ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Σχεδίαση Ρομποτικού Βραχίονα για Αναγνώριση και Συλλογή Αντικειμένων

Διπλωματική Εργασία

Ζέρβας Χρήστος

Επιβλέπουσα: Τσομπανοπούλου Παναγιώτα

Φεβρουάριος 2022

ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Σχεδίαση Ρομποτικού Βραχίονα για Αναγνώριση και Συλλογή Αντικειμένων

Διπλωματική Εργασία

Ζέρβας Χρήστος

Επιβλέπουσα: Τσομπανοπούλου Παναγιώτα

Φεβρουάριος 2022

UNIVERSITY OF THESSALY

SCHOOL OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

Robotic Arm Design for Object Recognition and Collection

Diploma Thesis

Zervas Christos

Supervisor: Tsompanopoulou Panagiota

February 2022

Εγκρίνεται από την Επιτροπή Εξέτασης:

Επιβλέπουσα **Τσομπανοπούλου Παναγιώτα**
Αναπληρώτρια Καθηγήτρια, Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών, Πανεπιστήμιο Θεσσαλίας

Μέλος **Βασιλακόπουλος Μιχαήλ**
Καθηγητής, Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών, Πανεπιστήμιο Θεσσαλίας

Μέλος **Φεύγας Αθανάσιος**
Ε.ΔΙ.Π, Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών, Πανεπιστήμιο Θεσσαλίας

Ευχαριστίες

Αρχικά, θα ήθελα να ευχαριστήσω την οικογένεια μου και κυρίως τη γυναίκα μου Δήμητρα για τη συμπαράσταση και τη δύναμη που μου έδωσαν σε όλα τα χρόνια της φοίτησης μου. Επίσης, τον δάσκαλο μου Θοδωρή για την ενθάρρυνση και τη στήριξη να εισέλθω στη συγκεκριμένη σχολή. Τέλος, την επιβλέπουσα καθηγήτρια κα. Τσομπανοπούλου Παναγιώτα για την εμπιστοσύνη και τη καθοδήγηση της ώστε να έρθει εις πέρας η εργασία αυτή.

ΥΠΕΥΘΥΝΗ ΔΗΛΩΣΗ ΠΕΡΙ ΑΚΑΔΗΜΑΪΚΗΣ ΔΕΟΝΤΟΛΟΓΙΑΣ ΚΑΙ ΠΝΕΥΜΑΤΙΚΩΝ ΔΙΚΑΙΩΜΑΤΩΝ

«Με πλήρη επίγνωση των συνεπειών του νόμου περί πνευματικών δικαιωμάτων, δηλώνω ρητά ότι η παρούσα διπλωματική εργασία, καθώς και τα ηλεκτρονικά αρχεία και πηγαίοι κώδικες που αναπτύχθηκαν ή τροποποιήθηκαν στα πλαίσια αυτής της εργασίας, αποτελεί αποκλειστικά προϊόν προσωπικής μου εργασίας, δεν προσβάλλει κάθε μορφής δικαιώματα διανοητικής ιδιοκτησίας, προσωπικότητας και προσωπικών δεδομένων τρίτων, δεν περιέχει έργα/εισφορές τρίτων για τα οποία απαιτείται άδεια των δημιουργών/δικαιούχων και δεν είναι προϊόν μερικής ή ολικής αντιγραφής, οι πηγές δε που χρησιμοποιήθηκαν περιορίζονται στις βιβλιογραφικές αναφορές και μόνον και πληρούν τους κανόνες της επιστημονικής παράθεσης. Τα σημεία όπου έχω χρησιμοποιήσει ιδέες, κείμενο, αρχεία ή/και πηγές άλλων συγγραφέων, αναφέρονται ευδιάκριτα στο κείμενο με την κατάλληλη παραπομπή και η σχετική αναφορά περιλαμβάνεται στο τμήμα των βιβλιογραφικών αναφορών με πλήρη περιγραφή. Δηλώνω επίσης ότι τα αποτελέσματα της εργασίας δεν έχουν χρησιμοποιηθεί για την απόκτηση άλλου πτυχίου. Αναλαμβάνω πλήρως, ατομικά και προσωπικά, όλες τις νομικές και διοικητικές συνέπειες που δύναται να προκύψουν στην περίπτωση κατά την οποία αποδειχθεί, διαχρονικά, ότι η εργασία αυτή ή τμήμα της δεν μου ανήκει διότι είναι προϊόν λογοκλοπής».

Ο Δηλών

Ζέρβας Χρήστος

Διπλωματική Εργασία

Σχεδίαση Ρομποτικού Βραχίονα για Αναγνώριση και Συλλογή Αντικειμένων

Ζέρβας Χρήστος

Περίληψη

Στη συγκεκριμένη εργασία παρουσιάζεται ένας κινούμενος αυτόνομος ρομποτικός βραχίονας για αναγνώριση και συλλογή αντικειμένων. Η λειτουργία του βασίζεται σε ένα Raspberry Pi και μέσω επικοινωνίας με έναν υπολογιστή είναι ικανό να συλλέξει αυτόνομα κουτάκια αλουμινίου σε εσωτερικό χώρο. Μέσω μίας κάμερας και ενός ανιχνευτή αντικειμένων αναγνωρίζει τα κουτάκια αλουμινίου, κινείται προς αυτά και τα συλλέγει με τη δαγκάνα αυτόματα. Τα αποτελέσματα των δοκιμών δείχνουν τις περισσότερες φορές να επιτυγχάνει τη συλλογή του αντικειμένου, ενθαρρύνοντας έτσι τη περαιτέρω μελέτη και βελτίωση του συγκεκριμένου ρομποτικού βραχίονα. Επιπλέον, σε πραγματικές συνθήκες θα έχει τη δυνατότητα να αναγνωρίζει και να συλλέγει διάφορα αντικείμενα, επιτρέποντας με αυτό το τρόπο τη χρήση του σε ποικίλες εργασίες και στη μείωση της χρήσης ανθρώπινου δυναμικού σε απλά και επαναλαμβανόμενα καθήκοντα.

Λέξεις-κλειδιά:

Ρομποτικός βραχίονας, Ανίχνευση αντικειμένων, Αναγνώριση αντικειμένων, Βαθιά μάθηση

Diploma Thesis

Robotic Arm Design for Object Recognition and Collection

Zervas Christos

Abstract

This work presents a moving autonomous robotic arm for object recognition and collection. Its operation is based on a Raspberry Pi and through communication with a computer it is able to collect aluminium cans in indoor spaces. Through a camera and an object detector it recognizes the aluminium cans, moves towards them and collects them with a gripper autonomously. Testing results show that the object collection is successful most of the times, encouraging further study and improvement of the specific robotic arm. Furthermore in real conditions it will have the capability to recognize and collect different objects, allowing in this way its use in various tasks and reducing the use of human resources in simple and repetitive duties.

Keywords:

Robotic Arm, Object Detection, Object Recognition, Deep Learning

Πίνακας περιεχομένων

Κατάλογος σχημάτων

Κεφάλαιο 1

Εισαγωγή

Η εξέλιξη στη Ρομποτική, έχει οδηγήσει τα ρομπότ να εκτελούν απλές και επαναλαμβανόμενες εργασίες μέχρι πολύπλοκες και ακριβείς [1], είτε λειτουργώντας αυτόνομα είτε σε συνεργασία με τον άνθρωπο [2]. Ένα αυτόνομο κινούμενο ρομπότ, κινείται με μικρή ή καθόλου ανθρώπινη παρέμβαση και ακολουθεί μία καθορισμένη διαδρομή σε ένα περιβάλλον [3]. Χρειάζεται να γνωρίζει τη θέση και τη κατεύθυνση που έχει, το περιβάλλον στο οποίο βρίσκεται και να σχεδιάσει τη διαδρομή που θα ακολουθήσει, με τη χρήση αισθητήρων ή κάμερας [4]. Χρησιμοποιούνται σε αρκετές εφαρμογές όπως κατασκευή, αυτοματισμούς, ιατρική περίθαλψη, επιτήρηση, αναγνώριση, ψυχαγωγία και άλλα [5].

Ο συνδυασμός μηχανικής μάθησης και ρομποτικής είναι ένα θέμα το οποίο μελετάται ενεργά τα τελευταία χρόνια [6]. Μέσω της επεξεργασίας εικόνων, τα ρομπότ έχουν οπτική αντίληψη, αποφεύγουν εμπόδια, ανιχνεύουν και αναγνωρίζουν αντικείμενα [7]. Η ανίχνευση αντικειμένων είναι μία από τις κύριες εργασίες της όρασης υπολογιστών και χρησιμοποιείται ευρέως στο χειρισμό ρομπότ, στη λειτουργία αυτόνομων οχημάτων και άλλες εφαρμογές [8]. Με τη χρήση δικτύων βαθιάς μάθησης οι ανιχνευτές αντικειμένων εκπαιδεύονται σε συγκεκριμένες εικόνες και στη συνέχεια μπορούν να εντοπίζουν αντικείμενα σε νέες εικόνες που δεν έχουν ξαναδεί [9].

Η ευρεία χρήση των ρομπότ οφείλεται στη δυνατότητα να αντικαθιστούν ανθρώπους σε επικίνδυνα περιβάλλοντα και να βελτιώνουν τη παραγωγικότητα και την αποτελεσματικότητα [10]. Η χρήση ενός ρομποτικού βραχίονα για τον καθαρισμό παραθύρων σε μεγάλα ύψη, μειώνει το κίνδυνο πτώσης του προσωπικού και δίνει τη δυνατότητα να τα καθαρίζουν από ασφαλές περιβάλλον [11]. Μέσω ενός τηλεκατευθυνόμενου ρομπότ, παρέχονται οι υπηρεσίες περίθαλψης σε περιβάλλοντα με αυξημένο ρίσκο μετάδοσης ασθενειών, μειώνοντας

το κίνδυνο έκθεσης των υγειονομικών από μολύνσεις και λοιμώξεις [12]. Επίσης για την εξερεύνηση του Διαστήματος χρησιμοποιούνται ρόβερς [13] και του βυθού της θάλασσας υποβρύχιες ρομποτικές συσκευές [14].

Στη σύγχρονη γεωργία τα ρομπότ αποκτούν όλο και μεγαλύτερο ενδιαφέρον λόγω των οικονομικών οφελών και προοπτικών αγοράς [15]. Η επιτήρηση μεγάλων καλλιεργήσιμων εκτάσεων μπορεί να επιτευχθεί με τη χρήση ενός κινούμενου ρομπότ και μίας κάμερας, όπου ελέγχει την ανάπτυξη της σοδειάς απομακρυσμένα, στέλνοντας φωτογραφίες από το κάθε φυτό για περαιτέρω ανάλυση [16]. Ακόμη ένα παράδειγμα αποτελεί και η λειτουργία ρομποτικού βραχίονα για συγκομιδή μήλων, χρησιμοποιώντας δίκτυο βαθιάς μάθησης για ανίχνευση της θέσης του κάθε μήλου και υπολογισμός της διαδρομής του βραχίονα για να το συλλέξει [17].

Τα σκουπίδια έχουν αυξηθεί ραγδαία τα τελευταία χρόνια λόγω της παραγωγής αγαθών μίας χρήσης, φτάνοντας τα 2 δισεκατομμύρια τόνους κάθε χρόνο [18]. Η καθαριότητα των αστικών χώρων αποτελεί κυρίαρχη προτεραιότητα αρκετών Ευρωπαικών χωρών για τους πολίτες και την οικονομία τους [19]. Με τη χρήση τεχνικών μηχανικής μάθησης, όπως η ανίχνευση αντικειμένων, επιτυγχάνεται η επιτήρηση και η αναγνώριση σκουπιδιών με ελάχιστη ανθρώπινη παρέμβαση [9]. Έχουν αναπτυχθεί διάφορα ρομπότ για το καθαρισμό, όπως εκείνου του πλεούμενου συλλέκτη σκουπιδιών με δυνατότητα λειτουργίας, είτε αυτόνομα, είτε ελεγχόμενα μέσω σύνδεσης Bluetooth [20], όπως επίσης και ενός οχήματος για το καθαρισμό παραλιών και πάρκων, χρησιμοποιώντας ειδικά εξαρτήματα και GPS (Global Positioning System) για την αυτόνομη λειτουργία του [21].

Ένας κινούμενος ρομποτικός βραχίονας, μπορεί να εκτελέσει ποικίλες εργασίες έχοντας τη δυνατότητα να δρα ελεύθερα στο περιβάλλον εργασίας, δημιουργώντας νέες δυνατότητες [22]. Μελέτες έχουν υλοποιήσει τέτοια ρομπότ όπως εκείνο όπου ακολουθεί μία γραμμή και μέσω υπερηχητικών αισθητήρων καταλαβαίνει ότι υπάρχει αντικείμενο μπροστά του και το συλλέγει [23] και ενός άλλου όπου χρησιμοποιεί σύστημα GPS για τη πλοήγηση στο χώρο και δίκτυο βαθιάς μάθησης για την ανίχνευση σκουπιδιών και τη συλλογή τους σε ένα κάδο που είναι τοποθετημένο μαζί του [24].

Η συγκεκριμένη εργασία στοχεύει στην υλοποίηση ενός κινούμενου ρομποτικού βραχίονα ελεγχόμενο από ένα Raspberry Pi, όπου θα ανιχνεύει αντικείμενα και θα τα μαζεύει αυτόματα. Για να συμβάλει στο πρόβλημα των σκουπιδιών όπου αναφέρθηκε παραπάνω, το μοντέλο ανίχνευσης αντικειμένων εκπαιδεύτηκε να ανιχνεύει κουτάκια αλουμινίου. Επίσης

χρησιμοποιήθηκαν απλά υλικά με σκοπό τη διατήρηση του κόστους όσο πιο χαμηλά γίνεται. Η πλοήγηση του ρομπότ είναι προκαθορισμένη σε συγκεκριμένο χώρο, σε αντίθεση με τις παραπάνω μελέτες όπου χρησιμοποιούσαν GPS ή μία γραμμή για τη θέση και δεν υφίστανται άλλοι αισθητήρες εκτός μίας USB κάμερας, μέσω της οποίας εκτελείται η ανίχνευση και η καθοδήγηση για τη συλλογή.

Η εργασία διαρθρώνεται ως εξής. Μία σύντομη περιγραφή της εξέλιξης του ρομπότ παρουσιάζεται στο Κεφάλαιο 2. Το Κεφάλαιο 3 περιγράφει τη σχεδίαση και τη κατασκευή του ρομποτικού βραχίονα, καθώς και τη σύνδεση όλων των εξαρτημάτων για τη λειτουργία του. Το Κεφάλαιο 4 εισάγει τον αναγνώστη σε βασικές έννοιες που είναι απαραίτητες για τη κατανόηση της εργασίας, με έμφαση στην όραση υπολογιστών και τη μηχανική μάθηση. Η ρύθμιση και ο προγραμματισμός για τη λειτουργία, κίνηση του ρομποτικού βραχίονα και για την ανίχνευση αντικειμένων αναλύονται στο Κεφάλαιο 5. Τέλος στο Κεφάλαιο 6 αναφέρονται τα συμπεράσματα της εργασίας, τονίζοντας τις δυνατότητες και μελλοντικές επεκτάσεις που έχει ο συγκεκριμένος ρομποτικός βραχίονας, καθώς επίσης και οι δυσκολίες που συναντήθηκαν κατά τη διάρκεια της μελέτης.

Κεφάλαιο 2

Ιστορική Αναδρομή

2.1 Εισαγωγή

Η σύλληψη της ιδέας ενός μηχανήματος όπου θα εκτελεί βαριές και επαναλαμβανόμενες εργασίες χρονολογείται από την αρχαιότητα. Όμως με την εξέλιξη της τεχνολογίας μετά την Αναγέννηση, δόθηκε η δυνατότητα σε εφευρέτες να δημιουργήσουν αυτόνομες συσκευές [25].

Ο όρος «ρομπότ» προέρχεται από τη Τσέχική λέξη «Robota» που σημαίνει εξαναγκασμένη δουλειά και εμφανίστηκε το 1921 από τον Τσέχο συγγραφέα Karel Capek (1890-1938) στο έργο του Rossum Universal Robots [26].

Ένας επίσημος ορισμός της λέξης ρομπότ δόθηκε από το Ινστιτούτο Ρομποτικής της Αμερικής (Robot Institute of America, RIA), «Ένα ρομπότ είναι ένας επαναπρογραμματίσιμος, πολυλειτουργικός χειριστής, σχεδιασμένος να μετακινεί υλικά, εξαρτήματα, εργαλεία ή εξειδικευμένες μηχανές μέσω ποικίλων προγραμματισμένων κινήσεων για την απόδοση πληθώρων εργασιών» [27].

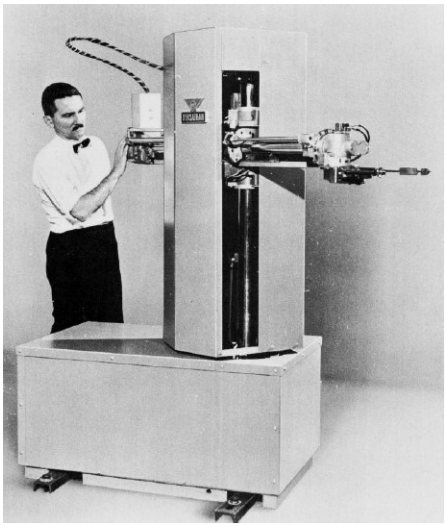
2.2 Εξέλιξη

Αν και ο αυτοματισμός εμφανίστηκε μετά τη βιομηχανική επανάσταση, η ευρεία χρήση των ρομποτικών μηχανημάτων ξεκίνησε στο βιομηχανικό τομέα τη δεκαετία του 1950. Η εξέλιξη τους μπορεί να χωριστεί σε 4 γενιές-περιόδους ανάλογα με τη τεχνολογία και τις δυνατότητες που είχαν.

Η πρώτη γενιά χρονολογείται από το 1950 μέχρι το 1967 όπου ελέγχονταν από ενεργητές

πεπιεσμένου αέρα και οι κινήσεις τους περιοριζόντουσαν από μηχανικά εμπόδια. Τα καθήκοντα που είχαν ήταν απλές κινήσεις χειρισμού. Σε αυτή τη περίοδο δημιουργήθηκαν και οι πρώτες εταιρείες κατασκευής ρομπότ όπου προμήθευαν αυτοκινητοβιομηχανίες στον τομέα παραγωγής. Χαρακτηριστικά ήταν τα Unimate ρομπότ (Σχήμα 2.1) της εταιρείας Unimation και χρησιμοποιήθηκαν από τη General Motors, καθώς επίσης και τα Versatran ρομπότ (Σχήμα 2.2) της εταιρείας AMF Coorporation όπου τα χρησιμοποίησε η Ford [28].

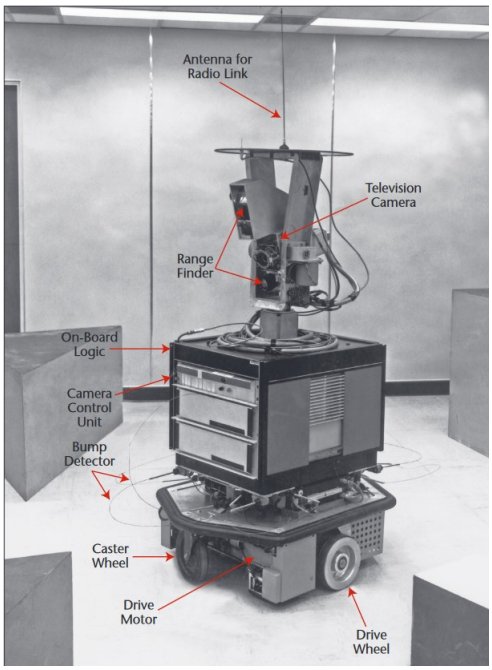
Σχήμα 2.1: Το ρομπότ Unimate [28].



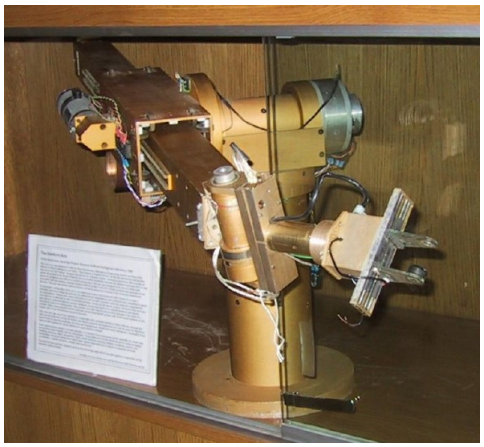
Σχήμα 2.2: Το ρομπότ Versatran [28].

Η 2η γενιά των ρομπότ (1968-1977) χαρακτηρίζεται από την ενσωμάτωση αισθητήρων, έχοντας ως αποτέλεσμα να αλληλοεπιδρούν με το περιβάλλον. Παράδειγμα αυτού αποτελεί το Shakey (Σχήμα 2.3) όπου είχε αισθητήρες αφης και κάμερα [29]. Επίσης ενσωματώθηκαν σέρβο χειριστές και ελεγχόντουσαν από PLC (Programmable Logic Controllers), ψηφιακούς υπολογιστές όπου μπορούσαν να επαναπρογραμματιστούν χωρίς να αλλάζει το υλικό [30]. Το 1969 ο Scheinman, ένας φοιτητής του Stanford Πανεπιστημίου, κατασκεύασε το Stanford

Arm (Σχήμα 2.4) ένα πρωτότυπο ρομπότ ελεγχόμενο από ηλεκτρικά μοτέρ. Σε σχέδιο του Scheinman βασίστηκε και η εταιρεία Unimation, όπου κατασκεύασε το Puma ρομπότ (Σχήμα 2.5) και για αρκετά χρόνια αποτελούσε πρότυπο ανθρωπομορφικού ρομπότ. Τέλος το 1974 κυκλοφόρησε και το πρώτο εμπορικά διαθέσιμο ρομπότ, T3 (Σχήμα 2.6) από τον Cincinnati Milacron [28].



Σχήμα 2.3: Το ρομπότ Shakey με τους αισθητήρες [29].



Σχήμα 2.4: Το ρομπότ Stanford [28].

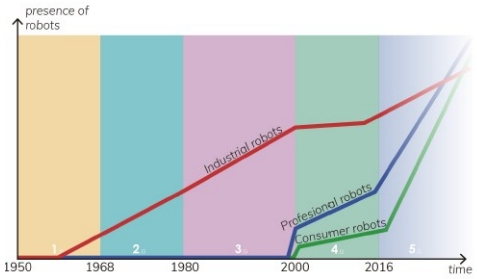
Η 3η γενιά (1978-1999) θεωρείται ως η εποχή των ρομπότ και χρησιμοποιούνταν ευρέως για πολλές δραστηριότητες. Κύριος στόχος όμως παρέμενε η αυτόματη και ακριβής επαναλαμβανόμενη κίνηση, στη κατασκευή και συναρμολόγηση στις βιομηχανίες. Τα ρομπότ εξοπλίστηκαν με ειδικούς υπολογιστές, δημιουργήθηκαν νέες γλώσσες προγραμματισμού για

Σχήμα 2.5: Το ρομπότ Puma [28].

Σχήμα 2.6: Το ρομπότ T3 [28].

τον έλεγχο τους και υπήρχε η δυνατότητα επαναπρογραμματισμού. Επίσης η εξέλιξη και άλλων τεχνολογιών όπως το Ethernet και το Linux βοήθησαν στη περαιτέρω ανάπτυξη των ρομποτικών κατασκευών. Τέλος στα τέλη του 1990 γεννήθηκε η ιδέα για χρήση ρομπότ εκτός βιομηχανικού τομέα και κατασκευάστηκαν αρκετά κιτ ρομπότ [31].

Συνεχίζοντας, στη 4η γενιά (2000-2017), κατασκευάζονται ρομπότ εξελιγμένα ως προς τον υπολογιστή, καθώς και στους αισθητήρες που ήταν εξοπλισμένα, επιτυγχάνοντας καλύτερη κατανόηση και δράση στις διάφορες καταστάσεις. Οι εταιρείες ξεκίνησαν να στοχεύουν στους καταναλωτές και να φτιάχνουν ρομπότ συσκευές όπου να βελτιώνουν τον τρόπο ζωής (Σχήμα 2.7). Σε πολλά σπίτια χρησιμοποιείται η ρομποτική σκούπα [32], γίνονται βελτιώσεις στην ασφάλεια λειτουργίας των ρομπότ, με αποτέλεσμα να συνεργάζονται με ανθρώπους στο ίδιο περιβάλλον. Επιπρόσθετα, δημιουργήθηκαν πλατφόρμες, όπως το Raspberry Pi [33], ένας υπολογιστής φτιαγμένος εξ ολοκλήρου σε μία πλακέτα (Single Board Computer),

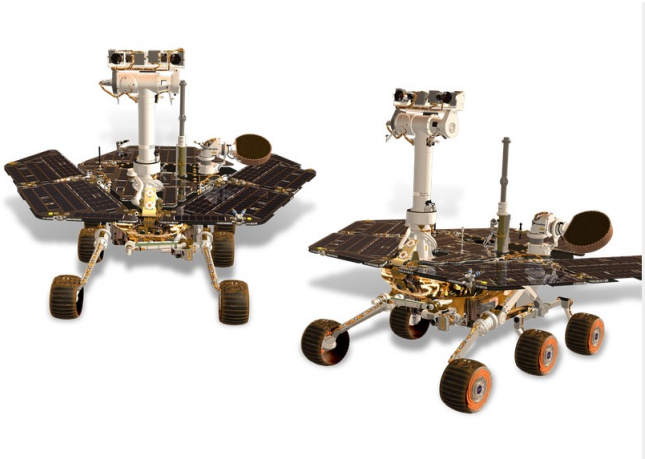


Σχήμα 2.7: Σύνοψη και πρόβλεψη διανομής των ρομπότ στο κόσμο στις 3 κύριες αγορές [31].

δίνοντας έτσι τη δυνατότητα στο καθένα, να φτιάξει ένα δικό του ρομπότ [31].

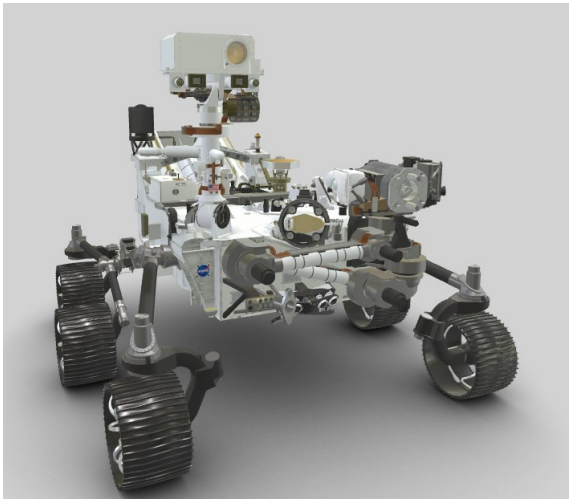
Τα τελευταία χρόνια, με τη χρήση αισθητήρων ακριβείας, μείωσης του κόστους των υλικών, και τη πρόοδο σε ασύρματες επικοινωνίες, αλγορίθμους και υλισμικό, τα ρομπότ χρησιμοποιούνται ευρέως σε πολλές εφαρμογές αλληλεπιδρώντας μαζί με ανθρώπους. Χαρακτηριστικά παραδείγματα αποτελούν τα αυτόνομα οχήματα καθώς επίσης και η χρήση των ρομπότ σε δυσπρόσιτα περιβάλλοντα, όπως τα βάθη των ωκεανών και το διάστημα [34].

