial Noise increases following the same trend.

Lateral noise:

$$\sigma_{NL}(\theta_y) = 0.8 + \frac{0.035 \cdot \theta_y}{\frac{\pi}{2} - \theta_y}, \quad (6.4)$$

in pixel units [pxls].

$$\begin{aligned} \sigma_{N_L^x}(z, \theta_y) &= \sigma_{NL}[\text{pxls}] \cdot z \frac{c_x}{f_x} \\ \sigma_{N_L^y}(z, \theta_y) &= \sigma_{NL}[\text{pxls}] \cdot z \frac{c_y}{f_y} \end{aligned} \quad (6.5)$$

in meters [m].

$c_{x/y}$ are the camera center components and $f_{x/y}$ the corresponding focal lengths.

The Lateral components, on the other hand, depend lightly on the depth z , especially as it increases.

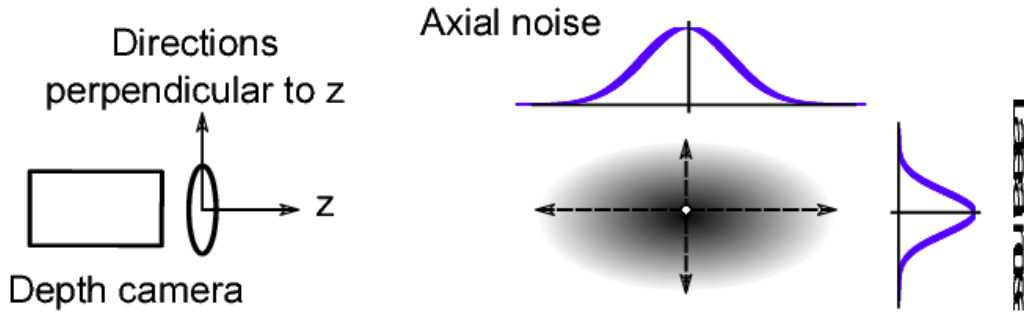


Figure 6.24: A 3D Gaussian Distribution that we use to model the Kinect noise, which is the product of 2 Lateral and 1 Axial noise components w.r.t. the camera's principle axis[91].

The final 3D Kinect noise distribution is a product of the three (fig.6.24):

$$\mathcal{N}(0, \sigma_{Kinect3D}) = \mathcal{N}(0, \sigma_{N_A}) * \mathcal{N}(0, \sigma_{N_L^{(x)}}) * \mathcal{N}(0, \sigma_{N_L^{(y)}}) \quad (6.6)$$

- Finally, we further augment the depth modality by adding a random number of different hole patterns to each Observed depth map. In particular, these holes are predefined patches with random scale. A random number of patches is added to depth at a random x-y pixel position.

Next, the reader may observe several alterations of the same Observed RGB-D frames under random combinations of the aforementioned augmentation strategies.

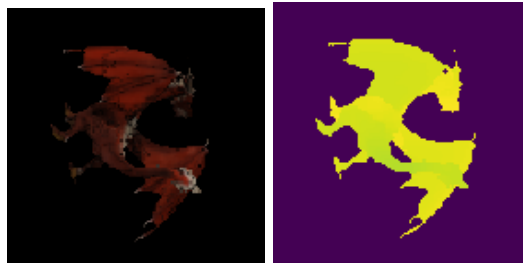
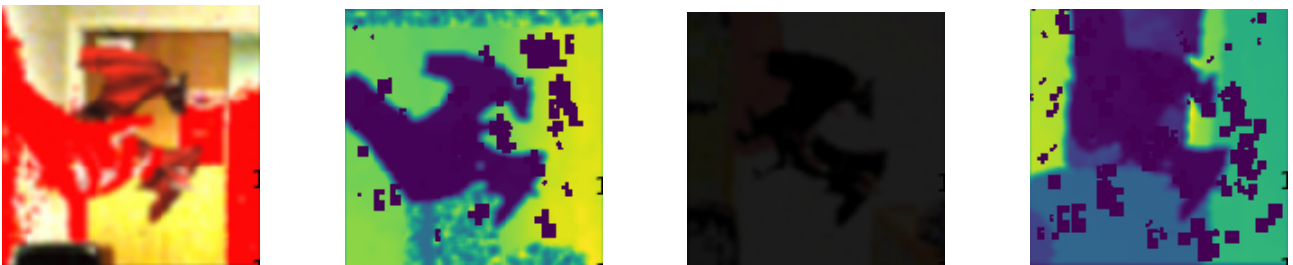


Figure 6.25: An indicative random Observed RGB-D frame pair that we showcase random augmentation combinations of our overall strategy.

We remind that the Predicted RGB-D frame is in no need of augmentation, as it will be synthetically rendered, during inference, as well.



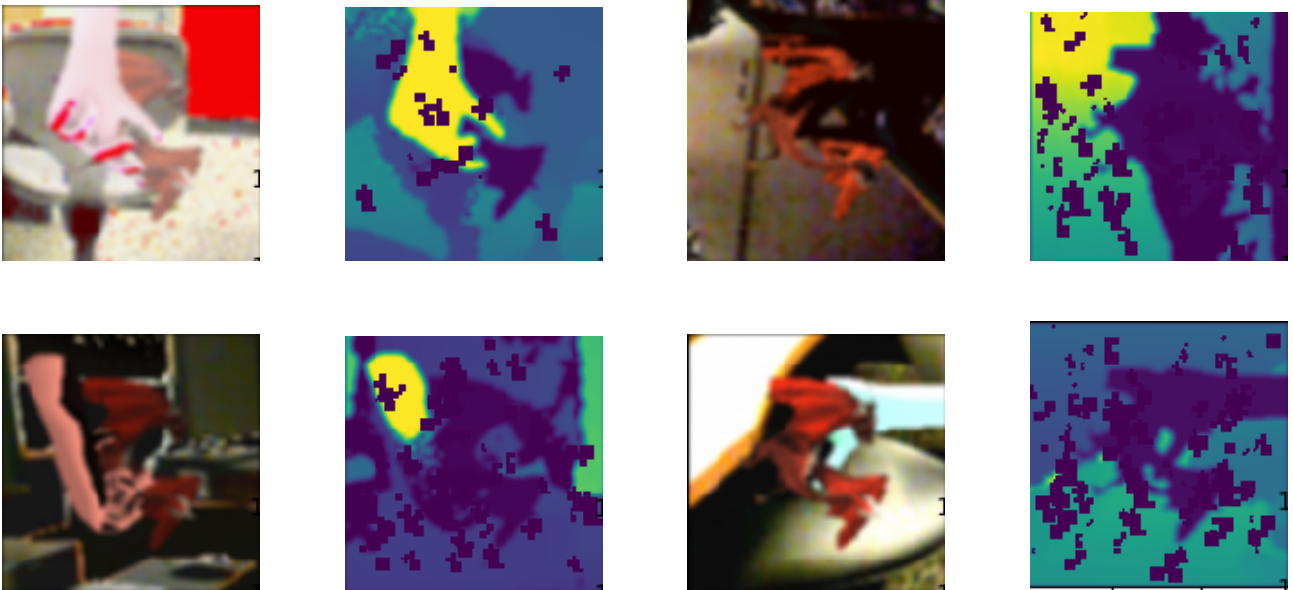


Figure 6.26: Full random augmentation examples of the same Observed RGB(left)-D(right) frame.

6.4 Baseline Approach

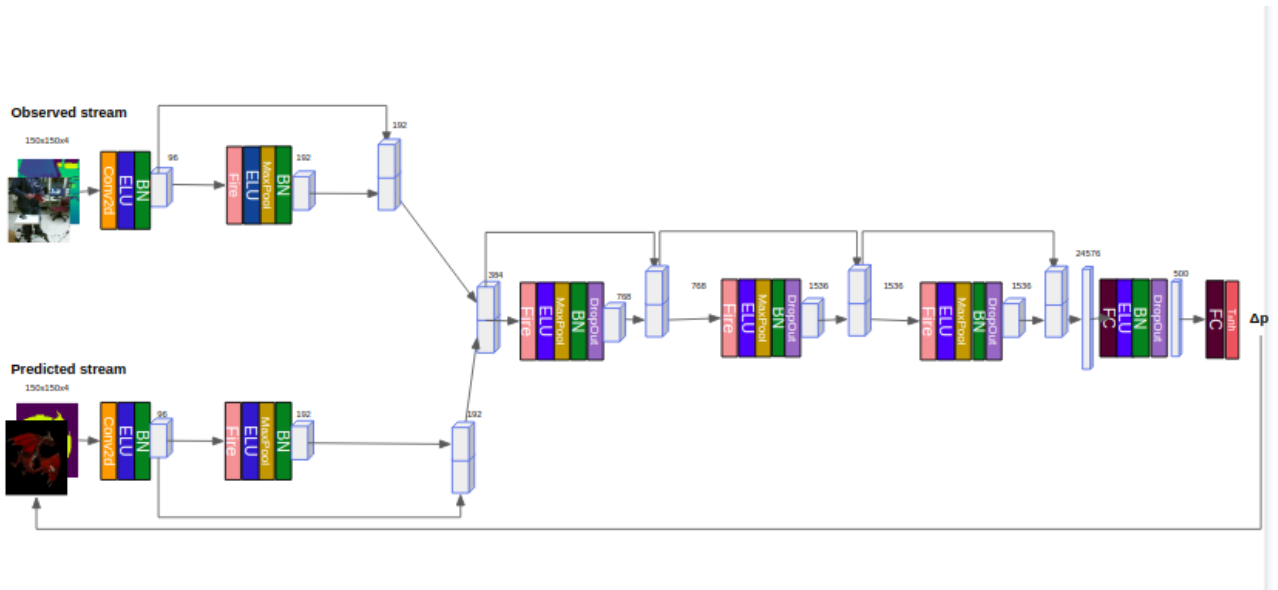


Figure 6.27: Overview of the SoA CNN architecture of Garon et al. [83], for object pose tracking.

The main scope of this work is to improve the results of Garon et al. [83]. What they did is to, grosso modo, combine the data generation procedure we analyzed, and improved, above, with the data augmentation strategy that we have, also, unrolled (without our own insightful interventions in it) and the baseline strategy presented in this section. This architectural design consists of two convolutional streams that process two RGB-D images pairs, of the same object in adjacent pose configurations, that they fuse at the end of the second layer and proceeds to regress the pose transformation between the two frames. At the end, the estimated pose transformation is fed back to the input layer and is applied to the previous pose of one of the two streams (the “Predicted” one) to generate the current pose and re-iterate the regressed estimation.

6.5 Proposed Architecture

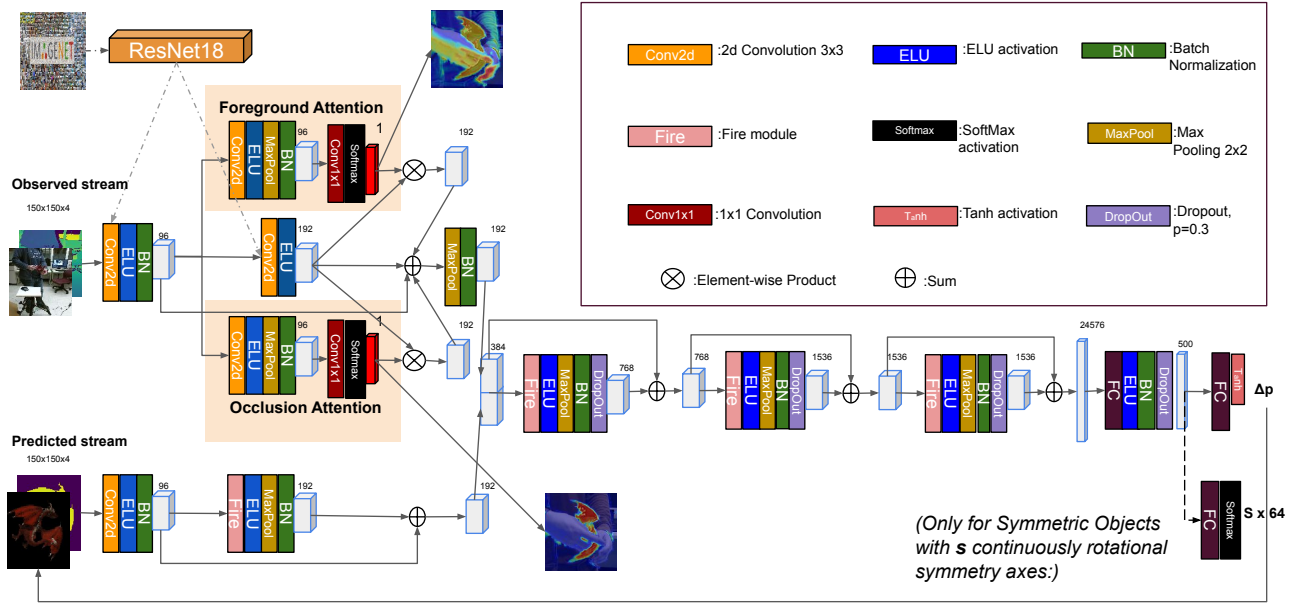


Figure 6.28: Overview of the proposed CNN architecture for object pose tracking.

6.6 Input Normalization

After augmentation, following [[29],[83]], we standardize all cue channels using their mean and standard deviation values and then, we normalize them to $[-1,1]$.

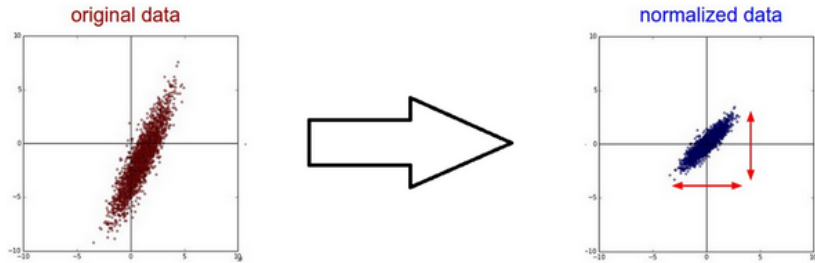


Figure 6.29: Standardization of the data distributions of the CNN inputs (both streams). The new mean converges to zero and the new standard deviation converges to 1.[105]

Following, we explicitly report the details of these procedures.

Normalization: After standardizing all input channels to a Normal distribution $\mathcal{N}(0, 1)$, we normalize their values in the range $[-1,1]$ as follows:

$$\mathbf{I}_t^{Norm} = 2 \frac{\mathbf{I}_t - \min(\mathbf{I}_t)}{\max(\mathbf{I}_t) - \min(\mathbf{I}_t)} - 1 \in [-1, 1], \quad (6.7)$$

$$\mathbf{I}_t = [I_t, \hat{I}_t] \in \mathbb{R}^{H \times W \times 8}$$

Standardization:Welford's Algorithm [151]

This is not as straightforward as it initially seems to. That is because the synthetic dataset is too big to make use of the basic standard definition of mean and variance for their respective calculations:

$$\begin{aligned}\bar{x} &= \frac{1}{N} \sum_{i=1}^N x_i \\ \sigma^2 &= \frac{1}{N} \sum_{i=1}^N (x_i - \bar{x})^2,\end{aligned}\tag{6.8}$$

where N is the number of total train samples (in our case: 20,000 RGB-D pairs). To this end, we need to calculate the mean and the squared difference recursively, in mini-batches, with a complexity of $\Theta(2N)$. Mean can be easily calculated recursively in the following way:

$$\bar{x}_N = \frac{N-1}{N} \bar{x}_{N-1} + \frac{1}{N} x_N\tag{6.9}$$

However, this method is proven to be arithmetically unstable and can be substituted by its stable alternative version [105]:

$$\bar{x}_N = \bar{x}_{N-1} + \frac{x_N - \bar{x}_{N-1}}{N}\tag{6.10}$$

Notice that the former formula is unstable because of the factor $\frac{1}{N}$ (we are multiplying \bar{x}_{N-1} by $\frac{N-1}{N}$ and x_N by $\frac{1}{N}$, quantities that lead to very different numbers in case N is either large or small).

From now on, the notation N changes from the cardinality of the whole dataset to the cardinality of each mini-batch under study.

As for the variance, by expanding the sum we can rewrite σ^2 in the following manner:

$$\sigma^2 = \frac{1}{N} \sum_{i=1}^N (x_i - \bar{x})^2 = \frac{1}{N} \sum_{i=1}^N x_i^2 + \bar{x}^2 - 2x_i \bar{x} = \frac{N\bar{x}^2 - 2N\bar{x}^2 + \sum_{i=1}^N x_i^2}{N}\tag{6.11}$$

which is equal to

$$\sigma^2 = \frac{\sum_{i=1}^N x_i^2 - N\bar{x}^2}{N}\tag{6.12}$$

But, we haven't answered our problem yet: we need a $O(1)$ recursive algorithm. Usually, to end up with a recursive formula of the type:

$$S_N = S_{N-1} + M_N\tag{6.13}$$

is best to calculate the difference

$$S_N - S_{N-1}!\tag{6.14}$$

In our case, then, we need to calculate

$$\sigma_N^2 - \sigma_{N-1}^2,\tag{6.15}$$

but, for simplicity we can just consider

$$M_N = N\sigma_N^2 - (N-1)\sigma_{N-1}^2:\tag{6.16}$$

$$M_N = N\sigma_N^2 - (N-1)\sigma_{N-1}^2 = \sum_{i=1}^N (x_i - \bar{x}_N)^2 - \sum_{i=1}^{N-1} (x_i - \bar{x}_{N-1})^2.\tag{6.17}$$

Consider the first sum from 1 to $N-1$ so that we can write only one big sum:

$$M_N = (x_N - \bar{x}_N)^2 - \sum_{i=1}^{N-1} ((x_i - \bar{x}_N)^2 - (x_i - \bar{x}_{N-1})^2).\tag{6.18}$$

Focusing on the term inside the series we can expand it:

$$\sum_{i=1}^{N-1} ((x_i - \bar{x}_N)^2 - (x_i - \bar{x}_{N-1})^2) = \sum_{i=1}^{N-1} (\bar{x}_N^2 - 2x_i\bar{x}_N + 2x_i\bar{x}_{N-1} - \bar{x}_{N-1}^2) \quad (6.19)$$

and notice that we can collect the terms inside the series as follows:

$$\sum_{i=1}^{N-1} (\bar{x}_N^2 - 2x_i\bar{x}_N + 2x_i\bar{x}_{N-1} - \bar{x}_{N-1}^2) = \sum_{i=1}^{N-1} (\bar{x}_N - \bar{x}_{N-1})(\bar{x}_N + \bar{x}_{N-1} - 2x_i), \quad (6.20)$$

and notice that:

$$\sum_{i=1}^{N-1} \bar{x}_N + \bar{x}_{N-1} - 2x_i = (N-1)(\bar{x}_N + \bar{x}_{N-1} - 2\bar{x}_{N-1}) = (N-1)(\bar{x}_N - \bar{x}_{N-1}) \quad (6.21)$$

Still we are not done, because we would like to get rid of the term $(N-1)$ somehow.

$$(N-1)(\bar{x}_N - \bar{x}_{N-1}) \quad (6.22)$$

is simply:

$$(N-1)(\bar{x}_N - \bar{x}_{N-1}) = \frac{N-1}{N} \sum_{i=1}^N x_i - \sum_{i=1}^{N-1} x_i = \frac{(N-1) \sum_{i=1}^N x_i - N \sum_{i=1}^{N-1} x_i}{N}, \quad (6.23)$$

which is equal to:

$$(N-1)(\bar{x}_N - \bar{x}_{N-1}) = \frac{\sum_{i=1}^N x_i - Nx_N}{N} = \bar{x}_N - x_N. \quad (6.24)$$

Therefore, collecting all pieces, we can state that:

$$M_N = (x_N - \bar{x}_N)^2 - (\bar{x}_N - \bar{x}_{N-1})(\bar{x}_N - x_N) = (x_N - \bar{x}_N)(x_N - \bar{x}_N + \bar{x}_N - \bar{x}_{N-1}), \quad (6.25)$$

which leaves us with the final formulae:

$$M_N = (x_N - \bar{x}_N)(x_N - \bar{x}_{N-1}). \quad (6.26)$$

Finally:

$$S_N = S_{N-1} + M_N = S_{N-1} + (x_N - \bar{x}_N)(x_N - \bar{x}_{N-1}) \Rightarrow \sigma_N^2 = \frac{S_N}{N}, \quad (6.27)$$

where we divide S_N by N to obtain the variance (remember that we collected N and $N-1$ at the beginning).

Last but not least, we divide S_N by $N-1$, to get an unbiased Standard deviation estimator and we calculate the square root of the quotient:

$$\sigma_N = \sqrt{\frac{S_N}{N-1}}. \quad (6.28)$$

Practical aspects:

Here, we need to note that we do not know a-priori the image mean pixel and standard deviation of pixels per input sample, as they are originated from real test samples that we have no access to. As a result, following a ‘‘Domain Randomization’’ (see next Section) thought process, we estimate their value using the only set we have available: the training set. As it is natural, for the ‘‘Predicted’’ input streams, mean \pm standard deviation values are estimated closely to the real ones, since no extra postprocessing is applied to feedback-rendered pose prediction during the inference stage, no occluder covers parts of the object and the background is kept black both for the training and the inference stage. Naturally, this is not the case for the ‘‘Observed’’ input streams. There, the means are different, as a large set

of different background conditions has to be accounted for and the standard deviation is larger, since background and occluder texture variety of the training set far exceeds that of the test video clip for each scenario. However, they are of the same scale, something that makes this tactic reasonable. As a matter of fact:

Real “Dragon” model image means \pm standard deviation:

Mean vector:

"Pred" R mean	"Pred" G mean	"Pred" B mean	"Pred" Depth mean	"Obs" R mean	"Obs" G mean	"Obs" B mean	"Obs" Depth mean
11.81	5.54	3.97	4047.27	75.17	66.54	64.94	1060.17

Std vector:

"Pred" R mean	"Pred" G mean	"Pred" B mean	"Pred" Depth mean	"Obs" R mean	"Obs" G mean	"Obs" B mean	"Obs" Depth mean
2.09	0.55	0.41	98.47	9.84	9.64	8.99	385.58

Synthetic “Dragon” model image means \pm standard deviation:

Mean vector:

"Pred" R mean	"Pred" G mean	"Pred" B mean	"Pred" Depth mean	"Obs" R mean	"Obs" G mean	"Obs" B mean	"Obs" Depth mean
10.78	4.80	3.40	4043.63	94.90	81.80	73.89	1161.820

Std vector:

"Pred" R mean	"Pred" G mean	"Pred" B mean	"Pred" Depth mean	"Obs" R mean	"Obs" G mean	"Obs" B mean	"Obs" Depth mean
2.26	0.77	0.57	112.96	54.09	49.87	48.53	6199.96

Real “Cookie Jar” model image means \pm standard deviation:

Mean vector:

"Pred" R mean	"Pred" G mean	"Pred" B mean	"Pred" Depth mean	"Obs" R mean	"Obs" G mean	"Obs" B mean	"Obs" Depth mean
11.91	11.37	9.44	3772.02	68.53	64.11	56.83	304.62

Std vector:

"Pred" R mean	"Pred" G mean	"Pred" B mean	"Pred" Depth mean	"Obs" R mean	"Obs" G mean	"Obs" B mean	"Obs" Depth mean
2.83	2.86	2.08	78.99	6.71	5.56	5.33	104.02

Synthetic “Cookie Jar” model image means \pm standard deviation:

Mean vector:

"Pred" R mean	"Pred" G mean	"Pred" B mean	"Pred" Depth mean	"Obs" R mean	"Obs" G mean	"Obs" B mean	"Obs" Depth mean
8.41	7.58	7.98	3829.42	89.44	79.66	73.79	1090.39

Std vector:

"Pred" R mean	"Pred" G mean	"Pred" B mean	"Pred" Depth mean	"Obs" R mean	"Obs" G mean	"Obs" B mean	"Obs" Depth mean
2.83	2.33	1.75	119.90	50.49	47.30	46.46	5740.17

6.6.1 Pretrained Resnet Weights - Initialization

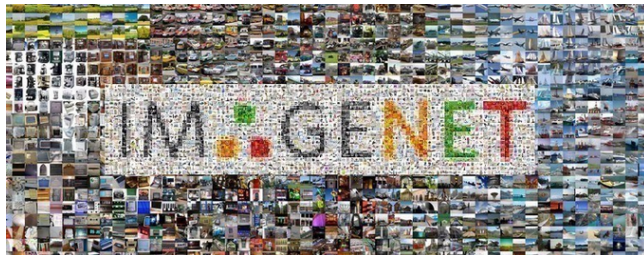
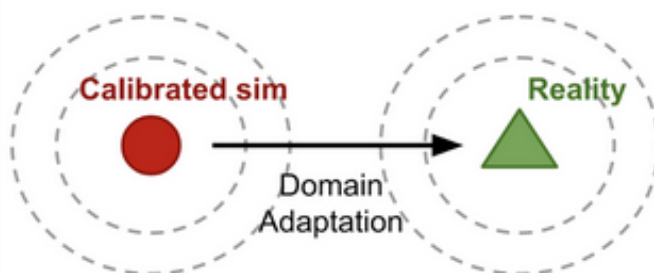


Figure 6.30: The “ImageNet” classification dataset, first presented in [42]. It is considered one of the toughest challenges for image-based classification and it contains 1 Million training images. As a result, weights of NN architectures pretrained on it constitute a very reliable initialization attempt for architectures later finetuned on other tasks. That is because the Imagenet dataset possesses great appearance and categorical variability, giving the current architecture a robust prior understanding of the world.

The proposed CNN architecture takes as input the two RGB-D 150×150 image frames: $I(t), \hat{I}(t)$ and regresses a continuous output pose representation $\Delta \mathbf{p} \in \mathbb{R}^9$, with 3 parameters for translation and 6 for rotation. We explain our rotation representation choice in Section 7.14. We notate as $X_{Source}^{(l)}$ the $C \times H \times W$ feature maps of the $l + 1$ layer, derived from the corresponding image source $Source = \{Pred/Obs\}$.

The full architecture of our Network is shown in fig. 6.28.

Domain Adaptation refers to a set of transfer learning techniques developed to update the data distribution in simulation to match the real one through a mapping or regularization enforced by the task model.



The first two layers of the Observed convolutional stream are initialized with the weights of Resnet18 (Section 3.30) pretrained on Imagenet to narrow down the real-synthetic domain adaptation gap, as proposed in [42]. Note that, in contrary to their work, the dimension of our RGB-D input is 4, instead of 3. Since depth only consists of one channel and there are no equivalent ImageNet datasets available for depth, we average out copies of each of the RGB pretrained weights for the depth input and we concatenate them with the corresponding weights that are responsible for the RGB stream, as normal. The motivation for this is that the network looks for similar features in both the depth maps and RGB images, and thus will still benefit from RGB pretrained weights over randomly initialized weights. In contrary to their results, however, we find beneficial not to freeze those two layers during training. We justify this result because we just aim to track the pose of the single objects we train on and not to

generalize to unseen ones, so overfitting to a specific object’s features improves the ability to distinguish its pose change. One would state that the mismatch between our results and those of the paper is caused by the absence of the depth modality in the study of Hintenkoisser et al.[42]. This would seem like a reasonable objection, at first, but we assess that it is not the cause, since we have taken this hinder into account and have left the weights that correspond to the depth modality to finetune in both scenarios. We apply the learnable Soft Spatial Attention Maps for Background Extraction and Occlusion Handling (that we describe in detail in Sections 6.6.3,6.6.4) to the output of the second Observed layer and we add their output feature maps with the one of the second layer, along with a Residual connection [37] from the first layer. Generally, we employ Residual connections between all convolutional consecutive layers in our Network. After that, we pass our feature map from three sequential Fire modules [51]. We prefer the use of Fire modules instead of standard 2D convolutional layers, as they require almost half the parameters without compromising the network’s performance (see Section 3.3.6).

The Predicted stream’s weights are initialized via the Kaiming He strategy (see Section 3.3.10), as they are in no need of domain adaptation. The concatenated feature map of the second layer passes from three sequential Fire modules with cross-layer Residual connections. Those Fire layers are, also, initialized with the Kaiming He strategy. Lastly, we employ a sequence of 2 Fully Connected layers, the first one initialized by Kaiming He strategy and the second one by the Xavier strategy (Section 3.3.10). As for the two parallel Attention blocks, the first one is initialized with the Kaiming He scheme and the second one using the one proposed by Xavier et al.

Next, we highlight the most important components of our design:

6.6.2 Residual Connections

Generally, we employ Res-connections (Section 6.6.2) between all consecutive Convolutional layers of our Network (instead of the Dense connections of 3.3.12, as they boost our performance). In particular, a big percentage of the decrease our architecture brings to both of the error metrics comes from replacing the Dense connections [49] (concatenation) with Residual ones [37]. This is shown even if this is the only architectural design choice we make in converting the baseline architecture of [83]. This effectiveness, may seem unreasonable at first, especially since previous experimental results (Section 6.6.2) are indicative of the opposite, but it can easily be justified if we look closer to the differences between those two methodologies. Since pose tracking is a more heavily spatial task than object classification, where the Dense connections were first introduced, it is profoundly more intuitive to add propagated pixels from previous layers to the current ones, instead of concatenating them, something that disrupts this strict spatial relationship.

6.6.3 Soft Spatial Attention for Background Extraction

For the auxiliary goal of background clutter, we construct a convolutional Spatial Attention module. In particular, we pass the output of the second Observed convolutional layer via a specially dedicated 2D convolutional layer. Then, we pass its output feature map via a 1×1 convolutional layer to squeeze its 96 channels to 1. In order to assure that this Spatial Weight map has a probability-like structure, we pass it via a 2D Softmax activation function. Mathematically,

$$Att_{Foregr} = Softmax^{(2D)}(Conv_{1 \times 1}(ELU(Conv_{2D}^{(Foregr)}(X_{Obs}^{(1)})))) \quad (6.29)$$

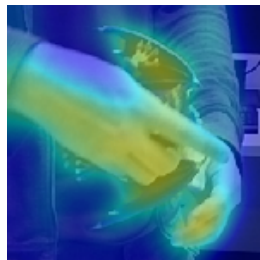


Figure 6.31: A Spatial Attention weight map dedicated to Foreground extraction. The highlighted pixels are the ones of the moving parts of the scene.

6.6.4 Soft Spatial Attention for Occlusion Handling

For the auxiliary goal of occlusion handling, we construct a convolutional Spatial Attention module. In particular, we pass the output of the second Observed convolutional layer via a specially dedicated 2D convolutional layer. Then, we pass its output feature map via a 1×1 convolutional layer to squeeze its 96 channels to 1. In order to assure that this Spatial Weight map has a probability-like structure, we pass it via a 2D Softmax activation function. Mathematically,

$$Att_{Occl} = Softmax^{(2D)}(Conv_{1 \times 1}(ELU(Conv2D^{(Occl)}(X_{Obs}^{(1)})))) \quad (6.30)$$

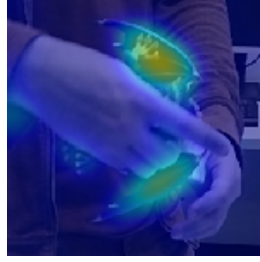


Figure 6.32: A Spatial Attention weight map dedicated to Occlusion handling. The highlighted pixels are the ones that correspond only to the object of interest.

6.6.5 Linear Layers

After the last (5th) sequential Convolutional layer, we flatten its output $X_{Source}^{(4)}$ and then we pass it from a sequence of 2 Fully Connected (Linear) layers: the first with 500 neurons and the second one with 9 neurons: 3 for translation and 6 for rotation. In the next section, we explain the nature of each output component and how they are properly transformed in order to be included in the overall tracking loss.

6.6.6 Output Transformation

As far as it concerns the translation parameters, we constrain the translation parameters ($\hat{t}_{x,y,z}$) to $[-1,1]$ by passing them through a Tanh (Section 7.8) activation function. As for rotation, in order to minimize the rotation errors due to ambiguities caused by the parameterization choice, we employ the 6D continuous rotation representation that was introduced in [171]: $\Delta \mathbf{r} = (\Delta \mathbf{r}_x^T, \Delta \mathbf{r}_y^T)^T$, where $\Delta \mathbf{r}_{x/y} \in \mathbb{R}^3$. Given $\Delta \mathbf{r}$, the matrix $\Delta R = (\Delta \mathbf{R}_x, \Delta \mathbf{R}_y, \Delta \mathbf{R}_z)^T$ is obtained by:

$$\begin{aligned} \Delta \mathbf{R}_x &= N(\Delta \mathbf{r}_x) \\ \Delta \mathbf{R}_y &= N[\Delta \mathbf{r}_y - (\Delta \mathbf{R}_x^T \cdot \mathbf{r}_y) \cdot \Delta \mathbf{R}_x] \\ \Delta \mathbf{R}_z &= \Delta \mathbf{R}_x \times \Delta \mathbf{R}_y \end{aligned} \quad (6.31)$$

where $\Delta \mathbf{R}_{x/y/z} \in \mathbb{R}^3$, $N(\cdot) = \frac{(\cdot)}{\|(\cdot)\|}$ is the normalization function.

6.6.7 Multitask Loss function

We train our network by minimizing a Multi-Task loss function that is comprised of a primary learning task: Pose Tracking and two auxiliary ones: Foreground and Occlusion handling.

As far as it concerns Pose Tracking, we weigh the translation and rotation losses using a pair of learnable weights $\mathbf{v} = [v_1, v_2]^T$ that are trained along with the rest of the network's parameters using a Gradient Descent-based optimization method, as proposed by [60].

$$\begin{aligned} L_{Track}(\Delta \hat{\mathbb{P}}, \Delta \mathbb{P}_{GT}) &= e^{-v_1} \cdot L_{Transl}(\Delta \hat{\mathbf{t}}, \Delta \mathbf{t}_{GT}) + e^{-v_2} \cdot L_{Rot}(\Delta \hat{R}, \Delta R_{GT}) + v_1 + v_2 = \\ e^{-v_1} \cdot \left\| (\Delta \hat{\mathbf{t}}, \Delta \mathbf{t}_{GT}) \right\|_2^2 &+ e^{-v_2} \cdot \arccos \left(\frac{\text{tr}((\Delta \hat{R} \cdot \Lambda_{(G.S.)})^T \cdot \Delta R_{GT} \cdot \Lambda_{(G.S.)}) - 1}{2} \right) + v_1 + v_2 \end{aligned} \quad (6.32)$$

The side losses that address the tasks of Foreground and Occlusion handling are two Binary Cross-Entropies (Section 3.3.17) between the corresponding Spatial Attention maps and the equivalent Binary Masks, saved during data augmentation (Section 6.3.1).

•

$$\begin{aligned} L_{Occl}(Att_{Occl}, M_{Occl}^{(Binary)}) &= BCE(Att_{Occl}, M_{Occl}^{(Binary)}) = \\ &= - \sum_{i=1}^H \sum_{j=1}^W M_{Occl}^{(Binary)}[i, j] \cdot \log(Att_{Occl}[i, j]) \end{aligned} \quad (6.33)$$

•

$$\begin{aligned} L_{Foregr}(Att_{Foregr}, M_{Foregr}^{(Binary)}) &= BCE(Att_{Foregr}, M_{Foregr}^{(Binary)}) = \\ &= - \sum_{i=1}^H \sum_{j=1}^W M_{Foregr}^{(Binary)}[i, j] \cdot \log(Att_{Foregr}[i, j]) \end{aligned} \quad (6.34)$$

Finally, we use a similar external multi-task learnable weighting scheme ($\mathbf{s} = [s_1, s_2, s_3]^T$), introduced in Section 3.3.20 and we combine our primary task, with the two auxiliary ones:

$$Loss = e^{-s_1} \cdot L_{Track} + e^{-s_2} \cdot L_{Occl} + e^{-s_3} \cdot L_{Foregr} + s_1 + s_2 + s_3 \quad (6.35)$$

Important Note 1:

Something that could be passed unnoticed is that those learnable weights have their own functional limitations. As we have stated in the “Background” (Chap.3) chapter, this scheme can learnably scale up or down the individual tracking component losses in order to balance out their impact and facilitate training. However, its capabilities are not limitless. For example, the rotational distance must be calculated in **rad** in order for the scheme to be successful. That is because, in the opposite case of degree-based calculation the two numerical scales between translation and rotation become so uneven that an extremely large learning rate would be required for a true balancing attempt. However, we have seen that such large learning rates are unfavorable, as it would destabilize the training of those two parameters. In the radiant case, training is much smoother and the learnable weights converge extremely fast to relatively flat-local minimum- surface.

Important Note 2:

Although the mathematic formulation of Sect.8.2 dictates that the multi-task regularization penalty should be of the fashion $2 \cdot s_i$, omitting the weighting factor of “2” and, thus, imposing less regularization to the learning process was shown to be practically better along a series of experiments, making us to alter the formulation.

6.6.8 Symmetric Object Handling

Object Geometry-induced symmetries: Beyond the problem of proper rotational representation, pose ambiguities, also, derive from the object’s geometrical structure. In particular, as we have already stated in Section 2.10.2, objects with one or more axes of rotational symmetry cause pose representations to have multiple equivalent instances that do not change the underlying pose context (see fig.6.33). When the object appearance modality is available, along with its 3D shape one, it helps breaking this symmetry and reducing alternative pose representations to a unique one. However, this breaking is far from absolute and it leaves room for some remaining uncertainty as repeated appearance patterns are frequent in most household items. Besides, even if this assumption (of the appearance-based pose disentanglement) is the case, leaving it for the Network to figure out by itself, without explicit guidance in the loss function, is susceptible to suboptimal regressive performance.

Viewpoint-induced symmetries: However, one should take other sources of symmetry into consideration when designing a Pose Estimator/Tracker. The most prominent case left, is that of symmetries caused not by any special geometric feature of the object itself, but by set of camera configurations allowed in dataset generation. Whether one should account for them in their loss function creation is a

topic much more visible than in the previous case. That is because, in this scenario, the symmetry set is more frequently discrete and the symmetry-induced errors can not be marginal rotational drifts, caused by jittered regression, that accumulate over time, but are constrained to be extreme phenomena (e.g. the object’s pose prediction turned upside down with a registered error of 180° in one of its rotational components). If that is the case, a more simple and straightforward strategy (similar to the one we analyzed in the Pix2Pose [98] approach) of assigning the right rotational value from the symmetry set when the rotational error is extremely high in one component only, can be successfully employed. Thus, an incorporation of this corner-case in the training loss function structure is not necessary.

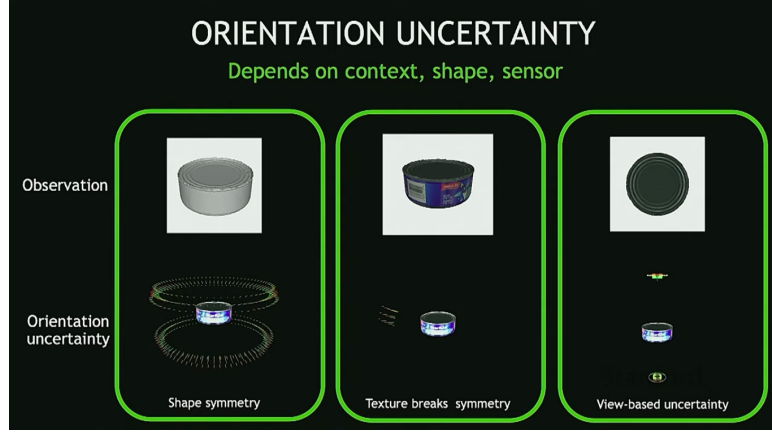


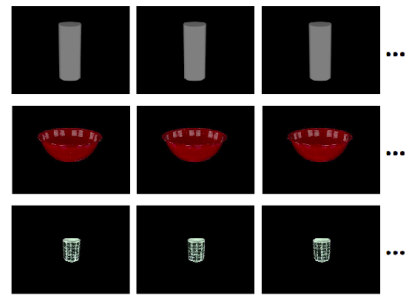
Figure 6.33: All possible rotational uncertainties induced by all possible symmetrical scenarios.[27]

Based on the discussion above, in the special case of symmetric objects, with continuous symmetry, we disentangle the ambiguities inserted due to this property from the core of the rotation estimation. We regress a separate Euler angle triplet of symmetry-based parameters $\hat{\mathbf{g}} \in \mathbb{R}^3$ that is converted to a rotation matrix \hat{G} , which gets right-multiplied with $\Delta\hat{R}$ before being weighted by the parameters of $\Lambda_{(G.S.)}$. We used a cylindrical Cookie Jar model for the symmetric object case, the shape of which has only one axis of symmetry and a repetitive texture pattern. Consequentially, we estimate a single symmetry parameter, that of the object-centric z-axis. Before the conversion, that parameter is passed through a tanh function and multiplied by π to constrain its values.

Examples of identified discrete symmetries



Examples of identified continuous symmetries



As a result, our overall tracking loss function is transformed to:

$$L_{Track}(\Delta\hat{\mathbb{P}}, \Delta\mathbb{P}_{GT}) = e^{(-v_1)} \cdot MSE[(\Delta\hat{\mathbf{t}}, \Delta\mathbf{t}_{GT})] + v_1 + v_2 + e^{(-v_2)} \cdot \arccos\left(\frac{\text{tr}((\Delta\hat{R} \cdot \hat{G} \cdot \Lambda_{(G.S.)})^T \cdot (\Delta R_{GT} \cdot \Lambda_{(G.S.)})) - 1}{2}\right) \quad (6.36)$$

Minimizing of the overall tracking loss w.r.t. the symmetry matrix \hat{G}^* (see Brégier et al. [11]) does not explicitly impose a global-solution constraint, something that gives us the freedom to select how the continuously rotational symmetry parameter(s) will be learned. Our selection spectrum ranges from training a unique version of them, alongside the rest of the Network parameters, during the train

stage and keeping them frozen during the inference stage, up to adding an appropriate extra fully connected regression layer on top of the first linear layer of our tracker. This layer will be responsible for regression the rotational symmetry parameter(s) per pose pair, in order to provide the best symmetry matrix possible. An intermediate approach is learning a batch of the continuously rotational symmetry parameters (per rotational component) and then either take their mean as the proposed estimate or select the most suitable for the pose pair at hand using a classification layer on top of the first fully connected layer of the tracker. That last approach is the one we ultimately propose. With one extra notable observation: the classification layer must have a wide range of selections to make per pose pair in order to exploit its efficiency to the fullest. To this end, we should force the parameters of each component batch to be as widespread (and, thus, uniform) as possible, something that must be incorporated in the learning procedure of the network.

As a result, for objects with continuously rotational symmetry the loss becomes:

$$Loss^{(Symm)} = Loss + e^{(-s_4)} \left(\frac{1}{B} \sum_{b=1}^B \frac{1}{\xi_b} \right) + s_4, \text{ with} \quad (6.37)$$

$$\xi_b = \frac{1}{B_2(B_2 - 1)} \sum_{j=1}^{B_2} \sum_{k \neq j} d_{Rot}^{(Geod)}(\hat{G}_k, \hat{G}_j), \quad (6.38)$$

The extra term added to the multi-task loss is a penalty that adversarially trains the classification layer(s) that select(s) the proper rotational symmetry parameter(s) at each timestep. It encourages the geodesic rotational distances between all pairs of parameters in the batch to be maximum and, thus, ultimately converge to as a uniform distribution as possible. Here, we train $B_2=64$ such parameters for each continuous rotational component. All symmetry parameters are initialized via randomly sampling from a uniform distribution $U[-180^\circ, 180^\circ]$.

Discrete Rotational Symmetries:

As we have already mentioned, for the second symmetry case: that of **Discrete/Viewpoint-induced object symmetries**, we articulate the following thought process that leads to a heuristic algorithm.

The careful reader would, reasonably, formulate their following objection: Why we need to manually solve the discrepancy induced by this discrete symmetry family? Is not the Feedback loop itself adequate enough for giving a strong prior to the network and guide it to choose the right orientation? Besides, rotational differences between the two estimates are extraordinary and they would be one of the first things we would anticipate of the learning algorithm to correct.

This concern is reasonable, indeed. However, reality belies this trajectory of thought potentially expressed by the reader, as we can clearly see in the (left) following figure. One the one side, we see the erroneous pose estimation provided by the CNN itself, without manual correction. In one, particular frame, the estimation shifts by 180 degrees and adds a disproportional error to the measured rotational metric that consists nothing more than a fallacy. In fact, due to the set patience we have already coerced our algorithm into, before deciding to re-initialize the tracker, this error is possible to quickly accumulate and add error that corresponds to the estimates of the following frames, as well. To this end, we follow the next, very simple, algorithm.



Figure 6.34 Our Rotational Estimation for the 126th frame of the “Hard Interaction Scenario”.

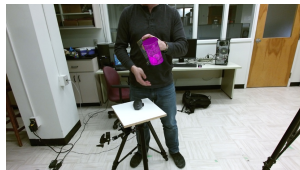


Figure 6.35 Our Rotational Estimation for the 127th frame of the “Hard Interaction Scenario” **without the proposed algorithm**.

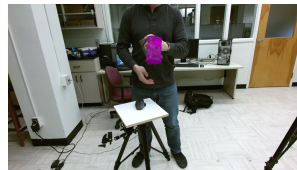


Figure 6.36 Our Rotational Estimation for the 126th frame of the “Hard Interaction Scenario”.

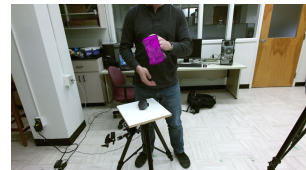


Figure 6.37 Our Rotational Estimation for the 127th frame of the “Hard Interaction Scenario” **with the proposed algorithm**.

This algorithm is exclusively based on our observation that pose estimates of the current frame are close to that of the previous one, or at least much closer than the error induced by a discrete symmetry wrong estimate. So, we partition the 360 degrees of the rotational component space, for each component, by the number of discrete/viewpoint-induced symmetries of the object of interest. At every step, after the CNN outputs its prediction, we check if the rotational distance between the current and the previous rotational estimation $d_A^o(\hat{r}_i(t), \hat{r}_i(t-1))$:

$$d_A^o(\hat{r}_i(t), \hat{r}_i(t-1)) = 180^\circ - (180^\circ - |\hat{r}_i - r_i^{(GT)}|) \% 360^\circ, \quad i = a, b, c. \quad (6.39)$$

exceeds this quotient, minus a safety threshold. If it does, then we assign as a current rotation estimation, the estimation of the previous frame, that we have kept. For example, for the ‘‘Cokiejar’’ model, we have 2 discrete symmetries (at the two lits of the the Cookie Jar) and we set $th = 80^\circ$.

Algorithm 2: Discrete/Viewpoint-induced Rotational Symmetries discrepancy resolution

```

for every Rotation estimation  $\hat{R}(t)$  do
   $\hat{r}_a(t), \hat{r}_b(t), \hat{r}_c(t) = \text{mat2eulerXYZ}(\hat{R}(t))$  (in degrees)
  for every euler angle  $r_i, i=a,b,c$ : do
    if  $d_A^o(\hat{r}_i(t), \hat{r}_i(t-1)) \geq \left[ \frac{360^\circ}{N_{DiscrSymm}} - th \right]$  then
       $\hat{r}_i(t-1) \rightarrow \hat{r}'_i(t)$ 
       $\hat{R}(t) = \text{eulerXYZ2mat}(\hat{r}'_a(t), \hat{r}'_b(t), \hat{r}'_c(t))$ 
    else
       $\hat{R}(t)$  is a valid estimation
    end
  end
end

```

We note that, in the future, we aim to include this algorithm in the learnable architecture and have rotational classification estimation provided by the same forward pass of the Network, as happens with Continuous Rotational Symmetries, as well. In this way, we will not be in need of knowing the details of those rotations at test time (i.e. their number, distribution and axis), something that will facilitate a possible passage to an object-agnostic approach.

6.6.9 Initialization Loss

Since the Geodesic distance suffers from multiple local minima, following [89], we first warm-up the weights, aiming to minimize the LogCosh loss function for 25 epochs.

$$L_{Track,Init}((\Delta\hat{\mathbb{P}}, \Delta\mathbb{P}_{GT})) = L_{Track,Init}((\Delta\hat{\mathbf{p}}, \Delta\mathbf{p}_{GT})) = \log(\cosh(\Delta\hat{\mathbf{p}} - \Delta\mathbf{p}_{GT})) \quad (6.40)$$

Then, we train until convergence, minimizing the loss (6.35).

6.7 Implementation Details

We start from 96 convolutional filters in the first layer pair and we double them in every consecutive convolutional (2D Convolutional or Fire) module of the main Network stream. At every Fire layer, we squeeze all input channels to half and then we excite them back to the input size via each of the two excitation paths. As for the convolutional modules dedicated to Foreground/Occlusion handling, we assign 96 convolutional filters to each one and then we squeeze them to a single map via a corresponding 1×1 convolutional layer (Section 3.3.5).

We use ELU activation functions, a minibatch size of 128, Dropout with probability 0.3, an Adam optimizer with corrected weight decay (Section 3.3.21) by a factor $1e-5$, learning rate $1e-3$ and a cosine annealing-based scheduler with warm restarts [79] every 10 epochs. We, also, clip the gradients of the output layer to a norm threshold of 1. The average training time is 12 hours in a single GeForce 1080 Ti GPU.

Validation procedure:

One insightful look in our method would immediately reveal an obvious discrepancy: **The training loss weights are optimized along with the Networks parameters. As a result, we are unable to use the same Tracking Loss for the validation stage of our training, since it will be reduced, overall, independently from the optimizer’s convergence.** Besides, a different validation loss would be crucial in our effort to compare our method with the State-of-the-Art of Garon et al. [83] and it would aid in avoiding overfitting to the test metrics (a discussion that we are about to have in a following Section). To this end, in the validation stage, we employ the MSE loss function, after flattening our 9 predicted parameters. The MSE’s drawback of unfair error weighting is known, but it will be insignificant here, as this validation loss is only used to decide if we have to stop training or not. In other words, its values compete only with its total older values and there is no comparison between the individual pose error components.

Spatial Attention Visualization:

To begin with, the Attention weight maps are bilinearly interpolated to the original input frame dimensions. Then, it is transformed to a jet heatmap. Following, we convolute this heatmap with a Gaussian kernel in order to allow the map to demonstrate relative differences in the various spatial dimensions of the Attention. Lastly, we interpose the heatmap on the original “Observed” RGB frame using a weighted scheme of 50% – 50%.

6.8 Testing Metrics

In our work, we use the Metrics introduced in [83], which unify the components of movement.

- The Translation Error is simply defined as the L2 norm between the two translation vectors:

$$\delta_t(\Delta\hat{\mathbf{t}}, \Delta\mathbf{t}_{GT}) = \|\Delta\hat{\mathbf{t}} - \Delta\mathbf{t}_{GT}\|_2 \quad (6.41)$$

- The Distance between two Rotation Matrices is computed using:

$$\delta_R(\Delta\hat{R}, \Delta R_{GT}) = \arccos\left(\frac{\text{tr}(\Delta\hat{R}^T \cdot \Delta R_{GT}) - 1}{2}\right), \quad (6.42)$$

where $\text{tr}(\cdot)$ denotes the matrix trace.

Alternatively, testing metrics that have been introduced in the literature, are:

- **ADD:**

$$ADD = \frac{1}{m} \sum_{\mathbf{x} \in \mathcal{M}} \left\| (\Delta R\mathbf{x} + \Delta\mathbf{t}) - (\Delta\hat{R}\mathbf{x} + \Delta\hat{\mathbf{t}}) \right\|_2 \quad (6.43)$$

where \mathcal{M} denotes the set of 3D model points and m is the number of points. The 6D pose is considered to be correct if the average distance is smaller than a predefined threshold.

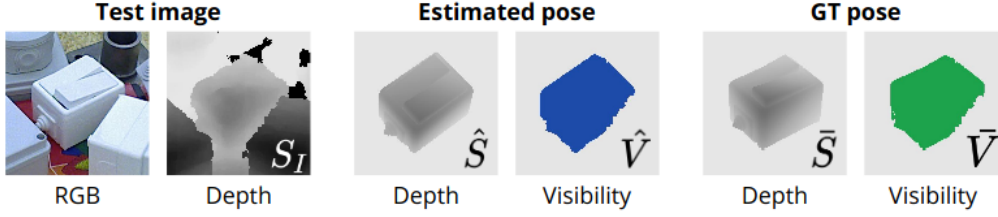
As for symmetric objects:

$$ADD_S = \frac{1}{m} \sum_{\mathbf{x}_1 \in \mathcal{M}} \min_{\mathbf{x}_2 \in \mathcal{M}} \left\| (\Delta R\mathbf{x}_1 + \Delta\mathbf{t}) - (\Delta\hat{R}\mathbf{x}_2 + \Delta\hat{\mathbf{t}}) \right\|_2 \quad (6.44)$$

Our design of the loss function for rotation regression is motivated by these two evaluation metrics. Using a fixed threshold in computing pose accuracy cannot reveal how a method performs on these incorrect poses with respect to that threshold. Therefore, we vary the distance threshold in evaluation. In this case, we can plot an accuracy-threshold curve, and compute the area under the curve for pose evaluation. Instead of computing distances in the 3D space, we can project the transformed points onto the image, and then compute the pairwise distances in the image space. This metric is called the reprojection error that is widely used for 6D pose estimation when only color images are used.

This metric is suboptimal as it generates more computations than those absolutely necessary, since the object is rigid, by adding noise to measurements, as well. We claim so, since all 3D object points, should be sampled for measuring pose errors.

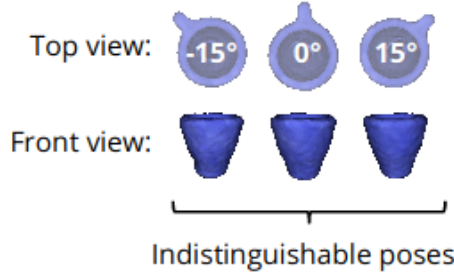
- **Pose component MSE [29]:** In [29], Garon et al. calculate the pose error as the average of each axis component in translation and rotation separately. The drawback of this metric is that **a large error on a single axial component is less penalized than in a unified translation and rotation ones.**
- **Visible Surface Discrepancy (VSD) [46]:** A pixel-based error metric, proposed by Hodan et al. [46], is the VSD metric which is founded on the visibility masks obtained by the Observed, Predicted and Ground Truth depth maps.



Visible Masks are obtained by comparing \hat{S} and \tilde{S} with S_I and the Error Metric is:

$$Err_{VSD}(\hat{S}, \tilde{S}, S_I, \hat{V}, \tilde{V}, \tau) = avg_{p \in (\hat{V} \cup \tilde{V})} \begin{cases} 0, & \text{if } p \in \hat{V} \cap \tilde{V} \wedge |\hat{S}(p) - \tilde{S}(p)| < \tau \\ 1, & \text{otherwise.} \end{cases} \quad (6.45)$$

Intuitively, it penalizes only object-pixels that are not included inside the object's rendered silhouette, at the intersection of the Predicted and Ground Truth image planes. This pose error metric is an image-based one, i.e. it is calculated over the visible part of the object, and especially, solely its 3D shape projection. This has the negative side effect that indistinguishable poses of symmetric objects are handled equivalently. Furthermore, it ignores the pose disentanglement that color information provides, even that is not complete.



- **Maximum Symmetry-Aware Surface Distance (MMSD) [46]:** Another candidate that takes into account the object's symmetries, this times is MMSD:

$$Err_{MMSD}(\hat{P}, P_{GT}) = \min_{\mathcal{G} \in \mathcal{G}} \left\{ \max_{\mathbf{x} \in \mathcal{M}} \left\{ \left\| \hat{P}\mathbf{x} - P_{GT}\mathcal{G}\mathbf{x} \right\|_2 \right\} \right\} \quad (6.46)$$

Due to the max operation in its formula, it is less dependent on sampling of the model surface, than its ADD_S counterpart. Symmetric and asymmetric objects treated in the same way. However, its drawback is that only pose ambiguities induced by the global object symmetries are considered, not pose ambiguities induced by occlusion/self-occlusion.

- **Maximum Symmetry-Aware Projection Distance (MSPD) [46]:**

$$Err_{MSPD}(\hat{P}, P_{GT}) = \min_{\mathcal{G} \in \mathcal{G}} \left\{ \max_{\mathbf{x} \in \mathcal{M}} \left\{ \left\| proj(\hat{P}\mathbf{x}) - proj(P_{GT}\mathcal{G}\mathbf{x}) \right\|_2 \right\} \right\} \quad (6.47)$$

MSPD, here, is less dependent on sampling of the model surface is dominated by finer parts. It is more suitable for AR applications and evaluation of RGB-only methods. Only pose ambiguities induced by the global object symmetries are considered, not pose ambiguities induced by occlusion/self-occlusion.

In the recent, “6D Pose Estimation” International Challenge, the performance w.r.t. each pose error function (VSD, MSSD or MSPD) is measured by the Average Recall (AR), i.e. the average of the recall rates calculated for multiple threshold settings.

$$Err_{Total}^{(AR)}(\hat{P}, P_{GT}) = \frac{Err_{VSD}(\hat{P}, P_{GT}) + Err_{MSSD}(\hat{P}, P_{GT}) + Err_{MSPD}(\hat{P}, P_{GT})}{3} \quad (6.48)$$

However, since both an accurate object 3D model and its refined ground truth pose labels are available, we see no reason to assess our method using an image-based metric. Besides, since the object is rigid and one 3D point (its center) is adequate for recovering its full pose, the pose metric pair we utilize in this work is more efficient in terms of calculations as well.

Chapter 7

Ablation Study

Good judgement is the result of
experience and experience the result
of bad judgement.

Mark Twain

In this chapter, we will explain the thought process that lead us to the architecture that was described in Chapter 6 and we will present quantitative and qualitative results of its variations. Specifically, we started from the simple baseline architecture of [83] and we experimented with its characteristics, making the assumption that each component type choice is independent from the rest (for example using an L2 or L1 loss function on the pose parameters has little to do with employing Dense or Residual connections between the layers of the architecture). This initial study allowed us to gain a direction towards which component (e.g. information source type, connection type, loss function choice etc.) changes were to be the most beneficial for the Pose Tracking problem, as well as understanding the designing choices that the authors of [29],[83] have made in their works. Later, the optimal strategies arisen from this ablation study were delicately combined to build our proposed scheme. In the following chapter, Chapter 8, we will compare the pros and cons of our proposed method in comparison to the State-of-the-Art [83] in terms of pose errors, tracking failures and inference speed.

Here, we have to notice that all of our comparisons will be made using the 3D Translational and Rotational error metrics presented in Section 6.8. Frequently in literature, those two metrics have shown to follow the same trends, however that is not necessary. That is why we chose to combine them in a joint **Normalised Pose Error (N.P.E.)** metric.

Mathematically, it is described as follows:

$$N.P.E = \frac{\delta_{\mathbf{t}}(\hat{\mathbf{t}}, \mathbf{t})}{\sqrt{3 \cdot MaxTranslRange(mm)^2}} + \frac{\delta_{\mathbf{R}}(\hat{R}, R)}{\sqrt{3 \cdot MaxRotRange(o)^2}}. \quad (7.1)$$

Using the **N.P.E.** formula, the mean and standard deviations of the 3D Translation and Rotation errors will be combined to mean and standard deviations of the N.P.E.:

- $$mean(N.P.E.) = \frac{mean(\delta_{\mathbf{t}}(\mathbf{t}_1, \mathbf{t}_2))}{\sqrt{3 \cdot MaxTranslRange(mm)^2}} + \frac{mean(\delta_{\mathbf{R}}(\mathbf{R}_1, \mathbf{R}_2))}{\sqrt{3 \cdot MaxRotRange(o)^2}} \quad (7.2)$$

- $$std(N.P.E.) = \frac{std(\delta_{\mathbf{t}}(\mathbf{t}_1, \mathbf{t}_2))}{\sqrt{3 \cdot MaxTranslRange(mm)^2}} + \frac{std(\delta_{\mathbf{R}}(R_1, R_2))}{\sqrt{3 \cdot MaxRotRange(o)^2}} \quad (7.3)$$

The design choices with the lowest Normalised Pose Errors (N.P.E.), were selected as the optimal ones and were finally intergrated in the proposed full architectural scheme.

In the following Ablation Study, we will see how independent architectural choice changes affect the tracker’s performance. Those that benefit us the most are finally blended in the proposed architecture.

7.1 State-of-the-Art Architecture - Baseline Results

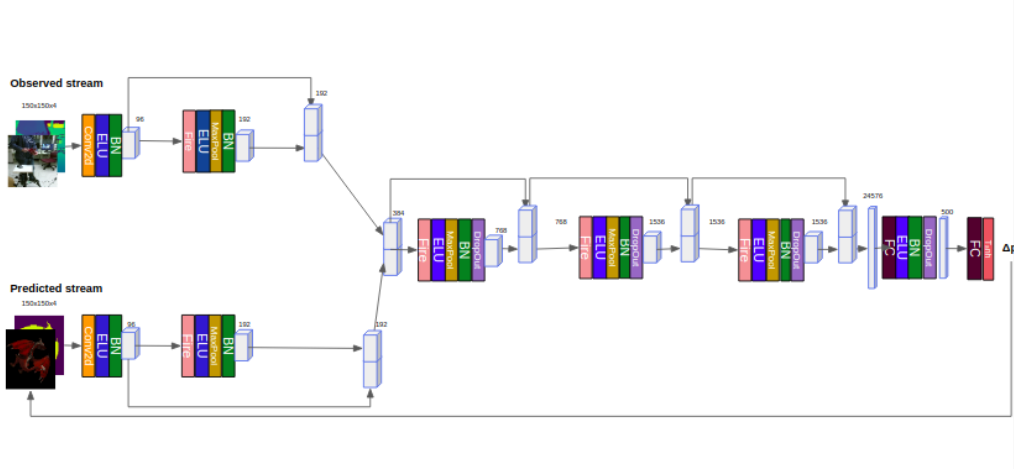


Figure 7.1: An overview of the State-of-the-Art CNN architecture proposed in [83].

We consider the model presented in [83] to be our baseline and we describe and justify each change performed to it. In [83], where the model is trained on a dataset of 200,000 RGB-D pairs, they do not report the exact translational and rotational errors in the “Hard Interaction scenario” for every object and scenario, rather than their mean values. To this end, we reproduced their approach and we report both the errors and the tracking failures, by training only on 10% of it (20,000 pairs). We did so because of limitation in computational resources, but the Pose Space was still covered sufficiently enough for the trends presented in the Ablation Study below to be similar as if we had trained each tracker on the full dataset of 200,000 samples.

Architecture	Translational Error (mm)	Rotational Error (o)
[83] (200k training pairs)	10.75 ± 2.16	18.9 ± 1.24
[83] (20k training pairs)	48.58 ± 38.23	35.43 ± 34.74

Table 7.1: The mean and the standard deviation of both the Translation and Rotation errors of the baseline tracker of [83], when it is trained on the full dataset (200,000 samples) stated in [83] and only on 10%, just for comparison reasons in the Ablation Study.

Architecture	Normalized pose error (mean \pm std)
[83] (200k training pairs)	1.40 ± 0.13
[83] (20k training pairs)	3.45 ± 3.11

Table 7.2: The Normalised Pose Errors and the Inverses of the Coefficient of Variation of both the Translation and Rotation errors of the baseline tracker of [83], when it is trained on the full dataset (200,000 samples) stated in [83] and only on 10%, just for comparison reason in the Ablation Study.

As we see in Tables 7.1,7.2, training on 10 times more data pairs improves the tracker’s accuracy, but much less than a proportional amount. For this phenomenon, we blame the fact that the synthetic training dataset contains instances that are heavily correlated (e.g. poses that are too close to be distinguished apart or training pairs which share almost the same “Observed” pose). So, we conclude that the reduced dataset of only 20,000 RGB-D pairs covers the Pose Space sufficiently enough for our study. We can, also, see this qualitatively, as both versions of the tracker lose the object at, approximately, the same points of the 3D trajectory, but in the second case the errors are larger. This

means that after the first 20,000 pairs, increasing the dataset size does not add previously heavily unknown poses, but rather refinements between pose differences, previously contained in it.

7.2 The necessity of the Feedback-Predicted Stream

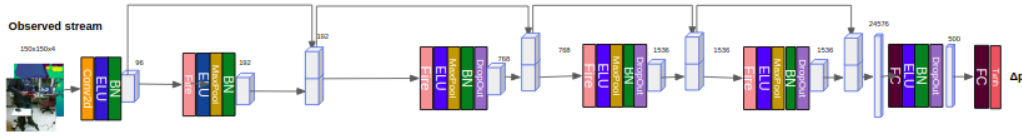


Figure 7.2: An overview of the State-of-the-Art CNN architecture proposed in [83] without the pose rendering-based Feedback from the previous estimation, both unavailable in the training and in the inference mode of the tracker.

One could justifiably wonder why we do not try to regress the 6D pose just from the current RGB-D video stream. That is because it would create increasing pose errors over time that would make the tracker fail irrecoverably, due to the lack of a feedback loop, as it has been stated numerous times before [29], as well as in it. Besides, most of the recent frame-by-frame 6D Pose Detection frameworks, described in Chapter 4 utilize a CNN module iteratively at the same timestep after their initial, coarse-grained estimate, for pose refinement. However, they are not suited for a real-time application, due to the presence of a preceding Deep preprocessing module (e.g. MaskRCNN [39] in some implementations etc.). So, in fact, we utilize for tracking a module similar to what they use just for the refinement process, but in the time-level instead. In case of a tracking failure, a Pose Detector (like those described in Chapter 4) is employed.

Architecture	Translational Error (mm)	Rotational Error (o)
[83] (20k training pairs)	48.58 ± 38.23	35.43 ± 34.74
[83] Without Feedback Predicted Stream (20k training pairs)	86.76 ± 51.42	52.43 ± 40.04

Table 7.3: The mean and the standard deviation of both the Translation and Rotation errors of the baseline tracker of [83], trained only on 10% of its overall dataset and of a modified version of it without an available pose feedback, in order to indicate the values that this feedback brings to stabilizing the tracker’s predictions.

Architecture	Normalized pose error (mean \pm std)
[83] (20k training pairs)	3.45 ± 3.11
[83] Without Predicted stream feedback (20k training pairs)	5.53 ± 3.79

Table 7.4: The means and the standard deviations of the Normalised Pose Errors of both the Translation and Rotation errors of the baseline tracker of [83], trained only on 10% of its overall dataset and of a modified version of it without an available pose feedback, in order to indicate the values that this feedback brings to stabilizing the tracker’s predictions.

We see in Tables 7.3,7.4, that this guess is confirmed as the absence of a feedback rendered pair leaves the single stream model unstable as it can never really track the object straight-forward. The only way to do so, in this framework, is through a pose comparison between the two RGB-D frame pairs: the real and the rendered one.

7.3 Translation-Rotation range exploration

In this section, we search for the optimal Translation and Rotation ranges of the standard deviation of a zero-meaned Gaussian Distribution from which the object pose parameters will be sampled. This

study was performed in [83], but we revise it explicitly for the “Hard interaction” scenario. We started from the range pair that [83] proposes: $30mm/15^\circ$ and we tried out a **Small** range pair: $10mm/5^\circ$, a **Medium**: $20mm/10^\circ$ and a **Big** one: $50mm/45^\circ$.

The first thing we observe from the Tables 7.5,7.6 is that the **Big** range scenario is highly unsuitable for sampling a dataset of this magnitude. That is caused by three reasons: firstly, this range does not model the testing dataset pose distribution accurately enough. The second one is that by allowing the Observed and the Predicted poses to be sampled so far one from the other, we create a sparser synthetic training dataset that with our predefined number of samples we cannot cover the pose space well enough. The last one is that if we sample a pose Transformation of such a big range, the bounding box calculated by the Predicted frame pair will leave an important part of the object outside of the Observed crop.

As for the other two ranges, the “Medium” range pair arises as the optimal selection.

We need to note here that the **N.P.E.** metric that we use to compare the various architectural choices uses the Translation and Rotation ranges in its denominators. So, from now on, we set that the that pair would be: $20mm/10^\circ$, and will be used across all the following comparisons.

Architecture	Translational Error (mm)	Rotational Error (o)
[83] (20k training pairs) (<i>Translation Range:30mm & Rotation Range:15°</i>)	50.27 ± 43.36	37.45 ± 40.28
[83] (20k training pairs) Small Range (<i>Translation Range:20mm & Rotation Range:5°</i>)	55.67 ± 31.60	32.97 ± 31.44
[83] (20k training pairs) Medium Range (<i>Translation Range:20mm & Rotation Range:10°</i>)	48.58 ± 38.23	35.43 ± 34.74
[83] (20k training pairs) Big Range (<i>Translation Range:50mm & Rotation Range:45°</i>)	93.30 ± 57.53	93.60 ± 56.19

Table 7.5: The mean and the standard deviations of both the Translation and Rotation errors of the baseline tracker of [83], trained only on 10% of its overall dataset, with the samples of this dataset taken from a distribution with ranges $30mm/15^\circ$ and the modified versions where the samples are taken from mean-centered Gaussian distributions with **Big,Medium** and **Small** ranges.

Architecture	Normalized pose error (mean \pm std)
[83] (20k training pairs) (<i>Translation Range:30mm & Rotation Range:15°</i>)	3.61 ± 3.59
[83] (20k training pairs) Small Range (<i>Translation Range:10mm & Rotation Range:5°</i>)	3.52 ± 2.73
[83] (20k training pairs) Medium Range (<i>Translation Range:20mm & Rotation Range:10°</i>)	3.45 ± 3.11
[83] (20k training pairs) Big Range (<i>Translation Range:30mm & Rotation Range:45°</i>)	5.93 ± 4.90

Table 7.6: The means and the standard deviations of the Normalised Pose Errors of both the Translation and Rotation errors of the baseline tracker of [83], trained only on 10% of its overall dataset, with the samples of this dataset taken from a distribution with ranges $30mm/15^\circ$ and the modified versions where the samples are taken from mean-centered Gaussian distributions with **Big,Medium** and **Small** ranges.

7.4 Insufficiencies of Unimodal architectures (only RGB/Depth-based streams)

In this section, we experimentally examine the necessity of both the input modalities. As it has stated above, in Chapter 2, they both have their own distinct disadvantages, so achieving the same performance, employing only one of them would give us the chance to get rid of the cons of the other. That is most important in the **solely-RGB** case, where we could employ our architecture on mobile devices that do not provide online Depth maps, as well and not be limited by the need for a Kinect sensor.

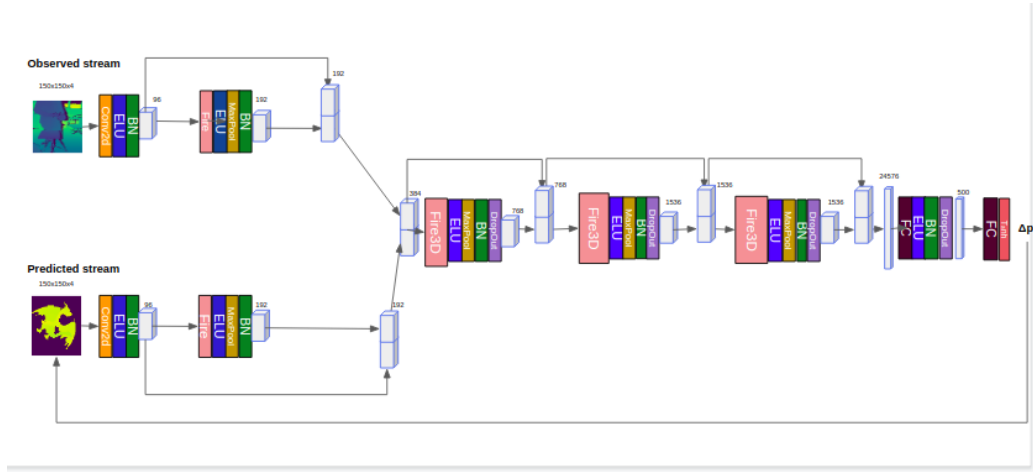


Figure 7.3: An Overview of a modification of the architecture of [83], with the use only of the 3D shape modality, as described by the the “Predicted” and “Observed” Depth maps.

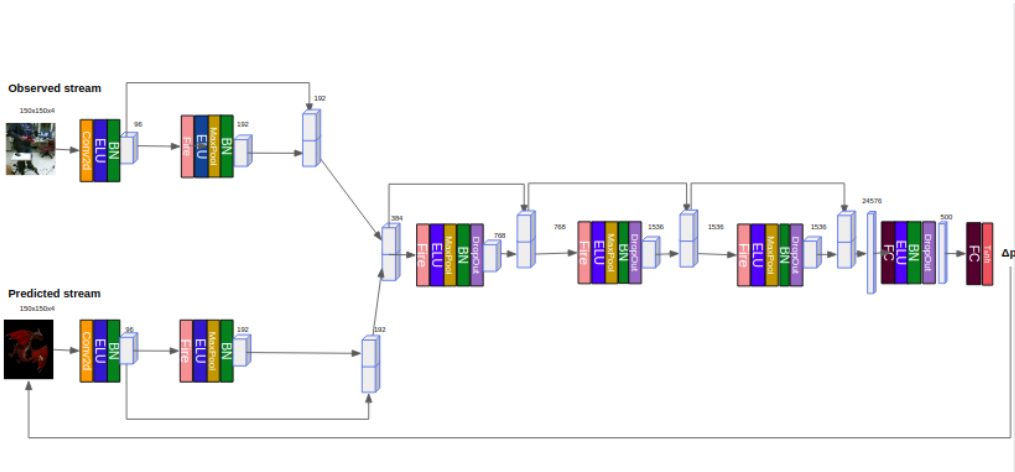


Figure 7.4: An Overview of a modification of the architecture of [83], with the use only of the appearance modality, as described by the the “Predicted” and “Observed” RGB images.

Architecture	Translational Error (mm)	Rotational Error (o)
[83] (20k training pairs)	48.58 ± 38.23	35.43 ± 34.74
[83] with only RGB information source (20k training pairs)	61.45 ± 54.06	40.24 ± 35.74
[83] with only Depth information source (20k training pairs)	53.68 ± 40.89	37.33 ± 36.47

Table 7.7: The mean and the standard deviations both of the Translational and Rotational errors of the architecture of [83], trained only on 10% of the overall dataset, and the two modifications of it, each using only one of the two input modalities:RGB and Depth.

Architecture	Normalized pose error (mean ± std)
[83] (20k training pairs)	3.45 ± 3.11
[83] with only RGB information source (20k training pairs)	4.10 ± 3.62
[83] with only Depth information source (20k training pairs)	3.70 ± 3.29

Table 7.8: The Normalised Pose Errors both of the Translational and Rotational errors of the architecture of [83], trained only on 10% of the overall dataset, and the two modifications of it, each using only one of the two input modalities:RGB and Depth.

However, Tables 7.7,7.8 indicate that this is not the case. We see that the Depth maps that describe the 3D object shape have the principal effect on the tracker’s performance. The noisy and ambiguous (by nature) 3D information source prevails to the high resolution 2D appearance-based one for a 3D task, such as Pose Tracking. This means that, even for a highly-detailed task, after accurate pose labelling, meticulous modeling of the lighting and shading parameters, and extensive relational data augmentation (e.g. the object being in front of a background and behind an occluder (to some extent)) 2D-3D unprojection reasoning is harder for the Network to estimate than Pose difference reasoning between two noisy 3D sources. The observation above is logical, especially for **non-Symmetric** objects. That is because **Symmetric** objects’ 3D shape alone inserts an ambiguity around their symmetry ax(is/es) that translates into some extra jitter for the Pose prediction. When the appearance information is added into the mix, both Pose error metrics decrease, stating that both modalities contain useful complementary information. However, for the symmetric object case, some pose jitter remains. This has given us the motivation to model that ambiguity as a disentangled rotation representation \hat{G}_{Rot} that is right-multiplied with the weighted Rotation matrix ($R \cdot \Lambda_{G.S.}$) in our approach and trained in such a way that symmetry-based jitter would be minimized.

