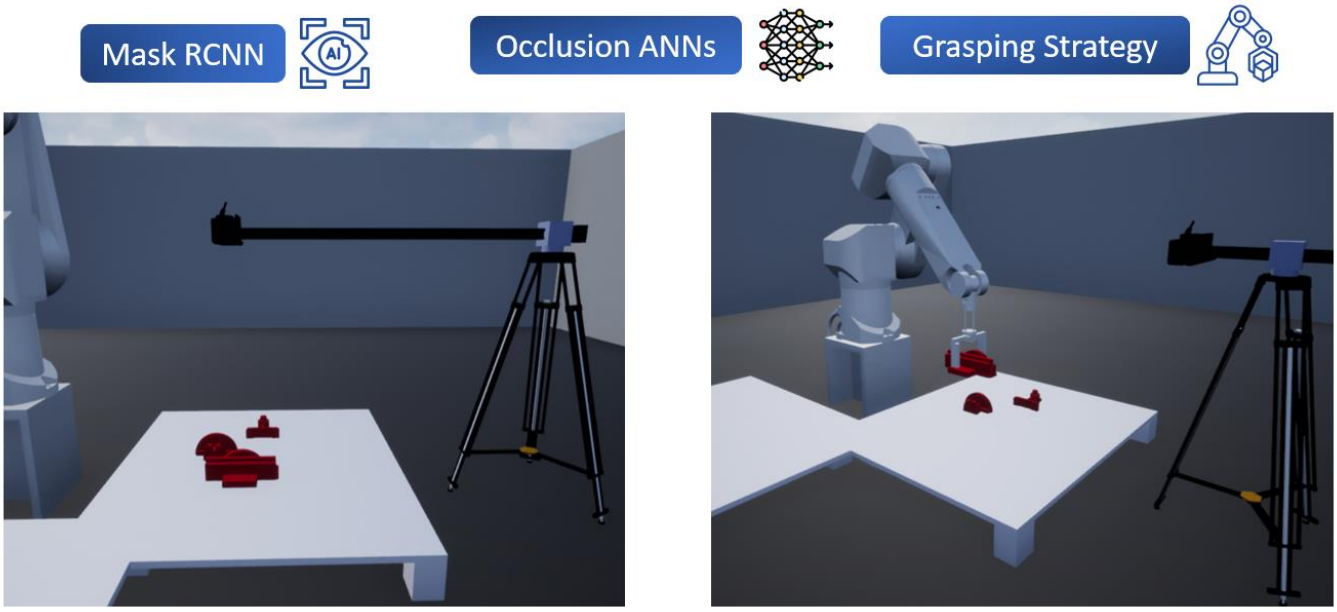
National Technical University of Athens
School of Mechanical Engineering
Section of Manufacturing Technology

Robotic Arm Manipulation for Object Detection & Grasping in Occlusion Environments Using Machine Vision & Neural Networks

Diploma Thesis

Pavlos Chionidis



Supervisor: Prof. Panorios Benardos (NTUA)

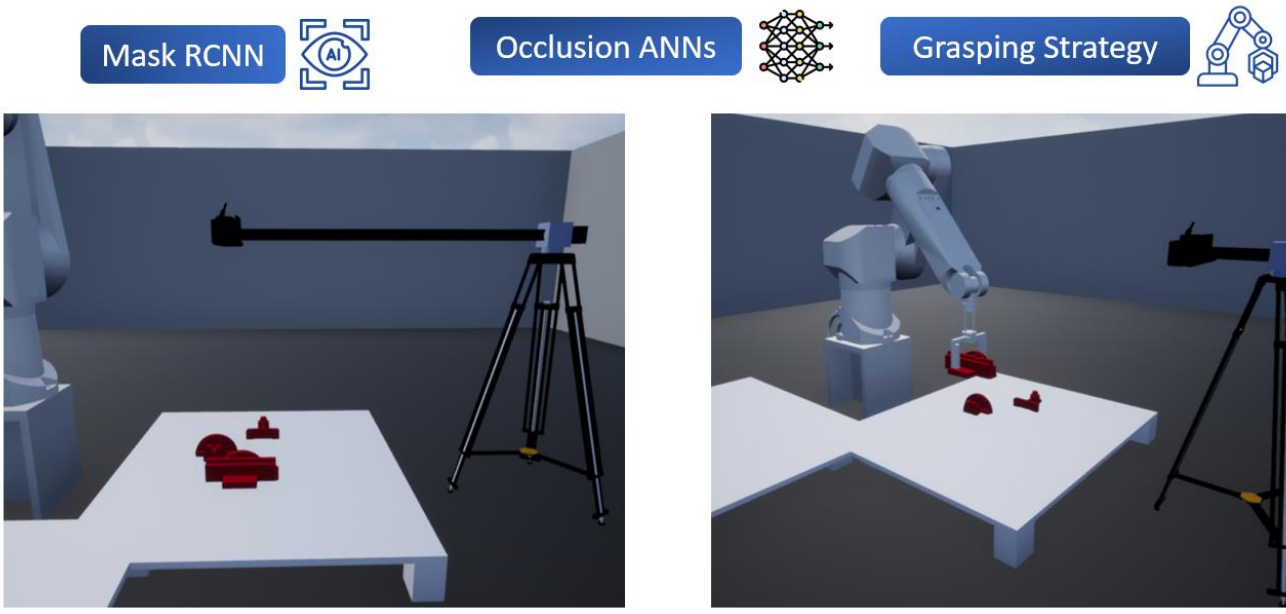
Athens 2024

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Μηχανολόγων Μηχανικών
Τομέας Τεχνολογίας των Κατεργασιών

Ρομποτικός Βραχίονας με Σύστημα Μηχανικής Όρασης & Λογικής για την Αναγνώριση & Αρπαγή Αλληλοεπικαλυπτόμενων Αντικειμένων

Διπλωματική Εργασία

Παύλος Χιονίδης



Υποβλέπων: Πανώριος Μπενάρδος, Καθηγητής ΕΜΠ

Αθήνα 2024

--Σημειώνεται ότι το παρών κείμενο αποτελεί μια εκτεταμένη περίληψη στα Ελληνικά της διπλωματικής μου εργασίας που είναι γραμμένη στα Αγγλικά--

-- Κενή Σελίδα --

Ευχαριστίες

Η παρούσα διπλωματική εργασία πραγματοποιήθηκε κατά το ακαδημαϊκό έτος 2023-2024 στον Τομέα Τεχνολογίας των Κατεργασιών της Σχολή Μηχανολόγων Μηχανικών του Εθνικού Μετσόβιου Πολυτεχνείου.

Αρχικά θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή Πανώριο Μπενάρδο, για την καθοδήγηση και την υποστήριξή του σε όλη τη διάρκεια αυτής της εργασίας. Επιπλέον θα ήθελα να ευχαριστήσω και τον Τομέα Τεχνολογίας των Κατεργασιών για την παροχή πρόσβασης στον απαραίτητο εξοπλισμό.

Επίσης θα ήθελα να ευχαριστήσω την οικογένειά μου για τη συνεχή ενθάρρυνση και υποστήριξη που μου προσφέρανε κατά τις ακαδημαϊκές μου διατριβές.

Έπειτα θα ήθελα να ευχαριστήσω τους φίλους μου για την υποστήριξή και τη βοήθειά που μου προσφέρανε στη βελτίωση των συγγραφικών μου δεξιοτήτων καθ’ όλα τα ακαδημαϊκά έτη.

Η λήξη της παρούσας διπλωματικής, αντικατοπτρίζει όχι μόνο την ατομική προσπάθεια, αλλά τη συλλογική υποστήριξη που έλαβα από τα κοντινά μου πρόσωπα όλα αυτά τα χρόνια. Ευελπιστώ ότι η εργασία αυτή θα συμβάλει θετικά στην εξέλιξη του τομέα της μηχανικής όρασης και της ρομποτικής.

Παύλος Χιονίδης
Αθήνα, Φεβρουάριος 2024

Περιεχόμενα

Abstract

This diploma thesis focuses on the simulation of a robotic manipulator with machine vision to grasp objects in occlusion environments. Specifically, the robotic arm in use is the Stäubli RX90L with 6 degrees of freedom, equipped with a gripper. The study delves into the assessment of inverse kinematics and trajectory control for the robotic arm during grasping operations. A crucial augmentation to the system involves the incorporation of machine vision systems for object detection. The machine vision component encompasses the calibration of Kinect V2 RGB and depth images, along with the mapping of RGB images to depth images. Subsequently, the theoretical underpinnings of Convolutional Neural Networks are elucidated, with the chosen network for this thesis being the Mask RCNN. The methodology for training the networks is explicated, followed by comprehensive testing in diverse occlusion environments.

A paramount aspect of this work involves addressing the occlusion problem in the grasping methodology, determining which objects are occluded, by implementing neural networks that utilize features extracted from the Mask RCNN network and depth images. The acquired occlusion data undergoes a logic-based algorithm, delineating the item to be grasped and generating an effective grasping strategy. The entire process is simulated within the integrated MATLAB-Simulink Unreal Engine environment, providing a holistic evaluation of the proposed methodologies.

Περίληψη

Η παρούσα διπλωματική εργασία εστιάζει στην προσομοίωση ενός ρομποτικού βραχίονα με μηχανική όραση για να συλλαμβάνει αντικείμενα σε περιβάλλοντα αλληλοεπικάλυψης αντικειμένων. Πιο συγκεκριμένα, ο ρομποτικός βραχίονας που χρησιμοποιείται είναι ο Stäubli RX90L με 6 βαθμούς ελευθερίας, εξοπλισμένος με αρπάγη. Η μελέτη εμβαθύνει στην υλοποίηση της αντίστροφης κινηματικής και του ελέγχου της τροχιάς του βραχίονα κατά τη σύλληψη αντικειμένων. Κρίσιμη είναι η ενσωμάτωση του συστήματος μηχανικής όρασης για την ανίχνευση αντικειμένων. Το κεφάλαιο της μηχανικής όρασης περιλαμβάνει τη βαθμονόμηση και την αντιστοίχιση των εικόνων RGB στις εικόνες βάθους του Kinect V2. Στη συνέχεια, διευκρινίζονται οι θεωρητικές βάσεις των συνελικτικών νευρωνικών δικτύων, με το δίκτυο που επιλέχθηκε για τη διατριβή αυτή, να είναι το Mask RCNN. Έπειτα ακολουθείται η μεθοδολογία για την εκπαίδευση του δικτύου, με την δοκιμή του σε διάφορα περιβάλλοντα αλληλοεπικάλυψης.

Μια πρωταρχική πτυχή της εργασίας περιλαμβάνει η αντιμετώπιση του προβλήματος αλληλοεπικάλυψης των αντικειμένων. Για τον προσδιορισμό των αντικειμένων που επικαλύπτονται, χρησιμοποιήθηκαν νευρωνικά δίκτυα με είσοδο χαρακτηριστικά που εξάγονται από το δίκτυο Mask RCNN και τις εικόνες βάθους. Η έξοδος από τα νευρωνικά δίκτυα χρησιμοποιείται από αλγόριθμο, ο οποίος οριοθετεί το αντικείμενο που πρέπει να συλληφθεί και δημιουργεί την στρατηγική σύλληψης. Η όλη διαδικασία προσομοιώνεται στο περιβάλλον του MATLAB-Simulink-Unreal Engine, παρέχοντας μια ολική εικόνα των προτεινόμενων μεθοδολογιών.

Συντομογραφίες

2D: Two Dimensional

3D: Three Dimensional

ANN: Artificial Neural Network

API: Application Programming Interface

ASIC: Application Specific Integrated Circuits

BFGS: Broyden–Fletcher–Goldfarb–Shanno

CAD: Computer Aided Design

CNN: Convolutional Neural Network

CPU: Central Processing Unit

DOF: Degrees Of Freedom

FBX: Filmbox Format

GPU: Graphics Processing Unit

MS COCO/ COCO: Microsoft Common Objects in Context

PAR: Pixel Aspect Ratio

RCNN: Region-based Convolutional Neural Network

RAM: Random Access Memory

ReLU: Rectified Linear Unit

RGB: Red Green Blue

RGB-D: Red Green Blue - Depth

RPN: Region Proposal Network

STL: Standard Triangle Language

UE: Unreal Engine

URDF: Unified Robot Description Format

VOC: Visual Object Classes

VRAM: Video Random Access Memory

YOLO: You Only Look Once

1. Εισαγωγή

Η μηχανική όραση παράλληλα με τις προηγμένες τεχνικές σύλληψης αντικειμένων σε ρομποτικούς βραχίονες βρίσκει ευρεία εφαρμογή σε σενάρια που χαρακτηρίζονται από δυναμικά και ποικίλα περιβάλλοντα. Παρόλα αυτά το πρόβλημα της αλληλοεπικάλυψης μεταξύ των αντικειμένων που βρίσκονται εντός της περιοχής λειτουργείας του βραχίονα δεν έχει επιλυθεί και παραμένει στο επίκεντρο της έρευνας για πολλά ιδρύματα και ερευνητές παγκοσμίως. Η λύση του προβλήματος αυτού θα μπορούσε να φέρει πιο κοντά την ανθρωπότητα προς την υλοποίηση ενός συστήματος μηχανικής όρασης - ρομποτικού βραχίονα με ανθρώπινη αντίληψη. Μια τέτοια υλοποίηση έχει τη δυνατότητα να φέρει επανάσταση σε διάφορους τομείς, όπως στην παραγωγή υλικών αγαθών άλλα και στην καθημερινή διεπαφή του ανθρώπου με τα ρομπότ [5][6].

Παρόλα αυτά ενώ η παράγεται σημαντική έρευνα στον τομέα αυτό, η εφαρμογή των μεθοδολογιών που προτείνονται συχνά δεν διατίθεται ο ακριβές τρόπος υλοποίησης, αφήνοντας αβεβαιότητες σχετικά με την εφαρμογή των αντίστοιχων μεθοδολογιών. Επιπλέον, στις περιπτώσεις που υπάρχουν πρακτικά παραδείγματα, συνήθως αυτά δοκιμάζονται σε ρομποτικούς βραχίονες μικρής κλίμακας έτσι ώστε να αποφευχθούν πιθανές ζημιές που μπορούν να προκύψουν από σφάλματα στην αναπτυγμένη μεθοδολογία και τον κώδικα. Επομένως, η προσομοίωση του συστήματος σε ένα εικονικό περιβάλλον είναι μεγάλης αξίας, μιας και μπορεί να βελτιώσει την εφαρμογή των μεθοδολογιών και να επιταχύνει τη διαδικασία δοκιμής, ελαχιστοποιώντας παράλληλα τους κινδύνους που σχετίζονται με τον πειραματισμό στον πραγματικό κόσμο.

Σε αυτή τη διατριβή κατασκευάστηκε ένα σύστημα προσομοίωσης της ανίχνευσης αντικειμένων που βρίσκονται πάνω σε μια τράπεζα, την επιλογή του αντικειμένου προς αρπαγή και τέλος της εικονικής αναπαράστασης της όλης διαδικασίας. Η υλοποίηση αυτού του συστήματος πραγματοποιείται εντός του περιβάλλοντος MATLAB-Unreal Engine. Επιπλέον δίνεται η πλήρης μεθοδολογία για την εκμάθηση του συστήματος μηχανικής όρασης με πραγματικά δεδομένα, όπου το εξαγόμενο δίκτυο μπορεί να ανιχνεύσει και αντικείμενα στο εικονικό περιβάλλον. Έπειτα, το πρόβλημα της αλληλοεπικάλυψης αντιμετωπίζεται με την εφαρμογή νευρωνικών δικτύων, στην οποία βασίζεται και η μεθοδολογία σύλληψης των αντικειμένων. Τέλος μια σχετικά απλή αλλά και αποτελεσματική τεχνική σύλληψης υλοποιείται με βάση τη στροφή των αντικειμένων σε σχέση με το καθολικό σύστημα συντεταγμένων.

2. Βιβλιογραφική Ανασκόπηση

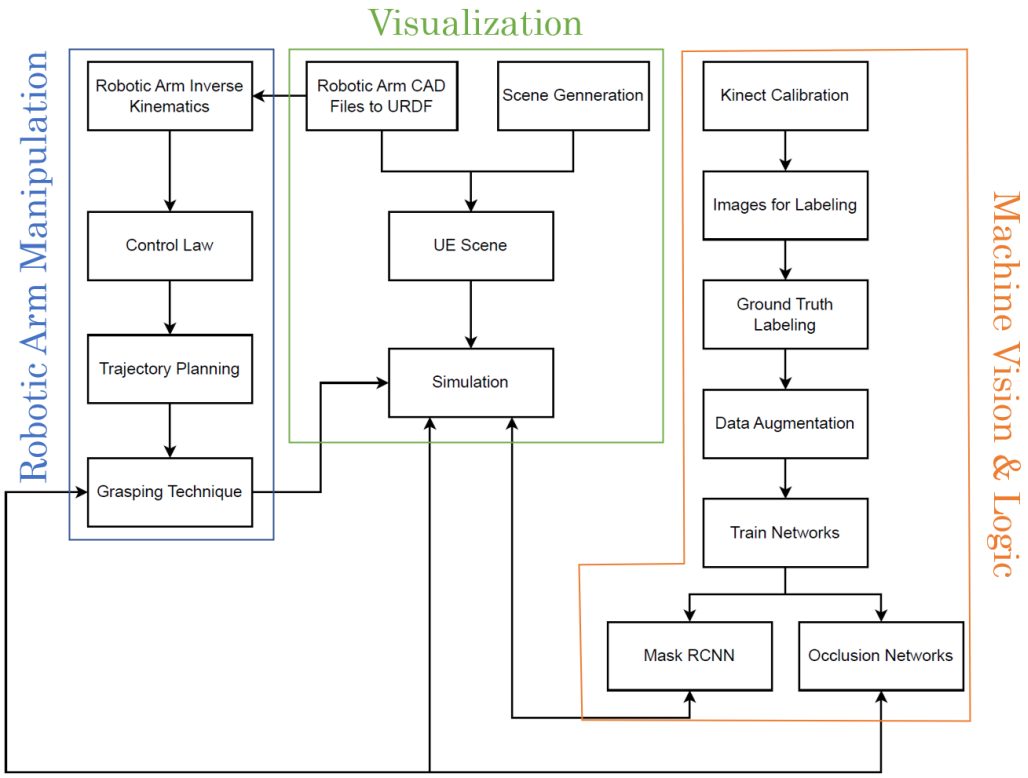
Το 2014, η δημιουργία R-CNN [7] σηματοδότησε μια σημαντική πρόοδο στην ανίχνευση αντικειμένων σε εικόνες, πετυχαίνοντας 30% αύξηση σε επίδοση σε σχέση με το προηγούμενο καλύτερο αποτέλεσμα στο Pascal VOC 2012. Αυτή η καινοτομία επέτρεψε την ανίχνευση πολλαπλών αντικειμένων σε μια εικόνα, αναζωπυρώνοντας το ενδιαφέρον για την ενσωμάτωση ρομποτικών βραχιόνων με μηχανική όραση. Επιπλέον, η εμφάνιση προσιτών αισθητήρων βάθους, όπως το Microsoft Kinect (2010) και το Asus Xtion (2011), προκάλεσε αύξηση σε έρευνες που επικεντρώνονται σε μεθόδους βασισμένες σε χαρακτηριστικά εικόνας και σε δεδομένα βάθους.

Οι Aarth R. και Rishma G. διερεύνησαν τις δυνατότητες του Mask R-CNN στην ανίχνευση αντικειμένων απορριμμάτων στο έδαφος. Τα ευρήματά τους έδειξαν ανώτερη απόδοση σε σύγκριση με άλλα δίκτυα, επιτυγχάνοντας ακρίβεια ανίχνευσης ίση με 97%. Σε μια ξεχωριστή μελέτη, οι Zhen Li et al. [11] ερεύνησαν τη χρήση του YOLO για την ανίχνευση αντικειμένων σε ένα κινητό ρομπότ εξοπλισμένο με ρομποτικό βραχίονα και κάμερα RGB-D. Πρότειναν ένα επιπλέον συνελικτικό νευρωνικό δίκτυο για σύλληψη αντικειμένων μέσω του βραχίονα, δημιουργώντας δύο επιπλέον «εικόνες» - με την μία να απεικονίζει την ποιότητα σύλληψης σε ολόκληρη και η άλλη δείχνει τη γωνία σύλληψης. Το παράδειγμα αυτό δείχνει την ευελιξία των συνελικτικών δικτύων πέρα από την ανίχνευση αντικειμένων, επιδεικνύοντας την αποτελεσματικότητά τους σε διάφορες εργασίες.

Οι Tuan-Tang Le et al. [8] χρησιμοποίησαν μια παρόμοια μέθοδο με αυτή που χρησιμοποιείται σε αυτή την διπλωματική εργασία, όπου εκπαιδεύσαν δίκτυο Mask RCNN, εισάγοντας και τα δεδομένα βάθους της εικόνας. Έδειξαν ότι σε σενάρια πραγματικού κόσμου το σύστημα μπορεί να επιτύχει ακρίβεια 90%. Ο Ziyad Tareq N. διεξήγαγε μια έρευνα σχετικά με την αποτελεσματικότητα του Mask RCNN στην ανίχνευση μασκών αντικειμένων σε έντονα αλληλεπικαλυπτόμενα περιβάλλοντα. Στη μελέτη του, το δίκτυο εκπαιδεύτηκε χρησιμοποιώντας τις πλήρες μάσκες των αντικειμένων έναντι μόνο των εμφανίσιμων. Η προσέγγιση αυτή είχε στόχο να αξιολογήσει την απόδοση του δικτύου στον χειρισμό σεναρίων αλληλοεπικάλυψης χωρίς διαχωρισμό των πληροφοριών μάσκας σε εμφανές και μη εμφανές, επιτυγχάνοντας σχεδόν τα ίδια αποτελέσματα εάν είχε κάνει τον διαχωρισμό.

3. Μεθοδολογία

Η μεθοδολογία που ακολουθήθηκε σε αυτή την διπλωματική, μπορεί να χωριστεί σε 3 διαφορετικές υποενότητες. Η Εικόνα 3-1 δείχνει το διάγραμμα ροής που ακολουθήθηκε για την αντιμετώπιση όλων των διαφορετικών προβλημάτων που διερευνήθηκαν κατά τα πλαίσια της εργασίας.



Εικόνα 3-1: Διάγραμμα ροής μεθοδολογίας.

- **Robotic Arm CAD Files to URDF**

Μέσα στο SOLIDWORKS συναρμολογείται ο ρομποτικός βραχίονας και μέσω ενός επιπρόσθετου πακέτου εξάγεται το αρχείο τύπου ".urdf".

- **Scene Generation**

Μέσα στο SOLIDWORKS κατασκευάζονται όλα τα απαραίτητα αντικείμενα για την προσομοίωση και αυτά εξάγονται ως ".stl". Έπειτα μέσα στο Blender τοποθετούνται στις κατάλληλες θέσεις και εξάγεται όλη η σκηνή σε μορφή αρχείου τύπου ".fbx" για να μπορεί να διαβαστεί από το Unreal Engine.

- **UE Scene**

Μέσα στο MATLAB-Simulink εισάγονται τα αντικείμενα της σκηνής μέσω των αντίστοιχων στοιχείων (Blocks). Ο ρομποτικός βραχίονας εισάγεται σαν ".urdf" αρχείο, ενώ μια επιπλέων συνάρτηση χρησιμοποιείται για να μετατρέψει το σύστημα συντεταγμένων από αυτό του MATLAB σε αυτό του Unreal Engine. Τέλος

εισάγεται και μια RGB-D κάμερα για την λήψη φωτογραφιών στην τράπεζα εργασίας.

- **Kinect Calibration**

Για την διόρθωση της εικόνας που προέρχεται από τον αισθητήρα RGB, χρησιμοποιούνται εργαλεία εντός του MATLAB, αξιοποιώντας φωτογραφίες ενός χαρτιού επιδιόρθωσης (calibration paper) που λήφθηκαν από το Kinect. Έπειτα γίνεται και η επιδιόρθωση του αισθητήρα βάθους λαμβάνοντας πολλές φωτογραφίες (σε διαφορετικά βάθη) ενός αντικειμένου γνωστών διαστάσεων. Τέλος γίνεται η αντιστοίχιση των εικόνων RGB στις εικόνες βάθους.

- **Images for labeling**

Λήψη εικόνων των αντικειμένων με την χρήση του Kinect και ιδικά τοποθετημένων φωτορυθμηκών για την μείωση των σκιών.

- **Ground Truth labeling**

Δημιουργία των δεδομένων εισόδου των νευρωσικών δικτύων μέσω τη εφαρμογής Image Labeler του MATLAB. Χρήση του COCO API για την κατάλληλη μετατροπή των δεδομένων για την εκμάθηση του Mask RCNN.

- **Data augmentation**

Δημιουργία νέων δεδομένων από τις αρχικές εικόνες, επιβάλλοντας διαφορετικά φόντα και μεταφέροντας/περιστρέφοντας τα τεμάχια με τυχαίο τρόπο.

- **Train the Networks**

Εκμάθηση των νευρωνικών δικτύων με την χρήση μεθόδων που υποστηρίζονται από την κάρτα γραφικών για την εξοικονόμηση χρόνου.

- **Robotic Arm Inverse Kinematics**

Δημιουργία της αναλυτικής αντίστροφης κινηματικής του ρομποτικού βραχίονα χρησιμοποιώντας την εντολή analyticalinversekinematics() εντός του MATLAB.

- **Control Law**

Εύρεση της τελικής θέσης του ρομποτικού βραχίονα για την αρπαγή των αντικειμένων χρησιμοποιώντας τα δεδομένα βάθους, τις μάσκες που παράγονται από το Mask RCNN και τα αποτελέσματα από τα νευρωνικά δίκτυα.

- **Trajectory Planning**

Χρήση πολυωνύμου $3^{ου}$ βαθμού για την κατάστρωση των τροχιών στο χώρο τον αρθρώσεων. Για την εύρεση της καλύτερης τελικής πόζας (pose) – σε περίπτωση που υπάρχουν περισσότερες από μια λύσεις- του ρομποτικού βραχίονα, επιλέγεται η πόζα που ελαχιστοποιεί το άθροισμα των τετραγωνικών διαφορών των περιστροφών κάθε άρθρωσης από την αρχική στην τελική λύση.

- **Grasping Technique**

Χρήση των δεδομένων μάσκας και βάθους κάθε αντικειμένου για την επιλογή της σωστής προσέγγισης και αρπαγής.

- **Simulation**

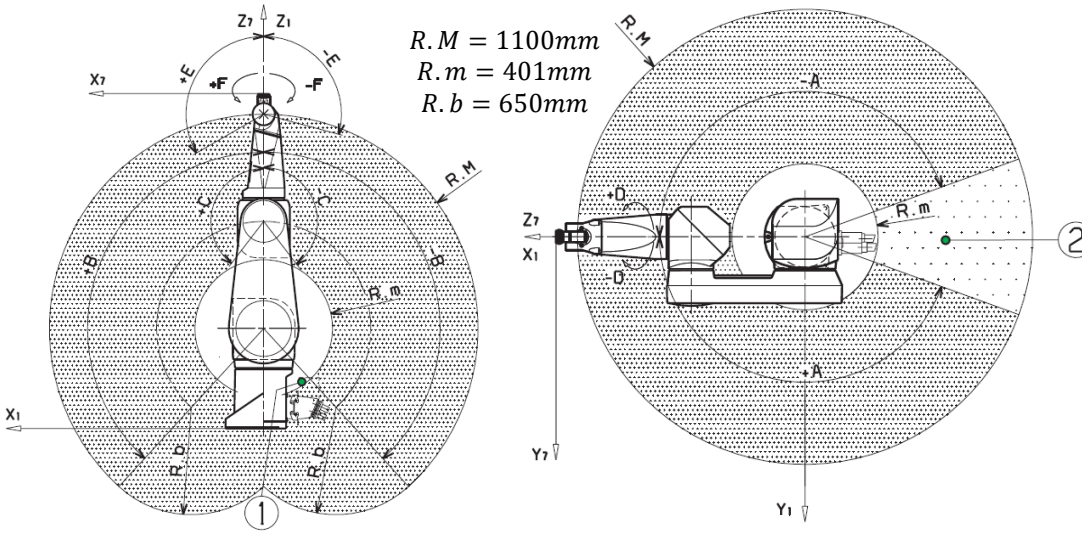
Προσομοίωση όλων των παραγόμενων μεθοδολογιών/δικτύων σε εικονικό περιβάλλον εντός του MATLAB-Simulink και του Unreal Engine.

4. Παράμετροι Συστήματος

Σε αυτή την εργασία, χρησιμοποιείται ένα εικονικό μοντέλο του ρομποτικού βραχίονα "Stäubli RX90L", με μια αρπάγη δύο σημείων. Ο βραχίονας αυτός αποτελείται από 6 αρθρώσεις των οποίων τα χαρακτηριστικά φαίνονται στον Πίνακας 4-1. Ο επιδέξιος χώρος εργασίας του ρομποτικού βραχίονα φαίνεται στην Εικόνα 4-1 ενώ μια πραγματική εικόνα του βραχίονα φαίνεται στην Εικόνα 4-2.

Πίνακας 4-1: Παράμετροι του Stäubli RX90L.

Joint	1	2	3	4 (1)	5	6
Amplitude (°)	320	275	285	540	225	540 (2)
Working range distribution (°)	A ± 160	B ± 137.5	C ± 142.5	D ± 270	E +120 -105	F ± 270
Nominal speed (°/s)	236	200	286	401	320	580
Maximum speed (°/s)	356	356	296	409	800	1125 (3)
Angular resolution ($°.10^{-3}$)	0.87	0.87	0.72	1	1.95	2.75



Εικόνα 4-1: Προσπελάσιμος χώρος εργασίας του Stäubli RX 90L.

Εικόνα 4-2: Εικόνα του Stäubli RX 90L.

Για τους σκοπούς της μηχανικής όρασης έχει χρησιμοποιηθεί το Kinect V2 που φαίνεται στην Εικόνα 4-3.

Εικόνα 4-3: Φωτογραφία του Kinect V2.

5. Ρομποτική

5.1 Θεωρία

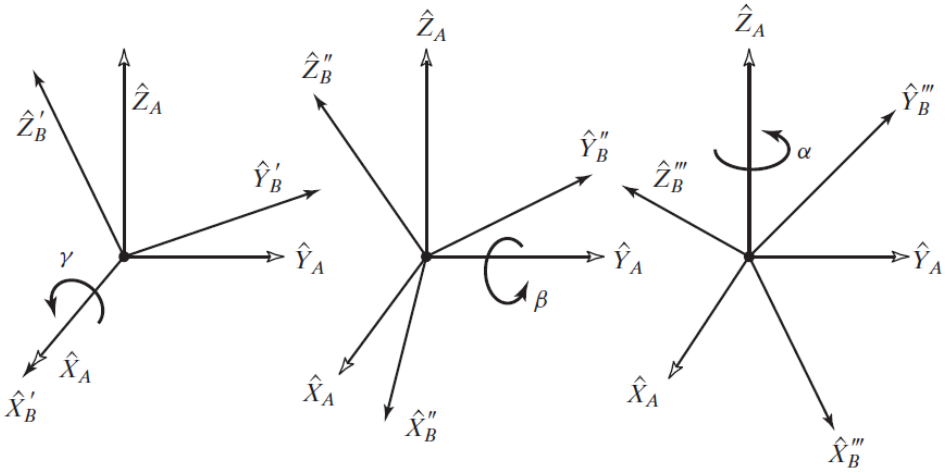
5.1.1 Κινηματική

Για την πλήρη οριοθέτηση ενός σώματος στον καρτεσιανό χώρο δεν απαιτείται μόνο η θέση του [x,y,z] άλλα και ο προσανατολισμός του. Το εργαλείο που επιτρέπει την μοντελοποίηση του προσανατολισμού με μαθηματικό τρόπο είναι ο πίνακας περιστροφής που φαίνεται στην Εξίσωση *5-1*.

