ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΑΓΡΟΝΟΜΩΝ ΤΟΠΟΓΡΑΦΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΓΕΩΠΛΗΡΟΦΟΡΙΚΗΣ

Υλοποίηση μεθοδολογίας για την εκπαίδευση αλγορίθμου μηχανικής μάθησης για τον προσδιορισμό 6D πόζας αντικειμένου χωρίς υφή

Διπλωματική εργασία

Παναγιώτης Σαπουτζόγλου

Μάρτιος, 2023

Ευχαριστίες

Θα ήθελα να ευχαριστήσω ιδιαίτερα την επιβλέπουσα καθηγήτρια της διπλωματικής μου εργασίας Μαρία Πατεράκη για την πολύτιμη καθοδήγηση της, το χρόνο που αφιέρωσε συζητώντας μαζί μου για την επίλυση κάθε προβλήματος, προτείνοντας λύσεις αλλά και για την υποστήριξη των ιδεών μου. Ακόμα, την ευχαριστώ για τις ευκαιρίες που μου έχει δώσει να ασχοληθώ με ενδιαφέροντα ερευνητικά αντικείμενα.

Ευχαριστώ θερμά τον κ. Μανόλη Λουράκη, Ερευνητή στο Ίδρυμα Τεχνολογίας και Έρευνας για την πολύτιμη βοήθεια, την καθοδήγηση αλλά και το υλικό που μου παρείχε. Με τη βοήθεια εκείνου και της Μ. Πατεράκη η παρούσα διπλωματική έγινε αντικείμενο δημοσίευσης σε συνέδριο.

Τέλος, θα ήθελα να ευχαριστήσω την οικογένεια μου που με στήριξαν με κάθε δυνατό τρόπο κατά τη διάρκεια φοίτησης μου αλλά και τους Μ. Λιανό, Γ. Κουγιανό, Γ. Ταλαβέρο, Ι. Βουτσίνο και Ε. Μποσδελεκίδου που μου συμπαραστάθηκαν όχι μόνο σαν συμφοιτητές σε θέματα σχολής αλλά γενικά στη ζωή μου σαν πολύτιμοι φίλοι.

Ακρωνύμια

AAE	Absolute Angular Error
APE	Absolute Positional Error
API	Application Programming Interface
BB	Bounding Box
CNN	Convolutional Neural Network
CUDA	Compute Unified Architecture
DL	Deep Learning
DLT	Direct Linear Transform
FPS	Frames Per Second
GLSL	OpenGL Shading Language
GNSS	Global Navigation Satellite System
GPU	Graphics Processing Unit
GT	Ground Truth
HLR	Hidden Line Removal
ICP	Iterative Closest Point
LCM	Least Common Multiple
MAE	Mean Absolute error
ML	Machine Learning
MSE	Mean Squared Error
NDC	Normalized Device Coordinates
NN	Neural Network
PnP	Perspective-n-Point
PSO	Particle Swarm Optimization
RAE	Relative Angular Error
RMSE	Root Mean Squared Error
RPE	Relative Positional Error
RPN	Region Proposal Network
RTK	Real-time Kinematic positioning
SFM	Structure For Motion
SGD	Stohastic Gradient Descent
SVD	Singular Value Decomposition
UI	User Interface
VAO	Vertex Array Object
VBO	Vertex Buffer Object

Περιεχόμενα

Κατάλογος σχημάτων

Κατάλογος πινάκων

Κατάλογος δειγμάτων κώδικα

Περίληψη

Η αναγνώριση αντικειμένων και ο προσδιορισμός της θέσης και της στροφής τους ως προς το σύστημα της κάμερας είναι ένα σημαντικό και βασικό πρόβλημα του τομέα της υπολογιστικής όρασης και της φωτογραμμετρίας, το οποίο βρίσκει εφαρμογή σε διάφορους τομείς όπως είναι η ρομποτική, η επαυξημένη πραγματικότητα και τα έξυπνα περιβάλλοντα λόγω των αυξανόμενων απαιτήσεων αυτοματοποίησης διαδικασιών αναγνώρισης, εκπαίδευσης, ασφάλειας,κ.ά.. Οι δύο αυτοί τομείς βασίζονται στις ίδιες γεωμετρικές αρχές της προβολικής γεωμετρίας. Συγκεκριμένα, από τη σχετική βιβλιογραφία για την εκτίμηση της 6D πόζας αντικειμένων φαίνεται ότι υπάρχει η τάση να αναπτύσσονται μέθοδοι όπου για δεδομένα εισόδου χρησιμοποιούν τόσο RGB όσο και RGB-D εικόνες. Ωστόσο, η απόκτηση της πληροφορία του βάθους από ενεργητικούς αισθητήρες δεν είναι εφικτή στον ίδιο βαθμό για όλα τα αντικείμενα (πχ. μεταλλικά η ημιδιαφανή) και η τοποθέτηση πολλαπλών αισθητήρων (πχ. στερεοκάλυψη) αυξάνει σημαντικά το υπολογιστικό κόστος. Εκτός από αυτό, σε τέτοια περιβάλλοντα είναι συχνές οι περιπτώσεις όπου το αντικείμενο έχει αποκρύψεις (occlusions) - έως και μεγάλου μέρους του - στην εικόνα λόγω παρεμβολής άλλων αντικειμένων μεταξύ αυτού και της κάμερας, ή το παρασκήνιο αποτελείται από πάρα πολλές οντότητες με αποτέλεσμα το αντικείμενο να αναγνωρίζεται πιο δύσκολα. Για την αντιμετώπιση των παραπάνω, έχουν προταθεί τα τελευταία χρόνια πολλές μέθοδοι με εκείνες οι οποίες βασίζονται σε δίκτυα βαθιάς μάθησης να έχουν εξέχουσα θέση. Αν και πολύ αποτελεσματικά τόσο σε ακρίβεια όσο και σε χρόνο, τα δίκτυα βαθιάς μάθησης είναι άμεσα εξαρτημένα από τον όγκο των δεδομένων εκπαίδευσης, την ποιότητα τους αλλά και το είδος τους. Λόγω της εντατικοποίησης της χρήσης έξυπνων συστημάτων που βασίζονται στην υπολογιστική όραση στο χώρο των μεταφορών και της εφοδιαστικής σε λιμάνια (port logistics) η παρούσα διπλωματική ασχολείται με τον προσδιορισμό της 6D πόζας αρπάγης για container από μία μόνο RGB εικόνα.

Αρχικά, παραθέτονται κάποιες βασικές έννοιες και το απαραίτητο θεωρητικό υπόβαθρο για την διεκπεραίωση της εργασίας. Συγκεκριμένα, γίνεται αναφορά στη μοντελοποίηση του συστήματος της κάμερας που χρησιμοποιεί η υπολογιστική όραση, στις ομογενείς συντεταγμένες και το σημαντικό ρόλο τους στην πραγματοποίηση μετασχηματισμών με τη χρήση διανυσμάτων και πινάκων, στην αναπαράσταση της στροφής στο χώρο μέσω τετραδρονιών αλλά και σε βασικές έννοιες της υλοποίησης 3D γραφικών μέσω της OpenGl. Ακόμα, αναφέρονται σημαντικές έννοιες της βαθιά μάθησης όπως η διαδικασία εκπαίδευσης των νευρωνικών δικτύων, η αρχιτεκτονική τους και οι υπερ-παράμετροι που τη συνοδεύουν. Τα παραπάνω στοχεύουν στην βαθύτερη κατανόηση της λειτουργίας των συνελικτικών νευρωνικών δικτύων (CNN) σαν μέσο για την επίτευξη καλύτερων αποτελεσμάτων, τη αντίληψη των περιορισμών τους αλλά και τρόπους για την βελτιστοποίηση τους.

Έπειτα, γίνεται αναλυτική αναφορά στη σχετική βιβλιογραφία και ο διαχωρισμός των διάφορων μεθόδων προσδιορισμού της 6D πόζας σε διακριτές κατηγορίες επισημαίνοντας τη συνεισφορά, τα πλεονεκτήματα και τις αδυναμίες τους. Οι κύριες κατηγορίες είναι μέθοδοι που βασίζονται σε μια διαδικασία εκμάθησης και οι κλασσικές μέθοδοι. Οι μεν πρώτες χρησιμοποιούν τα CNN για την εξαγωγή χαρακτηριστικών στις εικόνες ενώ οι δεύτερες χρησιμοποιούν κλασσικούς αλγορίθμους εύρεσης χαρακτηριστικών σημείων. Πιο συγκεκριμένα, οι μέθοδοι εκμάθησης χωρίζονται σε επιμέρους κατηγορίες ανάλογα με το πως αντιμετωπίζουν το πρόβλημα του προσδιορισμού της 6D πόζας. Κάποιες προβλέπουν αντιστοιχίες 2D-3D και λύνουν το πρόβλημα PnP, άλλες δημιουργούν ένα περιορισμένο ομοιόμορφο σετ από πόζες του αντικειμένου και αναζητούν με βάση την εικόνα τη πλησιέστερη από αυτές και άλλες που χρησιμοποιούν απευθείας παλινδρόμηση της 6D πόζας του αντικειμένου.

Για την εκπαίδευση του αλγορίθμου προσδιορισμού της 6D πόζας της αρπάγης χρειάζονται εικόνες στις οποίες αυτή απεικονίζεται συνοδευόμενες από τις πόζες αναφοράς. Η απόκτηση των τελευταίων είναι δύσκολο να γίνει λόγω των μεγάλων διαστάσεων του αντικειμένου και λόγω του γεγονότος ότι δεν

μπορεί εύκολα να συστηματοποιηθεί η λήψη τους. Ακόμα, η χειροκίνητη απόκτηση και επεξεργασία τους είναι χρονοβόρα. Για τον λόγο αυτό μέρος της εργασίας αποτελεί η δημιουργία ενός συνθετικού σετ εκπαίδευσης. Συγκεκριμένα, με τη χρήση ενός φωτορεαλιστικού μοντέλου της αρπάγης παρήχθησαν εικόνες της υπό διάφορες γωνίες λήψεις και αποστάσεις. Για την δημιουργία του φωτορεαλιστικού μοντέλου της αρπάγης έγινε η απόδοση της υφής του στο διαθέσιμο γεωμετρικό μοντέλο από εικόνες της αρπάγης που ελήφθησαν στο πεδίο με μία ερασιτεχνική κάμερα. Έγινε πειραματισμός πάνω σε δύο προγράμματα ανοιχτού λογισμικού ο οποίος κατέληξε τελικά στην χειροκίνητη προβολή των απαραίτητων εικόνων στο μοντέλο για κάθε τρίγωνο της επιφάνειας του. Με βάση το σετ δεδομένων που προέκυψε εκπαιδεύτηκε ένας αλγόριθμος προσδιορισμού 6D πόζας (EPOS). Για την εκπαίδευση ακολούθησε πειραματισμός με το μέγεθος των εικόνων αλλά και με τις υπερ-παραμέτρους του μοντέλου. Τα αποτελέσματα της εκπαίδευσης αξιολογήθηκαν ποιοτικά και ποσοτικά στο σετ ελέγχου (validation set) με τον υπολογισμό διάφορων μετρικών για τον υπολογισμό των σφαλμάτων που χρησιμοποιούνται στη σχετική βιβλιογραφία.

Στο τελικό στάδιο της εργασίας, η αποτελεσματικότητα της προτεινόμενης μεθόδου αξιολογείται σε πραγματικές εικόνες οι οποίες ανακτήθηκαν από σειρές video που καταγράφουν τη διαδικασία φόρτωσης/εκφόρτωσης container. Για να λυθεί το πρόβλημα της μεγάλης διαφοράς που παρουσιάζουν τα συνθετικά δεδομένα από τις πραγματικές εικόνες (πχ. δυναμικό φόντο - background) υιοθετήθηκε μία προ-επεξεργασία των εικόνων και συγκεκριμένα η εξαγωγή της 2D μάσκας της αρπάγης και η εφαρμογή διαφόρων μετασχηματισμών συμβατότητας. Για την αξιολόγηση της απόδοσης του μοντέλου ποσοτικά, και σε πραγματικές εικόνες, υπολογίστηκαν τα ίδια σφάλματα όπως στο σετ ελέγχου ενώ για την ποιοτική αξιολόγηση χρησιμοποιήθηκε ο αλγόριθμος για την απόκρυψη των μη ορατών ακμών και η υπέρθεση της εκτιμώμενης πόζας πάνω στην αρχική εικόνα.

Τέλος, η εργασία παραθέτει μια σύνοψη της διαδικασίας, τα συμπεράσματα της επίτευξης ή μη των αρχικών απαιτήσεων αλλά και τις δυνατές διορθώσεις και ιδέες για μελλοντική έρευνα και ανάπτυξη. Μέρος της ερευνητικής εργασίας παρουσιάστηκε στο συνέδριο 8th International Conference on Computer Vision Theory and Applications, Lisbon Portugal,19-22 February 2023 με τίτλο “Crane Spreader Pose Estimation from a Single View”.

National Technical University of Athens, Greece
School of Rural, Surveying and Geoinformatics Engineering

Implementation of a Methodology for Training a Machine Learning Algorithm for 6D Pose Estimation of a Textureless Object

Diploma Thesis
Sapoutzoglou Panagiotis

March, 2023

Abstract

Object detection and 6D object pose estimation are important and fundamental problems in computer vision and photogrammetry, relevant in robotics, augmented reality and smart environments applications due to the rapidly growing automation of recognition, training and security tasks. Photogrammetry and computer vision have a lot in common both being structured by the same fundamental principles of projective geometry. Specifically for object pose estimation, the literature tends to use as input modality RGB and RGB-D images. However, the acquisition of depth information based on active sensors is not always feasible in outdoor environments or in the case of objects with special characteristics (with metallic or semi-transparent surfaces). Furthermore, the installation of multiple sensors (e.g. stereo) to support additional viewpoints to extract 3D information via passive means translates to higher computational cost. Moreover, in such environments, occlusions and background clutter are frequently encountered. The object may be partially or fully occluded, whereas in the case of background clutter other objects and similar-looking distractors may aggravate the problem. To confront with the above, many approaches have been proposed over the years for 6D pose estimation with deep learning techniques showing a prominent role. Even though they are very effective and robust, they are extremely data-driven depending not only on the size of the training data but also on their quality and type. Digitization is currently a trend that is gaining momentum in container logistics, aiming to make related processes more automated, efficient and traceable. Under these premises, this thesis deals with the 6D pose estimation of a crane spreader from a single view.

The thesis starts by providing some important theoretical background and key concepts. This part describes the mathematical modeling of the camera adopted in computer vision, the importance of homogenous coordinates in vector and matrix computations, quaternion representation, as well as key concepts regarding the implementation of 3D computer graphics with OpenGL. Furthermore, key concepts of deep learning are highlighted describing the training process of NNs, their architecture and hyperparameters. The aforementioned part focuses on a better understanding of how NNs and CNNs operate internally as a means to achieve optimal results, being able to interpret possible shortcomings and ways to overcome them.

Next, relative literature is extensively analyzed by categorizing the 6D pose estimation methods and describing the main contributions, advantages, and disadvantages of each category. The recent methods proposed for the pose estimation problem are divided into two main categories: learning-based approaches and classical non-learning-based methods. Learning-based methods exploit CNNs

for feature extraction whilst non-learning methods use conventional 2D feature extractors. Moreover, learning-based approaches model the 6D pose estimation task in different ways with some predicting 2D-3D correspondences and solving the PnP, others by using a finite set of representations of the model and searching for an identical pose in the image (template-based) and others by directly regressing the 6D pose of the object.

In order to train the 6D pose estimation algorithm a training dataset must be obtained. The automatic acquisition of training images and their corresponding ground truth poses is challenging given the object's dimensions and the difficult task of systematically obtaining them. Manual acquisition and annotation of the dataset are both labor-intensive and time-consuming. The above are confronted by creating a photorealistic texture model of the spreader and rendering a synthetic training dataset by systematically sampling RGB images from various viewpoints and distances. In this context, there was experimentation with two different open-source software to obtain the texture of the spreader using images taken from the field with a commodity camera. This experimentation resulted in the manual mapping of the texture by introducing each input image separately and projecting it to the model's faces. Given this photorealistic model of the spreader, a synthetic dataset is constructed using a computer graphics shader pipeline. By the means of this dataset, a state-of-the-art 6D pose estimation algorithm is trained (EPOS). The model is fine-tuned by experimenting with hyperparameters and image size. Training results are quantitatively and qualitatively assessed using various error metrics from related literature.

Finally, the effectiveness of this approach is evaluated using real images acquired from video sequences of the spreader during container loading/unloading operations. In order to overcome the domain gap between real and synthetic data, a methodology consisting of a 2D segmentation task of the spreader, and various transformations is constructed. Besides the quantitative analysis, the results are also assessed visually by rendering the model's predicted pose with the Hidden Line Removal (HLR) algorithm and super-imposing it on the input images.

The thesis concludes by providing a summary of the approach implementation and discusses whether the requirements for the specific application are satisfied and to what extent. Finally, possible improvements to the procedure are discussed as well as interesting ideas for future work.

Part of this work was presented at the 18th International Conference on Computer Vision Theory and Applications, Lisbon Portugal,19-22 February 2023 as a short paper entitled "Crane Spreader Pose Estimation from a Single View".

1 Εισαγωγή

1.1 Περιγραφή προβλήματος

Η γνώση της 3D θέσης ενός αντικειμένου στο χώρο με τη χρήση απεικονίσεων του σε μία εικόνα ή με τη χρήση άλλων οργάνων και αισθητήρων είναι σημαντική σε εφαρμογές ρομποτικής, επαυξημένης πραγματικότητας, παρακολούθησης σε έξυπνα περιβάλλοντα κ.ά. Πολλές διαδικασίες σε διάφορους τομείς της βιομηχανίας οι οποίες αυτοματοποιούνται όλο και περισσότερο τα τελευταία χρόνια βασίζονται σε τέτοια συστήματα για τη διαρκή παρακολούθηση της θέσης αντικειμένων ενδιαφέροντος. Ένας τέτοιος τομέας είναι και αυτός των μεταφορών και της εφοδιαστικής. Τα container αποτελούν παγκοσμίως τον πιο σημαντικό τρόπο μεταφοράς εμπορευμάτων.Η φόρτωση/εκφόρτωση container γίνεται με τη χρήση κατάλληλων μηχανημάτων που επιτρέπουν τη διαχείριση του μεγάλου βάρους αλλά και των μεγάλων σχετικά διαστάσεων τους. Η μεταφορά των container βασίζεται κυρίως σε γερανούς που φέρουν μία αρπάγη (spreader) η οποία κλειδώνει μηχανικά στο πάνω μερός τους. Η ψηφιοποίηση αυτής της διαδικασίας στοχεύει στην αυτοματοποίηση της, την εξασφάλιση της αποδοτικότητας της και στην συγκρότηση μιας ελεγχόμενης προσέγγισης. Με βάση τα παραπάνω ενδιαφέρει ο προσδιορισμός της 6D πόζας της αρπάγης των container κατά της διαδικασία φόρτωσης/εκφόρτωσης. Η αρπάγη αποτελεί ένα μεταλλικό συμμετρικό αντικείμενο χωρίς υφή με μεγάλες διαστάσεις. Αυτά τα χαρακτηριστικά το καθιστούν απαιτητικό αντικείμενο αναφορικά με το 3D εντοπισμό του (tracking). Παράλληλα, το δυναμικό περιβάλλον και οι μη ελεγχόμενες συνθήκες φωτισμού του εξωτερικού περιβάλλοντος συνδράμουν σε μεγαλύτερη πολυπλοκότητα. Αρκετές μέθοδοι 3D tracking έχουν αναπτυχθεί (Lepetit and Fua [2005], Marchand et al. [2016]) οι οποίες όμως συνήθως προϋποθέτουν αρχικοποίηση της πόζας του αντικειμένου γνωστό και ως bootstrapping. Εκτός όμως από την αρχικοποίηση χρειάζονται και περιοδικές διορθώσεις λόγω σφαλμάτων στη διαδικασία της παρακολούθησης. Με την προτεινόμενη μεθοδολογία που ακολουθείται δίνεται η δυνατότητα να αυτοματοποιηθεί η αρχικοποίηση του 3D tracker αλλά και να προσδιοριστεί η 6D πόζα της αρπάγης σε εικόνες όπου ο 3D tracker χάνει το αντικείμενο λόγω έντονων αποκρύψεων ή θορύβου στο περιβάλλον/φόντο (severe occlusion,background clutter). Άλλοι σημαντικοί λόγοι και οφέλη της παρακολούθησης της θέσης της αρπάγης είναι η διευκόλυνση του χειριστή του γερανού στην καλύτερη αντίληψη του χώρου λειτουργίας της και η ασφάλεια των εργατών (Lourakis and Pateraki [2022a]). Ακόμα, δίνεται η δυνατότητα της εξασφάλισης σωστών κινήσεων από τη μεριά του χειριστή του γερανού με το να ελέγχεται αν οι κινήσεις του είναι μέσα στα ασφαλή όρια λειτουργίας. Η ταλάντωση της αρπάγης είναι επίσης ένα σημαντικό πρόβλημα, η οποία με απροσεξία του χειριστή ή έλλειψη εμπειρίας μπορεί να επιφέρει ζημίες ή και ατυχήματα. Τέτοιου είδους ταλαντώσεις μπορεί εν μέρει να αποφεύγονται με εφαρμογή αντίθετων μικρο-κινήσεων για την ισορροπία της (Ngo and Hong [2012]).

1.2 Απαιτήσεις

Η μεθοδολογία που θα αναπτυχθεί υπάγεται σε κάποιους περιορισμούς και υπάρχουν συγκεκριμένες απαιτήσεις. Αρχικά, τα δεδομένα που πρόκειται να χρησιμοποιηθούν να έχουν αποκτηθεί με τη χρήση μίας τυπικής κάμερας δηλαδή RGB εικόνες χωρίς συμπληρωματική πληροφορία από άλλες κάμερες, οι οποίες παρέχουν στερεοκάλυψη, ή αισθητήρες βάθους. Ακόμα, δεν μπορούν να χρησιμοποιηθούν δεδομένα από άλλους μη οπτικούς αισθητήρες (πχ. GNSS) λόγω παρεμβολής του σήματος με άλλα μεταλλικά αντικείμενα. Ο προσδιορισμός της πόζας είναι επιθυμητό να παρέχει μια καλή ισορροπία μεταξύ ακρίβειας και ταχύτητας. Είναι δύσκολο να επιτευχθούν χρόνοι για επεξεργασία σε πραγματικό χρόνο μέσω ανίχνευσης από αναγνώριση (tracking by detection) αλλά επιδιώκεται όσο το δυνατό μικρότερος χρόνος. Όσον αφορά την ακρίβεια, η γωνιακή απόκλιση της αρπάγης δεν

πρέπει να ξεπερνάει τις 5° ενώ το σφάλμα μέσης ευθυγράμμισης (ADD) δεν πρέπει να ξεπερνάει το 10% της διαμέτρου του αντικειμένου που στην περίπτωση της αρπάγης είναι 1.244m (Hinterstoisser et al. [2012]).

1.3 Προσέγγιση επίλυσης

Η μεθοδολογία που αναπτύχθηκε αρχικά δημιουργεί ένα φωτορεαλιστικό μοντέλο της αρπάγης με τη διαδικασία εφαρμογής υφής (texturing) ενός ήδη διαθέσιμου 3D γεωμετρικού μοντέλου της και το πειραματισμό με δύο διαφορετικά λογισμικά (Meshlab και Blender). Στη συνέχεια, αναπτύσσεται ένας renderer με τη χρήση βιβλιοθηκών 3D γραφικών και της OpenGL. Με τη χρήση του renderer και των shaders δημιουργούνται συνθετικές αναπαραστάσεις του φωτορεαλιστικού μοντέλου της αρπάγης αλλά και εικόνες στις οποίες αποκρύπτονται οι μη ορατές ακμές του μοντέλου της σε wireframe mode. Δημιουργείται το σετ δεδομένων χρησιμοποιώντας ένα σφαιρικό σύστημα αναφοράς ώστε να γίνει συστηματική δειγματοληψία συνθετικών RGB εικόνων από διάφορες οπτικές γωνίες και αποστάσεις. Από το σύνολο αυτών των δεδομένων επιλέγονται τυχαία κάποιες εικόνες για την εκπαίδευση και κάποιες για το σετ ελέγχου (validation set). Με τα δεδομένα που προκύπτουν εκπαιδεύεται ένας state-of-the-art αλγόριθμος προσδιορισμού 6D πόζας μετά από αρκετές δοκιμές με τη ρύθμιση των υπερ-παραμέτρων του δικτύου αλλά και τις κατάλληλες διαστάσεις εικόνας. Μετά την εκπαίδευση η απόδοση του μοντέλου αξιολογείται τόσο ποσοτικά με τον υπολογισμό διάφορων μετρικών αλλά και ποιοτικά με την οπτικοποίηση της εκτιμώμενης πόζας σε σχέση με την αντίστοιχη πόζα αναφοράς (ground truth). Το μοντέλο αξιολογείται σε ένα μεγάλο αριθμό εικόνων που προέρχονται από βίντεο της διαδικασίας φόρτωσης/εκφόρτωσης container στο πραγματικό περιβάλλον. Σε αυτό το σημείο, γίνεται εμφανές το πρόβλημα της διαφοράς των συνθετικών δεδομένων (μαύρο background) με τις πραγματικές εικόνες (δυναμικό background). Για την επίλυση του τελευταίου προστίθεται στη μεθοδολογία ένα στάδιο προ-επεξεργασίας των εικόνων με την εφαρμογή ενός αλγορίθμου για το 2D segmentation της αρπάγης και την εξαγωγή μιας μάσκας που προσδιορίζει τη περιοχή της αρπάγης στην εικόνα. Έπειτα, η μάσκα εφαρμόζεται στην εικόνα και με τη χρήση διάφορων μετασχηματισμών εισάγεται στον αλγόριθμο για τον προσδιορισμό της 6D πόζας. Τέλος, αξιολογείται η απόδοση του μοντέλου στις πραγματικές εικόνες ποσοτικά και ποιοτικά για οκτώ σειρές εικόνων.

1.4 Δομή κεφαλαίων

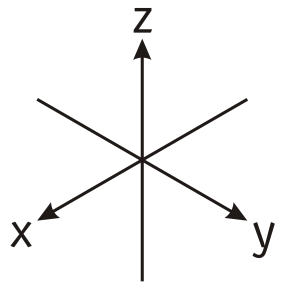
Η εργασία αποτελείται από 6 κεφάλαια. Στο **2 κεφάλαιο** παρουσιάζεται το θεωρητικό υπόβαθρο και οι βοηθητικές γνώσεις που χρειάστηκαν για την διεκπεραίωση τεχνικών ζητημάτων. Στο **3 κεφάλαιο** γίνεται η ανάλυση της βιβλιογραφίας στο θέμα της εκτίμησης της 6D πόζας αντικειμένων. Περιγράφονται οι διάφορες μέθοδοι που έχουν προταθεί, ο οποίες διαχωρίζονται σε κατηγορίες και υποκατηγορίες με διαφορετικά χαρακτηριστικά. Στο **4 κεφάλαιο** περιγράφεται η υλοποίηση της μεθοδολογίας της εφαρμογής υφής (texturing) στο μοντέλο, της απόκρυψης των μη ορατών ακμών σε wireframe mode, η ανάπτυξη αλγορίθμου για τη δημιουργία των συνθετικών δεδομένων αλλά και οι μετρικές που χρησιμοποιούνται για τον υπολογισμό των σφαλμάτων για την αξιολόγηση της εκτιμώμενης πόζας. Στο **5 κεφάλαιο** περιγράφονται τα αποτελέσματα της εκπαίδευσης του αλγορίθμου, η αξιολόγηση των αποτελεσμάτων στο σετ ελέγχου (validation), η προ-επεξεργασία των πραγματικών εικόνων, οι αναγκαίοι μετασχηματισμοί και η αξιολόγηση του μοντέλου σε πραγματικές εικόνες. Τέλος, στο **6 κεφάλαιο** παρουσιάζονται τα συμπεράσματα της εργασίας καθώς και ιδέες για μελλοντική εργασία και έρευνα.

2 Κεφάλαιο 2: Θεωρητικό υπόβαθρο

2.1 Συστήματα και μετασχηματισμοί

2.1.1 Τρισδιάστατο Καρτεσιανό σύστημα συντεταγμένων

Το σύστημα συντεταγμένων του Καρτέσιου στο επίπεδο αποτελείται από δύο προσανατολισμένες ευθείες, κάθετες μεταξύ τους, οι οποίες καλούνται συμβατικά άξονας τετμημένων (οριζόντιος άξονας) και άξονας τεταγμένων (κατακόρυφος άξονας) και συμβολίζονται αντίστοιχα με x και y. Το σημείο όπου τέμνονται λέγεται αρχή του συστήματος συντεταγμένων.



Σχήμα 1: Δεξιόστροφο σύστημα αναφοράς

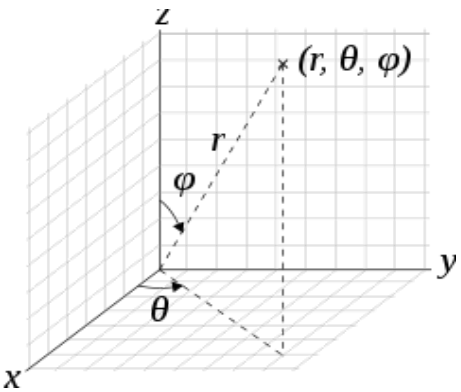
Ένα οποιοδήποτε διάνυσμα μπορεί να γραφτεί ως γραμμικός συνδυασμός d γραμμικά ανεξάρτητων διανυσμάτων, όπου d η διάσταση του χώρου που μελετάμε. Στην περίπτωση των τριών διαστάσεων, μπορούμε να ορίσουμε τα ορθομοναδιαία διανύσματα τα οποία έχουν διεύθυνση κατά τη θετική φορά των αξόνων x, y και z αντίστοιχα. Έχοντας ορίσει την προηγούμενη βάση, μπορούμε να γράψουμε το διάνυσμα θέσης ενός σημείου με συντεταγμένες (x, y, z) στο χώρο με τον εξής τρόπο:

$$\boldsymbol{r} = x\hat{\boldsymbol{i}} + y\hat{\boldsymbol{j}} + z\hat{\boldsymbol{k}}$$

Σε ένα δεξιόστροφο σύστημα συντεταγμένων η σχετική διάταξη των αξόνων ακολουθεί τον κανόνα του δεξιού χεριού: οι άξονες x, y, z έχουν τον προσανατολισμό, αντίστοιχα, του αντίχειρα, του δείκτη και του μέσου του δεξιού χεριού (Σχήμα 1). Αντίθετα, σε ένα αριστερόστροφο σύστημα συντεταγμένων ακολουθείται ανάλογα ο κανόνας του αριστερού χεριού. Στις εφαρμογές Γραφικών και Εικονικής Πραγματικότητας χρησιμοποιούνται κατά περίπτωση τόσο δεξιόστροφα όσο και αριστερόστροφα συστήματα συντεταγμένων.

2.1.2 Σφαιρικό σύστημα αναφοράς

Με την χρήση του καρτεσιανού συστήματος καθώς και των πολικών συντεταγμένων στο επίπεδο μπορεί να οριστεί το τρισδιάστατο σφαιρικό σύστημα αναφοράς (Weisstein [2023]) με παραμέτρους (r, θ, ϕ).



Σχήμα 2: Σφαιρικό σύστημα αναφοράς (Πηγή: Wikipedia [2023])

Το θ είναι το αζιμούθιο το οποίο ορίζεται ως η γωνία στο επίπεδο xy με αρχή τον άξονα x, ϕ η πολική ή ζενίθια γωνία με αρχή των z άξονα και δεξιόστροφη φορά προς στο επίπεδο xy και την ακτίνα r ενός σημείου από την αρχή του συστήματος. Οι σφαιρικές συντεταγμένες εκφράζονται μέσω των καρτεσιανών μέσω των παρακάτω σχέσεων:

$$r = \sqrt{x^2 + y^2 + z^2}, \quad r\epsilon[0, \infty]$$
$$\theta = \tan^{-1}\left(\frac{y}{x}\right), \quad \theta\epsilon[0, 2\pi)$$
$$\phi = \cos^{-1}\left(\frac{z}{r}\right), \quad \phi\epsilon[0, 2\pi]$$

Αντίστοιχα οι καρτεσιανές συντεταγμένες προκύπτουν από

τις σφαιρικές μέσω των εξισώσεων:

$$x = r\cos\theta\sin\phi, \quad y = r\sin\theta\sin\phi, \quad z = r\cos\phi$$

Το διάνυσμα της ακτίνας που αντιπροσωπεύει ένα σημείο στο χώρο μπορεί να εκφραστεί στο καρτεσιανό σύστημα αναφοράς ώς:

$$\vec{\boldsymbol{r}} = \begin{bmatrix} r\cos\theta\sin\phi \\ r\sin\theta\sin\phi \\ r\cos\phi \end{bmatrix} \tag{1}$$

2.1.3 Ομογενείς συντεταγμένες

Στην υπολογιστική όραση οι ομογενής συντεταγμένες χρησιμοποιούνται για την αναπαράσταση σημείων και γραμμών σε μία εικόνα. Το πλεονέκτημα στη χρήση των ομογενών συντεταγμένων έγκειται στο ότι μπορούν να αντιπροσωπευτούν σημεία στο άπειρο και συνεπώς διευκολύνεται η αναπαράσταση ευθειών. Το δεύτερο μεγάλο πλεονέκτημα της χρήσης των ομογενών συντεταγμένων είναι η σημαντική απλοποίηση των υπολογισμών και των μετασχηματισμών αφού μπορούν να αντιπροσωπευτούν από πολλαπλασιασμό πινάκων. Για παράδειγμα, μετασχηματισμοί όπως μετάθεση, στροφή και αλλαγή κλίμακας μπορούν να αντιπροσωπευτούν με ένα πίνακα με τον οποίο πολλαπλασιάζεται ένα διάνυσμα. Αντίθετα, η μετάθεση και η προοπτική προβολή δεν μπορούν να εκφραστούν με την πράξη του πολλαπλασιασμού χρησιμοποιώντας τις καρτεσιανές συντεταγμένες.

Με την χρήση των παραπάνω ιδιοτήτων απλοποιείται σημαντικά η διαχείριση των γεωμετρικών οντοτήτων όπως οι γραμμές και οι ευθείες και μειώνεται παράλληλα το υπολογιστικό κόστος. Για αυτό τον λόγο οι ομογενείς συντεταγμένες χρησιμοποιούνται πολύ συχνά στην αναπαράσταση 3D γραφικών με την χρήση των shaders.

Ένα δισδιάστατο διάνυσμα (x, y) μπορεί να αντιπροσωπευτεί στον προβολικό χώρο ως:

$$\begin{bmatrix} x \\ y \end{bmatrix} \rightarrow \begin{bmatrix} x \\ y \\ w \end{bmatrix}, \epsilon\Re$$

Δηλαδή ένα σημείο στο επίπεδο μπορεί να αντιπροσωπεύεται από άπειρα διανύσματα στο προβολικό χώρο. Στη προβολική γεωμετρία υπάρχει ένα σημείο στο άπειρο που αντιστοιχεί σε κάθε διεύθυνση ευθείας το οποίο δίνεται αριθμητικά από την κλίση της ευθείας. Οι παράλληλες γραμμές θεωρείται ότι τέμνονται στο άπειρο σε ένα σημείο το οποίο αποτελεί την κοινή τους διεύθυνση.Το w ουσιαστικά επιδρά σαν μεταβολή στη κλίμακα του συστήματος συνταγμένων. Για αυτό τον λόγο οι καρτεσιανές συντεταγμένες στο διάνυσμα $(x, y, z, 1)$ παραμένουν αναλλοίωτες αφού οι καρτεσιανές προκύπτουν από τις ομογενείς διαιρώντας με το w (προοπτική διαίρεση - perspective devision).

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} \rightarrow \begin{bmatrix} x/w \\ y/w \\ z/w \end{bmatrix}$$

Αν το $w = 0$ το σημείο που αντιπροσωπεύεται βρίσκεται στο άπειρο ενώ η αρχή του συστήματος είναι το (0,0,0,1).

2.1.4 Εσωτερικό και εξωτερικό γινόμενο διανυσμάτων

Υπάρχουν δύο ορισμοί τόσο για το εσωτερικό όσο και για το εξωτερικό γινόμενο, ένας γεωμετρικός και ένας αλγεβρικός.

Γεωμετρικός ορισμός εσωτερικού γινομένου:

Τα διανύσματα στο χώρο έχουν διεύθυνση και μέτρο. Γεωμετρικά το εσωτερικό γινόμενο δύο διανυσμάτων $\vec{x}, \vec{y}$ ορίζεται ως:

$$\vec{x} \cdot \vec{y} = \|\vec{x}\|\|\vec{y}\| \cos\theta$$

όπου θ η γωνία που σχηματίζουν τα δύο διανύσματα. Τα δύο αυτά διανύσματα είναι φυσικά συνεπίπεδα. Η σχέση αυτή χρησιμεύει κυρίως για τον υπολογισμό της γωνίας που σχηματίζουν τα δύο διανύσματα. Λύνοντας ως προς αυτή είναι:

$$\theta = \left(\arccos \frac{\vec{x} \cdot \vec{y}}{\|\vec{x}\|\|\vec{y}\|}\right)$$

Αλγεβρική έκφραση εσωτερικού γινομένου:

Το εσωτερικό γινόμενο (dot product) δύο διανυσμάτων $\boldsymbol{x} = (x_1, x_2, ..., x_n), \boldsymbol{y} = (y_1, y_2, ..., y_n)$ μπορεί επίσης να εκφραστεί ως ένα άθροισμα:

$$\boldsymbol{x} \cdot \boldsymbol{y} = \sum_{i=1}^{n} x_i y_i = x_1 y_1 + x_2 y_2 + ... + x_n y_n$$

όπου το n συμβολίζει τη διάσταση των διανυσμάτων. Αν τα διανύσματα αναπαριστώνται ως διανύσματα στήλες (column vectors) το εσωτερικό γινόμενο μπορεί να γραφεί και ως πολλαπλασιασμός πινάκων:

$$\boldsymbol{x} \cdot \boldsymbol{y} = \boldsymbol{x}\boldsymbol{y}^{\boldsymbol{T}} = \begin{bmatrix} x_1 \\ x_2 \\ ... \\ x_n \end{bmatrix} \begin{bmatrix} y_1 \; y_2 \; ... \; y_n \end{bmatrix} \tag{2}$$

Γεωμετρικός ορισμός εξωτερικού γινομένου διανυσμάτων:

Ορίζεται στον τρισδιάστατο χώρο και συμβολίζεται $\vec{\boldsymbol{a}} \times \vec{\boldsymbol{b}}$. Το εξωτερικό γινόμενο (cross product) των $\boldsymbol{a}, \boldsymbol{b}$ ορίζει ένα διάνυσμα $\boldsymbol{c}$ το οποίο είναι κάθετο στο επίπεδο που σχηματίζουν τα $\boldsymbol{a}, \boldsymbol{b}$ και η διεύθυνση του καθορίζεται από τον κανόνα του δεξιού χεριού, ενώ το μέτρο του υπολογίζεται με τη μέθοδο του παραλληλογράμμου. Ορίζεται ως:

$$\boldsymbol{a} \times \boldsymbol{b} = \|\boldsymbol{a}\|\|\boldsymbol{b}\| \sin\theta \; \boldsymbol{n} \tag{3}$$

όπου:

- θ: η γωνία που σχηματίζουν τα δύο διανύσματα στο κοινό τους επίπεδο
- $\|\boldsymbol{a}\|$ και $\|\boldsymbol{b}\|$: τα μέτρα των διανυσμάτων $\boldsymbol{a}, \boldsymbol{b}$

- n: ένα μοναδιαίο διάνυσμα με διεύθυνση που προκύπτει από τον κανόνα του δεξιού χεριού

Έκφραση εξωτερικού γινομένου μέσω των μοναδιαίων διανυσμάτων:

Παράλληλα με τον παραπάνω ορισμό το εξωτερικό γινόμενο μπορεί να αντιπροσωπευτεί από την ορίζουσα:

$$\boldsymbol{a} \times \boldsymbol{b} = \begin{vmatrix} \boldsymbol{i} & \boldsymbol{j} & \boldsymbol{k} \\ a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \end{vmatrix} = \begin{vmatrix} a_2 & a_3 \\ b_2 & b_3 \end{vmatrix} \boldsymbol{i} - \begin{vmatrix} a_1 & a_3 \\ b_1 & b_3 \end{vmatrix} \boldsymbol{j} + \begin{vmatrix} a_1 & a_2 \\ b_1 & b_2 \end{vmatrix} \boldsymbol{k} \tag{4}$$

Το εσωτερικό γινόμενο θα χρησιμεύσει στην πορεία για τον υπολογισμό της γωνίας που σχηματίζουν δύο διανύσματα, ενώ το εξωτερικό για την κατασκευή ενός ορθογώνιο συστήματος αναφοράς με τα μοναδιαία διανύσματα των επιθυμητών αξόνων.

2.1.5 Μετασχηματισμοί διανυσμάτων

Με τη χρήση των ομογενών συντεταγμένων είναι δυνατόν να μετασχηματιστούν τόσο το σύστημα του 3D μοντέλου όσο και το σύστημα της κάμερας. Όπως αναφέρθηκε και στην υποενότητα 2.1.3 με τη πράξη του πολλαπλασιασμού μεταξύ πινάκων μπορούμε να πραγματοποιήσουμε εύκολα οποιοδήποτε μετασχηματισμό χρησιμοποιώντας τις ομογενείς συντεταγμένες (πίνακας ή συνδυασμός πολλών πινάκων πχ. στροφής, μετάθεσης). Για να μεταθέσουμε ένα διάνυσμα $\boldsymbol{p} = \begin{bmatrix} p_1, p_2, p_3 \end{bmatrix}$ κατά το διάνυσμα $\boldsymbol{T} = \begin{bmatrix} T_x, T_y, T_z \end{bmatrix}$ αυτό μπορεί να γίνει με τον παρακάτω πολλαπλασιασμό:

$$\boldsymbol{p'} = \left[\begin{array}{c|c} \boldsymbol{I} & \boldsymbol{T} \\ \hline \boldsymbol{0} & 1 \end{array}\right] \boldsymbol{p} = \begin{bmatrix} 1 & 0 & 0 & \boldsymbol{T_x} \\ 0 & 1 & 0 & \boldsymbol{T_y} \\ 0 & 0 & 1 & \boldsymbol{T_z} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ 1 \end{bmatrix} = \begin{bmatrix} p_1 + \boldsymbol{T_x} \\ p_2 + \boldsymbol{T_y} \\ p_3 + \boldsymbol{T_z} \\ 1 \end{bmatrix} \tag{5}$$

Αντίστοιχα για να στρέψουμε το ίδιο διάνυσμα $\boldsymbol{p}$ ως προς κάποιον άξονα και υπό συγκεκριμένη γωνία μπορεί να σχηματιστεί ένας **3x3** πίνακας στροφής και να πολλαπλασιαστεί με το διάνυσμα όπως παρακάτω:

$$\boldsymbol{p'} = \left[\begin{array}{c|c} \boldsymbol{R_{3x3}} & \boldsymbol{0} \\ \hline \boldsymbol{0} & \boldsymbol{1} \end{array}\right] \boldsymbol{p} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & 0 \\ a_{21} & a_{22} & a_{23} & 0 \\ a_{31} & a_{32} & a_{33} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ 1 \end{bmatrix} \tag{6}$$

Για να στραφεί το διάνυσμα **p** γύρω από τον άξονα x αρκεί να πολλαπλασιαστούν οι συντεταγμένες y, z με το πίνακα στροφής $\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$ ή ομοίως με τον αντίστοιχο 4x4 πίνακα στροφής.

$$R_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad R_y = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad R_z = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Στροφή ως προς τον άξονα x | Στροφή ως προς τον άξονα y | Στροφή ως προς τον άξονα z

Για να στραφεί ένα διάνυσμα ως προς ένα άλλο (πχ. ένα ray σε ένα άλλο - κοινό σημείο το πρωτεύον σημείο της κάμερας) μπορεί να υπολογιστεί ο άξονας περιστροφής ως το εξωτερικό γινόμενο των δύο διανυσμάτων. Ακόμα για την απλούστευση των υπολογισμών μπορεί να κανονικοποιηθεί έτσι ώστε να προκύψει ένα μοναδιαίο διάνυσμα αφού το μέτρο του διανύσματος που στρέφεται δεν θα μεταβληθεί.