We are now persuaded that both streams and both modalities are essential for the tracker’s performance. We proceed to experiment with the rest of its parameters.

7.5 Weight Initialization Algorithm comparison

In this section, we will discuss which Weight Initialization strategy we should prefer in our implementation. We start from a uniformly random weight initialization proposed in [29],[83] and we try out its “Xavier” 3.3.10 and “Kaiming He” 3.3.10 variations, leaving the architecture details the same as in [83].

Architecture	Translational Error (mm)	Rotational Error (o)
[83] (20k training pairs)	48.58 ± 38.23	35.43 ± 34.74
[83] (20k training pairs) with Uniform Xavier Weight Initialization	45.67 ± 37.22	33.86 ± 34.21
[83] (20k training pairs) with Uniform K.He Weight Initialization	40.28 ± 35.76	32.28 ± 31.98
[83] (20k training pairs) with Uniform K.He Weight Initialization in Conv. layers with (R)ELU activation function and Xavier Initialization in Conv. layers with Tanh activation function	34.31 ± 31.58	32.28 ± 31.98

Table 7.9: The means and the standard deviations both of the Translational and Rotational errors of the architecture of [83], trained only on 10% of the overall dataset, and its strategic modifications, where the (Uniformly) random weight initialization is replaced by Xavier Uniform (Section 3.3.10) and Kaiming He (Section 3.3.10) Uniform initialization schemes, correspondingly.

Architecture	Normalised Pose Error (mean ± std)
[83] (20k training pairs) (Random Weight Initialization)	3.45 ± 3.11
[83] (20k training pairs) with Uniform Xavier Weight Initialization	3.22 ± 3.29
[83] (20k training pairs) with Uniform K.He Weight Initialization	3.03 ± 2.88
[83] (20k training pairs) with Uniform K.He Weight Initialization in Conv. layers with (R)ELU activation function and Xavier Initialization in Conv. layers with Tanh activation function	2.62 ± 2.54

Table 7.10: The means and the standard deviations of the Normalised Pose Errors both of the Translational and Rotational errors of the architecture of [83], trained only on 10% of the overall dataset, and its strategic modifications, where the (Uniformly) random weight initialization is replaced by Xavier Uniform (Section 3.3.10) and Kaiming He (Section 3.3.10) Uniform initialization schemes, correspondingly.

As the Tables 7.23,7.10 the combination of the “Kaiming He” and “Xavier-Glorot” weight initialization algorithms arises as the best choice. The first one is used for convolutional layers with an

asymmetrical activation function (i.e. (R)eLU) and the second one for its symmetric counterparts: (i.e. Tanh, igmoid etc.) That is logical because, as it is stated in the original paper [38], “K.He” initialization approach fits best to Networks employing “ReLU-like” activation functions, while the Xavier et al.[31] method is older than the development of ReLU. This is the case here, as [83] employs the ELU activation function (see. Section 3.3.4), which differs from ReLU in just reducing, instead of completely negating, the negative inputs’ impact on the next layer, but still works quite similarly with classical ReLU.

7.6 Data Augmentation improvements

After a careful examination of the Data Augmentation strategies employed in [29],[83], we soon realized that we had the ability to further enhance them. To this end, as described in Section 7.6, we changed the Kinect sensor noise model from a simple Gaussian distribution to the 3D distribution described in Section 6.3.1, we required 15% of the occlusion cases to be full occlusions where the object is completely hidden behind the occluder object model (e.g. the user’s hand) and we employed the extra illumination variational augmentation as described in Section 6.3.1. The Network architecture was kept the same as in Garon et al. [83]. These improvement proved to be beneficial for the tracker’s performance as they decreased the mean Translation error by 8% and the Rotation one by 10%. This improvement did not have a dominant impact in any specific case of testing tracking sequence, but is rather consistent throughout the video-clip, which means that the more meticulous crafting of our augmentation scheme that we employ here, helps bridging the domain gap between the synthetic training and the real-world test dataset. Although it seems not to be the principal reason for the improvement our overall approach provided over the State-of-the-Art, it consists one of the secondary, but nonetheless, one of the more important components.

Architecture	Translational Error (mm)	Rotational Error (o)
[83] (20k training pairs)	48.58 ± 38.23	35.43 ± 34.74
[83] with extra 15% Full Occlusion,Illumination variation and 3D modeling of Kinect sensor noise (20k training pairs)	44.69 ± 33.74	31.74 ± 30.55

Table 7.11: The mean and the standard deviations both of the Translational and Rotational errors of the architecture of [83], trained only on 10% of the overall dataset, and its modifications where the Data Augmentation policy proposed in [83] is further sophisticated to include the cases of Full occlusion, illumination variation and a realistic modelling of a 3D Kinect depth sensor noise.

Architecture	Normalised Pose Error (mean ± std)
[83] (20k training pairs)	3.45 ± 3.11
[83] with extra 15% Full Occlusion,Illumination variation and 3D modeling of Kinect sensor noise (20k training pairs)	3.12 ± 2.73

Table 7.12: The means and the standard deviations of the Normalised Pose Errors of the architecture of [83], trained only on 10% of the overall dataset, and its modifications where the Data Augmentation policy proposed in [83] is further sophisticated to include the cases of Full occlusion, illumination variation and a realistic modeling of a 3D Kinect depth sensor noise.

7.7 Frozen vs Unfrozen Transfer Learning ResNet layers

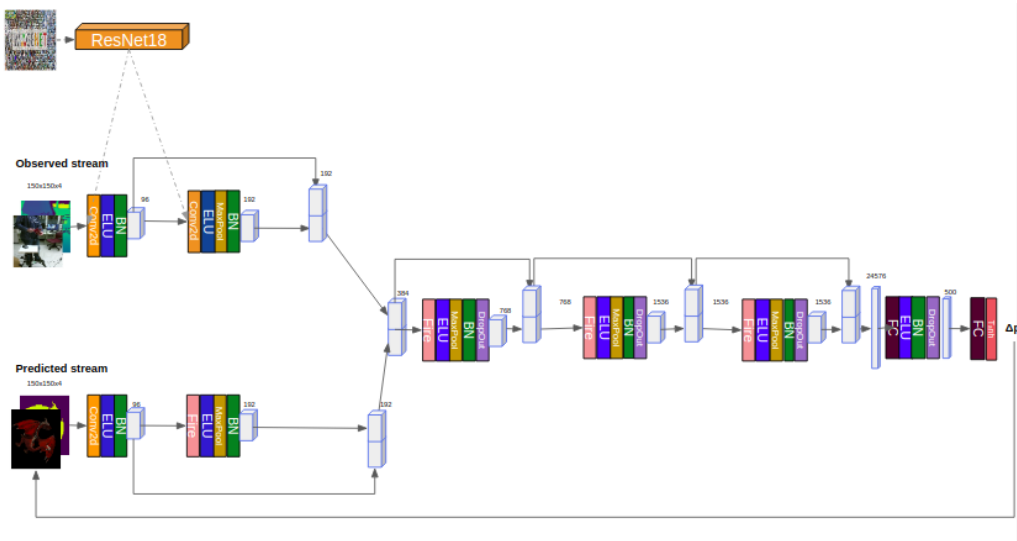


Figure 7.5: An overview of the architecture of [83], where the first two “Observed” layers are initialized with the corresponding weights of a ResNet18 pretrained on ImageNet.

Initializing some or the whole of the Network’s weights using an (ResNet18) architecture pretrained on Imagenet would be, de facto, beneficial for the tracker’s convergence speed and accuracy during training, as well as for its performance in the inference stage. Besides, it consists of the best known domain daptation strategy in literature. As we have already described in details in Section 6.3, the authors of [42] claim that if the “Observed” stream that is trained on synthetic and tested on real RGB-D frame pairs would be initialized by such a pretrained ResNet weights, we would bridge the domain gap. It is profound that the “Predicted” stream is not in the need of any kind of such an initialization in its case there is not a domain gap: it is both trained and tested on synthetically rendered data. However, in our experiments we tried such an initialization approach and then let the weights of this stream to finetune, along with the rest of the architecture hoping just to provide a “warmer” starting point. In [42], Hintentoisser et al. have experimentally shown that the optimal performance is achieved if we initialize the “Observed” stream with the pretrained ResNet18’s weights and keep it frozen during the Network training.

However, our experiments does not show this to be the case. In detail, we tried out all the combinations of initialization between the two streams. Those are: Weight Transferring and Freezing only for the “Observed” stream/Weight Transferring and Finetuning only for the “Observed” stream/Weight Transferring for both streams and Freezing only for the “Observed” stream/Weight Transferring and Finetuning for both streams. As one can easily spot in Tables 7.13,7.14, in contrary with what is proposed in [42], our best results (lowest N.P.E. metric) are given when we initialize only the “Observed” stream with the pretrained ResNet18’s weights and then let it finetune along with the rest of the architecture. We claim that this difference rises because of the following fact: in [42] they train their model on a variety of objects. So, freezing the stream’s weights in the “Multi-Object” case would give a two-fold end: bridging the real-synthetic domain gap and avoiding overfitting on the features of only one of the objects. However, in our scenario, we train and test each architecture on one, single object. As a result, overfitting to its features is a desirable outcome that increases the tracker’s performance along the whole testing video-clip, rather than a generalization disadvantage, as generalization across different objects is not one of our requirements. According to the Tables’ results, we assume that in a possible extension of this work, similarly to what was conducted in [83], following the method proposed in [42] (i.e. Transfer Learning to both streams and Freezing the “Observed” stream) would provide the desired generalization capabilities if the tracker would be trained on multiple objects and tested on previously unseen ones.

Architecture	Translational Error (mm)	Rotational Error (o)
[83] (20k training pairs)	48.58 ± 38.23	35.43 ± 34.74
[83] (20k training pairs) (<i>Obs-Stream Frozen ResNet (1D → 3D) ImageNet initialization</i>)	81.07 ± 58.30	40.54 ± 41.76
[83] (20k training pairs) (<i>Obs-Stream Finetuned ResNet (1D → 3D) ImageNet initialization</i>)	38.78 ± 35.28	35.12 ± 33.64
(<i>Obs (Frozen) & Pred (Frozen) Streams ResNet (1D → 3D) ImageNet initialization</i>)	62.83 ± 42.27	37.66 ± 33.64
(<i>Obs (Frozen) & Pred Streams (Finetuned) ResNet (1D → 3D) ImageNet initialization</i>)	38.94 ± 38.15	54.82 ± 52.03

Table 7.13: The mean and the standard deviations both of the Translational and Rotational errors of the architecture of [83], trained only on 10% of the overall dataset, and its modifications where the first two “Observed” of both the two “Observed” and “Predicted” Convolutional layers are initialized using the corresponding weights of a ResNet18[37], pretrained on ImageNet [42].

Architecture	Normalised Pose Error (mean ± std)
[83] (20k training pairs)	3.44 ± 3.11
[83] (20k training pairs) (<i>Obs-Stream Frozen ResNet (1D → 3D) ImageNet initialization</i>)	4.68 ± 4.09
[83] (20k training pairs) (<i>Obs-Stream Finetuned ResNet (1D → 3D) ImageNet initialization</i>)	3.06 ± 2.96
(<i>Obs (Frozen) & Pred (Frozen) Streams ResNet (1D → 3D) ImageNet initialization</i>)	3.99 ± 3.25
(<i>Obs (Finetuned) & Pred (Frozen) Streams ResNet (1D → 3D) ImageNet initialization</i>)	4.29 ± 4.02

Table 7.14: The mean and the standard deviations of the Normalized Pose Errors of the architecture of [83], trained only on 10% of the overall dataset, and its modifications where the first two “Observed” of both the two “Observed” and “Predicted” Convolutional layers are initialized using the corresponding weights of a ResNet18[37], pretrained on ImageNet [42].

7.8 Activation function choice

In this section, we defy the choice of the Activation function that was made in [29] and [83] and we examine the alternatives’ impact on the tracker’s performance, by keeping the rest of the architecture’s details the same as in [83].

Architecture	Translational Error (mm)	Rotational Error (o)
[83] (20k training pairs) (<i>ELU</i>)	48.58 ± 38.23	35.43 ± 34.74
[83] (20k training pairs) (<i>Sigmoid</i>)	87.43 ± 53.71	42.78 ± 38.25
[83] (20k training pairs) (<i>Tanh</i>)	72.64 ± 44.22	40.16 ± 35.09
[83] (20k training pairs) (<i>ReLU</i>)	62.21 ± 47.7	37.43 ± 35.63
[83] (20k training pairs) (<i>Leaky Relu</i>)	52.60 ± 45.76	37.20 ± 33.00
[83] (20k training pairs) (<i>SeLU</i>)	50.78 ± 42.53	36.01 ± 34.00
[83] (20k training pairs) (<i>Swish</i>)	51.64 ± 40.27	37.04 ± 39.01

Table 7.15: A comparison between the mean and standard deviations of the translational and rotation errors among different selections of activation function. The experiments concern the main coprus of the architecture (meaning the exclusion of branches and output layers.) Promising candidates tested here are: ReLU, Leaky ReLU, PReLU, Tanh, Simgoid, Swift, SELU.

Architecture	Normalised Pose Error (mean \pm std)
[83] (20k training pairs) (<i>ELU</i>)	3.45 \pm 3.11
[83] (20k training pairs) (<i>Sigmoid</i>)	4.99 \pm 3.76
[83] (20k training pairs) (<i>Tanh</i>)	4.42 \pm 3.30
[83] (20k training pairs) (<i>ReLU</i>)	3.84 \pm 3.38
[83] (20k training pairs) (<i>Leaky Relu</i>)	3.67 \pm 3.23
[83] (20k training pairs) (<i>SeLU</i>)	3.61 \pm 3.23
[83] (20k training pairs) (<i>Swish</i>)	3.63 \pm 3.41

Table 7.16: A comparison between the mean and standard deviations of the Normalized Pose Errors among different selections of activation function. The experiments concern the main coprus of the architecture (meaning the exclusion of branches and output layers.) Promising candidates tested here are: ReLU, Leaky ReLU, PReLU, Tanh, Simgoid, Swift, SELU.

The results that shown in Tables 7.15,7.16 are what we expected. The ELU candidate is shown to be a clear winner as it combines all the advantages of the ReLU function, by mitigating its “dead neurons” side-effect, at the same time.

7.9 Losses

In this section, we defy the choice of the MSE parameter loss function that was made by Garon et al. in [83] and we examine other possible alternatives. In detail, we keep the same architecture of [83] and we try out the L1 Loss (see Section 3.3.17), the Log Cosh loss (see Section 3.3.17), which is a smooth intermediate alternative between the L1 and L2 losses and the adaptive Regression loss function (see Section 3.3.17) that was recently proposed by Barron in [4], initialized with a=1 and c=2.

Architecture	Translational Error (mm)	Rotational Error (o)
[83] (20k training pairs) <i>MSE Loss</i>	48.58 \pm 38.23	35.43 \pm 34.74
[83] (20k training pairs) <i>L1 Loss</i>	51.63 \pm 43.44	33.80 \pm 35.38
[83] (20k training pairs) <i>Log Cosh Loss</i> [89]	42.54 \pm 31.20	31.30 \pm 34.50
[83] (20k training pairs) <i>Barron Loss</i>	61.25 \pm 40.51	38.72 \pm 33.40

Table 7.17: The mean and the standard deviations both of the Translational and Rotational errors of the architecture of [83], trained only on 10% of the overall dataset, and its modifications where the MSE parameter loss is substituted by a L1/LogCosh/Adaptive Barron loss alternative.

Architecture	Normalised Pose Error (mean \pm std)
[83] (20k training pairs) <i>MSE Loss</i>	3.45 \pm 3.11
[83] (20k training pairs) <i>L1 Loss</i>	3.44 \pm 3.29
[83] (20k training pairs) <i>Log Cosh Loss</i> [89]	3.04 \pm 2.89
[83] (20k training pairs) <i>Barron Loss</i>	4.00 \pm 3.01

Table 7.18: The mean and the standard deviations of the Normalised Pose Error measurements for the architecture of [83], trained only on 10% of the overall dataset, and its modifications where the MSE parameter loss is substituted by a L1/LogCosh/Adaptive Barron loss alternative.

According to the Tables 7.17, 7.18, the results of this comparison are clear, however somewhat unexpected. Logically, Log Cosh rises as winner as it utilizes the best of both L1 and L2 losses and causes smaller pose errors. However, the fact that its use overcomes that of Barron Loss is not straightforward. We blame this difference to the following reason: while LogCosh has a standard shape and its only task is to find the optimal set of weights that minimize its value, Barron Loss has the extra burden of finding the most suitable Loss shape as well, which introduces some extra complexity. Since this is handled by an extra regularization loss [4], having to scale that tradeoff reduces the testing performance of the Network trained with the Barron Loss.

7.10 Dense vs Residual connections

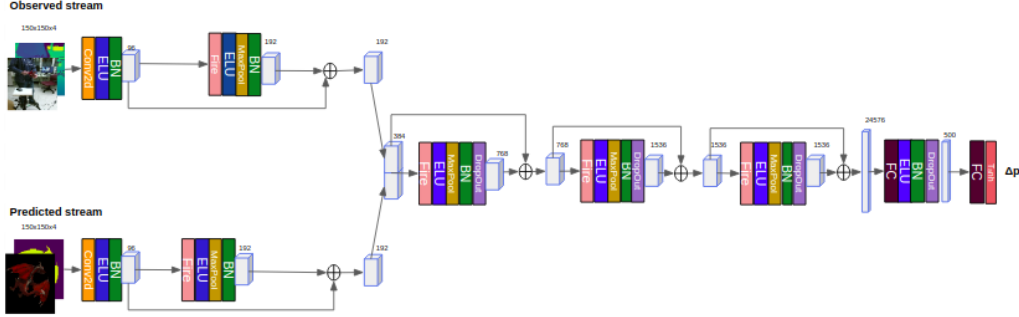


Figure 7.6: A modified version of the Network architecture proposed in [83] with the “Dense” cross-layer connections [49] replaced by Residual [37] connections.

In this section, we defy the choice of the Dense connections (see Section 3.3.12) that was made in the architectural design of [83] and we examine their possible substitution by the Residual ones of Section 6.6.2.

Architecture	Translational Error (mm)	Rotational Error (o)
[83] (20k training pairs) (<i>Dense Connections</i>)	48.58 ± 38.23	35.43 ± 34.74
[83] (20k training pairs) (<i>Residual Connections</i>)	10.58 ± 7.99	13.81 ± 18.22

Table 7.19: The 3D Translational and Rotational error metrics comparison between the architecture of [83] and its variation with the use of Residual connections.

Architecture	Normalised Pose Error (mean \pm std)
[83] (20k training pairs) (<i>Dense Connections</i>)	3.45 ± 3.11
[83] (20k training pairs) (<i>Residual Connections</i>)	1.11 ± 1.28

Table 7.20: The 3D Normalised Pose Error metric comparison between the architecture of [83] and its variation with the use of Residual connections.

The results presented in the Tables 7.19,7.20 are impressive! Just by replacing the Dense concatenation between the consecutive layers’ feature maps with Residual addition of them, the tracker’s Translation error plummets by 78% and the Rotation one by 61%. This major difference seems counter-intuitive at first as the Loss curves, when we use Dense connections, seem to be much more smooth and convex than the ones with Residual ones. However, this property does not assure a smaller global minimum than a local one owned by the Loss function landscape in the Residual connection case. At the pixel-level, now, watching the figures 3.29,3.34, we may provide a second explanation of this situation: We notice that the figures above were a product of the Imagenet classification task, where the focus was to extract a global assumption for the whole image, putting aside special spatial considerations. On the other hand, since the Residual connections perform pixel-level addition, they preserve and, more explicitly, connect, spatial properties throughout the network layers. This has a special effect for the Pose Tracking task, as it heavily relies on spatial pixel properties to make conclusions about the object and/or its position and orientation w.r.t. the background and/or any possible occluders.

Now that we have made up our minds as far as it concerns the inter-layer connectivity type, the best convex regression loss function and the proper initialization scheme, we visualize their gradual effect on

the training procedure, in order to give the reader an intuitive understanding.

Visualization of The effect of weight initialization and Residual connections

Next, we present 3 pairs of solely Train and Train/Validation loss visualizations.

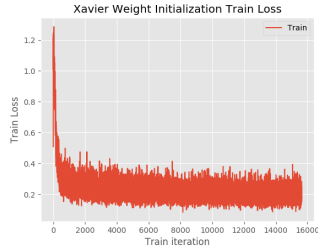


Figure 7.7 LogCosh train loss progress for the architecture of Garon et al. [83], with random weight initialization.

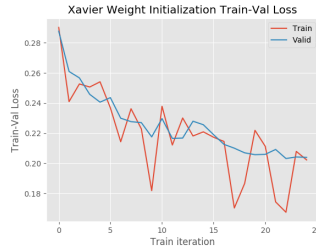


Figure 7.8 LogCosh train (in red) and validation (in blue) loss progress for the architecture of Garon et al. [83], with random weight initialization.

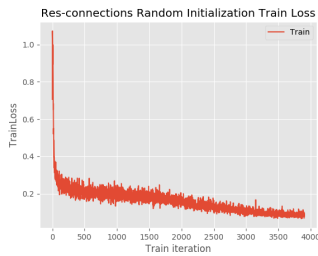


Figure 7.9 LogCosh train loss progress for the architecture of Garon et al. [83], with random weight initialization and Residual inter-layer connections.



Figure 7.10 LogCosh train (in red) and validation (in blue) loss progress for the architecture of Garon et al. [83], with random weight initialization and Residual inter-layer connections.

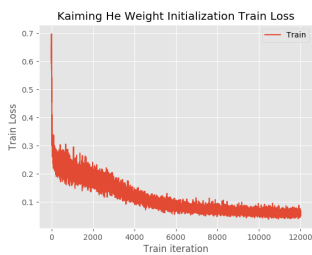


Figure 7.11 LogCosh train loss progress for the architecture of Garon et al. [83], with Residual inter-layer connections and “K.He-Xavier” proper combination weight initialization.

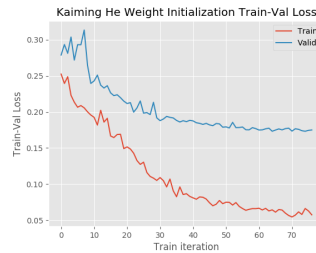


Figure 7.12 LogCosh train (in red) and validation (in blue) loss progress for the architecture of Garon et al. [83], with Residual inter-layer connections and “K.He-Xavier” proper combination weight initialization.

As it is obvious in the figures above, the Residual connections play a decisive role in smoothing out training and eliminating overfitting. Then, adding the optimal weight initialization combination of strategies makes the Network to converge faster, as it starts from a lower value and results in a deeper local validation minimum.

7.11 Stream Fusion choices

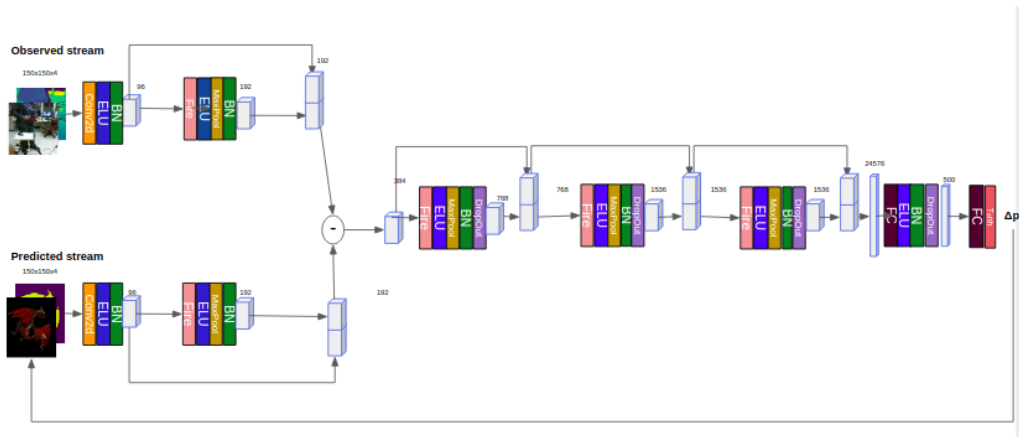


Figure 7.13: An alternative version of the architecture of Garon et al.[83], with the “Observed”-“Predicted” feature fusion been executed by subtraction of the corresponding feature maps. A similar attempt has been recorder in the DeepIM [73] architecture.

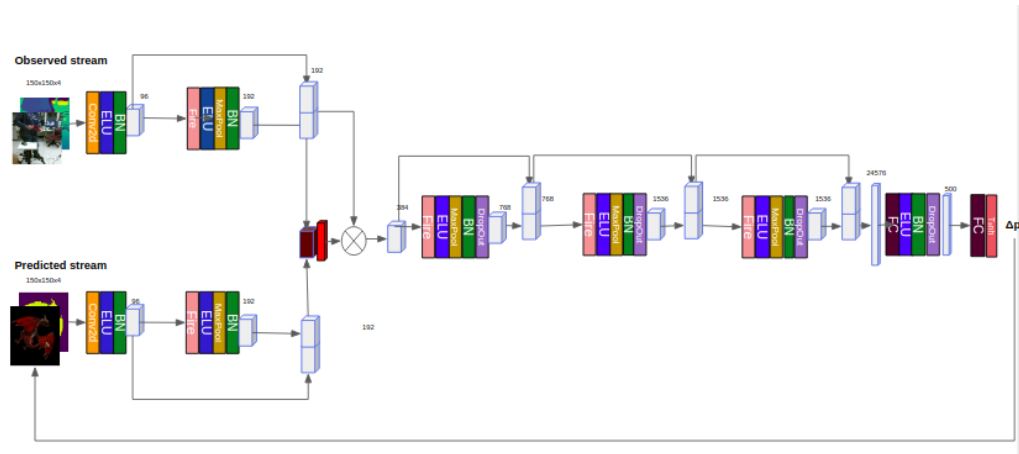


Figure 7.14: An alternative version of the architecture of Garon et al.[83], with the “Observed”-“Predicted” feature fusion been executed by taking the Cosine Similarity (Section 3.1.4) between the corresponding feature maps. Then, the resulted constraint 1D weight map is applied to the “Observed” feature map. Lastly, the outcome is enhanced by a Residual connection from the “Observed” feature map.

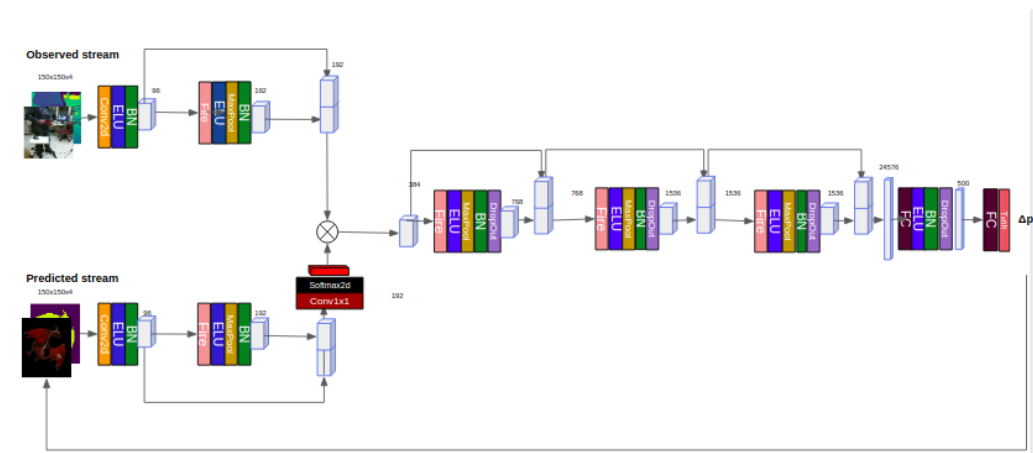


Figure 7.15: An alternative version of the architecture of Garon et al.[83], with the “Observed”-“Predicted” feature fusion been executed by applying an attentional weight map, extracted immediately from the “Predicted” stream, to the feature map of the “Observed” stream.

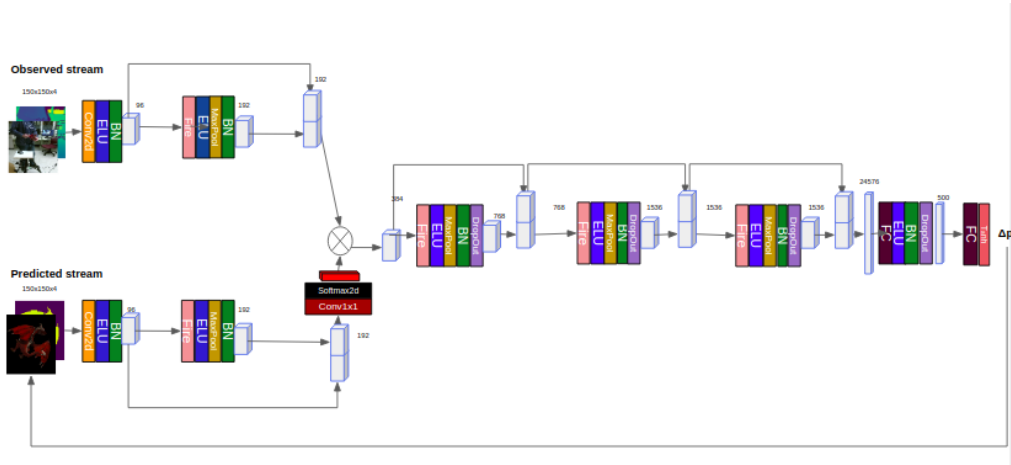


Figure 7.16: An alternative version of the architecture of Garon et al.[83], with the “Observed”-“Predicted” feature fusion been executed in “Late” fashion, i.e. at the output feature maps of the last convolutional layer.

In this section, we defy the Stream concatenation Fusion choice made in [83] and we experiment with a number of possible modifications of it.

- **Late Stream Concatenation Fusion:** we extend the depth of both the “Observed” and the “Predicted” network streams from two to five layers each and we concatenate their output feature maps at the end before passing the output of this concatenation to the sequence of the two Linear layers. See fig.7.16.
- **Stream Subtraction:** we substitute modeling the two Stream comparison using Early concatenation of their feature maps with subtracting the one produced by the “Predicted” Stream by the one produced by the “Observed” stream. It consists of a popular variation used in other works as well, such as [73]. See fig.7.13.
- **Spatial Attention based on the “Predicted” Stream and applied on the “Observed” Stream:** we substitute modeling the two Stream comparison using Early concatenation of their feature maps with the following Stream to Stream Spatial Attention mechanism: we pass the output feature map of the “Predicted Stream” through a 1×1 Convolution 3.3.5 followed by a 2D Spatial Softmax 3.3.4 activation function that outputs a single-channeled weight map. This weight map is multiplied element-wise with all the feature map channels of the output feature map of the “Observed” Stream. Finally, we enhance the weighted “Observed” feature map using a residual connection that adds it with its pre-weighted version. In this way, both the pixels that the “Predicted”-based Attention weight map highlights will have enhanced values and the effect of rest of the pixels of the “Observed”-based feature map to the following layers will not be deleted entirely (a similar strategy to what is employed in [75]). See fig.7.15.
- **Spatial Attention using “Obs”-“Pred” Cosine Similarity 3.1.4:** we substitute modeling the two Stream comparison using Early concatenation of their feature maps with the following Stream to Stream Spatial Attention mechanism, this time based on the Cosine Similarity function (Section 3.1.4) between the two streams’ outputs. Specifically, we take the Cosine Similarity 3.1.4 between the “Observed” and the “Predicted” feature maps and we produce a probability spatial weight map (with values between 0 and 1). We multiplicatively apply it channel-wise to the “Observed” feature map, getting a weighted version of it as output. Finally, we use a Residual connection that add to this weighted “Observed” feature map, its previous pre-weighted version. In this way, both the pixels that the “Predicted”-based Attention weight map highlights will have enhanced values and the effect of rest of the pixels of the “Observed”-based feature map to the following layers will not be deleted entirely (a similar strategy to what is employed in [75]). See fig.7.14.

Architecture	Translational Error (mm)	Rotational Error (o)
[83] (20k training pairs) (<i>Early Stream Concatenation Fusion</i>)	48.58 ± 38.23	35.43 ± 34.74
[83] (20k training pairs) (<i>Late Stream Concatenation Fusion</i>)	62.21 ± 35.90	37.43 ± 36.73
[83] (20k training pairs) (<i>Stream Subtraction</i>)	59.49 ± 46.18	40.92 ± 35.84
[83] (20k training pairs) (<i>Spatial Attention based on the “Predicted” Stream and applied on the “Observed” Stream</i>)	51.04 ± 39.82	37.89 ± 35.21
[83] (20k training pairs) (<i>Spatial Attention using “Obs-Pred” Cosine Similarity</i>)	49.22 ± 38.11	36.32 ± 35.45

Table 7.21: The 3D Translational and Rotational error metrics comparison between the architecture of [83] and its modified versions employing different Stream Fusion strategies.

Architecture	Normalised Pose Error (mean ± std)
[83] (20k training pairs) (<i>Early Stream Concatenation Fusion</i>)	3.45 ± 3.11
[83] (20k training pairs) (<i>Late Stream Concatenation Fusion</i>)	3.95 ± 3.16
[83] (20k training pairs) (<i>Stream Subtraction</i>)	3.79 ± 3.48
[83] (20k training pairs) (<i>Spatial Attention based on the “Predicted” Stream and applied on the “Observed” Stream</i>)	3.66 ± 3.18
[83] (20k training pairs) (<i>Spatial Attention using “Obs”-“Pred” Cosine Similarity</i>)	3.52 ± 3.15

Table 7.22: The Normalised Posed Error metric comparison between the architecture of [83] and its modified versions with different Stream Fusion strategies.

In the Tables 7.21,7.22, we see that the Stream Fusion choice that was made in [83] is verified as the best overall. However, if we see things from an intuitive perspective, both the Attention-based fusion strategies seem to model the Pose difference model better than the “comparison-via-concatenation” proposed in [83], as what they really show is the pixel-region in the “Observed” frame pair that should be highlighted, according to the depiction of the object Pose on the “Predicted” pair. So, the most interesting result that we need to interpret is why the Stream concatenation Fusion exceeds both of them, even slightly. We think that this is because both Attention Streams employ either a Softmax activation function or a Cosine Similarity function (Section 3.1.4). Indeed, the derivatives of both of them present a strong denominator value which severely diminishes the gradients passing to the first two layers of the architecture during the Backpropagation pass, resulting in worse Network training.

7.12 Clutter and Occlusion Handling Attention

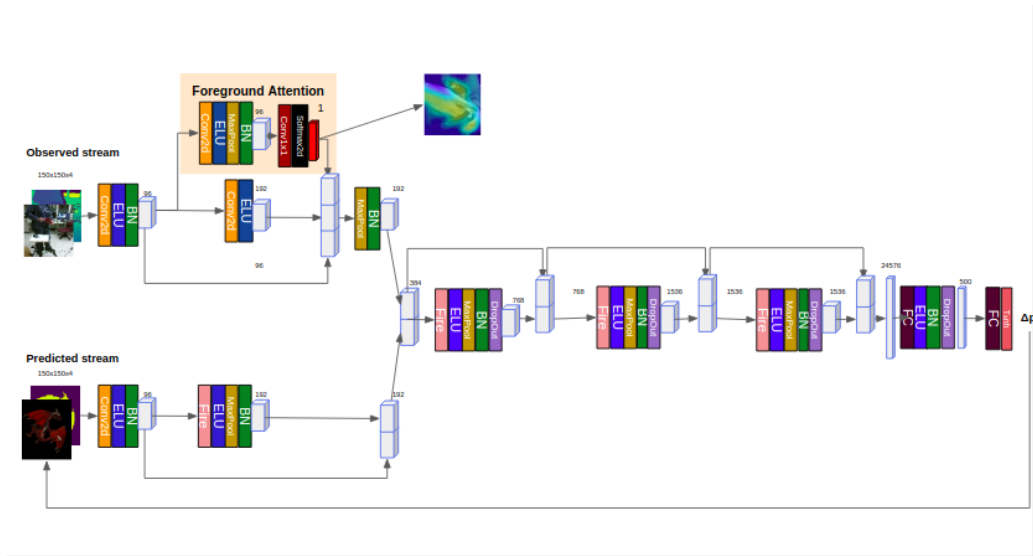


Figure 7.17: An overview of a modified version of the SoA CNN architecture of Garon et al. [83], where a Foreground pixel Attention Weight map is applied to the “Observed” feature map of the second corresponding layer, for handling Background Clutter.

7.12.1 Clutter Handling

In this section we examine, both qualitatively and quantitatively, the impact that different architectural design strategies for handling background clutter have on the tracker’s performance and we choose the best of them. Specifically, we search for the optimal way to highlight the pixels of the “Observed” feature map that correspond to the object-foreground with respect to those of the background. To accomplish so, we examine the following options and compare the results:

- **Soft Spatial Attention module for Foreground Extraction:** we take the feature map output of the first Convolutional layer of the “Observed” stream and we use it as input to a Soft Spatial Attention module consisted of: a 2D Convolutional and an 1×1 Convolutional (see Section 3.3.5) layer with 2D Spatial Softmax (Section 3.3.4) output. This Softmax activation outputs a single-channelled probability-weight Attention matrix. This specific Attention weight map is applied to the feature map output of the second “Observed” Fire (see Section 3.2.5) layer and it is added to a unitary mapping from an incoming residual connection between itself and the feature map of the previous layer.
- **Soft Spatial Attention module for Foreground Extraction using Binary Cross Entropy with steady weights:** We use the same Attention scheme as before. The difference, now, is that the Attention weight map is not let free to generate whatever patterns will emerge, but it is rather strictly supervised by adding a Binary Cross Entropy loss function (see Section 3.3.17) to the Track Loss (MSE) of [83]. The ground truth signal used to guide the Binary Cross Entropy loss function is a binary foreground-background map, kept during the Data Augmentation process.
- **Soft Spatial Attention module for Foreground Extraction using Binary Cross Entropy with learnable weights:** Here, we employ the same supervised Attention scheme as above. The difference, now, is that instead of just adding the Binary Cross Entropy Loss to the Track one, we leverage both using the learnable MultiTask weighting scheme, described in Section 8.2.

Next, we visualize the output of the three different Foreground Extraction Spatial Attention maps, superimposed on the same “Observed” RGB frame:

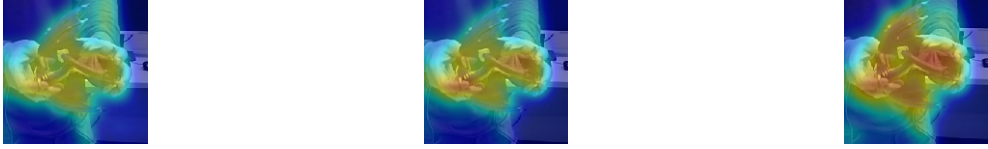


Figure 7.18: The Spatial Attention weight map of the unsupervised module. Its accuracy is remarkable, if you consider the fact of its complete lack of supervision. It’s qualitative differences are, indeed, existent, but mostly, this map presents brighter attention leakage from the object features to the user’s hand, in comparison to its supervised counterparts. Although this is not forbidden, its profoundly impedes the tracker from reaching optimal tracking accuracy and robust generalization to difficult pose configurations.

Figure 7.19: Here, in the Spatial Attention map that corresponds to the second approach (Binary Cross Entropy with steady inter-loss weighting), we start to see the benevolent effects of binary mask supervision. Note that this extra supervision is of no extra computational cost, as these masks are to be computed, anyway, during the augmentation stage. The only difference of this approach is that we keep them as a guiding signal, so, if someone would want to be fully diligent, we could only speak about extra memory, instead of computational cost (insignificant nonetheless). The effectiveness of our approach starts to show from the fact that peaks, here, are sharper and the attention leakages are significantly less, since the whole module’s focus is concentrated on the feature of the object of interest.

Figure 7.20: The optimal Spatial Attention map weights’ learning strategy: the two losses, the main tracking one and the auxiliary BCE for Attention supervision are inter-weighted via a learnable multi-task scheme. It is clear that peaks here are the most consistently concentrated on the object’s surface, they are the sharpest and give clear hints of the object’s keypoints that are the most interesting for tracking. This success is the aftermath of leaving the weights of different losses to be free to scale up or down in order to balance out the learning difficulties of loss hardness discrepancy.

Architecture	Translational Error (mm)	Rotational Error (o)
[83] (20k training pairs)	48.58 ± 38.23	35.43 ± 34.74
[83] (20k training pairs) (<i>Soft Spatial Attention Background Extraction</i>)	25.19 ± 17.87	38.19 ± 35.70
[83] (20k training pairs) (<i>Soft Spatial Attention Background Extraction using Binary Cross Entropy with steady weights</i>)	20.78 ± 15.39	36.42 ± 35.22
[83] (20k training pairs) (<i>Soft Spatial Attention Background Extraction using Binary Cross Entropy with learnable weights</i>)	15.75 ± 10.24	35.39 ± 34.52

Table 7.23: The 3D Translational and Rotational error metrics comparison between the architecture of [83] and its modified versions employing the various Foreground pixel Attention schemes described in this section.

Architecture	Normalised Pose Error (mean ± std)
[83] (20k training pairs)	3.44 ± 3.11
[83] (20k training pairs) (<i>Soft Spatial Attention Background Extraction</i>)	2.93 ± 2.58
[83] (20k training pairs) (<i>Soft Spatial Attention Background Extraction using Binary Cross Entropy with steady weights</i>)	2.70 ± 2.48
[83] (20k training pairs) (<i>Soft Spatial Attention Background Extraction using Binary Cross Entropy with learnable weights</i>)	2.50 ± 2.29

Table 7.24: The 3D Normalised Pose Error metric comparison between the architecture of [83] and its modified versions employing the various Foreground pixel Attention schemes described in this section.

According to the Tables 7.24,7.24, as well as to figures 7.18,7.19,7.20, we make the following observations:

- Highlighting the Foreground pixels is always beneficial for the main target of our work: tracking the object pose. The influence clutter has on regressing the pose is reduced even in the unsupervised case as pose comparison between the two streams becomes easier at the fusion level.
- This Foreground Attention scheme highlights not only the object of interest, but rather all moving parts of the scene: i.e. both the object and the user’s hand. We, also need to note that although

the user’s torso is not part of the SUN3D [163] dataset, used for augmentation and the Attention maps are not explicitly trained to ignore it, they do so implicitly, as a side effect of training. This shows that the Foreground Attention module performs a kind of motion reasoning.

- Supervising the Foreground Spatial Attention module with an auxiliary loss enhances its effect. While the unsupervised version highlights parts of the background as well, supervising it cuts out this ambiguity and clears out the Foreground soft segmentation even further. In fact, it does so without adding any computational burden during inference.
- Between the two Multi-Task weighting schemes, the learning-based one provides the lowest tracking errors and arises as the optimal choice. That happens due to the fact that the main minimization loss (tracking MSE) and the auxiliary one (foreground extraction Binary Cross Entropy) are of different scale, and the weighting process normalizes them to the same one before the BackPropagation-induced optimization.

7.12.2 Occlusion Handling

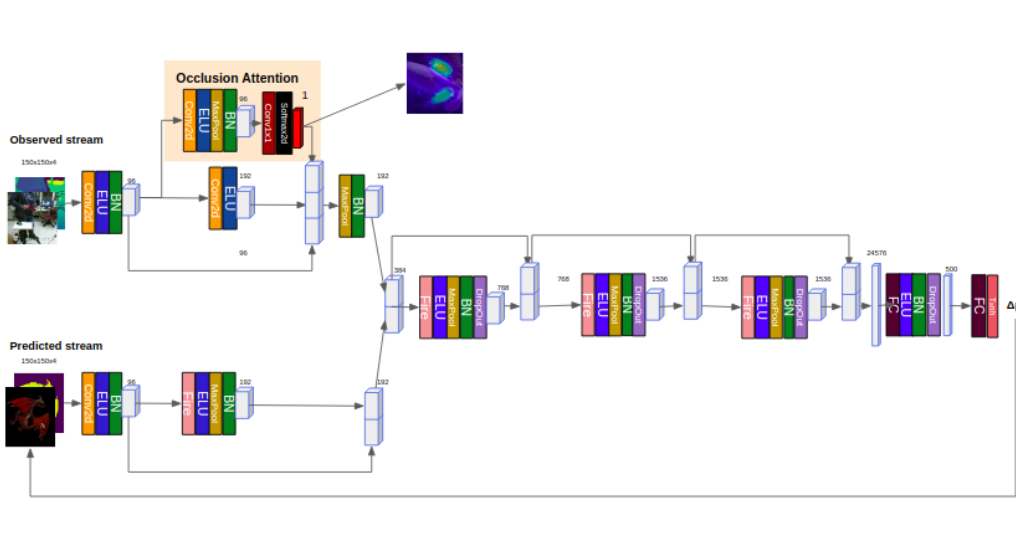


Figure 7.21: An overview of a modified version of the SoA CNN architecture of [83], where an unoccluded pixel Attention Weight map is applied to the “Observed” feature map of the second corresponding layer, for handling occlusions.

In this subsection, we discuss the optimal way to highlight regions of interest in the image space where the object is unoccluded by the user’s hand in order to enhance the Network’s capability to regress the pose under occlusion conditions. To this end, we, again start from the baseline architecture of [83] and we examine the following configurations for employing a corresponding Occlusion Attention module:

- **Soft Spatial Attention module for Occlusion Handling:** we take the feature map output of the first Convolutional layer of the “Observed” stream and we use it as input to a Soft Spatial Attention module consisted of: a 2D Convolutional and an 1×1 Convolutional 3.3.5 layer with 2D Spatial Softmax (Section 3.3.4) output, which output a single-channel probability-weight Attention matrix. This Attention weight map is applied on the feature map output of the second “Observed” Fire (see Section 3.25) layer and it is added to a unitary mapping from an incoming residual connection between itself and the feature map of the previous layer.

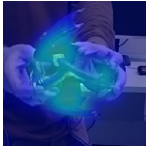


Figure 7.22: A visualized Spatial Attention weight map for Occlusion Handling overlaid on an “Observed” RGB frame. We see here, that the Occlusion Handling task is considered more difficult for the tracker’s convolutional module to learn, w.r.t. its foreground extraction counterpart (see above). The need for supervision is more important here and will, inevitably improve the accuracy of the prediction of unoccluded pixels that are important for proper estimating of the object’s pose.

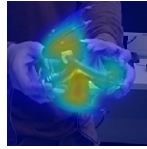


Figure 7.23: Clearly, the incorporation of a uniform binary unoccluded pixel mask ameliorates the sharpness of the attentional regions. The dragon’s wings emerge as its two parts (in this layout, at least). Both the main, tracking loss and the auxiliary one were inter-weighted with steady and equal weights (simple addition) during training.

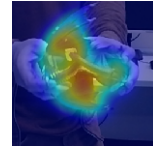


Figure 7.24: Again, we can ratify the effectiveness the adaptive learnable multi-task loss weighting scheme has not only in the numerical tracking accuracy but in its visual reasoning as well. Both wings remain the most important feature for the tracker to focus in order to accomplish its goal. However, deeper learning of the auxiliary task has generated an implicit key-point hierarchy, where attention has shifted to one wing in a more bold manner than the other.

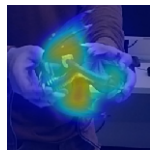


Figure 7.25: A visual demonstration of the Occlusion Handling Spatial Attention weight map that is learned by substituting the convolutional unit in the relevant module with a ConvLSTM one. Attention leakage is slightly increased and distributional peaks have been mitigated, two factors that justify the slightly diminished accuracy of this alternative method. It seems that LSTM (even this Convolutional variant) justify their reputation of being hard to train due to the combination of the backpropagation algorithm with sequential and parallel weight adaptation during the BPTT algorithm backward run.

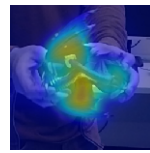


Figure 7.26: Although Gumbel Softmax, whose use in the Occlusion Handling Attentional module we witness in this figure, seems like a methodological modification with theoretical charismas, (since it allows for increased degrees of freedom in shaping the attention distribution and balancing out attention sharpness), in practice, it is proven suboptimal as simultaneous learning of the hardness level of the attentional distribution is proven an extra task that loads extra, unnecessary performance burden. As a result, nor the tracker’s accuracy nor its robustness see any spike of statistical significance and this alternation can be considered extraneous.

- **Soft Spatial Attention module for Occlusion Handling using Binary Cross Entropy with steady weights:** We use the same Attention scheme as before. The difference, now, is that the Attention weight map is not let free to generate whatever patterns will emerge, but it is rather strictly supervised adding to the Track Loss (MSE) a Binary Cross Entropy loss function (Section 3.3.17). The ground truth signal that we use as a guidance for this Binary Cross Entropy loss function is a binary occluder-unoccluded pixels map, kept during the Data Augmentation process.
- **Soft Spatial Attention module for Occlusion Handling using Binary Cross Entropy with learnable weights:** Here, we employ the same supervised Attention scheme as above. The difference, now, is that instead of just adding the Binary Cross Entropy Loss to the Track one, we leverage both using the learnable Multi-Task weighting scheme, described in 3.3.20.
- **Soft Spatial Attention ConvLSTM-based Occlusion Handling using Binary Cross Entropy with learnable weights:** Here, we employ the same strategy as in Section 7.12.2, but we substitute the last Fire (see Section 3.25) module with a ConvLSTM one. That choice is intuitively justified by the observation that the most occlusions presented in the testing sequence are not only static, but mostly dynamic and for this reason, a sequence-based model would capture better those occlusion patterns.
- **Gumbel Soft Spatial Attention Occlusion Handling using Binary Cross Entropy with learnable weights:** Here, we employ the same strategy as in 7.12.2, but we replace the 2D Spatial

Softmax module with a Gumbel Softmax one, presented in Section 3.3.16. In detail, we let the Temperature value T needed in the Gumbel Softmax value formula to be a learnable parameter of the network and we initialize it to a very high value (corresponding to Uniform weight map distribution). Before using it in the Gumbel Softmax formula, we pass it from a ReLU function to ensure that it is positive.

Architecture	Translational Error (mm)	Rotational Error (o)
[83] (20k training pairs)	48.58 ± 38.23	35.43 ± 34.74
[83] (20k training pairs) (<i>Soft Spatial Attention Occlusion Handling</i>)	25.19 ± 17.87	38.19 ± 35.70
[83] (20k training pairs) (<i>Soft Spatial Attention ConvLSTM Occlusion Handling using Binary Cross Entropy with steady weights</i>)	25.06 ± 16.24	39.67 ± 37.10
[83] (20k training pairs) (<i>Soft Spatial Attention Occlusion Handling using Binary Cross Entropy with learnable weights</i>)	15.75 ± 10.24	35.39 ± 34.52
[83] (20k training pairs) (<i>Soft Spatial Attention Occlusion Handling, with Gumbel Softmax activation, using Binary Cross Entropy with learnable weights</i>)	15.20 ± 9.02	39.94 ± 34.86
[83] (20k training pairs) (<i>Soft Spatial Attention Occlusion Handling, with ConvLSTM module, using Binary Cross Entropy with learnable weights</i>)	15.20 ± 9.02	39.94 ± 34.86

Table 7.25: The 3D Translational and Rotational error metrics comparison between the architecture of [83] and its modified versions employing the various Unoccluded pixel Attention schemes described in this section.

Architecture	Normalised Pose Error (mean \pm std)
[83] (20k training pairs)	3.44 ± 3.11
[83] (20k training pairs) (<i>Soft Spatial Attention Occlusion Handling</i>)	2.93 ± 2.58
[83] (20k training pairs) (<i>Soft Spatial Attention Occlusion Handling using Binary Cross Entropy with steady weights</i>)	3.01 ± 2.61
[83] (20k training pairs) (<i>Soft Spatial Attention Occlusion Handling using Binary Cross Entropy with learnable weights</i>)	2.50 ± 2.29
[83] (20k training pairs) (<i>Soft Spatial Attention Occlusion Handling, with Gumbel Softmax activation, using Binary Cross Entropy with steady weights</i>)	3.01 ± 2.61
[83] (20k training pairs) (<i>Soft Spatial Attention Occlusion Handling, with ConvLSTM module, using Binary Cross Entropy with steady weights</i>)	3.01 ± 2.61

Table 7.26: The 3D Normalised Pose Error metrics comparison between the architecture of [83] and its modified versions employing the various Unoccluded pixel Attention schemes described in this section.

After a careful look at the Tables 7.25,7.26 and in figures 7.12.2, we make the following observations:

- The unsupervised version of the Attention module gives an initial boost to the tracking performance by itself. However, its corresponding weight maps are too uniform, resulting in a blurry depiction of the probability weight spread.
- The introduction of the corresponding auxiliary Binary Cross Entropy loss function clearly improves this situation, as it makes the Attention peaks sharper on the unoccluded regions of the object of interest.
- Furthermore, the learnable weighting scheme arises as the optimal solution. Same as before, the main target loss: MSE and the auxiliary one: Binary Cross Entropy, are of significantly different scale due to their mathematical computation formula. As a result, learning the appropriate weighting scheme between them leaves the BackPropagation algorithm free of the burden to learn the loss scaling along with the other task during the main Network’s parameters’ optimization.
- In general, while the object is unoccluded, the Attention focuses mostly on the object’s 3D center. This is not a point that was presented explicitly to the algorithm, but rather a behaviour that it learned by itself, matching our cognitive perception that the most important point on the object is its center, unless there is a reason not to be. Indeed, as the occlusion percentage increases in the frame, the Attention weight map searches for parts of interest that stand out of the user’s grip (i.e. the neck, the wings and the tail for the dragon case) to help the algorithm to figure out the object’s pose using this remaining clues.

- Trying to replace the Soft Spatial Attention weighting of the “Observed” stream feature map with a Gumbel distribution-based variation seems intuitive at first. We think so, as, at a first glance, making the Attention’s “softness” a learnable extra parameter could give the Network the capability to converge to a local optimum where this weight map would resemble the ground truth binary mask even further. In this way, we could enhance the sharpness of our weight map peaks and boost the tracker’s performance more than in the Soft Spatial Attention case. However, we experimentally see that this is not the case as both error metrics increase. We speculate that leaving the “softness” parameter free to be optimized is proven to be an extra burden for the optimization algorithm that contradicts the main tracking target.
- Last but not least, we try to make a distinction between static and dynamic occlusion patterns. To this end, since the occlusions of our case are of the dynamic kind, we experiment with replacing the Soft Spatial 2D Convolutional Attention module with a corresponding ConvLSTM one. The results produced by this approach are close to those by the per-frame self-Attention version, but they are not proven to be superior in any way. Qualitatively, this occlusion handling strategy focuses mostly to the scene parts with the highest motion, so it leaves a higher amount of Attention to the occluder’s pixel, as well. (We feel the need to note that in order to examine the replacement of the 2D Convolutional module with the ConvLSTM one, we change the frame sampling strategy from a random/discrete to a continuous one, whose effects have been studied above.)

7.13 Hierarchical vs Parallel Clutter/Occlusion Attentions

As we have already seen in the two previous sections, both the Foreground and the unoccluded pixels’ Attention weight maps play a significant role in clearing out the adverse pixels from the feature map of the real-“Observed” stream and easing the pose difference comparison between itself and the corresponding feature map from the “Predicted” stream. The careful reader, however, would come up with the next intuitive question: Isn’t the unoccluded feature map sufficient for both clutter and occlusion handling, since the latter task is a subtask of the former? Indeed, theoretically, that is the case. However, as it has been proposed before (e.g. in DPOD [167]), avoiding mixing aliquot subtasks and assigning a single, clear target to each convolutional module improves the tracker’s performance in terms of the translation and rotation errors. In the following table, we verify this experimentally.

Then the question that come immediately to our minds is the hierarchy that these two Attention modules should be put into. Two equally intuitive options are the **Hierarchical** and the **Parallel** one. In the first of them,

- **Hierarchical Clutter and Occlusion Handling:** Here, we employ a 2-layer hierarchical Attention module: at first, we put the Foreground handling Soft Attention module which it is followed by the Occlusion handling one. They are both supervised by their corresponding binary mask and adaptively weighted with learnable weights, along with the Track Loss. With this scheme, we aim to extract the foreground pixels at first (i.e. those corresponding to the object or the user’s hand) and then further distill the feature map, highlighting only the unoccluded pixels of the object.

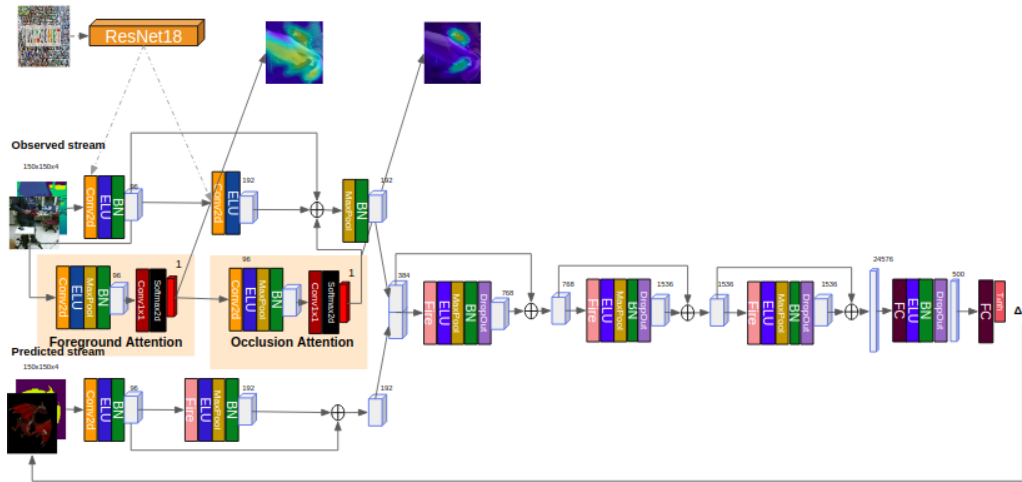


Figure 7.27: An overview of the modification of the SoA CNN architecture of [83], where we employ an hierarchical Foreground Extraction/Occlusion handling Soft Spatial Attention scheme to highlight the desired regions of interest in the feature maps’s spatial dimensions.

- Parallel Clutter and Occlusion Handling:** In this case, we use those two Attention modules in parallel. Their Attention weight outputs are separately applied to the feature map output of the second “Observed” convolutional layer. Their outputs are added together, along with that feature map and the residual mapping from the feature map of the previous layer.

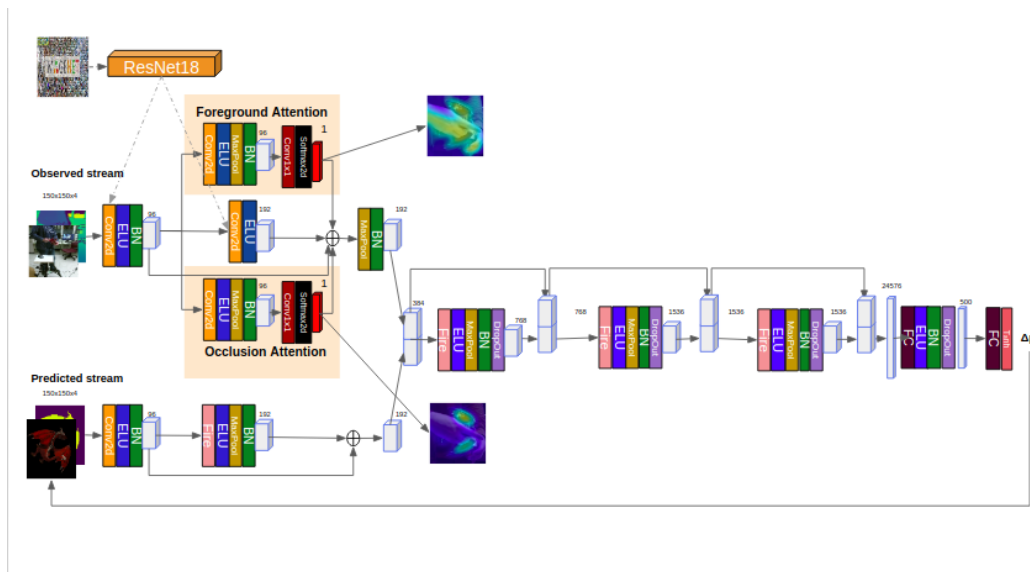


Figure 7.28: An overview of the modification of the SoA CNN architecture of [83], where we employ a parallel Foreground Extraction/Occlusion handling Soft Spatial Attention scheme to highlight the desired regions of interest in the feature maps’s spatial dimensions.

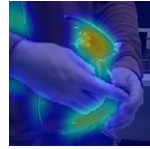
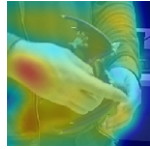


Figure 7.29: The Foreground/Clutter handling Spatial Attention weight map of the Hierarchical method, overlaid on an “Observed” RGB frame. Although the module has assigned no focus at all to the static background (i.e. the visual attributes of the room), most of the attentional distributional peaks have been gathered towards the human user instead of the object of interest. We blame this learning inefficiency to the short-term vanishing gradient effect that inevitably occurs as the gradient backward stream passes multiple softmax activations in the row. As a result, although occlusion detection is of quite high reliability the hierarchical implementation of one weight map over the other would result to a pair of sharp peaks that lie inside the object visual region (i.e. on the dragon’s wings) and a third one that stays on the user’s hand, which results in confusing the tracker and, thus, reducing its performance.

Figure 7.30: The Occlusion handling Spatial Attention weight map of the Hierarchical method, overlaid on an “Observed” RGB frame. Since this map has been sufficiently learned and due to its topographical location in the architecture (i.e. a layer after the foreground attention module), we can verify that our vanishing gradient arguments stands as it is.

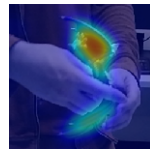


Figure 7.31: The Foreground/Clutter handling Spatial Attention weight map of the Parallel method, overlaid on an “Observed” RGB frame. This time, the Foreground extraction task is parallel to the Occlusion handling one, which exonerates gradients of the extra burden to pass two heavily nonlinear activation function in the row, before being connected again with their main convolutional stream counterparts.

Figure 7.32: The Occlusion handling Spatial Attention weight map of the Parallel method, overlaid on an “Observed” RGB frame. Maybe surprisingly, the fact that the two tasks are learned in parallel has given the extra freedom to the Occlusion handling module to weight the most important keypoint in a more explicit manner and, thus, create a lucid region of unoccluded pixel visual interest, since the other regions are left for its siamese Foreground extraction module to highlight.

Architecture	Translational Error (mm)	Rotational Error (o)
[83] (20k training pairs)	48.58 ± 38.23	35.43 ± 34.74
[83] (20k training pairs) (<i>Occlusion Handling Attention using Binary Cross Entropy Loss with learnable weights</i>)	17.60 ± 10.74	37.10 ± 35.08
[83] (20k training pairs) (<i>Hierarchical Clutter and Occlusion Handling using Binary Cross Entropy Loss with learnable weights</i>)	14.99 ± 9.89	39.07 ± 33.22
[83] (20k training pairs) (<i>Parallel Clutter and Occlusion Handling using Binary Cross Entropy Loss with learnable weights</i>)	14.35 ± 10.21	36.28 ± 32.29

Table 7.27: The 3D Translational and Rotational error metrics comparison between the architecture of [83] and its modified versions where we employ a lone Occlusion handling Attention module and two different pairing options of it with a corresponding Foreground extraction one: one hierarchical and one in parallel, as described in this section.

Architecture	Normalised Pose Error (mean \pm std)
[83] (20k training pairs)	3.45 \pm 3.11
[83] (20k training pairs) (<i>Occlusion Handling Attention using Binary Cross Entropy Loss with learnable weights</i>)	2.65 \pm 2.34
[83] (20k training pairs) (<i>Hierarchical Clutter and Occlusion Handling using Binary Cross Entropy Loss with learnable weights</i>)	2.53 \pm 2.16
[83] (20k training pairs) (<i>Parallel Clutter and Occlusion Handling using Binary Cross Entropy Loss with learnable weights</i>)	2.51 \pm 2.13

Table 7.28: The 3D Normalised Pose Error metrics comparison between the architecture of [83] and its modified versions where we employ a lone Occlusion handling Attention module and two different pairing options of it with a corresponding Foreground extraction one: one hierarchical and one in parallel, as described in this section.

According to Tables 7.13,7.28 and the figures 7.13, we make the following observations:

- In general, when both types of Attention are used, we observe the next tradeoff: when the object is mostly unoccluded the Foreground extraction weight map is more heavily highlighted than the Occlusion one. As the occlusion percentage increases in the frame, the focus shifts to the module responsible for the Occlusions. This was not a function that the tracker was optimized to perform, but it was learnt implicitly during the parameter optimization and it consists of a behaviour that matches our cognitive understanding of how humans track objects.
- The Attention blurring of the Occlusion module was more intense in the hierarchical case, resulting in worse performance. On the other hand, peaks in both modules were presented sharper in the parallel configuration.

7.14 Rotation Loss choices

In this section, we start from the observation that the MSE regression loss employed in the works of Garon et al. [[29, 83]] and applied to the 6 normalised (3 for translation and 3 for rotation) pose parameters is far from the ideal learning configuration for the Pose Estimation problem. As we have already explained, that is because of the fact that, on one hand, the 3D Translation component is a Euclidean element, but on the other hand the 3D Rotation one lies on the non-Euclidean Lie Group SO(3). So, we assume that disentangling the Pose regression loss to a (weighted) sum of a Euclidean-based Translation and SO(3)-based Rotation one would benefit the tracker’s convergence, since such a configuration consists of a Geodesic in the holistic SE(3) space. Furthermore, we examine the effect of replacing the 3D rotation representation proposed in [[29, 83]] with the 6D continuous one proposed by Zhou et al. in [171] as well as our attempt to weigh different rotation components differently according to what we have described in Section 3.3.18.

At first, we try to train the Deep Neural Network proposed in [83] separately for the 3D Translation and Rotation task. The results that we observe in the Tables 7.29,7.30 are very interesting. Our first conclusion, according to these error measurements, is that the 3D translation task is far easier to learn than the 3D Rotation one. That sounds profound as both the RGB and the Depth modality provide immediate 3D translation reasoning. Secondly, we see that the rotation estimation is benefited when we jointly estimate the object’s 3D translation as well. However, the translation one does not present a similar improvement under the coupled extra requirement for rotation estimation. We speculate this phenomenon’s cause to be the fact that 3D new position joint estimation helps the tracker to focus more to the object of interest and present a more refined estimation of its rotation. On the contrary, it is intuitive that accurately estimating the object’s orientation has little effect in knowing its 3D position.

As for the pure rotation estimation gradual modeling, in a nutshell, in Tables 7.29,7.30, we demonstrate the value of every component included in our rotation loss (leaving symmetries temporarily out of study), by: (i) regressing only the rotational parameters with the baseline architecture of [83], (ii), replacing the MSE loss with the Geodesic one, (iii), replacing the rotation parameterization of [83] with the continuous one of ([171]), and, (iv) including the Inertial Tensor weighting of each rotational component.

Architecture	Translational Error (mm)	Rotational Error (o)
[83] (20k training pairs) (<i>Joint MSE Loss for Translation and Rotation parameters</i>)	48.58 ± 38.23	35.43 ± 34.74
[83] (20k training pairs) (<i>Only Translation parameters MSE Loss</i>)	15.5 ± 11.12	-
[83] (20k training pairs) (<i>Only Rotation parameters MSE Loss</i>)	-	46.55 ± 40.88
[83] (20k training pairs) (<i>Only Rotation Geodesic Loss</i>)	-	37.69 ± 35.39
[83] (20k training pairs) (<i>Only Rotation Geodesic Loss and Continuous 6D parameterization (cross product-based orthogonalization)</i>)	-	15.25 ± 24.31
[83] (20k training pairs) (<i>Only Rotation Geodesic Loss and Continuous 6D parameterization (Gramm-Schmidt-based orthogonalization)</i>)	-	14.90 ± 21.76
[83] (20k training pairs) (<i>Only Rotation Geodesic Loss, Continuous 6D parameterization and (cross-product-based)Orthonormalized Rotation components weighting</i>)	-	13.96 ± 24.12
[83] (20k training pairs) (<i>Only Rotation Geodesic Loss, Continuous 6D parameterization and (Gramm-Schmidt-based)Orthonormalized Rotation components weighting</i>)	-	9.99 ± 13.76

Table 7.29: Comparison of the 3D rotational error metric as the proposed rotational loss scheme is gradually developed by adding one improvement after the other. Note that, in the first row, we include an experimentation on the sole 3D translation estimation, in order to understand which of the two components is easier to train to and what is the effect of coupling them to the accuracy of each one, independently.

Architecture	Normalised Pose Error (mean ± std)
[83] (20k training pairs) (<i>Joint MSE Loss for Translation and Rotation parameters</i>)	3.45 ± 3.11
[83] (20k training pairs) (<i>Only Translation parameters MSE Loss</i>)	0.45 ± 0.32
[83] (20k training pairs) (<i>Only Rotation parameters MSE Loss</i>)	2.68 ± 2.36
[83] (20k training pairs) (<i>Only Rotation Geodesic Loss</i>)	2.180 ± 2.06
[83] (20k training pairs) (<i>Only Rotation Geodesic Loss and Continuous 6D parameterization (cross product-based orthogonalization)</i>)	0.88 ± 1.40
[83] (20k training pairs) (<i>Only Rotation Geodesic Loss and Continuous 6D parameterization (Gramm-Schmidt-based orthogonalization)</i>)	0.86 ± 1.02
[83] (20k training pairs) (<i>Only Rotation Geodesic Loss, Continuous 6D parameterization and (cross-product-based)Orthonormalized Rotation components weighting</i>)	0.81 ± 1.39
[83] (20k training pairs) (<i>Only Rotation Geodesic Loss, Continuous 6D parameterization and (Gramm-Schmidt-based)Orthonormalized Rotation components weighting</i>)	0.58 ± 0.80

Table 7.30: Comparison of the Normalized Pose Error metric as the proposed rotational loss scheme is gradually developed by adding one improvement after the other. Note that, in the first row, we include an experimentation on the sole 3D translation estimation, in order to understand which of the two components is easier to train to and what is the effect of coupling them to the accuracy of each one, independently.

Our results for the Rotation Loss choice are both impressive and intuitive. Both tables justify our progressive design selections in formulating our overall rotation loss, as with the addition of each

ambiguity modeling, the 3D rotation error metric decreases, starting from $46.55^\circ \pm 40.88^\circ$ and reaching $9.99^\circ \pm 13.76^\circ$.


Chapter 8

Evaluation

But I am very poorly today and very stupid and I hate everybody and everything. One lives only to make blunders.

Charles Darwin, Diary

According to our ablation study, we proceed to merge our parallel attention modules with the Geodesic rotation loss of eq.6.36, along with the remaining elements of Section 6. We present an extensive evaluation of our method on two objects contained in the dataset of Garon et al. [83]: one asymmetrical with rich texture and complex shape (dragon) and one symmetrical with poor texture and simple 3D shape (cookiejar). However, our tracker is tested not only on them, but on three more object models, as well: the “Dog”, the “Lego block” and the “Watering can”, each of them with its own distinct characteristics. This wide spectrum of characteristics is what makes our evaluation process reliable and our results generalizable.



Object	Attributes				
	Size	Symmetry	Shape	Texture	Distinctive parts
“Dragon”	Medium	No	Complex	Rich	Yes
“Cookie Jar”	Medium	Rotoreflective	Simple	Poor and Repetitive	No
“Dog”	Medium	No	Complex	Almost None	Yes
“Lego”	Small	No	Complex	Rich and Repetitive	No
“Watering Can”	Big	No	Simple	Poor	Yes

Table 8.1: Characteristics of the five objects we test our approach on. The fact that there are no two identical items validates the generalization capabilities of our tracker.

8.1 Evaluation of modifying the baseline architecture to our proposed approach

Based on the conclusions we extracted in the previous section, we start from the layout proposed by Garon et al. in [83]

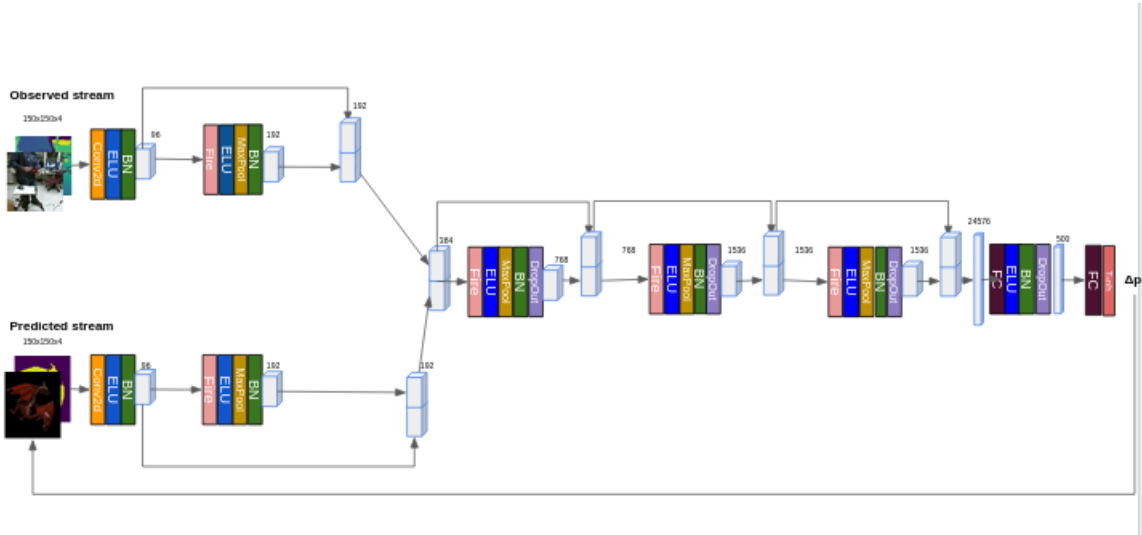


Figure 8.1: The SoA architectural design of Garon et al. [83]

and we proceed to initialize the “Observed” stream’s convolutional filter weights with the corresponding weights of a ResNet18, pretrained on Imagenet. The convolutional filter number is retained at the same level as before and the corresponding Fire modules of the “Predicted” stream do not squeeze the intermediate layers to half of the input filter number, but keep the same number of them. This happens in order to balance out the bias that will occur by replacing the Fire modules of the “Observed” convolutional stream with a fully convolutional filters, provided by the ResNet18 and pretrained on Imagenet.