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{21} \\ r_{31} & r_{31} & r_{33} \end{bmatrix} \qquad 5\text{-}1$$

Διάφοροι μέθοδοι υπάρχουν για την κατασκευή του πίνακα προσανατολισμούς άλλα ο πιο γνωστός είναι αυτός των γωνιών του Euler όπου μόνο 3 γωνίες απαιτούνται για την πλήρη αναπαράσταση του προσανατολισμού. Η Εικόνα 5-1 δείχνει ένα παράδειγμα των γωνιών του Euler με την Εξίσωση *5-2* να δείχνει τον τρόπο υλοποίησης του [16].

$$R_{XYZ} = R_X(\gamma) \cdot R_{Y'}(\beta) \cdot R_{Z''}(a) \qquad 5\text{-}2$$



Εικόνα 5-1: Γωνίες Euler X-Y-Z [16].

Παρά την χρησιμότητα και την απλότητα των γωνιών Euler, αυτοί μπορεί να οδηγήσουν το σύστημα σε ιδιόμορφα σημεία. Για τον λόγο αυτό χρησιμοποιούνται μέθοδοι που αξιοποιούν 4 παραμέτρους για την αναπαράσταση του πίνακα περιστροφής. Η πιο συχνά χρησιμοποιούμενη μέθοδος αξιοποιεί τους παραμέτρους Euler (ε_i) με την Εξίσωση *5-3* να δείχνει τον τρόπο με τον οποίο αξιοποιούνται.

$$R = (2 \cdot \varepsilon_4^2 - 1) \cdot I_3 + 2 \cdot \varepsilon_4 \cdot \boldsymbol{\varepsilon}^{\times} + 2 \cdot \boldsymbol{\varepsilon} \cdot \boldsymbol{\varepsilon}^T$$

Όπου:

Πρώτοι τρείς παράμετροι: $\boldsymbol{\varepsilon} = \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \end{bmatrix} = \hat{\boldsymbol{k}} \cdot \sin\left(\frac{\theta}{2}\right) = \frac{1}{4 \cdot \varepsilon_4} \begin{bmatrix} r_{32} - r_{23} \\ r_{13} - r_{31} \\ r_{21} - r_{12} \end{bmatrix}$ *5-3*

Τέταρτη παράμετρος: $\varepsilon_4 = \cos\left(\frac{\theta}{2}\right) = 0.5 \cdot \sqrt{1 + r_{11} + r_{22} + r_{33}}$

Skew Symmetric Πίνακας: $\varepsilon^{\times} = \begin{bmatrix} 0 & -\varepsilon_3 & \varepsilon_2 \\ \varepsilon_3 & 0 & -\varepsilon_1 \\ -\varepsilon_2 & \varepsilon_1 & 0 \end{bmatrix}$

Με την χρήση του πίνακα περιστροφής (R_{3x3}) και του διανύσματος θέσης (P_{3x1}) μπορεί να κατασκευαστεί ο ομογενής μετασχηματισμός (Transformation Matrix) όπως φαίνεται στην Εξίσωση *5-4* που περιγράφει πλήρως την θέση και τον προσανατολισμό ενός σώματος.

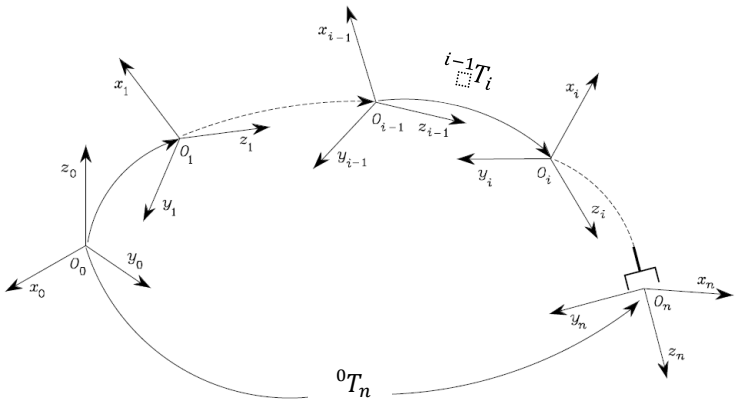
$$T = \begin{bmatrix} R_{3x3} & P_{3x1} \\ 0_{1x3} & 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad 5\text{-}4$$

Στην περίπτωση μιας ανοικτής κινηματικής αλυσίδας όπως φαίνεται στην Εικόνα 5-2, είναι δυνατό να βρεθεί ο μετασχηματισμός από την αρχή (βάση) της αλυσίδας, έως το τέλος αυτής χρησιμοποιώντας την Εξίσωση *5-5*

$$^{0}T_e = {}^{0}T_1 \cdot {}^{1}T_2 \cdot \ldots \cdot {}^{6}T_e \qquad 5\text{-}5$$

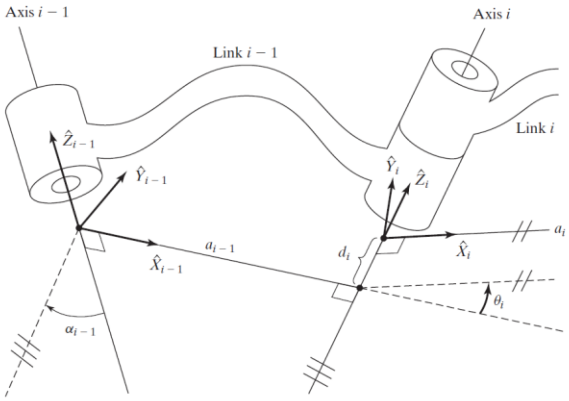
Όπου:

Μετασχηματισμός από το του συστήματος i σε σχέση με το σύστημα συντεταγμένων του j: ${}^{j}T_i$



Εικόνα 5-2: Μετασχηματισμοί μεταξύ συστημάτων συντεταγμένων σε μια ανοικτή κινηματική αλυσίδα [17].

Οι μετασχηματισμοί μεταξύ των συνδέσμων της ανοικτής κινηματικής αλυσίδας-ενός ρομποτικού βραχίονα, προέρχονται από πως είναι συνδεδεμένοι οι σύνδεσμοι μεταξύ τους. . Ο ομογενής μετασχηματισμός($^{i-1}T_i$) μεταξύ δύο συνδέσμων μπορεί να κατασκευαστεί με διάφορους τρόπους όπως αυτός του «Denavit Hardenberg» [16][17].



Εικόνα 5-3: Συσχέτιση συστημάτων συντεταγμένων μεταξύ δύο αρθρώσεων [16].

Στην περίπτωση αυτής της διπλωματικής, οι μετασχηματισμοί μεταξύ ενός συνδέσμου όπως φαίνεται στην Εικόνα 5-3 εξάγονται απευθείας από το ".urdf" αρχείο.

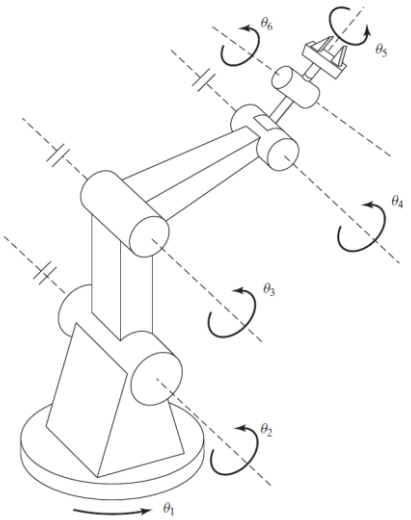
Επομένως σε περίπτωση απαίτησης ενός συγκεκριμένου μετασχηματισμού για την αρπαγή ενός αντικειμένου (${}^0T_{e,desired}$), απαιτείται μια λύση της μορφής που φαίνεται στην Εξίσωση *5-6*:

$$ {}^0T_{e,desired} = X(q_1, q_2, \dots, q_n) \qquad 5\text{-}6$$

Όπου:

Γωνίες περιστροφής των αρθρώσεων: q_i

Από το 2022, το MATLAB έχει εισάγει την εντολή analyticalinversekinematics(),η οποία μπορεί να βρει την αναλυτική αντίστροφη κινηματική ενός βραχίονα που αποτελείται μόνο από αρθρωτούς συνδέσμους με τους τελευταίους τρεις να είναι σε μορφή καρπού «wrist configuration» όπως φαίνεται στην Εικόνα 5-4[18]. Σημειώνεται ότι ρομποτικός βραχίονας που χρησιμοποιείται στην διπλωματική αυτή, ανήκει σε αυτή την κατηγορία.



Εικόνα 5-4: Βραχίονας με τις τελευταίες τρεις αρθρώσεις σε μορφή καρπού[16].

5.2 Έλεγχος Βραχίονα

5.2.1 Νόμος ελέγχου

Για τον έλεγχο του ρομποτικού βραχίονα χρησιμοποιούνται τα χαρακτηριστικά που εξάγονται από το σύστημα μηχανικής όρασης. Τα χαρακτηριστικά αυτά είναι σημεία εντός της εικόνας με δεδομένα βάθους. Επομένως χρησιμοποιώντας την Εξίσωση *5-7* [19] μπορεί να βρεθεί η συσχέτιση τον σημείων αυτόν (στην εικόνα) με της αποστάσεις στον καρτεσιανό χώρο από το σύστημα συντεταγμένων της κάμερας.

$$X_P = \frac{u_x - c_x}{f_x} \cdot Z_P$$
$$Y_P = \frac{u_y - c_y}{f_y} \cdot Z_P \qquad 5\text{-}7$$

Όπου

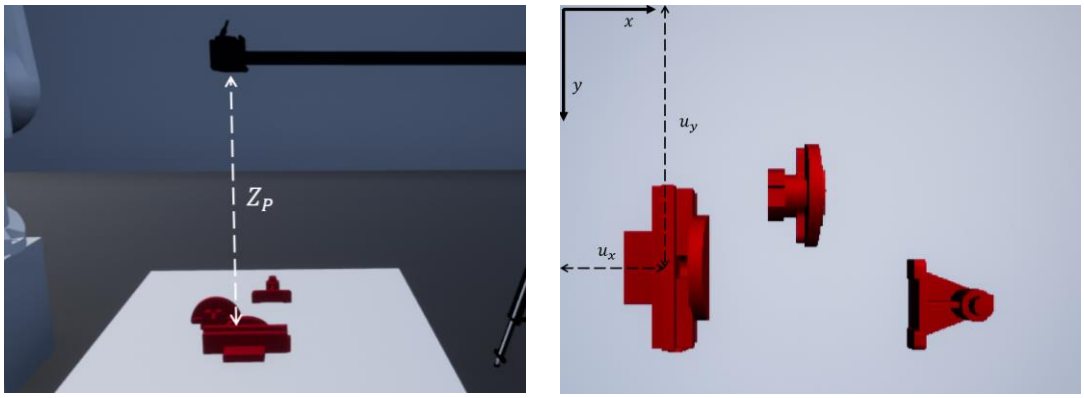
u_x, u_y: Είναι οι αποστάσεις ενός σημείου κατά τον x και y άξονα.

c_x, c_y: Είναι το σημείο του κέντρου της εικόνας κατά τον x και y άξονα.

f_x, f_y: Είναι τα εστιακά βάθη.

X_P, Y_P, Z_P: Είναι οι καρτεσιανές συντεταγμένες του αντικειμένου σε σχέση με την κάμερα.

Η απόσταση του σημείου από την κάμερα Z_P βρίσκεται από τα δεδομένα βάθους. Επόμενος το X_P και το Y_P μπορούν να υπολογιστούν. Εικόνα 5-5 δείχνει ένα παράδειγμα του υπολογισμού αυτών των συντεταγμένων.



(a) (b)

Εικόνα 5-5: Παράδειγμά απόστασης Z_P από την κάμερα(a) και της θέσης του αντικειμένου στην εικόνα (b).

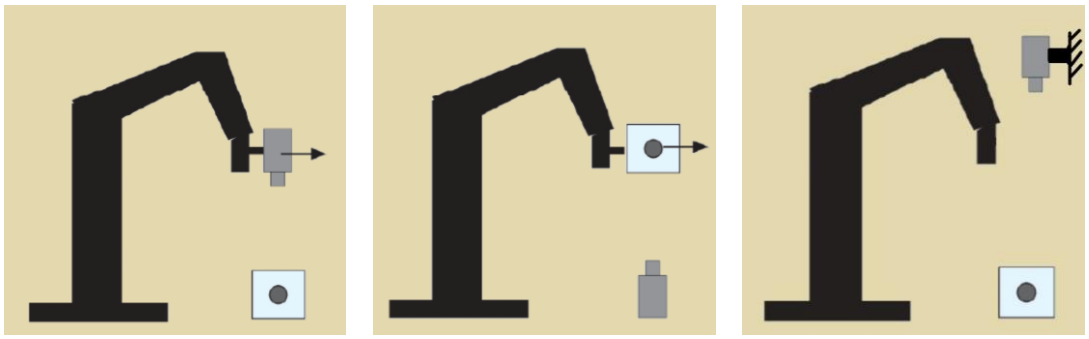
Επομένως για να πιάσει ο ρομποτικός βραχίονας το αντικείμενο μπορεί να χρησιμοποιηθεί ο ομογενής μετασχηματισμός με διάνυσμα θέσης το $[X_P, Y_P, Z_P]$ και προσανατολισμό κάθετο στο τραπέζι. Άρα αξιοποιώντας τον μετασχηματισμό της κάμερας με την βάση του ρομπότ μπορεί να βρεθεί ο μετασχηματισμός μεταξύ αντικειμένου και ρομπότ όπως φαίνεται στην Εξίσωση *5-8*. Επομένως χρησιμοποιώντας την αναλυτική αντίστροφη κινηματική μπορεί να βρεθούν οι γωνίες περιστροφής του ρομποτικού βραχίονα για τον συγκεκριμένο πίνακα μετασχηματισμού.

$$^{b}T_{obj} = {}^{b}T_{c} \cdot {}^{c}T_{obj} \qquad 5\text{-}8$$

Όπου

Ο ομογενής μετασχηματισμός μεταξύ της βάσης του βραχίονα και της κάμερας: $^{b}T_{c}$

Στην περίπτωση της εργασίας αυτής θεωρείται ότι η κάμερα διατηρείται σταθερή όπως φαίνεται στην Εικόνα 5-6 (c).

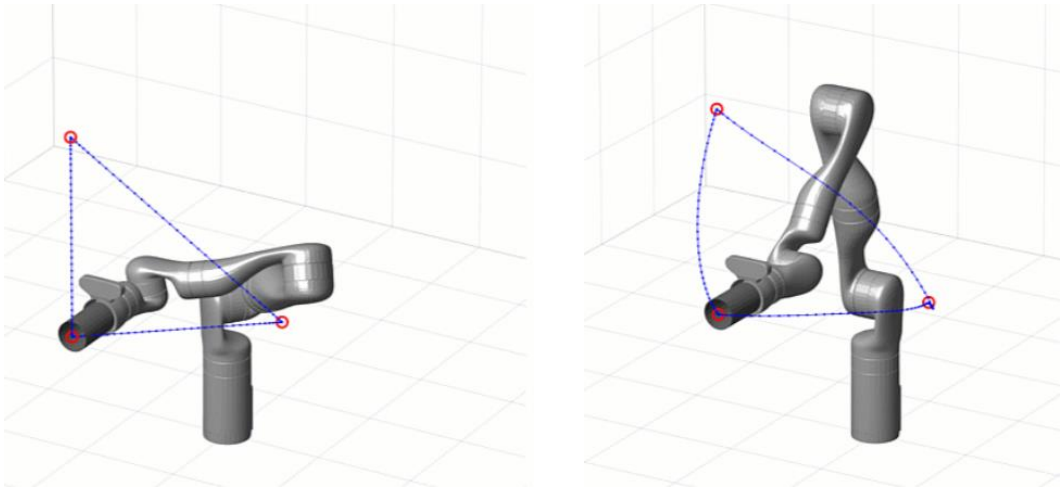


(a) (b) (c)

Εικόνα 5-6: Μάτι πάνω στο χέρι (a), Μάτι κοιτάει το χέρι (b), Ξεχωριστό μάτι και χέρι (c)

5.2.2 Έλεγχος Τροχιάς

Οι ρομποτικοί βραχίονες παρουσιάζουν δύο κύριους τύπους τροχιών: Τον καρτεσιανό έλεγχο τροχιάς και έλεγχο τροχιάς στον χώρο των αρθρώσεων. Στον καρτεσιανό έλεγχο, έμφαση δίνεται στη ρύθμιση της κίνησης του τελικού δράσης μέσω καθορισμένων θέσεων και προσανατολισμών στον Καρτεσιανό χώρο. Από την άλλη πλευρά, ο έλεγχος της τροχιάς στον χώρο των αρθρώσεων εστιάζει στη διαχείριση της κίνησης μεμονωμένων αρθρώσεων για την επίτευξη επιθυμητών θέσεων. Σημειώνεται επίσης ότι στον έλεγχο τροχιάς, δεν υπάρχει πιθανότητα ο βραχίονας να βρεθεί σε ιδιόμορφο σημείο μιας και δεν χρησιμοποιείται η αντίστροφη κινηματική. Η διαφορά μεταξύ αυτών των δύο τύπων ελέγχου φαίνεται στο Εικόνα 5-7.



(a) (b)

Εικόνα 5-7: Έλεγχος στον καρτεσιανό χώρο (a) και έλεγχος στον χώρο των αρθρώσεων (b).

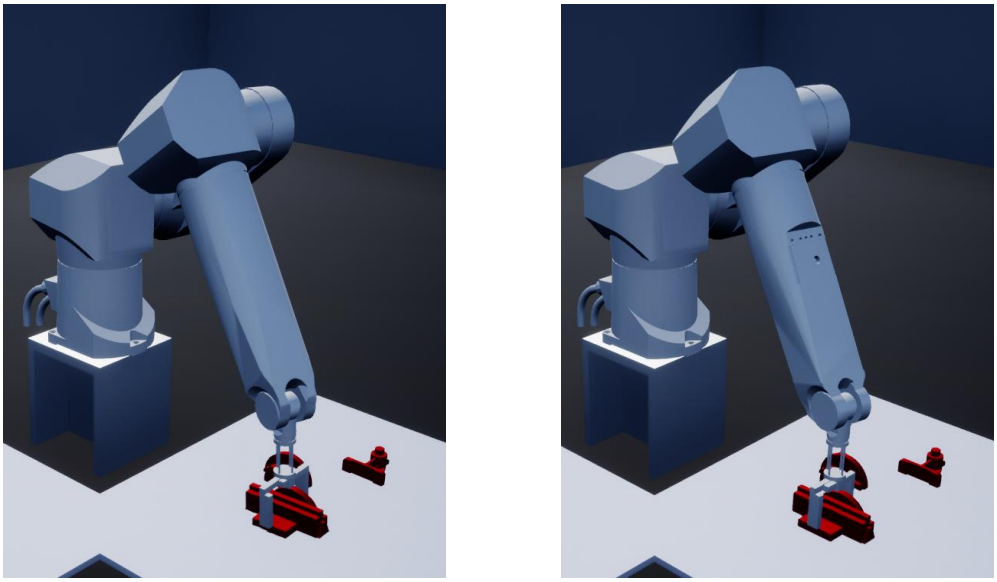
Κατά την αντίστροφη κινηματική είναι πιθανό να υπάρχουν παραπάνω από μια λύσεις που δίνουν τον ίδιο μετασχηματισμό του τελικού σημείου δράσης όπως φαίνεται στην Εικόνα 5-8. Σε αυτή την εργασία η συνάρτηση του αθροίσματος των τετραγώνων χρησιμοποιείται ως προς ελαχιστοποίηση όπως φαίνεται στην Εξίσωση *5-9* για την επιλογή της τελικής λύσης.

$$min\{L_j\} = \min\{\sum_{i=1}^{6}(q_{i,end} - q_{i,init})^2\} \qquad 5\text{-}9$$

Όπου:

$q_{i,end}$: Είναι η τελική γωνία της άρθρωσης i.

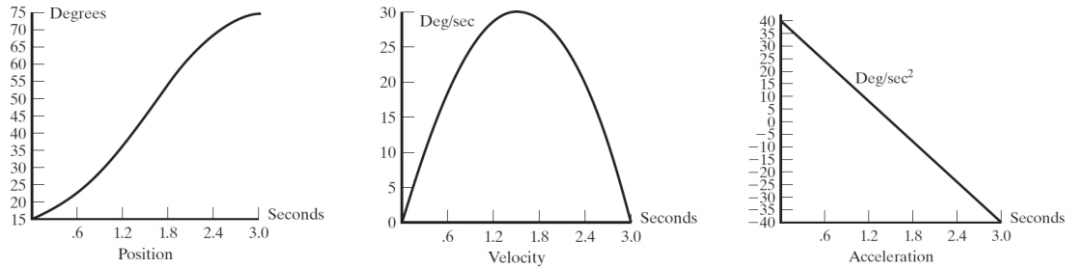
$q_{i,init}$: Είναι η αρχική γωνία της άρθρωσης i.



(a) *(b)*

Εικόνα 5-8: Ίδιο σημείο και προσανατολισμός του τελικού σημείου δράσης σε δυο διαφορετικές πόζες.

Η συναρτήσεις παρεμβολής των γωνιών των αρθρώσεων ήταν τα πολυώνυμα τρίτου βαθμού με μηδενικές ταχύτητες περιστροφής στο τελικό και το αρχικό σημείο όπως φαίνεται στην Εικόνα 5-9.



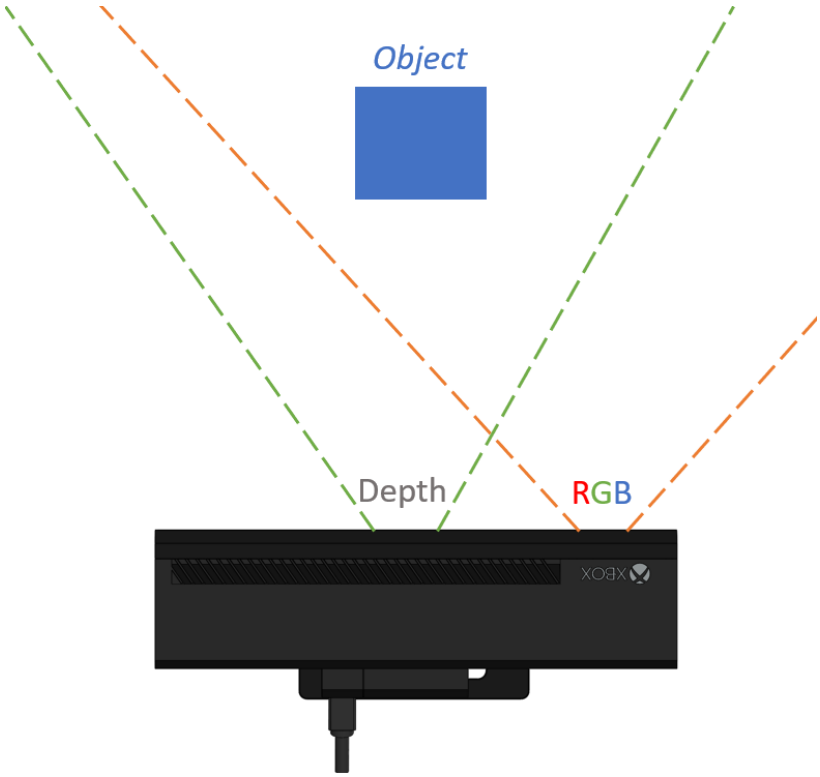
(a) *(b)* *(c)*

Εικόνα 5-9: Παράδειγμα πολυωνυμικής παρεμβολής.

6. Σύστημα Μηχανικής Όρασης

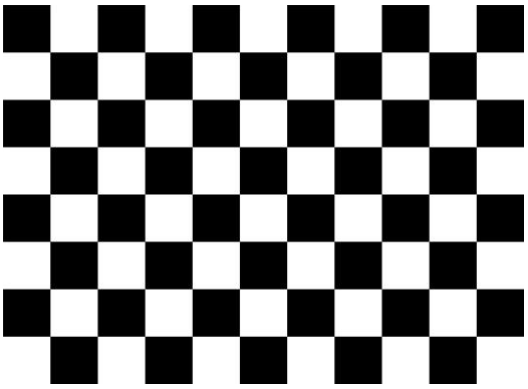
6.1 Βαθμονόμηση του Kinect V2

Η κάμερες του Kinect V2 -RGB και βάθους- δεν βρίσκονται στο ίδιο φυσικό σημείο, δεν έχουν ίδια ανάλυση και ούτε την ίδια αναλογία ύψους-πλάτους όπως φαίνεται στην Εικόνα 6-1 και από τις παραμέτρους των αισθητήρων στον Πίνακα 6-2 και 6-3. Επομένως επιλέχθηκε να αντιστοιχηθεί η εικόνα RGB στην εικόνα βάθους μιας και η εικόνα βάθους έχει μικρότερη ανάλυση.



Εικόνα 6-1: Παράδειγμα της τοποθέτησης των αισθητήρων του Kinect V2

Για την εύρεση των παραμέτρων της κάμερας RGB χρησημοποιείθηκε χαρτί βαθμονόμησης με το σχέδιο της Εικόνα 6-2 και το επιπρόσθετο πρόγραμμα βαθμονόμησης καμερών εντός του MATLAB.



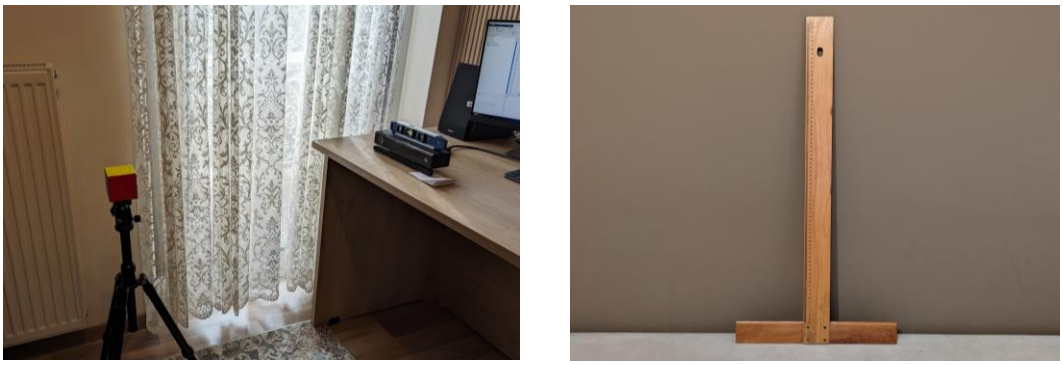
Εικόνα 6-2: Σχέδιο βαθμονόμησης με τετράγωνα μεγέθους 25x25mm.

Για την εύρεση της φυσικής απόστασης της κάμερας RGB από την κάμερα βάθους, χρησιμοποιήθηκε ένας χάρακας με ορθή γωνία ο οποίος τοποθετήθηκε στο κέντρο του Kinect. Έπειτα τοποθετήθηκε ένας κύβος του Rubik's περίπου στο κέντρο της εικόνας βάθους σύμφωνα με τις ενδείξεις του χάρακα και λήφθηκαν διάφορες φωτογραφίες. Στην Εικόνα 6-3 φαίνονται μερικές ενδεικτικές φωτογραφίες από την διαδικασία ενώ η Εξίσωση *6-1* χχρησημοποιείθηκε για να βρεθεί η απόσταση της κάμερας RGB από το κέντρο του Kinect που είναι ίση με 95mm (όπως είναι και στα αρχεία CAD του Kinect).

$$X = \frac{(u_x - c_x) \cdot Z}{f_x}$$

Όπου
Η απόσταση του σημείου κατά την x κατεύθυνση: $X[m]$
Το εστιακό βάθος κατά την x κατεύθυνση: $f_x\ [m]$
Η απόσταση της κάμερας από το σημείο: $Z\ [m]$
Η απόσταση του σημείου από το κέντρο της εικόνας: $u_x - c_x\ [pixels]$

6-1



(α) *(b)*

Εικόνα 6-3: Μέθοδος λήψη των φωτογραφιών για την βαθμονόμησης (α), χάρακας βαθμονόμησης (β)

Έπειτα υπολογίζεται η αναλογία μεταξύ των εικονοστοιχείων των δυο εικόνων να είναι ίση με $PAR = 3.05$ σύμφωνα με την Εξίσωση *6-2*.

$$PAR = \frac{pixel\ distance\ from\ RGB\ image}{pixel\ distance\ from\ Depth\ image}$$

6-2

Μετά η αντιστοίχιση της εικόνας RGB στην εικόνα βάθους ακολούθησε λαμβάνοντας διαφορετικές φωτογραφίες με διαφορετικά βάθη (Z) του ίδιου αντικειμένου σε ένα προκαθορισμένο διάστημα βάθος [500,1400] mm. Ο Πίνακας 6-1 δείχνει τις υπολογιζόμενες διαφορές του ίδιου σημείου στις δύο εικόνες. Σημειώνεται ότι το μέγεθος που χρησιμοποιείται είναι εικονοστοιχεία βάθους. Όπως είναι εμφανές η διαφορά κατά τον y άξονα παραμένει σταθερή και ίση με $dy = -9$, το οποίο σημαίνει ότι η εικόνα RGB πρέπει α μεταφερθεί 9 εικονοστοιχεία βάθους προς τα πάνω. Σε αντίθεση η διαφορά κατά τον x άξονα δε παραμένει σταθερή και άρα χρησιμοποιώντας πολυωνυμική παλινδρόμηση στα σημεία εξάγεται η Εξίσωση *6-3*.

Πίνακας 6-1: Διαφορές των εικονοστοιχείων κατά x και y μεταξύ της εικόνας RGB και της εικόνας βάθους.V2.

Depth Value [mm]	dx [pixels]	dy [pixels]
543	25	-9
561	24	-9
622	20	-9
764	15	-9
855	13	-8
949	12	-9
1036	10	-9
1355	6	-9

$$x_{new} = x_{old} - 5 \cdot 10^{-5} \cdot d^2 + 0.1078 \cdot d + 68.46 \quad \text{6-3}$$

Where the depth value in [mm] at the specified position (x_{old}, y): d

Κόβοντας την εικόνα κατά 25 εικονοστοιχεία από τα δεξιά για να μην υπάρχουν σημεία μηδενικού βάθους, η υπολειπόμενη εικόνα είναι διαστάσεων: [417,340] (width, height). Σημαντικό είναι ότι η εικόνα RGB μπορεί να λάβει διαστάσεις ίσες με: [1273,1038] όπου το κάθε εικονοστοιχείο βάθους αντιστοιχεί σε 3.05 RGB εικονοστοιχεία.