Η NASA (National Aeronautics and Space Administration) έχει στείλει στο πλανήτη Άρη αρκετά ρομπότ για την εξερεύνηση του. Το 2003 προσγειώθηκαν δύο ρόβερ, Opportunity και Spirit (Σχήμα 2.8) για την αναζήτηση έμβιων όντων [35]. Το 2012 το ρόβερ Curiosity (Σχήμα 2.9) ξεκίνησε την αποστολή του, εάν ο πλανήτης Άρης είχε τις κατάλληλες συνθήκες για την επιβίωση έμβιων όντων [36]. Τέλος το 2021 προσγειώθηκε και το ρόβερ Mars 2020 Perseverance (Σχήμα 2.10), όπου αποστολή του είναι να συλλέξει πέτρες και απολιθώματα με στόχο να σταλθούν στη Γη [37].



Σχήμα 2.8: Το δίδυμο Opportunity - Spirit [35].

Σχήμα 2.9: Το ροβερ Curiosity [36].



Σχήμα 2.10: Το ρόβερ Mars 2020 Perceverance [38].

Κεφάλαιο 3

Ανάλυση του Ρομποτικού Βραχίονα

Στο συγκεκριμένο κεφάλαιο αναφέρονται τα υλικά που χρησιμοποιήθηκαν για την κατασκευή του ρομπότ, το κόστος αυτών και πως συνδέθηκαν μεταξύ τους.

3.1 Εξοπλισμός και εργαλεία

Για το συγκεκριμένο ρομποτικό βραχίονα, επιλέχθηκαν συγκεκριμένα υλικά ώστε να είναι πλήρως λειτουργικό και το κόστος όσο το δυνατόν πιο χαμηλό.

3.1.1 Υλικά

Τα υλικά από τα οποία αποτελείται ο ρομποτικός βραχίονας είναι τα εξής:

- Ξύλο διαστάσεων 25x17 cm
- Πλεξιγκλάς πάχους 3mm και διαστάσεων 25 cm (x2) και 20 cm (x2)
- Βίδες και παξιμάδια (x4)
- Μεταλλικές γωνίες (x4)
- Πλαστικές Ρόδες Arduino (x4)
- Μοτέρ Συνεχούς ρεύματος Arduino (x4)
- Δαγκάνα
- USB Κάμερα

- Σερβομηχανισμός MG995 (x3)
- Σερβομηχανισμός SG90
- Jumper Cables
- Raspberry Pi 4 μοντέλο B
- Οδηγός Μοτέρ (Motor Driver L298N)
- Πειραματική διάταξη (Breadboard)
- Καλώδια
- Δεματικά (Cable tie)
- Μονωτική ταινία

Τα εργαλεία που χρειάστηκαν για τη συναρμολόγηση και επεξεργασία των υλικών είναι τα εξής:

- Πολύμετρο
- Ηλεκτρικό Κολλητήρι
- Δραπανοκατσάβιδο
- Ηλεκτρικό πολυεργαλείο
- Πριόνι χειρός
- Ηλεκτρικό πιστόλι σιλικόνης

3.1.2 Κόστος

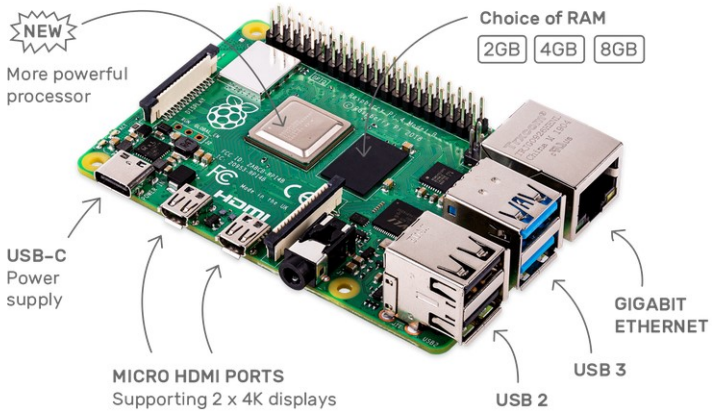
Το κόστος των υλικών του ρομποτικού βραχίονα έφτασε τα 150 ευρώ. Υπήρχε και η δυνατότητα να κατασκευαστεί βραχίονας με χρήση 3D εκτύπωσης, αλλά το κόστος θα ανέβαινε στα 230 ευρώ.

3.2 Κατασκευή του Ρομποτικού Βραχίονα

Παρακάτω περιγράφεται αναλυτικά η δομή του ρομποτικού βραχίονα. Αρχικά γίνεται μία αναφορά στα βασικά ηλεκτρονικά εξαρτήματα και στη συνέχεια περιγράφεται η μηχανική και ηλεκτρονική σύνδεση των εξαρτημάτων.

3.2.1 Raspberry Pi

Το Raspberry Pi (Σχήμα 3.1) είναι ένας υπολογιστής σε μέγεθος περίπου πιστωτικής κάρτας και δημιουργήθηκε το 2012. Το λειτουργικό σύστημα του είναι βασιζόμενο στο Linux και περιέχει ARM επεξεργαστή, κάρτα γραφικών, μνήμη RAM, GPIO (General Purpose Input/Output) και άλλες συνδέσεις για περιφερειακά [33]. Δεν έχει σκληρό δίσκο και για αυτό χρειάζεται micro SD κάρτα όπου θα έχει το λειτουργικό σύστημα και το χώρο αποθήκευσης. Έχουν κατασκευαστεί αρκετά μοντέλα ανάλογα με τη χρήση που επιθυμεί κάποιος και στο συγκεκριμένο ρομποτικό βραχίονα, χρησιμοποιήθηκε το Raspberry Pi 4 Model B με 4GB Ram .

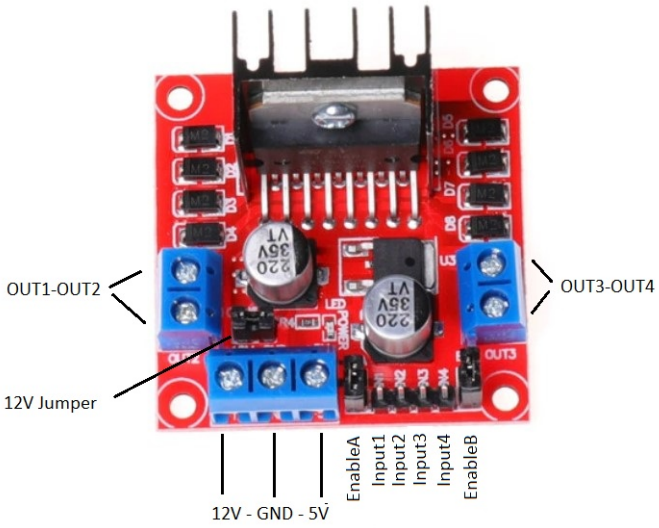


Σχήμα 3.1: Raspberry Pi 4 Model B [39].

3.2.2 Οδηγός L298N

Ο οδηγός μοτέρ L298N (Motor Driver Module) είναι μία μονάδα ελέγχου μοτέρ συνεχούς ρεύματος. Μπορεί να ελέγξει μέχρι 4 μοτέρ, αλλά 2 μοτέρ με έλεγχο ταχύτητας και κατεύθυνσης, δηλαδή να περιστρέφονται αριστερόστροφα ή δεξιόστροφα και αργά ή γρήγορα. Παρακάτω περιγράφονται αναλυτικά οι ακροδέκτες όπως φαίνονται στο Σχήμα 3.2

- Input 1 και Input 2. Χρησιμοποιούνται για τον έλεγχο της κατεύθυνσης του μότορα A
- Input 3 και Input 4. Χρησιμοποιούνται για τον έλεγχο της κατεύθυνσης του μότορα B
- EnA. Ενεργοποιεί το σήμα που δέχεται από το Raspberry Pi για το μότορα A
- EnB. Ενεργοποιεί το σήμα που δέχεται από το Raspberry Pi για το μότορα B
- OUT1-OUT2. Σύνδεση του μότορα A
- OUT3-OUT4. Σύνδεση του μότορα B
- 12V τροφοδοσία
- 5V τροφοδοσία
- GND. Γείωση

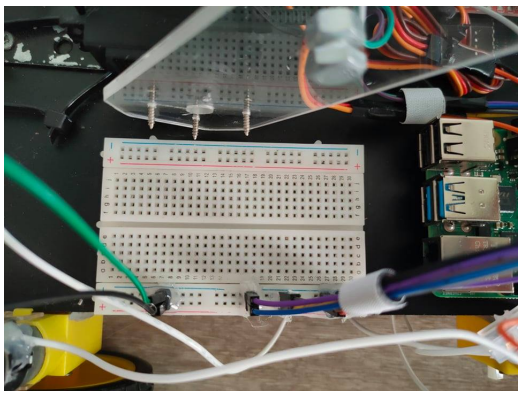


Σχήμα 3.2: L298N Motor Driver.

3.2.3 Πειραματική Διάταξη

Η πειραματική διάταξη (Breadboard) (Σχήμα 3.3 είναι μία πλακέτα όπου μας επιτρέπει να συνδέσουμε πολλά καλώδια οργανωμένα και γρήγορα. Συνήθως χρησιμοποιείται για τη αρχική και σωστή σύνδεση των καλωδίων και δεν είναι απαραίτητη η ύπαρξη του στη

συνέχεια, αλλά στο συγκεκριμένο βραχίονα διατηρήθηκε. Επίσης επιτρέπει και τη σύνδεση αντιστάσεων, πυκνωτών και άλλων εξαρτημάτων ανάλογα με τη περίσταση.



Σχήμα 3.3: Πειραματική διάταξη (Breadboard).

3.2.4 Σερβομηχανισμός

Ο σερβομηχανισμός είναι ένας μότορας όπου έχει τη δυνατότητα να στρέφεται σε συγκεκριμένη γωνία. Αποτελείται από μότορα συνεχούς ρεύματος, ηλεκτρικό κύκλωμα και από τρεις εισόδους (Σχήμα 3.4) . Ανάλογα το σήμα που δέχεται μεταφράζεται σε αντίστοιχη γωνία. Στο συγκεκριμένο ρομποτικό βραχίονα χρησιμοποιήθηκαν τα SG90 και MG995 (Σχήμα 3.5).

Σχήμα 3.4: Είσοδοι των σερβομηχανισμών.

3.2.5 Arduino μότορας

Τα μοτέρ που χρησιμοποιήθηκαν είναι απλά μοτέρ συνεχούς ρεύματος μαζί με τη βάση στήριξης, έτοιμα για εργασίες ρομποτικής. Η τάση εισόδου κυμαίνεται από 1.5 – 6.5 V και με τον οδηγό μοτέρ που χρησιμοποιήθηκε τροφοδοτούνται με 5V (Σχήμα 3.6).

(α□) SG90.

(β□) MG995.

Σχήμα 3.5: Οι σερβομηχανισμοί του ρομποτικού βραχίονα.

Σχήμα 3.6: Μοτέρ συνεχούς ρεύματος μαζί με τη βάση του.

3.2.6 Σύνδεση Μηχανικών Εξαρτημάτων

Το ξύλο διαστάσεων 20x15 αποτελεί τη πλατφόρμα όπου στηρίζεται ο βραχίονας και τα ηλεκτρονικά εξαρτήματα. Η πλατφόρμα έχει τη δυνατότητα κίνησης χάρη σε τέσσερις ρόδες συνδεμένα με 4 μοτέρ. Η βάση του βραχίονα, αποτελείται από δύο σερβομηχανισμούς MG995 για την καλύτερη στήριξη και ενώνεται με τη πλατφόρμα με τέσσερις μεταλλικές γωνίες. Ο βραχίονας είναι κατασκευασμένος από πλεξιγκλάς πάχους 3mm και ο λόγος που επιλέχθηκε το συγκεκριμένο υλικό είναι διότι αν και πολύ δυνατό σε αντοχή υλικό, επεξεργάζεται εύκολα επιτρέποντας να τρυπηθεί και να κοπεί σε συγκεκριμένα σημεία. Επίσης σε περίπτωση αστοχίας του υλικού είναι εύκολο να αντικατασταθεί λόγω χαμηλού κόστους. Τα πλεξιγκλάς ενώνονται μεταξύ τους με βίδες και παξιμάδια και τροποποιήθηκαν στις άκρες επιτυγχάνοντας πιο ελαφριά κατασκευή. Στον αγκώνα τοποθετήθηκε ένας σερβομηχανισμός MG995, επειδή εκεί η επιβάρυνση είναι σχετικά μικρότερη από τη βάση. Η δαγκάνα αποτελούμενη και αυτή από πλεξιγκλάς, αγοράσθηκε από το εμπόριο έτοιμη προς χρήση και ελέγχεται από ένα σερβομηχανισμό SG90. Η USB κάμερα βρίσκεται στη πλατφόρμα, ακριβώς

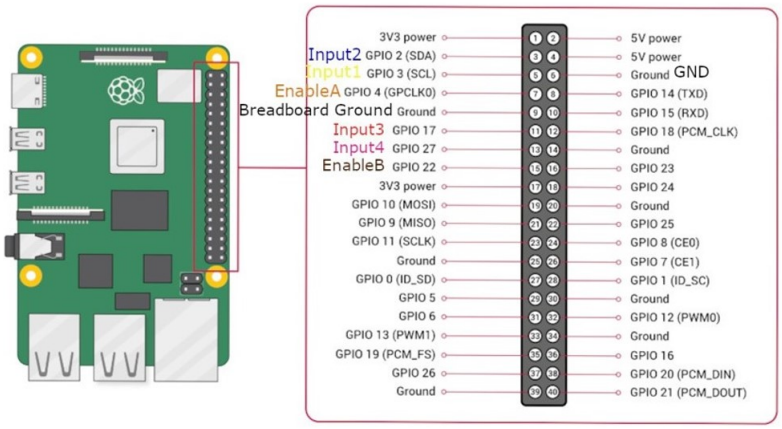
κάτω από τη δαγκάνα, επιτυγχάνοντας έτσι καλύτερη οπτική γωνία. Το Raspberry Pi τοποθετήθηκε πάνω στο ξύλο με μεταλλικούς αποστάτες για τη καλύτερη στήριξη και ασφαλή λειτουργία. Η θερμόκολλα χρησιμοποιήθηκε για να στηριχθούν καλύτερα τα μοτέρ, να μην μετακινούνται τα καλώδια στη πειραματική διάταξη, και να στερεωθεί ο οδηγός μοτέρ. Σημαντική είναι και η διαχείριση των καλωδίων, όσο είναι δυνατόν, ώστε με τη κίνηση της πλατφόρμας και του βραχίονα να μην εμποδίζουν και να μην προκληθεί ζημιά.

3.2.7 Σύνδεση Ηλεκτρονικών Εξαρτημάτων

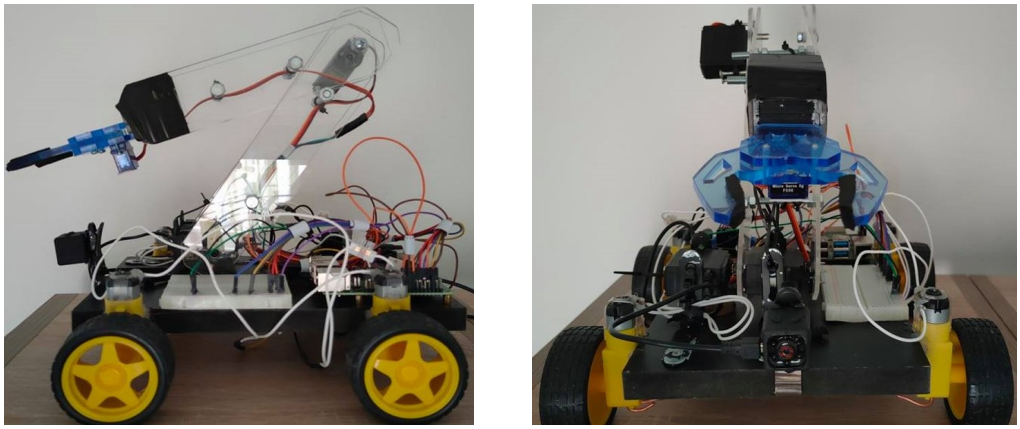
Τα μοτέρ στα οποία στηρίζονται οι ρόδες, συνδέονται στη μονάδα οδηγού μοτέρ στα σημεία Motor A και Motor B. Η μονάδα οδηγού μοτέρ, τροφοδοτείται με τροφοδοτικό 12 Volt - 1.5 Ampere οπότε αφαιρέθηκε και το 12V Jumper. Οι ακροδέκτες του οδηγού μοτέρ συνδέονται με τους ακροδέκτες του Raspberry Pi όπως παρακάτω.

- Enable A – GPIO 4 (GPCLK0)
- Input 1 – GPIO 3 (SCL)
- Input 2 – GPIO 2(SDA)
- Input 3 – GPIO 17
- Input 4 – GPIO 27
- Enable B – GPIO 22
- GND – Ground (Pin 6)

Το Raspberry Pi τροφοδοτείται με το γνήσιο τροφοδοτικό 3.1 Volt - 5 Ampere. Οι σερβομηχανισμοί της βάσης συνδέονται στο GPIO 23 και 24 αντίστοιχα, ο σερβομηχανισμός του αγκώνα στο GPIO 25 και ο σερβομηχανισμός της δαγκάνας στο GPIO 26. Τα παραπάνω τροφοδοτούνται από ξεχωριστή πηγή, με τροφοδοτικό 5 Volt – 2 Ampere συνδεδεμένα στη πειραματική διάταξη, ώστε να μην επιβαρύνεται το Raspberry Pi με παραπάνω φορτίο. Όλα τα εξαρτήματα πρέπει να έχουν κοινή γείωση, δηλαδή, οι ακροδέκτες 5 και 9 του Raspberry Pi που είναι οι γειώσεις (Ground) συνδέονται με τη γείωση του Οδηγού L298N και τη γείωση της πειραματικής διάταξης αντίστοιχα.



Σχήμα 3.7: Συνδεσμολογία του Raspberry Pi και του οδηγού L298N.



Σχήμα 3.8: Ο ρομποτικός βραχίονας.

Κεφάλαιο 4

Θεωρητικό Υπόβαθρο

4.1 Εισαγωγή

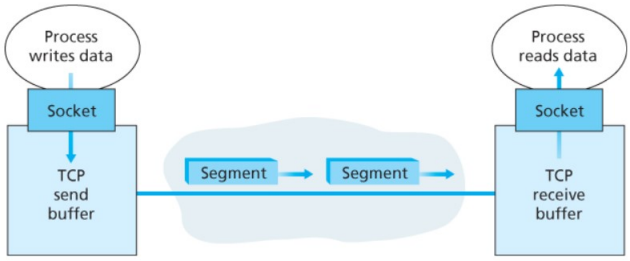
Ο σκοπός του κεφαλαίου είναι να δοθεί στον αναγνώστη, μία γενική ιδέα του λογισμικού που χρησιμοποιήθηκε για την λειτουργία του ρομποτικού βραχίονα. Για το προγραμματισμό και την υλοποίηση του ρομποτικού βραχίονα χρησιμοποιήθηκε η γλώσσα Python. Οι εντολές από τον υπολογιστή στάλθηκαν μέσω TCP/IP σύνδεσης και η βιβλιοθήκη Numpy για πράξεις μεγάλων δεδομένων. Τα «μάτια» του ρομποτικού βραχίονα αποτελεί μία κάμερα, όπου με τη χρήση της βιβλιοθήκης OpenCV θα λαμβάνει πληροφορίες για το περιβάλλον μέσω εικόνας και η ανίχνευση αντικειμένων επιτυγχάνεται μέσω της βιβλιοθήκης tensorflow και ενός ειδικού μοντέλου όπου εκπαιδεύτηκε με το SSD (Single Shot Multibox Detector). Τέλος οι κινήσεις της πλατφόρμας και του βραχίονα ελέγχονται μέσω PWM (Pulse Width Modulation) σημάτων από το Rasberry Pi.

4.2 Python

Η Python είναι μία αντικειμενοστραφής, υψηλού επιπέδου γλώσσα προγραμματισμού. Αποτελείται από δομές δεδομένων και σε συνδυασμό με τη δυναμική πληκτρολόγηση και τη δυναμική δέσμευση χρησιμοποιείται για την ανάπτυξη εφαρμογών, καθώς επίσης και για τη σύνδεση διαφορετικών εξαρτημάτων με σενάρια (scripts) ή συνδυασμό διαφορετικών τύπων κώδικα. Ένα από τα πολλά πλεονεκτήματα της είναι η ευκολία στην αναγνωσιμότητα, στη σύνταξη, και με τη χρήση δομοστοιχείων (modules) και πακέτων (packages), επιτυγχάνεται η αρθρωτή μορφή και η επαναχρησιμοποίηση του προγράμματος [40].

4.2.1 TCP/IP σύνδεση

Σε ένα δίκτυο, οι εφαρμογές ανταλλάσσουν μηνύματα μεταξύ τους. Όταν είναι να σταλθεί ένα μεγάλο μήνυμα, το τερματικό προέλευσης το διαιρεί σε μικρότερα κομμάτια δεδομένων. Τα κομμάτια αυτά ονομάζονται πακέτα (segments). Τα sockets αποτελούν «πόρτες» μέσω των οποίων περνούν τα δεδομένα από την διεργασία στο δίκτυο και αντίστοιχα από το δίκτυο στη διεργασία. Το IP (Internet Protocol), είναι το πρωτόκολλο Διαδικτύου και είναι υπεύθυνο για τη μορφή των πακέτων και τη δρομολόγηση τους. Υπάρχουν δύο εκδόσεις, η IPv4 η οποία είναι και η πιο διαδεδομένη και η IPv6. Το TCP (Transmission Control Protocol) είναι ένα αξιόπιστο και συνδεσμικό πρωτόκολλο μεταφοράς δεδομένων μεταξύ δύο διεργασιών. Συνδεσμικό, διότι πριν αρχίσει η ανταλλαγή δεδομένων μεταξύ των εφαρμογών, πρέπει πρώτα να σταλθούν κάποια τμήματα που καθορίζουν τις παραμέτρους της μεταφοράς, που αποκαλείται ως «χειραψία» και αξιόπιστο, διότι εξασφαλίζει το ρεύμα δεδομένων που λαμβάνει ο αποδέκτης να είναι αναλλοίωτο, συνεχές και με τη σωστή σειρά. Επειδή το πρωτόκολλο IP δεν εξασφαλίζει την εγγυημένη παράδοση και σωστή σειρά του δεδομενογράμματος, χρησιμοποιείται ο συνδυασμός TCP/IP (Σχήμα 4.1) [41].



Σχήμα 4.1: Σύνδεση TCP/IP [41].

4.2.2 Numpy

Η Numpy (Numerical Python), είναι μία βιβλιοθήκη της Python που χρησιμοποιείται για επιστημονικούς υπολογισμούς. Αυτό που παρέχει η συγκεκριμένη βιβλιοθήκη, είναι μία πολυδιάστατη δομή δεδομένων (Numpy Array) που αποτελείται από δεδομένα συγκεκριμένου τύπου, σε αντίθεση με τις λίστες της Python, που μπορεί να αποτελούνται από διαφορετικού τύπου δεδομένα. Με αυτό το τρόπο, η Numpy εκτελεί πράξεις σε βελτιστοποιημένο κώδικα χαμηλού επιπέδου, μία διαδικασία που ονομάζεται διανυσματοποίηση (vectorization), επιτυγχάνοντας έτσι καλύτερη απόδοση σε χρόνο εκτέλεσης των πράξεων [42].

4.3 Όραση υπολογιστών

Η όραση των υπολογιστών, είναι ένας κλάδος της επιστήμης των υπολογιστών, όπου παρέχει τη δυνατότητα στους υπολογιστές μέσω μίας κάμερας να αναγνωρίζει αντικείμενα όπως ακριβώς ο άνθρωπος. Πιο γενικά, ο σκοπός της όρασης των υπολογιστών, είναι να εξάγει πληροφορίες από μία εικόνα και να συμπεράνει κάτι για το κόσμο στον οποίο βρίσκεται.

4.3.1 Μοντέλο RGB

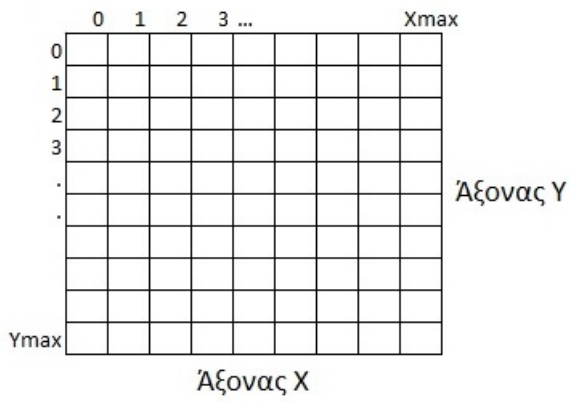
Ο άνθρωπος αντιλαμβάνεται το ορατό φως, ένα τμήμα από το ηλεκτρομαγνητικό φάσμα, με μήκος κύματος από 380nm (μωβ) έως 700nm (κόκκινο) [43]. Τα μάτια είναι πιο ευαίσθητα, στο κόκκινο (περίπου 630 nm), στο πράσινο (περίπου 530nm) και στο μπλε (περίπου 450nm). Αυτό αποτελεί και τη βάση της προβολής της έγχρωμης εξόδου στην οθόνη ενός υπολογιστή, με το συνδυασμό των παραπάνω χρωμάτων και ονομάζεται χρωματικό μοντέλο RGB (Red-Green-Blue). Αυτό το μοντέλο μπορεί να αναπαρασταθεί και με ένα μοναδιαίο κύβο πάνω σε ένα τρισδιάστατο σύστημα αξόνων, όπου οι άξονες αποτελούν τα βασικά χρώματα, η αρχή το μαύρο και το άθροισμα των διανυσμάτων του κάθε χρώματος αντιστοιχεί σε ένα διαφορετικό χρώμα του ορατού φάσματος [44]. Μερικά χαρακτηριστικά παραδείγματα αποτελούν:

- Πράσινο + Κόκκινο = Κίτρινο
- Μπλε + Κόκκινο = Πορφυρό (μωβ απόχρωση)
- Πράσινο + Μπλε = Κυανό
- Πράσινο + Κόκκινο + Μπλε = Λευκό

4.3.2 Απεικόνιση Οθόνης

Η απεικόνιση στην οθόνη του υπολογιστή, αποτελείται από εικονοστοιχεία (pixels), που περιγράφονται σε ακέραιες συντεταγμένες ενός καρτεσιανού επιπέδου. Οι θέσεις των εικονοστοιχείων, διευθυνσιοδοτούνται από τη επάνω αριστερά γωνία της οθόνης (Σχήμα 4.2) και το εικονοστοιχείο αποτελείται από ένα αριθμό, ο οποίος ορίζει το χρώμα με βάση το μοντέλο RGB. Η ανάλυση μίας εικόνας ορίζεται από τα πόσα εικονοστοιχεία αποτελείται στον

άξονα x και στον άξονα y. Για παράδειγμα μία εικόνα ανάλυσης 1080x720 έχει 1080 εικονοστοιχεία στον άξονα X και 720 εικονοστοιχεία στον άξονα Y. Καρέ (frame), ορίζεται ως μία απλή, ακίνητη εικόνα από ένα σύνολο συνεχόμενων εικόνων (βίντεο), που μπορεί να κάνει λήψη μία κάμερα. Ανάλογα με τα πόσα καρέ λαμβάνει μία κάμερα ή πόσες εικόνες δείχνει μία οθόνη σε ένα δευτερόλεπτο, ορίζεται και ο ρυθμός ανανέωσης της κάμερας ή της οθόνης αντίστοιχα (fps-frames per second). Όσο μεγαλύτερος είναι ο ρυθμός ανανέωσης, τόσο πιο ομαλό φαίνεται το αποτέλεσμα στην οθόνη.



Σχήμα 4.2: Οι θέσεις των εικονοστοιχείων σε σχέση με την πάνω αριστερή γωνία της οθόνης.