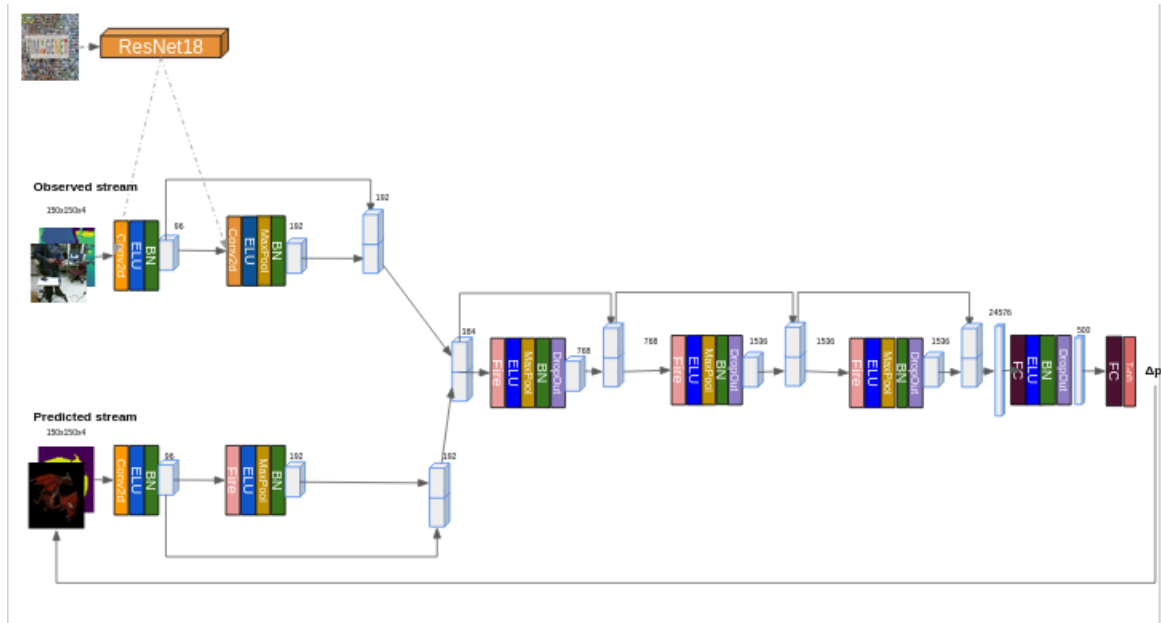


Figure 8.2: The SoA architectural design of Garon et al. [83], with the “Observed” stream initialized by the weights of a pretrained ResNet18.

Then, we replace the concatenation-based inter-layer connections, introduced in the “DenseNet” [49] paper for the first time, with Residual connections, something that our Ablation Study revealed to be a critical factor in easing the gradients’ backpropagation. Next, we complete the Data Augmentation process of Chapter 6 during our training effort.

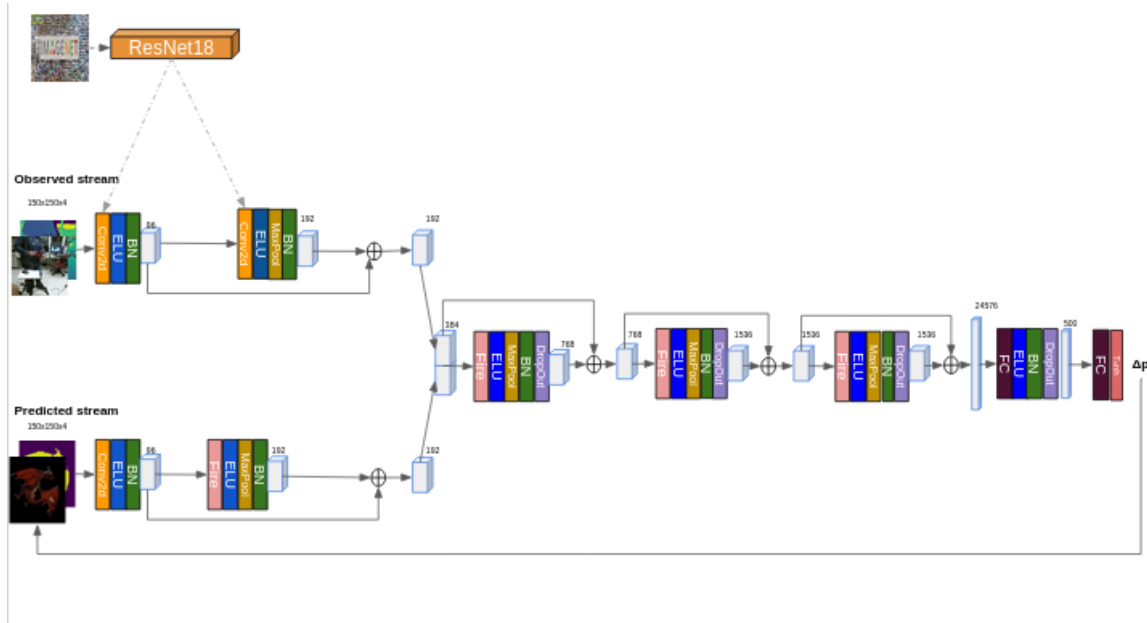


Figure 8.3: The SoA architectural design of Garon et al. [83], with the “Observed” stream initialized by the weights of a pretrained ResNet18 and with residual inter-layer connection.

Finally, we employ the Multi-task Tracking Loss, based on a 9D pose parameterization, and we add the two parallel Soft Spatial Attention modules at the “Observed” stream level. An overview of the architecture we propose is, now, completed and presented the figure, above:

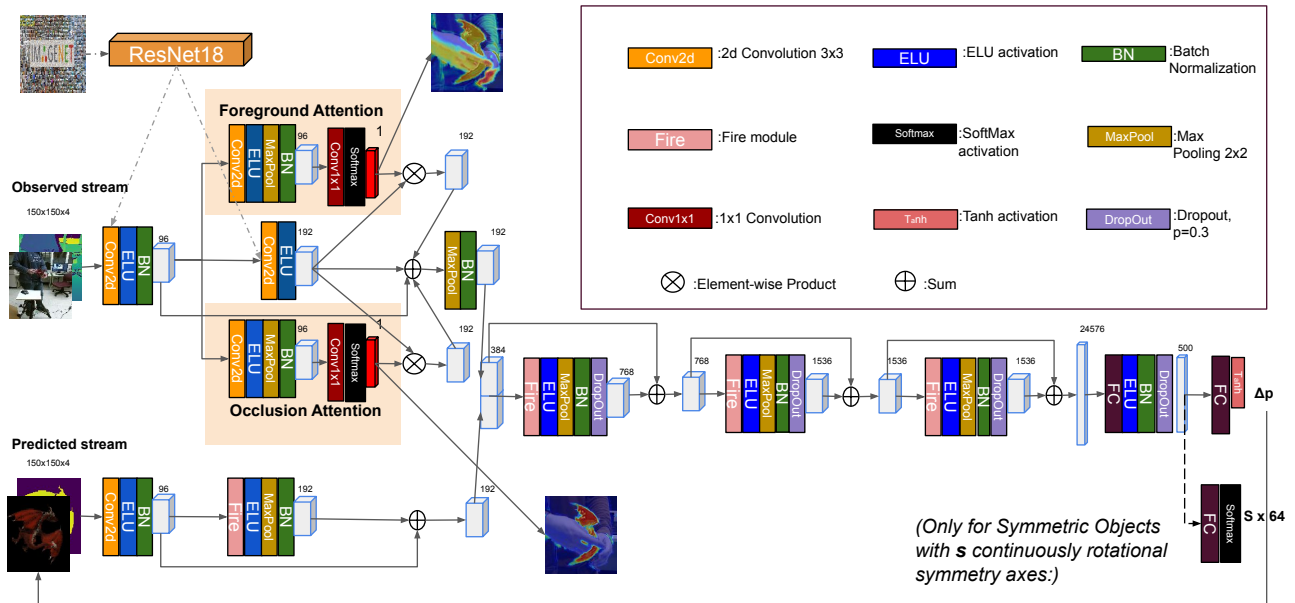


Figure 8.4: Overview of the architecture of the proposed approach. The previous layout is enhanced by the two parallel attention modules and the 9D pose parameter regression, suitable for the use of Geodesic rotation loss.

In the next table, we see the impact each of the evolutionary steps of our architecture design has to the tracker’s performance:

The last two things we need to establish is which way we will use the prior convex Loss to warmup the tracker’s weights and which weighting scheme between the various losses is found to be the optimal one.

Architecture	Translational Error (mm)	Rotational Error (o)
Garon et al. [83] (20k training pairs)	48.58 ± 38.23	35.43 ± 34.74
Garon et al. [gar] with Res-connections (<i>Asymmetric assumption</i>) (20k training pairs)	18.29 ± 23.43	22.61 ± 24.96
Garon et al. [83] with Res-connections + Data Augmentation of Chap.6 (<i>Asymmetric assumption</i>) (20k training pairs)	15.77 ± 22.42	19.94 ± 21.45
Garon et al. [83] with Res-connections + Data Augmentation of Chap.6 + Geodesic Rotation Loss (with Weight warmup) (<i>Asymmetric assumption</i>) (20k training pairs)	13.20 ± 10.42	16.28 ± 20.52
Approach of Chapter 6 (<i>Asymmetric assumption</i>) (20k training pairs)	11.63 ± 8.79	8.31 ± 6.76

Table 8.2: Quantitative overview of the impact of the gradual inclusion of every consecutive design amelioration in the proposed neural architecture.

8.2 Weighting the Multi-Task Loss

Here, we explore various weighting schemes of the multiple loss functions of our approach, both the main and the auxiliary ones. Our first approach is the crude addition of the tracking and the two Binary Cross Entropy losses. A second one is standardizing the three losses by subtracting their batchwise means and dividing by their batchwise standard deviations. Again, the recursive algorithm used for their calculation is the Welford algorithm [151]. Lastly, we consider the learnable weighting strategy that we, ultimately, propose.

- **Multiple Task Addition:**

$$Loss = L_{Transl}(\Delta\hat{\mathbf{t}}, \Delta\mathbf{t}_{GT}) + L_{Rot}(\Delta\hat{R}, \Delta R_{GT}) + L_{Clutter}^{Att} + L_{Occl}^{Att} \quad (8.1)$$

- **Rescaling of Multiple Tasks via Batch Standardization:**

$$Loss = \mathcal{N}(L_{Transl}(\Delta\hat{\mathbf{t}}, \Delta\mathbf{t}_{GT})) + \mathcal{N}(L_{Rot}(\Delta\hat{R}, \Delta R_{GT})) + \mathcal{N}(L_{Clutter}^{Att}) + \mathcal{N}(L_{Occl}^{Att}), \quad (8.2)$$

where $\mathcal{N}(\cdot)$ is the batch standardization of each loss function: $\mathcal{N}(L) = \frac{L - \text{mean}^{(Batch)}(L)}{\text{std}^{(Batch)}L}$.

- **Learnable Multi-Task hyperparameters:**

$$Loss = e^{-s_1} \cdot (e^{-v_1} \cdot L_{Transl}(\Delta\hat{\mathbf{t}}, \Delta\mathbf{t}_{GT}) + e^{-v_2} \cdot L_{Rot}(\Delta\hat{R}, \Delta R_{GT}) + v_1 + v_2) + e^{-s_2} \cdot L_{Clutter}^{Att} + e^{-s_3} \cdot L_{Occl}^{Att} + s_1 + s_2 \quad (8.3)$$

Architecture	Translational Error (mm)	Rotational Error (o)
Garon et al. [83] (20k training pairs)	48.58 ± 38.23	35.43 ± 34.74
Approach of Chapter 6 (20k training pairs) with steady and equal weights (<i>Asymmetric assumption</i>)	11.38 ± 8.94	10.98 ± 16.94
Approach of Chapter 6 (20k training pairs) with standardization of the various Losses (<i>Asymmetric assumption</i>)	13.97 ± 10.23	14.76 ± 19.24
Approach of Chapter 6 (20k training pairs) (<i>Asymmetric assumption</i>)	11.63 ± 8.79	8.31 ± 6.76

Table 8.3: Comparison of the 3D Translational and Rotational errors when evaluating different multi-task weighting schemes that leverage the main tracking and the auxiliary attention losses.

Architecture	Normalised Pose Error (mean \pm std)
Approach of Chapter 6 (20k training pairs) with steady and equal weights (<i>Asymmetric assumption</i>)	3.45 \pm 3.11
Approach of Chapter 6 (20k training pairs) with steady and equal weights (<i>Asymmetric assumption</i>)	0.96 \pm 1.24
Approach of Chapter 6 (20k training pairs) with standarization of the various Losses (<i>Asymmetric assumption</i>)	1.26 \pm 1.41
Approach of Chapter 6 (20k training pairs) (<i>Symmetric assumption</i>)	0.82 \pm 0.64

Table 8.4: Comparison of the Normalized Pose Errors (N.P.E.) when evaluating different multi-task weighting schemes that leverage the main tracking and the auxiliary attention losses.

8.3 Weight warm-up alternatives

As we have already mentioned, the Geodesic rotation loss that we have used in eq.6.36 suffers from multiple local minima. In Table 8.5, we firstly showcase this problem by reporting the error metrics measured when straightforwardly minimizing the Multitask loss with the Tracking Loss of eq.3.78.

In order to cure this adversity, we try different strategies involving convex regression losses. The first one is warming up the Network’s weights by initially minimizing the either the LogCosh or the MSE normalized parameter loss for 25 epochs and then finetuning them with eq.6.36. The second one is adding both pose regression losses (the LogCosh one and the sum of MSE and the Geodesic) and inter-weigh them with two factors: λ and $1 - \lambda$. λ is initialised to 1 and reduced by 0.1 at every epoch. The added losses are commonly weighted by the learnable intra-weight: e^{-s_1} :

$$L'_{Track} = e^{-s_1} \cdot (\lambda \cdot L_{Track}^{LogCosh} + (1 - \lambda) \cdot L_{Track}^{Geod.}) + \dots \quad (8.4)$$

Architecture	Translational Error (mm)	Rotational Error (o)
Approach of Chapter 6 with gradual intra-Track Loss weight decay (20k training pairs) (<i>Asymmetric assumption</i>)	15.29 \pm 12.16	13.03 \pm 22.54
Approach of Chapter 6 with learnable MultiTask Track Loss weighting (20k training pairs) (<i>Asymmetric assumption</i>)	12.40 \pm 10.16	13.50 \pm 9.16
Approach of Chapter 6 (20k training pairs) (<i>Asymmetric assumption</i>)	10.49 \pm 7.85	11.47 \pm 9.15

Table 8.5: The 3D Translational and Rotational error metrics’ comparison between the various weight warm-up schemes that alter the baseline architecture of Garon et al. [83]

Architecture	Normalised Pose Error (mean \pm std)
Approach of Chapter 6 with gradual intra-Track Loss weight decay (20k training pairs) (<i>Asymmetric assumption</i>)	1.19 \pm 1.65
Approach of Chapter 6 with learnable MultiTask Track Loss weighting (20k training pairs) (<i>Asymmetric assumption</i>)	1.14 \pm 0.82
Approach of Chapter 6 (20k training pairs) (<i>Asymmetric assumption</i>)	0.97 \pm 0.81

Table 8.6: The Normalized Pose Error (N.P.E.) metrics’ comparison between the various weight warm-up schemes that alter the baseline architecture of Garon et al. [83]

8.4 Object cases

We, initially, assess the validity of our method on two distinctive object cases, with very different characteristics between them. This way, we will have the chance to evaluate the way our approach handles the various challenges of 6D Pose Tracking, both qualitatively and quantitatively and have a spherical overview of the capabilities and boundaries of our tracker. In all the figures below, we demonstrate the frames of the video sequence where the baseline tracker failed the worst (or at least one of the worst and the most indicative one) and is accompanied by the frames produced by the improved tracker variations, where the changes we propose in this work are gradually applied.

Addressing the elephant in the room: Do we overfit to the pose error metrics?

The careful reader would immediately raise objections w.r.t. the scientific validity of our experimentation, occasioned by the observation that pose tracking loss that we train our network to minimize is a combination of the two pose error metrics that will use to validate the success of our tracker. So, do we overfit in purpose to the evaluation metrics in order to present better results? Since we had the same concern, we will not only report the absolute translation and rotation error values (in terms of temporal mean and standard deviation of our observations), but we will also include in our analysis the **unbiased tracking failures**. This way, if the translation and rotation error metrics decrease and the tracking failures become more rare, after applying our method, the absolute values of those errors are completely trustworthy. Only the opposite scenario (i.e. both errors decreasing and tracking failures increasing) could justifiably make us question the quality of our results. Besides, the qualitative analysis will, also, clearly demonstrate the superiority of our method and will consist a disincentive to any, logical, worries.

Inference Speed

Although more computationally intense, the speed of our CNN (**40 frames/sec.**), still lies within the boundaries of real-time performance set by Garon et al. [83].

Color Coding for reading the following error plots:

In the next error plots, we present the temporal Translational (in mm) and Rotational (in degrees) error differences between predicted and actual poses. Temporal intervals with high speed are depicted in **green** color background and temporal intervals of high occlusion patterns are depicted in **grey** color background.

Moreover, we plot the 6D temporal trajectories of the object models we test our algorithm on, for the 'Hard Interaction' scenario, which is the most interesting and inclusive of them all. In simple words, we plot the whole 3D positional trajectories and we sample 20 rotational 3D orientations along the overall length of its path. For each object, we plot the ground truth recorded trajectories of the dataset of Garon et al.[83] in **black**, the trajectories produced by the State-of-the-Art method of Garon et al.[83] in **blue** and our, improved, trajectories, in **red**. For the special case of the continuously rotational symmetric "Cookie Jar" model, we also depict the trajectory of the best symmetric handling algorithm, that is incorporated in our tracker, from all those we try in this work, in **orange** this time. Evidently, the best predicted trajectory is the one that most closely matches the Ground Truth one, both in terms of translation and rotation.

Note that, in the rest of this Chapter, we report only the 3D Translational and 3D Rotational pose errors, as well as the overall tracking failures (when the tracker is re-iterated only when it fails irrecoverably, according to the metric we have set above). As a result, the Normalized Pose Error metric is omitted as it only served in selecting the optimal design principles.

8.4.1 The Dragon model: the asymmetric case

The first object that we test our approach on is the "**Dragon**" 3D CAD model provided in the dataset of Garon et al.[83]. It is the one with the richest texture and most complex shape available. It is equipped with similar colors, in different texture patterns though and with parts that cause self-occlusions in 3D.

Therefore, it will be a hard crash test for our tracker and will help us identify which characteristics in the test object structure are more beneficial to the algorithm and which ones hinder its success. These effects will become more bold when this, first, object will be compared to the second one: an object with completely opposite characteristics.

We estimate the Inertia Tensor of the “Dragon” 3D object model:

$$\Lambda^{(Dragon)} = \begin{bmatrix} 0.0433 & -0.0016 & -0.0011 \\ -0.0016 & 0.0336 & -0.0032 \\ -0.0011 & -0.0032 & 0.0287 \end{bmatrix} \quad (8.5)$$

and after the Gramm-Schmidt orthonormalization:

$$\Lambda_{G.S.}^{(Dragon)} = \begin{bmatrix} 0.9989 & 0.0351 & 0.0297 \\ -0.0378 & 0.9946 & 0.0963 \\ -0.0261 & -0.0973 & 0.9949 \end{bmatrix} \quad (8.6)$$

We observe that $\Lambda^{(Dragon)}$ is almost diagonal, as its diagonal elements are at least one order of magnitude larger than the non-diagonal ones and that its axis with the maximum inertia is its x-axis.

The converged Multi-Task Loss weights are:

Multi-Task Weight Name	Multi-Task Weight Value
$\mathbf{w}_1^{(s)} = \mathbf{e}^{-s_1}$	0.0113
$\mathbf{w}_2^{(s)} = \mathbf{e}^{-s_2}$	0.5420
$\mathbf{w}_3^{(s)} = \mathbf{e}^{-s_3}$	1.6673
$\mathbf{w}_1^{(v)} = \mathbf{e}^{-v_1}$	0.0754
$\mathbf{w}_2^{(v)} = \mathbf{e}^{-v_2}$	0.0707

Static Object Pose Tracking on a 2D turntable:

The first case that we need to assess our method on is the **static** one. It provides an estimate of the tracker’s robustness to pose jitter and its generalization capabilities to different static scales. Although the dataset of Garon et al.[83] contains 4 such sequences, for 4 different scale changes of near-far away from the camera, here, we demonstrate our experimentation only on the last one.

Subcase 1: The object stands near the camera

In general, we observe that this, first, subcase is the easiest one and is the root cause of less pose errors.

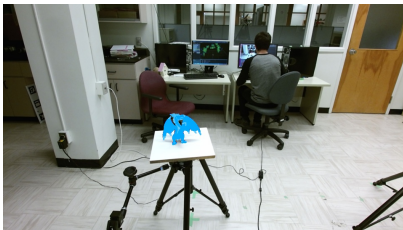


Figure 8.5 (a) Demonstration of an “Observed” RGB frame of the “Stability near” scenario with a rendered predicted pose of the “Dragon” model, generated by the baseline tracker of Garon et al.[83].



Figure 8.6 (b) Demonstration of an “Observed” RGB frame of the “Stability near” scenario with a rendered predicted pose of the “Dragon” model, generated by the baseline tracker of Garon et al.[83] with Residual inter-layer connections and augmentation/initialization/sampling strategic improvements.

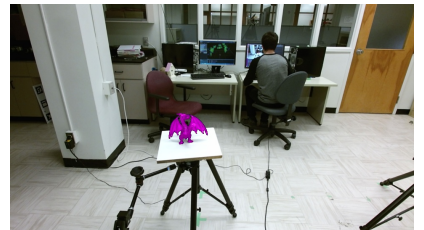


Figure 8.7 (c) Demonstration of an “Observed” RGB frame of the “Stability near” scenario with a rendered predicted pose of the “Dragon” model, generated by the proposed tracker.

Architecture	Translational Error (mm)	Rotational Error (o)	Fails
Approach of Garon et al. [83] (20k training pairs) (<i>Asymmetric assumption</i>) re-iterate every 15 frames	11.71 ± 6.66	3.52 ± 2.17	-
Approach of Garon et al. [83] (20k training pairs) (<i>Asymmetric assumption</i>) re-iterate every time the tracker fails	19.14 ± 10.39	5.95 ± 3.48	3
Approach of Garon et al. [83] with Residual inter-layer connections and improved sampling/initialization/augmentation strategy (20k training pairs) (<i>Asymmetric assumption</i>) re-iterate every 15 frames	8.73 ± 3.81	2.35 ± 0.87	-
Approach of Garon et al. [83] with Residual inter-layer connections and improved sampling/initialization/augmentation strategy (20k training pairs) (<i>Asymmetric assumption</i>) re-iterate every time the tracker fails	12.54 ± 1.73	2.37 ± 0.31	1
Approach of Chapter 6 (20k training pairs) (<i>Asymmetric assumption</i>) re-iterate every 15 frames	5.59 ± 1.99	1.18 ± 0.39	-
Approach of Chapter 6 (20k training pairs) (<i>Asymmetric assumption</i>) re-iterate every time the tracker fails	7.07 ± 0.78	1.44 ± 0.22	0

Table 8.7: 3D Translational and Rotational Pose Errors and Fail count of the “Dragon” model, for the “Stability near” for the baseline approach of Garon et al. [83], without/with sampling/initialization/augmentation improvements we propose and the approach of Chapter 6.

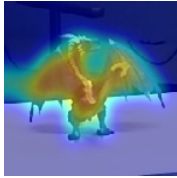


Figure 8.8 (a) Demonstration of the Foreground Extraction Attention map provided by the proposed tracker in the “Stability near” scenario.

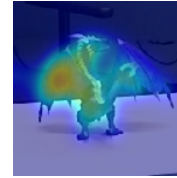


Figure 8.9 (b) Demonstration of the Occlusion Handling Attention map provided by the proposed tracker in the “Stability near” scenario.

Qualitative analysis:

Baseline tracker of Garon et al[83](*subfig(a)*):

- We have to admit that the baseline tracker of Garon et al.[83] is a fairly good starting point. Indicatively, we observe zero translation error in x,y-dimensions and an extremely small scale error in z-dimension. As for this subcase of the dataset, the translation problem is considered solved.
- As for rotations, initially insignificant rotational drift errors accumulate over time and cause tracking failures. The amount of these failures may be medium, but is presented as wanton since this is the easiest case study in the whole dataset.
- In balance, the baseline tracker is mediocly robust to the overall pose jitter. Although the object does not move, the tracker loses details of the object’s pose state without being challenged in any significant way.

Baseline with Res-connections+Sampling/Augmentation amelioration(*subfig(b)*):

- These algorithmic changes (the Residual architectural additions and the sampling/augmentation-randomization improvements) that we apply, reduce the pose jitter. This is obviously caused by facilitating the gradient trajectory from the latter layers to the former ones and by modeling the noise distribution variability and the domain adaptation procedure in a more careful way.
- **However**, the rotational errors eventually accumulate and cause the tracker to fail. This clearly indicates that a more studious formulation of rotational properties (w.r.t. rotational representation and weighting) is more of a necessity and less of a luxury.

Ours(*subfig(c)*):

- Both the pose error metrics and the tracking failures become the minimum we have measured so far. Rotational errors do not accumulate anymore. We can see that qualitatively in the figures above. For the Baseline method of Garon et al.[83] the frame presented has gathered a pose error that makes the estimated position of the wings of the dragon to intersect with the ground truth location of its mouth. On the contrary, for the same frame, our approach is far more accurate in placing all of the dragon’s parts at their right positions and orientations.
- *Foreground Attention*: As for this first Attention weight map, we observe that it, indeed, successfully distinguishes the object’s features from the features of the background. The Attention spike is distributed uniformly throughout the whole object shape.
- *Occlusion Attention*: We observe that, since no external occluder exists in the scene, this module learns to recognize **self-occlusions** implicitly and without any specific supervision. Thus, it focuses its Attention, between the two wings, only to the one which remains unoccluded by the tail.

Error Plotting (left: re-iterate every 15 frames) and (right: re-iterate when the tracker fails):

Subcase 2 The object stands far away from the camera

This scenario is a little more difficult for the tracker to figure out. However, since, during preprocessing, we crop and rescale the input frames, the feedback process functions pretty well, in general.

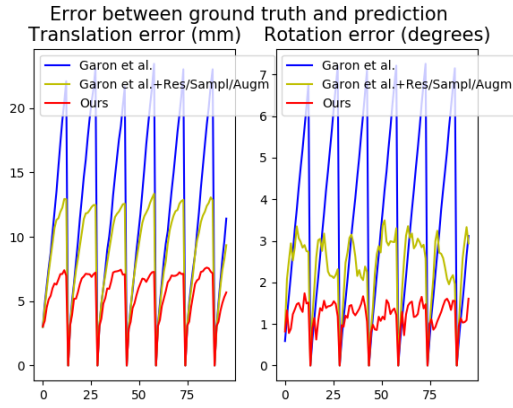


Figure 8.10 3D Translational and Rotational error metrics' plot, for the “Stability near” scenario of the Dragon 3D CAD model, in the case of tracker re-iteration every 15 frames.

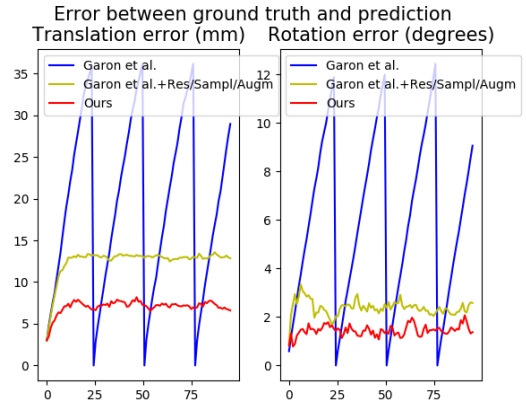


Figure 8.11 3D Translational and Rotational error metrics' plot, for the “Stability near” scenario of the Dragon 3D CAD model, in the case of tracker re-iteration only after a failure.

Architecture	Translational Error (mm)	Rotational Error (o)	Fails
Approach of Garon et al. [83] (20k training pairs) (<i>Asymmetric assumption</i>) re-iterate every 15 frames	20.35 ± 12.13	5.74 ± 3.32	-
Approach of Garon et al. [83] (20k training pairs) (<i>Asymmetric assumption</i>) re-iterate every time the tracker fails	22.73 ± 13.59	6.40 ± 3.50	6
Approach of Garon et al. [83] with Residual inter-layer connctions and improved sampling/initialization/augmentation strategy (20k training pairs) (<i>Asymmetric assumption</i>) re-iterate every 15 frames	15.37 ± 6.69	3.20 ± 0.97	-
Approach of Garon et al. [83] with Residual inter-layer connctions and improved sampling/initialization/augmentation strategy (20k training pairs) (<i>Asymmetric assumption</i>) re-iterate every time the tracker fails	24.08 ± 3.54	3.32 ± 0.30	2
Approach of Chapter 6 (20k training pairs) (<i>Asymmetric assumption</i>) re-iterate every 15 frames	14.63 ± 6.04	1.96 ± 0.58	-
Approach of Chapter 6 (20k training pairs) (<i>Asymmetric assumption</i>) re-iterate every time the tracker fails	21.76 ± 2.99	1.81 ± 0.19	0

Table 8.8: 3D Translational and Rotational Pose Errors and Fail count of the “Dragon” model, for the “Stability far” for the baseline approach of Garon et al. [83], without/with sampling/initialization/augmentation improvements we propose and the approach of Chapter 6.



Figure 8.12 (a) Demonstration of an “Observed” RGB frame of the “Stability far” scenario with a rendered predicted pose of the “Dragon” model, generated by the baseline tracker of Garon et al.[83].

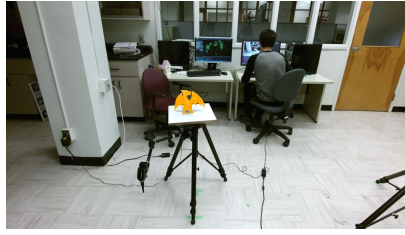


Figure 8.13 (b) Demonstration of an “Observed” RGB frame of the “Stability far” scenario with a rendered predicted pose of the “Dragon” model, generated by the baseline tracker of Garon et al.[83], with Residual inter-layer connections and augmentation/initialization/sampling strategic improvements.

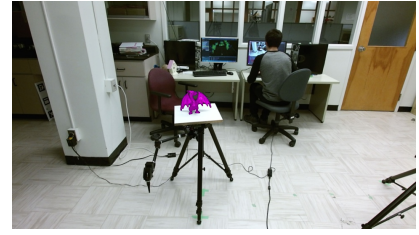


Figure 8.14 (c) Demonstration of an “Observed” RGB frame of the “Stability far” scenario with a rendered predicted pose of the model generated by the proposed tracker.

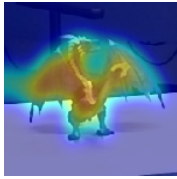


Figure 8.15 (a) Demonstration of the Foreground Extraction Attention map provided by the proposed tracker in the “Stability far” scenario.



Figure 8.16 (b) Demonstration of the Occlusion Handling Attention map provided by the proposed tracker in the “Stability far” scenario.

In a nutshell:

Qualitative analysis:

Baseline tracker of Garon et al[83](*subfig(a)*):

- Rotational errors accumulate relatively faster and, thus, tracking failures are increased in this subcase.
- Translation estimation remains extremely accurate in this subcase, as well. Translation estimation in the z-dimension appears to present a more substantial uncertainty. However, it never becomes the root cause of a tracking failure.

Baseline with Res-connections+Sampling/Augmentation amelioration(*subfig(b)*):

- Adding the residual connections and the more sophisticated data sampling and augmentation strategy in fact eliminates translation error.
- Rotational errors become smaller, making drift accumulation longer and rotational failures more rare. However, they remain and are still quite frequent. Again, proper rotational modeling rises as a need in reducing rotational drift.

Ours(*subfig(c)*):

- More accurate rotational estimation. Pose errors that correspond to rotations are, in general, small and thus, leave feedback to correct small drifts before they accumulate over time.
- *Foreground Attention*: In spite of the fact that the object is further away than in the previous subcase, cropping the input frames and rescaling both modalities make both Attentional modules to correspond in the same way as before. Foreground attention is, again, successful in distinguishing the object shape from its background and still shares focus uniformly to pretty much all its parts, in a uniform way.

- *Occlusion Attention*: The same pattern emerges here, as in the previous subcase. Again, since there is no human intervention (or from another object) this module implicitly recognizes self-occlusions and chooses (at least in the frame presented) to focus more at the wing that is in front of the tail instead of the other other that is (partially) hidden by it.

Error Plotting (left:re-iterate every 15 frames) and (right:re-iterate when the tracker fails):

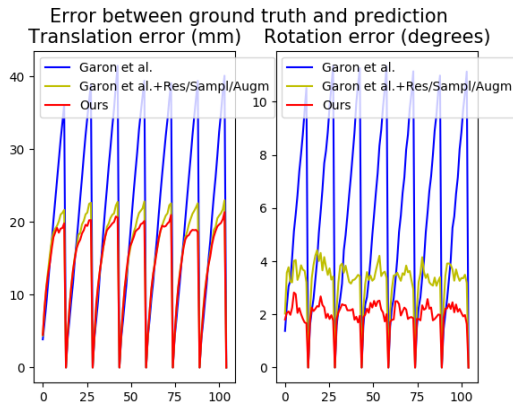


Figure 8.17 3D Translational and Rotational error metrics' plot, for the “Stability far” scenario of the Dragon 3D CAD model, in the case of tracker re-iteration every 15 frames.

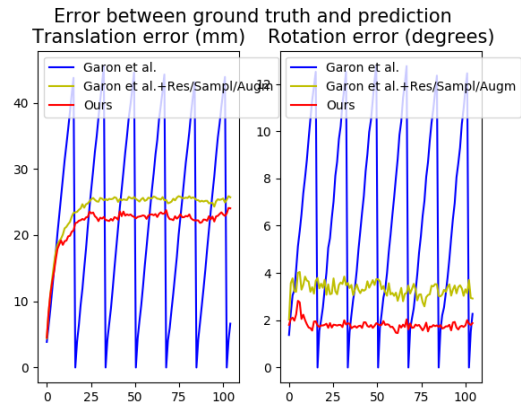


Figure 8.18 3D Translational and Rotational error metrics' plot, for the “Stability far” scenario of the Dragon 3D CAD model, in the case of tracker re-iteration only after a failure.

Object Pose Tracking on a 2D turntable with high occlusion degree (75 %):

Now, we shift our focus to a second study case where the object is placed at a turntable which rotates slowly, but the catch here is that the object is occluded at a gradual percentage per subcase. In particular, in order to avoid making our analysis too rambling, we only assess our tracker in the last subcase where 75% or more of the object is occluded all the time. The occluder in this scenario is a white plane which will be positioned in two possible configurations: (a) vertically and (b) horizontally.

Vertical Occlusion

Architecture	Translational Error (mm)	Rotational Error (o)	Fails
Approach of Garon et al. [83] (20k training pairs) (<i>Asymmetric assumption</i>) re-iterate every 15 frames	18.20 ± 11.81	13.14 ± 8.85	-
Approach of Garon et al. [83] (20k training pairs) (<i>Asymmetric assumption</i>) re-iterate every time the tracker fails	12.54 ± 8.56	17.24 ± 11.03	13
Approach of Garon et al. [83] with Residual inter-layer connections and improved sampling/initialization/augmentation strategy (20k training pairs) (<i>Asymmetric assumption</i>) re-iterate every 15 frames	17.07 ± 11.72	17.10 ± 10.88	-
Approach of Garon et al. [83] with Residual inter-layer connections and improved sampling/initialization/augmentation strategy (20k training pairs) (<i>Asymmetric assumption</i>) re-iterate every time the tracker fails	12.18 ± 7.94	17.52 ± 10.22	10
Approach of Chapter 6 (20k training pairs) (<i>Asymmetric assumption</i>) re-iterate every 15 frames	12.87 ± 10.49	14.66 ± 12.98	-
Approach of Chapter 6 (20k training pairs) (<i>Asymmetric assumption</i>) re-iterate every time the tracker fails	14.25 ± 9.30	13.64 ± 8.13	8

Table 8.9: 3D Translational and Rotational Pose Errors and Fail count of the “Dragon” model, for the “75% Vertical Occlusion” scenario for the baseline approach of Garon et al. [83], without/with sampling/initialization/augmentation improvements we propose and the approach of Chapter 6.

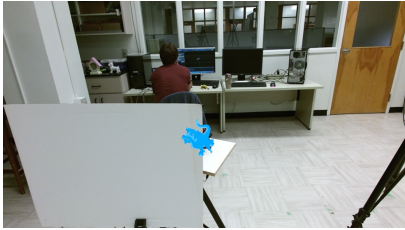


Figure 8.19 (a) Demonstration of an “Observed” RGB frame of the “75% Vertical Occlusion” scenario with a rendered predicted pose of the model generated by the baseline tracker of Garon et al.[83]

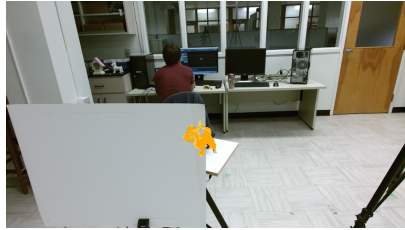


Figure 8.20 (b) Demonstration of an “Observed” RGB frame of the “75% Vertical Occlusion” scenario with a render predicted pose of the model generated by the baseline tracker of Garon et al.[77] with Residual inter-layer connections and augmentation/initialization/sampling strategic improvements.

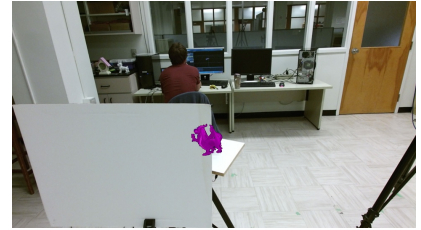


Figure 8.21 (c) Demonstration of an “Observed” RGB frame of the “75% vertical occlusion scenario” scenario with a render predicted pose of the model generated by the proposed tracker.



Figure 8.22 (a) Demonstration of the Foreground Extraction Attention map provided by the proposed tracker in the “75% Vertical Occlusion” scenario.



Figure 8.23 (b) Demonstration of the Occlusion Handling Attention map provided by the proposed tracker in the “75% Vertical Occlusion” scenario.

Qualitative analysis:

Baseline tracker of Garon et al[83](*subfig(a)*):

- The baseline tracker is highly sensitive to occlusions. The high occlusion percentage of the case that we examine causes it to lose track of rotation (primarily) and, even, translation (more rare, but still, existent scenario) many times during the inference stage.
- As soon as both wings and the head of the dragon disappear from the camera’s field of view, it cannot estimate the object’s pose by any means. Very large errors are instantly created and failures occur many times in a very short period of time.
- The right hand of the dragon seems to be the most indicative of its pose. This is fairly logical since the fact that it has two wings may confuse the tracker and the spinal and head texture are highly ambiguous under certain viewpoints. Additionally, this right hand is never occluded, under any viewpoint.
- It is profound that the most weak component in terms of estimation is the rotation around the z-axis. This is completely normal as it is the only one in which the object is moving (due to the turntable’s movement) and the one that causes these changes in its appearance and hard occlusion patterns.

Baseline with Res-connections+Sampling/Augmentation amelioration(*subfig(b)*):

- The algorithmic ameliorations that we primarily apply improve the tracker’s accuracy across all components. Pose errors are genuinely smaller and take longer to accumulate. For example, it does not lose track of the 13th frame (which was the case previously) and of the last frame, for the same reasons as well. Even for frames that the tracker loses, pose errors are both smaller. However, there still cases that very difficult to figure out (e.g. when most salient object parts are hidden from the sensor’s sight).

Ours(*subfig(c)*):

- Our approach produces the least failures up to date. It may still produce tracking errors (especially rotational ones) and failures but they are more rare. Besides, the reader must not forget that such a high occlusion percentage will hide over 3 quarters of the object’s characteristics.

Following, we observe the particular Attention weight’s effect on the tracker’s functionality:

- *Foreground Attention*: This Foreground Attentional module transitions focus (in that order) from the dragon’s tail to its head and, finally, its wings (first the one and then the other). Besides the features that strictly belong to the object, attention spreads to pixels that correspond to the plane-occluder as well. Sharp attention patterns implicitly occur, logically, on the few parts of the object that are still visible at each frame.
- *Occlusion Attention*: Here, the general pattern is mostly the same with its counterpart that is dedicated to Foreground Extraction. Differences can be spotted to the fact that this, separate, module does not assign any attention weights to parts of the occluders. It, also, has a different focus pattern sequence with the attention shifting from the wing to the head, e.g., as the object turns. Here, we feel the need to spot, as a proof of concept, the fact that, for the frame sequence

that most outstanding object parts are absent from the scene, the Occlusion Attention module has nowhere to form a sharp Attentional peak, so it spreads its weights throughout the whole image space.

Error Plotting (left:re-iterate every 15 frames) and (right:re-iterate when the tracker fails):

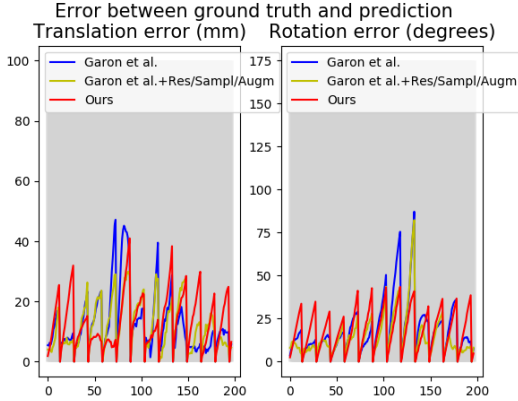


Figure 8.24 3D Translational and Rotational error metrics' plot, for the “75 % Horizontal Occlusion” scenario of the Dragon 3D CAD model, in the case of tracker re-iteration every 15 frames.

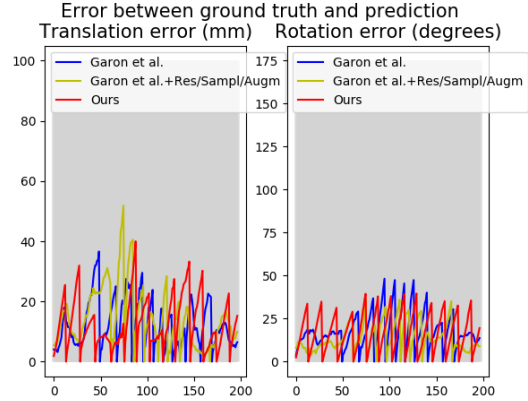


Figure 8.25 3D Translational and Rotational error metrics' plot, for the “75% Horizontal Occlusion” scenario of the Dragon 3D CAD model, in the case of tracker re-iteration only after a failure.

Horizontal Occlusion

The second occlusion subcase, of 75% or more, again, is the one where the plane-occluder is placed horizontally w.r.t. the object position on the turntable.

Architecture	Translational Error (mm)	Rotational Error (o)	Fails
Approach of Garon et al. [83] (20k training pairs) (<i>Asymmetric assumption</i>) re-iterate every 15 frames	16.02 ± 8.42	18.35 ± 11.71	-
Approach of Garon et al. [83] (20k training pairs) (<i>Asymmetric assumption</i>) re-iterate every time the tracker fails	12.54 ± 8.56	17.24 ± 11.03	13
Approach of Garon et al. [83] with Residual inter-layer connctions and improved sampling/initialization/augmentation strategy (20k training pairs) (<i>Asymmetric assumption</i>) re-iterate every 15 frames	14.23 ± 8.97	6.71 ± 9.12	-
Approach of Garon et al. [83] with Residual inter-layer connctions and improved sampling/initialization/augmentation strategy (20k training pairs) (<i>Asymmetric assumption</i>) re-iterate every time the tracker fails	14.16 ± 9.31	24.12 ± 16.20	20
Approach of Chapter 6 (20k training pairs) (<i>Asymmetric assumption</i>) re-iterate every 15 frames	12.68 ± 11.49	13.00 ± 9.14	-
Approach of Chapter 6 (20k training pairs) (<i>Asymmetric assumption</i>) re-iterate every time the tracker fails	12.73 ± 9.52	14.19 ± 9.36	10

Table 8.10: 3D Translational and Rotational Pose Errors and Fail count of the “Dragon” model, for the “75% Horizontal Occlusion” scenario for the baseline approach of Garon et al. [83], without/with sampling/initialization/augmentation improvements we propose and the approach of Chapter 6.

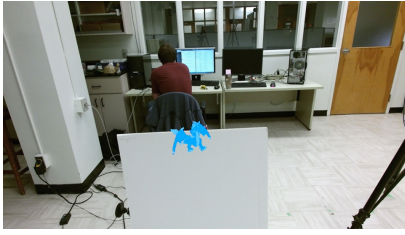


Figure 8.26 (a) Demonstration of an “Observed” RGB frame of the “75% Vertical Occlusion” scenario with a render predicted pose of the model generated by the baseline tracker of Garon et al.[83].

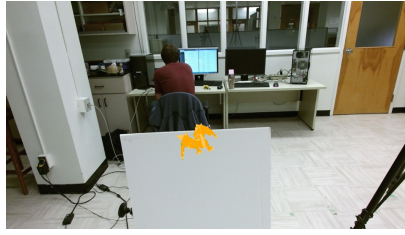


Figure 8.27 (b) Demonstration of an “Observed” RGB frame of the “75% Horizontal Occlusion” scenario with a render predicted pose of the model generated by the baseline tracker of Garon et al.[83] with Residual inter-layer connections and augmentation/initialization/sampling strategic improvements.

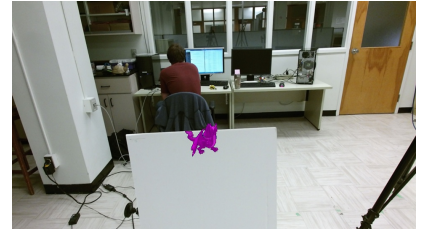


Figure 8.28 (c) Demonstration of an “Observed” RGB frame of the “75% Horizontal Occlusion” scenario with a render predicted pose of the model generated by the proposed tracker.



Figure 8.29 (a) Demonstration of the Foreground Extraction Attention map provided by the proposed tracker in the “75% Horizontal Occlusion” scenario.



Figure 8.30 (b) Demonstration of the Occlusion Handling Attention map provided by the proposed tracker in the “75% Horizontal Occlusion” scenario.

Qualitative analysis:

Overall, surprisingly, this subcase is proven to be the more difficult of the two and causes various types of failures, along all components. More ambiguities seem to be created here and less distinctive features remain uncovered. So, we have less information to feed our trackers. The following analysis shows how delicately we utilized these limited visual resources.

Baseline tracker of Garon et al[83](*subfig(a)*):

- The baseline tracker of Garon et al.[83] faces a lot of problems in this subcase with major translation and rotational errors appearing in many frames and along all components. The most prominent feature that escapes occlusion is the head of the dragon, so the complicated nature of its texture (e.g. white stripes break the red skin both in the front part of its nose and at its back, at the horns) misguides the tracker in many frames. The dragon’s wings appear more in certain subsequences than the head, their utility is of slightly higher value but they are rarely sufficient enough to save the tracker from failing.

Baseline with Res-connections+Sampling/Augmentation amelioration(*subfig(b)*):

- Classically, estimations of this variant are more robust to jitter and thus drifting is smoother. However, the tracker is still very reliant on seeing the whole of the important object features (e.g. the whole of the dragon’s head) and when only a small portion of them remains visible, it completely loses track of the object until it returns to such a state.

Ours(*subfig(c)*):

- Here, the Foreground and Occlusion Attention modules start to show their true value. Our proposed tracker keeps track of the object for much longer and accomplishes to balance out, through feedback, many of the errors created through time.
- *Foreground Attention*: This Attention map shows that as soon both the wings and the tail of the dragon are lost from the sensor’s sight, the object is ambiguous enough to cause multiple and irrecoverable failures. The dragon has to be reset to the ground truth position and orientation

before reinitiating the tracker. The attention temporal transition starts from focusing on the head, and when the wings appear, they take, initially half, and then, more of the tracker’s sharp weights. This visualization proves their value in estimating the pose.

- *Occlusion Attention*: This module enhances the intuition provided above in the most prominent manner. It is stuck on the head all the time, since this is the object feature that is unoccluded most of the time, but the Attentional peak it provides is rather dull.

Error Plotting (left:re-iterate every 15 frames) and (right:re-iterate when the tracker fails):

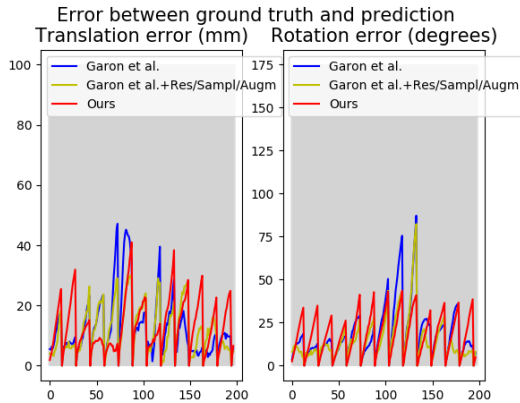


Figure 8.31 3D Translational and Rotational error metrics’ plot, for the “75% Horizontal Occlusion” scenario of the Dragon 3D CAD model, in the case of tracker re-iteration every 15 frames.

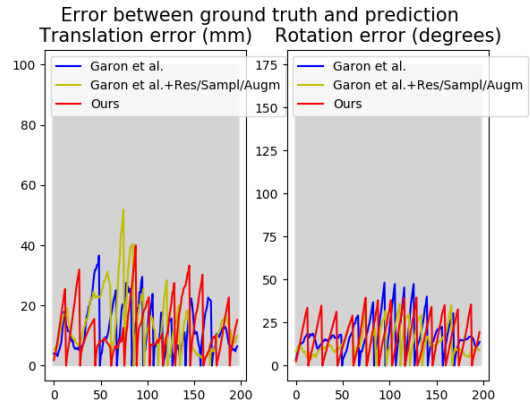


Figure 8.32 3D Translational and Rotational error metrics’ plot, for the “75% Horizontal Occlusion” scenario of the Dragon 3D CAD model, in the case of tracker re-iteration only after a failure.

8.4.2 3D Object Movement

Now, let’s move to the testing scenarios that are the most interesting ones: those with movement of the object in 3D at a variety of extra adversities. They are the most important in our study for three main reasons:

1. First of all, they are the only parts of the dataset that do not have replicas in the other pose estimation datasets that we have mentioned so far. None of them incorporates full 3D motion with reliable ground truth labels without any extra constraint.
2. Secondly, they gradually incorporate various challengies (starting from combinations of 3D translation and rotation motion, abrupt direction changes, scales outside the training set, dynamic occlusion patterns, different backgrounds across time, high speeds etc.)
3. Thirdly, the changes we applied in the baseline architecture of Garon et al.[83] were made with these scenarios in mind, as they are the most general ones.

Free 3D Object Movement:Translation Only

The first scenario that we are going to discuss is the “Translation Only” one. Here, the dragon object is translated freely in 3D, almost at the same configuration, and without occlusions. Thus, this scenario, de facto, assesses the responsiveness accuracy of our tracker to one of the two motion components: translation.

Architecture	Translational Error (mm)	Rotational Error (o)	Fails
Approach of Garon et al. [83] (20k training pairs) (<i>Asymmetric assumption</i>) re-iterate every 15 frames	41.60 ± 39.92	11.55 ± 15.58	-
Approach of Garon et al. [83] (20k training pairs) (<i>Asymmetric assumption</i>) re-iterate every time the tracker fails	30.17 ± 24.62	8.44 ± 9.43	15
Approach of Garon et al. [83] with Residual inter-layer connections and improved sampling/initialization/augmentation strategy (20k training pairs) (<i>Asymmetric assumption</i>) re-iterate every 15 frames	14.68 ± 10.46	6.61 ± 4.21	-
Approach of Garon et al. [83] with Residual inter-layer connections and improved sampling/initialization/augmentation strategy (20k training pairs) (<i>Asymmetric assumption</i>) re-iterate every time the tracker fails	17.14 ± 9.05	8.87 ± 5.18	5
Approach of Chapter 6 (20k training pairs) (<i>Asymmetric assumption</i>) re-iterate every 15 frames	11.05 ± 8.20	3.55 ± 2.27	-
Approach of Chapter 6 (20k training pairs) (<i>Asymmetric assumption</i>) re-iterate every time the tracker fails	12.78 ± 8.15	4.35 ± 3.16	1

Table 8.11: 3D Translational and Rotational Pose Errors and Fail count of the “Dragon” model, for the “Translation Only Interaction” scenario for the baseline approach of Garon et al. [83], without/with sampling/initialization/augmentation improvements we propose and the approach of Chapter 6.



Figure 8.33 (a) Demonstration of an “Observed” RGB frame of the “Translation Only Interaction” scenario with a render predicted pose of the model generated by the baseline tracker of Garon et al.[83].



Figure 8.34 (b) Demonstration of an “Observed” RGB frame of the “Translation Only Interaction” scenario with a render predicted pose of the model generated by the baseline tracker of Garon et al.[83] with Residual inter-layer connections and augmentation/initialization/sampling strategic improvements.



Figure 8.35 (c) Demonstration of an “Observed” RGB frame of the “Translation Only Interaction” scenario with a render predicted pose of the model generated by the proposed tracker.

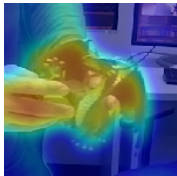


Figure 8.36 (a) Demonstration of the Foreground Extraction Attention map provided by the proposed tracker in the “Translation Only Interaction” scenario.

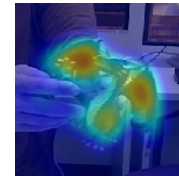


Figure 8.37 (b) Demonstration of the Occlusion Handling Attention map provided by the proposed tracker in the “Translation Only Interaction” scenario.

Qualitative analysis:

Baseline tracker of Garon et al[83](*subfig(a)*):

- The baseline tracker of Garon et al.[83] loses track of the dragon in high speeds and high (although far from abrupt) scale changes.
- In general, the free 3D translation challenge of this scenario causes the tracker to fail often, across all translational components (even at the easier x,y ones).

Baseline with Res-connections+Sampling/Augmentation amelioration(*subfig(b)*):

- Though improved in comparison to the baseline method, it fails in scales that are at the boundaries or outside of the training set (too big or too small). However, their potential absence in the training set is not an excuse for this failure, as the “Predicted” stream regularizes the scale of all input frames. So, increasing the scale ranges of the training set and retraining the tracker does not seem like a possible solution as it would be a clear attempt to overfit to the test set, as well as, a danger to sample poses too sparsely (in case of not increasing the train dataset size) and become the root cause of bigger errors in other pose configurations.
- At the only time the video sequence presents an (even) medium rotational movement, it is then that a small rotational drift appears. A drift that erases over time due to the feedback pose rendering.

Ours(*subfig(c)*):

- Although all pose estimations are clearly the most accurate across all previous architectures, too big and too small scales are still a weakness of the tracker. However, those errors are again smaller w.r.t. the previous architectural attempts.
- *Foreground Attention:* A characteristic behaviour of this module is that attention is, initially, focused on the dragon’s wings, then it shares it to them and the user’s hand and, finally, converges to the object again when the user catches it with both hands. A careful reader would have observed that we have given no supervising signal as far as it concerns the occluder’s behaviour for this module. So, it is more than logical to assume that these convolutional layers implicitly identify foreground extraction with motion segmentation, since they also focus on the user’s hand when it moves with keen speed. This observation gives us a hint, that we will thoroughly discuss in the next chapter, that an extra supervising signal of Optical Flow information would help the network to clear out these two concepts and, thus, distinguish the motion of the hand from the motion of the object in order to focus exclusively on the latter.
- *Occlusion Attention:* Here, the Attention shifts from one wing to both and then to the dragon’s belly, when the viewpoint changes. Of course, these are observations extra to the main one: the module achieves to focus only on the object’s features and the small part of it that is covered by the user’s fingers remains dark.

Error Plotting (left:re-iterate every 15 frames) and (right:re-iterate when the tracker fails):

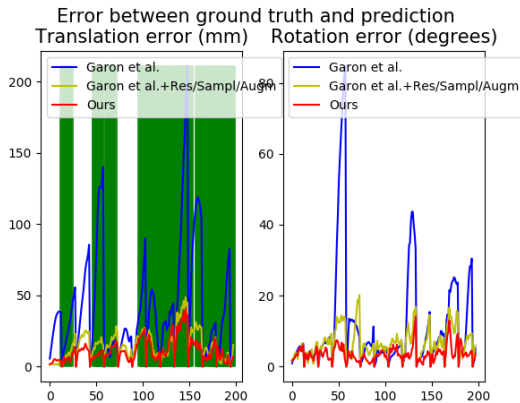


Figure 8.38 3D Translational and Rotational error metrics’ plot, for the “Translation Only Interaction” scenario of the Dragon 3D CAD model, in the case of tracker re-iteration every 15 frames.

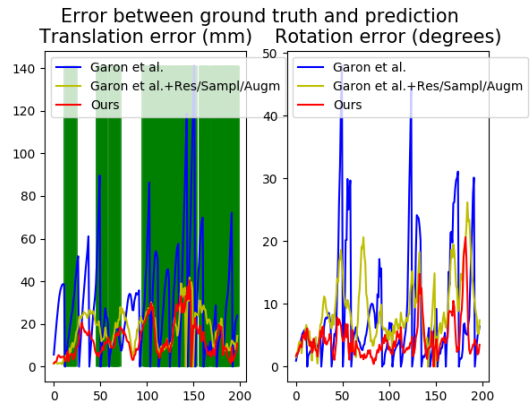


Figure 8.39 3D Translational and Rotational error metrics’ plot, for the “Translation Only Interaction” scenario of the Dragon 3D CAD model, in the case of tracker re-iteration only after a failure.

Free 3D Object Movement:Rotation Only

The second subcase of 3D movement is the opposite one: the object remains pretty much at the same position and is heavily rotated in complex trajectory patterns. This scenario is de facto more difficult than the previous one for two reasons:

- The first one is that rotations are an inherently more difficult task than translations for the network to estimate, as we have already ascertained in our Ablation Study of Chapter 7.
- The second one is that, in order to achieve all those complex rotational movement patterns and abrupt directional changes, the user is forced to intervene more forcefully to the object and occlude a more significant percentage of it for more time.

Architecture	Translational Error (mm)	Rotational Error (o)	Fails
Approach of Garon et al. [83] (20k training pairs) (<i>Asymmetric assumption</i>) re-iterate every 15 frames	23.86 ± 17.44	27.21 ± 22.40	-
Approach of Garon et al. [83] (20k training pairs) (<i>Asymmetric assumption</i>) re-iterate every time the tracker fails	21.22 ± 13.75	19.28 ± 15.17	15
Approach of Garon et al. [83] with Residual inter-layer connections and improved sampling/initialization/augmentation strategy (20k training pairs) (<i>Asymmetric assumption</i>) re-iterate every 15 frames	12.16 ± 8.94	12.76 ± 18.10	-
Approach of Garon et al. [83] with Residual inter-layer connections and improved sampling/initialization/augmentation strategy (20k training pairs) (<i>Asymmetric assumption</i>) re-iterate every time the tracker fails	14.63 ± 7.44	12.76 ± 10.48	5
Approach of Chapter 6 (20k training pairs) (<i>Asymmetric assumption</i>) re-iterate every 15 frames	9.37 ± 6.07	7.86 ± 6.69	-
Approach of Chapter 6 (20k training pairs) (<i>Asymmetric assumption</i>) re-iterate every time the tracker fails	11.88 ± 6.50	8.62 ± 6.88	2

Table 8.12: 3D Translational and Rotational Pose Errors and Fail count of the “Dragon” model, for the “Rotation Only Interaction” scenario for the baseline approach of Garon et al. [83], without/with sampling/initialization/augmentation improvements we propose and the approach of Chapter 6.

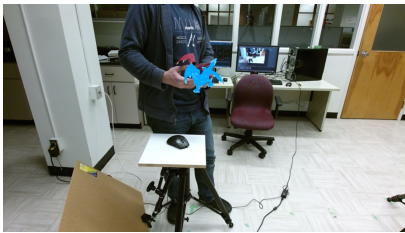


Figure 8.40 (a) Demonstration of an “Observed” RGB frame of the “Rotation Only Interaction” scenario with a render predicted pose of the model generated by the baseline tracker of Garon et al.[83].



Figure 8.41 (b) Demonstration of an “Observed” RGB frame of the “Rotation Only Interaction” scenario with a render predicted pose of the model generated by the baseline tracker of Garon et al.[83] with Residual inter-layer connections and augmentation/initialization/sampling strategic improvements.



Figure 8.42 (c) Demonstration of an “Observed” RGB frame of the “Rotation Only Interaction” scenario with a render predicted pose of the model generated by the proposed tracker.

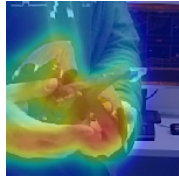


Figure 8.43 (a) Demonstration of the Foreground Extraction Attention map provided by the proposed tracker in the "Rotation Only Interaction" scenario.

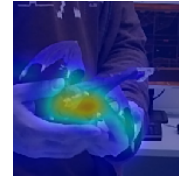


Figure 8.44 (b) Demonstration of the Occlusion Handling Attention map provided by the proposed tracker in the "Rotation Only Interaction" scenario.

Qualitative analysis:

Baseline tracker of Garon et al[83](*subfig(a)*):

- This baseline variation has a really hard time with complex rotations. It loses track even of the slowest ones.
- When no occlusions are present and the tracker has clear view of the most distinctive object features (i.e. the dragon's wings and tail) rotational errors are not significant.
- However, it tends to fail even when the slightest ambiguity, due to occlusion, is inserted in the sensor's field of view. This is especially obvious when it has to figure out the pose and the dragon's tail plays the first role in the scene. This is the case primarily due to its spurious shape which complicates the estimate, even for trained human users, and due to the fact that its configurations also produce self-occlusions that the tracker of Garon et al.[83] is unaware of, both implicitly and explicitly.

Baseline with Res-connections+Sampling/Augmentation amelioration(*subfig(b)*):

- Rotational errors make their appearance here mainly for quick motions.
- However, there are cases of tracking failure, where small rotational errors have accumulated over time and result in an irrecoverably erroneous estimate.

Ours(*subfig(c)*):

- Our proposed approach is the most robust one w.r.t. rotations, something completely normal as soon as one thinks about the sophisticated modeling of the rotational component of our tracking loss. The tail still causes problems but they are not so significant this time.
- We have to note here that our tracker is robust is even to very difficult cases that appear: for example when an appearance ambiguity and high self-occlusion co-exist, our method is capable of balancing out an, initially, small rotational error and avoid an extra failure.
- Indicative of the success of our method is that the Dragon completes a full 3D rotational maneuver without a fail!
- *Foreground Attention:* This Attention module separates the elements of the foreground (i.e. the features both of the object and the user) from those of the background. In particular, it mostly focuses on the object's body with the exemption of the following case: when the dragon is held upside down by the user, this module mostly enhances the head's features, as it is indicative of its pose. One very noticeable attribute of this module's behaviour is its persistence in focusing not only on the object, when it is moved at a higher speed, but also on the user's hands, even when they are occluding the object (frequently at high rates). This attribute is sometimes the root cause of losing track of certain rotations as they focalize on the moving hands, instead of the object of interest and insert extra ambiguities. For example, in frame number 144, where the user's hand is leaving the object, the Attention weight map peaks at the moving hand, instead of the object. This raises the value of the Occlusion-centric module, in such scenarios, as cascading the impact of the two Attention modules, even though their configuration is not hierarchical, tends to balance out ambiguities inserted by their individual function.

- *Occlusion Attention*: This module shifts focus from the dragon’s neck, to its belly, to its head (when the object is turned upside-down), to its wings (at first only on the single feather that is unoccluded by the dragon’s helical tail, and then on both of them, when the viewpoint alters) and, finally, results in its belly, when the object is clearly unoccluded. Next, we showcase two distinct snapshots when the impact of our approach is irrefutably validated:
 - At first, when the object is turned upside down, this module also chooses to enhance just the features of the head, and not to any of the parts below that are occluded by other parts of the same object, as it has implicitly learned to disentangle and avoid self-occlusions by itself.
 - Secondly, and maybe most noticeably, in frame number 109, where the object is fully occluded by the user’s hand, our proposed tracker does not produce very large errors. We appoint this success to the inclusion of the full occlusion scenario in our data augmentation procedure, along with the integration of the Occlusion Attentional module in our architecture, which has allowed the model to learn this pattern as well, and figure out the proper way to handle it (e.g. implicitly, rely, almost exclusively, on the prediction of the previous frame, instead of the current one, as the object is mostly missing from it). And we can confidently say that both parts of this combination is vital, as just the data augmentation improvement was presented in the previous architectural amelioration (the one which had Dense-Res connection substitution at its core) and it had not provide us with so impressive results in terms of handling complete occlusion of the object and still accurately estimating its rotation.

Error Plotting (left:re-iterate every 15 frames) and (right:re-iterate when the tracker fails):

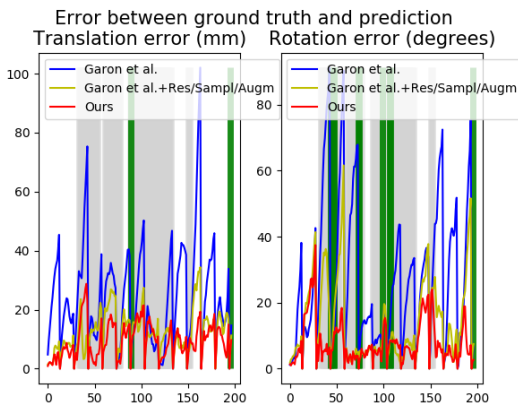


Figure 8.45 3D Translational and Rotational error metrics’ plot, for the “Rotation Only Interaction” scenario of the Dragon 3D CAD model, in the case of tracker re-iteration every 15 frames.

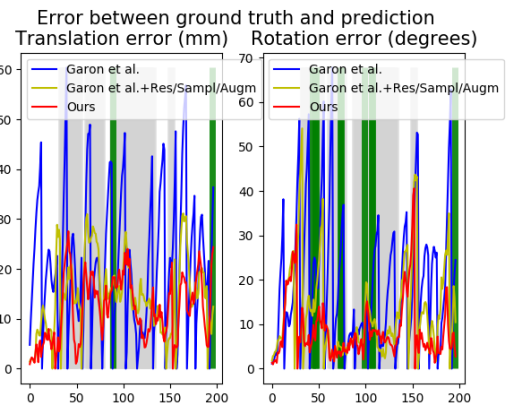


Figure 8.46 3D Translational and Rotational error metrics’ plot, for the “Rotation Only Interaction” scenario of the Dragon 3D CAD model, in the case of tracker re-iteration only after a failure.

Free 3D Object Movement: Translation and Rotation

Now, we proceed to mix up 3D Translational and Rotational movements and, thus, test our approach in Full 3D motions. Speed has also risen, in this case, from slow to medium. The user is careful, though, not to produce severe occlusions as he moves the object. Including adversarial and dynamic consecutive occlusion patterns in our testing dataset is a very challenging case that will be examined last, in the following subsection discussion.

Architecture	Translational Error (mm)	Rotational Error (o)	Fails
Approach of Garon et al. [83] (20k training pairs) (<i>Asymmetric assumption</i>) re-iterate every 15 frames	35.23 ± 31.97	34.98 ± 29.46	-
Approach of Garon et al. [83] (20k training pairs) (<i>Asymmetric assumption</i>) re-iterate every time the tracker fails	23.70 ± 21.37	22.97 ± 18.06	18
Approach of Garon et al. [83] with Residual inter-layer connections and improved sampling/initialization/augmentation strategy (20k training pairs) (<i>Asymmetric assumption</i>) re-iterate every 15 frames	9.31 ± 5.92	12.78 ± 18.61	-
Approach of Garon et al. [83] with Residual inter-layer connections and improved sampling/initialization/augmentation strategy (20k training pairs) (<i>Asymmetric assumption</i>) re-iterate every time the tracker fails	13.73 ± 8.88	11.37 ± 8.75	6
Approach of Chapter 6 (20k training pairs) (<i>Asymmetric assumption</i>) re-iterate every 15 frames	10.31 ± 8.66	6.40 ± 4.52	-
Approach of Chapter 6 (20k training pairs) (<i>Asymmetric assumption</i>) re-iterate every time the tracker fails	12.22 ± 9.14	7.78 ± 6.69	1

Table 8.13: 3D Translational and Rotational Pose Errors and Fail count of the “Dragon” model, for the “Full Interaction” scenario for the baseline approach of Garon et al. [83], without/with sampling/initialization/augmentation improvements we propose and the approach of Chapter 6.

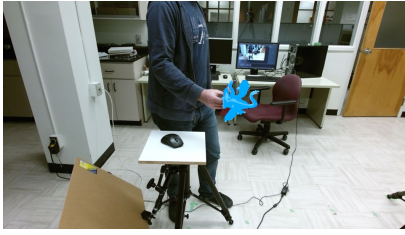


Figure 8.47 (a) Demonstration of an “Observed” RGB frame of the “Full Interaction” scenario with a render predicted pose of the model generated by the baseline tracker of Garon et al.[83].



Figure 8.48 (b) Demonstration of an “Observed” RGB frame of the "Full Interaction" scenario with a render predicted pose of the model generated by the baseline tracker of Garon et al.[83] with Residual inter-layer connections and augmentation/initialization/sampling strategic improvements.



Figure 8.49 (c) Demonstration of an “Observed” RGB frame of the "Full Interaction" scenario with a render predicted pose of the model generated by the proposed my tracker.

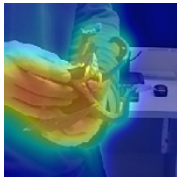


Figure 8.50 (a) Demonstration of the Foreground Extraction Attention map provided by the proposed tracker in the "Full Interaction" scenario.

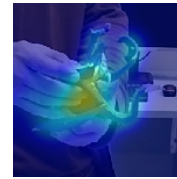


Figure 8.51 (b) Demonstration of the Occlusion Handling Attention map provided by the proposed tracker in the “Full Interaction” scenario.

Qualitative analysis:

Baseline tracker of Garon et al[83](*subfig(a)*):

- Firstly, as for the translational component of motion, the approach of Garon et al.[83] tracks the object’s pose adequately well, producing only infinitesimal errors that get zeroed-out by feedback.
- Now, as far as it concerns rotations, it fails emphatically even in small speed and medium occlusions or just medium speeds.

Baseline with Res-connections+Sampling/Augmentation amelioration(*subfig(b)*):

- This, first, architectural improvement is still very effective in tracking the object’s 3D translation, even for long-term translational trajectories.
- Now, as for rotations, their estimates are more accurate than before, as it is more faithful, more responsive to small errors and with reduced jitter. However it still presents genuine problems in high rotational speeds and scale changes.

Ours(*subfig(c)*):

- Our approach accumulates tracking errors much slower. Translational errors have been practically zeroed out and overall tracking fails, occasioned by this component, are really rare.
- The Rotational estimation has been much improved by our full architecture. We are more responsive to pose feedback, almost never lose track of the object’s rotation in medium speeds and much of the generated drift, created mostly in difficult scenarios with incidental translational and rotational variation, at large scales, is countered and not allowed to lead to easy fails. Specifically, even when the tracker loses track of the rotation and the object’s speed reduces recently, has the chance to catch it again. Nevertheless, as we can also see at the Table below, tracking fails have not diminished completely, a sign that extra work should be done.
- To give credit where credit is due, even if our tracker presents this image of general success, we have not managed to make it completely invariant to appearance ambiguity and disentangle pose predictions from difficult views that contain confusing visual features.

- **Besides**, we, also, observe that no matter how refined the tracking loss is, how inclusive is the domain randomization and adaptation performed at the data augmentation level and no matter how qualified inputs the rendering software produces, the tracker is weak in keeping account of the object’s pose in the long run. We may reduce pose drifts and slow down the error accumulation process, but the tracker is designed to be unaware of long-term dependencies and seems convicted in failing after some time has passed. It has not seen trajectories longer than two frames during training and has not acquired the long-term planning intuition that evolution has equipped people with. To this end, an idea that we explore in one of the following subsections is the enhancement of algorithm with such samples that smooth out the tracker’s predictions and generalizes its understanding at the trajectory, instead of the frame pair, level.
- *Foreground Attention*: Here that occlusions are mostly absent, this module’s peaks are presented the most dominant across the two parallel attributes. Here, both of the user’s hands are mostly stuck with the object of interest and thus do not produce extra unwanted ambiguities. In general, the module distinguishes the object very sharply from its background (the rest of the room and the user’s torso) and it mostly peaks near the dragon’s belly.
- *Occlusion Attention*: This module’s Attention weight map is more frequently spread out along the object’s surface and provides no extra meaningful information w.r.t. its Foreground extraction counterpart. It generally follows it in enhancing the same object regions.

Error Plotting (left:re-iterate every 15 frames) and (right:re-iterate when the tracker fails):

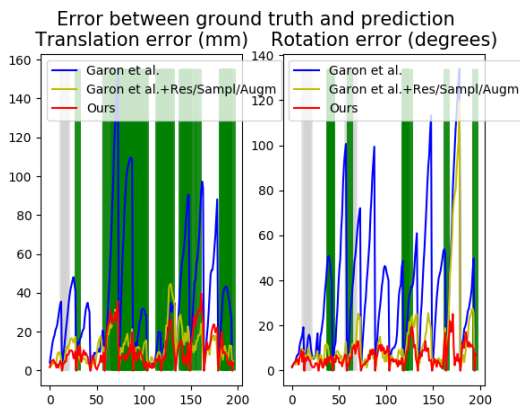


Figure 8.52 3D Translational and Rotational error metrics’ plot, for the “Full Interaction” scenario of the Dragon 3D CAD model, in the case of tracker re-iteration every 15 frames.

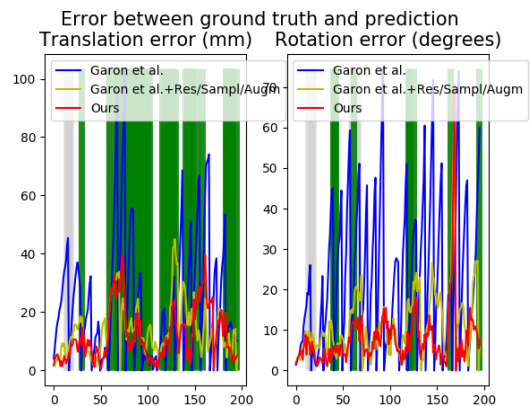


Figure 8.53 3D Translational and Rotational error metrics’ plot, for the “Full Interaction” scenario of the Dragon 3D CAD model, in the case of tracker re-iteration only after a failure.

Free 3D Object Pose Tracking - the “Hard Interaction” scenario:

Last, but certainly not least, we discuss the impact of the architectural evolution of this work in the most challenging scenario of them all. Here, the user is left free to move the object in all configurations and speeds possible and he intentionally produces dynamic occlusion patterns that vary greatly in scale and may change abruptly. It is the scenario we had, primarily, in mind when we designed the method as it consists the most intuitive use case for an object pose tracker. Additionally, in the “Hard Interaction” scenario for the “Dragon” object, the user performs some of the fastest and most abrupt maneuvers in the whole dataset.

