ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ

Αυτόνομο Ρομποτικό Όχημα με χρήση Μηχανικής Μάθησης και Μηχανικής Όρασης

ΤΖΑΝΑΚΗΣ ΙΩΑΝΝΗΣ

Α.Μ.: 44622

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ

Δρ. ΝΙΚΟΛΑΟΥ ΓΡΗΓΟΡΙΟΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΑΥΤΟΜΑΤΙΣΜΟΥ

ΑΘΗΝΑ 2020

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΠΤΥΧΙΑΚΗΣ ΕΡΓΑΣΙΑΣ

Ο κάτωθι υπογεγραμμένος ΤΖΑΝΑΚΗΣ ΙΩΑΝΝΗΣ, του ΓΕΩΡΓΙΟΥ, φοιτητής του Τμήματος ΜΗΧΑΝΙΚΩΝ ΒΙΟΜΗΧΑΝΙΚΗΣ ΣΧΕΔΙΑΣΗΣ ΚΑΙ ΠΑΡΑΓΩΓΗΣ του Πανεπιστημίου Δυτικής Αττικής, πριν αναλάβω την εκπόνηση της Πτυχιακής Εργασίας μου, δηλώνω ότι ενημερώθηκα για τα παρακάτω:

«Η Πτυχιακή Εργασία (Π.Ε) αποτελεί προϊόν πνευματικής ιδιοκτησίας τόσο του συγγραφέα, όσο και του Ιδρύματος και θα πρέπει να έχει μοναδικό χαρακτήρα και πρωτότυπο περιεχόμενο.

Απαγορεύεται αυστηρά οποιοδήποτε κομμάτι κειμένου της να εμφανίζεται αυτούσιο ή μεταφρασμένο από κάποια άλλη δημοσιευμένη πηγή. Κάθε τέτοια πράξη αποτελεί προϊόν λογοκλοπής και εγείρει θέμα Ηθικής Τάξης για τα πνευματικά δικαιώματα του άλλου συγγραφέα. Αποκλειστικός υπεύθυνος είναι ο συγγραφέας της Π.Ε, ο οποίος φέρει και την ευθύνη των συνεπειών, ποινικών και άλλων, αυτής της πράξης.

Πέραν των όποιων ποινικών ευθυνών του συγγραφέα, σε περίπτωση που το Ίδρυμα του έχει απονείμει Πτυχίο, αυτό ανακαλείται με απόφαση της Συνέλευσης του Τμήματος. Η Συνέλευση του Τμήματος με νέα απόφασή της, μετά από αίτηση του ενδιαφερόμενου, του αναθέτει εκ νέου την εκπόνηση Π.Ε με άλλο θέμα και διαφορετικό επιβλέποντα καθηγητή. Η εκπόνηση της εν λόγω Π.Ε πρέπει να ολοκληρωθεί εντός τουλάχιστον ενός ημερολογιακού 6μήνου από την ημερομηνία ανάθεσής της.

Ο Δηλών

ΤΖΑΝΑΚΗΣ ΙΩΑΝΝΗΣ

Ημερομηνία

5/02/2020

Περίληψη

Τα τελευταία χρόνια οι εξελίξεις στην Τεχνητή Νοημοσύνη και την Ρομποτική είναι ραγδαίες και τεράστιες. Στην εποχή μας εντοπίζονται αυτόνομα οχήματα που κυκλοφορούν στους δρόμους και αυτόνομα ρομποτικά οχήματα που χρησιμοποιούνται όχι μόνο για την εξερεύνηση του βυθού, αλλά και του διαστήματος. Παράλληλα, είναι διαθέσιμα στην αγορά ρομπότ που καθαρίζουν το σπίτι και κινητά τηλέφωνα που χρησιμοποιούν την Τεχνητή Νοημοσύνη για την αναγνώριση προσώπων και την λήψη καλύτερων φωτογραφιών.

Στην παρούσα πτυχιακή εργασία σχεδιάζεται και υλοποιείται ένα αυτόνομο ρομποτικό όχημα, που πλοηγείται στο περιβάλλον του για τον εντοπισμό και την προσέγγιση ενός συγκεκριμένου αντικειμένου χρησιμοποιώντας Μηχανική Μάθηση και Μηχανική Όραση. Το αυτόνομο ρομποτικό όχημα μέσω του αισθητήρα αποστάσεως Time-of-Flight που διαθέτει, μαθαίνει να κινείται στο περιβάλλον του χρησιμοποιώντας τον αλγόριθμο Q-Learning της Ενισχυτικής Μάθησης. Για τον εντοπισμό του αντικειμένου διαθέτει κάμερα και χρησιμοποιεί τον αλγόριθμο ORB, την μέθοδο FLANN και την Homography. Η επίτευξη της εργασίας του αυτόνομου ρομποτικού οχήματος πραγματοποιήθηκε έπειτα από πολλαπλά πειράματα τα οποία και παρουσιάζονται αναλυτικά.

Λέξεις Κλειδιά: Αυτόνομο Ρομποτικό Όχημα, Μηχανική Μάθηση, Μηχανική Όραση, Ενισχυτική Μάθηση, Time-of-Flight, Κάμερα, Q-Learning, ORB, FLANN, Homography.

Abstract

In recent years, developments in Artificial Intelligence and Robotics have been rapid and enormous. Nowadays, autonomous vehicles are found on the roads and autonomous robotic vehicles used not only to explore the seabed, but also to explore space. At the same time, robots for home cleaning and cell phones that use Artificial Intelligence to recognize faces and take better photos are available in the market.

In this thesis presents the design and implementation of an autonomous robotic vehicle, which navigates its environment to detect and approach a specific object using Machine Learning and Machine Vision. The autonomous robotic vehicle through its Time-of-Flight distance sensor learns to move around its environment using Q-Learning algorithm of Reinforcement Learning. To detect the object it has camera and uses ORB algorithm, FLANN method and Homography. The work of autonomous robotic vehicle was accomplished after multiple experiments which are presented in detail.

Keywords: Autonomous Robotic Vehicle, Machine Learning, Machine Vision, Reinforcement Learning, Time-of-Flight, Camera, Q-Learning, ORB, FLANN, Homography.

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή μου Δρ. Γρηγόριο Νικολάου για την δυνατότητα που μου έδωσε και την εμπιστοσύνη που μου έδειξε, να εκπονήσω την παρούσα αυτή εργασία.

Τέλος, θα ήθελα να ευχαριστήσω τους γονείς μου και την αδερφή μου για την υπομονή τους και την στήριξη που μου παρείχαν καθ' όλη την διάρκεια των σπουδών μου.

Περιεχόμενα

Κατάλογος Σχημάτων

Κατάλογος Εικόνων

Κατάλογος Πινάκων

1. Εισαγωγή

1.1 Ρομποτική

Η *Ρομποτική (Robotics)* ασχολείται με την μελέτη μηχανών που μπορούν να αντικαταστήσουν τον άνθρωπο σε κάποιο έργο. Η αντικατάσταση αυτή έχει να κάνει με το επίπεδο της φυσικής δραστηριότητας και το επίπεδο λήψης αποφάσεων. Στην πραγματικότητα η ρομποτική ορίζεται συνήθως ως ο επιστημονικός κλάδος του μηχανικού που μελετά την σχέση μεταξύ αντίληψης και δράσης.

Σύμφωνα με αυτόν τον ορισμό ένα ρομποτικό σύστημα είναι ένα πολύπλοκο σύστημα που λειτουργικά αποτελείται από πολλαπλά υποσυστήματα.

Είναι ένας κλάδος που συνδυάζει πολλούς τομείς, όπως ο αυτόματος έλεγχος, η μηχανική, η ηλεκτρονική, η πληροφορική, η τεχνητή νοημοσύνη και η μηχανική όραση.

Για να ανατρέξουμε στις ρίζες της ρομποτικής, θα πρέπει να γυρίσουμε χιλιάδες χρόνια πίσω στην ανθρώπινη ιστορία. Μια μεγάλη φιλοδοξία του ανθρώπου είναι να κατασκευάσει μηχανές που θα μπορούσαν να τον μιμηθούν στην μορφή, αλλά και στην συμπεριφορά του αλληλεπιδρώντας στο περιβάλλον. Στην αρχαία Ελληνική μυθολογία συναντάμε τον Τιτάνα Προμηθέα που έπλασε τους ανθρώπους από άργιλο. Επίσης, τον Τάλω που ήταν το πρώτο ρομπότ από χαλκό που περπάτησε στην Γη και τον δημιούργησε ο θεός Ήφαιστος για να φυλάσσει την Κρήτη από τους εισβολείς.

Ο όρος *ρομπότ (robot)* καθιερώθηκε για πρώτη φορά το 1920 από τον Τσέχο θεατρικό συγγραφέα Karel Čapek μέσα από το θεατρικό του έργο με τίτλο *Rossum's Universal Robots (R.U.R.)*. Η ρίζα της λέξης προέρχεται από τον όρο *robota* όπου στις Σλαβικές γλώσσες σημαίνει εργασία.

Τη δεκαετία του 1940 ο Ρώσος συγγραφέας επιστημονικής φαντασίας Isaac Asimov σκέφτηκε το ρομπότ ως ένα αυτόματο με ανθρώπινη μορφή, αλλά χωρίς συναισθήματα. Η συμπεριφορά του επιβαλλόταν από ένα «ποζιτρονικό» εγκέφαλο προγραμματισμένο από έναν άνθρωπο για ικανοποιεί κάποιους κανόνες ηθικής συμπεριφοράς. Στη συνέχεια ο όρος ρομπότ εισήχθη από τον Asimov ως την επιστήμη που θα είναι αφοσιωμένη στη μελέτη των ρομπότ στηριζόμενη από τρεις βασικούς νόμους:

1. Ένα ρομπότ να μην μπορεί να τραυματίσει ή μέσω της αδράνειας του να μην βλάψει ένα ανθρώπινο όν.
2. Ένα ρομπότ θα πρέπει να υπακούει τις εντολές που δίνονται από τον άνθρωπο, εκτός και εάν οι εντολές είναι αντίθετες με τον πρώτο νόμο.
3. Ένα ρομπότ θα πρέπει να προστατεύει την ύπαρξή του, εκτός και αν αυτή η προστασία έρχεται σε αντίθεση με τον πρώτο ή δεύτερο νόμο.

Αργότερα, ο Asimov πρόσθεσε ακόμη ένα νόμο τον «μηδενικό» για προστατέψει το συμφέρον της ανθρωπότητας. Ο μηδενικός νόμος λέει ότι ένα ρομπότ να μην μπορεί να κάνει ζημιά στην ανθρωπότητα ή με την αδράνειά του να μην βλαφτεί η ανθρωπότητα.

Τα ρομπότ σήμερα διαθέτουν *συσκευές δράσης (effectors)*, όπως πόδια, τροχούς, αρθρώσεις και λαβές. Επίσης, διαθέτουν *αισθητήρες (sensors)* που χρησιμοποιούνται για να αντιλαμβάνονται το περιβάλλον τους. Υπάρχουν διάφορα είδη αισθητήρων που χρησιμοποιεί η ρομποτική σήμερα, όπως κάμερες, λέιζερ για μετρήσεις στο περιβάλλον, γυροσκόπια και επιταχυνσιόμετρα για την μέτρηση της κίνησης του ίδιου του ρομπότ.

Στην σύγχρονη εποχή τα πιο πολλά ρομπότ ανήκουν σε μία από τις τρεις κύριες κατηγορίες, τα *χειριστήρια (manipulators)*, τα *κινητά ρομπότ (mobile robot)* και τα *ανθρωποειδή ρομπότ (humanoid robot)*.

Στην πρώτη κατηγορία τα χειριστήρια ή ρομποτικοί βραχίονες είναι εγκαταστημένα στο χώρο εργασίας τους, για παράδειγμα, ένας ρομποτικός βραχίονας σε μια αυτοκινητοβιομηχανία ή στον Διεθνή Διαστημικό Σταθμό.

Στην δεύτερη κατηγορία τα κινητά ρομπότ, κινούνται στο περιβάλλον τους χρησιμοποιώντας τροχούς, πόδια ή παρόμοιους μηχανισμούς. Για παράδειγμα, χρησιμοποιούνται στην βιομηχανία για την αποθήκευση και την μετακίνηση κιβωτίων σε αποθήκες. Κάποια είδη κινητών ρομπότ είναι τα *μη επανδρωμένα οχήματα εδάφους (unmanned ground vehicles, UGVs)* που μπορούν να κινηθούν αυτόνομα χωρίς οδηγό σε αυτοκινητοδρόμους και εκτός δρόμους. Άλλα είδη κινητών ρομπότ είναι τα *μη επανδρωμένα οχήματα αέρος (unmanned air vehicles, UAVs)* τα οποία χρησιμοποιούνται συνήθως για επιτήρηση, αεροψεκασμούς γεωργικών καλλιεργειών και στρατιωτικές επιχειρήσεις. Τα *αυτόνομα υποβρύχια οχήματα (autonomous underwater vehicles, AUVs)* τα οποία χρησιμοποιούνται για εξερευνήσεις μεγάλου βάθους στις θάλασσες.

Στην τρίτη κατηγορία ανήκουν τα ρομπότ υβρίδια που είναι κινητά ρομπότ με χειριστήρια. Τέτοια ρομπότ είναι τα ανθρωποειδή που προσπαθούν να μιμούνται τον ανθρώπινο κορμό. Τα υβρίδια έχουν την δυνατότητα να χρησιμοποιούν τις συσκευές δράσεις τους και να κινούνται σε μεγαλύτερες αποστάσεις σε σχέση με τα χειριστήρια που είναι τοποθετημένα στο χώρο εργασίας τους. Όμως το έργο τους γίνεται δύσκολο επειδή δεν έχουν την σταθερότητα που προσφέρει η στήριξη.

Το πεδίο της ρομποτικής περιλαμβάνει ακόμη προσθετικές συσκευές, όπως τεχνητά άκρα, αυτιά και μάτια για τους ανθρώπους. Επιπλέον, ευφυή περιβάλλοντα, όπως ένα σπίτι που διαθέτει αισθητήρες και συσκευές δράσης, αλλά και πολυσωματικά συστήματα, όπως ένα ρομποτικό σμήνος αποτελούμενο από μικρά ρομπότ που συνεργάζονται για την επίτευξη ενός σκοπού.

1.2 Τεχνητή Νοημοσύνη

Ο άνθρωπος εδώ και πολλά χρόνια έχει ονειρευτεί μηχανές που θα μπορούν να σκέφτονται. Το πεδίο της *Τεχνητής Νοημοσύνης (T.N., Artificial Intelligence, AI)* ασχολείται με την κατανόηση της ευφυής συμπεριφοράς και την κατασκευή ευφυών μηχανών. Μηχανές οι οποίες θα είναι σε θέση για παράδειγμα να χειρίζονται την γνώση, ώστε να μαθαίνουν και να επιλύουν προβλήματα. Στο πεδίο αυτό έχουν συνεισφέρει η φιλοσοφία, τα μαθηματικά, τα οικονομικά, οι νευροεπιστήμες, η ψυχολογία, η γλωσσολογία, η μηχανική υπολογιστών, η θεωρία αυτομάτου ελέγχου και κυβερνητική.

Ο Αριστοτέλης (384-322 π.Χ.) θα μπορούσαμε να αναφέρουμε ότι ήταν ο πρώτος που ασχολήθηκε με την έννοια της Τεχνητής Νοημοσύνης, αφού μέσα από την ανάλυση και την κωδικοποίηση της συμβατικής λογικής εξέφρασε τους «συλλογισμούς» του (*Αριστοτέλειος συλλογισμός*).

Σοβαρή δουλειά πάνω στην Τεχνητή Νοημοσύνη αρχίζει να γίνεται μετά τον Β΄ Παγκόσμιο Πόλεμο. Το 1950 ο Alan Turing σχεδίασε την δοκιμασία Turing (*Turing Test*) για να εξεταστεί η ευφυΐα ενός υπολογιστή. Για να καταφέρει ένας υπολογιστής να περάσει την δοκιμασία θα πρέπει να διαθέτει κάποιες δυνατότητες, όπως επεξεργασία φυσικής γλώσσας, αναπαράσταση γνώσης, αυτοματοποιημένη συλλογιστική, μηχανική μάθηση, μηχανική όραση και ρομποτική. Έξι χρόνια μετά στο Dartmouth College συμφωνήθηκε η ονομασία Τεχνητή Νοημοσύνη για το πεδίο όπως το γνωρίζουμε σήμερα.

Τις τελευταίες δεκαετίες η Τεχνητή Νοημοσύνη έχει αναπτυχθεί αρκετά ώστε να παρέχει μια σειρά ισχυρών υπολογιστικών εργαλείων. Αυτό οφείλεται στους ισχυρούς υπολογιστές, τις μεγάλες βάσεις δεδομένων και την ανάπτυξη του Παγκόσμιου Ιστού. Η Τεχνητή Νοημοσύνη μπορεί να εντοπιστεί σε πολλές εφαρμογές. Κάποιες από αυτές μπορούμε να τις εντοπίσουμε στην ρομποτική, τον αυτόματο έλεγχο, στον αυτόνομο σχεδιασμό και χρονοπρογραμματισμό, στα παιχνίδια για υπολογιστές και στην σύνθεση καλλιτεχνημάτων τόσο στη μουσική, όσο και στη ζωγραφική. Επίσης, σε στρατιωτικές εφαρμογές, στην κατανόηση της γλώσσας και στην ιατρική με προγράμματα διάγνωσης ασθενειών.

Έχουν περάσει 70 χρόνια από την δοκιμασία του Tuning και στο διάστημα αυτό έχει γίνει μεγάλη πρόοδος στο πεδίο της Τεχνητής Νοημοσύνης, ειδικά τα τελευταία χρόνια που η ανθρωπότητα οδεύει προς την *4η Βιομηχανική Επανάσταση (Industry 4.0)*.

2. Μηχανική Μάθηση (Machine Learning)

Ο όρος *Μάθηση (Learning)* αναφέρεται στον τρόπο με τον οποίο κάθε ζωντανός οργανισμός προσαρμόζεται στο περιβάλλον και στις απαιτήσεις του, μεταβάλλοντάς το με την σειρά του για να κερδίσει από αυτό όσο το δυνατόν περισσότερα πράγματα προς όφελός του. Ο μηχανισμός της μάθησης θεωρείται όμοιος στο ζώο και στον άνθρωπο και μπορεί να εντοπιστεί για την απόκτηση απλών συνηθειών μέχρι την μάθηση εξαιρετικά πολύπλοκων ικανοτήτων. [68]

2.1 Μηχανική Μάθηση

Όπως έχει ήδη αναφερθεί, η Τεχνητή Νοημοσύνη επιχειρεί την δημιουργία ευφυών μηχανών που είναι ικανές να χειρίζονται την γνώση για να μάθουν. Η γνώση αυτή για τις μηχανές είναι τα δεδομένα που μπορεί να προέρχονται για παράδειγμα, από στατιστικά δεδομένα, μετρήσεις και την ανθρώπινη εμπειρία. Επίσης, δεδομένα είναι και οι εικόνες, τα βίντεο ή ο ήχος που μια μηχανή μπορεί να δεχτεί και να τα επεξεργαστεί κατάλληλα για την επίτευξη ενός αποτελέσματος.

Η *Μηχανική Μάθηση (Machine Learning)* αποτελεί μέρος της Τεχνητής Νοημοσύνης και ασχολείται με τον προγραμματισμό ενός υπολογιστικού συστήματος για την βελτιστοποίηση των εργασιών που εκτελεί. Αυτό επιτυγχάνεται μέσω της εκμάθησης, δηλαδή την εκτέλεση ενός προγράμματος χρησιμοποιώντας μια σειρά από παραδείγματα δεδομένων ή την προηγούμενη εμπειρία. Η εργασία που εκτελεί το υπολογιστικό σύστημα, μπορεί να είναι προβλέψιμη για να κάνει προβλέψεις στο μέλλον, περιγραφική για την απόκτηση γνώσεων από τα δεδομένα ή και τα δύο.