Πίνακας 6-2: Παράμετροι κάμερας RGB

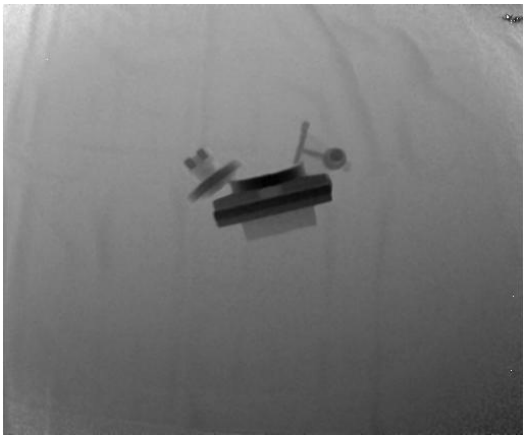
Resolution	Principal Point	Focal Length	Radial Distortion[1]
$[1920,1080]_{w,h}$	$[945,557]$	$[1053,1053]_{x,y}$	$K_1 = 0.04$

Πίνακας 6-3: Παράμετροι κάμερας βάθους.

Resolution	Principal Point	Focal Length	Radial Distortion
$[512,424]_{w,h}$	[256,212]	$[365,365]_{x,y}$	$K_1 = 0.2$

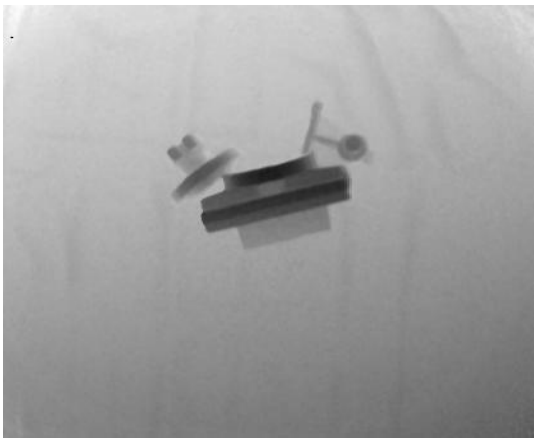
(α)

[1] Παράμετροι σύμφωνα με το μοντέλο Brown-Conrady: https://en.wikipedia.org/wiki/Distortion_(optics). Όλοι οι υπόλοιποι παράμετροι είναι ίσοι με το 0.



(b)

Εικόνα 6-4: Αρχικές εικόνες, RGB εικόνα (a) εικόνα βάθους (b).



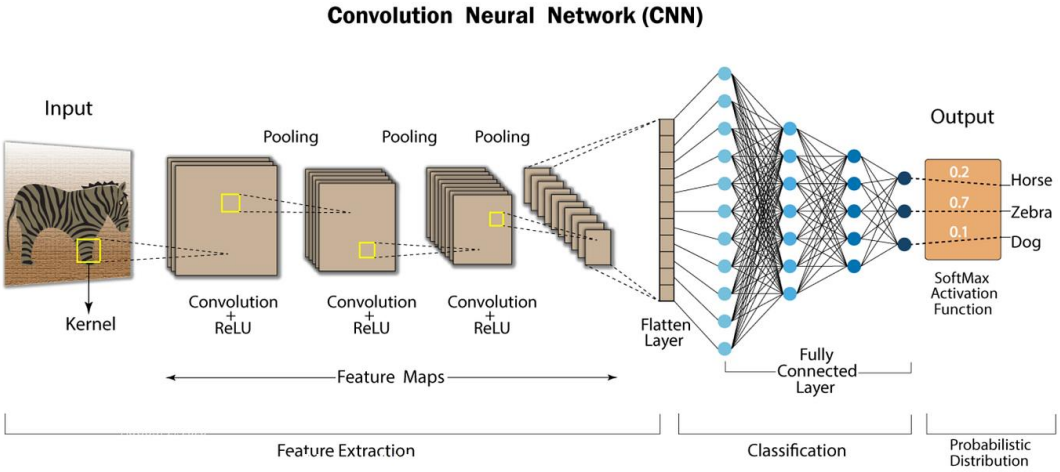
(a) (b)

Εικόνα 6-5: Βαθμονομημένες εικόνες, RGB εικόνα (a) εικόνα βάθους (b).

6.2 Θεωρία Συνελικτικών Νευρωνικών Δικτύων

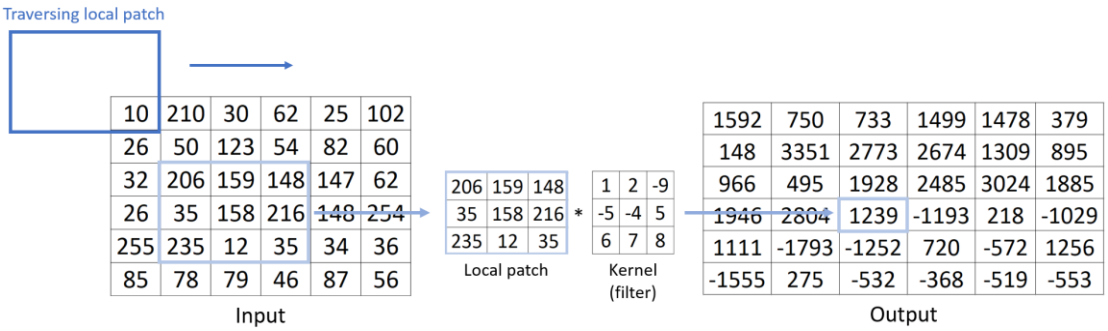
6.2.1 Ταξινομητής Εικόνας

Τα ΣΝΔ (Συνελικτικά Νευρωνικά Δίκτυα) ή στα αγγλικά CNN (Convolutional Neural Network) είναι μία ιδική κατηγορία των νευρωνικών δικτύων που ασχολούνται με την ανάλυση εικόνων. Μια συνηθισμένη υποκατηγορία αυτών είναι η ταξινόμηση των εικόνων σε διαφορετικές κατηγορίες, σύμφωνα με τα αντικείμενα που βρίσκονται εντός της εικόνας (Image Classifier). Η Εικόνα 6-6 δείχνει μια σχηματική αναπαράσταση ενός τέτοιου δικτύου.



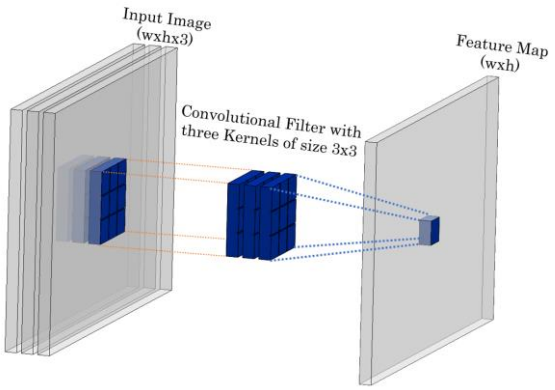
Εικόνα 6-6: Σχηματική αναπαράσταση ενός CNN.

Όπως το όνομα προδίδει, ένα συνελικτικό δίκτυο έχει την πράξη της συνέλιξης ως κύρια μέθοδο εξαγωγής στοιχείων από την εικόνα. Ένα απλό παράδειγμα της πράξης της συνέλιξης φαίνεται στην Εικόνα 6-7. Σημειώνεται ότι στα συνελικτικά δίκτυα αυτό που μεταβάλλεται κατά την εκμάθηση είναι οι συντελεστές επίδρασης του φίλτρου συνέλιξης.



Εικόνα 6-7: Παράδειγμα της συνέλιξης με φίλτρο 3x3.

Στην περίπτωση που η είσοδος εκτείνεται και στην τρίτη διάσταση (πχ RGB εικόνα) η συνέλιξη γίνεται σε κάθε διαφορετικό κανάλι με ένα φίλτρο (filter) τριών διαστάσεων όπως φαίνεται στην Εικόνα 6-8.



Εικόνα 6-8: Παράδειγμα συνελικτικού φίλτρου να επενεργεί σε μία εικόνα.

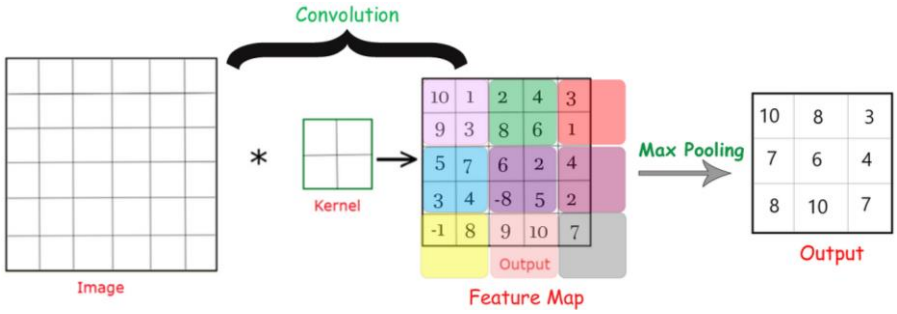
Το αποτέλεσμα της συνέλιξης περνάει από μια συνάρτηση ενεργοποίησης όπως φαίνεται στην Εικόνα 6-10. Σημειώνεται ότι μπορεί να υπάρχουν περισσότερα από

ένα φίλτρα που δρουν την ίδια εικόνα – χάρτη χαρακτηριστικών (feature map). Οι πιο συνήθης συναρτήσεις ενεργοποίησης είναι αυτές που φαίνονται στις Εξισώσεις *6-4* & *6-5*.

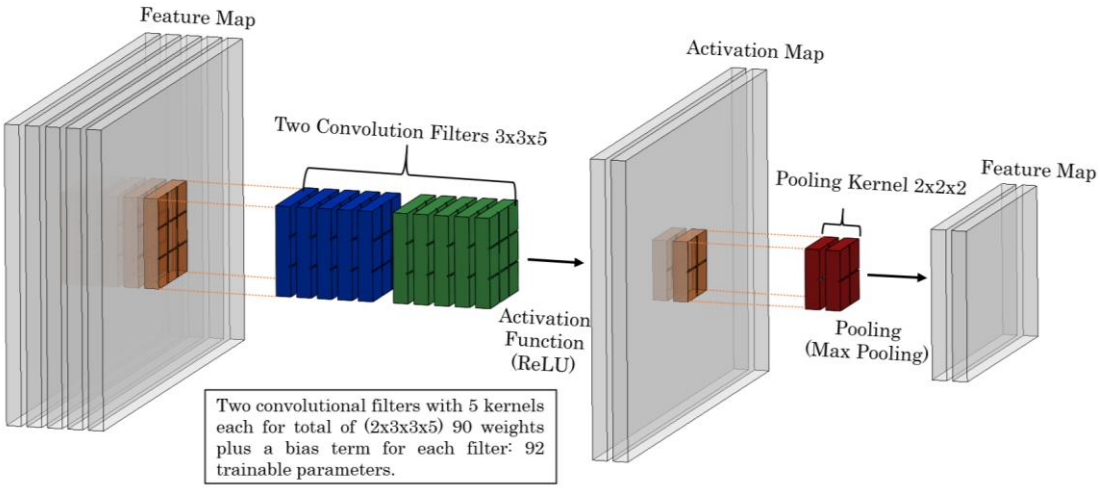
$$ReLU: f(x) = \max(0, x) \quad 6\text{-}4$$

$$Leaky\ ReLU: f(x) = \max(ax, x)\,, a \leq 0.01 \quad 6\text{-}5$$

Μετά την συνάρτηση ενεργοποίησης, συνήθως χρησιμοποιείται η πράξη συγκέντρωσης (pooling) των χαρακτηριστικών η οποία μειώνει την διάσταση του χάρτη χαρακτηριστικών. Η πιο συνήθης πράξη είναι η εξαγωγή του πιο δυνατού χαρακτηριστικού (max pooling), όπου ένα απλό παράδειγμα φαίνεται στην Εικόνα 6-9.

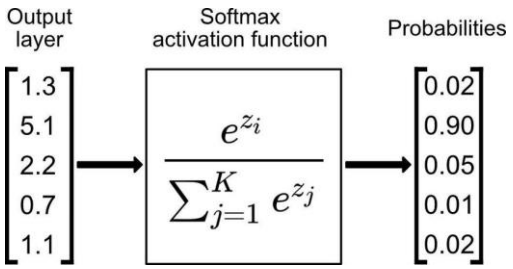


Εικόνα 6-9: Παράδειγμα του max pooling με φίλτρο 2x2.



Εικόνα 6-10: Παράδειγμα ενός συνελικτικού μπλοκ.

Οι παραπάνω πράξεις καταστρώνουν ένα συνελικτικό μπλοκ (Convolutional Block) όπως φαίνεται στην Εικόνα 6-10. Συνδέοντας πολλά τέτοια μπλοκ μεταξύ τους, κατασκευάζεται το δίκτυο εξαγωγής χαρακτηριστικών (Feature extractor - Backbone). Για την ταξινόμηση, χρησιμοποιούνται ένα σύνολο νευρώνων συνδεδεμένων μεταξύ τους. Οι νευρώνες αυτοί λαμβάνουν τα αποτελέσματα της εξαγωγής χαρακτηριστικών και «μαθαίνουν» πια από αυτά αντιστοιχούν σε πιο αντικείμενο. Τέλος για την καθολική ταξινόμηση της εικόνας χρησιμοποιείται στην έξοδο ει συνάρτηση SoftMax όπου ένα απλό παράδειγμα φαίνεται στην Εικόνα 6-11.



Εικόνα 6-11: Συνάρτηση SoftMax.

Η εκμάθηση του δικτύου γίνεται μέσω της μεθόδου οπισθοδρόμησης (Back Propagation) με μια συνήθη μέθοδο μεταβολής των συντελεστών επίδρασης να είναι αυτή της στοχαστικής καθόδου με ορμή (stochastic gradient descent with momentum- SGDM) [1] που φαίνεται στην Εξίσωση 6-6.

$$u_{i+1} = u_i - a\nabla L(u_i) + \gamma(u_i - u_{i-1})$$

Where:
Updated value of the Parameter: u_{i+1}
Previous value of the Parameter: u_i *6-6*
Gradient of the Loss Function: ∇L
Momentum value: γ
Learning rate: a

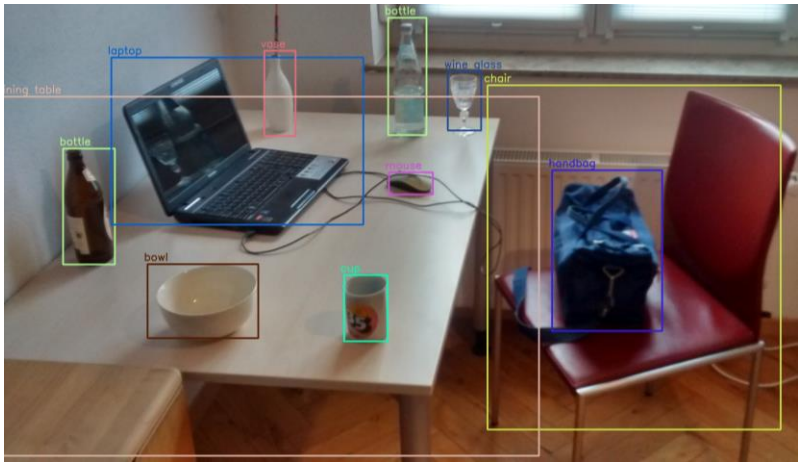
Με συνάρτηση σφάλματος να λαμβάνεται η Κατηγορική Διασταυρούμενη Εντροπία (Categorical Cross-Entropy), που δίνεται στην Εξίσωση 6-7.

$$L = -\sum_{i}^{n} y_i \cdot \log_2 \hat{y}_i$$

Where:
Loss Function: L *6-7*
The true probability (ground truth) of class i: y_i
The predicted probability of class i from the CNN: $\hat{y}_i$
The number of classes: n

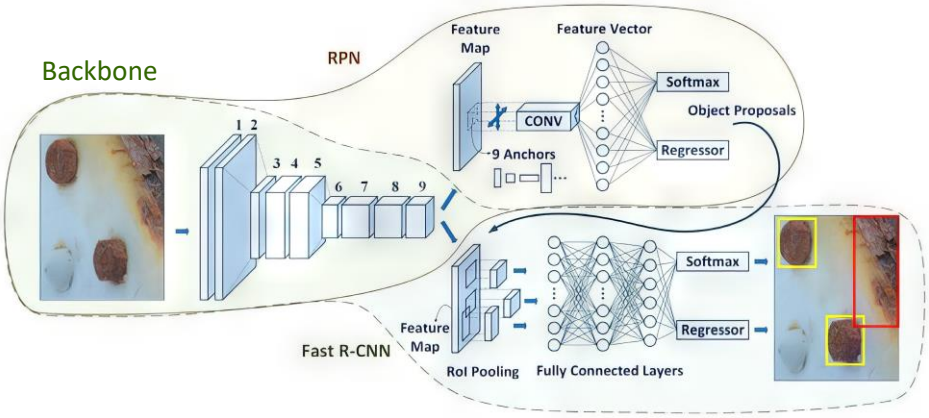
6.2.2 Αναγνώριση Αντικειμένων

Μέχρι στιγμής έχει περιγραφεί το μοντέλο ενός ταξινομητή εικόνας, όμως κατά αυτό τον τρόπο δεν λαμβάνεται πληροφορία για το αν υπάρχουν αντικείμενα διαφορετικών κλάσεων στην εικόνα και το που βρίσκονται αυτά. Για τον λόγο αυτό εξελίχθηκαν μοντέλα αναγνώρισης αντικειμένων (object detector) που καταστρώνουν πλαίσια οριοθέτησης (bounding box) σε κάθε αντικείμενο που βρίσκεται στην εικόνα, όπως φαίνεται στην Εικόνα 6-12.



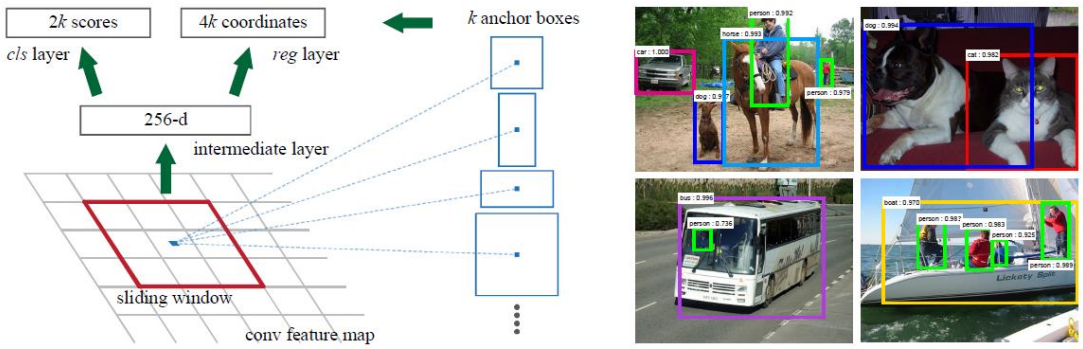
Εικόνα 6-12: Παράδειγμα ταξινόμησης αντικειμένων σε πλαίσια οριοθέτησης.

Στην εργασία αυτή θα γίνει αναφορά στο Faster RCNN μιας και το χρησιμοποιούμενο δίκτυο (Mask RCNN) είναι μια εξέλιξη αυτού. Στην Εικόνα 6-13 φαίνεται η σχηματική αναπαράσταση του δικτύου, όπου υπάρχει ακόμη το δίκτυο εξαγωγής χαρακτηριστικών (backbone) άλλα έχουν προστεθεί κάποια επιπλέον δίκτυα.



Εικόνα 6-13: Σχηματική αναπαράσταση του Faster RCNN.

Η κυριότερη προσθήκη είναι το δίκτυο πρότασης περιοχών (Region Proposal Network - RPN), το οποίο χρησιμοποιώντας 2 «μικρά» δίκτυα: το δίκτυο σύγκλισης περιοχής (Bounding Box Regressor - reg) και ο ταξινομητής (Clasiffier - cls) όπως φαίνονται στην Εικόνα 6-14. Πρακτικά το RPN δείχνει στο κύριο δίκτυο, σε πιες περιοχές να εστιάσει [2]. Επομένως παράγονται περιοχές εστίασης (Region Proposals) οι οποίες περνάνε από μια τεχνική παραγωγής χαρακτηριστικών σταθερού μεγέθους (RoI Pooling) για να μπορεί να τα λάβει η «κεφαλή» του Faster RCNN, δηλαδή να περάσει από το σύστημα νευρώνων. Το σύστημα νευρώνων παράγει στην έξοδο του N·5 εξόδους, όπου N είναι το πλίθος των αντικειμένων που βρέθηκαν στην εικόνα. Οι N έξοδοι έχουν να κάνουν με την κατηγοριοποίηση της περιοχής (classification), δηλαδή πιο αντικείμενο βρίσκεται στην εκάστοτε περιοχή. Ενώ οι άλλοι 4·N έξοδοι έχουν να κάνουν με την μεταβολή του μεγέθους της περιοχής, ώστε το εξαγόμενο πλαίσιο οριοθέτησης να εφαρμόζεται καλύτερα στα όρια του αντικειμένου.



Εικόνα 6-14: Σχηματική αναπαράσταση του RPN [2].

Οι παρακάτω εξισώσεις δείχνουν τις αντίστοιχες συναρτήσεις σφάλματος που λαμβάνονται για την εκπαίδευση του δικτύου.

$$L_{reg} = \sum_i P_i \cdot l_i$$

$$\text{With: } l_i = \begin{cases} \Sigma_j^4 \frac{0.5(y_j - \hat{y}_j)^2}{\beta} \; if \; |y_j - \hat{y}_j| < \beta \\ |y_j - \hat{y}_j| - 0.5 \cdot \beta \end{cases} \qquad 6\text{-}8$$

Όπου
Ομαλή συνάρτηση σφάλματος (Smooth L1 Loss): L
Τιμή πλαισίου (0 or 1) : P_i
Υπολογιζόμενες τιμές παραμέτρων πλαισίου (συντεταγμένες, πλάτος και ύψος) : $\hat{y}_j$
Πραγματικές τιμές παραμέτρων: y_j

$$L_{cls} = \sum_{i=1} y_i \cdot \log \hat{y}_i + (1 - y_i) \cdot \log (1 - \hat{y}_i) \qquad 6\text{-}9$$

Όπου
Δυαδική διασταυρούμενη εντροπία (Binary Cross-Entropy Loss): L
Πραγματικής τάξεις (0 or 1) για την περιοχή i: y_i
Υπολογιζόμενη πιθανότητα της περιοχής i από το cls: $\hat{y}_i$

Το συνολικό σφάλμα του RPN είναι ίσο με:

$$RPN\ Loss = \frac{L_{cls}}{N_{cls}} + \lambda \cdot \frac{L_{reg}}{N_{reg}} \qquad 6\text{-}10$$

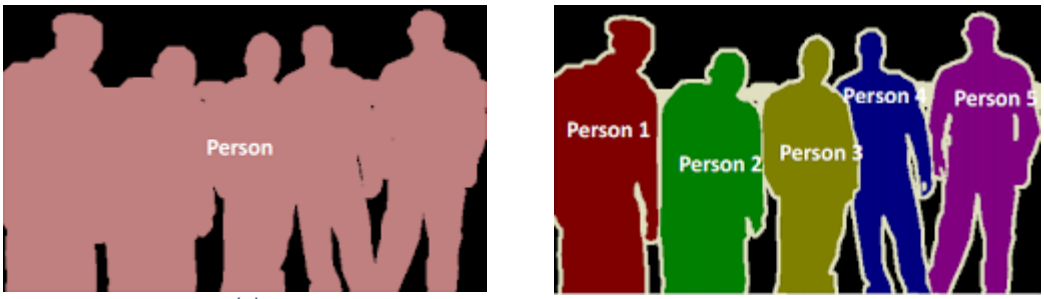
Όπου
Συντελεστής κανονικοποίησης: $\lambda = 10$
Mini batch size: $N_{cls} = 128$
Σημεία «σκαναρίσματος» των αρχικών κουτιών: $N_{reg} = 1250$

6.2.1 Μάσκες Αντικειμένων & Μέθοδοι Ταξινόμησης

Παρόλο που τα πλαίσια οριοθέτησης σε μια εικόνα δίνουν μια ιδέα για το πού βρίσκονται τα αντικείμενα, μια πιο ακριβής προσέγγιση χρειάζεται. Μια τέτοια προσέγγιση μπορεί να γίνει προσθέτοντας μια μάσκα (mask) που να δείχνει την περιοχή που βρίσκεται το κάθε αντικείμενο. Η μάσκα είναι μια δυαδική (με 0 ή 1)

εικόνα όπου στα σημεία που υπάρχουν τιμές ίσες με 1 σημαίνει ότι στο εικονοστοιχείο αυτό, υπάρχει ένα αντικείμενο.

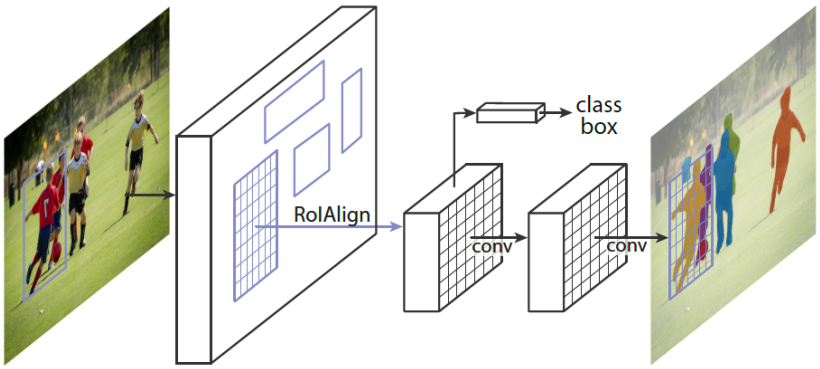
Γενικώς υπάρχουν δυο είδη μάσκας, οι σχηματικές μάσκες (Schematic Segmentation) και οι μάσκες ανά αντικείμενο (instance segmentation). Στην Εικόνα 6-15 φαίνονται οι διαφορές μεταξύ των δυο αυτών μασκών, όπου στην διπλωματική αυτή χρησιμοποιείται η μάσκα ανά αντικείμενο.



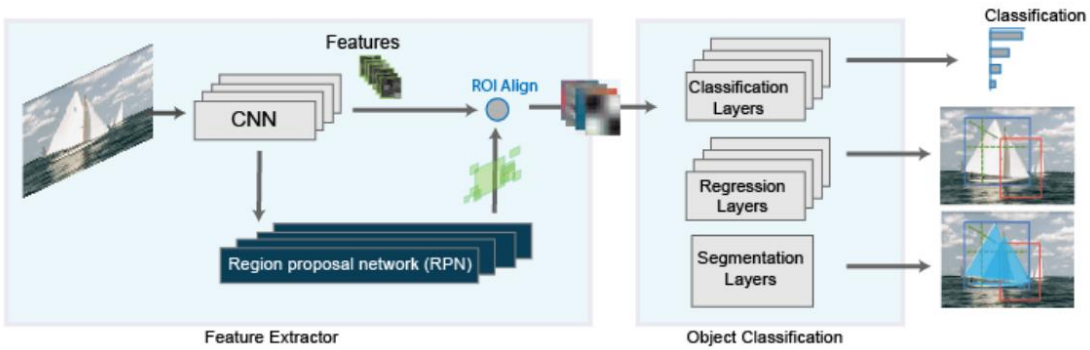
(a) *(b)*

Εικόνα 6-15: Σχηματική μάσκα (a) Και μάσα ανά αντικείμενο (b) [20].

Όπως αναφέρθηκε και προηγουμένως το Mask RCNN είναι μια επέκταση του Faster RCNN. Πιο συγκεκριμένα το Mask RCNN μπορεί και καταστρώνει τις μάσκες για κάθε περιοχή εστίασης που παράγεται από το RPN. Οι μάσκες κατασκευάζονται προσθέτοντας ένα επιπλέον συνελικτικό δίκτυο όπως φαίνεται στην Εικόνα 6-16. Μία απλοποιημένη σχηματική αναπαράσταση του Mask RCNN φαίνεται στην Εικόνα 6-17.



Εικόνα 6-16: Mask R-CNN δίκτυο μάσκας [3].

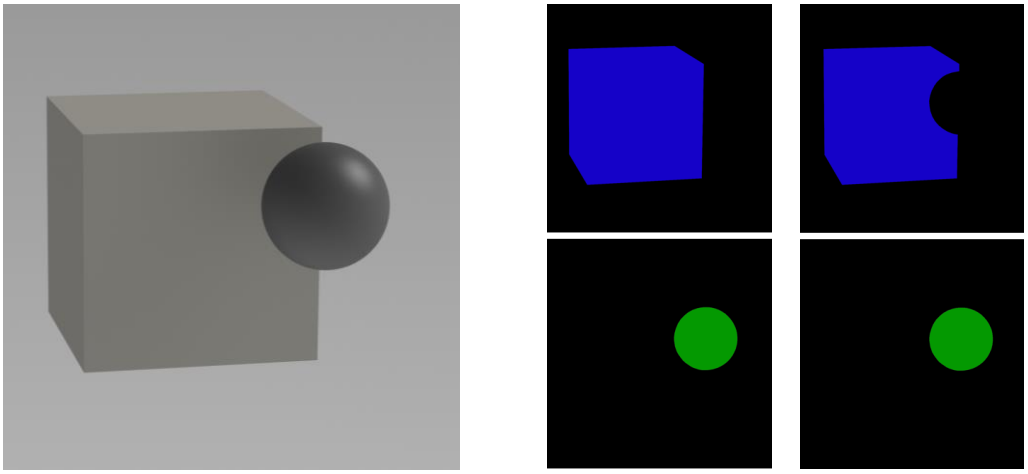


Εικόνα 6-17: Σχηματική αναπαράσταση του Mask R-CNN [20].

Το σφάλμα της μάσκας υπολογίζεται με την Εξίσωση 6-9, ενώ το άθροισμα όλων των σφαλμάτων (Box regression- Classification, RPN and Mask loss) ισούται με το συνολικό σφάλμα του δικτύου.

6.3 Δημιουργία Δικτύου Mask RCNN

Σε αυτή την εργασία το Mask R-CNN προ-εκπαιδευμένο με τα δεδομένα από το MS COCO χρησημοποιείθηκε για το κομμάτι της μηχανική όρασης [20]. Η διατήρηση του προβλήματος της αλληλοεπικάλυψης των αντικειμένων είναι πολύ σημαντικό στην περίπτωση αρπαγής αντικειμένων. Για τον λόγο αυτό μερικοί ερευνητές χρησιμοποιούνε τις πλήρες μάσκες [14] ενώ άλλη χρησιμοποιούν μόνο τις εμφανές μάσκες των αντικειμένων [8], με την διαφορά αυτών να φαίνεται στην Εικόνα 6-18. Ο λόγος επιλογής της μίας έναντι της άλλης μεθόδου έχει να κάνει με το πως ο κάθε ερευνητής επιλέγει να λύσει -σε επόμενο στάδιο- το πρόβλημα της αλληλοεπικάλυψης. Σε αυτή την διπλωματική επιλέχθηκε να χρησιμοποιηθούν μόνο οι εμφανές μάσκες.