Architecture	Translational Error (mm)	Rotational Error (o)	Fails
Approach of Garon et al. [83] (20k training pairs) (<i>Asymmetric assumption</i>) re-iterate every 15 frames	35.18 ± 25.78	38.33 ± 34.84	-
Approach of Garon et al. [83] (20k training pairs) (<i>Asymmetric assumption</i>) re-iterate every time the tracker fails	24.89 ± 17.12	24.56 ± 18.52	18
Approach of Garon et al. [83] with Residual inter-layer connections and improved sampling/initialization/augmentation strategy (20k training pairs) (<i>Asymmetric assumption</i>) re-iterate every 15 frames	13.27 ± 10.46	16.34 ± 20.64	-
Approach of Garon et al. [83] with Residual inter-layer connections and improved sampling/initialization/augmentation strategy (20k training pairs) (<i>Asymmetric assumption</i>) re-iterate every time the tracker fails	14.40 ± 9.82	14.14 ± 10.78	7
Approach of Chapter 6 (20k training pairs) (<i>Asymmetric assumption</i>) re-iterate every 15 frames	11.63 ± 8.79	8.31 ± 6.76	-
Approach of Chapter 6 (20k training pairs) (<i>Asymmetric assumption</i>) re-iterate every time the tracker fails	13.28 ± 8.49	9.06 ± 7.25	2

Table 8.14: 3D Translational and Rotational Pose Errors and Fail count of the “Dragon” model, for the “Hard Interaction” scenario for the baseline approach of Garon et al. [83], without/with sampling/initialization/augmentation improvements we propose and the approach of Chapter 6.



Figure 8.54 (a) Demonstration of an “Observed” RGB frame of the “Hard Interaction” scenario with a render predicted pose of the model generated by the baseline tracker of Garon et al.[83].



Figure 8.55 (b) Demonstration of an “Observed” RGB frame of the “Hard Interaction” scenario with a render predicted pose of the model generated by the baseline tracker of Garon et al.[83] with Residual inter-layer connections and augmentation/initialization/sampling strategic improvements.

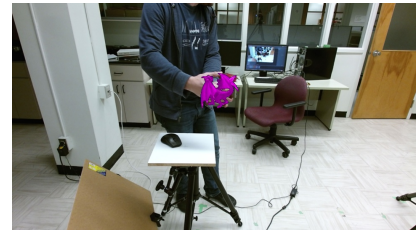


Figure 8.56 (c) Demonstration of an “Observed” RGB frame of the “Hard Interaction” scenario with a render predicted pose of the model generated by the proposed tracker.

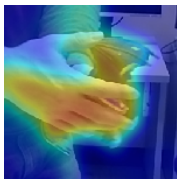


Figure 8.57 (a) Demonstration of the Foreground Extraction Attention map provided by the proposed tracker in the “Hard Interaction” scenario.

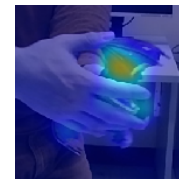


Figure 8.58 (b) Demonstration of the Occlusion Handling Attention map provided by the proposed tracker in the “Hard Interaction” scenario.

Qualitative analysis:

Baseline tracker of Garon et al[83](*subfig(a)*):

- The approach of Garon et al. [83] presents genuine weaknesses in almost all challenging scenarios of this subcase:
 - As soon as an increased occlusion percentage appears, the tracker fails irrecoverably.
 - As soon as the motion speed increases, the tracker fails irrecoverably.
 - As soon as we reach extreme scales and/or abrupt pose changes, the tracker fails irrecoverably.

Baseline with Res-connections+Sampling/Augmentation amelioration(*subfig(b)*):

- Similarly to the previous subcases, this improvement reduces pose jitter and the tracker’s stability. However, its response to dynamic occlusions (both of low and high percentage) and high speeds remains disappointing.

Ours(*subfig(c)*):

- In general, our approach reduces mean errors by about 40.5% for translation and 57.7% for rotation w.r.t. the baseline approach of [83], for the “Dragon” object model. When the object is not occluded, the tracker focuses mostly on its 3D center, implicitly realizing in this way that this is the main 3D point of tracking interest. When the user’s hand occludes parts of the dragon, the attention shifts to its body parts of interest that stand out of the grip, like its neck, wings or tail. A different parts is lightened up each time, the one that mostly stands out from the user’s grip. Overall, the effectiveness of our method is demonstrated by the fact that, while [83] (even with the gradual improvements we have applied to the intra-connectivity and Data Augmentation) keeps track only of the object’s 3D position under medium or extreme occlusions, our improved version extends this property to 3D rotations as well. In particular, even in the most challenging scenarios, where all difficulties are combined and the object, e.g. performs a very long, quick and complex maneuver that includes translations, rotations and occlusions of variable patterns, the tracker makes infinitesimal mistakes, under 10° at average.
- The most enlightening example is the 52th frame and its neighbouring ones. There, in the middle of a prolonged rotational trajectory which covers its whole domain, the object is dynamically occluded at a degree of 80% or more. However, due to the proper combination of data preparation, rotational modeling and occlusion handling, the tracker suffices to only few visual clues and correctly guesses the dragon’s future pose, leaving no more than a small rotational error range, which is zeroed out when visibility is chiefly restored, saving a previous failing of the tracker.

We assess this particular frame subsequence as one of the most valuable of the test dataset, w.r.t. the comprehension of the method’s success and, thus, we report that parallel Attention maps’ behaviour on it:

- *Foreground Attention in frame number 52:*

This, first, parallel, Attentional module lights up a general blurred region around the dragon and the hands hiding it from the sensor’s field of view. Its peaks are rather spread out and the regions of interest are rather blurred out. This is logical as the interest has mostly shifted to the “Occlusion” module, as this is, for now, the most significant error source.

- *Occlusion Attention in frame number 52:*

- This “Occlusion” module sieves the remaining object parts from the user’s occluding hands (in this case one of the dragon’s wings, as well as, its head, partially). Its peaks are more edgy than the corresponding peaks of its “Foreground” counterpart.

- One important feature that is revealed when we closely examine the behaviour of this module is that, without the slightest idea of temporal smoothing out or modeling of temporal consistency, it has implicitly learn medium temporal correlation. This becomes obvious as the deoccluded patterns smoothly transition from one object part of interest to the next and do not jump to the region of interest that seems more important at an individual level. Indicatively, one of the few tracking fails that occur using our approach, happens when this implicit temporal consistency breaks and one of the dragon’s wings abruptly reappears in the scene causing this Attentional module to enhance it and, probably, confuse the tracker.

Error Plotting (left:re-iterate every 15 frames) and (right:re-iterate when the tracker fails):

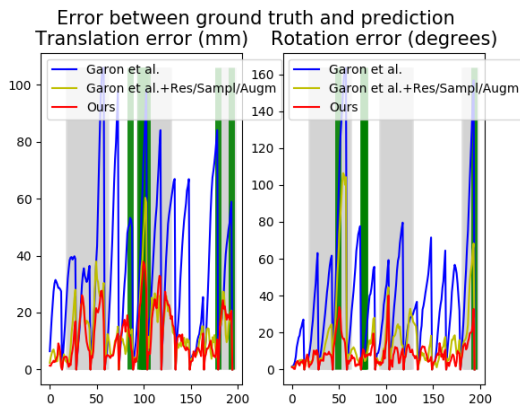


Figure 8.59 3D Translational and Rotational error metrics’ plot, for the “Hard Interaction” scenario of the Dragon 3D CAD model, in the case of tracker re-iteration every 15 frames.

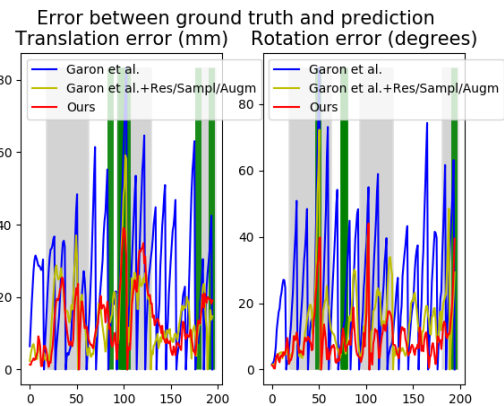
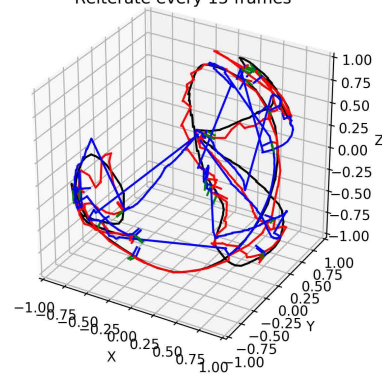


Figure 8.60 3D Translational and Rotational error metrics’ plot, for the “Hard Interaction” scenario of the Dragon 3D CAD model, in the case of tracker re-iteration only after a failure.

6D Temporal Pose Tracking Visualization
Dragon --- Hard Interaction scenario
Reiterate every 15 frames



6D Temporal Pose Tracking Visualization
Dragon --- Hard Interaction scenario
Reiterate at every tracking failure

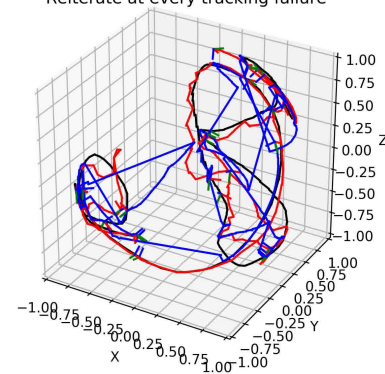


Figure 8.61: The 6D Pose Trajectories of the “Dragon” object model, in time, for the “Hard Interaction” scenario. It is evident, in both cases, from this perspective as well, that our proposed approach outperforms the SoA of Garon et al.[83] by far, as it produces smaller errors and fewer failures. All length units are in mm and all rotations in degrees.

8.4.3 the Cookie Jar model: the symmetric case

For the special case of rotorefective symmetry, we also report our results without/with accounting for the object’s symmetry axis(-es) in formulating our rotation loss. What is, in fact, special about rotational symmetry around the object-centric z-axis is that it adds an extra adversity in accurately estimating the object pose at any given moment: the depth modality that insinuates its 3D shape information inserts extra rotational jitter as the exact degree that the object’s symmetry influences the

estimate remains obscure. One would counter, though, as we have already seen, that the accompanying RGB cue irrefutably sorts out this discrepancy. This is what we examine in the second part of every one of the following evaluations, as well as the impact of different modeling selections of this symmetry, when the RGB modality is proven unadequate in its purpose.

We estimate the Inertia Tensor of the ‘‘CookieJar’’ 3D object model:

$$\Lambda^{(CookieJar)} = \begin{bmatrix} 0.0030 & 0 & -0.0003 \\ 0 & 0.0030 & -0.0004 \\ -0.0003 & -0.0004 & 0.0055 \end{bmatrix} \quad (8.7)$$

and after the Gramm-Schmidt orthonormalization:

$$\Lambda_{G.S.}^{(CookieJar)} = \begin{bmatrix} 0.9999 & 0.0012 & 0.0110 \\ -0.0013 & 0.9999 & 0.0137 \\ -0.0110 & -0.0137 & 0.9998 \end{bmatrix} \quad (8.8)$$

We observe that $\Lambda^{(CookieJar)}$ is almost diagonal, as its diagonal elements are at least one order of magnitude larger than the non-diagonal ones and that its axis with the maximum inertia is its z-axis.

The converged Multi-Task Loss weights are:

Multi-Task Weight Name	Multi-Task Weight Value
$\mathbf{w}_1^{(s)} = \mathbf{e}^{-s_1}$	0.0226
$\mathbf{w}_2^{(s)} = \mathbf{e}^{-s_2}$	0.4798
$\mathbf{w}_3^{(s)} = \mathbf{e}^{-s_3}$	1.7224
$\mathbf{w}_1^{(v)} = \mathbf{e}^{-v_1}$	0.0956
$\mathbf{w}_2^{(v)} = \mathbf{e}^{-v_2}$	0.05393

Experimentation with different ways to handle object symmetries:

Due to the fact that the minimization of the Tracking Loss w.r.t. the symmetry matrix \hat{G}^* (see Br egier et al. [11]) does not explicitly impose a global-solution constraint, it gives birth to a new degree of freedom in selecting the optimal way to regress the symmetric parameter(s). In this wok, we try the following different approaches:

- **Unique Symmetry parameter optimum:** Learning a unique symmetry parameter over all possible pose changes in the training set and keeping it frozen during inference.

$$L_{Track}^*(\Delta\mathbb{P}_{GT}, \Delta\hat{\mathbb{P}}) = e^{-v_1} \cdot MSE(\Delta\hat{\mathbf{t}}, \Delta\mathbf{t}_{GT}) + e^{-v_2} \cdot \arccos\left(\frac{\text{tr}\left(\left(\Delta\hat{R} \cdot \hat{G}(z^*) \Lambda_{(G.S.)}\right)^T (R_{GT} \cdot \Lambda_{(G.S.)})\right)}{2}\right) + \dots, \quad (8.9)$$

where z^* is a frozen, uniquely learned rotational symmetry parameter: $z^* = \arg \min_z L_{Track}(\Delta\mathbb{P}_{GT}, \Delta\hat{\mathbb{P}}, z)$.

- **Batch of Symmetry parameter optima:** Learning a batch of symmetry parameters and taking their average during inference.

$$L_{Track}^*(\Delta\mathbb{P}_{GT}, \Delta\hat{\mathbb{P}}) = e^{-v_1} \cdot MSE(\Delta\hat{\mathbf{t}}, \Delta\mathbf{t}_{GT}) + e^{-v_2} \cdot \arccos\left(\frac{\text{tr}\left(\left(\Delta\hat{R} \cdot \hat{G}(\bar{z}^*) \cdot \Lambda_{(G.S.)}\right)^T (R_{GT} \cdot \Lambda_{(G.S.)})\right)}{2}\right) + \dots, \quad (8.10)$$

where \bar{z}^* is the mean of a batch of frozen, learnable rotational symmetry parameters:

$$\bar{z}^* = \frac{1}{B} \sum_{b=1}^B z_b^*, \quad z_b^* = \arg \min_z L_{Track}(\Delta\mathbb{P}_{GT}, \Delta\hat{\mathbb{P}}, z_b). \quad (8.11)$$

- **Regression of Symmetry parameter:** Regressing a different symmetry parameter per pose pair.

$$L_{Track}^*(\Delta\mathbb{P}_{GT}, \Delta\hat{\mathbb{P}}z) = e^{-v_1} \cdot MSE(\Delta\hat{\mathbf{t}}, \Delta\mathbf{t}_{GT}) + e^{-v_2} \cdot \arccos\left(\frac{\text{tr}\left((\Delta\hat{R}\hat{G}(\hat{z})\Lambda_{(G.S.)})^T (R_{GT}\Lambda_{(G.S.)})\right)}{2}\right) + \dots, \quad (8.12)$$

where \hat{z} is the per-train pose regressed, learnable rotational symmetry parameter. An extra single-output, Fully Connected layer is employed over the latent convolutional representation and its output is constrained by a tanh activation function and then multiplied by π to ensure that the resulted parameter will lie in the interval $[-\pi, \pi]$. The difference with the previous two alternative versions is that, here, the rotational symmetry parameter is not frozen in the inference stage, but is regressed along with the other 6 rotational parameters for each input combination.

$$\hat{z} = \pi * \tanh(FC(f)), f \in \mathbb{R}^{500}. \quad (8.13)$$

- **Per-pose pair Optimal Selection out of an approximately Uniform batch of Continuously Rotational Symmetry parameters:** Selecting the optimal symmetry parameter from the aforementioned batch using an appropriate classification layer while encouraging the values of this batch to be as uniform as possible, at the same time.

Again,

$$L_{Track}^*(\Delta\mathbb{P}_{GT}, \Delta\hat{\mathbb{P}}) = e^{-v_1} \cdot MSE(\Delta\hat{\mathbf{t}}, \Delta\mathbf{t}_{GT}) + e^{-v_2} \cdot \arccos\left(\frac{\text{tr}\left((\Delta\hat{R} \cdot \hat{G}(\hat{z}^*) \cdot \Lambda_{(G.S.)})^T (R_{GT} \cdot \Lambda_{(G.S.)})\right)}{2}\right) + \dots, \quad (8.14)$$

and the overall loss function augments the one of eq.6.35 as follows:

$$Loss^{(Symm)} = Loss + e^{(-s_4)} \left(\frac{1}{B} \sum_{b=1}^B \frac{1}{\xi_b}\right) + s_4, \text{ with} \quad (8.15)$$

$$\xi_b = \frac{1}{B_2(B_2 - 1)} \sum_{j=1}^{B_2} \sum_{k \neq j} d_{Rot}^{(Geod)}(\hat{G}_k, \hat{G}_j), \quad (8.16)$$

This, extra, penalty term, added to the overall loss function, encourages the symmetry triplets to be as uniform as possible.

Indicatively, the converged Multi-Task Loss weights for this symmetric case are:

Multi-Task Weight Name	Multi-Task Weight Value
$\mathbf{w}_1^{(s)} = \mathbf{e}^{-s_1}$	0.0919
$\mathbf{w}_2^{(s)} = \mathbf{e}^{-s_2}$	0.4956
$\mathbf{w}_3^{(s)} = \mathbf{e}^{-s_3}$	1.8410
$\mathbf{w}_4^{(s)} = \mathbf{e}^{-s_4}$	0.0182
$\mathbf{w}_1^{(v)} = \mathbf{e}^{-v_1}$	0.0581
$\mathbf{w}_2^{(v)} = \mathbf{e}^{-v_2}$	0.0878

The last symmetry handling approach that we, ultimately, propose, results in learning a batch of continuously rotational symmetry parameters **with a range of about** $[-20^\circ, 20^\circ]$ and a distribution proximal to the uniform. Here, we randomly sample and showcase 10 of them (as we have already seen the whole batch is of size $B_2=64$):

Table 8.15: A random subset of the overall batch of $B_2 = 64$ learnable rotational symmetry parameters.

6.58° | -1.34° | -4.34° | 16.16° | -0.76° | -13.70° | -19.32° | 2.32° | -15.78° | 14.31°

Static Object Pose Tracking on a 2D turntable:

Subcase 1: Object Near the camera

As before, for our first testing subcase, the object is placed at its most convenient position w.r.t. to the sensor: steady and close to it.

Architecture	Translational Error (mm)	Rotational Error (o)	Fails
Approach of Garon et al. [83] (20k training pairs) (<i>Symmetric assumption</i>) re-iterate every 15 frames	31.86 ± 15.86	27.98 ± 13.90	-
Approach of Garon et al. [83] (20k training pairs) (<i>Symmetric assumption</i>) re-iterate every time the tracker fails	26.64 ± 14.35	20.45 ± 11.03	14
Approach of Garon et al. [83] with Residual inter-layer connctions and improved sampling/initialization/augmentation strategy (20k training pairs) (<i>Symmetric assumption</i>) re-iterate every 15 frames	8.55 ± 4.78	3.20 ± 1.89	-
Approach of Garon et al. [83] with Residual inter-layer connctions and improved sampling/initialization/augmentation strategy (20k training pairs) (<i>Symmetric assumption</i>) re-iterate every time the tracker fails	17.95 ± 9.33	5.89 ± 3.61	2
Approach of Chapter 6 (20k training pairs) (<i>Asymmetric assumption</i>) re-iterate every 15 frames	13.39 ± 5.97	8.04 ± 5.20	-
Approach of Chapter 6 (20k training pairs) (<i>Asymmetric assumption</i>) re-iterate every time the tracker fails	10.18 ± 4.90	6.38 ± 5.86	0
Approach of Chapter 6 (20k training pairs) (<i>Unique Parameter Symmetric assumption</i>) re-iterate every 15 frames	12.31 ± 15.93	5.97 ± 3.13	-
Approach of Chapter 6 (20k training pairs) (<i>Unique Parameter Symmetric assumption</i>) re-iterate every time the tracker fails	14.86 ± 16.34	7.97 ± 4.60	0
Approach of Chapter 6 (20k training pairs) (<i>Batch of Parameters Symmetric assumption</i>) re-iterate every 15 frames	11.54 ± 12.98	5.32 ± 3.20	-
Approach of Chapter 6 (20k training pairs) (<i>Batch of Parameters Symmetric assumption</i>) re-iterate every time the tracker fails	13.21 ± 17.89	8.22 ± 4.93	0
Approach of Chapter 6 (20k training pairs) (<i>Parameter Regression Symmetric assumption</i>) re-iterate every 15 frames	17.53 ± 16.76	7.94 ± 5.89	-
Approach of Chapter 6 (20k training pairs) (<i>Parameter Regression Symmetric assumption</i>) re-iterate every time the tracker fails	19.82 ± 16.85	8.74 ± 5.52	0
Approach of Chapter 6 (20k training pairs) (<i>Optimal Selection out of a uniform Batch of Parameters Symmetric assumption</i>) re-iterate every 15 frames	8.43 ± 4.67	5.02 ± 3.15	-
Approach of Chapter 6 (20k training pairs) (<i>Optimal Selection out of a uniform Batch of Parameters Symmetric assumption</i>) re-iterate every time the tracker fails	12.88 ± 15.34	6.33 ± 5.37	0

Table 8.16: 3D Translational and Rotational Pose Errors and Fail count of the “Cookie Jar” model, for the “Stability near” scenario, for the baseline approach of Garon et al. [83], without/with sampling/ initialization/augmentation improvements we propose and the approach of Chapter 6. and its symmetry handling alternatives.

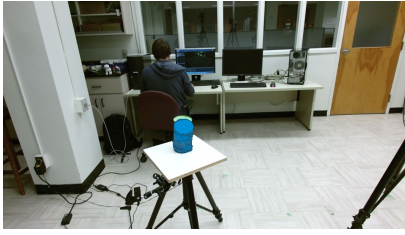


Figure 8.62 (a) Demonstration of an “Observed” RGB frame of the “Stability near” scenario with a rendered predicted pose of the “Cookie Jar” model generated by the baseline tracker of Garon et al. [83].



Figure 8.63 (b) Demonstration of an “Observed” RGB frame of the “Stability near” scenario with a rendered predicted pose of the “Cookie Jar” model generated by the baseline tracker of Garon et al. [83] with Residual inter-layer connections and sampling/initialization/augmentation improvements.

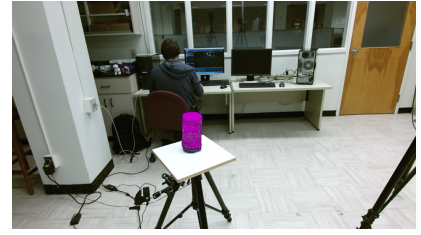


Figure 8.64 (c) Demonstration of an “Observed” RGB frame of the “Stability near” scenario with a rendered predicted pose of the “Cookie Jar” model generated by proposed tracker of this work.



Figure 8.65 (d) Demonstration of an “Observed” RGB frame of the “Stability near” scenario with a rendered predicted pose of the “Cookie Jar” model generated by proposed tracker, with the special case of unique rotational symmetry parameter, that remains frozen after training, of this work.



Figure 8.66 (e) Demonstration of an “Observed” RGB frame of the “Stability near” scenario with a rendered predicted pose of the “Cookie Jar” model generated by proposed tracker, with a learnable batch of rotational parameters, of this work.



Figure 8.67 (f) Demonstration of the Occlusion Handling Attention map provided by the proposed tracker, with a rotational parameter regression, in the “Stability near” scenario and tested on the “Cookie Jar” 3D model.



Figure 8.68 (g) Demonstration of the Occlusion Handling Attention map provided by the proposed tracker, with a per-pose pair optimal selection of the symmetry parameter, out of a nearly Uniform batch of Continuously Rotational Symmetry parameters, in the “Stability near” scenario and tested on the “Cookie Jar” 3D model.



Figure 8.69 (a) Demonstration of the Foreground Extraction Attention map of the “Cookie Jar” 3D model, provided by the proposed tracker in the “Stability near” scenario, without accounting for symmetries.



Figure 8.70 (b) Demonstration of the Occlusion Handling Attention map of the “Cookie Jar” 3D model, provided by the proposed tracker in the “Stability near” scenario, without accounting for symmetries.



Figure 8.71 (c) Demonstration of the Foreground Extraction Attention map of the “Cookie Jar” 3D model, provided by the proposed tracker, with a unique, frozen, learnable symmetrical rotational parameter, in the “Stability near” scenario.



Figure 8.72 (d) Demonstration of the Occlusion Handling Attention map of the “Cookie Jar” 3D model, provided by the proposed tracker, with a unique, frozen, learnable symmetrical rotational parameter, in the “Stability near” scenario.



Figure 8.73 (e) Demonstration of the Foreground Extraction Attention map of the “Cookie Jar” 3D model, provided by the proposed tracker, with a learnable batch of symmetrical rotational parameters, in the “Stability near” scenario.



Figure 8.74 (f) Demonstration of the Occlusion Handling Attention map of the “Cookie Jar” 3D model, provided by the proposed tracker, with the mean of a batch of frozen, learnable symmetrical rotational parameters, in the “Stability near” scenario.



Figure 8.75 (g) Demonstration of the Foreground Extraction Attention map of the “Cookie Jar” 3D model, provided by the proposed tracker, with a per-pose regressed learnable symmetrical rotational parameter, in the “Stability near” scenario.



Figure 8.76 (h) Demonstration of the Occlusion Handling Attention map of the “Cookie Jar” 3D model, provided by the proposed tracker, with a per-pose regressed learnable symmetrical rotational parameter, in the “Stability near” scenario.

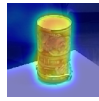


Figure 8.77 (i) Demonstration of the Foreground Extraction Attention map of the “Cookie Jar” 3D model, provided by the proposed tracker, with a per-pose pair optimal selection of the symmetry parameter, out of a nearly Uniform batch of Continuously Rotational Symmetry parameters, in the “Stability near” scenario.



Figure 8.78 (j) Demonstration of the Occlusion Handling Attention map of the “Cookie Jar” 3D model, provided by the proposed tracker, with a per-pose pair optimal selection of the symmetry parameter, out of a nearly Uniform batch of Continuously Rotational Symmetry parameters, in the “Stability near” scenario.

Qualitative analysis:

Baseline tracker of Garon et al[83](*subfig(a)*):

- The baseline tracker, proposed by Garon et. al[83], emits no translational errors in neither of the x,y-dimensions. The same happens for the z-direction, with the small exception of some scale ambiguity that is, however, countered out by the rendered feedback stream.
- As for rotations, the tracker makes small rotational drifts that may, though, accumulate over time and cause limited failures and force manual reinitialization of the tracker.

Baseline with Res-connections+Sampling/Augmentation amelioration(*subfig(b)*):

- The connectivity, sampling and augmentational improvements that we induce in our Neural tracking architecture, classically, have the effect of more accurate estimations, increased robustness and jitter resistance. Pose errors have practically been vanished and, even in case small errors occur, drift is

not let to accumulate and the feedback rendering stream returns the pose estimation back in the reasonable range.

Ours(*subfig(c)*):

- Since most problems are solved out by the previous methodological ameliorations, we have few remarks to make here. The estimation remains fairly robust using our proposed tracker and the errors of the rotational components are slightly reduced.
- As for the two parallel Attentional modules, they have more or less the same behaviour since the object is not occluded nor faces severe background cluttering. The Attention is concentrated on the object’s surface, it is mostly uniform in both weight maps and the only appearance patterns that slightly stand out are the esoteric ones that limit caricatural details of the Cookie Jar’s ornament.

Ours+Unique,frozen, learnable continuously rotational Symmetry parameter (*subfig(d)*):

Since the object is static on the table, the uniquely-imposed rotational symmetry parameter yields the minimum pose error w.r.t. both the translational and rotational metric, across all other symmetry learning efforts.

Ours+Mean of a Batch of frozen,learnable continuously rotational Symmetry parameters (*subfig(e)*):

This, second, approach for learning the continuously rotational symmetry z-parameter tries to balance out the jitter that a full regression (see below) would cause and the inflexibility of a unique-solution constrain (see above). In this scenario, this approach seems to be working quite reliably, but the, first, unique-solution attempt is still ahead of it.

Ours+per Viewpoint Regression of Symmetry parameter (*subfig(f)*):

This approach is the most flexible of them all and leaves the parameter(s) free to take any value in the accepted range for each pose pair. However, this property makes it the most unstable, as well, since test RGB-D pairs belong in the same domain distribution as those of the training dataset of the tracker, but are not identical to them (both in terms of appearance and absolute depth distance, as well as accurately defined pose difference), something that adds more noise to the overall rotational (and eventually translational, due to the rendering-based feedback) estimation. This becomes evident after a careful observation of the figure pairs below.

Ours+Optimal Selection out of a Batch of frozen,learnable, continuously rotational Symmetry parameters (*subfig(g)*):

In this work, we propose this symmetry parameter learning approach as the overall optimal one, since its performance is one of the best or the best, straightforwardly, across all scenarios. Indeed, we can see that it renders the second lowest pose error (for both metrics). However, the simplicity of the task and the imperfections of the task at hand learning procedure make the extra liberty it offers for the parameter values to be redundant and add small amounts of unnecessary pose jitter. As a result, the uniquely-defined parameter solution is the winner for this, particular scenario, by a slight amount.

Error Plotting (left:re-iterate every 15 frames) and (right:re-iterate when the tracker fails):

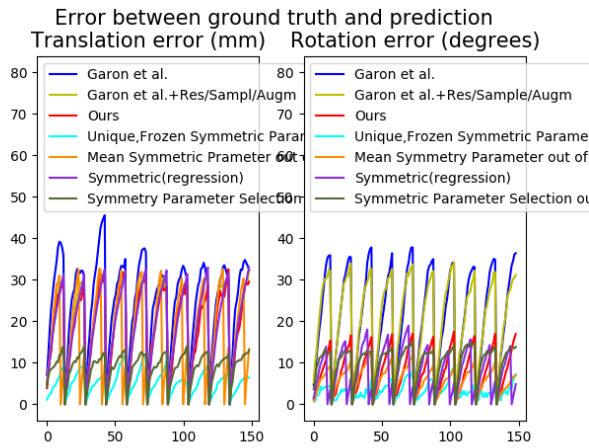


Figure 8.79 3D Translational and Rotational error metrics' plot, for the “Stability near” scenario of the Cookie Jar 3D CAD model, in the case of tracker re-iteration every 15 frames.

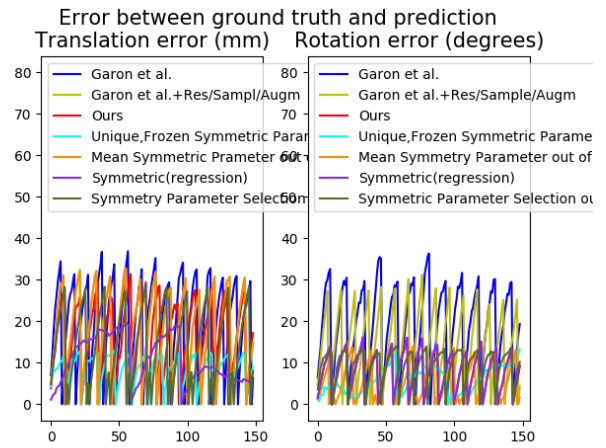


Figure 8.80 3D Translational and Rotational error metrics' plot, for the “Stability near” scenario of the Cookie Jar 3D CAD model, in the case of tracker re-iteration only after a failure.

Subcase 2: Object Far Away from the camera

In order to make the tracking procedure gradually more challenging, we repeat the same experiment but with the Cookie Jar placed far away from the Kinect RGB-D sensor and write our results down.

Architecture	Translational Error (mm)	Rotational Error (o)	Fails
Approach of Garon et al. [83] (20k training pairs) (<i>Symmetric assumption</i>) re-iterate every 15 frames	23.92 ± 11.96	28.43 ± 15.21	-
Approach of Garon et al. [83] (20k training pairs) (<i>Symmetric assumption</i>) re-iterate every time the tracker fails	24.55 ± 11.40	7.42 ± 12.73	13
Approach of Garon et al. [83] with Residual inter-layer connections and improved sampling/initialization/augmentation strategy (20k training pairs) (<i>Symmetric assumption</i>) re-iterate every 15 frames	7.71 ± 2.23	2.15 ± 1.17	-
Approach of Garon et al. [83] with Residual inter-layer connections and improved sampling/initialization/augmentation strategy (20k training pairs) (<i>Symmetric assumption</i>) re-iterate every time the tracker fails	15.11 ± 3.47	11.24 ± 3.98	0
Approach of Chapter 6 (20k training pairs) (<i>Asymmetric assumption</i>) re-iterate every 15 frames	14.26 ± 6.07	3.44 ± 1.18	-
Approach of Chapter 6 (20k training pairs) (<i>Asymmetric assumption</i>) re-iterate every time the tracker fails	11.40 ± 4.82	7.73 ± 2.28	0
Approach of Chapter 6 (20k training pairs) (<i>Unique Parameter Symmetric assumption</i>) re-iterate every 15 frames	11.87 ± 3.46	2.63 ± 0.97	-
Approach of Chapter 6 (20k training pairs) (<i>Unique Parameter Symmetric assumption</i>) re-iterate every time the tracker fails	9.88 ± 2.73	4.60 ± 1.52	0
Approach of Chapter 6 (20k training pairs) (<i>Batch of Parameters Symmetric assumption</i>) re-iterate every 15 frames	14.02 ± 5.97	3.81 ± 1.15	-
Approach of Chapter 6 (20k training pairs) (<i>Batch of Parameters Symmetric assumption</i>) re-iterate every time the tracker fails	10.36 ± 4.97	7.54 ± 2.16	0
Approach of Chapter 6 (20k training pairs) (<i>Parameter Regression Symmetric assumption</i>) re-iterate every 15 frames	16.82 ± 5.67	4.87 ± 1.64	-
Approach of Chapter 6 (20k training pairs) (<i>Parameter Regression Symmetric assumption</i>) re-iterate every time the tracker fails	13.33 ± 6.00	8.29 ± 3.50	0
Approach of Chapter 6 (20k training pairs) (<i>Batch of Parameters Symmetric assumption</i>) re-iterate every 15 frames	14.03 ± 5.56	3.00 ± 1.23	-
Approach of Chapter 6 (20k training pairs) (<i>Batch of Parameters Symmetric assumption</i>) re-iterate every time the tracker fails	11.26 ± 4.00	6.86 ± 2.14	0

Table 8.17: 3D Translational and Rotational Pose Errors and Fail count of the “Cookie Jar” model, for the “Stability far” for the baseline approach of Garon et al. [83], without/with sampling/ initialization/augmentation improvements we propose and the approach of Chapter 6 and its symmetry handling variations.



Figure 8.81 (a) Demonstration of an “Observed” RGB frame of the “Stability far” scenario with a rendered predicted pose of the “Cookie Jar” model generated by the baseline tracker of Garon et al. [83].



Figure 8.82 (b) Demonstration of an “Observed” RGB frame of the “Stability far” scenario with a rendered predicted pose of the “Cookie Jar” model generated by the baseline tracker of Garon et al. [83] with Residual inter-layer connections and sampling/initialization/augmentation improvements.



Figure 8.83 (c) Demonstration of an “Observed” RGB frame of the “Stability far” scenario with a rendered predicted pose of the “Cookie Jar” model generated by proposed tracker of this work.



Figure 8.84 (d) Demonstration of an “Observed” RGB frame of the “Stability far” scenario with a rendered predicted pose of the “Cookie Jar” model generated by proposed tracker, with the special case of unique rotational symmetry parameter, that remains frozen after training, of this work.



Figure 8.85 (e) Demonstration of an “Observed” RGB frame of the “Stability far” scenario with a rendered predicted pose of the “Cookie Jar” model generated by proposed tracker, with a learnable batch of rotational parameters, of this work.

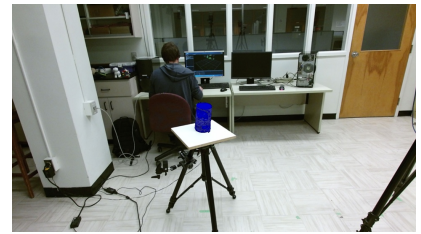


Figure 8.86 (f) of an “Observed” RGB frame of the “Stability far” scenario with a rendered predicted pose of the “Cookie Jar” model generated by proposed tracker, with a per-pose rotational parameter regression.



Figure 8.87 (g) Demonstration of an “Observed” RGB frame of the “Stability far” scenario with a rendered predicted pose of the “Cookie Jar” model generated by proposed tracker, with a per-pose pair optimal selection out of a nearly uniform batch of continuously rotational symmetry parameters.



Figure 8.88 (a)
Demonstration of the Foreground Extraction Attention map of the “Cookie Jar” model, provided by the proposed tracker in the “Stability far” scenario



Figure 8.89 (b)
Demonstration of the Occlusion Handling Attention map of the “Cookie Jar” model, provided by the proposed tracker in the “Stability far” scenario.



Figure 8.90 (c)
Demonstration of the Foreground Extraction Attention map provided by the proposed tracker, with a unique, frozen, rotational symmetry parameter, in the “Stability far” scenario



Figure 8.91 (d)
Demonstration of the Occlusion Handling Attention map of the “Cookie Jar” model, provided by the proposed tracker, with a unique, frozen, learnable rotational symmetry parameter in the “Stability far” scenario.



Figure 8.92 (e)
Demonstration of the Foreground Extraction Attention map provided by the proposed tracker, with a batch of frozen, learnable rotational symmetry parameters, in the “Stability far” scenario



Figure 8.93 (f)
Demonstration of the Occlusion Handling Attention map of the “Cookie Jar” model, provided by the proposed tracker, with the mean of a batch of frozen learnable in the “Stability far” scenario.



Figure 8.94 (g)
Demonstration of the Foreground Extraction Attention map provided by the proposed tracker, with per-pose regression of a learnable rotational symmetry parameter, in the “Stability far” scenario



Figure 8.95 (h)
Demonstration of the Occlusion Handling Attention map of the “Cookie Jar” model, provided by the proposed tracker, with a per-pose regressed learnable rotational symmetry parameter, in the “Stability far” scenario.



Figure 8.96 (i)
Demonstration of the Foreground Extraction Attention map provided by the proposed tracker, with a per-pose pair optimal selection out of a nearly uniform batch of continuously rotational symmetry parameters, in the “Stability far” scenario



Figure 8.97 (j)
Demonstration of the Occlusion Handling Attention map of the “Cookie Jar” model, provided by the proposed tracker, with a per-pose pair optimal selection out of a nearly uniform batch of continuously rotational symmetry parameters, in the “Stability far” scenario.

Qualitative analysis:

Baseline tracker of Garon et al[83](*subfig(a)*):

- The baseline tracker of Garon et al.[83] provides fairly good pose estimations. Its predictions are robust, in general, and its pose errors small. The tracking errors take time to accumulate.
- However, complete tracking failures are a rare, but still existing situation. The largest pose errors are mostly caused due to scale ambiguities, along the z-translational component.

Baseline with Res-connections+Sampling/Augmentation amelioration(*subfig(b)*):

- By replacing the dense connections of the SoA tracker with Residual ones and further enhancing their sampling and augmentation strategies, the remaining pose errors of the SoA approach are distinctively reduced and tracking failures are eliminated. The most profound improvement concerns the z-translational component whose predictions are more accurate.
- However, scale ambiguities have not been extincted, yet.

Ours(*subfig(c)*):

- Our, overall approach, improves the z-translational component error even more, obviously thanks to the foreground extraction attention module and the 6D continuous rotational parameter representation choice. Furthermore, our proposed tracker, again, presents no complete failures.
- In general, the pose errors presented in this scenario are bigger than in the “Near the camera” standing one.
- *Foreground Attention:*
These Attention weight maps intensely highlight the entirety of the target object with a response that is uniform through time.
- *Occlusion Attention:*
The Occlusion Attention weight map focus more on the distinct animal-like attributes of the Cookie Jar’s appearance.

Ours+Optimization of Unique Symmetry parameter (*subfig(d)*):

- This is the approach that results in the optimal predictions, overall. Again, we observe the recurrent pattern of unique rotational symmetry parameter convergence making 6D pose predictions more robust, when the object is standing still.
- Pose jitter is reduced, especially for the rotational component and, thus, drift is limited. However, scale ambiguities are omnipresent, even though limited. This discrepancy could motivate approaches that aim in possible future anisotropic weighting of different pose components, since, we clearly see one component being an important source of error, even in the simplest usecases. This is true for the translational part of our estimation. On the other hand, the z-rotational component, may be the most significant source of 3D rotational errors, with respect to the other two, but the mean values of the overall metrics have been dramatically decreased. This is something that validates the effectiveness of our modeling, especially for the symmetries’ handling, although the ablation study for this parameter has been for another, different real-world scenario, that of solely rotational 3D movement (see below).
- Looking at the attention map visualizations, we may conclude that, in this optimal case, between the various symmetry parameter selection options, the optimal one is the one that allows for balance between the two modules. Please note that this is the case only when the object is unoccluded and there is no severe clutter in the background (for example other objects with extravagant shapes or colouring, or, more importantly, other objects that the tracker has been trained on (in a possible extension of this work)).

Ours+Mean of a Batch of frozen,learnable continuously rotational Symmetry parameters (*subfig(e)*):

- Regressing a batch of different rotational symmetry parameters during training and then, freezing them and taking their mean during inference, produces results close to the ones of the uniquely-constrained solution above, but slightly worse. However, the robustness of the temporal prediction is assessed as visually acceptable.
- We, also, feel, here, the need to mention the perenial, as peculiar, pattern of different symmetry parameter selection algorithm(s) affecting the translational component error, as well. Moreover, we observe both pose metrics being increased or decreased together, something that validates the physical coupling of the two and justifies our single fully-connected layer incorporation, for simultaneous prediction, in the architecture.

Ours+per Viewpoint Regression of Symmetry parameter (*subfig(f)*):

- On the other hand, between the various symmetry parameter estimation techniques tried, plain regression emerges as the most unstable, and, thus, unsuitable for this scenario. Allowing the extra freedom of regressing a different parameter per different “Predicted”-“Observed” pose pair input, especially in large and correlated training datasets like ours, could easily perturb the parameter estimation during inference due to possible pose drift of the feedback-generated input.

- Rotational jitter (especially for the z-component) is at its highest and although estimations are not so bad, here, the other choices provide much ameliorated outcomes.
- However, it is fair to note that in this choice, as well as in any of the other ones, tracking failures have fallen to zero, after the more careful modeling of learning 3D rotations and their proper anisotropic weighting.

Ours+Optimal Selection out of a Batch of frozen,learnable continuously rotational Symmetry parameters (*subfig(g)*):

- The, overall, proposed strategy for selecting the rotational symmetry parameter is not the optimal one in this scenario (as it was not in the previous standing object scenario), but it, again, remains a close second.
- Although it offers a better tradeoff between taking the mean of the frozen, regressed parameters, and letting the parameter estimation free to be affected by the input pose pair drift, it slightly less stable than the “unique” symmetry solution. This time, the pose jitter of the z-rotational component is the one that increases the overall error, something that carries the z-translational error with it. The other two component predictions are the same as in the optimal case.

Error Plotting (left:re-iterate every 15 frames) and (right:re-iterate when the tracker fails):

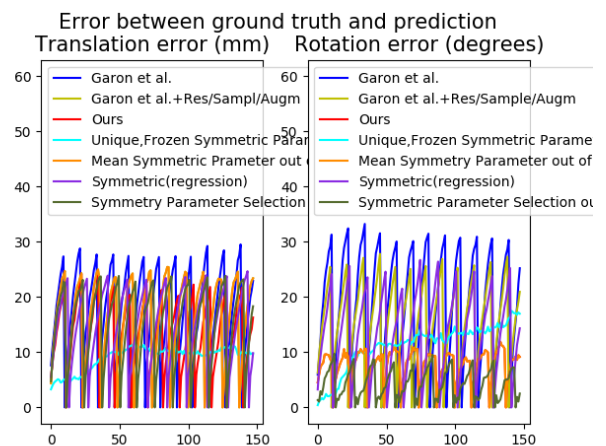
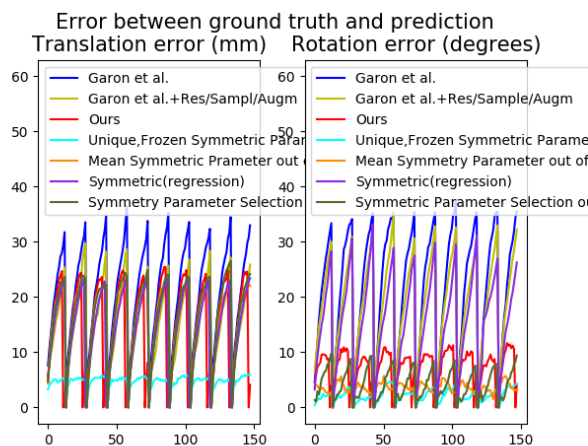


Figure 8.98 **Special case of rotational symmetry parameter estimation:** 3D Translational and Rotational error metrics’ plot, for the “Stability far” scenario of the Cookie Jar 3D CAD model, in the case of tracker re-iteration every 15 frames.

Figure 8.99 **Special case of rotational symmetry parameter estimation:** 3D Translational and Rotational error metrics’ plot, for the “Stability far” scenario of the Cookie Jar 3D CAD model, in the case of tracker re-iteration every 15 frames.

Object Pose Tracking on a 2D turntable with high occlusion degree (75 %):

Next, we examine the difference in the tracker’s performance when the object of interest is occluded by a static interposed plane, but at very high percentages (over 75%), similarly to the “Dragon” case.

75% Horizontal Occlusion

Initially, we horizontally occlude the Cookie Jar and we assess its impact when the occluder covers more than 75% of the object.

Architecture	Translational Error (mm)	Rotational Error (o)	Fails
Approach of Garon et al. [83] (20k training pairs) (<i>Symmetric assumption</i>) re-iterate every 15 frames	53.04 ± 29.56	18.12 ± 8.63	-
Approach of Garon et al. [83] (20k training pairs) (<i>Symmetric assumption</i>) re-iterate every time the tracker fails	30.25 ± 20.76	11.76 ± 6.86	16
Approach of Garon et al. [83] with Residual inter-layer connctions and improved sampling/initialization/augmentation strategy (20k training pairs) (<i>Symmetric assumption</i>) re-iterate every 15 frames	23.35 ± 11.92	8.55 ± 3.68	-
Approach of Garon et al. [83] with Residual inter-layer connctions and improved sampling/initialization/augmentation strategy (20k training pairs) (<i>Symmetric assumption</i>) re-iterate every time the tracker fails	25.14 ± 12.48	8.97 ± 3.75	14
Approach of Chapter 6 (20k training pairs) (<i>Asymmetric assumption</i>) re-iterate every 15 frames	9.51 ± 4.17	7.25 ± 3.99	-
Approach of Chapter 6 (20k training pairs) (<i>Asymmetric assumption</i>) re-iterate every time the tracker fails	9.73 ± 3.84	14.35 ± 8.23	11
Approach of Chapter 6 (20k training pairs) (<i>Unique Parameter Symmetric assumption</i>) re-iterate every 15 frames	8.16 ± 3.69	13.57 ± 8.07	-
Approach of Chapter 6 (20k training pairs) (<i>Unique Parameter Symmetric assumption</i>) re-iterate every time the tracker fails	8.90 ± 3.95	15.66 ± 9.08	13
Approach of Chapter 6 (20k training pairs) (<i>Batch of Parameters Symmetric assumption</i>) re-iterate every 15 frames	6.37 ± 2.14	15.48 ± 9.50	-
Approach of Chapter 6 (20k training pairs) (<i>Batch of Parameters Symmetric assumption</i>) re-iterate every time the tracker fails	6.39 ± 2.17	16.02 ± 9.80	15
Approach of Chapter 6 (20k training pairs) (<i>Parameter Regression Symmetric assumption</i>) re-iterate every 15 frames	8.36 ± 3.55	11.82 ± 7.37	-
Approach of Chapter 6 (20k training pairs) (<i>Parameter Regression Symmetric assumption</i>) re-iterate every time the tracker fails	9.20 ± 3.65	14.79 ± 8.90	12
Approach of Chapter 6 (20k training pairs) (<i>Optimal Selection out of a uniform Batch of Parameters Symmetric assumption</i>) re-iterate every 15 frames	6.37 ± 2.14	7.22 ± 3.97	-
Approach of Chapter 6 (20k training pairs) (<i>Optimal Selection out of a uniform Batch of Parameters Symmetric assumption</i>) re-iterate every time the tracker fails	5.44 ± 1.99	13.57 ± 8.03	11

Table 8.18: 3D Translational and Rotational Pose Errors and Fail count of the “Cookie Jar” model, for the “75% Horizontal Occlusion” scenario for the baseline approach of Garon et al. [83], without/with sampling/initialization/augmentation improvements we propose and the approach of Chapter 6.



Figure 8.100 (a) Demonstration of an “Observed” RGB frame of the “75% Horizontal Occlusion” scenario with a rendered predicted pose of the “Cookie Jar” model generated by the baseline tracker of Garon et al.[83].

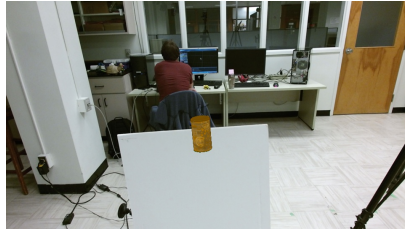


Figure 8.101 (b) Demonstration of an “Observed” RGB frame of the “75% Horizontal Occlusion” scenario with a rendered predicted pose of the “Cookie Jar” model generated by the baseline tracker of Garon et al.[83] with Residual inter-layer connections and sampling/initialization/augmentation improvements.



Figure 8.102 (c) Demonstration of an “Observed” RGB frame of the “75% Horizontal Occlusion” scenario with a rendered predicted pose of the “Cookie Jar” model generated by proposed tracker of this work.

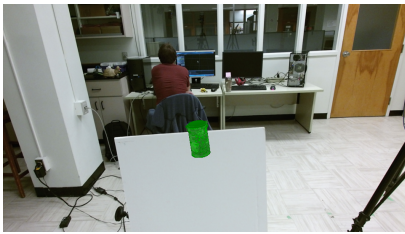


Figure 8.103 (d) Demonstration of an “Observed” RGB frame of the “75% Horizontal Occlusion” scenario with a rendered predicted pose of the “Cookie Jar” model generated by proposed tracker, with the special case of unique rotational symmetry parameter, that remains frozen after training, of this work.

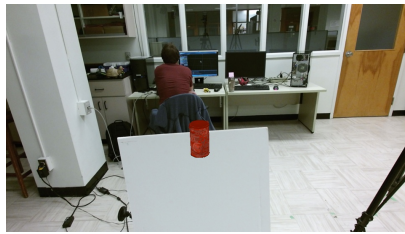


Figure 8.104 (e) Demonstration of an “Observed” RGB frame of the “75% Horizontal Occlusion” scenario with a rendered predicted pose of the “Cookie Jar” model generated by proposed tracker, with a learnable batch of rotational parameters, of this work.

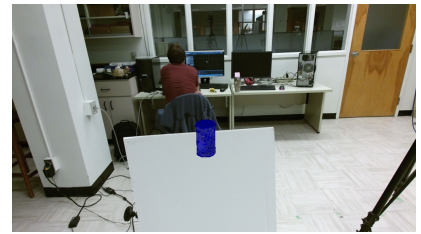


Figure 8.105 (f) Demonstration of the Occlusion Handling Attention map provided by the proposed tracker, with a continuous rotational parameter regression, in the “75% Horizontal Occlusion” scenario and tested on the “Cookiejar” 3D model.

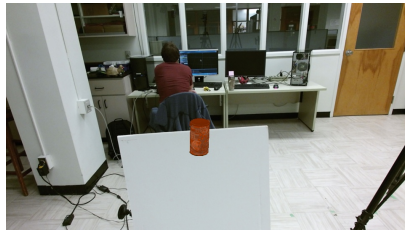


Figure 8.106 (g) Demonstration of the Occlusion Handling Attention map provided by the proposed tracker, with a per-pose pair optimal selection out of a nearly uniform batch of continuously rotational symmetry parameters, in the “75% Horizontal Occlusion” scenario and tested on the “Cookiejar” 3D model.

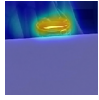


Figure 8.107 (a)
Demonstration of the Foreground Extraction Attention map of the “Cookie Jar” model, provided by the proposed tracker in the “75% Horizontal Occlusion” scenario

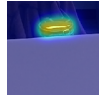


Figure 8.108 (b)
Demonstration of the Occlusion Handling Attention map of the “Cookie Jar” model, provided by the proposed tracker in the “75% Horizontal Occlusion” scenario.

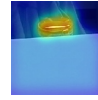


Figure 8.109 (c)
Demonstration of the Foreground Extraction Attention map of the “Cookie Jar” model, provided by the proposed tracker, with a unique, frozen learnable rotational symmetry parameter, in the “75% Horizontal Occlusion” scenario

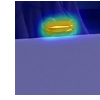


Figure 8.110 (d)
Demonstration of the Occlusion Handling Attention map of the “Cookie Jar” model, provided by the proposed tracker, with a unique, frozen learnable rotational symmetry parameter, in the “75% Horizontal Occlusion” scenario.

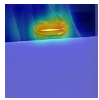


Figure 8.111 (e)
Demonstration of the Foreground Extraction Attention map of the “Cookie Jar” model, provided by the proposed tracker, with the mean of a batch of frozen, learnable continuous rotational symmetry parameters, in the “75% Horizontal Occlusion” scenario

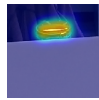


Figure 8.112 (f)
Demonstration of the Occlusion Handling Attention map of the “Cookie Jar” model, provided by the proposed tracker, with the mean of a batch of frozen learnable continuous rotational symmetry parameters, in the “75% Horizontal Occlusion” scenario.

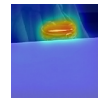


Figure 8.113 (g)
Demonstration of the Foreground Extraction Attention map of the “Cookie Jar” model, provided by the proposed tracker, with the per-pose regressed learnable continuous rotational symmetry parameter, in the “75% Horizontal Occlusion” scenario

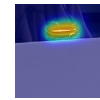


Figure 8.114 (h)
Demonstration of the Occlusion Handling Attention map of the “Cookie Jar” model, provided by the proposed tracker, with a per-pose regressed learnable continuous rotational symmetry parameter, in the “75% Horizontal Occlusion” scenario.

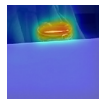


Figure 8.115 (i)
Demonstration of the Foreground Extraction Attention map of the “Cookie Jar” model, provided by the proposed tracker, with the per-pose pair optimal selection out of a nearly uniform batch of continuously rotational symmetry parameters, in the “75% Horizontal Occlusion” scenario

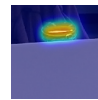


Figure 8.116 (j)
Demonstration of the Occlusion Handling Attention map of the “Cookie Jar” model, provided by the proposed tracker, with a per-pose pair optimal selection out of a nearly uniform batch of continuously rotational symmetry parameters, in the “75% Horizontal Occlusion” scenario.

Qualitative analysis:

Baseline tracker of Garon et al[83](*subfig(a)*):

- From an object standpoint, this is the hardest scenario for our tracker due to the geometrical structure of our object. Since the occluder leaves only an almost completely symmetrical part visible for us to base our estimations on, “Predicted” frame feedback seems to be our only source of information left make an educated guess. This is especially hard for rotational estimation, as one degree of freedom (that of the z-component) is essentially non-constrained in its estimates.
- All this previous discussion, obviously justifies large pose errors of the baseline tracker, as far as both metrics are concerned.
- A qualitative analysis would demonstrate that the previous reasons are the source for many tracking failures, caused especially by the translational pose component, after a few feedback iterations.

- However, one positive side of this, first kind of static occlusion imprint that we have to recognize, in comparison to its vertical counterparts, is that occlusion-induced symmetries are more stable and not relied on different viewpoint configurations. This allows for more stable pose predictions of the neural architecture, and that is why overall tracking failures are half here, in response to the vertical case.

Baseline with Res-connections+Sampling/Augmentation amelioration(*subfig(b)*):

- Our first architectural improvements have classically mollified error peaks, more in the direction of the translational, than the rotational component.
- Despite the observed pattern of this architectural variation in previous scenarios, this time overall failures are not significantly reduced. This shows the severity of symmetry-induced ambiguities in estimating the pose. This is the strongest argument for the rest of our design additions.

Ours(*subfig(c)*):

- Since it is obvious that accurate rotational estimation is the “Achilles’s heel” of our tracker in this scenario, due to severe static occlusion and its implications in generating new pose ambiguities, we clearly characterize our modeling of this component, accompanied with the two parallel attention module addition as the gamechanger that reduces the overall tracking failures and more than halves the mean and the standard estimation of both errors.
- Due to the presence of rotational symmetry, the high occlusion percentage of mote than 75% “breaks” our visual features and leaves very little information for the tracker to exploit.

This becomes more obvious as we examine the two depicted parallel attentional maps. In general, one thing that we witness for the attentional patterns is that they are limned with higher intensity here, in comparison to the following vertical occlusion scenario. This is a qualitative factor that validates the effectiveness of our method. In more detail:

- *Foreground Attention:* This, first, attention map presents sharper peaks when we compare it to the ones of the next type of occlusion, the vertical one. This happens as it detects the Cookie Jar’s lid as a feature of interest that makes it visually different than its background. Extra to that, the lit’s appearance is consistent and allows for a more stable cue for defining the pose, in comparison with the ever-changing visual patterns that emerge in the following scenario.
- *Occlusion Attention:* However, we can see visual attention leaking of the foreground extraction module to parts of the occluder, as well. So, this, second module, which is dedicated to occlusion handling comes in play and weighs in only the pixels that correspond to the object of interest. When the two augmented feature maps are added, the object pixels are at the strongest position and, thus, prevail in the pose prediction of the following layers.

In particular, an interesting emerging pattern is that when the rotational symmetry breaks, due to the appearance of the lit’s handle, the Attentional peak becomes sharper and is cented on the these two, symmetrically positioned, handles.

Ours+Optimization of Unique Symmetry parameter (*subfig(d)*):

- This, first alternative of estimating the regressed symmetry rotational z-parameter, the uniquely-imposed one, is one of the few cases we encounter in our study where translational and rotational components do not follow the same trend. While the rotational component is compromised, by a respectable amount, with respect to the asymmetric case, the translational (and especially its scale component) is slightly improved. Overall, we are more interested, in this case, in what happens with the 3D rotations, so we are obliged to conclude that this approach is far from optimal for this scenario.

Ours+Mean of a Batch of frozen, learnable continuously rotational Symmetry parameters (*subfig(e)*):

- The same effect is observed in this second selection option, as well. This time, translation predictions are even better, but rotational jitter is aggrandized by an even larger percentage.

Ours+per Viewpoint Regression of Symmetry parameter (*subfig(f)*):

- As we would expect, direct and unconstrained regression of the rotational symmetry (z-)parameter is not our optimal choice. Surprisingly, though, we see its estimations being better than its more constrained two counterparts, above, and even exceed the accuracy of the asymmetric method.
- We appoint this development into the fact that, in the “Occlusion” scenarios, the object is performing a keener movement (even if this is only a 1-DOF rotation) than in the previous two standing still object scenarios (both near and far away from the camera), something that upgrades the role of the direct regression symmetry parameter method. However, as we see below, it is not the optimal strategy-to-go.

Ours+Optimal Selection out of a Batch of frozen,learnable,continuously rotational Symmetry parameters (*subfig(g)*):

- This, overallly proposed, approach is the one producing the least amount of pose tracking errors across all metrics.
- However, we should not be very enthusiastic about this success of ours. Both translational and rotational errors are pretty high, compared to the interaction scenarios presented below and to the other objects that we test our approach on.

Error Plotting (left:re-iterate every 15 frames) and (right:re-iterate when the tracker fails):

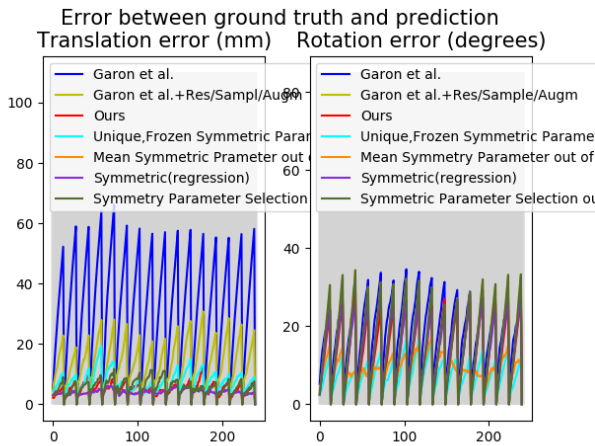


Figure 8.117 **Special case of rotational symmetry parameter estimation: 3D Translational and Rotational error metrics' plot**, for the “75% Horizontal Occlusion” scenario of the Cookie Jar 3D CAD model, in the case of tracker re-iteration only after a failure.

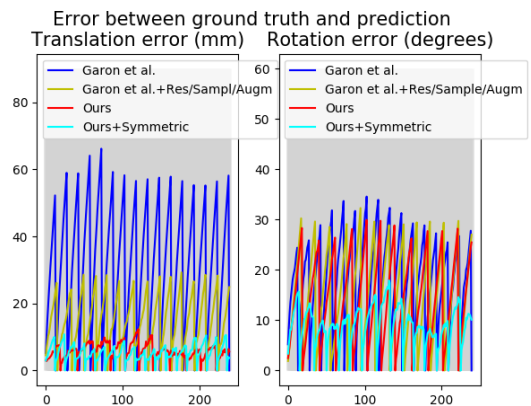


Figure 8.118 **Special case of rotational symmetry parameter estimation: 3D Translational and Rotational error metrics' plot**, for the “75% Horizontal Occlusion” scenario of the Cookie Jar 3D CAD model, in the case of tracker re-iteration only after a failure.

75% Vertical Occlusion

Then, we vertically occlude the object with the static plane and report our observations.

Architecture	Translational Error (mm)	Rotational Error (o)	Fails
Approach of Garon et al. [83] (20k training pairs) (<i>Symmetric assumption</i>) re-iterate every 15 frames	45.69 ± 26.65	16.99 ± 8.84	-
Approach of Garon et al. [83] (20k training pairs) (<i>Symmetric assumption</i>) re-iterate every time the tracker fails	43.56 ± 35.03	17.58 ± 17.49	32
Approach of Garon et al. [83] with Residual inter-layer connections and improved sampling/initialization/augmentation strategy (20k training pairs) (<i>Symmetric assumption</i>) re-iterate every 15 frames	18.20 ± 7.21	18.32 ± 10.49	-
Approach of Garon et al. [83] with Residual inter-layer connections and improved sampling/initialization/augmentation strategy (20k training pairs) (<i>Symmetric assumption</i>) re-iterate every time the tracker fails	17.86 ± 7.16	16.98 ± 9.69	16
Approach of Chapter 6 (20k training pairs) (<i>Asymmetric assumption</i>) re-iterate every 15 frames	20.97 ± 7.32	7.22 ± 3.97	-
Approach of Chapter 6 (20k training pairs) (<i>Asymmetric assumption</i>) re-iterate every time the tracker fails	20.83 ± 7.23	15.35 ± 9.06	14
Approach of Chapter 6 (20k training pairs) (<i>Unique Parameter Symmetric assumption</i>) re-iterate every 15 frames	19.47 ± 9.60	14.36 ± 9.98	-
Approach of Chapter 6 (20k training pairs) (<i>Unique Parameter Symmetric assumption</i>) re-iterate every time the tracker fails	17.90 ± 8.48	9.21 ± 8.55	8
Approach of Chapter 6 (20k training pairs) (<i>Batch of Parameters Symmetric assumption</i>) re-iterate every 15 frames	19.12 ± 7.86	15.58 ± 9.69	-
Approach of Chapter 6 (20k training pairs) (<i>Batch of Parameters Symmetric assumption</i>) re-iterate every time the tracker fails	19.22 ± 7.77	16.14 ± 10.06	15
Approach of Chapter 6 (20k training pairs) (<i>Parameter Regression Symmetric assumption</i>) re-iterate every 15 frames	28.91 ± 12.73	14.03 ± 8.52	-
Approach of Chapter 6 (20k training pairs) (<i>Parameter Regression Symmetric assumption</i>) re-iterate every time the tracker fails	26.66 ± 11.35	12.83 ± 8.37	18
Approach of Chapter 6 (20k training pairs) (<i>Optimal Selection out of a uniform Batch of Parameters Symmetric assumption</i>) re-iterate every 15 frames	19.01 ± 7.53	13.00 ± 7.49	-
Approach of Chapter 6 (20k training pairs) (<i>Optimal Selection out of a uniform Batch of Parameters Symmetric assumption</i>) re-iterate every time the tracker fails	16.75 ± 6.83	8.66 ± 7.03	14

Table 8.19: 3D Translational and Rotational Pose Errors and Fail count of the “Cookie Jar” model, for the “75% Vertical Occlusion” scenario for the baseline approach of Garon et al. [83], without/with sampling/initialization/augmentation improvements we propose and the approach of Chapter 6 and its symmetry handling variations.

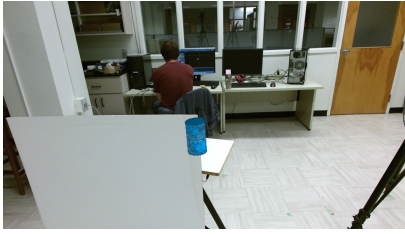


Figure 8.119 (a) Demonstration of an “Observed” RGB frame of the “75% Vertical Occlusion” scenario with a rendered predicted pose of the “Cookie Jar” model generated by the baseline tracker of Garon et al. [83].

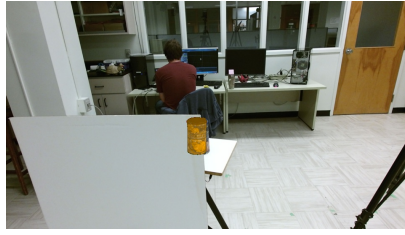


Figure 8.120 (b) Demonstration of an “Observed” RGB frame of the “75% Vertical Occlusion” scenario with a rendered predicted pose of the “Cookie Jar” model generated by the baseline tracker of Garon et al. [83] with Residual inter-layer connections and sampling/initialization/augmentation improvements.

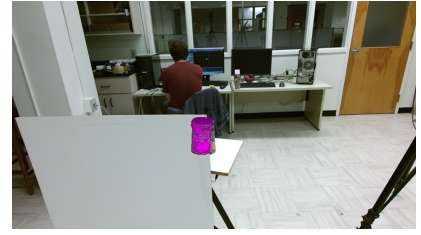


Figure 8.121 (c) Demonstration of an “Observed” RGB frame of the “75% Vertical Occlusion” scenario with a rendered predicted pose of the “Cookie Jar” model generated by proposed tracker of this work.



Figure 8.122 (d) Demonstration of an “Observed” RGB frame of the “75% Vertical Occlusion” scenario with a rendered predicted pose of the “Cookie Jar” model generated by proposed tracker, with the special case of unique rotational symmetry parameter, that remains frozen after training, of this work.



Figure 8.123 (e) Demonstration of an “Observed” RGB frame of the “75% Vertical Occlusion” scenario with a rendered predicted pose of the “Cookie Jar” model generated by proposed tracker, with a learnable batch of rotational parameters, of this work.

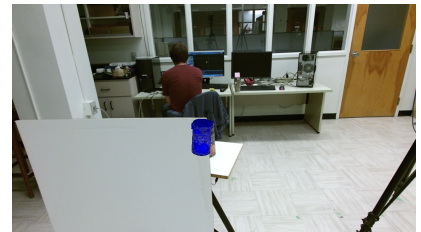


Figure 8.124 (f) Demonstration of the Occlusion Handling Attention map provided by the proposed tracker, with a rotational parameter regression, in the “75% Vertical Occlusion” scenario and tested on the “Cookiejar” 3D model.



Figure 8.125 (g) Demonstration of the Occlusion Handling Attention map provided by the proposed tracker, with a rotational parameter regression, in the “75% Vertical Occlusion” scenario and tested on the “Cookiejar” 3D model.



Figure 8.126 (a)
Demonstration of the Foreground Extraction Attention map of the “Cookie Jar” model, provided by the proposed tracker in the “75% Vertical Occlusion” scenario.



Figure 8.127 (b)
Demonstration of the Occlusion Handling Attention map of the “Cookie Jar” model, provided by the proposed tracker in the “75% Vertical Occlusion” scenario.

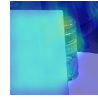


Figure 8.128 (c)
Demonstration of the Foreground Extraction Attention map of the “Cookie Jar” model, provided by the proposed tracker, with a unique, frozen learnable continuous rotational symmetry parameter, in the “75% Vertical Occlusion” scenario.



Figure 8.129 (d)
Demonstration of the Occlusion Handling Attention map of the “Cookie Jar” model, provided by the proposed tracker, with a unique, frozen learnable rotational symmetry parameter, in the “75% Vertical Occlusion” scenario.



Figure 8.130 (e)
Demonstration of the Foreground Extraction Attention map of the “Cookie Jar” model provided by the proposed tracker, with the mean of a batch of frozen, learnable continuous rotational symmetry parameters, in the “75% Vertical Occlusion” scenario.



Figure 8.131 (f)
Demonstration of the Occlusion Handling Attention map of the “Cookie Jar” model, provided by the proposed tracker, with the mean of a batch of frozen, learnable, rotational symmetry parameters, in the “75% Vertical Occlusion” scenario.



Figure 8.132 (g)
Demonstration of the Foreground Extraction Attention map of the “Cookie Jar” model, provided by the proposed tracker, with a per-pose regressed learnable continuous rotational symmetry parameter, in the “75% Vertical Occlusion” scenario.

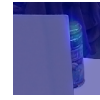


Figure 8.133 (h)
Demonstration of the Occlusion Handling Attention map of the “Cookie Jar” model, provided by the proposed tracker, with a per-pose regressed learnable rotational symmetry parameter, in the “75% Vertical Occlusion” scenario.



Figure 8.134 (g)
Demonstration of the Foreground Extraction Attention map of the “Cookie Jar” model, provided by the proposed tracker, with a per-pose regressed learnable continuous rotational symmetry parameter, in the “75% Vertical Occlusion” scenario.



Figure 8.135 (h)
Demonstration of the Occlusion Handling Attention map of the “Cookie Jar” model, provided by the proposed tracker, with a per-pose regressed learnable rotational symmetry parameter, in the “75% Vertical Occlusion” scenario.

Qualitative analysis:

Baseline tracker of Garon et al[83](*subfig(a)*):

- The baseline tracker of Garon et al.[83], firstly, presents increased pose jitter, both translational and rotational, that drifts the estimation and result in frequent failures.
- Secondly, there is an extra symmetry induced error that originates from inappropriate modeling of the rotational components and the object’s symmetry around the object-centric z-axis. We claim that, with proper modeling, this error could be disentangled and, eventually, reduced.

Baseline with Res-connections+Sampling/Augmentation amelioration(*subfig(b)*):

- Clearly, error peaks are severely mitigated, though the effect on the translational component is more profound in this scenario.
- Complete tracking failures have been halved and the rate of accumulation of pose drift has severely been shortened.

- However, rotational errors are still, unacceptably high.
- Please note that all the aforementioned observations have been made with respect to the difficulty the task at hand presents for the tracker. This object geometry is highly symmetrical and this, highest percentage scenario present in the dataset, generated a great deal of shape-induced ambiguity in properly estimating, not just the object’s rotation, but its translation as well, since scale information fades with the increase of static occlusions. Although more expressive than in the previous scenario, high static occlusion percentages create appearance-based ambiguities, as well, since partially hiding of complete visual patterns match multiple copycat alternate instances that the neural network has interacted with, but in different pose layout. Under the light of such observations, its is profound that viewpoint-symmetries occur as an extra source of uncertainty (especially one that the netowrk has not thoroughly trained on and on scaled percentages) making large pose errors something to be anticipated.

Ours(*subfig(c)*):

- Our approach presents smaller drifts, w.r.t. to the previous case, that are more prone to get erased by the feedback rendering stream.
- However,
 - it is evident that this is not a general conclusion and that there are many times that pose errors are big enough to cause failures, something universal to all pose components.
 - the rotational error that are related to the symmetry discrepancy are not zeroed out by any means, something that causes a need for proper modeling of these symmetry parameter to emerge.

• *Foreground Attention:*

This Attention weight map is successful in disentagling the object features (those of the foreground) from the ones corresponding to the background, but they also lighten up a part of the occluder, something that inserts ambiguities. As far as it concerns more specific attentional patterns that occur in an unsupervised way, this module focuses mostly on the lid of the Cookie Jar, as the shapes that were the main point of interest in the previous testing datasets are occluded to their most part.

• *Occlusion Attention:*

This Attentional weight map has a universal effect, similar to the one of the “Foreground” module, that avoids the occluder’s regions and it enhances the lit’s pixels, too.

Ours+Optimization of Unique Symmetry parameter (*subfig(d)*):

- This first approach in modeling the rotational symmetry component slightly ameliorates the rotational estimation’s accuracy. In fact, it makes it more difficult to the tracker to change its orientation output w.r.t. to the previous frame. Maybe, however, that is too strict and resulting in an intransigent optimized rotational symmetry value more or less deprives the tracker from a rotational degree of freedom. This case was present in our theoretical analysis but it is more obvious in this case.

Ours+Mean of a Batch of frozen,learnable continuously rotational Symmetry parameters (*subfig(e)*):

- By relaxing the single solution requirement of the symmetrical parameter optimization, we allow for a more flexible handling of the rotational component, something that we observe in the amelioration of the tracking accuracy of that, rotational component.

Ours+per Viewpoint Regression of Symmetry parameter (*subfig(f)*):

- Lastly, the per Viewpoint Regression seems to be the most successful approach as it lets the tracker to reach a different optimum w.r.t. to different input sample pose neighbourhood. It is allows for increased flexibility, something useful as in some viewpoint the resulting parameter solution range may be sparser and in others denser.

Ours+Optimal Selection out of a Batch of frozen,learnable,continuously rotational Symmetry parameters ($subfig(g)$):

- The optimal strategy, on the contrary, for this scenario, is the overallly proposed one: optimally selecting one parameter out of a learned,frozen batch of them, approximately evenly distributed. It is the strategy that, in general, seems to disentangle the core of rotation estimation better than symmetry-based ambiguities.
- From a qualitative standpoint, this approach produces the least amount of rotational jitter and stabilizes, mutatis mutantis, the 3D translational estimations.
- However, one must not neglect that this is a particularly difficult testing scenario, that we are working on and that, even in the optimal case, overall tracking failures are at an, almost, all-scenario high. Inaccurate translational predictions, especially in the z-component, inflame 3D rotational errors and, even though our methods improve the final results, pose drifts accumulates at a moderately fast pace.

Error Plotting (left:re-iterate every 15 frames) and (right:re-iterate when the tracker fails):

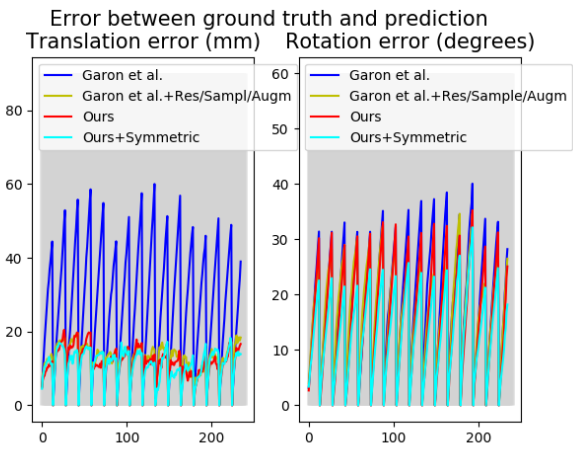


Figure 8.136 **Special case of rotational symmetry parameter estimation:** 3D Translational and Rotational error metrics’ plot, for the “Translation Only Interaction” scenario of the Cookie Jar 3D CAD model, in the case of tracker re-iteration only after a failure.