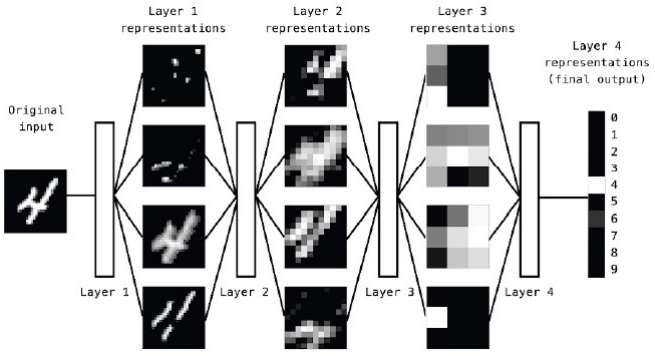
4.3.3 Open CV

Η OpenCv (Open Source Computer Vision Library), είναι μία βιβλιοθήκη ανοιχτού κώδικα (κώδικας που είναι προσβάσιμος σε όλους) και χρησιμοποιείται σε εφαρμογές όρασης υπολογιστών και μηχανικής μάθησης. Παρέχει αρκετούς αλγορίθμους, όπως αλγορίθμους εύρεσης και παρακολούθησης αντικειμένων, επεξεργασίας εικόνων και κυρίως κλίνει σε εφαρμογές όρασης πραγματικού χρόνου [45].

4.4 Μηχανική Μάθηση

Η Μηχανική Μάθηση αποτελεί ένα τομέα της Τεχνητής Νοημοσύνης, όπου ασχολείται με αλγόριθμους μάθησης. Η διαφορά της Μηχανικής Μάθησης είναι ότι ένα σύστημα εκπαιδεύεται παρά προγραμματίζεται. Δηλαδή αντί να δοθούν σε ένα σύστημα τα δεδομένα και οι κανόνες ώστε να βγει ένα αποτέλεσμα, στη Μηχανική Μάθηση δίνονται τα δεδομένα με τα

αποτελέσματα και ορίζονται οι κανόνες του συστήματος. Τα κύρια σημεία της Μηχανικής Μάθησης είναι τα δεδομένα εισόδου, η επιθυμητή έξοδος και ένας τρόπος ελέγχου εάν το σύστημα εκπαιδεύεται σωστά. Σε ένα τέτοιο μοντέλο, οι είσοδοι μετατρέπονται σε έξοδο με τη χρήση κατάλληλων αναπαραστάσεων (Σχήμα 4.3), δηλαδή μετατροπή των δεδομένων σε άλλη μορφή, χρησιμοποιώντας νευρωνικά δίκτυα και μεθόδους βαθιάς μάθησης [46].



Σχήμα 4.3: Αναπαραστάσεις που δημιουργούνται από το νευρωνικό δίκτυο ενός μοντέλου αναγνώρισης αριθμών [46].

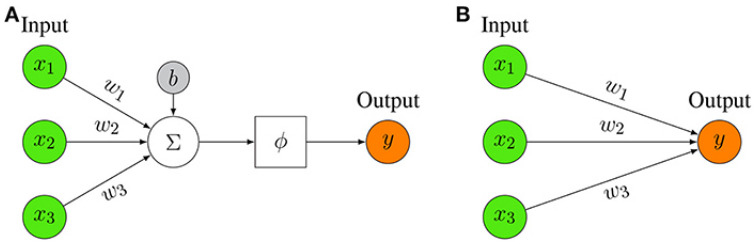
4.4.1 Νευρωνικό Δίκτυο

Τα νευρωνικά δίκτυα (Neural Networks), είναι γνωστά από τη δεκαετία του 1940, όπου η πρόκληση ήταν να προσομοιωθεί ο ανθρώπινος εγκέφαλος για τη λύση προβλημάτων εκμάθησης, αλλά λόγω της χαμηλής επεξεργαστικής ισχύς και των περιορισμένων δεδομένων, μειώθηκε το ενδιαφέρον στις αρχές του 20ου αιώνα, αλλά επανήλθε το 2006, λόγω μίας σημαντικής ανακάλυψης στην αναγνώριση ομιλίας [47].

Ένα νευρωνικό δίκτυο αποτελείται από κόμβους συνδεδεμένους μεταξύ τους, όπου ονομάζονται νευρώνες και οι ακμές που τους ενώνουν αποτελούν τα βάρη (Σχήμα 4.4Β). Οι νευρώνες χωρίζονται σε στρώματα όπου έχουν εισόδους - εξόδους και αποτελούν το αρχικό στρώμα, το ενδιάμεσο στρώμα (κρυμμένο) και το τελικό στρώμα. Το αρχικό στρώμα δέχεται τιμές από το περιβάλλον (ακατέργαστα δεδομένα) και το τελικό στρώμα δίνει το αποτέλεσμα [48].

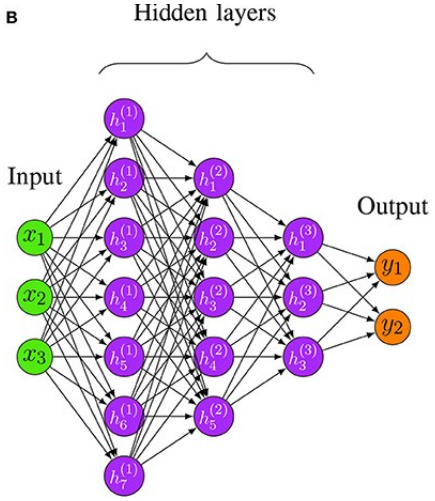
4.4.2 Βαθιά Μάθηση

Βαθιά μάθηση (Deep Learning), είναι μία μέθοδος Μηχανικής Μάθησης, αποτελούμενη από πολλαπλές διαδοχικές αναπαραστάσεις. Ένα δίκτυο βαθιάς μάθησης, είναι ένα τεχνητό



Σχήμα 4.4: (Α) Αναπαράσταση ενός νευρωνικού δικτύου όπου αθροίζονται τα γινόμενα των βαρών (w), με τις τιμές εισόδου (χ) συν τη πόλωση b και ελέγχεται από μία συνάρτηση ενεργοποίησης (φ). (Β) Απλή αναπαράσταση εισόδου-εξόδου ενός νευρωνικού δικτύου [49].

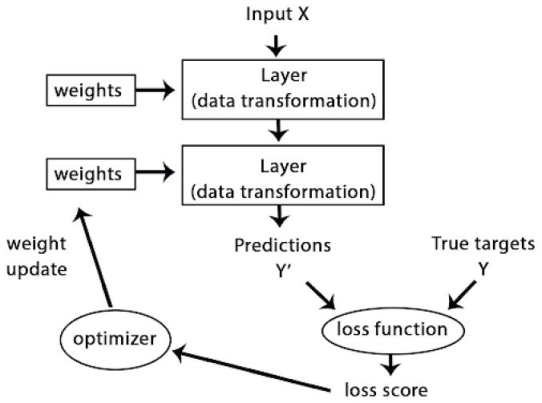
νευρωνικό δίκτυο με παραπάνω από ένα ενδιάμεσο στρώμα (Σχήμα 4.5), όπου τα χαρακτηριστικά των στρωμάτων δεν ορίζονται από τον άνθρωπο, αλλά δημιουργούνται μέσω της μάθησης χρησιμοποιώντας δεδομένα [48]. Η αρχιτεκτονική που χρησιμοποιείται ευρέως για τη σύνδεση ενός νευρωνικού δικτύου βαθιάς μάθησης, είναι η τροφοδότηση προς τα εμπρός (feedforward architecture), όπου τα δεδομένα τροφοδοτούνται από το ένα στρώμα στο επόμενο. Η έξοδος ενός νευρώνα-κόμβου, υπολογίζεται από το άθροισμα των γινομένων των τιμών του προηγούμενου στρώματος πολλαπλασιασμένο με το βάρος, συν τη πόλωση (ένας αριθμός που αθροίζεται σε κάθε νευρώνα). Έπειτα μέσω μίας συνάρτησης ενεργοποίησης καθορίζεται η τελική τιμή του νευρώνα (Σχήμα 4.4Α).



Σχήμα 4.5: Ένα βαθύ νευρωνικό δίκτυο με τρεις εισόδους, τρία κρυμμένα στρώματα και δύο εξόδους [49].

Η μάθηση με επίβλεψη (supervised learning), είναι η πιο γνωστή μορφή Μηχανικής Μάθησης στην οποία έχουν δοθεί η είσοδος και η επιθυμητή έξοδος, και μέσω της εκπαίδευσης τροποποιούνται οι παράμετροι (βάρη και πολώσεις), ώστε η διαφορά της επιθυμητής εξόδου

από την έξοδο του μοντέλου να μειωθεί όσο το δυνατόν περισσότερο. Αυτό ελέγχεται μέσω μια συνάρτησης απώλειας (loss function), όπου μετράει τη διαφορά της πρόβλεψης του δικτύου y_i από την αληθή έξοδο t_i και μετά αναλαμβάνει ένας βελτιστοποιητής όπου εκτελεί τον αλγόριθμο «προς τα πίσω διάδοση» (Backpropagation) (Σχήμα 4.6). Η παραπάνω διαδικασία αποτελεί μία εποχή (epoch) και η εκπαίδευση αποτελείται από συγκεκριμένο αριθμό εποχών που ορίζει ο προγραμματιστής.



Σχήμα 4.6: Η τιμή του σφάλματος χρησιμοποιείται σαν σήμα ανατροφοδότησης για να τροποποιηθούν τα βάρη από το βελτιστοποιητή [49].

Όταν ένα δίκτυο έχει παραπάνω από μία εξόδους, τότε η κάθε έξοδος έχει τη δικιά της συνάρτηση απώλειας, και χρησιμοποιείται το μέσο τετραγωνικό σφάλμα (Mean Square Error) (Εξίσωση 4.1) για τον υπολογισμό μίας τιμής που θα χρησιμοποιήσει ο βελτιστοποιητής.

$$J = \frac{1}{P}\sum_{i\in P}||t_i - y_i||^2 = \frac{1}{2P}\sum_{i=1}^{P}(t_i - y_i)^2 \tag{4.1}$$

Για να ρυθμιστούν οι παράμετροι κατά τη διάρκεια της μάθησης, υπολογίζεται ένα διάνυσμα κλίσης για το κάθε βάρος, όπου δείχνει πως μεταβάλλεται το σφάλμα, αν μεταβληθεί το βάρος κατά ένα μικρό ποσοστό. Τότε το βάρος διορθώνεται στην αντίθετη φορά από ότι του διανύσματος (Εξίσωση 4.2). Η μέθοδος αυτή ονομάζεται βαθμωτή κατάβαση (Gradient Descent) και είναι μία μέθοδος εύρεσης ελαχίστου μίας συνάρτησης. Στον Backpropagation αλγόριθμο χρησιμοποιείται η στοχαστική βαθμωτή κατάβαση [50], στην οποία τα δεδομένα χωρίζονται σε μικρές ομάδες (batches) και σε κάθε εποχή (epoch) χρησιμοποιούνται όλες οι ομάδες. Η ενημέρωση των παραμέτρων (βαρών και πολώσεων) γίνεται μετά από κάθε

ομάδα, επιτυγχάνοντας ταχύτερη σύγκλιση (μείωση) του κόστους σφάλματο . [51].

$$W_{new} - W_{old} = -\beta \frac{\vartheta J}{\vartheta W_{old}} \tag{4.2}$$

Το πως επηρεάζεται το σφάλμα (J) της εξόδου του στρώματος L σε σχέση με τους νευρώνες του προηγούμενου στρώματος L-1, μπορεί να φανεί από το κανόνα αλυσίδας παραγώγισης,

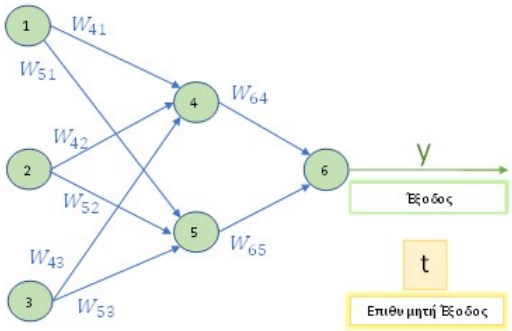
$$\frac{\partial J_0}{\partial a_k^{L-1}} = \sum_{j=0}^{n^L-1} \frac{\partial u_j^L}{\partial a_k^{L-1}} \frac{\partial a_j^L}{\partial u_j^L} \frac{\partial J_0}{\partial a_j^L}$$

όπου

$$\frac{\partial u_j^L}{\partial a_k^{L-1}} = W_{jk}^L$$

$$\frac{\partial a_j^L}{\partial u_j^L} = 1 \qquad (ReLU)$$

$$\frac{\partial J_0}{\partial a_j^L} = 2(a_j^L - y_j)$$



Σχήμα 4.7: Αναπαράσταση μίας εξόδου ενός νευρωνικού δικτύου και η επιθυμητή έξοδος. Η πόλωση του κάθε νευρώνα έχει αγνοηθεί.

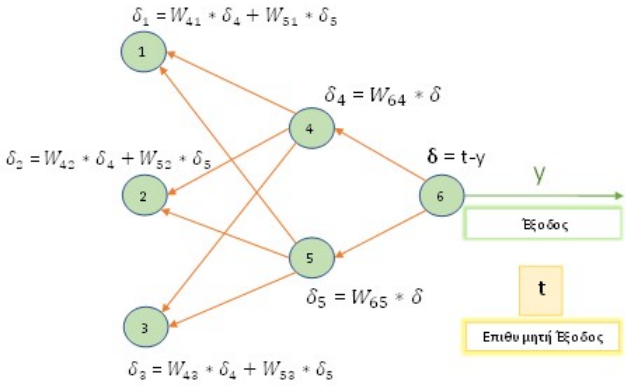
Στο Σχήμα 4.7 φαίνεται μία έξοδος ενός νευρωνικού δικτύου και δύο προηγούμενα κρυμμένα στρώματα, όπου οι τιμές του κάθε νευρώνα πριν τη συνάρτηση ενεργοποίησης f δίνεται από το τύπο

$$u_j^L = \sum_{k=0}^{n^{(L-1)}-1} (W_{jk}^L * a_k^{L-1})$$

και η έξοδος μετά τη συνάρτηση ενεργοποίησης

$$a_k^L = f\left(u_k^L\right)$$

Ο υπολογισμός του σφάλματος δ της κάθε εξόδου του τελευταίου στρώματος ενός νευρωνικού δικτύου, γίνεται από την Εξίσωση 4.3 όπου $f^{'}(u_i^k(L))$ η παράγωγος της συνάρτησης



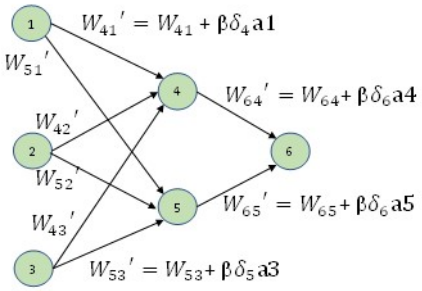
Σχήμα 4.8: Υπολογισμός των σφαλμάτων δ του κάθε νευρώνα στα στρώματα του νευρωνικού δικτύου. Έχει θεωρηθεί ως συνάρτηση ενεργοποίησης (f) η ReLU, όπου η παράγωγος της είναι 1 και δεν φαίνεται παραπάνω.

ενεργοποίησης. Στα κρυμμένα στρώματα, ο υπολογισμός του σφάλματος δ γίνεται από την Εξίσωση 4.4, όπου φαίνεται ότι εξαρτάται από τα σφάλματα του επόμενου στρώματος. Στο Σχήμα 4.8 φαίνεται η χρήση των παραπάνω εξισώσεων για τον υπολογισμό των σφαλμάτων.

$$\delta_i^{(k)}(L) = f'(u_i^k(L))(t_i^k - y_i^k) \tag{4.3}$$

$$\delta_i^{(k)}(l) = f'(u_i^k(l)) \sum_{m=1}^{N(l+1)} W_{mi}(l+1)\delta_m(l+1)) \qquad l = 1, .., L-1 \tag{4.4}$$

$$W_{ij}(l, k+1) = W_{ij}(l, k) + \beta\delta_i^{(k)}(l)a_j^k(l-1) \qquad j = 0, ..N(l), \qquad l = 1, .., L \tag{4.5}$$

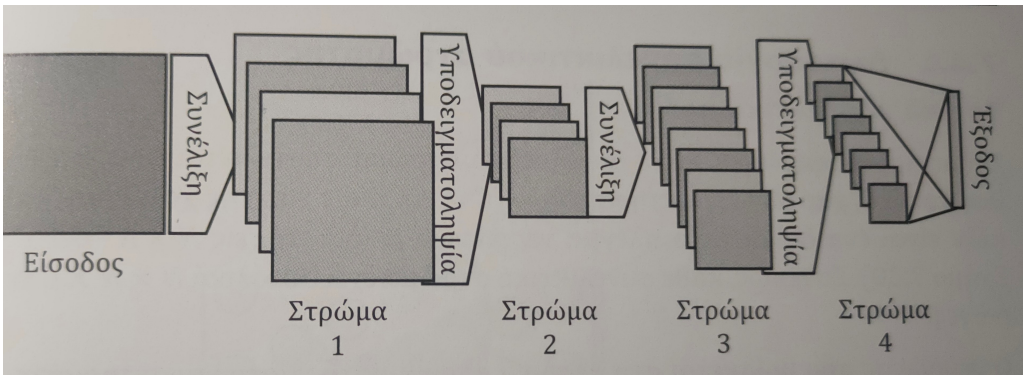


Σχήμα 4.9: Υπολογισμός των νέων βαρών με βάση το βήμα εκπαίδευσης β, τα σφάλματα δ και τις εξόδους a των νευρώνων.

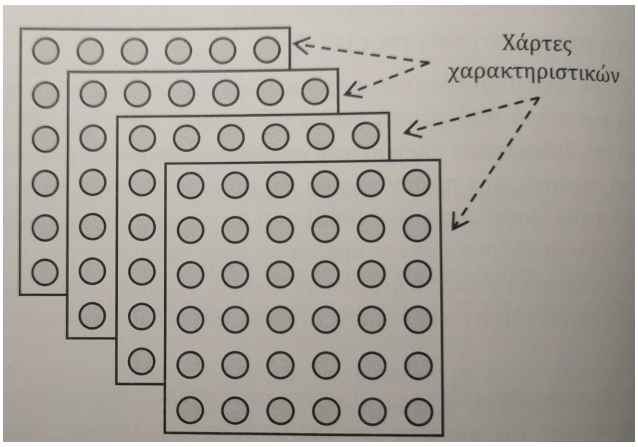
Όταν έχουν υπολογιστεί τα σφάλματα όλων των νευρώνων, υπολογίζονται τα νέα βάρη με την Εξίσωση 4.5 (Σχήμα 4.9). Η διαδικασία αυτή συνεχίζεται ως ότου το κόστος εκπαίδευσης είναι μικρότερο από κάποιο αριθμό ή το σφάλμα δεν μεταβάλλεται σημαντικά μετά από διαδοχικές εποχές [52][53].

4.4.3 Συνελικτικό νευρωνικό δίκτυο

Για την ανίχνευση αντικειμένων σε μία εικόνα, τα κατάλληλα δίκτυα βαθιάς μάθησης είναι τα συνελικτικά νευρωνικά δίκτυα. Αποτελούνται από στρώματα συνέλιξης (convolution) και υποδειγματοληψίας (pooling) που μετασχηματίζουν την εικόνα σε χάρτες χαρακτηριστικών (feature maps) και μέσω ενός κατηγοριοποιητή (classifier) διακρίνονται τα αντικείμενα σε κλάσεις (Σχήμα 4.10). Ένας χάρτης χαρακτηριστικών αποτελεί μία ομάδα από δισδιά-

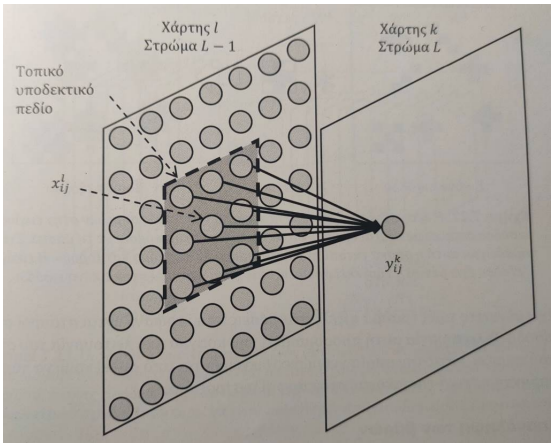


Σχήμα 4.10: Η αρχιτεκτονική ενός συνελικτικού νευρωνικού δικτύου [54].



Σχήμα 4.11: Ένα στρώμα ενός συνελικτικού νευρωνικού δικτύου όπου οι νευρώνες αναπαρίστανται ως κύκλοι [54].

στατα πλέγματα νευρώνων δημιουργώντας ένα στρώμα του νευρωνικού δικτύου (Σχήμα

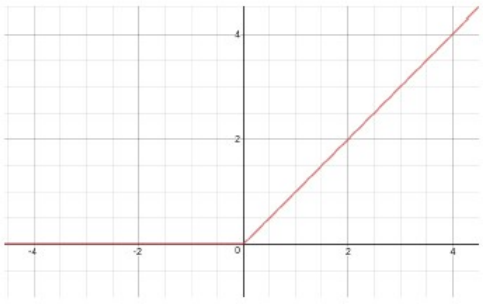


Σχήμα 4.12: Αναπαράσταση του τοπικού υποδεκτικού πεδίου και του νευρώνα του επόμενου στρώματος [54].

4.11). Ο νευρώνας ενός χάρτη δέχεται εισόδους από μία μικρή περιοχή του προηγούμενου στρώματος, το ονομαζόμενο τοπικό υποδεκτικό πεδίο (Σχήμα 4.12). Η τιμή του

$$y_{ij}^{k} = f\left(\sum_{n=1}^{C'}\sum_{a=1}^{m}\sum_{b=1}^{m} W_{a,b,n}^{k} * x_{i-a,j-b}^{n} + b^{k}\right)$$

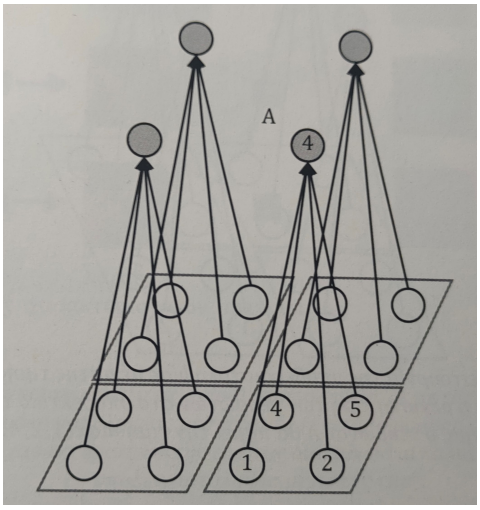
όπου W^k αποτελούν τα βάρη και b^k η πόλωση του κάθε νευρώνα. Η συνάρτηση ενεργοποίησης f, είναι μία μη γραμμική συνάρτηση η οποία είναι υπεύθυνη αν θα ενεργοποιηθεί ο νευρώνας ή όχι. Συνήθως για την εκπαίδευση ενός μοντέλου χρησιμοποιείται η ReLU(Rectified Linear Unit) (Σχήμα 4.13) η οποία επιστρέφει μηδέν για αρνητικές τιμές, ενώ σε θετικές επιστρέφει τη τιμή της εισόδου $(f(z) = max(0, z))$ [55]. Η συνέλιξη η οποία είναι



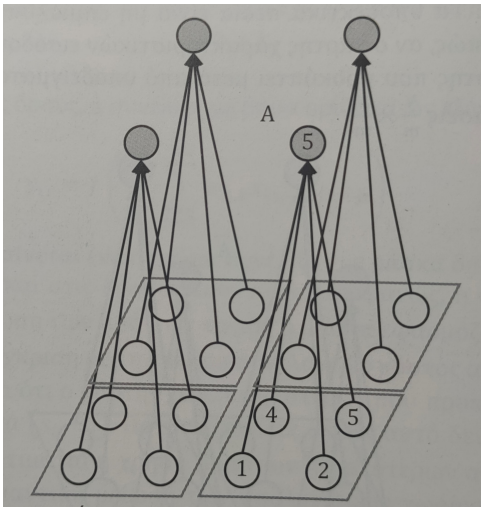
Σχήμα 4.13: Η συνάρτηση ενεργοποίησης ReLU (Rectified Linear Unit). Έχει ως έξοδο 0 για αρνητικά x, ενώ έξοδο τη τιμή x για θετικά x [55].

η μαθηματική πράξη μέσα στη παρένθεση, φιλτράρει την εικόνα δίνοντας τη μέγιστη απόκριση στα σημεία που ταιριάζουν με το φίλτρο (ο πίνακας που περιέχει όλα τα βάρη των

νευρώνων ενός συγκεκριμένου χάρτη χαρακτηριστικών). Μετά την συνέλιξη, γίνεται υποδειγματοληψία, όπου τα δεδομένα συμπιέζονται και επιτυγχάνεται καλύτερη αναπαράσταση των χαρακτηριστικών καθώς το σύστημα γίνεται λιγότερο ευαίσθητο σε παραμορφώσεις και μετατοπίσεις των αντικειμένων. Στο στρώμα αυτό, η τιμή του κάθε νευρώνα προκύπτει είτε με υποδειγματοληψία μέσης τιμής (average pooling), όπου υπολογίζεται η μέση τιμή των νευρώνων του τοπικού υποδεκτικού πεδίου του συνελεκτικού στρώματος(Σχήμα 4.14), είτε με υποδειγματοληψία μέγιστης τιμής (max pooling), όπου λαμβάνεται υπ'όψιν η μέγιστη τιμή των νευρώνων(Σχήμα 4.15). [54]



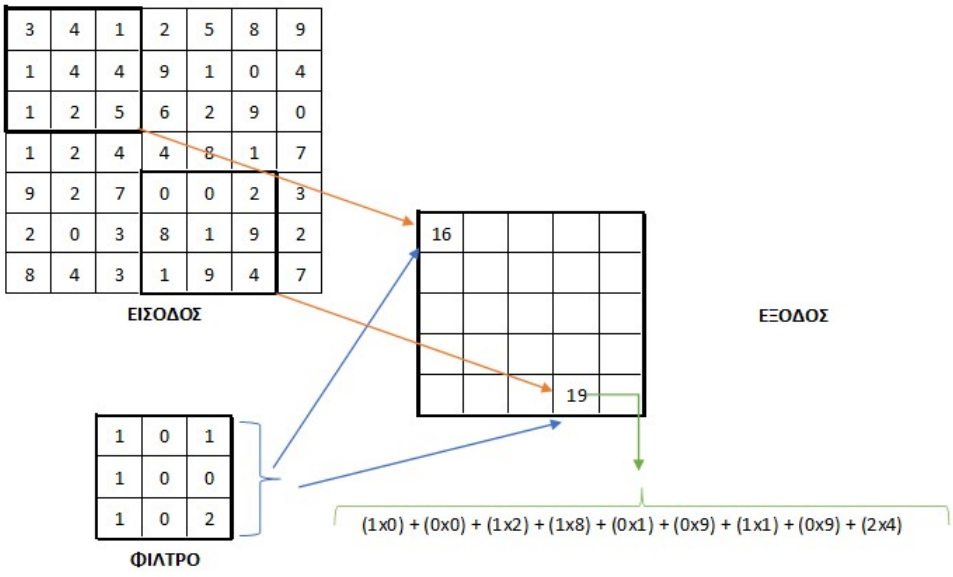
Σχήμα 4.14: Υποδειγματοληψία μέσης τιμής. Ο νευρώνας έχει τη τιμή $1/4 * (1 + 2 + 4 + 5)$ [54].



Σχήμα 4.15: Υποδειγματοληψία μέγιστης τιμής. Ο νευρώνας έχει τη μέγιστη τιμή από τις τιμές των νευρώνων του υποδεκτικού στρώματος [54].