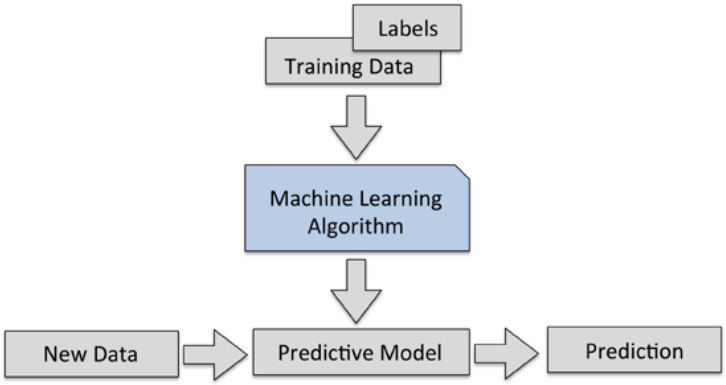
2.2 Κατηγορίες Μηχανικής Μάθησης

Ο τομέας της Μηχανικής Μάθησης χρησιμοποιεί αλγορίθμους για την εκτέλεση εργασιών στα υπολογιστικά συστήματα. Οι αλγόριθμοι αυτοί μπορούν να κατηγοριοποιηθούν ανάλογα με την εργασία τους σε τρείς κατηγορίες μάθησης, την *Επιβλεπόμενη Μάθηση*, την *Μη Επιβλεπόμενη Μάθηση* και την *Ενισχυτική Μάθηση*.

2.2.1 Επιβλεπόμενη Μάθηση (Supervised Learning)

Στην Επιβλεπόμενη Μάθηση το σύστημα μαθαίνει από παραδείγματα δεδομένων, ώστε να προβλέπει την έξοδο μελλοντικών δεδομένων.

Πιο συγκεκριμένα, το σύστημα χρησιμοποιεί αλγόριθμο για την κατασκευή συνάρτησης από δεδομένα που μαθαίνει να απεικονίζει τις εισόδους και τις γνωστές επιθυμητές εξόδους τους, από ένα σύνολο παραδειγμάτων ή αλλιώς *δεδομένα εκπαίδευσης (training data)* που δέχεται το σύστημα στην είσοδό του. Η συνάρτηση αυτή χρησιμοποιείται από το σύστημα για να προβλέψει την έξοδο νέων δεδομένων.



Σχήμα 2.2.1: Διάγραμμα αναπαράστασης Επιβλεπόμενης Μάθησης.

Στην Επιβλεπόμενη Μάθηση η έξοδος σε ένα σύστημα μπορεί να λάβει τιμές διακριτές ή συνεχείς. Εάν η έξοδος έχει διακριτές τιμές, τότε η εργασία ονομάζεται *ταξινόμηση (classification)*, ενώ αν έχει συνεχείς τιμές ονομάζεται *παλινδρόμηση (regression)*.

2.2.2 Μη Επιβλεπόμενη Μάθηση (Unsupervised Learning)

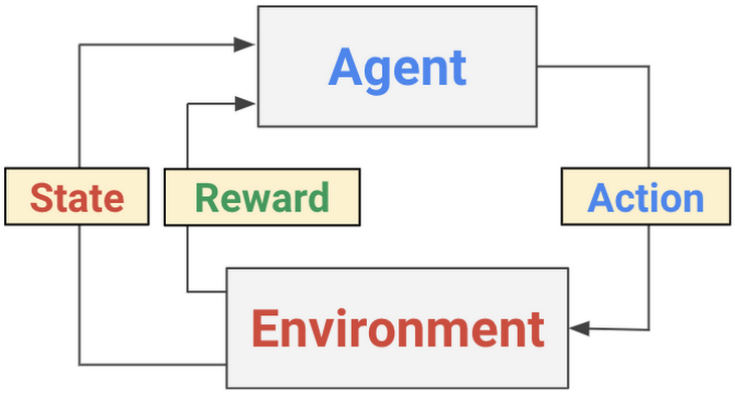
Στην Μη Επιβλεπόμενη Μάθηση το σύστημα πρέπει να μάθει να εξάγει πληροφορίες από άγνωστα δεδομένα χωρίς να έχει έτοιμα παραδείγματα δεδομένων.

Πιο συγκεκριμένα, το σύστημα δέχεται στην είσοδό του δεδομένα που δεν γνωρίζει ποια είναι η έξοδός τους και χρησιμοποιεί αλγόριθμο για να οργανώσει τα δεδομένα αυτά σε κατηγορίες. Εάν η κατηγοριοποίηση γίνεται με βάση τις ομοιότητες που έχουν τα δεδομένα, τότε η εργασία ονομάζεται *ομαδοποίηση (clustering)*. Στην περίπτωση που τα δεδομένα είναι αρκετά μεγάλα, δηλαδή περιέχουν μεγάλο αριθμό χαρακτηριστικών ή μετρήσεων, μπορεί να περιέχουν πληροφορίες που είναι περιττές ή ανήκουν σε άλλα χαρακτηριστικά. Τότε τα δεδομένα αυτά συμπιέζονται διατηρώντας τις χρήσιμες πληροφορίες. Η εργασία αυτή ονομάζεται *μείωση διαστάσεων (dimensionality reduction)*.

2.2.3 Ενισχυτική Μάθηση (Reinforcement Learning)

Στην Ενισχυτική Μάθηση το σύστημα έχει συνεχή αλληλεπίδραση με το περιβάλλον λαμβάνοντας πληροφορίες από αυτό για την βελτιστοποίηση της εργασίας του.

Πιο συγκεκριμένα, το σύστημα ή αλλιώς *πράκτορας (Agent)* επιλέγει τις *ενέργειες (Actions)* που θα εκτελέσει και από το *περιβάλλον (Environment)* λαμβάνει τις ανάλογες *ανταμοιβές (Rewards)* στην *κατάσταση (State)* που βρίσκεται. Έτσι ο πράκτορας μαθαίνει από τις εμπειρίες που έχει αποκτήσει κάτι που του επιτρέπει να επιλέγει αργότερα βέλτιστες ενέργειες.



Σχήμα 2.2.2: Διάγραμμα αναπαράστασης Ενισχυτικής Μάθησης.

3. Ενισχυτική Μάθηση (Reinforcement Learning)

Στην *Ενισχυτική Μάθηση* ο πράκτορας (π.χ. ένα ρομπότ) μαθαίνει τι πρέπει να κάνει, πώς να απεικονίσει τις καταστάσεις σε ενέργειες, έτσι ώστε να μεγιστοποιήσει την αριθμητική τιμή του σήματος της ανταμοιβής. Ο πράκτορας δεν γνωρίζει εξαρχής ποιες ενέργειες πρέπει να εκτελέσει, αλλά πρέπει να ανακαλύψει από μόνος του δοκιμάζοντας τις ενέργειες που θα αποφέρουν την μεγαλύτερη ανταμοιβή. Οι ενέργειες μπορούν να επηρεάσουν όχι μόνο την άμεση ανταμοιβή, αλλά και την επόμενη κατάσταση με αποτέλεσμα και την επιρροή των μεταγενέστερων ανταμοιβών. Κάθε πράκτορας θα πρέπει να αισθάνεται (π.χ. μέσω αισθητήρων) την κατάσταση του περιβάλλοντος, ώστε να μπορεί να εκτελεί ενέργειες που επηρεάζουν την κατάσταση. Ο πράκτορας για να μάθει αλληλεπιδρά με το περιβάλλον του, δοκιμάζει και κάνει λάθη (*trial and error learning*) για να επιτύχει τον στόχο ή τους στόχους του.

3.1 Στοιχεία της Ενισχυτικής Μάθησης

Τα στοιχεία που εντοπίζονται σε ένα σύστημα Ενισχυτικής Μάθησης είναι ο πράκτορας, το περιβάλλον, μια πολιτική, ένα σήμα ανταμοιβής, μια συνάρτηση αξίας και ένα μοντέλο του περιβάλλοντος.

Ο πράκτορας είναι ο εκπαιδευόμενος που μαθαίνει και λαμβάνει αποφάσεις. Μπορεί και εκτελεί ενέργειες από ένα σύνολο ενεργειών, ανάλογα με την κατάσταση του περιβάλλοντος στην οποία βρίσκεται.

Το περιβάλλον περιλαμβάνει οτιδήποτε υπάρχει σε αυτό εξωτερικά του πράκτορα.

Η πολιτική π εκφράζει την συμπεριφορά του πράκτορα σε μια δεδομένη στιγμή και είναι η απεικόνιση των καταστάσεων του περιβάλλοντος σε ενέργειες. Η πολιτική μπορεί να είναι μια συνάρτηση ή πίνακας αναζήτησης.

Το *σήμα ανταμοιβής (Reward signal)* είναι μια αριθμητική τιμή που ονομάζεται ανταμοιβή και την λαμβάνει ο πράκτορας κάθε χρονική στιγμή από το περιβάλλον. Καθορίζει εάν μια κατάσταση στην οποία βρεθεί ο πράκτορας είναι καλή ή κακή. Στόχος του πράκτορα είναι η μεγιστοποίηση της συνολικής ανταμοιβής που λαμβάνει με την πάροδο του χρόνου.

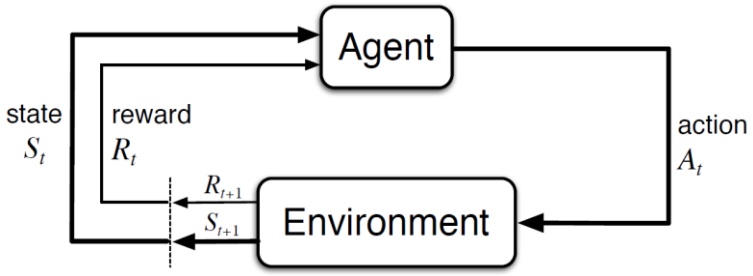
Η *συνάρτηση αξίας (Value function)* καθορίζει τι είναι καλό μακροπρόθεσμα. Πιο συγκεκριμένα, η αξία μιας κατάστασης είναι το συνολικό ποσό μιας ανταμοιβής που έχει

συγκεντρώσει ο πράκτορας με την πάροδο του χρόνου ξεκινώντας από αυτήν την κατάσταση. Στις ανταμοιβές καθορίζεται η άμεση επιθυμία των καταστάσεων του περιβάλλοντος, ενώ στις αξίες καθορίζεται η μακροπρόθεσμη επιθυμία των καταστάσεων έχοντας υπ'όψιν τις καταστάσεις που μπορεί να ακολουθήσουν και τις ανάλογες ανταμοιβές.

Το *μοντέλο του περιβάλλοντος (Model of the environment)* μιμείται την συμπεριφορά του περιβάλλοντος, δηλαδή του επιτρέπει να συμπεράνει για το πώς θα συμπεριφερθεί το περιβάλλον. Η χρήση ενός μοντέλου θα μπορούσε να ήταν η πρόβλεψη της επόμενης κατάστασης, από μια κατάσταση και μια ενέργεια που του έχουν δοθεί αρχικά.

3.2 Μαρκοβιανές Διαδικασίες Απόφασης

Κάθε εργασία ή αλλιώς πρόβλημα στην Ενισχυτική Μάθηση αποτελεί *Μαρκοβιανή Διαδικασία Απόφασης (Markov Decision Process, MDPs)*. Πιο συγκεκριμένα, οι Μαρκοβιανές Διαδικασίες Απόφασης παρέχουν τη μαθηματική βάση για την μοντελοποίηση του προβλήματος στην Ενισχυτική Μάθηση.



Σχήμα 3.2.1: Η αλληλεπίδραση μεταξύ πράκτορα και περιβάλλον.

Μια Μαρκοβιανή Διαδικασία Απόφασης ορίζεται σύμφωνα με τα παρακάτω στοιχεία:

Για την αλληλεπίδραση πράκτορα – περιβάλλον θεωρούμε τον χρόνο διακριτό, $t = 0,1,2\ldots$.

- Ένα σύνολο των καταστάσεων S του περιβάλλοντος.
- Μια κατάσταση του περιβάλλοντος S_t που λαμβάνει ο πράκτορας, $S_t \in S$.
- Ένα σύνολο των ενεργειών A του πράκτορα. Με τον πράκτορα να μπορεί να επιλέξει την ενέργεια A_t από το σύνολο των πιθανών ενεργειών $A(s)$ για την κατάσταση που βρίσκεται, $A_t \in A(s)$.
- Η *Συνάρτηση Πιθανότητας Μετάβασης (Transition Probability Function)*

$$p(s' \mid s,a) = \Pr\{S_{t+1} = s' \mid S_t = s, A_t = a\} \qquad (3.2.1)$$

όπου s είναι η κατάσταση που βρίσκεται ο πράκτορας και a είναι η ενέργεια που εκτελεί ο πράκτορας για να μεταβεί στην επόμενη πιθανή κατάσταση s'.

- Η *Συνάρτηση Πιθανότητας Ανταμοιβής (Reward Probability Function)*

$$r(s,a,s') = \mathbb{E}\left[R_{t+1} \middle| S_t = s, A_t = a, S_{t+1} = s' \right] \tag{3.2.2}$$

όπου ο πράκτορας λαμβάνει την ανταμοιβή όταν μεταβεί στην κατάσταση s' με την διαδικασία που περιγράφτηκε προηγουμένως. Όπου $\mathbb{E}$ είναι η *Αναμενόμενη Τιμή (Expected Value)*.

3.3 Επιστρεφόμενη Ανταμοιβή

Όπως έχει ήδη αναφερθεί, ο στόχος του πράκτορα είναι να μεγιστοποιήσει την ανταμοιβή που λαμβάνει από το περιβάλλον. Αν για χρόνο t ο πράκτορας λαμβάνει μια σειρά από ανταμοιβές $R_{t+1}, R_{t+2}, R_{t+3}, ...,$ τότε η μέγιστη αναμενόμενη ανταμοιβή που θα επιστραφεί για ένα *Μοντέλο Πεπερασμένου Χρονικού Ορίζοντα (Finite Horizon)* ή αλλιώς *Επεισοδιακό (Episodic) Μοντέλο* είναι η εξίσωση

$$R_t = r_{t+1} + r_{t+2} + r_{t+3} + ... + r_T = \sum_{i=1}^{T} r_t + i \tag{3.3.1}$$

όπου T είναι η τελευταία χρονική στιγμή.

Στα Μοντέλα Πεπερασμένου Ορίζοντα ο πράκτορας έχει ένα συγκεκριμένο αριθμό επεισοδίων που κάθε επεισόδιο ξεκινάει με μια αρχική κατάσταση και τελειώνει με μια ειδική κατάσταση που λέγεται *terminal state*. Στη συνέχεια ξεκινάει καινούργιο επεισόδιο από την ίδια αρχική κατάσταση που χρησιμοποιήθηκε στο προηγούμενο επεισόδιο.

Στην περίπτωση όπου η αλληλεπίδραση μεταξύ πράκτορα και περιβάλλον δεν σταματάει φυσικά από συγκεκριμένο αριθμό επεισοδίων, αλλά συνεχίζει χωρίς κανένα περιορισμό, τότε η χρήση της εξίσωσης από το Μοντέλο Πεπερασμένου Ορίζοντα για την περίπτωση αυτή είναι προβληματική. Γιατί το $T = \infty$ και το άθροισμα μπορεί να πλησιάζει στο άπειρο. Το πρόβλημα αυτό λύνεται με την εισαγωγή της έννοιας της *έκπτωσης (discounting)* και ο πράκτορας τότε προσπαθεί να εκτελέσει ενέργειες, ώστε το άθροισμα των μειωμένων ανταμοιβών που λαμβάνει να μεγιστοποιείται στο μέλλον.

Η περίπτωση αυτή είναι ένα *Μοντέλο Άπειρου Χρονικού Ορίζοντα (Infinity Horizon)* και η εξίσωση που έχει είναι

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \ldots = \sum_{i=0}^{\infty} \gamma^i r_{t+i+1} \tag{3.3.2}$$

όπου γ είναι ο *συντελεστής μείωσης (discount factor)* με τιμές $0 \le \gamma < 1$.

Ο συντελεστής μείωσης γ καθορίζει την τρέχουσα αξία των μελλοντικών ανταμοιβών. Όταν $\gamma = 0$ ο πράκτορας συμπεριφέρεται μυωπικά και ενδιαφέρεται μόνο για τις άμεσες ανταμοιβές που θέλει να μεγιστοποιήσει. Ενώ όταν ο συντελεστής μείωσης πλησιάζει προς την μονάδα ($\gamma \to 1$), ο πράκτορας γίνεται πιο προβλέψιμος, αφού οι μελλοντικές ανταμοιβές γίνονται ολοένα και πιο σημαντικές.

Στο *Μοντέλο Μέσης Ανταμοιβής (Average Reward)* ο πράκτορας θέλει να μεγιστοποιήσει την μακροπρόθεσμη μέση ανταμοιβή σε κάθε χρονική στιγμή. Η εξίσωση της Μέσης Ανταμοιβής είναι

$$R_t = \lim_{h \to \infty} \frac{1}{h} \sum_{i=0}^{h} r_{t+i} \tag{3.3.3}$$

3.4 Συναρτήσεις Αξίας

Πολλοί μέθοδοι στην Ενισχυτική Μάθηση χρησιμοποιούν *συναρτήσεις αξίας (value functions)* για την επίλυσή τους. Οι συναρτήσεις αξίας αφορούν καταστάσεις ή ζεύγη κατάστασης – ενέργειας και εκτιμούν την κατάσταση που βρίσκεται ο πράκτορας ή για την ενέργεια που εκτέλεσε για συγκεκριμένη κατάσταση. Η εκτίμηση αυτή έχει να κάνει με τις μελλοντικές ανταμοιβές που αναμένει ο πράκτορας ανάλογα με τις ενέργειες που θα αποφασίσει να εκτελέσει. Στην Ενισχυτική Μάθηση οι μέθοδοι ορίζουν το πώς αλλάζει η πολιτική του πράκτορα μέσα από την εμπειρία του.

Η *συνάρτηση αξίας κατάστασης (state-value function)* υπό μια πολιτική π ορίζεται ως $V^{\pi}(s)$ και εκφράζει την αναμενόμενη ανταμοιβή, όταν ο πράκτορας βρίσκεται στην κατάσταση s και ακολουθήσει την πολιτική π.

$$V^{\pi}(s) = \mathbb{E}_{\pi}\left[\sum_{i=0}^{\infty} \gamma^i r_{t+i+1} \,\middle|\, S_t = s \right] \tag{3.4.1}$$

Η *συνάρτηση αξίας ενέργειας (action-value function)* υπό μια πολιτική π ορίζεται ως $Q^{\pi}(s,a)$ και εκφράζει την αναμενόμενη ανταμοιβή, όταν ο πράκτορας από την κατάσταση s επιλέξει την ενέργεια a και ακολουθήσει την πολιτική π.

$$Q^{\pi}(s,a) = \mathbb{E}_{\pi}\left[\sum_{i=0}^{\infty}\gamma^{i} r_{t+i+1} \,\middle|\, S_t = s, A_t = a\right] \tag{3.4.2}$$

Οι συναρτήσεις αξίας $V^{\pi}(s)$ και $Q^{\pi}(s,a)$ έχουν την ιδιότητα να ικανοποιούν τις αναδρομικές *εξισώσεις Bellman* που εκφράζουν την σχέση μεταξύ της αξίας μιας κατάστασης με την αξία των καταστάσεων που ακολουθούν.

$$V^{\pi}(s) = \sum_{a}\pi(s,a)\sum_{s'}p(s'|s,a)\left[r(s,a,s') + \gamma V^{\pi}(s')\right] \tag{3.4.3}$$

$$Q^{\pi}(s,a) = \sum_{s'}p(s'|s,a)\left[r(s,a,s') + \gamma\sum_{a}\pi(s',a')\,Q^{\pi}(s',a')\right] \tag{3.4.4}$$

3.4.1 Βέλτιστη Πολιτική και Συναρτήσεις Αξίας

Η πολιτική π που ψάχνει ο πράκτορας, έχει να κάνει με την μέγιστη ανταμοιβή που θα του επιστρέψει μακροπρόθεσμα. Μια πολιτική π είναι καλύτερη ή ίση με μία πολιτική $\pi'(\pi \geq \pi')$, όταν ισχύει $V^{\pi}(s) \geq V^{\pi'}(s)$, $\forall s \in S$. Η πολιτική αυτή ονομάζεται *βέλτιστη πολιτική (optimal policy)* και συμβολίζεται με π^{*}.