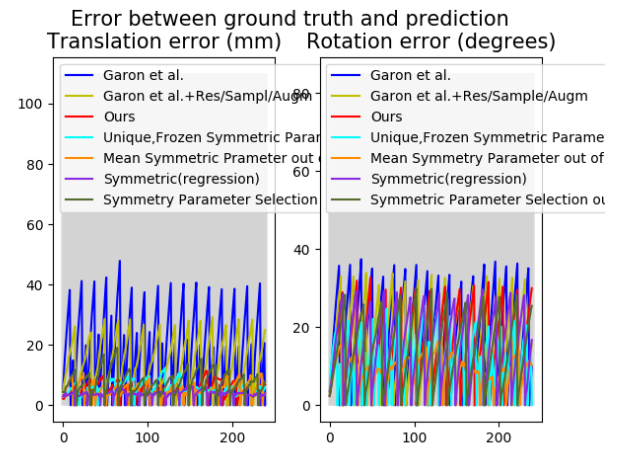


Figure 8.137 **Special case of rotational symmetry parameter estimation:** 3D Translational and Rotational error metrics’ plot, for the “Translation Only Interaction” scenario of the Cookie Jar 3D CAD model, in the case of tracker re-iteration only after a failure.

Free 3D Object Movement: Translation Only

Architecture	Translational Error (mm)	Rotational Error (o)	Fails
Approach of Garon et al. [83] (20k training pairs) (<i>Symmetric assumption</i>) re-iterate every 15 frames	61.59 ± 46.26	27.33 ± 26.54	-
Approach of Garon et al. [83] (20k training pairs) (<i>Symmetric assumption</i>) re-iterate every time the tracker fails	43.56 ± 46.26	27.33 ± 17.49	32
Approach of Garon et al. [83] with Residual inter-layer connections and improved sampling/initialization/augmentation strategy (20k training pairs) (<i>Symmetric assumption</i>) re-iterate every 15 frames	28.06 ± 35.15	11.05 ± 9.01	-
Approach of Garon et al. [83] with Residual inter-layer connections and improved sampling/initialization/augmentation strategy (20k training pairs) (<i>Symmetric assumption</i>) re-iterate every time the tracker fails	22.08 ± 17.43	12.11 ± 7.83	13
Approach of Chapter 6 (20k training pairs) (<i>Asymmetric assumption</i>) re-iterate every 15 frames	13.83 ± 9.88	8.31 ± 5.97	-
Approach of Chapter 6 (20k training pairs) (<i>Asymmetric assumption</i>) re-iterate every time the tracker fails	14.63 ± 8.02	9.19 ± 6.70	5
Approach of Chapter 6 (20k training pairs) (<i>Unique Parameter Symmetric assumption</i>) re-iterate every 15 frames	15.71 ± 9.28	7.14 ± 7.47	-
Approach of Chapter 6 (20k training pairs) (<i>Unique Parameter Symmetric assumption</i>) re-iterate every time the tracker fails	18.20 ± 8.58	9.27 ± 8.85	7
Approach of Chapter 6 (20k training pairs) (<i>Batch of Parameters Symmetric assumption</i>) re-iterate every 15 frames	15.74 ± 9.61	5.83 ± 5.50	-
Approach of Chapter 6 (20k training pairs) (<i>Batch of Parameters Symmetric assumption</i>) re-iterate every time the tracker fails	18.25 ± 8.58	11.18 ± 6.37	5
Approach of Chapter 6 (20k training pairs) (<i>Parameter Regression Symmetric assumption</i>) re-iterate every 15 frames	15.58 ± 8.36	6.68 ± 6.31	-
Approach of Chapter 6 (20k training pairs) (<i>Parameter Regression Symmetric assumption</i>) re-iterate every time the tracker fails	17.95 ± 7.54	9.96 ± 7.07	5
Approach of Chapter 6 (20k training pairs) (<i>Optimal Selection out of a uniform Batch of Parameters Symmetric assumption</i>) re-iterate every 15 frames	15.52 ± 9.38	5.83 ± 5.50	-
Approach of Chapter 6 (20k training pairs) (<i>Optimal Selection out of a uniform Batch of Parameters Symmetric assumption</i>) re-iterate every time the tracker fails	14.97 ± 7.02	6.35 ± 7.08	3

Table 8.20: 3D Translational and Rotational Pose Errors and Fail count of the “Cookie Jar” model, for the “Translation Only Interaction” scenario for the baseline approach of Garon et al. [83], without/with sampling/initialization/augmentation improvements we propose and the approach of Chapter 6 and its symmetric handling variations.



Figure 8.138 (a) Demonstration of an “Observed” RGB frame of the “Translation Only Interaction” scenario with a rendered predicted pose of the “Cookie Jar” model generated by the baseline tracker of Garon et al.[83].



Figure 8.139 (b) Demonstration of an “Observed” RGB frame of the “Translation Only Interaction” scenario with a rendered predicted pose of the “Cookie Jar” model generated by the baseline tracker of Garon et al.[83] with Residual inter-layer connections and sampling/initialization/augmentation improvements.



Figure 8.140 (c) Demonstration of an “Observed” RGB frame of the “Translation Only Interaction” scenario with a rendered predicted pose of the “Cookie Jar” model generated by proposed tracker of this work.



Figure 8.141 (d) Demonstration of an “Observed” RGB frame of the “Translation Only Interaction” scenario with a rendered predicted pose of the “Cookie Jar” model generated by proposed tracker, with the special case of unique rotational symmetry parameter, that remains frozen after training, of this work.

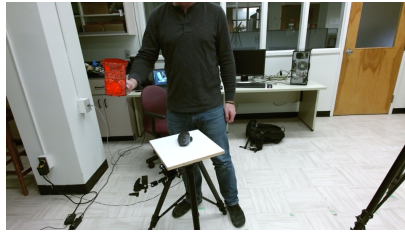


Figure 8.142 (e) Demonstration of an “Observed” RGB frame of the “Translation Only Interaction” scenario with a rendered predicted pose of the “Cookie Jar” model generated by proposed tracker, with a learnable batch of rotational parameters, of this work.

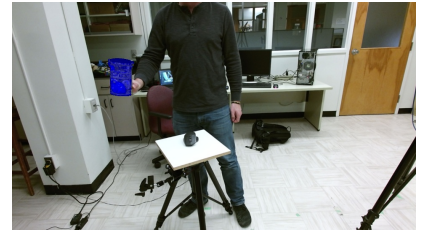


Figure 8.143 (h) Demonstration of the Occlusion Handling Attention map provided by the proposed tracker, with a rotational parameter regression, in the “Translation Only Interaction” scenario and tested on the “Cookiejar” 3D model.



Figure 8.144 (g) Demonstration of the Occlusion Handling Attention map provided by the proposed tracker, with a per-pose pair optimal selection out of a nearly uniform batch of continuously rotational symmetry parameters, in the “Translation Only Interaction” scenario and tested on the “Cookiejar” 3D model.



Figure 8.145 (a)
 Demonstration of the Occlusion Handling Attention map of the “Cookie Jar” model, provided by the proposed tracker, in the “Translation Only Interaction” scenario.



Figure 8.146 (b)
 Demonstration of the Occlusion Handling Attention map of the “Cookie Jar” model, provided by the proposed tracker, in the “Translation Only Interaction” scenario.



Figure 8.147 (c)
 Demonstration of the Occlusion Handling Attention map of the “Cookie Jar” model, provided by the proposed tracker, with a unique, frozen learnable continuous rotational symmetry parameters, in the “Translation Only Interaction” scenario.



Figure 8.148 (d)
 Demonstration of the Occlusion Handling Attention map of the “Cookie Jar” model, provided by the proposed tracker, with a unique, frozen learnable rotational symmetry parameter, in the “Translation Only Interaction” scenario.



Figure 8.149 (e)
 Demonstration of the Occlusion Handling Attention map of the “Cookie Jar” model, provided by the proposed tracker, with the mean of a batch of frozen, learnable learnable continuous rotational symmetry parameter, in the “Translation Only Interaction” scenario.



Figure 8.150 (f)
 Demonstration of the Occlusion Handling Attention map of the “Cookie Jar” model, provided by the proposed tracker, with the mean of a batch of frozen, learnable rotational symmetry parameters, in the “Translation Only Interaction” scenario.



Figure 8.151 (g)
 Demonstration of the Occlusion Handling Attention map of the “Cookie Jar” model, provided by the proposed tracker, with a per-pose regressed learnable rotational symmetry parameter, in the “Translation Only Interaction” scenario.



Figure 8.152 (h)
 Demonstration of the Occlusion Handling Attention map of the “Cookie Jar” model, provided by the proposed tracker, with a per-pose regressed learnable rotational symmetry parameter, in the “Translation Only Interaction” scenario.



Figure 8.153 (i)
 Demonstration of the Occlusion Handling Attention map of the “Cookie Jar” model, provided by the proposed tracker, with a per-pose regressed learnable rotational symmetry parameter, in the “Translation Only Interaction” scenario.



Figure 8.154 (j)
 Demonstration of the Occlusion Handling Attention map of the “Cookie Jar” model, provided by the proposed tracker, with a per-pose regressed learnable rotational symmetry parameter, in the “Translation Only Interaction” scenario.

Qualitative analysis:

Baseline tracker of Garon et al[83](*subfig(a)*):

- At the first few frames, we observe an important amount of symmetry pose jitter induced in our estimations. This jitter inserts prediction noise and aggravates the tracker’s errors.
- Although the careful reader would initially assume that since the object in this scenario is mostly translated in the 3D space and kept at a rather constant orientation, the pose-based rotational error would be low. However, this is not the case and a second, more insightful look is needed to figure out why: since the object is kept pretty much at the same orientation across the temporal video clip development, the rotational difference we need to estimate is non-essential and all rotational error left in our visual inspection is caused by continuous symmetries around the principal axis of the object.

- The object is not moving in extremely fast speeds under this testing scenario dataset. However, the baseline tracker of Garon et al. [83] fails in some cases in the highest speeds of this, overall, medium distribution.

Baseline with Res-connections+Sampling/Augmentation amelioration(*subfig(b)*):

- As we start to improve the baseline approach of Garon et al.[83] with our designing and sampling/augmentation choices, we observe the negative artifact of increasing the symmetry-induced errors.
- The problem of producing larger tracking errors as the object speed increases remains. It is not intact, but has not been completely solved by those improvements.
- However, albeit the previous drawbacks, those first improvements of our tracker yield lower mean and small scale pose errors that result in fewer irrecoverable failures.

Ours(*subfig(c)*):

- Due to the proper rotational parameter representation during the learning procedure and the use of a weighted Geodesic rotational loss function, the overall rotational 3D error, as well as its symmetry-based component, have both been reduced. However, the symmetry jitter remains in the estimation.
- Our approach yields smaller errors in slower speeds: an indicative improvement over the core of the approach of Garon et al. [83].
- Please notice that better modeling of rotations and attention learning of foreground extraction and occlusion handling improves the translation component, the one with the most variation in this problem. This is something that verifies our assumption that the two components are interconnected in terms of performance in this framework. Besides, the reader should not forget that the feedback-based pose rendering of the previous prediction shapes the bounding box that crops the both the “Predicted” and “Observed” frame pairs and has a crucial impact to estimating the object’s 3D position.
- *Foreground Attention:*
 - This, first, Spatial Attention module focuses on the object of interest while it is not occluded by the user’s hand. However, when it comes forward and covers it, it spreads its peaks to both visual attributes.
 - The appearance cue this module focuses more is the depiction of an animal on the cover of the Cookie Jar. This is indicative of the fact that it searches for regions of interest that would facilitate pose estimation, without an explicit such supervision.

• *Occlusion Attention:*

The Occlusion handling Attentional module prefers the same pattern as it focuses on the animal depicted on the Cookie Jar texture when it is unoccluded. In general, it successfully highlights the parts of the object that stay out of the dynamic trajectory of the user’s grip. This is something that gives a strong visual hint as far as it concerns the object’s 3D position in space.

Ours+Optimization of Unique Symmetry parameter (*subfig(d)*):

- Providing a single, frozen, learnable continuous symmetry rotational parameter reduces the symmetry-based pose error of rotations. In fact, since almost no extra rotation is added by the user, it gives the highest accuracy, along with the next variation.

Ours+Mean of a Batch of frozen,learnable continuously rotational Symmetry parameters (*subfig(e)*):

- This, next, variation is the mean of a batch of frozen such learnable parameters that have been optimized during training. It is one of the two cases with the best results that succeed to reduce symmetry-base pose jitter in rotation estimation. It is indicative that the translation component error is improved, as well.

Ours+per Viewpoint Regression of Symmetry parameter (*subfig(f)*):

- Our last approach of regressing this symmetry parameter exceeds the performance of our asymmetrical rotational modeling but it is slightly subpar to the equivalent frozen parameter ones. We blame the fact that the tracker is in reality tested in a single rotational configuration in this scenario and for this reason, the rotation estimation through symmetry pre-pose pair regression is subconstrained with respect to the realistic needs of the problem.

Ours+Optimal Selection out of a Batch of frozen,learnable,continuously rotational Symmetry parameters (*subfig(g)*):

- Evidently the proposed symmetry parameter selection method results in optimal predictions. Due to the fact that this scenario contains slight amounts of rotational movement, we observe certain values of the batch having high frequencies of selection, as the classification layer approximates the functionality of the uniquely learnable symmetry parameter selection choice.
- However, the resulted pose errors with this approach are significantly lower than those produced by the “Unique-solution” one. This is the effect of leaving the degree of freedom of selecting the optimal parameter during inference unconstrained, instead of consolidating a value during training and keeping it frozen during the testing stage, at an extra computational burden, of course.

Error Plotting (left:re-iterate every 15 frames) and (right:re-iterate when the tracker fails):

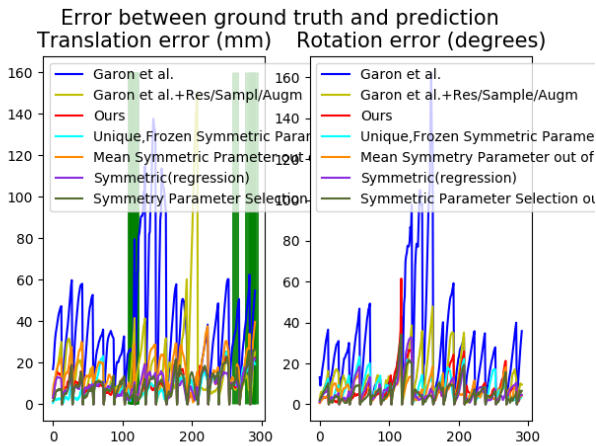


Figure 8.155 **Special case of rotational symmetry parameter estimation: 3D Translational and Rotational error metrics’ plot, for the “Rotation Only Interaction” scenario of the Cookie Jar 3D CAD model, in the case of tracker re-iteration only after a failure.**

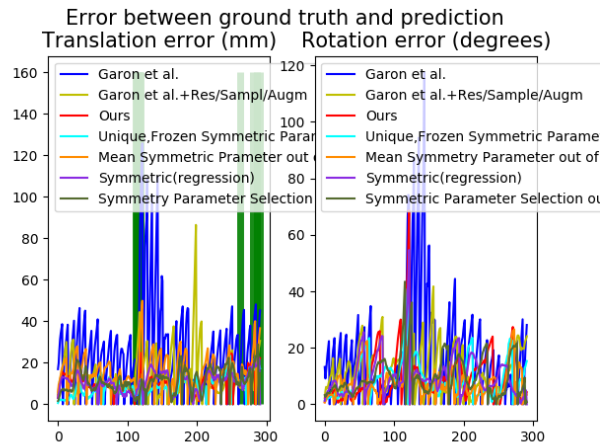


Figure 8.156 **Special case of rotational symmetry parameter estimation: 3D Translational and Rotational error metrics’ plot, for the “Translation Only Interaction” scenario of the Cookie Jar 3D CAD model, in the case of tracker re-iteration only after a failure.**

Free 3D Object Movement:Rotation Only

Architecture	Translational Error (mm)	Rotational Error (o)	Fails
Approach of Garon et al. [83] (20k training pairs) (<i>Symmetric assumption</i>) re-iterate every 15 frames	62.78 ± 46.26	27.33 ± 26.54	-
Approach of Garon et al. [83] (20k training pairs) (<i>Symmetric assumption</i>) re-iterate every time the tracker fails	44.35 ± 35.03	19.75 ± 14.06	31
Approach of Garon et al. [83] with Residual inter-layer connections and improved sampling/initialization/augmentation strategy (20k training pairs) (<i>Symmetric assumption</i>) re-iterate every 15 frames	12.70 ± 18.09	22.03 ± 16.02	-
Approach of Garon et al. [83] with Residual inter-layer connections and improved sampling/initialization/augmentation strategy (20k training pairs) (<i>Symmetric assumption</i>) re-iterate every time the tracker fails	11.16 ± 10.20	18.24 ± 11.86	19
Approach of Chapter 6 (20k training pairs) (<i>Asymmetric assumption</i>) re-iterate every 15 frames	10.22 ± 7.21	20.53 ± 18.03	-
Approach of Chapter 6 (20k training pairs) (<i>Asymmetric assumption</i>) re-iterate every time the tracker fails	10.87 ± 8.14	16.46 ± 12.49	16
Approach of Chapter 6 (20k training pairs) (<i>Unique Parameter Symmetric assumption</i>) re-iterate every 15 frames	21.52 ± 19.18	20.04 ± 17.41	-
Approach of Chapter 6 (20k training pairs) (<i>Unique Parameter Symmetric assumption</i>) re-iterate every time the tracker fails	19.09 ± 15.32	15.50 ± 11.20	16
Approach of Chapter 6 (20k training pairs) (<i>Batch of Parameters Symmetric assumption</i>) re-iterate every 15 frames	10.32 ± 11.47	13.97 ± 14.87	-
Approach of Chapter 6 (20k training pairs) (<i>Batch of Parameters Symmetric assumption</i>) re-iterate every time the tracker fails	12.38 ± 11.69	15.93 ± 12.16	16
Approach of Chapter 6 (20k training pairs) (<i>Parameter Regression Symmetric assumption</i>) re-iterate every 15 frames	13.26 ± 8.09	20.79 ± 17.33	-
Approach of Chapter 6 (20k training pairs) (<i>Parameter Regression Symmetric assumption</i>) re-iterate every time the tracker fails	13.03 ± 6.88	17.25 ± 12.40	16
Approach of Chapter 6 (20k training pairs) (<i>Optimal Selection out of a uniform Batch of Parameters Symmetric assumption</i>) re-iterate every 15 frames	9.98 ± 10.63	13.84 ± 11.87	-
Approach of Chapter 6 (20k training pairs) (<i>Optimal Selection out of a uniform Batch of Parameters Symmetric assumption</i>) re-iterate every time the tracker fails	10.04 ± 7.42	10.00 ± 8.06	16

Table 8.21: 3D Translational and Rotational Pose Errors and Fail count of the “Cookie Jar” model, for the “Rotation Only Interaction” scenario for the baseline approach of Garon et al. [83], without/with sampling/initialization/augmentation improvements we propose and the approach of Chapter 6.

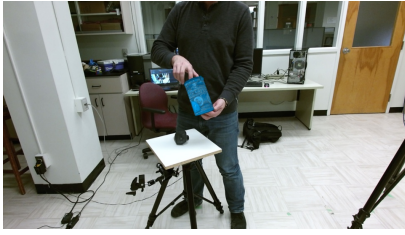


Figure 8.157 (a) Demonstration of an “Observed” RGB frame of the “Rotation Only Interaction” scenario with a rendered predicted pose of the “Cookie Jar” model generated by the baseline tracker of Garon et al.[83].

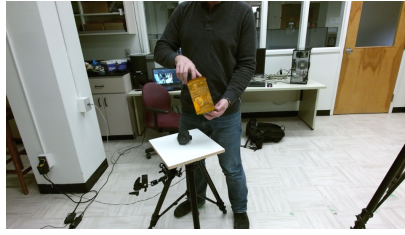


Figure 8.158 (b) Demonstration of an “Observed” RGB frame of the “Rotation Only Interaction” scenario with a rendered predicted pose of the “Cookie Jar” model generated by the baseline tracker of Garon et al.[83] with Residual inter-layer connections and sampling/initialization/augmentation improvements.



Figure 8.159 (c) Demonstration of an “Observed” RGB frame of the “Rotation Only Interaction” scenario with a rendered predicted pose of the “Cookie Jar” model generated by proposed tracker of this work.



Figure 8.160 (d) Demonstration of an “Observed” RGB frame of the “Rotation Only Interaction” scenario with a rendered predicted pose of the “Cookie Jar” model generated by proposed tracker, with the special case of unique rotational symmetry parameter, that remains frozen after training, of this work.



Figure 8.161 (e) Demonstration of an “Observed” RGB frame of the “Rotation Only Interaction” scenario with a rendered predicted pose of the “Cookie Jar” model generated by proposed tracker, with a learnable batch of rotational parameters, of this work.



Figure 8.162 (f) Demonstration of the Occlusion Handling Attention map provided by the proposed tracker, with a rotational parameter regression, in the “Rotation Only Interaction” scenario and tested on the “Cookiejar” 3D model.



Figure 8.163 (g) Demonstration of the Occlusion Handling Attention map provided by the proposed tracker, with a rotational parameter regression, in the “Rotation Only Interaction” scenario and tested on the “Cookiejar” 3D model.



Figure 8.164 (a)
Demonstration of the Occlusion Handling Attention map of the “Cookie Jar” model, provided by the proposed tracker in the “Rotation Only Interaction” scenario.



Figure 8.165 (b)
Demonstration of the Occlusion Handling Attention map of the “Cookie Jar” model, provided by the proposed tracker in the “Rotation Only” scenario.



Figure 8.166 (c)
Demonstration of the Occlusion Handling Attention map of the “Cookie Jar” model, provided by the proposed tracker, with a unique, frozen learnable continuous rotational symmetry parameter, in the “Rotation Only Interaction” scenario.

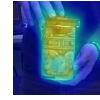


Figure 8.167 (d)
Demonstration of the Occlusion Handling Attention map of the “Cookie Jar” model, provided by the proposed tracker, with a unique, frozen learnable rotational symmetry parameter, in the “Rotation Only” scenario.



Figure 8.168 (e)
Demonstration of the Occlusion Handling Attention map of the “Cookie Jar” model, provided by the proposed tracker, with the mean of a batch of frozen, learnable continuous rotational symmetry parameter, in the “Rotation Only Interaction” scenario.



Figure 8.169 (f)
Demonstration of the Occlusion Handling Attention map of the “Cookie Jar” model, provided by the proposed tracker, with the mean of a batch of frozen, learnable rotational symmetry parameters, in the “Translation Only” scenario.



Figure 8.170 (g)
Demonstration of the Occlusion Handling Attention map of the “Cookie Jar” model, provided by the proposed tracker, with a per-pose regressed learnable rotational symmetry parameter, in the “Rotation Only Interaction” scenario.



Figure 8.171 (h)
Demonstration of the Occlusion Handling Attention map of the “Cookie Jar” model, provided by the proposed tracker, with a per-pose pair regressed learnable rotational symmetry parameter, in the “Translation Only” scenario.



Figure 8.172 (i)
Demonstration of the Occlusion Handling Attention map of the “Cookie Jar” model, provided by the proposed tracker, with a per-pose regressed learnable rotational symmetry parameter, in the “Rotation Only Interaction” scenario.



Figure 8.173 (j)
Demonstration of the Occlusion Handling Attention map of the “Cookie Jar” model, provided by the proposed tracker, with a per-pose pair regressed learnable rotational symmetry parameter, in the “Translation Only” scenario.

Qualitative analysis:

Baseline tracker of Garon et al [83](*subfig(a)*):

- The baseline tracker of Garon et al. [83] adds a great amount of pose jitter at the beginning of the video sequence. However, predictions are smoothed out at the later stages of the clip.
- We observe significant rotational errors whose source is the suboptimal modeling of rotational representation, as well as the improper rotation loss function selection. As a result, rotation estimations do not remain free from noise for long temporal intervals and small rotational errors accumulate fast and result in many irrecoverable failures.
- We, also, observe the major negative impact that dynamic hand occlusions have on rotation estimations as they change at a wide range over time.

Baseline with Res-connections+Sampling/Augmentation amelioration(*subfig(b)*):

- The results that are produced by improving the initial architectural characteristics and the sampling/initialization/augmentation strategy confirm our intuition. The improvement they yield is less impressive in this scenario as it is evident that most of the overall rotational refinement of the predictions of our overall approach come from the proper rotational component modeling.
- However, its impact is important for translation, whose (small either way) jitter is eliminated.

Ours(*subfig(c)*):

- Our proper rotational modeling reduces the relative error metric significantly and saves the tracker from frequent irrecoverable rotational failures.
- *Foreground Attention:*
Sharp attentional peaks focus on visual attributes of particular patterns. As the object is rotated in high speed, their attention shifts to the object region furthest from its 3D center and transitions from visual cues that were visible in the beginning of the motion to newer ones that are visible at its end.
- *Occlusion Attention:*
On the other hand, the Occlusion handling Attention module is more robust to the change of direction and large 3D rotational motions. However, it also focuses on particular patterns that help figuring out the object’s pose. This is not something we have explicitly supervised our auxiliary loss for.

Ours+Optimization of Unique Symmetry parameter (*subfig(d)*):

- The unique, frozen learnable continuous rotational symmetry parameter we use for handling the object’s rotational ambiguity is helpful in reducing the symmetry-induced pose error. However, it is not the optimal one.

Ours+Mean of a Batch of frozen,learnable continuously rotational Symmetry parameters (*subfig(e)*):

- Using the mean of a batch of frozen, learnable rotational symmetry parameters improves the results of the previous, unique approach and slightly reduces tracking failures.

Ours+per Viewpoint Regression of Symmetry parameter (*subfig(f)*):

- This, last approach yields the optimal results for this scenario, as the regression is finetuned for each particular pose pair instead of a mean of a crude, discrete, batch of frozen symmetry parameters.

Ours+Optimal Selection out of a Batch of frozen,learnable,continuously rotational Symmetry parameters (*subfig(g)*):

- Our, finally, proposed method clearly outperforms all others in, almost, all pose errors metrics and statistical tests. Online optimal selections out of a pool of candidate learned parameters leaves enough freedom to the algorithm to select the parameter that suits the CNN the most, in order to achieve its goal in disentangling the two kinds of 3D rotation and, thus, improve both the tracker’s accuracy and robustness to jitter.
- However, here, we need to note that this amelioration affects mostly the peaks of tracking failures, as their number and timing remains, more or less, intact accross all symmetry parameter estimation methods. As a matter of fact, the “Rotation Only” scenario is proven to be an extremely challenging benchmark for our CNN and witnesses a margin for future improvement that could be achieved with more dilligent mitigating of pose drift across time.

Error Plotting (left:re-iterate every 15 frames) and (right:re-iterate when the tracker fails):

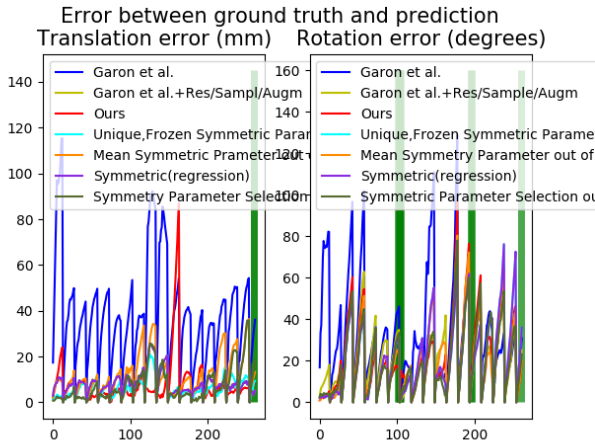


Figure 8.174 **Special case of rotational symmetry parameter estimation:** 3D Translational and Rotational error metrics’ plot, for the “Rotation Only Interaction” scenario of the Cookie Jar 3D CAD model, in the case of tracker re-iteration only after a failure.

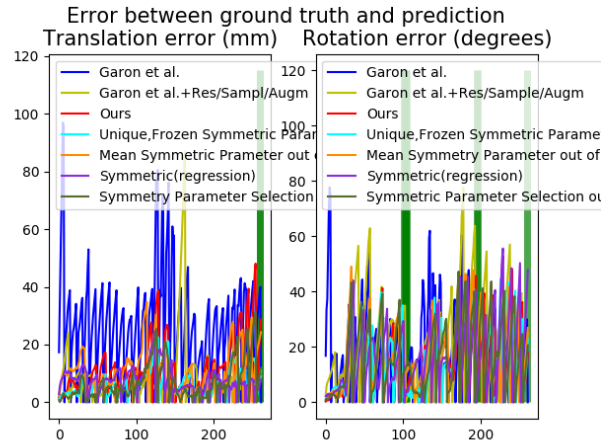


Figure 8.175 **Special case of rotational symmetry parameter estimation:** 3D Translational and Rotational error metrics’ plot, for the “Rotation Only Interaction” scenario of the Cookie Jar 3D CAD model, in the case of tracker re-iteration only after a failure.

Here, we also have the chance to present the angular distance between the predicted and the ground truth z-rotations in order to acquire a more precise notion of the effect each symmetry handling variation of our approach has on the accuracy of predicting the object’s orientation w.r.t. its symmetry axis.

Architecture	$d_A^{(o)}(\hat{r}(t), r_{G.T.}(t))$
Approach of Garon et al. [83] (20k training pairs) re-iterate every 15 frames	10.60 ± 38.90
Approach of Chapter 6 (20k training pairs) (<i>Asymmetric assumption</i>) re-iterate every 15 frames	4.60 ± 35.42
Approach of Chapter 6 (20k training pairs) (<i>Unique Parameter Symmetric assumption</i>) re-iterate every 15 frames	2.98 ± 25.07
Approach of Chapter 6 (20k training pairs) (<i>Batch of Parameters Symmetric assumption</i>) re-iterate every 15 frames	2.84 ± 24.95
Approach of Chapter 6 (20k training pairs) (<i>Parameter Regression Symmetric assumption</i>) re-iterate every 15 frames	2.16 ± 29.25
Approach of Chapter 6 (20k training pairs) (<i>Optimal Selection out of a uniform Batch of Parameters Symmetric assumption</i>) re-iterate every 15 frames	2.07 ± 24.52

Table 8.22: Angular distance between the predicted and ground truth z-rotational components (in degrees) of the “Cookie Jar” model, for the “Rotation-Only Interaction” scenario for the baseline approach of Garon et al. [83], the approach of Chapter 6 and its symmetry handling variations.

It is more than evident that accounting for symmetries makes a significant difference from this perspective and that our approach is the one that emerges as the golden medium between optimal accuracy and robustness.

Free 3D Object Movement: Translation and Rotation

Architecture	Translational Error (mm)	Rotational Error (o)	Fails
Approach of Garon et al. [83] (20k training pairs) (<i>Symmetric assumption</i>) re-iterate every 15 frames	54.05 ± 29.90	31.12 ± 27.96	-
Approach of Garon et al. [83] (20k training pairs) (<i>Symmetric assumption</i>) re-iterate every time the tracker fails	38.53 ± 23.11	19.10 ± 15.76	33
Approach of Garon et al. [83] with Residual inter-layer connections and improved sampling/initialization/augmentation strategy (20k training pairs) (<i>Symmetric assumption</i>) re-iterate every 15 frames	24.37 ± 19.01	22.84 ± 20.34	-
Approach of Garon et al. [83] with Residual inter-layer connections and improved sampling/initialization/augmentation strategy (20k training pairs) (<i>Symmetric assumption</i>) re-iterate every time the tracker fails	20.17 ± 14.07	17.24 ± 12.93	22
Approach of Chapter 6 (20k training pairs) (<i>Asymmetric assumption</i>) re-iterate every 15 frames	17.03 ± 11.94	22.24 ± 20.86	-
Approach of Chapter 6 (20k training pairs) (<i>Asymmetric assumption</i>) re-iterate every time the tracker fails	17.75 ± 11.60	16.70 ± 14.63	20
Approach of Chapter 6 (20k training pairs) (<i>Unique Parameter Symmetric assumption</i>) re-iterate every 15 frames	24.04 ± 15.17	24.47 ± 23.46	-
Approach of Chapter 6 (20k training pairs) (<i>Unique Parameter Symmetric assumption</i>) re-iterate every time the tracker fails	21.96 ± 13.74	17.42 ± 15.56	18
Approach of Chapter 6 (20k training pairs) (<i>Batch of Parameters Symmetric assumption</i>) re-iterate every 15 frames	14.63 ± 11.19	15.93 ± 16.85	-
Approach of Chapter 6 (20k training pairs) (<i>Batch of Parameters Symmetric assumption</i>) re-iterate every time the tracker fails	16.80 ± 11.83	18.90 ± 15.37	16
Approach of Chapter 6 (20k training pairs) (<i>Parameter Regression Symmetric assumption</i>) re-iterate every 15 frames	13.78 ± 10.02	17.43 ± 16.91	-
Approach of Chapter 6 (20k training pairs) (<i>Parameter Regression Symmetric assumption</i>) re-iterate every time the tracker fails	15.55 ± 8.11	19.76 ± 14.43	16
Approach of Chapter 6 (20k training pairs) (<i>Optimal Selection out of a uniform Batch of Parameters Symmetric assumption</i>) re-iterate every 15 frames	14.63 ± 11.19	15.71 ± 13.80	-
Approach of Chapter 6 (20k training pairs) (<i>Optimal Selection out of a uniform Batch of Parameters Symmetric assumption</i>) re-iterate every time the tracker fails	12.88 ± 8.74	14.92 ± 12.80	21

Table 8.23: 3D Translational and Rotational Pose Errors and Fail count of the “Cookie Jar” model, for the “Full Interaction” scenario for the baseline approach of Garon et al. [83], without/with sampling/ initialization/augmentation improvements we propose and the approach of Chapter 6 and its symmetry handling variations.

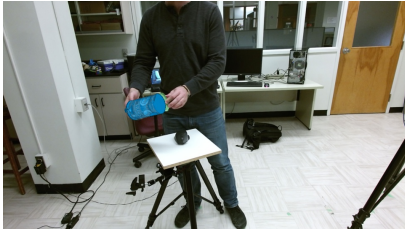


Figure 8.176 (a) Demonstration of an “Observed” RGB frame of the “Full Interaction” scenario with a rendered predicted pose of the “Cookie Jar” model generated by the baseline tracker of Garon et al.[83].



Figure 8.177 (b) Demonstration of an “Observed” RGB frame of the “Full Interaction” scenario with a rendered predicted pose of the “Cookie Jar” model generated by the baseline tracker of Garon et al.[83] with Residual inter-layer connections and sampling/initialization/augmentation improvements.



Figure 8.178 (c) Demonstration of an “Observed” RGB frame of the “Full Interaction” scenario with a rendered predicted pose of the “Cookie Jar” model generated by proposed tracker of this work.

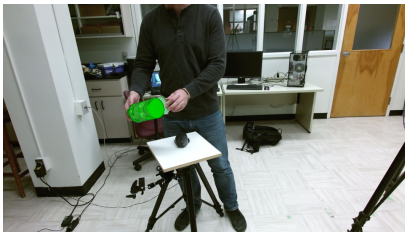


Figure 8.179 (d) Demonstration of an “Observed” RGB frame of the “Full Interaction” scenario with a rendered predicted pose of the “Cookie Jar” model generated by proposed tracker, with the special case of unique rotational symmetry parameter, that remains frozen after training, of this work.



Figure 8.180 (e) Demonstration of an “Observed” RGB frame of the “Full Interaction” scenario with a rendered predicted pose of the “Cookie Jar” model generated by proposed tracker, with a learnable batch of rotational parameters, of this work.

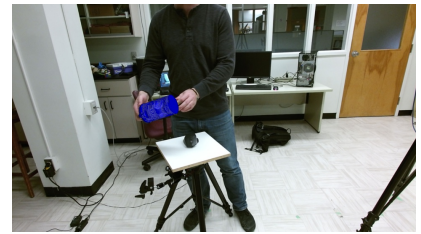


Figure 8.181 (f) Demonstration of the Occlusion Handling Attention map provided by the proposed tracker, with a rotational parameter regression, in the “Full Interaction” scenario and tested on the “Cookiejar” 3D model.



Figure 8.182 (g) Demonstration of the Occlusion Handling Attention map provided by the proposed tracker, with per-pose pair optimal selection out of a nearly uniform batch of continuously rotational symmetry parameters, in the “Full Interaction” scenario and tested on the “Cookiejar” 3D model.



Figure 8.183 (a)
Demonstration of the Occlusion Handling Attention map of the “Cookie Jar” model, provided by the proposed tracker in the “Full Interaction” scenario.

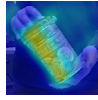


Figure 8.184 (b)
Demonstration of the Occlusion Handling Attention map of the “Cookie Jar” model, provided by the proposed tracker in the “Full Interaction” scenario.



Figure 8.185 (c)
Demonstration of the Occlusion Handling Attention map of the “Cookie Jar” model, provided by the proposed tracker, with a unique, frozen learnable continuous rotational symmetry parameter, in the “Full Interaction” scenario.



Figure 8.186 (d)
Demonstration of the Occlusion Handling Attention map of the “Cookie Jar” model, provided by the proposed tracker, with a unique, frozen learnable rotational symmetry parameter, in the “Full Interaction” scenario.



Figure 8.187 (e)
Demonstration of the Occlusion Handling Attention map of the “Cookie Jar” model, provided by the proposed tracker, with the mean of a batch of frozen learnable continuous rotational symmetry parameter, in the “Full Interaction” scenario.

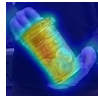


Figure 8.188 (f)
Demonstration of the Occlusion Handling Attention map of the “Cookie Jar” model, provided by the proposed tracker, with the mean of a batch of frozen, learnable rotational symmetry parameters, in the “Rotation Only” scenario.



Figure 8.189 (g)
Demonstration of the Occlusion Handling Attention map of the “Cookie Jar” model, provided by the proposed tracker, with a per-pose regressed learnable continuous rotational symmetry parameter, in the “Full Interaction” scenario.

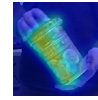


Figure 8.190 (h)
Demonstration of the Occlusion Handling Attention map of the “Cookie Jar” model, provided by the proposed tracker, with a per-pose pair learnable rotational symmetry parameter, in the “Full Interaction” scenario.

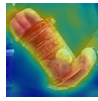


Figure 8.191 (i)
Demonstration of the Occlusion Handling Attention map of the “Cookie Jar” model, provided by the proposed tracker, with a per-pose pair optimal selection out of a nearly uniform batch of continuously rotational symmetry parameters, in the “Full Interaction” scenario.

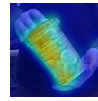


Figure 8.192 (j)
Demonstration of the Occlusion Handling Attention map of the proposed model, provided by the proposed tracker, with a per-pose pair optimal selection out of a nearly uniform batch of continuously rotational symmetry parameters, in the “Full Interaction” scenario.

Qualitative analysis:

Baseline tracker of Garon et al[83](*subfig(a)*):

- The baseline tracker of Garon et al. [83] presents significant errors in scale changes if significant range that spread throughout an important time interval.
- Furthermore, we observe that its estimations are noisy (both in terms of rotational symmetry and not) and especially for the first few frames when the complete object motion has not started yet.

Baseline with Res-connections+Sampling/Augmentation amelioration(*subfig(b)*):

- These first improvements help the tracker to adapt better to scale change and generalize to a wider range of them.
- Moreover, we yield smaller pose drifts that accumulate over longer periods of time and, thus, create less tracking failures.

- A drawback that we observe is that, albeit the aforementioned improvements, tracking of this object fails under medium speed profiles.
- Another con is that the problem of rotational pose jitter is present in this scenario as well, when this particular architectural variation is employed.

Ours(*subfig(c)*):

- Our approach solves the problem of medium speeds. However, it does not succeed to eliminate medium and big errors in the fastest temporal intervals.
- Nevertheless, symmetry-induced errors are reduced but still present.
- *Foreground Attention:* Similarly to the “Dragon” case, the attention is focused on the object parts (in general, in a rather uniform distribution) and spreads to the object of interest and the user’s hand, when it grasps it and starts to move it around.
- *Occlusion Attention:* On the contrary, this second module is focused solely on the object’s characteristics. Due to the simple object shape of this, second, case, implicit keypoint learning is not so obvious here. However, one could easily notice that the tracker focuses on particular appearance patterns of coherent shape, when it has the chance.

Ours+Optimization of Unique Symmetry parameter (*subfig(d)*):

- Contrarily to the previous scenarios where the object’s rotation is steadier, here, the unique, frozen symmetry parameter reports a rotational error lower even than the asymmetrical approach. This means that we overconstrain this aspect of the problem for this scenario by using a single parameter.

Ours+Mean of a Batch of frozen,learnable continuously rotational Symmetry parameters (*subfig(e)*):

- Estimating the continuous symmetry rotational parameter through the batch of frozen learned parameters saves the tracker from developing high rotational errors in high speed intervals.
- Here, two particular patterns emerge for the parallel Attention modules:
 - *Foreground Attention:* Its peaks are sharper when the user’s hand covers the object and they move together in the same direction both in terms of translation and rotation.
 - *Occlusion Attention:* On the other hand, the Occlusion handling Spatial Attention weight focuses more on the part of the object that has the most significant visual difference through this motion. For example, if only the top half of the object is covered by the hand of the user and the object performs exclusively a rotational move, this module will add more weight to the parts of the object that stand the furthest away from its 3D center.

Ours+per Viewpoint Regression of Symmetry parameter (*subfig(f)*):

- It presents one of the top two accuracy measurements, along with the previous approach. This is intuitive as the extra degree of freedom this approach offers is better utilized in this scenario in comparison with the other ones where the object is rotationally stable for the most part.
- Although, one has to admit that it inserts some initial pose jitter in the beginning that the Cookie Jar is at the aforementioned state.
- As for the dual Attention modules, our observations here are no different in comparison to the previous approach.

Ours+Optimal Selection out of a Batch of frozen,learnable,continuously rotational Symmetry parameters (*subfig(g)*):

- Here, we note that our proposed approach comes as a close second to the accuracy provided by its “Free symmetry parameter Regression” counterpart as far as the translational component is concerned. On the contrary, it is (by a small margin, indeed) the optimal method for proper rotational estimation.

- However, where our approach gets beaten is at the counting of complete tracking failures, something that showcases that the frozen batch size of symmetry parameters is inadequate to express the full complexity of the full 3D movement.

Error Plotting (left:re-iterate every 15 frames) and (right:re-iterate when the tracker fails):

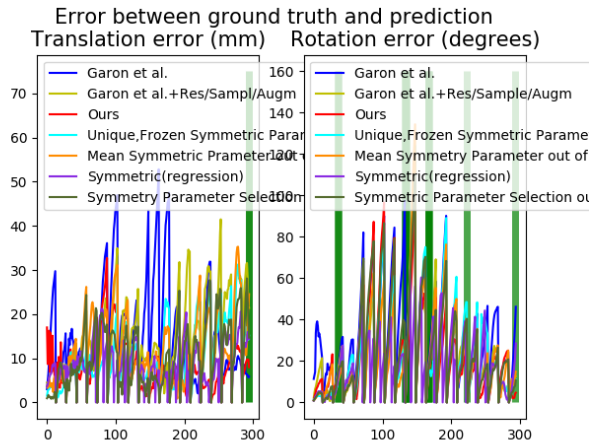


Figure 8.193 **Special case of rotational symmetry parameter estimation:** 3D Translational and Rotational error metrics' plot, for the "Full Interaction" scenario of the 3D CAD model, in the case of tracker re-iteration only after a failure.

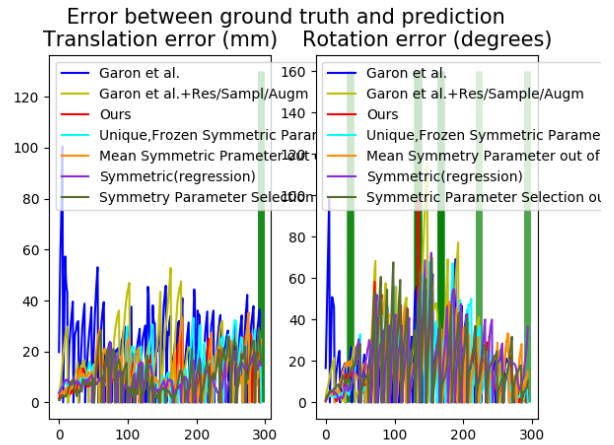


Figure 8.194 **Special case of rotational symmetry parameter estimation:** 3D Translational and Rotational error metrics' plot, for the "Full Interaction" scenario of the 3D CAD model, in the case of tracker re-iteration only after a failure.

Free 3D Object Pose Tracking - the 'Hard Interaction' scenario:



Figure 8.195 (a) Demonstration of an "Observed" RGB frame of the "Hard Interaction" scenario with a rendered predicted pose of the tracking model generated by the baseline tracker of Garon et al.[83].

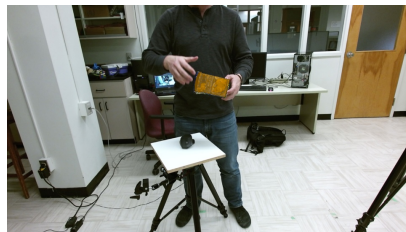


Figure 8.196 (b) Demonstration of an "Observed" RGB frame of the "Hard Interaction" scenario with a rendered predicted pose of the tracking model generated by the baseline tracker of Garon et al.[83] with Residual inter-layer connections and sampling/initialization/augmentation improvements.



Figure 8.197 (c) Demonstration of an "Observed" RGB frame of the "Hard Interaction" scenario with a rendered predicted pose of the tracking model generated by proposed tracker of this work.

Architecture	Translational Error (mm)	Rotational Error (o)	Fails
Approach of Garon et al. [83] (20k training pairs) (<i>Symmetric assumption</i>) re-iterate every 15 frames	10.75 ± 6.89	23.53 ± 18.85	-
Approach of Garon et al. [83] (20k training pairs) (<i>Symmetric assumption</i>) re-iterate every time the tracker fails	15.20 ± 9.56	18.41 ± 12.96	15
Approach of Garon et al. [83] with Residual inter-layer connections and improved sampling/initialization/augmentation strategy (20k training pairs) (<i>Symmetric assumption</i>) re-iterate every 15 frames	9.49 ± 5.97	15.46 ± 13.53	-
Approach of Garon et al. [83] with Residual inter-layer connections and improved sampling/initialization/augmentation strategy (20k training pairs) (<i>Symmetric assumption</i>) re-iterate every time the tracker fails	14.44 ± 8.76	16.77 ± 11.03	11
Approach of Chapter 6 (20k training pairs) (<i>Asymmetric assumption</i>) re-iterate every 15 frames	10.87 ± 8.94	20.55 ± 18.06	-
Approach of Chapter 6 (20k training pairs) (<i>Asymmetric assumption</i>) re-iterate every time the tracker fails	16.32 ± 17.83	17.54 ± 16.00	11
Approach of Chapter 6 (20k training pairs) (<i>Unique Parameter Symmetric assumption</i>) re-iterate every 15 frames	11.38 ± 8.94	16.26 ± 14.11	-
Approach of Chapter 6 (20k training pairs) (<i>Batch of Parameters Symmetric assumption</i>) re-iterate every 15 frames	11.98 ± 9.23	13.84 ± 11.87	-
Approach of Chapter 6 (20k training pairs) (<i>Batch of Parameters Symmetric assumption</i>) re-iterate every time the tracker fails	16.80 ± 11.83	18.90 ± 15.37	9
Approach of Chapter 6 (20k training pairs) (<i>Parameter Regression Symmetric assumption</i>) re-iterate every 15 frames	13.03 ± 6.88	17.25 ± 12.40	-
Approach of Chapter 6 (20k training pairs) (<i>Parameter Regression Symmetric assumption</i>) re-iterate every time the tracker fails	15.74 ± 11.87	17.62 ± 15.01	9
Approach of Chapter 6 (20k training pairs) (<i>Optimal Selection out of a uniform Batch of Parameters Symmetric assumption</i>) re-iterate every 15 frames	10.43 ± 6.63	9.57 ± 10.01	-
Approach of Chapter 6 (20k training pairs) (<i>Optimal Selection out of a uniform Batch of Parameters Symmetric assumption</i>) re-iterate every time the tracker fails	9.97 ± 5.80	8.86 ± 9.91	8

Table 8.24: 3D Translational and Rotational Pose Errors and Fail count of the tracking model, for the “Hard Interaction” scenario for the baseline approach of Garon et al. [83], without/with sampling/ initialization/augmentation improvements we propose and the approach of Chapter 6 and its symmetry handling variations.

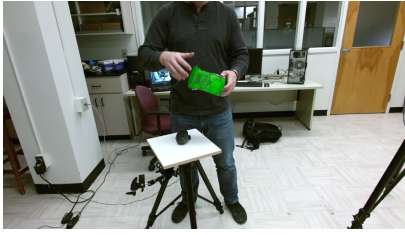


Figure 8.198 (d) Demonstration of an “Observed” RGB frame of the “Hard Interaction” scenario with a rendered predicted pose of the tracking model generated by proposed tracker, with the special case of unique rotational symmetry parameter, that remains frozen after training, of this work.



Figure 8.199 (e) Demonstration of an “Observed” RGB frame of the “Hard Interaction” scenario with a rendered predicted pose of the “” model generated by proposed tracker, with a learnable batch of rotational parameters, of this work.

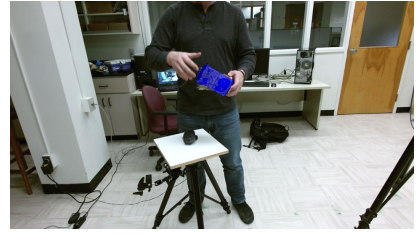


Figure 8.200 (f) Demonstration of the Occlusion Handling Attention map provided by the proposed tracker, with a rotational parameter regression, in the “Hard Interaction” scenario and tested on the “” 3D model.



Figure 8.201 (g) Demonstration of the Occlusion Handling Attention map provided by the proposed tracker, with a per-pose pair optimal selection out of a nearly uniform batch of continuously rotational symmetry parameters, in the “Hard Interaction” scenario and tested on the “” 3D model.



Figure 8.202 (a) Demonstration of the Foreground Extraction Attention map of the tracking model, provided by the proposed tracker in the “Hard Interaction” scenario.



Figure 8.203 (b) Demonstration of the Occlusion Handling Attention map of the tracking model, provided by the proposed tracker in the “Hard Interaction” scenario.



Figure 8.204 (c) Demonstration of the Occlusion Handling Attention map of the tracking model, provided by the proposed tracker, with a unique, frozen learnable continuous rotational symmetry parameter, in the “Hard Interaction” scenario.



Figure 8.205 (d) Demonstration of the Occlusion Handling Attention map of the tracking model, provided by the proposed tracker, with a unique, frozen learnable rotational symmetry parameter, in the “Hard Interaction” scenario.



Figure 8.206 (e)
 Demonstration of the Occlusion Handling Attention map of the tracking model, provided by the proposed tracker, with the mean of a batch of frozen learnable, continuous rotational symmetry parameters, in the “Hard Interaction” scenario.

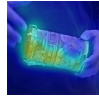


Figure 8.207 (f)
 Demonstration of the Occlusion Handling Attention map of the tracking model, provided by the proposed tracker, with the mean of a batch of frozen learnable rotational symmetry parameters, in the “Hard Interaction” scenario.



Figure 8.208 (g)
 Demonstration of the Occlusion Handling Attention map of the tracking model, provided by the proposed tracker, with a per-pose regressed learnable rotational symmetry parameter, in the “Hard Interaction” scenario.

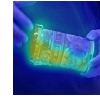


Figure 8.209 (h)
 Demonstration of the Occlusion Handling Attention map of the tracking model, provided by the proposed tracker, with a per-pose pair regressed learnable rotational symmetry parameter, in the “Hard Interaction” scenario.



Figure 8.210 (i)
 Demonstration of the Occlusion Handling Attention map of the tracking model, provided by the proposed tracker, with a per-pose pair optimal selection out of a nearly uniform batch of continuously rotational symmetry parameters, in the “Hard Interaction” scenario.



Figure 8.211 (j)
 Demonstration of the Occlusion Handling Attention map of the tracking model, provided by the proposed tracker, with a per-pose pair optimal selection out of a nearly uniform batch of continuously rotational symmetry parameters, in the “Hard Interaction” scenario.

Qualitative analysis:

Baseline tracker of Garon et al[83](*subfig(a)*):

- The baseline tracker’s predictions present intense jitter during the first few frames where the object is kept more or less still in the user’s hands. This jitter is not of the continuous symmetry-induced kind and is reduced over time thanks to the fact that the object never leaves the “Predicted” pose-based bounding box.
- The same happens with high and medium dynamic occlusions that (temporarily) throw off the tracker’s prediction, but it succeeds to recover it in time, before the object gets out of the aforementioned boundary box’s boundaries.
- We, also, observe that the tracker of Garon et al.[83] struggles with scale ambiguity.
- Under extreme rotations that keep on going for a long period and abrupt changes of direction, the tracker fails irrecoverably.
- The tracker is not particularly robust to occlusions of high degree. For example, in the last few frames that the object of interest is mostly covered by the user’s hand, the tracker completely loses its pose.

Baseline with Res-connections+Sampling/Augmentation amelioration(*subfig(b)*):

- The situation of this, first, alteration of the approach of Garon et al. [83] has not only significant quantitative differences, but also strong qualitative similarities with it. Small pose drifts are inherently smaller in this case and, as a result, they take longer to accumulate and cause the tracker to fail. That is the reason for the reduced number of irrecoverable tracking failures reported in the table above.
- Ostensibly, although the user is entirely covering the object in the last few frames and the tracker starts to present erroneous prediction, it manages to never lose complete track of the object’s position and make negative feedback an aggravating source of prediction noise.

Ours(*subfig(c)*):

- Our approach results in predictions with less pose jitter.
- The heuristic algorithm for handling discrete symmetry ambiguities saves us from adding extravagant rotational error measurement of about 180° in the mean error calculation and even prevents our estimate noise from spreading across the entirety of the object’s temporal trajectory. For visual details, see the relevant section, where we showcase the germane frames without/with the impact of our algorithm.
- An important visual observation is that the Cookie Jar’s lid is the most difficult geometric and appearance feature to use as an information source for tracking the object’s pose. That has to do with a multiplicity of reasons.
 1. Due to the rotationally symmetric nature of the model, the previous frame is less significant in this video subsequence since the object could result in this configuration, with the lid facing the camera, from infinitely many previous poses.
 2. That has a direct impact to the Network’s training, as it obviously confuses the proper source and target pose configuration due to the duality of the Cookie Jar’s lids in the model. It is not clear to it, which initial pose it has to hypothesize as the starting one, in order to predict the right future state. That has the extra impact of inserting noise to rotational components other than the principal axis of discrete symmetry (here, the x,y-axes (or x,y-plane)).

- *Foreground Attention:*

Again, similarly to the dragon case, the Foreground Attention module focuses on an extensive spatial area that contains not only object pixels, but the user’s hand as well. Thus, its peaks are not particularly sharp and the overall attentional impact has been balanced out between this module and its Occlusion Attention counterpart.

- *Occlusion Attention:*

This, siamese, Attentional modules has a slightly bigger impact to the tracker’s performance, as it highlights the most essential parts that disentangle the covered and uncovered regions of interest. It is something that can also be shown by the value the corresponding multitask loss converges to: the tracker ultimately gives a bolder assiduity to the occlusion parallel auxiliary task (which is objectively harder) than the foreground ones. Its peaks are more coherent, narrower and, typically, sharper.

However, due to the more plain shape and symmetric nature of the “Cookie Jar” model, our parallel Attentional modules have less of an impact in this case’s performance, in comparison to the more articulated dragon 3D model, with the richer visual texture.

Ours + Optimization of Unique Symmetry parameter (*subfig(d)*):

- The tracker has estimated and frozen a single learnable continuous rotational symmetry parameter and this is what it uses during the inference stage. Although it aids in stabilizing the rotational pose noise that was observed during the assymmetric handling case of the tracker, its impact is less beneficial in this scenario. Due to the abrupt and more fluid object motion, the rotational prediction produced by this variation obviously lack a necessary extra degree of rotational freedom, that would aid in completely disentangling symmetry ambiguities from the core of the rotation estimation subtask.

Ours+Mean of a Batch of frozen,learnable,continuously rotational Symmetry parameters (*subfig(e)*):

- Next, we try to cure this discrepancy with training and freezing a family (batch) of such rotational symmetry parameters and taking their mean as the ultimate tracker prediction. Our hope is that (similarly to the main task way of thinking), the tracker will learn each parameter in the batch with respect to a corresponding, different pose difference configuration and that those parameters will cover the pose space as densely as they can.
- Indeed, the tracker’s accuracy is increased in this variation and symmetry-induced jitter error is lowered.

Ours + per Viewpoint Regression of Symmetry parameter (*subfig(f)*):

- Last but not least, we proceed to the other extreme: that of regressing a per-pose continuous rotational symmetry parameter for each pose pair during training, and consequentially, leaving the tracker with the degree of freedom necessary to estimate the Symmetry rotation closest to the frame pair at hand and, thus, implicitly increasing the overall rotational estimation quality.
- We remember that we have seen that this approach does not yield the best results for scenarios where the object is static or does not move freely in 3D. However, in this scenario it rises as the optimal one and produces the lowest tracking errors with respect to both metrics.

Ours+Optimal Selection out of a Batch of frozen,learnable,continuously rotational Symmetry parameters (*subfig(g)*):

- Our proposed approach presents the less tracking failures and decreases 3D translational errors only to 66% and 3D rotational ones to the 50% of its closes alternative.
- It is evident, at a qualitative level, that the tracker outputs smoother predictions, lower error peaks across both components and is more robust where it matters the most, under rapid movement and high dynamic occlusion percentages, combined for the two pose components.
- The clear advantage this method has over all the others in the two most challenging scenarios for rotations: the “Rotation Only” and the “Hard Interaction” one ratifies our decision to select it as the optimal strategy-to-go for the general use case.

As a result, the overall golden medium that seems to result in the highest tracking accuracy is that of learning and freezing a (big enough) batch of rotational symmetry parameters and then using their mean to disentangle symmetry-caused ambiguities from the pure temporal rotational motion from frame to frame.

As expected, the two Parallel Attention modules present similar heatmaps to each of the asymmetric and symmetric handling variations of the tracker. If we were forced to recognize and report even the slightest of the differences, we would note the fact that the Occlusion handling Attention module presents slightly sharper peaks in the case of the per-pose regression of continuous rotational symmetry parameter. This is completely intuitive, as the reader must keep in her/his mind that the inference of these Attentional weight maps, as well as the rest of the tracker’s predictions, is an online procedure that is both directly and indirectly affected by the outcome quality of the prediction of the previous frame pair(s). This has not only to do with the feedback-rendering of the predicted pose to the “Predicted” frames, but also with the boundaries of the generated bounding box that crops the “Observed” input frame pair, as well, something that was not taken into consideration during training, since that cropping was performed once, during the generation stage, and was kept constant for the whole family of augmentation transformations.

Error Plotting (left:re-iterate every 15 frames) and (right:re-iterate when the tracker fails):

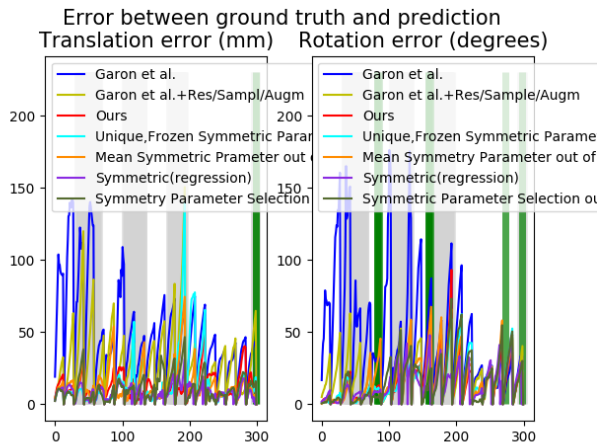


Figure 8.212 **Special case of rotational symmetry parameter estimation:** 3D Translational and Rotational error metrics' plot, for the “Hard Interaction” scenario of the 3D CAD model, in the case of tracker re-iteration only after a failure.

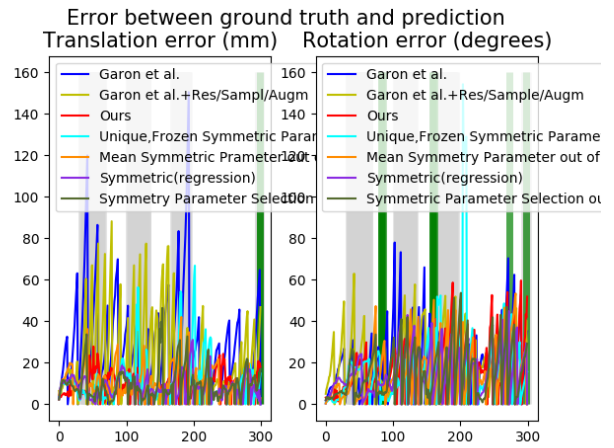
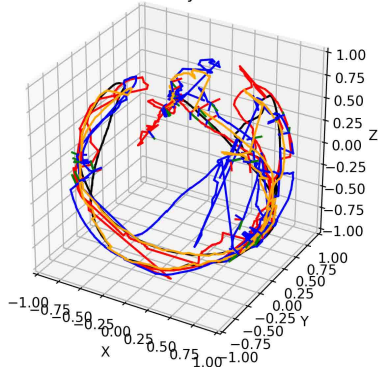


Figure 8.213 **Special case of rotational symmetry parameter estimation:** 3D Translational and Rotational error metrics' plot, for the “Hard Interaction” scenario of the 3D CAD model, in the case of tracker re-iteration only after a failure.

6D Temporal Pose Tracking Visualization
Cookie Jar --- Hard Interaction scenario
Reiterate every 15 frames



6D Temporal Pose Tracking Visualization
Cookie Jar --- Hard Interaction scenario
Reiterate at every tracking failure

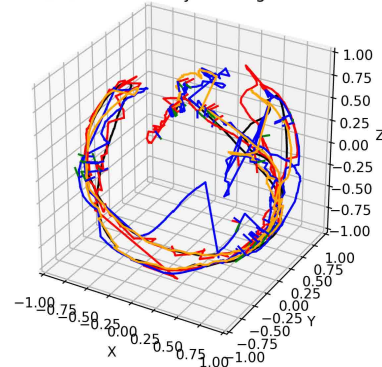


Figure 8.214: The 6D Pose Trajectories of the “Cookie Jar” object model, in time, for the “Hard Interaction” scenario. It is evident, in both cases, from this perspective as well, that our proposed approach outperforms the SoA of Garon et al. [83] by far, as it produces smaller errors and fewer failures. Also, the effectiveness of our symmetries' handling is ratified. All length units are in mm and all rotations in degrees.

General comments on the results concerning the two objects:

- First of all, we observe that the object case is much more difficult than the corresponding Dragon one. Zero failures are more rare there, if not non-existent. The source for this adversity should be searched for in the simplicity both of its shape and its texture, as well as in the appearance ambiguities that it discloses. It is not uncommon for the user to handle it in configurations that correspond to multiple pose results. Besides, the model lacks attributes of interest that stand out and can help recognize its pose, something extremely prominent in the “Dragon” case.
- Secondly, we can easily observe the need for incorporating the handling of extreme, discrete rotational ambiguities in the tracking loss function. Our algorithm heuristically sets a threshold for abandoning a rotational prediction as erroneous and replacing it with an older one (that we can next choose if we wish to re-pass it from a forward NN stream to refine it or not). This inserts the extra problem of tuning this threshold. However, if we try not to set it just at a logical value, but finetune it on the test set, we would severely harm the validity of our approach. Incorporating such a discrete decision (in the form of classification, for example) in the learning process we believe would be the solution to this discrepancy.

- Thirdly, our plots indicate that the user moves the “Dragon” at much higher speed profiles than the “Cookie Jar”. However, its distinctive features help the Network identify its pose more clearly than in the other case.
- Fourthly, contrarily to our initial intuitive belief, the hardest challenge for our tracker is not the one that combines all motion and occlusion patterns in a dynamic way. What seems to arise as the biggest challenge, on the contrary, is the scenario where the object is constantly occluded at higher percentage of its surface. We point out as a possible reason the fact that dynamic occlusions may be short in time and leave the tracker the freedom to catch up with the right estimation before the object gets out of the bounding box that is generated by each consecutive previous pose prediction.
- Fifthly, we observe the following peculiar situation. While it is obvious from our Ablation Study that estimating the object’s rotation is much more difficult than the corresponding translation task, our heuristic handling of abrupt discrete rotational ambiguities smooths out rotational errors at a significant margin since it disallows absurdly large rotational error outliers inserting in the calculation of our mean and standard deviation measures. However, things are not so disentangled between the two components, as we need to remind that ameliorating our rotational predictions has a positive impact to our next 3D translation estimation, as well.
- Lastly, we can understand that handling continuously symmetrical rotational ambiguities cannot be solved with a single, optimal approach. On the contrary, we see a clear pattern emerging as more degrees of freedom are added in selecting the Rotational Symmetry matrix of our choice, in order to diminish rotational errors: they match more complex object movements better. That is because the tracker has the flexibility to recalculate that part of rotation estimation in a viewpoint-dependent way that would minimize rotational errors “on the fly”.

8.5 Evaluation of our algorithms on other objects: Dog, Lego, Watering Can

Following, in order to establish the effectiveness of our tracker and to fully demonstrate its generalization capabilities, we report its results on 3 more objects: the “Dog”, the “Lego” block and the “Watering Can” models of the dataset of Garon et al.[83]. In the following figures, one may observe the estimated poses provided by the SoA and our tracker, blended on an “Observed” RGB frame each. Furthermore, we showcase the corresponding pair of Foreground and Occlusion handling attentions, as well as three **randomly selected** error plots (with a re-iteration every 15 frames). Finally, we present 3D Translational and Rotational errors, as well as overall tracking failures, for the most difficult scenarios of the dataset: the two “75% Occlusion” and all the “Interaction” scenarios.

8.5.1 Dog model

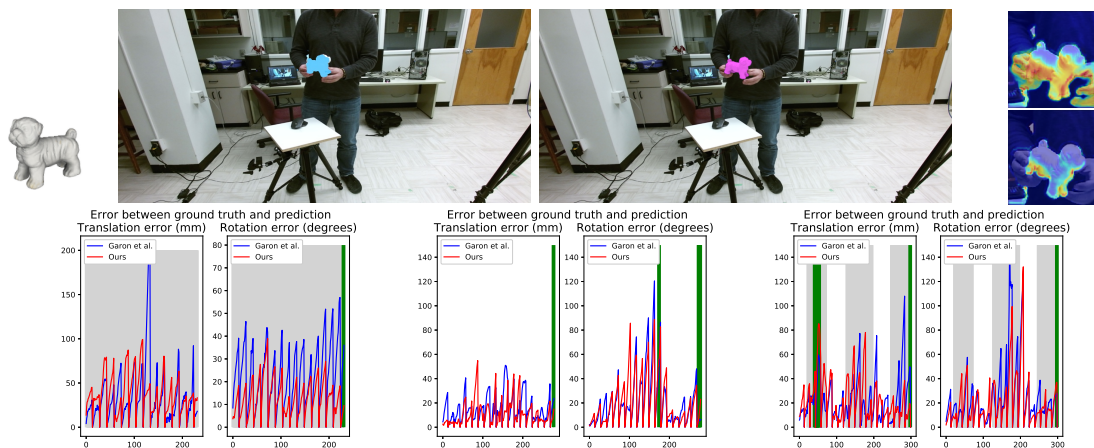
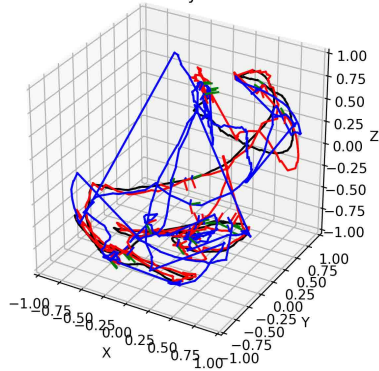


Figure 8.215: Comparison of the SoA[83] (light blue) and our (pink) approaches for the “Dog” in 3 scenarios: “75% Vertical Occlusion”, “Rotation Only” and “Hard Interaction”.

6D Temporal Pose Tracking Visualization
Dog --- Hard Interaction scenario
Reiterate every 15 frames



6D Temporal Pose Tracking Visualization
Dog --- Hard Interaction scenario
Reiterate at every tracking failure

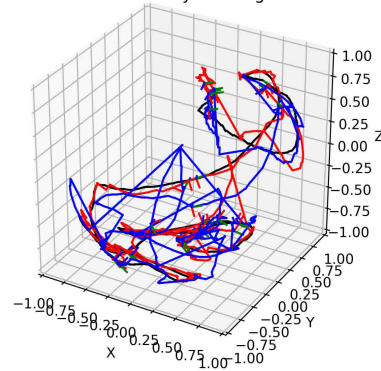


Figure 8.216: The 6D Pose Trajectories of the “Dog” object model, in time, for the “Hard Interaction” scenario. It is evident, in both cases, from this perspective as well, that our proposed approach outperforms the SoA of Garon et al. [83] by far, as it produces smaller errors and fewer failures. All length units are in mm and all rotations in degrees.

8.5.2 Lego model

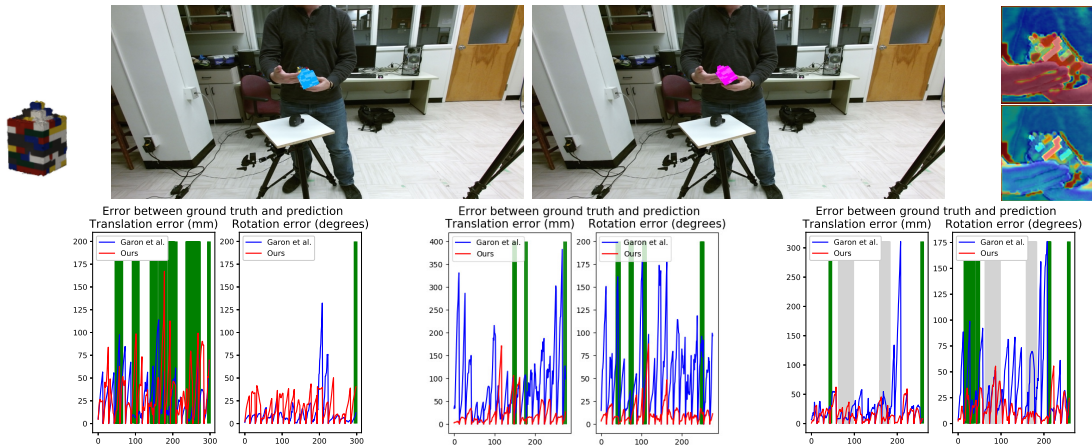
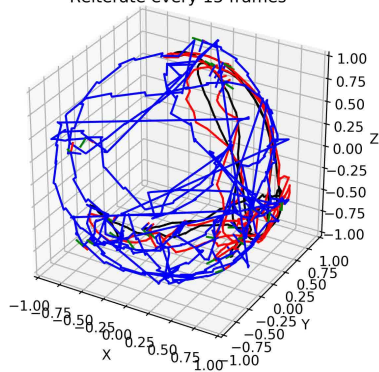


Figure 8.217: Comparison of the SoA [83] (light blue) and our (pink) approaches for the “Lego” in 3 scenarios: “Translation Only”, “Full” and “Hard Interaction”.

6D Temporal Pose Tracking Visualization
Lego --- Hard Interaction scenario
Reiterate every 15 frames



6D Temporal Pose Tracking Visualization
Lego --- Hard Interaction scenario
Reiterate at every tracking failure

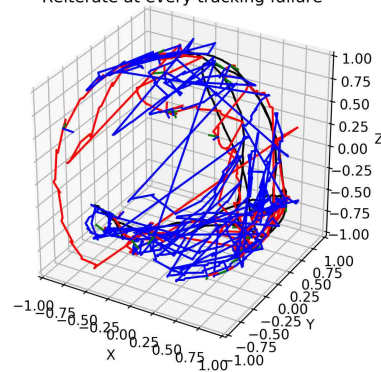


Figure 8.218: The 6D Pose Trajectories of the “Lego block” object model, in time, for the “Hard Interaction” scenario. It is evident, in both cases, from this perspective as well, that our proposed approach outperforms the SoA of Garon et al. [83] by far, as it produces smaller errors and fewer failures. All length units are in mm and all rotations in degrees.

8.5.3 Watering Can model

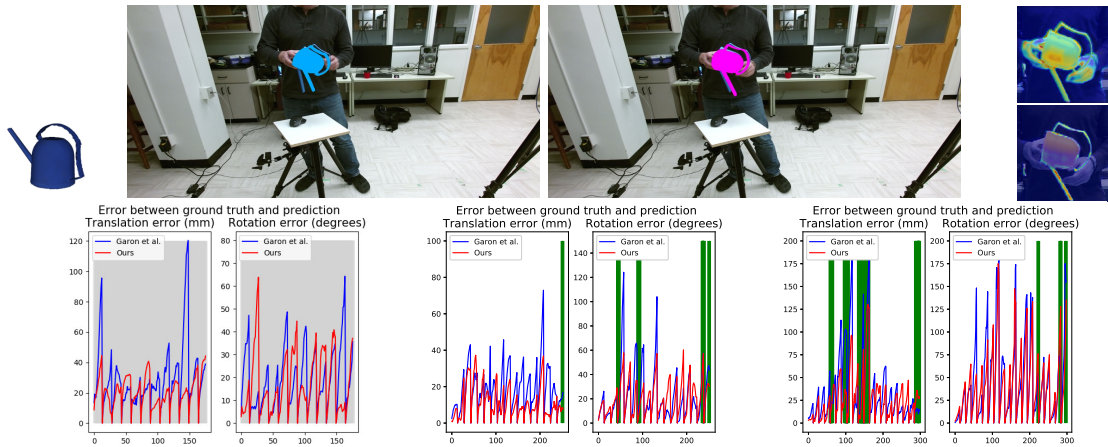


Figure 8.219: Comparison of the SoA [83] (*light blue*) and our (*pink*) approaches for the “Watering Can” in 3 scenarios: “75% Horizontal Occlusion”, “Rotation Only” and “Full Interaction”.