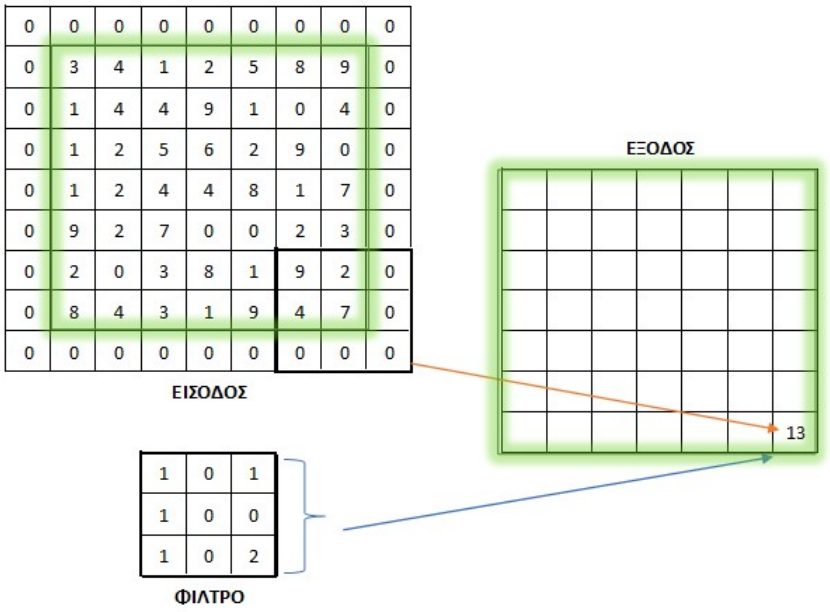
Μία εικόνα αποτελεί ένα δισδιάστατο πλέγμα από εικονοστοιχεία όπου η τιμή τους αποτελεί τη θέση στην εικόνα αυτή. Για να κωδικοποιηθεί το χρώμα από το οποίο αποτελείται το κάθε εικονοστοιχείο, χρησιμοποιείται ένα διάνυσμα που περιλαμβάνει τις τιμές RGB στο κάθε σημείο του πλέγματος.

Το κάθε στρώμα του συνελεκτικού δικτύου έχει διαστάσεις $L_q \times B_q \times d_q$ όπου L_q το ύψος, B_q το πλάτος και d_q το βάθος. Στο πρώτο στρώμα (στρώμα εισόδου) οι τιμές για τα L_q, B_q είναι το ύψος και το πλάτος της εικόνας αντίστοιχα και το βάθος (κανάλια) $d_q = 3$ (λόγω του ότι η εικόνα αποτελεί συνδυασμό των τριών χρωμάτων RGB και το κάθε κανάλι είναι οι τιμές του αντίστοιχου χρώματος). Οι παράμετροι (βάρη) του συνελεκτικού δικτύου ονομάζονται φίλτρα ή πυρήνες και είναι τρισδιάστατες δομές με διαστάσεις $F_q \times F_q \times d_q$, όπου συνήθως η τιμή F είναι μικρή και περιττή και το βάθος πάντα ίδιο με το βάθος του στρώματος στο οποίο εφαρμόζεται. Η συνέλιξη τοποθετεί το φίλτρο σε κάθε σημείο της εικόνας ή του χάρτη χαρακτηριστικών στα ενδιάμεσα στρώματα και εκτελεί πολλαπλασιασμό των παραμέτρων του φίλτρου με τις αντίστοιχες τιμές του χάρτη χαρακτηριστικών και τα προσθέτει (Σχήμα 4.16). Το επόμενο στρώμα θα έχει διαστάσεις $(L_q - F_q + 1) \times (B_q - F_q + 1) \times (d_q)$, όπως φαίνεται και στο Σχήμα 4.16, ότι με την εφαρμογή ενός φίλτρου 3x3 σε ένα πλέγμα 7x7 έχει ως έξοδο ένα πλέγμα 5x5 (7-3+1=5). Το βάθος του επόμενου στρώματος (αριθμός των χαρτών χαρακτηριστικών) είναι ίσο με τον αριθμό των διαφορετικών φίλτρων που εφαρμόζονται σε



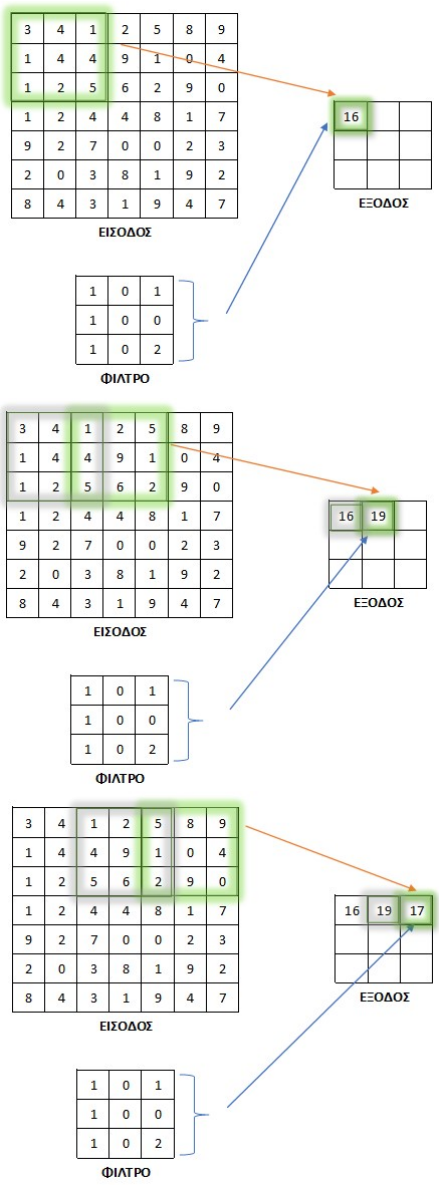
Σχήμα 4.16: Παράδειγμα χρήσης ενός φίλτρου 3x3x1 σε ένα δισδιάστατο πλέγμα μεγέθους 7x7x1.

κάθε στρώμα. Δηλαδή αν εφαρμοστούν τρία φίλτρα, τότε στο επόμενο στρώμα το βάθος θα είναι τρία. Στα κρυφά στρώματα, το βάθος μπορεί να ξεπερνάει και τον αριθμό των 500, διότι το κάθε φίλτρο προσπαθεί να βρει ένα μοτίβο σε κάθε περιοχή της εικόνας και χρειάζονται πολλά φίλτρα να συνδυάσουν πιθανές μορφές και να φτάσουν σε ένα τελικό αποτέλεσμα. Επειδή η χρήση των φίλτρων μειώνει τη διάσταση του επόμενου στρώματος, οδηγεί σε χάσιμο πληροφοριών περιμετρικά της εικόνας. Αυτό αποφεύγεται με τη προσθήκη $(F_q - 1)/2$ εικονοστοιχείων με μηδενική τιμή περιμετρικά της εικόνας (half-padding) (Σχήμα 4.17). Υπάρχει και η τεχνική full padding, όπου προσθέτει $F_q - 1$ εικονοστοιχεία, με αποτέλεσμα να αυξάνει τη διάσταση του επόμενου στρώματος κατά ένα.



Σχήμα 4.17: Παράδειγμα χρήσης ενός φίλτρου 3x3x1 σε ένα δισδιάστατο πλέγμα μεγέθους 7x7x1 με τη μέθοδο προσθήκης (3-1)/2 = 1 εικονοστοιχείων περιμετρικά. Η έξοδος συνεχίζει να έχει την ίδια διάσταση με της εισόδου.

Ο βαθμός της λεπτομέρειας μπορεί να μειωθεί και με τη χρήση βηματισμού (stride), δηλαδή το βαθμό μετακίνησης του φίλτρου πάνω στο πλέγμα. Η έξοδος θα έχει ύψος $(L_q - F_q)/S_q + 1$ και πλάτος $(B_q - F_q)/S_q + 1$ όπου S_q η τιμή του βηματισμού. Στις παραπάνω περιπτώσεις ο βηματισμός ήταν ένα. Στα Σχήματα 4.18 ο βηματισμός έχει οριστεί ως δύο και δείχνει πως το φίλτρο μετακινείται κατά 2 εικονοστοιχεία κάθε φορά. Ο βηματισμός αυξάνει ραγδαία το πεδίο δεκτικότητας (receptive field), τη περιοχή δηλαδή που καλύπτει το φίλτρο στην είσοδο, ενώ μειώνει τη διάσταση όλου του στρώματος (spatial footprint). Αυ-



Σχήμα 4.18: 3 διαδοχικές αναπαραστάσεις τοποθέτησης φίλτρου με βηματισμό 2.

ξάνοντας το πεδίο δεκτικότητας, επιτυγχάνεται η λήψη περίπλοκων χαρακτηριστικών σε ένα μεγαλύτερο μέρος της εικόνας και η συνήθης τεχνική είναι να λαμβάνει περίπλοκες μορφές στα τελευταία στρώματα [46].

Μετά τη συνέλιξη ακολουθεί η συνάρτηση ενεργοποίησης (συνήθως ReLU) και υποδειγματοληψία μέγιστης ή μέσης τιμής. Η υποδειγματοληψία γίνεται μετά τη συνάρτηση ενεργοποίησης και ο στόχος της είναι να μειώσει τις διαστάσεις ύψους και πλάτους αλλά όχι του βάθους, μειώνοντας τον αριθμό των παραμέτρων, καθώς και το υπολογιστικό κόστος. Έτσι εφόσον είναι γνωστό ένα συγκεκριμένο χαρακτηριστικό (μία ακμή ή γωνία), είναι σημαντικό η τοποθεσία του σε σχέση με άλλα χαρακτηριστικά και όχι η τοποθεσία του μέσα

στην εικόνα. Αυτό βασίζεται στο γεγονός ότι συνδυασμοί ακμών οδηγούν σε μοτίβα, τα μοτίβα σε κομμάτια και τα κομμάτια σε αντικείμενα δίνοντας τη δυνατότητα στο μοντέλο να αναγνωρίζει ίδια αντικείμενα σε διαφορετικές καταστάσεις και μορφές [51] [46].

Κατηγοριοποίηση αντικειμένων και λογιστική παλινδρόμηση

Κατηγοριοποίηση (ή ταξινόμηση) ονομάζεται η διαδικασία ανάθεσης ετικετών (labels) σε ένα σύνολο χαρακτηριστικών. Χρησιμοποιείται ένα σύνολο εκπαίδευσης, το οποίο περιέχει τις τιμές και τις ετικέτες των χαρακτηριστικών και με τη χρήση ενός αλγορίθμου μάθησης δημιουργείται το μοντέλο κατηγοριοποίησης. Η ακρίβεια ενός μοντέλου κατηγοριοποίησης υπολογίζεται ως εξής:

$$Accuracy = \frac{N_{correct}}{N_{total}}$$

όπου $N_{correct}$ το πλήθος των σωστών προβλέψεων και N_{total} το συνολικό πλήθος των προβλέψεων. Οι τύποι κατηγοριοποιητών με χαρακτηριστικά παραδείγματα είναι οι εξής:

- Δυαδικοί, όπου υπάρχουν δύο πιθανές ετικέτες, συνήθως θετική ή αρνητική απόφαση εάν ανήκει σε μία κατηγορία.
- Πολλών κατηγοριών, όπου υπάρχουν παραπάνω από δύο ετικέτες.
- Αιτιοκρατικοί, όπου παράγουν μία ετικέτα με μία συγκεκριμένη τιμή.
- Πιθανοφανών, όπου η τιμή της ετικέτας κυμαίνεται από 0 έως 1, δείχνοντας τη πιθανότητα να ανήκει σε μία κατηγορία (δίκτυα Bayes, Λογιστική παλινδρόμηση).
- Γραμμικοί ή μη, όπου χρησιμοποιείται ένα επίπεδο γραμμικό (Perceptron) ή μη γραμμικό (νευρωνικό δίκτυο πολλών στρωμάτων) για το διαχωρισμό των κατηγοριών.
- Ολικοί, όπου το μοντέλο προσαρμόζεται σε ολόκληρο το σύνολο δεδομένων.
- Τοπικοί, όπου το σύνολο δεδομένων χωρίζεται σε μικρότερες περιοχές και προσαρμόζεται ένα μοντέλο σε κάθε περιοχή (k- πλησιέστερου γείτονα).
- Παραγωγικοί, όπου εκτός από τη πρόβλεψη της ετικέτας, περιγράφεται και πως παράχθηκε (απλοϊκός Bayes, δίκτυα Bayes).

- Διαχωριστικοί, όπου γίνεται απευθείας πρόβλεψη χωρίς τις πληροφορίες εξαγωγής της κάθε ετικέτας (δένδρα απόφασης, κατηγοριοποιητές κανόνων, τεχνητά νευρωνικά δίκτυα και άλλα).

Το πόσο πιθανό είναι να συμβεί ένα γεγονός ε, ονομάζεται πιθανότητα P(x) και ισχύει $0 < P(x) < 1$. Η πιθανότητα να συμβεί ένα γεγονός Β όταν έχει συμβεί ή θα συμβεί ένα γεγονός Α, ορίζεται ως υπό συνθήκη πιθανότητα και συμβολίζεται με $P(A|B)$ [56].
Στην απλή παλινδρόμηση (Linear Regression) η εκτίμηση του μοντέλου είναι

$$t = w_1x_1 + w_2x_2 + \cdots + w_nx_n + b + \epsilon \tag{4.6}$$

όπου w και b οι παράμετροι του μοντέλου και ϵ το σφάλμα προσέγγισης $t - y$. Επειδή όμως οι εκτιμήσεις y λαμβάνουν τιμές από $-\infty$ ως $+\infty$ χρειάζεται ένα άλλο μοντέλο για διακριτές ετικέτες [54].
Η λογιστική παλινδρόμηση (Logistic Regression), αποτελεί ένα πιθανοφανή, διαχωριστικό μοντέλο κατηγοριοποίησης, όπου εξάγει τη πρόγνωση της πρότυπης εισόδου x ως:

$$\frac{P(y=1|x)}{P(y=0|x)} = e^z = e^{w^Tx+b} \tag{4.7}$$

όπου $w^Tx = w_1x_1 + w_2x_2 + \cdots + w_nx_n$ και εάν $e^z > 1$ τότε το x ανήκει στη κατηγορία 1 αλλιώς ανήκει στη κατηγορία 0 (δυαδική κατηγοριοποίηση). Έχει ονομαστεί έτσι από τη σιγμοειδής (λογιστική) συνάρτηση (Σχήμα 4.19)

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

Αυτό επειδή ισχύει $P(y=0|x) + P(y=1|x) = 1$ η Εξίσωση 4.7 μπορεί να γραφτεί ως

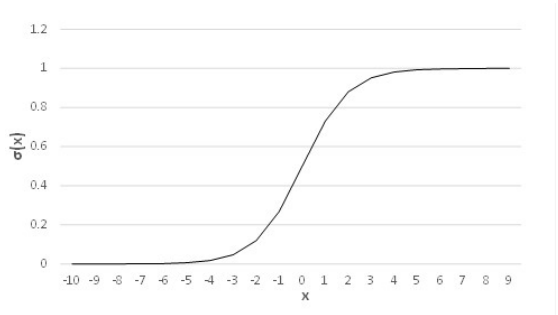
$$\frac{P(y=1|x)}{1-P(y=1|x)} = e^z \Rightarrow P(y=1|x) = \frac{1}{1+e^{-z}} = \sigma(z)$$

και

$$P(y=0|x) = 1 - \sigma(z) = 1 - \frac{1}{1+e^z} = \frac{1}{1+e^z} = \sigma(-z)$$

Κατά τη διάρκεια της εκπαίδευσης, οι παράμετροι (w,b) αλλάζουν με στόχο τη μεγιστοποίηση της κοινής πιθανότητας των παρατηρήσεων y:

$$L(w,b) = \prod_{i=1}^{n} P(y_i|x_i,w,b) = \prod_{i=1}^{n} P(y=1|x_i)^{y_i} \times P(y=0|x_i)^{1-y_i} \tag{4.8}$$



Σχήμα 4.19: Η σιγμοειδής (λογιστική) συνάρτηση. Εάν το z τείνει στο $+\infty$ η τιμή της πρόβλεψης θα είναι 1, ενώ όσο τείνει στο $-\infty$ η πρόβλεψη θα είναι 0.

Η συνάρτηση κόστους θεωρείται ο αρνητικός λογάριθμος της Εξίσωσης 4.8

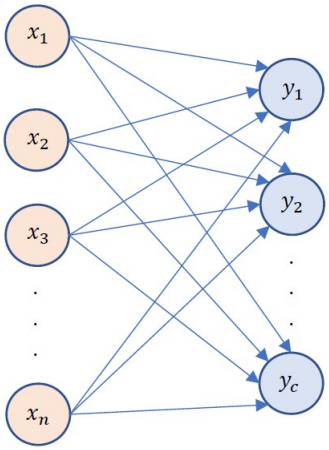
$$\begin{aligned} E(w,b) &= -\ln\left(L(w,b)\right) = -\sum_{i=1}^{n}(y_i \ln\left(P(y=1|x_i)\right) + (1-y_i)\ln P(y=0|x_i)) \\ &= -\sum_{i=1}^{n}(y_i \ln\left(\sigma(w^T x_i + b)\right) + (1-y_i)\ln\left(1-\sigma(w^T x_i + b)\right)) \end{aligned} \tag{4.9}$$

Η Εξίσωση 4.9 μπορεί να ελαχιστοποιηθεί με διάφορες μεθόδους όπως εκείνης της βαθμωτής κατάβασης ή Newton-Raphson, για την ενημέρωση των παραμέτρων του μοντέλου λογιστικής παλινδρόμησης μετά από κάθε επανάληψη [56].

Softmax

Ένα δίκτυο Softmax αποτελείται από ένα στρώμα νευρώνων, όπου εφαρμόζεται η λογιστική παλινδρόμηση για τη κατηγοριοποίηση C κλάσεων (Σχήμα 4.20). Η έξοδος του m-οστού νευρώνα δίνει τη πιθανότητα η είσοδος x να ανήκει στη κλάση m ($y_m = P(t = m|x)$) και η τιμή του υπολογίζεται από τη παρακάτω σχέση [54].

$$y_m = \frac{\exp\left(W_m^T x + W_{m,0}\right)}{\sum_{j=1}^{C}\exp\left(W_j^T x + W_{j,0}\right)}$$



Σχήμα 4.20: Ένα νευρωνικό δίκτυο Softmax για τη κατηγοριοποίηση C κλάσεων

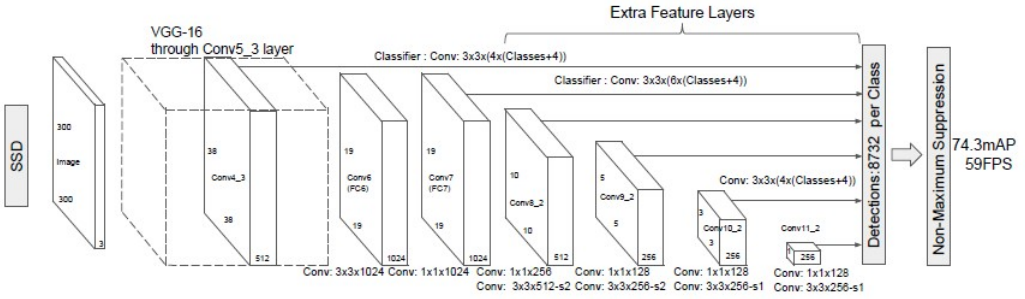
4.4.4 Ανίχνευση Αντικειμένων

Η ανίχνευση αντικειμένων (Object Detection) αποτελεί μία από τις βασικές εργασίες της Όρασης των Υπολογιστών [47] και η χρήση της πηγάζει από συστήματα Μηχανικής Μάθησης [51]. Έχει ως στόχο να διευκρινίσει σε ποιά θέση βρίσκεται ένα αντικείμενο σε μία εικόνα (εντοπισμός-localization) και σε ποια κατηγορία ανήκει (ταξινόμηση-classification). Σε μία εικόνα αυτό επιτυγχάνεται με τη χρήση πλαισίων οριοθέτησης (Bounding Box - BB) όπου δείχνει το ποσοστό ύπαρξης μίας κλάσης (αντικειμένου). Η δομή της γενικής ανίχνευσης αντικειμένων μπορεί να χωριστεί σε δύο κατηγορίες, αυτής της προτεινόμενης περιοχής (Region Proposal) με κάποια παραδείγματα μοντέλων όπως R-CNN (Regions with CNN) [57], Faster R-CNN [58] και αυτής της παλινδρόμησης/ταξινόμησης (Regression/Classification) με μοντέλα όπως YOLO [59] (You Only Look Once) και SSD [60] (Signle Shot MultiBox Detector). Αν και γενικά οι μέθοδοι Region Proposal έχουν καλύτερα αποτελέσματα σε ακρίβεια, οι μέθοδοι Regression επιτυγχάνουν καλύτερα αποτελέσματα σε χρόνους με κόστος στην ακρίβεια, βοηθώντας σε εφαρμογές πραγματικού χρόνου [61]. Για το λόγο αυτό, προτιμήθηκε ο SSD σε συνδυασμό με Mobilenet [62] για την υλοποίηση της ανίχνευσης αντικειμένων στον ρομποτικό βραχίονα και αναλύονται παρακάτω.

SSD - Single Shot Multibox Detector

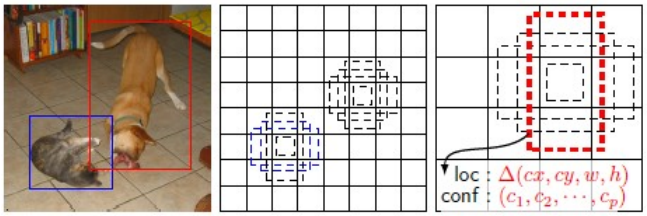
Ο SSD είναι ένας ανιχνευτής αντικειμένων, που βασίζεται σε βαθύ νευρωνικό δίκτυο και παράγει πλαίσια οριοθέτησης (Bounding Boxes-BB) και σκορ για κάθε κλάση που υφίσταται σε μία εικόνα. Το μοντέλο βασίζεται σε δίκτυα αναγνώρισης εικόνας (VGG-16 [63], Mo-

bilenet [62]) και έχουν προστεθεί συνελικτικά στρώματα τα οποία μειώνονται σε μέγεθος προοδευτικά, με στόχο τη πρόβλεψη σε διάφορες κλίμακες (Σχήμα 4.21). Κάθε στρώμα, πα-



Σχήμα 4.21: Η αρχιτεκτονική ενός μοντέλου SSD, όπου φαίνονται το VGG-16 δίκτυο σαν βάση και τα πρόσθετα στρώματα που χρησιμοποιούνται για τις προβλέψεις [60].

ράγει ένα σταθερό αριθμό προβλέψεων με τη χρήση φίλτρων και συγκριμένα χρησιμοποιείται ένα φίλτρο $3 \times 3 \times p$, όπου p το βάθος του στρώματος (κανάλια) που εφαρμόζεται το φίλτρο και παράγεται μία τιμή εξόδου. Οι έξοδοι των πλαισίων οριοθέτησης, είναι οι μετατοπίσεις σε σχέση με μία προκαθορισμένη θέση σε κάθε χάρτη χαρακτηριστικών. Στο κάθε κελί του χάρτη χαρακτηριστικών, υπολογίζεται η μετατόπιση στα προκαθορισμένα κουτιά του κελιού αυτού και το σκορ της ύπαρξης της κλάσης σε αυτά τα κουτιά (Σχήμα 4.22). Κατά τη διάρκεια της εκπαίδευσης, γίνεται υπολογισμός ποια προκαθορισμένα κουτιά αντιστοιχούν με τα αληθή κουτιά που δίνονται στην είσοδο. Χρησιμοποιούνται εκείνα τα κουτιά τα οποία υπερκαλύπτουν τα αληθή, με ένα όριο 0.5. Ο σκοπός αυτού είναι να προβλέπει πολλαπλά προκαθορισμένα κουτιά που αλληλοκαλύπτονται παρά να διαλέγει ένα με τη μεγαλύτερη επικάλυψη απλοποιώντας την εκπαίδευση. Για κάθε χάρτη χαρακτηριστικών χρησιμοποιούνται έξι προκαθορισμένα κουτιά και συνδυάζοντας προβλέψεις σε διαφορετικές κλίμακες και αναλογίες, καλύπτουν διαφορετικές εισόδους σε μέγεθος και μορφή (στο Σχήμα 4.22 η γάτα ταιριάζει σε δύο προκαθορισμένα κουτιά ενώ ο σκύλος μόνο σε ένα σε διαφορετική κλίμακα). Τα περισσότερα προκαθορισμένα κουτιά προκύπτουν αρνητικά (δεν ταιριάζουν με τα αληθή) και η στρατηγική που χρησιμοποιείται είναι να διατηρηθούν εκείνα τα οποία έχουν τη μεγαλύτερη απώλεια σιγουριάς με ένα ποσοστό 3:1 μεταξύ αρνητικών : θετικών. Έτσι είναι πιο σταθερή και γρήγορη η εκπαίδευση. Έστω ότι το i-οστό προκαθορισμένο κουτί ταιριάζει με το j-οστο αληθές κουτί της κλάσης p και $x_{ij}^{p} = 1, 0$ ένας δείκτης αυτών. Η συνάρτηση απώλειας του μοντέλου, είναι το σταθμισμένο άθροισμα της απώλειας εντοπισμού (localization



Σχήμα 4.22: Αριστερά, φαίνεται η εικόνα που ορίζεται ως είσοδος στο δίκτυο με τα αληθή κουτιά, για το κάθε αντικείμενο. Μέση, ένας χάρτης χαρακτηριστικών με μέγεθος 8×8 και 4 προκαθορισμένα κουτιά. Δεξιά, ένας χάρτης χαρακτηριστικών 4×4 και για κάθε προκαθορισμένο κουτί, γίνεται πρόβλεψη των μετατοπίσεων και το σκορ για όλες τις κατηγορίες. Φαίνεται η χρησιμότητα των διαφορετικών κλιμάκων, όπου το αληθές κουτί του σκύλου ταιριάζει με το προκαθορισμένο κουτί του χάρτη χαρακτηριστικών 4×4 (κόκκινο χρώμα) και σε κανένα εκείνου του χάρτη χαρακτηριστικών με μέγεθος 8×8 [60].

loss - loc) και της απώλειας σιγουριάς (confidence loss - conf).

$$L(x, c, l, g) = \frac{1}{N}(L_{conf}(x, c) + L_{loc}(x, l, g))$$

όπου N ο αριθμός των προκαθορισμένων κουτιών που ταιριάζουν με τα αληθή, c η σιγουριά των κλάσεων και l και g οι παράμετροι του κουτιού πρόβλεψης και του αληθές αντίστοιχα. Οι μετατοπίσεις των κουτιών είναι σε σχέση με το κέντρο (c_x, c_y) του προκαθορισμένου κουτιού και του ύψους h και του πλάτους του w. Η απώλεια εντοπισμού είναι η Smooth L1 απώλεια μεταξύ των παραμέτρων του κουτιού πρόβλεψης (l) από του αληθές κουτιού (g).

$$L_{loc}(x, l, g) = \sum_{i \in Pos}^{N} \sum_{m \in c_x, c_y, w, h} (x_{i,j}^{k} smooth_{L1}(l_i^m - \hat{g}_j^m)$$

$$\hat{g}_j^{cx} = (g_j^{cx} - d_i^{cx})/d_i^w$$

$$\hat{g}_j^{cy} = (g_j^{cy} - d_i^{cy})/d_i^h$$

$$\hat{g}_j^{w} = \log\left(\frac{g_j^w}{d_i^w}\right)$$

$$\hat{g}_j^{h} = \log\left(\frac{g_j^h}{d_i^h}\right)$$

όπου d το προκαθορισμένο κουτί και Smooth L1 [64] αποτελεί

$$Smooth_{L1}(x) = \begin{cases} \frac{1}{2}x^2, & if\ |x| < 1 \\ |x| - \frac{1}{2} & , otherwise \end{cases}$$

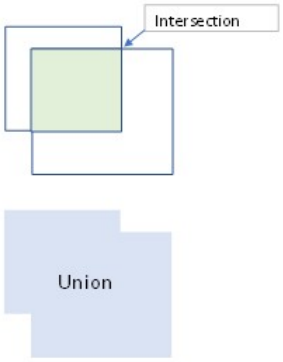
Η απώλεια σιγουριάς είναι η απώλεια Softmax [65] στα σκορ κάθε κλάσης (c)

$$L_{conf}(x,c) = -\sum_{i\in Pos}^{N} x_{ij}^{p} \log \hat{c}_i^p - \sum_{i\in Neg} \log \hat{c}_i^0$$

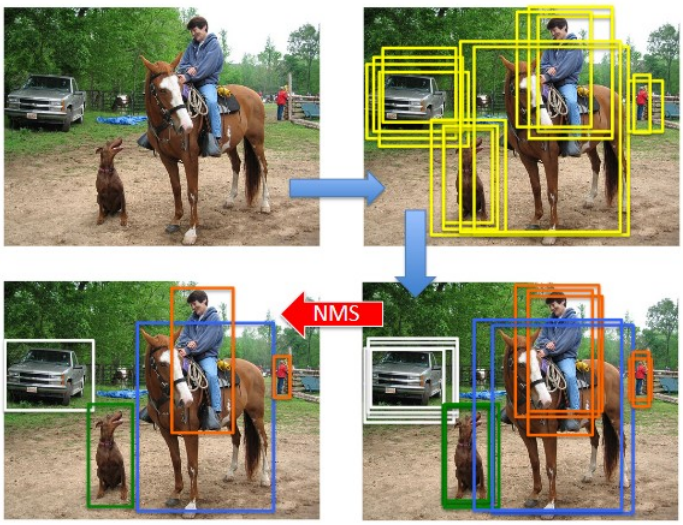
όπου

$$\hat{c}_i^p = \frac{\exp c_i^p}{\sum_p \exp c_i^p}$$

Για να παραχθούν τα τελικά αποτελέσματα, δηλαδή τα πλαίσια οριοθέτησης που ταιριάζουν καλύτερα σε κάθε αντικείμενο, χρησιμοποιείται η μη μέγιστη καταστολή (non-maximum suppression - NMS). Στον αλγόριθμο NMS [66], το κάθε πλαίσιο οριοθέτησης με το μεγαλύτερο σκορ σιγουριάς (confidence score), συγκρίνεται με τα υπόλοιπα ως προς το λόγο του κοινού μέρους προς την ένωση αυτών (Intersection over Union- IoU) (Σχήμα 4.23) και αν ξεπερνάει κάποιο κατώφλι (Nt) τότε απορρίπτεται. Με αυτό το τρόπο στο τέλος μένει εκείνο το πλαίσιο το οποίο οριοθετεί καλύτερα το αντικείμενο (Σχήμα 4.24).