Στην βέλτιστη πολιτική π^{*} η συνάρτηση $V^{\pi}(s)$ ονομάζεται *βέλτιστη συνάρτηση αξίας κατάστασης (optimal state-value function)* και ορίζεται ως $V^{*}(s)$.

$$V^{*}(s) = \max_{\pi} V^{\pi}(s), \ \forall s \in S \tag{3.4.5}$$

Ενώ για την συνάρτηση $Q^{\pi}(s,a)$ στην βέλτιστη πολιτική π^{*} ονομάζεται *βέλτιστη συνάρτηση αξίας ενέργειας (optimal action-value function)* και ορίζεται ως $Q^{*}(s,a)$.

$$Q^{*}(s,a) = \max_{\pi} Q^{\pi}(s,a), \ \forall s \in S, a \in A(s) \tag{3.4.6}$$

Οι *εξισώσεις του Bellman* για μια βέλτιστη πολιτική π^* ονομάζονται *βέλτιστες εξισώσεις του Bellman* και ορίζονται ως εξής.

$$V^*(s) = \max_a \sum_{s'} p(s'|s,a)\left[r(s,a,s') + \gamma V^*(s')\right] \tag{3.4.7}$$

$$Q^*(s,a) = \sum_{s'} p(s'|s,a)\left[r(s,a,s') + \gamma \max_{a'} Q^*(s',a')\right] \tag{3.4.8}$$

3.5 Μάθηση Χρονικών Διαφορών

Η *Μάθηση Χρονικών Διαφορών (Temporal-Difference Learning)* αποτελεί κεντρική και καινοτόμα ιδέα για την Ενισχυτική Μάθηση. Οι μέθοδοι Μάθησης Χρονικών Διαφορών αποτελούν ένα συνδυασμό των μεθόδων *Δυναμικού Προγραμματισμού (Dynamic Programming)* και των μεθόδων *Monte Carlo*. Όπως συμβαίνει στις μεθόδους Monte Carlo, έτσι και στις μεθόδους Χρονικών Διαφορών για την μάθηση χρησιμοποιείται η εμπειρία αλληλεπίδρασης του πράκτορα με το περιβάλλον, χωρίς όμως να είναι αναγκαία η γνώση για το μοντέλο του περιβάλλοντος. Επιπλέον, όπως στις μεθόδους Δυναμικού Προγραμματισμού, έτσι και στις μεθόδους Χρονικών Διαφορών οι εκτιμήσεις για τις συναρτήσεις αξίας ενημερώνονται αμέσως σε κάθε *βήμα (step)* έχοντας γνώσει τις εκτιμήσεις που έχει μάθει ο πράκτορας μέχρι εκείνη την στιγμή. Χωρίς όμως να περιμένουν το τέλος ενός *επεισοδίου (episode)* για να τις ενημερώσουν όπως συμβαίνει στις μεθόδους Monte Carlo.

3.5.1 Αλγόριθμος Μάθησης Q-Learning

Ο αλγόριθμος *Q-Learning* (Watkins 1989, Watkins and Dayan, 1992) [62] [63] αποτελεί μια μέθοδο Χρονικών Διαφορών και είναι αλγόριθμος *εκτός πολιτικής (off-policy)*, που δεν ακολουθεί κάποια συγκεκριμένη πολιτική για την επιλογή των επόμενων ενεργειών. Αποτελεί έναν από τους πιο σημαντικούς και δημοφιλής αλγορίθμους της Ενισχυτικής Μάθησης.

Σκοπός του αλγορίθμου Q-Learning είναι να μάθει τις βέλτιστες τιμές της συνάρτησης Q που αντιστοιχούν στην εκμάθηση μιας βέλτιστης πολιτικής. Ο πράκτορας μαθαίνει την βέλτιστη πολιτική ακόμη και όταν οι ενέργειες επιλέγονται από την εξερεύνηση ή την εκμετάλλευση.

Ο τρόπος με τον οποίο λειτουργεί ο αλγόριθμος Q-Learning είναι με τον πράκτορα να παρατηρεί την κατάσταση s που βρίσκεται και να επιλέγει, αλλά και να εκτελεί την ενέργεια a. Παρατηρεί την καινούργια κατάσταση s' που έχει μεταβεί, λαμβάνει αμέσως μια ανταμοιβή r και υπολογίζεται η ενημερωμένη τιμή της συνάρτησης αξίας $Q(s,a)$.

Η συνάρτηση αξίας για τον αλγόριθμο Q-Learning ορίζεται ως εξής.

$$Q(s,a) \leftarrow Q(s,a) + \alpha\left[r + \gamma \max_{a'} Q(s',a') - Q(s,a)\right] \tag{3.5.1}$$

όπου γ είναι ο *συντελεστής μείωσης (discount factor)* με τιμές $0 \le \gamma < 1$ και α ο *ρυθμός μάθησης (learning rate)* με τιμές $0 < \alpha \le 1$.

Ο ρυθμός μάθησης α καθορίζει την αξία των νέων τιμών σε σχέση με τις προηγούμενες τιμές. Όταν ο ρυθμός μάθησης πλησιάζει προς το μηδέν ($\alpha \to 0$), οι ενημερωμένες τιμές της συνάρτησης Q παραμένουν σχεδόν σταθερές και ο πράκτορας μαθαίνει πάρα πολύ αργά έως και σχεδόν καθόλου. Όταν $\alpha = 1$ οι ενημερωμένες τιμές της συνάρτησης Q αλλάζουν και ο πράκτορας ενώ μαθαίνει πολύ γρήγορα εστιάζει στις πρόσφατες τιμές αγνοώντας την προηγούμενη γνώση που είχε αποκτήσει.

Αλγόριθμος $Q-Learning$

Intialize $Q(s,a),\ \forall s \in S, a \in A(s)$
Loop for each episode:
  Intialize s
  Loop for each step of episode:
    Choose a from s using policy derived from Q (e.g., ε-greedy)
    Take action a, observe r, s'
    Update $Q(s,a)$:
      $Q(s,a) \leftarrow Q(s,a) + \alpha\left[r + \gamma \max_{a'} Q(s',a') - Q(s,a)\right]$
    $s \leftarrow s'$
  until s is terminal

3.5.2 Αλγόριθμος Μάθησης SARSA

Ο αλγόριθμος *SARSA* (Rummery and Niranjan, 1994) [47] αποτελεί μια μέθοδο Χρονικών Διαφορών και είναι αλγόριθμος *εντός πολιτικής (on-policy)* που ακολουθεί μια πολιτική για την επιλογή των επόμενων ενεργειών. Τα στοιχεία (s,a,r,s',a') είναι αυτά που έδωσαν και το όνομα στον αλγόριθμο.

Ο τρόπος με τον οποίο λειτουργεί ο αλγόριθμος SARSA είναι με τον πράκτορα να βρίσκεται σε μια κατάσταση s και να επιλέγει την ενέργεια a που θα εκτελέσει. Λαμβάνει μια ανταμοιβή r, μεταβαίνει σε μια καινούργια κατάσταση s', επιλέγει την επόμενη ενέργεια a'

για την πολιτική που ακολουθεί και υπολογίζεται η ενημερωμένη τιμή της συνάρτησης αξίας $Q(s,a)$.

Η συνάρτηση αξίας για τον αλγόριθμο SARSA ορίζεται ως εξής.

$$Q(s,a) \leftarrow Q(s,a) + \alpha\left[r + \gamma Q(s',a') - Q(s,a)\right] \tag{3.5.2}$$

όπου γ είναι ο *συντελεστής μείωσης (discount factor)* με τιμές $0 \le \gamma < 1$ και α ο *ρυθμός μάθησης (learning rate)* με τιμές $0 < \alpha \le 1$.

Αλγόριθμος *SARSA*

Intialize $Q(s,a),\ \forall s \in S, a \in A(s)$
Loop for each episode:
  Intialize s
  Choose a from s using policy from Q (e.g., ε-greedy)
  Loop for each step of episode:
    Take action a, observe r, s'
    Choose a' from s' using policy derived from Q (e.g., ε-greedy)
    Update $Q(s,a)$:
      $Q(s,a) \leftarrow Q(s,a) + \alpha\left[r + \gamma Q(s',a') - Q(s,a)\right]$
    $s \leftarrow s'$
    $a \leftarrow a'$
  until s is terminal

3.6 Το Δίλλημα Εξερεύνησης ή Εκμετάλλευσης

Ένα σημαντικό ζήτημα που δημιουργείται κατά την διάρκεια της μάθησης στην Ενισχυτική Μάθηση και απασχολεί την μέθοδο Q-Learning είναι η *αντιστάθμιση (trade-off)* μεταξύ *εξερεύνησης (exploration)* και *εκμετάλλευσης (exploitation)*.

Ο πράκτορας για να λάβει όσο το δυνατόν την μεγαλύτερη ανταμοιβή, θα πρέπει να επιλέξει ενέργειες που έχει δοκιμάσει στο παρελθόν που αποδείχτηκαν ότι του επιστρέφουν αυτή την ανταμοιβή. Προκειμένου να συμβεί αυτό, θα πρέπει πρώτα ο πράκτορας να βρεθεί σε καταστάσεις και να επιλέξει ενέργειες που δεν έχει δοκιμάσει προηγουμένως.

Ο πράκτορας στην εξερεύνηση θέλει να συγκεντρώσει πληροφορία για το περιβάλλον και τις ανταμοιβές όταν εξερευνεί άγνωστες καταστάσεις και ενέργειες, με σκοπό να κάνει καλύτερες επιλογές ενεργειών στο μέλλον. Επιπλέον, την πληροφορία που έχει συλλέξει θα

πρέπει να την εκμεταλλευτεί για τη μεγιστοποίηση της ανταμοιβής του. Μια μέθοδος για την ισορροπία μεταξύ εξερεύνησης και εκμετάλλευσης είναι η ε-greedy.

Στην μέθοδο ε-greedy η παράμετρος $\varepsilon \in [0,1]$ είναι η πιθανότητα επιλογής τυχαίας ενέργειας για την κατάσταση στην οποία βρίσκεται ο πράκτορας και $1-\varepsilon$ είναι η πιθανότητα επιλογής της καλύτερης ενέργειας. Συνήθως η υλοποίηση της μεθόδου γίνεται αρχικά με ένα υψηλό ε (εξερεύνηση) που σταδιακά μειώνεται και φτάνει κοντά στο μηδέν (εκμετάλλευση).

4. Μηχανική Όραση (Machine Vision)

Μια από τις πιο σημαντικές αισθήσεις του ανθρώπου είναι η όραση μιας και τα περισσότερα δεδομένα προέρχονται από αυτή. Το ανθρώπινο οπτικό σύστημα έχει την δυνατότητα να λαμβάνει δεδομένα και να τα ερμηνεύει αρκετά γρήγορα και με ιδιαίτερη ευκολία. Οι ερμηνείες αυτές έχουν να κάνουν για παράδειγμα, με την αναγνώριση προσώπων, χρωμάτων, σχημάτων ή την ανάλυση μιας σκηνής (βουνό, δάσος, δέντρα, πέτρες, χώμα). Σήμερα υπάρχουν μηχανές που χρησιμοποιούν την Μηχανική Όραση και καταφέρνουν σε ικανοποιητικό επίπεδο να μιμηθούν τον τρόπο λειτουργίας του ανθρώπινου οπτικού συστήματος.

Το πεδίο της *Μηχανικής Όρασης (Machine Vision)* αναπτύσσει και χρησιμοποιεί αλγορίθμους, ώστε ένα υπολογιστικό σύστημα να επεξεργαστεί τα δεδομένα που λαμβάνει από εικόνες και βίντεο με σκοπό την επίτευξη της εργασίας του. Προκειμένου να συμβεί αυτό, θα πρέπει να ληφθούν οι εικόνες και τα βίντεο μέσω αισθητήρων εικόνας και να γίνει ανάλυση της εικόνας χρησιμοποιώντας εργαλεία της επεξεργασίας εικόνας. Λόγω της απλότητας που προσφέρει η Μηχανική Όραση στην εισαγωγή πληροφορίας (εικόνα και βίντεο) σε ένα σύστημα την καθιστούν σημαντική σε διάφορους τομείς.

Κάποιες εφαρμογές που εντοπίζονται είναι στην ρομποτική και στα αυτοκινούμενα οχήματα, όπως η αυτόνομη πλοήγηση των ρομποτικών οχημάτων για την εξερεύνηση του πλανήτη Άρη, αλλά και σε ορισμένα αυτοκινούμενα οχήματα για την αυτόνομη πλοήγηση τους στους δρόμους.

Επιπλέον, εντοπίζονται εφαρμογές στην ιατρική, όπως η χειρουργική επέμβαση στα μάτια με λέιζερ και στην βιομηχανία, όπως ο έλεγχος ποιότητας των προϊόντων στις γραμμές παραγωγής. Τέλος, εντοπίζονται εφαρμογές σε συστήματα ασφαλείας, όπως η αναγνώριση προσώπου και πινακίδων κυκλοφορίας οχημάτων, η αναγνώριση δακτυλικών αποτυπωμάτων και προσώπων από τις αστυνομικές αρχές και τις υπηρεσίες πληροφοριών, αλλά και στρατιωτικές εφαρμογές για την αναγνώριση και παρακολούθηση στόχων.

4.1 Εξαγωγή Τοπικών Χαρακτηριστικών Εικόνας

Για την αναγνώριση ενός αντικειμένου πρέπει να εντοπιστούν κάποια τοπικά χαρακτηριστικά από περιοχές της εικόνας ή του βίντεο όπου εμφανίζεται το αντικείμενο. Τα τοπικά χαρακτηριστικά αποτελούνται από συγκεκριμένα σημεία στην εικόνα, όπως ακμές, γωνίες ή

σφαιρίδια και ονομάζονται *σημεία ενδιαφέροντος (keypoint feature ή interest points)*. Τα σημεία ενδιαφέροντος εμφανίζονται ως μικρές επιφάνειες *εικονοστοιχείων (pixels)* γύρω από τα συγκεκριμένα σημεία.

Η εξαγωγή των τοπικών χαρακτηριστικών γίνεται με το να εντοπιστούν τα σημεία ενδιαφέροντος και να γίνει η περιγραφή αυτών. Τα τοπικά χαρακτηριστικά θεωρούνται καλά όταν δεν επηρεάζονται από τις μεταβολές της κλίμακας και της περιστροφής, αλλά και από τον θόρυβο και την διακύμανση της φωτεινότητας.

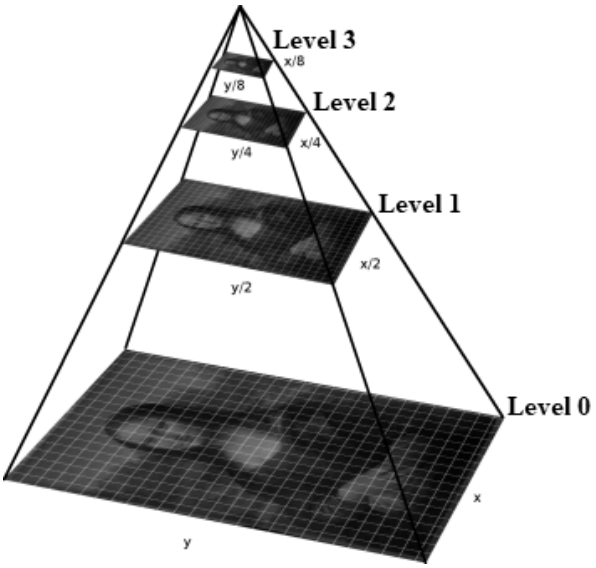
Ο εντοπισμός των σημείων ενδιαφέροντος γίνεται από μεθόδους που χρησιμοποιούν αλγορίθμους και αναφέρονται ως *ανιχνευτές χαρακτηριστικών (feature detectors)*. Η περιγραφή ενός σημείου ενδιαφέροντος γίνεται με ένα διάνυσμα που αναπαριστά την μικρή επιφάνεια εικονοστοιχείων γύρω από το συγκεκριμένο σημείο. Ο υπολογισμός της περιγραφής κάθε σημείου ενδιαφέροντος γίνεται από μεθόδους που χρησιμοποιούν αλγορίθμους και αναφέρονται ως *περιγραφείς χαρακτηριστικών (feature descriptors)*.

4.1.1 ORB (Oriented FAST and Rotated BRIEF)

Η εξαγωγή των τοπικών χαρακτηριστικών από μια εικόνα μπορεί να γίνει με διάφορους αλγορίθμους. Ένας τέτοιος αλγόριθμος είναι ο *ORB* (Ethan Rublee, Vincent Rabaud, Kurt Konolige και Gary Bradski, 2011) [46] που αποτελεί μια αποτελεσματική εναλλακτική σε σχέση με τους αλγορίθμους *SIFT* και *SURF*.

Ο ORB παρουσιάζει σταθερότητα στην περιστροφή, ανθεκτικότητα στον θόρυβο της εικόνας και μπορεί να χρησιμοποιηθεί σε εφαρμογές πραγματικού χρόνου και χαμηλής κατανάλωσης ενέργειας. Επίσης είναι δύο φορές πιο γρήγορος και ελεύθερος προς χρήση σε σχέση με τον SIFT και τον SURF. Ο αλγόριθμος ORB βασίζεται στον ανιχνευτή χαρακτηριστικών *FAST (Features from Accelerated Segment Test)* [43] [44] [45] και στον περιγραφέα χαρακτηριστικών *BRIEF (Binary Robust Independent Elementary Features)* [11] με κάποιες τροποποιήσεις αυτών για να γίνει πιο αποδοτικός.

Αρχικά χρησιμοποιείται ο FAST για τον εντοπισμό των σημείων ενδιαφέροντος και γίνεται εφαρμογή της μέτρησης γωνίας *Harris* [17] για να εντοπιστούν τα καλύτερα N σημεία. Επιπλέον, για μεταβολές της κλίμακας γίνεται εφαρμογή πυραμίδας κλιμάκωσης της εικόνας, ώστε να δημιουργηθούν τα σημεία και στα αλλά επίπεδα της πυραμίδας.



Σχήμα 4.1.1: Παράδειγμα πυραμίδας εικόνας (image pyramid) με διαφορετικές κλίμακες.

Για τον προσανατολισμό ενός σημείου ενδιαφέροντος γίνεται μέτρηση του προσανατολισμού της γωνίας υπολογίζοντας την μέθοδο *κεντροειδής έντασης (intensity centroid)* του Paul L. Rosin [42]. Σε αυτή την μέθοδο οι ροπές μιας επιφάνειας εικονοστοιχείων ορίζονται ως εξής.

$$m_{pq} = \sum_{x,y} x^p y^q I(x,y) \tag{4.1.1}$$

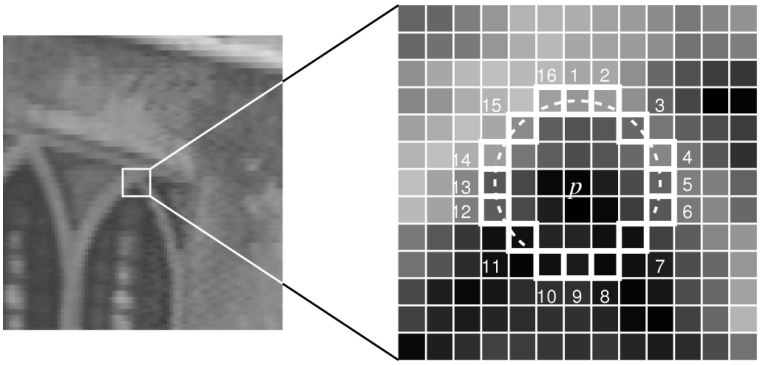
Οι *ροπές (moments)* περιγράφουν την μορφή ενός σχήματος σε μια εικόνα, όπως η διάταξη των εικονοστοιχείων.

Με αυτές τις ροπές οι συντεταγμένες της κεντροειδής υπολογίζονται ως εξής.

$$C = \left(\frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}} \right) \tag{4.1.2}$$

Έχοντας βρει την C, κατασκευάζεται το διάνυσμα $\overrightarrow{OC}$ με O το κέντρο της γωνίας και ο προσανατολισμός της επιφάνειας υπολογίζεται ως εξής.

$$\theta = a\tan 2(m_{01}, m_{10}) \tag{4.1.3}$$



Σχήμα 4.1.2: Παράδειγμα επιφάνειας εικόνας (image patch) ενός σημείου ενδιαφέροντος με *p* να είναι το κέντρο της γωνίας.