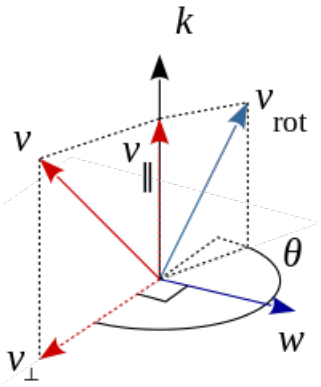
Το διάνυσμα v μπορεί να αναλυθεί στα επιμέρους κάθετα διανύσματα $v_{\parallel}, v_{\perp}$ ως το άθροισμα τους.

$$v = v_{\parallel} + v_{\perp}$$

όπου:

- $v_{\parallel} = (v \cdot \boldsymbol{k})\,\boldsymbol{k}$ η προβολή του διανύσματος **v** στο **k**
- $v_{\perp} = v - v_{\parallel} = v - (v \cdot \boldsymbol{k})\,\boldsymbol{k} = -k \times (k \times v)$

Το διάνυσμα $w = k \times v$ ταυτίζεται με το $v_{\perp}$ αν το τελευταίο στραφεί αριστερόστροφα κατά $90°$ γύρω από το k. Ομοίως το διάνυσμα $k \times (k \times v)$ ταυτίζεται με το $v_{\perp}$ αν το $v_{\perp}$ στραφεί αριστερόστροφα κατά $180°$ γύρω από το k. Η προβολή των διανυσμάτων v και v_{rot} ως προς το k αποτελούν το ίδιο διάνυσμα $v_{\parallel}$. Το μόνο που αλλάζει είναι η κατεύθυνση των διανυσμάτων $v_{\perp}$ και $v_{\perp rot}$ αφού τα μέτρα τους διατηρούνται. Δηλαδή κατά την περιστροφή του διανύσματος v γύρω από το k αλλάζει μόνο η κατεύθυνση της συνιστώσας $v_{\perp rot}$. Το διάνυσμα $v_{\perp rot}$ μπορεί να αναλυθεί σε επιμέρους συνιστώσες ως προς τα διανύσματα $v_{\perp}$ και $\boldsymbol{w} = k \times v$. Επειδή τα διανύσματα $v_{\perp rot}$,$v_{\perp}$,$k \times v$ έχουν ίδιο μήκος είναι:



Σχήμα 3: Περιστροφή διανύσματος γύρω από άξονα

$$v_{\perp rot} = \cos(\theta) v_{\perp} + \sin(\theta) \boldsymbol{k} \times v$$

Επειδή τα διανύσματα k και $v_{\parallel}$ είναι παράλληλα το εξωτερικό τους γινόμενο είναι 0 και ισχύει ότι:

$$k \times v_{\perp} = k \times \left(v - v_{\parallel}\right) = k \times v - k \times v_{\parallel} = k \times v$$

Τελικά το στραμμένο διάνυσμα από την εξίσωση 2.1.5 και 7 γίνεται:

$$v_{\perp rot} = v_{\parallel} + \cos(\theta)\left(v - v_{\parallel}\right) + \sin(\theta) k \times v$$

Τελικά είναι (Liang [2018]):

$$v_{\perp \boldsymbol{rot}} = \cos(\theta) v + (1 - \cos\theta)\,(\boldsymbol{k} \cdot v)\,\boldsymbol{k} + \sin(\theta) \boldsymbol{k} \times v$$

Για να χρησιμοποιηθεί η παραπάνω εξίσωση θα πρέπει να αντιπροσωπευτεί με πράξεις μεταξύ πινάκων και σταθερών μεταβλητών. Το εξωτερικό γινόμενο των διανυσμάτων v, k μπορεί να αναγραφεί με τη μορφή πίνακα ως:

$$\begin{bmatrix} (\boldsymbol{k} \times v)_x \\ (\boldsymbol{k} \times v)_y \\ (\boldsymbol{k} \times v)_z \end{bmatrix} = \begin{bmatrix} k_y v_z - k_z v_y \\ k_z v_x - k_x v_z \\ k_x v_y - k_y v_x \end{bmatrix} = \begin{bmatrix} 0 & -k_z & k_y \\ k_z & 0 & -k_x \\ -k_y & k_x & 0 \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} = [k]_{\times} \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix}$$

όπου $[k]_{\times}$ είναι ο πίνακας του εξωτερικού γινομένου των διανυσμάτων. Από τα παραπάνω προκύπτει ότι:

$$\boldsymbol{K}v = \boldsymbol{k} \times v$$

Πολλαπλασιάζοντας με τον πίνακα $[\boldsymbol{k}]_\times$ από αριστερά προκύπτει:

$$[\boldsymbol{k}]_\times \left([\boldsymbol{k}]_\times v\right) = {[\boldsymbol{k}]_\times}^2 v = \boldsymbol{k} \times [\boldsymbol{k}]_\times v = \boldsymbol{k} \times (\boldsymbol{k} \times v)$$

Από την σχέση 9 και την παραπάνω ισοδυναμία προκύπτει:

$$v_{\perp \boldsymbol{rot}} = v + \sin(\theta)[\boldsymbol{k}]_\times v + (1 - \cos(\theta)) {[\boldsymbol{k}]_\times}^2$$

$$\therefore v_{\boldsymbol{rot}} = \boldsymbol{R}v = \left(\boldsymbol{I} + \sin(\theta)[\boldsymbol{k}]_\times + (1 - \cos(\theta)) {[\boldsymbol{k}]_\times}^2\right) v \tag{7}$$

2.1.6 Τετραδρόνια(quaternions)

Τα τετραδρόνια (Hamiltonian quaternions, Hamilton [1844]) είναι τετραδιάστατες μαθηματικές οντότητες που χρησιμεύουν για την αναπαράσταση της στροφής στον τρισδιάστατο χώρο. Ένα τετραδρόνιο q αποτελείται από τους πραγματικούς αριθμούς s, x, y, z και τα φανταστικά μοναδιαία διανύσματα i, j, k.

$$q = s + ix + jy + kz \ , \ i^2 = j^2 = k^2 = ijk = -1 \tag{8}$$

Στο κεφάλαιο 2.1.5 αναφέρθηκε ότι μια στροφή γύρω απο κάποιο άξονα μπορεί να εκφραστεί με ένα 3x3 πίνακα στροφής R δηλαδή με 9 στοιχεία ενώ με τη χρήση τετραδρονίων ο μετασχηματισμός αυτός είναι εφικτός με τη γνώση 4 στοιχείων. Για δύο διανύσματα $\vec{a}, \vec{b}$ μπορεί να οριστεί ο τελεστής τετραδρονίου q ώστε $\vec{b} = q\vec{a}$ ο οποίος μετασχηματίζει το a (μέτρο και κατεύθυνση) έτσι ώστε να προκύψει το b. Το τετραδρόνιο της εξίσωσης 8 μπορεί να αναλυθεί σε δύο μέρη: ένα σταθερό (scalar) και ένα διανυσματικό (vector).

$$q = \boldsymbol{S}(q) + \boldsymbol{V}(q) = s + (ix + jy + kz) = [s, v] \tag{9}$$

Το κομμάτι που ενδιαφέρει για τα πλαίσια αυτής της διπλωματικής είναι η στροφή στον τρισδιάστατο χώρο. Πολλαπλασιάζοντας ένα τετραδρόνιο με ένα άλλο δεν διατηρείται το μέτρο του διανύσματος του διανυσματικού μέρους του ακόμα και να πρόκειται για το μοναδιαίο τετραδρόνιο ($|q| = 1$). Για να πραγματοποιηθεί ένας μετασχηματισμός μέσω των τετραδρονίων που να διατηρεί τα μήκη χρησιμοποιείται ο συζυγής μιγαδικός του q, q^* (conjugate).

$$q^* = [s, -v] = (s, -ix, -jy, -kz) \tag{10}$$

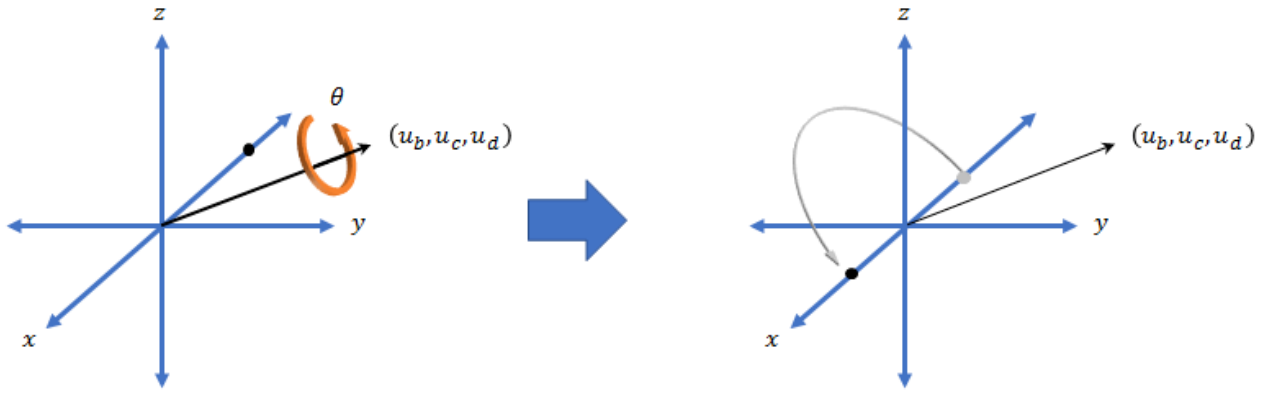
Για να στραφεί ένα τρισδιάστατο διάνυσμα v κατά γωνία θ δημιουργείται ένα τετραδρόνιο για το διάνυσμα $P_q = [0, p_x, p_y, p_z]$, ένα τετραδρόνιο στροφής $q = [\cos\theta/2, n_x \sin\theta/2, n_y \sin\theta/2, n_z \sin\theta/2]$ και το στραμμένο τετραδρόνιο προκύπτει ως:

$$p' = qpq^*$$

όπου: $n = [n_x, n_y, n_z]$ είναι ο άξονας περιστροφής (Dam et al. [2000], Jia [2013]). Ουσιαστικά αυτός ο διπλός πολλαπλασιασμός γίνεται για να ακυρωθεί η αλλαγή στη μήκος του διανύσματος που επιφέρει το qp και για αυτό τον λόγο η γωνία στροφής θ που εκφράζει το τετραδρόνιο είναι η μισή ($\theta/2$). Αν γνωρίζουμε δηλαδή τον άξονα περιστροφής και την επιθυμητή γωνία στροφής μπορούμε να εκφράσουμε

αυτήν την ενέργεια με το τετραδρόνιο $q = [q_0, q_1, q_2, q_3] = [\cos\frac{\theta}{2}, \vec{x}\sin\frac{\theta}{2}, \vec{y}\sin\frac{\theta}{2}, \vec{z}\sin\frac{\theta}{2}]$. Αντίστοιχα ο αντίστροφος μετασχηματισμός μπορεί να γίνει όπως παρακάτω:

$$\begin{cases} \theta = 2\cos^{-1}(q_0) \\ (\vec{x}, \vec{y}, \vec{z}) = \left(\frac{q_1}{\sin\frac{\theta}{2}}, \frac{q_2}{\sin\frac{\theta}{2}}, \frac{q_3}{\sin\frac{\theta}{2}}\right) \end{cases} \tag{11}$$



Σχήμα 4: Στροφή σημείου του $\Re^3$ γύρω από άξονα u κατα γωνία θ η οποία μπορεί να εκφραστεί ως το τετραδρόνιο $q = \cos\frac{\theta}{2} + \sin\frac{\theta}{2}(u_b i + u_c j + u_d k)$ (Πηγή: MATLAB [2022])

Αν και τα τετραδρόνια είναι ενας πολύ αποτελεσματικός τρόπος διαχείρισης της στροφής και της θέσης στον τρισδιάστατο χώρο οι περισσότεροι υπολογισμοί γίνονται με τη χρήση πινάκων. Είναι χρήσιμη λοιπόν, η μετατροπή ενός τετραδρονίου σε ένα 3x3 πίνακα στροφής. Αρχικά κάθε διάνυσμα του $\Re^3$ μπορεί να γραφεί ως ένα τετραδρόνιο με μηδενική τη σταθερή παράμετρο και τους συντελεστές των i,j,k να είναι οι τιμές του διανύσματος στους άξονες x,y,z δηλαδή με τη μορφή $q_o = [0, ix, jy, kz]$. Οι δύο παρακάτω μορφές δίνουν τον αντίστοιχο πίνακα στροφής του τετραδρονίου στροφής (Shoemake [1985]).

$$R = qpq^* = \begin{bmatrix} {q_0}^2 + {q_1}^2 - {q_2}^2 - {q_3}^2 & 2q_1q_2 - 2q_0q_3 & 2q_1q_3 + 2q_0q_2 \\ 2q_1q_2 + 2q_0q_3 & {q_0}^2 - {q_1}^2 + {q_2}^2 - {q_3}^2 & q_2q_3 - 2q_0q_1 \\ 2q_1q_3 - 2q_0q_2 & 2q_2q_3 + 2q_0q_1 & {q_0}^2 - {q_1}^2 - {q_2}^2 + {q_3}^2 \end{bmatrix} \tag{12}$$

ή όταν το q πρόκειται για το μοναδιαίο τετραδρόνιο ($|q| = 1 \Rightarrow {q_0}^2 + {q_1}^2 + {q_2}^2 + {q_3}^2 = 1$) ο πίνακας απλοποιείται σε:

$$R = \begin{bmatrix} 1 - 2{q_2}^2 - 2{q_3}^2 & 2q_1q_2 - 2q_0q_3 & 2q_1q_3 + 2q_0q_2 \\ 2q_1q_2 + 2q_0q_3 & 1 - 2{q_1}^2 - 2{q_3}^2 & q_2q_3 - 2q_0q_1 \\ 2q_1q_3 - 2q_0q_2 & 2q_2q_3 + 2q_0q_1 & 1 - 2{q_1}^2 - 2{q_2}^2 \end{bmatrix} \tag{13}$$

Για τους περισσότερους μετασχηματισμούς χρησιμοποιούνται μοναδιαία τετραδρόνια αφού κυρίως ενδιαφέρει η θέση (κατεύθυνση) ενός διανύσματος και αποφεύγονται με αυτόν τον τρόπο πιθανές αστάθειες στην ακρίβεια των υπολογισμών. Για την μετατροπή ενός τετραδρονίου σε μοναδιαίο τετραδρόνιο αρκεί αυτό να διαιρεθεί με το μέτρο του.

$$\boldsymbol{q} = \frac{q}{||q||} = \left[\frac{s}{||q||}, \frac{x}{||q||}, \frac{y}{||q||}, \frac{z}{||q||}\right]$$

Συγκεκριμένα για τα μοναδιαία τετραδρόνιο ισχύει ότι το αντίστροφο του είναι ίσο με το συζυγές μιγαδικό του ή $q^{-1} = \frac{q^*}{||q||^2} = q^*$ (Jia [2013], Dam et al. [2000]).

2.2 Αρχές υπολογιστικής όρασης

2.2.1 Γεωμετρία και κάμερα κεντρικής προβολής

2.2.1.1 Μοντέλο κεντρικής προβολής

Η κάμερα είναι μια συσκευή η οποία μπορεί να αναπαριστά τον τρισδιάστατο κόσμο σε δύο διαστάσεις (εικόνα). Οι 3D συντεταγμένες του χώρου σχετίζονται με τις 2D απεικονίσεις τους με βάση το μοντέλο της κάμερας σημειακής οπής (pinhole camera) που αντιστοιχεί σε μια κεντρική προβολή. Στόχος είναι η ανάπλαση της μορφής της δέσμης των ακτίνων όπως αυτή υπήρχε κατά την στιγμή της λήψης. Στην φωτογραμμετρία η σχέση που συνδέει τις εικονοσυντεταγμένες ενός σημείου με τις συντεταγμένες του στον τρισδιάστατο χώρο ονομάζεται συνθήκη συγγραμικότητας. Η σχέση αυτή εξασφαλίζει ότι το σημείο στην εικόνα, το προβολικό κέντρο και το αντίστοιχο σημείο στο χώρο ανήκουν στην ίδια ευθεία (Förstner and Wrobel [2016]).

$$\begin{aligned} x &= x_0 - c\frac{r_{11}(X - X_0) + r_{21}(Y - Y_0) + r_{31}(Z - Z_0)}{r_{13}(X - X_0) + r_{23}(X - X_0) + r_{33}(X - X_0)} \\ y &= y_0 - c\frac{r_{12}(X - X_0) + r_{22}(Y - Y_0) + r_{32}(Z - Z_0)}{r_{13}(X - X_0) + r_{23}(X - X_0) + r_{33}(X - X_0)} \end{aligned} \tag{14}$$

όπου:

- r_{ij} τα στοιχεία ενός πίνακα στροφής $R(\omega, \phi, \kappa)$ 3x3
- c η σταθερά της μηχανής η οποία προκύπτει απο την πιο πρόσφατη βαθμονόμηση πριν την λήψη
- x_0, y_0 το πρωτεύων σημείο σημείο της εικόνας(ιδανικά ταυτίζεται με το κέντρο της εικόνας)
- x, y οι εικονοσυνταγμένες του σημείου στην εικόνα
- X, Y, Z οι συντεταγμένες του σημείου στο χώρο
- X_0, Y_0, Z_0 οι συντεταγμένες του κέντρου προβολής στο χώρο

Στις εξισώσεις 14 συμμετέχουν οι παράμετροι c, x_0, y_0 οι οποίες αποτελούν τα στοιχεία του εσωτερικού προσανατολισμού και οι παράμετροι $X_0, Y_0, Z_0, \omega, \phi, \kappa$ που αποτελούν τα στοιχεία του εξωτερικού προσανατολισμού.

2.2.1.2 Παράμετροι εσωτερικού προσανατολισμού (Intrinsic parameters)

Στην υπολογιστική όραση οι παράμετροι του εσωτερικού προσανατολισμού περιγράφονται από έναν άνω τριγωνικό πίνακα K (calibration matrix). Αυτοί οι παράμετροι προσδιορίζουν την εσωτερική γεωμετρία της κάμερας. Εδώ η σταθερά της μηχανής συμβολίζεται με f αλλά εξακολουθεί να αποτελεί τη βαθμονομημένη εστιακή απόσταση.

$$K_{3x3} = \begin{bmatrix} f_x & \gamma & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{15}$$

όπου:

- f_x, f_y οι εστιακές αποστάσεις εκφρασμένες σε pixel στον άξονα x και y. Αυτές προκύπτουν από τις σχέσεις $f_x = F/Px, f_y = F/Py$ όπου F η εστιακή απόσταση σε χιλιοστά και Px, Py το μέγεθος του pixel εκφρασμένο επίσης σε χιλιοστά. Αν το pixel δεν είναι τετράγωνου σχήματος ($f_x = f_y = f$) τότε $f_x \neq f_y$.
- γ η στρέβλωση των αξόνων των pixel. Όταν το σχήμα των pixel δεν αποτελείται αποκλειστικά από ορθές γωνίες.
- u_0, v_0 οι συντεταγμένες του πρωτεύοντος σημείου

2.2.1.3 Παράμετροι εξωτερικού προσανατολισμού (Extrinsic parameters)

Οι παράμετροι του εξωτερικού προσανατολισμού περιγράφουν την θέση της κάμερας στον χώρο (”πόζα κάμερας ή camera pose”) ως προς ένα σύστημα αναφοράς. Η θέση της περιγράφεται από τη μετάθεση της σε σχέση με την αρχή του συστήματος αλλά και τη στροφή της γύρω από τους τρεις άξονες. Ο εξωτερικός προσανατολισμός μπορεί να περιγραφεί με τη χρήση ενός πίνακα στροφής R και ενός διανύσματος μετάθεσης T.

$$[R|T] = \begin{bmatrix} R_{3x3} & T_{3x1} \\ 0_{1x3} & 1 \end{bmatrix}_{4x4} \tag{16}$$

2.2.1.4 Camera matrix

Για να αντιστοιχηθεί ένα 3D σημείο το χώρου στο ομόλογο του σημείο στην εικόνα πρέπει να πολλαπλασιαστεί από αριστερά με το πίνακα που ορίζει το μοντέλο της κάμερας (camera matrix). Αυτός είναι ο συνδυασμός των παραμέτρων του εσωτερικού και εξωτερικού προσανατολισμού μέσω του οποίου οποιοδήποτε σημείο πάνω στην εικόνα μπορεί να μετασχηματιστεί στο ομόλογο του στο χώρο και αντιστρόφως:

$$u \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = P \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = K\,[R|T] \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & \gamma & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & T_1 \\ r_{21} & r_{22} & r_{23} & T_2 \\ r_{31} & r_{32} & r_{33} & T_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{17}$$

ενώ ο αντίστροφος μετασχηματισμός γίνεται ως:

$$\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = P^{-1} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

2.2.2 Computer vision PnP problem

Το πρόβλημα Perspective-n-Point έγκειται στον προσδιορισμό της σχετικής θέσης της κάμερας σε σχέση με ένα αντικείμενο έχοντας ως δεδομένα ομόλογα σημεία στο τρισδιάστατο χώρο του αντικειμένου με τα αντίστοιχα στην προβολή του στην εικόνα (opencv [2022], Zheng et al. [2013]). Αυτό ισοδυναμεί με τον υπολογισμό των παραμέτρων του μοντέλου της κάμερας (camera matrix) όπως αυτό

περιγράφηκε στην ενότητα 2.2.1.4. Είναι ένα θεμελιώδες πρόβλημα που χρησιμοποιείται σε πολλές εφαρμογές στην υπολογιστική όραση και χωρίζεται σε υποκατηγορίες. Ανάλογα με τον αριθμό των δεδομένων ομόλογων σημείων διακρίνονται διαφορετικοί αλγόριθμοι επίλυσης του προβλήματος:

- Τα ομόλογα σημεία είναι παραπάνω απο τέσσερα: το πρόβλημα θεωρείται καλά δομημένο και για τη λύση του υπάρχουν αλγόριθμοι όπως οι:
  - Direct Linear Transform (DLT): Χρησιμοποιεί τη μέθοδο των ελαχίστων τετραγώνων για να προσδιορίσει το camera matrix. Συχνά χρησιμοποιείται στη φωτογραμμετρία για αυτοβαθμονόμηση και υπολογισμό καλών αρχικών παραμέτρων (Abdel-Aziz and Karara [2015]).
  - Linear PnP algorithm (Quan and Lan [1999]): Χρησιμοποιεί τη μέθοδο της διάσπασης ιδιάζουσας τιμής (Singular Value Decomposition (SVD)).
  - Iterative Closest Point (ICP): Σε αντίθεση με τους παραπάνω είναι μη γραμμικός αλγόριθμος και χρησιμοποιείται για να βελτιώσει την προσέγγιση των γραμμικών μεθόδων (Arun et al. [1987]).
  - EPnP (Lepetit et al. [2009])
  - PnP + RANSAC (Fischler and Bolles [1981])
- Τα ομόλογα σημεία είναι ακριβώς τρία. Τότε το σύστημα δεν είναι καλά ορισμένο και δεν υπάρχει μοναδική λύση. Ωστόσο υπάρχουν αλγόριθμοι που χρησιμοποιούνται για την επίλυση του:
  - Grunert's P3P (Grunert [1841])
  - Fischler and Bolles' P3P (Fischler and Bolles [1981])
  - Gao's P3P (Gao et al. [2003])
  - Kneip P3P (Kneip et al. [2011])

2.2.3 Προβολικός μετασχηματισμός - Ομογραφία (Homography)

Ο επίπεδος προβολικός μετασχηματισμός (ομογραφία) απεικονίζει σημεία από ένα επίπεδο σε ένα άλλο (2D-2D). Είναι ένας γραμμικός μετασχηματισμός ο οποίος εφαρμόζεται σε ομογενή τρισδιάστατα διανύσματα και μπορεί να περιγραφεί από έναν 3x3 πίνακα H (Roth [2010], Berkeley [2019]).

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = H \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

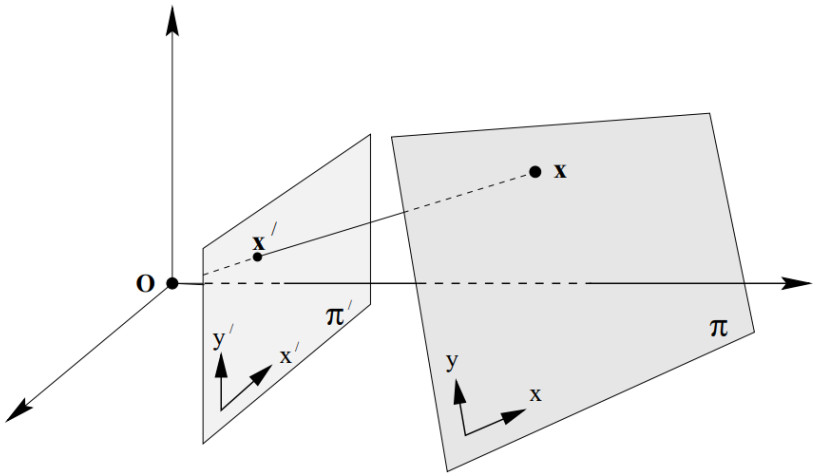
ή

$$x' = Hx \tag{18}$$

2.3 3D αναπαράσταση γραφικών με OpenGL

2.3.1 Γενικά

Η OpenGL αποτελεί ένα Application Programming Interface (API), το οποίο υποστηρίζει διαφορετικές πλατφόρμες (cross platform) αλλά και γλώσσες προγραμματισμού (cross-language). Χρησιμεύει



Σχήμα 5: Κεντρική προβολή ενός σημείου στο επίπεδο σε σημείο σε ένα άλλο επίπεδο (Berkeley [2019])

στην διαχείριση και δημιουργία 2D και 3D γραφικών (Khronos [2023]). Η OpenGL υποστηρίζει την χρήση των καρτών γραφικών (Graphics Processing Unit (GPU)) για πιο γρήγορη και αποτελεσματική επεξεργασία των γραφικών (hardware accelerated rendering). Παρέχει πολλές συναρτήσεις οι οποίες μπορούν να κληθούν από τον χρήστη-προγραμματιστή για την επίτευξη διάφορων λειτουργιών.

Βιβλιοθήκη	Περιγραφή
GLAD	Εύρεση των συναρτήσεων της OpenGL (queried at run-time)
GLFW	Διεπαφή με το χρήστη (input callbacks), δημιουργία παραθύρου κ.α
GLM	OpenGL Mathematics - Μαθηματικές συναρτήσεις, χρήσιμη για τους μετασχηματισμούς και τις πράξεις μεταξύ πινάκων
OpenCV	Computer Vision library, χρήση για επεξεργασία εικόνας
nlohmann json	Εγγραφή αρχείου json

Πίνακας 1: Σύνοψη βιβλιοθηκών που χρησιμοποιήθηκαν

Header file	Περιγραφή
SHADER.h	Custom made βιβλιοθήκη για shader loading, compilation
TrackBall.h	Custom made camera class. Για τη διαχείριση της θέσης και του προσανατολισμού της κάμερας
Model.h	Custom made βιβλιοθήκη για την ανάγνωση αρχείου μοντέλου και την άντληση του περιεχομένου σε επεξεργάσιμη από το υπόλοιπο πρόγραμμα μορφή
Format.h	Custom made βιβλιοθήκη για το format του αρχείου του μοντέλου που κατευθύνει ανάλογα την παραπάνω
Renderer.h	Custom made βιβλιοθήκη για την υλοποίηση χρήσιμων συναρτήσεων που χρησιμοποιούνται συχνά και αποτελούνται από πολλές γραμμές κώδικα(πχ. δημιουργία VAO)

Πίνακας 2: Σύνοψη header files που δημιουργήθηκαν

Σημαντικό είναι ότι η OpenGL έχει ενεργή κοινότητα και νέες εκδόσεις ανακοινώνονται για να συμ-

βαδίζουν με την ανάπτυξη του hardware στους υπολογιστές. Οι κύριες γλώσσες προγραμματισμού που υποστηρίζει είναι η C/C++ και η python με την pyOpenGL. Ακόμα, υποστηρίζεται από πολλά λειτουργικά συστήματα όπως Windows, Linux και Mac. Για αυτή την εργασία λόγω της καλύτερης κατά την τότε στιγμή γνώση σε C++ επιλέχθηκε αυτή η εναλλακτική. Άλλες παρόμοιες εναλλακτικές αποτελούν οι Microsoft DirectX και Vulkan.

2.3.2 Shaders

Οι shaders είναι προγράμματα που εκτελούνται στην GPU και η λειτουργία που επιτελούν είναι να μετασχηματίζουν δεδομένα εισόδου σε δεδομένα εξόδου και από πίνακες και μετασχηματισμούς να καταλήξουν στην απεικόνιση των γραφικών. Είναι απομονωμένοι μεταξύ τους και η μόνη επικοινωνία πραγματοποιείται μέσω των δεδομένων εισόδου και εξόδου τους. Είναι γραμμένοι σε γλώσσα που μοιάζει με C (C-like) και ονομάζεται OpenGL Shading Language (GLSL). Η τελευταία παρέχει χρήσιμες συναρτήσεις για τη διαχείριση διανυσμάτων, πινάκων και πράξεων μεταξύ τους. Τα τρία είδη shader που χρησιμοποιούνται για αυτή την εφαρμογή είναι:

- **Vertex shader:**

  Κάθε μεταβλητή εισόδου λέγεται και vertex attribute ή χαρακτηριστικό κόμβου ο αριθμός των οποίων καθορίζεται από την έκδοση της OpenGL αλλά και τα χαρακτηριστικά του συστήματος του υπολογιστή (hardware). Η GLSL όπως κάθε γλώσσα προγραμματισμού έχει τύπους μεταβλητών όπως int, float, double, uint και bool. Ακόμα παρέχει και δύο containers για διανύσματα (vectors) και πίνακες (matrices).

  Για να οριστούν τα δεδομένα εισόδου και εξόδου χρησιμοποιούνται οι λέξεις κλειδιά **in** και **out**. Είναι πολύ σημαντικό επίσης τα δεδομένα εισόδου ενός shader να ταυτίζονται με τα δεδομένα εξόδου που του μεταβιβάζει ο προηγούμενος shader. Ο vertex shader λαμβάνει τα δεδομένα εισόδου του κατευθείαν από τα δεδομένα των σημείων του μοντέλου.

  Η μεταβλητή τύπου uniform δίνει την δυνατότητα να μεταβιβαστούν δεδομένα εισόδου από το κύριο πρόγραμμα στον shader με βάση το όνομα της μεταβλητής. Έτσι γίνονται οι απαραίτητοι μετασχηματισμοί στον κυρίως κώδικα και έπειτα μεταβιβάζονται στους shaders. Αυτό έχει το μεγάλο πλεονέκτημα ότι γίνεται πολύ πιο εύκολη αποσφαλμάτωση (debugging), ενώ στους shaders είναι δύσκολο να διορθωθούν κυρίως λογικά λάθη και λάθη που έχουν να κάνουν με τις μεταβλητές εισόδου και εξόδου.

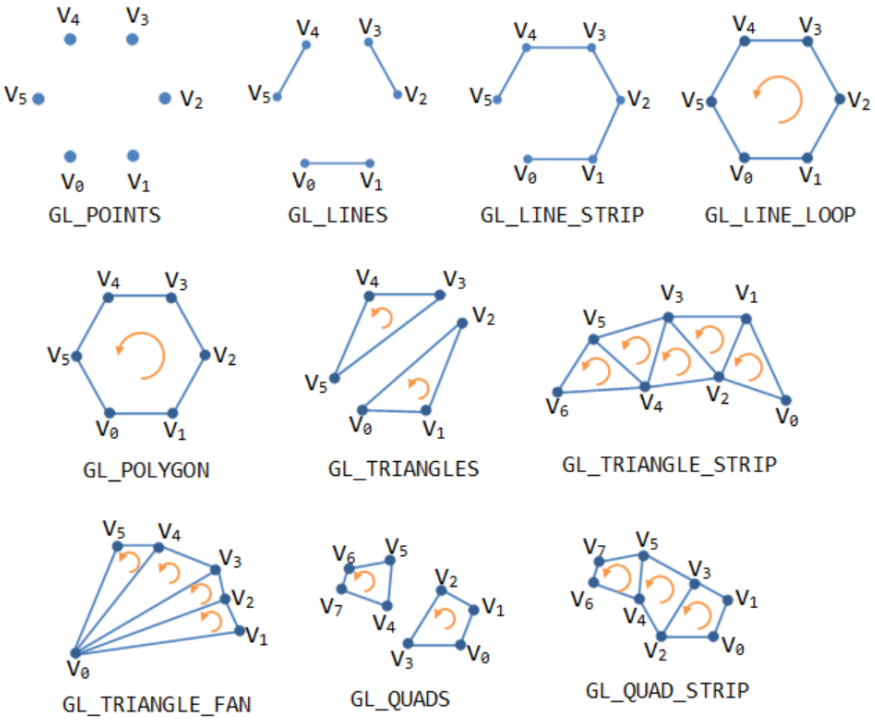
- **Fragment shader**

  Η λειτουργία του fragment shader είναι να δημιουργήσει ένα τελικό χρώμα για κάθε pixel της παραγόμενης εικόνας. Σε αυτή τη φάση είναι δυνατό να υπολογιστεί το χρώμα βάση κάποιας συνάρτησης η και ακόμα μέσω των δεδομένων εισόδου από προηγούμενους shader. Για παράδειγμα θα μπορούσαμε να δημιουργήσουμε έναν fragment shader που το τελικό χρώμα κάθε pixel θα υπολογίζεται με βάση την απόσταση της κάμερας από το αντικείμενο, με άλλα λόγια μια εικόνα που αντιπροσωπεύει το βάθος. Χρησιμοποιήθηκαν τρεις τέτοιοι shader: ένας για την εμφάνιση της υφής (texture) του μοντέλου, ένας για την δημιουργία της εικόνας βάθους και ένας άλλος για τις απλές διαδικασίες όπως το wireframe mode που απλά ορίζεται μόνο το χρώμα των primitives.

- **Geometry shader**

  Μεταξύ του vertex και του fragment shader μπορεί να χρησιμοποιηθεί ένας προαιρετικός shader ο οποίος ονομάζεται geometry shader. Όπως φαίνεται και από την ονομασία του σχετίζεται με

την γεωμετρία των δεδομένων. Η λειτουργία του είναι είναι να δέχεται τις κορυφές από το vertex shader (πχ. δεδομένα του τύπου vertex 1,vertex 2, vertex 3 κ.τ.λ) οι οποίες σχηματίζουν κάποιο γεωμετρικό σχήμα (primitive), πχ. τρίγωνο. Μπορεί να μετασχηματίσει αυτές τις βασικές γεωμετρικές οντότητες (primitives - Σχήμα 6) σε εντελώς νέες, πιο σύνθετες δημιουργώντας επιπλέον κορυφές στα τα αρχικά δεδομένα (πχ. να έχει σαν είσοδο γραμμές και να δημιουργεί αντίστοιχα τρίγωνα). Στην περίπτωση αυτή τα κατάλληλα primitives σχηματίστηκαν με τη χρήση των Vertex Array Object (VAO) και Vertex Buffer Object (VBO) τα οποία περιγράφονται στην ενότητα 2.3.3. Δηλαδή οι geometry shaders που χρησιμοποιήθηκαν είναι pass through δηλαδή δεν επεμβαίνουν στην γεωμετρία των αρχικών primitive. Στα δείγματα κώδικα 4, 5 παραθέτονται δύο παραδείγματα για τους geometry shader που χρησιμοποιούνται για το render με τρίγωνο (triangle primitive) και με γραμμή (line) αντίστοιχα.

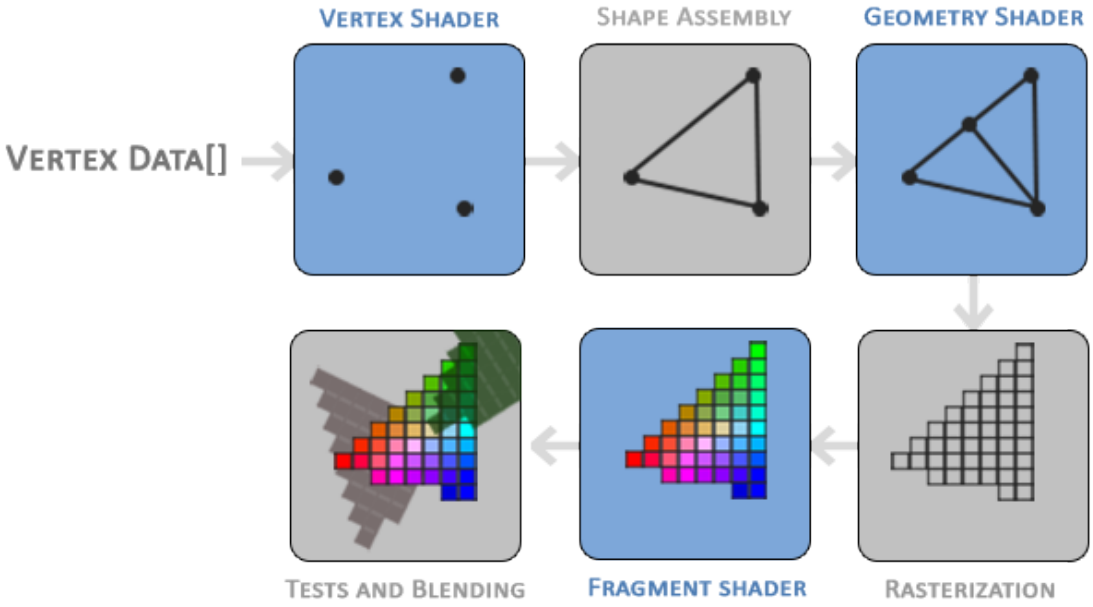


Σχήμα 6: OpenGL primitives (Πηγή: de Vries [2020])

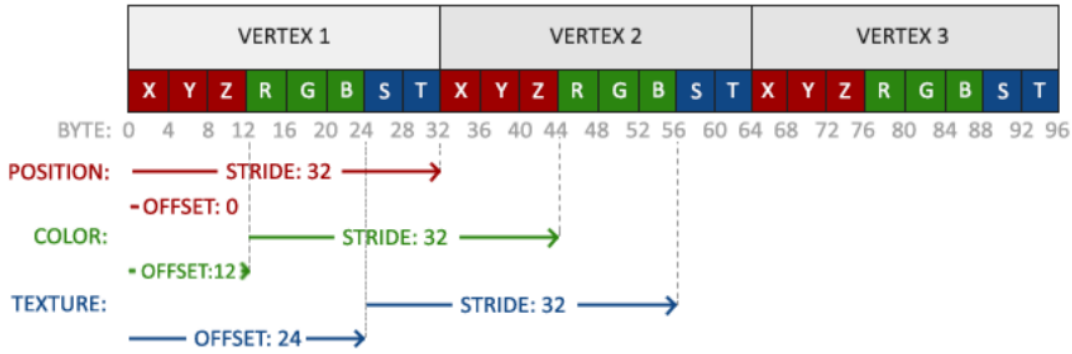
2.3.3 Vertex Buffer Objects (VBOs) και Vertex Array Objects (VAOs)

Ο vertex shader δίνει τη δυνατότητα για μεταβίβαση των πληροφοριών που σχετίζονται με τις κορυφές (vertices) και με τις ιδιότητες τους (vertex attributes). Οι πληροφορίες αυτές αποθηκεύονται σε έναν μέρος προσωρινής μνήμης (buffer) που έχει την μορφή του σχήματος 8. Ο όρος stride ορίζει ανά πόσες θέσεις ακολουθεί το επόμενο χαρακτηριστικό ίδιου τύπου ενώ το offset τη θέση που βρίσκεται το συγκεκριμένο χαρακτηριστικό από την αρχή του buffer.

Για να μεταφερθούν αυτές οι πληροφορίες στον vertex shader στη GPU χρησιμοποιείται ένας buffer που λέγεται Vertex Buffer Object ο οποίος δεσμεύει μνήμη στη κάρτα γραφικών. Για την αξιοποίηση αυτών των πληροφοριών μπορεί να δημιουργηθεί ένα Vertex Array Object το οποίο στην ουσία αποτελεί έναν πίνακα που περιέχει τους δείκτες στις θέσεις μνήμης του VBO. Με αυτόν τον τρόπο μπορεί να δημιουργηθεί το VAO και να χρησιμοποιηθεί αργότερα. Αυτό μας δίνει τη δυνατότητα να έχουμε διαφορετικά VAO με διαφορετικά vertex attributes στα VBO και να χρησιμοποιείται



Σχήμα 7: OpenGL shader pipeline (Πηγή: de Vries [2020])



Σχήμα 8: VBO με Vertex attributes που περιλαμβάνουν τη 3D θέση,το χρώμα και τις συντεταγμένες του texture (Πηγή: de Vries [2020])

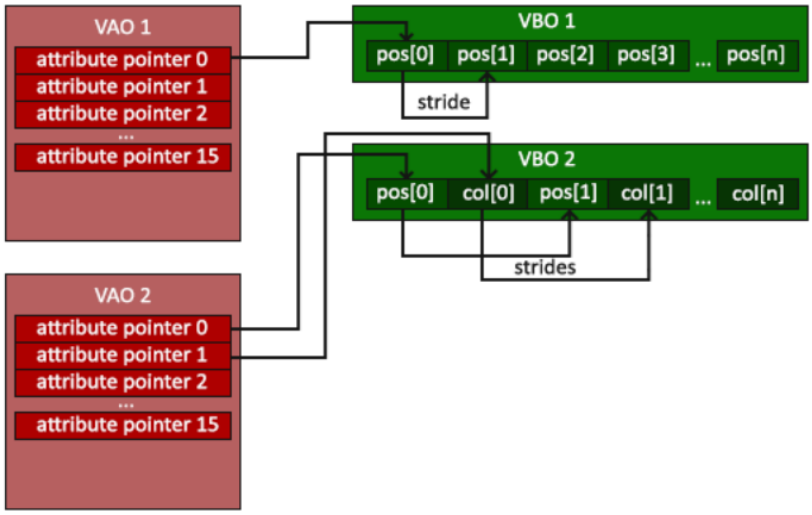
κάποιο από αυτά με μία απλή δέσμευση με το συγκεκριμένο VAO.

2.3.4 Framebuffers

Ένας framebuffer αποτελεί μια συλλογή από buffers (attachments) όπως ο color buffer/texture buffer (που περιέχει πληροφορίες χρώματος) και ο depth buffer (τιμές βάθους σε σχέση με τη κάμερα). Είναι δυνατό να υπάρχουν πολλοί framebuffers με διαφορετικές εσωτερικές παραμέτρους όπως ο τύπος των τιμών τους (float, unsigned char) και μπορούν όπως και με τα VAO να δεσμευτούν (bind) και να αποδεσμευτούν (unbind) για να χρησιμοποιηθούν. Με τη χρήση των framebuffers μπορεί να γίνει το rendering off-screen και έπειτα να διαβιβαστεί στον default framebuffer για την εμφάνιση του.

2.3.4.1 Render σε έναν framebuffer

Αφού δημιουργηθεί ο framebuffer όπως στο δείγμα κώδικα 7 για να μπορέσει να γίνει render στο δημιουργημένο παράθυρο πρέπει να δεσμευτεί ο συγκεκριμένος framebuffer. Σε αυτό το σημείο



Σχήμα 9: Σχέση VAO με VBO (Πηγή: de Vries [2020])

υπάρχει η δυνατότητα να γίνει δέσμευση ολόκληρου του framebuffer ή μόνο τμήματα από αυτόν. Μπορεί να οριστεί ένας framebuffer για πρόσβαση μόνο για reading operations και ένας άλλος για writing operations. Ο framebuffer που έχει δεσμευτεί με τον `GL_DRAW_FRAMEBUFFER` θα είναι υπεύθυνος για το rendering. Συνοπτικά η διαδικασία που ακολουθείται φαίνεται στο σχήμα 10.

2.3.4.2 Ανάκτηση εικόνας από framebuffer

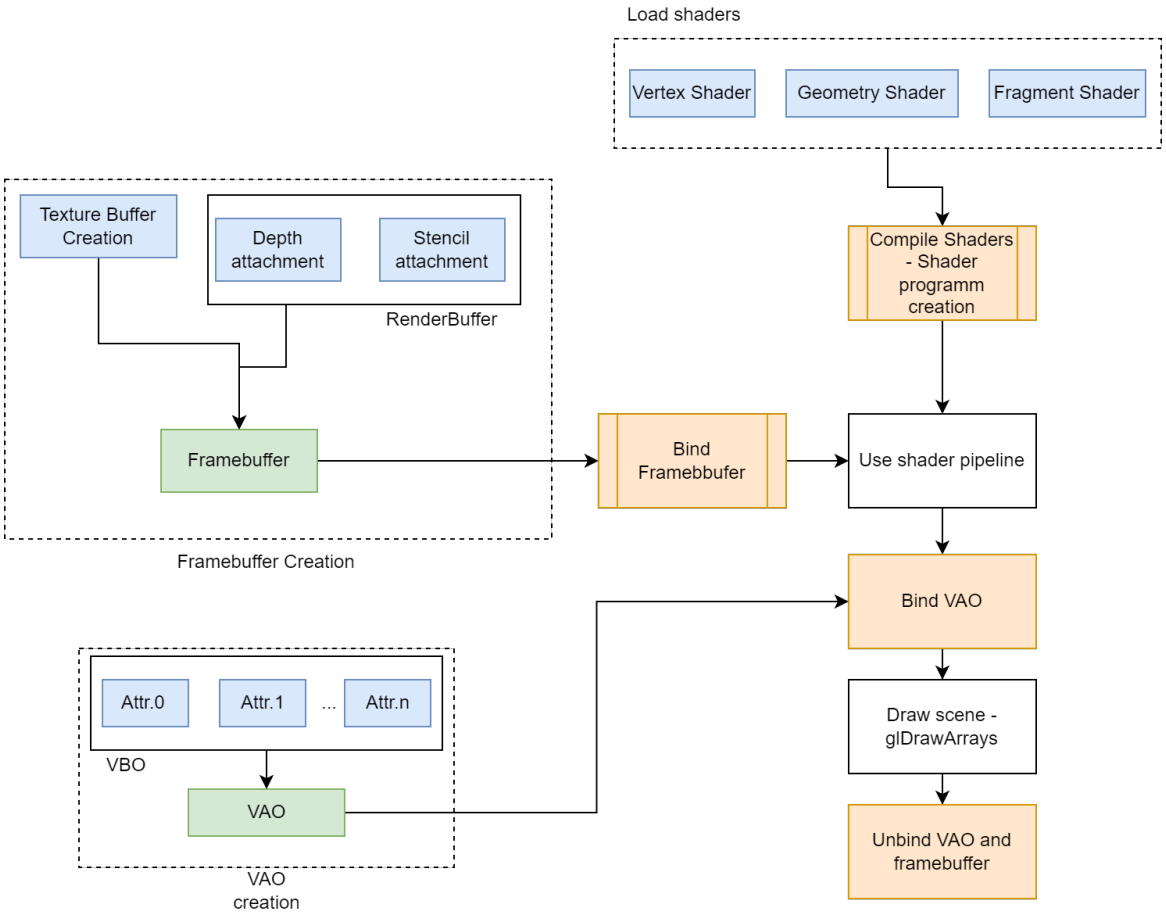
Όπως αναφέρθηκε και στην προηγούμενη ενότητα για να υπάρχει πρόσβαση στα περιεχόμενα του framebuffer πρέπει αυτός να δεσμευθεί ως ο ενεργός framebuffer με τη δυνατότητα διαβάσματος. Αυτό γίνεται με την εντολή `glBindFramebuffer` και τις παραμέτρους `GL_READ_FRAMEBUFFER` και το ID του επιθυμητού Framebuffer. Για να γίνει η ανάγνωση των περιεχομένων του framebuffer χρησιμοποιείται η συνάρτηση `glReadPixels` η οποία αντιγράφει τα περιεχόμενα του buffer σε έναν τοπικό buffer ο οποίος πρέπει να δημιουργηθεί από τον χρήστη (memory allocation). Ωστόσο, δεν αρκεί αυτό διότι ο buffer αποθηκεύει την εικόνα στραμμένη κατακόρυφα. Για αυτό το λόγω δημιουργείται ένας νέος buffer στον οποίο αντιγράφονται κατά γραμμές οι συμμετρικές ως προς τον x άξονα (η τελευταία γραμμή γίνεται πρώτη). Η διαδικασία αυτή γίνεται σύμφωνα με το κώδικα 8.

2.3.5 Συστήματα συντεταγμένων

Η OpenGL προϋποθέτει ότι οι συντεταγμένες όλων των κορυφών που πρόκειται να εμφανιστούν στην οθόνη είναι στο εύρος $(-1, 1)$ (NDC). Συντεταγμένες με τιμές εκτός αυτού του εύρος δεν θα εμφανιστούν στην οθόνη και τα αντίστοιχα ατνικείμενα θα αποκοπούν (clip). Ο μετασχηματισμός αυτός των συντεταγμένων του μοντέλου γίνεται με τη χρήση του vertex shader όπου η διαδικασία μπορεί να πραγματοποιηθεί με διαδοχικά στάδια και μετασχηματισμούς.

Υπάρχουν 5 διαφορετικά συστήματα αναφοράς (Σχήμα 12) τα οποία είναι σημαντικά για την μετατροπή των συντεταγμένων σε Normalized Device Coordinates (NDC) και σε δεύτερο χρόνο στο σύστημα της εικόνας.

- Τοπικό σύστημα ή σύστημα του μοντέλου (Local/Object space): Οι τοπικές συντεταγμένες είναι οι συντεταγμένες του μοντέλου με σημείο αναφοράς την αρχή των αξόνων του.



Σχήμα 10: Η διαδικασία του rendering σε έναν Framebuffer

- World Space: Οι συντεταγμένες με αρχή του συστήματος του γενικότερου 3D περιβάλλοντος (κόσμου) της OpenGL. Σε αυτό το ευρύτερο σύστημα μπορεί να τοποθετηθούν πολλά αντικείμενα σε συντεταγμένες σχετικές με το world origin. Για ένα μοντέλο τα δύο συστήματα αυτά μπορεί να ταυτίζονται. Για να μετασχηματιστούν οι τοπικές συντεταγμένες του μοντέλου στο wold space αρκεί να πολλαπλασιαστούν με ένα πίνακα (local-to-world matrix) δηλαδή:

$$P_{world} = P_{local} * M_{l2w} \tag{19}$$

ενώ για τον αντίθετο μετασχηματισμό είναι:

$$P_{local} = P_{world} * M_{l2w}^{-1} \tag{20}$$

- View space ή Eye space: Το σύστημα στο οποίο τα σημεία εκφράζονται σε σχέση με το σύστημα συντεταγμένων της κάμερας. Για να μετασχηματιστούν οι συντεταγμένες όλων των σημείων στο σύστημα της κάμερας ομοίως εφαρμόζεται ένας πολλαπλασιασμός των συντεταγμένων που αναφέρονται στο world ή local space με έναν πίνακα M_{w2c} (world-to-camera matrix). Αρχικά, η προκαθορισμένη θέση της κάμερας είναι στην αρχή του συστήματος world space και με κατεύθυνση προς τον αρνητικό z άξονα.

$$P_{camera} = P_{world} * M_{w2c} \tag{21}$$

- Clip space ή NDC space: Μετασχηματισμένες συντεταγμένες από το σύστημα της κάμερας (view space) στο εύρος **-1.0** έως **1.0** (συντεταγμένες NDC). Σε αυτό το στάδιο εφαρμόζεται ένας προβολικός μετασχηματισμός με ένα πίνακα M_{prj} με τον οποίο το w κάθε σημείου αυξάνεται ανάλογα με την απόσταση του αντικειμένου από τη κάμερα. Το εύρος των σημείων που θα διατηρηθούν εκφράζεται από τις zfar και znear παραμέτρους (βλ. ενότητα 4.2.3). Μετά τον προβολικό μετασχηματισμό (perspective projection) το εύρος των τιμών των συντεταγμένων είναι από $-w$ έως w και με τη διαίρεση των ομογενών συντεταγμένων (perspective devision) οι τιμές μετασχηματίζονται τελικά στο εύρος -1 έως 1.

$$
\begin{aligned}
P_{prj} &= M_{prj} * M_{w2c} * M_{l2w} \\
P_{canvas}x &= \frac{P_{prj}x}{P_{prj}w} \\
P_{canvas}y &= \frac{P_{prj}y}{P_{prj}w}
\end{aligned}
\tag{22}
$$

- Screen space: Οι NDC συντεταγμένες μετασχηματισμένες σε συντεταγμένες οθόνης οι οποίες μετασχηματίζονται μέσω ενός γραμμικού μετασχηματισμού ανάλογα του μεγέθους του παραθύρου η καλύτερα του framebuffer (Σχήμα 11). Ο μετασχηματισμός αυτός προγραμματιστικά υλοποιείται από τη συνάρτηση `glViewport(0, 0, FBO_WIDTH , FBO_HEIGHT)` όπως φαίνεται και στο δείγμα κώδικα 10. Πρακτικά η συνάρτηση μετασχηματίζει τις NDC σε συντεταγμένες στο σύστημα της οθόνης δηλαδή $p : [-1, 1] \rightarrow [0, F_{width}|F_{hegiht}]$ μέσω των παρακάτω σχέσεων:

$$
\begin{aligned}
P_{screen}x &= (P_{canvas}x + 1) * (\frac{B_w}{2}) + x \\
P_{screen}y &= (P_{canvas}y + 1) * (\frac{B_h}{2}) + y
\end{aligned}
$$

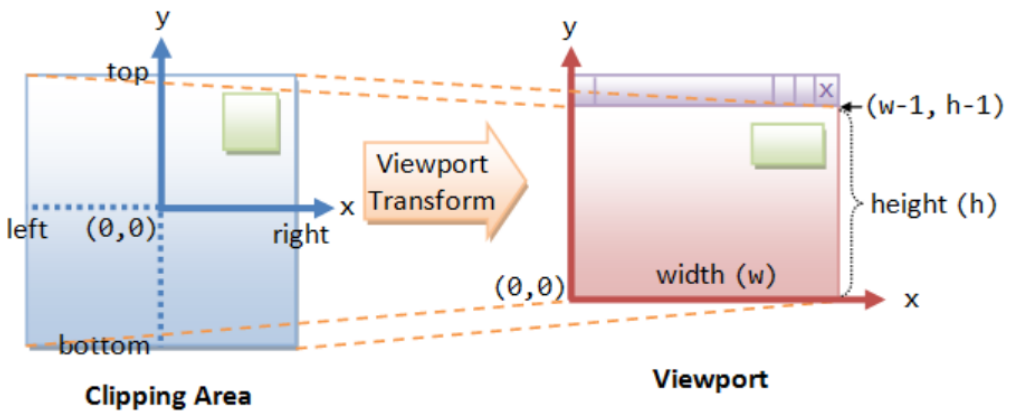
όπου:

  - $P_{canvas}x, P_{canvas}y$ οι NDC συντεταγμένες
  - $P_{screen}x, P_{screen}y$ οι συντεταγμένες στο σύστημα της οθόνης (screen space)
  - (B_w, B_h) είναι οι γραμμές και οι στήλες του χρησιμοποιούμενου Framebuffer
  - x, y Η κάτω γωνία του παραθύρου που θα δημιουργηθεί. Η προεπιλεγμένη τιμή είναι (0,0) δηλαδή η κάτω αριστερά γωνία του Framebuffer.