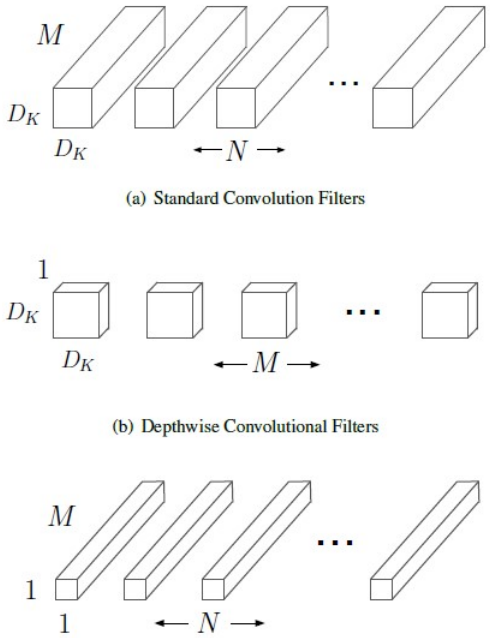
Σχήμα 4.23: Δύο κουτιά οριοθέτησης που αλληλοκαλύπτονται. Όταν το κοινό τους μέρος (πράσινο) προς την ένωση τους (γαλάζιο) ξεπερνάει ένα κατώφλι, τότε απορρίπτεται.



Σχήμα 4.24: Η χρήση του αλγορίθμου NMS οδηγεί στο βέλτιστο κουτί οριοθέτησης [66].

MobileNet

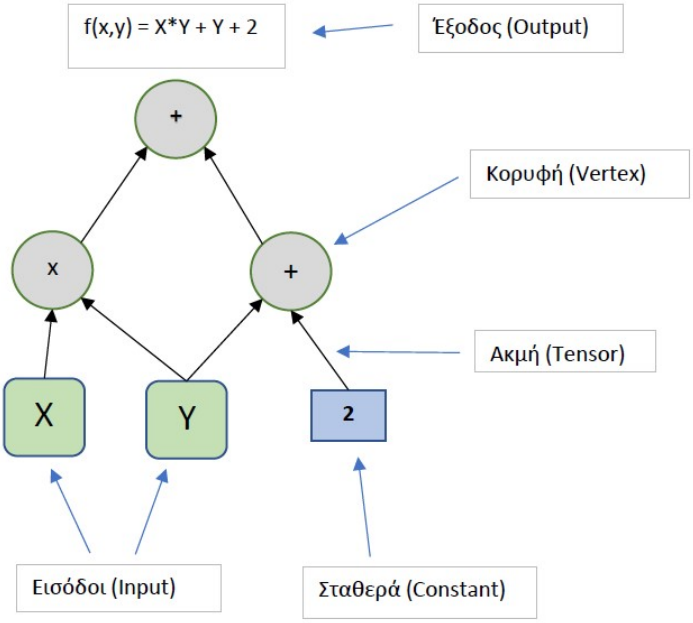
Τα MobileNet είναι συνελικτικά νευρωνικά δίκτυα για κινητές και ενσωματωμένες εφαρμογές όρασης. Στοχεύουν σε μικρού μεγέθους δίκτυα και σε μείωση της καθυστέρησης μεταφοράς (latency). Στα κοινά συνελικτικά δίκτυα η συνέλιξη φιλτράρει χαρακτηριστικά με τη χρήση πυρήνων-φίλτρων και με συνδυασμό αυτών δημιουργεί μία αναπαράσταση. Στο μοντέλο MobileNet, η παραπάνω διαδικασία εκτελείται μόνο στην είσοδο της εικόνας, ενώ στη συνέχεια χωρίζεται σε δύο βήματα με τη χρήση εις βάθους διαχωρίσιμων συνελίξεων (Depthwise Seperable Convolutions). Μία εις βάθος διαχωρίσιμη συνέλιξη αποτελείται από εις βάθος συνελίξεις(depthwise convolutions) και σημειακές συνελίξεις (pointwise convolutions). Στο Σχήμα 4.25 φαίνονται τα φίλτρα μίας απλής συνέλιξης, που αντικαθίστανται από τα φίλτρα της εις βάθους και σημειακής συνέλιξης. Η εις βάθος συνέλιξη μόνο φιλτράρει τα κανάλια εισόδου (βάθος εισόδου) με ένα φίλτρο $D_k \times D_k \times 1$ για κάθε κανάλι. Για τη δημιουργία νέων χαρακτηριστικών, είναι υπεύθυνη η σημειακή συνέλιξη όπου φιλτράρει την έξοδο της εις βάθους συνέλιξης με φίλτρα $1 \times 1 \times M$. Η υποδειγματοληψία επιτυγχάνεται με τη χρήση βηματισμού (stride) σε κάθε εις βάθους συνέλιξη και πριν το τελικό στρώμα εφαρμόζεται και υποδειγματοληψία μέσης τιμής (average pooling). Το τελικό στρώμα είναι ένα πλήρες συνδεδεμένο στρώμα (fully connected layer), δηλαδή ένα στρώμα όπου ο κάθε νευρώνας εξαρτάται από όλους τους νευρώνες του προηγούμενου στρώματος και μέσω ενός στρώματος νευρώνων Softmax γίνεται η κατηγοριοποίηση [62].



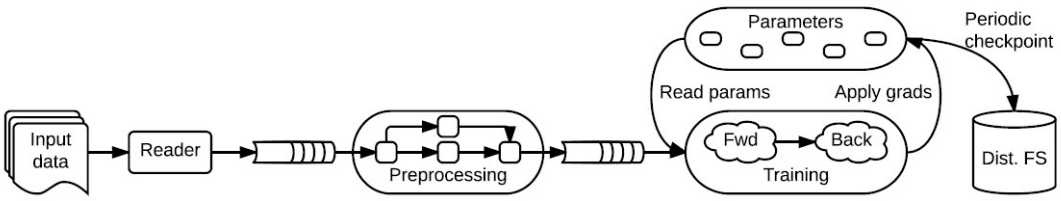
Σχήμα 4.25: a) Φίλτρα απλής συνέλιξης, b) Φίλτρα εις βάθους συνέλιξης, c) Φίλτρα σημειακής συνέλιξης [62].

Tensorflow

Το Tensorflow (Tensor + flow) είναι ένα σύστημα μηχανικής μάθησης, που χρησιμοποιεί γραφήματα ροής δεδομένων (dataflow graphs) για αριθμητικούς υπολογισμούς. Υποστηρίζει πολλές εφαρμογές, αλλά χρησιμοποιείται κυρίως για εκπαίδευση και χρήση βαθιών νευρωνικών δικτύων. Σε ένα ενοποιημένο γράφημα ροής δεδομένων, αναπαρίσταται ο υπολογισμός αλλά και η κατάσταση στην οποία επιχειρεί ένας αλγόριθμος μηχανικής μάθησης. Με το γράφημα είναι σαφής η σύνδεση των υπολογισμών και έτσι εκτελούνται ταυτόχρονα ανεξάρτητοι υπολογισμοί. Οι κορυφές (Vertices) του γραφήματος αποτελούν υπολογισμούς όπου μεταβάλλουν τη κατάσταση και οι ακμές (Edges) αποτελούν τις εισόδους και εξόδους από τις κορυφές. Στο tensorflow τα δεδομένα αναπαρίστανται ως ν-διάστατα διανύσματα, τα αποκαλούμενα tensors και μεταφέρονται από τις ακμές (Σχήμα 4.26). Οι κορυφές αποκαλούνται πράξεις (operations) και δέχονται tensors ως είσοδο και έξοδο. Επίσης χρησιμοποιούνται μεταβλητές (variables), οι οποίες είναι πράξεις (operations) χωρίς είσοδο αλλά μόνο έξοδο που χειρίζεται μία μνήμη (buffer). Σε αυτή τη μνήμη αποθηκεύονται και διαβάζονται οι παράμετροι του μοντέλου (βάρη και πολώσεις) ανάλογα με τη αναφορά που δέχεται. Ο συντονισμός επιτυγχάνεται με τη χρήση ουρών (queue) όπως FIFO (First In First Out). Ένα τυπικό παράδειγμα εκπαίδευσης φαίνεται στο Σχήμα 4.27 όπου εκτελούνται υπογραφήματα ταυ-



Σχήμα 4.26: Ένα απλοποιημένο παράδειγμα γραφήματος ροής δεδομένων.



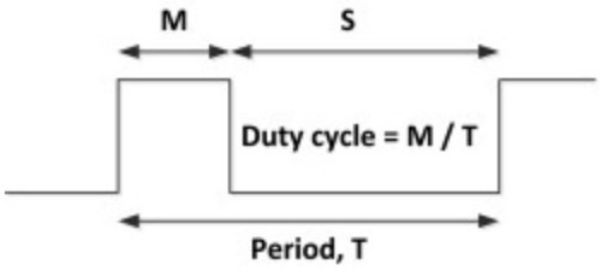
Σχήμα 4.27: Ένα γράφημα ροής δεδομένων Tensorflow με υπογραφήματα [67].

τόχρονα με τη χρήση μεταβλητών και ουρών. Το αρχικό υπογράφημα διαβάζει τις εισόδους, έπειτα γίνεται μία προεπεξεργασία πριν δοθούν στο βασικό τμήμα, όπου ένα υπογράφημα εκπαιδεύει τις παραμέτρους του ανάλογα με ταυτόχρονες παρτίδες που εισέρχονται. Επίσης χρησιμοποιείται και ένα υπογράφημα σημείου ελέγχου όπου ανά τακτά χρονικά διαστήματα αποθηκεύονται οι παράμετροι σε περίπτωση σφάλματος. Έτσι μπορεί μία εκπαίδευση μπορεί να συνεχίσει από το τελευταίο σημείο ελέγχου που είχε αποθηκευτεί [67].

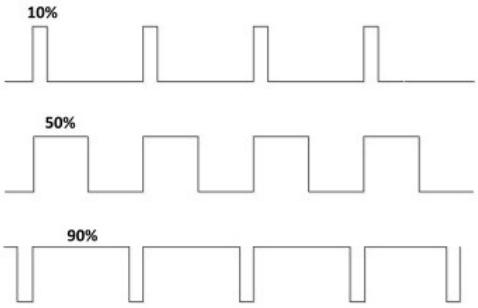
4.5 Ανάλυση κινήσεων

4.5.1 Παλμός Διαμόρφωσης Πλάτους

Ο παλμός διαμόρφωσης πλάτους (Pulse Width Modulation - PWM), χρησιμοποιείται για τον έλεγχο αναλογικών κυκλωμάτων με ψηφιακές εξόδους. Οι έξοδοι αυτοί στη συγκεκριμένη εργασία, είναι οι ακροδέκτες (pins) του Raspberry Pi . Ένα σήμα PWM, είναι ένα τετραγωνικό σήμα σε μορφή κύματος και προσδιορίζει για πόσο χρόνο ένα σήμα θα μένει ενεργό ή ανενεργό. Στο Σχήμα 4.28 όπου φαίνεται ένα σήμα PWM, ως Duty Cycle ορίζεται ως ο λόγος του ενεργού σήματος (M) προς τη περίοδο του (T) και κυμαίνεται από 0 έως 1 και S είναι ο χρόνος όπου το σήμα είναι ανενεργό [68]. Για τον έλεγχο των μοτέρ, αλλάζοντας το Duty Cycle αλλάζει η μέση τάση που τροφοδοτεί το αντίστοιχο κύκλωμα. Για παράδειγμα η τάση τροφοδοσίας στον οδηγό μοτέρ (L298N), είναι 12 βολτ.



Σχήμα 4.28: Ένας παλμός διαμόρφωσης πλάτους [68].



Σχήμα 4.29: Διάφορες τιμές του Duty Cycle [68].

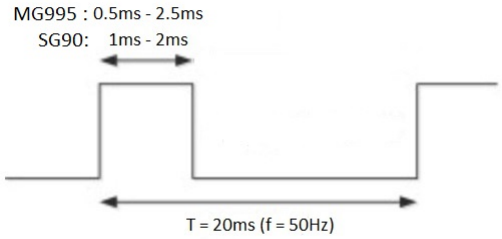
4.5.2 Έλεγχος Μοτέρ

Για τον έλεγχο των μοτέρ, χρησιμοποιήθηκε ο οδηγός L298N. Στο κεφάλαιο 3 αναφέρθηκε η σύνδεση του οδηγού με το Raspberry Pi και τα μοτέρ. Οι ακροδέκτες EnA και EnB του οδηγού, δέχονται PWM σήμα και αλλάζοντας το Duty Cycle αλλάζει η μέση τάση που τροφοδοτεί τα αντίστοιχα μοτέρ. Για παράδειγμα η τάση εισόδου στον οδηγό L298N είναι 12 βολτ και με σήμα PWM 0.5 Duty Cycle, η τάση στους ακροδέκτες των μοτέρ θα είναι 6 βολτ. Τροποποιώντας τη τάση αλλάζει και η ταχύτητα περιστροφής των μοτέρ, ελέγχοντας έτσι τη ταχύτητα της πλατφόρμας.

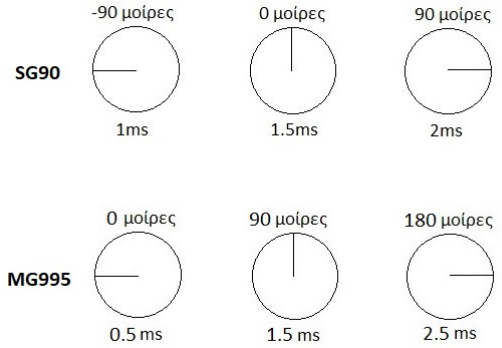
Στους ακροδέκτες In1, In2, In3, In4 ορίζεται η φορά κίνησης των μοτέρ, ελέγχοντας ουσιαστικά τις εισόδους στο κύκλωμα του L298N, και ανάλογα με το όρισμα ως HIGH ή LOW, οι έξοδοι OUT1 και OUT2 του κάθε μοτέρ δίνουν τη τάση για να κινούνται δεξιόστροφα ή αριστερόστροφα. Για παράδειγμα θέτοντας In1 = LOW και In2 = HIGH, ο μότορας A θα κινηθεί εμπρός, ενώ αν τεθεί In1 = HIGH και In2 = LOW, ο μότορας A θα κινηθεί προς τα πίσω. Τα τέσσερα μοτέρ της πλατφόρμας, είναι χωρισμένα σε δύο ομάδες, αριστερά και δεξιά. Για να κινηθεί μπροστά ή πίσω όλα τα μοτέρ έχουν τη ίδια φορά, αλλά για να στρίψει η πλατφόρμα, τότε έχουν αντίθετη φορά. Για παράδειγμα, όταν θέλει να στρίψει αριστερά, τα μοτέρ της αριστερής μεριάς έχουν φορά προς τα πίσω ενώ της δεξιάς μεριάς προς τα μπροστά. Έτσι στρέφεται χωρίς όμως να κινείται εμπρός ή πίσω.

4.5.3 Έλεγχος Σερβομηχανισμών

Για τους σερβομηχανισμούς, το duty cycle του σήματος PWM μεταφράζεται σε αντίστοιχη γωνία. Για το σερβομηχανισμού SG90, που είναι εκείνο της δαγκάνας, η περίοδος T του σήματος είναι 20ms που αντιστοιχεί σε συχνότητα f = 50 Hz και το ενεργό σήμα κυμαίνεται από 1ms έως 2ms (Σχήμα 4.30). Αντίστοιχα για τον έλεγχο του σερβομηχανισμού MG995, που είναι εκείνο της βάσης και του βραχίονα, η περίοδος T του σήματος είναι 20ms και το ενεργό σήμα κυμαίνεται από 0.5ms έως 2.5ms (Σχήμα 4.31). Στο Σχήμα 4.29 φαίνονται οι τρεις χαρακτηριστικές θέσεις των σερβομηχανισμών ανάλογα με το πόσο χρόνο είναι ενεργό το σήμα PWM.



Σχήμα 4.30: Ο παλμός διαμόρφωσης πλάτους για το χειρισμό των σερβομηχανισμων SG90 και MG995.



Σχήμα 4.31: Οι τρεις γωνίες των σερβομηχανισμών SG90 και MG995 ανάλογα με το Duty Cycle.

Κεφάλαιο 5

Χρήση Λογισμικού και Μεθοδολογία

5.1 Εισαγωγή

Στο παρόν κεφάλαιο περιγράφεται το λογισμικό και η μεθοδολογία όπου χρησιμοποιήθηκε. Αρχικά γίνεται αναφορά στο περιβάλλον όπου ο ρομποτικός βραχίονας λειτουργούσε, τη προετοιμασία του Raspberry Pi και στη συνέχεια αναλύεται η διαδικασία ολόκληρης της λειτουργίας του ρομποτικού βραχίονα.

5.2 Περιβάλλον

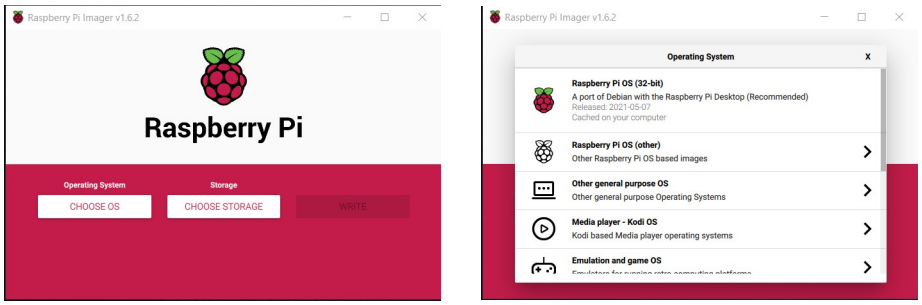
Το περιβάλλον όπου δρα ο ρομποτικός βραχίονας, είναι εσωτερικός χώρος, λόγω του περιορισμού με τη τροφοδοσία ρεύματος. Κινείται σε συγκεκριμένο χώρο και ο στόχος του θα είναι να εντοπίσει κουτάκια αναψυκτικού να τα συλλέξει και να αφήνει σε ένα συγκεκριμένο σημείο.

5.3 Λογισμικό του Raspberry Pi

Το λογισμικό του Raspberry Pi είναι το Raspbian, μία έκδοση Linux ειδικά σχεδιασμένη για τη πλατφόρμα και είναι φορτωμένο σε κάρτα μνήμης Micro SD. Παρακάτω περιγράφονται λεπτομερώς τα βήματα της προετοιμασίας.

1. Από την επίσημη ιστοσελίδα του Raspberry Pi γίνεται λήψη της εφαρμογής Raspberry Pi Imager, όπου επιτρέπει την εγκατάσταση στην κάρτα SD το λογισμικό Raspbian και εγκαθίστανται στον υπολογιστή.

2. Με την εκτέλεση της εφαρμογής εμφανίζεται ένα νέο παράθυρο. Στο πεδίο Choose Os, επιλέγεται ποια έκδοση εξυπηρετεί. Επιλέχθηκε η Raspberry Pi with Desktop έκδοση (1η επιλογή), ώστε να ρυθμιστεί το Raspberry μέσω μίας οθόνης και αποτελεί προσέγγιση, φιλική ως προς το χρήστη. Στη συνέχεια μέσω της επιλογής Choose Storage, επιλέγεται η κάρτα Micro SD, όπου έχει εισαχθεί στον υπολογιστή και τέλος με το Write ξεκινάει η εγγραφή (Σχήμα 5.1).



(α□) Αρχική οθόνη της εφαρμογής. (β□) Επιλογή της έκδοσης.

Σχήμα 5.1: Η εφαρμογή Raspberry Pi Imager.

3. Μόλις ολοκληρωθεί η εγκατάσταση, αφαιρείται η κάρτα μνήμης και τοποθετείται στο Raspberry Pi. Η σύνδεση με την οθόνη επιτυγχάνεται μέσω θύρας HDMI και το τροφοδοτικό στη θύρα USB Type C και μόλις εκκινήσει, πρέπει να εκτελεστούν τα βήματα για τις ρυθμίσεις που εμφανίζει. Μόλις ολοκληρωθούν οι ρυθμίσεις, εμφανίζεται η επιφάνεια εργασίας και το Raspberry Pi είναι έτοιμο προς χρήση. Ακολουθούν τα βήματα για την απομακρυσμένη σύνδεση με τον υπολογιστή ώστε να μην χρειάζεται να συνδέεται κάθε φορά το HDMI καλώδιο.

4. Αρχικά συνδέεται το Raspberry Pi στο τοπικό δίκτυο και σε ένα τερματικό (terminal) εκτελούνται οι παρακάτω εντολές όπως φαίνεται στο Σχήμα 5.2, για να εγκατασταθούν τυχόν αναβαθμίσεις του λογισμικού. Έπειτα εκτελείται η παρακάτω εντολή (Σχήμα 5.3) για να εγκατασταθεί το πακέτο xrdp, το οποίο επιτρέπει να συνδεθεί ο χρήστης απομακρυσμένα στο τοπικό δίκτυο. Με αυτό το βήμα, το Raspberry Pi είναι έτοιμο και δεν χρειάζεται κάποια άλλη ενέργεια.

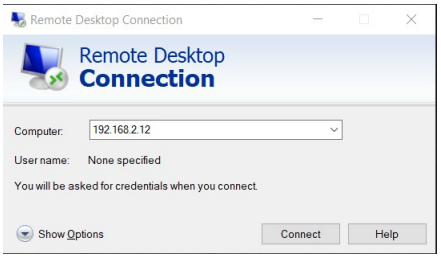
```
sudo apt-get update
sudo apt-get upgrade
```

Σχήμα 5.2: Εντολές για αναβάθμιση του λογισμικού.

```
sudo apt-get install xrdp
```

Σχήμα 5.3: Εντολή για εγκατάσταση του πακέτου xrdp.

5. Στον υπολογιστή εκτελείται η προεγκατεστημένη εφαρμογή, απομακρυσμένη σύνδεση (Remote Desktop Connection) και εμφανίζεται το Σχήμα 5.4. Η διεύθυνση IP που έχει το Raspberry Pi, μπορεί να βρεθεί από τις ρυθμίσεις του ρούτερ ή με τη χρήση της εντολής «nslookup raspberrypi» στη γραμμή εντολών του υπολογιστή και τοποθετείται στο πεδίο ”Computer” και ολοκληρώνεται η σύνδεση. Εμφανίζεται ένα παράθυρο όπου ζητάει όνομα χρήστη και κωδικό. Το όνομα, είναι προεπιλεγμένα το «pi» και ο κωδικός είναι εκείνος όπου επιλέχθηκε στις αρχικές ρυθμίσεις. Πατώντας σύνδεση, εμφανίζεται η επιφάνεια εργασίας του Raspberry Pi και έτσι η σύνδεση με τον υπολογιστή ολοκληρώθηκε επιτυχώς.



Σχήμα 5.4: Η εφαρμογή απομακρυσμένης σύνδεσης.

5.4 Επικοινωνία Υπολογιστή - Ρομποτικού Βραχίονα

Στον υπολογιστή, μέσω της Usb κάμερας, θα εκτελείται η αναγνώριση των αντικειμένων και θα στέλνει οδηγίες-εντολές στο Ρομποτικό βραχίονα, μέσω σύνδεσης TCP/IP Python Sockets. Παρακάτω αναλύονται τα τμήματα του κώδικα στο raspberry pi και στον υπολογιστή που είναι υπεύθυνα για τη σύνδεση τους.

5.4.1 Ρομποτικός βραχίονας

Το raspberry pi θα αποτελεί το «σέρβερ» στη συγκεκριμένη περίπτωση και θα συνδέεται ο υπολογιστής σαν «πελάτης» (client). Παρακάτω αναλύεται το τμήμα του κώδικα του raspberry pi για τη σύνδεση:

Αρχικά εισάγεται η βιβλιοθήκη socket, ένα δομοστοιχείο (module) για τη χρήση σημείων

σύνδεσης επικοινωνιών και η βιβλιοθήκη threading, ένα δομοστοιχείο για τη χρήση νημάτων, ροής εκτέλεσης ενός προγράμματος. Επίσης τοποθετείται και η διεύθυνση IP που έχει το raspberry pi στο τοπικό δίκτυο και μία θύρα της επιλογής του χρήστη.

```
import socket
import threading

HOST = '192.168.2.12'
PORT = 8485
```

Στη συνέχεια, ορίζεται ότι η σύνδεση θα είναι TCP ότι το σημείο σύνδεσης θα έχει τη συγκεκριμένη διεύθυνση και θύρα όπου τέθηκε προηγουμένως.

```
rPiSocket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
print('Socket created')
rPiSocket.bind((HOST, PORT))
print('Socket bind complete')
```

Παρακάτω χρησιμοποιείται η κλάση ClientThread όπου επιτρέπει τη σύνδεση με έναν «πελάτη» (client) και να δέχεται μηνύματα συνεχώς. Τα μηνύματα θα είναι τύπου συμβολοσειρών.

```
class ClientThread(threading.Thread):
    def __init__(self,clientAddress,clientsocket):
        threading.Thread.__init__(self)
        self.csocket = clientsocket
        print ("New connection established: ", clientAddress)

    def run(self):
        print("Connection from : ", clientAddress)
        msg = ''
        while True:
            data = self.csocket.recv(2048)
            msg = data.decode()



        self.csocket.send(bytes(msg, 'UTF-8'))
    print("Pc at ", clientAddress, " terminated the connection.")
```

```
# Accept connections
while True:
    rPiSocket.listen(1)
    (clientConnected, clientAddress) = rPiSocket.accept()

    newthread = ClientThread(clientAddress, clientConnected)
    newthread.start()
```

5.4.2 Υπολογιστής

Αφού εισαχθεί η βιβλιοθήκη socket, ορίζεται η διεύθυνση IP και η θύρα που έχουν εκχωρηθεί στο Raspberry Pi.

```
import socket

# Connecting Pc to Robotic Arm
HOST = '192.168.2.12'  # the Robotic Arms' IP
PORT = 8485    # the Robotic Arms' port
```

Στη συνέχεια ορίζεται ότι η σύνδεση θα είναι TCP και θα δοθεί διεύθυνση IP και θύρα .

```
# Create Socket
pc_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

Τέλος γίνεται σύνδεση και στέλνονται τα δεδομένα (go,left,right,back κλπ).