Ο BRIEF χρησιμοποιείται για την περιγραφή των σημείων ενδιαφέροντος που είναι η δυαδική αναπαράσταση της επιφάνειας εικόνας, μέσω της σύγκρισης της έντασης των εικονοστοιχείων σε δυαδική μορφή που βρίσκονται γύρω από το σημείο.

Ένα *δυαδικός έλεγχος (binary test)* μπορεί να γίνει ως εξής.

$$\tau(P;x,y) := \begin{cases} 1 \ : \ P(x) < P(y) \\ 0 \ : \ P(x) \le P(y) \end{cases} \tag{4.1.4}$$

όπου P είναι η ένταση του εικονοστοιχείου στο σημείο x. Ο περιγραφέας ορίζεται από το διάνυσμα μήκους $n = 256$ ως εξής.

$$f_n(P) := \sum_{1 \le i \le n} 2^{i-1} \tau(P;x_i,y_i) \tag{4.1.5}$$

Για την περιστροφή γίνεται χρήση των προσανατολισμένων σημείων ενδιαφέροντος. Πιο συγκεκριμένα, για ένα σύνολο n δυαδικών ελέγχων στην θέση (x_i, y_i) ορίζεται ο πίνακας $2 \times n$ ως εξής.

$$S = \begin{pmatrix} x_1,...,x_n \\ y_1,...,y_n \end{pmatrix} \tag{4.1.6}$$

Χρησιμοποιώντας τον προσανατολισμό της επιφάνειας θ και τον πίνακα περιστροφής R_θ μπορεί να κατασκευαστεί η κατευθυνόμενη έκδοση S_θ του S ως εξής.

$$S_\theta = R_\theta S \tag{4.1.7}$$

όπου $R_\theta = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$ και η γωνία περιστροφής διακριτοποιείται σε αυξήσεις $2\pi/30$, που είναι 12 μοίρες.

Τότε ο τελεστής του περιστρεφόμενου περιγραφέα BRIEF ορίζεται ως εξής.

$$g_n(P,\theta) := f_n(p) \big| (x_i, y_i) \in S_\theta \tag{4.1.8}$$

4.2 Αντιστοίχιση Χαρακτηριστικών Εικόνας

Για να γίνει η ταύτιση μεταξύ δύο εικόνων θα πρέπει να εντοπιστούν και να περιγραφούν τα σημεία ενδιαφέροντος για κάθε εικόνα ξεχωριστά. Έτσι ώστε στην συνέχεια να γίνει η διαδικασία της αντιστοίχισης για να εξακριβωθεί εάν υπάρχει ταύτιση των δύο εικόνων.

Η διαδικασία της αντιστοίχησης ψάχνει για ομοιότητες μεταξύ των χαρακτηριστικών των εικόνων μέσω της σύγκρισης των περιγραφόμενων σημείων τους. Αυτό επιτυγχάνεται από μεθόδους *αντιστοίχισης χαρακτηριστικών (feature matching)*, όπως ο Brute-Force και ο FLANN.

Ο *Brute-Force* χρησιμοποιεί ένα περιγραφόμενο σημείο ενδιαφέροντος από την πρώτη εικόνα και αναζητεί ένα όμοιο του, εξετάζοντας ξεχωριστά το κάθε περιγραφόμενο σημείο ενδιαφέροντος από την δεύτερη εικόνα, αλλά και υπολογίζοντας την απόσταση μεταξύ τους. Η μικρότερη απόσταση είναι αυτή που θα καθορίσει την επιλογή του χαρακτηριστικού της δεύτερης εικόνας. Για την μέτρηση της απόστασης χρησιμοποιείται η *Ευκλείδεια* απόσταση που είναι καλή για την SIFT, SURF ή η απόσταση *Hamming* για περιγραφείς με δυαδική μορφή, όπως στον ORB.

Ο *FLANN (Fast Library for Approximate Nearest Neighbors)* [32] είναι μια βιβλιοθήκη που περιέχει αλγορίθμους για να βρίσκει αντιστοιχίες και λειτουργεί πιο γρήγορα σε σχέση με τον Brute-Force. Οι αλγόριθμοι, όπως ο *randomized kd-trees* και ο *hierarchical k-means tree* χρησιμεύουν για την γρήγορη αναζήτηση των κοντινότερων γειτόνων σε μεγάλα σύνολα δεδομένων και χαρακτηριστικά που έχουν μεγάλες διαστάσεις. Επιπλέον, στον FLANN υπάρχει η δυνατότητα αυτόματης επιλογής του βέλτιστου αλγορίθμου.

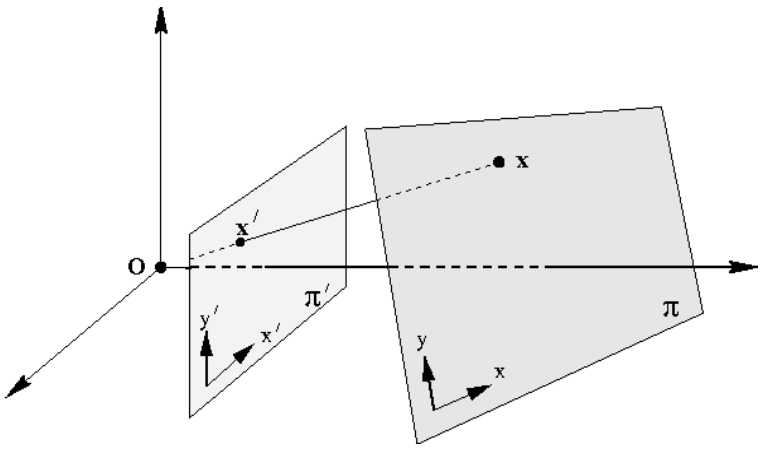
4.3 Ομογραφία (Homography)

Η *ομογραφία (homography)* είναι μαθηματικός όρος και αναφέρεται στην απεικόνιση σημείων μιας επιφάνειας σε μια άλλη. Στην Μηχανική Όραση αναφέρεται στην προβολική

απεικόνιση των σημείων μιας εικόνας σε μία άλλη. Συνώνυμοι όροι της ομογραφίας είναι η *προβολικότητα (projectivity)*, ο *συγγραμμισμός (collineation)* και ο *προβολικός μετασχηματισμός (projective transformation)*. Στο επίπεδο η ομογραφία έχει *8 βαθμούς ελευθερίας (8 DoF)* και είναι ένας γραμμικός μετασχηματισμός που γίνεται σε ομογενή διανύσματα τριών στοιχείων και αναπαριστάται από έναν 3×3 πίνακα ως εξής.

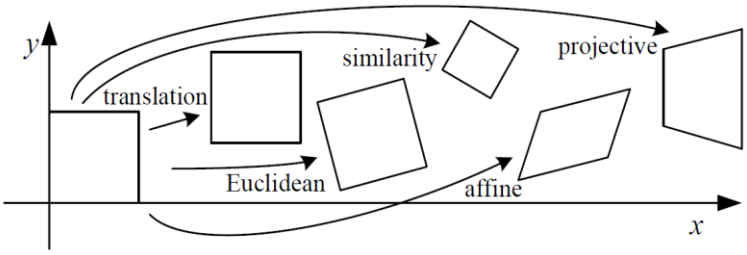
$$x' = Hx \Leftrightarrow \begin{pmatrix} x_1' \\ x_2' \\ x_3' \end{pmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \tag{4.3.1}$$

όπου H είναι ο πίνακας μετασχηματισμού που για να βρεθεί χρειάζονται οι αντιστοιχίες τουλάχιστον τεσσάρων σημείων, x είναι το σημείο στο $\mathbb{P}^2$ που παρουσιάζεται ως ομογενές διάνυσμα τριών στοιχείων και x' είναι το μετασχηματισμένο διάνυσμα. Το $\mathbb{P}^2$ είναι το *δισδιάστατο (2D)* προβολικό επίπεδο.



Σχήμα 4.3.1: Προβολική απεικόνιση σημείων ενός επιπέδου στο άλλο επίπεδο.

Τα διάφορα είδη μετασχηματισμών στο δισδιάστατο επίπεδο είναι ο *μετασχηματισμός μετατόπισης (translation transformation)* με 2 βαθμούς ελευθερίας, ο *μετασχηματισμός περιστροφής και μετατόπισης (rotation + translation transformation)* ή αλλιώς *Ευκλείδειος μετασχηματισμός (Euclidean transformation)* με 3 βαθμούς ελευθερίας, ο *μετασχηματισμός ομοιότητας (similarity transformation)* με 4 βαθμούς ελευθερίας, ο *συσχετισμένος μετασχηματισμός (affine transformation)* με 6 βαθμούς ελευθερίας και ο *προβολικός μετασχηματισμός (projective transformation)* με 8 βαθμούς ελευθερίας.



Σχήμα 4.3.2: Είδη μετασχηματισμών στο δισδιάστατο επίπεδο.

Πίνακας 4.3.1: Ιεραρχία των Μετασχηματισμών στο Δισδιάστατο Επίπεδο

Transformation	Matrix	# DoF	Preserves	Icon
translation	$\left[\, \boldsymbol{I} \mid \boldsymbol{t} \,\right]_{2\times3}$	2	orientation	
rigid (Euclidean)	$\left[\, \boldsymbol{R} \mid \boldsymbol{t} \,\right]_{2\times3}$	3	lengths	
similarity	$\left[\, s\boldsymbol{R} \mid \boldsymbol{t} \,\right]_{2\times3}$	4	angles	
affine	$\left[\, \boldsymbol{A} \,\right]_{2\times3}$	6	parallelism	
projective	$\left[\, \tilde{\boldsymbol{H}} \,\right]_{3\times3}$	8	straight lines	

4.3.1 RANSAC (Random Sample Consensus)

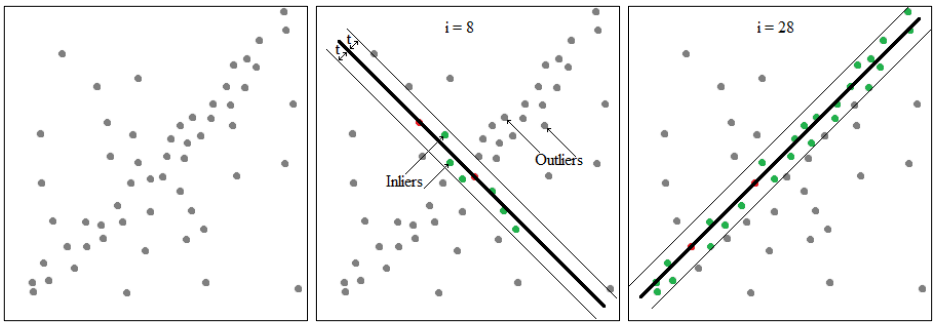
Κατά την διαδικασία της αντιστοίχησης των χαρακτηριστικών της μιας εικόνας με την άλλη, είναι δυνατόν να γίνει εσφαλμένη ταύτιση μερικών χαρακτηριστικών μεταξύ των δύο εικόνων που μπορεί να οδηγήσει σε αλλοίωση του τελικού αποτελέσματος. Για τον λόγο αυτό θα πρέπει να γίνει ένα είδος φιλτραρίσματος, ώστε να βελτιωθεί η αντιστοίχηση των εικόνων.

Ο αλγόριθμος *RANSAC* (Fischler and Bolles, 1981) [15] είναι ένας ισχυρός εκτιμητής που στόχο έχει την εκτίμηση παραμέτρων ενός μαθηματικού μοντέλου από ένα σύνολο δεδομένων που μπορεί να περιέχουν δεδομένα με ακραίες τιμές (*outliers*). Η διαδικασία που ακολουθείται στον αλγόριθμο είναι η εξής.

1. Αρχικά επιλέγεται ένα τυχαίο δείγμα σημείων δεδομένων από ένα σύνολο δεδομένων και δημιουργείται το μοντέλο από αυτές τις παραμέτρους.
2. Προσδιορίζεται ο αριθμός των δεδομένων με τις αποδεκτές τιμές (*inliers*) που βρίσκονται εντός του προκαθορισμένου ορίου απόστασης *t* του μοντέλου.
3. Εάν ο αριθμός των inliers είναι μεγαλύτερός από ένα προκαθορισμένο όριο *T* γίνεται επανεκτίμηση των παραμέτρων του μοντέλου χρησιμοποιώντας όλα τα αναγνωρισμένα inliers που έχουν εντοπιστεί και τερματίζεται.

4. Εάν ο αριθμός των inliers είναι μικρότερος από το προκαθορισμένο όριο *T* επιλέγονται καινούργια τυχαία σημεία δεδομένων και επαναλαμβάνεται η ίδια διαδικασία, όπως έχει περιγραφεί προηγουμένως.
5. Τέλος, μετά από έναν αριθμό δοκιμών *N* επιλέγεται το μεγαλύτερο σύνολο των inliers και το μοντέλο επανεκτιμάται χρησιμοποιώντας όλα τα αναγνωρισμένα inliers.

Στον RANSAC εντοπίζονται οι παράμετροι *t*, *T* και *N*. Η παράμετρος *t* χρησιμοποιείται για να καθοριστεί αν ένα δεδομένο από την απόστασή του από το μοντέλο είναι inlier. Η παράμετρος *T* είναι ο αριθμός των inliers που χρησιμοποιούνται για να προσδιοριστεί ότι έχει βρεθεί το σωστό μοντέλο. Η παράμετρος *N* καθορίζει τον αριθμό των δοκιμών που χρειάζονται για βρεθεί το σωστό μοντέλο.



Σχήμα 4.3.3: Εύρεση σωστού μοντέλου από ένα σύνολο δεδομένων που περιέχει outliers. Το τυχαίο δείγμα σημείων δεδομένων έχει χρώμα κόκκινο και το i είναι το σύνολο των inliers.

5. Σχεδιασμός Αυτόνομου Ρομποτικού Οχήματος

Σκοπός του ρομπότ είναι να κινηθεί στον περιβάλλον του χρησιμοποιώντας Μηχανική Μάθηση και να προσεγγίσει ένα συγκεκριμένο αντικείμενο, αφού πρώτα το έχει εντοπίσει χρησιμοποιώντας Μηχανική Όραση. Για την επίτευξη αυτού του σκοπού, θα πρέπει να σχεδιαστεί ένα σύστημα που να ανταποκρίνεται στις απαιτήσεις του προβλήματος.

Για το σχεδιασμό του ρομπότ έχει επιλεγεί η σχεδίαση ενός κινητού ρομπότ που για την κίνηση του χρησιμοποιούνται για παράδειγμα, κινητήρες, τροχοί, αισθητήρας απόστασης και για τον εντοπισμό του αντικειμένου χρησιμοποιείται μια κάμερα. Τα εξαρτήματα αυτά, μαζί με άλλα συνδέονται με ένα μικρό υπολογιστή που με τον κατάλληλο προγραμματισμό επιτυγχάνεται η εργασία του ρομποτικού οχήματος.

5.1 Υλικό (Hardware)

5.1.1 Raspberry Pi 3 Model B

Το *Raspberry Pi* είναι ένας μικρός υπολογιστής σε μέγεθος πιστωτικής κάρτας και αποτελεί τον εγκέφαλο του ρομποτικού οχήματος. Είναι μια μικρή πλακέτα που διαθέτει όλα τα κύρια χαρακτηριστικά ενός υπολογιστή. Για παράδειγμα, διαθέτει επεξεργαστή, μνήμη RAM, κάρτα γραφικών, θύρα Ethernet, Bluetooth, Wi-Fi, θύρες USB για την σύνδεση περιφερειακών συσκευών, όπως οθόνη, ποντίκι, πληκτρολόγιο.

Αναπτύχθηκε στο Ηνωμένο Βασίλειο από την *Raspberry Pi Foundation* με σκοπό την προώθηση της εκπαίδευσης της επιστήμης των υπολογιστών στα σχολεία. Παρόλα αυτά, λόγω της μεγάλης υπολογιστικής ισχύς και του χαμηλού κόστους που διαθέτει η μικρή πλακέτα, εντοπίζεται σε διάφορους τομείς στην έρευνα και στην βιομηχανία, αλλά και σε διάφορα projects, όπως ρομποτικά οχήματα και οικιακούς αυτοματισμούς.

Η υπολογιστική ισχύς έχει σημαντικό ρόλο σε ένα σύστημα ειδικά όταν αυτό δέχεται πολλά δεδομένα και θα πρέπει να τα επεξεργαστεί γρήγορα για την επίτευξη της εργασίας του. Το Raspberry Pi με τις όλο και πιο αναβαθμισμένες εκδόσεις του παρέχει αυτή την δυνατότητα, καθώς επίσης και με τους διάφορους ακροδέκτες εισόδου-εξόδου (*I/O*) που διαθέτει, το καθιστούν μια καλή επιλογή για ένα σύστημα.

Για τον σχεδιασμό του ρομποτικού οχήματος έχει επιλεγεί το *Raspberry Pi 3 Model B* που κυκλοφόρησε τον Φεβρουάριο του 2016 και διαθέτει επεξεργαστή 1.2GHz 64-bit Quad-Core

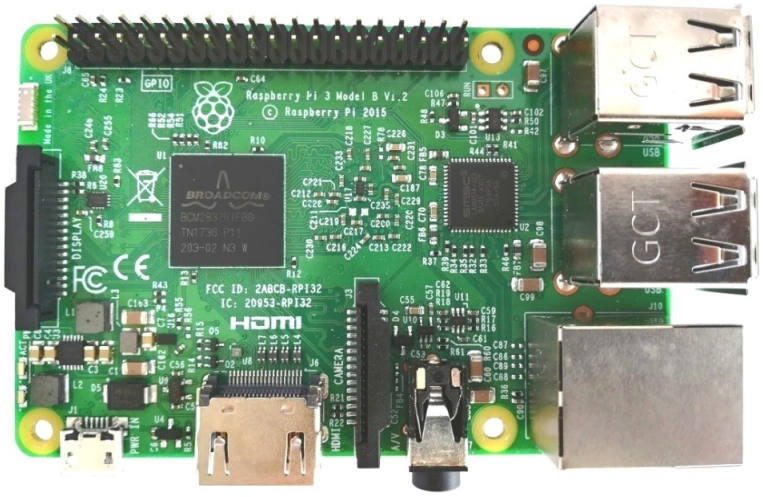
ARM Cortex-A53 (ARMv8), 1GB RAM, ενσωματωμένο 802.11b/g/n Wireless LAN, Bluetooth 4.1 και μια θύρα Micro USB 5V DC 2.5A για την τροφοδοσία. Έχει διαστάσεις 85 × 56 × 17 mm και διαθέτει υποδοχή για κάρτα μνήμης Micro SD για να φορτώνεται το λειτουργικό σύστημα και να αποθηκεύονται δεδομένα. Αναλυτικά με τα χαρακτηριστικά και την συνδεσιμότητα του Raspberry Pi 3 Model B παρουσιάζονται παρακάτω.