(a) (b) (c)

Εικόνα 6-18: Παράδειγμα (a) πλήρης μάσκες (b) και μη επικαλυπτόμενες μάσκες (c).

6.3.1 Ταξινόμηση Εικόνων

Τα αντικείμενα προς αναγνώριση είναι αυτά που φαίνονται στην Εικόνα 6-19.

(a) (b) (c)

Εικόνα 6-19: Αντικείμενα που χρησιμοποιήθηκαν σε αυτή την εργασία. "PT1" (a), "PT2" (b), "PT3" (c).

Λόγο του πρωτότυπου σχήματος των αντικειμένων, έπρεπε να δημιουργηθεί ένα σύνολο δεδομένων για να εκπαιδευτεί το δίκτυο Mask RCNN. Για τον λόγο αυτό 114 φωτογραφίες λήφθηκαν μέσω του Kinect διαστάσεων [1273,1038]. Η Εικόνα 6-20 δείχνει τον τρόπο με τον οποί λήφθηκαν οι εικόνες.

Εικόνα 6-20: Λήψη εικόνων.

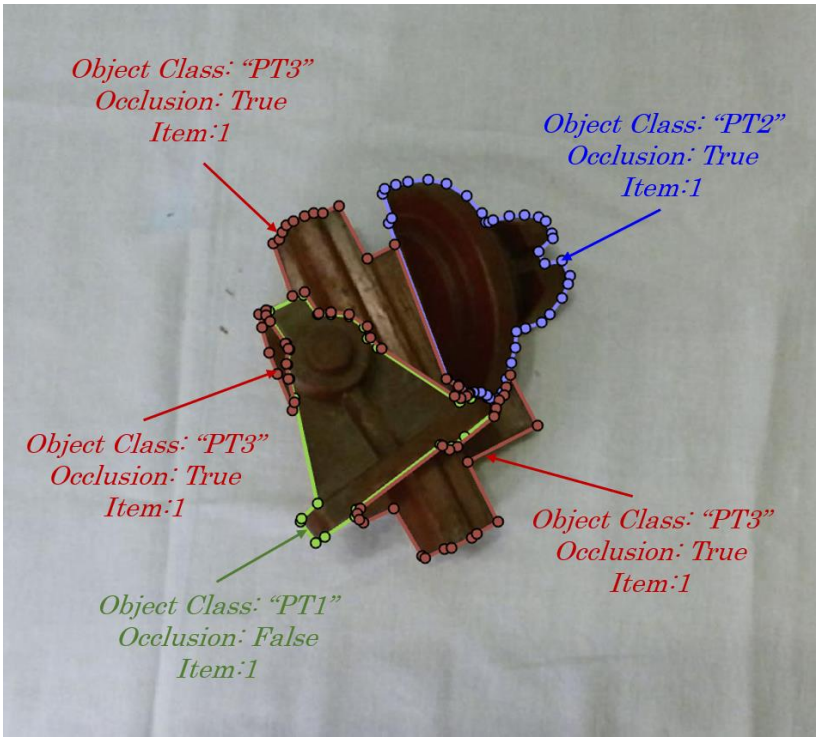
Χρησιμοποιώντας “Image Labeler” εντός του MATLAB, μάσκες από πολύγωνα καταστρώθηκαν για την κάθε κλάση αντικειμένου, με την κάθε μάσκα να έχει δύο επιπλέων μεταβλητές:

- **Αντικείμενο - Item (Ακέραιος)**

  Αυτή η μεταβλητή δείχνει για πιο αντικείμενο αναφέρεται η μάσκα εντός μιας συγκεκριμένης κλάσης όπως φαίνεται στην Εικόνα 6-21, όπου το αντικείμενο “PT3” είναι μοναδικό άλλα έχει 3 διαφορετικές μάσκες.

- **Αλληλοεπικάλυψη - Occlusion (Δυαδικός)**

  Αυτή η μεταβλητή δείχνει αν το αντικείμενο καλύπτεται από κάποιο άλλο, αν καλύπτεται τότε λαμβάνει την τιμή 1, αν όχι τότε την τιμή 0.



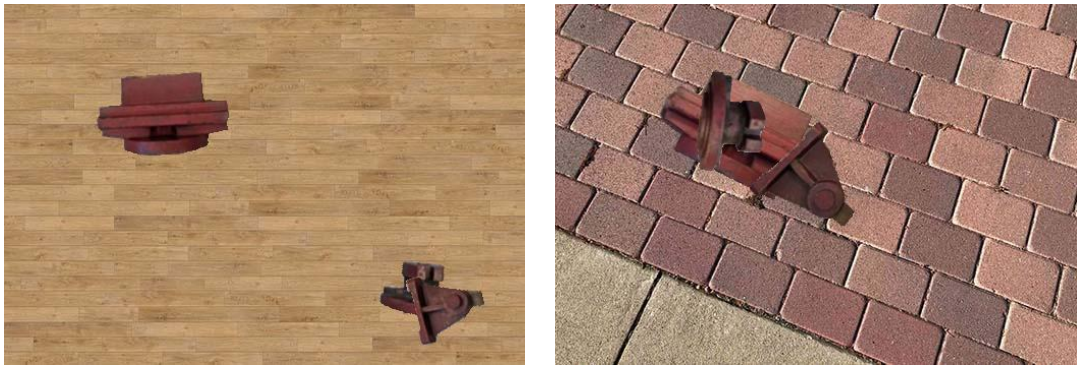
Εικόνα 6-21: Παράδειγμα δημιουργίας μασκών με τις παραμέτρους τους.

Η Εικόνα 6-22 δείχνει μερικά παραδείγματα δημιουργίας των μασκών των αντικειμένων για την εκπαίδευση. Σημειώνεται ότι οι μάσκες καταστρώθηκαν χειροκίνητα και περίπου 2 λεπτά χρειάστηκαν για την δημιουργία τους σε κάθε εικόνα.

Εικόνα 6-22: Παραγόμενες εικόνες για την εκπαίδευση.

6.3.2 Data augmentation

Για την αύξηση των δεδομένων που θα εκπαιδεύσουν το Mask RCNN καταστρώθηκε μια διαδικασία επεξεργασίας των εικόνων. Πιο συγκεκριμένα τα αντικείμενα σε κάθε εικόνα μεταφέρθηκαν και περιστράφικαν με τυχαίο τρόπο τοποθετώντας και ένα διαφορετικό παρασκήνιο με μερικά παραδείγματα να φαίνονται στην Εικόνα 6-23. Έτσι εκπαιδεύοντας το δίκτυο με αυτές τις εικόνες επιτυγχάνεται η σύγκλιση του σε ένα πιο γενικό σύνολο, το οποίο αυξάνει και την ικανότητα εύρεσης αντικειμένων.



Εικόνα 6-23: Επεξεργασμένες εικόνες.

6.3.3 Εκμάθηση

Η εκπαίδευση του δικτύου χωρίστηκε σε δύο στάδια: Στο αρχικό στάδιο χρησιμοποιήθηκαν οι επεξεργασμένες εικόνες (1140 εικόνες), ενώ στο επόμενο στάδιο χρησιμοποιήθηκαν οι πραγματικές εικόνες (114 εικόνες). Η εκπαίδευση έγινε με την χρήση της κάρτας γραφικών (RTX 3090) ενώ στον Πίνακας 6-4 και τον

Πίνακας 6-5 δίνονται οι παράμετροι εκμάθησης.

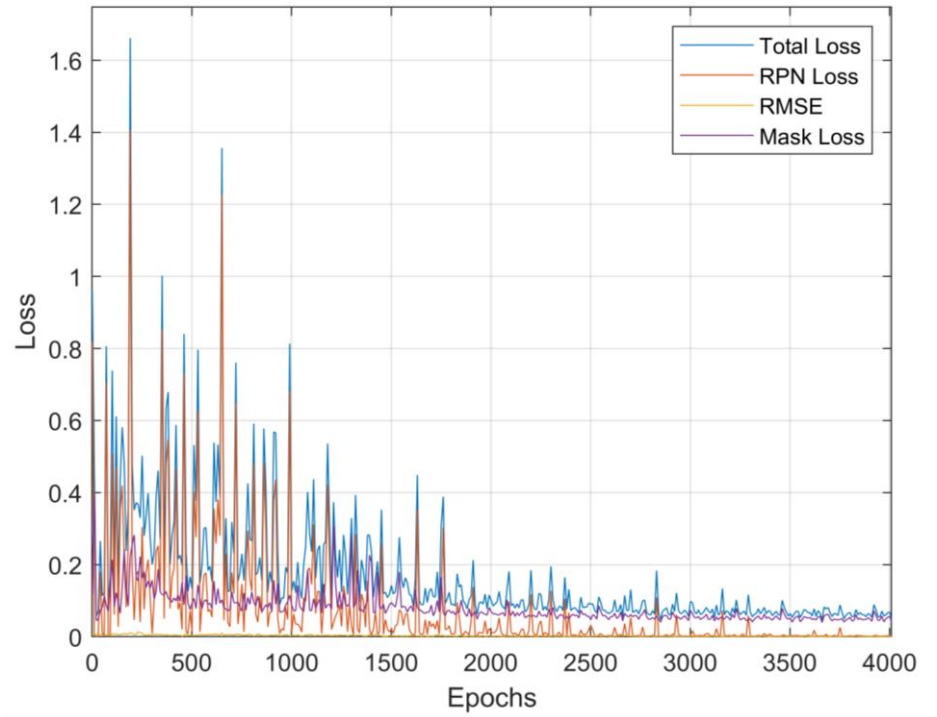
Πίνακας 6-4: Παράμετροι εκμάθησης.

Παράμετροι	**Τιμές**
Training Algorithm	Stochastic Gradient Decent with Momentum
Initial Learn Rate	0.001
Momentum	0.9
Learn Rate Schedule	Piecewise
Learn Rate Drop Period	1
Lear Rate Drop Factor	0.95
Max Epochs	10
Mini Batch Size	2
Bach Normalization Statistics	Moving

Πίνακας 6-5: Παράμετροι του Mask R-CNN.

Παράμετροι	**Τιμές**
Number of Regions to Sample	128
Number of Strongest Regions	1300
Positive Overlap Range	[0.75, 1]
Negative Overlap Range	[0, 0.75]
Number of Anchor Boxes	15

Σημειώνεται ότι το MATLAB και άλλοι ερευνητές[8] επεξηγούν σε μεγάλο βάθος πως οι παράμετροι αυτοί επιδρούν στο δικτύου. Ο Πίνακας 6-6 δείχνει τα σφάλματα κατά το τελευταίο πέρασμα των εικόνων, όπου το σφάλμα του RPN υπολογίζεται σύμφωνα με την Εξίσωση 6-10, το RMSE είναι το μέσω τετραγωνικό σφάλμα για το δίκτυο σύγκλισης πλαισίων οριοθέτησης (Bounding Box Regressor) της κεφαλής, το σφάλμα της μάσκας δίνεται στην Εξίσωση 6-9 (binary cross entropy) και τέλος το συνολικό σφάλμα ισούται με το άθροισμα αυτών των τριών σφαλμάτων. Η Εικόνα 6-24 δείχνει το διάγραμμα των σφαλμάτων κατά την εκμάθηση.



Εικόνα 6-24: Διάγραμμα εκπαίδευσης Mask RCNN .

Πίνακας 6-6: Απόδοση δικτύου Mask RCNN στα δεδομένα της εκπαίδευσης.

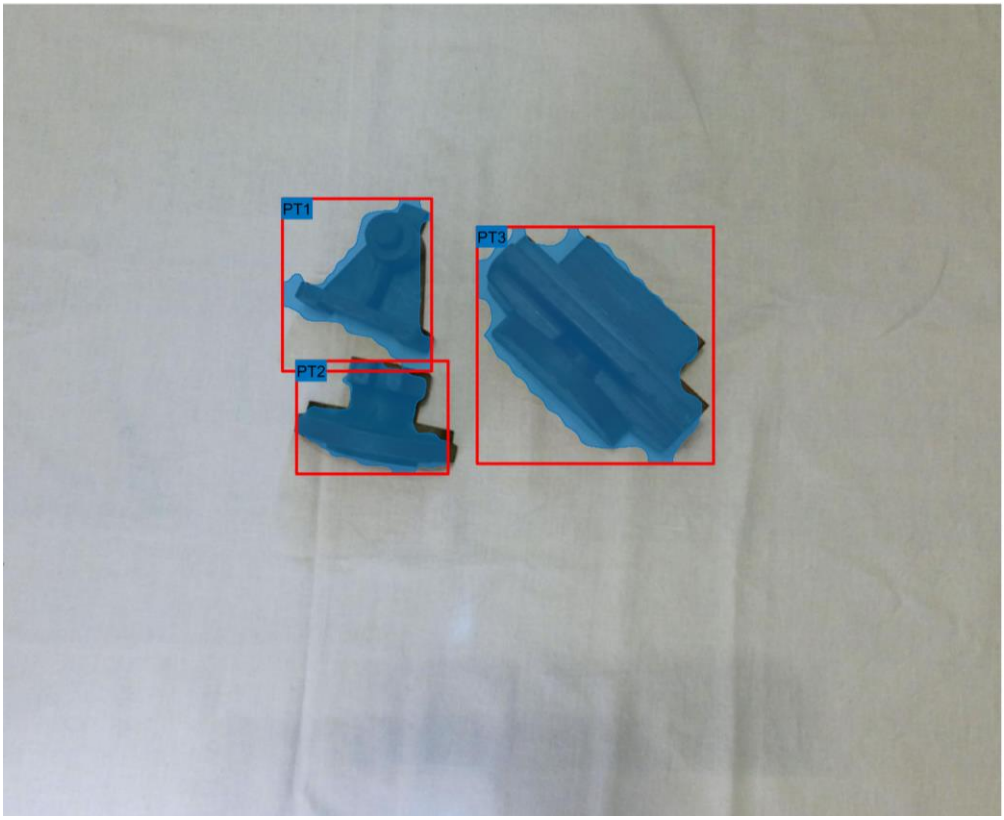
RPN Loss	RMSE	Mask Loss	Total Loss
0.0016	0.0004	0.0153	0.0223

6.3.4 Αποτελέσματα Δοκιμών

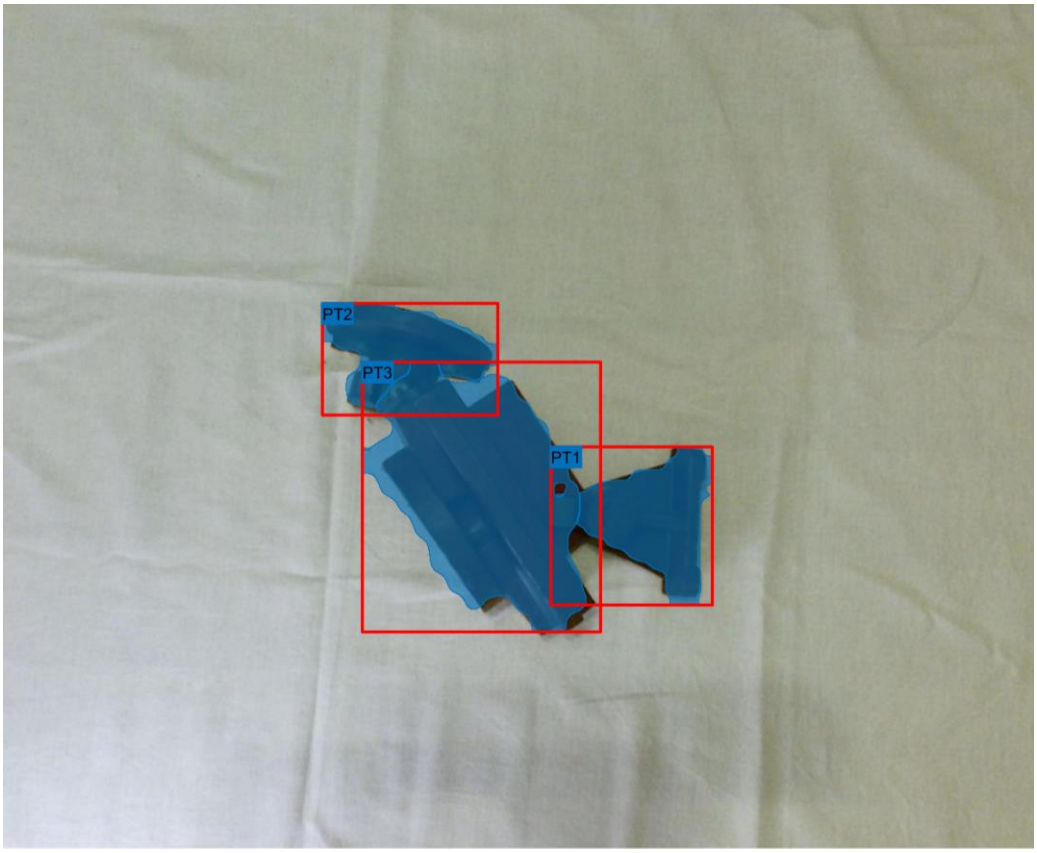
Για την δοκιμή των ικανοτήτων του δικτύου χρησιμοποιήθηκαν 15 εικόνες από 3 διαφορετικές κλάσεις αλληλοεπικάλυψης. Στην πρώτη κλάση δεν υπάρχει αλληλοεπικάλυψη, στην δεύτερη υπάρχει μέτρια αλληλοεπικάλυψη (<25% της επιφάνειας), ενώ στην τελευταία υπάρχει πολύ αλληλοεπικάλυψη (>25 της επιφάνειας). Ο Πίνακας 6-7 Δείχνει τα αποτελέσματα των δοκιμών και οι Εικόνα 6-25, Εικόνα 6-26 και Εικόνα 6-27 δείχνουν ένα παράδειγμα από την κάθε κλάση.

Πίνακας 6-7: Απόδοση του Mask RCNN στα δεδομένα δοκιμών.

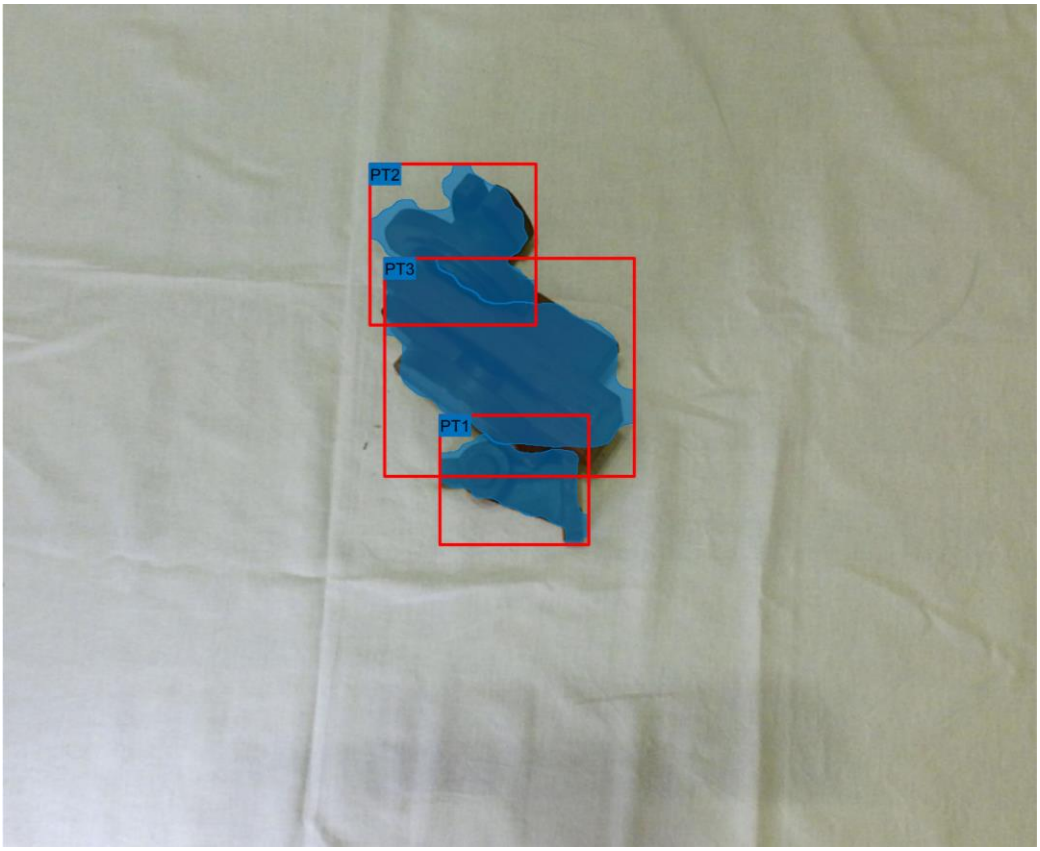
Case	RPN Loss	RMSE	Mask Loss	Total Loss
No Occlusion	0.0022	0.0005	0.0179	0.0258
Mild Occlusion	0.0031	0.0005	0.0251	0.0352
High Occlusion	0.0032	0.0009	0.0381	0.0520



Εικόνα 6-25: Δοκιμή χωρίς καθόλου αλληλοεπικάλυψη.



Εικόνα 6-26: Δοκιμή όταν υπάρχει μέτρια αλληλοεπικάλυψη.



Εικόνα 6-27: Δοκιμή όταν υπάρχει πολύ αλληλοεπικάλυψη.

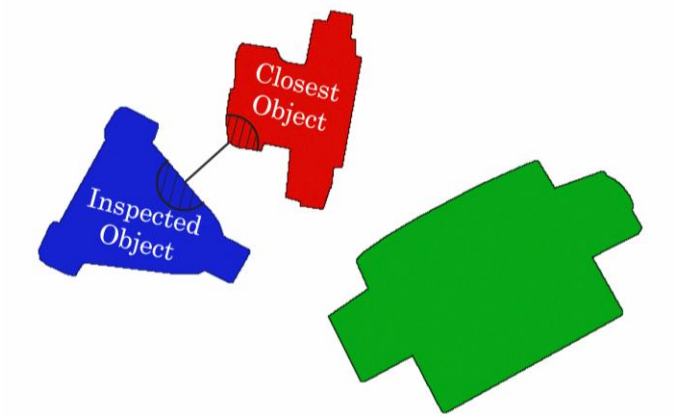
Παρατηρείται ότι όσο αυξάνεται η αλληλοεπικάλυψη τόσο αυξάνεται και το σφάλμα το οποίο ήταν και αναμενόμενο. Παρόλα αυτά τα αποτελέσματα τα οποία επιτυγχάνονται είναι αρκετά καλά συγκρίνοντας με άλλες περιπτώσεις της βιβλιογραφίας [8] (συνολικό σφάλμα 0.08). Πρέπει να αναφερθεί όμως ότι στο [8] υπήρχαν περισσότερες κλάσεις αντικειμένων και περισσότερα αντικείμενα ανά εικόνα το οποίο αυξάνει το συνολικό σφάλμα. Παρόλα αυτά το δίκτυο που αναπτύχθηκε φαίνεται να είναι ικανό να διαχειριστεί ακόμα και τις πιο δύσκολες περιπτώσεις.

7. Νευρωνικά Δίκτυα Αλληλοεπικάλυψης

7.1 Παράμετροι

Σε αυτή την διπλωματική χρησιμοποιήθηκαν νευρωνικά δίκτυα (ένα για κάθε διαφορετική κλάση αντικειμένου) ώστε να βρεθεί αν τα αντικείμενα επικαλύπτονται ή αν δεν επικαλύπτονται. Δηλαδή τα δίκτυα αυτά έχουν μια δυαδική έξοδο (1 αν επικαλύπτεται και 0 αν δεν επικαλύπτεται) και έξι εισόδους που περιγράφονται παρακάτω:

- **Ύψος Παρασκηνίου - Background Height**
  Το ύψος του παρασκηνίου βρίσκεται προσθέτοντας όλες τις μάσκες των αντικειμένων (με την πράξη and), αντιστρέφοντας την μάσκα (δηλαδή όπου υπάρχουν 1, τοποθετούνται 0 και το αντίθετο) και λαμβάνοντας τον μέσο όρο του ύψους στα αληθές εικονοστοιχεία της μάσκας. Άρα λαμβάνεται ο μέσος όρος του ύψους του παρασκηνίου.
- **Ανοιγμένη Περίμετρος - Opened Perimeter**
  Είναι η περίμετρος (σε εικονοστοιχεία) της εκάστοτε μάσκας διαιρεμένη με το ύψος του παρασκηνίου.
- **Ανοιγμένη Επιφάνεια - Opened Area**
  Είναι η επιφάνεια (σε εικονοστοιχεία) διαιρεμένη με το τετράγωνο του ύψους παρασκηνίου.
- **Ύψος Αντικειμένου - Object Height**
  Είναι το μέσο ύψος του αντικειμένου σε σχέση με το ύψος του παρασκηνίου.
- **Distance Between Closest Object**
  Είναι η μικρότερη απόσταση μεταξύ του αντικειμένου προς επιθεώρηση με οποιοδήποτε άλλο αντικείμενο στην εικόνα όπως απεικονίζει και η Εικόνα 7-1.
- **Height Difference Between Closest Object**
  Είναι η διαφορά ύψους μεταξύ του αντικειμένου προς επιθεώρηση και του κοντινότερου αντικειμένου. Για την αποφυγή τοπικών σφαλμάτων στα δεδομένα βάθους λαμβάνεται ο μέσος όρος του ύψους σε μια περιοχή κοντά στα κοντινότερα σημεία όπως φαίνεται στην Εικόνα 7-1.

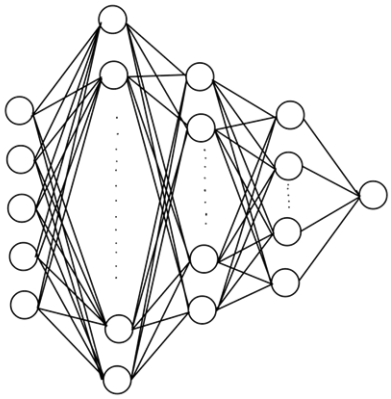


Εικόνα 7-1: Σχηματική αναπαράσταση των παραμέτρων εκπαίδευσης.

7.2 Διαδικασία Εκμάθησης

Για την εκμάθηση των νευρωνικών δικτύων χρησιμοποιήθηκαν οι 114 φωτογραφίες που πάρθηκαν κατά την κατάστρωση των δεδομένων για την εκμάθηση του Mask RCNN. Επομένως τα δεδομένα χωρίζονται σε 114 περιπτώσεις για κάθε διαφορετικό νευρωνικό δίκτυο (σε κάθε εικόνα υπήρχαν όλα τα αντικείμενα μέσα). Τα δεδομένα αυτά χωρίζονται σε τρεις επιπλέων υποκατηγορίες εκπαίδευσης / επικύρωσης / δοκιμής (training/validation/testing) με κατανομή 80/10/10 (%) αντίστοιχα.

Η εκμάθηση έγινε μέσω της μεθόδου αδιατάραχτης οπισθοδρόμησης (resilient backpropagation - RP), εφόσον επιτρέπει την εκμάθηση του δικτύου μέσω της κάρτας γραφικών. Η συνάρτηση σφάλματος επιλέχθηκε να είναι αυτή του μέσου τετραγωνικού σφάλματος, η οποία υπολογίστηκε με ισόποση βαρύτητα για την κάθε υποκατηγορία (εκπαίδευσης / επικύρωσης / δοκιμής).



Εικόνα 7-2: Structure of Neural Networks implemented in this work.

Η συνάρτηση ενεργοποίησης στα κρυφά επίπεδα είναι η υπερβολική εφαπτομένη "tansig", ενώ στην έξοδο εφαρμόζεται στρογγυλοποίηση του αποτελέσματος για να ληφθεί 0 ή 1. Στον Πίνακας 7-1 φαίνονται οι παράμετροι εκμάθησης που εφαρμόστηκαν ενώ ο Πίνακας 7-2 δίνει τα αποτελέσματα της εκπαίδευσης.

Πίνακας 7-1: Παράμετροι εκπαίδευσης των νευρωνικών δικτύων.

Maximum Epochs	Maximum Validation Increases	Regularization	Normalization	Training algorithm
1000	20	false	false	Resilient backpropagation

Πίνακας 7-2: Κρυμμένα επίπεδα και σφάλματα των δικτύων.

Κλάση Αντικειμένου	Κρυμμένα επίπεδα	Εκμάθηση	Επικύρωση	Δοκιμή
PT1	[50,50,25,25,10,10,5,5]	2	1	1
PT2	[100,100,50,50,25,25,10,10,5,5]	0	0	2
PT3	[100,100,50,50,25,25,10,10,5,5]	4	1	1

7.3 Διαδικασία Εκπαίδευσης

Τα αποτελέσματα που βρέθηκαν από την εκμάθηση αντιστοιχούν ή ξεπερνάνε αυτά που έχουν βρεθεί από άλλες εργασίες- ερευνητές. Πιο συγκεκριμένα στην εργασία του [14] η απόδοση της μεθόδου που ακολουθήθηκε ήταν ίση με 68.3%, ενώ σε αυτή την διπλωματική η απόδοση των δικτύων ήταν ίση με 90.5%. Είναι σημαντικό όμως να επισημανθεί η διαφορά μεταξύ των δεδομένων εκμάθησης και δοκιμής όπου στο [14] παραπάνω αντικείμενα και κλάσεις αντικειμένων χρησιμοποιήθηκαν. Επίσης οι δοκιμές έγιναν με επισημασμένες εικόνες (ground truth data) έναντι εικόνων που έχουν εξαχθεί από το Mask RCNN, το οποίο θα γίνει στην πρακτική εφαρμογή. Ο Πίνακας 7-3 δείχνει τους τύπους των σφαλμάτων που βρέθηκαν κατά την εκμάθηση και δοκιμή των δικτύων.

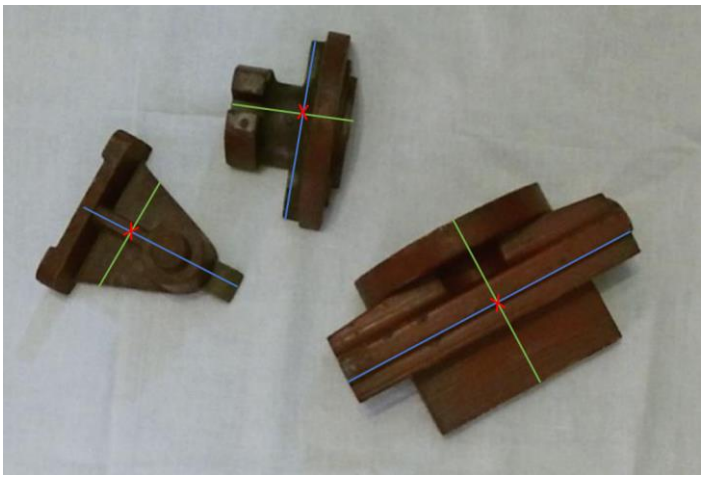
Πίνακας 7-3: Πίνακας αληθείας των νευρωνικών δικτύων.