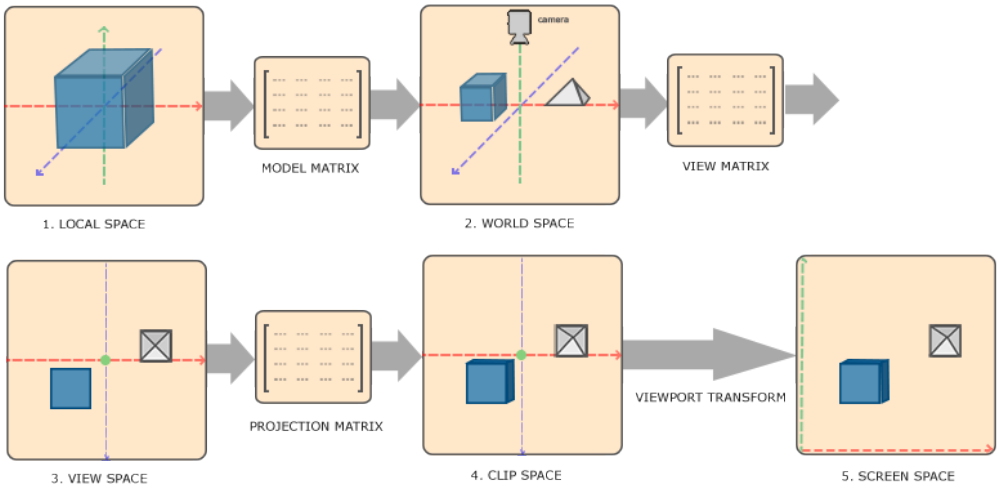
Σε αυτή τη φάση το σύστημα της οθόνης έχει αφετηρία την κάτω αριστερή γωνία της εικόνας. Για την μετάβαση στο σύστημα της εικόνας με αφετηρία την πάνω αριστερή γωνία αρκεί μία μετάθεση κατά τον y άξονα και η στρογγυλοποίηση των τιμών τύπου float στους κοντινότερους ακεραιους οι οποίοι πλέον θα αντιπροσωπεύουν pixel. Τέλος, το χρώμα σε κάθε pixel προκύπτει ανάλογα και με τον fragment shader.

$$
\begin{aligned}
P_{raster}x &= nint(P_{screen}x) \\
P_{raster}y &= nint(B_h - P_{screen}y)
\end{aligned}
$$

όπου $nint()$ η συνάρτηση που μετασχηματίζει τις τιμές στους κοντινότερους ακεραίους.



Σχήμα 11: Αριστερά: Συντεταγμένες NDC. Δεξιά: Το σύστημα της οθόνης(screen space) (Πηγή: Opengl and tutorials [2023])



Σχήμα 12: Συστήματα συντεταγμένων στην OpenGL (Πηγή: de Vries [2020])

2.4 Βαθιά μάθηση (Deep Learning) και έννοιες

2.4.1 Μηχανική μάθηση (Machine Learning)

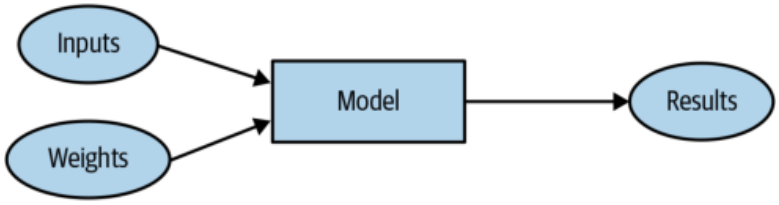
Τα μοντέλα βαθιάς μάθησης (Deep Learning models) χρησιμοποιούν τα νευρωνικά δίκτυα (Neural Network (NN)) και αποτελούν μια υποκατηγορία της Μηχανικής Μάθησης (Machine Learning (ML)). Η μηχανική μάθηση αποτελεί μια διαδικασία με την οποία ένας υπολογιστής “μαθαίνει” να λύνει συγκεκριμένα προβλήματα. Η διαδικασία αυτή διαφέρει από την κλασσική προγραμματιστική προσέγγιση. Για να λυθεί ένα πρόγραμμα προγραμματιστικά με έναν αλγόριθμο αρκεί να αντιπροσωπευτούν οι επιμέρους διαδικασίες και υπολογισμοί προκειμένου να προκύψει το αποτέλεσμα. Μία τέτοια προσέγγιση μπορεί να περιγραφεί συνοπτικά στο διάγραμμα του σχήματος 13.

Σχήμα 13: Η αυστηρά προγραμματιστική προσέγγιση (Πηγή: Howard and Gugger [2020])

Η παραπάνω μοντελοποίηση επίλυσης ενός προβλήματος δεν είναι αποτελεσματική για προβλήματα όπως η αναγνώριση ενός αντικειμένου σε μία εικόνα. Είναι πολύ δύσκολο να μοντελοποιηθεί ένα

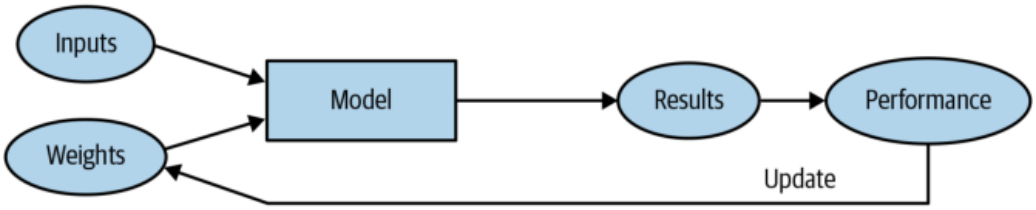
τέτοιο πρόβλημα με μεγάλη διαφοροποίηση στα δεδομένα εισόδου δεδομένου ότι πρέπει να οριστούν πολύ συγκεκριμένα βήματα για την επίλυση του.

Υιοθετήθηκε μία μέθοδος ”εκμάθησης” που επιτρέπει στον υπολογιστεί να προβλέπει το αποτέλεσμα με βάση την εμπειρία του με δεδομένα εκπαίδευσης. Το μοντέλο της εικόνας 14 περιγράφει πολύ γενικά την έννοια της χρήσης βαρών στην διαδικασία επίλυσης. Τα βάρη (weights) είναι μεταβλητές ανάλογα με την τιμή των οποίων αναλύονται τα δεδομένα εισόδου. Η διαφοροποίηση τους οδηγεί σε διαφορετικό τελικό αποτέλεσμα του μοντέλου.



Σχήμα 14: Μοντέλο το οποίο χρησιμοποιεί βάρη για την εξαγωγή του αποτελέσματος (Πηγή: Howard and Gugger [2020])

Για να μπορέσει το μοντέλο να αυτοβελτιώσει τις προβλέψεις του θα πρέπει να υπάρχει ένας μηχανισμός ο οποίος θα αξιολογεί την επίδοση του μοντέλου και θα μεταβάλει τις τιμές των βαρών με σκοπό την ελαχιστοποίηση της διαφοράς του ορθού αποτελέσματος (εκ των προτέρων γνωστό κατά την εκπαίδευση) και τη πρόβλεψης του μοντέλου (loss)(σχήμα 14). Όταν τελικά βρεθούν οι καλύτερες παράμετροι/βάρη του μοντέλου πλέον τις θεωρούμε σταθερές και με αυτές το μοντέλο καλείται να κάνει προβλέψεις σε ένα άλλο σετ δεδομένων που δεν έχει επεξεργαστεί κατά την εκπαίδευση του (test set).



Σχήμα 15: Διαδικασία ελαχιστοποίησης του σφάλματος με την μεταβολή των βαρών (Πηγή: Howard and Gugger [2020])

2.4.2 Νευρωνικά Δίκτυα (Neural Networks)

Ένα νευρωνικό δίκτυο (ΝΔ) αποτελεί μία συνάρτηση με την οποία μπορεί να λυθεί οποιοδήποτε πρόβλημα μεσώ της μεταβολής και της εύρεσης των κατάλληλων βαρών του δικτύου. Σύμφωνα με το **universal approximation theorem** ένα νευρωνικό δίκτυο μπορεί να λύσει οποιοδήποτε πρόβλημα και σε κάθε βαθμό ακρίβειας αρκεί η απόκτηση των βέλτιστων τιμών των παραμέτρων για την συγκεκριμένη εφαρμογή μέσω της εκπαίδευσης του. Ένα ΝΔ γενικά αποτελείται από:

- **Επίπεδα εισόδου (Input Layers)**: Αυτά τα layers αντιπροσωπεύουν τα δεδομένα εισόδου που δέχεται το δίκτυο και μπορεί να είναι περισσότερα από ένα
- **Κρυφά επίπεδα (Hidden layers)**: Είναι συνδετικά layers που συνδέουν τα input layers με τα output layers πραγματοποιώντας μετασχηματισμούς πάνω σε αυτά. Σε αυτό το επίπεδο περιέχονται κόμβοι (nodes). Διαφορετικός αριθμός κόμβων μπορεί να χρησιμοποιηθεί για την πιο πολύπλοκη μοντελοποίηση της αρχιτεκτονικής του δικτύου σε συνδυασμό με την χρήση των

συναρτήσεων ενεργοποίησης (activation functions) που μεταβάλουν τις τιμές των κόμβων των hidden layers.

- **Επίπεδα εξόδου (Output layer)**: Τελικοί κόμβοι που αποτελούν το τελικό αποτέλεσμα της πρόβλεψης του δικτύου. Ο αριθμός των κόμβων εξόδου εξαρτάται από τη φύση του προβλήματος. Για παράδειγμα, αν η το τελικό αποτέλεσμα είναι η εκτίμηση μιας συνεχόμενης μεταβλητής (continuous variable) το output layer θα αποτελείται από έναν κόμβο. Αντίθετα, αν η μεταβλητή υπάγεται σε συγκεκριμένες κατηγορίες (categorical variable) (πχ. ένα pixel ανήκει στο αντικείμενο 1 ή το αντικείμενο 2) οι κόμβοι θα είναι σε αριθμό όσοι οι δυνατές κατηγορίες που μπορεί να ταξινομηθεί η μεταβλητή.

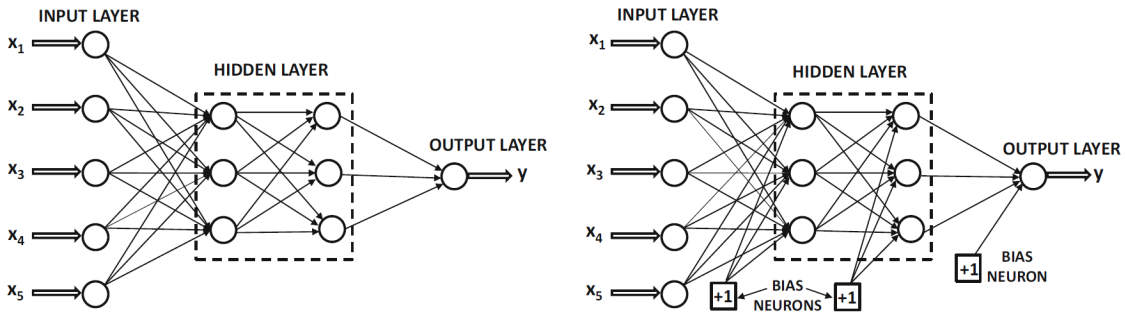
Ένας κόμβος του σχήματος 16 ή ένας νευρώνας δέχεται σαν είσοδο το διάνυσμα $\boldsymbol{x}$ των δεδομένων εισόδου και το μετασχηματίζει με την χρήση του διανύσματος βαρών $\boldsymbol{w}$ των βαρών (Σχήμα 16). Η τελική τιμή του κόμβου διαμορφώνεται αφού εφαρμοστεί μία συνάρτηση που ονομάζεται activation function που είναι ιδιαίτερα χρήσιμη στο να κάνει το μοντέλο μη-γραμμικό και κατ' επέκταση δυνατό να λύσει περίπλοκα μη-γραμμικά προβλήματα. Ο όρος βαθιά μάθηση (Deep Learning (DL)) αναφέρεται στο μεγάλο αριθμό των hidden layers με σκοπό να μοντελοποιηθούν πολύπλοκα προβλήματα όπως η αναγνώριση ενός αντικειμένου σε μια εικόνα.

Η έξοδος y του νευρώνα (post activation) διαμορφώνεται από την σχέση:

$$y = f(w_0 + \sum_{i=1}^{n} w_i x_i)$$

όπου:

- x_i είναι οι μεταβλητές εισόδου
- w_i οι αντίστοιχες τιμές βαρών
- f η συνάρτηση ενεργοποίησης (activation function)
- w_0 μια ποσότητα που λέγεται bias και χρησιμεύει στον να μετατοπίζει τη καμπύλη της συνάρτησης ενεργοποίησης ώστε να εφαρμόζει καλύτερα στα δεδομένα και τη μορφή της προσδιοριζόμενης μεταβλητής



Σχήμα 16: Αρχιτεκτονική ενός ΝΔ με και χωρίς biases (Πηγή: Aggarwal [2018])

2.4.2.1 Activation functions

Η συνάρτηση ενεργοποίησης όπως αναφέρθηκε και στην προηγούμενη ενότητα χρησιμοποιείται για να αποδώσει μη-γραμμικότητας στο μοντέλο. Οι πιο συχνές συναρτήσεις που χρησιμοποιούνται είναι:

- Σιγμοειδής συνάρτηση (Sigmoid activation):

$$Sigmoid(x) = \frac{1}{1 + e^{-x}}$$

- Ανορθωμένη Γραμμική Μονάδα - ReLU (Rectified Linear Unit):

$$ReLU(x) = \begin{cases} x, & x > 0 \\ 0, & x \leq 0 \end{cases}$$

- Συνάρτηση υπερβολικής εφαπτομένης (Hyperbolic tangent):

$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

- Γραμμική ενεργοποίση - Linear activation:

$$LA(x) = x$$

- Κανονικοποιημένη εκθετική συνάρτηση (Normalized exponential function) ή Softmax:

$$softmax(z)_i = \frac{e^{z_i}}{\sum_{i=j}^{n} e^{z_j}}$$

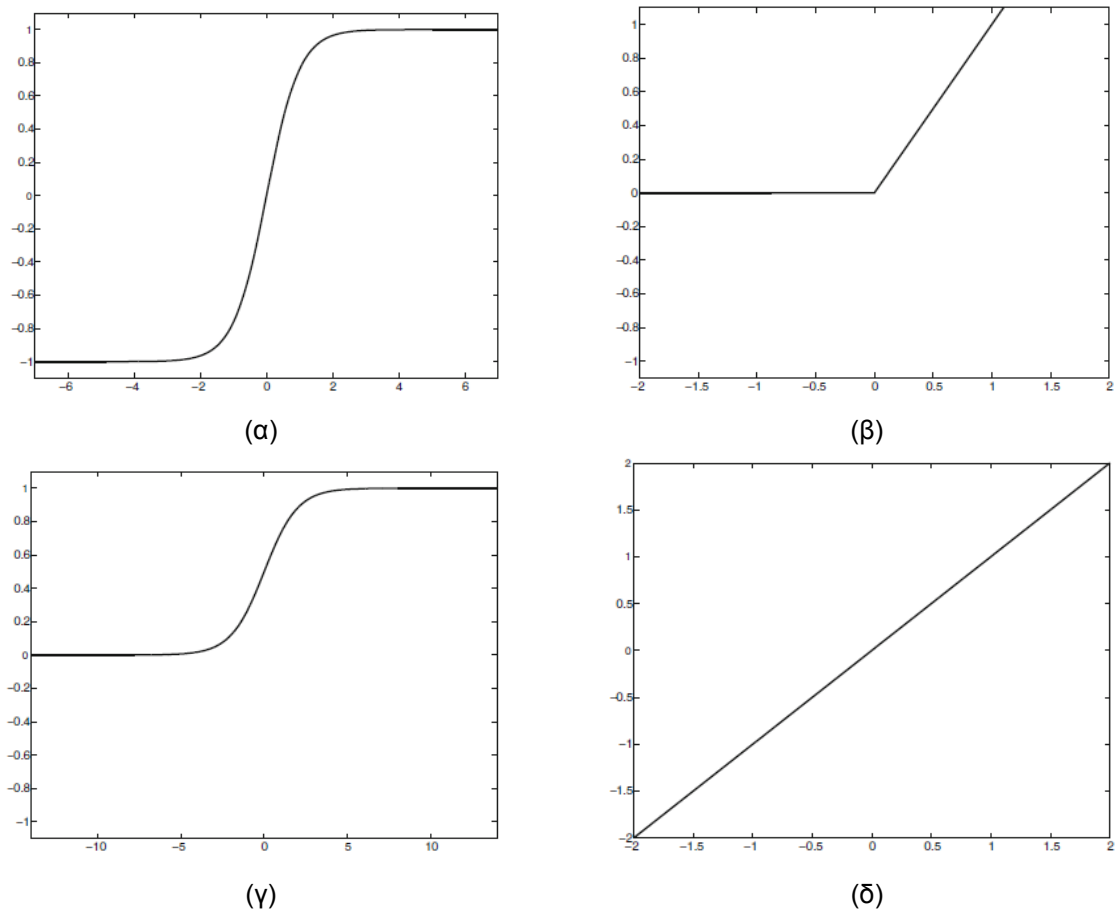
  Η συνάρτηση softmax έχει σαν είσοδο ένα διάνυσμα z των τιμών πριν την ενεργοποίηση των επιπέδων εξόδου (pre-activation raw outputs). Χρησιμοποιείται όταν το ζητούμενο είναι η εκτίμηση της κατηγορίας που ανήκουν τα δεδομένα εισόδου (categorical values). Το i στοιχείο του διανύσματος που δίνει η εφαρμογή της συνάρτησης αντιπροσωπεύει τη προβλεπόμενη πιθανότητα τα δεδομένα εισόδου να ανήκουν στην κατηγορία i. Το e^{x} στον αριθμητή θα μετασχηματίσει όλες τα στοιχεία του διανύσματα σε θετικούς αριθμούς ενώ η κανονικοποίηση που γίνεται με τη διαίρεση μετασχηματίζει τα στοιχεία στο διάστημα 0-1. Αυτό το εύρος ταυτίζεται με την πιθανότητα η είσοδος να ανήκει σε μία κλάση. Με βάση τα παραπάνω για όλες τις πιθανές κατηγορίες ισχύει ότι το άθροισμα των softmax πιθανοτήτων τους έχει σαν άθροισμα 1 ή 100%.

2.4.2.2 Υπερ-παράμετροι (Hyperparameters)

Οι υπερ-παράμετροι ενός ΝΔ είναι οι μεταβλητές που ορίζουν την δομή του όπως ο αριθμός των hidden layers αλλά και που καθορίζουν τη συμπεριφορά του ΝΔ κατά την εκπαίδευση όπως η ταχύτητα εκμάθησης (learning rate).

Hyperparameters που σχετίζονται με τη δομή

- Αριθμός hidden layers: Με την αύξηση των hidden layers και τις κατάλληλες τεχνικές μπορεί να αυξηθεί η ακρίβεια του δικτύου.



(α) (β)

(γ) (δ)

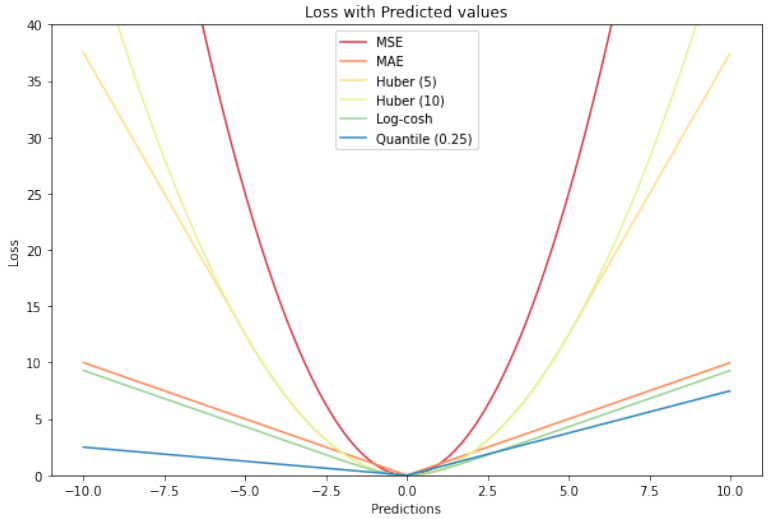
Σχήμα 17: Actiovation functions - (a) Tanh, (β) ReLU (γ) Sigmoid, (δ) Linear (Πηγή: Aggarwal [2018])

- Dropout: Είναι μια τεχνική κανονικοποίησης με σκοπό την αποφυγή του overfitting και κατ' επέκταση την γενίκευση του μοντέλου. Ο όρος dropout αναφέρεται στην προσωρινή απενεργοποίηση κάποιον νευρώνων και την διακοπή input/outputs σε και από αυτούς έτσι ώστε να μην ανανεώνονται τόσο συχνά τα βάρη (βλ. ενότητες 2.4.2.6 και 2.4.2.8).
- Activation functions (βλ. ενότητα 2.4.2.1) Με τις συναρτήσεις ενεργοποίησης προσδιορίζονται οι έξοδοι των νευρώνων με βάση τα δεδομένα εισόδου σε αυτούς. Αποδίδουν μη γραμμικότητα στο μοντέλο και του επιτρέπουν να επιλύσει οποιοδήποτε πρόβλημα.
- Μέθοδος αρχικοποίησης βαρών (weight initialization) Υπάρχουν διάφορες τεχνικές αρχικοποίησης των βαρών. Μία από αυτές είναι η αρχικοποίηση με τη χρήση τυχαίων τιμών. Η τεχνική αυτή έχει αδυναμίες αν τα βάρη που θα προκύψουν είναι πολύ μεγάλες η μικρές τιμές όπως περιγράφεται στην ενότητα 2.4.2.5. Για τον παραπάνω λόγο έχουν αναπτυχθεί νεότερες τεχνικές όπως οι αρχικοποίηση He (He initialization, He et al. [2015]) και αρχικοποίηση Xavier (Xavier initialization, Xavier and Yoshua [2010]). Η πρώτη χρησιμοποιείται με τη ReLU συνάρτηση ενεργοποίησης ενώ η δεύτερη με την $tanh$ (βλ. ενότητα 2.4.2.1). Συγκεκριμένα για την αρχικοποίηση Xavier όλα τα βάρη ενός layer l επιλέγονται τυχαία από μια κανονική κατανομή με μέσο όρο $\mu = 0$ και διασπορά $\sigma^2 = \frac{1}{n^{[l-1]}}$ όπου $n^{[l-1]}$ είναι ο αριθμός των νευρώνων στο layer $l-1$ ενώ τα biases αρχικοποιούνται με τιμή 0.

Hyperparameters που σχετίζονται με τη διαδικασία εκπαίδευσης

- Learning Rate (lr): Το learning rate αποτελεί μία απο τις βασικότερες εκ των υπερ-παραμέτρων. Αυτή η ποσότητα ρυθμίζει το πόσο μικρή η μεγάλη θα είναι η αλλαγή στα βάρη στην κατεύθυνση

που ορίζεται από την παράγωγο της συνάρτησης του σφάλματος (βλ. ενότητα 2.4.2.4). Αν το learning rate είναι πολύ μεγάλο αυτό συνοδεύεται από μεγάλη αλλαγή στην τιμή των βαρών και το μοντέλο μπορεί να φτάσει σε σημείο να αποκλίνει και να μη βρίσκει ποτέ το ελάχιστο. Αντίθετα, αν το learning rate είναι πολύ μικρό η εκπαίδευση μπορεί να πάρει πολύ περισσότερες επαναλήψεις αλλά και να σταθεροποιηθεί σε τοπικό ελάχιστο και όχι ολικό ελάχιστο. Για τους παραπάνω λόγους είναι σημαντικό να οριστεί το κατάλληλο learning rate. Αυτό μπορεί να μην επιτευχθεί με τη πρώτη δοκιμή αλλά μετά από πλήθος δοκιμών και πειραματισμό. Η τιμή του learning rate κατά τη διάρκεια της εκπαίδευσης είναι θεμιτό να μην παραμένει σταθερή. Ορίζεται μία αρχική τιμή (initial learning rate) και κατά τη διάρκεια της εκπαίδευσης το learning rate μειώνεται σταδιακά ακολουθώντας κάποιο πρότυπο (policy) όπως για παράδειγμα πολυωνυμική μείωση.



Σχήμα 18: Γραφικές παραστάσεις διαφόρων συναρτήσεων loss (Πηγή: Grover [2018])

- Momentum: Λειτουργεί όπως η έννοια της αδράνειας στη φυσική. Χρησιμοποιεί την προηγούμενη ανανέωση στη τιμή των βαρών στην επόμενη επανάληψη. Για παράδειγμα αν τα βάρη ανανεώθηκαν κατά δw_1 στην επόμενη επανάληψη θα ανανεωθούν κατά $\delta w_2 + a\delta w_1$ όπου $0 \leq a < 1$ είναι το μέγεθος της ορμής. Η ποσότητα αυτή συμβάλλει στην αποφυγή ολικών ελαχίστων και μειώνει το χρόνο για τη σύγκλιση. Ωστόσο, πρέπει να ρυθμιστεί κατάλληλα η τιμή της ανάλογα και με το learning rate.

- Αριθμός εποχών (epochs): Μία εποχή εκπαίδευσης έχει επιτευχθεί όταν το μοντέλο έχει επεξεργαστεί όλα τα δεδομένα εκπαίδευσης (training dataset - βλ. ενότητα 5.1.1) μία φορά. Ο αριθμός των εποχών αυξάνεται κατά τον πειραματισμό όσο ακόμα μειώνεται το σφάλμα στο validation set (βλ. ενότητα 5.1.2).

- Batch size: Για να υπολογιστεί το loss (βλ. ενότητα 2.4.2.3) και να ανανεωθούν τα βάρη είναι δυνατόν να χρησιμοποιηθεί όλο το dataset (full batch) ή να χωριστεί σε μικρότερα μέρη (mini-batches) ακόμα και για κάθε στοιχείο στο dataset ξεχωριστά (online batch size). Αυτό που χρησιμοποιείται συνήθως είναι τα mini-batches και είναι συνήθως δυνάμεις του 2 όπως 32,64,256. Όσο πιο μικρό είναι το batch size εισάγεται περισσότερες θόρυβος και μεγαλύτερος παράγοντας αβεβαιότητας πάνω στα δεδομένα. Με μεγαλύτερο batch size γίνεται μία κανονικοποίηση του υπολογιζόμενου loss και αποκτάται μια πιο αντικειμενική εικόνα για τη κατεύθυνση που πρέπει

να ανανεωθούν τα βάρη. Ωστόσο, το μέγεθος του batch size εξαρτάται από τις δυνατότητες του hardware αφού απαιτούν μεγαλύτερη δέσμευση μνήμης.

2.4.2.3 Forward propagation - Loss

Σε ένα δίκτυο του σχήματος 16 υπολογίζονται οι τιμές των κόμβων των hidden layers πριν την ενεργοποίηση (pre-activation) και μετά από αυτήν (post-activation). Η τιμή κάθε κόμβου πριν την ενεργοποίηση είναι $a = w_0 + \sum_{i=1}^{n} w_i x_i$ ενώ μετά είναι η $a' = f(a)$. Με αυτόν το τρόπο υπολογίζονται όλες οι τιμές των κόμβων μέχρι τους κόμβους του output layer. Στην ενότητα 2.4.1 αναφέρθηκε ότι πρέπει να υπάρχει ένας μηχανισμός ο οποίος θα αξιολογεί την απόδοση του μοντέλου, θα υπολογίζει τη διαφορά από την επιθυμητή πραγματική τιμή (ground truth) και θα ανανεώνει τα βάρη αναλόγως. Η διαφορά αυτή της πραγματικής τιμής από την πρόβλεψη του μοντέλου ονομάζεται στη διεθνή βιβλιογραφία loss.

- Loss για την πρόβλεψη συνεχόμενη μεταβλητής (continues variables)
  - Mean Squared Error (MSE) ή L2 Loss

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

  - Mean Absolute error (MAE) ή L1 Loss

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

  - Root Mean Squared Error (RMSE)

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

- Loss για την πρόβλεψη μεταβλητών κατηγορίας (categorical variables)
  - Binary cross-entropy: Χρησιμοποιείται όταν το αποτέλεσμα είναι δυαδικό δηλαδή υπάρχουν μόνο δύο δυνατές κατηγορίες.

$$L = -\frac{1}{m}\sum_{i=1}^{m}\sum_{i=1}^{2} y_i \log(p_i) = -\frac{1}{m}\sum_{i=1}^{m}[y_i \log(p(y_i)) + (1 - y_i)\log(1 - p(y_i))]$$

  - Categorical cross-entropy: Για περισσότερες απο δύο πιθανές κατηγορίες:

$$L = -\frac{1}{m}\sum_{j=1}^{C}\sum_{i=1}^{m} y_i \log(p_i), \quad C \geq 2$$

    όπου:
    - y είναι η πραγματική τιμή
    - p η πιθανότητα softmax
    - m το σύνολο των δεδομένων εισόδου
    - C ο αριθμός των πιθανών κλάσεων

2.4.2.4 Back propagation

Με τη διαδικασία του feedforward propagation συνδέονται τα input layers με τα hidden layers και τελικά με το output layer. Στην πρώτη επανάληψη αρχικοποιούνται τα βάρη με τυχαίες τιμές (η με κάποια άλλη μέθοδο) και υπολογίζεται το loss που προκύπτει στην πρώτη επανάληψη. Αφού υπολογιστεί το loss πρέπει να μεταβληθούν αναλόγως τα weights ως προς την κατεύθυνση που ελαχιστοποιείται το loss. Το τελευταίο γίνεται μέσω του υπολογισμού της παραγώγου ως προς τα βάρη για όλα τα layers. Η διαδικασία αυτή ελαχιστοποίησης του loss για μέγεθος batch ίσο με 1 ονομάζεται Stohastic Gradient Descent (SGD). Εκτός από την SGD υπάρχουν και άλλες τεχνικές ελαχιστοποίησης του loss (optimizers) όπως οι Adagrad, Adadelta, RMSprop, Adam.

Η παράγωγος της συνάρτησης του loss ως προς το αντίστοιχο βάρος μπορεί να υπολογιστεί με τη χρήση του κανόνα της αλυσίδας (chain rule)(Aggarwal [2018]).

$$\frac{\partial L}{\partial w_{(h_{r-1},h_r)}} = \frac{\partial L}{\partial \hat{y}} \cdot \left[\frac{\partial \hat{y}}{\partial h_k} \prod_{i=r}^{k-1} \frac{\partial h_{i+1}}{\partial h_i} \right] \frac{\partial h_r}{\partial w_{(h_{r-1},h_r)}}$$

όπου:

- L η συνάρτηση του loss ή cost function
- h_i οι κόμβοι του δικτύου
- $w_{(i-1,i)}$ το βάρος που σχετίζει το κόμβο h_{r-1} με τον h_r

Η νέα τιμή του βάρους είναι η παλιά τιμή μειωμένη κατά της παράγωγο του loss ως προς το βάρος πολλαπλασιασμένη με το learning rate:

$$w'_{(h_{r-1},h_r)} = w_{(h_{r-1},h_r)} - lr \cdot \frac{\partial L}{\partial w_{(h_{r-1},h_r)}}$$

Από την παραπάνω διαδικασία φαίνεται η σημαντικότητα του learning rate αφού η κατάλληλη επιλογή του μπορεί να συμβάλλει σε ταχύτερη σύγκλιση. Από την άλλη μικρή τιμή μπορεί να οδηγήσει σε μεγάλο χρόνο εκπαίδευσης και σε διακυμάνσεις ενώ υψηλή σε πολύ μεγάλες αλλαγές στα βάρη και στην απόκλιση του ο αλγόριθμου.

2.4.2.5 Data scaling

Πριν την έναρξη της εκπαίδευσης είναι σημαντικό τα δεδομένα να μετασχηματίζονται σε ένα σταθερό εύρος τιμών το οποίο να αποτελείται από μικρές τιμές. Για παράδειγμα για μία 8 bit εικόνα ο μετασχηματισμός αυτός θα ήταν από το εύρος [0,255] στο [0,1] (ακέραιο σε πραγματικό εύρος). Αυτός ο μετασχηματισμός γίνεται με την διαίρεση της τιμής κάθε pixel κάθε καναλιού της εικόνας με 255. Η συγκεκριμένη διαδικασία κανονικοποιεί τα δεδομένα εισόδου και τα μετατρέπει σε παρόμοιες αναπαραστάσεις της εικόνας αλλά με τη χρήση μικρότερων τιμών. Αυτή η τεχνική μπορεί να αυξήσει σημαντικά την ακρίβεια του μοντέλου. Αν για παράδειγμα χρησιμοποιείται ως συνάρτηση ενεργοποίησης η σιγμοειδής συνάρτηση, δηλαδή υπολογίζεται η τιμή

$$Sigmoid = \frac{1}{1 + e^{-(Input * Weight)}}$$

αυτή δεν διαφοροποιεί τις τιμές της σημαντικά για μεγάλους αριθμούς των δεδομένων εισόδου και σταθερά βάρη. Αντίθετα, όταν οι τιμές των δεδομένων εισόδου είναι πιο μικρές (0,1) η διαφοροποίηση είναι πιο έντονη. Αυτό συμβαίνει γιατί η εκθετική συνάρτηση για μεγάλους αρνητικούς αριθμούς

(μεγάλη είσοδος * βάρος) προσεγγίζει το 0, με αποτέλεσμα η τιμή της συνάρτησης να είναι 1 (Πίνακας 3).

Input	Weight	Sigmoid
255	0.00001	0.501
255	0.0001	0.506
255	0.001	0.563
255	0.01	0.928
255	0.1	1.000
255	0.2	1.000
255	0.3	1.000
255	0.4	1.000
255	0.5	1.000
255	0.6	1.000
255	0.7	1.000
255	0.8	1.000
255	0.9	1.000
255	1.0	1.000

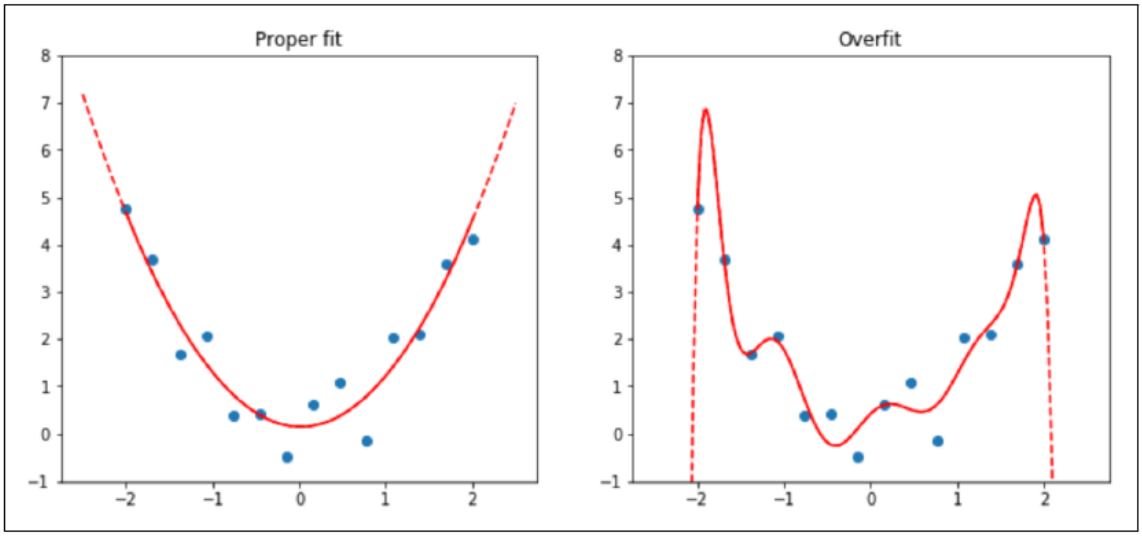
Input	Weight	Sigmoid
1	0.00001	0.500
1	0.0001	0.500
1	0.001	0.500
1	0.01	0.502
1	0.1	0.525
1	0.2	0.550
1	0.3	0.574
1	0.4	0.599
1	0.5	0.622
1	0.6	0.646
1	0.7	0.668
1	0.8	0.690
1	0.9	0.711
1	1.0	0.731

Πίνακας 3: Παράδειγμα των τιμών της σιγμοειδής συνάρτησης οι οποίες δεν αλλάζουν σημαντικά εκτός αν τα βάρη είναι πάρα πολύ μικρά (Πηγή: Ayyadevara and Yeshwanth [2020])

2.4.2.6 Underfitting και Overfitting

Κατά την εκπαίδευση ενός νευρωνικού δικτύου μπορεί το μοντέλο να επιτυγχάνει ακρίβεια εκπαίδευσης μέχρι και 99%. Αυτό δεν σημαίνει απαραίτητα ότι αν του δοθούν λίγο διαφορετικά δεδομένα θα έχει τόσο ακριβή πρόβλεψη. Η έννοιά του overfitting αντικατοπτρίζει το γεγονός ότι το μοντέλο έχει προσαρμοστεί τόσο αυστηρά στα δεδομένα εκπαίδευσης που είναι αδύνατο να γενικεύσει τις προβλέψεις του σε καινούργια δεδομένα. Αντίστοιχα, το underfitting συμβαίνει όταν το μοντέλο δεν αποδίδει καλά κατά την διαδικασία της εκπαίδευσης και κατ' επέκταση είναι σχεδόν σίγουρο ότι τα αποτελέσματα του σε καινούργιες εικόνες θα είναι πολύ πλησιέστερα στα τυχαία και όχι ακριβή. Γενικά, αυξάνοντας τα δεδομένα εκπαίδευσης αυξάνεται και η ικανότητα γενίκευσης του μοντέλου ενώ αυξάνοντας την πολυπλοκότητα του μοντέλου αυτή μειώνεται. Για αυτό το λόγο μοντέλα με πολλές παραμέτρους χαρακτηρίζονται ως high capacity δηλαδή ότι απαιτούν μεγάλο αριθμό δεδομένων εκπαίδευσης για να γενικεύσουν τις προβλέψεις τους σε νέα δεδομένα.

Το overfitting μπορεί να ανιχνευθεί όταν ενώ το σφάλμα κατά την εκπαίδευση μειώνεται το σφάλμα του validation αρχίζει και αυξάνεται. Αυτό εξυπηρετεί η παραλληλότητα του training με το validation διότι κάποιος μπορεί να παρακολουθεί τις αντίστοιχες τιμές του loss και να σταματήσει την εκπαίδευση αν κάτι τέτοιο αρχίζει να συμβαίνει. Κάποιες πλατφόρμες έχουν ενσωματώσει συναρτήσεις που παρακολουθούν αυτά τα σφάλματα και αν υπερβούν κάποια συγκεκριμένη τιμή σταματάνε τη διαδικασία της εκπαίδευσης. Στην αποφυγή ή καθυστέρηση του overfitting (επιτρέπει στο δίκτυο να εκπαιδευτεί για περισσότερες εποχές πριν εμφανιστεί overfitting) συμβάλει σε μεγάλο βαθμό το data augmentation (βλ. ενότητα 2.4.3.5) , η κανονικοποίηση (regularization - βλ. ενότητα 2.4.2.7) και η τεχνική του dropout (βλ. ενότητα 2.4.2.2).



Σχήμα 19: Παράδειγμα overfitting δεδομένων στη συνάρτηση $f(x) = x^2$ (Πηγή: Howard and Gugger [2020])

2.4.2.7 Κανονικοποίηση (Regularization)

Εκτός από την επίδραση που έχει το overffiting στην ακρίβεια που επιτυγχάνει το μοντέλο έχει την τάση να ευνοεί μεγάλες τιμές για κάποια βάρη. Πολύ μεγάλες τιμές βαρών αποτελούν μία ένδειξη ότι το μοντέλο αρχίζει να προσαρμόζεται αυστηρά στα δεδομένα. Σε αυτή τη περίπτωση συμβαίνει κάτι ανάλογο με αυτό που περιγράφηκε στην υποενότητα 2.4.2.5 μόνο που τώρα τα βάρη έχουν τις πολύ υψηλές τιμές. Για την κανονικοποίηση των βαρών ενσωματώνεται στην συνάρτηση του loss ένα παράγοντας ποινής για το μοντέλο όταν τα βάρη του έχουν πολύ μεγάλες τιμές. Οι δύο τύποι κανονικοποίησης που χρησιμοποιούνται συνήθως είναι οι:

- Κανονικοποίηση L1 (L1 Regularization)

$$L1 = Loss + \Lambda \sum_{j=1}^{m} w_i$$

όπου το Loss μπορεί να είναι της μορφής $MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$ (MSE- συνεχόμενη μεταβλητή) είτε της μορφής $L = -\frac{1}{m}\sum_{j=1}^{C}\sum_{i=1}^{m} y_i \log(p_i)$ (Categorical crossentropy - μεταβλητή κατηγορίας) είτε άλλης μορφής. Ο συντελεστής Λ είναι ένας σταθερός συντελεστής που η τιμή του καθορίζει τη μείωση στις τιμές των βαρών. Όσο υψηλότερη τιμή έχει το Λ τόσο μεγαλύτερη είναι και η ποινή και κατ' επέκταση η μείωση των βαρών.

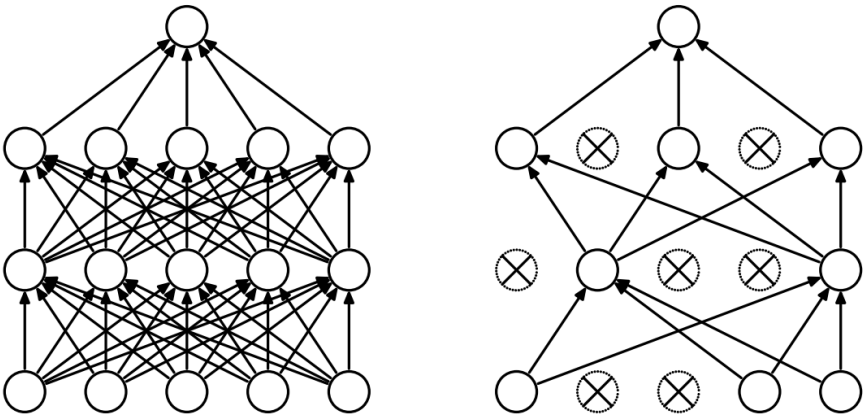
- Κανονικοποίηση L2 (L2 Regularization ή Weight Decay)

$$L1 = Loss + \Lambda \sum_{j=1}^{m} {w_i}^2$$

Σε αυτή τη περίπτωση η ποινή που επιβάλλεται είναι ανάλογη του αθροίσματος των τετραγώνων των τιμών των βαρών και για αυτό αναφέρεται και ως weight decay διότι μειώνει αρκετά τις τιμές των βαρών προς το 0. Για τον παραπάνω λόγω πρέπει να ρυθμιστεί κατάλληλα η παράμετρος Λ.

2.4.2.8 Dropout

Ο όρος drop-out αναφέρεται στην απενεργοποίηση κάποιων κόμβων του δικτύου (input και hidden layers). Όλες οι ενώσεις από και προς αυτούς τους κόμβους διακόπτονται προσωρινά δημιουργώντας ουσιαστικά μια νέα αρχιτεκτονική. Οι κόμβοι που απενεργοποιούνται καθορίζονται με μία παράμετρο πιθανότητας p. Για παράδειγμα dropout 20% σημαίνει ότι 20% των κόμβων θα απενεργοποιηθούν τυχαία δηλαδή η τιμή τους θα γίνει 0. Με αυτό το τρόπο η τεχνική αυτή ενσωματώνει θόρυβο τόσο στα δεδομένα εισόδου (σε μικρότερο βαθμό) όσο και στα κρυφά layers. Πιο συγκεκριμένα, αποτρέπει ένα φαινόμενο που αναφέρεται ως συν-προσαρμογή χαρακτηριστικών (feature co-adaptation) (Srivastava et al. [2014]). Με λίγα λόγια, κατά τη διαδικασία της εκπαίδευσης κάποιοι νευρώνες μπορεί να αντισταθμίζουν τα λάθη των υπόλοιπων κάτι το οποίο συνάδει με το overfitting. Απενεργοποιώντας νευρώνες τυχαία αποτρέπεται το τελευταίο από το να συμβαίνει σε μεγάλο βαθμό και οδηγεί σε ακριβέστερα αποτελέσματα.



(a) Standard Neural Net (b) After applying dropout.

Σχήμα 20: Αριστερά: Ένα τυπικό νευρωνικό δίκτυο με δύο κρυφά layers. Δεξιά: Το νευρωνικό δίκτυο με την εφαρμογή της τεχνικής του Dropout (Πηγή: Srivastava et al. [2014])

2.4.2.9 Batch Normalization

Εκτός από την αρνητική επίδραση των μεγάλων τιμών στα δεδομένα εισόδου κάτι ανάλογο ισχύει και όταν αυτά έχουν πολύ μικρές τιμές. Για πολύ μικρή τιμή των δεδομένων για να μεταβληθεί το αποτέλεσμα της συνάρτησης ενεργοποίησης απαιτείται πολύ μεγάλη αλλαγή στη τιμή του βάρους(gradient). Το τελευταίο οδηγεί σε πολύ μεγάλη τιμή των βαρών κάτι το οποίο όπως διατυπώθηκε και παραπάνω πρέπει να αποφεύγεται. Ακόμα, εκτός από τις μικρές τιμές στα δεδομένα εισόδου αυτό το πρόβλημα μπορεί να εμφανιστεί στα ενδιάμεσα κρυφά layers όταν κάποιος πολλαπλασιασμός μεταξύ της εισόδου του νευρώνα και του αντίστοιχου βάρους καταλήξει σε πολύ μεγάλη τιμή. Για να μειωθεί αυτό το πρόβλημα της εκτοξευόμενης παραγώγου της συνάρτησης του loss (exploding gradient problem) χρησιμοποιείται πάλι μία τεχνική κανονικοποίησης η οποία πραγματοποιείται σε κάθε κόμβο/νευρώνα. Δηλαδή προστίθενται ενδιάμεσα layers που αντισταθμίζουν αυτό το πρόβλημα με το να εισάγουν δύο ακόμα παραμέτρους β_i και γ_i οι οποίες ρυθμίζουν το μέτρο της κανονικοποίησης. Σημαντικό είναι ότι οι παράμετροι αυτοί αφήνονται στο μοντέλο να τις προσδιορίσει μέσω της εκπαίδευσης. Υπάρχουν δύο επιλογές για την εφαρμογή αυτής της τεχνικής:

- Η κανονικοποίηση να πραγματοποιηθεί μετά την ενεργοποίηση (post-activation).

- Η κανονικοποίηση να πραγματοποιηθεί πριν την ενεργοποίηση αλλά μετά τον πολλαπλασιασμό της εισόδου με τα βάρη (pre-activation)

Σύμφωνα με τη σχετική βιβλιογραφία η δεύτερη μέθοδος παρουσιάζει περισσότερα πλεονεκτήματα (Ioffe and Szegedy [2015]). Για να πραγματοποιηθεί η κανονικοποίηση υπολογίζονται αρχικά στατιστικά μεγέθη όπως ο μέσος όρος και η διασπορά του batch (Ayyadevara and Yeshwanth [2020]):

$$\mu_B = \frac{1}{m}\sum_{i=1}^{m} x_i$$

$$\sigma_B{}^2 = \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu_B)^2$$

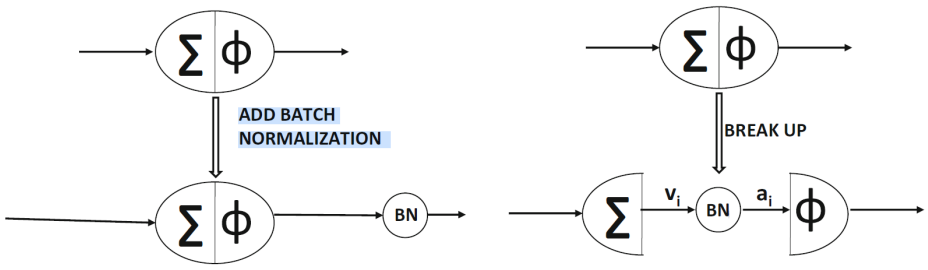
$$\bar{x_i} = \frac{(x_i - \mu_B)}{\sqrt{sigma_B{}^2 + \epsilon}}$$

$$\alpha_i = \gamma_i \cdot \bar{x_i} + \beta_i$$

όπου:

- μ_B ο μέσος όρος του batch
- $\sigma_B{}^2$ η διασπορά του batch
- $\bar{x_i}$ η κανονικοποιημένη τιμή πριν την κανονικοποίηση με τους παραμέτρους β_i και γ_i
- α_i η τελική τιμή μετά την κανονικοποίηση με τους παραμέτρους β_i και γ_i πριν όμως την εφαρμογή της συνάρτησης ενεργοποίησης

Με το να αφαιρείται από κάθε είσοδο ο μέσος όρος και έπειτα να γίνεται η διαίρεση με την τυπική απόκλιση μετασχηματίζονται όλα τα δεδομένα σε ένα σταθερό εύρος. Αν και αυτή η τεχνική αποτελεί αυστηρή κανονικοποίηση (hard normalization) με τον συνδυασμό των παραμέτρων β_i και γ_i το δίκτυο υπολογίζει τις καλύτερες παραμέτρους για την κανονικοποίηση.



(a) Post-activation normalization (b) Pre-activation normalization

Σχήμα 21: Batch Normalization (Πηγή: Aggarwal [2018])

2.4.3 Συνελικτικά Νευρωνικά δίκτυα (Convolutional Neural Network (CNN))

Τα συνελικτικά νευρωνικά δίκτυα είναι βαθιά δίκτυα μάθησης που χρησιμοποιούνται όταν τα δεδομένα εισόδου είναι εικόνες. Βασικά προβλήματα στην όραση υπολογιστών είναι η αναγνώριση προτύπων και αντικειμένων. Για να εξαχθούν τέτοια χαρακτηριστικά (features) από μία εικόνα το μοντέλο πρώτα πρέπει να μάθει να αναγνωρίζει πιο χαμηλού επιπέδου χαρακτηριστικά (low level features). Για να επιτευχθεί αυτό χρησιμοποιείται η διαδικασία της συνέλιξης.

2.4.3.1 Συνέλιξη (Convolution)

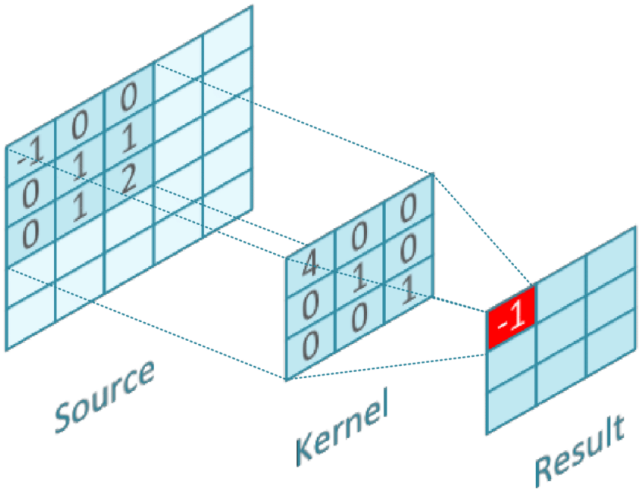
Η συνέλιξη εκφράζει έναν πολλαπλασιασμό μεταξύ δύο πινάκων. Χρησιμοποιείται ένας πίνακας που λέγεται kernel (μικρού μεγέθους σε σχέση με την εικόνα) και γίνεται προσπέλαση της εικόνας με αυτόν. Η νέα τιμή ενός εικονοστοιχείου υπολογίζεται με τη σχέση:

$$y(m,n) = x(m,n) * h(m,n) = \sum_{j=-\infty}^{\infty} \sum_{i=-\infty}^{\infty} x(i,j) \cdot h(m-i, n-j) \tag{23}$$

όπου:

- $[m, n]$ ο αριθμός στηλών και γραμμών αντίστοιχα
- $x(m, n)$ το υποσύνολο της εικόνας που γίνεται συνέλιξη
- $x(m, n)$ το φίλτρο η ο πίνακας συνέλιξης

Στην παραπάνω σχέση η αρχή του συστήματος αρίθμησης του φίλτρου h(0,0) βρίσκεται στο κέντρο του ενώ η αρίθμηση του υποσυνόλου της εικόνας που με το οποίο γίνεται η συνέλιξη είναι από πάνω αριστερά (x(0,0)).