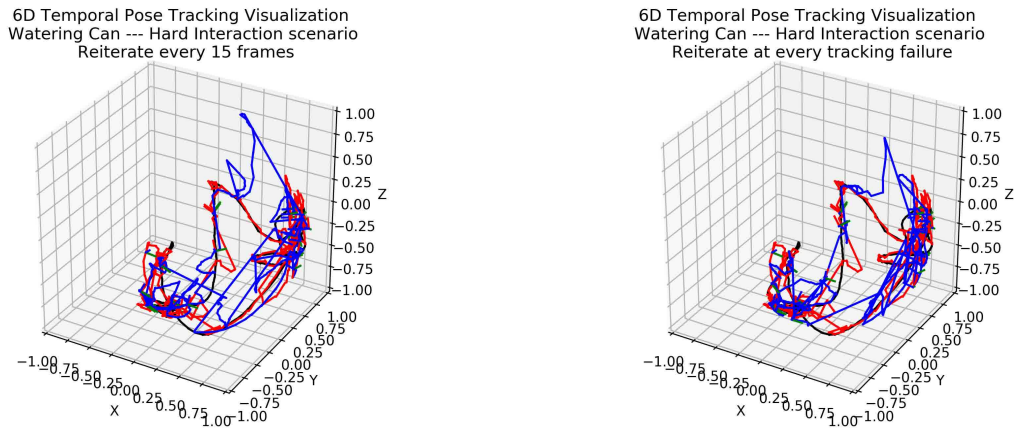


Figure 8.220: The 6D Pose Trajectories of the “Watering Can” object model, in time, for the “Hard Interaction” scenario. It is evident, in both cases, from this perspective as well, that our proposed approach outperforms the SoA of Garon et al.[83] by far, as it produces smaller errors and fewer failures. All length units are in mm and all rotations in degrees.

Quantitative results on the other 3 objects:

Approach	75% Horizontal Occlusion			75% Vertical Occlusion		
	Translational Error(<i>mm</i>)	Rotational Error(<i>degrees</i>)	Fails	Translational Error(<i>mm</i>)	Rotational(<i>degrees</i>)	Fails
Garon et al.[83] (“Dog”)	37.96 ± 23.39	47.94 ± 31.55	21	32.84 ± 34.07	22.44 ± 13.60	21
Ours (“Dog”)	24.43 ± 18.92	17.24 ± 12.41	25	36.53 ± 22.39	12.67 ± 7.95	20
Garon et al.[83] (“Lego”)	68.25 ± 46.97	40.04 ± 47.37	28	40.04 ± 47.37	35.30 ± 31.32	20
Ours (“Lego”)	72.04 ± 34.10	18.41 ± 13.84	28	12.92 ± 5.73	12.92 ± 9.02	20
Garon et al.[83] (“Watering Can”)	21.59 ± 11.32	23.99 ± 16.95	14	32.76 ± 24.12	26.74 ± 19.05	18
Ours (“Watering Can”)	20.71 ± 10.24	17.00 ± 18.99	13	17.66 ± 17.95	13.46 ± 10.43	12

Approach	Translation Interaction			Rotation Interaction		
	Translational Error(<i>mm</i>)	Rotational Error(<i>degrees</i>)	Fails	Translational Error(<i>mm</i>)	Rotational(<i>degrees</i>)	Fails
Garon et al.[83] (“Dog”)	58.87 ± 71.86	16.42 ± 13.51	20	11.16 ± 10.28	20.00 ± 21.31	17
Ours (“Dog”)	21.64 ± 22.78	9.27 ± 8.03	14	10.68 ± 7.53	20.07 ± 19.29	17
Garon et al.[83] (“Lego”)	27.90 ± 23.53	11.89 ± 18.50	29	16.42 ± 10.90	17.83 ± 15.90	32
Ours (“Lego”)	22.66 ± 24.58	9.08 ± 7.60	12	10.13 ± 6.79	7.22 ± 4.55	4
Garon et al.[83] (“Watering Can”)	24.95 ± 42.91	13.26 ± 11.34	16	13.14 ± 8.99	22.19 ± 25.93	15
Ours (“Watering Can”)	24.30 ± 21.51	8.79 ± 6.35	16	12.22 ± 9.46	18.66 ± 15.51	15

Approach	Full Interaction			Hard Interaction		
	Translational Error(<i>mm</i>)	Rotational Error(<i>degrees</i>)	Fails	Translational Error(<i>mm</i>)	Rotational(<i>degrees</i>)	Fails
Garon et al. [83](“Dog”)	37.73 ± 42.32	20.77 ± 19.66	23	23.95 ± 38.86	24.38 ± 26.39	20
Ours (“Dog”)	24.88 ± 35.85	28.52 ± 25.38	20	19.32 ± 15.97	19.72 ± 20.17	19
Garon et al. [83](“Lego”)	30.96 ± 31.44	22.10 ± 20.20	20	30.71 ± 42.62	36.38 ± 34.99	20
Ours (“Lego”)	23.58 ± 27.73	11.80 ± 12.28	13	16.47 ± 12.95	14.29 ± 11.68	11
Garon et al. [83](“Watering Can”)	33.76 ± 37.62	40.16 ± 35.90	26	28.31 ± 19.49	23.04 ± 24.27	28
Ours (“Watering Can”)	19.82 ± 19.98	28.76 ± 30.27	26	18.03 ± 14.99	19.57 ± 17.47	23

Table 8.25: 3D Translational and Rotational errors and overall tracking failures in six different scenarios for the final three employed objects.

As we can see, the accuracy of our approach exceeds that of the SoA [83], across all objects and in almost all scenarios, especially for the 3D rotation component (generally considered the more challenging one). According to Table 8.25, both our reported errors are generally lower (in terms of both mean and standard deviation) and our tracker fails equally or less often. Our approach presents aggravated errors in fast object motions in less occasions than [83] and handles both static and dynamic high-percentage occlusion patterns more successfully. The effectiveness of our method is verified by the fact that, not only it keeps track of the object’s 3D position under severe occlusions, but extends this property to 3D rotations as well.

Under a global view, evidently, the object most benefited by our methodological improvements is the “Dragon”. Since its geometry is the most complex, its texture is rich and it has several distinctive parts that stand out of the user’s grip, both our geometric modeling and the parallel attention modules find their best application in this case. When the user’s hand occludes parts of the “Dragon”, the attention shifts to its body parts of interest that stand out of the grip, like its neck, wings or tail. For the symmetric “Cookie Jar”, the differences between our method and the baseline are lower. The attentions’ effect is less prominent here since this model is of simpler, symmetric shape and poorer texture. This replaces the distinctive clues of the dragon case with ambiguities, denying the corresponding modules of the ability to easily identify the pose. Alongside the “Lego” model, they make the most out of the reflective symmetry handling algorithm as they avoid large abrupt errors that propagate to future frames. Although our CNN primarily focuses on the object’s shape, appearance seems to play a significant role in its predictions, as well, since the errors of the “Dog” and the “Watering Can” models, the less textured ones, decrease more mildly. The foreground attention map aids disentangling the “Dog” model from its background, a table of the same color, in the “75% Occlusion” scenarios. On the other hand, for the “Watering Can”, we observe that the most ambiguities are presented when its nozzle faces straight at the RGB-D sensor and when it is hidden due to its viewpoint configuration, as the effect of both attention modules and the geometric rotational modeling is diminished in this case.

Chapter 9

Ongoing Work

Now I know what a ghost is.
Unfinished business, that's what.

Salman Rushdie, "The Satanic Verses"

9.1 Unaddressed challenges

So far, our approach has dealt with a series of challenges that are innate in the Pose Tracking problem formulation, as enlisted in Section 2.9 i.e. domain representation mismatch, sensor noise modeling, clutter and occlusion handling, fast object movement, ambiguities caused by rotation representation and geometry-based uncertainties. However, there are many more of them left to consider in order to possibly extend our established framework.

1. modeling of Temporal Continuity:

First of all, object pose tracking is a temporal problem at its core and every attempt to address it via a learning-based approach needs to explicitly take the continuity of its 3D motion into account. This holistic concept may be able to enhance a Neural Network's ability to disentangle different poses from similar image imprints, as the whole 3D trajectory history will be taken into account, not only the exact previous pose, provided by the feedback stream.

2. Accumulation of small pose drift:

During inference, small tracking errors accumulate over time creating substantial differences between the estimated and the anticipated poses. To this end, implicitly reducing the temporal pose difference based on an RGB-D image pair may be proven to be not enough, and the need for an explicit difference representation at the image level may arise as crucial for the tracker's further improvement.

3. Depth information distillation:

As it has already been discussed, pose tracking algorithms have the potential to be proven very handy for mobile applications. However, our approach is severely based on having an available Kinect sensor that provides as clear depth maps as possible, a sensor that is not embedded in mobile devices. To this end, we plan to utilize depth information, available only during training, in order to distill it and enhance the appearance modality's pose distinguish ability.

4. Bridging 2D image source with the 3D task via Pose-Equivariant filtering:

Another important future task that we believe will boost the tracker's performance is process 3D features in order to produce pose results that live in the 3D world. Moreover, based on the observation that classic convolutional filters are only translationally equivariant, we make the extra

requirement of filtering the 3D features with SE(3)-equivariant convolutional filters, as we believe that this is a more intuitive way to succeed more refined estimations of the pose.

5. Extension to multiple objects without/with identical characteristics:

So far, our learning-based tracking approach is trained and tested on single known objects. So, one could train multiple identical trackers on multiple objects with different/the same model and then employ them in parallel without taking into account their interactions. However, previous literature has shown this to be suboptimal as it is frequent for trackers to mix objects or stick all to one of them (the one with the most extreme characteristics for example or one of the many with identical attributes). An intuitive possible approach for solving this is a reinforcement learning-based framework that will finetune multiple trackers (pretrained on single objects) with an additional loss term which will penalize collisions between objects. Of course, such an effort will require to further extend our data generation procedure to multiple relative 3D motions between the objects and a family of possible occluders.

6. Extension to unseen objects:

Following the work of Garon et al. [83], an interesting research question that arises is whether training our NN on multiple independent object models and then testing on unseen ones will make our proposed adjustments to generalize their performance and how this lack of prior model information effects the tracker’s performance.

In this chapter, we have tried to address the first two challenges and, appropriately, extend our Network architecture in order to achieve higher tracking performance. As far as it concerns the former, we present an alternative way to create the training dataset by planning continuous 3D object trajectories. As for the latter, we experiment with dense Optical Flows, produced based on the “Observed” and “Predicted” RGB pairs. Those Optical Flow image representations were processed by convolutional filters, in different configurations, just like their RGB-D counterparts, and then, their features were fused with the corresponding appearance-depth based ones. Finally, the motion information was distilled in order to keep the tracker’s time complexity within the real-time boundaries. In this chapter, we discuss our work, so far, and some observations of the effect these two extensions have on the tracker’s performance.

The rest of the challenges are discussed in Chapter 10, where a crude description is provided for each extension idea.

9.2 Continuous 3D Trajectories Learning

For fair comparison with the baseline work of Garon et al. [83] and our developed approach of Chapter 6, both in terms of performance and training speed, we wish to create a training dataset of 20,000 RGB-D image pairs of I_t, \hat{I}_t divided into 25-frame video clips of continuous object 3D trajectories. This arises from our need to model the motion continuity in our Deep architecture via a Recurrent unit. We, again, follow the golden spiral sampling approach. At first, we sample 20,000 poses. Then, we apply k-Means Clustering into 800 centroids (as $800 \times 25 = 20,000$ pair samples), for training, and 160 centroids ($160 \times 25 = 4000$ validation pair samples), for validation, that consist our initial poses.

We annotate these poses as $P_{Pred}^{(n)}(0)$, where n is the video subclip index, and we move from timestep to timestep to sample $P_{Pred}^{(n)}(t)$, for $t=0, \dots, 24$. In a nutshell, we carefully plan physically plausible pose transformations $T_{P_{Pred}(t)}^{P_{Pred}(t+1)}$, with a high variety that aims to fully explore the pose trajectory space. Then, for each timestep, we sample a random 6-D Rigid Transformation: $T_{P_{Pred}(t)}^{P_{Obs}(t)}$. In detail, we sample random translations $\delta t_{x,y,z} \sim \mathcal{N}(0, \Delta t)$, with, and rotations $\delta r_{a,b,c} \sim \mathcal{N}(0, \Delta r)$, (in Axis-Angle notation) to acquire $T_{P_{Pred}(t)}^{P_{Obs}(t)}$, which we apply on $P_{Pred}(t)$ to get $P_{Obs}(t)$. The Rigid Transformation $T_{P_{Pred}(t)}^{P_{Obs}(t)}$ (in the camera reference frame) serves as our ground truth learning target, the same as before.

The rendering pipeline remains the same.

What is left to explain, in detail, is the selection process of the $T_{P_{Pred}(t)}^{P_{Pred}(t+1)}$ transformation. Here, we need to state that for this particular trajectory planning process, we disentangle the translation and rotation components, and we follow a different strategy for each.

9.2.1 3D Translation continuous interpolation

We plan random 3D trajectories that start from initial 3D positions and result into equivalent final ones. Between them, we define a number of intermediate 3D positions that the “Cookie Jar” object must pass from. When all these random discrete positions are set, we perform cubic spline interpolation between them to acquire the continuous trajectories, divided into the discrete timesteps. Our intuition indicates the existence of the following tradeoff:

- On one hand, there is the requirement that position trajectories must diverge as much as possible from straight lines (i.e. the geodesic paths in the 3D Euclidean space) in order to achieve a big enough variance of the training instances. We expect that the tracker will be tested in scenarios that contain interchangeably fast and slow movements (like the “Free Hard Interaction” one), so we would like it to be trained into motion patterns that are complex enough. This need, has the natural consequence of randomly sampling as many intermediate 3D positions as possible.
- On the other hand, since the overall time interval is fixed, increasing the number of intermediate 3D positions will only increase the speed of the moving object, globally, causing a potential mismatch between the trajectories will train on and the, possibly, slower movements of the testing set.

We solve the tradeoff by establishing 2 intermediate positions at fixed timesteps: the first in the first and the second in the third quarter of the overall time interval. At those timesteps, random 3D positions will be sampled, similarly to the sampling procedure for $P_{Pred}^{(n)}(0)$. When the first intermediate position will be close to the initial one or the second intermediate positions will be close to the final one or the two intermediate positions will be close to each other, the object movement will be equivalently slower at this part of the trajectory, and vice versa.

Next, we define the process of Cubic Spline Interpolation that we use to create the trajectories of the 3D translations.

Cubic Spline Interpolation

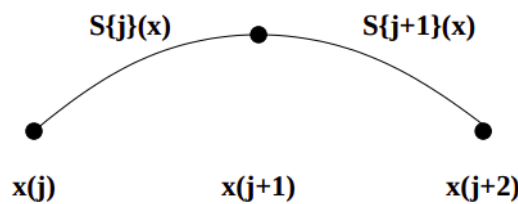


Figure 9.1: A graphical representation of two neighbouring segments of a cubic spline that interpolates a function $f(x)$. [93]

Given a function f defined on $[a, b]$ and a set of points $a = x_0 < x_1 < \dots < x_n = b$, a **cubic spline interpolant** \mathbf{S} for the function f , is a function that can be separated into cubic polynomials $S_j(x)$, (see fig.9.1), on the sub-intervals $[x_j, x_{j+1}]$, $\forall j = 0, 1, \dots, n - 1$. Those polynomials satisfy the following conditions:

- **"Left" interpolation:** $S_j(x_j) = f(x_j)$, $\forall j = 0, 1, \dots, (n - 1)$.
- **"Right" interpolation:** $S_j(x_{j+1}) = f(x_{j+1})$, $\forall j = 0, 1, \dots, (n - 1)$.
- **Slope match:** $S'_j(x_{j+1}) = S'_{j+1}(x_{j+1})$, $\forall j = 0, 1, \dots, (n - 2)$.
- **Curvature-match:** $S''_j(x_{j+1}) = S''_{j+1}(x_{j+1})$, $\forall j = 0, 1, \dots, (n - 2)$.

We separate the total $[a, b]$ interval into the sub-intervals $[x_j, x_{j+1}]$ on which the spline function is segmented into $S_j(x)$. Similarly, every spline segment $S_{j+1}(x)$ lives on the interval $[x_{j+1}, x_{j+2}]$.

In order to ensure the interpolant's smoothness, we require in every interior point x_{j+1} :

- the spline segments to be equal between them and equal to the discrete function values: $S_j(x_{j+1}) = S_{j+1}(x_{j+1}) = f(x_{j+1})$
- their derivatives to be equal: $S'_j(x_{j+1}) = S'_{j+1}(x_{j+1})$
- and their second derivatives to be equal: $S''_j(x_{j+1}) = S''_{j+1}(x_{j+1})$

Applying the conditions and solving the system:

For convenience we introduce the notation $h_j(x) = x - x_j$ at every point $x \in [a, b]$.

We start from the cubic polynomial

$$S_j(x) = a_j + b_j(x - x_j) + c_j(x - x_j)^2 + d_j(x - x_j)^3, \forall j = 0, \dots, n - 1 \quad (9.1)$$

and we apply the boundary conditions:

1. $S_j(x_j) = a_j = f(x_j)$
2. $S_{j+1}(x_{j+1}) = S_j(x_{j+1}) = a_{j+1} = a_j + b_j h_j + c_j h_j^2 + d_j h_j^3$
3. $S'_j(x_j) = b_j \iff b_{j+1} = b_j + 2c_j h_j + 3d_j h_j^2$
4. $S''_j(x_j) = 2c_j \iff c_{j+1} = c_j + 3d_j h_j$

We start from 4. and we solve for d_j :

$$d_j = \frac{c_{j+1} - c_j}{3h_j} \quad (9.2)$$

and we plug it into 2., 3. to get:

1)

$$a_{j+1} = a_j + b_j h_j + \frac{h_j^2}{3}(2c_j + c_{j+1}) \iff b_j = \frac{1}{h_j}(a_{j+1} - a_j) - \frac{h_j}{3}(2c_{j-1} + c_j) \quad (9.3)$$

2)

By reducing the index j by 1: $b_{j-1} = \frac{1}{h_{j-1}}(a_j - a_{j-1}) - \frac{h_{j-1}}{3}(2c_{j-2} + c_{j-1}) \quad (9.4)$

3)

$$b_{j+1} = b_j + h_j(c_j + c_{j+1}) \iff b_j = b_{j-1} + h_{j-1}(c_{j-1} + c_j) \quad (9.5)$$

and by combining 2),3), we end up with a linear system of equations:

$$h_{j-1}c_{j-1} + 2(h_{j-1} + h_j)c_j + h_j c_{j+1} = \frac{3}{h_j}(a_{j+1} - a_j) - \frac{3}{h_{j-1}}(a_j - a_{j-1}), \forall j = 1, 2, \dots, n - 1. \quad (9.6)$$

So, we wish to find the coefficients $\{c_j\}_{j=0}^n$. When we succeed in doing so, we will be able to compute b_j and d_j (a_j are known and equal to $f(x_j)$):

- $$b_j = \frac{a_{j+1} - a_j}{h_j} - \frac{h_j(2c_j + c_{j+1})}{3} \quad (9.7)$$

- $$d_j = \frac{c_{j+1} + c_j}{3h_j} \quad (9.8)$$

However, in order to solve for $\{c_j\}_{j=0}^n, \forall j = 1, \dots, n - 1$, we only have $(n - 1)$ equations for $(n + 1)$ unknowns. So, we need to add two more boundary conditions:

- **Natural boundary conditions:**

- $S_0''(x_0) = 2c_0 = 0 \iff c_0 = 0$
- $S_n''(x_n) = 2c_n = 0 \iff c_n = 0$

- The derivative of the function must be known at the endpoints in order to get the **Clamped boundary conditions**

- $S_0'(x_0) = b_0 = f'(x_0) \iff 2h_0c_0 + h_0c_1 = \frac{3}{h_0}(a_1 - a_0) - 3f'(x_0)$
- $S_{n-1}'(x_n) = b_n = b_{n-1} + h_{n-1}(c_{n-1} + c_n) = f'(x_n) \iff h_{n-1}c_{n-1} + 2h_{n-1}c_n = 3f'(x_n) - \frac{3}{h_{n-1}}(a_n - a_{n-1})$

Combining all the boundary conditions, we end up to the following linear systems of the $A\tilde{\mathbf{x}} = \mathbf{y}$, where

- $$A = \begin{bmatrix} 2h_0 & h_0 & 0 & \dots & \dots & 0 \\ h_0 & 2(h_0 + h_1) & h_1 & \ddots & \ddots & \vdots \\ 0 & h_1 & 2(h_1 + h_2) & h_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & h_{n-2} & 2(h_{n-2} + h_{n-1}) & h_{n-1} \\ 0 & \dots & \dots & 0 & h_{n-1} & 2h_{n-1} \end{bmatrix} \quad (9.9)$$

- $$\tilde{\mathbf{x}} = \begin{bmatrix} c_0 \\ c_1 \\ \dots \\ c_{n-1} \\ c_n \end{bmatrix} \quad (9.10)$$

- $$\mathbf{y} = \begin{bmatrix} \frac{3(a_1 - a_0)}{h_0} - 3f'(x_0) \\ \frac{3(a_2 - a_1)}{h_1} - \frac{3(a_1 - a_0)}{h_0} \\ \dots \\ \frac{3(a_n - a_{n-1})}{h_{n-1}} - \frac{3(a_{n-1} - a_{n-2})}{h_{n-2}} \\ 3f'(x_n) - \frac{3(a_n - a_{n-1})}{h_{n-1}} \end{bmatrix} \quad (9.11)$$

This linear system is solved (in $O(n)$ time complexity (where n is the matrix A 's size: $A \in \mathbb{R}^{n \times n}$)) numerically using the LU factorization method [122] of A . Following the algorithmic steps below, we calculate:

$$\begin{aligned} \tilde{\mathbf{z}} &= L^{-1}\mathbf{y} \\ \tilde{\mathbf{x}} &= U^{-1}\tilde{\mathbf{z}} \end{aligned} \quad (9.12)$$

Algorithm 3: The solution of the $A \tilde{\mathbf{x}} = \mathbf{y}$ linear system, with the use of LU factorization of matrix A .

```

 $L[1, 1] = A[1, 1];$ 
 $U[1, 2] = \frac{A[1, 2]}{L[1, 1]};$ 
 $z[1] = \frac{y[1]}{L[1, 1]};$ 
 $i = 2;$ 
while  $i \leq n - 1$  do
   $L[i, i - 1] = A[i, i - 1];$ 
   $L[i, i] = A[i, i] - L[i, i - 1]U[i - 1, i];$ 
   $U[i, i + 1] = \frac{A[i, i + 1]}{L[i, i]};$ 
   $z[i] = \frac{y[i] - L[i, i - 1]z[i - 1]}{L[i, i]};$ 
   $i = i + 1;$ 
end
 $L[n, n - 1] = A[n, n - 1];$ 
 $L[n, n] = A[n, n] - L[n, n - 1]U[n - 1, n];$ 
 $z[n] = \frac{y[n] - L[n, n - 1]z[n - 1]}{L[n, n]};$ 
 $\tilde{\mathbf{x}}[n] = \tilde{\mathbf{z}}[n];$ 
 $i = n - 1;$ 
while  $i > 1$  do
   $\tilde{\mathbf{x}}[i] = \tilde{\mathbf{z}}[i] - U[i, i + 1]\tilde{\mathbf{x}}[i + 1];$ 
   $i = i - 1;$ 
end

```

We plan the trajectories of each translation component independently by executing 3 parallel spline interpolations between each intermediate sampled point, provided by the previous procedure.

In the following figure, we cite some trajectory examples for the translation component. With the blue line, we annotate straight lines that connect between the 4 (one initial, one final and 2 intermediate) 3D points and with the red one the smooth interpolated trajectories that pass from all of them. We can clearly see that our training dataset contains a wide variety of different translational motion patterns and speeds.

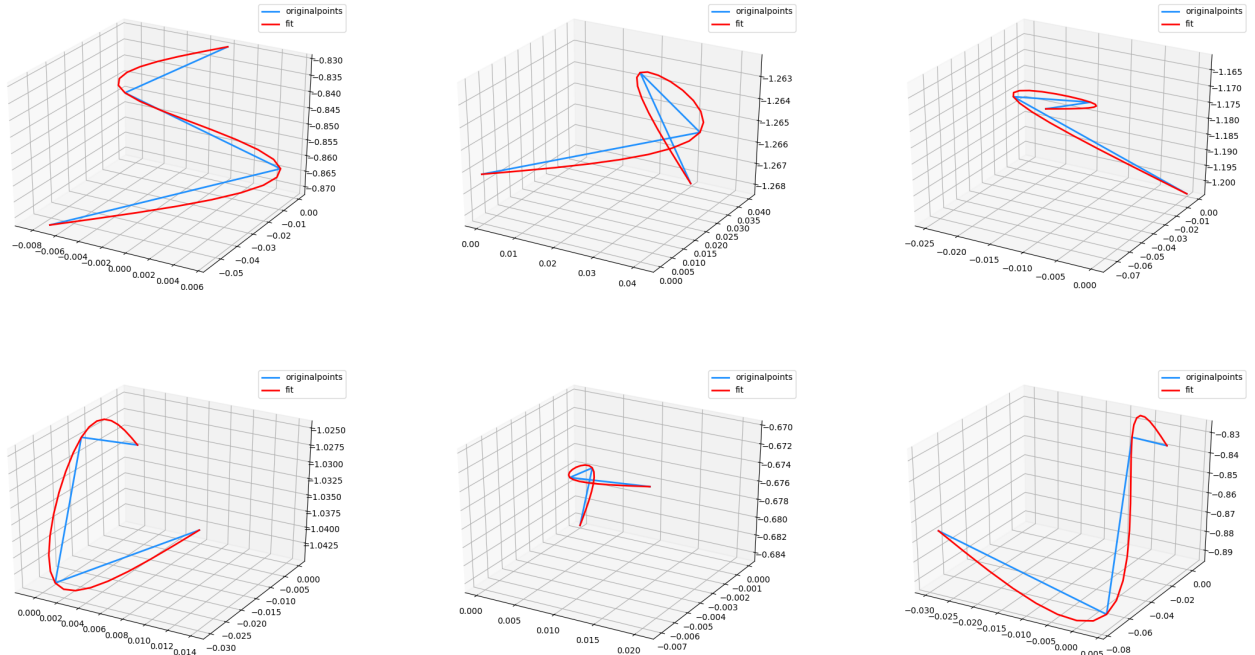


Figure 9.2: Examples of 3D continuous positional trajectories. The blue line linearly interpolates between the 4 discrete points (in meters) and the red one represents the smooth trajectories that pass all 4 of them, interpolated using the cubic spline method.

9.2.2 3D Rotation continuous interpolation

As for the rotation component, things are rather different. Since object rotations result into more severe appearance alterations than translations, there is a need for simpler rotational motion patterns in every video subclip of 25 frames. Since recurrent modules are known to function using a learnable memory unit, an, otherwise, convoluted rotational movement will more likely confuse the module's understanding of the object's dynamics. That happens because the rotational pose component lies on the $SO(3)$ Lie manifold and, thus, requiring the presence of a many different intermediate rotational checkpoints will result in the frequent resurgence of the same rotation configurations in a single clip. To this end, we again, sample one random initial and one final pose orientation, as in Chapter 6 and the intermediate orientations are interpolated along the Geodesic rotational path in $SO(3)$.

Geodesic Rotational Interpolation

If we think about the Lie Group $SE(3)$ and we are given 2 poses P_1, P_2 that live in it: $P_1 = \begin{bmatrix} R_1 & \mathbf{t}_1 \\ \mathbf{0}^T & 1 \end{bmatrix}$ and $P_2 = \begin{bmatrix} R_2 & \mathbf{t}_2 \\ \mathbf{0}^T & 1 \end{bmatrix}$, the object motion between them, $P(t) = \begin{bmatrix} R(t) & \mathbf{t}(t) \\ \mathbf{0}^T & 1 \end{bmatrix}$, lies on that manifold as well.

We may annotate a time-dependent velocity vector $\mathbf{V}(t) = \begin{bmatrix} \omega \\ v \end{bmatrix}$ whose norm is calculated:

$$V(t) = \frac{dP(t)}{dt}. \quad (9.13)$$

Then, according to the work of Zefran et al. [168], the trajectory of the shortest path is the one that minimizes the left invariant Riemannian metric:

$$J = \int_{t_0}^T \langle \mathbf{V}(t), \mathbf{V}(t) \rangle dt = \int_{t_0}^T \mathbf{V}^T(t) \mathbb{W} \mathbf{V}(t) dt, \quad (9.14)$$

proposed by Park et al. [96], where, in our case $t_0 = 0, T=24$ and the matrix \mathbb{W} is a diagonal matrix of the form:

$$\mathbb{W} = \begin{bmatrix} \alpha \mathbb{I} & \mathbf{0} \\ \mathbf{0} & \beta \mathbb{I} \end{bmatrix}. \quad (9.15)$$

α and β are positive scalars which act like scaling factors for angular velocities and linear velocities. In kinematic analysis there is no a priori justification for choosing them, so an intuitive choice we make is to assign $\beta = 1$ and $\alpha = \max_{i=1}^3 \{\lambda_{\Lambda}^{(i)}\}$, where Λ is the object Inertia Tensor estimation. According to Zefran et al.[168], this metric is minimized by the solution of the next differential system:

$$\begin{aligned} \frac{d\omega}{dt} &= 0 \\ \frac{dv}{dt} &= -\omega \times v, \end{aligned} \quad (9.16)$$

Keeping only the rotational component, the shortest rotational distance path (geodesic) (i.e. the one with the minimum length in $SE(3)$) that interpolates R_1, R_2 is given by:

$$R(t) = R_1 e^{\Omega_0 t}, \quad (9.17)$$

where $\Omega_0 = \log(R_1^T R_2)$ ($\log(\cdot)$ refers to the logarithmic matrix of Section 2.11).

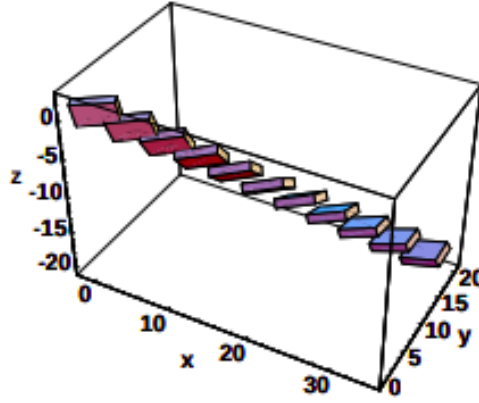


Figure 9.3: A graphical representation of the rotational geodesic 3D trajectory that an object follows to transform from one initial configuration (upper left corner) to a corresponding final one (lower right corner)[168]

We make the observation that, for rotation, we do not sample each component individually, following a SLERP-like [20] algorithm, that is common in Computer Graphics. That happens to match our intuition that the geodesic path is the optimal choice for such a small training time interval. The interval's choice was made in order to balance out a tradeoff that wants, maximum memory of the previous poses of the trajectory, on one hand, and the problems that Recurrent Neural modules face in learning such long trajectories, on the other. Next, we explain the functionality of those recurrent modules.

9.2.3 Recurrent Units

Recurrent Neural Networks (RNNs)

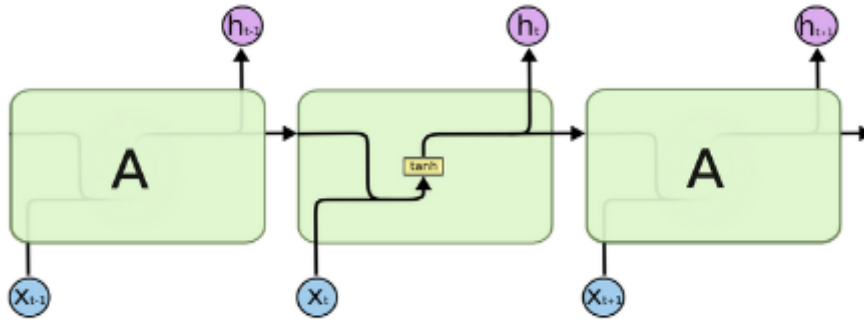


Figure 9.4: The looped module in a standard RNN [44] contains a single Fully Connected layer[33]

Recurrent neural networks or **RNNs** [44] are a family of neural networks for processing sequential data equipped with an internal state for each unit. At each time-step, the output of a layer depends on the current input and the previous state. Thus, for a sequence of T elements of vectors $\mathbf{x}_1, \dots, \mathbf{x}_T \in \mathbb{R}^d$, the output of a simple RNN, parameterized by the matrices $W \in \mathbb{R}^{m \times n}$ and $R \in \mathbb{R}^{n \times n}$ is a sequence of n -dimensional vectors $\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n$, given by the equation :

$$\mathbf{y}_t = \sigma(\mathbf{x}_t W + \mathbf{y}_{t-1} R) \quad (9.18)$$

A Recurrent Neural Network can be thought of as multiple copies of the same network, each passing a message to a successor. Consider what happens if we unroll the loop: this chain-like nature (see fig.9.5) reveals that RNNs are the natural NN architecture to select for processing to sequences.

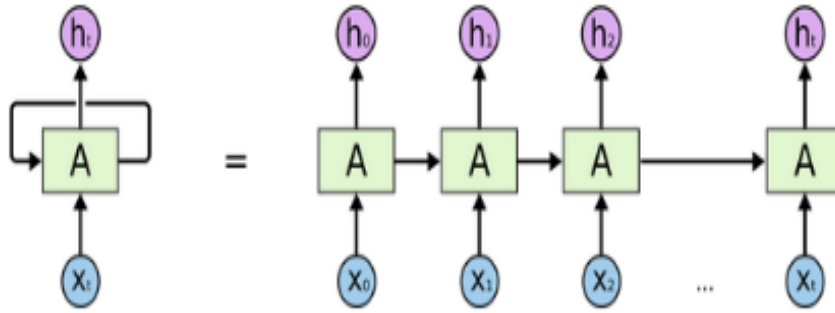


Figure 9.5: Unrolling the Recurrent Neural Networks [33]

They are trained using an expansion of the classic BackPropagation algorithm: **Back Propagation Through Time (BPTT)**. It begins by unfolding the RNN in time. The unfolded network contains T inputs and outputs, **but every copy of the network shares the same parameters**. Then, the classical BackPropagation algorithm is used to find the gradient of the cost function with respect to all the shared network parameters, considering, in practice, every timestep as an extra NN layer.

The Problem of Long-Term Dependencies:

In cases where the gap between the relevant information appearance and the place that it's needed is small, RNNs can learn to use that past information. But there are also cases where we need more long-term context. In such a case, there is the possibility for the gap between the relevant information and the point, where it is needed, to become very large. In theory, RNNs are absolutely capable of handling dependencies of arbitrary length. A human could carefully pick parameters for them to solve toy problems of this form. Sadly, in practice, RNNs don't seem to be able to learn them. Observing it from the backwards perspective, it consists of nothing more than a Vanishing Gradient problem. Since the BPTT algorithm tunes weights at each layer via the Chain Rule, calculated in time, their gradient values will exponentially shrink as the algorithm propagates through each time step, as the same happens to its classical BP version.

Long-Short Term Memory (LSTM)

Long Short Term Memory Networks [44], or LSTMs, are a special kind of RNN, capable of learning long-term dependencies. To succeed so, they are equipped with specific gating mechanisms that define the information percentage that should be kept or erased from the memory at each step, as shown mathematically in the following equation set.

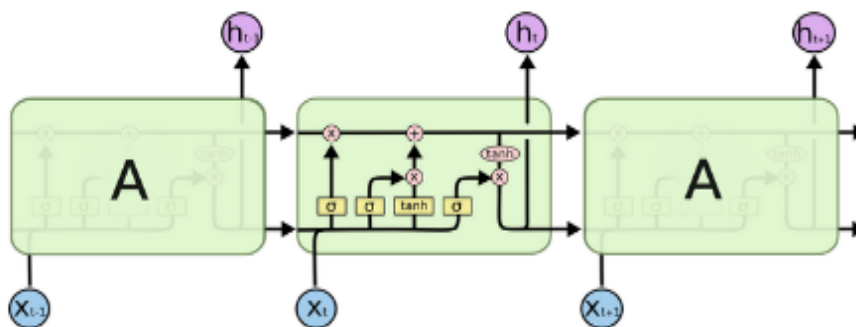


Figure 9.6: Internal Structure of LSTM Layer[44]

$$\begin{aligned}
f_t &= \sigma(W_f[h_{t-1}, x_t] + b_f) \\
i_t &= \sigma(W_i[h_{t-1}, x_t] + b_i) \\
\hat{C}_t &= \tanh(W_C[h_{t-1}, x_t] + b_C) \\
C_t &= f_t \cdot C_{t-1} + i_t \cdot \hat{C}_t \\
o_t &= \sigma(W_o[h_{t-1}, x_t] + b_o) \\
h_t &= o_t \cdot \tanh(C_t)
\end{aligned}
\tag{9.19}$$

The key to LSTMs is the **cell state**, the horizontal line running through the top of the diagram. The cell state is kind of like a conveyor belt. It runs straight down the entire chain, with only some minor linear interactions. It's very easy for information to just flow along it unchanged.

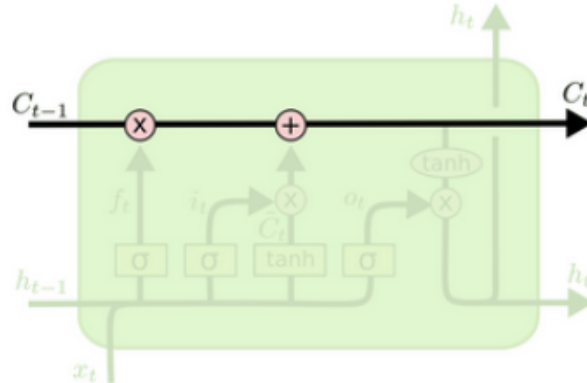


Figure 9.7: The LSTM's cell state highway.[33]

As mentioned above, the LSTM architecture has the ability to remove or add information to the cell state, carefully regulated by structures called **gates**. The information is controlled through three gates:

- the **Input Gate** i
- the **Forget Gate** f
- the **Output Gate** o

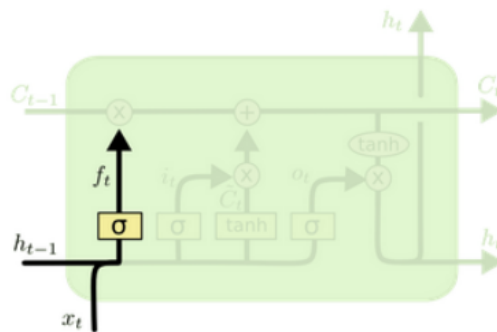


Figure 9.8: The **Forget-Gate** f of LSTM. In this step, the LSTM module decides what information will be thrown away from the cell state. This is implemented by a sigmoid layer which looks at the h_{t-1} and the x_t and assigns a number of 0 or 1 on each of the units of C_{t-1} . [94]

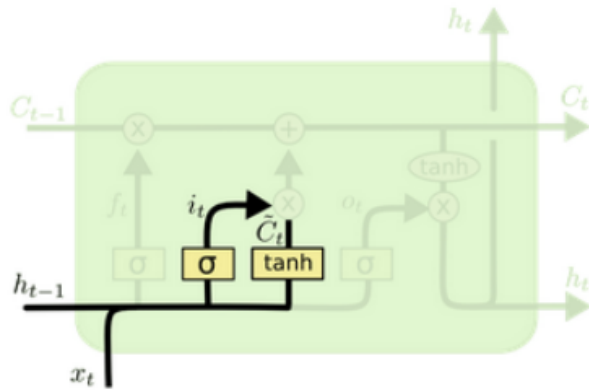


Figure 9.9: The **Input-Gate** i of LSTM. In this step, the LSTM decides what new information will be stored. This has two parts. First, a sigmoid layer called the “input gate layer” decides which values we’ll update. Next, a tanh layer creates a vector of new candidate values, \hat{C}_t , that could be added to the state. In the next step, we’ll combine these two to create an update to the state.[94]

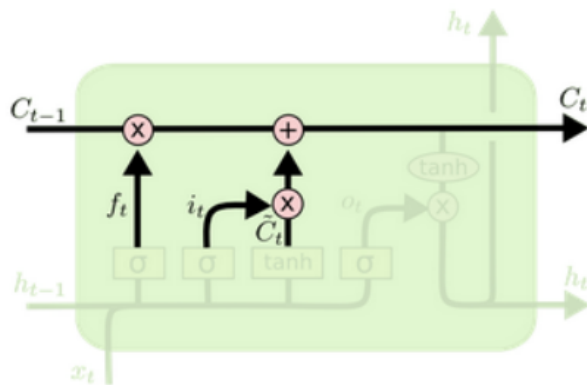


Figure 9.10: In order to update the old cell state, c_{t-1} , into the new cell state c_t , we multiply the old state by f_t , forgetting the things we decided to forget earlier. Then we add $i_t \odot c_t$. [94]

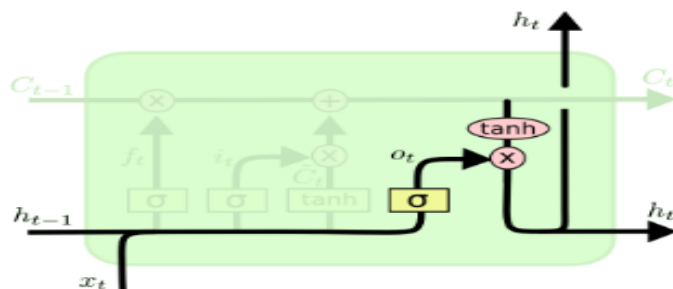


Figure 9.11: The **Output Gate** o of LSTM. In this step, the LSTM updates the C_{t-1} value with the new one C_t and its synthesized by how much information will be abandoned and what new information will be stored in the cell. [94]

Gated Recurrent Unit(GRU)

Gated Recurrent Units (GRUs) are an idea similar to LSTM. Their difference is that they get rid of the cell state and use the hidden state h_t to transfer information, instead. They, also, replace the 3 gating mechanisms of LSTM with only two:

- a **Reset Gate** $U^{(r)}$:

$$r_t = \sigma(W^{(r)}x_t + U^{(r)}h_{t-1}) \quad (9.20)$$

which is used from the model to decide how much of the past information to forget and

- an **Update Gate** $U^{(z)}$:

$$z_t = \sigma(W^{(z)}x_t + U^{(z)}h_t) \quad (9.21)$$

that helps the model to determine how much of the past information (from previous time steps) needs to be passed along to the future. That is really powerful because the model can decide to copy all the information from the past and eliminate the risk of vanishing gradient problem.

Combining them, we will update the current working memory content using the reset gate to store the relevant information from the past:

$$h'_t = \tanh(Wx_t + r_t \cdot Uh_{t-1}) \quad (9.22)$$

and then, as the last step, the network needs to calculate h_t — vector which holds information for the current unit and passes it down to the next layers. In order to do that the update gate z_t is needed:

$$h_t = z_t \cdot h_{t-1} + (1 - z_t)h'_t \quad (9.23)$$

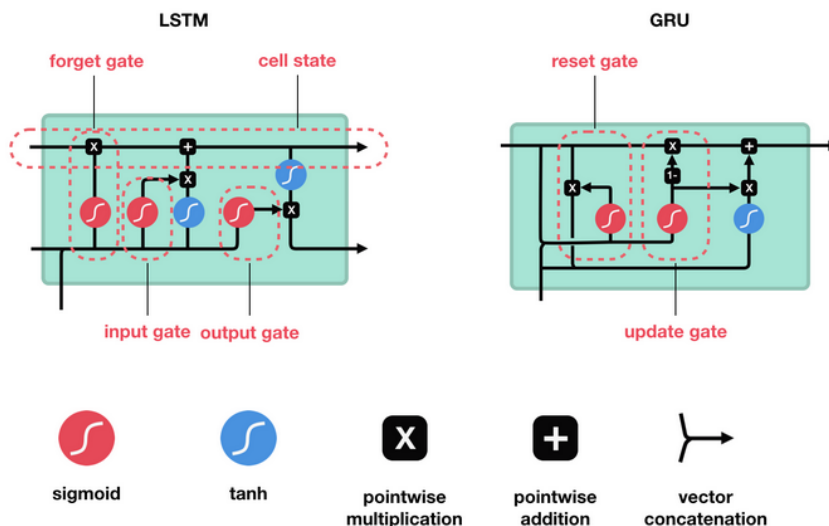


Figure 9.12: A comparison between a LSTM module (on the left) and a Gated Recurrent Unit (GRU) (on the right) functionalities.[92]

Compared to LSTM, the GRU has fewer tensor operations; therefore, they are a little speedier to train than the LSTM's. There isn't a clear winner which one is better. Researchers and engineers usually try both to determine which one works better for their use case. A rule of thumb to begin with would be prioritizing LSTM for longer, more complex sequences, as it employs more parameters, therefore more representation power without the danger of overfitting them.

Convolutional Long-Short Term Memory(Conv-LSTM)

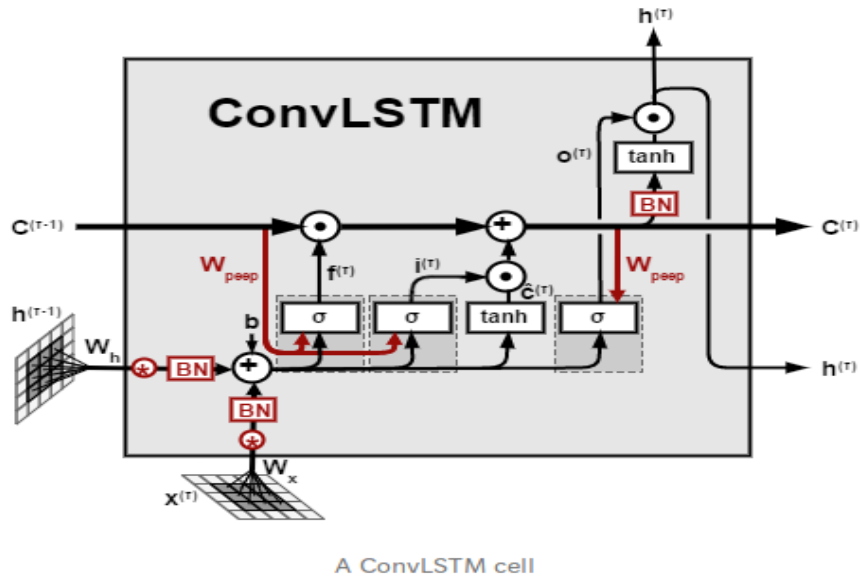


Figure 9.13: A ConvLSTM cell.[164]

As an insightful reader would understand, LSTMs (and GRUs, extensively) are comprised of Fully Connected layers. That alone means, that they are far from ideal for processing image streams that are temporally correlated, i.e. videos, as they do not take into consideration image spatial properties.

So, in [164], a variation of the classical LSTM architecture was introduced, where all FC modules were replaced with Convolutional ones. We call this modified architecture a **Convolutional LSTM**.

Mathematically, it is characterized by the corresponding equation set which is similar to the Linear LSTM one, but with Convolutional operations in place of the Matrix Multiplication ones:

$$\begin{aligned}
 f_t &= \sigma(W_f * [h_{t-1}, x_t] + b_f) \\
 i_t &= \sigma(W_i * [h_{t-1}, x_t] + b_i) \\
 \hat{C}_t &= \tanh(W_C * [h_{t-1}, x_t] + b_C) \\
 C_t &= f_t \cdot C_{t-1} + i_t \cdot \hat{C}_t \\
 o_t &= \sigma(W_o * [h_{t-1}, x_t] + b_o) \\
 h_t &= o_t \cdot \tanh(C_t)
 \end{aligned} \tag{9.24}$$

9.2.4 Temporal Convolutional Networks

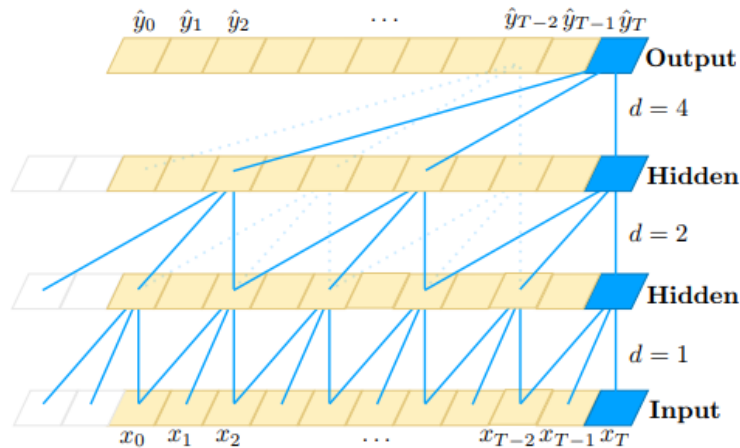


Figure 9.14: A dilated causal convolution with dilation factors $d = 1, 2, 4$ and filter size $k = 3$. [143]

Temporal Convolutional Networks (TCNs) are a variation of classical Convolutional Neural Network architectures for sequence modeling tasks. To this end, they employ 1D convolutional operations only on the elements from current timestamp or earlier in the previous layer (**casual convolutions**). The main idea is to use these TCNs instead of RNN variations, when processing sequential data, by forming a temporal hierarchical structure, where each component has a limited field of temporal 1D view (see fig.9.14).

The distinguishing characteristics of TCNs are:

- The architecture can take a sequence of any length and map it to an output sequence of the same length, just as RNNs do.
- The convolutions in the architecture are causal, meaning that there is no information “leakage” from future to past. To accomplish the first point, the TCN uses a 1D fully-convolutional network (FCN) architecture, where each hidden layer is the same length as the input layer, and zero padding of length (kernel size - 1) is added to keep subsequent layers the same length as previous ones. To achieve the second point, the TCN uses causal convolutions, i.e. convolutions where an output at time t is convolved only with elements from time t and earlier in the previous layer.

The above description insinuates that how much this kind of modules can look back at history depends, firstly, on the filter size and, following, on the depth of each network’s layer. Hence, capturing long term dependencies becomes challenging. One simple solution to aid our module in this direction is to use dilated convolutions.

Dilated Convolutions:

Similarly to the 2D case (Section 3.3.5), for a 1-D sequence input $\mathbf{x} \in \mathbb{R}^n$ and a filter $f: \{0, \dots, k-1\} \rightarrow \mathbb{R}$, the dilated convolution operation F on element s of the sequence is defined as:

$$F(s) = (\mathbf{x}_d^* f)(s) = \sum_{i=0}^{k-1} f(i) \mathbf{x}_{s-d \cdot i} \quad (9.25)$$

In above formula, the causal convolution criteria is also taken care of by using only non-negative values of i . Also, a dilation of size d (frequently proportional to the network’s depth) is applied to the filter’s kernel in order to increase the (temporal) receptive field of each layer.

Advantages of using TCNs for sequence modeling:

1. **Parallelism:** Unlike in RNNs, where the predictions for later time-steps must wait for their predecessors to complete, convolutions can be done in parallel since **the same filter is used in**

each layer. Therefore, in both training and evaluation, a long input sequence can be processed as a whole in TCN, instead of sequentially as in RNN.

2. **Flexible receptive field size:** A TCN can change its receptive field size in multiple ways. For instance, stacking more dilated (causal) convolutional layers, using larger dilation factors, or increasing the filter size are all viable options (with possibly different interpretations). TCNs thus afford better control of the model’s memory size, and are easy to adapt to different domains.
3. **Low memory requirement for training:** Especially in the case of a long input sequence, LSTMs and GRUs can easily use up a lot of memory to store the partial results for their multiple cell gates. However, in a TCN the filters are shared across a layer, with the back-propagation path depending only on network depth. Therefore in practice, gated RNNs are likely to use up to a multiplicative factor more memory than TCNs.
4. **Capturing local information:** Using convolution operation helps in capturing local information along with temporal information. All local information, provided by each layer, are combined into more global context, as we move forward the hierarchical structure of the TCN.

Data Sampling and Augmentations Continuous alterations

At the same time of creating a continuous object trajectory training dataset, we follow the same method to create a same occluded dataset, similar to the one used previously.

During the training process, we sample and load each subvideo pair continuously as well, allowing only **video-clip-wise shuffling**. This strategy, although it rises as a necessity from the presence of a Recurrent unit in our architecture, it reduces the appearance variation in a training minibatch by the factor of the sequence length, thus biasing the gradient step resulting to be harmful, by itself, to the learning process (see following experiments). However, we hope that the Recurrent unit will exploit the temporal coherence of our dataset trajectories and improve the overall tracking performance, showcasing that indeed, using this continuous generation and sampling method the algorithm learns to generalize not just to uncorrelated pose transformations, but to complete trajectories instead. In order to succeed in shuffling the background dataset in a continuous way, as well, we modify the SSIM 3.2.2 based pruning procedure described above. This time, we quantize each background RGB-D videoclip of the SUN3D database [163] into subclips of 25 frames and we greedily calculate the SSIM between the corresponding frames of each subclip. Finally, we take the average of those 25 SSIMs and we apply the same thresholding operation.

Finally, for the data augmentation part, besides sampling occluders and backgrounds of the same processed video subclip, we change the full occlusion scenario: at every frame of the subclip, we calculate the outcome of a Bernoulli random variable and if its 1, then the object gets fully occluded by the hand for the rest of the subclip sequence. Furthermore, we set the same HSV-based 3.2.1 discolorification for the same object sequence, instead of a random one per frame. The rest augmentation strategies are free to differ per frame as they are mostly noise models that come from external sources (i.e. sensors like Kinect) that have nothing to do with whether we will model the continuity of the object’s 3D trajectories or not.

9.2.5 Recurrent variations

In this section, we examine the possibility of modeling the continuity of the object’s 3D trajectory by employing recurrent modules on top of our previous architecture. So, we examine the following configurations that consist alterations of the improved CNN architecture, that was proposed in Chapter 6:

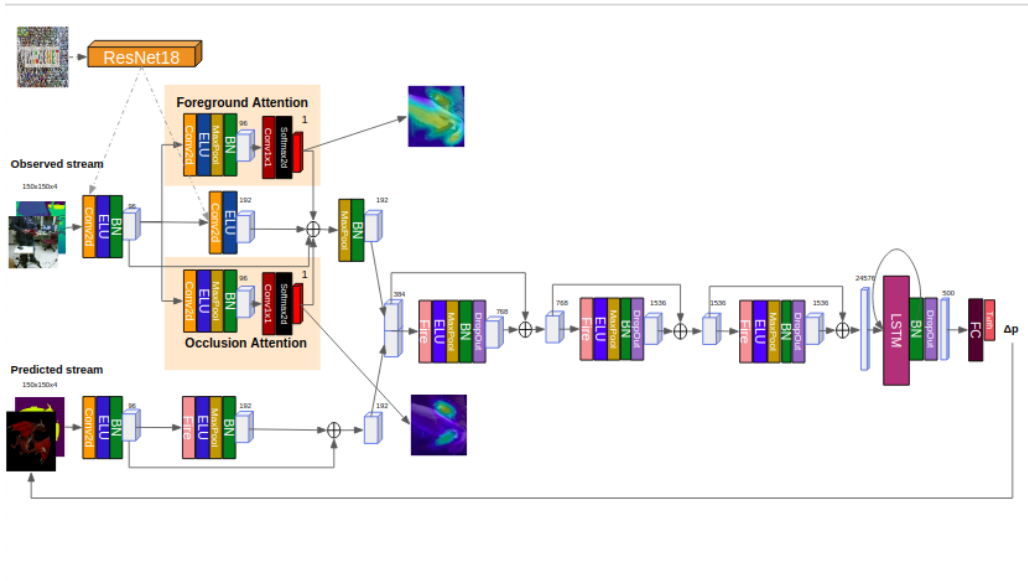


Figure 9.15: Overview of an alteration of our CNN architecture where the first Linear layer is replaced by a recurrent type (RNN/LSTM/GRU) one with the same number of neurons.

- the architecture of Chapter 6 with the first Linear layer replaced by an RNN module
- the architecture of Chapter 6 with the first Linear layer replaced by an LSTM module
- the architecture of Chapter 6 with the first Linear layer replaced by an GRU module
- the architecture of Chapter 6 with the last Fire layer replaced by an FireLSTM (a ConvLSTM (Section 9.2.3) layer where classic convolutional filters are replaced by Fire modules (Section 3.3.6)) module
- the architecture of Chapter 6 with the first Linear layer replaced by a sequence of 3 1d (Temporal) Convolutions (Section 9.2.4)

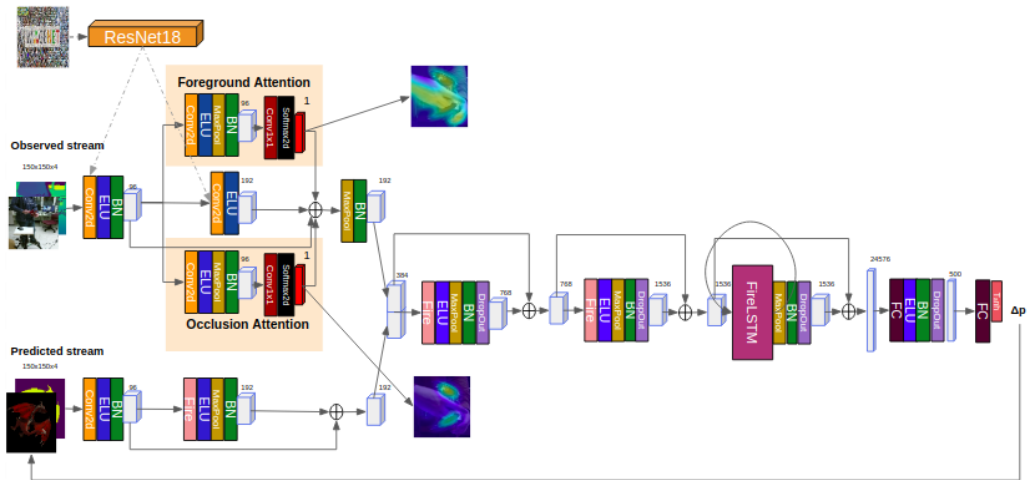


Figure 9.16: Overview of an alteration of our CNN architecture where the last Fire layer is replaced by a FireLSTM one with the same number of filters.

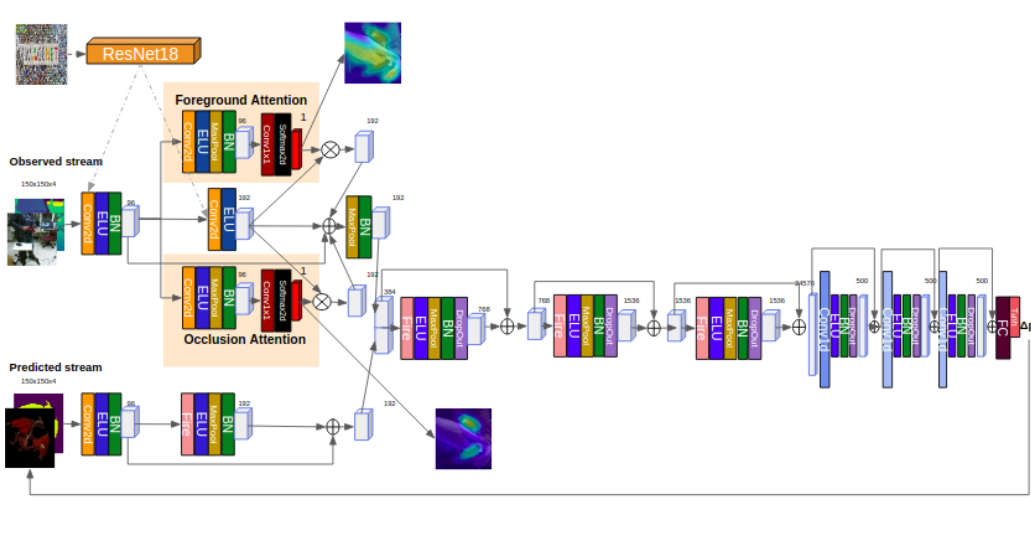


Figure 9.17: Overview of an alteration of our CNN architecture where the first Linear layer is replaced by a sequence of 3 1d Convolutional layers.

Discrete vs Continuous Sampler

But first, we need to assess the impact that changing the sampling strategy from a discrete (per-sample) to a continuous one (per video subclip) has on the tracker’s performance. For simplicity, for the baseline method of Garon et al. [83], we use both the continuous dataset that we created with the pose interpolation in the previous Section and its discrete counterpart of Chapter 6. We evaluate the tracker’s behaviour when the training pairs are sampled after Discrete/Continuous shuffling of the training set and/or Discrete/Continuous shuffling of the validation set. Finally, we, also, experiment with blending/binding the moving occluder with the object of interest during the generation process instead of the augmentation one, thus de facto reducing one of the most severe augmentation degrees of freedom of our training pipeline. We comment on the impact of this action and we discuss its value in validating/cancelling future expansion thoughts of our method. A numerical experimental comparison is provided in the Tables below:

Architecture	Translational Error (mm)	Rotational Error (o)
[83] (20k training pairs) (<i>Discrete Poses Training Dataset - Discrete Pose Train Sampler - Discrete Pose Validation Sampler</i>)	48.58 ± 38.23	35.43 ± 34.74
[83] (20k training pairs) (<i>Continuous Poses Training Dataset - Discrete Pose Train Sampler - Discrete Pose Validation Sampler</i>)	28.14 ± 18.49	58.70 ± 42.85
[83] (20k training pairs) (<i>Continuous Poses Training Dataset - Continuous Pose Train Sampler - Discrete Pose Validation Sampler</i>)	193.31 ± 103.78	46.78 ± 38.25
[83] (20k training pairs) (<i>Continuous Poses Training Dataset - Continuous Pose Train Sampler - Continuous Pose Validation Sampler</i>)	43.78 ± 12.29	31.03 ± 33.00
[83] (20k training pairs) (<i>Continuous Poses Training Dataset - Discrete Pose Train Sampler with Blended Hand - Discrete Pose Validation Sampler</i>)	39.13 ± 32.33	51.30 ± 47.8
[83] (20k training pairs) (<i>Continuous Poses Training Dataset - Continuous Pose Train Sampler with Blended Hand - Continuous Pose Validation Sampler</i>)	149.85 ± 85.48	63.04 ± 46.95

Table 9.1: The 3D Translational and Rotational error metrics comparison for the architecture of [83], when this CNN design is trained on the Discretely/Continuously sampled training dataset, shuffled in a Discrete/Continuous manner and without/with a-priori blending the same occluder 3D continuous trajectory for every batch iteration.

Architecture	Normalised Pose Error (mean \pm std)
[83] (20k training pairs) (Discrete Poses Training Dataset - Discrete Pose Train Sampler - Discrete Pose Validation Sampler)	3.45 \pm 3.11
[83] (20k training pairs)(Continuous Poses Training Dataset - Discrete Pose Train Sampler - Discrete Pose Validation Sampler)	4.20 \pm 3.01
[83] (20k training pairs) (Continuous Poses Training Dataset - Continuous Pose Train Sampler - Discrete Pose Validation Sampler)	8.28 \pm 5.20
[83] (20k training pairs)(Continuous Poses Training Dataset - Continuous Pose Train Sampler - Continuous Pose Validation Sampler)	3.06 \pm 2.26
[83] (20k training pairs) (<i>Continuous Poses Training Dataset - Discrete Pose Train Sampler with Blended Hand - Discrete Pose Validation Sampler</i>)	4.09 \pm 3.69
[83] (20k training pairs) (<i>Continuous Poses Training Dataset - Continuous Pose Train Sampler with Blended Hand - Continuous Pose Validation Sampler</i>)	5.97 \pm 5.18

Table 9.2: The Normalized Pose Error metrics comparison for the architecture of [83], when this CNN design is trained on the Discretely/Continuously sampled training dataset, shuffled in a Discrete/Continuous manner and without/with a-priori blending the same occluder 3D continuous trajectory for every batch iteration.

As we can clearly see in Tables 9.1, 9.2:

- Sampling our training RGB-D frames continuously makes it easier for our baseline network to identify 3D translations. However, this is not the case for 3D rotations, as they become much more difficult, in this case.
- Sampling the validation set in a continuous manner simulates the testing conditions better. As a result, since the validation error is our only multi-dimensional minimum criterion, both tracking errors are smaller, when the tracker is evaluated on the test set.
- The effect of sampling on data augmentation is profound. The network’s augmentation capabilities have been divided by the sampling length (i.e. 25 frames), which gives it less randomized samples to be trained on. The same effect is aggravated on the occluder dataset, since the hand-occluder trajectories are entrenched and sampled per sequence, along with each training batch.
- The last observation becomes even clearer when we experiment with blending the hand-occluder on the object of interest during the generation stage, something that reduces the occlusion patterns by $No^{(iterations)} \times B$ times. The motivation for such an action is the demand for online Optical Flow generation during training (either in a per sample or in a per batch strategy), since in the remaining part of this Chapter we attempt to incorporate this modality in our training schemes as well.

As a result, a clear pattern of needing maximum variance in the augmentation transformation emerges. Any kind of discount in this aspect reduces the tracker’s accuracy proportionally.

So, we understand that our new baseline is that of the “continuously” generated synthetic training/validation dataset, which is shuffled in a “continuous” way, i.e. $43.78 \pm 12.19/31.03 \pm 33.00$. That is because our goal is to incorporate a recurrent unit in the NN architecture and recurrent units have the need to be both trained/validated and tested on sequences of data instead of independent frame pairs.

Next, we experiment with substituting either the first Linear of the last Fire layer with a Recurrent Linear or a Recurrent Convolutional architectural variation, as shown in figures 9.19, 9.20. As can be noticed in the Tables below, none of those recurrent variations was shown to consist an improvement over the baseline architecture of Garon et al. [83], shown in fig.6.27.

So, we hypothesized that the reason for this weakness would be that, as it has stated in multiple works in the past, Recurrent architectures are more difficult to train and more prone to be trapped in local minima (e.g. see the works of Pascanu et al. [99], Vaswani et al. [144]). We, at first thought, blame this observation to the fact that recurrent variations are much harder to train than purely Convolutional ones due to the fact that the BackPropagation Through Time (BPTT) variation forces the module to share its parameters not only per batch, but per sequence length as well. So, we tried to replace the final Linear layer with a hierarchical sequence of 1d (Temporal) convolutions (see Section 9.2.4), with filters with size 3, 5 and 7 and a dilation parameter of 2, applied along the 25 temporal dimension. However, if we take a look at the Tables below, they are still subpar to the classical Convolutional-Linear combination.

We note that, regarding the inference stage of the tracker in this case, that we model long term temporal dynamics either by a recurrent linear/convolutional unit or by a hierarchical sequence of 1d Convolutions, we initialize the recurrent unit’s memory with a learned initial value \mathbf{h}_0 at the beginning of the testing sequence and at every time we re-initialize the tracker (either every 15 frames or whenever it exceeds the maximum pose error values allowed).

Hard Parameter Sharing

Hard parameter sharing is one of the most commonly used approach in Multi-Task Learning for Neural Networks. It is generally applied by sharing the hidden layers between all tasks, while keeping several task-specific output layers.

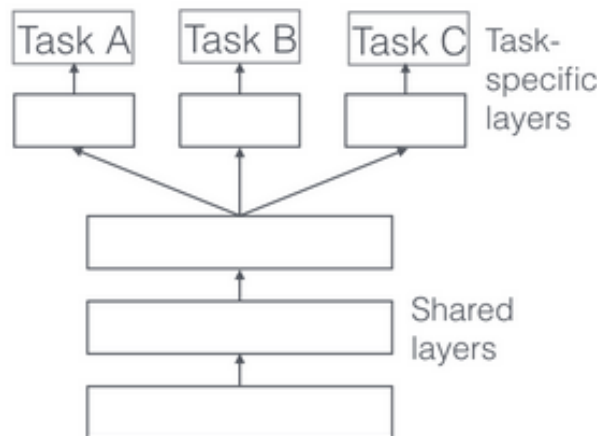


Figure 9.18: Hard parameter sharing for multi-task learning in deep neural networks. [115]

Hard parameter sharing greatly reduces the risk of overfitting. This makes sense intuitively: The more tasks we are learning simultaneously, the more our model has to find a representation that captures all of the tasks and the less is our chance of overfitting on our original task.

Occasioned by this property and with the intention to to incorporate the continuous trajectory information in our learning curriculum, but without disrupting the already satisfying convergence of the main Method proposed in this work, we employ the following strategy: we initialize the Network with the refined weights of the neural architecture of Chapter 6 and we alternate one batch of the “continuous trajectory” dataset with one, revised, batch of the old,faithfull “discrete” dataset in order to smooth out the learning procedure. Our aim is to, practically, incorporate the continuous motion patterns, but at the same time, not let the Net’s weights to diverge very much from the already reached local optimum.

In fact, we initialize the corresponding recurrent layer’s weights and biases with the equivalent weights and biases of the pre-tuned version of our tracker in order to give it a jumpstart. All layers **before and after** the recurrent one are shared.

Next, we observe the outcome of our experimentation.

Architecture	Translational Error (mm)	Rotational Error (o)
[83] (20k training pairs)	43.78 ± 12.19	31.03 ± 33.00
[83] (20k training pairs) (<i>RNN</i>)	48.93 ± 36.67	34.30 ± 37.22
[83] (20k training pairs) (<i>LSTM</i>)	45.63 ± 32.00	40.99 ± 35.58
[83] (20k training pairs) (<i>GRU</i>)	42.90 ± 29.51	31.82 ± 33.06
[83] (20k training pairs) (<i>FireLSTM</i>)	44.23 ± 24.97	33.98 ± 36.89
[83] (20k training pairs) (<i>TCNs</i>)	43.99 ± 18.78	32.86 ± 36.07

Table 9.3: The 3D Translational and Rotational error metrics comparison between the architecture of [83] and its variations with the use of single-layered recurrent modules (RNN / LSTM / GRU / FireLSTM / TCNs) in place of the first Linear/last Fire layer.

Architecture	Normalised Pose Error (mean ± std)
[83] (20k training pairs)	3.06 ± 2.26
[83] (20k training pairs) (<i>RNN</i>)	3.69 ± 3.98
[83] (20k training pairs) (<i>LSTM</i>)	3.25 ± 3.21
[83] (20k training pairs) (<i>GRU</i>)	3.08 ± 2.77
[83] (20k training pairs) (<i>FireLSTM</i>)	2.72 ± 2.85
[83] (20k training pairs) (<i>TCNs</i>)	3.17 ± 2.63

Table 9.4: The 3D Translational and Rotational error metrics comparison between the architecture of [83] and its variations with the use of single-layered recurrent modules (RNN / LSTM / GRU / FireLSTM / TCNs) in place of the first Linear/last Fire layer.

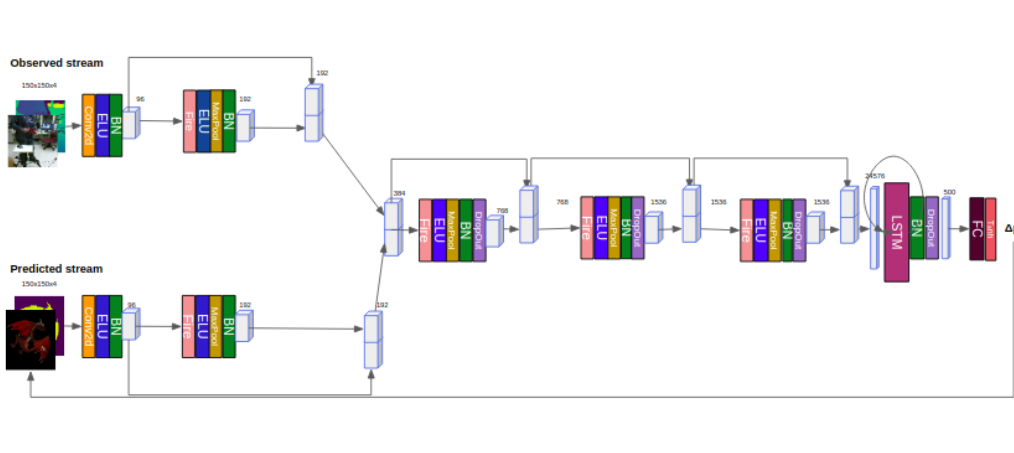


Figure 9.19: Overview of an alteration of the baseline [83] CNN architecture where the first Linear layer is replaced by a recurrent type (RNN/LSTM/GRU) one with the same number of neurons.

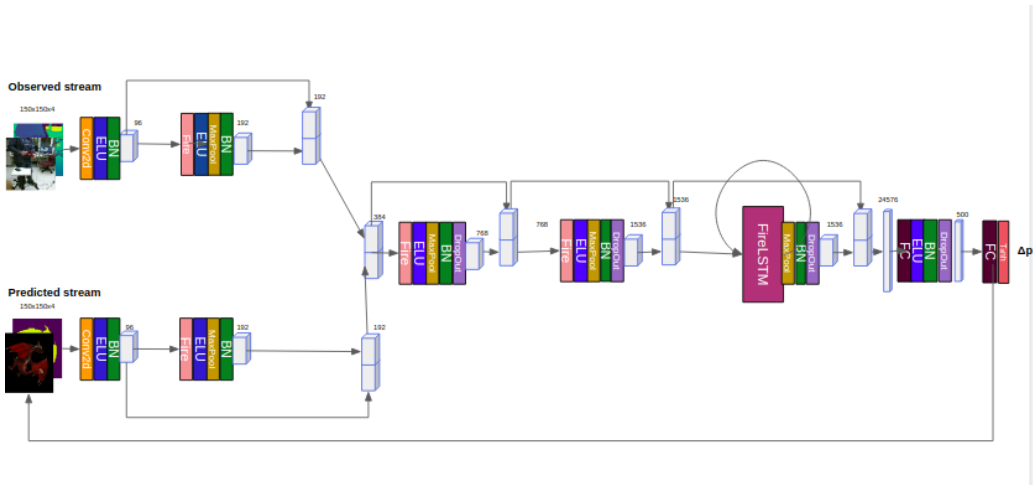


Figure 9.20: Overview of an alteration of the baseline [83] CNN architecture where the last Fire layer is replaced by a FireLSTM one with the same number of filters.

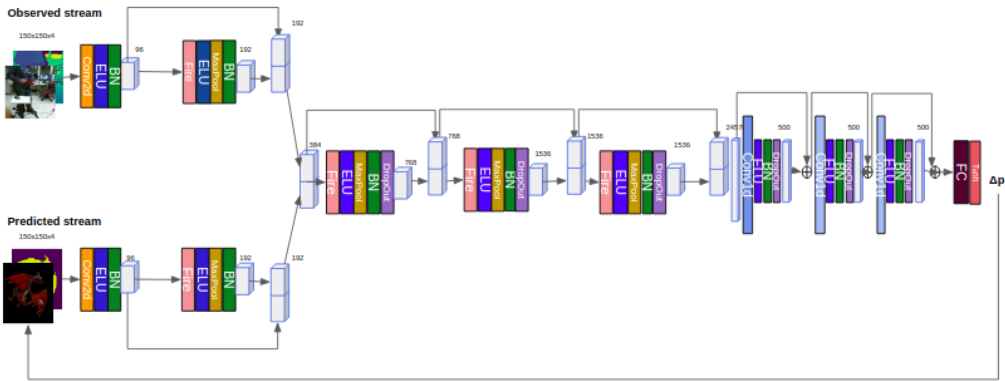


Figure 9.21: Overview of an alteration of our CNN architecture where the last Fire layer is replaced by a FireLSTM one with the same number of filters.

Since these results seem weird at a first sight, we repeat the experiments, this time, by making the same modifications to our, improved architecture. However, our measurements are not different.

We pinpoint the inefficiency of all this design mostly to the following factor

- Modeling the continuity of 3D trajectory along the time component is required to be realistic enough to match the real scenarios. Maybe there is room for improvement in the trajectory planning strategy described above in order to match the motion dynamics produced by the interaction of the object with the user’s hands.