```
# Connect to Robotic Arm
pc_socket.connect((HOST, PORT))
# Send the message to Robotic Arm
pc_socket.send(data.encode())
```

5.5 Ανίχνευση Αντικειμένων

Η ανίχνευση αντικειμένων εκτελείται εξολοκλήρου στον υπολογιστή. Χρησιμοποιήθηκε το PyCharm, ένα λογισμικό ανάπτυξης προγραμμάτων (IDE - Integrated Development En-

vironment), και δημιουργήθηκε ένα περιβάλλον όπου χρησιμοποιήθηκε Python 3.8, Tensorflow 2.4.1 και OpenCV 4.5.1. Η εγκατάσταση και η εκπαίδευση του ειδικού μοντέλου ανίχνευσης αντικειμένων, βασίστηκε στο διδακτικό υλικό «Tensorflow 2 Object Detection API» [69] [70]. Παρακάτω περιγράφονται οι διαφοροποιήσεις στη διαδικασία που χρειάστηκαν για το ρομποτικό βραχίονα.

5.5.1 Προετοιμασία Δεδομένων

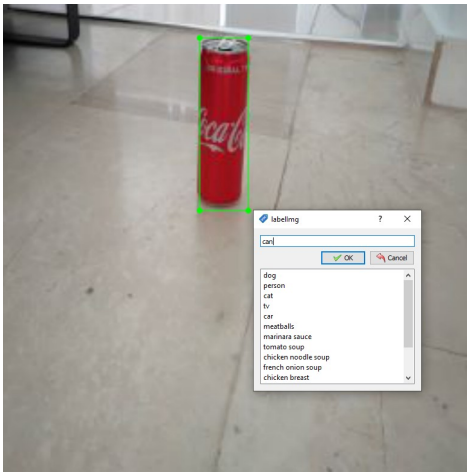
Στο διαδίκτυο υπάρχουν έτοιμα σετ εικόνων (dataset) όπου μπορούν χρησιμοποιηθούν για την εκπαίδευση ενός μοντέλου ανίχνευσης αντικειμένων. Στη συγκεκριμένη περίπτωση χρειάστηκε να ετοιμαστεί συγκεκριμένο σετ εικόνων. Ο στόχος του ρομποτικού βραχίονα είναι να εντοπίσει κουτιά αλουμινίου. Τα δεδομένα επομένως που χρησιμοποιήθηκαν, είναι 138 εικόνες από κουτάκια αναψυκτικού. Είναι σημαντικό οι εικόνες να έχουν διαφορετικά φόντα και οι διαστάσεις τους να ταιριάζουν με τις διαστάσεις που χρησιμοποιεί το μοντέλο (Σχήμα 5.5). Μεγαλύτερα σετ εικόνων, οδηγούν σε καλύτερα μοντέλα. Μια λύση στην επέκταση του σετ είναι οι εικόνες να επεξεργαστούν όπως αλλαγές στην απόχρωση (κλίμακα του γκρι, διαφορετικές τιμές RGB), στον προσανατολισμό (κατακόρυφα, οριζόντια και με διαφορετικές γωνίες) και στη θέση (το κουτάκι αναψυκτικού να βρίσκεται σε διαφορετικά σημεία στις εικόνες) [47].

Για κάθε φωτογραφία πρέπει να δημιουργηθεί ένα αρχείο xml, που δημιουργούνται από την εφαρμογή labelimg (Σχήμα 5.6). Τα αρχεία αυτά περιέχουν πληροφορίες για το οριοθετημένο κουτί της κάθε εικόνας, όπως σε τι κλάση ανήκει (can-κουτάκι αλουμινίου στη περίπτωση της εργασίας) και τις συντεταγμένες των τεσσάρων άκρων του (Σχήμα 5.7) .

5.5.2 Εκπαίδευση Μοντέλου

Το ειδικό μοντέλο του ρομποτικού βραχίονα βασίστηκε στο SSD MobileNet V2 FPN-Lite 320x320 [71], ένα προ-εκπαίδευσης μοντέλο (pre-trained model), το οποίο δεν απαιτεί μεγάλη επεξεργαστική ισχύ, ιδανικό για φορητές συσκευές όπως το Raspberry Pi, αλλά με κόστος στην ακρίβεια της ανίχνευσης αντικειμένων. Αυτό που χρειάζεται από το έτοιμο μοντέλο, είναι το pipeline.config αρχείο, στο οποίο βρίσκονται οι παράμετροι για την εκπαίδευση του ειδικού μοντέλου. Στο διδακτικό υλικό, αναφέρονται λεπτομερώς ποιες γραμμές κώδικα χρειάζονται αλλαγές και οι τιμές για το ρομποτικό βραχίονα που διαφέρουν, είναι οι εξής:

Σχήμα 5.5: Ενδεικτικές εικόνες των δεδομένων.



Σχήμα 5.6: Στιγμιότυπο της εφαρμογής LabelImg.

- batch size : 4 (Ο αριθμός των εικόνων που χρησιμοποιούνται σε μία επανάληψη στην εκπαίδευση)

- total steps : 4000 (Μια εποχή στο μοντέλο θεωρείται 100 steps άρα ορίζονται 40 epochs)

```
<annotation>
        <folder>cola_dataset</folder>
        <filename>1.jpg</filename>
        <path>C:\Users\cz_av\Pictures\cola_dataset\1.jpg</path>
        <source>
                <database>Unknown</database>
        </source>
        <size>
                <width>320</width>
                <height>320</height>
                <depth>3</depth>
        </size>
        <segmented>0</segmented>
        <object>
                <name>can</name>
                <pose>Unspecified</pose>
                <truncated>0</truncated>
                <difficult>0</difficult>
                <bndbox>
                        <xmin>30</xmin>
                        <ymin>27</ymin>
                        <xmax>292</xmax>
                        <ymax>290</ymax>
                </bndbox>
        </object>
</annotation>
```

Σχήμα 5.7: Περιεχόμενο ενός xml αρχείου.

- learning rate base : 0.001 (Παράμετρος για την εκπαίδευση μέσω της μεθόδου Back-Propagation)
- warmup steps : 20

Η εκπαίδευση του μοντέλου, μπορεί να γίνει στον υπολογιστή, αλλά ο χρόνος εκτέλεσης εξαρτάται από την ισχύ του επεξεργαστή και της κάρτας γραφικών του υπολογιστή. Για να αποφευχθεί ο μεγάλος χρόνος εκτέλεσης και η πιθανότητα αποτυχίας, χρησιμοποιήθηκε το Google Colaboratory [72], ένας επεξεργαστής κειμένου Python, όπου επιτρέπει τη χρήση κάρτας γραφικών και την εκτέλεση κώδικα διαδικτυακά, επιταχύνοντας έτσι τη διαδικασία της εκπαίδευσης. Μόλις η εκπαίδευση ολοκληρωθεί χρειάζονται τα περιεχόμενα του φακέλου myModel στο φάκελο ExportedModels, που έχουν δημιουργηθεί από τη διαδικασία και τα τοποθετούνται στο φάκελο myModels με την εξής δομή.

```
tensorflow
└─data
  └─models
    └─myModel
```

Συνεχίζοντας με τον παρακάτω κώδικα, χρησιμοποιώντας OpenCV και μία κάμερα USB θα γίνεται ο εντοπισμός του κουτιού αλουμινίου.

5.5.3 Κώδικας για την ανίχνευση αντικειμένων

Αρχικά εισάγονται οι βιβλιοθήκες που χρειάζονται για την ανίχνευση αντικειμένων

```
import os
```

```
import cv2
import numpy as np
import tensorflow as tf
from object_detection.utils import label_map_util
from object_detection.utils import config_util
from object_detection.utils import visualization_utils
                                        as viz_utils
from object_detection.builders import model_builder
```

Στη συνέχεια τίθενται σε σίγαση τυχόν πληροφορίες και προειδοποιήσεις που μπορεί να εμφανιστούν κατά τη διάρκεια της εκτέλεσης και ορίζονται τα αρχεία του μοντέλου που θα φορτωθούν για την ανίχνευση αντικειμένων.

```
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'

DATA_DIR = os.path.join(os.getcwd(), 'data')
MODELS_DIR = os.path.join(DATA_DIR, 'models')

MODEL_NAME = 'my_model'
PATH_TO_CKPT = os.path.join(MODELS_DIR,
                os.path.join(MODEL_NAME, 'checkpoint/'))
PATH_TO_CFG = os.path.join(MODELS_DIR,
                os.path.join(MODEL_NAME, 'pipeline.config'))

# Download labels file
LABEL_FILENAME = 'label_map.pbtxt'
PATH_TO_LABELS = os.path.join(MODELS_DIR,
                os.path.join(MODEL_NAME, LABEL_FILENAME))
```

Παρακάτω ενεργοποιείται η δυνατότητα να χρησιμοποιηθεί περισσότερη μνήμη από τη κάρτα γραφικών, φορτώνεται το αρχείο διαμόρφωσης και χτίζεται το μοντέλο. Στη συνέχεια φορτώνεται το τελευταίο σημείο ελέγχου που είχε δημιουργηθεί κατά τη διάρκεια της εκπαίδευσης.

```
# Enable GPU dynamic memory allocation
gpus = tf.config.experimental.list_physical_devices('GPU')
for gpu in gpus:
```

```
    tf.config.experimental.set_memory_growth(gpu, True)

# Load pipeline config and build a detection model
configs =
config_util.get_configs_from_pipeline_file(PATH_TO_CFG)
model_config = configs['model']
detection_model =
model_builder.build(model_config=model_config, is_training=False)

# Restore checkpoint
ckpt = tf.compat.v2.train.Checkpoint(model=detection_model)
ckpt.restore(os.path.join(PATH_TO_CKPT, 'ckpt-0')).expect_partial()
```

Στη συνέχεια ορίζεται η συνάρτηση η οποία θα ανιχνεύει τα αντικείμενα. Δέχεται ως είσοδο την εικόνα σαν ένα τετραδιάστατο διάνυσμα tensor, υλοποιεί μια προεπεξεργασία της εικόνας με διάφορες τροποποιήσεις, δημιουργεί τις προβλέψεις των κουτιών και τέλος μέσω της μετεπεξεργασίας υπολογίζει τις συνολικές σωστές προβλέψεις.

```
def detect_fn(image):
  image, shapes = detection_model.preprocess(image)
  prediction_dict = detection_model.predict(image, shapes)
  detections = detection_model.postprocess(prediction_dict, shapes)

  return detections, prediction_dict, tf.reshape(shapes, [-1])
```

Και τέλος φορτώνεται το αρχείο όπου περιέχει τις κλάσεις που αντιστοιχούν στους αριθμούς που έχει ως έξοδο το μοντέλο. Στη συγκεκριμένη περίπτωση υπάρχει μόνο μία κλάση, οπότε στο αρχείο αυτό υπάρχει η αντιστοίχιση του αριθμού 1 στη κλάση «Can». Εάν υπήρχαν παραπάνω από μία κλάσεις τότε θα υπήρχαν αντίστοιχα παραπάνω αριθμοί για την αντιστοίχιση τους.

```
category_index =
label_map_util.create_category_index_from_labelmap(PATH_TO_LABELS,
        use_display_name=True)
```

Σε αυτό το σημείο ορίζεται η πηγή της λήψης εικόνων. Επειδή ο υπολογιστής είναι φορητός και έχει μία κάμερα τοποθετημένη στην οθόνη χρειάζεται εικόνα από τη USB κάμερα, οπότε

τίθεται 1 στην εντολή VideoCapture και όχι 0. Στη συνέχεια ορίζεται ένα κουτί με μηδενικά άκρα που θα χρησιμοποιηθεί αργότερα και ξεκινάει ένας βρόγχος εντολών.

```
# Define the video stream
cap = cv2.VideoCapture(1)

# Define a box
box = (0, 0, 0, 0)
```

Στον βρόγχο παρακάτω ξεκινάει να μετράει ο χρόνος ώστε να υπολογιστεί ο ρυθμός ανανέωσης του κάθε καρέ που λαμβάνεται αμέσως μετά. Στη συνέχεια οι διαστάσεις της εικόνας μετατρέπονται στις διαστάσεις που απαιτούνται από το μοντέλο με τρία κανάλια όπως αναφέρθηκε στο Κεφάλαιο 4 και η νέα εικόνα μετατρέπεται σε διανύσματα tensor. Αυτά τα διανύσματα αποτελούν τις εισόδους στη συνάρτηση που θα ανιχνεύσει τα αντικείμενα.

```
while True:
  # Start timer
  timer = cv2.getTickCount()
  # Read frame from camera
  ret, image_np = cap.read()

  # Expand dimensions of images to shape: [1, None, None, 3]
  image_np_expanded = np.expand_dims(image_np, axis=0)

  #Convert the input image to tensors
  input_tensor = tf.convert_to_tensor(np.expand_dims(image_np, 0),
                                      dtype=tf.float32)
  detections, predictions_dict, shapes = detect_fn(input_tensor)
```

Αφού έχει ολοκληρωθεί η ανίχνευση, η αρχική εικόνα (καρέ βίντεο), επεξεργάζεται με τη παρακάτω συνάρτηση για να εμφανιστούν τα κουτιά οριοθέτησης στο κάθε αντικείμενο. Μέσω των παραμέτρων μπορεί να ρυθμιστεί πόσα κουτιά να εμφανιστούν ανάλογα με το κατώφλι και τον αριθμό των επιθυμητών κουτιών. Οι συντεταγμένες των κουτιών είναι μεταξύ 0 και 1.

```
  label_id_offset = 1
  image_np_with_detections = image_np.copy()
```

```
viz_utils.visualize_boxes_and_labels_on_image_array(
      image_np_with_detections,
      detections['detection_boxes'][0].numpy(),
      (detections['detection_classes'][0].numpy() +
                          label_id_offset).astype(int),
      detections['detection_scores'][0].numpy(),
      category_index,
      use_normalized_coordinates=True,
      max_boxes_to_draw=2,
      min_score_thresh=.70,
      agnostic_mode=False)
```

Μόλις ολοκληρωθεί η διαδικασία τοποθέτησης των κουτιών, υπολογίζεται ο χρόνος εκτέλεσης για την εμφάνιση του ρυθμού ανανέωσης (εάν είναι επιθυμητό) και εμφανίζεται στην οθόνη του υπολογιστή η εικόνα από τη κάμερα με τα κουτιά εφόσον έχουν ανιχνευθεί αντικείμενα, αλλιώς δείχνει ότι βλέπει η κάμερα εκείνη τη στιγμή. Ο τερματισμός του προγράμματος εκτελείται όταν πατηθεί το κουμπί Q.

```
# Calculate Frames per second (FPS) and display on frame
fps = cv2.getTickFrequency() / (cv2.getTickCount() - timer)
cv2.putText(image_np_with_detections, "FPS : "
  + str(int(fps)), (100, 50), cv2.FONT_HERSHEY_SIMPLEX, 0.75,
              (50, 170, 50), 2)

# Display output
cv2.imshow('object detection', image_np_with_detections)

if cv2.waitKey(25) & 0xFF == ord('q'):
    break
```

Όταν ολοκληρωθεί ο βρόγχος τότε με τις παρακάτω εντολές κλείνει η κάμερα και ότι άλλο παράθυρο έχει ανοίξει από το πρόγραμμα.

```
cap.release()
cv2.destroyAllWindows()
```

5.6 Κίνηση του Ρομποτικού Βραχίονα

Όπως αναφέρθηκε παραπάνω, οι οδηγίες για την κίνηση του ρομποτικού βραχίονα, στέλνονται από τον υπολογιστή. Αρχικά ο βραχίονας με τη δαγκάνα, παραμένει στην αρχική του θέση, και λαμβάνει εντολή να κινηθεί προς τα εμπρός. Μόλις εντοπίσει κάποιο κουτάκι αλουμινίου, σταματάει τη κίνηση στιγμιαία και ανάλογα με το κέντρο του κουτιού που δημιουργείται από το αλγόριθμο ανίχνευσης αντικειμένων, δίδονται και οι αντίστοιχες εντολές από τον υπολογιστή. Επειδή με τα μοτέρ η κίνηση στη πλατφόρμα δεν είναι ακριβής, δημιουργώντας εκτροπή προς τα δεξιά ή αριστερά, η κίνηση γίνεται διακοπτόμενα με τη χρήση της εντολής «sleep», επιτυγχάνοντας καλύτερο έλεγχο και καλύτερο εντοπισμό. Ο στόχος είναι ο ρομποτικός βραχίονας να έρθει σε τέτοια απόσταση από το κουτάκι αλουμινίου, ώστε μετά να αναλάβει ο βραχίονας και να το πιάσει με τη δαγκάνα. Αφού έχει λάβει τη σωστή θέση, τα μοτέρ σταματάνε και κινούνται οι σερβομηχανισμοί. Η δαγκάνα έχει προκαθορισμένη κίνηση σε συγκεκριμένες μοίρες. Όταν πλησιάσει το στόχο, το πιάνει με τη δαγκάνα, στρέφει δεξιά 90 μοίρες και το αφήνει. Στη συνέχεια συνεχίζει την αναζήτηση για υπόλοιπα κουτάκια αλουμινίου.

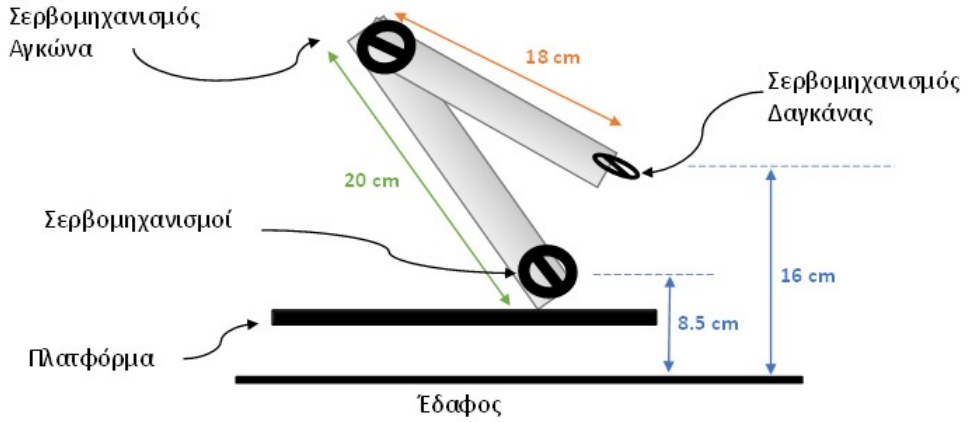
5.6.1 Έλεγχος του βραχίονα

Η κίνηση του βραχίονα γίνεται με τη χρήση σερβομηχανισμών. Η ουδέτερη θέση φαίνεται στην Εικόνα 5.8. Στην αρχή ανοίγει η δαγκάνα και έπειτα στρέφεται πρώτα ο αγκώνας και μετά η βάση. Ο σκοπός είναι να φτάσει η δαγκάνα σχεδόν στο πάτωμα και να πιάσει το κουτάκι αναψυκτικού, να το σηκώσει και κρατώντας το να κινηθεί το ρομπότ στη θέση όπου θα το αφήσει.

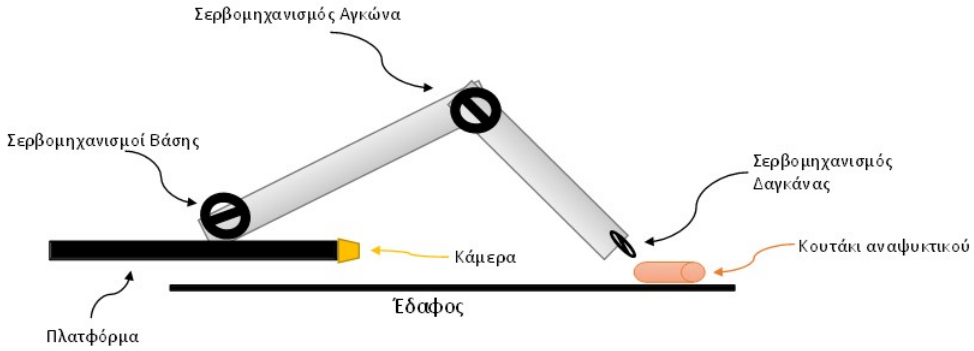
Κώδικας ελέγχου του βραχίονα

Ο παρακάτω κώδικας αποτελεί μέρος του κώδικα RpiControl που εκτελείται στο Raspberry Pi και είναι υπεύθυνος για τη ρύθμιση και τον έλεγχο των σερβομηχανισμών. Μόλις η πλατφόρμα πάρει τη σωστή θέση σε σχέση με το αντικείμενο προς συλλογή, ο βραχίονας αναλαμβάνει να το συλλέξει, να το κρατήσει και να το αφήσει δεξιά του.

```
import RPi.GPIO as GPIO


servoBase1 = 23
```



Σχήμα 5.8: Η αρχική θέση του βραχίονα. Η θέση αυτή διατηρείται και κατά τη διάρκεια αναζήτησης του ρομπότ.



Σχήμα 5.9: Η θέση του βραχίονα για τη συλλογή του αντικειμένου.

```
servoBase2 = 24
servoArm = 25
servoGrip = 26


GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
GPIO.setup(servoGrip, GPIO.OUT)
GPIO.setup(servoBase1, GPIO.OUT)
GPIO.setup(servoBase2, GPIO.OUT)
GPIO.setup(servoArm, GPIO.OUT)
```

```
base1 = GPIO.PWM(servoBase1, 50) # GPIO 23 for PWM with 50Hz
base2 = GPIO.PWM(servoBase2, 50) # GPIO 24 for PWM with 50Hz
arm = GPIO.PWM(servoArm, 50) # GPIO 25 for PWM with 50Hz
gripper = GPIO.PWM(servoGrip, 50) # GPIO 26 for PWM with 50Hz

gripper.start(0)
arm.start(0)
base1.start(0)
base2.start(0)
```

Στη συγκεκριμένη συνάρτηση η επιθυμητή γωνία μεταφράζεται σε duty για το PWM σήμα που ελέγχει τους σερβομηχανισμούς.

```
def DesiredAngle(angle):
    duty = angle/18 +2.5

    return duty
```

Ο παρακάτω κώδικας εκτελείται για τη συλλογή του αντικειμένου. Αρχικά ανοίγει η δαγκάνα, σηκώνεται ο αγκώνας και κατεβαίνει ο βραχίονας προς το αντικείμενο. Μόλις κατέβει κλείνει η δαγκάνα και σηκώνεται ο βραχίονας κρατώντας το αντικείμενο.

```
print("Initiating Grabbing Procedure")
# Prepare the Gripper
posGrip = DesiredAngle(120)
gripper.ChangeDutyCycle(posGrip)
time.sleep(1)

# Raise the Arm
posArm = DesiredAngle(110)
arm.ChangeDutyCycle(posArm)
time.sleep(1)

# Moving the arm towards the target
for i in range(1,6):
```

```
    ang = 32 + i*7
    posBase = DesiredAngle(ang)
    base1.ChangeDutyCycle(posBase)
    base2.ChangeDutyCycle(posBase)
    time.sleep(0.5)

# Grab the target
posGrip = DesiredAngle(30)
gripper.ChangeDutyCycle(posGrip)
time.sleep(1)

# Moving the arm back after grabbing it
posBase = DesiredAngle(20)
base1.ChangeDutyCycle(posBase)
base2.ChangeDutyCycle(posBase)
time.sleep(1)
print("Grabbing Accomplished")
holding = 1
```

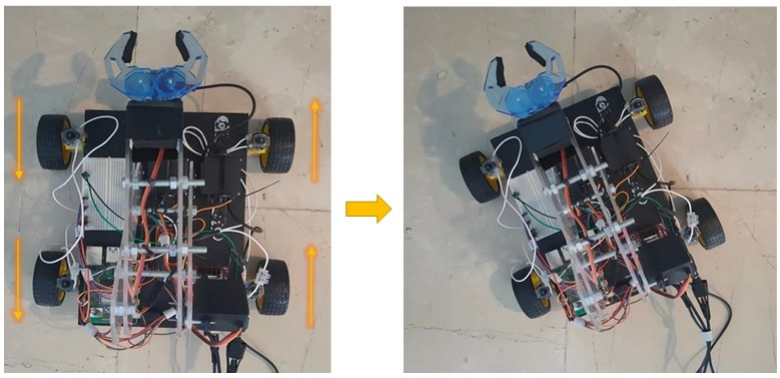
Μόλις έχει συλλεχθεί το αντικείμενο αναλαμβάνει ο παρακάτω κώδικας. Αρχικά στρέφεται δεξιά περίπου 90 μοίρες, αφήνει το αντικείμενο ανοίγοντας τη δαγκάνα, και χαμηλώνει τον αγκώνα. Στη συνέχεια στρίβει 90 μοίρες αριστερά για να συνεχίσει την αναζήτηση για νέα αντικείμενα.