Πίνακας 5.1.1: Τεχνικά Χαρακτηριστικά Raspberry Pi 3 Model B

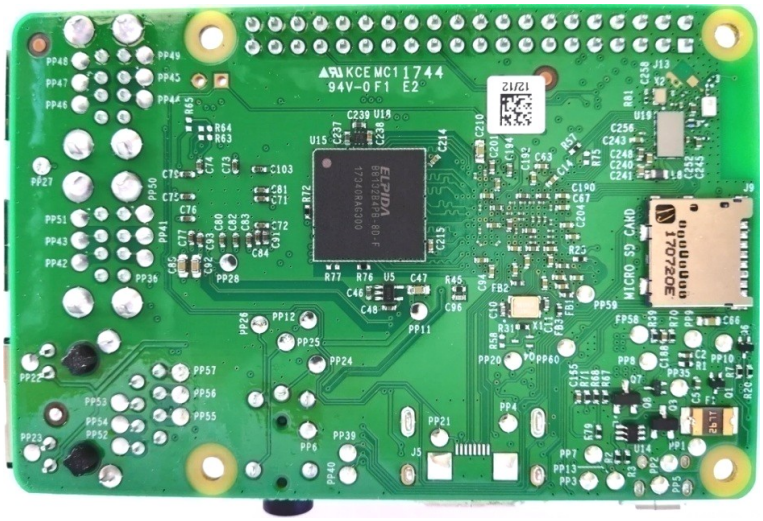
Feature	**Detail**
System on a Chip (SoC)	• Broadcom BCM2837
CPU	• 1.2GHz 64-bit Quad-Core ARM Cortex-A53 (ARMv8)
GPU	• Broadcom VideoCore IV 3D
Wireless Connectivity	• Broadcom BCM43438 chip • 2.4GHz 802.11b/g/n Wireless LAN • Bluetooth 4.1 Classic, Bluetooth Low Energy
RAM	• 1GB LPDDR2
Power	• Micro USB socket 5V DC 2.5A
Operating System	• Running a version of the Linux operating system or Windows 10 Internet of Things (*IoT*)
Dimensions	• 85 × 56 × 17 mm
Weight	• 45g

Πίνακας 5.1.2: Συνδεσιμότητα Raspberry Pi 3 Model B

Feature	**Detail**
Ethernet	• 10/100 Ethernet
Video Output	• 1 × full size HDMI
Audio Output	• 3.5mm analogue audio-video jack, HDMI, 4 × USB 2.0
Camera Connector	• 15-pin MIPI Camera Serial Interface (*CSI-2*)
Display Connector	• Display Serial Interface (*DSI*) 15 way Flexible Flat Cable (*FFC*) connector with two data lanes and a clock lane
GPIO Connector	• 40-pin GPIO header
Memory Card Slot	• Micro SD

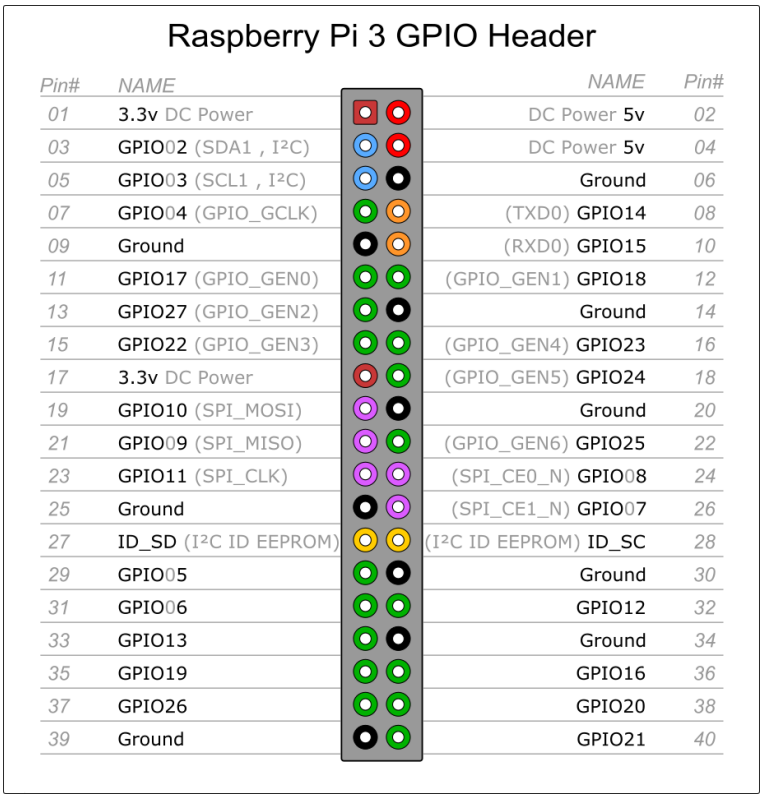


Εικόνα 5.1.1: Απεικόνιση του Raspberry Pi 3 Model B από πάνω.



Εικόνα 5.1.2: Απεικόνιση του Raspberry Pi 3 Model B από κάτω.

Ένα ισχυρό χαρακτηριστικό του Raspberry Pi 3 Model B είναι ακροδέκτες εισόδου – εξόδου που διαθέτει *GPIO (General Purpose Input/Output)* που δίνει την δυνατότητα να συνδεθούν διάφορα εξαρτήματα, όπως αισθητήρες, κινητήρες, διακόπτες, δίοδοι φωτοεκπομπής (*Light-Emitting Diodes, LEDs*). Διαθέτει 40 ακροδέκτες από τους οποίους 2 ακροδέκτες παρέχουν τάση 3.3V DC, 2 ακροδέκτες παρέχουν τάση 5V DC και 8 ακροδέκτες που είναι η γείωση. Οι υπόλοιποι ακροδέκτες είναι για γενική χρήση που μερικοί από αυτούς χρησιμοποιούνται για διαμόρφωση εύρους παλμού (*Pulse-Width Modulation, PWM*) και για συνδέσεις που απαιτούν *SPI (Serial Peripheral Interface)* για την επικοινωνία με άλλες πλακέτες ή περιφερειακές συσκευές. Ακόμη, για συνδέσεις που απαιτούν I^2C *(Inter-Integrated Circuit)* για την σύνδεση περιφερειακών συσκευών χαμηλής ταχύτητας και για συνδέσεις που απαιτούν *UART (Universal Asynchronous Receiver-Transmitter)* για την διασύνδεση αισθητήρων και άλλων συσκευών.



Σχήμα 5.1.1: Οι ακροδέκτες του Raspberry Pi 3 Model B.

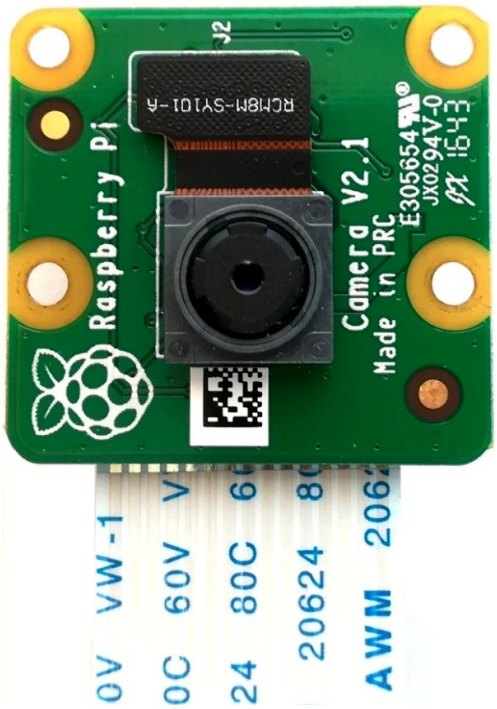
5.1.2 Raspberry Pi Camera V2

Για να μπορέσει το ρομποτικό όχημα να εκτελέσει το κομμάτι της Μηχανικής Όρασης χρησιμοποιείται ως όραση μια κάμερα που θα το βοηθήσει να εντοπίσει το αντικείμενο. Συγκεκριμένα, χρησιμοποιείται η κάμερα *Raspberry Pi Camera V2* λόγω της απόδοσης της, την εύχρηστη διασύνδεση με το Raspberry Pi και τις μικρές της διαστάσεις.

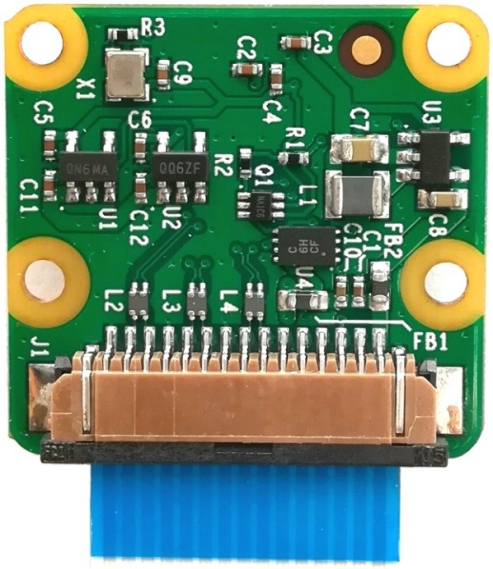
Η Raspberry Pi Camera V2 κυκλοφόρησε τον Απρίλιο του 2016 από την *Raspberry Pi Foundation* και διαθέτει τον αισθητήρα εικόνας Sony IMX219 8-megapixel CMOS. Έχει διαστάσεις 25 × 24 × 9 mm και ζυγίζει 3g. Η κάμερα μπορεί να χρησιμοποιηθεί για την λήψη βίντεο υψηλής ευκρίνειας και για την λήψη φωτογραφιών. Υποστηρίζει την λήψη βίντεο σε 1080p30, 720p60, VGA60/90 και για την σύνδεση με το Raspberry Pi στην θύρα CSI, χρησιμοποιείται καλωδιοταινία 15 κατευθύνσεων.

Πίνακας 5.1.3: Τεχνικά Χαρακτηριστικά Raspberry Pi Camera V2

Feature	Detail
Image Sensor	• Sony IMX219
Sensor Type	• CMOS
Resolution	• 8 Megapixels
Sensor Resolution	• 3280 × 2464 pixels
Sensor Image Area	• 3.68 × 2.76 mm (4.6 mm diagonal)
Pixel Size	• 1.12 µm × 1.12 µm
Optical Size	• 1/4"
Focal Length	• 3.04 mm
Horizontal Field of View	• 62.2 degrees
Vertical Field of View	• 48.8 degrees
Focal Ratio (F – Stop)	• 2.0
Video Modes	• 1080p30 • 720p60 • 640 × 480p60/90
Maximum frame rate	• 90fps
Linux Integration	• V4L2 driver available
C programming API	• OpenMax IL and others available
Connection to Raspberry Pi CSI port	• 15-pin ribbon cable
Automatic Control Functions	• Automatic exposure control (*AEC*) • Automatic white balance (*AWB*) • Automatic black level calibration (*ABLC*) • Automatic 50/60 Hz luminance detection
Size	• 25 × 24 × 9 mm
Weight	• 3g



Εικόνα 5.1.3: Εμπρόσθια απεικόνιση της Raspberry Pi Camera V2.



Εικόνα 5.1.4: Πίσω απεικόνιση της Raspberry Pi Camera V2.

5.1.3 VL53L0X Time-of-Flight (ToF) Distance Sensor

Για να μπορέσει το ρομποτικό όχημα να κινηθεί στο περιβάλλον του, διαθέτει έναν αισθητήρα απόστασης που του δίνει την δυνατότητα να αποφεύγει εμπόδια που μπορεί να συναντήσει στο δρόμο του. Για να τα αποφύγει, μετριέται η απόσταση μεταξύ του αισθητήρα και του αντικειμένου και ανάλογα με την μέτρηση αποφασίζει την κίνησή του.

Συγκεκριμένα, ο αισθητήρας απόστασης που χρησιμοποιείται είναι ο *VL53L0X* από την *STMicroelectronics* ο οποίος έχει διαστάσεις 4.4 × 2.4 × 1.0 mm και παρέχει ακριβή μέτρηση της απόστασης ανεξάρτητα από την ανακλαστικότητα του αντικειμένου. Για την μέτρηση της απόστασης ο αισθητήρας διαθέτει υπέρυθρη δέσμη λέιζερ και τεχνολογία *Time-of-Flight (ToF)*, που μοιάζει σαν ένα μικρό *LiDAR (Light Detection And Ranging)*.

Εικόνα 5.1.5: Αισθητήρας VL53L0X.

Ο εκπομπός 940 nm *VCSEL (Vertical Cavity Surface-Emitting Laser)* του αισθητήρα μαζί με τα εσωτερικά φυσικά υπέρυθρα φίλτρα, του επιτρέπουν να επηρεάζεται λιγότερο στο φως του περιβάλλοντος και να έχει καλύτερη αντοχή σε οπτικές παρεμβολές, από την κάλυψη του αισθητήρα με προστατευτικό γυαλί για την χρήση του σε δύσκολα περιβάλλοντα. Η τεχνολογία Time-of-Flight χρησιμοποιείται για την άμεση μέτρηση της απόστασης με βάση τον χρόνο που χρειάζεται η δέσμη λέιζερ να επιστρέψει στον αισθητήρα, από την αντανάκλαση που προήλθε από το αντικείμενο. Για την επικοινωνία με το Raspberry Pi ο αισθητήρας διαθέτει διασύνδεση I^2C με δύο γραμμές, την γραμμή σειριακών δεδομένων (*SDA*) και την γραμμή σειριακού ρολογιού (*SCL*), αλλά και προγραμματιζόμενη διεύθυνση I^2C.

Πίνακας 5.1.4: Τεχνικά Χαρακτηριστικά VL53L0X

Feature	**Detail**
Operating Voltage	• 2.6 to 3.5 V
Operating Temperature	• -20 to 70°C
Size	• 4.40 × 2.40 × 1.00 mm
Package	• Optical LGA12
Infrared Emitter	• 940 nm • VCSEL driver

Photodetector	• SPAD (Single Photon Avalanche Diodes)
I2C	• Up to 400 kHz (FAST mode) serial bus Address: 0x52
Ranging Distance	• 30 mm to 2000 mm
Ranging Accuracy (Depending on ambient lighting and distance)	• ±3% to 12%
Ranging Profiles	• Default mode • High accuracy • Long range • High speed
Ranging Time	• 20 ms (High Speed) • 30 ms (Default mode) • 33 ms (Long range) • 200 ms (High accuracy)
FoV (Field of View)	• 25 degrees
Eye Safe	• Class 1 laser device compliant with latest standard IEC 60825-1:2014 - 3rd edition

Ο VL53L0X είναι τοποθετημένος σε πλακέτα που διαθέτει το απαραίτητο κύκλωμα για την λειτουργία του. Η πλακέτα έχει διαστάσεις 21.0 × 18.0 × 2.8 mm, ζυγίζει 1.3g και διαθέτει 7 ακροδέκτες.

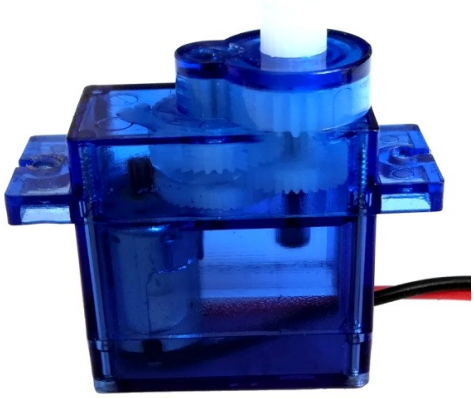
Εικόνα 5.1.6: Ο αισθητήρας VL53L0X τοποθετημένος στην πλακέτα.

Συγκεκριμένα, ο ακροδέκτης VIN χρησιμοποιείται για την τροφοδοσία της πλακέτας που δέχεται τάση εσόδου 3-5V DC και με έναν ρυθμιστή τάσης (*Voltage Regulator*) ρυθμίζεται η τάση για την ομαλή λειτουργία του VL53L0X. Ο ακροδέκτης GND είναι η γείωση, ενώ ο ακροδέκτης 2v3 παρέχει τάση με έξοδο 2.3V και ρεύμα μέχρι 100mA. Για την διασύνδεση I^2C υπάρχουν οι ακροδέκτες SDA και SCL. Ο GPIO είναι ακροδέκτης διακοπής που χρησιμοποιείται για να στείλει μια διακοπή όταν υπάρχει διαθέσιμη νέα μέτρηση και ο ακροδέκτης SHDN που χρησιμοποιείται για την απενεργοποίηση της λειτουργίας του αισθητήρα.

5.1.4 Ηλεκτρικοί Κινητήρες

Στο ρομποτικό όχημα εντοπίζονται δύο είδη ηλεκτρικών κινητήρων που μετατρέπουν την ηλεκτρική ενέργεια σε μηχανική ενέργεια. Ο κινητήρας συνεχούς ρεύματος (*DC Motor*) που χρησιμοποιείται για την κίνηση του ρομποτικού οχήματος και ο σερβοκινητήρας (*Servo Motor*) που χρησιμοποιείται στην περιστροφή του αισθητήρα VL53L0X για την μέτρηση της απόστασης σε τρείς διαφορετικές θέσεις.

Συγκεκριμένα, χρησιμοποιούνται δύο DC κινητήρες *FM90* από την *FEETECH* με κάθε κινητήρα να έχει διαστάσεις 23.2 × 12.5 × 22 mm και ταχύτητα περιστροφής που φτάνει τα 100 RPM (στροφές ανά λεπτό) με μέγιστη τάση τα 6V. Ο κινητήρας FM90 διαθέτει δύο ακροδέκτες έναν θετικό + (κόκκινο) και έναν αρνητικό − (μαύρο), αλλά και έναν μειωτήρα που είναι μια διάταξη οδοντωτών τροχών (γρανάζια) για την μετάδοση της κίνησης.



Εικόνα 5.1.7: Κινητήρας FM90.

Πίνακας 5.1.5: Τεχνικά Χαρακτηριστικά FM90

Feature	**Detail**	
Operating Voltage Range	• 4.8 V	• 6 V
Idle Current (at stopped)	• 5 mA	• 6 mA
No Load Speed	• 80 RPM	• 100 RPM
Running Current (at no load)	• 110 mA	• 130 mA
Maximum Stall Torque	• 1.3 kg.cm	• 1.5 kg.cm
Stall Current	• 700 mA	• 800 mA
Motor Type	• Metal brush motor	
Gear Type	• Plastic Gear (Nylon & POM)	
Bearing Type	• None	
Horn Gear Spline	• 20T (4.8 mm)	
Horn Type	• Plastic, POM	
Case	• Nylon & Fiberglass	
Connector Wire Length	• 100 ±5 mm	
Operating Temperature	• -15 to 70°C	
Size	• 23.2 × 12.5 × 22 mm	
Weight	• 9 ±0.2 g	

Στους σερβοκινητήρες ο άξονάς μετακινείται σε διάφορες θέσεις ανάλογα με το σήμα για την θέση που δέχεται και μπορεί να την διατηρήσει μέχρι να λάβει ένα νέο σήμα. Αποτελούνται από έναν DC κινητήρα, τον μειωτήρα, ένα ποτενσιόμετρο που είναι συνδεδεμένος ο άξονας και μια πλακέτα με το κύκλωμα ελέγχου που δέχεται τα σήματα PWM, αλλά και μετράει την τιμή του ποτενσιόμετρου.

Ο σερβοκινητήρας που χρησιμοποιείται στο ρομποτικό όχημα είναι ο *FS90MG* από την *FEETECH* με διαστάσεις 23.2 × 12.5 × 22 mm και ο άξονας περιστρέφεται από 0° έως 120°. Διαθέτει ένα καλώδιο για την τάση (κόκκινο), ένα καλώδιο για την γείωση (καφέ) και ένα καλώδιο σήματος (πορτοκαλί).

Πίνακας 5.1.6: Τεχνικά Χαρακτηριστικά FS90MG

Feature	**Detail**	
Operating Voltage Range	• 4.8 V	• 6 V
Idle Current (at stopped)	• 5 mA	• 6 mA
No Load Speed	• 0.12sec/60degree	• 0.10sec/60degree
Running Current (at no load)	• 120 mA	• 150 mA
Maximum Stall Torque	• 1.5 kg.cm	• 1.8 kg.cm
Stall Current	• 700 mA	• 800 mA
Control System	• Analog	
Command Signal	• PWM	
Operating Angle	• 120 degree	
Required Pulse	• 900 to 2100 us	
Neutral Position	• 1500 us	
Rotating Direction	• CW (1500 to 900 us) • CCW (1500 to 2100 us)	
Motor Type	• Metal brush motor	
Gear Type	• Metal Gear	
Bearing Type	• None	
Horn Gear Spline	• 20T (4.8 mm)	
Horn Type	• Plastic, POM	
Case	• ABS	
Connector Wire Length	• 200 ±5 mm	
Operating Temperature	• -15 to 70℃	
Size	• 23.2 × 12.5 × 22 mm	
Weight	• 11 ±0.2 g	

Εικόνα 5.1.8: Σερβοκινητήρας FS90MG.

5.1.5 Οδηγός DC Κινητήρων

Για τον έλεγχο της ταχύτητας και της κατεύθυνσης των DC κινητήρων χρησιμοποιείται ένας οδηγός (driver). Το ρεύμα που χρειάζεται ένας DC κινητήρας για τον έλεγχό του είναι υψηλό, σε σχέση με το ρεύμα που παρέχουν οι ακροδέκτες του Raspberry Pi. Επίσης, ο οδηγός δέχεται ένα σήμα χαμηλού ρεύματος και το μετατρέπει σε ένα σήμα υψηλού ρεύματος. Συγκεκριμένα, για τον έλεγχο χρησιμοποιείται πλακέτα οδήγησης DC κινητήρων με το μονολιθικό ολοκληρωμένο κύκλωμα (*IC*) *L298* από την *STMicroelectronics*.