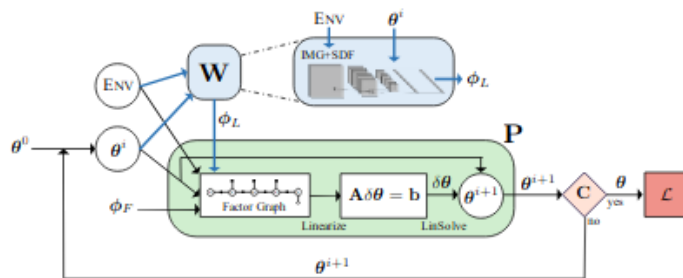


Figure 9.22: The computational graph of the “Differential Gaussian Process Motion Planner2” (dGMPM2) where ϕ_F are user defined planning parameters that are fixed and ϕ_L are learned planning parameters [6].

As a matter of fact, we blame the poor quality of our synthetically generated continuous trajectories to the fact that they were explicitly designed and we hope that, by turning to Trajectory Optimization-based Motion Planning we will have better luck. In particular, inspired by the approaches of VIBE[64] and CARL [80] frameworks, we intend to move towards an “Analysis-by-Synthesis” hypothesis.

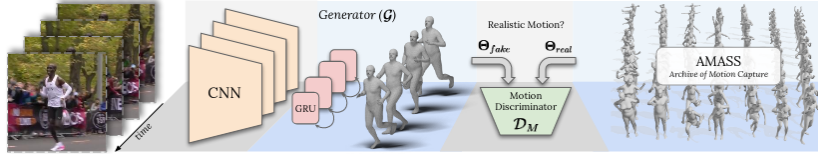


Figure 9.23: The “VIBE” Human Pose Estimation framework. [64]

We will formulate a GAN pair, where the Generator of pose trajectories we will use will be the learnable **dGPMP2** framework (fig. 9.22). It consists of a differentiable alteration of the State-of-the-Art “Gaussian Process Motion Planner 2 (GPMP2)”, that formulates 3D trajectory planning as an inference problem on factor graphs that is optimized by non-linear iterative algorithms. dGPMP2 will use half of the Ground Truth 6D Continuous Pose Trajectories of all the “Interaction” scenarios, for all other objects, except the one at hand at each time, to minimize a complex loss function that combines ground truth trajectory imitation, velocity and task specific constraint, as well as, curiosity regularization, sublosses. This differentiable motion planner will render the 6 degrees of freedom of motion for each frame, for T frames, and will be trained through the BPTT weight update algorithm. In an Generative Adversarial framework, it will be trained alongside with a combined ConvLSTM-Transformer[145] Discriminator design, that will be trained on the other half of the ground truth continuous trajectories available. The Discriminator’s duty would be to decide if each of the trajectory patterns produced by the differentiable Motion Planner. The non-realistic synthetic motion patterns would be discarded and replaced by new candidates. When this GAN pair converges, the Planner’s weights will be frozen and will yield a new continuous synthetic training dataset that we will feed to our recurrent variation of the approach proposed in Chap.6, hoping that it will bridge the motion domain gap and adapt the tracker to realistic conditions.

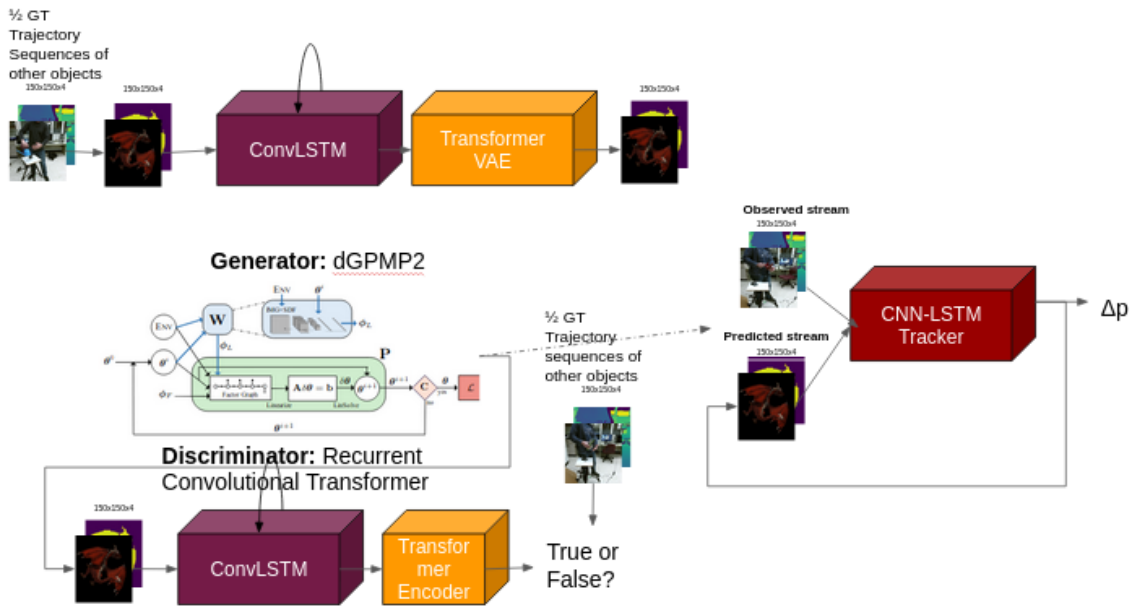


Figure 9.24: Coarse overview of the GAN framework for realistic 6D Pose Trajectory planning we are exploring for an extension of this work, in order to resolve the realism problem.

In the meantime, according to our experiments, intentional and explicit modeling of 3D long-term

object dynamics inside the Neural architecture has shown to be unfavorable compared with classical CNN approaches, thus far. This result comes into contradiction with our intuitive understanding of the continuity of object trajectories and our instinct stating that knowing the whole object pose history, in the past, will result in a more accurate pose estimation at the current timestep. However, these results vindicate all previous field researchers that had not tried to model 3D continuous synthetic trajectories in the past in order to train Neural architectures on them, as well as research teams in other fields that progressively turn their interest from explicit temporal modeling of sequences into independent instance-level configurations (e.g. the field of Natural Language Processing has, throughout the last few years, replaced recurrent modules with architectural schemes based on temporal attention such as (shallow) Transformer layouts [144]). Such layouts offer better training parallelization capabilities, as well as basic mathematical modular operations that facilitate the BackPropagation algorithm’s function. However, they are, at present, extremely computationally heavy; so much that they exceed the real-time performance barrier. If that is the answer for continuous trajectory modeling, we will have to wait for them to become more efficient before they are employed in real-time applications, such as object tracking.

9.3 Real-time Small Drifts counter Distilled Optical Flow fusion

A second thought, we experiment with in this Thesis in order to extend our proposed approach, is trying to explicitly model short term temporal dynamics in order to counter pose drift. With this in mind, we account for an extra information source cue that projects 3D pose differences in the 2D image plane in an immediate way: **(Dense) Optical Flow frames**.

Just like the RGB and the Depth image cue, Optical Flow information source has its own distinct pros and cons:

- **Advantages:**
 - It provides short term motion information that makes our prediction more sophisticated.
- **Disadvantages:**
 - Its production increases the time complexity of the tracker
 - It mostly contains information more about the 2D object translation and less about the rest of the components of the object’s pose.

We aim to find an optimal architectural design that utilizes its advantages in order to enhance the pose distinctive capabilities of the other two modalities, while finding a way to circumvent its disadvantages.

Based on our synthetic dataset structure, we experiment with three kinds of Optical Flow information: The “Observed” Optical Flow $OF_{Obs}(t)$, the “Predicted” Optical Flow $OF_{Pred}(t)$, the Augmented Reality Flow image $AF(t)$ and its inverse: $InvAF(t)$.

These three variations are obtained via a forward pass of the pretrained “FlowNet2” architecture:

$$\begin{aligned}
 AF(t) &= FlowNet2(\hat{I}(t), I(t)) \\
 InvAF(t) &= FlowNet2(I(t), \hat{I}(t)) \\
 OF_{Observed}(t) &= FlowNet2(I(t-1), I(t)) \\
 OF_{Predicted}(t) &= FlowNet2(\hat{I}(t-1), \hat{I}(t))
 \end{aligned}
 \tag{9.26}$$

(Note: since binding the object with the occluder model during generation drops the generalization capabilities of the tracker by a high margin, such an approach is rejected and we take as a premise that each of the three Optical Flow variations will be about only the object and not the occluder model movement. This way, we are certain that data augmentation has the same degrees of freedom as in Chapter 6, modulo the 25-frame length of the sequence. In fact, some short preliminary experiments show us this strategy to be more favourable, indeed, in comparison to making the opposite design choice.)

We chose the FlowNet2 learnable architecture to produce our Dense Optical Flow images. This choice was deliberately made for two reasons:

1. We chose a CNN based approach instead of a more traditional variational one, like TVL1 [103] or the approach of Brox et al. [10], as they are typically faster
2. We prefer the FlowNet2 architectural design over some of its more modern competitors (like the PWCNet [125] or the Spatial Pyramid Network (SpyNet) [109] that have presented optical flow predictions of better quality when tested on the toughest benchmark datasets of the field) because FlowNet2 was the only one that was pretrained on the 'Flying Chair' dataset [52], a synthetic dataset with objects that appear at the same scale as the ones found in our benchmark dataset of Garon et al. [83]. This way, we minimize the Optical Flow estimation error and we leave learning of the tracker's weights to be our main source of glitch.

At first, we have to find a way to get over the following adversity: even if adding an Optical Flow information of some kind will improve the tracker's performance (even if it is only for the translational component), an online computation of the Optical Flow stream during the inference stage will slow us down far beyond the real time boundaries, as a forward pass of FlowNet2 exceeds 200ms. Therefore, even if we find an appropriate information fusion configuration that, indeed, improves the tracker's accuracy, we need to accompany it with a second counterpart, that will learn to distill the Optical Flow information, from the samples that are available during the non-time-critical training, and enhance the tracker's capabilities for its inference.



Figure 9.25 The "Observed" synthetic dense Optical Flow image produced by passing two consecutive "Observed" RGB frames, from one of the continuous trajectories of the previous Section, via the pretrained FlowNet2.



Figure 9.26 The "Predicted" synthetic dense Optical Flow image produced by passing two consecutive "Predicted" RGB frames, from one of the continuous trajectories of the previous Section, via the pretrained FlowNet2.



Figure 9.27 The synthetic dense "Augmented Reality" Optical Flow image (AF) [100] produced by passing an "Observed"- "Predicted" RGB frame pair via the pretrained FlowNet2.



Figure 9.28 The synthetic dense Inverse "Augmented Reality" Optical Flow image (InvAF) [100] produced by passing an "Observed"- "Predicted" RGB frame pair via the pretrained FlowNet2.

Again, the comparison is performed for the, more general, asymmetric case, i.e. using the dragon 3D model. At first, we need to validate the distinctive ability of the Optical Flow source information as far as it concerns understanding the temporal pose difference. So, at first, we exploit the effect of using the Optical Flow as our only source for tracking the object's pose in a variety of configurations. Our intuition is that the AF cue will reduce the pose drift (at least for the translational component) and, thus, reduce the tracking error, while the use of the other two Optical Flow streams will enhance the robustness of the pose estimation.

- **AF - based tracking:**

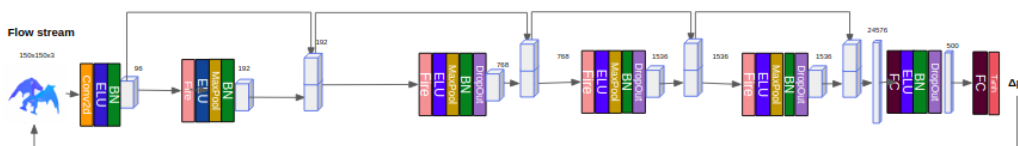


Figure 9.29: Overview of an alternative version of the SoA architecture of Garon et al. [83] with a single AF input.

- $\{OF_{Obs} + AF\}$ - based tracking:

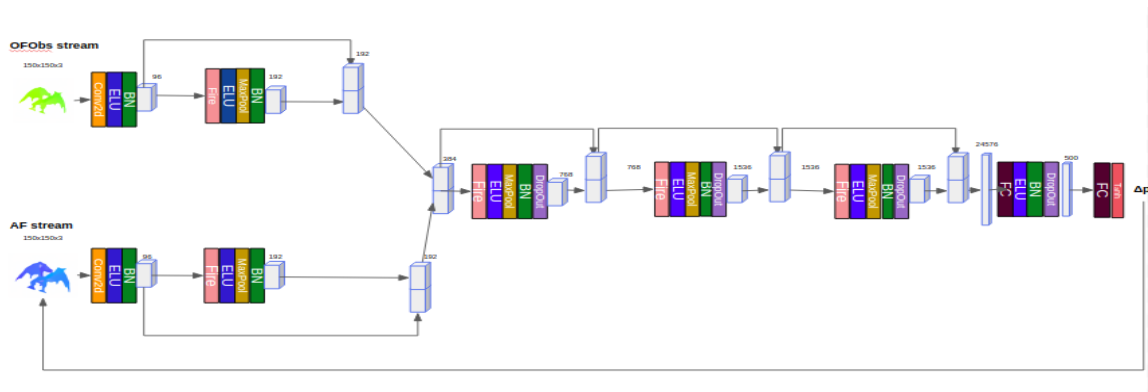


Figure 9.30: Overview of an alternative version of the SoA architecture of Garon et al. [83] with two streams with Observed Optical Flow and AF inputs, correspondingly.

- $\{OF_{Obs} + OF_{Pred}\}$ - based tracking:

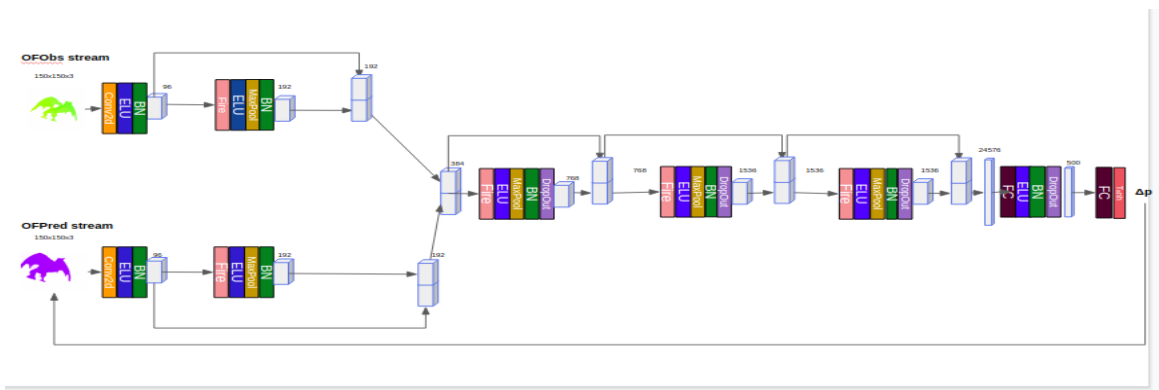


Figure 9.31: Overview of an alternative version of the SoA architecture of Garon et al. [83] with two streams with “Observed” Optical Flow and “Predicted” Optical Flow inputs, correspondingly.

- $\{OF_{Obs} + OF_{Pred} + AF\}$ - based tracking with concatenation at the input level:

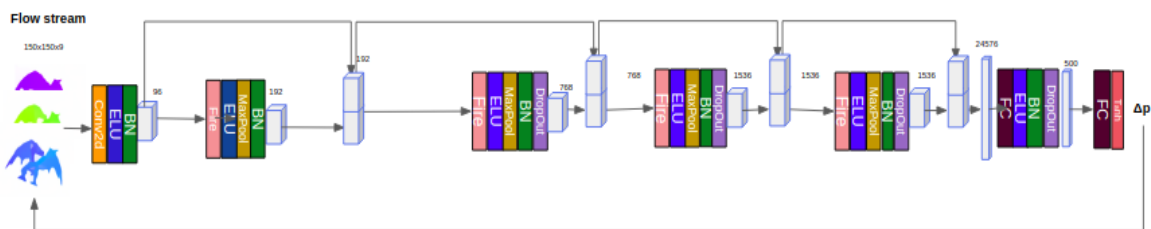


Figure 9.32: Overview of an alternative version of the SoA architecture of Garon et al. [83] with three concatenated input sources: “Observed” Optical Flow, “Predicted” Optical Flow and AF, correspondingly.

- $\{OF_{Obs} + OF_{Pred} + AF\}$ - based tracking with early fusion at the feature level:

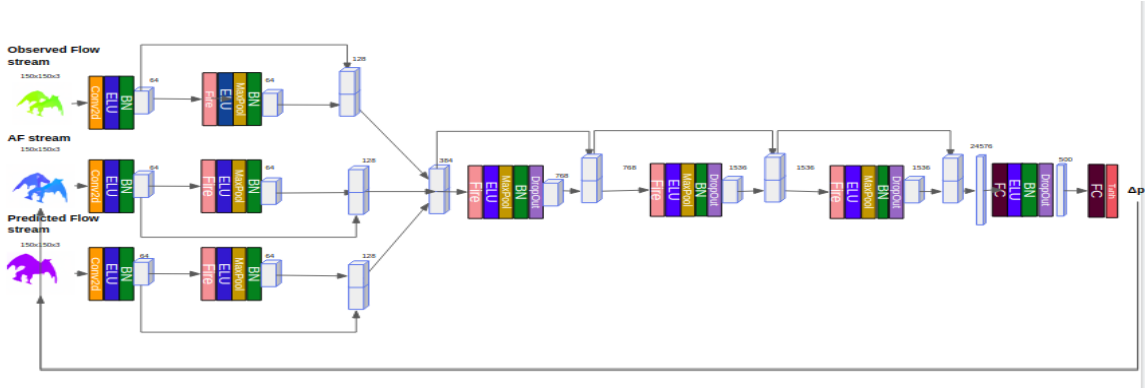


Figure 9.33: Overview of an alternative version of the SoA architecture of Garon et al. [83] with three streams, with “Observed” Optical Flow, “Predicted” Optical Flow and AF inputs, correspondingly.

Architecture	Translational Error (mm)	Rotational Error (o)
[83] (20k training pairs)	48.58 ± 38.23	35.43 ± 34.74
[83] (20k training pairs) with AF input	45.31 ± 37.74	48.30 ± 35.05
[83] (20k training pairs) with “Observed” Optical Flow and AF streams	41.21 ± 33.62	43.90 ± 33.68
[83] (20k training pairs) with “Observed” and “Predicted” Optical Flow streams	62.09 ± 40.11	55.18 ± 37.13
[83] (20k training pairs) with “Observed” Optical Flow, “Predicted” Optical Flow and AF inputs	51.04 ± 37.23	51.76 ± 36.05
[83] (20k training pairs) with “Observed” Optical Flow, “Predicted” Optical Flow and AF-based feature streams	49.08 ± 34.21	42.27 ± 33.02

Table 9.5: The Translational and Rotational Error comparison of the variations of the SoA architecture of Garon et al. [83], with different Optical Flow inputs, instead of RGB-D ones.

Architecture	Normalised Pose Error (mean \pm std)
[83] (20k training pairs)	3.45 ± 3.11
[83] (20k training pairs) with AF input	4.1 ± 3.11
[83] (20k training pairs) with “Observed” Optical Flow and AF streams	3.84 ± 2.92
[83] (20k training pairs) with “Observed” and “Predicted” Optical Flow streams	4.98 ± 3.30
[83] (20k training pairs) with “Observed” Optical Flow, “Predicted” Optical Flow and AF concatenated inputs	4.50 ± 3.16
[83] (20k training pairs) with “Observed” Optical Flow, “Predicted” Optical Flow and AF feature streams	3.86 ± 2.90

Table 9.6: The Normalized Pose Error comparison of the variations of the SoA architecture of Garon et al. [83], with different Optical Flow inputs, instead of RGB-D ones.

As we observe in Tables 9.5, 9.6:

- The AF modality gives us strong hints that will help in tracking the 3D position of the object of interest, if not suffice by itself. However, its performance for the rotation component estimation is rather weaker.
- Utilizing the “Observed” Optical Flow, for feature comparison with the Augmented Reality Flow (AF), seems to enhance this hint and improve the rotational component along with the translational one. In particular, beyond the tracking error, this cue improves the estimations’ robustness, as they are expressed by the standard deviation of the pose error metrics.
- Using the “Predicted” Optical Flow modality in contrast with the “Observed” one, without/with the

enhancing presence of the AF, appears to give no benefits to the tracker’s capabilities. Therefore, it will be excluded from the rest of our study.

- Fusing the Optical Flow inputs at the feature level appears to be of greater importance to the tracker’s generalization capabilities that fusing them at the input level.
- Since this family of experimental tests is performed by modifying the algorithm of Garon et al. [83], where the training is stopped by an Early Stopping mechanism, we observe that using Optical Flow information as inputs, in general, results in higher convergence speed. but also, in faster overfitting on the training set. For example, in the architectural layout of fig.9.29, the tracker converges to its local minimum in 16 epochs (instead of 25 for the RGB-D version of Garon et al.) and strongly intensely overfits on the training set for the rest of them, until reaching the 25th epoch. We observe similar behaviour in the other architectural modifications, with inputs of Optical Flow nature, as well. In fact, the more dense Optical Flow images we use as inputs, the faster this phenomenon occurs. We appoint this behaviour to the fact that the Optical Flows are not augmented as strongly as their RGB-D counterparts, as they are required to be saved a-priori on the dataset, for the training to be computationally feasible, and not recalculated over every augmented RGB pair batch. This symptom appears to be rather suspicious, as far as it concerns the limits of the Optical Flow modality to enhance the tracker’s performance. Our bet, in the rest of the Chapter is finding the most appropriate fusion scheme between this visual cue and its RGB-D counterparts that will surpass this adversity.

Next, we will explore whether combining these motion information with the RGB-D-based features of our proposed CNN will yield an improved performance. We will discuss the potential pros and cons of such an architectural alteration and experiment with a variety of possible architectural changes.

9.4 RGB-D + Optical Flow Fusion

The first fusion strategy family that comes to our mind is the straightforward one, where the AF and/or the “Observed” Optical Flow cues are fused with the RGB-D features, via concatenation, either at the input or the feature level. Therefore, we examine the viability of the following three configurations:

- **RGB-D+Flow concatenation at the input level:**

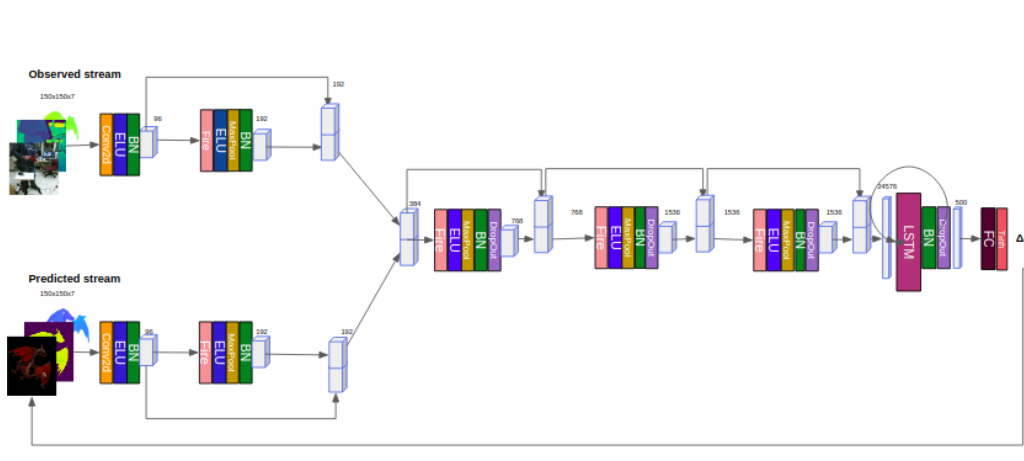


Figure 9.34: Overview of a variation of the SoA architecture of Garon et al. [83] with concatenated RGB-D-Flow types of input.

- **Early RGB-D + Flow feature concatenation fusion:**

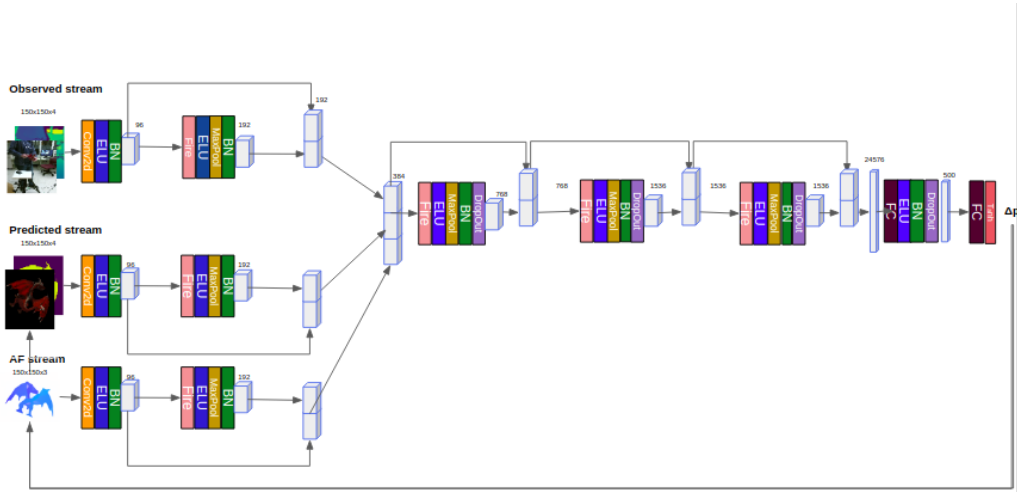


Figure 9.35: Overview of a variation of the SoA architecture of Garon et al. [83] with Early (concatenation-based) Fusion of the RGB-D and Flow-based feature streams.

- Late RGB-D + Flow feature concatenation fusion:

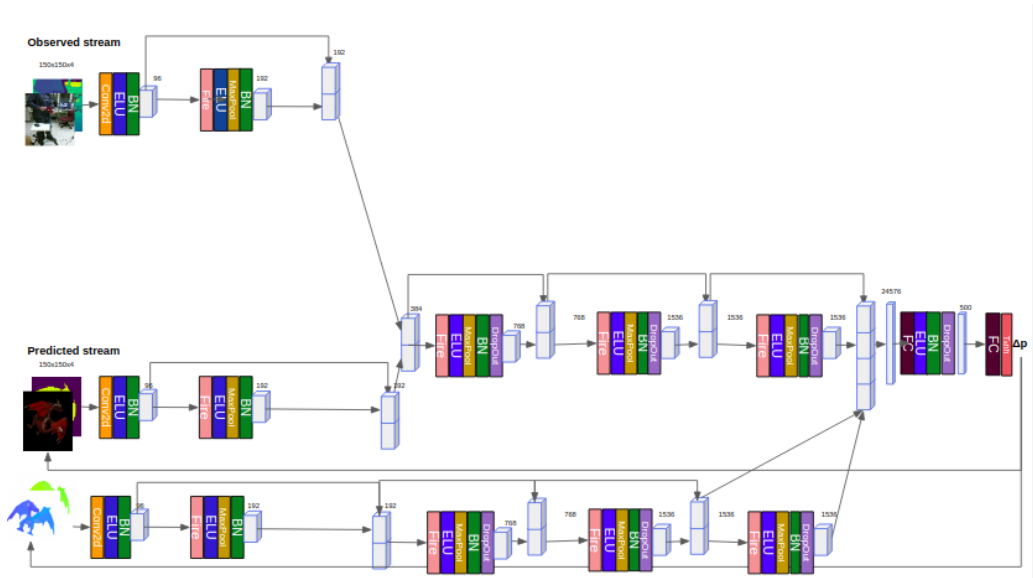


Figure 9.36: Overview of a variation of the SoA architecture of Garon et al. [83] with Late (concatenation-based) Fusion of the RGB-D and Flow streams.

Architecture	Translational Error (mm)	Rotational Error (o)
Approach of Chapter 6 (20k training pairs)	11.63 ± 8.79	8.31 ± 6.76
Approach of Chapter 6 with RGB-D + Flow concatenation at the input level (20k training pairs)	20.34 ± 9.42	14.71 ± 9.88
Approach of Chapter 6 with Early RGB-D + Flow feature concatenation (20k training pairs)	13.47 ± 10.28	10.01 ± 7.54
Approach of Chapter 6 with Late RGB-D + Flow feature concatenation (20k training pairs)	15.47 ± 9.63	11.22 ± 8.00

Table 9.7: The Translational and Rotation Error metric comparison of the SoA RGB-D architecture of Garon et al. [83] with its variations of Fusing the Flow-based information at different layers of the Network.

Architecture	Normalised Pose Error (mean \pm std)
Approach of Chapter 6 (20k training pairs)	0.82 \pm 0.64
Approach of Chapter 6 with RGB-D + Flow concatenation at the input level (20k training pairs)	1.43 \pm 0.85
Approach of Chapter 6 with Early RGB-D + Flow feature concatenation (20k training pairs)	0.97 \pm 0.73
Approach of Chapter 6 with Late RGB-D + Flow feature concatenation (20k training pairs)	1.09 \pm 0.73

Table 9.8: The Normalized Pose Error metric comparison of the SoA RGB-D architecture of Garon et al. [83] with its variations of Fusing the Flow-based information at different layers of the Network.

According to Tables 9.7,9.8, it seems that fusion-via-concatenation is not the way to go for the RGB-D + Optical Flow modalities. So, we need to think of more sophisticated strategies to incorporate the motion information in the core of the pose comparison. In the following subsections, we experiment with three completely different approaches and discuss our findings.

9.4.1 Optical Flow guided Foreground and Occlusion handling Attention modules

The first sophisticated incorporation of the Optical Flow modality that we thought is using either/both the “Observed” Optical Flow or/and the AF, but this time, its Backward version, as guiding signals for the Foreground Handling/Occlusion Handling Attention modules, correspondingly. Our intuition for such a decision is strong:

- Firstly, we observe that the background elements are not moving in the testing dataset, which propel the “Observed” Optical Flow to become the Clutter Handling Attention input signal (with or without the use of the corresponding binary supervising signal and the BCE loss function). We speculate that such an immediate correlation between the foreground features and the overall feature map could improve the effectiveness of this Attention module, in comparison to the self-attention variation of Chapter 6.
- Secondly, we think of the work of Wang et al. [150], which concerns the field of Unsupervised Optical Flow estimation. There, in a similar layout to ours, they use the Backward AF cue to produce occlusion weight maps that handle dynamic occlusions present in a synthetic scene. Their argument is that this occlusion map indicates the region in I_1 (here, the “Predicted” RGB frame) that is correspondingly occluded in I_2 (here, the “Observed” RGB frame) (i.e. region in I_1 that does not have a correspondence in I_2), i.e. if a pixel is not occluded in the initial frame, between the two, and occluded in the second one, this difference will be imprinted in the Backward AF image. Thus, Backward AF will serve as a suitable guiding input signal for occlusion handling. In our work, the first RGB frame is the “Predicted” frame and the second the “Observed” one, so we have no way to distinguish static from dynamic occlusions.

“Observed” Optical Flow guided Clutter Handling Attention

In the next figure, we present a variation of the architecture of Chapter 6, where the input signal for the Foreground Spatial Attention module is not the feature map that it will be applied to, but the feature maps generated by a stream with the “Observed” Optical Flow, as input.

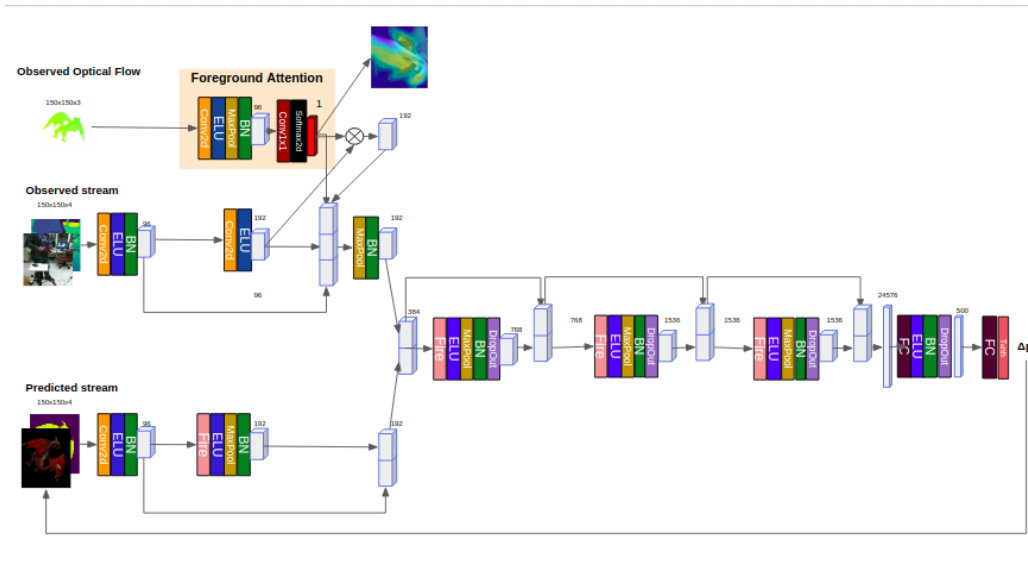


Figure 9.37: Overview of our approach of Chapter 6 where the Foreground Attention module is guided by the “Observed” Optical Flow stream.

In particular, we compare the approach of Chapter 6 with the following three variations, whose architectural design is that of fig.9.37:

- Soft Spatial Attention Background Clutter Handling using “Observed”-stream Optical Flow:** Here, we use the same Attention module scheme as in Chapter 6, but instead of using the “Observed” feature map as input, we leverage the feature map produced by a separate stream. That stream uses the “Observed-stream” Optical Flow image (converted to RGB color code). This idea is based on the observation that Optical Flow may aid Foreground-Background segmentation, due to the fact that moving objects are depicted to have very different Optical Flow values than the steady background.
- Soft Spatial Attention Background Clutter Handling using “Observed”-stream Optical Flow using Binary Cross Entropy with steady weights:** We use the same Attention scheme as before. The difference, now, is that the Optical Flow-based Attention weight map is not let free to generate whatever patterns will emerge, but it is rather strictly supervised adding to the Track Loss (MSE) a Binary Cross Entropy loss function (see Section 3.3.17). The ground truth signal used to guide the Binary Cross Entropy loss function is a binary foreground-background map, kept during the Data Augmentation process.
- Soft Spatial Attention Background Clutter Handling using “Observed”-stream Optical Flow using Binary Cross Entropy with learnable weights:** Here, we employ the same supervised Attention scheme as above. The difference, now, is that instead of just adding the Binary Cross Entropy Loss to the Track one, we leverage both using the learnable MultiTask weighting scheme, described in Section 8.2.

Architecture	Translational Error (mm)	Rotational Error (o)
Approach of Chapter 6 (20k training pairs)	11.63 ± 8.79	8.31 ± 6.76
Approach of Chapter 6 with Clutter Handling Spatial Attention guided by the “Observed” Optical Flow cue (20k training pairs)	62.21 ± 47.7	37.43 ± 35.63
Approach of Chapter 6 with Clutter Handling Spatial Attention guided by the “Observed” Optical Flow cue using Binary Cross Entropy loss with steady weights (20k training pairs)	25.06 ± 16.24	39.67 ± 37.10
Approach of Chapter 6 with Clutter Handling Spatial Attention guided by the “Observed” Optical Flow cue using Binary Cross Entropy loss with learnable weights (20k training pairs)	15.20 ± 9.02	39.94 ± 34.86

Table 9.9: The Translational and Rotation Error metric comparison between our approach of Chapter 6 and its variations where the Clutter Handling Spatial Attention module is guided by an “Observed” Optical Flow input type.

Architecture	Normalised Pose Error (mean \pm std)
Approach of Chapter 6	2.78 ± 2.63
Approach of Chapter 6 with Clutter Handling Spatial Attention guided by the “Observed” Optical Flow cue (20k training pairs)	4.00 ± 3.52
Approach of Chapter 6 with Clutter Handling Spatial Attention guided by the “Observed” Optical Flow cue (20k training pairs) using Binary Cross Entropy loss with steady weights (20k training pairs)	3.01 ± 2.61
Approach of Chapter 6 with Clutter Handling Spatial Attention guided by the “Observed” Optical Flow cue (20k training pairs) using Binary Cross Entropy loss with learnable weights (20k training pairs)	2.74 ± 2.27

Table 9.10: The Normalized Pose Error metric comparison between our approach of Chapter 6 and its variations where the Clutter Handling Spatial Attention module is guided by an “Observed” Optical Flow input type.

In Tables 9.7,9.8, we observe that our experiments do not validate our hint. In particular, the supervised “Observed” Optical Flow based attention scheme, with learnable weights, came first among the various Optical Flow guided strategies. However, their performance is worse than the one of the variation that used self-Attention. All of them achieve to enhance the Foreground pixels of the moving object of interest and, as we, gradually, supervise and weight our attention weight maps better and better, the quality of this enhancement improves. An interesting qualitative observation we make for these suite of strategies is that the attention maps they produce focus not only to the pixels of the object of interest, but they, also light up pixels of the moving parts of the background, in the video stream. This behaviour, profoundly, plays a significant distractive role in the distinctive capabilities of the trainable tracker. We assume, that this is the reason that this kind of strategies do not surpass the self-Attention variation of Chapter 6.

Backward “Augmented Reality” Optical Flow guided Occlusion Handling Attention

In the next figure, we present a variation of the architecture of Chapter 6, where the input signal for the Foreground Spatial Attention module is not the feature map that it will be applied to, but the feature maps generated by a stream with the Backward AF (see fig.9.28), as input.

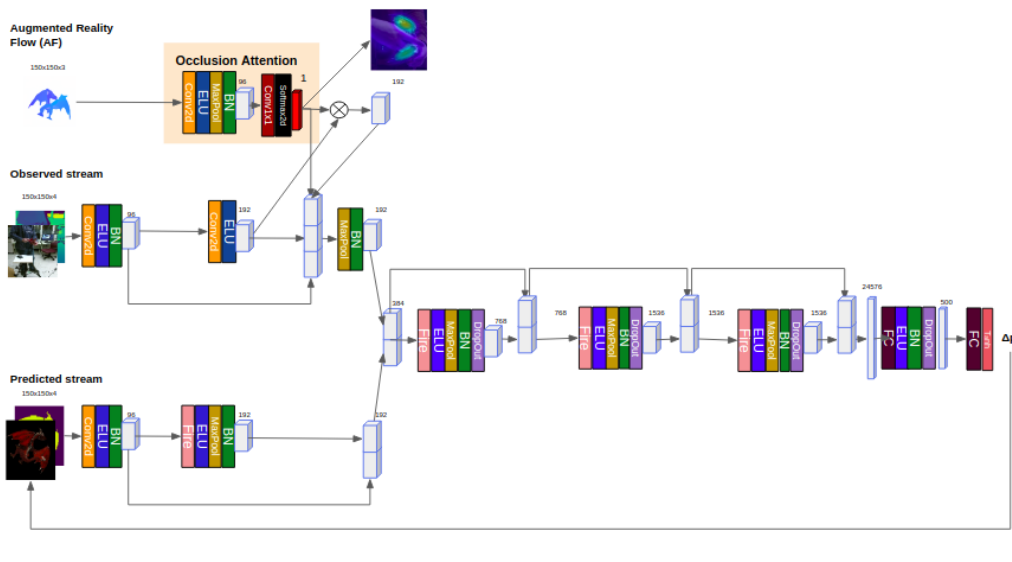


Figure 9.38: Overview of our approach of Chapter 6 where the Occlusion Handling Attention module is guided by the Backward AF stream, following the idea of the paper of Wang et al.[150].

In particular, we compare the approach of Chapter 6 with the following three variations, whose architectural design is that of fig.9.38:

- Soft Spatial Attention Occlusion Handling using Backward Augmented Reality Optical Flow (AF):** Here, we use the same Attention module scheme as in Chapter 6, but instead of using the “Observed” feature map as input, we leverage the feature map produced by a separate stream. That stream uses the “Observed-stream” Optical Flow image (converted to RGB color code). This idea is based on the observation that Backward AF depicts parts of the scene present in the “Observed” RGB frame (and thus, in the Depth map, as well) that are not present in the “Predicted” RGB frame, that we know to contain only object attributes and thus, could help guide the Occlusion Handling module in a more accurate way than the self-Attention convolutional stream.
- Soft Spatial Attention Occlusion Handling using Backward Augmented Reality Optical Flow (AF) using Binary Cross Entropy with steady weights:** We use the same Attention scheme as before. The difference, now, is that the Attention weight map is not let free to generate whatever patterns will emerge, but it is rather strictly supervised adding to the Track Loss (MSE) a Binary Cross Entropy loss function (see Section 3.3.17). The ground truth signal inputed to this Binary Cross Entropy loss function is a binary foreground-background map, kept during the Data Augmentation process.
- Soft Spatial Attention Occlusion Handling using Backward Augmented Reality Optical Flow (AF) using Binary Cross Entropy with learnable weights:** Here, we employ the same supervised Attention scheme as above. The difference, now, is that instead of just adding the Binary Cross Entropy Loss to the Track one, we leverage both using the learnable Multi-Task weighting scheme, described in Section ???.

Architecture	Translational Error (mm)	Rotational Error (o)
Approach of Chapter 6 with Occlusion Handling Spatial Attention guided by the Backward AF cue (20k training pairs)	22.67 ± 13.24	14.89 ± 9.34
[83] (20k training pairs) (<i>Soft Spatial Attention Occlusion Handling using Backward Augmented Reality Optical Flow (AF)</i>)	18.03 ± 11.56	11.96 ± 8.04
[83] (20k training pairs) (<i>Soft Spatial Attention Occlusion Handling using Backward Augmented Reality Optical Flow (AF) using Binary Cross Entropy with steady weights</i>)	15.46 ± 10.03	11.08 ± 7.99
[83] (20k training pairs) (<i>Soft Spatial Attention Occlusion Handling using Backward Augmented Reality Optical Flow (AF) using Binary Cross Entropy with learnable weights</i>)	12.88 ± 9.57	10.21 ± 8.01

Table 9.11: The Translational and Rotation Error metric comparison between our approach of Chapter 6 and its variations where each Spatial Attention module is guided by a Backward Augmented Reality Flow (AF) input type.

Architecture	Normalised Pose Error (mean \pm std)
Approach of Chapter 6 (20k training pairs)	0.82 ± 0.64
Approach of Chapter 6 with Occlusion Handling Spatial Attention guided by the Backward Augmented Reality Optical Flow (AF) cue (20k training pairs)	1.21 ± 0.80
Approach of Chapter 6 using Backward Augmented Reality Optical Flow (AF) cue (20k training pairs) using Binary Cross Entropy with steady weights	1.09 ± 0.75
[83] using Backward Augmented Reality Optical Flow (AF) cue (20k training pairs) using Binary Cross Entropy with learnable weights	0.96 ± 0.74

Table 9.12: The Normalized Pose Error metric comparison between our approach of Chapter 6 and its variations where each Spatial Attention module is guided by a Backward Augmented Flow (AF) input type.

In Tables 9.11,9.12, we observe that the use of the Backward AF does not give us the effects we wish for, at least at the extend self-Attention does in Chapter 6, and thus, comes short in replacing it as the new State-of-the-Art. Qualitatively, the unoccluded attention weight maps produced without the supervision by binary occlusion masks are blurry and tend more to a uniform pixel distribution instead of highlighting specific pixel areas of interest in a deliberate way. As we supervise these maps in a more strict way, exactly as we did in Chapter 7, we, indeed, see clearer attention patterns to rise and highlight object parts that are essential to recognizing its pose. However, the occluder receives very small to no attention at all, and that has a profound cause: as we use the Backward version of AF, the hand-occluder depiction is not present in it, and thus, does not guide the Network to focus on it, in comparison to other motion-based attention modifications we examined, such as the ones with the Occlusion ConvLSTMs, where the attention was spread at a percentage of the occluder, as well.

9.4.2 Feature map Warping using Spatial Transformer Networks

The second sophisticated RGB-D + Optical Flow fusion strategy that we examine is the one presented in the following figure.

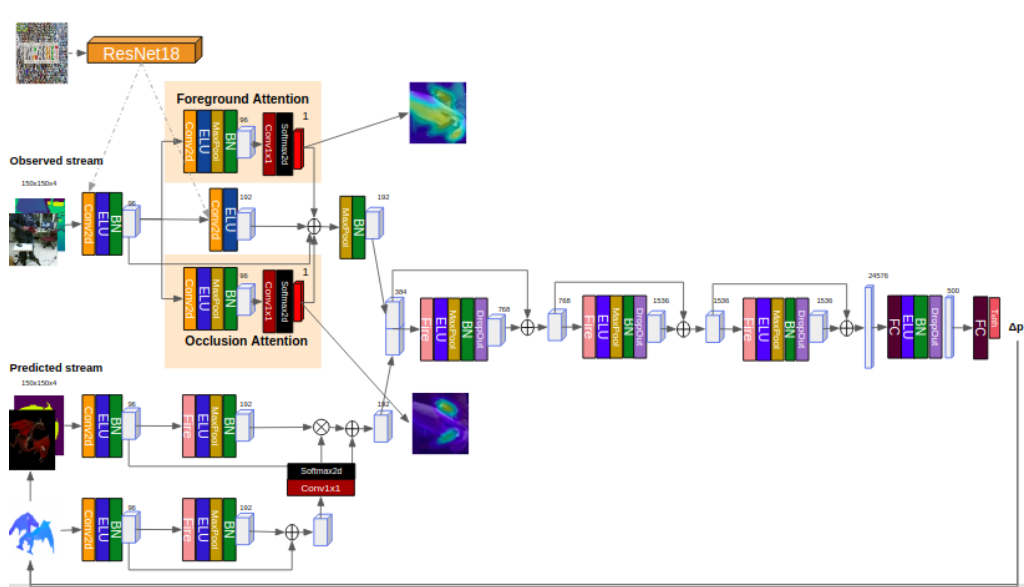


Figure 9.39: Overview of a modified version of our approach of Chapter 6 where the Augmented Reality Flow (AF) is warped at the feature level via a Spatial Transformer Network module.

Let’s briefly explain our thought pattern. A potential drift of pose estimation at every timestep consists one of the main causes our tracker may lose trace of the object at some point. Due to the way the preprocessing of Chapter 6 was performed, the 2D bounding box that is generated by the “Predicted” stream is the one guiding the information context of the “Observed” RGB-D testing frame that will be fed to the CNN. This dependence was clearly shown in our Ablation Study in Section 7.2. While the pose errors are small, the feedback-generated rendered RGB-D frame will be close to the optimal one. This will allow the 2D bounding boxes drawn around the object to have an **Intersection over Union (IoU)** with their “Observed” counterparts close to 0, and, thus, crop in the totality of the object’s attributes, as depicted in the “Observed” RGB-D frame. When that happens, the feedback of the tracker will be negative, in terms of the pose error and will force it to converge to zero. On the other hand, as the small pose errors accumulate over time, the 2D bounding box estimation will diverge from the optimal one and will crop out more and more important features of the object of interest, present in the “Observed” RGB-D frame. This will cause a positive error feedback, that will grow at every consecutive iteration and will, finally, cause the tracker to fail.

In a nutshell, since we wish to eliminate this pose estimation drift, we intend to propagate the features generated by the “Predicted” stream to be more correlated to the ones of the corresponding “Observed” features. Therefore, our goal is to warp them using a separate feature stream, provided by the Forward AF cue, a modality that is a depiction of the projected difference of the two poses, the “Predicted” and the “Observed” one. However, a problem occurs. In order for this warping to be part of an end-to-end CNN architecture, we wish to execute it using a differentiable component. To this end, we employ the last step of a module called **Spatial Transformer Network**, a modular and differentiable image warping mechanism. In particular, we squeeze the feature channels of the AF-based feature map of the second convolutional layer to only 2 via a 1×1 convolution and we pass it through a Tanh activation function to constrain its values between $[-1,1]$. Then, we use this AF-based generated grid to warp the “Predicted” features of the corresponding second convolutional layer, in a differentiable manner (look at the next subsection about Spatial Transformer Networks, for details).

Beyond that, during training, at every batch, we attempt to help in reducing the Network’s overfitting to the presence of the AF cue, by switching off the Flow Warping stream, in relation to the outcome of a Bernoulli probability with a parameter $p=0.5$. When the outcome of this variable is 1, we only keep the components of Chapter 6, which we know to consist the optimal layout, thus far. We are inspired by the classical Dropout [121] approach for avoiding overfitting, this time scaled up to a whole Network stream.

The careful reader may remember the gap between the speed requirements for a real-time performance of our tracker and the time complexity that FlowNet2 need to produce a single AF image. To this end, should this approach presents results that overcome those presented in Chapter 8, we will discuss a potential way to try to distil the motion information, just by using RGB information cues. In the opposite case, we will proceed to the next Section, in order to study our third and last, sophisticated RGB-D fusion effort with the Optical Flow modality. It is noteworthy to mention that a similar approach has been proposed and successfully implemented in the LINEMOD popular 6D object pose estimation dataset, with an approach very similar to us, but with render-based pose feedback realized in the iteration instead of the temporal domain, by Trabelsi et al. in their recent work: [135].

Next, we establish the theoretical foundations of image warping with the use of Optical Flow information and we explain the structure and properties of the Spatial Transformer Network (STN) module.

Image Warping using Optical Flow

The (sparse) Optical Flow is the displacement vector between 2 separate screenshots of the video frames' evolution. Warping one of those frames using the Optical Flow means translating its values on the image plane according to it. If the starting frame is earlier in time, we are talking about **Forward Warping**, while in the opposite case, we are talking about **Backward Warping**.

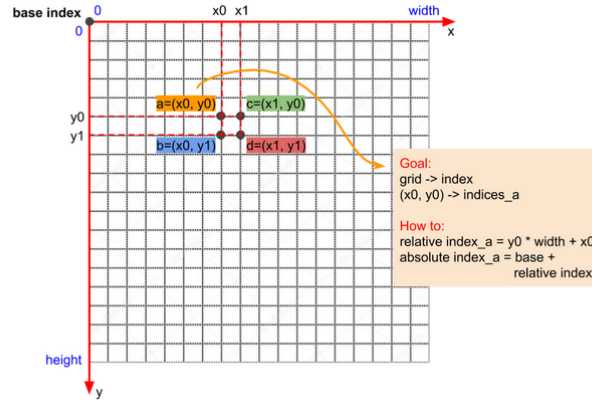


Figure 9.40: Bilinear pixel interpolation for completing Optical Flow-based image warping.[155]

Forward Warping:

We wish to apply the Optical Flow displacement vector $F(I_{t-1}, I_t)$ between 2 consecutive image samples to the former of the two: I_{t-1} in the forward case, and acquire a warped version of the latter: $I_t^{(W)} = \mathcal{W}(I_{t-1}, F(I_{t-1}, I_t)) = I_{t-1}(\mathbf{p} + \mathbf{F}(I_{t-1}, I_t))$. In order to achieve so, we translate every image pixel according to both the length and the direction of the aforementioned Optical Flow vector. One would assume that our work is done, at this point. However, the translated pixels are not guaranteed to lie in an exact pixel position of the future (warped) image grid (see fig.9.40), (resulting in potential holes in the image) but rather in between 2 or more of them. So, we apply a bilinear pixel interpolation to every one of its channels:

$$I'_{c,t}{}^{(W)}(\mathbf{p}) = \sum_{\mathbf{q}} K(\mathbf{q}, \mathbf{p} + \delta\mathbf{p}) I_{c,t}^{(W)}(\mathbf{q}), \quad (9.27)$$

in every channel $c=1, \dots, C$ and every pixel $\mathbf{p} = (x, y)$, $x=1, \dots, H, y=1, \dots, W$, where $\delta\mathbf{p} = \mathbf{p} + \mathbf{F}(I_{t-1}, I_t)$ and K is a bilinear convolution kernel. **Backward Warping:**

Backward Optical Flow-based warping repeats the same procedure in the opposite direction to estimate the prior image sample: $I_{t-1}^{(W)} = \mathcal{W}(I_t, F(I_t, I_{t-1})) = I_t(\mathbf{p} + \mathbf{F}(I_t, I_{t-1}))$. If needed, the Bilinear Interpolation of fig.9.40 is employed, here, as well.

Spatial Transformer Networks

Spatial Transformer Networks (STNs) are composed of **Localisation Net**, a **Grid Generator** and, then, a **Sampler**. These components allow the network to actively spatially transform feature maps conditioned on themselves without explicit supervision. The authors of [53] show that this module is able to perform translation, scaling, rotation and other general warping transformations. In [53], they applied this module to a standard CNN network for classification, making it invariant to a set of spatial transformations.

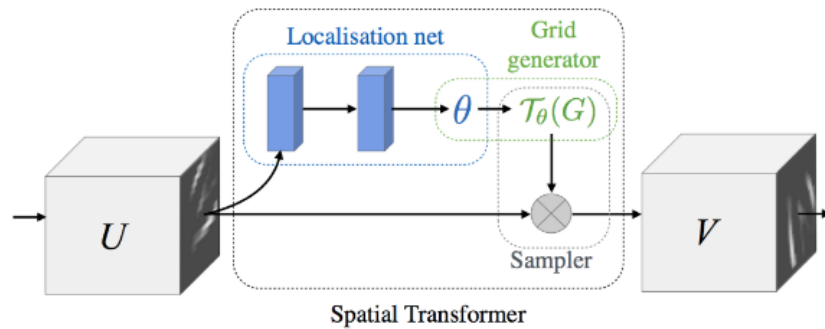


Figure 9.41: The diagram of a Spatial Transformer module[53].

The work of Spatial Transformer Networks is to transform the feature map into another vector space representation. There're 3 parts in STN: the Localisation Network, the Grid Generator and the Sampler. The Localisation Network is composed by fully-connected or convolution layers will generate the transformation parameters. We notate the input feature map as U , the width as W , the height as H , the channels as C , the outputs as θ and the parameters of transformation T_θ . This transformation can be learnt as an affine transform. For example, in the 2D affine case that transformation contains scaling and translation as below:

$$A_\theta = \begin{bmatrix} s & 0 & t_x \\ 0 & s & t_y \end{bmatrix} \quad (9.28)$$

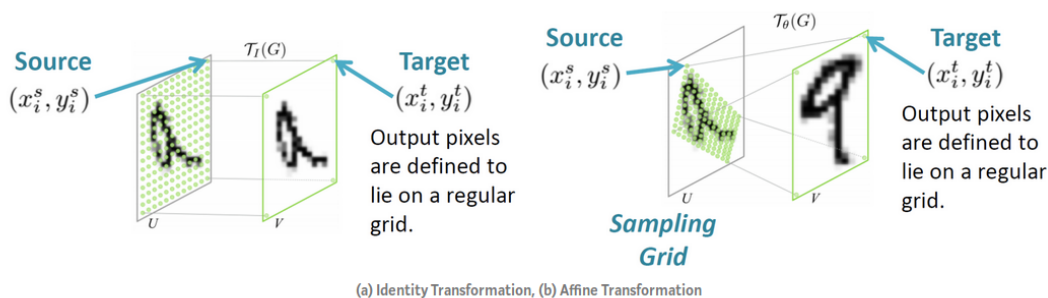


Figure 9.42: An example of the effect a STN module has on an example image of the MNIST dataset. The image source is quantized into a discrete grid and then transformed by an Affine 2D Transformation to a target image that lies on a regular grid.[53]

The second part is the Grid Generator. Suppose we have a regular grid G , this G is a set of points with source coordinates $(x_i^s, y_i^s)^T$, which acts as input. Then we apply transformation T_θ on G , i.e. $T_\theta(G)$. After $T_\theta(G)$, a set of points with destination coordinates $(x_i^t, y_i^t)^T$ is outputted. These points have been altered based on the transformation parameters. It can be Translation, Scale, Rotation or More Generic Warping depending on the definition of θ . After we get the parameters of the Affine

transformation A_θ , we can compute the opposite coordinate. Notice that the input of the transformation function is the coordinate of target feature map! That seems strange at first as what we know are the source coordinates instead of the target ones.

$$\begin{bmatrix} x_i^s \\ y_i^s \end{bmatrix} = \mathcal{T}_\theta(G_i) = A_\theta \begin{bmatrix} x_i^t \\ y_i^t \\ 1 \end{bmatrix} = \begin{bmatrix} \theta_1 & \theta_2 & \theta_3 \\ \theta_4 & \theta_5 & \theta_6 \end{bmatrix} \begin{bmatrix} x_i^t \\ y_i^t \\ 1 \end{bmatrix}, \quad (9.29)$$

where the superindex s notates the *source* and t one notates the *target* coordinates.

We want to compute where is the original coordinate for all target pixels. As a result, the task of the Grid Generator is only computing the coordinate in source feature map for each target pixel.

Discussing the Sampler component, based on the new set of coordinates $(x_i^t, y_i^t)^T$, we generate a transformed output feature map V. This V is translated, scaled, rotated, warped, projective transformed or affined. It is noted that STN can be applied to not only input image, but also to intermediate feature maps. If we need to sample a transformed grid, we need to select an appropriate kernel:

The Sampler’s kernel general form is:

$$V_i^c = \sum_{n=1}^H \sum_{m=1}^W U_{n,m}^c k(x_i^s - m; \Phi_x) k(y_i^s - n; \Phi_y), \quad (9.30)$$

$$\forall i \in [1, \dots, H' \cdot W'], c = [1, \dots, C].$$

The bilinear special form of the Sampler is:

$$V_i^c = \sum_{n=1}^H \sum_{m=1}^W U_{n,m}^c [1 - |x_i^s - m|]_+ [1 - |y_i^s - n|]_+ \quad (9.31)$$

This special form is (sub-)differentiable and, thus, convenient for Backpropagation:

$$\begin{aligned} \frac{\partial V_i^c}{\partial U_{n,m}^c} &= \sum_n^H \sum_m^W [1 - |x_i^s - m|]_+ [1 - |y_i^s - n|]_+ \\ \frac{\partial V_i^c}{\partial x_i^s} &= \sum_n^H \sum_m^W U_{n,m}^c [1 - |y_i^s - n|]_+ \begin{cases} 0, & |m - x_i^s| \geq 1 \\ 1, & m \geq x_i^s \\ -1, & m < x_i^s \end{cases} \end{aligned} \quad (9.32)$$

We note that, in this case, we will use the Forward AF-based (fig.9.27) convolutional features as our warping guidance. So, we will skip the localization component of the STN, as the affine transformation is provided by the Optical Flow convolutional features themselves.

If we carefully examine the potential of this approach to succeed we can articulate arguments both for and against its validity, before running the corresponding experiment:

- **Argument in favour of this approach:** Obviously, it is highly probable that warping the “Predicted” features towards the “Observed” ones, under the guidance of the Forward AF feature stream, would restrain the pose drift at every timestep and keep the feedback-generated 2D bounding boxes between tolerance boundaries. In this case, the tracker will be free to compensate for the pose drift at every iteration and, thus, rarely lose track of the object’s 3D trajectory.
- **Arguments against this approach:** On the other, one may find some really compelling arguments that counter the effectiveness of the architecture of fig.9.39.
 1. The first one is that warping the RGB-D-based convolutional features using the feature stream provided by the AF cue inserts a modeling mismatch error, as the AF is a by-product of comparing only the two RGB frames and ignores the depth information.
 2. The second one would be one of optimization nature. Training a STN module, and especially its differentiable sampling component is fairly unstable and frequently stacks the corresponding

weights to local optima, from which the CNN hardly recovers. So, the idea of feature warping may be a good one, but the component responsible for executing it may be shown to be an incompatible one.

By examining the effect of training the Network of fig.9.39, we will reveal which of the aforementioned two arguments does prevail.

Architecture	Translational Error (mm)	Rotational Error (o)
Approach of Chapter 6 (20k training pairs)	11.63 ± 8.79	8.31 ± 6.76
Approach of Chapter 6 with AF-based Feature Warping of the “Predicted” stream (fig.9.39) (20k training pairs)	13.24 ± 8.58	9.31 ± 6.80

Table 9.13: The Translational and Rotation Error metric comparison between our approach of Chapter 6 and its variation where Spatial Transformer AF-based feature warping is used for early fusion between the RGB-D and Flow modalities.

Architecture	Normalised Pose Error (mean \pm std)
Approach of Chapter 6 (20k training pairs) (Random Weight Initialization)	0.82 ± 0.64
Approach of Chapter 6 with AF-based feature warping of the “Predicted” stream (20k training pairs)	0.92 ± 0.64

Table 9.14: The Normalized Pose Error metric comparison between our approach of Chapter 6 and its variation where Spatial Transformer AF-based feature warping is used for early fusion between the RGB-D and Flow modalities.

We see in Tables 9.13,9.14, indeed the arguments against this improvement attempt seem to be stronger than the ones for it. Albeit the use of residual connections throughout the whole AF-based stream, incorporating it in the overall design does not compensate for the pose errors effectively enough. On the contrary, the tracker presents increased pose errors, in average, than in Chapter 8. We have no way of knowing if that happens due to improper training of the AF-based component or due to the fact that the depth-aware features are not taken into consideration the way they should. Here, we note that the AF-based warping has not any particular weakening effect on the tracker’s inference, either, something shown by the fact that the mean error standard deviation remains fairly the same. The CNN seems to perceive it as added noise, at the fusion point, and rely only on the two RGB-D-based convolutional streams.

9.4.3 Hierarchical Augmented Reality Flow Attentions

The third, and final, sophisticated way we use to incorporate the motion information directly at the feature extraction process of the tracker is a Multi-layer Hierarchical Spatial Attention fusion scheme. Following, we briefly explain the core of our idea: AF is the emblazonment of pose difference between the two streams, projected at the image plane. When the two feature streams get concatenated, we apply a different Spatial Attention Weight map at each of the three sequential Fire module outputs, in order to enhance their focus, at a different scale at a time. Those maps are produced by an independent AF-based convolutional stream, as shown in fig.7.27. However, the same problem about the real-time capabilities of the tracker, arises here as well. So, if this approach seems promising enough (i.e. surpasses the RGB-D based approach of Chapter 6, even for a small amount), we will add a context distillation term in our design. That means, that we will only use the AFs of the dataset during training: at first, we will train a Teacher Network with the structure of fig.7.27, we will freeze its weights, and then, we will train a similar Student Network with the same loss, an RGB difference instead of the AF cue and an additional distillation loss term, added to the overall loss function and weighted in the same way as the others.

Again, at every training batch, we switch on and off the Flow Attention stream according to the outcome of a Bernoulli probability with $p=0.5$, to avoid overfitting.

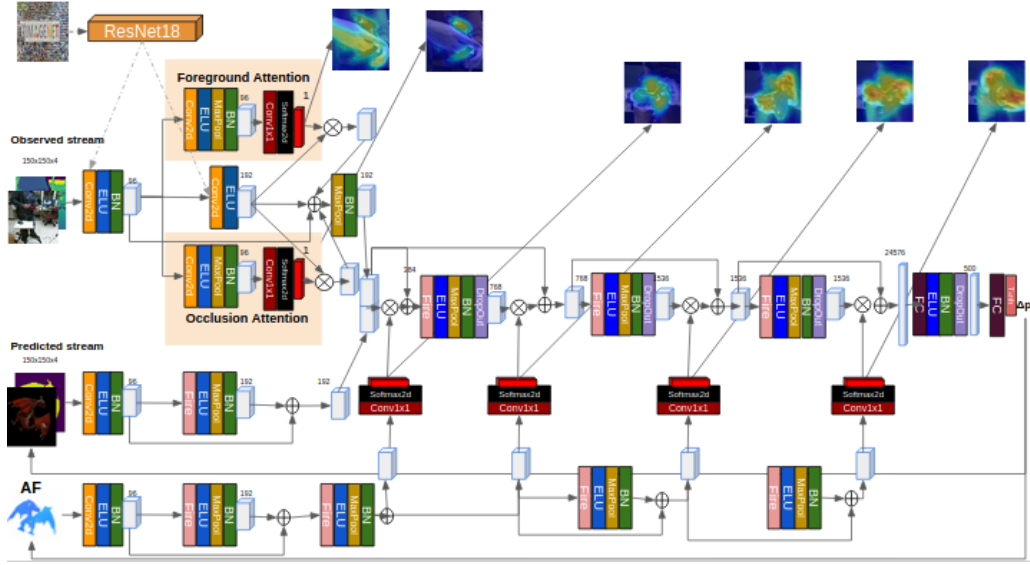


Figure 9.43: Overview of the Hierarchical Multi-Layer AF-based Spatial Attention fusion modification the approach of Chapter 6.

Architecture	Translational Error (mm)	Rotational Error (o)
Approach of Chapter 6 (20k training pairs)	11.63 ± 8.79	8.31 ± 6.76
Approach of Chapter 6 with a Multi-Layer Hierarchical AF-based Spatial Attention fusion (20k training samples)	11.55 ± 8.26	8.23 ± 7.02

Table 9.15: 3D Translational and Rotational Error metric comparison between the baseline architecture of Garon et al. [83], our improved approach of Chapter 6 and its modified version with an incorporated Multi-Layer Hierarchical AF-based Spatial Attention fusion scheme.

Architecture	Normalised Pose Error (mean \pm std)
Approach of Chapter 6 (20k training pairs)	0.82 ± 0.64
Approach of Chapter 6 with a Multi-Layer Hierarchical AF-based Spatial Attention fusion (20k training pairs)	0.81 ± 0.63

Table 9.16: Normalized Pose Error metric comparison between the baseline architecture of Garon et al. [83], our improved approach of Chapter 6 and its modified version with an incorporated Multi-Layer Hierarchical AF-based Spatial Attention fusion scheme.

We see in Tables 9.15,9.16, that enhancing the RGB-D fused stream via a hierarchical Optical-Flow based one will result in a slightly reduced pose tracking error. At this point, we need to remind that online Optical Flow estimation is extremely demanding from a computational standpoint, resulting in losing the real-time performance of our tracker by far (remember that our approach was estimating the full pose in ms and a single FlowNet2 pass (one of the best approaches in balancing the tradeoff between time complexity and Optical Flow frame quality [52]) is close to 200ms by itself). So, this Multi-layer Spatial Attention does not seem to be worthy of the extra computational burden it causes to the tracker’s inference.

Distillation of Hierarchical Augmented Reality Flow Attentions

However, the slight improvement we observe on the tracker’s performance gives us the spark to try to fit the Hierarchical Multi-layer AF Attention stream within the real-time speed boundaries. In order to achieve so, we will next try approximate the AF input cue with a simple RGB “Observed”, “Predicted” frame difference and distil the motion information. It is this RGB difference that will be used during the

tracker’s inference. We will employ both Network variations’ (fig.7.27 and fig.9.43) forward passes, but we will keep only the “Distilled” ones’ backward pass unfrozen. The make an effort to distil the motion context in a twofold way:

1. By adding an extra Distillation loss that would try to simulate the AF-based attention, at every level, with an RGB difference motivated corresponding one. We do not only want to confine ourselves in requiring a pixel-level similarity, but also, a semantic one. So, we inspire from the work of Liu et al. [76] and instead of employing a Mean Squared Error loss between the output Spatial Attention weight maps of every layer, we exploit their probability-like nature and use a Kullback-Leibler Divergence (see Section 3.3.17):

$$\begin{aligned}
L_{KL}^{Distil}(Att_{Teacher}, \hat{Att}_{Student}) = & e^{(-w_1)} \cdot D_{KL}(Att_{Teacher}^{(1)}, \hat{Att}_{Student}^{(1)}) + \\
& e^{(-w_2)} \cdot D_{KL}(Att_{Teacher}^{(2)}, \hat{Att}_{Student}^{(2)}) + \\
& e^{(-w_3)} \cdot D_{KL}(Att_{Teacher}^{(3)}, \hat{Att}_{Student}^{(3)}) + \\
& e^{(-w_4)} \cdot D_{KL}(Att_{Teacher}^{(4)}, \hat{Att}_{Student}^{(4)}) + \\
& w_1 + w_2 + w_3 + w_4
\end{aligned} \tag{9.33}$$

2. During the training phase, we have an extensive freedom, so we give the distilled stream a head-start by initializing its weights with the weights of the pretrained AF-based Multi-layer Attention stream. Besides that, we also, transfer the weights of the RGB-D stream.

Our overall learnably multitask loss function now is of the form:

$$\begin{aligned}
Loss' = & e^{(-s_1)} \cdot L_{Track} + e^{(-s_2)} \cdot L_{Uoccl} + e^{(-s_3)} \cdot L_{Foregr} + e^{(-s_4)} \cdot L_{KL}^{(Distil)} + \\
& s_1 + s_2 + s_3 + s_4
\end{aligned} \tag{9.34}$$

Of course, the data augmentation phase remains the same as above and we still drop out both the RGB-difference and the AF-based Attention streams according to the outcome of a Bernoulli probability with p=0.5, for extra regularization. In order to keep the Network’s speed within the real-time spectrum, imitating the Optical Flow information durind inference is not enough.

Before proceeding with the results in the Tables below, we state an intuitive factor that would contribute in this approach’s potential success (i.e. surpassing the method of Chapter 6, while remaining in the real-time speed spectrum) and one, contradictive, factor that would lead us to think otherwise:

- **Argument against the approach:**

Since incorporating the Optical Flow information in this Multi-Layer way return a marginal tracking improvement, in terms of Normalized Pose Error, we expect that the added error an imperfect distillation will cause, will be the reason for falling, again, under the performance of the solely RGB-D-based variation.

- **Argument for the approach:**

On the other hand, the RGB difference information source will have a greater degree of augmentation, giving more realistic data to the tracker to process and a, potentially, extra generalization capability to the real-world testing dataset.

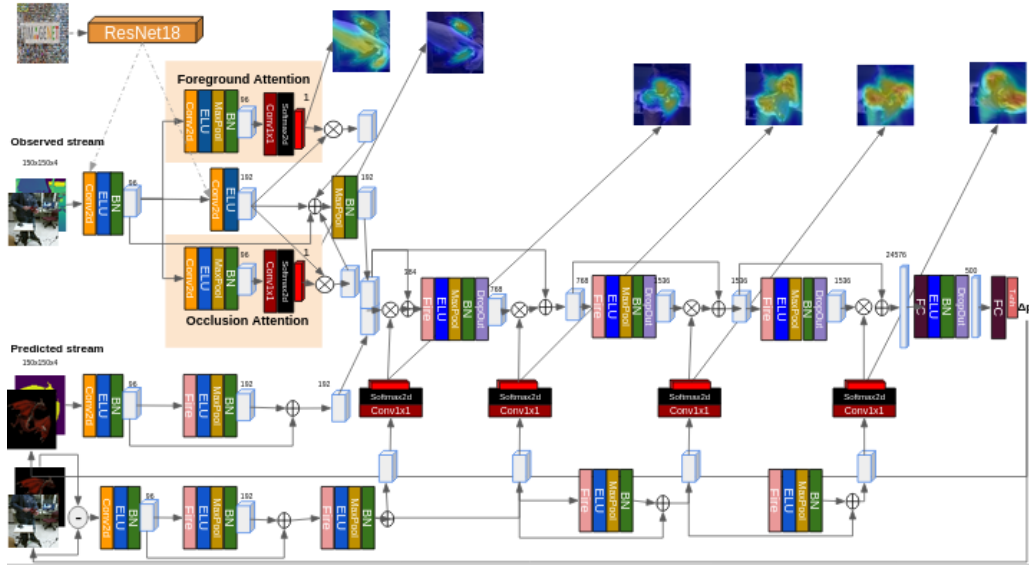


Figure 9.44: Overview of our approach of Chapter 6 with a Multi-Layer Hierarchical Distilled AF-based Attention to every layer concatenated RGB-D-based feature map.

Architecture	Translational Error (mm)	Rotational Error (o)
Approach of Chapter 6 (20k training pairs)	11.63 ± 8.79	8.31 ± 6.76
Approach of Chapter 6 with a Distilled Multi-Layer Hierarchical AF-based Spatial Attention fusion (20k training samples)	14.81 ± 9.96	11.46 ± 8.04

Table 9.17: 3D Translational and Rotational Error metric comparison between the baseline architecture of Garon et al. [83], our improved approach of Chapter 6 and its modified version with an incorporated Distilled Multi-Layer Hierarchical AF-based Spatial Attention fusion scheme.

Architecture	Normalised Pose Error (mean \pm std)
Approach of Chapter 6 (20k training pairs)	0.82 ± 0.64
Approach of Chapter 6 with a Distilled Multi-Layer Hierarchical AF-based Spatial Attention fusion (20k training pairs)	1.08 ± 0.75

Table 9.18: Normalized Pose Error metric comparison between the baseline architecture of Garon et al. [83], our improved approach of Chapter 6 and its modified version with an incorporated Distilled Multi-Layer Hierarchical AF-based Spatial Attention fusion scheme.

Indeed, as we can see in Tables 9.17 and 9.18, we find out that tracking accuracy drops significantly in comparison to the approach of Chapter 6, which has the extra advantage of simplicity.

As a matter of fact, our experiments show that an explicit modeling of the pose error feedback in the form of Optical Flow dense 2D images is not of any substantial value to the tracker’s understanding of the pose. We appoint two reasons for explaining this observation:

1. The first reason is that, for computational feasibility reasons, the Optical Flows are not created on the fly with augmented inputs, but they are standard in the overall training dataset. Besides that, the augmentation capabilities of the Optical Flow in the literature are rather limited and they are focused on the pose alteration of their RGB inputs (something standard for our problem). This has the effect that every two-stream CNN architecture or every other variation that incorporated the Optical Flow information source **is pretty much sentenced to overfit to it**.
2. The second reason is the inevitability of information distillation for a real-time tracking performance. Due to the high computational burden that producing accurate flow images cause, it is not

appropriate for time-critical applications. As a result, a need for imitating this kind of information, when it is successful in reducing the NN's error, arises and with it, a problem arises as well: the distillation process is highly improbable to be optimal, and as a result, an extra inference error (of the “model mismatch” type) is most likely to add up.

Consequentially, none of the two types of motion modeling has given any improvement to the performance reported by the approach of Chapter 6. However, we can neither exclude such a possibility in the future, as there is no mathematical foundation that would suggest so. Thus, we aim to experiment with more drastic and physically plausible augmentation schemes for the Optical Flow cue, motion projection sources of other kind (e.g. Optical Flow images extracted from Depth images difference of Range Flow (i.e. the 3D counter part of 2D Optical Flow)) or modular architectural modeling of temporal dynamics with the use of Temporal Attention schemes such as the popular “Transformer” [144].

Chapter 10

Future Work

Only the unknown frightens men. But
once a man has faced the unknown,
that terror becomes the known.

Antoine de Saint-Exupéry, 'Terre des
Hommes'

10.1 Model-agnostic Single Object Tracking

Following the work of Garon et al. [83], it arises as critical for our tracking algorithm to generalize over a wide variety of objects (i.e. the **multiple known object case**) or even to perform at the same quality when dealing with **previously unseen ones**. In the approach of Chapter 6, this requirements' fulfillment is not straightforward for two reasons:

1. Both the rotational anisotropy term (Λ) and the symmetry matrix G , that are incorporated in the rotational loss function, as well as the discrete rotational ambiguity handling algorithm of Chapter 6, are tailor-made for specific, known objects and it is not certain that they will have the same, beneficial, influence to the pose tracking objects the Network has never seen before, and thus, has not utilized, this kind of particular shape information during training.
2. We have no guarantee, neither theoretical nor experimental, that the parallel Attention modules will generalize their performance in the object-agnostic case or if the keypoints of interest they learn to focus to are of the same value to the main task, i.e. pose tracking, for objects the Network has never seen before.