Κλάση	Λανθασμένα Θετικό	Λανθασμένα Αρνητικό
PT1	1	3
PT2	0	2
PT3	1	5

Αν και τα αποτελέσματα των δοκιμών υποδεικνύουν μια "καλή" απόδοση, είναι σημαντικό να αναγνωριστεί ότι τα δίκτυα ενδέχεται να υποτιμούν την πιθανότητα επικάλυψης των αντικειμένων. Η διαφορά που παρατηρείται στα ψευδώς θετικά και ψευδώς αρνητικά αποτελέσματα μπορεί να αποδοθεί στη φύση των εισόδων που παρέχονται στα δίκτυα. Πιο συγκεκριμένα, σε περιπτώσεις όπου υπάρχει υψηλή αλληλοεπικάλυψη μεταξύ των αντικειμένων, οι παράμετροι "Κοντινότερη απόσταση αντικειμένου" και "Διαφορά ύψους μεταξύ πλησιέστερων αντικειμένων" ενδέχεται να μην δίνουν τα δεδομένα του αντικειμένου που επικαλύπτει το υπό εξέταση αντικείμενο. Συνεπώς, τα δεδομένα αυτά αναφέρονται σε ένα άλλο αντικείμενο το οποίο μπορεί να επικαλύπτεται το οποίο μπορεί να έχει και χαμηλότερο ύψος από το εξεταζόμενο, οδηγώντας στο εσφαλμένο συμπέρασμα ότι το επιθεωρημένο αντικείμενο δεν επικαλύπτεται. Σε τέτοιες περιπτώσεις, εάν τα δίκτυα αποτυγχάνουν να αναγνωρίσουν τις σωστές ιδιότητες επικάλυψης ενός αντικειμένου, η λογική σύλληψης μπορεί να οδηγήσει στην προσπάθεια σύλληψης του επικαλυμμένου αντικειμένου, το οποίο θέτει σε κίνδυνο σύγκρουσης του βραχίονα με άλλα αντικείμενα. Ωστόσο, η πιθανότητα του τελευταίου αποτρέπεται διότι στην λογική σύλληψης επιλέγεται πάντα το υψηλότερο αντικείμενο (που δεν επικαλύπτεται) να αρπαχθεί.

8. Μέθοδος Αρπαγής των Τεμαχίων

Σε αυτή την εργασία επιλέχθηκε να χρησιμοποιηθεί μια απλή και αποτελεσματική μέθοδος αρπαγής- σύλληψης των αντικειμένων η οποία βασίζεται στην γωνία στροφής των αντικειμένων σύμφωνα με το επίπεδο του παρασκηνίου. Πιο συγκεκριμένα οι αρπάγες πιάνουν το αντικείμενο από ένα ευθύγραμμο τμήμα που περνάει από το κέντρο της επιφάνεια όπως φαίνεται στην Εικόνα 8-1.



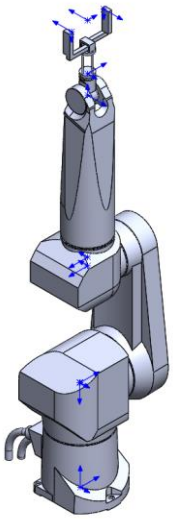
Εικόνα 8-1: Κέντρο επιφάνειας (Κόκκινα σημάδια), Άξονας Αρπαγής (Πράσινες γραμμές).

Επομένως ο προσανατολισμός του τελικού σημείου δράσεις επιλέγεται να είναι κάθετος του παρασκηνίου με κατεύθυνση παράλληλη του ευθύγραμμου τμήματος αρπαγής. Η θέση τοποθέτησης του τελικού σημείου δράσης είναι στο κέντρο της επιφάνειας του εκάστοτε αντικειμένου, με απόσταση από κατά τον Z ίση με 10mm κάτω από το ύψος του αντικειμένου. Σημειώνεται ότι κατά αυτόν τον τρόπο κάθε φορά που αφαιρείται ένα αντικείμενο από την εικόνα, πρέπει να γίνει καινούργια λήψη εικόνας ώστε να επιλεχθεί το επόμενο αντικείμενο. Κατά αυτόν τον τρόπο το σύστημα είναι πιο ικανό στην λήψη αντικειμένων σε περιβάλλοντα με υψηλή αλληλοεπικάλυψη, όμως αυξάνεται η υπολογιστική ισχύς αφού απαιτείται συνεχή λήψη και επεξεργασία εικόνων.

9. Δημιουργία Εικονικού Περιβάλλοντος

Η προσομοίωση στο εικονικό περιβάλλον έγινε με την χρήση της επικοινωνίας του MATLAB/Simulink – Unreal Engine. Τα μοντέλα καταστρώθηκαν εντός του SOLIDOWKRS ενώ η σκηνή κατασκευάστηκε μέσω του Blender. Η σκηνή εξάγεται σε μορφή αρχείου τύπου ".fbx" για να μπορεί να διαβαστεί από το Unreal Engine.

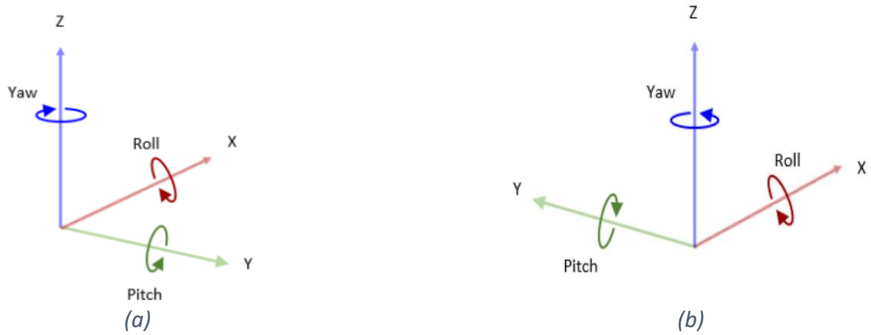
Ο ρομποτικός βραχίονας συναρμολογείτε εντός του SOLIDWORKS όπως φαίνεται στην Εικόνα 9-1. Μέσω του SW urdf exporter εξάγεται το αρχείο τύπου ".urdf" για να χρησιμοποιηθεί εντός του MATLAB και του Unreal Engine.



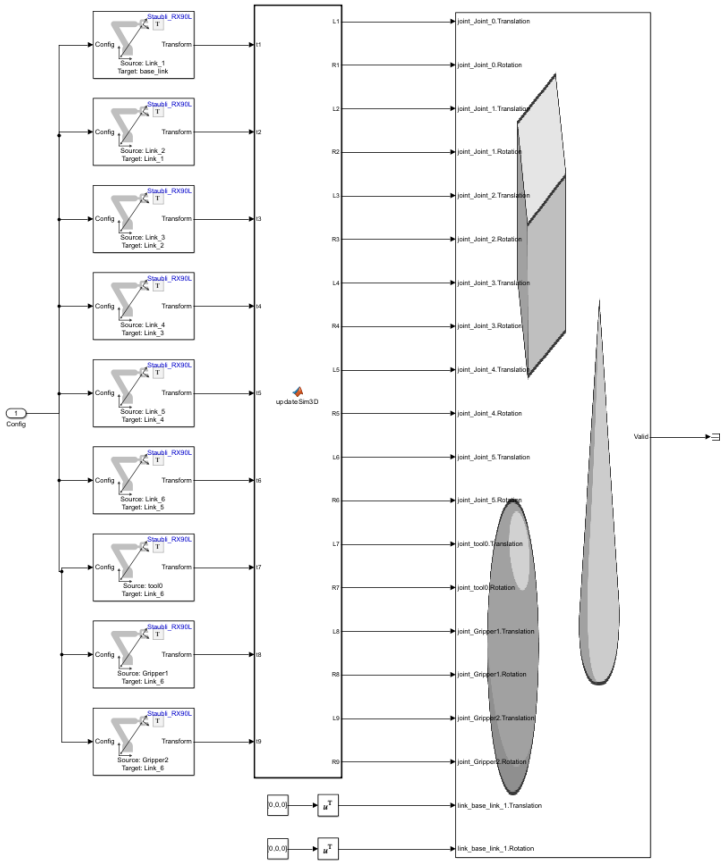
Εικόνα 9-1: Ο συναρμολογημένος βραχίονας Stäubli RX90L στο SOLIDWORKS.

Εντός του Simulink τοποθετείται και η κάμερα με την επιπλέον δυνατότητα εξαγωγής εικόνας βάθους. Τα χαρακτηριστικά της κάμερας είναι ίδια με αυτά που αναγράφονται στο κεφάλαιο της μηχανικής όρασης.

Για να είναι δυνατή η προσημείωση όμως του ρομποτικού βραχίονα άλλα και της κίνησης των αντικειμένων, ήταν απαραίτητος ο μετασχηματισμός των συντεταγμένων μεταξύ του συστήματος συντεταγμένων του MATLAB και του Unreal Engine. Η διαφορά μεταξύ των δυο συστημάτων φαίνεται στην Εικόνα 9-2, ενώ η Εικόνα 9-3 δείχνει το μοντέλο αυτής της μετατροπής εντός του Simulink.



Εικόνα 9-2: Συστήματα συντεταγμένων στο Unreal Engine (a) και στο MATLAB (b).



Εικόνα 9-3: Μετατροπή των συντεταγμένων του MATLAB σε αυτές του Unreal Engineγια τον βραχίονα.

9.1 Προσομοίωση Αρπαγής

Σημαντικό κομμάτι της προσομοίωσης ήταν και η μοντελοποίηση της αρπαγής του αντικειμένου, δηλαδή να φανεί πως κλείνουν οι αρπάγες και πιάνουν το αντικείμενο και πως αυτό μετακινείται μαζί με τον ρομποτικό βραχίονα. Για το τελευταίο η λύση που υλοποιήθηκε είναι η διατήρηση σταθερού ομογενούς μετασχηματισμού μεταξύ αντικειμένου και τελικού σημείου δράσης κατά την μεταφορά του. Αυτό μπορεί να υλοποιηθεί αξιοποιώντας την σχέση της Εξίσωσης *9-1*.

$$ {}^{e}T_{obj} = {}^{e}T_{b} \cdot {}^{b}T_{obj} $$

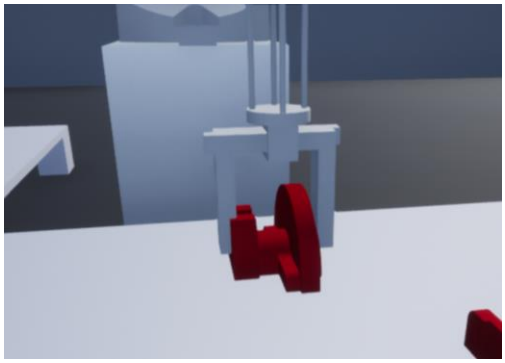
9-1

Όπου

Ο ομογενής μετασχηματισμός της βάσης του βραχίονα σε σχέση με την αρπάγη: ${}^{e}T_{b} = \left({}^{b}T_{e}\right)^{-1}$

Ο ομογενής μετασχηματισμός του αντικειμένου σε σχέση με την βάση : ${}^{b}T_{obj}$

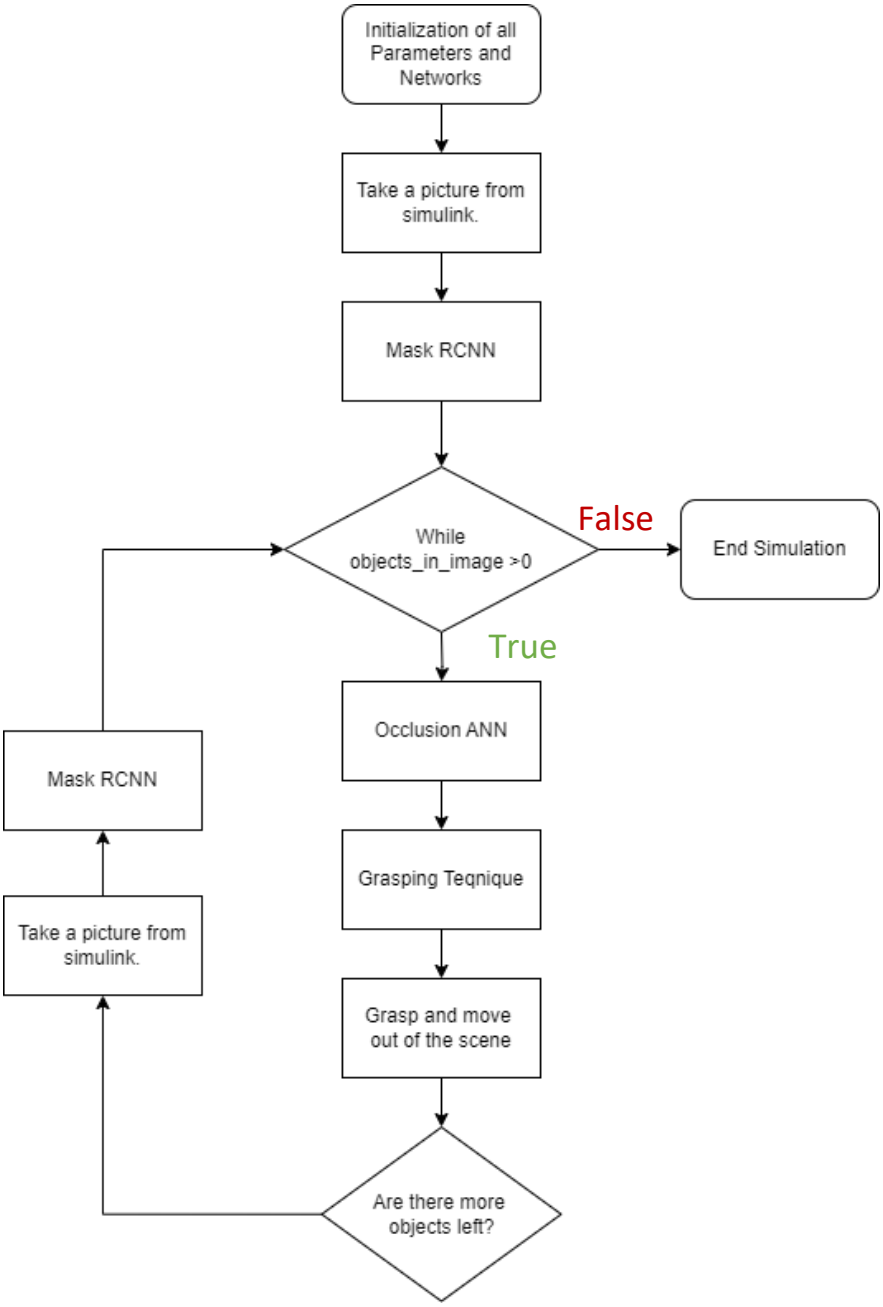
Για την προσομοίωση του κλεισίματος της αρπάγης χρησημοποιείθηκε το συμβάν (Event) τύπου χτυπήματος (Hit) το οποίο υποστηρίζει το Unreal Engine. Πιο συγκεκριμένα ενώ κλείνουν με σταθερό ρυθμό οι αρπάγες, όταν χτυπήσουν το αντικείμενο σταματάνε να κουνιούνται και άρα βρίσκονται στο σημείο αρπαγής. Η Εικόνα 9-4 δείχνει ένα παράδειγμα αυτής της διαδικασίας.



Εικόνα 9-4: Προσομοίωση του κλεισίματος της αρπάγης.

9.2 Προσομοίωση του Πλήρους Μοντέλου

Το τελευταίο στάδιο είναι η προσομοίωση όλων όσος έχουν κατασκευαστεί στην εργασία αυτή σε ένα εικονικό περιβάλλον or the code. Το διάγραμμα ροής φαίνεται στην Εικόνα 9-5.



Εικόνα 9-5: Διάγραμμα ροής κατά την προσομοίωση.

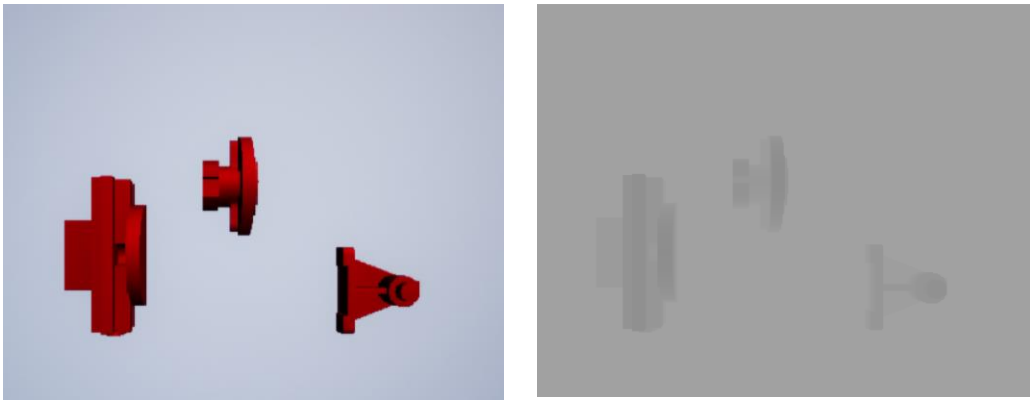
Μια συνοπτική περίληψη του κάθε βήματος δίνεται παρακάτω:

- **Initialization of All Parameters and Networks**

Το αρχικό βήμα περιλαμβάνει τη φόρτωση βασικών παραμέτρων, τον ρομποτικό βραχίονα, τους καταλόγους δεδομένων, το Mask RCNN και τα νευρωνικά δίκτυα. Η ολοκλήρωση αυτού του βήματος απαιτεί περίπου 5 δευτερόλεπτα.

- **Capture Image from Simulink**

Άνοιγμα του μοντέλου Simulink για 0,1 δευτερόλεπτα ώστε να γίνει η λήψη των εικόνων βάθους και RGB. Ωστόσο, αυτό το βήμα εκτείνεται σε περίπου 8 δευτερόλεπτα λόγω της έναρξης του Unreal Engine.

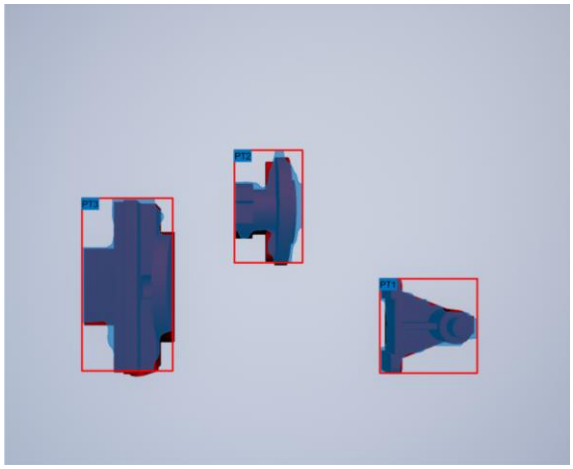


(a) (b)

Εικόνα 9-6: Αρχικές εικόνες και από τους δύο αισθητήρες.

- **Mask RCNN**

Χρησιμοποιώντας την εικόνα RGB που λαμβάνεται από την εικονική κάμερα, το δίκτυο Mask RCNN δημιουργεί ετικέτες, πλαίσια οριοθέτησης και μάσκες για ανιχνευμένα αντικείμενα. Συγκεκριμένα, ένα αναδυόμενο παράθυρο εμφανίζει τα αποτελέσματα.



Εικόνα 9-7: Αναγνώριση αντικειμένων και δημιουργία μασκών από το Mask RCNN.

- **Occlusion ANN**

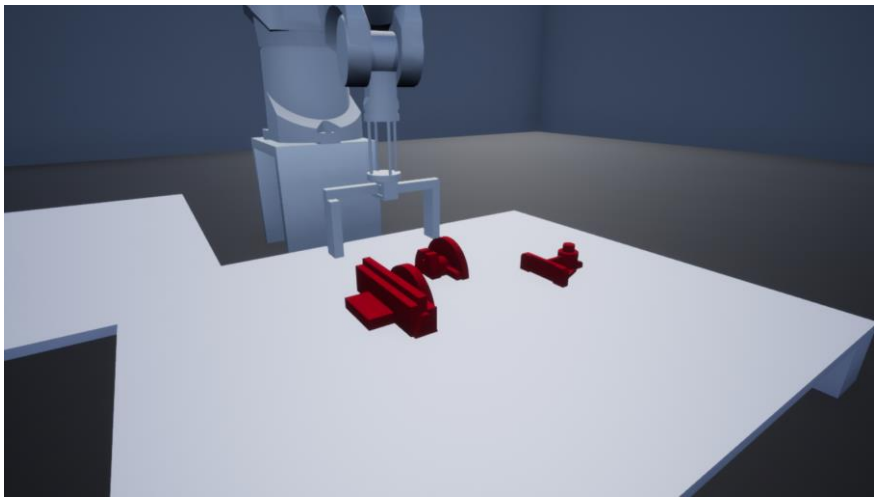
Χρησιμοποιώντας τις μάσκες και τις ετικέτες από το Mask RCNN μαζί με την εικόνα βάθους από το Kinect, δημιουργούνται τα χαρακτηριστικά για κάθε αντικείμενο, και εξάγονται οι δυαδικοί δείκτες επικάλυψης. Αυτό το βήμα συμβαίνει σχεδόν ακαριαία (λιγότερο από 0,5 δευτερόλεπτα).

- **Grasping Technique**

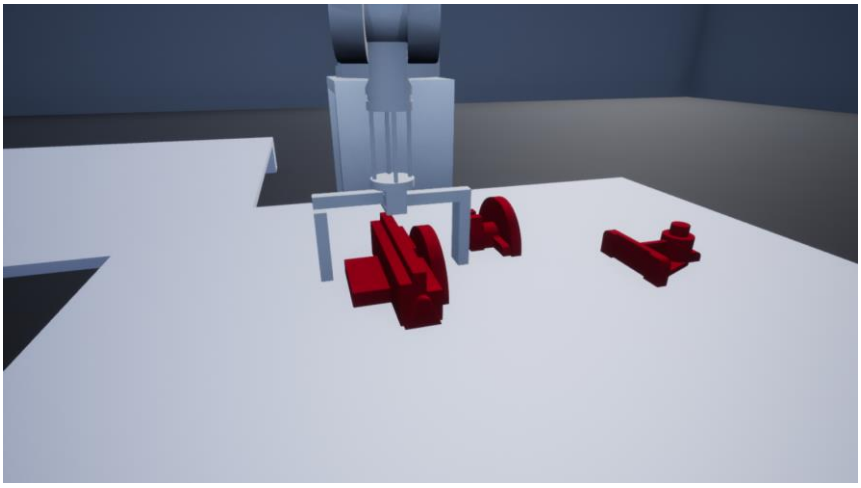
Συνδυάζοντας μάσκες, την εικόνα βάθους και την έξοδο από τα νευρωνικά , υπολογίζεται πιο αντικείμενο θα συλληφθεί και δημιουργείται η τροχιά

- **Grasp and Move Out of the Scene**

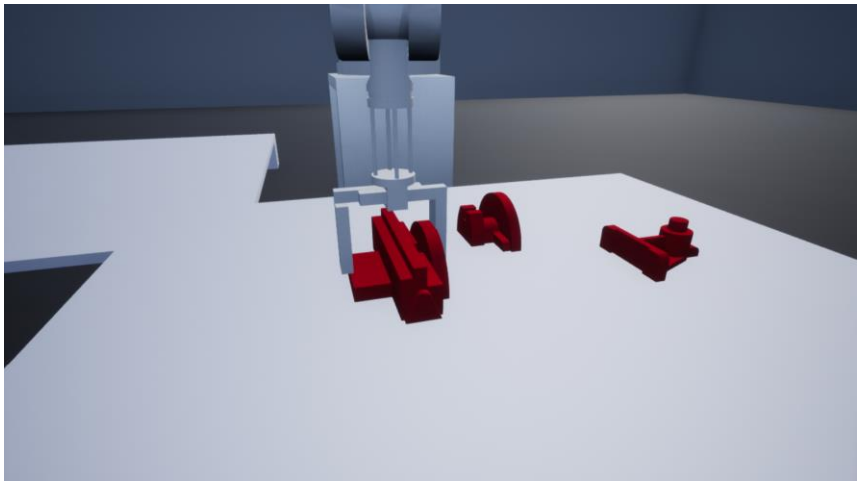
Γίνεται η προσομοίωση της αρπαγής συμπεριλαμβάνοντας με την προσέγγιση του τελικού σημείου δράσης στο τεμάχιο. Στις παρακάτω Εικόνες αποτυπώνεται η διαδικασία για έναν βρόχο της προσομοίωσης όπου επιλέχθηκε το "PT3".



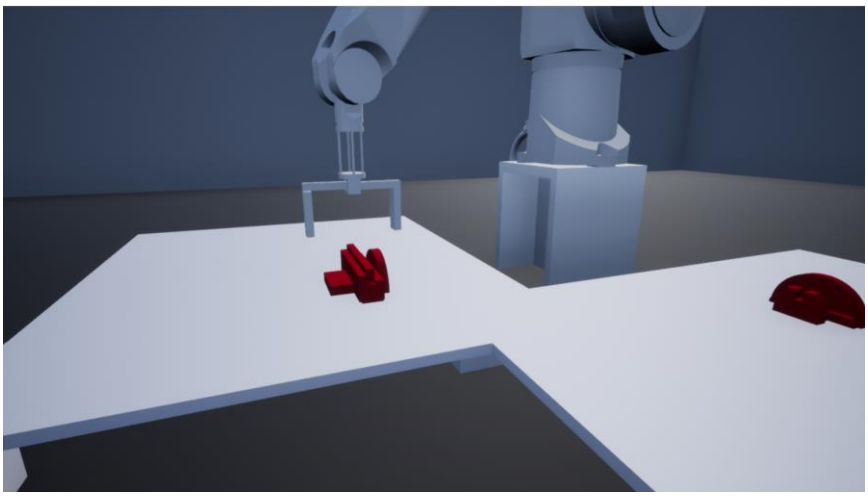
Εικόνα 9-8: Προσέγγιση αντικειμένου.



Εικόνα 9-9: Σημείο αρπαγής.



Εικόνα 9-10: Κλείσιμο αρπάγης.



Εικόνα 9-11: Απελευθέρωση αντικειμένου.

Το βίντεο με την προσομοίωση μπορεί να βρεθεί στο ακόλουθο σύνδεσμο: https://ntuagr-my.sharepoint.com/:v:/g/personal/mc19098_ntua_gr/ET8tc6sprSdJvt_Ezgmybm8BIFP8IXQMy33FBxDUm-ZB4A?nav=eyJyZWZlcnJhbEluZm8iOnsicmVmZXJyYWxBcHAiOiJPbmVEcml2ZUZvckJ1c2luZXNzIiwicmVmZXJyYWxBcHBQbGF0Zm9ybSI6IldlYiIsInJlZmVycmFsTW9kZSI6InZpZXciLCJyZWZlcnJhbFZpZXciOiJNeUZpbGVzTGlua0NvcHkifX0&e=SmhL6m

10. Συζήτηση

Για την διεξαγωγή της εργασίας αυτής απαιτήθηκε εξειδίκευση σε τρεις διαφορετικούς τομείς: ρομποτικά συστήματα, μεθόδους οπτικοποίησης και μηχανική όραση. Ενώ οι θεμελιώδεις γνώσεις σε αυτούς τους τομείς αποκτήθηκαν κατά τη διάρκεια της ακαδημαϊκής μου θητείας στη Σχολή Μηχανολόγων Μηχανικών, η ολοκλήρωση αυτής της διατριβής απαιτούσε πρόσθετες γνώσεις που αποκτήθηκαν κατά την ανάπτυξή της. Είναι σημαντικό να τονιστεί ότι αν και το αναπτυγμένο σύστημα παρουσιάζει συγκριτικά καλές επιδώσεις, υπάρχουν ακόμη περιθώρια βελτίωσης. Συγκεκριμένα, δυσκολίες εμφανίζονται όταν στις εικόνες υπάρχει υψηλή αλληλοεπικάλυψη μεταξύ των αντικειμένων, ένα θέμα που διερευνάται επί του παρόντος από πολλούς ακόμη ερευνητές.

Όπως αναφέρθηκε στη σχετική ενότητα εργασίας, ορισμένοι ερευνητές χρησιμοποίησαν δίκτυα Mask RCNN για να ενσωματώσουν δεδομένα βάθους με εικόνες RGB για να βελτιώσουν τις δυνατότητες κάλυψης. Δυστυχώς, στο MATLAB, αυτή η βαθιά συγχώνευση δεδομένων μέσω του δικτύου δεν είναι δυνατή, με αποτέλεσμα την υποχρησιμοποίηση των δεδομένων. Για να μετριαστεί αυτός ο περιορισμός, υπήρχε ο ιδεασμός να χρησιμοποιηθούν εναλλακτικές υλοποιήσεις σε περιβάλλοντα όπως η Python όπου υπάρχει διαθέσιμος κώδικας για αυτούς τους τύπους δικτύου ή να δημιουργηθεί ένα προσαρμοσμένο δίκτυο Mask RCNN στο MATLAB. Ωστόσο, και οι δύο μέθοδοι απαιτούν σημαντική προσπάθεια υλοποίησης και οι βελτιώσεις απόδοσης είναι αβέβαιες, όπως φαίνεται στην ενότητα δοκιμών δικτύου.

Όσον αφορά τις μεθόδους χειρισμού της επικάλυψης, παρόλο που υπάρχουν άλλες μέθοδοι στα δίκτυα Mask RCNN και περιλαμβάνουν επιπλέον CNN, η μέθοδος που επισημαίνεται σε αυτό το άρθρο ξεχωρίζει για την απλότητα, την αμεσότητα και την ταχύτητά της. Ωστόσο, αυτή η απόδοση συνδυάζεται με την απόδοση του δικτύου Mask RCNN, το οποίο με τη σειρά του μπορεί να οδηγήσει σε κακά αποτελέσματα πρόβλεψης εάν η μάσκα δεν είναι αρκετά ακριβής. Οι πιθανές βελτιώσεις (σε αυτή τη μέθοδο) έγκεινται στην ενσωμάτωση πρόσθετων εισόδων ή στην αλλαγή των υπαρχόντων εισόδων.