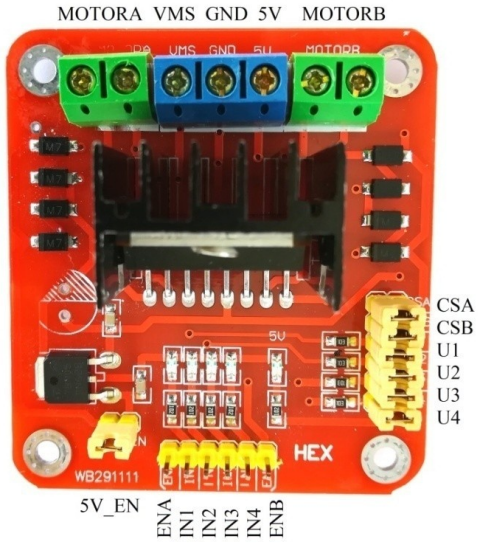
Το L298 είναι μια διπλή πλήρης γέφυρα οδήγησης που χρησιμοποιείται για την οδήγηση επαγωγικών φορτίων, όπως ηλεκτρονόμοι (ρελέ), πηνία, DC και βηματικοί κινητήρες. Το μέγιστο ρεύμα του L298 για κάθε κανάλι είναι 2A ή συνολικά 4A, ενώ η μέγιστη τάση λειτουργίας του φτάνει τα 46V και μπορεί να ελέγξει ξεχωριστά την ταχύτητα, αλλά και την κατεύθυνση κάθε DC κινητήρα.

Εικόνα 5.1.9: Πλακέτα οδήγησης με τον L298N.

Στην πλακέτα οδήγησης περιλαμβάνεται η έκδοση Multiwatt15 (*L298N*) που διαθέτει 15 ακροδέκτες. Η πλακέτα οδήγησης έχει διαστάσεις 55 × 60 × 30 mm και η τάση λειτουργίας της είναι 5-35V DC. Επίσης, διαθέτει 4 LEDs για την ένδειξη της κατεύθυνσης και 1 LED για την ένδειξη λειτουργίας.

Πίνακας 5.1.7: Τεχνικά Χαρακτηριστικά Πλακέτας Οδήγησης με το L298N

Feature	**Detail**
Driver	• ST L298N
Operating Voltage	• 5 to 35 V DC
Logic Voltage	• 5 V
Maximum Output Current (Each Channel)	• 2 A
Logic Current	• 0 to 36 mA
Input Voltage (IN1, IN2, IN3, IN4)	• Low: -0.3 to 1.5 V • High: 2.3 to Logic Voltage
Enable Voltage (ENA, ENB)	• Low: -0.3 to 1.5 V • High: 2.3 to Logic Voltage
Maximum Power	• 25 W
Operating Temperature	• -25 to 130°C
Size	• 55 × 60 × 30 mm
Weight	• 33g



Εικόνα 5.1.10: Απεικόνιση πλακέτας οδήγησης με τον L298N.

Οι ακροδέκτες MOTORA και MOTORB είναι για την σύνδεση των DC κινητήρων, ενώ ο ακροδέκτης VMS είναι για την τροφοδοσία των DC κινητήρων. Ο ακροδέκτης GND είναι η γείωση και ο ακροδέκτης 5V που γίνεται έξοδος ή είσοδος ανάλογα με τον ακροδέκτη 5V_EN. Όταν ο ακροδέκτης 5V_EN είναι ενεργοποιημένος, τότε μέσω ρυθμιστή τάσης παρέχονται 5V για την τροφοδοσία του λογικού κυκλώματος της πλακέτας και για την έξοδο του ακροδέκτη 5V. Στην αντίθετη περίπτωση με τον ακροδέκτη 5V_EN να είναι απενεργοποιημένος, ο ακροδέκτης 5V χρησιμοποιείται ως είσοδος τάσης 5V για την τροφοδοσία του λογικού κυκλώματος της πλακέτας.

Ο ακροδέκτης ENA είναι για τον έλεγχο του ενός DC κινητήρα με τους ακροδέκτες IN1 και IN2 να χρησιμοποιούνται για την κατεύθυνση του. Αντίστοιχα για τον άλλο κινητήρα υπάρχουν οι ακροδέκτες ENB, IN3 και IN4. Οι έξι αυτοί ακροδέκτες συνδέονται στους ακροδέκτες του Raspberry Pi και γίνεται ο έλεγχος της ταχύτητας με PWM και ο έλεγχος της κατεύθυνσης. Στον Πίνακα 5.1.8 παρουσιάζεται ο έλεγχος του DC κινητήρα Α όπου παρόμοια συμβαίνει ο έλεγχος και για τον DC κινητήρα Β με τους ανάλογους ακροδέκτες του.

Οι ακροδέκτες U1, U2, U3 και U4 είναι αντιστάσεις pull-up των IN1, IN2, IN3 και IN4, ενώ ο ακροδέκτης CSA χρησιμοποιείται για την μέτρηση του ρεύματος στο κανάλι Α δηλαδή του DC κινητήρα. Αντίστοιχα για την μέτρηση του ρεύματος στον άλλο DC κινητήρα χρησιμοποιείται ο ακροδέκτης CSB.

Πίνακας 5.1.8: Έλεγχος DC κινητήρα Α

Είσοδος (H = HIGH, L = LOW, X = H ή L)		**Κατάσταση DC κινητήρα Α**
ENA = H	*IN1 = H, IN2 = L*	• Περιστροφή Εμπρός
	IN1 = L, IN2 = H	• Περιστροφή Πίσω
	IN1 = H, IN2 = H	• Σταματάει η Περιστροφή
	IN1 = L, IN2 = L	• Σταματάει η Περιστροφή
ENA = L	*IN1 = X, IN2 = X*	• Σταματάει η Περιστροφή και η Λειτουργία

5.1.6 Τροφοδοσία

Το ρομποτικό όχημα για να είναι αυτόνομο θα πρέπει να έχει και την κατάλληλη τροφοδοσία για να λειτουργήσει ομαλά και για μεγάλο χρονικό διάστημα. Συγκεκριμένα, επιλέχτηκαν επαναφορτιζόμενες μπαταρίες για την τροφοδοσία των DC κινητήρων και ένα Power Bank για την τροφοδοσία του Raspberry Pi.

Για τους DC κινητήρες επιλέχτηκαν 5 επαναφορτιζόμενες μπαταρίες AA από την *GP Batteries* με κάθε μπαταρία να έχει ονομαστική τάση 1.2V, χωρητικότητα 2600mAh και βάρος ~31g.

Πίνακας 5.1.9: Τεχνικά Χαρακτηριστικά Μπαταρίας AA

Feature	Detail
Rechargeable Battery	• GP ReCyko^{+}
Battery Type	• AA
Electrochemical System	• Nickel-Metal Hydride (NiMH)
Nominal Voltage	• 1.2 V
Rated Capacity	• 2600 mAh
Diameter	• 14.5 mm
Height	• 50.5 mm
Gross Weight	• ~31g

Εικόνα 5.1.11: Επαναφορτιζόμενες μπαταρίες AA.

Για την τροφοδοσία του Raspberry Pi επιλέχτηκε ένα *Power Bank* από την *VARTA* με χωρητικότητα 6000mAh που παρέχει μία έξοδο 5V DC 2.4A και άλλη μία έξοδο 5V DC 1.0A. Διαθέτει 4 LEDs για την ένδειξη φόρτισης και αποφόρτισης, αλλά και μία θύρα Micro USB 5V DC 0.5-2A για την φόρτιση.

Πίνακας 5.1.10: Τεχνικά Χαρακτηριστικά Power Bank

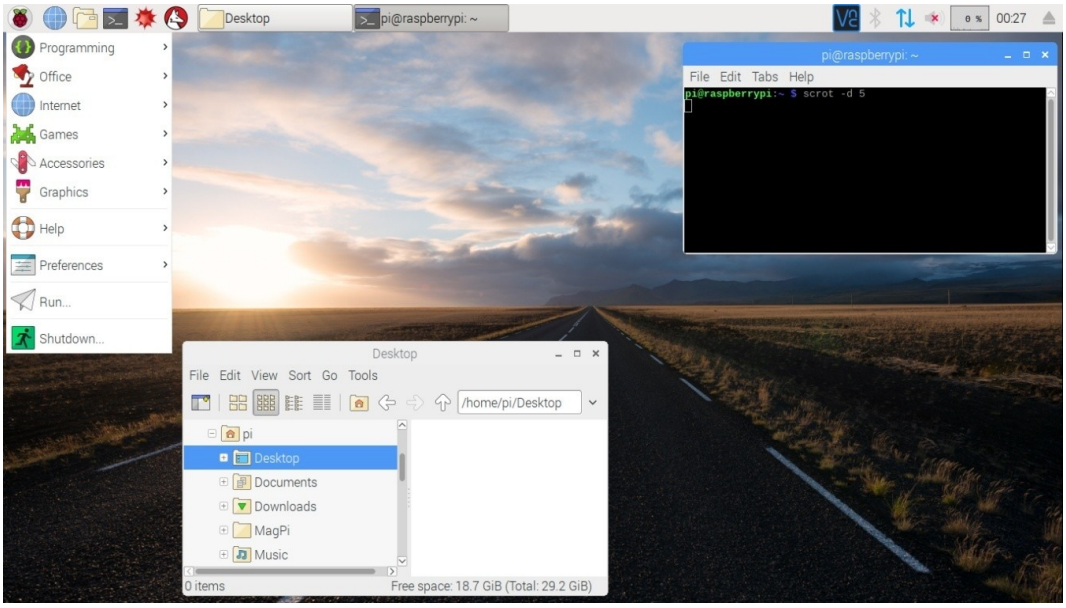
Feature	Detail
Capacity	• 6000 mAh
Input	• Micro USB: 5V DC 0.5-2A
Output	• USB I: 5V DC 1.0A • USB II: 5V DC 2.4A
Electrochemical System	• Lithium-ion (Li-ion)
LED	• 4 LEDs for charge and discharge indication • 1 LED flashlight
Size	• 64 × 96 × 22 mm
Weight	• 194g

Εικόνα 5.1.12: Power Bank.

5.2 Λογισμικό (Software)

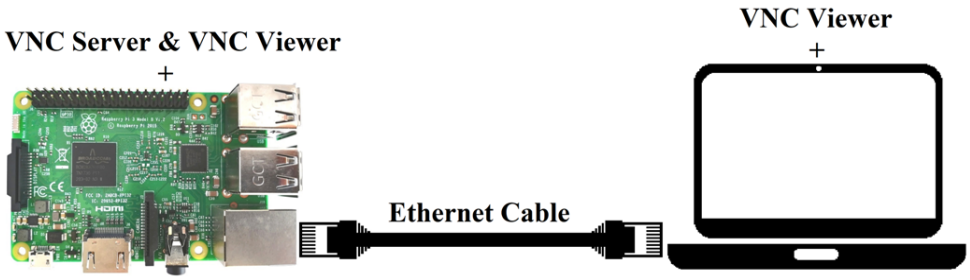
5.2.1 Λειτουργικό Σύστημα Raspbian Stretch

Το λειτουργικό σύστημα (*Operating System, OS*) που έχει επιλεγεί να εγκατασταθεί στο Raspberry Pi είναι το *Raspbian Stretch*. Το Raspbian είναι ένα δωρεάν λειτουργικό σύστημα που είναι βασισμένο στο Debian, μια διανομή Linux που έχει δεχτεί βελτιώσεις για το υλικό του Raspberry Pi. Το Raspbian διαθέτει πάνω από 35000 πακέτα και βρίσκεται σε διαρκή βελτίωση της σταθερότητας και της απόδοσης για όσο το δυνατόν περισσότερων πακέτων του Debian. Στο Raspbian υπάρχει προεγκατεστημένο πλήθος λογισμικού, όπως για παράδειγμα η Python, η Java και το LibreOffice.



Εικόνα 5.2.1: Το γραφικό περιβάλλον στο Raspbian.

Η εγκατάσταση του λειτουργικού συστήματος Raspbian Stretch γίνεται σε μια κάρτα μνήμης Micro SD χωρητικότητας 32 GB που τοποθετείται στο Raspberry Pi. Ο έλεγχος και η εργασία στο γραφικό περιβάλλον του Raspberry Pi γίνεται μέσω υπολογιστή χρησιμοποιώντας για την διασύνδεση ένα καλώδιο Ethernet και το VNC Connect. Το VNC Connect αποτελείται από το λογισμικό *VNC Viewer* και το λογισμικό *VNC Server*. Στο Raspberry Pi έχουν εγκατασταθεί και τα δύο λογισμικά, ενώ στον υπολογιστή έχει εγκατασταθεί το VNC Viewer.



Σχήμα 5.2.1: Διασύνδεση Raspberry Pi με υπολογιστή.

Συγκεκριμένα, για την διασύνδεση γίνονται κάποιες ρυθμίσεις στην κάρτα μνήμης που περιέχει το εγκατεστημένο λειτουργικό σύστημα Raspbian Stretch και τοποθετείται πάλι στο Raspberry Pi. Στην συνέχεια, συνδέεται το Raspberry Pi με τον υπολογιστή μέσω καλωδίου Ethernet και γίνονται κάποιες άλλες ρυθμίσεις στο Raspberry Pi μέσω του λογισμικού PuTTY που έχει εγκατασταθεί στον υπολογιστή. Με το τέλος όλων αυτών των ρυθμίσεων μπορεί να γίνει ο έλεγχος και η εργασία στο γραφικό περιβάλλον του Raspberry Pi από τον υπολογιστή διαμέσου του καλωδίου Ethernet και του VNC Connect.

5.2.2 Python

Η *Python* είναι μια διερμηνευόμενη, δυναμική, διαδραστική, αντικειμενοστραφής και υψηλού επιπέδου γλώσσα προγραμματισμού. Είναι εύκολη στην χρήση της, καθώς το συντακτικό της επιτρέπει να γίνεται εύκολα η ανάγνωση και η συγγραφή των προγραμμάτων με λιγότερες γραμμές κώδικα σε σχέση με την C++ και την Java.

Η χρήση της είναι συμβατή σε διάφορα λειτουργικά συστήματα, όπως Windows, Linux, Unix και Mac OS X. Ο μεγάλος αριθμός από βιβλιοθήκες που περιλαμβάνει, καθώς και οι εξωτερικές βιβλιοθήκες με τις οποίες είναι συμβατή την καθιστούν διάσημη επιλογή για διάφορες εφαρμογές στην επιστήμη και την μηχανική.

Για την συγγραφή του κώδικα και την δημιουργία εφαρμογών, η Python παρέχει ένα περιβάλλον ανάπτυξης που ονομάζεται IDLE. Στο περιβάλλον αυτό μπορεί να γίνει ο έλεγχος του κώδικα, η επεξεργασία, η εκτέλεση και η αποθήκευση των προγραμμάτων.

5.2.3 OpenCV

Η *OpenCV (Open Source Computer Vision)* είναι μια βελτιστοποιημένη βιβλιοθήκη που δίνει έμφαση στην δημιουργία εφαρμογών σε πραγματικό χρόνο. Η χρήση της εντοπίζεται σε διάφορες μη εμπορικές και εμπορικές εφαρμογές, όπως για παράδειγμα στην βιομηχανία, στην ρομποτική, σε συστήματα ασφαλείας και στην ιατρική απεικόνιση.

Η βιβλιοθήκη έχει πάνω από 2500 βελτιστοποιημένους αλγορίθμους που περιέχει ένα σύνολο από κλασσικούς και την τελευταία λέξη της τεχνολογίας αλγορίθμους στην Υπολογιστική Όραση και την Μηχανική Μάθηση.

Κάποιες από τις χρήσεις των αλγορίθμων είναι για την ανίχνευση και την αναγνώριση προσώπων, αλλά και αντικειμένων. Επίσης, για την παρακολούθηση κινούμενων αντικειμένων, την εξαγωγή τρισδιάστατων μοντέλων των αντικειμένων και την εξαγωγή της εικόνας μιας σκηνής σε υψηλή ανάλυση από την ένωση εικόνων. Ακόμη, για την αναγνώριση μιας σκηνής, την ακολούθηση των κινήσεων των ματιών και τον εντοπισμό όμοιων εικόνων από μια βάση δεδομένων με εικόνες.

Η OpenCV είναι γραμμένη στην C++ και είναι διαθέσιμη σε διάφορα λειτουργικά συστήματα όπως Windows, Linux και Mac OS. Επιπλέον, διαθέτει διεπαφές για τις γλώσσες προγραμματισμού C++, Python, Java και MATLAB.

5.2.4 Βιβλιοθήκες RPi.GPIO, pigpio, picamera και VL53L0X_rasp_python

Οι βιβλιοθήκες *RPi.GPIO*, *pigpio*, *picamera* και *VL53L0X_rasp_python* χρησιμοποιούνται στον προγραμματισμό του ρομποτικού οχήματος για τον έλεγχο των κινητήρων DC, του σερβοκινητήρα, της Raspberry Pi Camera V2 και του αισθητήρα VL53L0X.

Η βιβλιοθήκη RPi.GPIO περιέχεται στο Raspbian Stretch και χρησιμοποιείται για τον έλεγχο των ακροδεκτών GPIO του Raspberry Pi. Συγκεκριμένα, η βιβλιοθήκη χρησιμοποιείται για τον έλεγχο της κατεύθυνσης και της ταχύτητας με PWM των DC κινητήρων FM90.

Η βιβλιοθήκη pigpio (έκδοση 67) χρησιμοποιείται στο Raspberry Pi για τον έλεγχο των ακροδεκτών GPIO. Κάποια χαρακτηριστικά της βιβλιοθήκης είναι ότι χρησιμοποιείται για PWM, για τον έλεγχο του σήματος σε ένα σερβοκινητήρα και σε συνδέσεις, όπως σειριακές, I^2C και SPI. Συγκεκριμένα, η βιβλιοθήκη χρησιμοποιείται για τον έλεγχο της θέσης του σερβοκινητήρα FS90MG.

Η βιβλιοθήκη picamera (έκδοση 1.13) χρησιμοποιείται για τον έλεγχο της κάμερας Raspberry Pi. Κάποια χαρακτηριστικά του πακέτου είναι ότι μπορεί να ρυθμιστεί η ανάλυση της κάμερας, ο ρυθμός καρέ, η φωτεινότητα της κάμερας και να γίνει λήψη φωτογραφιών ή βίντεο.

Το πακέτο VL53L0X_rasp_python [78] παρέχει μια απλοποιημένη διασύνδεση του αισθητήρα VL53L0X στο Raspberry Pi με την χρήση της Python. Περιέχει έτοιμα παραδείγματα για την μέτρηση της απόστασης από έναν ή περισσότερους αισθητήρες VL53L0X. Για να μπορέσει να χρησιμοποιηθεί ο δίαυλος I^2C και από αλλά προγράμματα Python, η βιβλιοθήκη εφαρμόζει συγκεκριμένες λειτουργίες I^2C για τον αισθητήρα δια μέσου κλήσεων στην διεπαφή SMBus στην Python.

5.2.5 Βιβλιοθήκες NumPy, SciPy, Matplotlib

Σημαντικές είναι οι βιβλιοθήκες *NumPy*, *SciPy* και *Matplotlib* για την επεξεργασία των δεδομένων. Καθώς χρησιμοποιούνται στον υπολογισμό, στην εισαγωγή, την εξαγωγή και την απεικόνιση των δεδομένων.

Το NumPy αποτελεί το βασικό πακέτο για επιστημονικούς υπολογισμούς στην Python. Παρέχει την απαραίτητη λειτουργικότητα για πολυδιάστατους πίνακες, γραμμική άλγεβρα, μετασχηματισμό Fourier και για δυνατότητες τυχαίου αριθμού. Επίσης, διαθέτει εργαλεία για την ενσωμάτωση κώδικα Fortran και C/C++. Εκτός από την επιστημονική χρήση το πακέτο

μπορεί να χρησιμοποιηθεί ως ένα πολυδιάστατο δοχείο που περιέχει γενικά δεδομένα και να οριστούν αυθαίρετοι τύποι δεδομένων. Αυτό δίνει την δυνατότητα στο NumPy να ενσωματώνεται γρήγορα και χωρίς εμπόδια με ένα μεγάλο πλήθος βάσεων δεδομένων.