```
if holding == 1: # The target is grabbed. Maintain the angles.
print("Desposing the can")
for t in range(0,3):
    SteerThirty('right')
    sleep(0.5)

posGrip = DesiredAngle(120) # Drop the can
gripper.ChangeDutyCycle(posGrip)
time.sleep(0.5)

posArm = DesiredAngle(170) # Lower the Arm
```

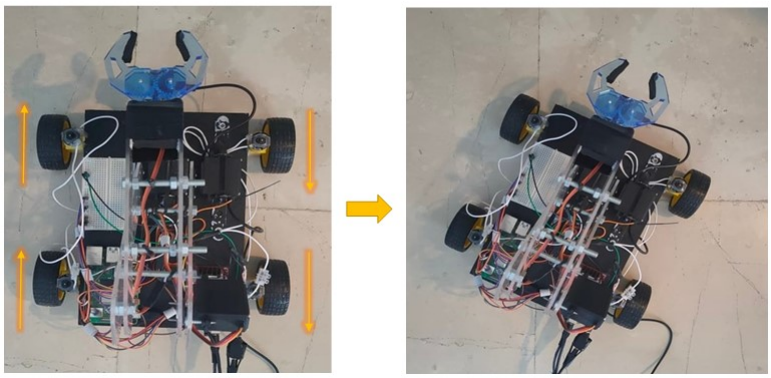
```
arm.ChangeDutyCycle(posArm)
time.sleep(0.5)

posBase = DesiredAngle(5) # Neutral position
base1.ChangeDutyCycle(posBase)
base2.ChangeDutyCycle(posBase)
time.sleep(0.5)

print("Continuing")
for t in range(0,3):
    SteerThirty('left')
    sleep(0.5)

holding=0
break
```

5.6.2 Έλεγχος της πλατφόρμας

Η πλατφόρμα κινείται με τέσσερα μοτέρ τα οποία δεν έχουν την δυνατότητα να στρέφονται. Η στροφή της πλατφόρμας επιτυγχάνεται με τη ταυτόχρονη αντίθετη κίνηση των τροχών (Εικόνα 5.10 και 5.11). Ο έλεγχος της φοράς γίνεται με βάση τη τάση που εφαρμόζεται στα άκρα του κάθε μοτέρ.



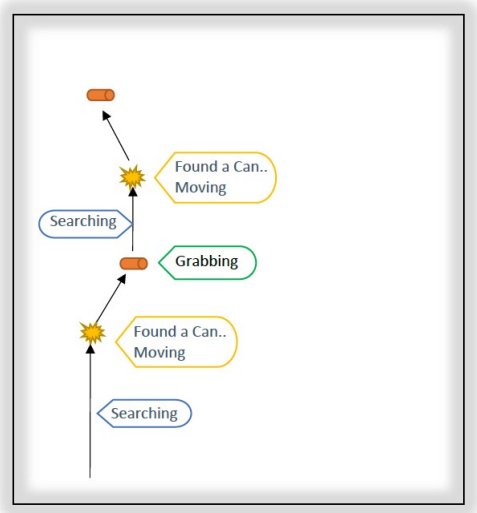
Σχήμα 5.10: Αριστερή στροφή με ταυτόχρονη κίνηση πίσω και εμπρός των αριστερών και δεξιών ροδών αντίστοιχα.



Σχήμα 5.11: Δεξιά στροφή με ταυτόχρονη κίνηση εμπρός και πίσω των αριστερών και δεξιών ροδών αντίστοιχα.

Κώδικας καθοδήγησης για τη συλλογή αντικειμένου

Ο συγκεκριμένος κώδικας αποτελεί μέρος του κώδικα που εκτελείται στον υπολογιστή. Αρχικά ο ρομποτικός βραχίονας ψάχνει για κάποιο κουτάκι αλουμινίου Εάν έχει εντοπιστεί κάποιο κουτάκι αλουμινίου, τότε ο υπολογιστής ανάλογα με το κέντρο του κουτιού οριοθέτησης που έχει δημιουργηθεί, δίνει οδηγίες να κινηθεί ανάλογα με τη σχετική του θέση από το κέντρο εκείνο (Εικόνα 5.12). Έχοντας υπολογιστεί η απόσταση η οποία οδηγεί σε επιτυχημένη συλλογή του αντικειμένου, ο ρομποτικός βραχίονας κινείται μέχρι το κέντρο του κουτιού να λάβει τις επιθυμητές συντεταγμένες.

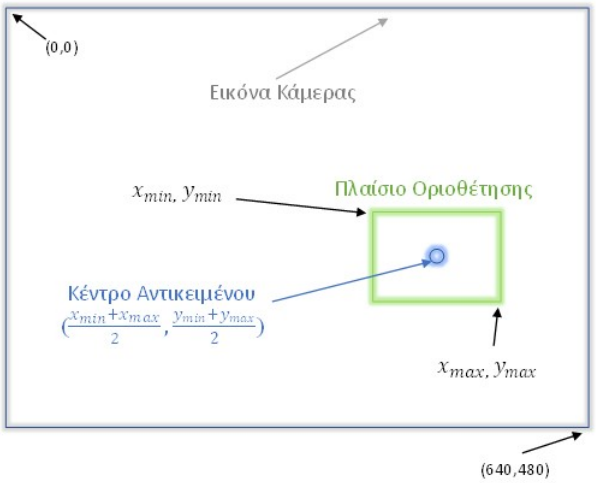


Σχήμα 5.12: Προσομοίωση λειτουργίας του ρομποτικού βραχίονα στο εσωτερικό χώρο.

Αρχικά όταν εντοπισθούν αντικείμενα δημιουργούνται κουτιά και επιλέγονται εκείνα τα οποία έχουν σκορ πάνω από το κατώφλι που έχει ορισθεί (0.7 στη περίπτωση της εργασίας).

```
boxes = np.squeeze(detections['detection_boxes'])
scores = np.squeeze(detections['detection_scores'])

# set a min thresh score
min_score_thresh = 0.7
# get the boxes with score > threshold
bboxes = boxes[scores > min_score_thresh]
```

Όταν εντοπισθεί κάποιο κουτάκι, αρχικά υπολογίζονται οι συντεταγμένες του. Επειδή όπως προαναφέρθηκε προηγουμένως, οι τιμές είναι μεταξύ 0 και 1, πρέπει να πολλαπλασιαστούν με τις διαστάσεις που λαμβάνει η κάμερα και στη συγκεκριμένη περίπτωση είναι 640x480. Στη συνέχεια υπολογίζονται οι συντεταγμένες του κέντρου του κουτιού και εμφανίζονται στην εικόνα της κάμερας με ένα κύκλο (Εικόνα 5.13).



Σχήμα 5.13: Στιγμιότυπο κατά τη διάρκεια ανίχνευσης.

```
# If we have found a can then load the coordinates of the box
if bboxes.any():
    yMin, xMin, yMax, xMax = bboxes[0]
    box = (int(xMin * 640), int(yMin * 480),
                    int(xMax * 640), int(yMax * 480))

    # Draw a circle in the center of the detection bounding box
    dc1 = int((box[0] + box[2])/2)
    dc2 = int((box[1] + box[3])/2)
```

```
cv2.circle(image_np_with_detections, (dc1, dc2), 4,
                                        (255, 0, 0), -1)

# Print the centroid coordinates on the image
text = "x: " + str(dc1) + ", y: " + str(dc2)
cv2.putText(image_np_with_detections, text,
    (dc1 - 10, dc2 - 10),
        cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 0, 0), 2)
```

Εφόσον είναι γνωστή η θέση του αντικειμένου προς συλλογή, ο υπολογιστής στέλνει εντολές να κινηθεί ανάλογα της θέσης του από τη προέκταση της δαγκάνας. Αφήνοντας κάποιο περιθώριο, περίπου 30 εικονοστοιχεία, η πλατφόρμα θα κινείται μέχρις ότου το κέντρο του κουτιού οριοθέτησης του αντικειμένου να είναι μεταξύ 280 με 310. Το κομμάτι που έχει πιο μεγάλο αντίκτυπο είναι ο άξονας y, διότι με αυτό καθορίζεται η απόσταση του αντικειμένου από το βραχίονα. Όταν το κέντρο του κουτιού είναι μεταξύ 340 με 350 εικονοστοιχεία, τότε η δαγκάνα είναι στο σωστό σημείο για τη συλλογή του. Μόλις η πλατφόρμα φτάσει στο σημείο αυτό τότε στέλνει μήνυμα στο Raspberry Pi να σταματήσει να κινείται.

```
# Directions for centering the Object on x-axis of frame
if dc1 > 310:  # if the object is right
    data = "Right"
    cv2.putText(image_np_with_detections,
        "Moving : " + str('Right'), (100, 80),
            cv2.FONT_HERSHEY_SIMPLEX, 0.75,(0, 0, 0), 2)
elif dc1 < 280:  # if the object is left
    data = "Left"
    cv2.putText(image_np_with_detections,
        "Moving : " + str('Left'), (100, 80),
            cv2.FONT_HERSHEY_SIMPLEX, 0.75,(0, 0, 0), 2)
else:  # the object is at the desired position
    data = 'half'
# Send data to server
pc_socket.send(data.encode())

# Directions for centering the Object on y-axis of frame
```

```
    if dc2 > 350:   # if the object is near
        data = "Back"
        cv2.putText(image_np_with_detections,
            "Moving : " + str('Back'), (100, 140),
                cv2.FONT_HERSHEY_SIMPLEX,0.75,(0, 0, 0), 2)
    elif dc2 < 340:  # if the object is far
        data = "Go"
        cv2.putText(image_np_with_detections,
            "Moving : " + str('Forward'), (100, 50),
                cv2.FONT_HERSHEY_SIMPLEX,0.75,(0, 0, 0), 2)
    else:  # the object is at the desired position
        if data == 'half':
            data = 'full'
            cv2.putText(image_np_with_detections,
                "Status : " + str('Stoped'), (100, 50),
                    cv2.FONT_HERSHEY_SIMPLEX,0.75,(0, 0, 0), 2)


    # Send data to server
    pc_socket.send(data.encode())

else:
    cv2.putText(image_np_with_detections,
        "Status : " + str('Searching'), (100, 50),
            cv2.FONT_HERSHEY_SIMPLEX, 0.75,(0, 0, 0), 2)

    data = " "
    pc_socket.send(data.encode())
```

Κώδικας ελέγχου μοτέρ

Παρακάτω αναλύεται ο κώδικας MotorDriver ο οποίος αρχικοποιεί τις συνδέσεις των ακροδεκτών του Raspberry Pi και του οδηγού των μοτέρ και ορίζει τις εντολές για την ενεργοποίηση, τη φορά κίνησης, τη ταχύτητα και απενεργοποίηση των μοτέρ. Ο κώδικας αυτός

εκτελείται στο Raspberry Pi και δεν επηρεάζεται από τον υπολογιστή. Αρχικά εισάγονται οι βιβλιοθήκες, ορίζεται ως σύστημα ορισμού των ακροδεκτών, οι αριθμοί GPIO του Raspberry Pi και δεν θα εμφανίζονται προειδοποιήσεις στη κονσόλα.

```
import RPi.GPIO as GPIO
from time import sleep
from pynput.keyboard import Key, Listener

GPIO.setmode(GPIO.BCM)  # Use GPIO numbering
GPIO.setwarnings(False) # Do not show warnings
GPIO.cleanup()          # Exit if nothing is called
```

Στη συνέχεια δημιουργείται η κλάση Motor. Αρχικά ορίζεται η συνάρτηση αρχικοποίησης της κλάσης όπου ρυθμίζονται οι συνδέσεις του οδηγού L298N ως έξοδοι του Raspberry Pi και στη συνέχεια ενεργοποιείται το σήμα PWM με 100 Hz συχνότητα.

```
class Motor():
    def __init__(self,EnaA,In1A,In2A,EnaB,In1B,In2B):
        GPIO.setmode(GPIO.BCM)
        GPIO.setwarnings(False)

        #--The inputs of Motor Driver---
        self.EnaA = EnaA
        self.In1A = In1A
        self.In2A = In2A
        self.EnaB = EnaB
        self.In1B = In1B
        self.In2B = In2B

        #----GPIO Outputs-----
        GPIO.setup(self.EnaA,GPIO.OUT)
        GPIO.setup(self.In1A,GPIO.OUT)
        GPIO.setup(self.In2A,GPIO.OUT)
        GPIO.setup(self.EnaB,GPIO.OUT)
        GPIO.setup(self.In1B,GPIO.OUT)
        GPIO.setup(self.In2B,GPIO.OUT)
```

```
        #---Initialize PWM on 100Hz frequency
        #---Start PWM with 0% duty cycle
        self.pwmA = GPIO.PWM(self.EnaA, 100);
        self.pwmA.start(0);
        self.pwmB = GPIO.PWM(self.EnaB, 100);
        self.pwmB.start(0);
```

Ο σκοπός της συνάρτησης move() είναι να ελέγχει τη φορά και τη ταχύτητα των μοτέρ. Η τιμή του turn καθορίζει αν θα κινηθεί δεξιά, αριστερά, εμπρός ή πίσω και η τιμή του speed καθορίζει τη ταχύτητα περιστροφής. Ανάλογα τι τίθεται στα In1 και In2 του κάθε μότορα καθορίζει τη φορά και η τιμή του Duty Cycle τη ταχύτητα.

```
    def move(self,speed=0.5,turn=0,t=0):
        speed *=100
        leftSpeed = speed
        rightSpeed = speed

        if turn == -1:      #-----Left
            leftSpeed = -speed
            rightSpeed = speed
        elif turn == 1:     #-----Right
            leftSpeed = speed
            rightSpeed = -speed
        elif turn == -0.5: #-----Backwards
            leftSpeed = -speed
            rightSpeed = -speed
        else:
            leftSpeed = speed #--Forward
            rightSpeed = speed

        #---Max Duty Cycle 100-----
        if leftSpeed>100: leftSpeed=100
        elif leftSpeed<-100: leftSpeed= -100
        if rightSpeed>100: rightSpeed=100
```

```
        elif rightSpeed<-100: rightSpeed= -100

        #--- Make the signal active----
        self.pwmA.ChangeDutyCycle(abs(leftSpeed))
        self.pwmB.ChangeDutyCycle(abs(rightSpeed))

        if leftSpeed>0: #---- Left wheels going forward
            GPIO.output(self.In1A,GPIO.HIGH)
            GPIO.output(self.In2A,GPIO.LOW)
        else:           #---- Left wheels going backwards
            GPIO.output(self.In1A,GPIO.LOW)
            GPIO.output(self.In2A,GPIO.HIGH)

        if rightSpeed>0: #---- Right wheels going forward
            GPIO.output(self.In1B,GPIO.LOW)
            GPIO.output(self.In2B,GPIO.HIGH)
        else:            #---- Right wheels going backwards
            GPIO.output(self.In1B,GPIO.HIGH)
            GPIO.output(self.In2B,GPIO.LOW)
        if t>0:
            sleep(t)
```

Η συνάρτηση Stop απενεργοποιεί τα μοτέρ θέτοντας 0 τη τάση στα άκρα τους.

```
    #----- Stop the motors-----
    def stop(self,t=0):
        self.pwmA.ChangeDutyCycle(0);
        self.pwmB.ChangeDutyCycle(0);
        sleep(t)
```

Τέλος στη συνάρτηση main δημιουργείται ένα αντικείμενο τύπου Motor με ορίσματα τους ακροδέκτες που συνδέεται το Raspberry Pi και ο οδηγός μοτέρ όπως σημειώθηκε και στο Κεφάλαιο 3.

```
if __name__ == '__main__':
    motor= Motor(4,3,2,22,17,27)
```

Κεφάλαιο 6

Συμπεράσματα

Σε αυτό το Κεφάλαιο συνοψίζεται η μελέτη του ρομποτικού βραχίονα, καθώς αναφέρονται τα αποτελέσματα, οι δυσκολίες που αντιμετωπίσθηκαν, οι μελλοντικές επεκτάσεις και εφαρμογές που δύναται να υλοποιηθούν.

6.1 Σύνοψη και συμπεράσματα

Ο σκοπός της συγκεκριμένης εργασίας ήταν να μελετηθεί και να κατασκευαστεί ένας αυτόνομος κινούμενος ρομποτικός βραχίονας για συλλογή αντικειμένων. Ο έλεγχος των συστημάτων του ρομποτικού βραχίονα καθώς επίσης και η επικοινωνία με τον υπολογιστή βασίστηκε στο Raspberry Pi 4, ενώ η ανίχνευση αντικειμένων εκτελούταν στον υπολογιστή. Το ειδικό μοντέλο που χρησιμοποιήθηκε στη μελέτη βασίστηκε στο προεκπαιδευμένο μοντέλο SSD MobileNet V2 και εκπαιδεύτηκε να εντοπίζει κουτιά αλουμινίου. Οι δοκιμές έδειχναν ότι ανίχνευε κουτάκια αλουμινίου με 70% μέσο όρο ακρίβεια με τη ταχύτητα αναγνώρισης να διατηρούταν σε ικανοποιητικά επίπεδα χωρίς να επηρεάζεται η κίνηση του ρομπότ προς το αντικείμενο.

6.2 Προβλήματα που εντοπίστηκαν

Η υλοποίηση ενός τέτοιου ρομπότ, εμφανίζει κάποιες δυσκολίες και περιορισμούς λόγω κατασκευής αλλά και ποιότητας υλικών. Αρχικά το Raspbery Pi σαν πλακέτα είναι πολύ ευαίσθητη σε στατικούς ηλεκτρισμούς και απαιτεί καλή τροφοδοσία για την αποφυγή φθοράς μέχρι και ανεπανόρθωτης βλάβης. Από μεριά κατασκευής, οι μη ακριβείς τοποθετήσεις (σε

βαθμό χιλιοστών) των σερβομηχανισμών αλλά και η παραμικρή κίνηση λόγω μη απόλυτης σταθερότητας τους, οδηγεί σε τρέμουλο κατά τη διάρκεια του χειρισμού του βραχίονα. Αυτό οφείλεται επίσης και στο Raspberry Pi λόγω των σημάτων που στέλνει, αλλά διορθώνεται σε μεγάλο βαθμό με τη χρήση βιβλιοθηκών. Τα μοτέρ που χρησιμοποιήθηκαν είχαν αρκετή ισχύ για τη κίνηση της πλατφόρμας, αλλά η ακρίβεια στις στροφές δεν ήταν ικανοποιητική και υπήρχε εκτροπή που οδηγούσε σε λάθος κατεύθυνση. Η δαγκάνα λόγω της σταθερής θέσης και η μη δυνατότητα στροφής, περιόριζε το ποσοστό επιτυχίας συλλογής του αντικειμένου. Επίσης αρχικά χρησιμοποιήθηκε κάμερα του Raspberry Pi και να στέλνει την εικόνα στον υπολογιστή ασύρματα. Η ανανέωση της εικόνας όμως καθυστερούσε τουλάχιστον 10 δευτερόλεπτα με αποτέλεσμα να μην μπορεί να εκτελεστεί η ανίχνευση αντικειμένων και έτσι χρησιμοποιήθηκε μία ενσύρματη USB κάμερα.

6.3 Μελλοντικές επεκτάσεις

Λόγω περιορισμένου χρόνου, χώρου αλλά και κόστους, αρκετές ιδέες δεν πραγματοποιήθηκαν, χωρίς να διακυβευτεί η λειτουργία του ρομποτικού βραχίονα. Πιο συγκεκριμένα, η χρήση ενός έτοιμου βραχίονα κατασκευασμένου μέσω τρισδιάστατης εκτύπωσης θα εξασφάλιζε πιο σταθερές και ακριβείς κινήσεις για τη συλλογή του αντικειμένου. Η τοποθέτηση ενός κάδου στο πίσω μέρος, είναι απαραίτητη για τη συλλογή, αλλά απαιτεί μεγαλύτερη κατασκευή. Για τη κίνηση της πλατφόρμας, η αποφυγή εμποδίων που μπορεί να υπάρξουν σε πραγματικές συνθήκες θα μπορούσε να επιτευχθεί με τη χρήση υπερηχητικών αισθητήρων και το πρόβλημα της ακριβής γνώσης της θέσης αλλά και της κατεύθυνσης θα λυνόταν με τη χρήση συστήματος GPS ή ηλεκτρονικής πυξίδας. Για τη λειτουργία σε εξωτερικό χώρο, πρέπει να λυθεί το θέμα της τροφοδοσίας και της κάμερας. Η χρήση μίας ασύρματης Wi-Fi κάμερας και μιας αξιόπιστης και συγκεκριμένων προδιαγραφών μπαταρίας θα έλυνε το πρόβλημα της ενσύρματης σύνδεσης.

Η εξέλιξη του συγκεκριμένου ρομποτικού βραχίονα να ανιχνεύει περισσότερα αντικείμενα όπως χαρτιά, πλαστικά μπουκάλια, μάσκες προστασίας και άλλων σκουπιδιών που συναντάται συχνά σε μία πόλη, θα βοηθούσε κατά πολύ το έργο της υπηρεσίας καθαρισμού. Επιπρόσθετα με τις παραπάνω βελτιώσεις που αναφέρθηκαν, θα μπορούσε να χρησιμοποιηθεί σε αρκετές ακόμη εφαρμογές, όπως επιτήρηση και καθαρισμός μονοπατιών ενός δάσους με στόχο την αποτροπή δημιουργίας εστιών φωτιάς.

Βιβλιογραφία

[1] Shreyas Kulkarni and Sarang Junghare. Robot based indoor autonomous trash detection algorithm using ultrasonic sensors. pages 1–5, 12 2013.

[2] Debasmita Mukherjee, Kashish Gupta, Li Chang, and Homayoun Najjaran. A survey of robot learning strategies for human-robot collaboration in industrial settings. *Robotics and Computer-Integrated Manufacturing*, 73:102231, 02 2022.

[3] Mary Alatise and Gerhard Hancke. A review on challenges of autonomous mobile robot and sensor fusion methods. *IEEE Access*, PP:1–1, 02 2020.

[4] Chao Duan, Steffen Junginger, Jiahao Huang, Kairong Jin, and Kerstin Thurow. Deep learning for visual slam in transportation robotics: A review. *Transportation Safety and Environment*, 1:177–184, 12 2019.

[5] Ahmed S. Al-Araji Noor Abdul Khaleq Zghair. A one decade survey of autonomous mobile robot systems. *International Journal of Electrical and Computer Engineering*, 11, Dec. 2021.

[6] Momomi Kanamura, Yuki Suga, and Tetsuya Ogata. Development of a basic educational kit for robot development using deep neural networks. pages 1360–1363, 01 2020.

[7] Christian Scharfenberger, Steven L. Waslander, John S. Zelek, and David A. Clausi. Existence detection of objects in images for robot vision using saliency histogram features. In *2013 International Conference on Computer and Robot Vision*, pages 75–82, 2013.

[8] Junhui Wu, Dong Yin, Jie Chen, Yusheng Wu, Huiping Si, and Kaiyan Lin. A survey on monocular 3d object detection algorithms based on deep learning. *Journal of Physics: Conference Series*, 1518:012049, 04 2020.

[9] Politikos Dimitris, Fakiris Elias, Davvetas Athanasios, Klampanos Iraklis, and Papatheodorou George. Automatic detection of seafloor marine litter using towed camera images and deep learning. *Marine Pollution Bulletin*, 164:111974, 03 2021.

[10] Lu Jiayuan. Space robot motion planning oriented to autonomous capture path. *Journal of Physics: Conference Series*, 1939(1):012118, may 2021.

[11] Alexi Delgado Taylor Sevillano-Visarga, Brian Meneses-Claudio. Design of a robotic arm window cleaner at the heights to avoid accidents due to falls. *International Journal of Emerging Technology and Advanced Engineering*, 11:70–80, Oct 2021.

[12] Zhi Li, Peter Moran, Qingyuan Dong, Ryan Shaw, and Kris Hauser. Development of a tele-nursing mobile manipulator for remote care-giving in quarantine areas. pages 3581–3586, 05 2017.

[13] Arturo Rankin, Mark Maimone, Jeffrey Biesiadecki, Nikunj Patel, Dan Levine, and Olivier Toupet. Mars curiosity rover mobility trends during the first 7 years. *Journal of Field Robotics*, 38(5):759–800, 2021.

[14] Michel L'Hour, Vincent Creuze, and Denis Dégez. Purpose-built robotic tools and methods for deep-sea archaeology. 08 2021.

[15] Deng Lino Li Jing, Yin Jialin. A robot vision navigation method using deep learning in edge computing environment. *EURASIP Journal on Advances in Signal Processing*, 22, May 2021.

[16] Anggit Wijanarko, Andri Prima Nugroho, Lilik Sutiarso, and Takashi Okayasu. Development of mobile robovision with stereo camera for automatic crop growth monitoring in plant factory. volume 2202, page 020100, 12 2019.

[17] Kang and Chen. Fruit detection and segmentation for apple harvesting using visual sensor in orchards. *Sensors*, 19(20):4599, Oct 2019.

[18] Sylwia Majchrowska, Agnieszka Mikołajczyk, Maria Ferlin, Zuzanna Klawikowska, Marta A. Plantykow, Arkadiusz Kwasigroch, and Karol Majek. Waste detection in pomerania: non-profit project for detecting waste in environment, 2021.

[19] Mohammad Saeed Rad, Andreas von Kaenel, Andre Droux, Francois Tieche, Nabil Ouerhani, Hazım Kemal Ekenel, and Jean-Philippe Thiran. A computer vision system to localize and classify wastes on the streets. *Computer Vision Systems*, page 195–204, 2017.

[20] Chengkai Yao. Floating garbage collector based on OpenMV. *Journal of Physics: Conference Series*, 1952(3):032058, jun 2021.

[21] Shuyuan Sun Yujiang Chen. Research and design of small semi-automated beach-lawn cleaning vehicle by optimizing the vibrating screen module. *Journal of Physics: Conference Series*, 1952(3):032066, jun 2021.

[22] Zhengxue Zhou, Leihui Li, Alexander Fürsterling, Hjalte Durocher, Jesper Mouridsen, and Xuping Zhang. Learning-based object detection and localization for a mobile robot manipulator in sme production. *Robotics and Computer-Integrated Manufacturing*, 73, 08 2021.

[23] Saidi Mazna, Hosni Ammar, Kharusi Ali, Montesines Nagayo Analene, Jamisola Rodrigo, and P.V. Bindu. Autonomous trash collector robot with wireless charging system in a campus environment. 06 2019.

[24] Bai Jinqiang, Lian Shiguo, Liu Zhaoxiang, Wang Kai, and Liu Dijun. Deep learning based robot for automatically picking up garbage on the grass. *IEEE Transactions on Consumer Electronics*, PP:1–1, 07 2018.

[25] A. Gasparetto. Robots in history: Legends and prototypes from ancient times to the industrial revolution. In *Explorations in the History of Machines and Mechanisms*, volume 32, pages 39–49. Springer, 2016.

[26] Teun Koetsier. *The Ascent of GIM, the Global Intelligent Machine*. Springer, Cham, Amsterdam, The Netherlands, 1 edition, 2019.

[27] M. Vidyasagar Mark W. Spong, Seth Hutchinson. *Robot Modeling and Control*. Jown Wiley and Sons, Ltd, 2 edition, 2020.

[28] A. Gasparetto and Lorenzo Scalera. A brief history of industrial robotics in the 20th century. *Advances in Historical Studies*, 08:24–35, Jan. 2019.

[29] Benjamin Kuipers, Edward A. Feigenbaum, Peter E. Hart, and Nils J. Nilsson. Shakey: From conception to history. *AI Magazine*, 38(1):88–103, Mar. 2017.

[30] Ahmad Alwan. Project design and management of programmable logic controllers for electrical technology. *IJES*, 2:322–333, 09 2012.

[31] Irati Zamalloa, Risto Kojcev, Alejandro Hernández, Iñigo Muguruza, Lander Usategui San Juan, Asier Bilbao, and Victor Mayoral. Dissecting robotics - historical overview and future perspectives. *CoRR*, abs/1704.08617, 2017.

[32] T.B. Asafa, T.M. Afonja, E.A. Olaniyan, and H.O. Alade. Development of a vacuum cleaner robot. *Alexandria Engineering Journal*, 57(4):2911–2920, 2018.

[33] Anand Nayyar and Vikram Puri. Raspberry pi-a small, powerful, cost effective and efficient form factor computer: A review. *International Journal of Advanced Research in Computer Science and Software Engineering (IJARCSSE)*, 5:720–737, 12 2015.

[34] Daniela Rus. A decade of transformation in robotics. In *Towards a New Enlightenment? A Trascendent Decade*, pages 188–202. BBVA, 2018.

[35] Mars exploration rovers overview. `https://mars.nasa.gov/mer/mission/overview/`. Ημερομηνία πρόσβασης: 11-09-2021.

[36] Curiosity rover mission overview. `https://mars.nasa.gov/msl/mission/overview/`. Ημερομηνία πρόσβασης: 11-09-2021.

[37] Mars 2020 mission overview. `https://mars.nasa.gov/mars2020/mission/overview/`. Ημερομηνία πρόσβασης: 11-09-2021.

[38] Mars 2020 perceverance rover. `https://mars.nasa.gov/mars2020/spacecraft/rover/`. Ημερομηνία πρόσβασης: 11-09-2021.

[39] Raspberry pi 4. `https://www.raspberrypi.org/products/raspberry-pi-4-model-b/`. Ημερομηνία πρόσβασης: 14-09-2021.

[40] What is python? executive summary. `https://www.python.org/doc/essays/blurb/`. Ημερομηνία πρόσβασης: 20-10-2021.

[41] Keith W Ross James F Kurose. *Computer networking : a top-down approach*. Pearson Education, 7 edition, 2017.

[42] Stefan van der Walt, S. Chris Colbert, and Gael Varoquaux. The numpy array: A structure for efficient numerical computation. *Computing in Science Engineering*, 13(2):22–30, 2011.

[43] electromagnetic spectrum. `https://www.britannica.com/science/electromagnetic-spectrum`. Ημερομηνία πρόσβασης: 28-10-2021.

[44] M Pauline Baker Donald Hearn. *Computer graphics with OpenGL*. Pearson Education Inc, 3 edition, 2004.

[45] About opencv. `https://opencv.org/about/`. Ημερομηνία πρόσβασης: 29-10-2021.

[46] Chollet Francois. *Deep Learning with Python*. Manning Publications Co., USA, 1st edition, 2017.

[47] Connor Shorten and Taghi Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6, 07 2019.

[48] Fuqiang Gu, Mu-Huan Chung, Mark Chignell, Shahrokh Valaee, Baoding Zhou, and Xue Liu. A survey on deep learning for human activity recognition. *ACM Comput. Surv.*, 54(8), October 2021.

[49] Frank Emmert-Streib, Zhen Yang, Han Feng, Shailesh Tripathi, and Matthias Dehmer. An introductory review of deep learning for prediction models with big data. *Frontiers in Artificial Intelligence*, 3:4, 2020.

[50] Sebastian Ruder. An overview of gradient descent optimization algorithms. *ArXiv*, abs/1609.04747, 2016.

[51] Yann LeCun, Y. Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521:436–44, 05 2015.

[52] Κ. Διαμαντάρας. *Τεχνητα Νευρωνικά Δίκτυα*. Κλειδάριθμος, Αθήνα, 1 edition, 2007.

[53] Simon O. Haykin. *Neural Networks and Learning Machines*. Pearson Education, Upper Saddle River, New Jersey, 3rd edition, 2009.

[54] Δ. Μπότσης Κ. Διαμαντάρας. *Μηχανική Μάθηση*. Κλειδάριθμος, Αθήνα, 1 edition, 2019.

[55] Abien Fred Agarap. Deep learning using rectified linear units (relu). 03 2018.

[56] PangNing Tan, Michael Steinbach, Anuj Karpatne, and Vipin Kumar. *Introduction to Data Mining*. Pearson Education, Upper Saddle River, New Jersey, 2nd edition, 2018.

[57] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 11 2013.

[58] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

[59] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. pages 779–788, 06 2016.

[60] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. Ssd: Single shot multibox detector. *Lecture Notes in Computer Science*, page 21–37, 2016.

[61] Zhong-Qiu Zhao, Peng Zheng, Shou-Tao Xu, and Xindong Wu. Object detection with deep learning: A review. *IEEE Transactions on Neural Networks and Learning Systems*, PP:1–21, 01 2019.

[62] Andrew Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. 04 2017.

[63] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv 1409.1556*, 09 2014.

[64] Zhen-Hua Feng, Josef Kittler, Muhammad Awais, Patrik Huber, and Xiao-Jun Wu. Wing loss for robust facial landmark localisation with convolutional neural networks. 06 2018.

[65] Fei Gao, Bing Li, Lei Chen, Zhongyu Shang, Xiang Wei, and Chen He. A softmax classifier for high-precision classification of ultrasonic similar signals. *Ultrasonics*, 112:106344, 2021.

[66] Navaneeth Bodla, Bharat Singh, Rama Chellappa, and Larry Davis. Improving object detection with one line of code. 04 2017.

[67] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, and Xiaoqiang Zhang. Tensorflow: A system for large-scale machine learning. 05 2016.

[68] Dogan Ibrahim. Advanced pic32 projects. In *Designing Embedded Systems with 32-Bit PIC Microcontrollers and MikroC*, volume 1, pages 359–442. Newnes, 2014.

[69] Tensorflow 2 object detection api tutorial. `https://tensorflow-object-detection-api-tutorial.readthedocs.io/en/latest/index.html`. Ημερομηνία πρόσβασης: 8-10-2021.

[70] Tensorflowobjectdetectiontutorial. `https://github.com/sglvladi/TensorFlowObjectDetectionTutorial`. Ημερομηνία πρόσβασης: 4-02-2022.

[71] Tensorflow 2 detection model zoo. `https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/tf2_detection_zoo.md`. Ημερομηνία πρόσβασης: 8-10-2021.

[72] Google colaboratory. `https://colab.research.google.com/?utm_source=scs-index`. Ημερομηνία πρόσβασης: 9-10-2021.

ΠΑΡΑΡΤΗΜΑΤΑ

Παράρτημα Α

Κώδικες

Παρακάτω δίδονται οι κώδικες συνολικά όπως εκτελέστηκαν στη λειτουργία του ρομποτικού βραχίονα. Πρώτα εκτελείται ο κώδικας RpiControl.py στο Raspberry Pi και μετά εκτελείται ο κώδικας ObjectDetectionCamera.py στον υπολογιστή.

A.1 MotorDriver.py