10.2 Bridging the 2D (source) - 3D (task) modeling mismatch

So far, we have only utilized 2D convolutional modules to extract features that will have discriminative 3D pose capabilities. In fact, the pose comparison, based on these features, is mainly done in the two Fully Connected layers of fig. 6.28. This, at a first glance, seems like a modeling mismatch as we speculate that conclusions about entities of the 3D space (such as the object's pose) must reside from 3D scene representations. This profound contradiction is of great importance, specifically to the pose tracking problem, as the 2D convolutional filters of our feature extractor are only translationally equivariant (and, specifically, 2D translationally equivariant) and not rotationally equivariant, a property that we suspect that would play a crucial role in potentially mapping 3D pose differences in a manifold-like subspace and imposing some order between different pose configurations.

In that spirit, we aim to modify our architectural design in the future, inspired by the approaches of Kehl et al. [57], Brachmann et al. [8],[9], Park et al. [98], and the iPose configuration [48]. In iPose, for example, the authors regress the Object Coordinates of an object of interest (from a single view) and then estimate its pose w.r.t the camera reference frame (C) using a P'n'P + RANSAC algorithmic

combination (see Section 4.1). On the contrary, we wish to transform the convolutional part of fig 6.28 into two U-Net-like [113] streams that regress the object’s 3D shape at the each of the “Observed” and “Predicted” poses, transform them from the object-oriented (O) to the camera-oriented (C) framework, with the use of the SVD and Umeyama algorithms, and then compare them using a **Tensor Field Network** layout, i.e. a CNN-based structure that processes 3D point inputs via SE(3)-equivariant convolutions.

Next, we briefly introduce the theoretical foundations of our the two modular components we aim to experiment with in the future: the concept of **(Normalized) Object Coordinates**, the **Umeyama Algorithm** that transforms reconstructed 3D object Coordinates from (O) to (C) and, finally, the **Tensor Field Networks, equipped with SE(3)-equivariant convolutions**.

10.2.1 (Normalized) Object Coordinate (NOCs) Regression

The NOCs are a reconstructed representation of the object’s 3D point cloud that allows each vertex of its shape to be represented as a tuple (x, y, z) within a unit cube i.e., $x, y, z \in [0, 1]$. Besides being interpreted as a shape reconstruction, they can be seen as dense pixel correspondences like in Pix2Pose [98]. These Object Coordinates have been used many times in the past as the information source of the P’n’P algorithm.

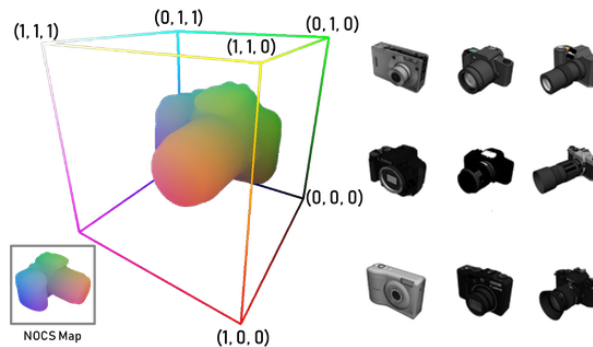


Figure 10.1: The **(Normalized) Object Coordinate Space (NOCS)** is a 3D space contained within a unit cube. For a given object category, canonically oriented instances are normalized to lie within the NOCS. Each (x, y, z) position in the NOCS is visualized as an RGB color tuple. [149]

10.2.2 Umeyama Algorithm

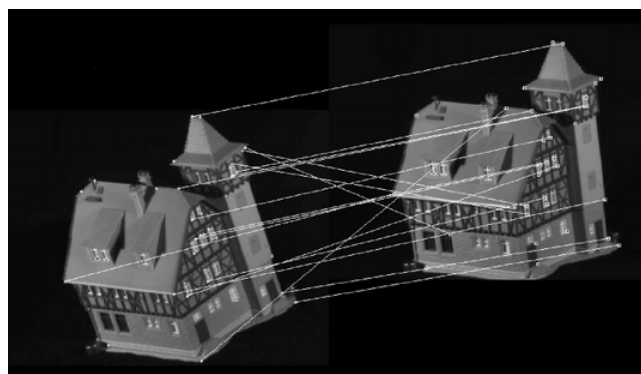


Figure 10.2: An iteration of the Umeyama algorithm with the use of 3D points of interest.[140]

Let’s consider two sets of paired points, P and Q. Each set of points can be represented as an $N \times 3$ matrix. The first row is the coordinates of the first point, the second row is the coordinates of the second

point, the Nth row is the coordinates of the Nth point: $\begin{bmatrix} x_1 & y_1 & z_1 \\ \dots & & \\ x_N & y_N & z_N \end{bmatrix}$. The algorithm that calculates the relative rotation of P into Q is named the **Umeyama Algorithm** [141].

The algorithm works in three steps: a translation, the computation of a covariance matrix, and the computation of the optimal rotation matrix.

Translation:

Both sets of coordinates must be translated first, so that their centroid coincides with the origin of the coordinate system. This is done by subtracting from the point coordinates the coordinates of the respective centroid.

Computation of the covariance matrix

The second step consists of calculating a cross-covariance matrix H. In matrix notation:

$$H = P^T \cdot Q \tag{10.1}$$

or, using summation notation:

$$H = \sum_{k=1}^N P_{ki} \cdot Q_{kj} \tag{10.2}$$

Rotation Matrix

It is possible to calculate the optimal rotation R based on the matrix formula:

$$R = (H^T \cdot H)^{\frac{1}{2}} H^{-1} \tag{10.3}$$

but implementing a numerical solution to this formula becomes complicated when all special cases are accounted for (for example, the case of H not having an inverse).

If singular value decomposition (SVD) routines are available, the optimal rotation, R, can be calculated using the following simple algorithm.

First, calculate the SVD of the covariance matrix H.

$$H = U \Sigma V^T \tag{10.4}$$

Next, decide whether we need to correct our rotation matrix to ensure a right-handed coordinate system

$$d = \det(VU^T) \tag{10.5}$$

Finally, calculate our optimal rotation matrix, R, as:

$$R = V \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & d \end{bmatrix} U^T \tag{10.6}$$

The optimal rotation matrix can also be expressed in terms of quaternions. This alternative description was recently used in the development of a rigorous method for removing rigid-body motions from molecular dynamics trajectories of flexible molecules.

The algorithm was described for points in a three-dimensional space. The generalization to D dimensions is immediate.

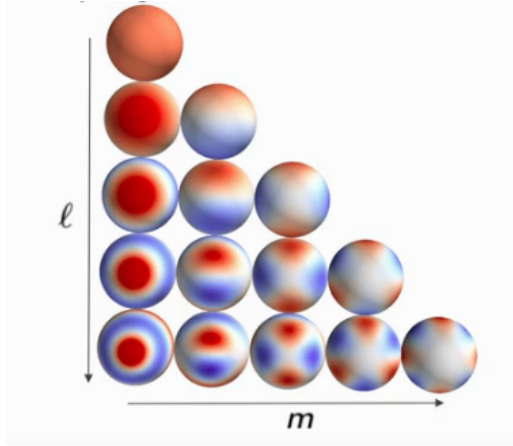


Figure 10.4: A visualization of the 3D Spherical Harmonic component of the SE(3)-equivariant filters of Tensor Field Networks.[132].

10.2.3 Tensor Field Networks

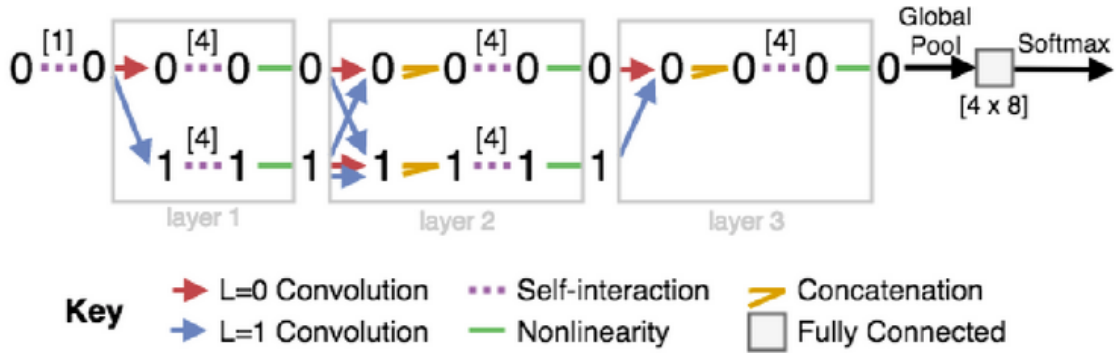


Figure 10.3: An example of a **Tensor Field Network** presented in [132].

CNNs have been constructed in a way that their filters are Translationally Equivariant (either 2D or 3D) and function on Feature maps. However, tasks like 3D Pose Estimation require 3D Rotational Equivariance as well. **Tensor Field Networks (TFNs)** [132] are a recent successful example of providing such a feature. They function on 3D Point Clouds instead of feature maps and their filters are locally equivariant to SE(3)-transforms of their corresponding inputs. Their existence removes the need for intensive 3D translational and/or 3D rotational data augmentation in order to produce the corresponding features of random poses. Those local features are now extracted by a predefined set of filters with global parameters.

We now, briefly, explain how this gets accomplished:

TFNs model the Point Cloud inputs as high-order geometric tensors and are comprised of Point Convolutions with kernels that are products of 3D Spherical Harmonics, that depend on the rotation direction, multiplied by learnable radial functions. The former, $Y_m^{(l)}(\hat{\mathbf{r}})$, $m \in [-l, l]$ (where l is the layer number and m : the rotation order) create an orthonormal basis for functions that lies on a sphere and provide SO(3) equivariance w.r.t. the input of the corresponding layer l . The latter, $R(r) = R(r_a - r_b)$ depend only on the radial distance between 2 corresponding 3D points and provide 3D translational equivariance w.r.t. the input of the corresponding layer l [132].

As shown in an example in [132], those Point Convolutions give the opportunity to correctly classify 3D object models that have seen in only one pose configuration, when tested in any configuration possible.

If we also think deeply about our problem, the various poses that we produce during the dataset

generation procedure (before projected in 2D images) are nothing more than $SE(3)$ data augmentation of one object pose. As a result, we hypothesize that a small training group of imperfect 3D Object Coordinate reconstructions from RGB-D frame pairs may give the power to an extension of our architecture performed by a multi-layer TFN component to correctly estimate 3D object poses, after having trained only on a few samples or in a One-Shot fashion, because of the $SE(3)$ -equivariance property of TFN’s kernels.

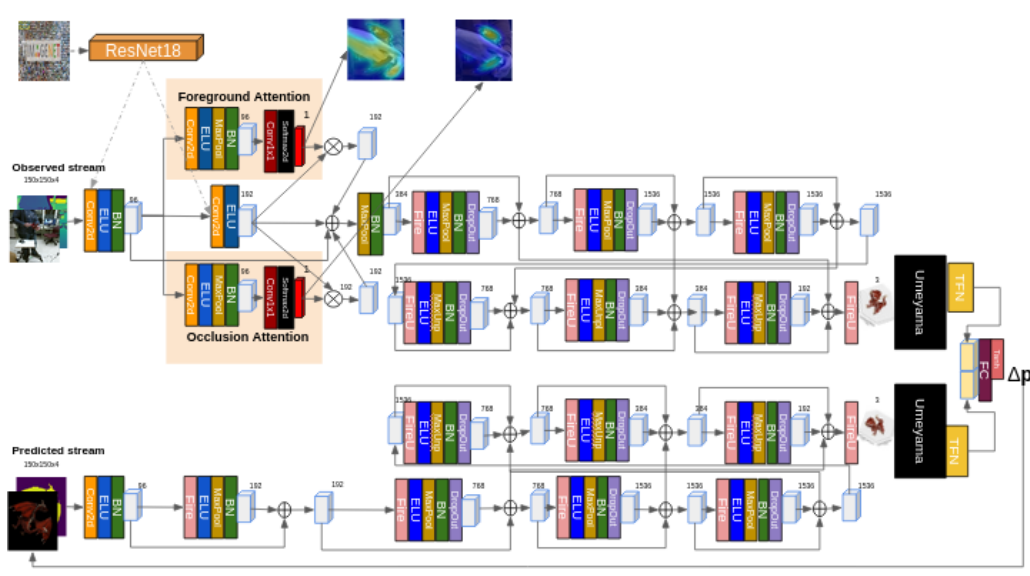


Figure 10.5: Overview of our future exploration of a modified version of the approach of Chapter 6 where we will reconstruct 3D Object Coordinates, we will process them with a pair of TFNs and we will, then, fuse them at a late stage.

10.3 Depth Information Distillation

If we bear in our minds the list of pose tracking challenges we created in this Thesis’ introduction (Chapter 2), we will remember that a severe one that we have not addressed so far is our desire to track the object’s 6D pose using only the RGB information cue, as not all devices carry Kinect sensors (i.e. mobile devices) on them. However, the preliminary study we conducted in Chapter 7 states that the Depth modality as the most important between the two. So, we need to think of an architectural scheme that distills the context of the Depth modality only from the pair of Depth maps available during training and equips the RGB-based stream with a scale-aware mechanism that variates its focus to different spatial dimensions of the image/feature map. In that context, we, next, introduce a new module called Pixel Adaptive Convolution.

Pixel-Adaptive Convolutions

As we have already discussed classical 2D convolutional filters (Section 3.3.5) have the property of spatial sharing their learned weights across all spatial dimensions of the input. This property has been presented as an advantage, as it reduces the computational complexity and increases the parallelization of the Network. However, it is **content agnostic** as it does not allow convolutional kernels to vary w.r.t. different spatial patterns in the image. Classical CNNs try to solve this problem by increasing the filter size, thus, practically allowing each filter to recognize one image pattern. However, this does not allow the kernels to be guided by any external auxiliary supervision signal.

That is why Su et al. [123] developed **Pixel Adaptive Convolution (PAC) operations**. These new operations are a simple modification of standard convolutions where a second information stream produces a guiding spatial varying kernel K that is multiplied by the main convolutional stream’s kernel weights in order to allow for spatial variability (see fig.10.6).

Mathematically, PAC modifies the spatially invariant convolution of Section 3.3.5 with a spatially varying kernel $K \in R^{c_{in} \times c_{out} \times k_W \times k_H}$ that depends on pixel features \mathbf{f} :

$$f'_i = \sum_{j \in \Omega(i)} K(\mathbf{f}_i, \mathbf{f}_j) W(\mathbf{p}_j - \mathbf{p}_i) \mathbf{f}_j + \mathbf{b} \quad (10.7)$$

where $\mathbf{p}_i = (x_i, y_i)^T$ are pixel coordinates, K is a kernel function that has a fixed parametric form such as Gaussian: $K(\mathbf{f}_i, \mathbf{f}_j) = e^{-\frac{1}{2}(\mathbf{f}_i - \mathbf{f}_j)^T(\mathbf{f}_i - \mathbf{f}_j)}$, $\Omega(\cdot)$ defines an $k_W \times k_H$ convolution window and \mathbf{b} denotes the bias vector. We call these pixel features \mathbf{f} as “adapting features” and the kernel K as “adapting kernel”. The adapting features \mathbf{f} can be either hand-specified such as position and color features $\mathbf{f} = (x, y, r, g, b)$ or can be deep features that are learned end-to-end. Of course, classic convolution is the special case with $K=1$. Other common kernel choices are bilateral filtering and average pooling.

Despite the simple formulation, PAC can be seen as a generalization of several popular filtering techniques such as standard convolution, bilateral filtering and several other pooling operations that are in widespread use in computer vision and computer graphics, with its main known application fields being image upsampling, “hot swapping” of pretrained filters. For example, in their research paper, they upsample low-resolution depth or optical flow signals using high-resolution an RGB image as guidance, achieving to restore details missing even after 16 times of spatial upsampling. Because the filters are adaptive to the guidance image, details can be recovered even when completely missing in the low-resolution input.

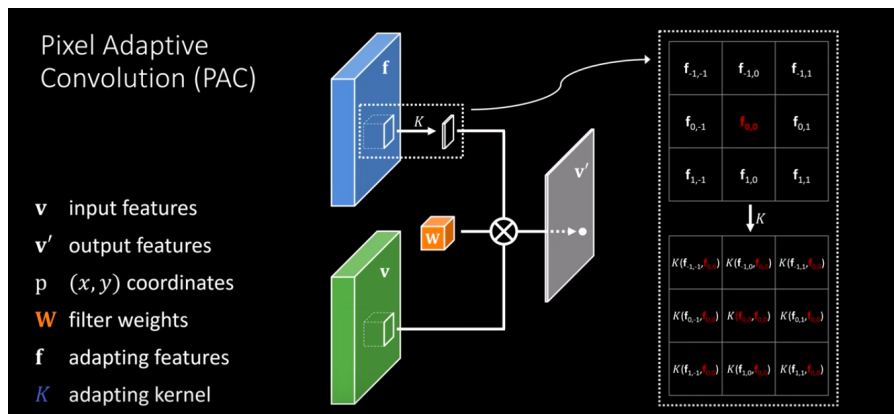


Figure 10.6: Pixel-Adaptive Convolution. PAC modifies a standard convolution on an input by modifying the spatially invariant filter W with an adapting kernel K . The adapting kernel is constructed using either predefined or learned features \mathbf{f} . \otimes denotes element-wise multiplication of matrices followed by a summation. Only one output channel is shown for the illustration. [123]

Inspired by their approach, our goal is to use the opposite configuration: apply Pixel Adaptive Convolutions to RGB-based features guided by the features of the Depth modalities, in each of the “Predicted” and “Observed” frames. This way, we aim to restrict the use of Depth maps only in the training phase, where they are available, practically for free, and distill their conceptual meaning into the weighting procedure of the RGB-based features, in order to allow our tracker to function in devices with availability only of the RGB cue, without a significant drop in performance. Practically, what we want in order to achieve so, we need to store the running average of the convolutional weighting kernel’s K (following the Batch Normalization methodology) and store them for use in the inference phase of the tracker (see fig.10.7).

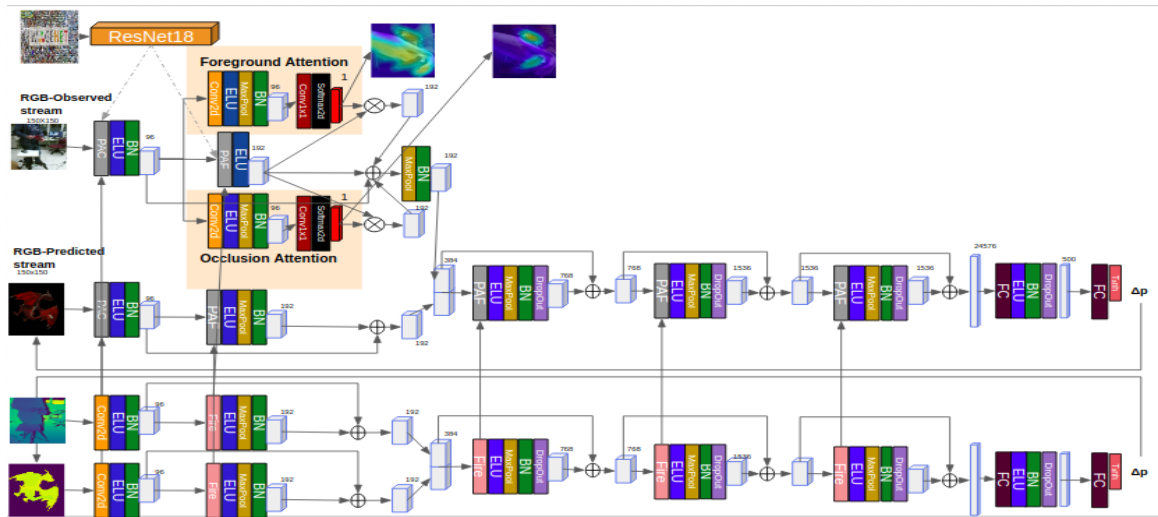


Figure 10.7: Overview of an exploration of a future variation of our approach of Chapter 6, which is based only on RGB inputs and the Depth information is distilled with the use of Pixel-Adaptive Convolutions [123] and its stream is available only during training.

10.4 Reinforcement Learning-based Multiple Object Tracking and Assembly Guidance

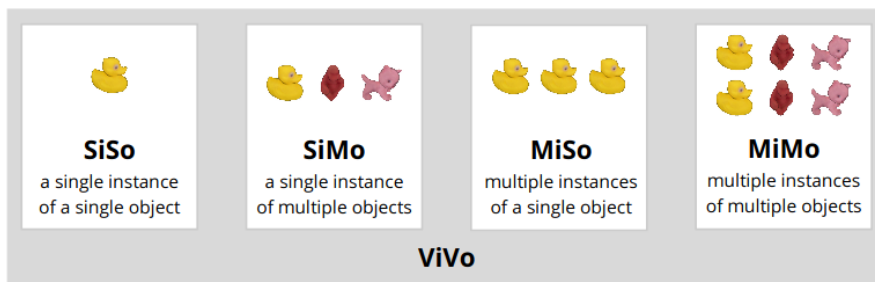


Figure 10.8: 6D localization of a Varying number of instances of a Varying number of objects in a single RGB-D image, the number of instances is known. A list of instances to localize provided with the image. [45]

One would, naively, think that generalizing our learnable tracker to the multi-object case is straightforward and just a matter of parallel computing resources’ availability. However, that is not the case, as it has been reported in various literature sources. That is because in cases of multiple objects, especially for those with similar appearance and/or shapes, the multiple trackers we employ will possibly either stick with only of the objects, neglecting the rest, intersect their different estimations when this is not the case or, even, mix up their targets from a timestep and onwards. A distinctive visual clue provided for this challenge is the fact that the “shoe” sequences of the testing dataset provided by Garon et al. [83] contains a user wearing only one shoe, instead of both, in order to avoid this tracker confusion. In a case where he would have worn both, the tracker would possibly switch between shoes between different frames or even estimate the pose of the shoe as the average between the two distinctive shoe poses.

Before, briefly, explaining our future action plan for this challenge, we refer to two previous important publications:

- At first, we, briefly, remind the reader of the work of Hadfield et al. [34], where a Rao-Blackwellized Particle Filter is employed for tracking multiple objects (in particular, building blocks) in order to aid a user assemble a more complex target structure. When an assembly connection is near its completion, an indicative score of how similar the observed pose is to the desired one is calculated.

Hadfield et al., [34], used this scoring to calculate the number of particles for the next iteration that we will assign to the next connection type. The scoring function is of the form:

$$score = \frac{1}{2} \left\{ \frac{1}{d(\mathbf{p}_1, \mathbf{p}_2)} + \frac{1}{d(\mathbf{p}'_1, \mathbf{p}'_2)} \right\}, \quad (10.8)$$

where \mathbf{p}_1 is the estimated pose parameter 6D vector (with quaternia parameterization of the rotational component), \mathbf{p}_2 the ground truth one and :

$$d(\mathbf{p}_1, \mathbf{p}_2) = \sqrt{\max\{K[(\frac{t_{1,x} - t_{2,x}}{m_{t,x}})^2 + \frac{t_{1,y} - t_{2,y}}{m_{t,y}})^2 + \frac{t_{1,z} - t_{2,z}}{m_{t,z}})^2 - 3], 0\} + 1} \quad (10.9)$$

where K is a sensitivity parameter and $m_{t,i}$, $i=x,y,z$ are the corresponding assembly tolerance at each axis. As K increases, this score metric will be lower when the two 3D translations diverge outside of the tolerance ranges. When the relative 3D position of the two objects is within these boundaries, $score=1$. As they diverge from them, the score metric will decrease and, finally, converge to 0.

There is the possibility of the two objects being really close, in terms of 3D position, but having a substantial rotational distance. For this reason, they convert the rotational pose components to their quaternia representation and use them to invert the relative poses between the two objects:

$$\begin{aligned} \mathbf{p}'_1 &= -\mathbf{q}_1^* \mathbf{p}_1 \\ \mathbf{p}'_2 &= -\mathbf{q}_2^* \mathbf{p}_2. \end{aligned} \quad (10.10)$$

Finally, they calculate the relative distance $d(\mathbf{p}'_1, \mathbf{p}'_2)$ between them, as well, and add its inverse to the score metric as a second term.

This distance metric is based solely on 3D position information, as their Recursive Bayesian Estimation framework is very accurate in estimating the object's 3D translation, but not so much when it comes to 3D rotations. The experiments we have conducted using our, improved, learnable tracker do not indicate such a pattern, so, according to Hadfield et al. [34], an extra rotational distance can be added to the overall metric:

$$d'(\mathbf{p}_1, \mathbf{p}_2) = d(\mathbf{p}_1, \mathbf{p}_2) + \arccos\left(\frac{\text{tr}(R_1^T \cdot R_2) - 1}{2}\right) \quad (10.11)$$

- We also inspire by the approaches of Ren et al. [111] and Milan et al. [87], where they address the multiple object tracking (and, especially, with identical appearance features) problem by formulating an overall energy function that penalizes multiple object estimates intersecting (by calculating their centers' distance).

Having thoroughly studied these past works, we aim to generalize our single object strategy as follows (let's discuss the case of similar objects that both Hadfield and Ren address): we employ multiple learnable trackers, pretrained on single, known objects, according to Chapter 6. Then, we need to generalize our training dataset to a scalable version, where it contains multiple copies of the same object and two hand occluders, which would occlude each object with a probability assigned to it. Finally, we would finetune the trackers in a Reinforcement Learning framework, where the overall inverse reward function would be consisted of an adding of the tracking losses of all the trackers available, along with an extra penalty term, like those Hadfield and Ren used in their respective pieces of work, which would penalize the proximity of the pose estimates of different objects, in order to avoid mixing them up either temporarily or permanently. These two types of penalties would be weighted via an adaptive scheme based on the relative pose score metric Hadfield et al.[34] introduced. In this scenario, the two pose vectors that we would use to compute their distance metric, would not be the desired-predicted pair, but the predicted ones of the multiple instances of the same object type.

10.5 Self-Supervised Object Pose 6D Tracking

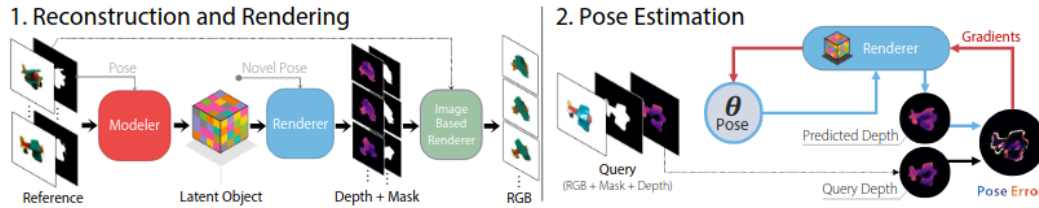


Figure 10.9: LatentFusion: An end-to-end reconstruction and rendering pipeline. This pipeline is used to perform pose estimation on unseen objects using simple gradient updates in a render-and-compare fashion[97].

Obtaining enough accurately labeled temporal 6D pose data for supervised training is a severe hinder in scaling deep learnable trackers to vaster databases. Even by generating synthetic training datasets to avoid this pitfall, one is required to balance the tradeoff between realism, abundance and cost-effectiveness. And, even in this case, the capabilities of such trackers are constrained in being aware of the object class or instance a-priori. On the other hand, gathering large amounts of unsupervised realistic video clips is much cheaper and avoids the Sim2Real gap. For this reason, another possible extension of this work would be to extend the Fire encoder to a sequential decoder which would neurally render the predicted 6D pose back to the image plane at the end of the overall pipeline. This would allow for end-to-end self-supervised approaches, where the tracking loss labels would be implicitly generated and direct supervision would be brought by the input images themselves. This is an approach that has recently started to gain popularity, as the research community strives to move away from the need of manually annotated data and it fits our architectural modifications, as well, since only the 3D shape a class of object instances is required a-priori. Then, the tracker would be free to process millions of unsupervised realistic video clips that would cover not only the pose space, but the dynamics space, as well (an effort that we have already classified as hard to be effectively engineered). Besides, recent scientific publications like the one of Harley et al. [36], has shown promise in tracking 3D position (in the context of Self-Driving Cars) just by learning spatial feature maps of unsupervised video streams without any sorts of supervised or self-supervised learning. And, although one can justifiably suspect that 3D rotations will be a much harder and demanding task to accomplish, this news can encourage this research direction. As far as complete frameworks are concerned, recently the “Self6D” (Wang et al.[148]) and the “Latent Fusion” (Park et al.[97]) (see fig.10.9) frameworks have utilized the recent advances in the Neural Rendering landscape [131] to build end-to-end self-supervised trackers and, although, their applicability does not yet exceed a narrow task domain due to a variety of necessary assumptions, the results one or two papers down the line will possibly be adequately close to general purpose unsupervised 6D tracking, just from untrimmed video clips.

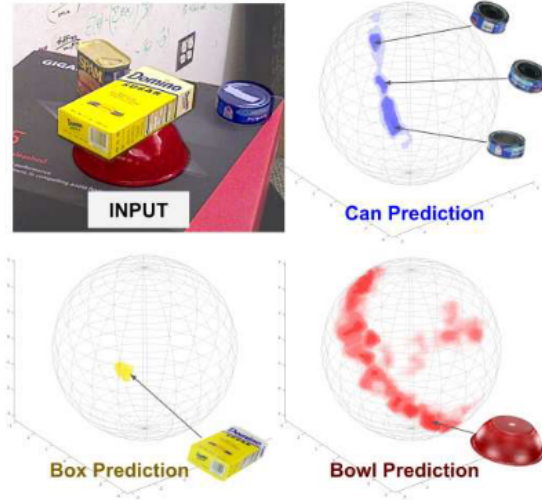


Figure 10.10: Multi-modal distributions estimated by a Learned Comparison Histogram approach of Okoron et al.(2020). These distributions are generated for the “tuna can”, “bowl” and “sugar box” object models using PoseCNN[162] featurizations. The estimator captures multiple possible viewpoint for the tuna can, while still placing most of the probability density on the correct mode. In the case of unambiguous poses, like the “sugar box”, it is still capable of producing tight uni-modal distributions.(Okorn et al.(2020)).

Another important factor that one could consider in order to properly extend this work is that not all poses in the relevant space are created equal. Different objects have different shapes with different properties and different appearance patterns that could facilitate or impede our effort to accurately predict the current and/or future pose configurations of a moving object. So far in this work, we dealt with anisotropic weighting of different pose components based on the geometry-based properties of the object shape representation. However, something similar could be done in the context of anisotropic pose weighting both in terms of proper balancing out of the training set with more dense or sparse pose distributions of particular layouts or during the training process itself. As a matter of fact, a self-supervised study and/or organization of the pose space would indisputably be proven beneficial in this task, as it has been proven in previous samples of literature [90],[62],[142] and in other tasks, like image classification. Noteworthy, in order to inspire the reader for the possibilities that such an approach would reveal to the algorithm designer, we interpose some characteristic examples: both Okonor et al. (2020), Wohlhart et al.[161], Bui et al.[13] and Deng et al.[21] disentangle similar from dissimilar poses(images) in the feature space and create an ordering inside the same pose(image) cluster, mostly in a contrastive manner. Under such a spirit, if we manage it we could even reduce our dependence on vast amounts of accurately labeled pose data and make do with weak supervision with the use of realistically generated samples.

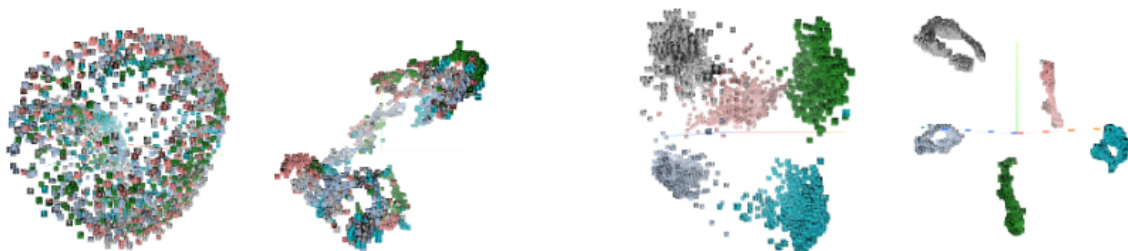


Figure 10.11 Obtained using direct pose regression[13] Figure 10.12 Obtained using our multi-task learning framework.[13]

Figure 10.13: In [13], Bui et al., by using a multi-task manifold learning framework, are able to improve feature descriptors learned for object pose estimation. Depicted here is the feature visualization using left: PCA and right: t-SNE for five objects of the LINEMOD dataset. Pose instances are disentangled in the pose space. If such an ordered input is given to a pose regressor, less samples will be needed for an accurate and consistent prediction not only of individual poses, but of trajectories, maybe.[13]

10.6 Pose Tracking of Deformable, Articulated and Compositional Object

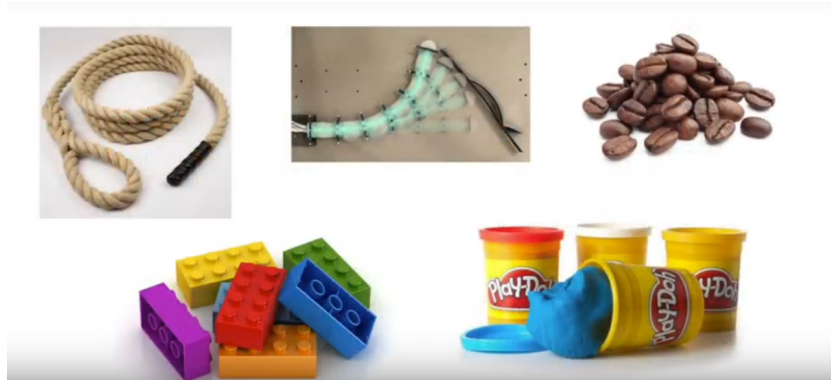


Figure 10.14: Indicative example cases of deformable, articulated and compositional objects whose pose needs to be tracked in future applications.[74]

We must not forget that the whole work presented in this thesis, as well as, all previous ongoing and future work descriptions above, have something in common: they take the rigidity of the object structure for granted. However, real life is frequent to repudiate this belief as there are multiple object classes that are of paramount importance for everyday human interaction and manipulation and do not fall under this category. And even if some of them, for example the “multiple lego blocks” case presented in fig.10.14 can be approached by multiple rigid object tracking methods, other instances like deformable, articulated or compositional objects are in need of novel, intermediate representation learning strategies that would integrate their specific characteristics. A significant amount of progress has been observed in this direction, during the past years. For example, in [74], Li et al. have used a Graph Neural Network[170] to project the visual representation of compositional objects into a higher dimensional space, where its dynamics can be described and controlled with Koopman Operators[84] and in [26], Florence et al. have constructed 3D deformable dense descriptors, via contrastive approaches, that greatly facilitate generalization properties of robotic manipulation. In such a scenario, it would be inevitable to involve ourselves into the object definition discussion and bind the boundaries of local (e.g. parts) vs global (e.g. whole object) structure. As a matter of fact, we are of the opinion that our existing framework is both flexible and modular enough to be generalized for this, holistic problem by, for example, alternating the purpose of attention maps to part-based extraction, with simultaneous generation of pertinent masks during creating the dataset, for supervision, and by decomposing the global object pose matrices (as well as the approximated inertia tensors) to a composition of the relevant properties of the poses of its individual parts. Besides, the unsupervised self-occlusion handling effect of our parallel attention maps hints the prospective effectiveness of such a modeling improvement attempt. If supervising uniform attention maps with binary masks increases the tracker’s accuracy by such a significant level, part-based guidance could improve its robustness to innate challenges by an unforeseen rate.

10.7 Ego-Motion Mitigation of a Moving Camera Sensor as a Pose Tracking prior

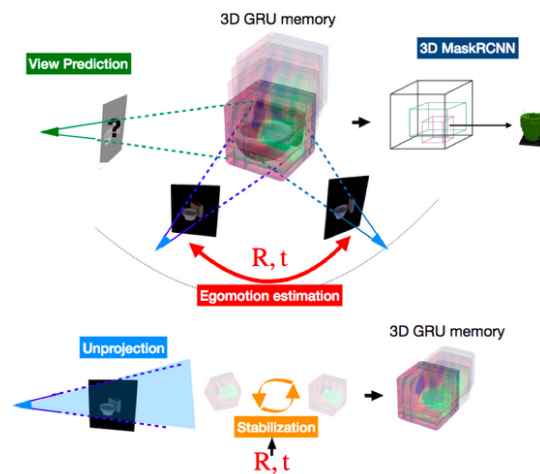


Figure 10.15: A visual demonstration of the work of Hsiao et al.[139] that recognizes, accurately estimates and counters sensor's ego-motion before proceeding to geometrical visual feature reasoning for the moving objects' of interest.

In every effort of studying the task of object pose tracking, in a research-oriented way, and in every benchmark dataset created so far, researchers have made the implicit assumption of a steady RGB(-D) camera sensor and the object being manipulated/standing in front of it. However, it is self-evident that the final consumer requirement is far more demanding in terms of degrees of freedom of movement. True intelligent robotic interaction projects would, for sure, be in need of deployment of embodied visual recognition algorithms [166],[146], with both the source-sensor and the object-target. Besides, this is something that is considered a sine qua non functionality of the human visual system. So, another future research direction would be to disentangle the two kinds of motion, in order to deduce this complex, coupled problem to the one we have discussed in this work, in some intermediate stage. This is far from trite, since visual coordinate systems refer to themselves instead of a global one. That is why the our object tracking was formulated with respect to the camera coordinate frame. The optical flow information could extricate us since it could be used to estimate the camera's ego-motion in order to compensate it, in an image preprocessing stage, before proceeding to the main task of object(s)'(s) pose tracking. Such an approach has been tried in the recent work of Hsiao et al.[139] with exceptional results, but it still remains just a part of an overall potential robotic visual interaction system.

10.8 Pose Tracking as a prior for Robotic Grasping



Figure 10.16: Overview of an example of using 6D Pose Estimation as a prior for successful household object grasping. Grasp hypotheses are generated, based on the Object Pose estimations, and they are verified or discarded at a later, cascaded level [137].

Robotic Grasping is a research subject closely tied to the one we examine in the Diploma Thesis: Object Pose Estimation/Tracking. In fact, it has seen a recent rise in interest and many vision-based Robotic Grasping frameworks (for example the work of Tremblay et al. [22]) use the Pose Estimation of static objects as a prior for optimizing both the grasping configuration and path planning. Their aim is to use the estimation of the object's pose as a "warm" start of their optimal grasping pose (e.g. if we are talking about a mug model, knowing its relative position and rotation w.r.t. the camera gives a strong hint to the robot to decide if its handle is the optimal grasping point, where to find it in space (and at which 3D angle) and how to approach it). Although, extensive literature has been developed so far in the context of static object, grasping of moving objects has not yet seen enough interest, as it is still considered a long shot and there is no big database available for training appropriate CNN configurations. So, a good idea is to incorporate our temporal tracking framework in an overall adaptive grasping layout, in the future, that decides the best time to reach for the object w.r.t its relative pose at every frame.

Chapter 11

Conclusion

If a conclusion is not poetically balanced, it cannot be scientifically true.

Isaac Asimov

In this work, we propose a convolutional framework for fast and accurate single object pose tracking. A CNN is trained on a synthetically generated dataset of RGB-D pairs, with known 6D pose differences between them. Then, it is tested on the hardest available scenario of the most complete realistic Pose Tracking dataset in the literature, up to date.

We mathematically define the problem of Object Pose Tracking, we explain how 3D Graphics are rendered and we showcase a full background analysis of the sum of the modules we utilized. Moreover, we present relevant literature sources in the domains of Object Pose Estimation & Tracking, as well as the dual problem of Camera Pose Estimation and discuss their pros and cons, in detail.

In the core of our method, we try to face innate challenges of the Pose Tracking problems that previous approaches had ignored or inefficiently modeled. We modify the synthetic data generation strategy of a previous approach, as well as enhancing the data augmentation one. As for the CNN, we perform explicitly modular design of clutter and occlusion handling and we account for the geometrical properties of both the pose space and the object model during training. As a result, we reduce both SoA pose errors by an average of 34.03% for translation and 40.01% for rotation and gain an intuitive understanding of our artificial tracking mechanism. Our approach is based on previous, simpler convolutional architecture, present in literature, that had left the aforementioned tracking challenges unaddressed. During our efforts to achieve a performance improvement, we discuss the prior design choices and record an ablation study on the impact each architectural component has on the tracker's properties. Furthermore, we experiment with a variety of possible modular extensions of our approach to model both short and long term temporal dynamics of the object's 3D motion and report our observations. In the future, we aim to extend this work in the object-agnostic case. We, also, plan to examine a single source scenario, via a modality distillation scheme, and to bridge the profound inconsistency between the 2D image source and the 3D pose tracking task. Last but not least, we have the intention of scaling up our approach to multiple objects without/with similar attributes in a Reinforcement Learning framework and to delve into the value of Pose Tracking as a component of Robotic Grasping of moving objects.

Regrets-Afterthoughts

I have not failed. I've just found 10,000 ways that won't work.

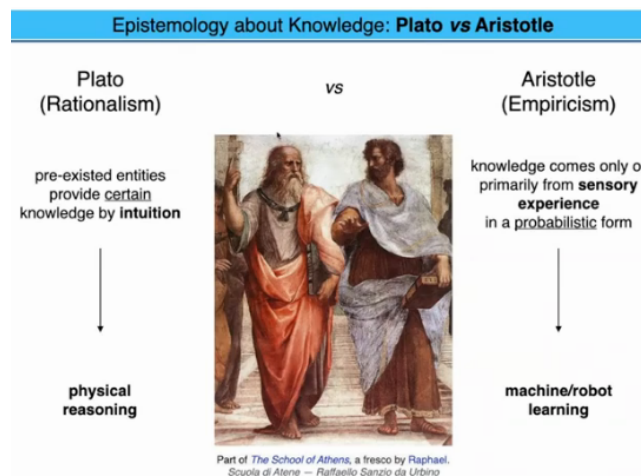
Thomas Edison

After an important time period dealing with this particular problem, we briefly express some inner shifts we experienced, from an ideological and methodological perspective, w.r.t. handling challenging

Machine Learning problems.

- First and foremost, the rise of Convolutional Neural Networks and their recent success in a variety of application has motivated many researchers to abandon mathematical-based modeling completely and employ an end-to-end Neural architecture to resolve challenging problems. During our research activity, we observed how difficult is to tune this kind of layouts and how easy it is to make mistakes that paralyze the distinctive capabilities of the Network. Not only debugging and troubleshooting becomes extensively more costly, in terms of time and computational speed, but also intuition is lost. And intuition does not only make the researcher understand the nature of the problem more accurately, but also indicates useful constraints that should be put in order to guide the CNN's weights to a better local optimum.

Moreover, during studying the Background of our work and the previous efforts that were done in the field through the years, we admired how well older, mathematically based approaches dealt with the tricky problem of pose estimation, even with using poor features and not finetuned to the task at hand. Particle Filtering, Iterative Closest Point, P'n'P etc. are really powerful algorithms on their own. Therefore, our intuition is that the limit of fully neural architectures, like ours, is near and that further research on the topic should be focused in incorporating neural components and learnably regressable parameters in the main core of classical algorithms in order to achieve a consensus. This path should we take, should we have started this Thesis again, from the beginning, today. With one distinctive difference: we would keep the neural architecture backbone as a feature extractor, as CNNs are known to have no match in this domain, and adapt a model-based tracker (e.g. a Particle Filter) to match the computational requirements by cascading it with a pose prior in order to avoid the burden of too many computations (e.g. disentangling translations and rotation particles / or defining particles only for regions of interest guided by an attention scheme similar to the one employed here).



Besides, a model-based approach that would process convolutional feature representations in a way that would predict the target's velocities, along with its pose components is something that matches our intuition as we observe object movements evolve in time. Although, in this work, we have moved more towards the fully learnable approach of “computational empiricism”, a school founded on the Aristotelean mode of thinking, we recognize the success that Platonic “physical reasoning” platforms have shown in the past decades in decision making with remarkable accuracy, without abundance of prior experimentation, finetuning and computational years of past data, but due to diligent modeling of the object's dynamics of motion.

- Secondly, we conceived the innate difficulty that Neural architectures face when trying to learn spatiotemporal relationships and that, frequently, the way to handle such complex relationships seems to be to disentangle the spatial and the temporal component and solve the problem in small parallel local subsets for the latter in order to bypass the problem of long range temporal dependencies. Similar conclusions appear to come out of studying the literature of other similarly complex

spatiotemporal domains of Machine Learning, e.g. Action Recognition, Video Summarization, 2D Object Detection in videos, Optical Flow Estimation etc. Therefore, it is common in such domains to avoid modeling sequential relationships in an explicit way, but either at a per frame basis, either by inputting bunches of images into a CNN and letting it free understand potential temporal dynamics present in the video.

- Last but not least, we became aware of the generalization capabilities of synthetically simulated datasets, even when their appearance does not really correlate with the appearance of the testing set. When we started dealing with this project, we were only aware of their stylistic disadvantages in terms of realism, but, soon, we understood their representational power and the low cost of production of millions of annotated data, even in cases where they are strongly correlated and do not fully imprint the properties of real world. Right now, we strongly believe that bridging the gap between the synthetic and the real world, as efficiently as possible, is the only obstacle in learning complex spatiotemporal conceptual tasks in the future, using learnable architecture and that their widespread usage will make the prospect for success of A.I. algorithms limitless.

Bibliography

- [1] “5 Regression Loss Functions All Machine Learners Should Know”. <https://heartbeat.fritz.ai/5-regression-loss-functions-all-machine-learners-should-know-4fb140e9d4b0>.
- [2] Andrew Baker. “Matrix groups: An introduction to Lie group theory”. In: (2012).
- [3] Vassileios Balntas et al. “Pose guided rgb-d feature learning for 3d object pose estimation”. In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 3856–3864.
- [4] Jonathan T. Barron. “A General and Adaptive Robust Loss Function”. In: (2017). eprint: [arXiv:1701.03077](https://arxiv.org/abs/1701.03077).
- [5] Paul J Besl and Neil D McKay. “Method for registration of 3-D shapes”. In: *Sensor fusion IV: control paradigms and data structures*. Vol. 1611. International Society for Optics and Photonics. 1992, pp. 586–606.
- [6] Mohak Bhardwaj, Byron Boots, and Mustafa Mukadam. “Differentiable Gaussian process motion planning”. In: *arXiv preprint arXiv:1907.09591* (2019).
- [7] Aleksandar Botev, Guy Lever, and David Barber. “Nesterov’s accelerated gradient and momentum as approximations to regularised update descent”. In: *arXiv preprint arXiv:1607.01981* (2016).
- [8] Eric Brachmann et al. “Learning 6d object pose estimation using 3d object coordinates”. In: *European conference on computer vision*. Springer. 2014, pp. 536–551.
- [9] Eric Brachmann et al. “Uncertainty-driven 6d pose estimation of objects and scenes from a single rgb image”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 3364–3372.
- [10] Thomas Brox et al. “High accuracy optical flow estimation based on a theory for warping”. In: *European conference on computer vision*. Springer. 2004, pp. 25–36.
- [11] Romain Brégier et al. “Defining the Pose of Any 3D Rigid Object and an Associated Distance”. In: *International Journal of Computer Vision* 126.6 (2017), 571–596. ISSN: 1573-1405. DOI: [10.1007/s11263-017-1052-4](https://doi.org/10.1007/s11263-017-1052-4). URL: <http://dx.doi.org/10.1007/s11263-017-1052-4>.
- [12] Mai Bui et al. “When regression meets manifold learning for object recognition and pose estimation”. In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2018, pp. 1–7.
- [13] Mai Bui et al. “When regression meets manifold learning for object recognition and pose estimation”. In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2018, pp. 1–7.
- [14] Benjamin Busam, Tolga Birdal, and Nassir Navab. “Camera pose filtering with local regression geodesics on the riemannian manifold of dual quaternions”. In: (2017), pp. 2436–2445.
- [15] Berk Calli et al. “Benchmarking in manipulation research: The YCB object and model set and benchmarking protocols”. In: *arXiv preprint arXiv:1502.03143* (2015).
- [16] Tony Chan and Wei Zhu. “Level set based shape prior segmentation”. In: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’05)*. Vol. 2. IEEE. 2005, pp. 1164–1170.

- [17] Changhyun Choi and Henrik I. Christensen. “RGB-D object tracking: A particle filter approach on GPU”. In: *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*. 2013, pp. 1084–1091.
- [18] Chua Hock Chuan. *3D Graphics with OpenGL Basic Theory*. URL: https://www.ntu.edu.sg/home/ehchua/programming/opengl/CG_BasicsTheory.html.
- [19] George Cybenko. “Approximation by superpositions of a sigmoidal function”. In: *Mathematics of control, signals and systems 2.4* (1989), pp. 303–314.
- [20] Erik B Dam, Martin Koch, and Martin Lillholm. *Quaternions, interpolation and animation*. Vol. 2. Citeseer, 1998.
- [21] Xinke Deng et al. “Poserbpf: A rao-blackwellized particle filter for 6d object pose tracking”. In: *arXiv preprint arXiv:1905.09304* (2019).
- [22] Guoguang Du, Kai Wang, and Shiguo Lian. “Vision-based robotic grasping from object localization, pose estimation, grasp detection to motion planning: A review”. In: *arXiv preprint arXiv:1905.06658* (2019).
- [23] Xiaomin Duan, Huafei Sun, and Linyu Peng. “Riemannian means on special Euclidean group and unipotent matrices group”. In: *The Scientific World Journal 2013* (2013).
- [24] Philipp Fischer et al. “Flownet: Learning optical flow with convolutional networks”. In: *arXiv preprint arXiv:1504.06852* (2015).
- [25] Martin A Fischler and Robert C Bolles. “Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography”. In: *Communications of the ACM 24.6* (1981), pp. 381–395.
- [26] Peter R. Florence, Lucas Manuelli, and Russ Tedrake. “Dense Object Nets: Learning Dense Visual Object Descriptors By and For Robotic Manipulation”. In: (2018). arXiv: [1806.08756](https://arxiv.org/abs/1806.08756) [cs.R0].
- [27] Dieter Fox. *oward robust manipulation in complex environments*. URL: <https://www.youtube.com/watch?v=jZa1S5LFa1o>.
- [28] Ge Gao et al. “Occlusion resistant object rotation regression from point cloud segments”. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 0–0.
- [29] Mathieu Garon and Jean-François Lalonde. “Deep 6-DOF Tracking”. In: *IEEE Transactions on Visualization and Computer Graphics 23.11* (2017).
- [30] “Generating optical flow using NVIDIA flownet2-pytorch implementation”. <https://towardsdatascience.com/generating-optical-flow-using-nvidia-flownet2-pytorch-implementation-d7b0ae6f8320>.
- [31] Xavier Glorot and Yoshua Bengio. “Understanding the difficulty of training deep feedforward neural networks”. In: *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. 2010, pp. 249–256.
- [32] Ian J. Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. <http://www.deeplearningbook.org>. Cambridge, MA, USA: MIT Press, 2016.
- [33] Purnasai Gudikandula. “Recurrent Neural Networks and LSTM explained”. <https://medium.com/@purnasaigudikandula/recurrent-neural-networks-and-lstm-explained-7f51c7f6bbb9>.
- [34] Jack Hadfield et al. “Object Assembly Guidance in Child-Robot Interaction using RGB-D based 3D Tracking”. In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2018), pp. 347–354.
- [35] Frederik Hagelskjær and Anders Glent Buch. “PointPoseNet: Accurate Object Detection and 6 DoF Pose Estimation in Point Clouds”. In: *arXiv preprint arXiv:1912.09057* (2019).
- [36] Adam W Harley et al. “Tracking Emerges by Looking Around Static Scenes, with Neural 3D Mapping”. In: *arXiv preprint arXiv:2008.01295* (2020).
- [37] Kaiming He et al. “Deep Residual Learning for Image Recognition”. In: (2015). eprint: [arXiv: 1512.03385](https://arxiv.org/abs/1512.03385).

- [38] Kaiming He et al. “Delving deep into rectifiers: Surpassing human-level performance on imagenet classification”. In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1026–1034.
- [39] Kaiming He et al. “Mask r-cnn”. In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2961–2969.
- [40] Yisheng He et al. “PVN3D: A Deep Point-wise 3D Keypoints Voting Network for 6DoF Pose Estimation”. In: *arXiv preprint arXiv:1911.04231* (2019).
- [41] David Held, Sebastian Thrun, and Silvio Savarese. “Learning to Track at 100 FPS with Deep Regression Networks”. In: *European Conference Computer Vision (ECCV)*. 2016.
- [42] Stefan Hinterstoisser et al. “On pre-trained image features and synthetic images for deep learning”. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 0–0.
- [43] Geoffrey Hinton, Nitish Srivastava, and Kevin Swersky. “Neural networks for machine learning lecture 6a overview of mini-batch gradient descent”. In: *Cited on 14.8* (2012).
- [44] Sepp Hochreiter and Jürgen Schmidhuber. “Long short-term memory”. In: *Neural computation* 9.8 (1997), pp. 1735–1780.
- [45] Thomas Hodan. “BOP: Benchmark for 6D Object Pose Estimation”. In: (2020). URL: <https://bop.felk.cvut.cz/challenges/bop-challenge-2020/>.
- [46] Tomas Hodan. “BOP Challenge 2019”. In: (2019). URL: http://cmp.felk.cvut.cz/sixd/workshop_2019/slides/r6d19_hodan_bop_challenge_2019.pdf.
- [47] Berthold KP Horn and Brian G Schunck. “Determining optical flow”. In: *Techniques and Applications of Image Understanding*. Vol. 281. International Society for Optics and Photonics. 1981, pp. 319–331.
- [48] Omid Hosseini Jafari et al. “iPose: Instance-Aware 6D Pose Estimation of Partly Occluded Objects”. In: *Lecture Notes in Computer Science* (2019), 477–492. ISSN: 1611-3349. DOI: [10.1007/978-3-030-20893-6_30](https://doi.org/10.1007/978-3-030-20893-6_30). URL: http://dx.doi.org/10.1007/978-3-030-20893-6_30.
- [49] Gao Huang et al. “Densely connected convolutional networks”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017.
- [50] Peter J Huber. “Robust estimation of a location parameter”. In: *Breakthroughs in statistics*. Springer, 1992, pp. 492–518.
- [51] Forrest N. Iandola et al. “SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size”. In: *arXiv:1602.07360* (2016).
- [52] Eddy Ilg et al. “FlowNet 2.0: Evolution of Optical Flow Estimation with Deep Networks”. In: (2016). eprint: [arXiv:1612.01925](https://arxiv.org/abs/1612.01925).
- [53] Max Jaderberg et al. “Spatial Transformer Networks”. In: (2015). arXiv: [1506.02025 \[cs.CV\]](https://arxiv.org/abs/1506.02025).
- [54] David Joseph Tan et al. “A versatile learning-based 3d temporal tracker: Scalable, robust, online”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015, pp. 693–701.
- [55] Roman Kaskman et al. “HomebrewedDB: RGB-D Dataset for 6D Pose Estimation of 3D Objects”. In: *Proceedings of the IEEE International Conference on Computer Vision Workshops*. 2019, pp. 0–0.
- [56] Wadim Kehl et al. “Deep learning of local RGB-D patches for 3D object detection and 6D pose estimation”. In: *European conference on computer vision*. Springer. 2016, pp. 205–220.
- [57] Wadim Kehl et al. “Deep learning of local RGB-D patches for 3D object detection and 6D pose estimation”. In: *European conference on computer vision*. Springer. 2016, pp. 205–220.
- [58] Wadim Kehl et al. “Ssd-6d: Making rgb-based 3d detection and 6d pose estimation great again”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 1521–1529.
- [59] Alex Kendall and Roberto Cipolla. “Geometric loss functions for camera pose regression with deep learning”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 5974–5983.

- [60] Alex Kendall, Yarin Gal, and Roberto Cipolla. “Multi-Task Learning Using Uncertainty to Weigh Losses for Scene Geometry and Semantics”. In: (2017). eprint: [arXiv:1705.07115](https://arxiv.org/abs/1705.07115).
- [61] Alex Kendall, Matthew Grimes, and Roberto Cipolla. “Posenet: A convolutional network for real-time 6-dof camera relocalization”. In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 2938–2946.
- [62] Prannay Khosla et al. “Supervised contrastive learning”. In: *arXiv preprint arXiv:2004.11362* (2020).
- [63] Diederik P Kingma and Jimmy Ba. “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980* (2014).
- [64] Muhammed Kocabas, Nikos Athanasiou, and Michael J Black. “VIBE: Video Inference for Human Body Pose and Shape Estimation”. In: *arXiv preprint arXiv:1912.05656* (2019).
- [65] Gregory R. Koch. “Siamese Neural Networks for One-Shot Image Recognition”. In: 2015.
- [66] Iasonas Kokkinos. “Introduction to Machine Learning: Neural Networks”. University College London. 2016.
- [67] Iasonas Kokkinos. “Introduction to Machine Learning: Optimization, Regularization and Applications of Deep Learning”. University College London. 2016.
- [68] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “Imagenet classification with deep convolutional neural networks”. In: *Advances in neural information processing systems*. 2012, pp. 1097–1105.
- [69] Alexander Krull et al. “Learning analysis-by-synthesis for 6D pose estimation in RGB-D images”. In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 954–962.
- [70] Nikolaos Kyriazis and Antonis Argyros. “Scalable 3d tracking of multiple interacting objects”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2014, pp. 3430–3437.
- [71] Vincent Lepetit, Francesc Moreno-Noguer, and Pascal Fua. “Epnnp: An accurate o (n) solution to the pnp problem”. In: *International journal of computer vision* 81.2 (2009), p. 155.
- [72] Hao Li et al. “Visualizing the loss landscape of neural nets”. In: *Advances in Neural Information Processing Systems*. 2018, pp. 6389–6399.
- [73] Yi Li et al. “Deepim: Deep iterative matching for 6d pose estimation”. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 683–698.
- [74] Yunzhu Li et al. “Learning Compositional Koopman Operators for Model-Based Control”. In: (2019). arXiv: [1910.08264](https://arxiv.org/abs/1910.08264) [cs.LG].
- [75] Miao Liu et al. “Attention Distillation for Learning Video Representations”. In: *arXiv preprint arXiv:1904.03249* (2019).
- [76] Miao Liu et al. “Paying More Attention to Motion: Attention Distillation for Learning Video Representations”. In: *arXiv preprint arXiv:1904.03249* (2019).
- [77] Wei Liu et al. “SSD: Single Shot MultiBox Detector”. In: To appear. 2016. URL: <http://arxiv.org/abs/1512.02325>.
- [78] Ilya Loshchilov and Frank Hutter. “Decoupled weight decay regularization”. In: *arXiv preprint arXiv:1711.05101* (2017).
- [79] Ilya Loshchilov and Frank Hutter. “Sgdr: Stochastic gradient descent with warm restarts”. In: *arXiv preprint arXiv:1608.03983* (2016).
- [80] Ying-Sheng Luo et al. “CARL: Controllable Agent with Reinforcement Learning for Quadruped Locomotion”. In: *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2020)* 39.4 (2020).
- [81] Siddharth Mahendran, Haider Ali, and Rene Vidal. “3D Pose Regression Using Convolutional Neural Networks”. In: *The IEEE International Conference on Computer Vision (ICCV) Workshops*. 2017.

- [82] Isidoros Maroukias et al. “How to track your dragon: A multi-attentional framework for real-time rgb-d 6-dof object pose tracking”. In: *European Conference on Computer Vision*. Springer. 2020, pp. 682–699.
- [83] Denis Laurendeau Mathieu Garon and Jean-François Lalonde. “A Framework for Evaluating 6-DOF Object Trackers”. In: *European conference on computer vision*. 2018.
- [84] Y Susuki Mauroy and I Mezić. *Koopman Operator in Systems and Control*. Springer, 2020.
- [85] Warren S McCulloch and Walter Pitts. “A logical calculus of the ideas immanent in nervous activity”. In: *The bulletin of mathematical biophysics* 5.4 (1943), pp. 115–133.
- [86] *Metrics on $SO(3)$ and Inverse Kinematics*.
- [87] Anton Milan. “Energy minimization for multiple object tracking”. PhD thesis. Technische Universität, 2013.
- [88] Andriy Myronenko and Xubo Song. “Point set registration: Coherent point drift”. In: *IEEE transactions on pattern analysis and machine intelligence* 32.12 (2010), pp. 2262–2275.
- [89] Ralph Neuneier and Hans Georg Zimmermann. “How to train neural networks”. In: *Neural networks: tricks of the trade*. Springer, 1998, pp. 373–423.
- [90] Alejandro Newell and Jia Deng. “How Useful is Self-Supervised Pretraining for Visual Tasks?” In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 7345–7354.
- [91] Chuong V. Nguyen, Shahram Izadi, and David R. Lovell. “Modeling Kinect Sensor Noise for Improved 3D Reconstruction and Tracking”. In: *2012 Second International Conference on 3D Imaging, Modeling, Processing, Visualization & Transmission* (2012), pp. 524–530.
- [92] Michael Nguyen. “Illustrated Guide to LSTM’s and GRU’s: A step by step explanation”. <https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21>.
- [93] *Numerical Analysis and Computing for Interpolation and Polynomial Approximation (Piecewise Polynomial Approximation, Cubic Splines)*.
- [94] Christopher Olah. “Understanding LSTM Networks”. <https://colah.github.io/posts/2015-08-Understanding-LSTMs/>.
- [95] Sinno Jialin Pan and Qiang Yang. “A survey on transfer learning”. In: *IEEE Transactions on knowledge and data engineering* 22.10 (2009), pp. 1345–1359.
- [96] Frank C Park and Roger W Brockett. “Kinematic dexterity of robotic mechanisms”. In: *The International Journal of Robotics Research* 13.1 (1994), pp. 1–15.
- [97] Keunhong Park et al. “LatentFusion: End-to-End Differentiable Reconstruction and Rendering for Unseen Object Pose Estimation”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 10710–10719.
- [98] Kiru Park, Timothy Patten, and Markus Vincze. “Pix2pose: Pixel-wise coordinate regression of objects for 6d pose estimation”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2019, pp. 7668–7677.
- [99] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. “On the difficulty of training recurrent neural networks”. In: *International conference on machine learning*. 2013, pp. 1310–1318.
- [100] Karl Pauwels et al. “Real-time model-based rigid object pose estimation and tracking combining dense and sparse visual cues”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2013, pp. 2347–2354.
- [101] Georgios Pavlakos et al. “6-dof object pose from semantic keypoints”. In: *2017 IEEE international conference on robotics and automation (ICRA)*. IEEE. 2017, pp. 2011–2018.
- [102] Sida Peng et al. “Pvnet: Pixel-wise voting network for 6dof pose estimation”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 4561–4570.
- [103] Javier Sánchez Pérez, Enric Meinhardt-Llopis, and Gabriele Facciolo. “TV-L1 optical flow estimation”. In: *Image Processing On Line* 2013 (2013), pp. 137–150.

- [104] Charles R Qi et al. “Pointnet: Deep learning on point sets for 3d classification and segmentation”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 652–660.
- [105] Alessio R. *Online/Recursive Variance calculation – Welford’s Method*. 2017. URL: <https://alessior.wordpress.com/2017/10/09/onlinerecursive-variance-calculation-welfords-method/>.
- [106] Mahdi Rad and Vincent Lepetit. “Bb8: A scalable, accurate, robust to partial occlusion method for predicting the 3d poses of challenging objects without using depth”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 3828–3836.
- [107] Prajit Ramachandran, Barret Zoph, and Quoc V Le. “Searching for activation functions”. In: *arXiv preprint arXiv:1710.05941* (2017).
- [108] *Random Points on a Sphere*.
- [109] Anurag Ranjan and Michael J. Black. “Optical Flow Estimation using a Spatial Pyramid Network”. In: (2016). eprint: [arXiv:1611.00850](https://arxiv.org/abs/1611.00850).
- [110] Joseph Redmon et al. “You Only Look Once: Unified, Real-Time Object Detection”. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016). DOI: [10.1109/cvpr.2016.91](https://doi.org/10.1109/cvpr.2016.91). URL: <http://dx.doi.org/10.1109/CVPR.2016.91>.
- [111] Carl Yuheng Ren et al. “3D tracking of multiple objects with identical appearance using RGB-D input”. In: *2014 2nd International Conference on 3D Vision*. Vol. 1. IEEE. 2014, pp. 47–54.
- [112] Colin Rennie et al. “A dataset for improved rgbd-based object detection and pose estimation for warehouse pick-and-place”. In: *IEEE Robotics and Automation Letters* 1.2 (2016), pp. 1179–1185.
- [113] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. “U-net: Convolutional networks for biomedical image segmentation”. In: *International Conference on Medical image computing and computer-assisted intervention*. Springer. 2015, pp. 234–241.
- [114] Frank Rosenblatt. “The perceptron: a probabilistic model for information storage and organization in the brain.” In: *Psychological review* 65.6 (1958), p. 386.
- [115] Sebastian Ruder.
- [116] Sebastian Ruder. “An overview of gradient descent optimization algorithms”. In: *arXiv preprint arXiv:1609.04747* (2016).
- [117] Arthur A Sagle and R Walde. “Introduction to Lie Groups and Lie Algebra, 51”. In: (1986).
- [118] Mathieu Salzmann and Raquel Urtasun. “Physically-based motion models for 3D tracking: A convex formulation”. In: *2011 International Conference on Computer Vision*. IEEE. 2011, pp. 2064–2071.
- [119] Aleksandr Segal, Dirk Haehnel, and Sebastian Thrun. “Generalized-icp.” In: *Robotics: science and systems*. Vol. 2. 4. Seattle, WA. 2009, p. 435.
- [120] Wenzhe Shi et al. “Is the deconvolution layer the same as a convolutional layer?” In: *arXiv preprint arXiv:1609.07009* (2016).
- [121] Nitish Srivastava et al. “Dropout: a simple way to prevent neural networks from overfitting”. In: *The journal of machine learning research* 15.1 (2014), pp. 1929–1958.
- [122] Gilbert Strang. *Linear algebra and its applications*. Belmont, CA: Thomson, Brooks/Cole, 2006. ISBN: 0030105676 9780030105678 0534422004 9780534422004. URL: <http://www.amazon.com/Linear-Algebra-Its-Applications-Edition/dp/0030105676>.
- [123] Hang Su et al. “Pixel-adaptive convolutional neural networks”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 11166–11175.
- [124] Jae Kyu Suhr. “Kanade-lucas-tomasi (klt) feature tracker”. In: *Computer Vision (EEE6503)* (2009), pp. 9–18.
- [125] Deqing Sun et al. “PWC-Net: CNNs for Optical Flow Using Pyramid, Warping, and Cost Volume”. In: (2017). eprint: [arXiv:1709.02371](https://arxiv.org/abs/1709.02371).

- [126] Martin Sundermeyer et al. “Implicit 3d orientation learning for 6d object detection from rgb images”. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 699–715.
- [127] Christian Szegedy et al. “Going deeper with convolutions”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 1–9.
- [128] David Joseph Tan, Nassir Navab, and Federico Tombari. “Looking beyond the simple scenarios: Combining learners and optimizers in 3d temporal tracking”. In: *IEEE transactions on visualization and computer graphics* 23.11 (2017), pp. 2399–2409.
- [129] Hai Tao, Harpreet S Sawhney, and Rakesh Kumar. “Object tracking with bayesian estimation of dynamic layer representations”. In: *IEEE transactions on pattern analysis and machine intelligence* 24.1 (2002), pp. 75–89.
- [130] Bugra Tekin, Sudipta N Sinha, and Pascal Fua. “Real-time seamless single shot 6d object pose prediction”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 292–301.
- [131] Ayush Tewari et al. “State of the Art on Neural Rendering”. In: *arXiv preprint arXiv:2004.03805* (2020).
- [132] Nathaniel Thomas et al. “Tensor Field Networks: Rotation- and Translation-Equivariant Neural Networks for 3D Point Clouds”. In: (2018). cite arxiv:1802.08219. URL: <http://arxiv.org/abs/1802.08219>.
- [133] Meng Tian et al. “Robust 6D Object Pose Estimation by Learning RGB-D Features”. In: (2020). arXiv: [2003.00188](https://arxiv.org/abs/2003.00188) [cs.CV].
- [134] Henning Tjaden, Ulrich Schwanecke, and Elmar Schomer. “Real-time monocular pose estimation of 3D objects using temporally consistent local color histograms”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 124–132.
- [135] Ameni Trabelsi et al. “A Novel Pose Proposal Network and Refinement Pipeline for Better Object Pose Estimation”. In: (2020). arXiv: [2004.05507](https://arxiv.org/abs/2004.05507) [cs.CV].
- [136] Jonathan Tremblay, Thang To, and Stan Birchfield. “Falling things: A synthetic dataset for 3d object detection and pose estimation”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2018, pp. 2038–2041.
- [137] Jonathan Tremblay et al. “Deep object pose estimation for semantic robotic grasping of household objects”. In: *arXiv preprint arXiv:1809.10790* (2018).
- [138] Yanghai Tsin and Takeo Kanade. “A correlation-based approach to robust point set registration”. In: *European conference on computer vision*. Springer. 2004, pp. 558–569.
- [139] Hsiao-Yu Fish Tung, Ricson Cheng, and Katerina Fragkiadaki. “Learning Spatial Common Sense with Geometry-Aware Recurrent Networks”. In: *CoRR* abs/1901.00003 (2019). arXiv: [1901.00003](https://arxiv.org/abs/1901.00003). URL: <http://arxiv.org/abs/1901.00003>.
- [140] Shinji Umeyama. “Least-squares estimation of transformation parameters between two point patterns”. In: *IEEE Transactions on Pattern Analysis & Machine Intelligence* 4 (1991), pp. 376–380.
- [141] Shinji Umeyama. “Least-Squares Estimation of Transformation Parameters Between Two Point Patterns”. In: *IEEE Trans. Pattern Anal. Mach. Intell.* 13.4 (Apr. 1991), pp. 376–380. ISSN: 0162-8828. DOI: [10.1109/34.88573](https://doi.org/10.1109/34.88573). URL: <https://doi.org/10.1109/34.88573>.
- [142] Wouter Van Gansbeke et al. “SCAN: Learning to Classify Images without Labels”. In: *European Conference on Computer Vision (ECCV)*. 2020.
- [143] Gül Varol, Ivan Laptev, and Cordelia Schmid. “Long-term temporal convolutions for action recognition”. In: *IEEE transactions on pattern analysis and machine intelligence* 40.6 (2017), pp. 1510–1517.
- [144] Ashish Vaswani et al. “Attention is all you need”. In: *Advances in neural information processing systems*. 2017, pp. 5998–6008.

- [145] Ashish Vaswani et al. “Attention is all you need”. In: *Advances in neural information processing systems*. 2017, pp. 5998–6008.
- [146] Marcus Wallenberg. “Embodied Visual Object Recognition”. PhD thesis. Linköping University Electronic Press, 2017.
- [147] Eric A Wan and Rudolph Van Der Merwe. “The unscented Kalman filter for nonlinear estimation”. In: *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No. 00EX373)*. Ieee. 2000, pp. 153–158.
- [148] Gu Wang et al. “Self6D: Self-Supervised Monocular 6D Object Pose Estimation”. In: *arXiv preprint arXiv:2004.06468* (2020).
- [149] He Wang et al. “Normalized Object Coordinate Space for Category-Level 6D Object Pose and Size Estimation”. In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.
- [150] Yang Wang et al. “Occlusion aware unsupervised learning of optical flow”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 4884–4893.
- [151] BP Welford. “Note on a method for calculating corrected sums of squares and products”. In: *Technometrics* 4.3 (1962), pp. 419–420.
- [152] Lilian Weng. “Domain Randomization for Sim2Real Transfer”. In: (2019). URL: <https://lilianweng.github.io/lil-log/2019/05/05/domain-randomization.html>.
- [153] Wikipedia contributors. *Activation functions Wikipedia page*. [Online; accessed 22-July-2004]. 2004. URL: https://en.wikipedia.org/wiki/Activation_function.
- [154] Wikipedia contributors. *Axis-Angles Wikipedia page*. [Online; accessed 22-July-2004]. 2004. URL: https://https://en.wikipedia.org/wiki/Axis%E2%80%93angle_representation.
- [155] Wikipedia contributors. *Bilinear Interpolation Wikipedia page*. [Online; accessed 22-July-2004]. 2004. URL: https://en.wikipedia.org/wiki/Bilinear_interpolation.
- [156] Wikipedia contributors. *Euler Angles Wikipedia page*. [Online; accessed 22-July-2004]. 2004. URL: https://https://en.wikipedia.org/wiki/Euler_angles.
- [157] Wikipedia contributors. *HSV Wikipedia page*. [Online; accessed 22-July-2004]. 2004. URL: https://en.wikipedia.org/wiki/HSL_and_HSV.
- [158] Wikipedia contributors. *kMeans Wikipedia page*. [Online; accessed 22-July-2004]. 2004. URL: https://en.wikipedia.org/wiki/K-means_clustering.
- [159] Wikipedia contributors. *Quaternia Wikipedia page*. [Online; accessed 22-July-2004]. 2004. URL: <https://en.wikipedia.org/wiki/Quaternion>.
- [160] Wikipedia contributors. *SVD Wikipedia page*. [Online; accessed 22-July-2004]. 2004. URL: https://https://en.wikipedia.org/wiki/Singular_value_decomposition.
- [161] Paul Wohlhart and Vincent Lepetit. “Learning descriptors for object recognition and 3d pose estimation”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 3109–3118.
- [162] Yu Xiang et al. “Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes”. In: *arXiv preprint arXiv:1711.00199* (2017).
- [163] Jianxiong Xiao, Andrew Owens, and Antonio Torralba. “SUN3D: A Database of Big Spaces Reconstructed Using SfM and Object Labels”. In: *2013 IEEE International Conference on Computer Vision* (2013), pp. 1625–1632.
- [164] SHI Xingjian et al. “Convolutional LSTM network: A machine learning approach for precipitation nowcasting”. In: *Advances in neural information processing systems*. 2015, pp. 802–810.
- [165] Huijuan Xu and Kate Saenko. “Ask, attend and answer: Exploring question-guided spatial attention for visual question answering”. In: *European Conference on Computer Vision*. Springer. 2016, pp. 451–466.
- [166] Jianwei Yang et al. “Embodied visual recognition”. In: *arXiv preprint arXiv:1904.04404* (2019).

- [167] Sergey Zakharov, Ivan Shugurov, and Slobodan Ilic. “DPOD: 6D Pose Object Detector and Refiner”. In: (2019). arXiv: [1902.11020](https://arxiv.org/abs/1902.11020) [[cs.CV](#)].
- [168] Milos Zefran and Vijay Kumar. “Planning of smooth motions on $SE(3)$ ”. In: *Proceedings of IEEE International Conference on Robotics and Automation*. Vol. 1. IEEE. 1996, pp. 121–126.
- [169] Huizhong Zhou, Benjamin Ummenhofer, and Thomas Brox. “Deeptam: Deep tracking and mapping”. In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 822–838.
- [170] Jie Zhou et al. “Graph Neural Networks: A Review of Methods and Applications”. In: (2018). arXiv: [1812.08434](https://arxiv.org/abs/1812.08434) [[cs.LG](#)].
- [171] Yi Zhou et al. “On the Continuity of Rotation Representations in Neural Networks”. In: (2018). eprint: [arXiv:1812.07035](https://arxiv.org/abs/1812.07035).