Η βιβλιοθήκη SciPy μπορεί να λειτουργήσει με τους πίνακες από το NumPy και περιέχει ένα εύρος λειτουργιών για υπολογισμούς που αφορούν τα μαθηματικά, την επιστήμη και την μηχανική στην Python. Παρέχει αποτελεσματικές αριθμητικές ρουτίνες για την αριθμητική ολοκλήρωση, βελτιστοποίηση και παρεμβολή. Επίσης, παρέχει αποτελεσματικές ρουτίνες για την γραμμική άλγεβρα, την στατιστική, την επεξεργασία σήματος και την επεξεργασία εικόνας.

Η βιβλιοθήκη Matplotlib χρησιμοποιείται για την απεικόνιση δεδομένων σε δισδιάστατα και τρισδιάστατα διαγράμματα στην Python. Η λειτουργία της βιβλιοθήκης μοιάζει σαν του MATLAB και μπορεί να λειτουργήσει με τους πίνακες από το Numpy. Μερικά παραδείγματα για την τύπο της απεικόνισης των δεδομένων που παρέχει η βιβλιοθήκη είναι ιστογράμματα, διαγράμματα σφαλμάτων, διαγράμματα διασποράς και προβολή εικόνας.

6. Υλοποίηση Αυτόνομου Ρομποτικού Οχήματος

Με την επιλογή του κατάλληλου υλικού και λογισμικού, το επόμενο βήμα είναι η υλοποίηση του ρομποτικού οχήματος. Πιο συγκεκριμένα, πραγματοποιείται η κατασκευή και η συνδεσμολογία του κυκλώματος, καθώς και ο προγραμματισμός του ρομποτικού οχήματος για την εκτέλεση της εργασίας του.

6.1 Κατασκευή

Για την κατασκευή του ρομποτικού οχήματος χρησιμοποιούνται συνολικά 178 εξαρτήματα. Αποτελείται από τα εξαρτήματα που περιγράφονται στην Ενότητα 5.1 Υλικό (Hardware)μαζί με κάποια επιπλέον εξαρτήματα, όπως για παράδειγμα σασί, τροχοί, μπαταριοθήκες, βίδες, αποστάτες, παξιμάδια, καλώδια. Το ρομποτικό όχημα έχει διαστάσεις 180 × 105 × 213 mm και ζυγίζει 725g.

Εικόνα 6.1.1: Το αυτόνομο ρομποτικό όχημα.

Συγκεκριμένα, το ρομποτικό όχημα διαθέτει μεταλλικό σασί με διαστάσεις 156 × 74 × 25 mm και δύο πλαστικούς τροχούς με ελαστικά σιλικόνης διαμέτρου 60 mm, που έχουν τοποθετηθεί στους δύο DC κινητήρες χρησιμοποιώντας βίδες. Στο σασί έχουν τοποθετηθεί δύο μπαταριοθήκες για τις 5 μπαταρίες AA με την χρήση αυτοκόλλητων διπλής όψεως και οι

δύο DC κινητήρες που έχουν στερεωθεί με βίδες και παξιμάδια. Επιπλέον, στο σασί έχει τοποθετηθεί μια βοηθητική ρόδα (ball caster) και μία επιφάνεια plexiglass χρησιμοποιώντας αποστάτες και βίδες.

Η επιφάνεια plexiglass έχει κοπεί κατάλληλα για να περνάνε καλώδια και για να τοποθετηθεί επάνω της η θήκη για το Raspberry Pi, η πλακέτα οδήγησης (L298N), αλλά και η βάση στήριξης με την Raspberry Pi Camera V2 χρησιμοποιώντας αποστάτες και βίδες. Ο σερβοκινητήρας έχει στερεωθεί με βίδες και παξιμάδια σε μια κατάλληλα διαμορφωμένη επιφάνεια plexiglass που ενώνεται με την πλακέτα οδήγησης (L298N) χρησιμοποιώντας αποστάτες και βίδες.

Για την πλακέτα με τον αισθητήρα VL53L0X και για την Raspberry Pi Camera V2 έχουν κατασκευαστεί δύο βάσεις στήριξης από φύλλο αλουμινίου και μονωτική ταινία. Η πλακέτα με τον αισθητήρα VL53L0X και η Raspberry Pi Camera V2 έχουν τοποθετηθεί στις βάσεις στήριξης χρησιμοποιώντας βίδες και αποστάτες. Η βάση στήριξης της πλακέτας με τον αισθητήρα VL53L0X έχει στερεωθεί στον βραχίονα του σερβοκινητήρα χρησιμοποιώντας βίδες και παξιμάδια.

Η θήκη για το Raspberry Pi έχει κατασκευαστεί από δύο επιφάνειες plexiglass που ενώνονται με αποστάτες και βίδες. Ακόμη, έχουν γίνει οι κατάλληλες διαμορφώσεις στην θήκη για να περνάνε καλώδια σύνδεσης και για να τοποθετηθεί ένας μικρός ανεμιστήρας 5V DC με διαστάσεις 40 × 40 × 10 mm. Στην θήκη τοποθετείται το Raspberry Pi χρησιμοποιώντας αποστάτες και βίδες, ενώ για τον μικρό ανεμιστήρα χρησιμοποιώντας βίδες και παξιμάδια.

Για την ψύξη του Raspberry έχουν τοποθετηθεί δύο ψύκτρες και χρησιμοποιείται ο μικρός ανεμιστήρας. Το Power Bank έχει τοποθετηθεί ανάμεσα στο σασί και στην επιφάνεια plexiglass. Επιπλέον, χρησιμοποιείται ένα καλώδιο USB σε Micro USB με μήκος 105 mm για την σύνδεση του Power Bank με το Raspberry Pi.

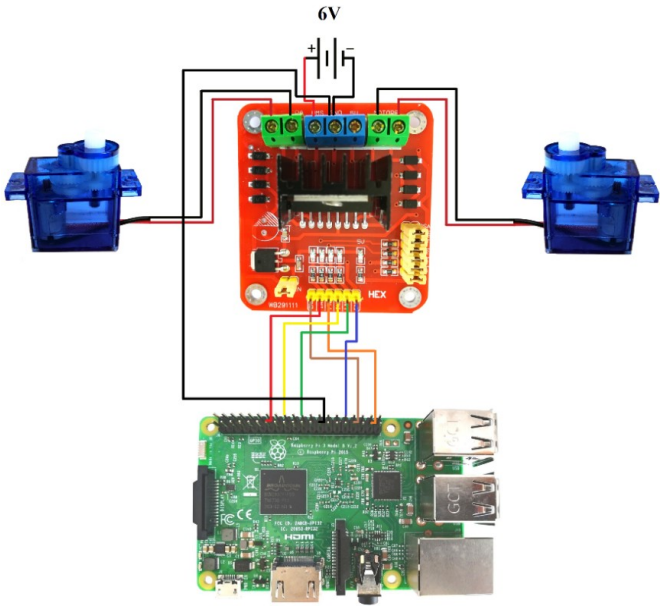
Στο Παράρτημα Β υπάρχει αναλυτικός πίνακας με τα εξαρτήματα που χρησιμοποιούνται για την υλοποίηση του ρομποτικού οχήματος και παρουσιάζεται φωτογραφικό υλικό από την συναρμολόγηση.

6.2 Συνδεσμολογία Κυκλώματος

Η συνδεσμολογία του κυκλώματος για το ρομποτικό όχημα παρουσιάζεται μέσα από πίνακες και σχήματα. Στο Παράρτημα Α υπάρχει το ολοκληρωμένο διάγραμμα κυκλώματος για το ρομποτικό όχημα.

Πίνακας 6.2.1: Συνδεσμολογία Πλακέτας Οδήγησης (L298N) με το Raspberry Pi, τους δύο DC Κινητήρες και την Τροφοδοσία από τις Μπαταρίες

Πλακέτα Οδήγησης (L298N)	**Raspberry Pi**		**Right DC Motor**		**Left DC Motor**		**6V from Batteries**
Pin Name	***Pin***	***Name***	***Cable***	***Cable***	***Cable***	***Cable***	***Polarity***
MOTORA			Positive (Red)	Negative (Black)			
VMS							(+)
GND	25	Ground					(−)
5V							
MOTORB					Negative (Black)	Positive (Red)	
ENA	33	GPIO13					
IN1	13	GPIO27					
IN2	37	GPIO26					
IN3	18	GPIO24					
IN4	22	GPIO25					
ENB	32	GPIO12					

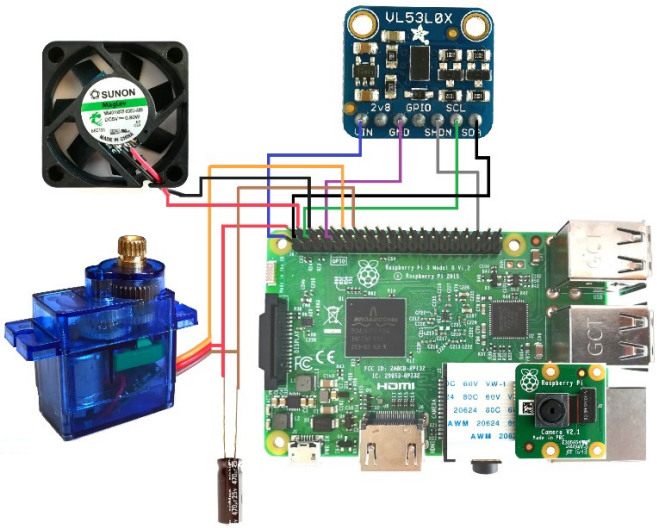


Σχήμα 6.2.1: Συνδεσμολογία πλακέτας οδήγησης (L298N) με το Raspberry Pi, τους δύο DC κινητήρες και την τροφοδοσία από τις μπαταρίες.

Πίνακας 6.2.2: Συνδεσμολογία Αισθητήρα VL53L0X με το Raspberry Pi

Αισθητήρας VL53L0X	**Raspberry Pi**	
Pin Name	***Pin***	***Name***
VIN	01	3.3V
2V8		
GND	09	Ground
GPIO		
SHDN	36	GPIO16
SCL	05	GPIO03
SDA	03	GPIO02

Στην συνδεσμολογία του σερβοκινητήρα με το Raspberry Pi το καλώδιο για την τάση (κόκκινο) συνδέεται στον ακροδέκτη Pin02 (5V), το καλώδιο για την γείωση (καφέ) συνδέεται με το Pin14 (Ground) και το καλώδιο σήματος (πορτοκαλί) συνδέεται με το Pin12 (GPIO18). Επίσης, στον σερβοκινητήρα έχει συνδεθεί ένας πυκνωτής 470μF 25V για την ομαλή λειτουργία του και η Raspberry Pi Camera V2 συνδέεται με καλωδιοταίνια 15 κατευθύνσεων στην θύρα CSI του Raspberry Pi. Στην συνδεσμολογία του μικρού ανεμιστήρα με το Raspberry Pi, το καλώδιο για την τάση (κόκκινο) συνδέεται στο Pin04 (5V) και το καλώδιο για την γείωση (μαύρο) συνδέεται στο Pin06 (Ground).



Σχήμα 6.2.2: Συνδεσμολογία του Raspberry Pi με τον αισθητήρα VL53L0X, τον σερβοκινητήρα, την κάμερα και τον μικρό ανεμιστήρα.

6.3 Προγραμματισμός

Ο προγραμματισμός του ρομποτικού οχήματος έχει γίνει στην Python 3.5.3, χρησιμοποιώντας τις βιβλιοθήκες που περιγράφονται στην Ενότητα 5.2 Λογισμικό (Software). Έχουν υλοποιηθεί συνολικά πέντε προγράμματα από τα οποία, τρία προγράμματα αφορούν την κίνηση με Μηχανική Μάθηση, ένα πρόγραμμα αφορά τον εντοπισμό του αντικειμένου με Μηχανική Όραση και ένα τελικό πρόγραμμα που συνδυάζει τα προηγούμενα τέσσερα προγράμματα.

Συγκεκριμένα, το ρομποτικό όχημα μαθαίνει να κινείται στο περιβάλλον του μέσω της Ενισχυτικής Μάθησης χρησιμοποιώντας τον αλγόριθμο Q-Learning. Τα τρία προγράμματα που έχουν υλοποιηθεί, αφορούν τους δύο DC κινητήρες, τον σερβοκινητήρα και τον αισθητήρα VL53L0X, αλλά και την εκμάθηση της κίνησης. Για τον εντοπισμό του αντικειμένου από το ρομποτικό όχημα χρησιμοποιείται η Raspberry Pi Camera V2, με το πρόγραμμα να περιέχει τον αλγόριθμος ORB, την μέθοδο FLANN και την χρήση της Homography. Το τελικό πρόγραμμα περιέχει την προσέγγιση του αντικειμένου σε συνδυασμό με τα προγράμματα για την Μηχανική Μάθηση και την Μηχανική Όραση με κάποιες τροποποιήσεις.

Στο Παράρτημα Α παρουσιάζονται όλα τα προγράμματα που χρησιμοποιούνται στο ρομποτικό όχημα.

6.3.1 Πρόγραμμα DC Κινητήρες και Actions

Στο πρόγραμμα γίνεται ο έλεγχος της κίνησης των δύο DC κινητήρων και δηλώνονται οι ενέργειες (actions) που μπορεί να εκτελέσει το ρομποτικό όχημα. Υπάρχουν συνολικά εννέα συναρτήσεις που κάθε μια αποτελεί και μια διαφορετική κίνηση. Οι συναρτήσεις με την σειρά που υπάρχουν στο πρόγραμμα είναι: FORWARD, BACKWARD, FRIGHT, FLEFT, BRIGHT, BLEFT, STOP, FORWARD2, STOP2.

Για τις ενέργειες του ρομποτικού οχήματος χρησιμοποιούνται οι συναρτήσεις, FORWARD, BACKWARD, FRIGHT, FLEFT, BRIGHT και BLEFT. Η συνάρτηση STOP χρησιμοποιείται όταν έχει τελειώσει η εκπαίδευση του ρομποτικού οχήματος και πρέπει να σταματήσουν οι κινητήρες. Οι κινητήρες απενεργοποιούνται κάνοντας LOW τους ακροδέκτες ENA και ENB. Οι συναρτήσεις FORWARD2 και STOP2 χρησιμοποιούνται στο Τελικό Πρόγραμμα για την προσέγγιση και το σταμάτημα του ρομποτικού οχήματος στο εντοπισμένο αντικείμενο.

Σε όλες τις συναρτήσεις η ταχύτητα ρυθμίζεται με μια τιμή που καθορίζει το PWM, ενώ για την κίνηση ρυθμίζονται τα IN1,IN2,IN3,IN4 σε True και False.

Πίνακας 6.3.1: Ανάλυση των Ενεργειών του Ρομποτικού Οχήματος

Ενέργεια (Action)	**Αποτέλεσμα Ενέργειας**
FORWARD	• Κίνηση Εμπρός
BACKWARD	• Κίνηση Πίσω
FRIGHT	• Κίνηση Εμπρός Δεξιά
FLEFT	• Κίνηση Εμπρός Αριστερά
BRIGHT	• Κίνηση Πίσω Δεξιά
BLEFT	• Κίνηση Πίσω Αριστερά

6.3.2 Πρόγραμμα Μέτρησης Απόστασης και States

Στο πρόγραμμα γίνεται η μέτρηση της απόστασης και δηλώνονται οι καταστάσεις (states) στις οποίες μπορεί να βρεθεί το ρομποτικό όχημα. Για την μέτρηση της απόστασης χρησιμοποιείται ο αισθητήρας VL53L0X μαζί με τον σερβοκινητήρα. Υπάρχουν συνολικά πέντε συναρτήσεις που η σειρά εμφάνισης τους στο πρόγραμμα είναι: centre, right, left, servo_stop, states.

Στην συνάρτηση centre ο σερβοκινητήρας περιστρέφει τον αισθητήρα VL53L0X στις 90° για την μέτρηση της απόστασης εμπρός. Η τιμή που επιστρέφεται από την centre είναι η μέτρηση της απόστασης. Στην συνάρτηση right ο σερβοκινητήρας περιστρέφει τον αισθητήρα VL53L0X στις 30° για την μέτρηση της απόστασης δεξιά. Η τιμή που επιστρέφεται από την right είναι η μέτρηση της απόστασης. Στην συνάρτηση left ο σερβοκινητήρας περιστρέφει τον αισθητήρα VL53L0X στις 150° για την μέτρηση της απόστασης αριστερά. Η τιμή που επιστρέφεται από την left είναι η μέτρηση της απόστασης. Η συνάρτηση servo_stop χρησιμοποιείται όταν έχει τελειώσει η εκπαίδευση του ρομποτικού οχήματος και πρέπει να σταματήσει η λειτουργία του σερβοκινητήρα και του αισθητήρα VL53L0X.

Στην συνάρτηση states περιέχονται εννέα καταστάσεις στις οποίες μπορεί να βρεθεί το ρομποτικό όχημα. Η συνάρτηση states καλεί πρώτα τις συναρτήσεις με την σειρά, right, centre, left και μετά ελέγχει ποια από τις εννέα καταστάσεις ισχύει. Εφόσον έχει βρεθεί ποια είναι η κατάσταση, η συνάρτηση επιστρέφει τον αριθμό της κατάστασης (state).

Πίνακας 6.3.2: Ανάλυση των Καταστάσεων του Ρομποτικού Οχήματος

Κατάσταση (State)	**Έλεγχος (R = right, C = centre, L = left, οι αριθμοί 5 και 20 είναι αποστάσεις σε εκατοστά (cm))**
0	• R >= 20 and C >= 20 and L >= 20
1	• R >= 20 and 5 < C < 20 and 5 < L < 20
2	• 5 < R < 20 and 5 < C < 20 and L >= 20
3	• R >= 20 and C >= 20 and 5 < L < 20
4	• 5 < R < 20 and C >= 20 and L >= 20
5	• 5 < R < 20 and 5 < C < 20 and 5 < L < 20
6	• R >= 20 and 5 < C < 20 and L >= 20
7	• 5 < R < 20 and C >= 20 and 5 < L < 20
8	• R <= 5 or C <= 5 or L <= 5

6.3.3 Πρόγραμμα Εκπαίδευσης Q-Learning

Στο πρόγραμμα γίνεται η εκμάθησης της κίνησης του ρομποτικού οχήματος στο περιβάλλον του χρησιμοποιώντας τα προηγούμενα δύο προγράμματα και τον αλγόριθμο Q-Learning. Οι συναρτήσεις που υπάρχουν στο πρόγραμμα είναι η move (action) και η R. Στην συνάρτηση move (action) ανάλογα με το action που είναι εκτελείται και η ανάλογη κίνηση.

Πίνακας 6.3.3: Ανάλυση Συνάρτησης move (action)

Action	Αποτέλεσμα Κίνησης
0	• FORWARD
1	• BACKWARD
2	• FRIGHT
3	• FLEFT
4	• BRIGHT
5	• BLEFT

Στην συνάρτηση R υπάρχει ένας 9×6 πίνακας με τις ανταμοιβές που μπορεί να πάρει το ρομποτικό όχημα. Στον πίνακα οι γραμμές αποτελούν τις καταστάσεις (states), ενώ οι στήλες αποτελούν τις ενέργειες (actions).

$$\begin{bmatrix} 10 & -10 & -10 & -10 & -10 & -10 \\ -10 & -5 & 1 & -10 & 0 & 0 \\ -10 & -5 & -10 & 1 & 0 & 0 \\ 10 & -5 & 0 & -10 & 0 & 0 \\ 10 & -5 & -10 & 0 & 0 & 0 \\ -10 & 0 & 0 & 0 & 0 & 0 \\ -10 & -5 & 0 & 0 & 0 & 0 \\ 10 & -5 & -1 & -1 & -1 & -1 \\ -100 & -100 & -100 & -100 & -100 & -100 \end{bmatrix}$$

Για παράδειγμα, στην πρώτη γραμμή το ρομποτικό όχημα θα πάρει ανταμοιβή 10 εάν κάνει την κίνηση FORWARD (εμπρός) και −10 για τις υπόλοιπες κινήσεις.