Σχήμα 22: Συνέλιξη με 3x3 πίνακα συνέλιξης (Πηγή: Wicht [2018])

2.4.3.2 Φίλτρα (Filters)

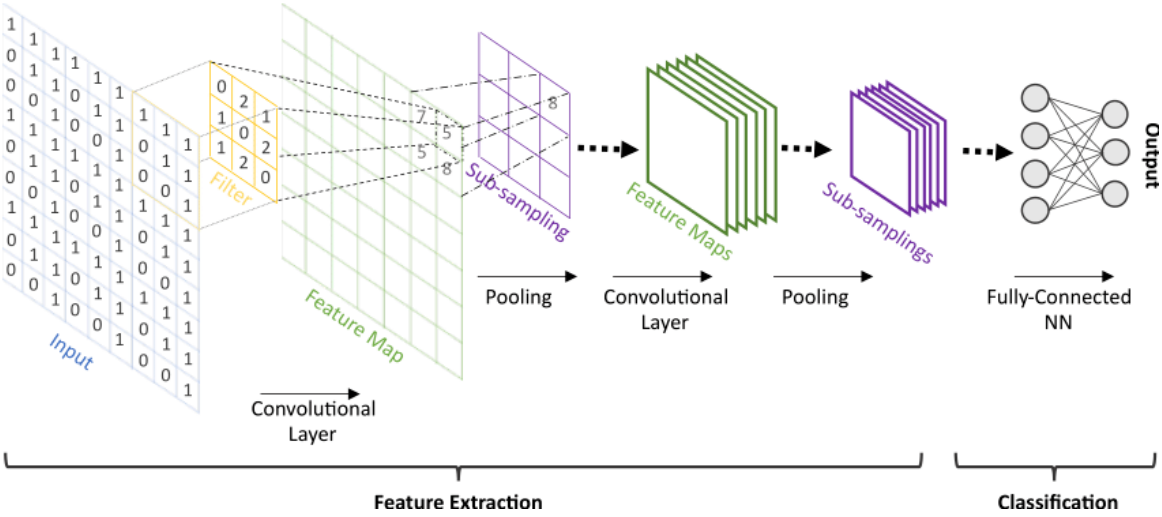
Τα φίλτρα είναι πίνακες από βάρη τα οποία αρχικοποιούνται τυχαία στην αρχή. Το μοντέλο μέσα απο τη διαδικασία της εκπαίδευσης σχηματίζει την καλύτερη τιμή των βαρών κάθε φίλτρου. Θεωρητικά με την αύξηση των φίλτρων αυξάνεται ο αριθμός των χαρακτηριστικών (features) που μπορεί να μάθει το δίκτυο. Στα αρχικά layers το μοντέλο μέσω της συνέλιξης εμφανίζει υψηλές ενεργοποιήσεις σε low level features ενώ σε πιο προχωρημένα στάδια συνέλιξης και επεξεργασίας αρχίζει να μαθαίνει πιο σύνθετα χαρακτηριστικά. Η διάσταση της εικόνας που προκύπτει μετά από την συνέλιξη μπορεί να υπολογιστεί ως:

$$n_{out} = \left[\frac{n_{in} + 2p - k}{s} \right] + 1 \tag{24}$$

όπου:

- n_{in}: η διάσταση της εικόνας (H ή W)
- n_{out}: η διάσταση μετά την συνέλιξη
- k: η διάσταση του φίλτρου
- p: padding συνέλιξης στην αντίστοιχη διάσταση
- s: stride συνέλιξης στην αντίστοιχη διάσταση

Για παράδειγμα για μία εικόνα εισόδου διάστασης $H * W * 3$ θα χρησιμοποιηθούν φίλτρα με 3 κανάλια (conv3d ή seperable conv2d). Ο αριθμός των τελευταίων μπορεί να είναι 32,64,128,512 κ.τ.λ. Το αποτέλεσμα της συνέλιξης H*W*3 με τα φίλτρα N*w*h*3 αποτελεί ένα feature map βάθους όσο ο αριθμός των φίλτρων(N) και διαστάσεων όπως αυτές προκύπτουν από τη σχέση 24.



Σχήμα 23: Οι διαδοχικές συνελίξεις και poolings αποτελούν το κομμάτι του δικτύου που εξάγει χαρακτηριστικά (feature extraction) το οποίο λέγεται και backbone. Το τελευταίο μέρος του δικτύου είναι υπεύθυνο για την ταξινόμηση (classification) ή/και τη παλινδρόμηση (regression) το οποίο λέγεται και head. (Πηγή: Cunha et al. [2022])

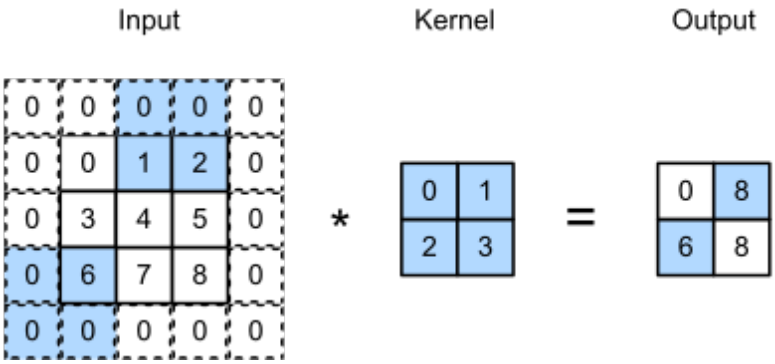
2.4.3.3 Strides και Padding

Η έννοια του stride αναφέρεται στο βήμα του πίνακα συνέλιξης κατά την προσπέλαση ενός πίνακα. Η χρήση τιμής stride μεγαλύτερη του 1 μπορεί να χρησιμοποιηθεί για downsampling της εικόνας ή κάποιου feaure map έτσι ώστε να διατηρηθούν μόνο high level features ενώ να περιοριστούν low level features. Ακόμα γίνονται λιγότεροι υπολογισμοί οπότε απαιτείται λιγότερη υπολογιστική ισχύς.

Κατά τη συνέλιξη ενός πίνακα με ένα φίλτρο δεν μπορούν να υπολογιστούν οι τιμές για τα ακριανά στοιχεία του πίνακα αφού δεν υπάρχουν δεδομένα για να γίνει ο πολλαπλασιασμός ένα προς ένα. Αυτό έχει ως αποτέλεσμα ο πίνακας που προκύπτει να έχει μικρότερες διαστάσεις από τον αρχικό. Το τελευταίο μπορεί να αποτελεί μειονέκτημα για κάποιες τεχνικές όπως η πρόσθεση του αποτελέσματος στον αρχικό πίνακα (residual addition). Η έννοια του padding επιλύει αυτό ο πρόβλημα με το να γεμίζει το εξωτερικό του πίνακα με κάποια τιμή (συνήθως 0).

2.4.3.4 Pooling

Το pooling είναι μία μέθοδος που χρησιμοποιείται για να μειώσει τη διάσταση των input feature maps και να προκύψει μια γενίκευση του feature map. Με αυτό τον τρόπο το δίκτυο μπορεί να γίνει



Σχήμα 24: Padding κατά μία γραμμή και μία στήλη με 0, συνέλιξη με stride 3 (height) και 2 (width) (Πηγή: Padding [2020])

ανεξάρτητο της θέσης των χαρακτηριστικών στην αρχική εικόνα αφού θα δημιουργείται μία μικρότερη έκδοση του χάρτη χαρακτηριστικών. Ακόμα, εκτός από τις συναρτήσεις ενεργοποίησης προσθέτει και έναν επιπλέον μη γραμμικό παράγοντα. Οι δύο κύριες μέθοδοι που χρησιμοποιούνται είναι το max pooling και το average pooling με το πρώτο να είναι πολύ πιο σύνηθες. Στο max pooling ενός χάρτη χαρακτηριστικών (feature map) εξάγεται το max των τιμών μίας υποπεριοχής με κάποιο stride ενώ στο average η μέση τιμή.

Αν χρησιμοποιηθεί τιμή stride 1 και N*N max pooling ο πίνακας που θα προκύψει μετά την εφαρμογή του είναι μεγέθους $(H - N + 1) \times (W - N + 1) \times f$. Συνήθως όμως χρησιμοποιείται stride μεγαλύτερο του 1 οπότε το μέγεθος προκύπτει ως

$$(\frac{(H - N)}{S_h} + 1) \times (\frac{(W - N)}{S_w} + 1) \times f$$

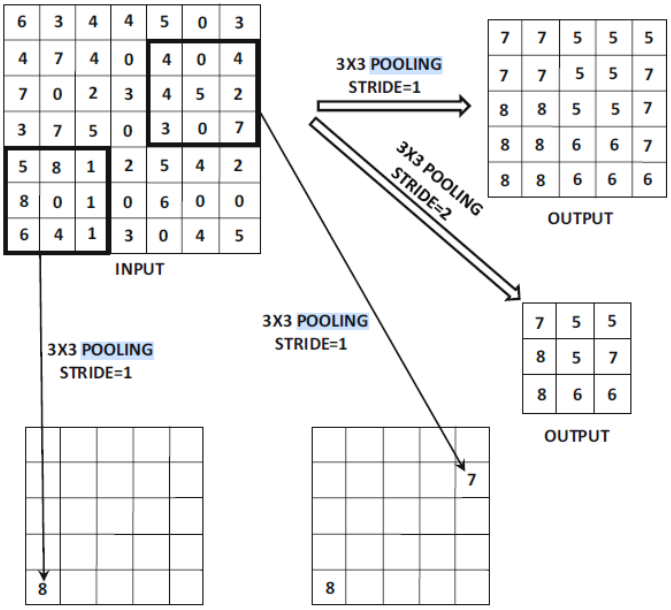
όπου:

- H, W γραμμές,στήλες του πίνακα εισόδου
- N η διάσταση του παραθύρου που θα πραγματοποιηθεί το pooling
- S_h, S_w το βήμα (stride) κατά γραμμές και στήλες
- f ο αριθμός των feature maps

2.4.3.5 Data augmentation

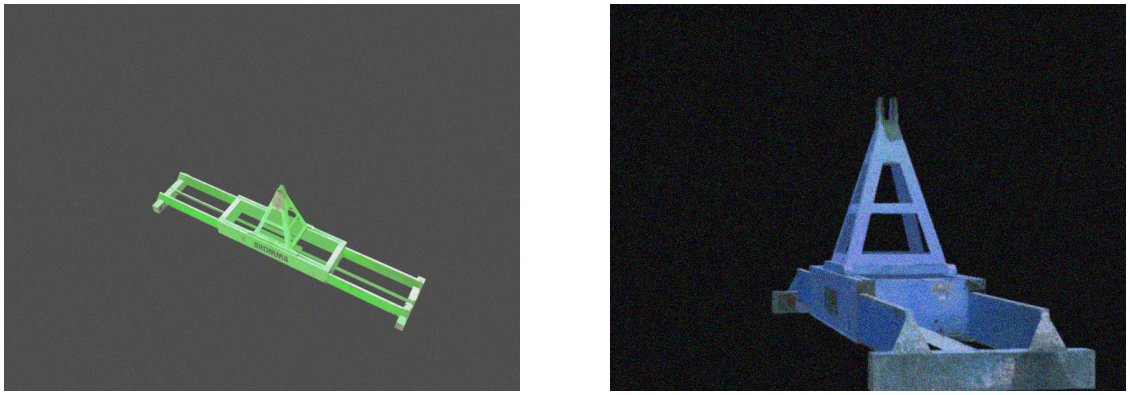
Στην πραγματικότητα οι εικόνες για τις οποίες θα κληθεί το μοντέλο να κάνει προβλέψεις δεν θα είναι ακριβώς όπως αυτές με τις οποίες εκπαιδεύτηκε. Για να γενικευθεί το μοντέλο και η πρόβλεψη να γίνει όσο το δυνατόν ανεξάρτητη από αλλαγές στις συνθήκες όπως ο φωτισμός, η μετάθεση, η στροφή του αντικειμένου στην εικόνα και ο θόρυβος ακολουθείται μια διαδικασία που λέγεται image augmentation και ενισχύει την ικανότητα του μοντέλου να προσαρμόζεται. Τέτοιου τύπου μετασχηματισμοί των δεδομένων εισόδου είναι:

- Αφινικοί μετασχηματισμοί (Affine transformations) που περιλαμβάνουν στροφή,μετάθεση, αλλαγή κλίμακας
- Εισαγωγή θορύβου όπως salt and pepper noise
- Θόλωμα



Σχήμα 25: Max pooling με stride 1 και 2 (Πηγή: Aggarwal [2018])

- Τυχαία μεταβολή στην απόχρωση, στην αντίθεση, στη φωτεινότητα και στο κορεσμό



Σχήμα 26: Image augmentation σε εικόνες εκπαίδευσης του αντικειμένου της αρπάγης. Τυχαία μεταβολή σε φωτεινότητα, αντίθεση, θόρυβο, θόλωμα, απόχρωση και κορεσμό.

3 Κεφάλαιο 3: Ανάλυση βιβλιογραφίας στην εξαγωγή της 6D πόζας αντικειμένων

3.1 Εισαγωγή

Ο εντοπισμός της 6D πόζας αντικειμένων και γενικότερα ο εντοπισμός αντικειμένων σε εικόνες είναι ευρέως διαδεδομένα ζητήματα στην κοινότητα της όρασης υπολογιστών. Η εξαγωγή της 6D πόζας ενός αντικειμένου αποτελεί πιο σύνθετη διαδικασία από τον 2D εντοπισμό του (object segmentation). Αυτό οφείλεται στην έλλειψη βάθους σε δεδομένα 2D RGB εικόνων. Οι πιο πρόσφατες προσεγγίσεις του προβλήματός εξετάζουν την χρήση της βαθιάς μάθησης η οποία έχει επιφέρει σημαντικές βελτιώσεις στο αντικείμενο. Με την ανάπτυξη των δικτύων βαθιάς μάθησης η εκτίμηση της 6D πόζας αντικειμένων χρησιμοποιείται σε εφαρμογές ρομποτικής, αυτόνομης οδήγησης (autonomus driving)

και επαυξημένης πραγματικότητας (augmented reality). Υπάρχουν επιστημονικές εργασίες που ταξινομούν τον τεράστιο όγκο των μεθόδων που έχουν αναπτυχθεί στο αντικείμενο της εκτίμησης της 6D πόζας αντικειμένων όπως οι Kim and Hwang [2021], He et al. [2021], Sahin and Kim [2018], Rahman et al. [2019]

Προκειμένου να αντιμετωπιστεί το πρόβλημα έλλειψης της πληροφορίας του βάθους χρησιμοποιούνται μέθοδοι όπως η χρήση πολλαπλών εικόνων από διαφορετικές θέσεις (Labbe et al. [2020], Sun et al. [2022]), η υιοθέτηση κάποιων γεωμετρικών υποθέσεων (Hu et al. [2022]) όπως η a priori γνώση του 3D μοντέλου του αντικειμένου και η χρήση τρισδιάστατης πληροφορίας από πρόσθετους αισθητήρες (He et al. [2021]). Με την απόκτηση μεγάλου όγκου δειγμάτων/annotation που περιγράφουν τις πόζες στις RGB εικόνες ευνοούνται σημαντικά οι τεχνικές deep learning όμως η διαδικασία απόκτησης τους μπορεί να είναι δύσκολη και χρονοβόρα (Hodan et al. [2017], Wang et al. [2020]). Για αυτόν τον λόγο μπορούν να δημιουργηθούν συνθετικά δεδομένα εκπαίδευσης μέσω της αναπαράστασης των μοντέλων με τη χρήση των γραφικών υπολογιστών (graphics rendering). Βέβαια, σε αυτή τη περίπτωση πρέπει να ληφθούν αποφάσεις για την προσαρμογή των μοντέλων σε πραγματικά δεδομένα αφού υπάρχει απόκλιση της απλουστευμένης συνθετικής εικόνας σε σχέση με τον πραγματικό κόσμο (Wang et al. [2020]).

3.2 RGB vs RGB-D

Λόγω της φύσης της συγκεκριμένης εφαρμογής επιλέχθηκε ο αλγόριθμος που θα χρησιμοποιηθεί τελικά για τον προσδιορισμό της 6D πόζας της αρπάγης να λειτουργεί με μοναδικό δεδομένο εισόδου μία RGB εικόνα. Το τελευταίο βασίζεται κυρίως στις απαιτήσεις της εφαρμογής, δηλαδή την εκτίμηση της πόζας της αρπάγης στο εξωτερικό περιβάλλον της αποβάθρας όπου υπάρχουν πολλά μεταλλικά αντικείμενα και κατασκευές τα οποία μπορούν να εισάγουν θόρυβο στις εικόνες βάθους λόγω των αντανακλάσεων. Ακόμα, δεν μπορούν να χρησιμοποιηθούν ενεργητικοί αισθητήρες λόγω κορεσμού της υπέρυθρης ακτινοβολίας από τον ήλιο και η εγκατάσταση περισσότερων από μία κάμερες για την επίτευξη στερεοκάλυψης αυξάνει το υπολογιστικό κόστος. Λόγω των παραπάνω και του τεράστιου όγκου της βιβλιογραφίας πάνω στην εκτίμηση της 6D πόζας αντικειμένων η ανάλυση εστιάζει κυρίως σε μεθόδους που χρησιμοποιούν RGB εικόνες.

3.3 Μέθοδοι για τον προσδιορισμό της 6D πόζας αντικειμένου

Ο κύριος διαχωρισμός των μεθόδων που έχουν αναπτυχθεί έγκειται στο αν βασίζονται στην εκπαίδευση ενός αλγορίθμου ή όχι (Learning-based ή Non-Learning-based) με την πρώτη κατηγορία να ενσωματώνει πρόσφατες υλοποιήσεις με την χρήση των CNN. Η κατηγοριοποίηση ακολουθεί την δομή των Du et al. [2019].

3.3.1 Non-learning based Methods

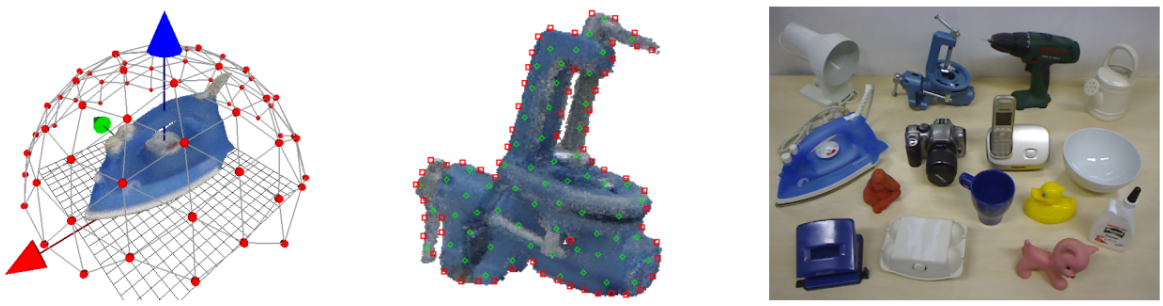
3.3.1.1 Κλασσικές μέθοδοι εξαγωγής χαρακτηριστικών σε συνδυασμό με δημιουργία ομολογιών 2D-3D

Οι κλασσικές μέθοδοι εξαγωγής χαρακτηριστικών (feature extraction) στοχεύουν κυρίως σε αντικείμενα με αρκετή υφή προσδιορίζοντας ομολογίες με την αναγνώριση 2D χαρακτηριστικών σημείων. Αρχικά, μπορούν να δημιουργηθούν εικόνες με την προβολή των 3D μοντέλων από διάφορες οπτικές γωνίες και θέσεις στις οποίες κάθε pixel του αντικειμένου θα αντιπροσωπεύει ένα 3D σημείο. Οι 2D-3D ομολογίες εντοπίζονται με την αντιστοίχηση χαρακτηριστικών σημείων στις πραγματικές εικόνες και τις δημιουργούμενες (Vacchetti et al. [2004], Lepetit and Fua [2005]). Τελικά η 6D πόζα προκύπτει με

τη χρήση αλγορίθμου PnP (Lepetit et al. [2009]). Για τη διαδικασία της αναγνώρισης και συνταύτισης των χαρακτηριστικών σημείων χρησιμοποιούνται αλγόριθμοι (2D descriptors) όπως ο SHIFT (Scale Invariant Feature Transform - Lowe [1999]), FAST (Features from accelerated segment test, Rosten and Drummond [2005]), SURF (Speeded Up Robust Features, Bay et al. [2006]) και ORB (Oriented FAST and Rotated BRIEF, Rublee et al. [2011]).

3.3.1.2 Μέθοδοι βασισμένες σε πρότυπα (template based)

Αντίστοιχα με τις τεχνικές που βασίζονται σε deep learning υπάρχουν και πιο συμβατικές προσεγγίσεις με βάση πρότυπα που χρησιμοποιούν σαν είσοδο RGB-D εικόνες. Η μέθοδος των Hinterstoisser et al. [2012] δημιουργεί ένα σετ εικόνων (templates) με βάση το 3D CAD μοντέλο του αντικειμένου στις οποίες αυτό απεικονίζεται από διάφορες οπτικές γωνίες. Τα κέντρα λήψης προκύπτουν από τον διαχωρισμό ενός 20εδρου σε μικρότερα τρίγωνα. Τελικά προκύπτουν 162 θέσεις λήψης για το πάνω ημισφαίριο του μοντέλου. Η επιλογή αυτή γίνεται τόσο για καλύτερη απόδοση του αλγορίθμου από άποψη χρόνου όσο και για το γεγονός ότι στις εικόνες του dataset δεν απαιτείται ανάκτηση πόζας για αντικείμενο που φαίνεται το κάτω μέρος του. Η παραπάνω διαδικασία δειγματοληψίας εικόνων πραγματοποιείται και για περισσότερες κλίμακες δηλαδή για μεγαλύτερα πολύεδρα. Η αντιστοίχηση των προτύπων εικόνων γίνεται με βάση χαρακτηριστικά αλλαγής χρώματος (Color gradient Features), τα οποία υπολογίζονται από την RGB εικόνα και χαρακτηριστικές ιδιότητες που αφορούν τα κάθετα διανύσματα στις επιφάνειες (normals) οι οποίες υπολογίζονται από το 3D μοντέλο του αντικειμένου. Τα υποψήφια πρότυπα που εντοπίζονται περνάνε από διάφορα τεστ με βάση το χρώμα και το βάθος και τελικά προσαρμόζεται καλύτερα η τελική πόζα με μία παραλλαγή της μεθόδου ICP (Arun et al. [1987]).



Σχήμα 27: Η μέθοδος των Hinterstoisser et al. [2012]. Αριστερά: η συστηματική συλλογή συνθετικών εικόνων του αντικειμένων. Στη μέση: τα χαρακτηριστικά σημεία που επιλέγονται στην επιφάνεια του αντικειμένου μέσω της αλλαγής χρώματος (color gradient - κόκκινα) αλλά και τα διανύσματα των επιφανειών (surface normals - πράσινα). Δεξιά: Αντικείμενα που χρησιμοποιήθηκαν

Οι Hodaň et al. [2015] πρότειναν μια ίδιας λογικής μεθοδολογία που βασιζόταν στη μέθοδο του κινούμενου παραθύρου (sliding window) για την αντιστοίχηση των προτύπων. Επειδή αυτή η διαδικασία εξαρτάται από τη διάσταση της εικόνας αλλά και από τον αριθμό των αντικειμένων και των προτύπων που μπορεί να είναι μεγάλα σε αριθμό ακολούθησαν μια προ-επεξεργασία για τον γρήγορο αποκλεισμό παραθύρων που δεν περιέχουν το αντικείμενο. Κάθε παράθυρο που απομένει εξετάζεται με κάθε πρότυπο και γίνεται η αντιστοίχηση. Τέλος, αφού προκύψει μία καλή εκτίμηση της πόζας αυτή βελτιστοποιείται με έναν αλγόριθμο Particle Swarm Optimization (PSO), Poli et al. [2007].

3.3.2 Learning-based μέθοδοι

Οι αλγόριθμοι μηχανικής μάθησης με τη παράλληλη ανάπτυξη της υπολογιστικής ισχύς έχουν παρουσιάσει μεγάλο ερευνητικό ενδιαφέρον λόγω της ανάπτυξης τεχνικών και μοντέλων βαθιάς μάθησης και τη χρήση των νευρωνικών δικτύων (NN). Σημαντικό παράδειγμα αποτελεί η μέθοδος PoseNet (Kendall et al. [2015]) η οποία διαμορφώνει το πρόβλημα της απόκτησης της 6D πόζας από μια RGB εικόνα σε πρόβλημα παλινδρόμησης (regression). Χρησιμοποιεί μία συνάρτηση loss η οποία αναγκάζει το δίκτυο να προσαρμόζεται σταδιακά εξίσου στη πρόβλεψη της θέσης του αντικειμένου όσο και της στροφής του. Ακόμα, τα δεδομένα που χρησιμοποιήθηκαν για την εκπαίδευση του CNN δημιουργήθηκαν με τη μέθοδο Structure For Motion (SFM) από βίντεο που περιγράφουν τα αντικείμενο.

Συμπληρωματικά, αναπτύχθηκαν μέθοδοι που αξιοποιούν τμηματικά τα 3D μοντέλο για τις προβλέψεις τους, όπως η μέθοδος του (Crivellaro et al. [2015]). Ο αλγόριθμος αυτός σχεδιάστηκε έτσι ώστε να αποδίδει ακόμα και σε περιπτώσεις όπου το αντικείμενο στην εικόνα αποκρύπτεται από κάποιο άλλο αντικείμενο της σκηνής. Το CNN που χρησιμοποιήθηκε εκπαιδεύεται να αναγνωρίζει τις αντιστοιχίες 2D-3D των επιλεγμένων χαρακτηριστικών περιοχών. Με αυτό το τρόπο όταν το αντικείμενο είναι μερικώς ορατό η 6D πόζα προκύπτει από τις αντίστοιχες ορατές χαρακτηριστικές περιοχές ενώ όταν το αντικείμενο είναι εξ ολοκλήρου ορατό (περισσότερες περιοχές και 2D-3D αντιστοιχίες) η 6D πόζα προκύπτει με τον συνδυασμό όλων των υποπεριοχών του και οδηγεί στην παραγωγή υψηλότερης ακρίβειας πόζας. Η συγκεκριμένη μέθοδος λειτουργεί αποκλειστικά με δεδομένα RGB και όχι δεδομένα βάθους κάτι που την κάνει ιδιαίτερα χρήσιμη για μεταλλικά και γυαλιστερά αντικείμενα.

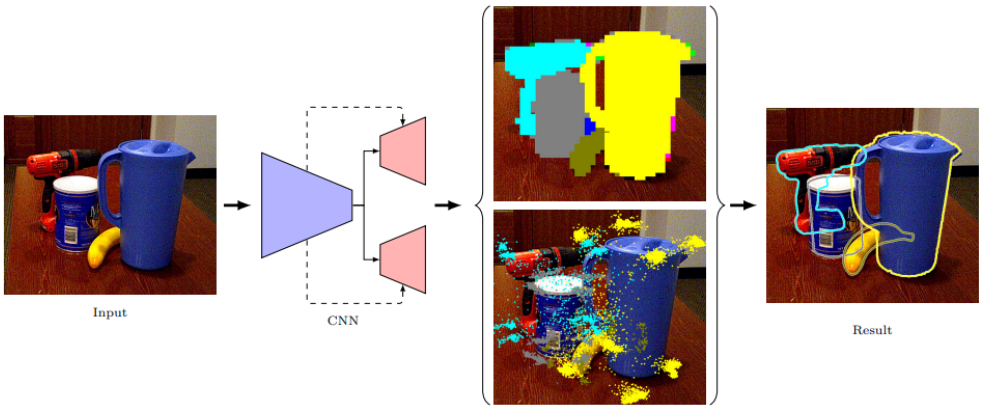
3.3.2.1 Μέθοδοι που βασίζονται σε χαρακτηριστικά σημεία (Keypoint-Based Approaches)

Οι μέθοδοι που λειτουργούν με χαρακτηριστικά σημεία, δημιουργούν ομολογίες μεταξύ της 2D εικόνας του αντικειμένου και του 3D μοντέλου του και μέσω αυτών προκύπτει η 6D πόζα του αντικειμένου σε κάθε εικόνα. Η διαδικασία αυτή χωρίζεται σε δύο διακριτά μέρη: στην εξαγωγή χαρακτηριστικών (feature extraction) στην εικόνα εισόδου και στον προσδιορισμό της 6D πόζας με την επίλυση του PnP προβλήματος.

Η μέθοδος BB8 (Rad and Lepetit [2017]) χρησιμοποίησε ένα CNN για τον προσδιορισμό των 2D απεικονίσεων 8 σημείων του 3D bounding box του μοντέλου και την εξαγωγή της 6D πόζας μέσω αλγορίθμου PnP. Η μέθοδος αυτή αξιολογήθηκε σε benchmark datasets όπως τα LINEMOD (Hinterstoisser et al. [2012]) και το T-LESS (Hodan et al. [2017]). Στα dataset αυτά υπάρχουν συμμετρικά αλλά και χωρίς έντονη υφή αντικείμενα που δυσκολεύουν τον προσδιορισμό της πόζας. Για να τα επιλύσει αυτά η BB8 περιορίζει τις δυνατές γωνίες στροφής των δεδομένων εκπαίδευσης με την ενσωμάτωση ενός ταξινομητή (classifier) ο οποίος προσδιορίζει το εύρος της γωνίας πριν την εκτίμηση από το CNN. Για την εκτίμηση της πόζας συμμετρικών αντικειμένων η εικόνα μετασχηματίζεται (mirror) έτσι ώστε να απεικονίζεται η συμμετρική πόζα του αντικειμένου στην οποία έχει εκπαιδευτεί ο αλγόριθμος. Μετά την πρόβλεψη η εκτιμώμενη πόζα μετασχηματίζεται με τον αντίθετο μετασχηματισμό (mirror back) ώστε να αναφέρεται στην αρχική πόζα του αντικειμένου στην εικόνα. Ωστόσο, η μέθοδος αυτή έχει αδυναμίες σε περιπτώσεις μερικής απόκρυψης του αντικειμένου.

Βελτίωση της προηγούμενης αποτελεί η μέθοδος Hu et al. [2019], η οποία βασίζεται στην συστηματική κατάτμηση της εικόνας σε ένα κάναβο και στην πρόβλεψη της παρουσίας των μοντέλων σε κάθε τμήμα του κανάβου. Η μέθοδος αυτή ενοποιεί τις διαδικασίες της κατάτμησης αντικειμένων (2D segmentation mask) και της εκτίμησης της 6D πόζας (σε αντίθεση με τη BB8). Ο αλγόριθμος πλεονεκτεί σημαντικά σε εικόνες με απόκρυψη αντικειμένων γιατί συνδυάζονται πολλές τοπικές προβλέψεις 2D χαρακτηριστικών σημείων που αντιστοιχούν σε κάθε αντικείμενο που ορίζεται από την προβλεπόμενη μάσκα. Η δυνατότητα εκτίμησης της πόζας από τμήμα του μοντέλου το κάνει να αποδίδει

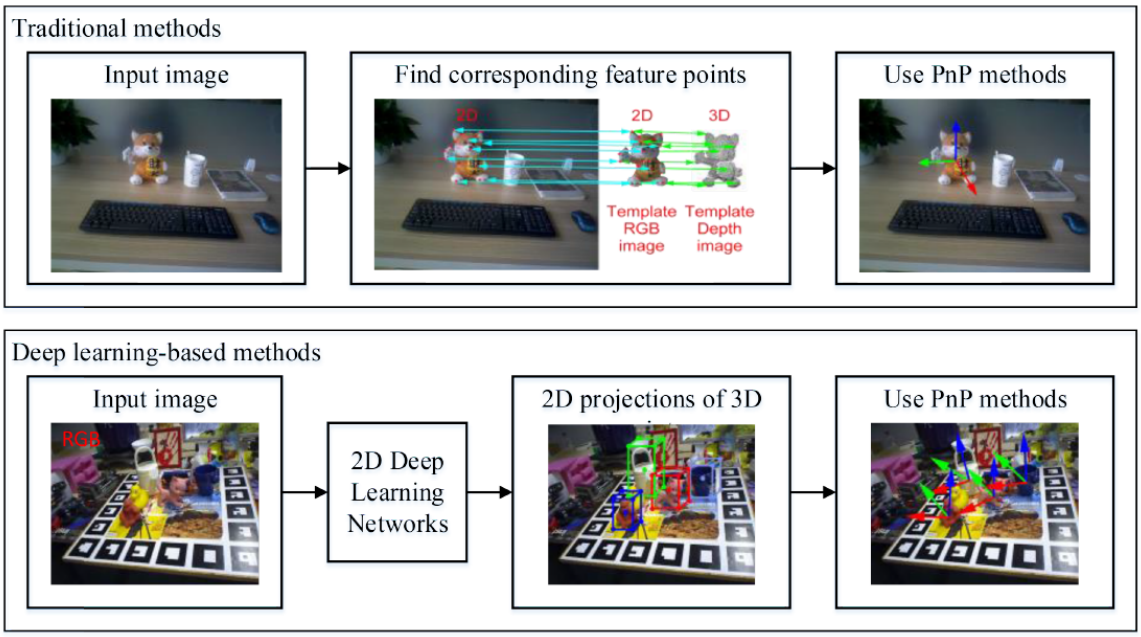
πολύ καλά σε περιπτώσεις απόκρυψης των αντικειμένων. Δεδομένου της κατεύθυνσης από κάθε pixel προς κάθε χαρακτηριστικό σημείο δημιουργούνται υποθέσεις για τις θέσεις τους. Βασισμένο στις τελευταίες, υπολογίζεται ο μέσος όρος και η χωρική διασπορά της κατανομής πιθανότητας των θέσεων των χαρακτηριστικών σημείων. Το πλεονέκτημα αυτής της μεθόδου είναι ότι επειδή χρησιμοποιεί τις κατευθύνσεις προς τα χαρακτηριστικά σημεία μπορεί να εκτιμήσει αποτελεσματικά 6D πόζες αντικειμένων που υφίστανται μεγάλη απόκρυψη ή βρίσκονται εκτός της εικόνας.



Σχήμα 28: Η αρχιτεκτονική της μεθόδου των Hu et al. [2019]: Η αρχιτεκτονική περιλαμβάνει δύο διαδικασίες. Η μία προβλέπει ποιο μοντέλο απεικονίζεται σε κάθε pixel του κανάβου ενώ η δεύτερη προβλέπει την 2D θέση των χαρακτηριστικών σημείων για το παρών αντικείμενο (Πηγή: Hu et al. [2019])

Για την επίλυση του προβλήματος του βάθους σε αδιαφανή αντικείμενα προσεγγίσεις όπως το δίκτυο KeyPose (Liu et al. [2020]) προβλέπει τις πόζες των αντικειμένων με τη χρήση 3D χαρακτηριστικών σημείων από στερεοεικόνες (δύο εικόνες από διαφορετική οπτική γωνία). Ακόμα οι Hu et al. [2020] πρότειναν μία μέθοδο, η οποία χρησιμοποιεί απευθείας παλινδρόμηση (regression) της 6D πόζας χρησιμοποιώντας 2D-3D αντιστοιχίες αλλά χωρίς τη χρήση αλγορίθμου RANSAC PnP. Το τελευταίο δικαιολογείται από τους συγγραφείς λόγω της πολυπλοκότητας του αλγορίθμου ειδικά κατά την ύπαρξη πολλών ακραίων τιμών (outliers) και της εξάρτησης του από την σειρά των ομολογιών 2D-3D (αν αλλάξει η σειρά των ομολογιών ο αλγόριθμος δεν θα υπολογίσει την ίδια 6D πόζα). Εκτός από αυτό, οι Hu et al. [2020] πρότειναν την χρήση μίας συνάρτησης loss που συνδυάζει το σφάλμα ADD (βλ. ενότητα 4.3) της 6D πόζας, το σφάλμα στη ταξινόμηση κάθε pixel του κανάβου σε μία κλάση (Focal Loss, Lin et al. [2017]) και το σφάλμα στον προσδιορισμό των αντιστοιχιών 2D από χαρακτηριστικά σημεία 3D αντικατοπτρίζοντας έτσι άμεσα το σφάλμα στη 6D πόζα.

Το 2020 οι Hodaň et al. [2020] πρότειναν τον αλγόριθμο EPOS όπου τα αντικείμενα χωρίζονται σε μικρές επιφάνειες (surface fragments) το οποίο δίνει τη δυνατότητα για διαχείριση καθολικής ή μερικής συμμετρίας σε αντικείμενα. Η μέθοδος προσδιορίζει ομολογίες 2D-3D μέσω ενός δικτύου τύπου encoder-decoder. Σε κάθε pixel το μοντέλο προσδιορίζει την πιθανότητα ύπαρξης του κάθε αντικειμένου, την πιθανότητα ύπαρξης fragment δεδομένης της ύπαρξης συγκεκριμένου μοντέλου και την ακριβή θέση του fragment. Τέλος, η συγκεκριμένη υλοποίηση πετυχαίνει μεγάλες ακρίβειες στα dataset αξιολόγησης (LINEMOD,T-LESS) και μπορεί να αντιμετωπίσει πολλαπλά αντικείμενα, αποκρύψεις αλλά και πολλαπλές εμφανίσεις του ίδιου αντικειμένου.



Σχήμα 29: Διαδικασία μεθόδων που βασίζονται σε χαρακτηριστικά σημεία και εκτίμηση 6D τελικής πόζας μέσω PnP αλγορίθμων (Πηγή: Du et al. [2019])

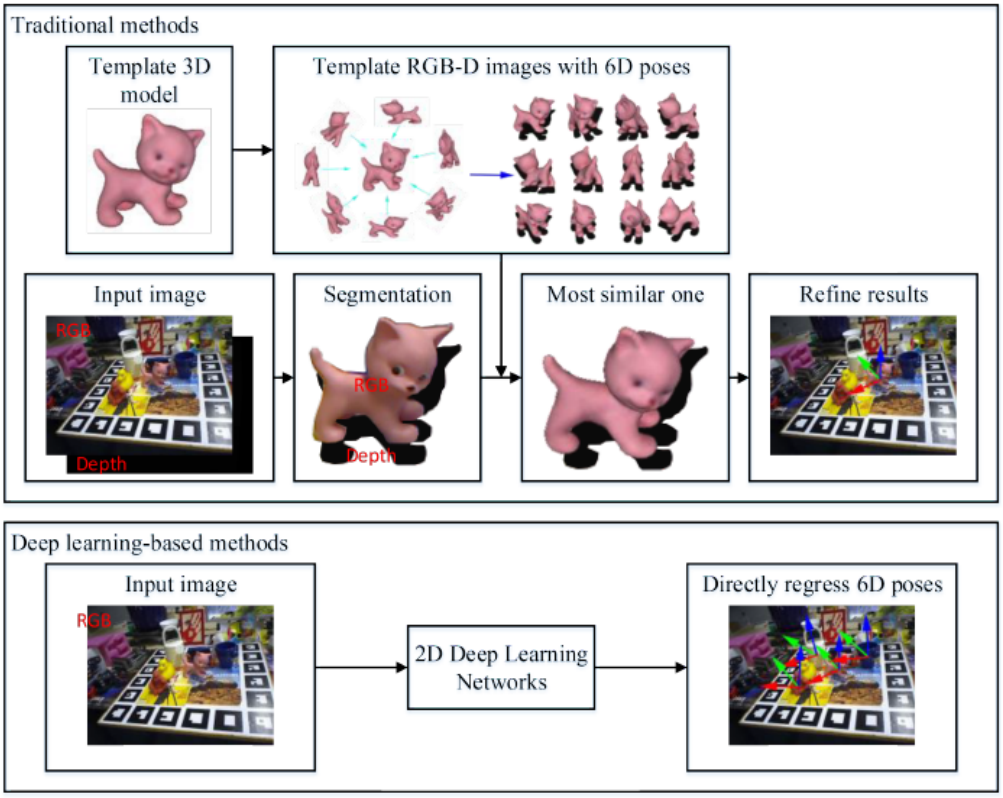
3.3.2.2 Μέθοδοι βασισμένες σε πρότυπα (template based)

Οι μέθοδοι αυτής της κατηγορίας βασίζονται στην εύρεση του πιο κοντινού/όμοιου προτύπου από τα διαθέσιμα Ground Truth (GT) πρότυπα που αναφέρονται σε μία 6D πόζα η κάθε μία. Όταν τα δεδομένα εισόδου είναι εικόνες, τα πρότυπα (templates) μπορεί να είναι οι προβολές του 3D αντικειμένου στο επίπεδο της εικόνας. Συνεπώς ο προσδιορισμός της πόζας μετασχηματίζεται σε ένα πρόβλημα δημιουργίας των προτύπων και η εύρεση του πλησιέστερου στην εικόνα. Αυτές οι μέθοδοι κυρίως στοχεύουν σε αντικείμενα με έλλειψη υφής όπου αλγόριθμοι βασισμένοι σε χαρακτηριστικά σημεία δεν αποδίδουν τόσο καλά. Υλοποιήσεις όπως το PoseCNN (Xiang et al. [2018]) προσδιορίζουν απευθείας την 6D πόζα. Συγκεκριμένα, η τελευταία προσδιορίζει το κέντρο του αντικειμένου στην εικόνα και προβλέπει την απόσταση του από τη κάμερα (3d μετάθεση) ενώ η στροφή του αντικειμένου σε σχέση με τη κάμερα αντιπροσωπεύεται με τη χρήση τετραδρονίων στα οποία γίνεται και η παλινδρόμηση των παραμέτρων τους. Ακόμα ενσωμάτωσε στη συνάρτηση loss ένα παράγοντα ο οποίος επιτρέπει στο δίκτυο να διαχειρίζεται αντικείμενα με συμμετρίες. Το 2017 οι Liu et al. [2016] πρότειναν το SSD (Single-shot Detector) για την 2D αναγνώριση αντικειμένων σε RGB εικόνες χρησιμοποιώντας προκαθορισμένα bounding boxes και την προσαρμογή τους κατά τη διαδικασία της εκπαίδευσης έτσι ώστε να περιγράφουν καλύτερα τα αντικείμενα ενδιαφέροντος. Το SSD-6D (Kehl et al. [2017]) επέκτεινε την υλοποίηση του SSD για τον προσδιορισμό 6D πόζας με εύκολη εκπαίδευση και διαχείριση συμμετρικών αντικειμένων. Ένα χρόνο αργότερα οι Do et al. [2018] πρότειναν μια end-to-end υλοποίηση (Deep-6DPose) η οποία χρησιμοποιούσε ένα δίκτυο Region Proposal Network (RPN) βασισμένο στο Mask R-CNN (He et al. [2017]). Η καινοτομία της συγκεκριμένης μεθόδου έγκειται στον τρόπο που πραγματοποιείται ο προσδιορισμός της πόζας με το να ανεξαρτητοποιεί τη μετάθεση και τη στροφή ώστε η τελευταία να μπορεί να προσδιοριστεί με παλινδρόμηση με χρήση της άλγεβρας Lie (Lie group). Ωστόσο, αυτή η μέθοδος εφόσον βασίζεται στις προτάσεις περιοχών (Region proposals) δεν έχει καλή απόδοση σε μικρά και συμμετρικά αντικείμενα. Παρόμοια πορεία ακολούθησαν και οι Li et al. [2019] με το δίκτυο Cdpn (Coordinates-based disentangled pose network for real-time rgb-based 6-dof object pose estimation) όπου χρησιμοποίησαν ένα γρήγορο δίκτυο για το αναγνώριση

Table 4: Summary of correspondence-based 6D object pose estimation methods.

Methods	Descriptions	Traditional methods	Deep learning-based methods
2D image-based methods	Find correspondences between 2D pixels and 3D points, and use PnP methods	SIFT [Lowe, 1999], FAST [Rosten and Drummond, 2005], SURF [Bay *et al.*, 2006], ORB [Rublee *et al.*, 2011]	LCD [Pham *et al.*, 2020], BB8 [Rad and Lepetit, 2017], Tekin et al. [Tekin *et al.*, 2018], Crivellaro et al. [Crivellaro *et al.*, 2017], KeyPose [Liu *et al.*, 2020b], Hu et al. [Hu *et al.*, 2020], HybridPose [Song *et al.*, 2020], Hu et al. [Hu *et al.*, 2019], DPOD [Zakharov *et al.*, 2019], Pix2pose [Park *et al.*, 2019b], EPOS [Hodan *et al.*, 2020]
3D point cloud-based methods	Find correspondences between 3D points	Spin Images [Johnson, 1997], 3D Shape Context [Frome *et al.*, 2004], FPFH [Rusu *et al.*, 2009a], CVFH [Aldoma *et al.*, 2011], SHOT [Salti *et al.*, 2014]	3DMatch [Zeng *et al.*, 2017a], 3DFeat-Net [Yew and Lee, 2018], Gojcic et al. [Gojcic *et al.*, 2019], Yuan et al. [Yuan *et al.*, 2020], StickyPillars [Simon *et al.*, 2020]

Σχήμα 30: Σύνοψη μεθόδων που βασίζονται σε ομολογίες σημείων και επίλυση PnP (Πηγή: Du et al. [2019])



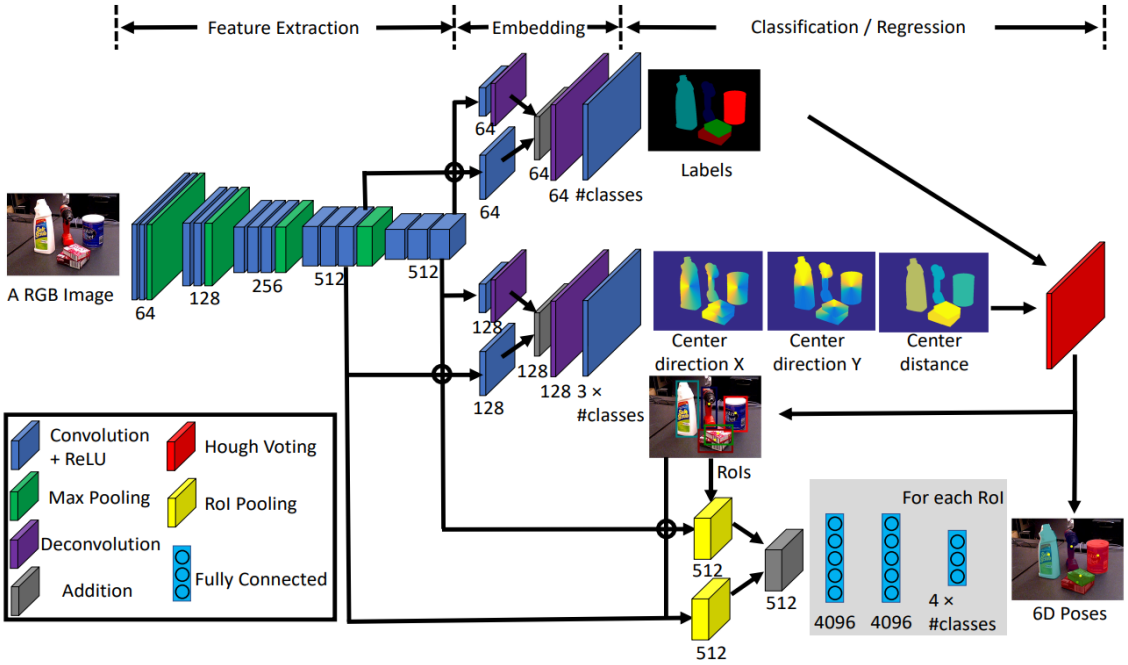
Σχήμα 31: Διαδικασία μεθόδων template-based (Πηγή: Du et al. [2019])

αντικειμένου (tiny YOLOv3 - Joseph Redmon [2018]) το οποίο ωστόσο δεν υστερούσε σε ακρίβεια λόγω της χρήσης της δυναμικής μεγέθυνσης (dynamic zoom) κατά την εκπαίδευση για τον εντοπισμό των περιοχών των αντικειμένων. Διαχώρισαν τον προσδιορισμό της στροφής και της μετάθεσης με τη δεύτερη να υπολογίζεται με τη χρήση μίας μεθόδου ανεξάρτητης της κλίμακας (SITE - Scale Invariant Translation Estimation) απευθείας από τις υποπεριοχές τις εικόνες που εντοπίστηκαν στο πρώτο στάδιο.

Table 5: Summary of template-based 6D object pose estimation methods.