```
import RPi.GPIO as GPIO
from time import sleep
from pynput.keyboard import Key, Listener

GPIO.setmode(GPIO.BCM)  # Use GPIO numbering
GPIO.setwarnings(False) # Do not show warnings
GPIO.cleanup()          # Exit if nothing is called

class Motor():
    def __init__(self,EnaA,In1A,In2A,EnaB,In1B,In2B):
        GPIO.setmode(GPIO.BCM)
        GPIO.setwarnings(False)

        #--The inputs of Motor Driver---
        self.EnaA = EnaA
        self.In1A = In1A
        self.In2A = In2A
        self.EnaB = EnaB
        self.In1B = In1B
        self.In2B = In2B

        #----GPIO Outputs-----
        GPIO.setup(self.EnaA,GPIO.OUT)
        GPIO.setup(self.In1A,GPIO.OUT)
        GPIO.setup(self.In2A,GPIO.OUT)
        GPIO.setup(self.EnaB,GPIO.OUT)
        GPIO.setup(self.In1B,GPIO.OUT)
        GPIO.setup(self.In2B,GPIO.OUT)

        #---Initialize PWM on 100Hz frequency
        #---Start PWM with 0% duty cycle
        self.pwmA = GPIO.PWM(self.EnaA, 100);
        self.pwmA.start(0);
```

```
        self.pwmB = GPIO.PWM(self.EnaB, 100);
        self.pwmB.start(0);

    def move(self,speed=0.5,turn=0,t=0):
        speed *=100
        leftSpeed = speed
        rightSpeed = speed

        if turn == -1:      #-----Left
            leftSpeed = -speed
            rightSpeed = speed
        elif turn == 1:     #-----Right
            leftSpeed = speed
            rightSpeed = -speed
        elif turn == -0.5: #-----Backwards
            leftSpeed = -speed
            rightSpeed = -speed
        else:
            leftSpeed = speed #--Forward
            rightSpeed = speed

        #---Max Duty Cycle 100-----
        if leftSpeed>100: leftSpeed=100
        elif leftSpeed<-100: leftSpeed= -100
        if rightSpeed>100: rightSpeed=100
        elif rightSpeed<-100: rightSpeed= -100

        #--- Make the signal active----
        self.pwmA.ChangeDutyCycle(abs(leftSpeed))
        self.pwmB.ChangeDutyCycle(abs(rightSpeed))

        if leftSpeed>0: #---- Left wheels going forward
            GPIO.output(self.In1A,GPIO.HIGH)
            GPIO.output(self.In2A,GPIO.LOW)
        else:           #---- Left wheels going backwards
            GPIO.output(self.In1A,GPIO.LOW)
            GPIO.output(self.In2A,GPIO.HIGH)

        if rightSpeed>0: #---- Right wheels going forward
            GPIO.output(self.In1B,GPIO.LOW)
            GPIO.output(self.In2B,GPIO.HIGH)
        else:            #---- Right wheels going backwards
            GPIO.output(self.In1B,GPIO.HIGH)
            GPIO.output(self.In2B,GPIO.LOW)
        if t>0:
            sleep(t)

    #----- Stop the motors-----
    def stop(self,t=0):
        self.pwmA.ChangeDutyCycle(0);
        self.pwmB.ChangeDutyCycle(0);
        sleep(t)


if __name__ == '__main__':
    motor= Motor(4,3,2,22,17,27)
```

A.2 RpiControl.py

```
import socket
import threading
import RPi.GPIO as GPIO
```

```
import MotorDriver
from time import sleep
import time
import numpy as np

motor = MotorDriver.Motor(4,3,2,22,17,27)

servoBase1 = 23
servoBase2 = 24
servoArm = 25
servoGrip = 26

GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
GPIO.setup(servoGrip, GPIO.OUT)
GPIO.setup(servoBase1, GPIO.OUT)
GPIO.setup(servoBase2, GPIO.OUT)
GPIO.setup(servoArm, GPIO.OUT)


base1 = GPIO.PWM(servoBase1, 50) # GPIO 23 for PWM with 50Hz
base2 = GPIO.PWM(servoBase2, 50) # GPIO 24 for PWM with 50Hz
arm = GPIO.PWM(servoArm, 50) # GPIO 25 for PWM with 50Hz
gripper = GPIO.PWM(servoGrip, 50) # GPIO 26 for PWM with 50Hz

gripper.start(0)
arm.start(0)
base1.start(0)
base2.start(0)

# Steer the robot 30 degrees
def SteerThirty(direction):
    if direction=='left':
        motor.move(0.5, -1, 0.33)
        motor.stop(0)
    elif direction=='right':
        motor.move(0.5, 1, 0.33)
        motor.stop(0)
    else:
        print("No Maneuvres")
        motor.stop(0)

# Steer the robot 1 degree
def Steer(direction):
    if direction=='left':
        motor.move(0.4, -1, 0.05)
        motor.stop(0)
    elif direction=='right':
        motor.move(0.4, 1, 0.05)
        motor.stop(0)
    else:
        print("No Maneuvres")
        motor.stop(0)

# Move the robot straight ahead
def Forward(time):
    motor.move(0.3,0.5,time)
    motor.stop(0)

# Move the robot back
def Backward(time):
    motor.move(0.3,-0.5,time)
    motor.stop(0)

def DistanceUpdate(distance,deg):
    distance = distance + np.cos(deg)*distance
```

```
    return distance

# Update of heading
def HeadingUpdate(heading, action):
    if action == 'left':
        heading = heading - 1
    else:
        heading = heading + 1
    return heading

# Stop moving the robot.
def Stop():
    print("Stoping the motors")
    motor.stop(0)

# Termination of robot
def Terminate():
    gripper.stop()
    arm.stop()
    base1.stop()
    base2.stop()
    GPIO.cleanup()
    motor.stop(0)
    print("Terminated. Servos and motors Disabled")

#--------Function that translates angle to duty-------
def DesiredAngle(angle):
    duty = angle/18 +2.5

    return duty

# Function that grabs the object
def grab(action,distance,heading):
    holding = 0
    try:
        while True:
            if action == 'get':
                if holding == 1: # The target is grabbed. Maintain the angles.
                    print("Desposing the can")
                    for t in range(0,3):
                        SteerThirty('right')
                        sleep(0.5)
                    posGrip = DesiredAngle(120) # Drop the can
                    gripper.ChangeDutyCycle(posGrip)
                    time.sleep(0.5)
                    posArm = DesiredAngle(170) # Lower the Arm
                    arm.ChangeDutyCycle(posArm)
                    time.sleep(0.5)
                    posBase = DesiredAngle(5) # Neutral position
                    base1.ChangeDutyCycle(posBase)
                    base2.ChangeDutyCycle(posBase)
                    time.sleep(0.5)
                    print("Continuing")
                    for t in range(0,3):
                        SteerThirty('left')
                        sleep(0.5)
                    holding=0
                    break
                else:
                    print("Initiating Grabbing Procedure")
                    # Prepare the Gripper
                    posGrip = DesiredAngle(120)
                    gripper.ChangeDutyCycle(posGrip)
                    time.sleep(1)

                    # Raise the Arm
```

```
                    posArm = DesiredAngle(110)
                    arm.ChangeDutyCycle(posArm)
                    time.sleep(1)

                    # Moving the arm towards the target
                    for i in range(1,6):
                        ang = 32 + i*7
                        posBase = DesiredAngle(ang)
                        base1.ChangeDutyCycle(posBase)
                        base2.ChangeDutyCycle(posBase)
                        time.sleep(0.5)

                    # Grab the target
                    posGrip = DesiredAngle(30)
                    gripper.ChangeDutyCycle(posGrip)
                    time.sleep(1)

                    # Moving the arm back after grabbing it
                    posBase = DesiredAngle(20)
                    base1.ChangeDutyCycle(posBase)
                    base2.ChangeDutyCycle(posBase)
                    time.sleep(1)
                    print("Grabbing Accomplished")
                    holding = 1
            else:
                continue


    except KeyboardInterrupt:
        print("Interrupted from User. Servos Disabled")
        #----------Termination-------------
        gripper.stop()
        arm.stop()
        base1.stop()
        base2.stop()
        GPIO.cleanup()
        motor.stop(0)

class ClientThread(threading.Thread):
    def __init__(self, clientAddress,clientsocket):
        threading.Thread.__init__(self)
        self.csocket = clientsocket
        print ("New connection established: ", clientAddress)

    def run(self):
        print("Connection from : ", clientAddress)
        msg = ''
        act = 0 #  If > 0 heading is right else is left
        head = 0 # Heading counter
        dist = 0 # Distance counter

        while True:
            data = self.csocket.recv(2048)
            msg = data.decode()
            # Status: Searching
            if msg == ' ':
                # Nothing is found after 1 meter.
                # Changing Location.
                print("Changing Searching Area")
                if dist != 0 and dist % 20 == 0:
                    SteerThirty('left')
                    sleep(0.3)
                    Backward(0.3)
                    sleep(0.3)
                    SteerThirty('right')
                    sleep(0.3)
```

```
                    Backward(0.8)
                    sleep(0.5)
                    dist=0
                    continue
                Forward(0.2)
                sleep(1.5)
                dist = dist + 2

            if msg == 'bye':
                break
            # "right" order from PC
            if msg == 'Right':
                Steer('right')
                head = HeadingUpdate(head,'right')
                act = act + 1
                print("right at",act)
                sleep(0.3)
            # "left" order from PC
            elif msg == 'Left':
                Steer('left')
                head = HeadingUpdate(head,'left')
                act = act - 1
                print("left at",act)
                sleep(0.3)
            # "forward" order from PC
            elif msg == 'Go':
                print("go")
                Forward(0.1)
                dist = dist + 1
                sleep(0.3)
            # "back" order from PC
            elif msg == 'Back':
                print("back")
                Backward(0.1)
                dist = dist - 1
                sleep(0.3)
            # "stop and get ready for grab" order from PC
            elif msg == 'full':
                print("full")
                Forward(0.24)
                dist = dist + 2
                print("The final distance and heading", dist, head)
#                 dist = DistanceUpdate(dist,act)
                grab('get', dist, head)
            else:
                continue

            self.csocket.send(bytes(msg, 'UTF-8'))

        print("Pc at ", clientAddress, " terminated the connection.")


HOST = '192.168.2.43'
PORT = 8485

rPiSocket = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
print('Creation of Socket successful')

rPiSocket.bind((HOST, PORT))
print('Socket bind accomplished')

# Accept connections
while True:
    rPiSocket.listen(1)
    (clientConnected, clientAddress) = rPiSocket.accept()
```

```
    newthread = ClientThread(clientAddress, clientConnected)
    newthread.start()
```

A.3 ObjectDetectionCamera.py

Ο παρακάτω κώδικας για το κομμάτι της ανίχνευσης αντικειμένων βασίστηκε στο κώδικα του Lyudmil Vladimirov, (2020), `https://github.com/sglvladi/TensorFlowObjectDetectionTutorial` όπου τροποποιήθηκε για τις ανάγκες της συγκεκριμένης εργασίας.

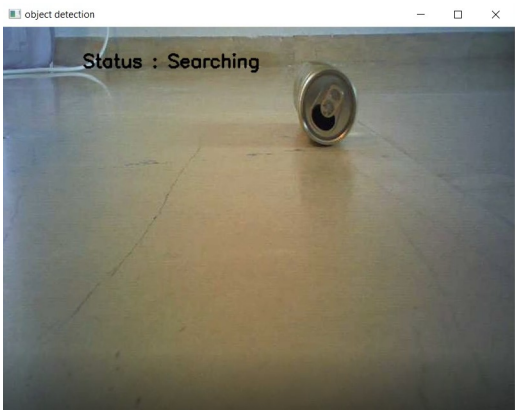
```
import os
import socket
import cv2
import numpy as np
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'    # Suppress TensorFlow logging
import tensorflow as tf
from object_detection.utils import label_map_util
from object_detection.utils import config_util
from object_detection.utils import visualization_utils as viz_utils
from object_detection.builders import model_builder

# Connecting Pc to Robotic Arm
HOST = '192.168.2.43'  # the Robotic Arms' IP
PORT = 8485   # the Robotic Arms' port
pc_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)  # Create Socket
pc_socket.connect((HOST, PORT))  # Connect to Robotic Arm

DATA_DIR = os.path.join(os.getcwd(), 'data')
MODELS_DIR = os.path.join(DATA_DIR, 'models')

MODEL_NAME = 'my_model'
PATH_TO_CKPT = os.path.join(MODELS_DIR, os.path.join(MODEL_NAME, 'checkpoint/'))
PATH_TO_CFG = os.path.join(MODELS_DIR, os.path.join(MODEL_NAME, 'pipeline.config'))

# Download labels file
LABEL_FILENAME = 'label_map.pbtxt'
PATH_TO_LABELS = os.path.join(MODELS_DIR, os.path.join(MODEL_NAME, LABEL_FILENAME))

# Enable GPU dynamic memory allocation
gpus = tf.config.experimental.list_physical_devices('GPU')
for gpu in gpus:
    tf.config.experimental.set_memory_growth(gpu, True)

# Load pipeline config and build a detection model
configs = config_util.get_configs_from_pipeline_file(PATH_TO_CFG)
model_config = configs['model']
detection_model = model_builder.build(model_config=model_config, is_training=False)

# Restore checkpoint
ckpt = tf.compat.v2.train.Checkpoint(model=detection_model)
ckpt.restore(os.path.join(PATH_TO_CKPT, 'ckpt-0')).expect_partial()


@tf.function
def detect_fn(image):
    """Detect objects in image."""

    image, shapes = detection_model.preprocess(image)
    prediction_dict = detection_model.predict(image, shapes)
```

```
    detections = detection_model.postprocess(prediction_dict, shapes)

    return detections, prediction_dict, tf.reshape(shapes, [-1])

# Load label map data (for plotting)
category_index = label_map_util.create_category_index_from_labelmap(PATH_TO_LABELS,
                                                                     use_display_name=True)

# Define the video stream
cap = cv2.VideoCapture(1)

# Define a box
box = (0, 0, 0, 0)

while True:
    # Start timer
    timer = cv2.getTickCount()
    # Read frame from camera
    ret, image_np = cap.read()

    # Expand dimensions since the model expects images to have shape: [1, None, None, 3]
    image_np_expanded = np.expand_dims(image_np, axis=0)

    #Convert the input image to tensors
    input_tensor = tf.convert_to_tensor(np.expand_dims(image_np, 0), dtype=tf.float32)
    detections, predictions_dict, shapes = detect_fn(input_tensor)

    label_id_offset = 1
    image_np_with_detections = image_np.copy()

    viz_utils.visualize_boxes_and_labels_on_image_array(
          image_np_with_detections,
          detections['detection_boxes'][0].numpy(),
          (detections['detection_classes'][0].numpy() + label_id_offset).astype(int),
          detections['detection_scores'][0].numpy(),
          category_index,
          use_normalized_coordinates=True,
          max_boxes_to_draw=2,
          min_score_thresh=.70,
          agnostic_mode=False)

    # Calculate Frames per second (FPS) and display on frame
    fps = cv2.getTickFrequency() / (cv2.getTickCount() - timer)
    # cv2.putText(image_np_with_detections, "FPS : " + str(int(fps)), (100, 50), cv2.FONT_HERSHEY_SIMPLEX, 0.75,
    #             (50, 170, 50), 2)

    boxes = np.squeeze(detections['detection_boxes'])
    scores = np.squeeze(detections['detection_scores'])

    # set a min thresh score
    min_score_thresh = 0.7
    # get the boxes with score > threshold
    bboxes = boxes[scores > min_score_thresh]

    # If we have found a can then load the coordinates of the box
    if bboxes.any():
        yMin, xMin, yMax, xMax = bboxes[0]
        box = (int(xMin * 640), int(yMin * 480), int(xMax * 640), int(yMax * 480))

        # Draw a circle in the center of the detection bounding box
        dc1 = int((box[0] + box[2])/2)
        dc2 = int((box[1] + box[3])/2)
        cv2.circle(image_np_with_detections, (dc1, dc2), 4, (255, 0, 0), -1)

        # Print the centroid coordinates on the image
        text = "x: " + str(dc1) + ", y: " + str(dc2)
```

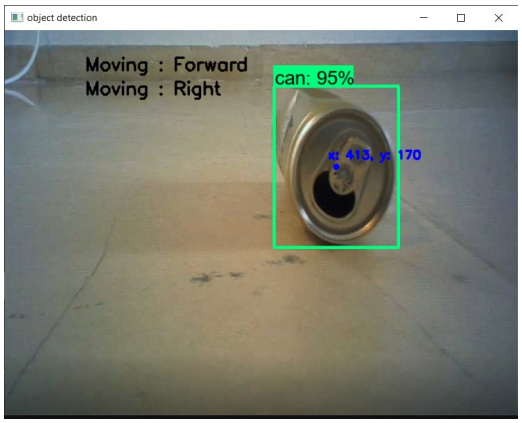
```
        cv2.putText(image_np_with_detections, text, (dc1 - 10, dc2 - 10),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 0, 0), 2)

        # Directions for centering the Object on x-axis of frame
        if dc1 > 310:  # if the object is right
            data = "Right"
            cv2.putText(image_np_with_detections, "Moving : " + str('Right'), (100, 80), cv2.FONT_HERSHEY_SIMPLEX, 0.75,
                        (0, 0, 0), 2)
        elif dc1 < 280:  # if the object is left
            data = "Left"
            cv2.putText(image_np_with_detections, "Moving : " + str('Left'), (100, 80), cv2.FONT_HERSHEY_SIMPLEX,
                        0.75,
                        (0, 0, 0), 2)
        else:  # the object is at the desired position
            data = 'half'
        # Send data to server
        pc_socket.send(data.encode())

        # Directions for centering the Object on y-axis of frame
        if dc2 > 350:   # if the object is near
            data = "Back"
            cv2.putText(image_np_with_detections, "Moving : " + str('Back'), (100, 140), cv2.FONT_HERSHEY_SIMPLEX,
                        0.75,
                        (0, 0, 0), 2)
        elif dc2 < 342:  # if the object is far
            data = "Go"
            cv2.putText(image_np_with_detections, "Moving : " + str('Forward'), (100, 50), cv2.FONT_HERSHEY_SIMPLEX,
                        0.75,
                        (0, 0, 0), 2)
        else:  # the object is at the desired position
            if data == 'half':
                data = 'full'
                cv2.putText(image_np_with_detections, "Status : " + str('Stoped'), (100, 50), cv2.FONT_HERSHEY_SIMPLEX,
                            0.75,
                            (0, 0, 0), 2)


        # Send data to server
        pc_socket.send(data.encode())

    else:
        cv2.putText(image_np_with_detections, "Status : " + str('Searching'), (100, 50), cv2.FONT_HERSHEY_SIMPLEX, 0.75,
                    (0, 0, 0), 2)
        data = " "
        pc_socket.send(data.encode())

    # Display output
    cv2.imshow('object detection', image_np_with_detections)

    if cv2.waitKey(25) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```

Παράρτημα Β

Φωτογραφίες του Ρομποτικού Βραχίονα



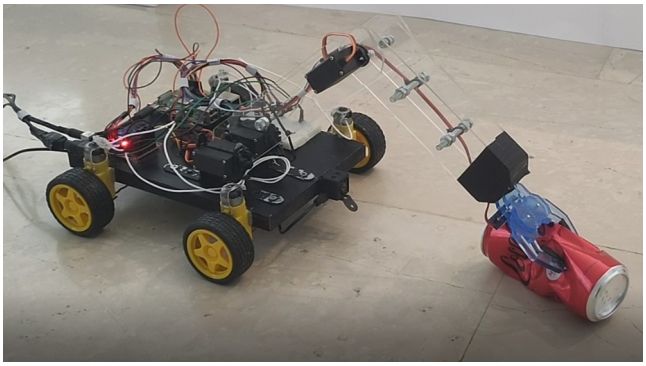
Σχήμα Β.1: Εκτέλεση αναζήτησης. Εαν και υπάρχει στην εικόνα ένα κουτί αλουμινίου το ποσοστό του σκορ είναι χαμηλότερο από το κατώφλι, οπότε δεν το έχει ανιχνέυσει ακόμα και δεν έχει εμφανιστεί πλαίσιο οριοθέτησης.



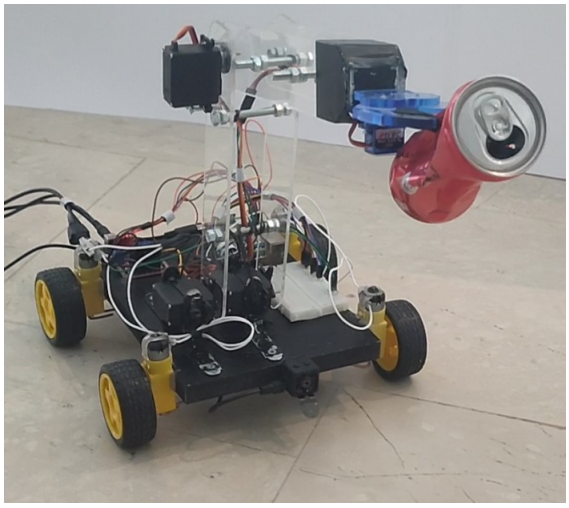
Σχήμα Β.2: Έχει ανιχνευθεί κουτί αλουμινίου και δίνονται οδηγίες από τον υπολογιστή για να λάβει τη σωστή θέση ο βραχίονας.

Σχήμα Β.3: Έχει ληφθεί η τελική θέση και κινείται ο βραχίονας για τη συλλογή.

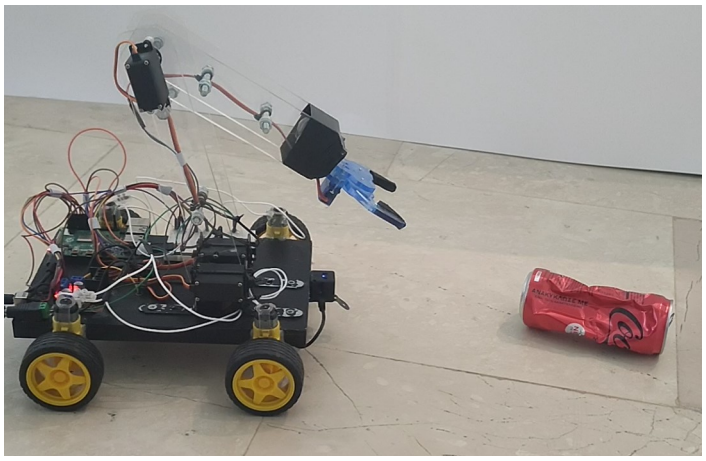
Σχήμα Β.4: Δοκιμή συλλογής 2 κουτιών αλουμινίου.



Σχήμα Β.5: Η τελική θέση της πλατφόρμας και του βραχίονα για τη συλλογή του 1ου κουτιού.



Σχήμα Β.6: Κρατώντας το κουτί αλουμινίου, στρέφει δεξιά με σκοπό να το αφήσει.



Σχήμα Β.7: Μετά τη 1η συλλογή συνεχίζει για το 2ο κουτί αλουμινίου που έχει ανιχνεύσει.