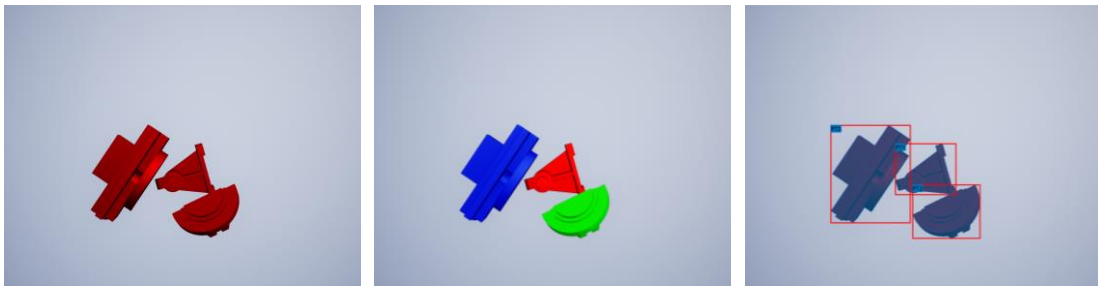
11. Συμπεράσματα

Συνοψίζοντας, αυτή η διπλωματική εργασία παρουσιάζει μια μεθοδολογία για την κατασκευή ενός συστήματος ρομποτικού βραχίονα-μηχανής όρασης ικανού να αντιμετωπίζει προκλήσεις που σχετίζονται με την αλληλοεπικάλυψη. Ολόκληρη η διαδικασία, από τη δημιουργία δεδομένων για εκπαίδευση έως την εφαρμογή μέσα στο περιβάλλον MATLAB-Simulink και το Unreal Engine, αναλύεται λεπτομερώς. Με την χρήση τόσο επεξεργασμένων (augmented) όσο και πραγματικών ταξινομημένων εικόνων, επιτυγχάνεται μια σχετικά υψηλή απόδοση σε σύγκριση με το [8], όπου στο παρουσιαζόμενο μοντέλο επιτυγχάνεται συνολικό σφάλμα 0,025 ενώ στο [8] επιτυγχάνεται συνολικό σφάλμα ίσο με 0,08. Επιπλέον, χρησιμοποιήθηκε μια εντελώς διαφορετική προσέγγιση για την αντιμετώπιση της αλληλοεπικάλυψης, όπου τα χαρακτηριστικά που εξάγονται από τις εικόνες βάθους και τις μάσκες χρησιμοποιούνται ως είσοδοι στα νευρωνικά δίκτυα. Η απόδοση σε δοκιμές έδειξε ένα σημαντικό ποσοστό επιτυχίας της τάξης του 90,5% στον εντοπισμό περιπτώσεων αλληλοεπικάλυψης σε σύγκριση με το 68,3% που βρέθηκε από το [14]. Η αρπαγή των αντικειμένων με μια μέθοδο λογικής που είναι ταχεία και αποτελεσματική. Το ίδιο μπορεί να ειπωθεί και για τον έλεγχο και τον σχεδιασμό τροχιάς του ρομποτικού βραχίονα, χρησιμοποιώντας τα δεδομένα βάθους. Τέλος, ολόκληρη η διαδικασία προσομοιώνεται σε ένα εικονικό περιβάλλον χρησιμοποιώντας τις υψηλές γραφικές δυνατότητες του Unreal Engine, επιδεικνύοντας την ικανότητα του συστήματος να αντιμετωπίσει προβλήματα.

12. Μελλοντική εργασία

Υπάρχουν πολλοί τρόποι για να επεκταθεί το εύρος της εργασίας που παρουσιάζεται την παρούσα διπλωματική, ξεκινώντας με τη χρήση δεδομένων βάθους εντός του Mask RCNN για τη βελτίωση της απόδοσής του. Αυτή η προσέγγιση έχει συζητηθεί σε προηγούμενη ενότητα, όπου η Python εμφανίζεται ως το προτιμώμενο εργαλείο για την υλοποίηση. Ενώ το Mask RCNN έχει χρησιμοποιηθεί ευρέως από ερευνητές για εργασίες τμηματοποίησης, οι πρόσφατες εξελίξεις σε δίκτυα όπως το YOLO V5 και το YOLO V8 έχουν δείξει μεγάλες δυνατότητες σε παρόμοιες περιπτώσεις. Επομένως, η ενσωμάτωση αυτών των δικτύων έναντι του Mask RCNN μπορεί να βελτιώσει περαιτέρω την απόδοση του συστήματος.

Μια άλλη πτυχή που δεν εξετάζεται αφορά τη δημιουργία εικόνων εντός του εικονικού περιβάλλοντος για την εκπαίδευση των δικτύων. Με βάση τις περιορισμένες γνώσεις του συγγραφέα για το θέμα, αυτός ο στόχος θα μπορούσε ενδεχομένως να επιτευχθεί στο περιβάλλον του Unreal Engine. Πιο συγκεκριμένα, η χρήση μιας τεχνικής τμηματοποίησης χρώματος θα επέτρεπε τη δημιουργία μασκών. Με τη λήψη δύο εικόνων ανά διαρρύθμιση των αντικειμένων—μία που απεικονίζει τα πραγματικά χρώματα των αντικειμένων και μία άλλη με διαφορετικά χρώματα—θα μπορούσαν να δημιουργηθούν μάσκες (και κατά συνέπεια οριοθετημένα πλαίσια) από τη δεύτερη, ενώ η πρώτη θα παρέχει εικόνες RGB και βάθους για την εκπαίδευση των δικτύων.



(a) *(b)* *(c)*

Εικόνα 12-1: Λήψη φωτογραφιών εντός του εικονικού περιβάλλοντος, Εικόνες για εκπαίδευση (a), Εικόνες για τον διαχωρισμό (b), Ταξινομημένη εικόνα (c).

Μια άλλη βασική πτυχή αυτής της διπλωματικής εργασίας θα περιλάμβανε την πρακτική προσομοίωση των μεθοδολογιών που αναπτύχθηκαν, σε ένα πραγματικό σενάριο. Δυστυχώς όμως, λόγο τεχνικών προβλημάτων δεν ήταν δυνατή η χρήση του ρομποτικού βραχίονα κατά την συγγραφή της διπλωματικής. Παρόλα αυτά παρακάτω παραδίδεται συνοπτικά η διαδικασία που θα είχε ακολουθηθεί:

1. Σειριακή Επικοινωνία

Δημιουργία μια σειριακής επικοινωνίας μεταξύ του MATLAB και της μονάδας CS7 που ελέγχει τον βραχίονα.

2. Έλεγχος του Βραχίονα

Δεδομένου ότι το Stäubli RX90L χρησιμοποιεί τη γλώσσα V+, ο έλεγχος του ρομποτικού βραχίονα μπορεί να επιτευχθεί με τη μετάδοση των επιθυμητών συντεταγμένων σε κώδικα V+.

3. Βαθμονόμηση συστήματος

Βασική προϋπόθεση είναι η βαθμονόμηση του συστήματος κάμερας-ρομποτικού βραχίονα για τον προσδιορισμό του ομογενούς μετασχηματισμού μεταξύ τους.

4. Εφαρμογή

Μετά την ολοκλήρωση των προαναφερθέντων βημάτων, το μοντέλο Simulink μπορεί να αντικατασταθεί με το πραγματικό μοντέλο και με μικρές αλλαγές στον κώδικα να γίνει η προσομοίωση στον πραγματικό κόσμο.

Λίστα Πινάκων

Λίστα Εικόνων

Βιβλιογραφία

[1] Murphy, K. P. *Machine Learning: A Probabilistic Perspective*. The MIT Press, Cambridge, Massachusetts, 2012.

[2] Shaoqing Ren, Kaiming He, Ross Girshick, Jian Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. https://doi.org/10.48550/arXiv.1506.01497

[3] Kaiming He, Georgia Gkioxari, Piotr Dollár, Ross Girshick. Mask R-CNN, 24 Jan 2018 https://doi.org/10.48550/arXiv.1703.06870

[4] Ramisa, A., Alenyà, G., Moreno-Noguer, F., & Torras, C. (2014). Learning RGB-D descriptors of garment parts for informed robot grasping. Engineering Applications of Artificial Intelligence, 35, 246–258. https://doi.org/10.1016/j.engappai.2014.06.025

[5] Zeng, A., Song, S., Yu, K.-T., Donlon, E., Hogan, F. R., Bauza, M., Ma, D., Taylor, O., Liu, M., Romo, E., Fazeli, N., Alet, F., Chavan Dafle, N., Holladay, R., Morona, I., Nair, P. Q., Green, D., Taylor, I., Liu, W., … Rodriguez, A. (2019). Robotic pick-and-place of novel objects in clutter with multi-affordance grasping and cross-domain image matching. The International Journal of Robotics Research, 41(7), 690–705. https://doi.org/10.1177/0278364919868017

[6] Hongkun Tian, Tianhai Wang, Yadong Liu, Xi Qiao, Yanzhou Li, Computer vision technology in agricultural automation —A review, Information Processing in Agriculture, Volume 7, Issue 1, 2020, Pages 1-19, ISSN 2214-3173, https://doi.org/10.1016/j.inpa.2019.09.006

[7] Girshick, R.B., Donahue, J., Darrell, T., & Malik, J. (2013). Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. *2014 IEEE Conference on Computer Vision and Pattern Recognition*, 580-587. https://doi.org/10.48550/arXiv.1311.2524

[8] Tuan-Tang Le, Trung-Son Le, Yu-Ru Chen, Joel Vidal, Chyi-Yeu Lin, 6D pose estimation with combined deep learning and 3D vision techniques for a fast and accurate object grasping, Robotics and Autonomous Systems, Volume 141, 2021, 103775, ISSN 0921-8890, https://doi.org/10.1016/j.robot.2021.103775.

[9] Y. Inagaki, R. Araki, T. Yamashita and H. Fujiyoshi, "Detecting layered structures of partially occluded objects for bin picking," *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Macau, China, 2019, pp. 5786-5791, doi: 10.1109/IROS40897.2019.8968093.

[10] Hongkun Tian, Kechen Song, Song Li, Shuai Ma, Jing Xu, Yunhui Yan, Data-driven robotic visual grasping detection for unknown objects: A problem-oriented review, Expert Systems with Applications, Volume 211, 2023, 118624, ISSN 0957-4174, https://doi.org/10.1016/j.eswa.2022.118624

[11] Zhen Li, Benlian Xu, Di Wu, Kang Zhao, Siwen Chen, Mingli Lu, Jinliang Cong, A YOLO-GGCNN based grasping framework for mobile robots in unknown environments, Expert Systems with Applications, Volume 225, 2023, 119993, ISSN 0957-4174, https://doi.org/10.1016/j.eswa.2023.119993

[12] R Aarthi., G Rishma., A Vision Based Approach to Localize Waste Objects and Geometric Features Exaction for Robotic Manipulation, Procedia Computer Science, Volume 218, 2023, Pages 1342-1352, ISSN 1877-0509, https://doi.org/10.1016/j.procs.2023.01.113.

[13] Silviu Răileanu, Theodor Borangiu, Florin Anton, Silvia Anton, Open source machine vision platform for manufacturing and robotics, IFAC-PapersOnLine, Volume 54, Issue 1, 2021, Pages 522-527, ISSN 2405-8963, https://doi.org/10.1016/j.ifacol.2021.08.060.

[14] Ziyad Tareq Nouri Master thesis

[15] Automate Virtual Assembly Line with Two Robotic Workcells. MATLAB & Simulink. (n.d.). https://www.mathworks.com/help/robotics/ug/automate-virtual-assembly-line.html
[16] Craig, J. J. (2022). Introduction to robotics: Mechanics and control. Pearson Education Limited.
[17] Siciliano, B., Sciavicco, L., Villani, L., & Oriolo, G. (2010). Robotics: Modelling, planning and control. Springer.
[18] Pieper, Donald. The Kinematics of Manipulators Under Computer Control. Stanford University, 1968.
[19]François Chaumette, S.Hutchinson. Visual servocontrol, PartI & II: Basicapproaches. IEEE Robotics and Automation Magazine, 2006, 13(4), pp.82-90. inria-00350283.
[20]MaskRCNN. Train Mask R-CNN network to perform instance segmentation - MATLAB. (n.d.). https://www.mathworks.com/help/vision/ref/trainmaskrcnn.html

Παράρτημα

I. Αρχεία MATLAB .m

Matlab_Simulink_Control.m

```
clc;
clear;
%% INPUTS
%add chp funcitons
addpath('B:\OneDrive - CHP\Σχολή μαθήματα\10. Diplomatikh\matlab code\Mask
CNN\CHP_Functions');
%Insert the urdf file of the robotic arm
robot_urdf_dir ="B:\OneDrive - CHP\Σχολή μαθήματα\10. Diplomatikh\Cad
Models\URDF\Staubli RX90.SLDASM\urdf\Staubli RX90.SLDASM.urdf";
%Insert the initial position of the robotic arm
Initial_Config = zeros(1,8);
%Insert Detector directory (TrainedDetector)
load('B:\OneDrive - CHP\Σχολή μαθήματα\10. Diplomatikh\matlab code\Mask
CNN\TrainedDetectors\test7.2.mat');
imageSize = [346 512 3]; %height-width (needed for detection purposes) [346
512 3]
%Depth sensor resolution
D_res = [340,417];
%Insert Simulink 3d World Directory File
simulink_dir = 'B:\OneDrive - CHP\Σχολή μαθήματα\10. Diplomatikh\matlab
code\Testing_Images\SIMULATION.slx';
%Insert ANNS
load('B:\OneDrive - CHP\Σχολή μαθήματα\10. Diplomatikh\matlab code\Mask
CNN\Best1_occlusion_net.mat');
net1 = best_net;
load('B:\OneDrive - CHP\Σχολή μαθήματα\10. Diplomatikh\matlab code\Mask
CNN\Best2_occlusion_net.mat');
net2 = best_net;
load('B:\OneDrive - CHP\Σχολή μαθήματα\10. Diplomatikh\matlab code\Mask
CNN\Best3_occlusion_net.mat');
net3 = best_net;
addpath('B:\OneDrive - CHP\Σχολή μαθήματα\10. Diplomatikh\matlab
code\Kinect');
%Insert Kinect (camera) position and orientation,
%additional camera parameters are set inside simulink make sure to change
%them if needed!
%Destances from base to camera
xcamera = 0.75;
ycamera = -0.0;
zcamera = 0.4;
%camera Rotation, vector outwards from camera
Camera_Rotation = [180,90,0];
%Camera Transformation
Rcamera = eul2rotm(Camera_Rotation*pi/180,"YZX"); %YXZ
Camera_Position = [xcamera,ycamera,zcamera];
T_base_camera= eye(4);
T_base_camera(1:3,1:3) = Rcamera;
T_base_camera(1:3,4)=(Camera_Position)';

f = 345; %focal length of D_camera fx = fy (almost);
cx = D_res(2)/2; %center point of depth camera;
cy = D_res(1)/2; %center point of depth camera;
% Be carful optical center is in form of [x,y] while image resolution is in
```

```
% form of [h,w] in simulink camera parameters.

%% Connect to the kinect
% colorDevice = imaq.VideoDevice('kinect',1);
% depthDevice = imaq.VideoDevice('kinect',2);
%% Take a Picture with Kinect
% [im_rgb,im_d]=KinectPicture(colorDevice,depthDevice);
% [rgb,d]=Kinect_RGBtoDepthMap(im_rgb,im_d);
% d_show = uint16_to_uint8(d);
% figure(1)
% montage({rgb,d_show});

%% Initialize robot
%Use simulink for the 3d world representation
%Use matlab to solve the problem

%import the robotic arm
Staubli_RX90L=importrobot(robot_urdf_dir,"urdf");
% Set the kinematic groop to the specified links
Kinematic_group =
struct(convertStringsToChars('BaseName'),'base_link',convertStringsToChars(
'EndEffectorBodyName'),'tool0');
% Solve the inverse kinematics by creating the robotIK function
%
https://www.mathworks.com/help/robotics/ref/analyticalinversekinematics.htm
l
invkin =
analyticalInverseKinematics(Staubli_RX90L,"KinematicGroup",Kinematic_group)
;
analsol = generateIKFunction(invkin,'robotIK');
% Set the data format to row manualy due to matlab restrictions
Staubli_RX90L.DataFormat='row';
%% Initialize Item Locations
%In UE coordinate system:
Item_Initial_Locs=[0.8,-0.2,-0.227; %item 1 [x,y,z]
    0.65,0.0,-0.227; %item 2
    0.75,0.15,-0.228];%item 3
Item_Initial_Rots = zeros(3);
Item_End_Locs=[-0.2,0.7,-0.227; %item 1 [x,y,z]
    0,0.7,-0.227; %item 2
    0.2,0.7,-0.228];%item 3
Item_End_Rots = zeros(3);

%% Parameters for simultion initialization
%(these do not matter for the initialization but are needed to use the
simulink model)
Item_to_grab=1;
end_effector_part_transform=zeros(4);
Gripper_Position2=0;
Gripper_Position1=0;

%% Parameters for simulation
VEL_BC = zeros(8,2); % velocity of robot at each configuration
Grab_time=30;
Letgo_time = 70;
Configs_Time=[0,20,Grab_time,40,60,Letgo_time,80,100];
%Camera Positions for Visulization
Camera_take_picture_loc = [1.699,0,-0.3581];
Camera_Hide_Loc = Camera_take_picture_loc+[0.75,0,0];
```

```
Camera_Locations_Visulization =
[Camera_take_picture_loc;Camera_Hide_Loc;Camera_Hide_Loc;Camera_take_pictur
e_loc]';
Camera_Speed_Visulization = zeros(3,size(Camera_Locations_Visulization,2));
Camera_Times_Visulization = [5,10,90,95];
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%

%% Robot Controll with matlab - simulink

%% Take a Picture with simulink
% Robot conficurations
Robot_Configs = [zeros(1,8);zeros(1,8)]';
%Velocity at each configuration
VEL_BC = zeros(8,2); %2 extra prismatic joints for grippers
%Time stamps to achieve the configurations
Configs_Time = [0,20];
% run the simulation for 0.1 sec to just get a picture
out = sim(simulink_dir,0.1);
%RGB image:
rgb = out.Image.signals.values;
%Depth image:
d = out.Depth.signals.values; % in this way we get distance in doubles
which is giving a value of meters
d=imresize(d,D_res); %change to actual depth resolution
%visualize images
figure(1)
montage({rgb,d})
% [x,y] =ginput(1)
d=d*1000; %to get the depth values in mm
%% Detection (mask RCNN)
[masks,labels, scores, bboxes] = segmentObjects(TrainedDetector,rgb);
%% visualize Detection
overlayeredImage = insertObjectMask(rgb,masks);
figure(3)
imshow(overlayeredImage)
hold on
showShape("rectangle",bboxes,"Label",labels,'LineColor',[1,0,0])
itemsonimage=size(labels,1);
% Change to match depth image resolution
[rgb,masks,bboxes]=ResizeImageMasksBoxes(rgb,masks,bboxes,D_res);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
while itemsonimage>0 %% Initiation of the loop

    %% Matlab coodrinate system transforms
    totalnumberofitems=size(Item_Initial_Locs,1);
    T_items_initial = zeros(4,4,totalnumberofitems);
    T_items_end = zeros(4,4,totalnumberofitems);
    for i=1:totalnumberofitems

T_items_initial(:,:,i)=UEtoMATLABtransform(Item_Initial_Locs(i,:),Item_Init
ial_Rots(i,:));

T_items_end(:,:,i)=UEtoMATLABtransform(Item_End_Locs(i,:),Item_End_Rots(i,:
));
    end
    %% Grasping Teqnique
    %%Gennerate Occlusion detection for all objects
```

```
    occlusion=Occlusion_Detector(net1,net2,net3,d,masks,labels);
    %Gennerate the grasping information for the object to be grabbed next

[itemcenter,itemorientation,itemz_grab,Item_to_grab,Gripper_Position2,Gripp
er_Position1] = Grab_Selection(masks,d,occlusion,f);
    %The name of the object to be grabbed next
    Item_to_grab_name = labels(Item_to_grab)
    %set the index for the simulation
    if Item_to_grab_name=="PT1"
        Item_to_grab=1;
    elseif Item_to_grab_name=="PT2"
        Item_to_grab=2;
    elseif Item_to_grab_name=="PT3"
        Item_to_grab=3;
    end

    %% Object world Transformation
    %Calculating X and Y diff from center of camera to the center of the
object
    X_item = (itemz_grab)*(itemcenter(1)-cx)/f;
    Y_item = (itemz_grab)*(itemcenter(2)-cy)/f;
    Z_item = itemz_grab;
    R_item = R_z(-itemorientation); % - sign is to change from image frame
to world frame
    % Item homogeneous transform from camera world frame to item world
frame
    T_camera_item = eye(4);
    T_camera_item(1:3,1:3)=R_item;
    T_camera_item(1:3,4)=[X_item;Y_item;Z_item];
    % Homogeneous transfrom from base to item
    T_base_item = T_base_camera*T_camera_item;

    %% Inverse Kinematics & Trajectories
    %Approach Transformation from end solution (z=10cm)
    T_approach_diff = [0,0,0,0;0,0,0,0;0,0,0,0.1;0,0,0,0];

    %Grab Approach
    T_base_item_approach=T_base_item+T_approach_diff;
    ikConfig = robotIK(T_base_item_approach);
    Robot_approach = best_sol(Initial_Config(1:6),ikConfig);

    %Grab
    clear ikConfig
    ikConfig = robotIK(T_base_item);
    Robot_Grab = best_sol(Robot_approach(1:6),ikConfig);
    end_effector_part_transform =
inv(T_base_item)*T_items_initial(:,:,Item_to_grab);% transform between the
grabbing point and the part coordinate system
%getTransform(Staubli_RX90L,Robot_Grab,'base_link','tool0')

    %Detach Approach
    clear ikConfig

T_base_letgo=T_items_end(:,:,Item_to_grab)*inv(end_effector_part_transform)
; % to get the letgo position
    T_base_letgo_approach =T_base_letgo+T_approach_diff;
    ikConfig = robotIK(T_base_letgo_approach);
    Robot_letgo_approach = best_sol(Robot_Grab(1:6),ikConfig);
```

```
    %Detach
    clear ikConfig
    ikConfig = robotIK(T_base_letgo);
    Robot_letgo = best_sol(Robot_letgo_approach(1:6),ikConfig);

    %Set the configurations and timings for simulation
    Robot_Configs =
[Initial_Config;Robot_approach;Robot_Grab;Robot_Grab;Robot_approach;Robot_l
etgo_approach;Robot_letgo;Robot_letgo_approach;Initial_Config]'; % Robot
configuration
    VEL_BC = zeros(8,size(Robot_Configs,2));
    Configs_Time= [0,20,Grab_time,40,50,60,Letgo_time,80,100];

    %Simulate
    out = sim(simulink_dir,100);
    I_rgb = out.Image.signals.values;
    I_d = out.Depth.signals.values;

    %Set new locations for items
    Item_Initial_Locs(Item_to_grab,:)=Item_End_Locs(Item_to_grab,:);
    Item_Initial_Rots(Item_to_grab,:)=Item_End_Rots(Item_to_grab,:);

    %% Take a Picture with simulink
    % Robot conficurations
    Robot_Configs = [zeros(1,8);zeros(1,8)]';
    %Velocity at each configuration
    VEL_BC = zeros(8,2); %2 extra prismatic joints for grippers
    %Time stamps to achieve the configurations
    Configs_Time = [0,20];
    %Run the simulation for 0.1 sec to just get a picture
    out = sim(simulink_dir,0.1);
    %RGB image:
    rgb = out.Image.signals.values;
    %Depth image:
    d = out.Depth.signals.values; % in this way we get distance in doubles
which is giving a value of meters
    d=imresize(d,D_res); %change to actual depth resolution
    %visualize images
    figure(1)
    montage({rgb,d})
    % [x,y] =ginput(1)
    d=d*1000; %to get the depth values in mm
    %% Detection (mask RCNN)
    [masks,labels, scores, bboxes] = segmentObjects(TrainedDetector,rgb);
    %% visualize Detection
    overlayeredImage = insertObjectMask(rgb,masks);
    figure(3)
    imshow(overlayeredImage)
    hold on
    showShape("rectangle",bboxes,"Label",labels,'LineColor',[1,0,0])
    itemsonimage=size(labels,1);
    % Change to match depth image resolution
    [rgb,masks,bboxes]=ResizeImageMasksBoxes(rgb,masks,bboxes,D_res);

end

Reply = 'No more objects where Found by the detector'
```

Kinect_Photos.m

```
%% INITIALIZATION (Run once)
% clc;
% clear;
%Directory to store High quallity mapped images
RGBHighQuallity_dir = 'B:\OneDrive - CHP\Σχολή μαθήματα\10.
Diplomatikh\matlab code\Kinect\Testing\RGBHigh Mapped';
%Directory to store mapped RGB images with the same resolution as depth
images
RGBFolder_dir = 'B:\OneDrive - CHP\Σχολή μαθήματα\10. Diplomatikh\matlab
code\Kinect\Testing\RGB Mapped';
%Directory to store mapped depth images
DepthFolder_dir = 'B:\OneDrive - CHP\Σχολή μαθήματα\10. Diplomatikh\matlab
code\Kinect\Testing\Depth Mapped';
%Directory to store raw images
RGB_RAW_dir = 'B:\OneDrive - CHP\Σχολή μαθήματα\10. Diplomatikh\matlab
code\Kinect\Testing\RGB RAW';
%Directory to store raw depth images
Depth_RAW_dir = 'B:\OneDrive - CHP\Σχολή μαθήματα\10. Diplomatikh\matlab
code\Kinect\Testing\Depth RAW';
%Directory to store visulization of depth images
Depth_Images_RAW_dir = 'B:\OneDrive - CHP\Σχολή μαθήματα\10.
Diplomatikh\matlab code\Kinect\Testing\Depth Images RAW';
Depth_Images_dir = 'B:\OneDrive - CHP\Σχολή μαθήματα\10. Diplomatikh\matlab
code\Kinect\Testing\Depth Images Mapped';
%Initialization of kinect sensors
colorDevice = imaq.VideoDevice('kinect',1);
depthDevice = imaq.VideoDevice('kinect',2);

%% Get Images
[img_rgb,img_d]=KinectPicture(colorDevice,depthDevice);
[rgb_c,d_c]=Kinect_RGBtoDepthMap(img_rgb,img_d);
%high resolution mapped image:
rgb = imcrop(img_rgb,[285,39,1272,1037]);

%% Display images
d_show =uint16_to_uint8(d_c);
figure(1)
title('Montage')
montage({rgb_c,d_show});
figure(2)
imshow(rgb)
dr_show = uint16_to_uint8(img_d);
figure(3)
montage({img_rgb,dr_show})

%% Save images
rgbh_dir = strcat(RGBHighQuallity_dir,'\',num2str(i),'.jpg');
imwrite(rgb,rgbh_dir);
rgb_dir = strcat(RGBFolder_dir,'\',num2str(i),'.jpg');
imwrite(rgb_c,rgb_dir);
d_dir = strcat(DepthFolder_dir,'\',num2str(i));
save(d_dir,"d_c")
rgbr_dir = [RGB_RAW_dir,'\',num2str(i),'.jpg'];
imwrite(img_rgb,rgbr_dir);
dr_dir = [Depth_RAW_dir,'\',num2str(i)];
save(dr_dir,"img_d")
di_dir = [Depth_Images_dir,'\',num2str(i),'.jpg'];
```

```
imwrite(d_show,di_dir)
dir_dir = [Depth_Images_RAW_dir,'\',num2str(i),'.jpg'];
imwrite(dr_show,dir_dir)
i=i+1;
```