Methods	Descriptions	Traditional methods	Deep learning-based methods
2D image-based methods	Retrieve the template image that is most similar with the observed image	LineMod [Hinterstoisser *et al.*, 2012], Hodaň et al. [Hodaň *et al.*, 2015]	AAE [Sundermeyer *et al.*, 2018], PoseCNN [Xiang *et al.*, 2018], SSD6D [Kehl *et al.*, 2017], Deep-6DPose [Do *et al.*, 2018a], Liu et al. [Liu *et al.*, 2019a], CDPN [Li *et al.*, 2019], Tian et al. [Tian *et al.*, 2020], NOCS [Wang *et al.*, 2019c], LatentFusion [Park *et al.*, 2020], Chen et al. [Chen *et al.*, 2020a]
3D point cloud-based methods	Find the pose that best aligns the observed partial 3D point cloud with the template full 3D model	Super4PCS [Mellado *et al.*, 2014], Go-ICP [Yang *et al.*, 2015]	PCRNet [Sarode *et al.*, 2019b], DCP [Wang and Solomon, 2019a], PointNetLK [Aoki *et al.*, 2019], PRNet [Wang and Solomon, 2019b], DeepICP [Lu *et al.*, 2019], Sarode et al. [Sarode *et al.*, 2019a], TEASER [Yang *et al.*, 2020a], DGR [Choy *et al.*, 2020], G2L-Net [Chen *et al.*, 2020c], Gao et al. [Gao *et al.*, 2020]

Σχήμα 32: Σύνοψη μεθόδων Template-based (Πηγή: Du et al. [2019])

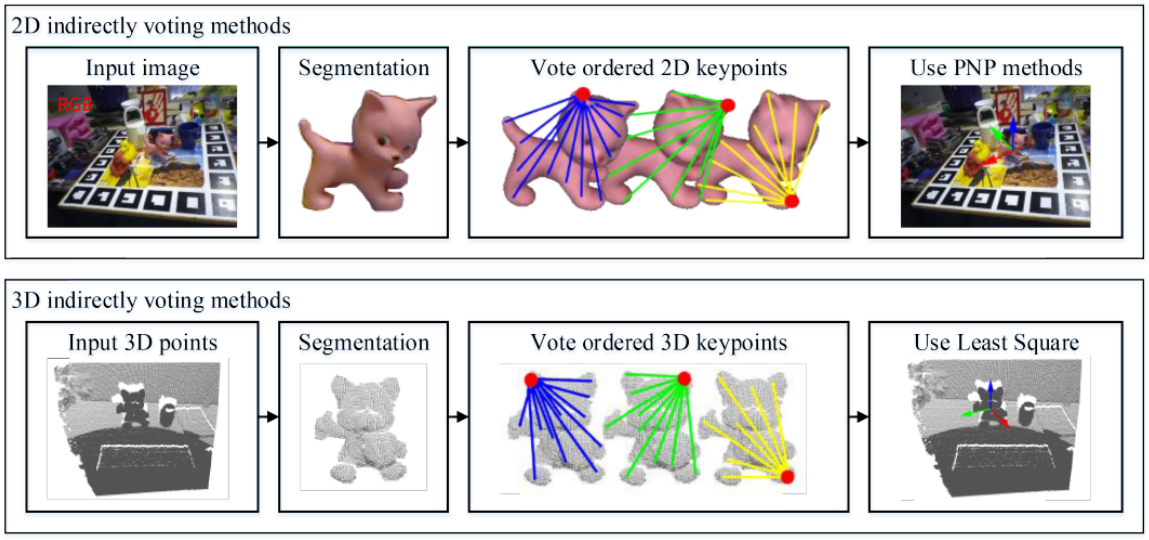


Σχήμα 33: Η αρχιτεκτονική του PoseCNN όπου το δίκτυο εκπαιδεύεται να πραγματοποιεί τρις διαδικασίες: Σημασιολογική κατάτμηση (semantic segmentation), εκτίμηση της 3D μετάθεσης, και τελικά την 3D στροφή του αντικειμένου σε σχέση με τη κάμερα (Πηγή: Xiang et al. [2018])

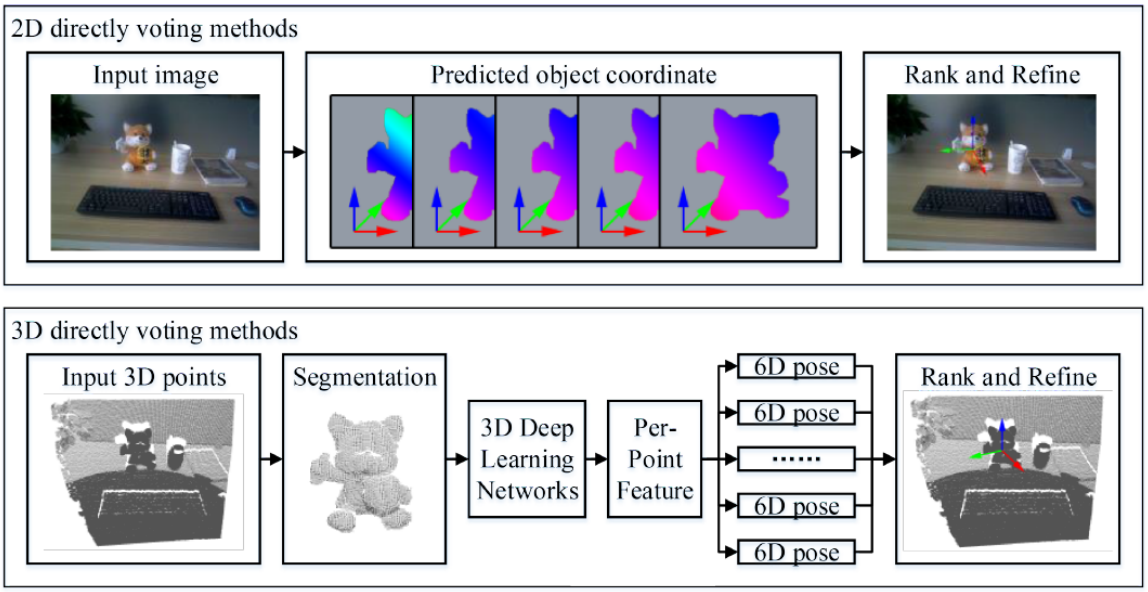
3.3.2.3 Μέθοδοι βασισμένες σε ψηφοφορία (voting-based)

Αυτές οι μέθοδοι ονομάζονται έτσι με την έννοια ότι κάθε pixel συνεισφέρει στον προσδιορισμό της 6D πόζας με το να τάσσεται υπέρ κάποιου χαρακτηριστικού σημείου ή των 6D συντεταγμένων αντικειμένου ή και της πόζας του. Αυτές είναι και οι δύο υποκατηγορίες αυτής της μεθόδου: έμμεσης συνεισφοράς μέθοδοι (Indirect voting methods) και άμεσης συνεισφοράς μέθοδοι (Direct voting methods). Οι πρώτες κυρίως αφορούν την συνεισφορά των pixel στον προσδιορισμό 2D χαρακτηριστικών σημείων για τον καθορισμό 2D-3D ομολογιών. Ένα σημαντικό παράδειγμα είναι το δίκτυο PVNet (Peng et al. [2019]) όπου εκτιμώνται διανύσματα για κάθε pixel τα οποία έχουν κατεύθυνση προς τα χαρακτηριστικά σημεία και έπειτα προσδιορίζονται ομολογίες 2D-3D μέσω συνεισφοράς (voting). Στη συνέχεια υπολογίζεται η χωρική κατανομή των 2D χαρακτηριστικών σημείων με τον αλγόριθμο RANSAC (Fischler and Bolles [1981]). Τελικά χρησιμοποιείται ένα αλγόριθμος για την επίλυση του PnP που

λαμβάνει υπόψιν του την αβεβαιότητα των υποψήφιων θέσεων. Με το παραπάνω τρόπο ο αλγόριθμος RANSAC έχει αρκετά σημεία για διαχειριστεί τα outliers. Όσον αφορά τις μεθόδους άμεσης συνεισφοράς κυρίως χρησιμοποιούνται για τις μεθόδους προτύπων (tamplate based) συνεισφέροντας απευθείας σε μία πόζα του αντικειμένου. Τέτοιου είδους μέθοδοι χρησιμοποιούνται για τον προσδιορισμό της πόζας αντικειμένων τα οποία υφίστανται αποκρύψεις μερών τους στην εικόνα. Για αυτά τα αντικείμενα κάθε pixel του κανάβου συνεισφέρει/"ψηφίζει" για την καθολική πόζα του αντικειμένου. Τέτοια παραδείγματα είναι οι μέθοδοι των Brachmann et al. [2014a] και Tejani et al. [2014], οι οποίες χρησιμοποιούν και οι δύο RGB και εικόνες βάθους. Η πρώτη μοντελοποιεί τα αντικείμενα με βάση τις 3D συντεταγμένες παράλληλα με την αναγνώριση των κλάσεων των αντικειμένων. Οι προβλεπόμενες συντεταγμένες του μοντέλου ορίζουν αντιστοιχίες 3D-3D και έτσι προσδιορίζονται οι 6D πόζες των αντικειμένων οι οποίες βελτιώνονται με τη χρήση μεθόδου τύπου ICP (Arun et al. [1987]).



Σχήμα 34: Λειτουργία μεθόδων έμμεσης συνεισφοράς (Indirect voting methods) (Πηγή: Du et al. [2019])

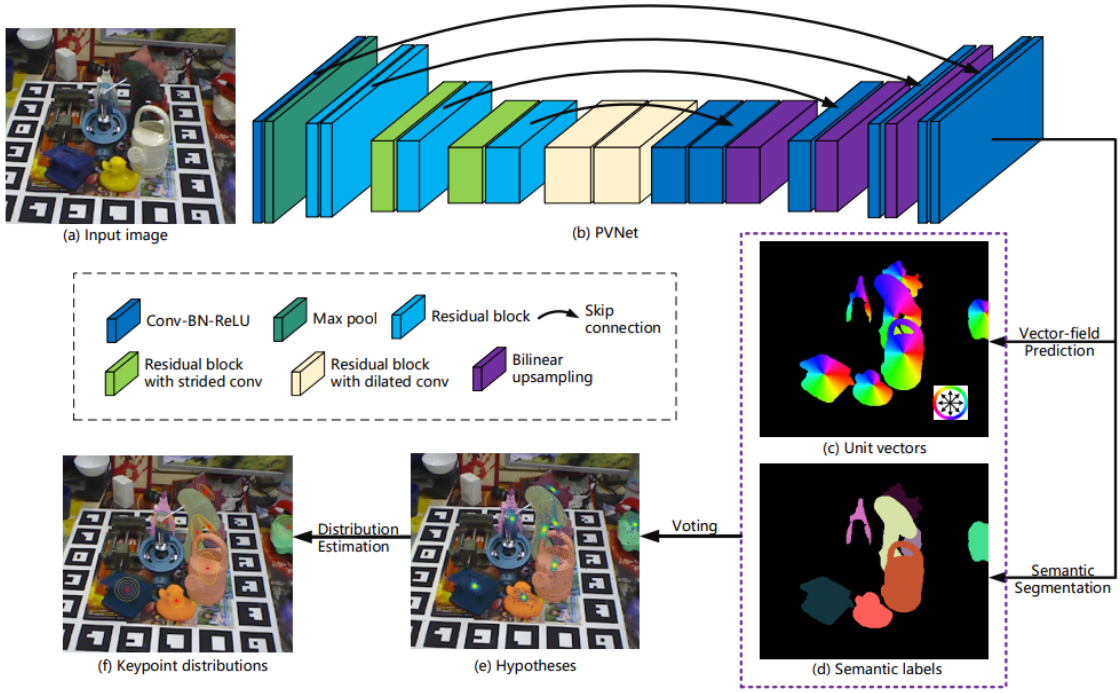


Σχήμα 35: Λειτουργία μεθόδων άμεσης συνεισφοράς (Direct voting methods) (Πηγή: Du et al. [2019])

Table 6: Summary of voting-based 6D object pose estimation methods.

Methods	Descriptions	2D image-based methods	3D point cloud-based methods
Indirect voting methods	Voting for correspondence-based methods	PVNet [Peng *et al.*, 2019], Yu et al. [Yu *et al.*, 2020]	PVN3D [He *et al.*, 2020], YOLOff [Gonzalez *et al.*, 2020], 6-PACK [Wang *et al.*, 2019a]
Direct voting methods	Voting for template-based methods	Brachmann et al. [Brachmann *et al.*, 2014], Tejani et al. [Tejani *et al.*, 2014], Crivellaro et al. [Crivellaro *et al.*, 2017], PPF [Drost and Ilic, 2012]	DenseFusion [Wang *et al.*, 2019b], MoreFusion [Wada *et al.*, 2020]

Σχήμα 36: Σύνοψη voting-based μεθόδων (Πηγή: Du et al. [2019])



Σχήμα 37: Η αρχιτεκτονική της μεθόδου PVNet (Peng et al. [2019]): (a) Εικόνα εισόδου, (b) Η αρχιτεκτονική του CNN, (c) τα εκτιμώμενα διανύσματα με κατεύθυνση προς τα χαρακτηριστικά σημεία, (d) Σημασιολογική κατάτμηση (semantic segmentation), (e) Εκτιμώμενες θέσεις χαρακτηριστικών σημείων, (f) Η χωρική κατανομή της πιθανότητας των σημείων κλειδιών (Με κόκκινο αστέρι αντιπροσωπεύεται ο μέσος όρος ενώ με έλλειψη η διασπορά)

3.4 Δεδομένα για συγκριτική αξιολόγηση (benchmarking)

Η ύπαρξη δεδομένων είναι ένας σημαντικός παράγοντας για τη μηχανική μάθηση. Ειδικά όταν πρόκειται για νευρωνικά δίκτυα χρειάζεται μεγάλος όγκος δεδομένων για την εκπαίδευσης τους αλλά και την αξιόπιστη αξιολόγηση τους. Λόγω της μεγάλης εξάρτησης της ανάπτυξης μεθόδων από τη διαθεσιμότητα δεδομένων έχουν κατασκευαστεί σετ δεδομένων για την συστηματική ανάπτυξη νέων μεθόδων αλλά και την ύπαρξη κοινών σημείων αναφοράς σε σχέση με την απόδοση και την ακρίβεια τους. Κάποια από τα dataset που χρησιμοποιούνται από τη βιβλιογραφία (Kim and Hwang [2021]) αναφέρονται συνοπτικά παρακάτω:

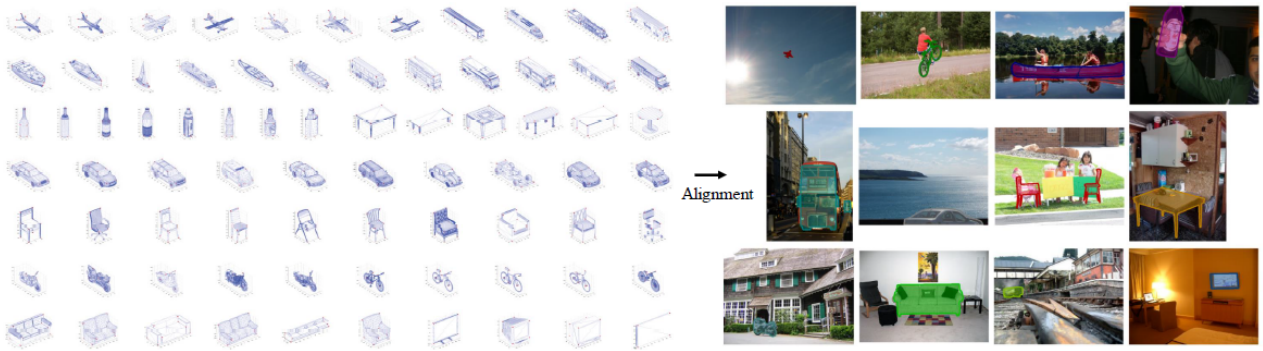
- **PASCAL3D+ (Xiang et al. [2014])**

  Αποτελεί συνέχεια του dataset PASCAL VOC (Everingham et al. [2010]) το οποίο δημιουργήθηκε για 2D αναγνώριση αντικειμένων και προσφέρει 3D annotations για 12 κατηγορίες. Οι

Dataset	Description	Data Type	Scene Type	Syn.?	# 3D Objects	# Images	Related References
PASCAL 3D+ [31]	A Benchmark for 3D Object Detection in the Wild (WACV 2014)	RGB + 3D models	Indoor + Outdoor	Real	3000 per cate. 12 categories	>20,000	PASCAL VOC [32], ImageNet [33], Google Warehouse
SUN RGB-D [34]	A Scene Understanding Benchmark Suite (CVPR 2015)	RGB-D	Indoor	Real	14.2 per image 800 categories	>10,335 in total	NYU depth v2 [35], BerkeleyB3DO [36] SUN3D [37]
ObjectNet 3D [38]	A Large Scale Database 3D Object Recognition (ECCV 2016)	RGB + 3D models	Indoor + Outdoor	Real + Syn.	201,888 inst. 100 objects	90,127	ImageNet [33], ShapeNet [39], Trimble Warehouse
FAT [40]	A Synthetic Dataset for 3D Object Detection (CVPRW 2018)	RGB + 3D models	Household Objects	Syn.	1–10 per image 21 objects	61,500	YCB [41]
BOP [42,43]	Benchmark for 6D Object Pose Estimation (ECCV 2018, ECCVW 2020)	RGB-D + 3D models	Indoor (various)	Real + Syn.	302,791 inst. in 97,818 real images (test) 171 objects (w/texture)	>800 K train, test RGB-D (mostly synthetic)	LM [44], LM-O [45], T-LESS [46], ITODD [47], YCB-V [27], HB [48], RU-APC [49], IC-BIN [50], IC-MI [51], TUD-L [42], TYO-L [42]
Objectron [52]	Object-Centric Videos in the Wild with Pose Annotations	RGB	Indoor + Outdoor	Real	17,095 inst. (multi-view) 9 categories	>4 M (14,819 videos)	Open Images [53] Similar to CAMERA [54] (Real/syn. data)
KITTI 3D [55]	KITTI Vision Benchmark Suite—3D Objects (CVPR 2012)	RGB (Stereo) + PointCloud	Driving Scenes	Real	80,256 inst. 3 categories	14,999	Virtual KITTI 2 [56] * Photo-realistically simulated DB
CityScape 3D [57]	Dataset and Benchmark for 9 DoF Vehicle Detection (CVPRW 2020)	RGB (Stereo)	Driving Scenes	Real	8 categories	5000	CityScape [58]
Synscapes [59]	A Photo Synthetic Dataset for Street Scenes	RGB	Driving Scenes	Syn.	8 categories	25,000	Similar to CityScape [58] (Structure, content)
SYNTHIA-AL [60]	Synthetic Collection of Imagery and Annotation —3D Boxes (CVPR 2012)	RGB	Driving Scenes	Syn.	3 categories	>143 K	ImageNet [33] SYNTHIA [61]

Σχήμα 38: Σετ δεδομένων που χρησιμοποιούνται για τον 3D προσδιορισμό αντικειμένων (Πηγή: Kim and Hwang [2021])

τελευταίες εμπλουτίζονται με την εισαγωγή εικόνων από την βάση δεδομένων Imagenet (Deng et al. [2009]). Για το 3D annotation χρησιμοποιήθηκαν τα 3D CAD μοντέλα των αντικειμένων και ευθυγραμμίστηκαν με τις προβολές τους στις εικόνες. Το dataset περιέχει περισσότερα από 3000 αντικείμενα ανά κατηγορία σε φυσικά περιβάλλοντα και όχι μόνο με ελεγχόμενες συνθήκες.



Σχήμα 39: Παραδείγματα αντικειμένων του PASCAL3D+ dataset και οι αντίστοιχες εικόνες στις οποίες έχουν ευθυγραμμιστεί αυτά (Πηγή: Kim and Hwang [2021])

- **SUN RGB-D (Song et al. [2015])**

Το SUN RGB-D είναι μία επέκταση του SUN3D (Xiao et al. [2013]) και περιέχει 10355 εικόνες με αντίστοιχες depth οι οποίες έχουν αποκτηθεί με τη χρήση διάφορων αισθητήρων. Περιέχει 3389 frames χωρίς σημαντικό θόλωμα από τα βίντεο του dataset SUN3D αλλά και RGB-D εικόνες από άλλα dataset όπως τα NYU Depth V2 (Silberman et al. [2012]) και B3DO (Janoch et al. [2011]). Συνολικά, περιέχει 47 διαφορετικές κατηγορίες σκηνής και 800 κατηγορίες αντικειμένων.

Τα αντίστοιχα annotations είναι 2D πολύγωνα αλλά και 3D bounding boxes. Τελικά, αυτό το dataset χρησιμεύει για αξιολόγηση και υλοποίηση μεθόδων που στοχεύουν στην αναγνώριση αντικειμένων κυρίως σε εσωτερικούς χώρους με δεδομένα εισόδου RGB ή RGB-D εικόνες.

- **ObjectNet3D (Xiang et al. [2016])**

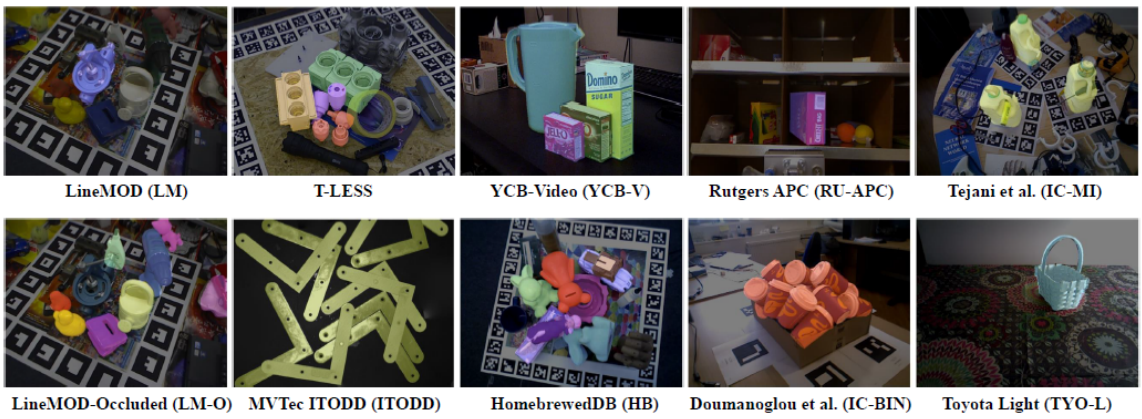
  Συνολικά ενσωματώνει 90127 εικόνες και 44147 3D μοντέλα που ανήκουν σε 100 διαφορετικές κατηγορίες αντικειμένων. Οι αρχικές εικόνες έχουν αποκτηθεί από το ImageNet (A Large-scale Hierarchical Image Database - Deng et al. [2009]) και συμπληρώθηκαν με εικόνες για κατηγορίες που δεν υπήρχαν αρκετά δεδομένα μέσα από αναζητήσεις Google. Ακόμα, επιλέχθηκαν κάποια μοντέλα από ShapeNet και Trimble Warehouse για να ευθυγραμμιστούν με τις προβολές τους στις εικόνες και δημιουργηθούν τα 3D annotations. Ομοίως με το PASCAL3D+ (Xiang et al. [2014]) το dataset αυτό συνεισφέρει στην εξαγωγή του σχήματος, τον εντοπισμό της θέσης του αντικειμένου (object detection) αλλά και την εκτίμηση της πόζας του (pose estimation).

- **Falling Things (FAT, Tremblay et al. [2018]**)

  Το FAT είναι προέκταση του Yale-CMU-Berkeley (YCB - Calli et al. [2015]). Περιέχει 61500 εικόνες από 21 οικιακά αντικείμενα. Σε αυτό το dataset εντάχθηκαν και συνθετικά δεδομένα με την δημιουργία ψηφιακών φωτορεαλιστικών εικόνων και αυτόματων annotation αυτών. Συνδυάζει, συνθετικές εικόνες μοντέλων και παρασκηνίων υψηλής ποιότητας. Για κάθε τέτοια εικόνα αποκτώνται με ακρίβεια λόγω της συνθετικής τους φύσης 2D/3D αντιστοιχίες, segmentation μάσκες και 3D πόζες. Το dataset στοχεύει στην βελτίωση της αντίληψης των ρομποτικών συστημάτων, στην ανάπτυξη και βελτιστοποίηση αλγορίθμων ταξινόμησης αλλά και την εκτίμηση της 6D πόζας αντικειμένων.

- **Benchmark for 6D Object Pose Estimation (BOP, Hodan et al. [2018]**)

  Το BOP αποτελεί ένα εκτεταμένο dataset αξιολόγησης των τεχνικών προσδιορισμού 6D πόζας που αποτελείται απο 8 ευρέως διαδεδομένα dataset και περιέχει εικόνες εκπαίδευσης με γνωστές τις 6D πόζες των αντικειμένων. Έχει επίσης την δυνατότητα απόδοσης υψηλής ποιότητας μοντέλων με υφή (textures). Ένα από το σύνολο των dataset είναι το LineMod (LM,Hinterstoisser et al. [2012]) που αποτελείται από 15 δίχως υφή αντικείμενα με διάφορα γεωμετρικά χαρακτηριστικά,μεγέθη και χρώματα που συναντώνται σε οικιακά αντικείμενα. Ακόμα, ένα μεγάλο και διαδεδομένο dataset είναι το T-LESS (Hodan et al. [2017]). Το τελευταίο αποτελείται από 30 βιομηχανικά αντικείμενα (πχ. πρίζες ρεύματος) με απεικονίσεις σε 20 διαφορετικές σκηνές (αλλαγή background). Σημαντικό είναι ότι τα αντικείμενα έχουν περιορισμένη υφή και παρουσιάζουν συμμετρίες, ενώ άλλα αντικείμενα είναι μικρότερα μέρη κάποιων άλλων κάνοντας το dataset αρκετά απαιτητικό. Ακόμα ένα dataset που χρησιμοποιείται μεταξύ των πολλών στο BOP είναι το YCB-V (YCB-Video, Calli et al. [2015]), το οποίο περιέχει 133827 εικόνες και 21 αντικείμενα επιλεγμένα από 92 βίντεο. Άλλα datasets που χρησιμοποιούνται είναι τα RU-APC (Rennie et al. [2016]), IC-BIN (Doumanoglou et al. [2015]), IC-MI (Tejani et al. [2018]) και TYO-L (Hodan et al. [2018]). Τέλος, όλα τα dataset που περιέχονται σε αυτήν την αξιολόγηση ακολουθούν κοινές αναπαραστάσεις των δεδομένων (BOP format).



Σχήμα 40: Παραδείγματα αντικειμένων που περιέχονται στο σύνολο των dataset του BOP dataset (Kim and Hwang [2021])

Dataset	Challenge	# Obj. Classes	Modality	# Total Frame	Obj. Dist. [mm]
LINEMOD	VP + C + TL	15	RGB-D	15770	600-1200
MULT-I	VP + C + TL + O + MI	6	RGB-D	2067	600-1200
OCC	VP + C + TL + SO	8	RGB-D	9209	600-1200
BIN-P	VP + SC + SO + MI + BP	2	RGB-D	180	600-1200
T-LESS	VP + C + TL + O + MI + SLD	30	RGB-D	10080	600-1200

Σχήμα 41: Δυσκολίες που παρουσιάζουν διάφορα dataset (Sahin and Kim [2018]). VP: σημείο λήψης (viewpoint), O: απόκρυψη μέρος του αντικειμένου (occlusion), C: σύγχυση λόγω ύπαρξης πολλών αντικειμένων και έντονου background (clutter), SO: Απόκρυψη μεγάλου μέρους του αντικειμένου (Sever occlusion), SC: μεγάλος βαθμός σύγχυσης λόγω ύπαρξης πολλών αντικειμένων και έντονου background (Severe Clutter), MI: Εμφάνιση αντικειμένου περισσότερο από μία φορές στην ίδια εικόνα (multiple instance), SLD: αντικείμενα με παρόμοια εμφάνιση που ”αποσπούν τον αλγόριθμο (similar looking distractors), BP: bin picking

3.5 Συμπεράσματα

Με βάση την ακρίβεια που παρέχουν οι διάφορες κατηγορίες μεθόδων καλύτερα αποτελέσματα προκύπτουν από τη χρήση μεθόδων που χρησιμοποιούν περισσότερη 3D πληροφορία σε σχέση με μεθόδους που επεξεργάζονται μόνο RGB εικόνες. Ακόμα, η ακρίβεια προσδιορισμού της πόζας με επιπλέον πληροφορία βάθους από αισθητήρες RGB-D μπορεί αποτελεσματικά να αυξήσει την ακρίβεια της λύσης.

Το βάθος μπορεί να περιγράψει τη γενική μορφολογία του αντικειμένου ενώ το χρώμα συμβάλει στην εγκαθίδρυση ομολογιών. Ωστόσο, ο προσδιορισμός του βάθους μπορεί να είναι δύσκολος για μεταλλικά, γυαλιστερά και ημιδιαφανή αντικείμενα κάτι το οποίο κάνει τις RGB learning-based μεθόδους πιο κατάλληλες. Ένας σημαντικός παράγοντας ο οποίος επηρεάζει την ακρίβεια είναι η εξάρτηση των περιγραφόμενων μεθόδων από τον θόρυβο και στις αλλαγές περιβάλλοντος. Σε αυτή τη περίπτωση learning-based προσεγγίσεις επιφέρουν τα καλύτερα αποτελέσματα. Οι αλγόριθμοι αυτοί εκπαιδεύονται να είναι περισσότερο ανεκτικοί σε σκηνές με πολλά αντικείμενα, διαφορές συνθηκών φωτισμού και θορύβου.

Η εισαγωγή θορύβου στα δεδομένα εισόδου (data augmentation) είναι αναπόσπαστο μέρος της διαδικασίας της εκπαίδευσης, έτσι ώστε η προβλέψεις των μοντέλων να γίνουν όσο γίνεται πιο ανεξάρτητες του θορύβου, της θέσης του αντικειμένου, της εικόνας και του φωτισμού. Ωστόσο, οι μέθοδοι εκμάθησης απαιτούν πολύ μεγάλο όγκο δεδομένων και κατ’ επέκταση είναι απαιτητικές σε αποθηκευ-

τικό χώρο σε σχέση με template based μεθόδους όπου ο αριθμός των tamplates είναι αρκετά μικρότερος από τις εικόνες εκπαίδευσης. Εκτός από το χώρο που χρειάζονται τα δεδομένα εκπαίδευσης, χρειάζεται επιπλέον χώρος για την αποθήκευση του εκπαιδευμένου μοντέλου με τις παραμέτρους και τα βάρη του.

Ο χρόνος που χρειάζεται είναι ακόμα ένας σημαντικός παράγοντας αξιολόγησης μιας μεθόδου. Οι περισσότερες μέθοδοι προσπαθούν να επιτύχουν μία καλή ισορροπία μεταξύ ακρίβειας και χρόνου. Όσο για τις μεθόδους εκμάθησης όταν η διαδικασία εκπαίδευσης έρθει εις πέρας ο χρόνος που απαιτείται για τον προσδιορισμό της 6D πόζας (inference time) είναι αρκετά μικρός. Αντίθετα, για μεθόδους όπως οι template based σε κάθε παράθυρο πρέπει να εξετάζονται πολλαπλές (αν όχι όλες) ταμπλέτες προκειμένου να βρεθεί η καταλληλότερη κάτι το οποίο είναι χρονοβόρο.

Συνοψίζοντας, τα τελευταία χρόνια η προσοχή των ερευνητών έχει στραφεί σε μεγάλο βαθμό στην ανάπτυξη νευρωνικών δικτύων για την λύση του προβλήματος του προσδιορισμού της 6D πόζας αντικειμένων με την ανάπτυξη μεθόδων περιορισμένων εισόδων (RGB) για την επίλυση απαιτητικών προβλημάτων όπως δυναμικά παρασκήνια, επικαλύψεις αντικειμένων, πολλαπλές εμφανίσεις, έλλειψη υφής, γυαλιστερά αντικείμενα κ.ά.

4 Κεφάλαιο 4: Μεθοδολογία και υλοποίηση

Όπως αναφέρθηκε και στην προηγούμενη ενότητα αναζητήθηκε μία μέθοδος η οποία θα χρησιμοποιεί αποκλειστικά RGB εικόνες για την εκτίμηση της 6D πόζας.Με την διερεύνηση των αποτελεσμάτων του BOP Challenge 2019 (Hodaň et al. [2019]) η μέθοδος EPOS (Hodaň et al. [2020]) είχε καλύτερη απόδοση σε σχέση με τις υπόλοιπες RGB μεθόδους. Συγκεκριμένα στο YCB-V (Xiang et al. [2018]) ξεπέρασε όλες τις RGB-D και D μεθόδους και στο T-LESS (Hodan et al. [2017]) πέτυχε τη μεγαλύτερη ακρίβεια. Εκτός από την καλή ακρίβεια που πέτυχε το EPOS έχει και σχετικά μικρούς χρόνους του προσδιορισμού της πόζας με μέσο όρο 0.75s ανά εικόνα (με έναν επεξεργαστή 6-core Intel i7-8700K CPU, 64GB RAM, και κάρτα γραφικών Nvidia P100). Ο παραπάνω χρόνος αφορά μια μη βελτιστοποιημένη πλήρως ως προς το χρόνο πρόβλεψης (σχετική αναφορά στην ενότητα 6). Για τους παραπάνω λόγους επιλέχθηκε ο προσδιορισμός της 6D πόζας της αρπάγης να γίνει με βάση την υλοποίηση του EPOS.

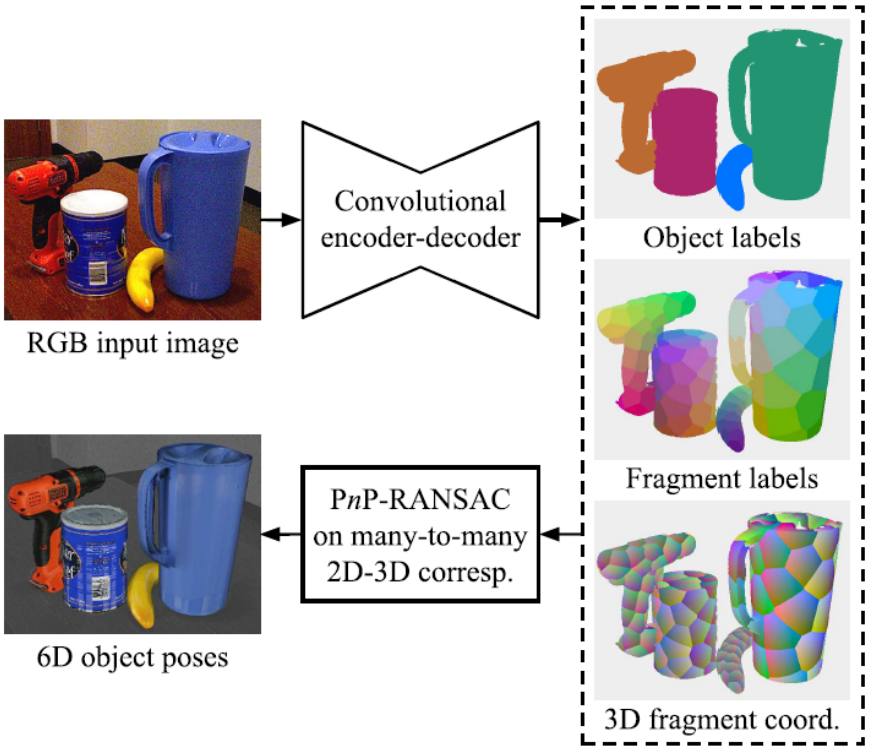
6D object pose estimation method	Image	T-LESS [24]		YCB-V [68]		LM-O [4]		Time
		AR	AR_{MSPD}	AR	AR_{MSPD}	AR	AR_{MSPD}	
EPOS	RGB	**47.6**	**63.5**	**69.6**	**78.3**	**44.3**	**65.9**	0.75
Zhigang-CDPN-ICCV19 [39]	RGB	12.4	17.0	42.2	51.2	37.4	55.8	0.67
Sundermeyer-IJCV19 [59]	RGB	30.4	50.4	37.7	41.0	14.6	25.4	0.19
Pix2Pose-BOP-ICCV19 [48]	RGB	27.5	40.3	29.0	40.7	7.7	16.5	0.81
DPOD-ICCV19 (synthetic) [69]	RGB	8.1	13.9	22.2	25.6	16.9	27.8	0.24
Pix2Pose-BOP_w/ICP-ICCV19 [48]	RGB-D	–	–	**67.5**	**63.0**	–	–	–
Drost-CVPR10-Edges [13]	RGB-D	**50.0**	**51.8**	37.5	27.5	**51.5**	**56.9**	144.10
Félix&Neves-ICRA17-IET19 [55, 53]	RGB-D	21.2	21.3	51.0	38.4	39.4	43.0	52.97
Sundermeyer-IJCV19+ICP [59]	RGB-D	48.7	51.4	50.5	47.5	23.7	28.5	1.10
Vidal-Sensors18 [66]	D	**53.8**	**57.4**	**45.0**	**34.7**	**58.2**	**64.7**	4.93
Drost-CVPR10-3D-Only [13]	D	44.4	48.0	34.4	26.3	52.7	58.1	10.47
Drost-CVPR10-3D-Only-Faster [13]	D	40.5	43.6	33.0	24.4	49.2	54.2	2.20

Σχήμα 42: Αποτελέσματα BOP Challenge Hodaň et al. [2019]. Σύγκριση των διάφορων μεθόδων με βάση την ακρίβεια και το χρόνο στα dataset T-LESS (Hodan et al. [2017]), YCB-V Xiang et al. [2018], LM-O (Brachmann et al. [2014b])

4.1 Εκτίμηση 6D πόζας με τον αλγόριθμο EPOS

Η μέθοδος EPOS:Estimating 6D Pose of Objects with Symmetries (Hodaň et al. [2020]) είναι μία learning-based προσσέγιση η οποία με μοναδικό δεδομένο μια RGB εικόνα μπορεί να εκτιμήσει την 6D πόζα αντικειμένων χωρίς υφή, μερικές η ολικές συμμετρίες, να διαχειριστεί πολλαπλές εμφανίσεις του ίδιου αντικειμένου στην εικόνα αλλά και αποκρύψεις μερών των αντικειμένων.

Η μέθοδος αυτή αντιστοιχεί στην κατηγορία των μεθόδων που βασίζονται σε αντιστοιχίες 2D-3D. Μία απο τις καινοτομίες της είναι η αναπαράσταση των 3D μοντέλων με τη χρήση μικρότερων επιφανειών (surface fragments). Οι αντιστοιχίες προσδιορίζονται με την χρήση ενός συνελικτικού νευρωνικού δικτύου (CNN) τύπου encoder-decoder. Πιο συγκεκριμένα, αντίστοιχα με άλλες μεθόδους που περιγράφηκαν στην ενότητα 3 (πχ. PoseCNN) πραγματοποιεί παράλληλα segmentation και regression. Η διαφορά είναι ότι για κάθε pixel προσδιορίζεται α) η πιθανότητα της παρουσίας κάθε μοντέλου β) η πιθανότητα της παρουσίας συγκεκριμένου fragment του αντίστοιχου μοντέλου και 3) την ακριβή 3D θέση καθενός από τα fragment.



Σχήμα 43: Παραδείγματα αντικειμένων που περιέχονται στο σύνολο των dataset του BOP dataset (Hodaň et al. [2020])

Οι αντιστοιχίες 2D-3D προκύπτουν από ένα CNN. Για το πρόβλημα της αβεβαιότητας λόγω συμμετρίας η μέθοδος μοντελοποιεί την πιθανότητα ένα fragment j ενός αντικειμένου i να αναγνωρίζεται στο pixel $u = (u, v)$ ως:

$$Pr(f = j, o = i|u) = Pr(f = j|o = i, u)Pr(o = i|u)$$

όπου:

- $Pr(f = j|o = i, u)$ η δεσμευμένη πιθανότητα της ύπαρξης συγκεκριμένου fragment j στο pixel u δεδομένης της ύπαρξης του μοντέλου i.
- $Pr(o = i|u)$ η δεσμευμένη πιθανότητα της ύπαρξης του αντικειμένου i στο pixel u

Αντί να προβλέπεται απευθείας η πιθανότητα $Pr(f = j, o = i|u)$ προβλέπονται ξεχωριστά οι $Pr(f = j|o = i, u)$ και $Pr(o = i|u)$. Η πιθανότητα $Pr(f = j, o = i|u)$ μπορεί να είναι μικρή εάν α) το μοντέλο i δεν αναγνωρίζεται στο pixel u ή β) το pixel u αντιστοιχεί σε πολλαπλά fragments λόγω μερικών η ολικών συμμετριών. Αν ισχύει το δεύτερο η πιθανότητα λαμβάνει χαμηλή τιμή ενώ το αντικείμενο μπορεί να αναγνωρίζεται στο συγκεκριμένο pixel. Με την παραπάνω διάσπαση ανεξαρτητοποιείται η αβεβαιότητα λόγω συμμετρίας του μοντέλου από την αβεβαιότητα της ύπαρξης του μοντέλου κάτι το οποίο επιτρέπει στο μοντέλο να διαχειρίζεται συμμετρικά αντικείμενα.

Για τον προσδιορισμό της ακριβής 3D θέσης κάθε fragment χρησιμοποιείται παλινδρόμηση (regression) και έτσι σε κάθε pixel u προβλέπεται η αντίστοιχη 3D θέση σε συντεταγμένες στο σύστημα των fragments (fragment coordinates) σε σχέση με το κέντρο (fragment center) του συστήματος.

Συνολικά για την πρόβλεψη των πιθανοτήτων $Pr(f = j|o = i, u)$ και $Pr(o = i|u)$ και των 3D θέσεων των fragments χρησιμοποιείται ένα βαθύ νευρωνικό δίκτυο με δομή encoder-decoder (DeepLabv3+, Chen et al. [2018]). Για m αντικείμενα, τα οποία αναπαριστώνται από n fragments το καθένα το δίκτυο έχει $4mn + m + 1$ κανάλια εξόδου ($m + 1$ για τις πιθανότητες ύπαρξης του κάθε μοντέλου συν 1 για το background, mn για τις πιθανότητες ύπαρξης των fragments και $3mn$ για τις 3D θέσεις των fragments).

Ενδιαφέρον παρουσιάζει η συνάρτηση του Loss που χρησιμοποιεί το EPOS για την εκπαίδευση. Η παρακάτω συνάρτηση υπολογίζεται για κάθε pixel u και ελαχιστοποιείται κατά την εκπαίδευση κατά μέσο όρο των pixel της εικόνας:

$$L(u) = \overbrace{E\left(\bar{a}(u), a(u)\right)}^{\text{Softmax cross entropy loss}} + \sum_{i \in I} \bar{a}_i(u) \left[\lambda_1 \overbrace{E\left(\bar{b}_i(u), b_i(u)\right)}^{\text{Softmax cross entropy loss}} + \sum_{j \in J} \bar{b}_{ij}(u) \lambda_2 \overbrace{H\left(\bar{r}_{ij}(u), r_{ij}(u)\right)}^{\text{Humber loss}} \right]$$

όπου:

- $\bar{a}(u), a(u)$ οι πιθανότητες $Pr(o = i|u)$ ground truth και προβλεπόμενη αντίστοιχα
- $\bar{b}_i(u), b_i(u)$ οι πιθανότητες $Pr(f = j|o = i, u)$ ground truth και προβλεπόμενη αντίστοιχα
- $\bar{r}_{ij}(u), r_{ij}(u)$ οι ground truth και εκτιμώμενες 3D θέσεις των fragment αντίστοιχα
- I, J τα σύνολα των αντικειμένων και των fragments κάθε αντικειμένου αντίστοιχα.
- λ_1, λ_2 βάρη που αφήνονται στο δίκτυο να προσδιορίσει την καλύτερη τιμής τους για να ισορροπήσουν τα σφάλματα
- E η συνάρτηση σφάλματος softmax cross entropy (βλ. ενότητα 2.4.2.3)
- H η συνάρτηση σφάλματος Humber (Huber [1964]) είναι μία επιλογή για τη συνάρτηση loss η οποία είναι λιγότερο ευαίσθητη σε outliers σε σχέση με την MSE (βλ. ενότητα 2.4.2.3) και δίνεται από τη συνάρτηση:

$$L_k(\bar{y} - y) = \begin{cases} \frac{1}{2}(\bar{y} - y)^2, & |\bar{y} - y| \leq \delta \\ \delta(\bar{y} - y) - \delta^2, & |\bar{y} - y| \geq \delta \end{cases}$$

Κάθε pixel u συσχετίζεται με μία 3D θέση για κάθε fragment όταν η πιθανότητα παρουσίας του μοντέλου αλλά και η πιθανότητα ύπαρξης συγκεκριμένου fragment είναι μεγαλύτερες από κάποια όρια. Μετά τον προσδιορισμό των ομολογιών 2D-3D από το CNN γίνεται εκτίμηση της 6D πόζας του αντικειμένου μέσω ενός αλγορίθμου P3P (Kneip et al. [2011]) από τις αντιστοιχίες οι οποίες προκύπτουν

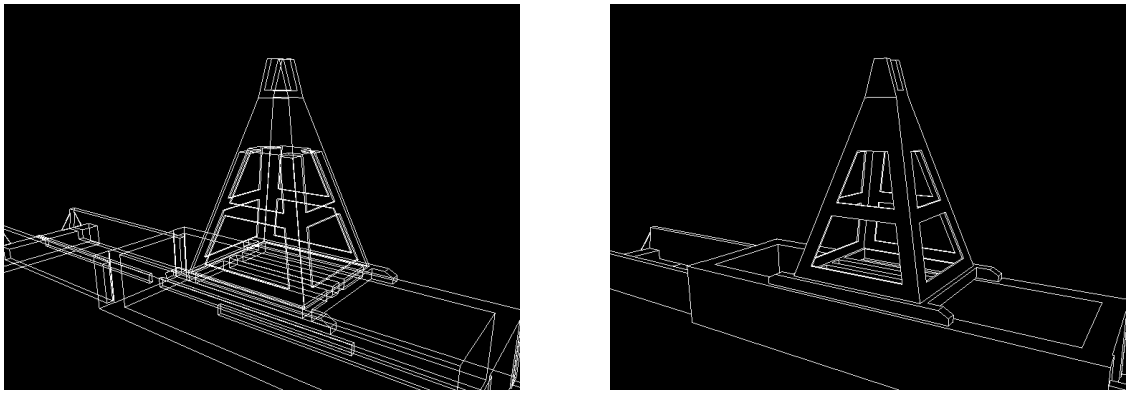
από την εφαρμογή - μιας παραλλαγής του αλγορίθμου RANSAC- του GC-RANSAC (Barath and Matas [2018]). Τέλος, η πόζα βελτιστοποιείται με τη χρήση του EPnP (Lepetit et al. [2009]) από όλα τα σημεία χωρίς πλέον τα outliers (inliers).

4.2 Δημιουργία δεδομένων εκπαίδευσης

4.2.1 Βασική μεθοδολογία rendering με απόκρυψη μη ορατών ακμών

Κατά τα αρχικά στάδια της διπλωματικής εργασίας έγινε ένας εκτενής πειραματισμός με βάση το καθαρά γεωμετρικό μοντέλο της αρπάγης. Συγκεκριμένα, εξετάστηκε η πιθανότητα απεικόνισης του μοντέλου σε wireframe mode αλλά με την απόκρυψη των μη ορατών ακμών από την οπτική γωνία της κάμερας. Τέτοιες αναπαραστάσεις θα μπορούσαν να χρησιμοποιηθούν για την εκπαίδευση ενός αλγορίθμου με τη λογική ότι γνωρίζοντας τη θέση και τη κατεύθυνση κάποιων χαρακτηριστικών ακμών στο αντικείμενο είναι δυνατή η απόκτηση της 6D πόζας τους. Ακόμα, αυτή η διαδικασία αποδείχθηκε πολύ χρήσιμη για την οπτικοποίηση της τελικής εκτιμώμενης 6D πόζας του αντικειμένου για τον οπτικό της έλεγχο.

Όταν γίνεται render ένα μοντέλο με τη παράμετρο `GL_FILL` (polygon fill) οι πίσω ακμές του μοντέλου δεν φαίνονται στο χρήστη λόγω της χρηματισμένης μπροστινής πλευράς. Αν το μοντέλο αναπαρασταθεί σε wireframe mode καμία πλευρά του δεν θα χρωματίζεται και θα είναι διάφανες συνεπώς θα φαίνονται όλες οι ακμές που κανονικά από την οπτική γωνία της κάμερας δεν θα φαίνονταν. Για να εμφανίζονται μόνοι οι ορατές ακμές ο αλγόριθμος που δημιουργήθηκε εκμεταλλεύεται κυρίως τον Z buffer (buffer που περιέχει τις τιμές βάθους για κάθε fragment) αλλά και έναν αλγόριθμο που θα αναγνωρίζει τις εσωτερικές ακμές (ακμές τριγώνων του μοντέλου που δεν αποτελούν πραγματικά ακμές) και θα τις απορρίπτει πριν το rendering του μοντέλου.



Σχήμα 44: Αποτέλεσμα εφαρμογής αλγορίθμου απόκρυψης μη ορατών ακμών (HRL). Χωρίς την απόκρυψη των μη ορατών ακμών (αριστερά), με απόκρυψη των μη ορατών ακμών (δεξιά)

Το παραπάνω γίνεται με την πραγματοποίηση δύο διαδοχικών renderings με διαφορετικές παραμέτρους. Στο πρώτο το μοντέλο αναπαρίσταται με χρωματισμό του δικτύου των τριγώνων που το αποτελούν αποθηκεύοντας σε κάθε θέση πάνω στο μοντέλο το βάθος των pixel στον Depth buffer. Έπειτα καθαρίζεται ο color buffer αλλά όχι depth buffer. Το δεύτερο rendering χρησιμοποιεί την wireframe αναπαράσταση του μοντέλου με το primitive να είναι πλέον οι γραμμές από τις οποίες έχουν αφαιρεθεί οι εσωτερικές ακμές. Οι ορατές ακμές προκύπτουν από έναν έλεγχο (depth test) από τον οποίο περνάνε μόνο όσα fragmetns έχουν τιμή μικρότερη η ίση (παράμετρος `GL_LEQUAL`, βλ. δείγμα κώδικα 17) των προσωρινών τιμών του depth buffer.

Για την παραγωγή αυτών των δύο renderings χρησιμοποιήθηκαν και διαφορετικοί shaders. Αυτό που άλλαξε κυρίως ήταν στον geometry shader λόγω του επιθυμητού primitive κάθε φορά (τρίγωνο

για το πρώτο, γραμμή για το δεύτερο). Η απόκρυψη των εσωτερικών ακμών έγινε σύμφωνα με το δείγμα κώδικα 9.

4.2.2 Εφαρμογή υφής στο γεωμετρικό μοντέλο της αρπάγης

Για τη δημιουργία ενός φωτορεαλιστικού μοντέλου της αρπάγης χρησιμοποιήθηκαν το γεωμετρικό μοντέλο της αρπάγης και RGB εικόνες που την απεικονίζουν από διάφορες οπτικές γωνίες στο πεδίο (Σχήμα 45).

Η υφή του μοντέλου (texture) πρακτικά είναι ένα raster αρχείο που περιέχει την αναγκαία πληροφορία της αναπαράστασης του αντικειμένου. Αυτή δεν είναι ανάγκη να παρουσιάζεται σαν κανονική φωτογραφία της αρπάγης (όπως αυτές που χρησιμοποιήθηκαν για τη δημιουργία της) αλλά μπορεί να είναι ”κομμάτια” του αντικειμένου απο διάφορες εικόνες και μέρη σε αυτές. Το μοντέλο αποκτάει ένα νέο σετ παραμέτρων τα u, v οι οποίες ονομάζονται συντεταγμένες UV (UV coordinates). Τα UV υποδηλώνουν τους δύο άξονες αντί για X και Y. Αντίστοιχα η διαδικασία αυτή λέγεται αντιστοίχηση UV (UV mapping) επειδή αντιστοιχίζονται οι 3D επιφάνειες ενός μοντέλου με τις 2D συντεταγμένες μια εικόνας. Πρακτικά οι παράμετροι u, v του μοντέλου ορίζουν από ποια περιοχή της εικόνας υφής πρέπει να πάρει το χρώμα η κάθε επιφάνεια.

Αρχικά έγινε μια πρώτη προσέγγιση με το open source λογισμικό **Meshlab** (Meshlab [2022]). Το πρόβλημα που αντιμετωπίστηκε εμπίπτει στο ότι σε ιδανικές συνθήκες η υφή που προβάλλεται στο κάθε τρίγωνο του μοντέλου της αρπάγης θα έπρεπε είναι από την εικόνα με την μικρότερη γωνία κάμερας- face normal. Αυτό όμως δεν εξασφαλίζεται (βλ. ενότητα 4.2.2.1), με αποτέλεσμα να υπάρχουν σημαντικές παραμορφώσεις στην απεικόνιση. Τελικά υιοθετήθηκε μια πιο χειροκίνητη διαδικασία στο Blender.

4.2.2.1 Texturing στο Meshlab

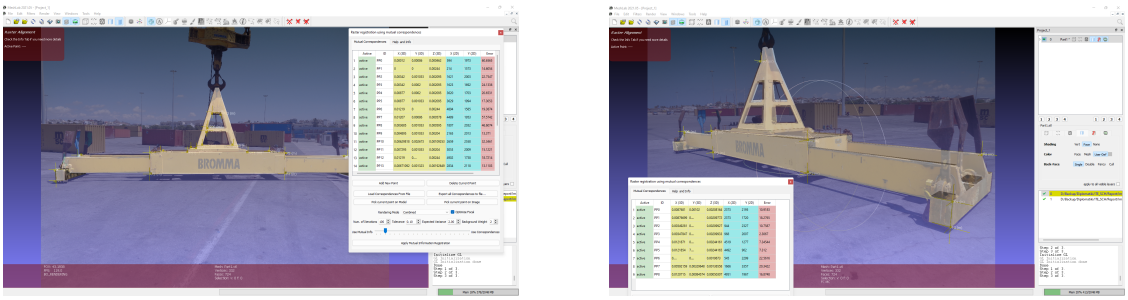
Μετά την εισαγωγή του μοντέλου και των εικόνων στο πρόγραμμα με τη χρήση του εργαλείου *Raster Alignment* πραγματοποιείται μια αρχική ευθυγράμμιση στρέφοντας και μετακινώντας το μοντέλο. Για να γίνει με ακρίβεια η ευθυγράμμιση εικόνας-μοντέλου (προσδιορισμός εξωτερικού προσανατολισμού κάμερας) επιλέγονται ομόλογα σημεία στο μοντέλο και στην εικόνα και εκτιμάται η 6D πόζα του αντικειμένου σε σχέση με τη κάμερα. Είναι δυνατή η ρύθμιση του βάρους των αυτόματων αντιστοιχιών που υπολογίζει το πρόγραμμα και εισαγωγή σημείων χειρωνακτικά από το χρήστη.

Μετά την εκτέλεση της εντολής *“Apply Mutal Information Registration”* εμφανίζονται τα σφάλματα επαναπροβολής. Με βάσει αυτά μπορεί ένα σημείο να αποκλειστεί από τον υπολογισμό (μεγάλο σφάλμα). Όταν πλέον βρεθούν οι καλύτεροι παράμετροι εξωτερικού προσανατολισμού οπτικά και ποσοτικά (μέσω των σφαλμάτων) πρέπει να γίνει καθορισμός του συγκεκριμένου σημείου λήψης. Αυτό γίνεται με την εντολή *“Set Raster Camera* $\rightarrow$ *Get shot* $\rightarrow$ *Apply”*. Η διαδικασία επαναλαμβάνεται και για δεύτερη εικόνα η οποία λήφθηκε υπό μεγαλύτερη γωνία σε σχέση με την μπροστινή πλευρά της αρπάγης. Τελικά, οι θέσεις των δύο σημείων λήψεις προέκυψαν όπως στην εικόνα 47α.

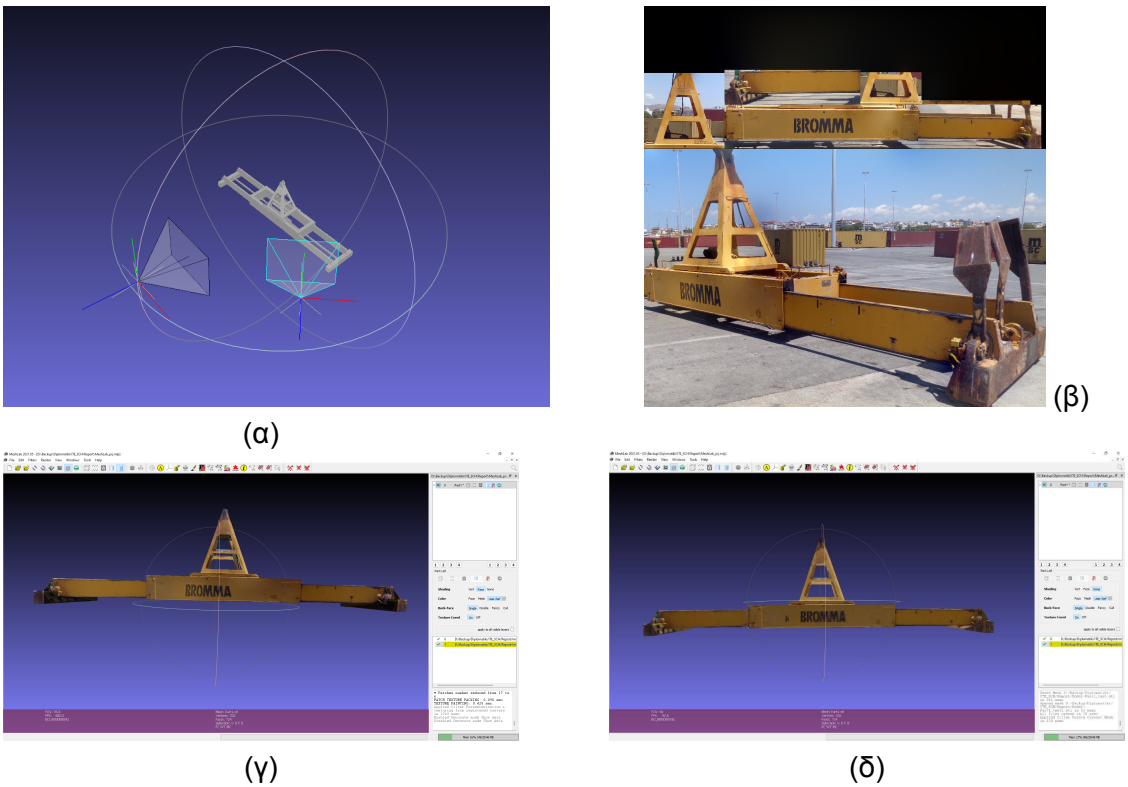
Για να γίνει η προβολή των εικόνων στο μοντέλο και να δημιουργηθεί το texture αρχείο χρησιμοποιήθηκαν δύο μέθοδοι. Η πρώτη ήταν μέσω του αλγορίθμου *Filters*$\rightarrow$*Texture*$\rightarrow$*Parametrization + texturing from registered rasters* και η δεύτερη με τη χρήση του *Render*$\rightarrow$*Texture/Camera*$\rightarrow$*Project active raster’s color to current mesh, filling the texture*. Η διαφορά των δύο μεθόδων έγκειται στις εικόνες που χρησιμοποιούν για να προκύψει η υφή του μοντέλου. Στην πρώτη περίπτωση η υφή προκύπτει από το συνδυασμό όλων των raster που συμμετέχουν ενώ στη δεύτερη γίνεται προβολή συγκεκριμένης εικόνας πάνω στο μοντέλο. Αυτό που παρατηρείται είναι ότι στην πρώτη μέθοδο είναι ότι αντί η υφή του μοντέλου για την μπροστινή πλευρά να προκύψει χρησιμοποιώντας τη προβολή της

Σχήμα 45: Ενδεικτικές εικόνες που χρησιμοποιήθηκαν για την διαμόρφωση της υφής της αρπάγης

εικόνας που είναι παράλληλη με την μεγάλη πλευρά της αρπάγης χρησιμοποιείται η πλαϊνή λήψη. Δηλαδή, η εικόνα θα έπρεπε να επιλέγεται με κριτήριο την μικρότερη γωνία μεταξύ του άξονα λήψης και το διάνυσμα της επιφάνειας του κάθε τριγώνου του μοντέλου. Η συγκεκριμένη διαδικασία δίνει καλά αποτελέσματα κυρίως σε πιο πυκνά μοντέλα και όχι σε απλά μοντέλα όπως στη προκειμένη περίπτωση με αποτέλεσμα το τελικό texture να έχει παραμορφώσεις (Σχήμα 47α). Για το παραπάνω λόγο υιοθετήθηκε μία χειροκίνητη διαδικασία εφαρμογής της υφής με τη χρήση του ανοιχτού λογισμικού **Blender**.



Σχήμα 46: Επιλογή ομόλογων σημείων για το προσδιορισμό του εξωτερικού προσανατολισμού της κάμερας



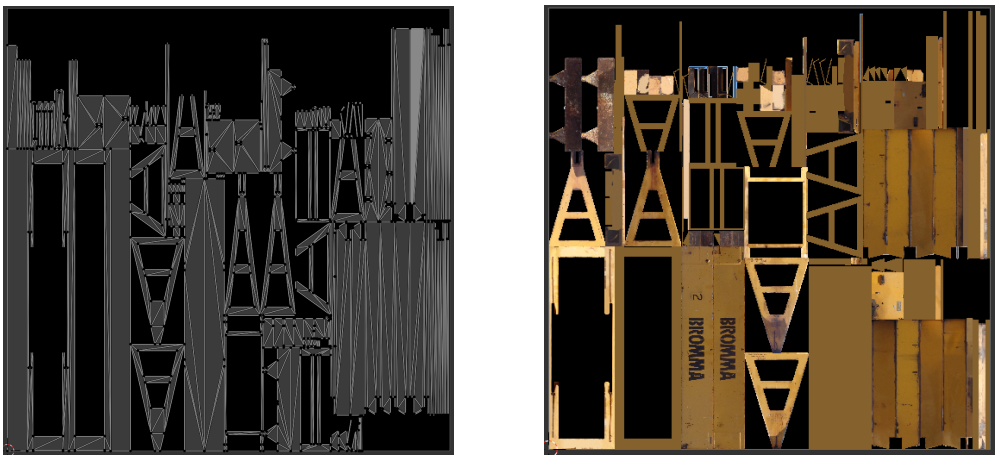
(α) (β)

(γ) (δ)

Σχήμα 47: (α) Προσδιορισμένες θέσεις λήψεις, (β) Αρχείο texture που προέκυψε με χρήση 2 εικόνων (γ) Μοντέλο με υφή αποτέλεσμα της πρώτης μεθόδου, (δ) Μοντέλο με υφή αποτέλεσμα της δεύτερης μεθόδου.

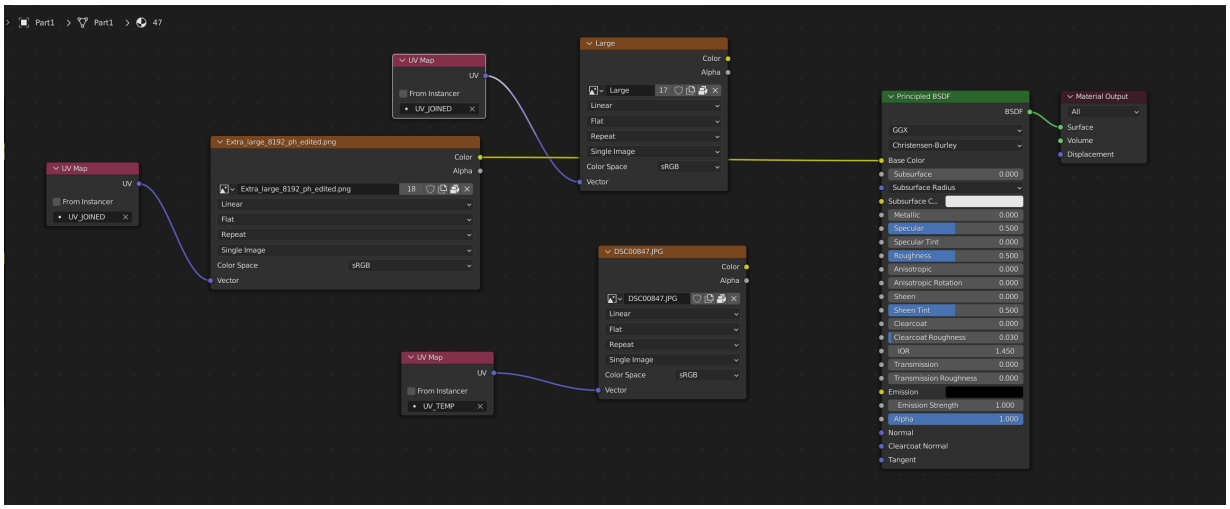
4.2.2.2 Texturing στο Blender

Για να προκύψει το αρχείο texture στο Blender (Blender [2023]) αρχικά πρέπει να δημιουργηθεί το ανάπτυγμα του μοντέλο με το να οριστούν οι ακμές ως προς τις οποίες θα “ξεδιπλωθεί” (seams). Επειδή το μοντέλο της αρπάγης έχει έντονη γεωμετρία με μεγάλες αλλαγές κλίσεων στις επιφάνειες μπορεί να χρησιμοποιηθεί η εντολή Smart UV project που χρησιμοποιεί τις “αιχμηρές” ακμές ή ακμές μεταξύ τριγώνων με μεγάλη διαφορά στη διεύθυνση των αντίστοιχων κάθετων διανυσμάτων (Sharp edges) του μοντέλου για να γίνει το ανάπτυγμα. Τα κομμάτια που προκύπτουν ονομάζονται από το λογισμικό islands (νησίδες). Ακόμα, για να μειωθεί η συνολική έκταση που καλύπτει το ανάπτυγμα του μοντέλο στο UV map και κατ’ επέκταση και στο τελικό texture αρχείο χρησιμοποιείται η εντολή pack UV islands η οποία κατανέμει τις επιφάνειες με τον καλύτερο δυνατό τρόπο ώστε να πιάνουν τον ελάχιστο χώρο. Το ζητούμενο μετά από αυτό είναι να συμπληρωθεί το UV map με πληροφορία από τις εικόνες πεδίου (Σχήμα 48).



Σχήμα 48: Το ανάπτυγμα του μοντέλου (UV Map) (Αριστερά), η τελική εικόνα υφής (texture file) (Δεξιά)

Μετά τη δημιουργία του UV map, για κάθε μία από τις επιφάνειες του μοντέλου που πρόκειται να εφαρμοστεί υφή δημιουργείται ένα καινούργιο υλικό (material). Πηγαίνοντας στην κάθε εικόνα και μετακινώντας τις κορυφές του αναπτύγματος των επιλεγμένων επιφανειών προκύπτει η πηγή υφής κάθε 2D επιφάνειας από τις διάφορες εικόνας.

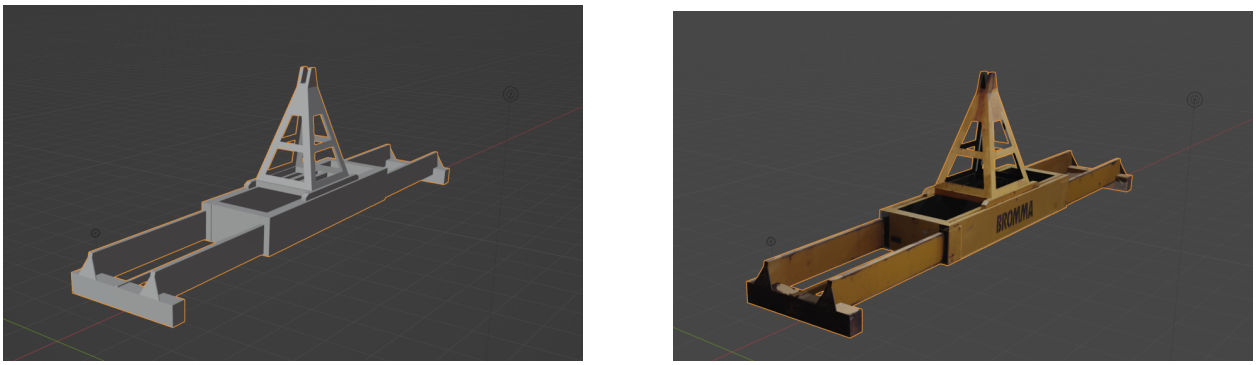


Σχήμα 49: Η διαδικασία δημιουργίας του raster υφής (texture) με τη χρήση των shader nodes

Σημεία του μοντέλου που δεν ενδιέφεραν τόσο στην εφαρμογή (πχ. το κάτω η το εσωτερικό τμήμα) αποδόθηκαν με τη χρήση ενός ενιαίου χρώματος ενώ αλλά που εν μέρη φαινόντουσαν στις εικόνες αλλά όχι εξ ολοκλήρου συμπληρώθηκαν με αντιγραφή υφής από άλλο τμήμα της αρπάγης. Όταν η διαδικασία ολοκληρωθεί και για τα τρίγωνα που ενδιαφέρουν έχει αντιστοιχηθεί ένα material το οποίο περιέχει τη προβολή της εικόνας στη συγκεκριμένη επιφάνεια δημιουργείται το τελικό texture με μια διαδικασία που λέγεται **bake texture** (Σχήμα 50).

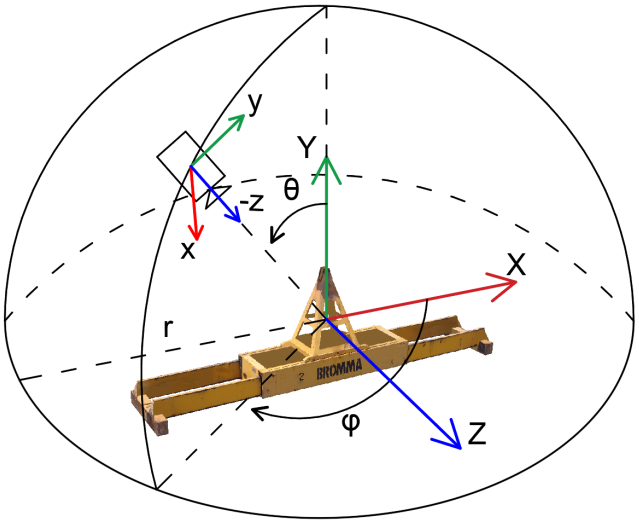
4.2.3 Αυτόματη παραγωγή εικόνων και δεδομένων 6D πόζας

Για την παραγωγή των εικόνων εκπαίδευσης είναι σημαντικός ο ορισμός ενός συστήματος αναφοράς, το οποίο θα συμβάλει στη μεθοδική συλλογή των εικόνων. Για το σκοπό αυτό επιλέχθηκε ένα σφαιρικό σύστημα αναφοράς με παραμέτρους ϕ, θ, ρ. Ο άξονας x αντιστοιχεί με τη μεγάλη διάσταση της αρπάγης, ο άξονα y με το ύψος της ενώ ο z το βάθος. Για τους παραπάνω παραμέτρους ισχύουν τα εξής: $\phi \epsilon [0, 360),\ \theta \epsilon [0, 90],\ \rho \epsilon \{10, 15, 20, 25, 30\}$. Η γωνία θ επιλέχθηκε μέχρι 90 μοίρες διότι λήψεις



Σχήμα 50: Αριστερά: Το αρχικό γεωμετρικό μοντέλο. Δεξιά: Το τελικό textured μοντέλο με τη χρήση του Blender

που απεικονίζουν το κάτω μέρος της αρπάγης δεν χρειάζονται λόγω της εφαρμογής (στις πραγματικές λήψεις η αρπάγη συνήθως δεν ξεπερνά το ύψος της κάμερας από το έδαφος). Ακόμα, επιλέχθηκαν αποστάσεις από 10 έως 30μ δεδομένης πάλι της απόστασης της αρπάγης στις πραγματικές συνθήκες. Τα βήματα των γωνιών και των αποστάσεων ρυθμίστηκαν ανάλογα με τις δοκιμές που έγιναν κατά την εκπαίδευση του αλγορίθμου EPOS (βλ. ενότητα 4.1).



Σχήμα 51: Ημισφαίριο συλλογής εικόνων

Για να γίνει η δειγματοληψία των εικόνων η κάμερα περιστρέφεται ακολουθώντας την πορεία των νοητών σφαιρών που σχηματίζονται με τις τιμές των παραπάνω παραμέτρων. Οι 3D καρτεσιανές συντεταγμένες της κάμερας C εκφράζονται με τη χρήση των παραμέτρων χρησιμοποιώντας τις σφαιρικές καρτεσιανές συντεταγμένες.

$$C = \begin{Bmatrix} r\cos\phi\sin\theta \\ r\sin\phi\sin\theta \\ r\cos\theta \end{Bmatrix}$$

Για την αποτελεσματικότερη διαχείριση του προσανατολισμού και της θέσης της κάμερας δημιουργήθηκε μία κλάση (Camera class - Πίνακας 2). Η τελευταία περιέχει συναρτήσεις που σε κάθε επανάληψη ενημερώνουν τις τιμές των παραμέτρων και υπολογίζονται εκ νέου οι συντεταγμένες της κάμερας. Η συνάρτηση `camera.UpdateView()` (βλ. δείγμα κώδικα 16) υπολογίζει ένα πίνακα τύπου LookAt (de Vries [2020]) ο οποίος μετασχηματίζει τις συντεταγμένες όλων των σημείων στο σύστημα

της κάμερας (**camera viewspace**). Στη συνάρτηση αυτή δίνεται σαν παράμετρος το σημείο προς το οποίο θα προσανατολιστεί ο z άξονας της κάμερας (η διεύθυνση - Look at), οι συντεταγμένες της κάμερας ως προς το world origin, και η κατεύθυνση του διανύσματος που θεωρούμε πάνω (σε αυτή τη περίπτωση ταυτίζεται με την κατεύθυνση του κατακόρυφου άξονα της αρπάγης).

Σε αυτή τη φάση οι συντεταγμένες βρίσκονται στο σύστημα της κάμερας και σε ομογενείς μορφή. Ο επόμενος πίνακας που υπολογίζεται είναι ο προβολικός πίνακας (projection matrix). Αυτός υπολογίζεται με τη χρήση του πίνακα **K** της κάμερας.

$$\boldsymbol{Projection}_{GL} = \begin{bmatrix} \frac{2K_{00}}{w} & \frac{-2K_{01}}{w} & \frac{(w-2K_{02}+2u_0)}{w} & 0 \\ 0 & \frac{-2K_{11}}{h} & \frac{(h-2K_{12}+2v_0)}{h} & 0 \\ 0 & 0 & \frac{(-z_{far}-z_{near})}{(z_{far}-z_{near})} & \frac{-2z_{far}z_{near}}{(z_{far}-z_{near})} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

όπου:

- w : Το πλάτος του αισθητήρα της κάμερας σε pixel
- h : Το μήκος του αισθητήρα της κάμερας σε pixel
- K_{3x3} : Ο πίνακας εσωτερικού προσανατολισμού της κάμερας όπως αυτός περιγράφηκε στην ενότητα 2.2.1.2
- z_{near}, z_{far} : Οι τιμές πέρα από τον οποίων πραγματοποιείται αποκοπή των εμφανιζόμενων αντικειμένων όταν ισχύει $z_{near} < z < z_{far}$

Οι NDC πλέον συντεταγμένες αποδίδονται από την OpenGL σε συντεταγμένες raster εικόνες με εύρος όσο το ορισμένο μέγεθος του framebuffer στον οποίο γίνονται rendered (βλ. ενότητα 2.3.5). Η παραπάνω διαδικασία αποτελεί τον τρόπο με τον οποίο προκύπτουν οι εικόνες της αρπάγης. Για την αποθήκευση της 6D πόζας του αντικειμένου δηλαδή έναν πίνακα $[R|T]$ πρέπει ο τελικός πίνακας που διαβιβάζεται για το render στην OpenGl να αναστραφεί και να πολλαπλασιαστεί με τον πίνακα I_{CV}. Ο μετασχηματισμός γίνεται γιατί η OpenGL αντιμετωπίζει τους πίνακες με κύρια διάσταση τις στήλες (column major) ενώ η OpenCV τις γραμμές (row major) και ο πολλαπλασιασμός με τον πίνακα I_{CV} λόγω των διαφορετικών συστημάτων αναφοράς. Οπότε το τελικό R|T που αντιστοιχεί σε κάθε πόζα προκύπτει ως:

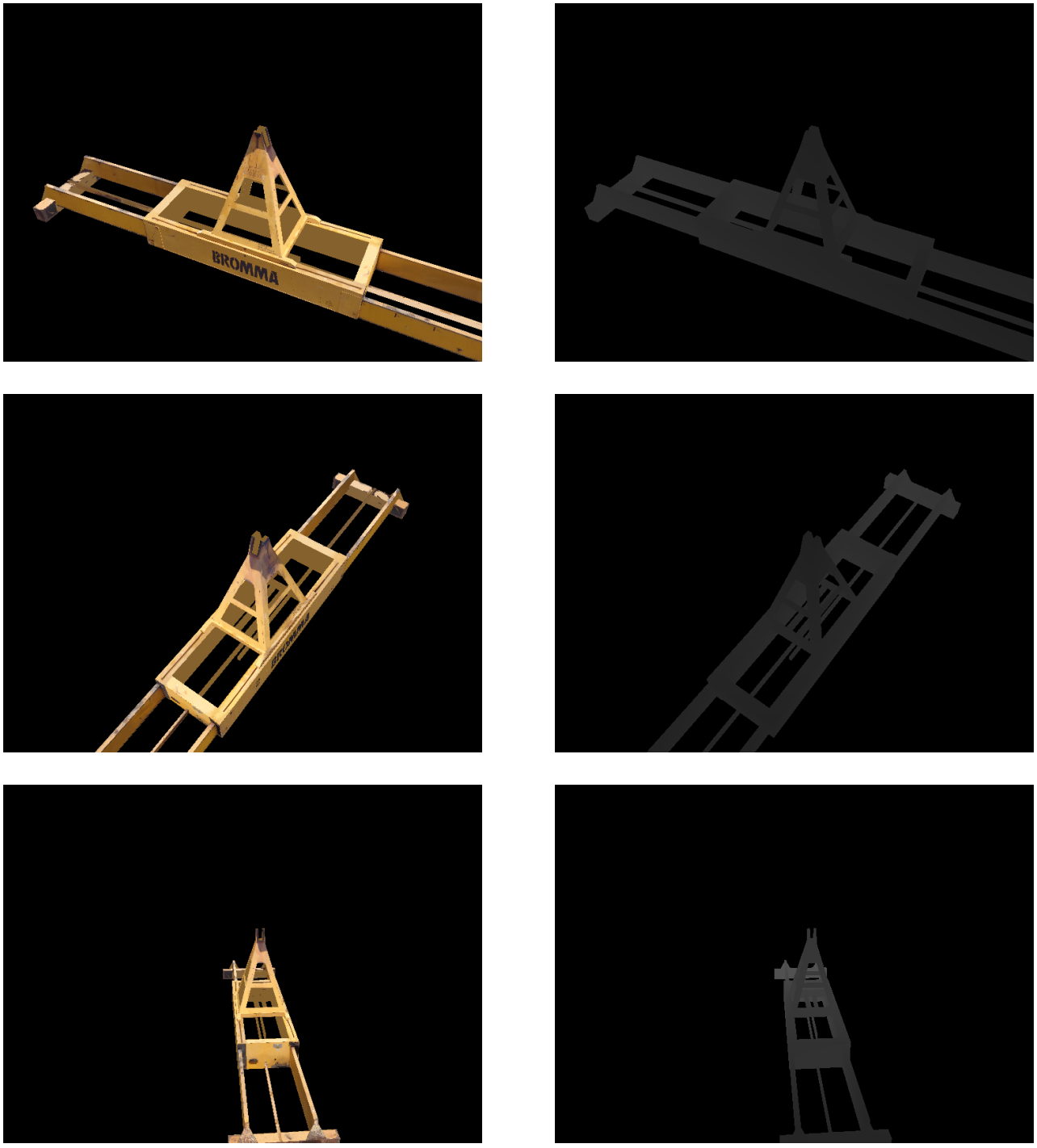
$$[R|T]_{out} = {[R|T]_{GL}}^T * I_{CV} = {[R|T]_{GL}}^T * \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Η παραπάνω διαδικασία παράγει τις RGB rendered εικόνες του textured μοντέλου αλλά και αντίστοιχες εικόνες βάθους (depth - σχέση 25). Αυτό που αλλάζει για την παραγωγή των depth εικόνων είναι η χρήση διαφορετικών vertex, fragment shaders και ο τύπος των τιμών του buffer. Για να προκύψει μια εικόνα depth πρέπει τα σημεία να εκφραστούν στο σύστημα της κάμερας προτού γίνει ο

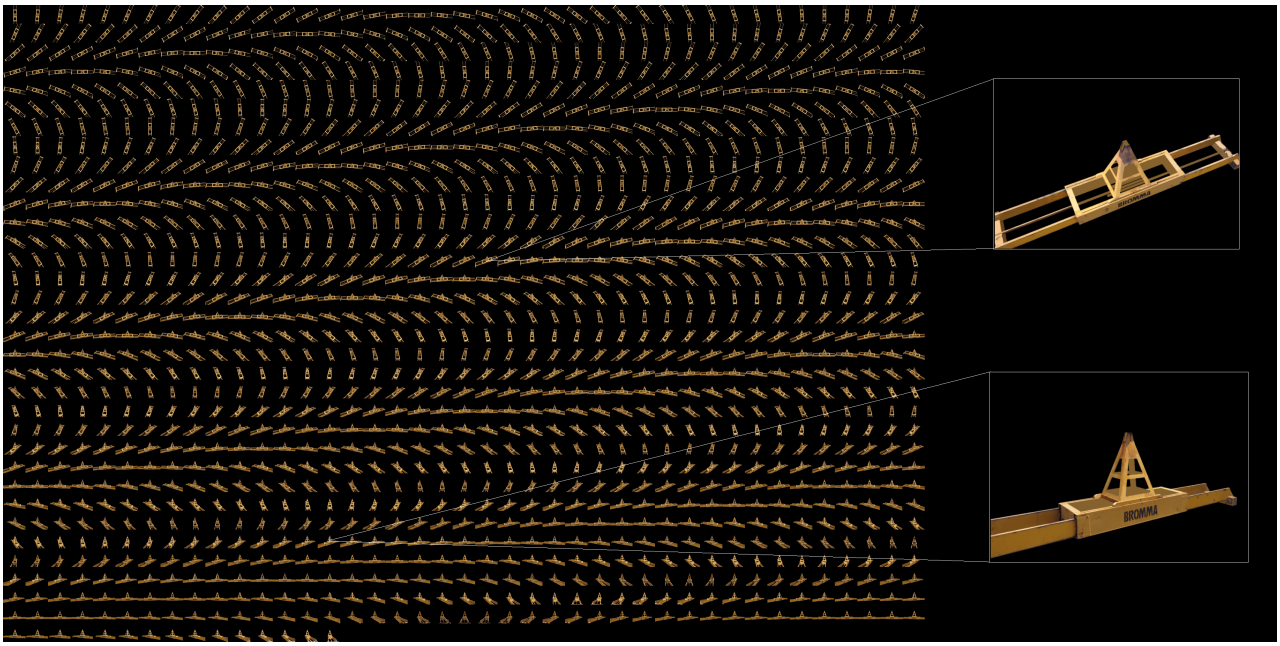
προβολικός μετασχηματισμός. Η εικόνα βάθους προκύπτει από τη παρακάτω σχέση:

$$Depth = -C_z = -[R|T]_{GL} * \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{25}$$

με τον fragment shader να αποδίδει “χρώμα” ανάλογα με το βάθος του σημείου όπως φαίνεται στο δείγμα κώδικα 11. Για την ανάγνωση του buffer δεσμεύεται μνήμη μεταβλητών FLOAT και έπειτα τα περιεχόμενα του στρογγυλοποιούνται στον κοντινότερο θετικό ακέραιο 16bit (unsigned short). Τέλος, αποθηκεύεται με τη χρήση της OpenCV σαν ένα κανάλι (δείγμα κώδικα 12).



Σχήμα 52: RGB rendered εικόνες και οι αντίστοιχες depth



Σχήμα 53: Training dataset που δημιουργήθηκε με την αυτόματη συλλογή εικόνων

4.3 Μετρικές για την ποσοτική αξιολόγηση του σφάλματος 6D πόζας

Για την ποσοτική αξιολόγηση του προσδιορισμού της 6D πόζας της αρπάγης χρησιμοποιήθηκαν διάφορες μετρικές οι οποίες χρησιμοποιούνται στη βιβλιογραφία (Hodan et al. [2020]). Ο κύριος διαχωρισμός τους είναι ότι υπολογίστηκαν μετρικές που ποσοτικοποιούν το σφάλμα στη στροφή της πόζας και αντίστοιχα της θέσης. Τα παραπάνω υπολογίστηκαν τόσο σαν απόλυτα σφάλματα (άμεση διαφορά της πόζας από την αληθινή) όσο και σαν σχετικά. Για το σφάλμα στη στροφή υπολογίστηκε το απόλυτο γωνιακό σφάλμα (Absolute angular error) ως τη γωνία που πρέπει να στραφεί η εκτιμώμενη πόζα R_e γύρω από μοναδιαίο διάνυσμα έτσι ώστε να προκύψει η αληθινή R_g (Huynh [2009]).

$$AAE = \arccos\left(\frac{1}{2}(tr(R_g {R_e}^T) - 1)\right)$$

όπου με tr συμβολίζεται το ίχνος του πίνακα το οποίο υπολογίζεται ως $\sum_{i=1}^{N} a_{ii}$ με a_{ii} τα στοιχεία της διαγωνίου. Το απόλυτο σφάλμα στη θέση (Absolute Positional Error) προκύπτει ως η διαφορά των μέτρων των διανυσμάτων μετάθεσης της αληθινής $\boldsymbol{t_g}$ και της εκτιμώμενης θέσης $\boldsymbol{t_e}$:

$$APE = ||\boldsymbol{t_g} - \boldsymbol{t_e}||$$

Η δεύτερη κατηγορία μετρικών περιλαμβάνει σχετικά σφάλματα για τη στροφή και τη μετάθεση. Το σχετικό σφάλμα στροφής καθώς και το σχετικό σφάλμα στη μετάθεση(Lepetit et al. [2009]) υπολογίζονται από τις σχέσεις:

$$RAE = \frac{||\boldsymbol{q_g} - \boldsymbol{q_e}||}{||\boldsymbol{q_e}||}$$

$$RPE = \frac{||\boldsymbol{t_g} - \boldsymbol{t_e}||}{||\boldsymbol{t_e}||}$$

όπου τα q_e, q_g είναι τα μοναδιαία τετραδρόνια που αντιστοιχούν στους πίνακες στροφής R_e, R_g (2.1.6) και $||\boldsymbol{q_g} - \boldsymbol{q_e}|| = \sqrt{(q_g0 - q_e0)^2 + (q_g1 - q_e1)^2 + (q_g2 - q_e2)^2 + (q_g3 - q_e3)^2}$. Εκτός από τα παραπάνω χρησιμοποιήθηκε και μία πιο γενική μετρική η οποία συνυπολογίζει πρακτικά το σφάλμα στη στροφή

και την μετάθεση με το να υπολογίζει την μέση απόσταση των κορυφών (vertices) του μοντέλου στην αληθινή πόζα από τις αντίστοιχες στην εκτιμώμενη πόζα (Average distance for distinguishable (ADD) objects).

$$ADD = \frac{1}{N}\sum_{i=1}^{N} ||(\boldsymbol{R_g x_i} + \boldsymbol{t_g}) - (\boldsymbol{R_e x_i} + \boldsymbol{t_e})||$$

όπου:

- $\boldsymbol{R_g}, \boldsymbol{t_g}$ είναι η αληθινή πόζα(ground truth) και $\boldsymbol{R_e}, \boldsymbol{t_e}$ είναι εκτιμώμενη.
- $\boldsymbol{x_i}$ είναι η 3D θέση κάθε κόμβου

5 Κεφάλαιο 5: Αποτελέσματα και αξιολόγηση

5.1 Εκπαίδευση αλγορίθμου

Για τη διαδικασία της εκπαίδευσης του αλγορίθμου ορίστηκαν δύο σετ δεδομένων με διαφορετική λειτουργία. Το σετ εκπαίδευσης (training set) και το σετ ελέγχου (validation set). Στο σύνολο τους δημιουργήθηκαν 6840 RGB εικόνες οι οποίες χωρίστηκαν σε 5130 (75%) για το training set και 1710 (25%) για το validation set. Ο υπολογισμός των εικόνων που θα προκύψουν μπορεί να υπολογιστεί πριν την παραγωγή τους με την γνώση των παραμέτρων ϕ, θ, r ως:

$$N = len(distances) * (int(180/d\phi)) * (int(90/d\theta) + 1)$$

όπου:

- $distances$ είναι ένας πίνακας με τις επιθυμητές αποστάσεις σε μέτρα πχ. [5,10,15,..] και η συνάρτηση len δηλώνει το αριθμό των αποστάσεων στον πίνακα
- $d\phi, d\theta$ το βήμα που αυξάνεται η γωνία ϕ και θ αντίστοιχα.

5.1.1 Δεδομένα εκπαίδευσης - training set

Το training set είναι αυτό με το οποίο θα εκπαιδευτεί το δίκτυο με μία διαδικασία ανάλογη αυτής που περιγράφηκε στην ενότητα 2.4.1. Περιέχει RGB εικόνες που έχουν γίνει render όπως αναφέρθηκε στην ενότητα 4.2.3 για τις οποίες έχουν οριστεί οι κατάλληλες παράμετροι ϕ, θ, r έτσι ώστε να υπάρχει επάρκεια στο δεδομένα εκπαίδευσης και οι γωνίες συλλογής των εικόνων να είναι αρκετά πυκνές. Αναλυτικότερα το training set περιλαμβάνει:

- 5130 RGB εικόνες με τις παραμέτρους ϕ, θ, r που αναφέρονται στην ενότητα 4.2.3.
- τους αντίστοιχους πίνακες $[R|T]$ που συνοδεύουν τις πόζες αναφοράς (GT) σε format που ορίζεται από τον αλγόριθμο EPOS
- συνοδευτικά αρχεία που περιέχουν το πίνακα K που χρησιμοποιήθηκε για το render των εικόνων καθώς και το μέγεθος των εικόνων σε γραμμές και στήλες.

5.1.2 Δεδομένα ελέγχου - validation set

Το validation set χρησιμοποιείται για τον έλεγχο της ακρίβειας του εκπαιδευμένου μοντέλου. Τα δεδομένα του validation (1710) εικόνες προέκυψαν από τον τυχαίο διαχωρισμό των συνολικών εικόνων σε training και validation set. Αυτό σημαίνει ότι οι διαδοχικές εικόνες δεν συνεπάγονται και διαδοχικές γωνίες ϕ, θ. Πρακτικά τα δεδομένα σε ένα validation set ακολουθούν την ίδια κατανομή με τα δεδομένα εκπαίδευσης αλλά δεν έχουν χρησιμοποιηθεί στο training set. Αυτό αποτελεί έναν έλεγχο για το αν το μοντέλο όντως έμαθε να κάνει σωστές προβλέψεις και σε καινούργιες εικόνες και δεν έχει απλά προσαρμοστεί αυστηρά στις εικόνες εκπαίδευσης (overfitting - 2.4.2.6). Σε πολλές πλατφόρμες (πχ. Pytorch, Tensorflow) η διαδικασία του training και του validation γίνεται παράλληλα κατά της διαδικασία της εκπαίδευσης έτσι ώστε να παρατηρούνται τα σφάλματα του training και του validation set ταυτόχρονα καθώς και αν υπάρχει overfitting. Ωστόσο στη συγκεκριμένη υλοποίηση του EPOS πρώτα ολοκληρώνεται η εκπαίδευση και στη συνέχεια ο έλεγχος με το validation dataset. Αυτό που είναι πρακτικά το σημαντικό είναι να ελαχιστοποιηθεί το σφάλμα του validation set.

5.1.3 Test set

Το test set έχει βασική διαφορά από το validation set. Χρησιμοποιείται για την αξιολόγηση του μοντέλου σε πραγματικές εικόνες της εφαρμογής και σε πολλές εφαρμογές δεν χρειάζονται δεδομένα αναφοράς (GT) όπως στο validation set. Ωστόσο, για να γίνει ποσοτική ανάλυση πρέπει να υπάρχουν δεδομένα αναφοράς. Μια ακόμα σημαντική διαφορά είναι ότι δεν πρέπει να παρθούν αποφάσεις με την επίβλεψη του σφάλματος του test set. Τα αποτελέσματα του validation υποδεικνύουν τις κινήσεις που πρέπει να γίνουν στην ρύθμιση των υπερ-παραμέτρων για να βελτιωθεί η απόδοση του μοντέλου. Δηλαδή υπάρχει “πρόσβαση” σε αυτά ενώ δεν πρέπει να γίνονται αλλαγές με βάση τα αποτελέσματα του test set γιατί τότε στο μοντέλο εισάγεται ένας παράγοντας μεροληψίας (bias) του χειριστή. Με άλλα λόγια είναι σαν ο χειριστής να κατευθύνει το μοντέλο στο να κάνει τη πρόβλεψη που επιθυμεί.

5.1.4 Μετασχηματισμοί για τη συμβατότητα των συστημάτων

Το EPOS απαιτεί τόσο το σύστημα του μοντέλου όσο και της κάμερας να είναι δεξιόστροφα με τον άξονα z του μοντέλου να δείχνει προς τα πάνω όταν το μοντέλο βρίσκεται στην “φυσιολογική” του πόζα (“standing naturally up-right”). Για την περίπτωση της αρπάγης ο άξονας αυτός είναι παράλληλος με τη διεύθυνση του ύψους της. Ακόμα, προϋποθέτει ότι το σύστημα αναφοράς του μοντέλου έχει ως αρχή το κέντρο του Bounding Box (BB) του μοντέλου και η μονάδα μέτρησης των συντεταγμένων να είναι τα χιλιοστά (mm). Για να μετασχηματιστεί το αρχικό μοντέλο σύμφωνα με τις παραπάνω απαιτήσεις αυτό μπορεί να γίνει μέσω κάποιου προγράμματος όπως το Blender ή κατά την εκτέλεση των αλγορίθμων προγραμματιστικά με τον ορισμό κάποιων πινάκων μετασχηματισμού. Για την αλλαγή του συστήματος αναφοράς αρκεί να γίνει μετάθεση της κάθε πόζας κατά το διάνυσμα

$$t_{BB} = \left[\frac{(X_{max} - X_{min})}{2}, \frac{(Y_{max} - Y_{min})}{2}, \frac{(Z_{max} - Z_{min})}{2} \right]$$

ή χρησιμοποιώντας τον πίνακα:

$$T_{bb} = \begin{bmatrix} 1 & 0 & 0 & T_x \\ 0 & 1 & 0 & T_y \\ 0 & 0 & 1 & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

ενώ η αλλαγή του άξονα y και z γίνεται μέσω αριστερόστροφης στροφής 90^o ως προς τον άξονα x (ο άξονας y γίνεται z) και αντιστρέφοντας τη φορά του νέου y άξονα. Αυτό γίνεται με τον πίνακα:

$$R_{yz} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

5.1.5 Ρύθμιση υπερ-παραμέτρων και δοκιμές

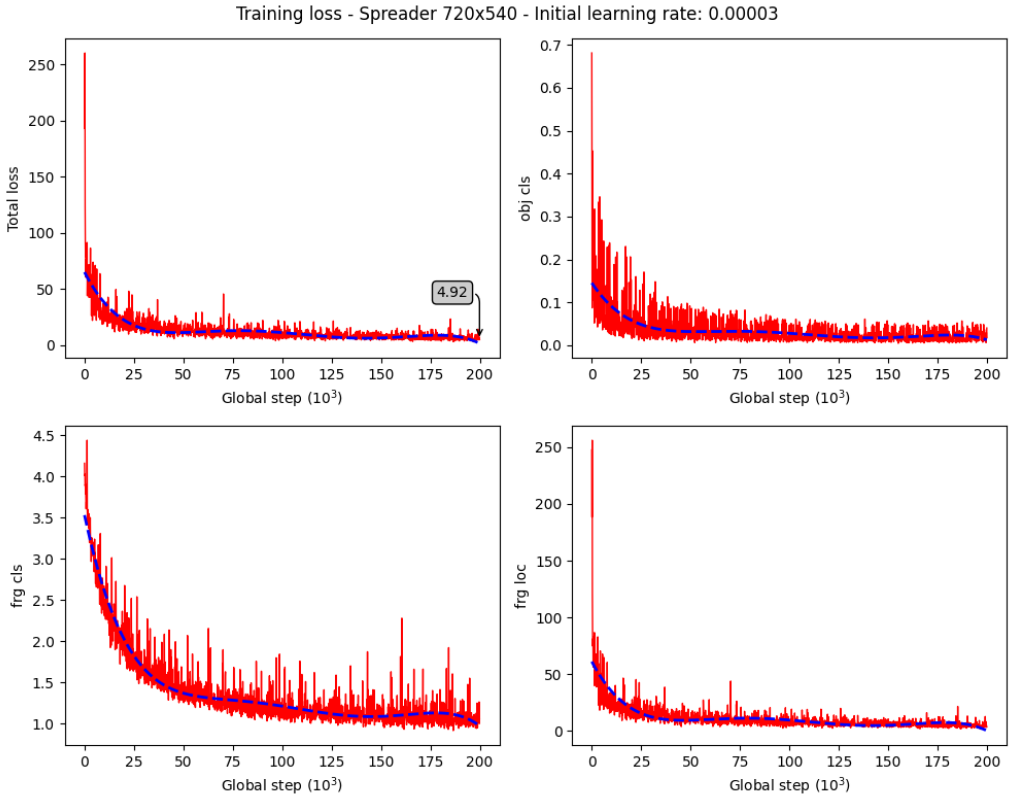
Πριν αρχίσει η εκπαίδευση πρέπει να ρυθμιστούν διάφοροι παράμετροι που επηρεάζουν τη συμπεριφορά του μοντέλου. Όπως αναφέρθηκε στην ενότητα 2.4.2.2 η πιο σημαντική παράμετρος είναι το learning rate. Για να βρεθεί το κατάλληλο learning rate επιλέγεται μια αρχική τιμή και γίνονται διαδοχικές δοκιμές εκπαίδευσης του μοντέλου παρακολουθώντας το loss. Ο στόχος είναι το loss να μειώνεται γρήγορα χωρίς πολλές διακυμάνσεις. Έγιναν συστηματικές δοκιμές με αλλαγή του learning rate και του learning power το οποίο αντιστοιχεί στο πόσο γρήγορα μειώνεται το learning rate με την αύξηση των επαναλήψεων. Το βασικό πρόβλημα που παρατηρήθηκε είναι ότι όταν το learning rate ρυθμίστηκε στην προτεινόμενη τιμή για datasets όπως το tless (0.0001) το loss από τις πρώτες επαναλήψεις έφτανε πολύ μεγάλες τιμές με αποτέλεσμα να μη μπορεί να επανέρθει (NaN, βλ. Πίνακα 4). Αυτό ήταν ένδειξη ότι έπρεπε να μειωθεί το αρχικό learning rate περαιτέρω.

Δοκιμή	Initial learning rate	Learning Power	Training steps	Loss
1	10^{-4}	0.9	200k	NaN
2	9×10^{-5}	0.9	200k	NaN
3	8×10^{-5}	0.9	200k	NaN
4	7×10^{-5}	0.9	200k	NaN
5	6×10^{-5}	0.9	200k	NaN
6	5×10^{-5}	0.9	200k	NaN
7	4×10^{-5}	0.7	200k	NaN
8	3×10^{-5}	0.7	200k	**4.92**

Πίνακας 4: Δοκιμές training - Hyperparameter tuning

Με τις παραμέτρους της δοκιμής 8 προέκυψε loss ίσο με 4.92. Η συγκεκριμένη υλοποίηση του EPOS δεν προσφέρει κάποια μετρική για την καλύτερη κατανόηση της απόδοσης του μοντέλου κατά την εκπαίδευση. Οι συναρτήσεις του loss χρησιμοποιούνται για να ελαχιστοποιηθεί το loss με τις επαναλήψεις αλλά δεν είναι κατάλληλες για την εξαγωγή συμπερασμάτων για την απόδοση. Οι υπόλοιπες τιμές που εμφανίζονται στα διαγράμματα του σχήματος 54 αντιπροσωπεύουν:

- Object classification loss (Obj cls): Αναφέρεται στο σφάλμα αναγνώρισης της κατηγορίας του αντικειμένου. Σε αυτήν τη περίπτωση το μόνο αντικείμενο που καλείται να αναγνωρίσει το μοντέλο είναι η αρπάγη. Οπότε οι δύο κατηγορίες είναι η αρπάγη και το background. Επειδή το background είναι μαύρο το μοντέλο μαθαίνει γρήγορα να το ταξινομεί οπότε το αντίστοιχο loss πέφτει πολύ γρήγορα και σταθεροποιείται κοντά στο 0. Επειδή πρόκειται για loss μεταβλητής βάσει κατηγορίας αυτό είναι της μορφής 2.4.2.3.
- Fragment classification loss (Frg cls): Ο αριθμός των fragments ορίστηκε σε 64 και αυτό το loss



Σχήμα 54: Διαγράμματα training loss

αναφέρεται στο σφάλμα της ταξινόμησης τους. Παρατηρείται ότι κατά την διάρκεια της εκπαίδευσης μειώνεται σταδιακά και τελικά σταθεροποιείται κοντά στο 1.