Dataset_Creation_With_Occlusion.m

```
clc;
clear;
% imageLabeler;

% Use the imageLabeler app to create polygon shaped masks and the bounding
% boxes will be created with the code bellow. export the gTrouth as "table"
% with the name "gTruth".

% In the image labler app you need to specify for each label 2 attributes:

%Item attribute with the name "Item", is a numeric value that identifies
%what number of object is the label for the same class of objects. This is
%needed in case one object was two or more polygon shaped masks (due to
%occlusion). So is two areas are fo the same object then both areas should
%have the same value for the attribute item. Eg 1,2,3,4....

% Occlusion Attribute with the name "Occlusion", is a logical value that
% specifies occlusion of the spesific object if an object is occluded then
% this value should be true, if it is not (or if it occludes other object)
% this value should be falase. If an object has multiple areas it should
% you should manually set the occlusion to all the areas (set to value
"true"
% because an item that is not occluded should not have many areas)
%% INPUTS
% Add path to custom made CHP Functions
addpath('B:\OneDrive - CHP\Σχολή μαθήματα\10. Diplomatikh\matlab code\Mask
CNN\CHP_Functions')
% Add the source directory to the path (add maskrcnn main functions)
addpath('B:\OneDrive - CHP\Σχολή μαθήματα\10. Diplomatikh\matlab code\Mask
CNN\mask-rcnn-main\src')
% Set the root directory for COCO API:
cocoAPIDir = 'B:\OneDrive - CHP\Σχολή μαθήματα\10. Diplomatikh\matlab
code\Mask CNN\MatlabAPI';
%Set the Image Training Folder (needs to contain images as described in
imageFile above):
trainImgFolder = 'B:\OneDrive - CHP\Σχολή μαθήματα\10. Diplomatikh\matlab
code\Kinect\Testing\RGBHigh Mapped'; % where the images are stored
%load the groud truth data
load("B:\OneDrive - CHP\Σχολή μαθήματα\10. Diplomatikh\matlab
code\Kinect\Testing\OCCLUSION CLASSES\gTruth_High_Occlusion.mat");%load the
gTruth table
h= 1037; %height of the images
w=1272; %width of the images
%Set the annotation folder where all the .mat files will be stored
AnnotationFolder = 'B:\OneDrive - CHP\Σχολή μαθήματα\10. Diplomatikh\matlab
code\Kinect\Testing\OCCLUSION CLASSES\Anottations_High_Occlusion';
%Set the occlison annotation folder where all the .mmat files will be
%stored
OcclusionFolder = 'B:\OneDrive - CHP\Σχολή μαθήματα\10. Diplomatikh\matlab
code\Kinect\Testing\OCCLUSION CLASSES\Anottations_High_Occlusion -
Occlusion';
% Set image size for training (you can change that if you need to lower the
res for faster training)
imageSize = [h w 3]; %height-width
mask_image_save_location = "B:\OneDrive - CHP\Σχολή μαθήματα\10.
Diplomatikh\matlab code\Kinect\Testing\OCCLUSION CLASSES\Masks_HO\";
%% Data Manipulation
```

```
Image_Number = size(gTruth,1); % number of images in the table
Object_Number =size(gTruth,2); % number of objects in the table

varnames = gTruth.Properties.VariableNames; % the names are set in the
gTrouth table
varnames= varnames(1,2:end);
blds_cell = cell(Image_Number,1); % Bounding box cell array. Working with
cell is easier (at least for me)
mask_cell = cell(Image_Number,1);
variable_names_cell=cell(Image_Number,1);
occlusion_cell = cell(Image_Number,1);
% the image cell is not needed for the datastore it is only cept in here
% for legacy reasons. The datastore needs only the image file names.
%image_cell = cell(Image_Number,1); %open images inside matlab and store
them inside image_cell for concatenation later
for n=1:Image_Number
    i=1; %goes from [1 to Total_items]
    clear masks_on_image var_name_categorical% delete the last atempts
    Total_items=0;
    for ni=2:Object_Number
        if isempty(gTruth{n,ni}{1,1})==0
            Total_items=Total_items+gTruth{n,ni}{1,1}(end).Item;
        end
    end
    masks_on_image=false([h,w,Total_items]);
    %total_var_names = 1:Total_items; %to create a categorical array of
size Total_items
    var_name_categorical =
cell(1,Total_items);%categorical(total_var_names);
    occlusion_property = false(1,Total_items);
    bboxes = zeros(Total_items,4);
    for nn=2:Object_Number% object classes are Object_Number - 1
        num_of_object_segments = size(gTruth{n,nn}{1,1},2);
        if isempty(gTruth{n,nn}{1,1})==0
            Objects_in_Image = gTruth{n,nn}{1,1}(end).Item;% find how many
objects of a specific type exist (e.g 2 rubiks cubes)
        else
            Objects_in_Image=0;
        end
        if Objects_in_Image > 0
            BD= zeros(Objects_in_Image,4); %zero-out the matrix and change
it's dimensions per iteration
            %
            Item=1; % to distinguish items of the same type [1 -
Objects_in_Image]
            nnn=1;
            while nnn<=num_of_object_segments

                    mask = false([h,w]); % create the mask for the specific
object
                    if gTruth{n,nn}{1,1}(nnn).Item==Item % every object has
1 or more segments (because of occlusion) therefore we have to create one
mask for all the segments together (of that item)
                        gT_polygon = gTruth{n,nn}{1,1}(nnn).Position; %
find the polygon coordinates inside the ground truth table
                        x = gT_polygon(:,1); % find the x coordinates of
the polygon points
                        y =gT_polygon(:,2); %find the y coordinates of the
polygon points
```

```
                        mask=poly2mask(x,y,h,w); % crete the polygon by
inserting ones (1) where there is an object
                        masks_on_image(:,:,i)=mask+masks_on_image(:,:,i);

occlusion_property(i)=gTruth{n,nn}{1,1}(nnn).Occlusion;
                    end
                    if nnn<num_of_object_segments %generate for every
objects its properties
                        if gTruth{n,nn}{1,1}(nnn+1).Item ~= Item
                            var_name_categorical(i) = varnames(nn-1);
                            bboxes(i,:) =
BoundingBox_From_Mask(masks_on_image(:,:,i)); %Create the vertical
rectangle (Bounding box) that encapsulates all the polygon points

occlusion_property(i)=gTruth{n,nn}{1,1}(nnn).Occlusion;

imwrite(masks_on_image(:,:,i),strcat(mask_image_save_location,num2str(n),nu
m2str(nn-1),num2str(nnn),'.png')) % write the image to a file
                            Item=Item+1;
                            i=i+1;
                        end
                    else %nnn==num_of_object_segments
                            var_name_categorical(i) = varnames(nn-1);
                            bboxes(i,:) =
BoundingBox_From_Mask(masks_on_image(:,:,i)); %Create the vertical
rectangle (Bounding box) that encapsulates all the polygon points

occlusion_property(i)=gTruth{n,nn}{1,1}(nnn).Occlusion;

imwrite(masks_on_image(:,:,i),strcat(mask_image_save_location,num2str(n),'_
',num2str(nn-1),'_',num2str(nnn),'.png')) % write the image to a file
                            Item=Item+1;
                            i=i+1;
                    end
                    nnn=nnn+1;

            end

            %blds_cell{n,nn-1} = BD; % instert bounding boxes in the
bounding box cell array
        end
    end
    %image_cell{n}= imread(gTruth{n,1}{1,1});
    occlusion_cell{n} = occlusion_property;
    blds_cell{n}=bboxes;
    mask_cell{n}=masks_on_image;
    variable_names_cell{n}=var_name_categorical';
end


[~,image_names,ext] = fileparts(gTruth{:,1});
image_names = strcat(image_names,ext);%get the name and the type (.jpg)

TrainingData = [image_names,blds_cell,variable_names_cell,mask_cell];

%% Create a different .mat file for every image
%In the annotation folder there need to be only .mat files the names does
%not matter but in this case we save them as 1.mat,2.mat,3.mat,4.mat.. etc
%for mask rccn training:
```

```
for n=1:Image_Number
 mat_to_save = TrainingData(n,:);
 %save averything in a different variable so as when the mat files are
 %loaded the open up 4 different variables with the names specified in the
 %save function:
 imageFile = mat_to_save{1}; % it is the image file name eg. '0001.png'
(string)
 boxes = mat_to_save{2}; % it is a 2d array (double) containing bounding
boxes in the form of: Mx4 (where M is the objects in image '0001.png')
 labels = categorical(mat_to_save{3}); % its is a categorical array
containing the categories for each bounding box Mx1
 masks = mat_to_save{4};% its a 3d array containing the masks as
HeightxWidthxM (logical) for each object

save(strcat(AnnotationFolder,'\',num2str(n)),"imageFile","boxes","labels","
masks"); % the names play a very big role check: cocoAnnotationMatReader.m
and put the correct names there
 occlusion= occlusion_cell{n};
 save(strcat(OcclusionFolder,'\',num2str(n)),"occlusion");
end


%% Save Variable Names
item_names = varnames(1:end);
save('item_names','item_names');
%% Using the coco api to create the datastores (because matlab
imagedatastore does not support multi image masks)
%% Using code from MaskRCNNTrainingExample.mlx
% COCO-MATLABAPI: https://github.com/cocodataset/cocoapi
% Add the API directory to the path
addpath(cocoAPIDir);
% Create the training datastore to read image and ground truth data from
% the unpacked annotation MAT file
ds =
fileDatastore(AnnotationFolder,'ReadFcn',@(x)helper.cocoAnnotationMATReader
(x, trainImgFolder));
%% OUTPUT
trainDS = transform(ds, @(x)helper.preprocessData(x, imageSize));
save('trainDS','trainDS'); % save the training datastore to use for the
training m file
%% Test the datastore!
trainDS.shuffle();
%preview the datastore needs to be in form of 1x4 cell array like this:
%{512×512×3 uint8}    {7×4 double}    {7×1 categorical}    {512×512×7
logical}
data = preview(trainDS) % done
%% visualise an image with its bounding box
%load("1.mat")
% overlayeredImage = insertObjectMask(test_img,masks);
% figure(2)
% imshow(overlayeredImage)
% hold on
% showShape("rectangle",boxes,"Label",labels,'LineColor',[1,0,0])
```

Image_Augmentation.m

```
clc;
clear;
%% INPUT
addpath('CHP_Functions') %Add the custom made functions
% Add the source directory to the path (add maskrcnn main functions)
addpath('B:\OneDrive - CHP\Σχολή μαθήματα\10. Diplomatikh\matlab code\Mask
CNN\mask-rcnn-main\src')
% Set the root directory for COCO API:
cocoAPIDir = 'B:\OneDrive - CHP\Σχολή μαθήματα\10. Diplomatikh\matlab
code\Mask CNN\MatlabAPI';
% Set the directory where the images to be augmented are located
Images_location = 'B:\OneDrive - CHP\Σχολή μαθήματα\10. Diplomatikh\matlab
code\Kinect\CleanImages\HighQuallity\RGBHigh\';
% Set the location of the initial dataset to be augmented
DS_location = 'B:\OneDrive - CHP\Σχολή μαθήματα\10. Diplomatikh\matlab
code\Kinect\CleanImages\HighQuallity\trainDS.mat'; % insert the Dataset
created with Datase_Creation.m file
%Set all the item names contained in the images
Item_Names = 'B:\OneDrive - CHP\Σχολή μαθήματα\10. Diplomatikh\matlab
code\Item images\v2\Testing\item_names.mat';
%Set the directory to save the augmented images
trainImgFolder= 'B:\OneDrive - CHP\Σχολή μαθήματα\10. Diplomatikh\matlab
code\Kinect\CleanImages\HighQuallity\Augmented\Images'; % location where
the images will be saved
%Set the directory where the annotatios are going to be stored
Augmented_Images_Annotation_Folder = 'B:\OneDrive - CHP\Σχολή μαθήματα\10.
Diplomatikh\matlab
code\Kinect\CleanImages\HighQuallity\Augmented\Annotations'; % location
where the annotations will be saved
%Set the directory where the Backround images are stored
Background_Images_Location = 'B:\OneDrive - CHP\Σχολή μαθήματα\10.
Diplomatikh\matlab
code\Kinect\CleanImages\HighQuallity\Augmented\Background\JPEG'; % location
of imagedatastore containing the backround images for the augmentation
%Set the ammount of augmented images created from 1 image.
Augmentation_Number = 10; %Ammount of images to create with random
rotations and translations of the Items per image using a random backround
%Set the image  sizes
h= 1042; %height of the images
w=1353; %width of the images
Image_Size = [w,h];%width,height
imageSize=[h,w,3];%height,width
% Set the directory for intermediate images to be stored.
Augmented_Images_Intermediate_Location= 'B:\OneDrive - CHP\Σχολή
μαθήματα\10. Diplomatikh\matlab
code\Kinect\CleanImages\HighQuallity\Augmented\Intemediate\';

%%
% Find the image center
Image_Center= Image_Size/2;
% Create an imagedatastore for the backround images (just to get their
names)
Background_IMDS = imageDatastore(Background_Images_Location);
% Number_of_Backround_Images = size(Background_IMDS,1);

DS = load(DS_location); % Load the datastore to be augmented
```

```
DS=DS.trainDS; % the datastore name should be trainDS for this to work

Number_of_Images = size(DS.UnderlyingDatastores{1,1}.Files,1); % Number of
images in the datastore

for n =1:Number_of_Images % for every image
load(DS.UnderlyingDatastores{1,1}.Files{n,1}) % load masks,
labels,boundingboxes and image name orresponding to an image from the
annotation folder
image_location = strcat(Images_location,imageFile); % directory of the
image
image = imread(image_location); %load the image to be augmented
%save the initial parameters of the annotations corresponding to the image
labels_n=labels;
boxes_n = boxes;
imageFile_n = imageFile;
masks_n=masks;
num_of_items_in_image = size(labels,1); %number of items in the image


    for nn=1:Augmentation_Number
        clear msk mask img boxes labels % clear the anotation parameters
for the augmented images
        masks = masks_n;
        boxes=boxes_n;
        labels=labels_n;
        imageFile = imageFile_n;
        number_of_bg_images = size(Background_IMDS.Files,1); % number of
backround images
        bg_image=floor(rand*number_of_bg_images+1); %pick a random
background image
        IMG = imread(Background_IMDS.Files{bg_image}); %initiate the
backround image

        %i is like nnn but it has randomly sorted the number (for random
occlusion purposes)
        i = 1:num_of_items_in_image;
        i = i(randperm(length(i))); %create an array of length num of items
in image but to have randomly distributed numbers. This number indicates
the occlusion order the i(end) item occludes all the others the i(1) is
occluded by all the other (it is placed first in the image)

            for nnn=1:num_of_items_in_image
                props = regionprops(masks(:,:,i(nnn)),'Centroid'); % find
the centroid of the mask (masks need to have only 1 region)
                centroid= props.Centroid;
                img = image;
                msk = masks(:,:,i(nnn));
                Translation = Image_Center - centroid;
                img = imtranslate(img,Translation,'FillValues',[0,0,0]);
                msk=imtranslate(msk,Translation,'FillValues',0);
                %image augmentation part
                theta = rand*360;
                img=imrotate(img,theta,"bilinear","crop");
                msk=imrotate(msk,theta,"bilinear","crop");
                translate= (rand*Image_Center-rand*Image_Center);  %random
translation
                img = imtranslate(img,translate,'FillValues',[0,0,0]);
                msk=imtranslate(msk,translate,'FillValues',0);
```

```
                saveloc = strcat(Augmented_Images_Intermediate_Location,
num2str(n),'_',num2str(nn),'_',num2str(nnn),'.png');
                Alpha = double(msk);
                imwrite(img,saveloc,'alpha',Alpha)
                masks(:,:,i(nnn)) = msk;
                img_intermediate = imread(saveloc);
                IMG = CHP_Blend(IMG,img_intermediate,msk);
                props=regionprops(masks(:,:,i(nnn)),'BoundingBox');
                boxes(i(nnn),:)=props.BoundingBox;% wrong
            end
        imageFile = strcat(num2str(n),'_',num2str(nn),'.jpg');%mask rcnn
does not support png!
        save_loc = strcat(trainImgFolder,'\',imageFile);
        imwrite(IMG,save_loc);

        [masks,boxes,labels]=Occlusion_Correction(masks,boxes,labels,i,50);

save(strcat(Augmented_Images_Annotation_Folder,'\',num2str(n),'_',num2str(n
n)),"imageFile","boxes","labels","masks");
    end
end


%% Using the coco api to create the datastores (because matlab
imagedatastore does not support multi image masks)
%% Using code from MaskRCNNTrainingExample.mlx
% COCO-MATLABAPI: https://github.com/cocodataset/cocoapi
% Add the API directory to the path
addpath(cocoAPIDir);
% Create the training datastore to read image and ground truth data from
% the unpacked annotation MAT file
ds =
fileDatastore(Augmented_Images_Annotation_Folder,'ReadFcn',@(x)helper.cocoA
nnotationMATReader(x, trainImgFolder));
trainDS = transform(ds, @(x)helper.preprocessData(x, imageSize));
save('trainDS','trainDS'); % save the training datastore to use for the
training m file
%% Test the datastore!
trainDS.shuffle();
%preview the datastore needs to be in form of 1x4 cell array like this:
%{512×512×3 uint8}    {7×4 double}    {7×1 categorical}    {512×512×7
logical}
data = preview(trainDS) % done
%% visualise an image with its bounding box
% im_pos = string(gTruth{1,1});
% figure(1)
% im = imread(im_pos);
% imshow(im)
% hold on
% rectangle("Position",TrainingData{1,2})
```

Resize_Datastore.m

```
clc;
clear;
%% Input
% Set the resolution to resize to
resize_res= [340,417];%height width
% Add the source directory to the path (add maskrcnn main functions)
addpath('B:\OneDrive - CHP\Σχολή μαθήματα\10. Diplomatikh\matlab code\Mask
CNN\mask-rcnn-main\src')
% Set the root directory for COCO API
cocoAPIDir = 'B:\OneDrive - CHP\Σχολή μαθήματα\10. Diplomatikh\matlab
code\Mask CNN\MatlabAPI';
% Set the directory of the RGB images to be rescaled
Highres_rgb_dir = "B:\OneDrive - CHP\Σχολή μαθήματα\10. Diplomatikh\matlab
code\UE Generated Pictures\High Res\RGB";
%Set the directory of the datastore to be rescaled
trainDS_dir = "B:\OneDrive - CHP\Σχολή μαθήματα\10. Diplomatikh\matlab
code\UE Generated Pictures\High Res\trainDS.mat";
%Initial Occlusion Folder
Occlusion_dir = "B:\OneDrive - CHP\Σχολή μαθήματα\10. Diplomatikh\matlab
code\UE Generated Pictures\High Res\Occlusion Annotations";

%The annotation folder to save the new annotation files
AnnotationFolder = "B:\OneDrive - CHP\Σχολή μαθήματα\10. Diplomatikh\matlab
code\UE Generated Pictures\Low Res\Annotations";
%The rescaled image dirirectory
Rescaled_rgb_dir = "B:\OneDrive - CHP\Σχολή μαθήματα\10. Diplomatikh\matlab
code\UE Generated Pictures\Low Res\RGB";
%Occlusion Annotations resized
Occlusion_dir_resized = "B:\OneDrive - CHP\Σχολή μαθήματα\10.
Diplomatikh\matlab code\UE Generated Pictures\Low Res\Occlusion
Annotations";

%% Output
%load the datastore
load(trainDS_dir)
%get the file directories
Files_dir = trainDS.UnderlyingDatastores{1,1}.Files;
%get the number of files in the datastore
num_of_files= size(Files_dir,1);

for i=1:num_of_files
    %load one annotation file at a time
    load(Files_dir{i}) %masks,labels,imageFile,boxes
    %get the directory of the image to be resized
    rgb_dir = strcat(Highres_rgb_dir,'\',imageFile);
    rgb=imread(rgb_dir);
    res = size(rgb,1,2);
    resize_ratio=resize_res./res;
    boxes_new = boxes;
    %find the amount of different objects in the image
    items_on_image = size(labels,1);
    masks_new = false([resize_res,items_on_image]);
    %Rescale the boxes and the masks
    for ii=1:items_on_image
        boxes_new(ii,1)=round(boxes(ii,1)*resize_ratio(2));
        boxes_new(ii,3)=round(boxes(ii,3)*resize_ratio(2));
        boxes_new(ii,2)=round(boxes(ii,2)*resize_ratio(1));
```

```
        boxes_new(ii,4)=round(boxes(ii,4)*resize_ratio(1));
        masks_new(:,:,ii)=imresize(masks(:,:,ii),resize_res);
    end
    %rescale the image
    rgb_new = imresize(rgb,resize_res);
    %save the data to a new annotation file
    imageFile = strcat(num2str(i),'.jpg');
    rgbwrite_dir = strcat(Rescaled_rgb_dir,'\',imageFile);
    imwrite(rgb_new,rgbwrite_dir)
    boxes = boxes_new;
    masks=masks_new;


save(strcat(AnnotationFolder,'\',num2str(i)),"imageFile","boxes","labels","
masks"); % the names play a very big role check: cocoAnnotationMatReader.m
and put the correct names there
    %for occlusion
    [n,name,ext]=fileparts(Files_dir{i});
    occlusion_dir = strcat(Occlusion_dir,'\',name,'.mat');
    load(occlusion_dir);
    save(strcat(Occlusion_dir_resized,'\',num2str(i)),"occlusion")
end
%% Using code from MaskRCNNTrainingExample.mlx
% COCO-MATLABAPI: https://github.com/cocodataset/cocoapi
% Add the API directory to the path
addpath(cocoAPIDir);
% Create the training datastore to read image and ground truth data from
% the unpacked annotation MAT file
trainImgFolder=Rescaled_rgb_dir;
clear trainDS
ds =
fileDatastore(AnnotationFolder,'ReadFcn',@(x)helper.cocoAnnotationMATReader
(x, trainImgFolder));
%% OUTPUT
imageSize = [resize_res,3];
trainDS = transform(ds, @(x)helper.preprocessData(x, imageSize));
save('trainDS','trainDS'); % save the training datastore to use for the
training m file
%% Test the datastore!
 trainDS.shuffle();
%preview the datastore needs to be in form of 1x4 cell array like this:
%{512×512×3 uint8}    {7×4 double}    {7×1 categorical}    {512×512×7
logical}
 data = preview(trainDS);
%% Visualise
% load('B:\OneDrive - CHP\Σχολή μαθήματα\10. Diplomatikh\matlab
code\Kinect\Testing\Resized to best net\Annotations\1.mat');
% test_img=imread('B:\OneDrive - CHP\Σχολή μαθήματα\10. Diplomatikh\matlab
code\Kinect\Testing\Resized to best net\RGB\1.jpg');
% overlayeredImage = insertObjectMask(test_img,masks);
% figure(2)
% imshow(overlayeredImage)
% hold on
% showShape("rectangle",boxes,"Label",labels,'LineColor',[1,0,0])
```

Occlusion_Data_Creation.m

```
clc;
clear;
%% Inputs
%Add CHP Functions
addpath('B:\OneDrive - CHP\Σχολή μαθήματα\10. Diplomatikh\matlab code\Mask
CNN\CHP_Functions');
%load the training ds created by the dataset creation
TrainDS_dir = "B:\OneDrive - CHP\Σχολή μαθήματα\10. Diplomatikh\matlab
code\Kinect\CleanImages\HighQuallity\Downscaled\trainDS.mat";
%Input the occlusion mat files for training
Occlusion_dir = "B:\OneDrive - CHP\Σχολή μαθήματα\10. Diplomatikh\matlab
code\Kinect\CleanImages\HighQuallity\Occlusion";
%Input the depth images folder
Depth_dir = "B:\OneDrive - CHP\Σχολή μαθήματα\10. Diplomatikh\matlab
code\Kinect\CleanImages\HighQuallity\Depth";
%load the names of the parts
load("B:\OneDrive - CHP\Σχολή μαθήματα\10. Diplomatikh\matlab
code\Kinect\Images\Mapped\item_names")
%give a location to visualize the intersection areas
intersection_dir = 'B:\OneDrive - CHP\Σχολή μαθήματα\10. Diplomatikh\matlab
code\Kinect\CleanImages\HighQuallity\Intersection_occlusion_dir';
%intersection circle radius (in pixels)
r_intersection = 10;
%% Main
load(TrainDS_dir);

TrainDS = trainDS.UnderlyingDatastores{1,1}.Files;
total_images = size(TrainDS,1);
id = 1;

for i=1:total_images
    %load the masks etc from annotation folder...
    load(TrainDS{i,1});
    % get the name of the annotation .mat file to open the corresponting
    % .mat file containing the depth image and the occlusion labels
    [folder,name,ext]=fileparts(TrainDS{i,1});
    load(strcat(Occlusion_dir,"\",name,'.mat')) % load the corresponding
mat file
    load(strcat(Depth_dir,'\',name,'.mat'));
    if size(labels,1)>=2
        [INPUT,lab] = Occlusion_Features(d_c,masks,labels);
        objects =size(occlusion,2);
        for ii=1:objects
            DATA(id,1:6)=INPUT(ii,:);
            DATA(id,7)=occlusion(ii);
            DATA_labels(id,1)=lab(ii,1);
            DATA_labels(id,2)=name;
            id=id+1;
        end
    end
end

i1=1;
i2=1;
i3=1;
%Split the data for each network
for i=1:size(DATA_labels,1)
```

```
    if DATA_labels(i,1)=='PT1'
        DATA1(i1,:) =DATA(i,:);
        i1=i1+1;
    elseif DATA_labels(i,1)=='PT2'
        DATA2(i2,:) =DATA(i,:);
        i2=i2+1;
    else
        DATA3(i3,:) =DATA(i,:);
        i3=i3+1;
    end
end
```

Occlusion_ANN_Training.m

```
clear;
clc;
%% INPUTS
load("B:\OneDrive - CHP\Σχολή μαθήματα\10. Diplomatikh\matlab code\UE
Generated Pictures\Low Res\DATA2.mat")
DATA = DATA2;
%The raw that the output values are located
output_ind = 7;
input_ind = 4;
datasize = size(DATA,1);
%Set the validation and testing indicies
validation_ind = 4:10:datasize;
test_ind = 6:10:datasize;
%% Input manipulation
test_datasize = size(test_ind,2);
input = DATA(:,1:input_ind);
output = DATA(:,output_ind);
%The remaining indices go to the training data = 80%
training_ind= 1:datasize;% create a matrix with all the indices

index = 1;
for n =1:test_datasize
    training_ind(validation_ind(n)+1-index)=[]; %remove validation indices
    index = index+1;
    training_ind(test_ind(n)+1-index)=[];%remove test indices
    index = index+1;
end

% Separation of training,validation and testing data
training_data=input(training_ind,1:input_ind);
training_output = output(training_ind);

validation_data = input(validation_ind,1:input_ind);
validation_output = output(validation_ind);

testing_data = input(test_ind,1:input_ind);
test_output = output(test_ind);
%% training
HiddenSizes = [100,100,50,50,25,25,10,10,5,5]; %400,400,200,200,100,100,
net = feedforwardnet(HiddenSizes,'trainrp');%to use gpu you have to change
from trainlm, trainscg, trainrp
%%Set ANN training parameters
net.trainParam.epochs = 10000; % maximum epochs
net.trainParam.goal = 0;  % We want 0 error
net.trainParam.max_fail = 20; % validation failures before rejection
net.divideFcn = 'divideind'; % User defined training/testing/validation
data
net.divideParam.trainInd = training_ind;
net.divideParam.valInd = validation_ind;
net.divideParam.testInd = test_ind;
net.performParam.regularization = 0;
net.performParam.normalization = 'none';
% net.layers{1}.transferFcn ='poslin';
%net.layers{end}.transferFcn ='softmax';
%net.performFcn = 'crossentropy';%'mse'; %Using mean absolute error
net.trainParam.showWindow = 0; % close window pop up for lowering
computation time
```

```
% net.trainFcn = 'trainscg';
perf=1000;
for i=1:100
    net.trainParam.showWindow = 0;
    [trained_net,tr] = train(net,input',output'); % train the network
%'useParallel','yes' for cpu // 'useGPU','yes'
    y = trained_net(input');% find the outputs for all the inputs
    %p = crossentropy(trained_net,output',y,{1});%  %Define the error as
described by the performace function (sse)
    p = perform(trained_net, output', y);
    if perf>p
        perf=p; %best performing network
        best_net=trained_net; %keep the best perfoming net
        y_best=y; %keep the relative error
        best_tr =tr; %keep matrix tr
    end
    i
end


%Check best solution
y=best_net(input');
y = round(y,0);
error = abs(y-output');
totalerrors = sum(error);

error_validation=sum(error(validation_ind));
error_testing = sum(error(test_ind));
dif = y-output';
false_positive=0;
false_negative=0;
for i=1:datasize
    if dif(i)>0
        false_positive=false_positive+1;
    elseif dif(i)<0
        false_negative = false_negative+1;
    end
end
```

Mask_RCNN_Training.m

```
clc;
clear;
%% INPUTS
% Add the source directory to the path (add maskrcnn main functions)
addpath('B:\OneDrive - CHP\Σχολή μαθήματα\10. Diplomatikh\matlab code\Mask
CNN\mask-rcnn-main\src')
% Set the root directory for COCO API:gTruth
cocoAPIDir = 'B:\OneDrive - CHP\Σχολή μαθήματα\10. Diplomatikh\matlab
code\Mask CNN\MatlabAPI';
% Add the API directory to the path
addpath(cocoAPIDir);
imageSize = [1042 1353 3]; %height-width
%load training datastore (trainDS)
load('B:\OneDrive - CHP\Σχολή μαθήματα\10. Diplomatikh\matlab
code\Kinect\CleanImages\HighQuallity\trainDS.mat')
%load cell array with item_names
load('B:\OneDrive - CHP\Σχολή μαθήματα\10. Diplomatikh\matlab
code\Kinect\Images\Mapped\item_names.mat')

%% Create the network
detector = maskrcnn('resnet50-
coco',item_names,'InputSize',imageSize,PoolSize=[14
14],MaskPoolSize=[14,14]);%https://www.mathworks.com/help/vision/ref/maskrc
nn.html

%% Train the network
options = trainingOptions("sgdm",  ...
    InitialLearnRate=0.002, ...
    Momentum=0.9, ...
    LearnRateSchedule="piecewise", ...
    LearnRateDropPeriod=1, ...
    LearnRateDropFactor=0.99, ...
    Plot="none", ...
    MaxEpochs=12, ...
    MiniBatchSize=2, ...
    BatchNormalizationStatistics="moving", ...
    ResetInputNormalization=false, ...
    ExecutionEnvironment="gpu", ...
    VerboseFrequency=10);
    % GradientThresholdMethod='l2norm',...
    % L2Regularization=10e-5,...

[TrainedDetector,info] =
trainMaskRCNN(trainDS,TrainedDetector,options,'NumRegionsToSample',128,'Num
StrongestRegions',1250,'PositiveOverlapRange',[0.75,1],'NegativeOverlapRang
e',[0  0.75]); %
https://www.mathworks.com/help/vision/ref/trainmaskrcnn.html
%FreezeSubNetwork="backbone"

%% test network for one image
% Read the image for inference
test_img = imread('B:\OneDrive - CHP\Σχολή μαθήματα\10. Diplomatikh\matlab
code\Kinect\Testing\Resized to best net\RGB\19.jpg');

% Define the target size of the image for inference
targetSize = imageSize; % same as the network image size
```

```
% Resize the image maintaining the aspect ratio and scaling the largest
% dimension to the target size.
imgSize = size(test_img);
[~, maxDim] = max(imgSize);
resizeSize = [NaN NaN];
resizeSize(maxDim) = targetSize(maxDim);

test_img = imresize(test_img, resizeSize);
% detect the objects and their masks for more info:
https://www.mathworks.com/help/vision/ref/maskrcnn.segmentobjects.html#mw_a
9899d3b-d637-4834-85c9-fe5cfae7f8af_sep_mw_c42f62bd-bea4-474a-96a3-
f99843001dde
[masks,labels, scores, boxes] = segmentObjects(TrainedDetector,test_img);

%% visualize test
overlayeredImage = insertObjectMask(test_img,masks);
figure(2)
imshow(overlayeredImage)
hold on
showShape("rectangle",boxes,"Label",labels,'LineColor',[1,0,0])
```

Mask_RCNN_Testing.m

```
clc;
clear;
%this file is named validation but it is more for testing purposes of the
%network
%% INPUTS
% Add the source directory to the path (add maskrcnn main functions)
addpath('B:\OneDrive - CHP\Σχολή μαθήματα\10. Diplomatikh\matlab code\Mask
CNN\mask-rcnn-main\src')
% Set the root directory for COCO API:
cocoAPIDir = 'B:\OneDrive - CHP\Σχολή μαθήματα\10. Diplomatikh\matlab
code\Mask CNN\MatlabAPI';
% Add the API directory to the path
addpath(cocoAPIDir);
h= 1038; %height of the images
w=1353; %width of the images
imageSize = [h w 3]; %height-width
%insert trained detector directory
Trained_detector_dir = 'B:\OneDrive - CHP\Σχολή μαθήματα\10.
Diplomatikh\matlab code\Mask CNN\TrainedDetectors\test7.2.mat';
%inert validation datastore directory
ValidationDS_dir ='B:\OneDrive - CHP\Σχολή μαθήματα\10. Diplomatikh\matlab
code\Kinect\CleanImages\HighQuallity\trainDS.mat';
%load cell array with item_names
load('B:\OneDrive - CHP\Σχολή μαθήματα\10. Diplomatikh\matlab code\Item
images\v2\Testing\item_names.mat')
%load trained detector
load(Trained_detector_dir);
%load validation datastore
load(ValidationDS_dir);

% Annotations = trainDS.UnderlyingDatastores{1,1}.Files;
% image_number = size(Annotations,1);

%% Validating-Testing
options = trainingOptions("sgdm",  ...
%https://www.mathworks.com/help/deeplearning/ref/trainingoptions.html %
    InitialLearnRate=0.00000001, ... %for testing set Initial Learn Rate to
10^-10 and epochs to 1 and verbose frequency to 1 thet take the average of
the results
    Momentum=0.9, ...
    LearnRateSchedule="piecewise", ...
    LearnRateDropPeriod=1, ...
    LearnRateDropFactor=0.95, ...
    Plot="none", ...
    MaxEpochs=1, ...
    MiniBatchSize=2, ...
    BatchNormalizationStatistics="moving", ...
    ResetInputNormalization=false, ...
    ExecutionEnvironment="gpu", ...
    VerboseFrequency=1);
%use the same properties in trainMaskRCNN as used in actual training
[TrainedDetector,info] =
trainMaskRCNN(trainDS,TrainedDetector,options,'NumRegionsToSample',128,'Num
StrongestRegions',1000,'PositiveOverlapRange',[0.8,1],'NegativeOverlapRange
',[0.1  0.8]); %
https://www.mathworks.com/help/vision/ref/trainmaskrcnn.html
%FreezeSubNetwork="backbone"
```

```
%total error is the summ of errors from last epoch devided by the total
%amount of evaluations
TotalLoss = 0;
RPNLoss = 0;
RMSE = 0;
MaskLoss = 0;

testing_size = size(info,1);
for n =1:testing_size

TotalLoss=info(n).TrainingLoss/testing_size+TotalLoss;
RPNLoss = info(n).TrainingRPNLoss/testing_size+RPNLoss;
RMSE = info(n).TrainingRMSE/testing_size+RMSE;
MaskLoss=info(n).TrainingMaskLoss/testing_size+MaskLoss;

end
```