- Fragment localization loss (frg loc): Αναφέρεται στο σφάλμα στη θέση των fragments και το loss είναι συνεχόμενης μεταβλητής. Αρχικά όπως και σε όλους τους τύπους σφάλματός παρατηρείται μεγάλη διακύμανση των τιμών κάτι που οφείλεται στην διαδικασία Stohastic Gradient Descent (SGD) και του batch size που έχει τιμή 1.

5.1.6 Πειραματικά αποτελέσματα validation

Στις ενότητες 5.1.1 και 5.1.2 αναφέρθηκε ότι το 25% των δεδομένων που δημιουργήθηκαν χρησιμοποιήθηκε για έλεγχο. Για την αξιολόγηση της απόδοσης του μοντέλου χρησιμοποιήθηκαν οι μετρικές που αναπτύχθηκαν στην ενότητα 4.3. Επειδή το μοντέλο παρουσιάζει συμμετρία ως προς τον άξονα z υπολογίστηκε το σφάλμα της κάθε πόζας και της αντίστοιχης συμμετρικής της. Το τελικό σφάλμα προκύπτει το μικρότερο εκ των δύο:

$$AAE = \min\left\{AAE(\boldsymbol{R_e}), AAE(M \cdot \boldsymbol{R_e})\right\}$$

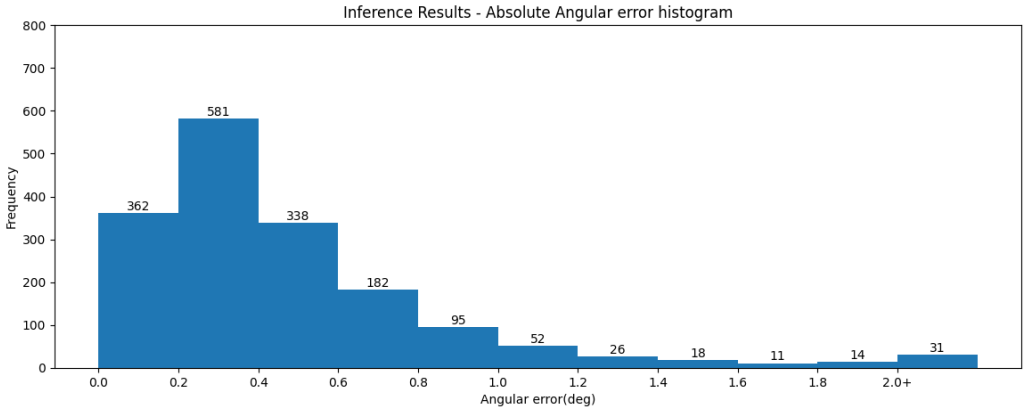
$$RAE = \min\left\{RAE(\boldsymbol{q_e}), RAE(\boldsymbol{q'_e})\right\}$$

όπου:

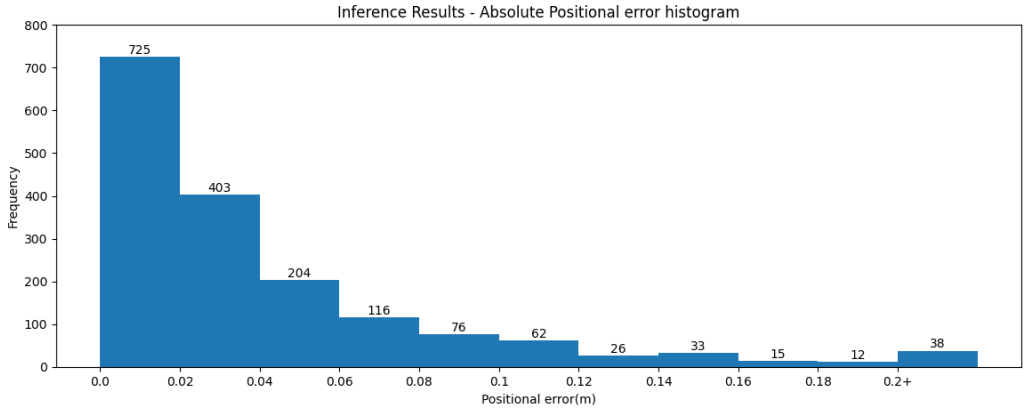
- $M = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ ο πίνακας για τη στροφή κατά π γύρω από τον άξονα z
- $\boldsymbol{R_e}$ η εκτιμώμενη πόζα

- $\boldsymbol{q'_e}$ το τετραδρόνιο που αντιστοιχεί στο πίνακα στροφής $M \cdot \boldsymbol{R_e}$ (2.1.6 - Σχέση 13)

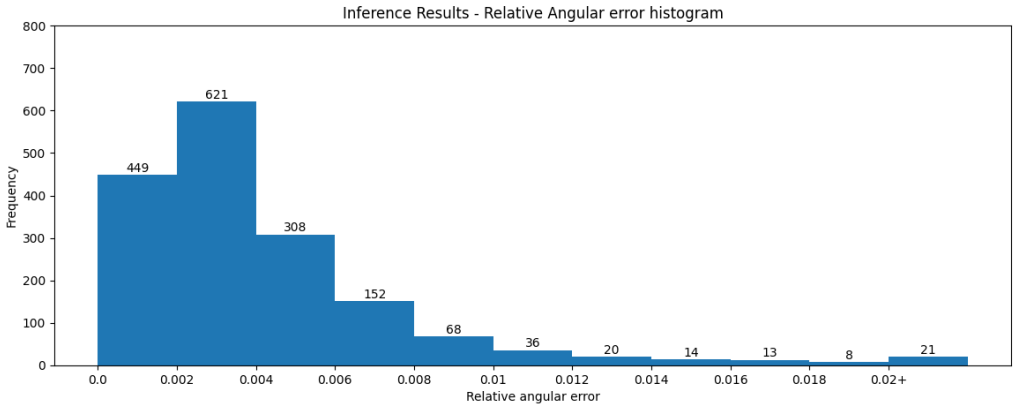
Για κάθε τύπο σφάλματος δημιουργήθηκαν διαφορετικά διαγράμματα. Τα διαγράμματα κατανομής δίνουν μια γενική εικόνα για το εύρος του σφάλματος της πλειονότητας των εικόνων ενώ τα γραμμικά (line plots) παρουσιάζουν πως αλλάζει το σφάλμα ανάλογα με την απόσταση και τη στροφή της αρπάγης. Στο σχήμα 55 παρατηρείται απόλυτο γωνιακό σφάλμα μικρότερο της 1^o στην πλειονότητα των εικόνων ενώ στο σχήμα 56 απόλυτο σφάλμα θέσης κάτω από 0.1μ. Ακόμα, στα σχήματα 57,58 τα αντίστοιχα σχετικά σφάλματα κυμαίνονται γενικά κάτω από 0.01 ενώ στο σχήμα 59 το σφάλμα ADD βρίσκεται κάτω από 0.04μ. Στο σχήμα 60 φαίνεται η αναμενόμενη μείωση της ακρίβειας με την αύξηση της απόστασης της κάμερας από την αρπάγη καθώς αυτή αντιπροσωπεύεται από λιγότερα pixel.



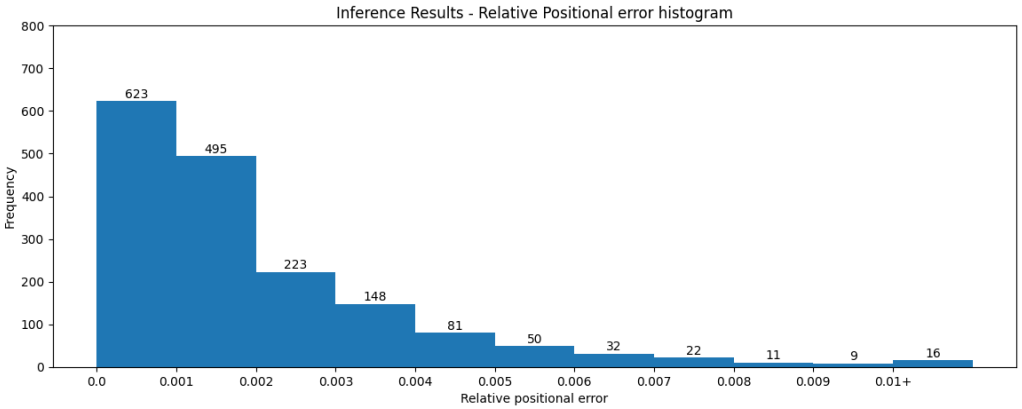
Σχήμα 55: Κατανομή Absolute Angular Error (AAE)



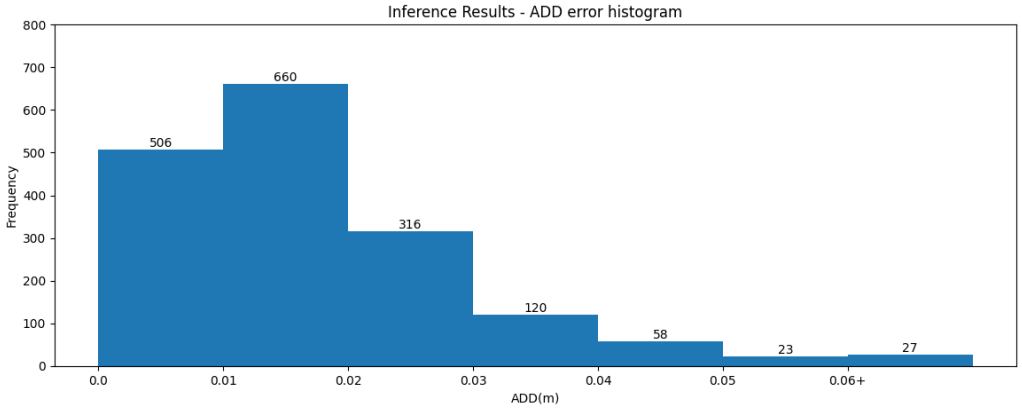
Σχήμα 56: Κατανομή Absolute Positional Error (APE)



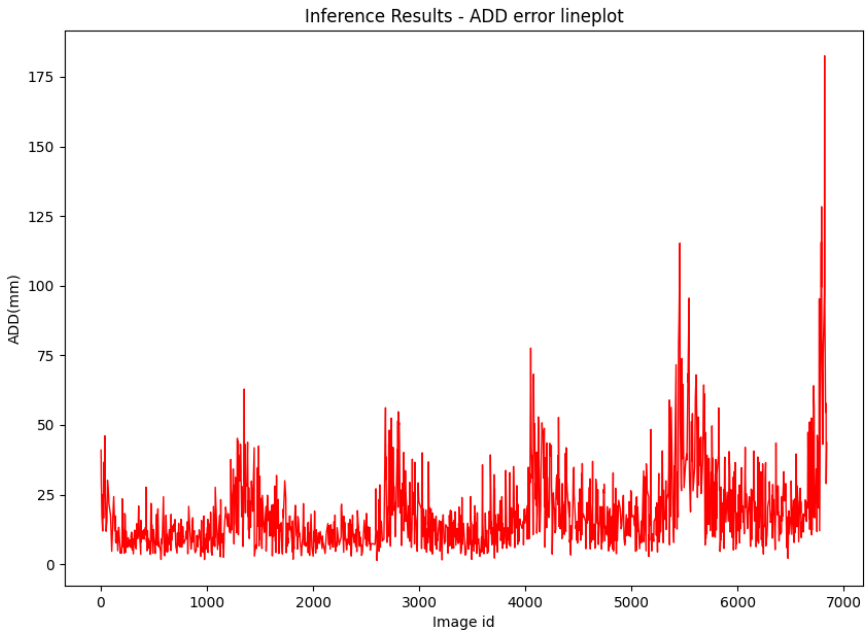
Σχήμα 57: Κατανομή Relative Angular Error (RAE)



Σχήμα 58: Κατανομή Relative Positional Error (RPE)



Σχήμα 59: Κατανομή σφάλματος ADD



Σχήμα 60: Γραμμική αναπαράσταση του σφάλματος ADD σε σχέση με το id της εικόνας

5.2 Σύγκριση αλγορίθμου με δεδομένα αναφοράς από πραγματικές εικόνες φορτο-εκφόρτωσης

5.2.1 Περιγραφή δεδομένων

Οι εικόνες που χρησιμοποιήθηκαν για την αξιολόγηση του μοντέλου είναι δημόσια διαθέσιμες (Lourakis and Pateraki [2022b]) και προέρχονται από βίντεο τα οποία καταγράφουν τη διαδικασία φόρτωσης/εκφόρτωσης container από και σε φορτηγά πλοία. Τα βίντεο καταγράφηκαν από κάμερα που ήταν τοποθετημένη στην καμπίνα του χειριστή του γερανού και σε απόσταση περίπου 20μ από την αποβάθρα. Οι εικόνες απεικονίζουν την αρπάγη με μεταβαλλόμενο παρασκήνιο (background) με διακυμάνσεις στην φωτεινότητα, την ύπαρξη σκιών αλλά και την απόκρυψη μερών της από άλλα αντικείμενα (occlusion). Κάθε εικόνα αυτών των δεδομένων συνοδεύεται από την αντίστοιχη ground truth πόζα της αρπάγης. Λόγω των συνθηκών και του περιβάλλοντος της αρπάγης πολλοί τρόποι απόκτησης των GT δεν μπορούσαν να υλοποιηθούν . Η χρήση δεκτών RTK GNSS (Lourakis et al. [2020]) δεν μπορούσε να εφαρμοστεί λόγω δυσκολίας εγκατάστασης τους πάνω στην αρπάγη και το γεγονός ότι οι μετρήσεις τους μπορεί να επηρεάζονται αρνητικά από παρεμβολές με το μεταλλικό σκελετό της αρπάγης. Ακόμα, δεν μπορούσαν να τοποθετηθούν φωτοσταθερά (Garrido-Jurado et al. [2014]) διότι αυτό θα άλλαζε τη φυσική εικόνα της αρπάγης και μπορεί να είχε αρνητικό αποτέλεσμα στον προσδιορισμό της πόζας της. Για τους παραπάνω λόγους ακολουθήθηκε μια ημιαυτόματη διαδικασία.

Αρχικά, αναγνωρίζονται στην εικόνα χαρακτηριστικές ακμές τις αρπάγης και μία εκτίμηση της πόζας προκύπτει από την εφαρμογή ενός αλγορίθμου Perspective-n-Line (PnL - Wang et al. [2019]) μαζί με έναν αλγόριθμο RANSAC (Fischler and Bolles [1981]) για την διαχείριση των outliers. Η τελική πόζα της αρπάγης προκύπτει από την βελτίωση της εκτίμησης περαιτέρω με τη χρήση μεθόδου ελαχίστων τετραγώνων και την ελαχιστοποίηση του σφάλματος επαναπροβολής (reprojection error) μεταξύ των πραγματικών ακμών και των εκτιμήσεων τους με τη μέθοδο Levenberg-Marquardt (Lourakis [2004]). Για την απόκτηση της πόζας της αρπάγης για κάθε εικόνα που προκύπτει από ένα βίντεο χρησιμοποιείται ένας 3D tracker (Lourakis and Pateraki [2021]) ο οποίος αρχικοποιείται για την πρώτη εικόνα

με τη διαδικασία που περιγράφηκε παραπάνω με τον ορισμό κάποιων αντιστοιχιών ακμών χειροκίνητα. Το αντικείμενο παρακολουθείται από τον tracker μέχρι το αποτέλεσμα του να αποκλίνει σημαντικά. Όταν αυτό συμβαίνει ο tracker ξανα-αρχικοποιείται με τον αλγόριθμο PnL και τη προσαρμογή Levenberg-Marquardt.

Με αυτό το τρόπο προέκυψαν όλες οι 6D πόζες της αρπάγης στις ακολουθίες εικόνων. Το πλεονέκτημα αυτής της μεθόδου είναι ότι επέτρεψε την εξαγωγή δεδομένων αναφοράς με μία αυτόματη διαδικασία με περιορισμένη απαίτηση για χειροκίνητο προσδιορισμό της. Ωστόσο, τα δεδομένα αναφοράς αποτελούν για την ακρίβεια pseudo ground truth και σε μερικές περιπτώσεις μπορεί να παρουσιάζουν αποκλίσεις από τα πραγματικά.

Στον πίνακα 5 φαίνεται το όνομα του κάθε dataset, από πόσα frames αποτελείται και στοιχεία του πίνακα K (f_x, c_x, f_y, c_y).

Dataset	Αριθμός εικόνων	f_x, c_x, f_y, c_y
0621_144051	538	1825.95758, 1034.02348, 1826.84981, 802.98851
0621_143959	581	1825.95758, 1034.02348, 1826.84981, 802.98851
090531	692	1222.81331, 1022.09675, 1224.01279, 718.52760
0621_124449	1300	1825.95758, 1034.02348, 1826.84981, 802.98851
083611	1060	1222.81331, 1022.09675, 1224.01279, 718.52760
081824	1336	1222.81331, 1022.09675, 1224.01279, 718.52760
091956	804	1222.81331, 1022.09675, 1224.01279, 718.52760
095324	695	1222.81331, 1022.09675, 1224.01279, 718.52760

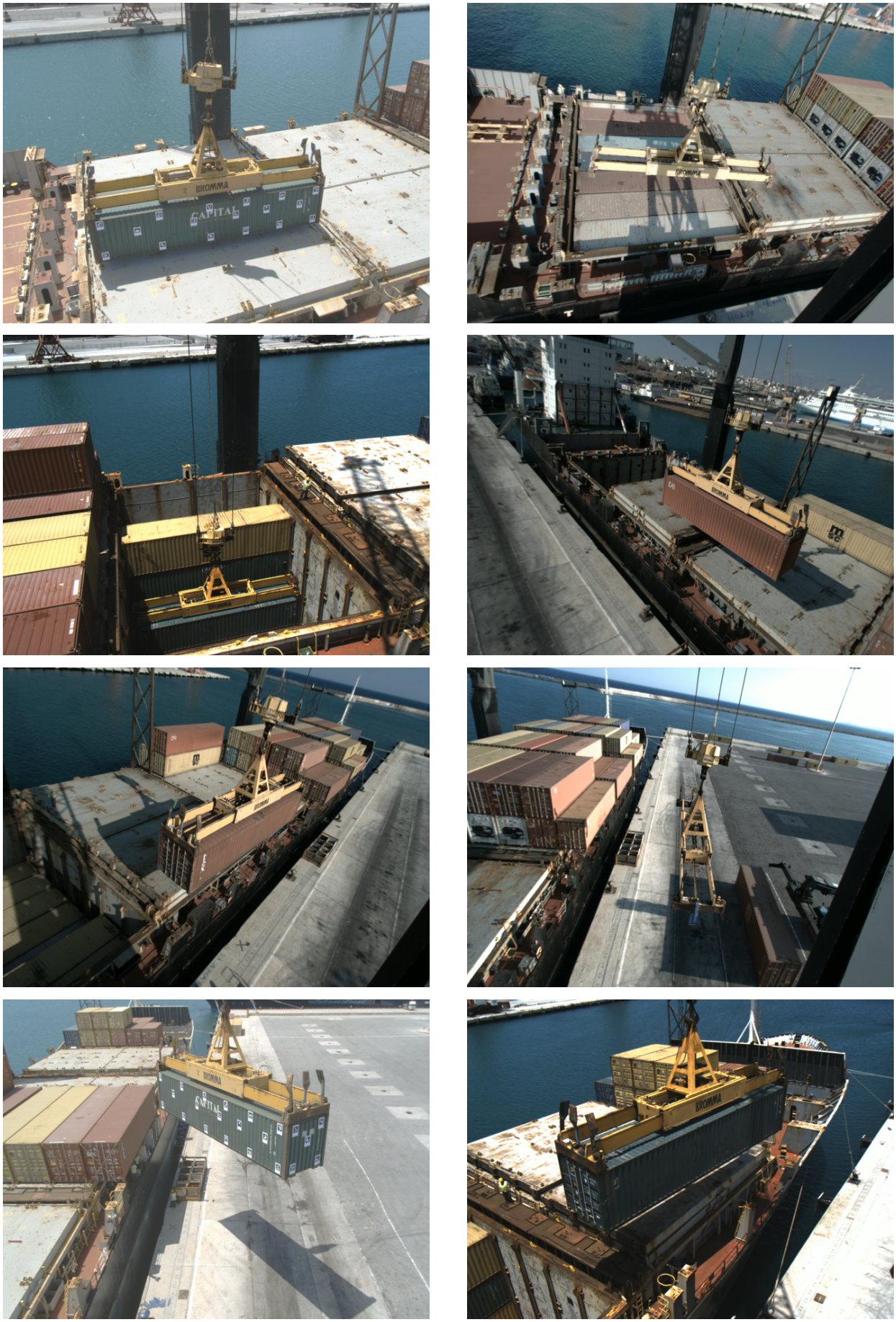
Πίνακας 5: Περιγραφή δεδομένων πραγματικών εικόνων

5.2.2 Αφαίρεση φόντου και χρήση μασκών

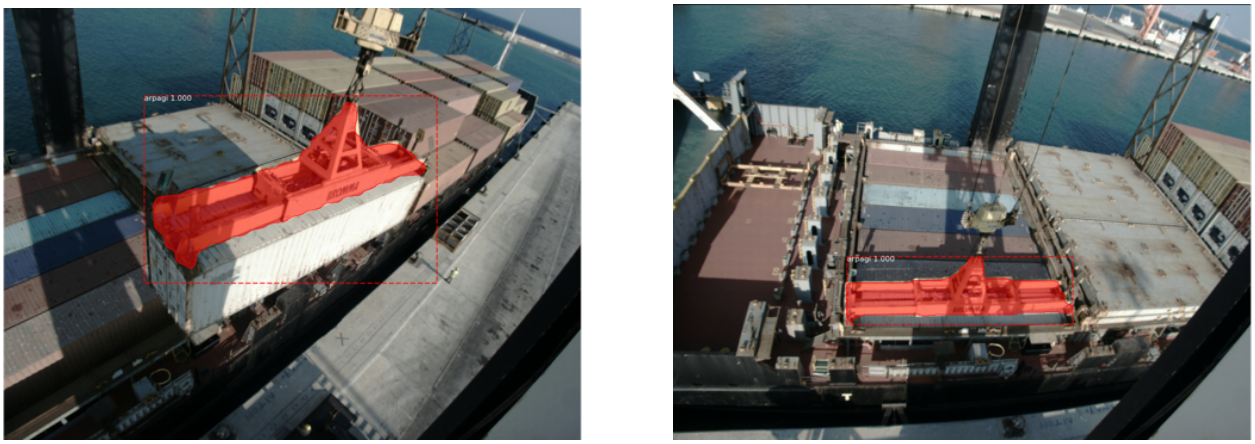
Οι πραγματικές εικόνες με τις εικόνες που χρησιμοποιήθηκαν για το training διαφέρουν ουσιαστικά λόγω του δυναμικού παρασκηνίου, τις συνθήκες φωτισμού, την μη κεντραρισμένη θέση της αρπάγης και την απόκρυψη μερών της κατά τη διαδικασία της φορτοεκφόρτωσης container. Είναι αναμενόμενο το μοντέλο να μην κάνει σωστές προβλέψεις της πόζας όταν του δοθούν εικόνες με δυναμικό background. Αυτό συμβαίνει γιατί οι εικόνες με τις οποίες εκπαιδεύτηκε είχαν ομοιόμορφο παρασκήνιο. Το μη κεντράρισμα της αρπάγης επίσης μπορεί να επηρεάζει σε κάποιο βαθμό το αποτέλεσμα γιατί δεν υπάρχει μεταβολή στην θέση της αρπάγης στα δεδομένα εκπαίδευσης πέρα από την αλλαγή των γωνιών και των αποστάσεων. Για τους παραπάνω λόγους αναπτύχθηκε μία μεθοδολογία με την οποία εξάγεται μία 2D μάσκα της αρπάγης στην εικόνα μεσω κατάτμησης (segmentation) και έπειτα κεντράρεται στο πρωτεύον σημείο της εικόνας με τη χρήση προβολικού μετασχηματισμού (ομογραφίας).

Για την εξαγωγή της μάσκας εκπαιδεύτηκε ένας αλγόριθμος Mask R-CNN[1] (Region-Based Convolutional Neural Network). Αρχικά, έγινε λήψη δειγμάτων εκπαίδευσης (annotation) της αρπάγης για 224 εικόνες εκπαίδευσης όπου οι 188 χρησιμοποιήθηκαν για εκπαίδευση και οι άλλες 36 για έλεγχο. Το annotation έγινε με τη χρήση του εργαλείου VGG Image Annotator (Dutta and Zisserman [2019]). Το μέγεθος των εικόνων επιλέχθηκε να είναι 1024 x 768 λόγω της διαθέσιμης μνήμης της κάρτας γραφικών. Τα αποτελέσματα ήταν αρκετά ακριβή χωρίς πολύ μεγάλη διακύμανση εκτός από κάποια frames στα οποία υπήρχαν κενά στις μάσκες και ένα εφέ “κυματισμού” στο όρια της μάσκας.

[1] Στα πλαίσια της διπλωματικής του Γιάννη Πρίφτη



Σχήμα 61: Δείγμα εικόνων από τα dataset αξιολόγησης



Σχήμα 62: Εξαγωγή 2D μάσκας

Μετά την εξαγωγή της μάσκας αυτή πρέπει να εφαρμοστεί στην αρχική εικόνα (Σχήμα 63). Για να γίνει αυτό γίνεται ανασύσταση στο αρχικό μέγεθος της εικόνας (resize). Με την εφαρμογή της μάσκας προκύπτει μία εικόνα όπου το background έχει αντικατασταθεί με μαύρα pixel. Ωστόσο, σε αυτήν την εικόνα η προβολή της του κέντρο της αρπάγης δεν ταυτίζεται με το πρωτεύον σημείο της εικόνας (δεν είναι κεντραρισμένη στην εικόνα). Για το λόγο αυτό εφαρμόζεται μία ομογραφία της μορφής:

$$\begin{bmatrix} wx \\ wy \\ w \end{bmatrix} = (KRK^{-1}) \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

όπου:

- ο πίνακας των εσωτερικών παραμέτρων της κάμερας
- R ο πίνακας στροφής που αντιπροσωπεύει τη στροφή του διανύσματος Ox του σχήματος 5 ώστε να ταυτιστεί με το διάνυσμα u (principal ray) όπου u είναι το πρωτεύον σημείο της εικόνας. Η στροφή αυτή υπολογίζεται με βάση την υποενότητα 2.1.5.

Για να υπολογιστεί το διάνυσμα (ray) που αντιστοιχεί στο κέντρο της αρπάγης στην αρχική εικόνα πρέπει να υπολογιστούν οι εικονοσυντεταγμένες του κέντρου του BB (βλ. δείγμα κώδικα 13 και ενότητα 2.3.5). Αυτό γίνεται παράλληλα με τη διαδικασία του rendering με την απόκρυψη των μη ορατών ακμών. Τελικά, τα δύο διανύσματα προκύπτουν ως:

$$a = K^{-1} \begin{bmatrix} u_x \\ u_y \\ 1 \end{bmatrix}, b = K^{-1} \begin{bmatrix} bb_x \\ bb_y \\ 1 \end{bmatrix}$$

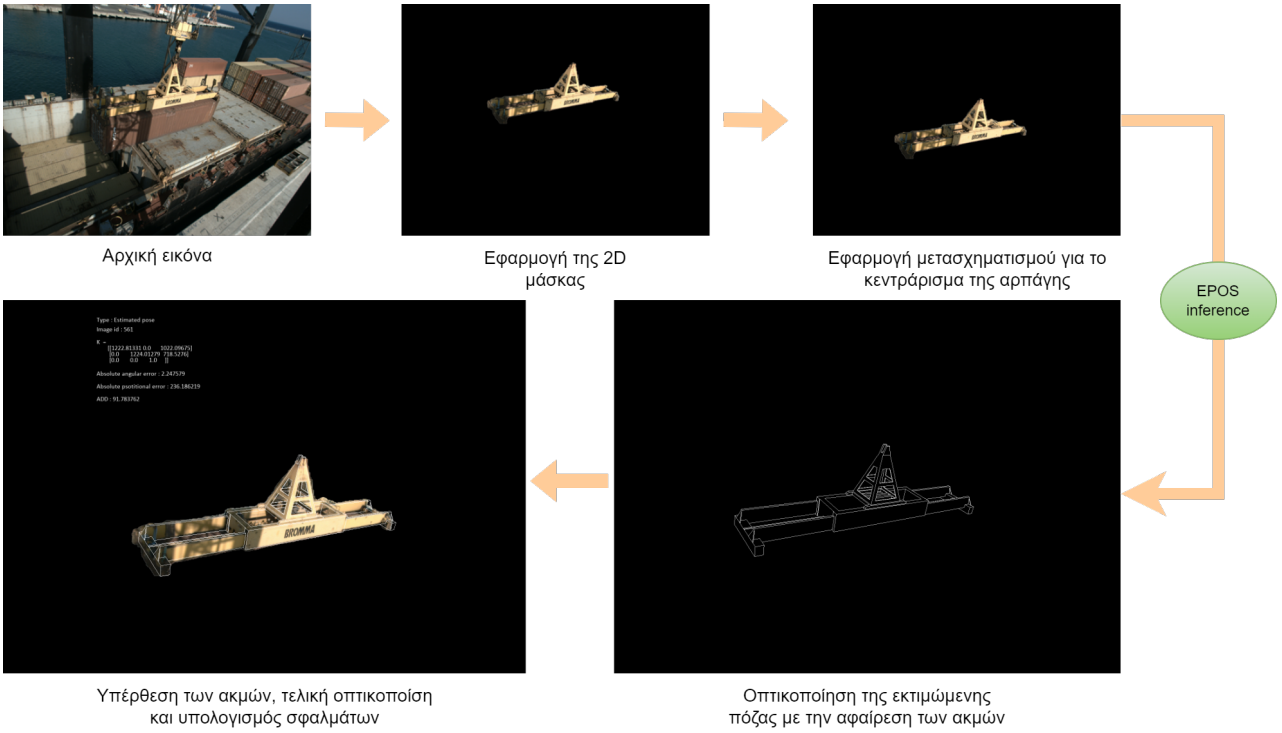
Με την παραπάνω διαδικασία σχηματίζονται οι τελικές εικόνες που θα κληθεί το μοντέλο να εκτιμήσει την πόζα. Αντίστοιχα, όμως πρέπει να μετασχηματιστούν και οι πίνακες R|T των ground truth. Με βάση όλες τα παραπάνω οι τελικές GT πόζες προκύπτουν με τον μετασχηματισμό:

$$[\boldsymbol{R}|\boldsymbol{t}]_c = \boldsymbol{R}_h \cdot [\boldsymbol{R}|\boldsymbol{t}]_{GT} \cdot \begin{bmatrix} \boldsymbol{R}_s & \boldsymbol{O} \\ \boldsymbol{0} & 1 \end{bmatrix} \cdot \begin{bmatrix} \boldsymbol{I} & \boldsymbol{t}_{tr} \\ \boldsymbol{0} & 1 \end{bmatrix} \tag{26}$$

όπου:

- $[\boldsymbol{R}|\boldsymbol{t}]_c$ είναι η τελική GT πόζα όπως απαιτείται από το EPOS

- $\boldsymbol{R}_h$ είναι ο πίνακας στροφής που κεντράρει την αρπάγη
- $\boldsymbol{R}_s$ είναι η αλλαγή των αξόνων y-z όπως αναφέρθηκε στην ενότητα 5.1.4
- $\boldsymbol{t}_{tr}$ η μετάθεση έτσι ώστε η αρχή του συστήματος του μοντέλου να είναι το κέντρο του BB (5.1.4)



Σχήμα 63: Διαγραμματική αναπαράσταση της διαδικασίας εξαγωγής της πόζας και των αποτελεσμάτων

5.3 Πειραματικά αποτελέσματα σε πραγματικές εικόνες

Όπως και στα αποτελέσματα του validation γενικά πόζες που η αρπάγη είναι σε μακρινές αποστάσεις και αντιπροσωπεύεται από λίγα pixels στην εικόνα παρουσιάζουν μεγαλύτερα σφάλματα. Επειδή οι ground truth πόζες είναι στην πραγματικότητα pseudo-ground truth και σε κάποιες περιπτώσεις η εξαγόμενη 2D μάσκα ήταν προβληματική (π.χ λόγω κοντινών αντικειμένων παρόμοιου χρώματος) επιλέχθηκε να αποκλειστούν από τον υπολογισμό το 10% αποτελεσμάτων που αντιστοιχούν στις πόζες με τα μεγαλύτερων σφάλματα γιατί αυτά κυρίως οφείλονταν σε απόκλιση των pseudo ground truth από την πραγματικότητα. Η 90η εκατοστιαία τιμή των σφαλμάτων υπολογίζεται με τον παρακάτω τύπο (όταν πρόκειται για ταξινομημένο (sorted) διάνυσμα):

$$q_{th} = \lfloor \frac{q}{100} \cdot len(v) \rfloor$$

όπου:

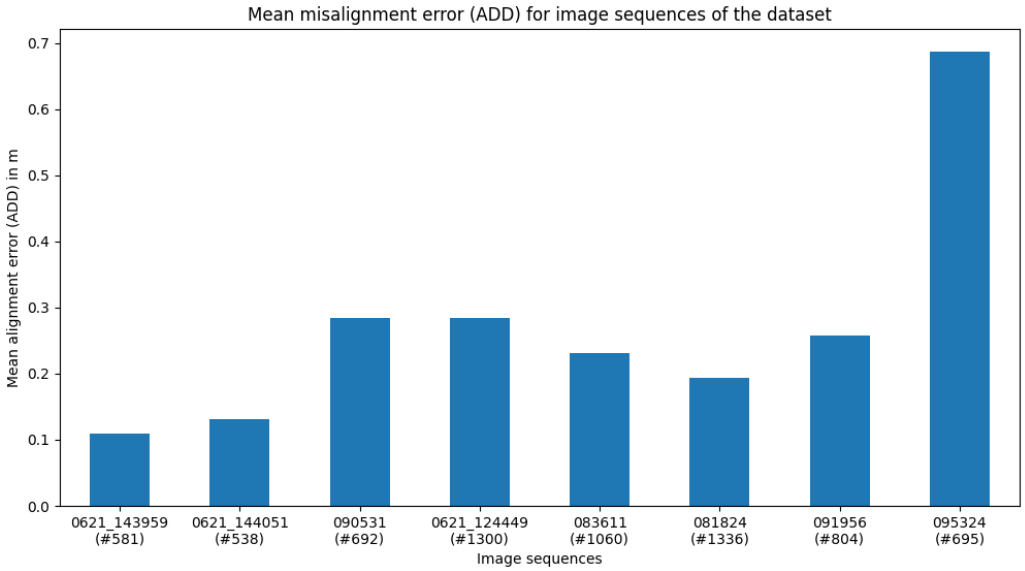
- το q_{th} αντιπροσωπεύει τη θέση της τιμής του ορίου q%. Για παράδειγμα για το διάνυσμα [1,6,8,15] το 70% των τιμών που βρίσκονται μέσα στο διάνυσμα είναι μικρότερα από το $v[\lfloor 0.7 \cdot 4 \rfloor] = v[2] = 8$
- με $\lfloor \quad \rfloor$ αντιπροσωπεύεται η συνάρτηση floor δηλαδή ο αμέσως μικρότερος ακέραιος του ορίσματος

Η παραπάνω διαδικασία επιτυγχάνεται μέσω της συνάρτησης numpy.percentile όπως φαίνεται και στο δείγμα κώδικα 15. Όπως φαίνεται και στον πίνακα 6 για την πλειονότητα των dataset το σφάλμα AAE βρίσκεται κάτω από 5° με την εξαίρεση της σειράς 095324 όπου το μέσο σφάλμα ανέρχεται περίπου στις 8°. Παρατηρήθηκε ότι τα σφάλματα κυμαίνονται μεταξύ $2^\circ - 3^\circ$ ενώ για σειρές εικόνων όπου η αρπάγη είναι πιο απομακρυσμένη αυξάνονται έως $4^\circ - 5^\circ$. Αντίστοιχα, το σφάλμα APE κυμαίνεται γενικά από 0.5 m - 0.6 m και για μακρινές αποστάσεις δεν ξεπερνάει τα 3.5 m.

Sequence	#frames	ADD error (m)			Absolute Angular Error (°)			Absolute Positional Error (m)		
		min	mean	max	min	mean	max	min	mean	max
0621_144051	538	0.012	0.131	0.359	0.11	1.81	5.79	0.028	0.309	1.917
0621_143959	581	0.015	0.109	0.230	0.06	1.50	4.57	0.008	0.273	1.256
090531	692	0.026	0.284	0.613	0.42	3.65	10.48	0.036	0.570	3.873
0621_124449	1300	0.012	0.284	0.949	0.25	4.66	19.15	0.017	0.934	4.761
083611	1060	0.029	0.230	0.642	0.17	4.35	11.77	0.034	0.535	3.525
081824	1336	0.016	0.194	0.563	0.11	1.87	6.65	0.018	0.641	3.028
091956	804	0.037	0.257	0.700	0.15	3.36	10.74	0.025	0.650	3.116
095324	695	0.08	0.686	1.286	0.47	7.93	16.36	0.029	0.905	2.389

Πίνακας 6: Ποσοτική αξιολόγηση για κάθε dataset. Min, Mean, Max για τα AAE, APE, ADD

Επειδή, ενδιαφέρει πιο πολύ η γενική απόκλιση που έχει η αρπάγη σε σχέση με τις ground truth πόζες παρά το μεμονωμένα απόλυτα γωνιακά και σφάλματα θέσης επιλέχθηκε η 90η εκατοστιαία τιμή του ADD να καθορίσει τις εικόνες που θα αποκλειστούν από τον υπολογισμό των στατιστικών μεγεθών. Για το σφάλμα ADD παρατηρείται ότι δεν ξεπερνάει τα 0.7 m και στην πλειονότητα των dataset αυτό βρίσκεται κάτω από 0.3 m. Ακόμα, για τις σειρές εικόνων με το μεγαλύτερο αριθμό εικόνων όπως τις 0621_124449 και 081824 το σφάλμα ADD δεν ξεπερνάει τα 0.2 m. Σε αυτό το σημείο είναι σημαντικό να αναφερθεί το πότε μια πόζα θεωρείται αποδεκτή με βάση το σφάλμα ADD. Στη βιβλιογραφία αναφέρεται ότι μία πόζα θεωρείται ”σωστή” όταν το σφάλμα ADD δεν ξεπερνάει το 10% της διαμέτρου του αντικειμένου (Hinterstoisser et al. [2012]). Στην περίπτωση της αρπάγης η διάμετρός είναι 12.44m οπότε πόζες που αποδίδουν σφάλματα μικρότερα του 1.244 θεωρούνται σωστές. Η τελευταία συνθήκη επαληθεύεται για όλες τις σειρές εικόνων και δεν επαληθεύεται για λίγες εικόνες στη σειρά 095324.



Σχήμα 64: Σφάλμα μέσης απόκλισης (Mean misalignment error - ADD) για όλες τις σειρές εικόνων. Στον οριζόντιο άξονα αναγράφονται τα ονόματα των dataset ενώ στον κατακόρυφο άξονα αναγράφεται το μέσο σφάλμα ADD των εικόνων του κάθε dataset.

Σχήμα 65: Ενδεικτικά αποτελέσματα και οπτικοποίηση από την αξιολόγηση σε πραγματικές εικόνες. Με άσπρο φαίνονται οι ορατές ακμές του μοντέλου. Πάνω αριστερά αναγράφεται το id της εικόνας αλλά και τα αντίστοιχα σφάλματα της εκτιμώμενης πόζας.

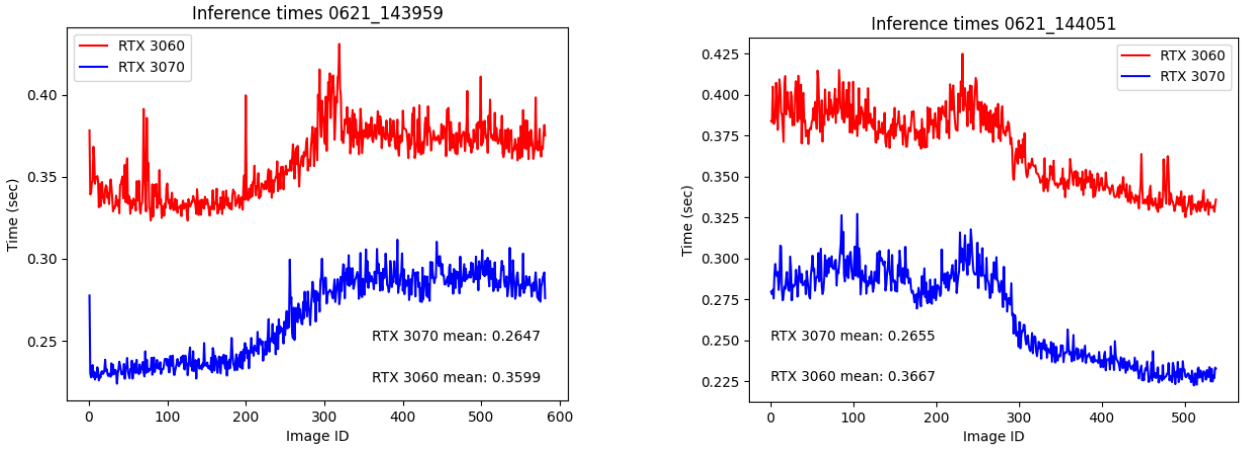
Εξίσου σημαντική με την ποιοτική και ποσοτική αξιολόγηση του μοντέλου είναι και ο χρόνος απόδοσης του αποτελέσματος. Η διαδικασία που αναπτύχθηκε ακολουθεί τη λειτουργικότητα του EPOS ως προς την εκτέλεση των προβλέψεων και των υπολογισμών με δημιουργία μιας συλλογής δεδομένων και όχι ανά μία εικόνα. Για αυτό το λόγο οι μέσοι όροι των χρόνων που καταγράφηκαν αφορούν το χρόνο πρόβλεψης του αλγορίθμου του EPOS για το σύνολο των εικόνων στη συλλογή. Ακόμα, είναι σημαντικό να τονιστεί ότι αυτοί οι χρόνοι αφορούν την κανονική έκδοση τους EPOS δηλαδή χωρίς κάποια βελτιστοποίηση για την απόδοση του. Για την εκπαίδευση και την αξιολόγηση της απόδοσης και την καταγραφή των χρόνων πρόβλεψης και επεξεργασίας χρησιμοποιήθηκαν δύο μηχανήματα με διαφορετικές κάρτες γραφικών του εργαστηρίου φωτογραμμετρίας της ΣΑΤΜ-ΜΓ τα χαρακτηριστικά των οποίων φαίνονται παρακάτω.

Και οι δύο GPU είναι της γενιάς RTX 3000 με την RTX 3060 να έχει περισσότερη VRAM (12 GB) σε σχέση με την RTX 3070 όμως η δεύτερη έχει αρκετά παραπάνω CUDA cores. Η CUDA (Compute Unified Architecture) είναι μια πλατφόρμα με την οποία υπολογισμοί μπορούν να παραλληλοποιηθούν με αποτέλεσμα τη γρηγορότερη επεξεργασία όγκου δεδομένων. Με άλλα λόγια, ο μεγαλύτερος

Μοντέλο	VRAM	CUDA Cores	Αρχιτεκτονική
NVIDIA GeForce RTX 3060	12 GB GDDR6	3584	Ampere
NVIDIA GeForce RTX 3070	8 GB GDDR6	5888	Ampere

Πίνακας 7: Τεχνικά χαρακτηριστικά των καρτών γραφικών που χρησιμοποιήθηκαν

αριθμός CUDA cores δηλώνει μια γρηγορότερη απόδοση της RTX 3070. Αν και υπάρχουν και άλλοι παράγοντες που επηρεάζουν την απόδοση μιας κάρτας γραφικών (Core Clock, Memory clock, memory bandwitdh) υπάρχει πιο άμεση σύγκριση διότι ανήκουν στην ίδια γενιά. Για την διαδικασία της εκπαίδευσης τα 8 GB μνήμες της RTX 3070 δεν ήταν αρκετά για να γίνει η αναγκαία δέσμευση μνήμης για τον όγκο και τις διαστάσεις των εικόνων των δεδομένων εκπαίδευσης με αποτέλεσμα να χρησιμοποιηθεί τη RTX 3060 η οποία αν και πιο αργή είχε περισσότερη διαθέσιμη μνήμη. Για την αξιολόγηση των προβλέψεων του μοντέλου (inference) χρησιμοποιήθηκαν και οι δύο κάρτες με πανομοιότυπα αποτελέσματα αλλά διαφορετικούς χρόνους όπως φαίνεται στα διαγράμματα του σχήματος 66. Η RTX 3070 είναι περίπου 25% πιο γρήγορη στις προβλέψεις της σε σχέση με το προηγούμενο μοντέλο της γενιάς. Η διερεύνηση των χρόνων που επιτυγχάνει το μοντέλο εξυπηρετούν και το σκοπό του κατά πόσο μια τέτοια διαδικασία μπορεί να υλοποιηθεί με μία προμηθεύσιμη -από την άποψη κόστους- κάρτα γραφικών.



Σχήμα 66: Χρόνοι πρόβλεψης και εξαγωγής της εκτιμώμενης πόζας για δύο σειρές εικόνων με τη χρήση δύο διαφορετικών GPU. Οι διαστάσεις των εικόνων είναι 720 x 540 και στις δύο περιπτώσεις.

Σχήμα 67: Οι εκτιμώμενες πόζες μετασχηματισμένες στο format των αρχικών εικόνων. Με άσπρο χρώμα έχει γίνει η υπέρθεση των ορατών ακμών (διακρίνονται καλύτερα σε μεγέθυνση) και με μπλε ημιδιάφανο χρώμα φαίνεται το 3D μοντέλο της αρπάγης

6 Κεφάλαιο 6: Συμπεράσματα και μελλοντική εργασία

Ο σκοπός της διπλωματικής εργασίας ήταν η ανάπτυξη μιας μεθοδολογίας δημιουργίας των κατάλληλων δεδομένων και την εκπαίδευση ενός αλγορίθμου μηχανικής μάθησης για το προσδιορισμό της 6D πόζας μιας αρπάγης container. Η δυσκολία έγκειται στα χαρακτηριστικά του εξωτερικού περιβάλλοντος (δυναμικές, μη ελεγχόμενες συνθήκες φωτισμού και φόντου), στις μεγάλες διαστάσεις του αντικειμένου οι οποίες δεν επιτρέπουν την εξαγωγή δειγμάτων εκπαίδευσης σε ελεγχόμενο περιβάλλον εργαστηρίου και των χαρακτηριστικών του αντικειμένου (μεταλλικό, χωρίς υφή, με συμμετρίες). Προκειμένου να αντιμετωπιστεί η έλλειψη των απαραίτητων εικόνων εκπαίδευσης από το πεδίο δημιουργήθηκε αρχικά το texure της αρπάγης και σε συνδυασμό με την υλοποίηση ενός renderer δημιουργήθηκαν συνθετικά δεδομένα εκπαίδευσης. Με τη χρήση των τελευταίων εκπαιδεύτηκε και αξιολογήθηκε ο αλγόριθμος EPOS. Τέλος, για την αξιολόγηση του μοντέλου σε πραγματικές εικόνες που αναπαριστούν την διαδικασία φόρτωσης/εκφόρτωσης αναπτύχθηκε μια μέθοδος προ-επεξεργασίας των εικόνων πριν την τελική πρόβλεψη της 6D πόζας από το μοντέλο. Τα αποτελέσματα αναλύθηκαν τόσο ποιοτικά όσο και ποσοτικά.

Για την δημιουργία του φωτορεαλιστικού μοντέλου ελέγχθηκαν δύο προσεγγίσεις σε δύο διαφορετικά λογισμικά χρησιμοποιώντας εικόνες από το πεδίο. Μετά από πειραματισμούς αποδείχθηκε ότι το Meshlab δεν είναι κατάλληλο την εφαρμογή υφής στο μοντέλο της αρπάγης το οποίο απαρτίζεται από σχετικά μεγάλα τρίγωνα (CAD) δεδομένου ότι υπήρχαν ανεπιθύμητες παραμορφώσεις. Για τον λόγο αυτό υιοθετήθηκε μία πιο χειροκίνητη διαδικασία στο Blender από την οποία τελικά προέκυψε το τελικό φωτορεαλιστικό μοντέλο της αρπάγης.

Η δημιουργία των δεδομένων εκπαίδευσης έγινε με την επιτυχή υλοποίηση ενός renderer χρησιμοποιώντας την OpenGl και τους shaders. Το πρόγραμμα αυτό επιτρέπει την συστηματική συλλογή εικόνων της αρπάγης από διαφορετικές γωνίες λήψης και αποστάσεις. Εκτός από τις εικόνες με το texture της αρπάγης αναπτύχθηκε και ένας υπο-αλγόριθμος ο οποίος κάνει render το μοντέλο της αρπάγης σε wireframe mode με την απόκρυψη των μη ορατών ακμών από την εκάστοτε οπτική γωνία της κάμερας. Το σύνολο των δεδομένων χωρίστηκε τυχαία σε training και validation set.