UE_Data_Creation.m

```
clc;
clear;
%CHP Functions
addpath('B:\OneDrive - CHP\Σχολή μαθήματα\10. Diplomatikh\matlab code\Mask
CNN\CHP_Functions');
%Initiate image numbering
% i=1;
%Add Simulink Path
simulink_dir='B:\OneDrive - CHP\Σχολή μαθήματα\10. Diplomatikh\matlab
code\Testing_Images\UE_Pictures.slx';
%Initial Color of Items
InitialColor = [0.5,0,0.009];
%Segmentation Color of Items
Color1_seg =[1,0,0];
Color2_seg=[0,1,0];
Color3_seg=[0,0,1];
%Depth sensor resolution
D_res = [340,417];
%RGB Save Location
RGB_dir = 'B:\OneDrive - CHP\Σχολή μαθήματα\10. Diplomatikh\matlab code\UE
Generated Pictures\High Res\RGB';
%Depth Save Location High res
Depth_dir_H = 'B:\OneDrive - CHP\Σχολή μαθήματα\10. Diplomatikh\matlab
code\UE Generated Pictures\High Res\Depth';
%Depth Save Location low res
Depth_dir_L='B:\OneDrive - CHP\Σχολή μαθήματα\10. Diplomatikh\matlab
code\UE Generated Pictures\Low Res\Depth';
%Annotation Folder
AnnotationFolder = 'B:\OneDrive - CHP\Σχολή μαθήματα\10. Diplomatikh\matlab
code\UE Generated Pictures\High Res\Annotations';

%% Take initial Pictures to save
Color1 = InitialColor;
Color2= InitialColor;
Color3= InitialColor;

out = sim(simulink_dir,0.1);
%RGB image:
RGB = out.RGB.signals.values;
%Depth image:
D = out.D.signals.values*1000;

% figure(1)
% imshow(RGB);
% figure(2)
% imshow(D)
imageFile = strcat(num2str(i),'.png');
rgb_h_dir = strcat(RGB_dir,'\',imageFile);
imwrite(RGB,rgb_h_dir);

d_c = imresize(D,D_res);
d_l_dir = strcat(Depth_dir_L,'\',num2str(i));
save(d_l_dir,"d_c");

%Segmentation
Color1 = Color1_seg;
```

```
Color2= Color2_seg;
Color3= Color3_seg;

out = sim(simulink_dir,0.1);
%RGB image:
rgb_seg = out.RGB.signals.values;
figure(3)
imshow(rgb_seg);

Mask1 = Image_to_Mask(rgb_seg,Color1*256);
Mask2 = Image_to_Mask(rgb_seg,Color2*256);
Mask3 = Image_to_Mask(rgb_seg,Color3*256);
% figure(4)
% montage({Mask1,Mask2,Mask3});

labels = categorical(["PT1";"PT2";"PT3"]);
masks = false([size(Mask1),3]);
masks(:,:,1)=Mask1;
masks(:,:,2)=Mask2;
masks(:,:,3)=Mask3;
boxes = zeros(3,4);
boxes(1,:)=BoundingBox_From_Mask(Mask1);
boxes(2,:)=BoundingBox_From_Mask(Mask2);
boxes(3,:)=BoundingBox_From_Mask(Mask3);

% visualise an image with its bounding box
%load("1.mat")
overlayeredImage = insertObjectMask(RGB,masks);
figure(2)
imshow(overlayeredImage)
hold on
showShape("rectangle",boxes,"Label",labels,'LineColor',[1,0,0])
%Save Annotation File
save(strcat(AnnotationFolder,'\',num2str(i)),"imageFile","boxes","labels","
masks");
i=i+1;
```

II. Συναρτήσεις MATLAB

best_sol.m

```
function [Final_Config] = best_sol(Initial_Config,Final_Configs)
%This function finds the "best" solution for the end configuration of a
% robotic arm based on angle rotations.
[num_of_sol,~] = size(Final_Configs);
minimum_movement=10000;
for n=1:num_of_sol
    movement = sum((Initial_Config-Final_Configs(n,:)).^2); %summed squered
error
    if movement<minimum_movement
        minimum_movement=movement;
        best_ik_sol=n;
    end
end
Final_Con = Final_Configs(best_ik_sol,:);
Final_Config = [Final_Con,0,0];
end
```

BoundingBox_From_Mask.m

```
function [Box] = BoundingBox_From_Mask(mask)
%This function creates a bounding box from a mask
props=regionprops(mask,'BoundingBox');
bboxes=cell2mat(struct2cell(props)');
bboxes(:,3)= bboxes(:,1)+bboxes(:,3);
bboxes(:,4)= bboxes(:,2)+bboxes(:,4);
x=min(bboxes(:,1));
y=min(bboxes(:,2));
w=max(bboxes(:,3))-x;
h=max(bboxes(:,4))-y;
Box=[x,y,w,h];
end
```

BoundingBox_From_Polygon.m

```
function [BD] = BoundingBox_from_Polygon(x,y)
%This function create a bounding box from a polygon
upper_y = floor(min(y)); %as it is in images y is from top to buttom
lower_y = ceil(max(y));
left_x = floor(min(x));
right_x = ceil(max(x));
w = right_x-left_x;
h=lower_y-upper_y;
BD = [left_x,upper_y,w,h];
end
```

BW_circle.m

```
function [out] = BW_circle(c,r,res)
% This function creates a binarry image of size res =[h,w] with a circle of
% radius r in pixels and a center at c = [y,x]
out = false(res);
```

```
for i=1:res(1)
    for ii=1:res(2)
        dist = sqrt((ii-c(2))^2+(i-c(1))^2);
        if dist<=r
            out(i,ii)=1;
        end
    end
end
end
```

CHP_Blend.m

```
function [out] = CHP_Blend(Background,img,mask)
%This function inserts an image on top of the background image as specified
%by the mask (alpha channel).
if size(Background)==size(img)
    H=size(Background,1);
    W = size(Background,2);
    out=Background;

    for n=1:H
        for nn=1:W
            if mask(n,nn)==1
                out(n,nn,:)=img(n,nn,:);
            end
        end
    end

else
end

end
```

DrawBW_line.m

```
function [BW1] = DrawBW_line(c1,c2,res)
%This function creates a binary image containing a line prom point c1 to
%point c2
%initial point c1 = [x,y] and ending point c2=[x,y];
%res = [h,w] the resolution of the image
c1=[ceil(c1(1)),ceil(c1(2))];
c2=[ceil(c2(1)),ceil(c2(2))];
dp = c2-c1;
a=dp(2)/dp(1);% dy/dx
BW=false(res);
x_pixels = dp(1);
if x_pixels>=0
for x=1:abs(x_pixels)
    y=a*x+c1(2);
    BW(ceil(y),x+c1(1))=1;
end
else
for x=1:abs(x_pixels)
    y=a*x+c2(2);
    BW(ceil(y),x+c2(1))=1;
end
end
se=strel('rectangle',[2,2]);
BW1 = imdilate(BW,se);
end
```

DrawBW_lineA.m

```
function [BW] = DrawBW_lineA(c,a,res)
% This function creates a binary images with a line  passing from point
% c=[x,y] with an angle of a.

BW=false(res);
A=tan(a);

b=c(2)-A*c(1);
if A>1 %if the angle is greater than 45deg use the form x=(y-b)/A
for i=1:res(1)
    xx=round((i-b)/A);
    if xx>0 && xx<=res(2)
        BW(i,xx)=1;
    end
end
else%if the angle is smaller than 45deg use the form y=Ax+b
for i=1:res(2)
    yy=round(A*i+b);
    if yy>0 && yy<=res(1)
        BW(yy,i)=1;
    end
end
end
```

Grab_selection.m

```
function [center,orientation,simplez,Pick_Item_Index,Gripper1,Gripper2] =
Grab_Selection(masks,depth,occlusion,f)

%% Select a part (the one that is not occluded but also is the tallest)
[height,background_height] = Height_of_Objects(masks,depth);
items_in_image = size(masks(1,1,:),3);
BiggestHeight =-10;
for i =1:items_in_image
    if occlusion(i)==0 && height(i)>BiggestHeight
        Pick_Item_Index=i;
        BiggestHeight=height(i);
    end
end

%% Get orientation of the part
mask = masks(:,:,Pick_Item_Index);
props = regionprops(mask,"Centroid","Orientation");
center = props(1).Centroid;
orientation=props(1).Orientation/180*pi; % angles in degs from x axis [-
90,90] positive with right hand rule with z towards us from the screen
simplez = (background_height-BiggestHeight+10)/1000; % to get the z
distance from item

%% Get Grpipper positions
res=size(mask);
%Get the perimeter of the mask
perim = bwperim(mask);
se = strel('rectangle',[2,2]);
%dilate it by 2
perim = imdilate(perim,se);
%Daw a line where the gripper is going to grab the part
```

```
Line =DrawBW_lineA(center,orientation-pi/2,res);
%Find the two grasping positions
Grab_BW =and(Line,perim);
% To get the correct gripper to move we proceed with the following:
R=R_z(pi-orientation);
T=eye(4);% tranfrorm from image coordinate system to item coordinate system
T(1:3,1:3)=R;
T(1:2,4)=center';
%b-> image coordinate system, %c-> item coordinate system
T_c_b = inv(T);
Grab_props=regionprops(Grab_BW,"Centroid");
Grab_locs = struct2array(Grab_props); %[x1,y1,x2,y2]
c1=Grab_locs(1:2);
c2=Grab_locs(3:4);
c1_new=T_c_b*[c1,0,1]'; %y coordinate indicates the distance needed to move
from the center of the item to the grabbing location
c2_new=T_c_b*[c2,0,1]';
Dist_of_gripper_from_center=0.085;
if c1_new(2)>0 %Gripper is located in the positive direction while gripper
2 is in the negative
Gripper1=Dist_of_gripper_from_center-c1_new(2)/f*simplez;%distance that
gripper needs to move for visulization
Gripper2=Dist_of_gripper_from_center-abs(c2_new(2)/f*simplez);
else
Gripper1=abs(Dist_of_gripper_from_center-c2_new(2)/f*simplez);%distance
that gripper needs to move for visulization
Gripper2=abs(Dist_of_gripper_from_center-abs(c1_new(2)/f*simplez));
end
end
```

Height_of_Objects.m

```
function [height,background_height] = Height_of_Objects(masks,depth)
%This function calculates the average height of an object specified by a
%mask and calculates the background height (where there are no objects).
res = size(masks(:,:,1));
items_in_image = size(masks,3);
background_mask= false(res);

for ii = 1:items_in_image % find backround height
        background_mask = or(masks(:,:,ii),background_mask);
end

background_mask = imcomplement(background_mask);
background_mask_height = double(background_mask).*double(depth);
background_area = sum(background_mask,"all");
background_height = sum(background_mask_height,"all")/background_area;
height = zeros(items_in_image,1);
for ii =1:items_in_image
    Area = sum(masks(:,:,ii),"all");
    mask_depth = double(masks(:,:,ii)).*double(depth);
    height(ii) =background_height-sum(mask_depth,"all")/Area;
end

end
```

Occlusion_Correction.m

```
function [masks,bboxes,labels] =
Occlusion_Correction(masks,bboxes,labels,i,min_occlusion_pixel_area)
loop = size(i,2);

for n =1:(loop-1)
    for nn=1:n
        masks(:,:,i(end-n)) = masks(:,:,i(end-
n)).*imcomplement(masks(:,:,i(end-nn+1)));
    end

end
ii=0;
while n <= loop-ii
    S = sum(masks(:,:,n),'all');
    if S<min_occlusion_pixel_area % it can be altered
        masks(:,:,n)=[];
        labels(n)=[];
        bboxes(n,:)=[];
        n=n-1;
        ii=ii+1;
    else
        stats = regionprops(masks(:,:,n),'BoundingBox');
        bb= stats.BoundingBox;
        if size(bb,1)==1
            bboxes(n,:)=bb;
        else
            bboxes(n,1)=min(bb(:,1));
            bboxes(n,2)=min(bb(:,2));
            % find the maximum x from all the regions that represent the
item
            xmax = max(bb(:,1)+bb(:,3));
            ymax = max(bb(:,2)+bb(:,4));
            bboxes(n,3) = xmax-bboxes(n,1);
            bboxes(n,4) = ymax-bboxes(n,2);
        end
    end
n=n+1;
end

end
```

Occlusion_Detector.m

```
function [occlusion] = Occlusion_Detector(net1,net2,net3,d_c,masks,labels)
%this function utilizes the occlusion networks to find occlusion properties
num_of_objects_detected = size(labels,1);
if num_of_objects_detected==1
    occlusion=0;
elseif num_of_objects_detected>1
    [Features,Features_labels] = Occlusion_Features(d_c,masks,labels);
    occlusion = false(num_of_objects_detected,1);
    for i=1:num_of_objects_detected
        if Features_labels(i,1)=='PT1'
            occlusion(i,1)=round(net1(Features(i,:)'),0);
        elseif Features_labels(i,1)=='PT2'
            occlusion(i,1)=round(net2(Features(i,:)'),0);
        else
            occlusion(i,1)=round(net3(Features(i,:)'),0);
        end
    end
else
    occlusion = [];
end

end
```

Occlusion_Features.m

```
function [DATA,DATA_labels] = Occlusion_Features(d_c,masks,labels)
%This functionextracts the features for the detection of occlusion
%d_c is the depth image
%mask is the masks on the image
%labels are the labels of the masks

%radius of intersection circle for local area height
r_intersection = 10;

%the ammount of objects-items in an image is equal to the ammount of
%labels
items_in_image = size(labels,1);
% find the resolution of the images
res = size(masks(:,:,1));
%% Generate the background mask and find its height
background_mask= false(res);
if items_in_image<=1 % if there is only one item then there is no occlusion
    %do not create any data
else
    for ii = 1:items_in_image % find backround height
        background_mask = or(masks(:,:,ii),background_mask);
    end
    background_mask=imcomplement(background_mask);
    background_mask_height = double(background_mask).*double(d_c);
    background_area = sum(background_mask,"all");
    background_height = sum(background_mask_height,"all")/background_area;
    %% Calculate the distance between all the items-objects in an image
    % Find minimum distance between objects
    %valuable info from:
https://nl.mathworks.com/matlabcentral/answers/91046-how-to-find-distance-
in-binary-image
    % gennerate a matrix with very high diagonal values
```

```
    %(maximum distance between objects is set to 100 pixels)
    min_distance_matrix = eye(items_in_image)*100;
    %matrix of cells containint [x,y] coordinates of pixels, to keep track
    %of the pixel positions that have the smallest distance between the
objects
    min_distance_index = cell(items_in_image);
    %find the local height at the closest points between objects
    local_height_matrix = eye(items_in_image)*1000;
    pixeldistance_mask=zeros(res(1),res(2),items_in_image);
    mask_perimeter = false(res(1),res(2),items_in_image);
    for ii = 1:items_in_image
        pixeldistance_mask(:,:,ii) = bwdist(masks(:,:,ii));
        mask_perimeter(:,:,ii) = bwperim(masks(:,:,ii));

    end
    for ii= 1:items_in_image
        for iii =1:items_in_image
            if ii~=iii
                %Distance of item iii from item ii
                perimeter_distance_ii_iii =
pixeldistance_mask(:,:,ii).*mask_perimeter(:,:,iii);
                % create a matrix with very high values where there is no
perimeter
                % (to add it to the distance so when I take the minimum I
take the distance and not 0)
                perimeter_inverse =
inverse_BW_CHP(mask_perimeter(:,:,iii))*1000000;
                Distances_matrix =
perimeter_distance_ii_iii+perimeter_inverse;
                %this gets the minimum from all columns
                [a,I]=min(Distances_matrix,[],1);
                %this gets the mimimum from the row (gets index as well).
                [min_distance_matrix(ii,iii),II]=min(a);
                %get the indexes to a matrix (pixel where there is the
closes distance)
                min_distance_index{ii,iii} = [I(II),II];
                % Gennerate a binary image with resolution res witch
contains a
                % circle with radius r and center at the closest point
between two objects
                circle =
BW_circle(min_distance_index{ii,iii},r_intersection,res);
                % find the local height of item ii in the closest point to
item iii.
                circle_intersection= and(masks(:,:,ii),circle);
                %find the area of intersection
                circle_intersection_area = sum(circle_intersection,'all');
                % gennerate a metrix containing the local heights between
objects
                local_height_matrix(ii,iii)=background_height-
sum(double(circle_intersection).*double(d_c),"all")/circle_intersection_are
a;

            end
        end
    end
    local_height_difference_matrix = eye(items_in_image)*1000;
    for ii=1:items_in_image
        for iii=1:items_in_image
```

```
            if iii~=ii
                local_height_difference_matrix(ii,iii)=
local_height_matrix(ii,iii)-local_height_matrix(iii,ii);
            end
        end
    end
    for ii = 1:items_in_image

        [closest_object_distance,closest_object_index] =
min(min_distance_matrix(ii,:));
        props =
regionprops(masks(:,:,ii),'Area','MajorAxisLength','MinorAxisLength','Perim
eter');

        Area = props(1).Area;
        Perimeter = props(1).Perimeter;
        aspect_ratio = props(1).MajorAxisLength/props(1).MinorAxisLength;
        mask_depth = double(masks(:,:,ii)).*double(d_c);

        avg_height =background_height-sum(mask_depth,"all")/Area;

        %hiest_point = background_height-min(smothed_mask_depth,[],'all');
        Openned_Area = Area/background_height^2;
        Openned_Perimeter = Perimeter/background_height;
        %find the minimum height ddifference of object ii in terms of all
        %other objects (for occluded objects it might be negative)
        min_height_difference =
local_height_difference_matrix(ii,closest_object_index);%min(local_height_d
ifference_matrix(ii,:));
        if isnan(min_height_difference)
            min_height_difference=0;
        end
        DATA(ii,:) =
[Openned_Area,avg_height,Openned_Perimeter,aspect_ratio,closest_object_dist
ance,min_height_difference];
        DATA_labels(ii,1)=labels(ii,1);
    end
end
end
```

R_z.m

```
function [R] = R_z(t)
%Rotation matrix when a coordinate system its rotated by t rad in the z
%axis
c=cos(t);
s=sin(t);
R=[c,-s,0;
    s,c,0;
    0,0,1];
end
```

ResizeImageMasksBoxes.m

```
function [im_new,masks_new,boxes_new] =
ResizeImageMasksBoxes(im,masks,boxes,newres)
%Resize the image,masks and boxes to new resolution specified by newres

res=size(im,1,2);
resize_ratio=newres./res;
boxes_new = boxes;
items_on_image = size(masks,3);
masks_new = false([newres,items_on_image]);
for ii=1:items_on_image
    boxes_new(ii,1)=round(boxes(ii,1)*resize_ratio(2));
    boxes_new(ii,3)=round(boxes(ii,3)*resize_ratio(2));
    boxes_new(ii,2)=round(boxes(ii,2)*resize_ratio(1));
    boxes_new(ii,4)=round(boxes(ii,4)*resize_ratio(1));
    masks_new(:,:,ii)=imresize(masks(:,:,ii),newres);
end
    im_new = imresize(im,newres);
end
```

UEtoMATLABtransfrom.m

```
function T = UEtoMATLABtransform(L,eul)
%Ureal Engine to MATLAB coordinate system tranfromation
lsgn = [1 -1 1];
rsgn = [1 -1 -1];
T=eye(4);
T(1:3,4)=(L.*lsgn)';
eul=eul.*rsgn;
T(1:3,1:3)=eul2rotm(eul,"XYZ");
end
```

Kinect_RGBtoDepthMap.m

```
function [rgb_cropped_fix,d_cropped_fix] =
Kinect_RGBtoDepthMap(RGB_im,D_im)
%This function maps the rgb image to the depth image
%resolutions
Pixel_Ratio = 3.05; %depth pixels are 3 times larger than rgb pixels
Scale_Ratio = 2; %How much bigger ratio should the rgb image have
%the center distance value in the x direction changes!!
c_x = 0; %pixels in depth image plane(move it towards the depth plane)
c_y = -15;%pixels in depth image plane
f = 365; %focal length of D_camera fx = fy (almost);
cx = 256; %center point of depth camera;
cy = 212; %center point of depth camera;
K_depth = [f,0,cx;
    0,f,cy;
    0,0,1];
RadialDistortion_depth = [0.2,0]; % it specifies an elipse(rx ,ry)
side_cut = 35; % is an additional cropping in pixels from each side that
needs to be done in order to compensate for the distortion correction;
% x = ii-0.0000500604*bg_depth^2-0.1078012098*bg_depth+68.4597461493;
%% Fix distortion of depth image
depth_params =
cameraParameters('K',K_depth,'RadialDistortion',RadialDistortion_depth);
D_im = undistortImage(D_im,depth_params);
res_d = size(D_im);
boundingBox = [side_cut,side_cut,floor(res_d(2)-2*side_cut),res_d(1)-
2*side_cut];
D_im = imcrop(D_im,boundingBox);
res_d = size(D_im);

%principal points
% principal_rgb = res_rgb/2;
% principal_d = res_d/2;
%
% res_ratio = res_rgb./res_d;
RGB_im = imresize(RGB_im,1/Pixel_Ratio);

res_rgb = size(RGB_im,[1,2]);
RGB_im = imtranslate(RGB_im,[c_x,c_y]); % move the rgb image to the depth
center
boundingBox = [0,0,floor(res_rgb(2)-abs(c_x)),floor(res_rgb(1)-abs(c_y))];%
Becarefull!!!! only if c_x and c_y are negative this works
% Crop the translated image to remove empty areas
RGB_im = imcrop(RGB_im, boundingBox);
res_rgb = size(RGB_im,[1,2]);
new_image_res = [min(res_rgb(1),res_d(1)),min(res_rgb(2),res_d(2))];
%355x512
x_diff = ceil((res_rgb(2)-res_d(2))/2);
y_diff = ceil((res_d(1)-res_rgb(1))/2);

rgb_cropped = imcrop(RGB_im, [x_diff,1,new_image_res(2)-
1,new_image_res(1)]);
d_cropped = imcrop(D_im, [1,y_diff,new_image_res(2),new_image_res(1)-1]);
% d_cropped = imgaussfilt(d_cropped,2);

%% Fix depth pixels to rgb mapping issue in x direction
cropped_res = size(d_cropped);
d_cropped_f = d_cropped;
```

```
for i=1:cropped_res(1)
    for ii = 1:cropped_res(2)
        if d_cropped(i,ii)==0
            %do nothing (should probably fill the area with the average of
the perimeter)
        else
            d_at_pos = double(d_cropped(i,ii));
            dx = (0.0000500604*d_at_pos^2 - 0.1078012098*d_at_pos +
68.4597461493);
            x = ii-dx;
            if x>=1 && x<=cropped_res(2)
                d_cropped_f(i,ceil(x))=d_cropped(i,ii);
                d_cropped_f(i,floor(x))=d_cropped(i,ii);
            end
        end
    end
end
%crop again both images
rgb_cropped_fix = imcrop(rgb_cropped,[1,1,cropped_res(2)-
27,cropped_res(1)]);
d_cropped_fix = imcrop(d_cropped_f,[1,1,cropped_res(2)-27,cropped_res(1)]);
end
```

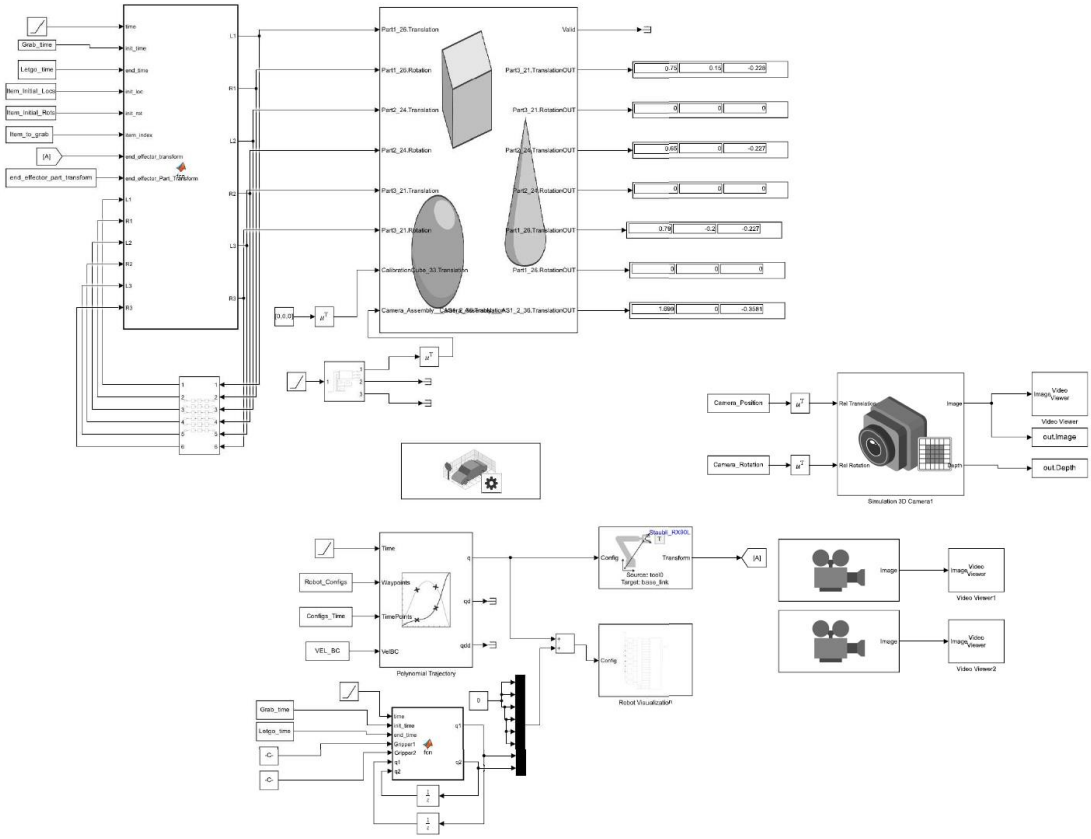
KinectPicture.m

```
function [img_rgb,img_d] = KinectPicture(colorDevice,depthDevice)
%This function takes a picture using the Kinect
step(colorDevice);
step(depthDevice);
step(depthDevice);
img_rgb = step(colorDevice);
img_d=step(depthDevice);
img_rgb = flip(img_rgb,2);
img_d = flip(img_d,2);

end
```

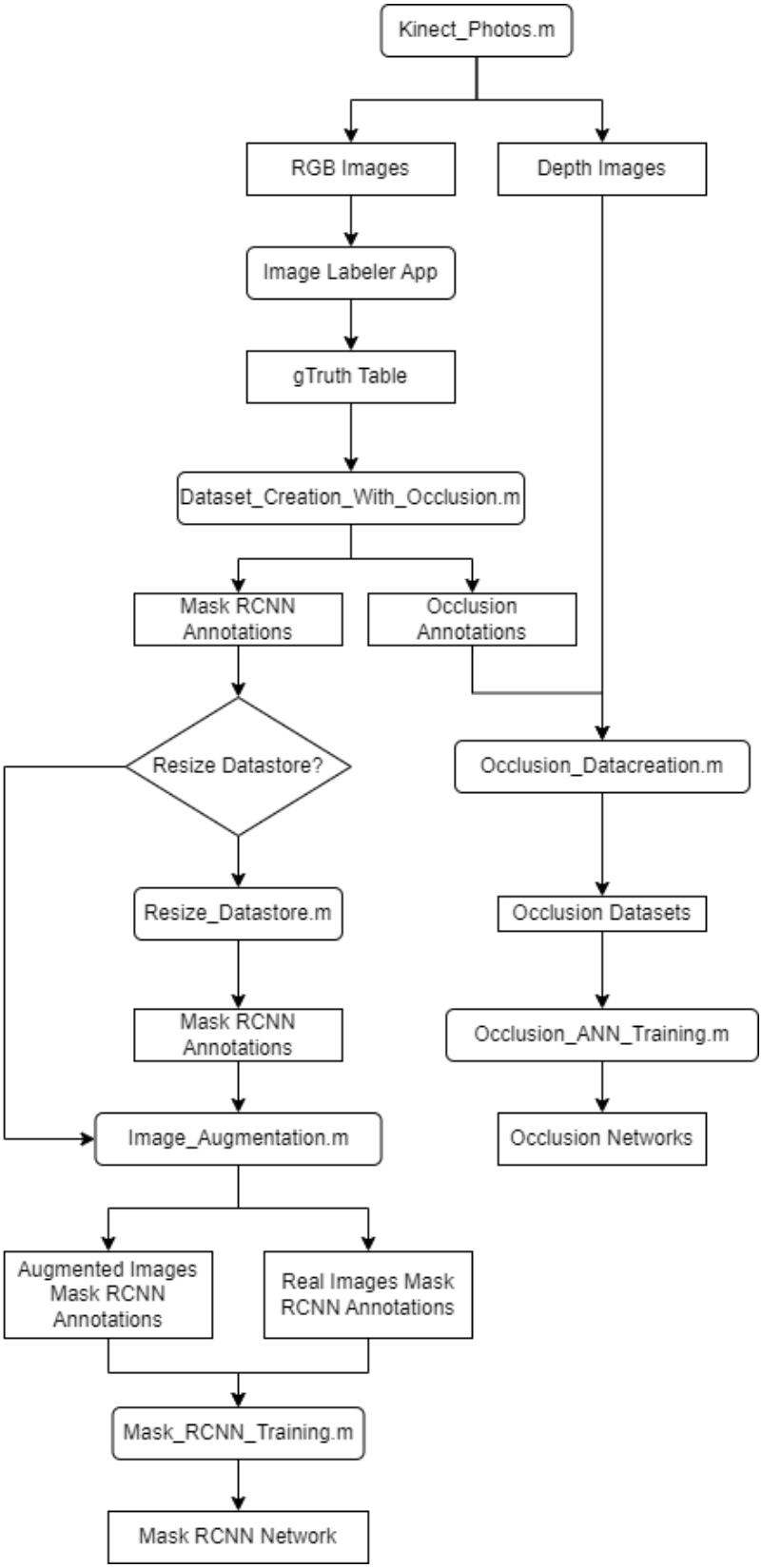
Uint16_to_uint8.m

```
function [out] = uint16_to_uint8(d)
%This function converts a depth image for visulization
max_z = max(d,[],'all');
dd = d;
dd(dd==0)= max_z;
min_z = min(dd,[],'all');

out = uint8(ceil((double(dd-min_z))*255/double(max_z-min_z)));
end
```

III. Μοντέλο Simulink



IV. Μεθοδολογία Εκπαίδευσης Mask RCNN & ANNs



V. Χαρακτηριστικά Υπολογιστή & Προγραμμάτων

CPU	AMD Ryzen 9 5900X @4.2GHz

RAM	64GB ddr4 @3600MHz
GPU	NVIDIA GeForce RTX 3090

MATLAB Version 2023a, update 3.

SOLIDWORKS 2023 SP3.

Blender 2.93.4.