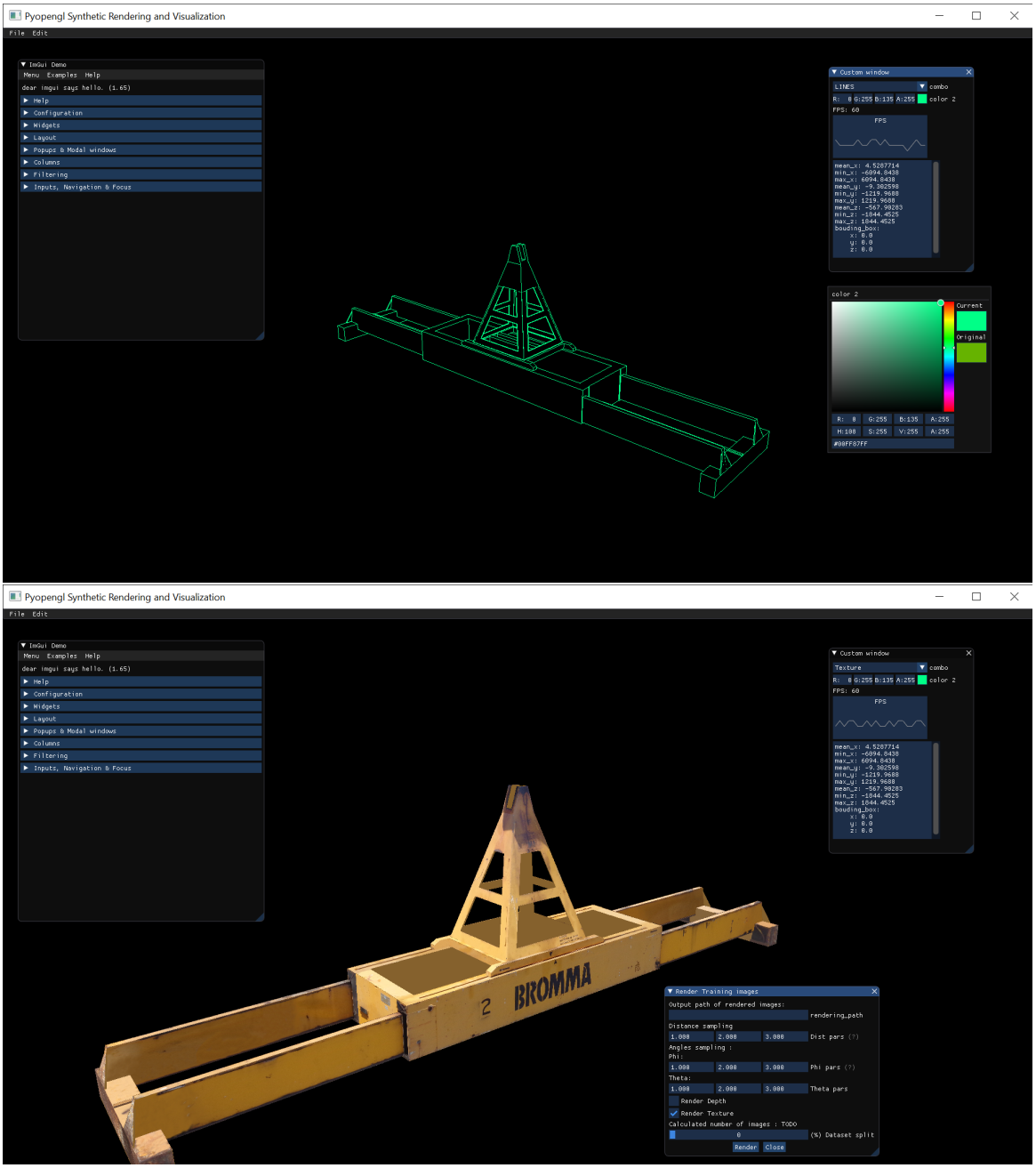
Με τα παραπάνω δεδομένα εκπαιδεύτηκε ένας state-of-the-art αλγόριθμος προσδιορισμού 6D πόζας αντικειμένων ο οποίος μπορεί να διαχειριστεί αποκρύψεις αντικειμένων αλλά και ολικές η μερικές συμμετρίες. Τα αποτελέσματα του αξιολογήθηκαν πάνω στο validation σετ με τον υπολογισμό κάποιων αντιπροσωπευτικών μετρικών.

Η τελική φάση της ερευνητικής εργασίας αφορούσε τον έλεγχο της απόδοσης του μοντέλου σε πραγματικές εικόνες. Στο σημείο αυτό, έπρεπε να λυθεί το πρόβλημα του δυναμικού φόντου το οποίο δεν υπήρχε στα συνθετικά δεδομένα εκπαίδευσης. Για αυτό το λόγο, έγινε προ-επεξεργασία των εικόνων για την αφαίρεση του φόντου με την απόκτηση της μάσκας της περιοχής που καλύπτει η αρπάγη στην εικόνα και κεντράροντας στη συνέχεια σε αυτήν. Τέλος, αξιολογήθηκαν οι ακρίβειες στις πραγματικές εικόνες οι οποίες ήταν μέσα στα όρια των απαιτήσεων ακρίβειας της εφαρμογής όπως περιγράφηκε στην ενότητα 5.3. Ακόμα, έγινε μια ανάλυση του χρόνου που χρειάζεται το μοντέλο για την πρόβλεψη με κάρτες γραφικών ευρέως χρησιμοποιούμενες. Οι χρόνοι που προέκυψαν δεν είναι ικανοί για μια υλοποίηση παρακολούθησης μέσω αναγνώρισης (tracking by detection) (30+ fps) αλλά είναι ικανοποιητικοί για το αρχικό ζητούμενο της εργασίας δηλαδή να μπορεί να γίνεται αρχικοποίηση ενός 3D tracker αλλά και η ανάκτηση της πόζας όταν ο 3D tracker αποτυγχάνει να αναγνωρίσει την αρπάγη (πχ. απόκρυψη της από άλλα αντικείμενα).

Παρόλα αυτά, υπάρχουν πράγματα στα οποία μπορούν να υπάρξουν βελτιώσεις. Ένας κύριος παράγοντας είναι ο χρόνος που απαιτείται για τον προσδιορισμό της 6D πόζας σαν σύνολο (προ-επεξεργασία + πρόβλεψη μοντέλου). Αρχικά, η προ-επεξεργασία δεν έγινε ενιαία σε ένα πρόγραμμα.

Για παράδειγμα κάποια μέρη της υλοποιήθηκαν με χρήση κώδικα C++ παράλληλα με άλλες διαδικασίες και άλλα με τη χρήση της MATLAB. Ενώ τα σφάλματα και τα διαγράμματα υπολογίστηκαν με κώδικα Python. Μία πρόταση που αξίζει να δοθεί έμφαση θα ήταν να υλοποιηθεί μία διαδικασία ενιαία και ομογενής. Ακόμα, εκτός από τον χρόνο της προ-επεξεργασίας μπορεί δραματικά να βελτιωθεί ο χρόνος πρόβλεψης του μοντέλου. Όπως αναφέρουν οι Hodaň et al. [2020] για το EPOS η συγκεκριμένη υλοποίηση δεν είναι βελτιστοποιημένη. Υπάρχουν τρόποι με τους οποίους το εκπαιδευμένο μοντέλο μπορεί να μετασχηματιστεί και να πραγματοποιηθούν τροποποιήσεις σε αυτό σε επίπεδο αρχιτεκτονικής έτσι ώστε να μειωθεί ο χρόνος πρόβλεψης. Τέτοιοι τρόποι είναι η μετατροπή του σε ONNX format (ONNX [2023]) και το inference να γίνεται με βάση βιβλιοθήκες όπως το ONNX Runtime αλλά και η NVIDIA TensorRT (NVIDIA [2023]). Με μια εκτίμηση το inference θα μπορούσε να λειτουργεί κοντά στα 10-20 FPS δηλαδή να μειωθεί ο χρόνος πρόβλεψης περίπου 2-5x. Άλλες πιθανές βελτιώσεις είναι η προσθήκη συνθετικού δυναμικού background στα δεδομένα εκπαίδευσης. Για παράδειγμα μπορούν να υπάρχουν σαν background διάφορες σκηνές από την αποβάθρα, τα πλοία και τα container και να γίνει υπέρθεση της σε διάφορες θέσεις της εικόνας. Ως προς αυτό θα μπορούσαν να μοντελοποιηθούν και containers έτσι ώστε το μοντέλο να εκπαιδευτεί και σε δεδομένα όπου κάποιο μέρος τους αποκρύπτεται από αυτά (κάτι που συναντάται κατά την διαδικασία φόρτωσης/εκφόρτωσης).

Τέλος, ενδιαφέρουσα πρόταση είναι η δημιουργία ενός προγράμματος με εξυπηρετικό UI (User interface - Διεπαφή χρήστη) για την δημιουργία των συνθετικών δεδομένων. Ο χρήστης θα μπορεί να εισάγει το επιθυμητό μοντέλο, να περιηγηθεί και να το περιστρέψει αλλά και να επιλέξει τις κατάλληλες παραμέτρους για τη δημιουργία του σετ δεδομένων. Μία τέτοια υλοποίηση έχει μερικώς δοκιμαστεί με τη χρήση της pyOpenGL και της βιβλιοθήκης pyimgui (pyimgui [2022])(Σχήμα 68).



Σχήμα 68: Δοκιμές δημιουργίας διεπαφής χρήστη για περιήγηση 3D μοντέλου και δημιουργία συνθετικών εικόνων

7 Αναφορές

Abdel-Aziz, Y. and Karara, H. (2015). Direct linear transformation from comparator coordinates into object space coordinates in close-range photogrammetry. *Photogrammetric Engineering & Remote Sensing*, 81(2):103–107.

Aggarwal, C. C. (2018). *Neural Networks and Deep Learning: A Textbook*. Springer Nature Switzerland.

Arun, K. S., Huang, T. S., and Blostein, S. D. (1987). Least-squares fitting of two 3-d point sets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-9(5):698–700.

Ayyadevara, V. K. and Yeshwanth, R. (2020). *Modern Computer Vision with PyTorch*. Packt Publishing Ltd.

Barath, D. and Matas, J. (2018). Graph-cut ransac. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6733–6741.

Bay, H., Tuytelaars, T., and Van Gool, L. (2006). Surf: Speeded up robust features. In Leonardis, A., Bischof, H., and Pinz, A., editors, *Computer Vision – ECCV 2006*, pages 404–417, Berlin, Heidelberg. Springer Berlin Heidelberg.

Berkeley, U. (2019). Advanced topics in signal processing: 3d image processing and computer vision, lecture 8: Homography. `https://inst.eecs.berkeley.edu/~ee290t/fa19/lectures/lecture8-homography.pdf`.

Blender (2023). blender.org - home of the blender project - free and open 3d creation software. `https://www.blender.org/`.

Brachmann, E., Krull, A., Michel, F., Gumhold, S., Shotton, J., and Rother, C. (2014a). Learning 6d object pose estimation using 3d object coordinates. In Fleet, D., Pajdla, T., Schiele, B., and Tuytelaars, T., editors, *Computer Vision – ECCV 2014*, pages 536–551, Cham. Springer International Publishing.

Brachmann, E., Krull, A., Michel, F., Gumhold, S., Shotton, J., and Rother, C. (2014b). Learning 6d object pose estimation using 3d object coordinates. In Fleet, D., Pajdla, T., Schiele, B., and Tuytelaars, T., editors, *Computer Vision – ECCV 2014*, pages 536–551, Cham. Springer International Publishing.

Calli, B., Walsman, A., Singh, A., Srinivasa, S., Abbeel, P., and Dollar, A. M. (2015). Benchmarking in manipulation research: The ycb object and model set and benchmarking protocols. *arXiv preprint arXiv:1502.03143*.

Chen, L.-C., Zhu, Y., Papandreou, G., Schroff, F., and Adam, H. (2018). Encoder-decoder with atrous separable convolution for semantic image segmentation. In Ferrari, V., Hebert, M., Sminchisescu, C., and Weiss, Y., editors, *Computer Vision – ECCV 2018*, pages 833–851, Cham. Springer International Publishing.

Crivellaro, A., Rad, M., Verdie, Y., Yi, K. M., Fua, P., and Lepetit, V. (2015). A novel representation of parts for accurate 3d object detection and tracking in monocular images. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 4391–4399.

Cunha, B., Droz, C., Zine, A., Foulard, S., and Ichchou, M. (2022). A review of machine learning methods applied to structural dynamics and vibroacoustic. *arXiv preprint arXiv:2204.06362*.

Dam, E. B., Koch, M., and Lillholm, M. (2000). Quaternions, interpolation and animation.

de Vries, J. (2020). *Learn OpenGL - Graphics Programming*. Kendall & Welling.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255.

Do, T.-T., Cai, M., Pham, T., and Reid, I. (2018). Deep-6dpose: Recovering 6d object pose from a single rgb image. *arXiv preprint arXiv:1802.10367*.

Doumanoglou, A., Kouskouridas, R., Malassiotis, S., and Kim, T.-K. (2015). Recovering 6d object pose and predicting next-best-view in the crowd. *arXiv preprint arXiv:1512.07506*.

Du, G., Wang, K., Lian, S., and Zhao, K. (2019). Vision-based robotic grasping from object localization, object pose estimation to grasp estimation for parallel grippers: A review. *arXiv preprint arXiv:1905.06658*.

Dutta, A. and Zisserman, A. (2019). The VGG image annotator (VIA). *CoRR*, abs/1904.10699.

Everingham, M., Gool, L. V., Williams, C. K. I., Winn, J. M., and Zisserman, A. (2010). The pascal visual object classes (voc) challenge. *Int. J. Comput. Vis.*, 88(2):303–338.

Fischler, M. and Bolles, R. (1981). Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395.

Förstner, W. and Wrobel, B. P. (2016). *Photogrammetric Computer Vision*. Springer Cham, Switzerland.

Gao, X.-S., Hou, X.-R., Tang, J., and Cheng, H.-F. (2003). Complete solution classification for the perspective-three-point problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(8):930–943.

Garrido-Jurado, S., Muñoz Salinas, R., Madrid-Cuevas, F., and Marín-Jiménez, M. (2014). Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognition*, 47(6):2280–2292.

Grover, P. (2018). 5 regression loss functions all machine learners should know. `https://heartbeat.comet.ml/5-regression-loss-functions-all-machine-learners-should-know-4fb140e9d4b0`. Τελευταία πρόσβαση 28 Φεβρουαρίου, 2023.

Grunert, J. (1841). Das pothenotische problem in erweiterter gestalt nebst ber seine anwendungen in der geodsie. *Grunerts Archiv für Mathematik und Physik*.

Hamilton, W. R. (1844). Ii. on a new species of imaginary quantities connected with a theory of quaternions. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 25(168):1–14.

He, K., Gkioxari, G., Dollár, P., and Girshick, R. (2017). Mask r-cnn. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2980–2988.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *arXiv preprint arXiv:1502.01852*.

He, Z., Feng, W., Zhao, X., and Lv, Y. (2021). 6D pose estimation of objects: Recent technologies and challenges. *Applied Sciences*, 11(1):228.

Hinterstoisser, S., Lepetit, V., Ilic, S., Holzer, S., Bradski, G., Konolige, K., and Navab, N. (2012). Model based training, detection and pose estimation of texture-less 3D objects in heavily cluttered scenes. In *Asian Conf. on Computer Vision*, pages 548–562. Springer.

Hodan, T., Haluza, P., Obdržálek, Š., Matas, J., Lourakis, M., and Zabulis, X. (2017). T-LESS: An RGB-D dataset for 6D pose estimation of texture-less objects. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 880–888. IEEE.

Hodan, T., Michel, F., Brachmann, E., Kehl, W., Buch, A. G., Kraft, D., Drost, B., Vidal, J., Ihrke, S., Zabulis, X., Sahin, C., Manhardt, F., Tombari, F., Kim, T.-K., Matas, J., and Rother, C. (2018). Bop: Benchmark for 6d object pose estimation. *arXiv preprint arXiv:1808.08319*.

Hodaň, T., Sundermeyer, M., Drost, B., Labbé, Y., Brachmann, E., Michel, F., Rother, C., and Matas, J. (2019). BOP challenge 2019 on 6D object localization.

Hodan, T., Sundermeyer, M., Drost, B., Labbé, Y., Brachmann, E., Michel, F., Rother, C., and Matas, J. (2020). Bop challenge 2020 on 6d object localization. In *ECCV Workshops*.

Hodaň, T., Baráth, D., and Matas, J. (2020). Epos: Estimating 6d pose of objects with symmetries. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11700–11709.

Hodaň, T., Zabulis, X., Lourakis, M., Obdržálek, □., and Matas, J. (2015). Detection and fine 3d pose estimation of texture-less objects in rgb-d images. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4421–4428.

Howard, J. and Gugger, S. (2020). *Deep Learning for Coders with Fastai and Pytorch: AI Applications Without a PhD*. O'Reilly Media, Incorporated.

Hu, H., Zhu, M., Li, M., and Chan, K.-L. (2022). Deep learning-based monocular 3D object detection with refinement of depth information. *Sensors*, 22(7).

Hu, Y., Fua, P., Wang, W., and Salzmann, M. (2020). Single-stage 6d object pose estimation. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2927–2936.

Hu, Y., Hugonot, J., Fua, P., and Salzmann, M. (2019). Segmentation-driven 6d object pose estimation. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3380–3389.

Huber, P. J. (1964). Robust Estimation of a Location Parameter. *The Annals of Mathematical Statistics*, 35(1):73 – 101.

Huynh, D. Q. (2009). Metrics for 3D rotations: Comparison and analysis. *J. Math. Imaging Vis.*, 35(2):155–164.

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*.

Janoch, A., Karayev, S., Jia, Y., Barron, J. T., Fritz, M., Saenko, K., and Darrell, T. (2011). A category-level 3-d object dataset: Putting the kinect to work. In *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pages 1168–1174.

Jia, Y.-B. (2013). *Quaternions and Rotations **.

Joseph Redmon, A. F. (2018). Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*.

Kehl, W., Manhardt, F., Tombari, F., Ilic, S., and Navab, N. (2017). Ssd-6d: Making rgb-based 3d detection and 6d pose estimation great again. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 1530–1538.

Kendall, A., Grimes, M., and Cipolla, R. (2015). Posenet: A convolutional network for real-time 6-dof camera relocalization. *arXiv preprint arXiv:1505.07427*.

Khronos, G. (2023). Opengl - the industry standard for high performance graphics. https://www.opengl.org/. Τελευταία πρόσβαση 1 Μαρτίου, 2023.

Kim, S.-h. and Hwang, Y. (2021). A survey on deep learning based methods and datasets for monocular 3D object detection. *Electronics*, 10(4):517.

Kneip, L., Scaramuzza, D., and Siegwart, R. (2011). A novel parametrization of the perspective-three-point problem for a direct computation of absolute camera position and orientation. In *CVPR 2011*, pages 2969–2976.

Labbe, Y., Carpentier, J., Aubry, M., and Sivic, J. (2020). CosyPose: Consistent multi-view multi-object 6D pose estimation. In *Proceedings of the European Conference on Computer Vision (ECCV)*.

Lepetit, V. and Fua, P. (2005). Monocular model-based 3D tracking of rigid objects: A survey. *Foundations and Trends in Computer Graphics and Vision*, 1(1).

Lepetit, V., Moreno-Noguer, F., and Fua, P. (2009). Epnp: An accurate o(n) solution to the pnp problem. *International Journal Of Computer Vision*, 81:155–166.

Li, Z., Wang, G., and Ji, X. (2019). Cdpn: Coordinates-based disentangled pose network for real-time rgb-based 6-dof object pose estimation. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 7677–7686.

Liang, K. K. (2018). Efficient conversion from rotating matrix to rotation axis and angle by extending rodrigues' formula. *arXiv preprint arXiv:1810.02999*.

Lin, T.-Y., Goyal, P., Girshick, R., He, K., and Dollár, P. (2017). Focal loss for dense object detection. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2999–3007.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., and Berg, A. C. (2016). SSD: Single shot multibox detector. In *ECCV*.

Liu, X., Jonschkowski, R., Angelova, A., and Konolige, K. (2020). Keypose: Multi-view 3d labeling and keypoint estimation for transparent objects. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11599–11607.

Lourakis, M. (2004). levmar: Levenberg-Marquardt nonlinear least squares algorithms in C/C++. [web page] http://www.ics.forth.gr/~lourakis/levmar/.

Lourakis, M. and Pateraki, M. (2021). Markerless visual tracking of a container crane spreader. In *IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, pages 2579–2586.

Lourakis, M. and Pateraki, M. (2022a). Computer vision for increasing safety in container handling operations. In *Human Factors and Systems Interaction, International Conference on Applied Human Factors and Ergonomics (AHFE)*, volume 52.

Lourakis, M. and Pateraki, M. (2022b). Container spreader pose tracking dataset. Zenodo, https://doi.org/10.5281/zenodo.7043890.

Lourakis, M., Pateraki, M., Karolos, I.-A., Pikridas, C., and Patias, P. (2020). Pose estimation of a moving camera with low-cost, multi-GNSS devices. *Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci.*, XLIII-B2-2020:55–62.

Lowe, D. (1999). Object recognition from local scale-invariant features. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 2, pages 1150–1157 vol.2.

Marchand, E., Uchiyama, H., and Spindler, F. (2016). Pose estimation for augmented reality: A hands-on survey. *IEEE Transactions on Visualization and Computer Graphics*, 22(12):2633–2651.

MATLAB (2022). https://www.mathworks.com/help/uav/ref/quaternion.html#d124e7178.

Meshlab (2022). https://www.meshlab.net/#download.

Ngo, Q. H. and Hong, K.-S. (2012). Sliding-Mode Antisway Control of an Offshore Container Crane. *IEEE/ASME Transactions on Mechatronics*, 17(2):201–209.

NVIDIA (2023). Nvidia tensorrt. https://developer.nvidia.com/tensorrt.

ONNX (2023). Onnx - open neural network exchange. https://onnx.ai/.

opencv (2022). Opencv pnp. https://docs.opencv.org/4.x/d5/d1f/calib3d_solvePnP.html.

Opengl and tutorials (2023). An introduction on opengl with 2d graphics. https://www3.ntu.edu.sg/home/ehchua/programming/opengl/cg_introduction.html#zz-3.

Padding (2020). Convolutional neural networks - padding and stride. https://d2l.ai/chapter_convolutional-neural-networks/padding-and-strides.html.

Peng, S., Liu, Y., Huang, Q., Zhou, X., and Bao, H. (2019). Pvnet: Pixel-wise voting network for 6dof pose estimation. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4556–4565.

Poli, R., Kennedy, J., and Tim, B. (2007). Particle swarm optimization. *Swarm Intelligence*, 1(1).

pyimgui (2022). pyimgui: Cython-based python bindings for dear imgui. https://github.com/pyimgui/pyimgui.

Quan, L. and Lan, Z. (1999). Linear n-point camera pose determination. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(8):774–780.

Rad, M. and Lepetit, V. (2017). Bb8: A scalable, accurate, robust to partial occlusion method for predicting the 3d poses of challenging objects without using depth. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 3848–3856.

Rahman, M. M., Tan, Y., Xue, J., and Lu, K. (2019). Recent advances in 3D object detection in the era of deep neural networks: A survey. *IEEE Transactions on Image Processing*, 29:2947–2962.

Rennie, C., Shome, R., Bekris, K. E., and De Souza, A. F. (2016). A dataset for improved rgbd-based object detection and pose estimation for warehouse pick-and-place. *IEEE Robotics and Automation Letters*, 1(2):1179–1185.

Rosten, E. and Drummond, T. (2005). Fusing points and lines for high performance tracking. In *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, volume 2, pages 1508–1515 Vol. 2.

Roth, G. (2010). Homography. http://people.scs.carleton.ca/~c_shu/Courses/comp4900d/notes/homography.pdf.

Rublee, E., Rabaud, V., Konolige, K., and Bradski, G. (2011). Orb: An efficient alternative to sift or surf. In *2011 International Conference on Computer Vision*, pages 2564–2571.

Sahin, C. and Kim, T.-K. (2018). Recovering 6D object pose: a review and multi-modal analysis. In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, pages 0–0.

Shoemake, K. (1985). Quaternion calculus and fast animation. *ACM SIGGRAPH Computer Graphics*, 19(3):273–278.

Silberman, N., Hoiem, D., Kohli, P., and Fergus, R. (2012). Indoor segmentation and support inference from rgbd images. In Fitzgibbon, A., Lazebnik, S., Perona, P., Sato, Y., and Schmid, C., editors, *Computer Vision – ECCV 2012*, pages 746–760, Berlin, Heidelberg. Springer Berlin Heidelberg.

Song, S., Lichtenberg, S. P., and Xiao, J. (2015). Sun rgb-d: A rgb-d scene understanding benchmark suite. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 567–576.

Srivastava, N., Hinton, G., Krizhevsky, A., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958.

Sun, J., Wang, Z., Zhang, S., He, X., Zhao, H., Zhang, G., and Zhou, X. (2022). OnePose: One-shot object pose estimation without CAD models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6815–6824.

Tejani, A., Kouskouridas, R., Doumanoglou, A., Tang, D., and Kim, T.-K. (2018). Latent-class hough forests for 6 dof object pose estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(1):119–132.

Tejani, A., Tang, D., Kouskouridas, R., and Kim, T.-K. (2014). Latent-class hough forests for 3d object detection and pose estimation. In Fleet, D., Pajdla, T., Schiele, B., and Tuytelaars, T., editors, *Computer Vision – ECCV 2014*, pages 462–477, Cham. Springer International Publishing.

Tremblay, J., To, T., and Birchfield, S. (2018). Falling things: A synthetic dataset for 3d object detection and pose estimation. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 2119–21193.

Vacchetti, L., Lepetit, V., and Fua, P. (2004). Stable real-time 3d tracking using online and offline information. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(10):1385–1391.

Wang, B., Zhong, F., and Qin, X. (2019). Robust edge-based 3D object tracking with direction-based pose validation. *Multimedia Tools and Applications*, 78(9):12307–12331.

Wang, G., Manhardt, F., Shao, J., Ji, X., Navab, N., and Tombari, F. (2020). Self6D: Self-supervised monocular 6D object pose estimation. In *European Conference on Computer Vision (ECCV)*, pages 108–125, Cham. Springer International Publishing.

Weisstein, E. W. (2023). `https://mathworld.wolfram.com/SphericalCoordinates.html`.

Wicht, B. (2018). *Deep Learning feature Extraction for Image Processing*. PhD thesis.

Wikipedia (2023). Spherical coordinate system. `https://en.wikipedia.org/wiki/Spherical_coordinate_system`.

Xavier, G. and Yoshua, B. (2010). Understanding the difficulty of training deep feedforward neural networks. In Teh, Y. W. and Titterington, M., editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256, Chia Laguna Resort, Sardinia, Italy. PMLR.

Xiang, Y., Kim, W., Chen, W., Ji, J., Choy, C., Su, H., Mottaghi, R., Guibas, L., and Savarese, S. (2016). Objectnet3d: A large scale database for 3d object recognition. In *European Conference Computer Vision (ECCV)*.

Xiang, Y., Mottaghi, R., and Savarese, S. (2014). Beyond pascal: A benchmark for 3d object detection in the wild. In *IEEE Winter Conference on Applications of Computer Vision*, pages 75–82.

Xiang, Y., Schmidt, T., Narayanan, V., and Fox, D. (2018). Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes.

Xiao, J., Owens, A., and Torralba, A. (2013). Sun3d: A database of big spaces reconstructed using sfm and object labels. In *2013 IEEE International Conference on Computer Vision*, pages 1625–1632.

Zheng, Y., Kuang, Y., Sugimoto, S., Åström, K., and Okutomi, M. (2013). Revisiting the pnp problem: A fast, general and optimal solution. In *2013 IEEE International Conference on Computer Vision*, pages 2344–2351.

8 Παράρτημα

```
#version 460 core
#extension GL_EXT_gpu_shader4 : enable
layout (location = 0) in vec3 aPos;
uniform mat4 projection;
uniform mat4 translation;
uniform mat4 LookAt;
uniform mat4 view;

void main(){
    gl_Position=projection*LookAt*view*translation*vec4(aPos.x,aPos.y,aPos.z,1.0);
}

```

Κώδικας 1: Vertex shader for 3D points

```
#version 460 core
#extension GL_EXT_gpu_shader4 : enable
layout (triangles) in;
layout (triangle_strip, max_vertices = 3) out;


void main() {
    gl_Position = gl_in[0].gl_Position;
    EmitVertex();

    gl_Position=gl_in[1].gl_Position;
    EmitVertex();

    gl_Position=gl_in[2].gl_Position;
    EmitVertex();
    EndPrimitive();
}
```

Κώδικας 2: Pass-through geometry shader for a triangle primitive

```
#version 460 core
#extension GL_EXT_gpu_shader4 : enable
layout (lines) in;
layout (line_strip, max_vertices = 2) out;

void main() {
    gl_Position = gl_in[0].gl_Position;
    EmitVertex();

    gl_Position=gl_in[1].gl_Position;
    EmitVertex();

    EndPrimitive();
}
```

Κώδικας 3: Pass-through geometry shader for a line primitive

```
#version 460 core
#extension GL_EXT_gpu_shader4 : enable
out vec4 FragColor;
void main(){
    FragColor = vec4(0.0f,1.0f,0.0f,1.0f);
}
```

Κώδικας 4: Fragment shader for wireframe rendering - Σταθερό χρώμα (άσπρο)

```
#version 460 core
#extension GL_EXT_gpu_shader4 : enable

in vec2 TexCoords;
out vec4 FragColor;

uniform sampler2D mtexture;

void main(){
    FragColor = texture(mtexture,TexCoords);

}
```

Κώδικας 5: Fragment shader for texture rendering

```
unsigned int Renderer::Create_VAO(float vertices_array[], int size, int stride, int
    attribs[]) {

    unsigned int vao, vbo;
    glGenVertexArrays(1, &vao);
    glGenBuffers(1, &vbo);

    //bind vertex array
    glBindVertexArray(vao);

    glBindBuffer(GL_ARRAY_BUFFER, vbo);
    glBufferData(GL_ARRAY_BUFFER, sizeof(float) * size, vertices_array, GL_STATIC_DRAW);

    for (int i = 0; i < sizeof(attribs)/sizeof(attribs[0]); i++) {
        glVertexAttribPointer(i, attribs[i], GL_FLOAT, GL_FALSE, stride, (void*)(i *
    sizeof(float)));
        glEnableVertexAttribArray(i);
    }

    //unbinding
    glBindBuffer(GL_ARRAY_BUFFER, 0);
    glBindVertexArray(0);

    return vao;
}
```

Κώδικας 6: Δημιουργία VAO

```
FrameBuffer framebuffer = renderer.Create_framebuffer(FINAL_FBO_WIDTH, FINAL_FBO_HEIGHT,
    GL_RGB32F,GL_RGB,GL_FLOAT);

FrameBuffer Renderer::Create_framebuffer(int FBO_WIDTH, int FBO_HEIGHT,GLenum
    interanlformat,GLenum format,GLenum type) {

  //creating a framebuffer object
  FrameBuffer framebuffer{
    NULL,
    FBO_WIDTH,
    FBO_HEIGHT,
    NULL
  };
  glGenFramebuffers(1, &framebuffer.ID);
  glBindFramebuffer(GL_FRAMEBUFFER, framebuffer.ID);

  //create a color attachment - texture
  unsigned int textureColorbuffer;
  glGenTextures(1, &textureColorbuffer);
  glBindTexture(GL_TEXTURE_2D, textureColorbuffer);
  glTexImage2D(GL_TEXTURE_2D, 0, interanlformat, FBO_WIDTH, FBO_HEIGHT, 0, format, type,
    NULL);
  glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
  glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
  glFramebufferTexture2D(GL_FRAMEBUFFER, GL_COLOR_ATTACHMENT0, GL_TEXTURE_2D,
    textureColorbuffer, 0);
  // create a renderbuffer object for depth attachment
  unsigned int rbo;
  glGenRenderbuffers(1, &rbo);
  glBindRenderbuffer(GL_RENDERBUFFER, rbo);
  glRenderbufferStorage(GL_RENDERBUFFER, GL_DEPTH32F_STENCIL8, FBO_WIDTH, FBO_HEIGHT); //
     use a single renderbuffer object for both a depth AND stencil buffer.
  glFramebufferRenderbuffer(GL_FRAMEBUFFER, GL_DEPTH_STENCIL_ATTACHMENT, GL_RENDERBUFFER,
     rbo); // now actually attach it
  //check if the framebuffer is complete
  if (glCheckFramebufferStatus(GL_FRAMEBUFFER) != GL_FRAMEBUFFER_COMPLETE)
    cout << "ERROR::FRAMEBUFFER:: Framebuffer is not complete!" << endl;
  //binding again to default framebuffer
  glBindFramebuffer(GL_FRAMEBUFFER, 0);

  framebuffer.colorBuffer = textureColorbuffer;
  return framebuffer;
```

Κώδικας 7: Δημιουργία Framebuffer

```

unsigned char* pix = new unsigned char[FBO_WIDTH * FBO_HEIGHT * 3];
unsigned char* inverted = new unsigned char[FBO_WIDTH * FBO_HEIGHT * 3];

glReadPixels(0, 0, FBO_WIDTH, FBO_HEIGHT, GL_RGB,/*type of data in bytes*/
    GL_UNSIGNED_BYTE, pix/* returned data*/);

for (int i = 0; i < FBO_HEIGHT; i++) {
    //first line is FBO_WIDTH * 3
    memcpy(&inverted[(FBO_HEIGHT - 1 - i) * FBO_WIDTH * 3], &pix[i * FBO_WIDTH * 3],
    sizeof(unsigned char) * FBO_WIDTH * 3);

```

```
...
delete[] pix;
delete[] inverted;
}
```

Κώδικας 8: Ανάγνωση buffer εικόνας

```

    void Model::LinesOut(list<Model::triangle> triangles){
    list<triangle>::iterator tr1;
    list<triangle>::iterator tr2;

    for (tr1 = triangles.begin(); tr1 != triangles.end();tr1++){
        //for all lines in this triagnle

            for(tr2 = triangles.begin(); tr2 != triangles.end();tr2++){ // for every
    ohter triangle

                vector3D crossP = CrossVectorNormals(tr1->face_mormal, tr2->face_mormal);
                vector3D zero{ 0,0,0 };

                if (tr1!=tr2 && RoundVector3D(crossP).operator==(zero)) {
                    for (int i = 0; i < 3; i++) {
                        for (int j = 0; j < 3; j++) {
                            if (tr1->lines[i].operator==(tr2->lines[j])) {

                                lines.remove(tr1->lines[i]);
                                lines.remove(tr2->lines[j]);

                            }
                        }
                    }
                }
                else
                {
                    continue;
                }
            }
    }
}
```

Κώδικας 9: Απόκρυψη πλασματικών ακμών

```

    for (int r : dist_array)
        {
            for (int theta = 0; theta <= 90; theta += theta_interval)
            {
                for (int phi = 0; phi < 360; phi += phi_interval)
                {
                    glViewport(0, 0, FBO_WIDTH, FBO_HEIGHT);
                    ...
                    camera.spcifySphCoords(theta, phi);
                    camera.updateRadius(r_m);
                    LookAt = camera.UpdateView();

                    \\convert to Zup due to EPOS model axis requirements
```

```
                    LookAt = camera.transform2Zup();

                    ...
                    glBindFramebuffer(GL_FRAMEBUFFER, SSAA_BUFFER.ID);
                    glBindTexture(GL_TEXTURE_2D, texture);

                    shader_texture.use();

                    ...
                    glBindVertexArray(vao_texture);
                    glDrawArrays(GL_TRIANGLES, 0, vrt.size() / 4);
                    ...
                    //read the buffer with ReadPixels and copy it's data

                    unsigned char* pix = new unsigned char[FBO_WIDTH * FBO_HEIGHT * 3];
                    unsigned char* inverted = new unsigned char[FBO_WIDTH * FBO_HEIGHT *
3];

                    glReadPixels(0, 0, FBO_WIDTH, FBO_HEIGHT, GL_RGB,/*type of data in
bytes*/ GL_UNSIGNED_BYTE, pix/* returned data*/);

                    for (int i = 0; i < FBO_HEIGHT; i++) {
                        //first line is FBO_WIDTH * 3
                        memcpy(&inverted[(FBO_HEIGHT - 1 - i) * FBO_WIDTH * 3], &pix[i *
FBO_WIDTH * 3], sizeof(unsigned char) * FBO_WIDTH * 3);
                    }
                    glm::mat4 out_lookat = glm::transpose(LookAt) * camera.ICV_MATRIX;

                    //save the image using OpenCV wiht LookAT matrix!!!
                    try {


                            cv::Mat image(FBO_HEIGHT, FBO_WIDTH, CV_8UC3, &inverted[0]);
                            unsigned char* hl = new unsigned char[FINAL_FBO_WIDTH *
FINAL_FBO_HEIGHT * 3];
                            cv::cvtColor(image, image, cv::COLOR_BGR2RGB);
                            //cv::pyrDown(image,resized,cv::Size(FINAL_FBO_WIDTH,
FINAL_FBO_HEIGHT));


                            cv::imwrite(path + "/rgb/" + makeImgPrefix(image_id) +
to_string(image_id) + ".png", image);

                            //write the R|T to file using out_lookat!!
                            imgid2 = "\"" + to_string(image_id) + "\": [{\"cam_R_m2c\": [
" + to_string(out_lookat[0][0]) + ", " + to_string(out_lookat[0][1]) + ", " +
to_string(out_lookat[0][2]) + ", " + to_string(out_lookat[1][0]) + ", " + to_string(
out_lookat[1][1]) + ", " + to_string(out_lookat[1][2]) + ", " + to_string(out_lookat
[2][0]) + ", " + to_string(out_lookat[2][1]) + ", " + to_string(out_lookat[2][2]) + "
], \"cam_t_m2c\": [" + to_string(out_lookat[0][3]) + ", " + to_string(out_lookat
[1][3]) + ", " + to_string(out_lookat[2][3]) + "], \"obj_id\": 1}]";
                    catch (cv::Exception& e) {
                            cerr << e.msg << "OPENCV error" << endl;
                        }

                    //free memory
```

```
                    delete[] pix;
                    delete[] inverted;
                    delete[] pixD;
```

Κώδικας 10: Ανανέωση παραμέτρων rendering και αλλαγή της θέσης και του προσανατολισμού της κάμερας

```

#version 460 core
#extension GL_EXT_gpu_shader4 : enable
layout (location = 0) in vec3 aPos;
layout (location = 1) in float index;



uniform mat4 projection;
uniform mat4 translation;
uniform mat4 LookAt;
uniform mat4 view;

out float depth;

void main(){
  gl_Position=projection*LookAt*view*translation*vec4(aPos.x,aPos.y,aPos.z,1.0);
  vec3 v_eye_pos = (LookAt*view * vec4(aPos.x,aPos.y,aPos.z,1.0)).xyz; // In eye coords.
  depth = -v_eye_pos.z;

}
```

Κώδικας 11: Depth vertex shader

```

float* pixD = new float[FINAL_FBO_WIDTH * FINAL_FBO_HEIGHT * 3];
float* invertedD = new float[FINAL_FBO_WIDTH * FINAL_FBO_HEIGHT * 3];
unsigned short* conv = new unsigned short[FINAL_FBO_HEIGHT * FINAL_FBO_WIDTH * 3];

glReadPixels(0, 0, FINAL_FBO_WIDTH, FINAL_FBO_HEIGHT, GL_RGB,/*type of data in bytes*/
    GL_FLOAT, pixD/* returned data*/);

for (int i = 0; i < FINAL_FBO_HEIGHT; i++) {
    //first line is FBO_WIDTH * 3
    memcpy(&invertedD[(FINAL_FBO_HEIGHT - 1 - i) * FINAL_FBO_WIDTH * 3], &pixD[i *
    FINAL_FBO_WIDTH * 3], sizeof(float) * FINAL_FBO_WIDTH * 3);


}
for (int i = 0; i < FINAL_FBO_HEIGHT * FINAL_FBO_WIDTH * 3; i++)
{
    //cout<<"\t"<<invertedD[i];
    conv[i] = (unsigned short)(round(invertedD[i]));

}
cv::Mat imaged(FINAL_FBO_HEIGHT, FINAL_FBO_WIDTH, CV_16UC3, &conv[0]);
cv::cvtColor(imaged, imaged, cv::COLOR_BGR2GRAY);
```

Κώδικας 12: Ανάγνωση depth buffer και μετασχηματισμός των τιμών του

```

glm::vec2 getScreenCoords(glm::mat4 projection, glm::mat4 LookAt2, glm::vec3 point, int
    num_pnts, int FBO_WIDTH, int FBO_HEIGHT) {
    glm::vec2 cameraSpaceCoords;

    glm::vec4 ClippedCoordinates = projection * LookAt2 * glm::vec4(point, 1.0f) ;
    cout << "Clip_coords  " << ClippedCoordinates.x << "\t" << ClippedCoordinates.y << "\
    t" << ClippedCoordinates.w << endl;
    double canvas_normalized_x = ClippedCoordinates.x / ClippedCoordinates.w;
    double canvas_normalized_y = ClippedCoordinates.y / ClippedCoordinates.w;

    cout << "canvas norm x  " << canvas_normalized_x << endl;
    cout << "canvas norm y  " << canvas_normalized_y << endl;



    double Screen_x = (canvas_normalized_x + 1) * (FBO_WIDTH / 2);
    double Screen_y = (canvas_normalized_y + 1) * (FBO_HEIGHT / 2);



    double raster_x = round(Screen_x);
    double raster_y = round(FBO_HEIGHT - Screen_y);

    cout << "Screen x , y " << raster_x << "\t" << raster_y << endl;

    cameraSpaceCoords = glm::vec2(raster_x, raster_y);

    return cameraSpaceCoords;
}
...
glm::vec2 raster_coords;
raster_coords = camera.getScreenCoords(projection, viewEst, glm::vec3(0.0,0.0,0.0), 1,
    FBO_WIDTH, FBO_HEIGHT);
```

Κώδικας 13: Υπολογισμός εικονοσυντεταγμένων του κέντρου του BB

```
def main():

    # 1st step : upscale masks to original images size
    images = [f for f in listdir(paths['DATA_PATH']) if isfile(
        join(paths['DATA_PATH'], f)) and os.path.splitext(f)[1] == '.png']

    if args.resize is not None:

        print("Resizing masks to {} , {} ...".format(
            args.resize[0], args.resize[1]))

        for im in images:

            img = Image.open(paths['DATA_PATH']+'/'+im)
            resized = img.resize(
                tuple((int(args.resize[0]), int(args.resize[1]))))
            resized.save(paths['APPLYIED_MASKS']+'/'+im)

        print("Done resizing")

```

```
    images_res = [f for f in listdir(paths['APPLYIED_MASKS']) if isfile(
        join(paths['APPLYIED_MASKS'], f)) and os.path.splitext(f)[1] == '.png']

    if args.rename == True:
        print("Renaming images...")
        for im in images_res:
            os.rename(paths['APPLYIED_MASKS']+'/'+im, paths['APPLYIED_MASKS'] +
                      '/'+str(int(im.split('_')[1].split('.')[0]))+".png")
        print("Done renaming")
    if args.apply_mask == True:

        print("Applying mask to rgb images...")
        # get the initial RGB images
        rgb = [f for f in listdir(paths['RGB_IMAGES']) if isfile(
            join(paths['RGB_IMAGES'], f)) and os.path.splitext(f)[1] == '.png']

        for im in rgb:

            noprefix = int(im.split('.')[0])

            or_mask = cv2.imread(
                paths['APPLYIED_MASKS'] + "/"+str(noprefix)+'.png')
            kernel = np.ones((5, 5), np.uint8)
            dila_mask = cv2.dilate(or_mask, kernel, iterations=1)
            cv2.imwrite(paths['DILATED_MASKS'] + "/" +
                        str(noprefix)+'.png', dila_mask)

            rgbimg = Image.open(paths['RGB_IMAGES'] + "/"+im)

            maskmg = Image.open(
                paths['DILATED_MASKS'] + "/"+str(noprefix)+'.png')
            mskimg_mask = maskmg.convert("RGB")

            new = Image.composite(rgbimg, mskimg_mask, maskmg.convert('L'))

            # maskmg.paste(rgbimg,mask=mskimg_mask)

            new.save(paths['RGB_MASKED'] + "/"+im, format="png")
        print("Masks applied")

    # run matlab script

    if args.homography == True:
        print("Applying homography...")
        os.chdir(paths['MATLAB_PROJECT'])
        matlabexe = 'cmd /k  "C:/Program Files/MATLAB/R2022a/bin/matlab.exe" '
        options = "/minimize /nosplash /nodesktop /r "
        matfunc = "BbcTran('"+paths['RGB_MASKED'] + \
            "','"+paths['MASKED_ROTATED']+"','"+paths['CENTROIDS_FILE']+"','"+paths['
    OUT_ROT']+"');exit; "
        print(matlabexe+options+matfunc)
        os.system(matlabexe+options+matfunc)

        print("Done applying the homography")


```

```
if __name__ == "__main__":
    main()
```

Κώδικας 14: Resize εικόνων, εφαρμογή μασκών και ομογραφίας σε γλώσσα Python

```
def calc_bins(error):
  bin_size = np.ceil(np.percentile(error,90)) / 20
  plot_vals = np.ceil(np.percentile(error,90))

  return bin_size,plot_vals
def histogram_angularABS(abspt,frequency,bins,angerror,title=None,typE="angular"):
  bin_size,plot_vals = calc_bins(angerror)
  bin_size = 0.2
  if path.exists(os.path.join(abspt,'plots')) == False:
    os.mkdir(os.path.join(abspt,'plots'))

  ang_plot = [(plot_vals+2*bin_size + 1) if i>plot_vals else i for i in angerror]

  bins=np.arange(0,plot_vals+2*bin_size,bin_size)
  fig=plt.figure(figsize=(14,5))
  print(bins)
  freq, bins, patches = plt.hist(np.clip(ang_plot, bins[0], bins[-1]), bins=bins)
  plt.gca().set(title=title, ylabel='Frequency',xlabel=typE)
  #plt.xticks(range(int(max(angerror))))



  bin_centers = np.diff(bins)*0.5 + bins[:-1]
  n = 0
  for fr, x, patch in zip(freq, bin_centers, patches):
    height = int(freq[n])
    plt.annotate("{}".format(height),
                 xy = (x, height),             # top left corner of the histogram bar
                 xytext = (0,0.2),             # offsetting label position above its bar
                 textcoords = "offset points", # Offset (in points) from the *xy* value
                 ha = 'center', va = 'bottom'
                 )
    n = n+1

  ticks_to_plot = np.arange(0,plot_vals+bin_size,bin_size);
  xlabels = np.array(np.round(ticks_to_plot,1), dtype=str)
  xlabels[-1] = str(plot_vals)+'+'
  print("Bins size : ",bin_size)
  print("Xlabels size :",len(xlabels))
  print(xlabels)

  N_labels = len(xlabels)
  #plt.xlim([0, 600])
  plt.xticks(ticks_to_plot, xlabels)
  #ax.set_xticklabels(xlabels)

  plt.ylim(0,800)
  plt.savefig(os.path.join(os.path.join(abspt,'plots'),title+'.png'))
 ...
plotPE.histogram_angularABS(plot_path, len(imID), 26, finalAR[0:len(
        imID), 2], title="Inference Results - Absolute Angular error histogram", typE="
```

```
    Angular error(deg)")
```

Κώδικας 15: Παράδειγμα αυτόματης δημιουργίας ιστογράμματος

```

glm::mat4 UpdateView() {
    if (theta_angle == 0) {
      CameraPos.x = radius * cos(glm::radians(float(phi_angle))) * sin(glm::radians(float
    (0.00001)));
      CameraPos.z = radius * sin(glm::radians(float(phi_angle))) * sin(glm::radians(float
    (0.00001)));
      CameraPos.y = radius * cos(glm::radians(float(0.00001)));
    }
    else {
      CameraPos.x = radius * cos(glm::radians(float(phi_angle))) * sin(glm::radians(float
    (theta_angle)));
      CameraPos.z = radius * sin(glm::radians(float(phi_angle))) * sin(glm::radians(float
    (theta_angle)));
      CameraPos.y = radius * cos(glm::radians(float(theta_angle)));


    }

    glm::vec3 centerVec = glm::vec3(stats.Xmean, stats.Ymean, stats.Zmean);
    glm::mat4 LookAT = glm::lookAt(CameraPos + centerVec, centerVec, glm::vec3(0, 1, 0));

    return LookAT;
  }
```

Κώδικας 16: Συνάρτηση υπολογισμού του πίνακα LookAt

```
    ifstream scene_gt("path/to/json/file.json");
    ordered_json js = ordered_json::parse(scene_gt);
    //cout << js["3"].at(0)["cam_R_m2c"].at(0);
    std::list<std::array<float, 16>> gt_poses;

    for (auto& el : js.items())
    {

        auto r = el.value()[0]["cam_R_m2c"];
        auto t = el.value()[0]["cam_t_m2c"];

        float tt = float(r.at(0));
        //cout << tt;

        float rt_test[16] = {
            float(r.at(0)),float(r.at(1)),float(r.at(2)),float(t.at(0)),
            float(r.at(3)),float(r.at(4)),float(r.at(5)),float(t.at(1)),
            float(r.at(6)),float(r.at(7)),float(r.at(8)),float(t.at(2)),
            0.0,0.0,0.0,1.0
        };
        std::array<float, 16> rt_test_arr;
        std::copy(rt_test, rt_test + 16, begin(rt_test_arr));
        gt_poses.push_back(rt_test_arr);
    }

     if (!glfwWindowShouldClose(window)) {
```

```
27
28          int imid = 1;
29          for (std::list<std::array<float, 16>>::iterator it = gt_poses.begin(); it !=
    gt_poses.end(); it++) {
30
31                  //based on the K matrix calculate the projection matrix for rendering
32                  float prj[16] = {
33                      2 * fx / float(FBO_WIDTH),-2 * 0 / float(FBO_WIDTH),0,0,
34                      0,2 * fy / float(FBO_HEIGHT),0,0,
35                      1 - 2 * (cx / float(FBO_WIDTH)),2 * (cy / float(FBO_HEIGHT)) - 1,-(
    farP + nearP) / (farP - nearP),-1,
36                      0,0,2 * farP * nearP / (nearP - farP),0
37                  };
38
39                  glm::mat4 projection = glm::make_mat4(prj);
40
41                  ...
42
43                  glBindFramebuffer(GL_FRAMEBUFFER, framebuffer.ID);
44                  glViewport(0, 0, FBO_WIDTH, FBO_HEIGHT);
45
46                  renderer.__init__(); //clear and initialize draw buffer
47
48                  //enable depth testing
49                  glPolygonMode(GL_FRONT_AND_BACK, GL_FILL);
50                  glEnable(GL_DEPTH_TEST);
51
52                  ...
53
54                  shader.use();
55
56                  ...
57
58                  glBindVertexArray(vao);
59
60                  glEnable(GL_POLYGON_OFFSET_FILL);
61                  glDisable(GL_BLEND);
62                  glPolygonOffset(1, 1);
63
64                  glDrawArrays(GL_TRIANGLES, 0, vrt.size() / 4);
65                  shader1.use();
66
67                  ...
68
69                  glBindVertexArray(0);
70                  glBindVertexArray(VAO_lines);
71                  ...
72
73                  glPolygonMode(GL_FRONT_AND_BACK, GL_LINE);
74                  glDepthFunc(GL_LEQUAL);
75
76                  ...
77
78                  glDrawArrays(GL_LINES, 0, model.vertices.size() * 2);
79
80                  //unbind vertex array
```

```
81                  glBindVertexArray(0);
```

Κώδικας 17: Ανάγνωση του .json αρχείου με τις εκτιμώμενες πόζες στο BOP format και διαδικασία οπτικοποίησης με τη χρήση του HLR