Στο πρόγραμμα φορτώνεται ο πίνακας qtable που έχει δημιουργηθεί από προηγούμενες εκπαιδεύσεις ή δημιουργείται ένας μηδενικός πίνακας εάν γίνεται πρώτη φορά εκπαίδευση.

Η epsilon είναι η παράμετρος ε για την εξερεύνηση και την εκμετάλλευση. Στην πρώτη εκπαίδευση η epsilon έχει τιμή 1 και σε κάθε εκπαίδευση μειώνεται κατά 0.25, ενώ στην τελευταία εκπαίδευση έχει τιμή 0.1. Το alpha είναι ο ρυθμός μάθησης με τιμή 0.5 και το g είναι ο συντελεστής μείωσης με τιμή 0.95. Η επιλογή της τιμής για το alpha και το g έγινε μέσα από πειράματα. Το ρομποτικό όχημα εκτελεί συνολικά πέντε εκπαιδεύσεις που κάθε μια διαρκεί 00:55:23 με τα episodes να είναι 100 και τα steps 20.

Στο τέλος κάθε εκπαίδευσης αποθηκεύεται ο τελικός πίνακας qtable (9×6) στον ίδιο φάκελο από τον οποίο και εκτελείται το πρόγραμμα. Ο πίνακας qtable προβάλει την διαδικασία της εκμάθησης του ρομποτικού οχήματος όπου κάθε τιμή q είναι το ζευγάρι κατάσταση (state) – ενέργεια (action). Το ρομποτικό όχημα χρησιμοποιεί τον πίνακα qtable για επιλέξει την καλύτερη ενέργεια με βάση την τιμή q.

Στην συνέχεια εμφανίζεται στην οθόνη, ο qtable, μήνυμα ότι έχει τελειώσει η εκπαίδευση και η συνολική ανταμοιβή (reward) που συγκέντρωσε το ρομποτικό όχημα. Επίσης, εμφανίζεται ο χρόνος εκτέλεσης της εκπαίδευσης και γίνεται απενεργοποίηση του σερβοκινητήρα και του αισθητήρα VL53L0X. Ακόμη, το πρόγραμμα αποθηκεύει σε κάθε episode την ανταμοιβή και τα τρακαρίσματα (crash) σε δύο διαφορετικούς πίνακες.

6.3.4 Πρόγραμμα Εντοπισμού Αντικειμένου

Στο πρόγραμμα γίνεται ο εντοπισμός του αντικειμένου χρησιμοποιώντας τον αλγόριθμο ORB, την μέθοδο FLANN και την Homography που περιέχονται στην βιβλιοθήκη OpenCV 4.0-alpha. Γίνονται οι κατάλληλες αρχικοποιήσεις για την κάμερα και φορτώνεται στο πρόγραμμα η φωτογραφία του αντικειμένου που μετατρέπεται από έγχρωμη σε ασπρόμαυρη (grayscale).

Γίνεται η ρύθμιση των παραμέτρων του ORB για να βρεθούν τα τοπικά χαρακτηριστικά της φωτογραφίας του αντικειμένου, καθώς και η ρύθμιση των παραμέτρων στον FLANN αλλά και στην Homography.

Σε έναν βρόχο επανάληψης χρησιμοποιείται η κάμερα για την μετάδοση συνεχούς ροής εικόνας που το κάθε καρέ (frame) μετατρέπεται από έγχρωμο σε ασπρόμαυρο. Ο ORB χρησιμοποιείται για να βρεθούν τα τοπικά χαρακτηριστικά από την συνεχόμενη ροή εικόνας, ενώ ο FLANN για την αντιστοίχηση των χαρακτηριστικών της φωτογραφία του αντικειμένου με την συνεχόμενη ροή εικόνας από την κάμερα. Εάν έχουν εντοπιστεί πάνω από 30 όμοια σημεία, τότε εφαρμόζεται η Homography και το εντοπισμένο αντικείμενο στην συνεχόμενη ροή εικόνας εμφανίζεται με ένα μπλε περίγραμμα γύρω του. Η συνεχόμενη ροή εικόνας με το εντοπισμένο αντικείμενο εμφανίζεται έγχρωμη, ενώ όταν δεν υπάρχει ταύτιση η συνεχόμενη ροή εικόνας είναι ασπρόμαυρη.

6.3.5 Τελικό Πρόγραμμα

Το τελικό πρόγραμμα χρησιμοποιείται για την πλοήγηση του ρομποτικού οχήματος στο περιβάλλον του και για τον εντοπισμό, αλλά και την προσέγγιση του αντικειμένου. Για την υλοποίηση του τελικού προγράμματος, χρησιμοποιείται το Πρόγραμμα Εντοπισμού και το Πρόγραμμα Εκπαίδευσης Q-Learning.

Το πρόγραμμα για τον εντοπισμό του αντικειμένου έχει χρησιμοποιηθεί ολόκληρο, ενώ στο πρόγραμμα για την εκπαίδευση της κίνησης του ρομποτικού οχήματος έχει αφαιρεθεί η διαδικασία της εκπαίδευσης. Μέσα στον βρόχο επανάληψης του Προγράμματος Εντοπισμού έχει προστεθεί από το Πρόγραμμα Εκπαίδευσης Q-Learning, ο έλεγχος της κατάστασης (state), η επιλογή και η εκτέλεση της ενέργειας (action) με βάση το πίνακα qtable της τελευταίας εκπαίδευσης. Οι τιμές του πίνακα qtable παραμένουν ως έχουν, καθώς δεν γίνεται κάποια εκπαίδευση για να ανανεώνονται.

Με τον εντοπισμό του αντικειμένου ο σερβοκινητήρας περιστρέφει τον αισθητήρα VL53L0X στις 90° για την μέτρηση της απόστασης μεταξύ ρομποτικού οχήματος και αντικειμένου. Αν η απόσταση είναι μεγαλύτερη από 15 cm, τότε η προσέγγιση γίνεται αργά και το ρομποτικό όχημα σταματάει μπροστά από το αντικείμενο.

Για την προσέγγιση και το σταμάτημα μπροστά από το αντικείμενο χρησιμοποιούνται οι συναρτήσεις FORWARD2 και STOP2. Στην συνάρτηση FORWARD2 ρυθμίζεται η κίνηση των δύο DC κινητήρων να είναι εμπρός και η ταχύτητα της προσέγγισης να είναι αργή μέσω της ρύθμισης του PWM. Η συνάρτηση STOP2 χρησιμοποιείται για το σταμάτημα μπροστά στο αντικείμενο με την κίνηση των δύο DC κινητήρων να σταματάει ρυθμίζοντας το PWM σε 0.

7. Πειράματα και Αποτελέσματα

Τα πειράματα που πραγματοποιούνται, έχουν σκοπό την συλλογή και την εξαγωγή δεδομένων τα οποία αξιολογούνται με κύριο στόχο την επίτευξη της εργασίας του αυτόνομου ρομποτικού οχήματος. Τα εξαγόμενα δεδομένα είναι σε μορφή πίνακα που μπορούν να χρησιμοποιηθούν για την δημιουργία διαγραμμάτων μέσω των βιβλιοθηκών Matplotlib και SciPy

Έχουν πραγματοποιηθεί πειράματα που αφορούν την επιλογή των κατάλληλων παραμέτρων για την εκμάθηση της κίνησης και τον εντοπισμού του αντικειμένου. Επίσης, πειράματα για την αξιολόγηση της εκπαίδευσης και της τελικής εργασίας του αυτόνομου ρομποτικού οχήματος.

Σχεδόν σε όλα τα πειράματα η τροφοδοσία που χρησιμοποιείται για τους DC κινητήρες προέρχεται από τροφοδοτικό 5V DC 2000mA. Δεν χρησιμοποιούνται οι επαναφορτιζόμενες μπαταρίες AA επειδή τα πειράματα είναι χρονοβόρα και υπάρχει πιθανότητα να μην ολοκληρωθούν, αλλά και να αλλοιωθούν τα αποτελέσματα λόγω της αποφόρτισης. Οι μπαταρίες χρησιμοποιούνται μόνο στην τελική εργασία του αυτόνομου ρομποτικού οχήματος.

7.1 Περιβάλλον Εργασίας

Για την διεξαγωγή των περισσότερων πειραμάτων έχει κατασκευαστεί μια πίστα που περιέχει εμπόδια και αντικείμενα η οποία αποτελεί και το περιβάλλον του αυτόνομου ρομποτικού οχήματος. Το συγκεκριμένο περιβάλλον χρησιμοποιείται στην εκμάθηση της κίνησης, στον εντοπισμό και στην προσέγγιση του αντικειμένου.

Το περιβάλλον περιέχει δύο εμπόδια και πέντε διαφορετικά αντικείμενα, όπου ένα από αυτά είναι το αντικείμενο εντοπισμού. Τα δύο εμπόδια είναι ένα βιβλίο και ένα κουτί, ενώ τα αντικείμενα είναι διαφορετικές πλακέτες τυπωμένων κυκλωμάτων που έχουν τοποθετηθεί σε διάφορα σημεία της πίστας. Οι διαστάσεις της πίστας είναι 1500 × 790 mm και το αντικείμενο (πλακέτα) προς εντοπισμό έχει τοποθετηθεί επάνω στο κουτί.

Εικόνα 7.1.1: Το περιβάλλον του αυτόνομου ρομποτικού οχήματος.

7.2 Παράμετροι alpha και g

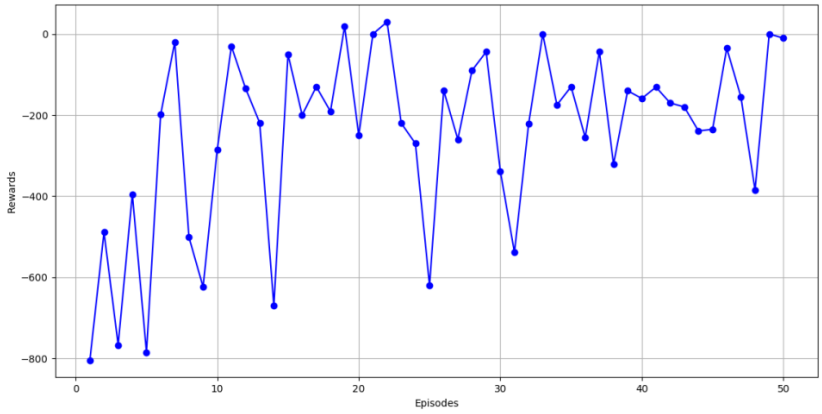
Στην διαδικασία της εκπαίδευσης του ρομποτικού οχήματος για την εκμάθηση της κίνησης έχουν οριστεί κάποιες τιμές για τις παραμέτρους alpha και g. Οι τιμές αυτές επιλέχτηκαν μετά από πολλά πειράματα χρησιμοποιώντας το ρομποτικό όχημα στο περιβάλλον του.

Συνολικά διεξάγονται δώδεκα πειράματα όπου στο ρομποτικό όχημα εκτελείται το Πρόγραμμα Εκπαίδευσης Q-Learning. Τα έξι πειράματα αφορούσαν την εύρεση της τιμής του alpha και τα υπόλοιπα έξι για την εύρεση της τιμής του g. Για τα πειράματα του alpha και g επιλέχτηκαν οι τιμές 0.1, 0.25, 0.5, 0.75, 0.9 και 0.95. Επιλέχτηκαν οι συγκεκριμένες τιμές, επειδή το alpha, ο ρυθμός μάθησης (α) παίρνει τιμές $0 < \alpha \leq 1$, ενώ το g, ο συντελεστής μείωσης (γ) παίρνει τιμές $0 \leq \gamma < 1$.

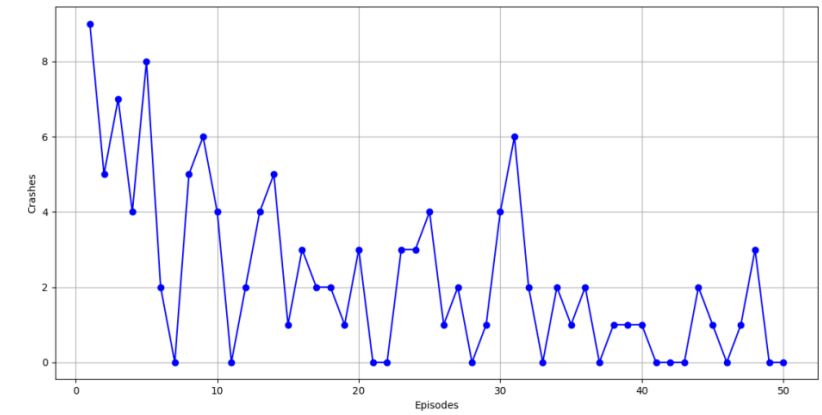
Σε όλα τα πειράματα η τιμή των παραμέτρων epsilon = 1.0, episodes = 50 και steps = 20 παραμένει σταθερή, όπως και η τιμή της παραμέτρου alpha ή g, ανάλογα με το πείραμα που διεξάγεται. Ο συνολικός χρόνος κάθε πειράματος είναι 00:27:42 όπου στο τέλος τα δεδομένα που εξάγονται είναι η ανταμοιβή και τα τρακαρίσματα για κάθε επεισόδιο σε δύο διαφορετικούς πίνακες, αλλά και ο πίνακας qtable.

7.2.1 Πειράματα Παραμέτρου alpha

Πείραμα $\alpha 1$ με παραμέτρους epsilon = 1.0, alpha = 0.1, g = 0.95, episodes = 50, steps = 20. Το τελικό αποτέλεσμα είναι το ρομποτικό όχημα να έχει συγκεντρώσει συνολική ανταμοιβή -12200.0 και να έχει τρακάρει 114 φορές.

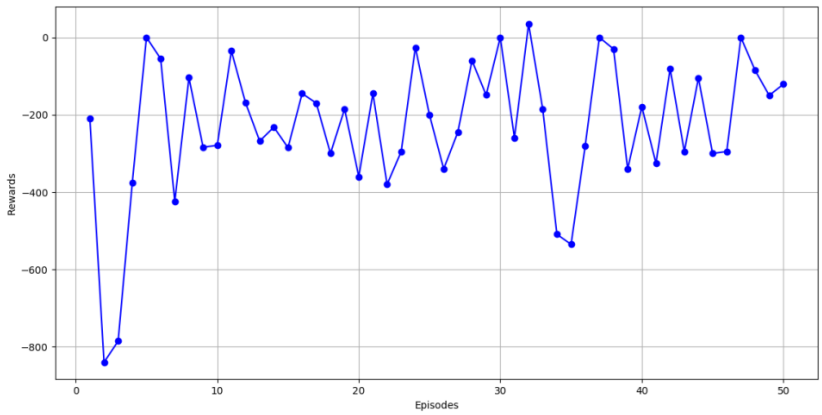


Σχήμα 7.2.1: Διάγραμμα ανταμοιβών για alpha = 0.1.

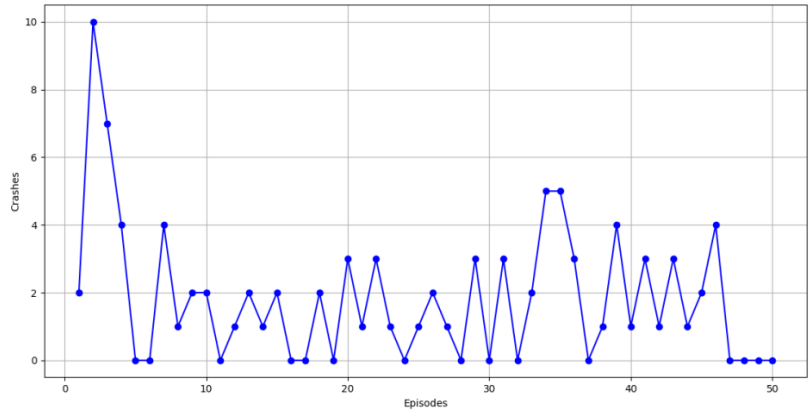


Σχήμα 7.2.2: Διάγραμμα τρακαρισμάτων για alpha = 0.1.

Πείραμα $\alpha 2$ με παραμέτρους epsilon = 1.0, alpha = 0.25, g = 0.95, episodes = 50, steps = 20. Το τελικό αποτέλεσμα είναι το ρομποτικό όχημα να έχει συγκεντρώσει συνολική ανταμοιβή -11388.0 και να έχει τρακάρει 93 φορές.

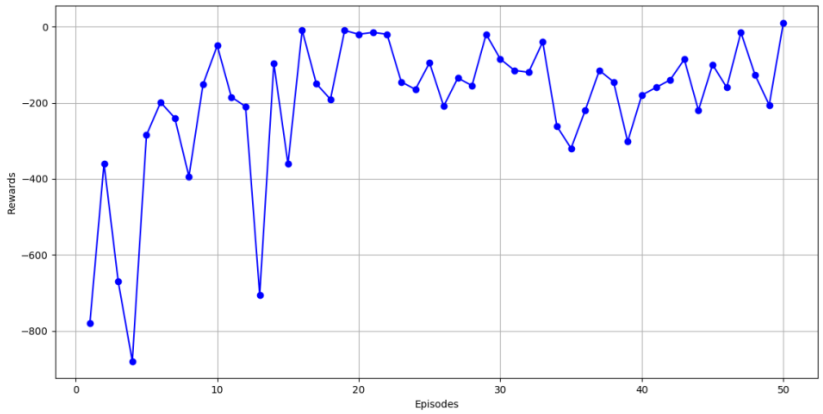


Σχήμα 7.2.3: Διάγραμμα ανταμοιβών για alpha = 0.25.

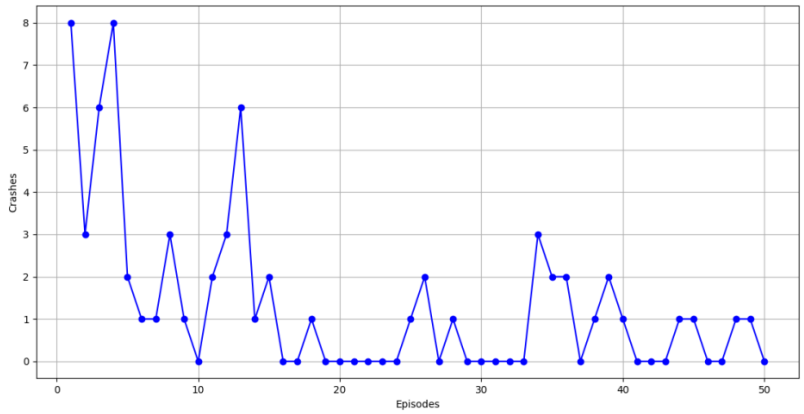


Σχήμα 7.2.4: Διάγραμμα τρακαρισμάτων για alpha = 0.25.

Πείραμα $\alpha 3$ με παραμέτρους epsilon = 1.0, alpha = 0.5, g = 0.95, episodes = 50, steps = 20. Το τελικό αποτέλεσμα είναι το ρομποτικό όχημα να έχει συγκεντρώσει συνολική ανταμοιβή -10013.0 και να έχει τρακάρει 67 φορές.

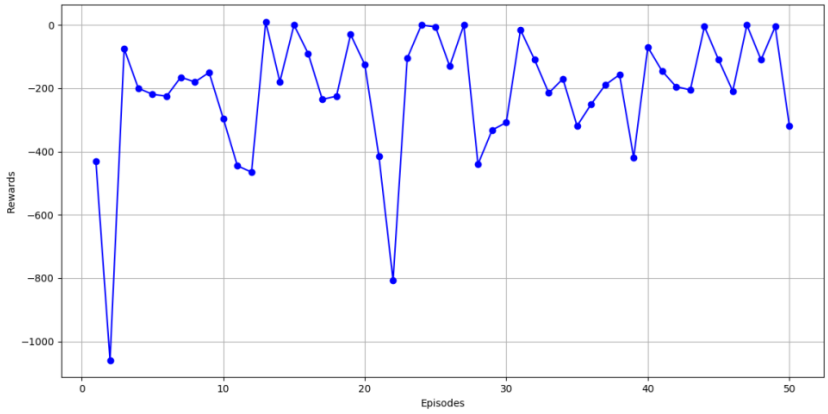


Σχήμα 7.2.5: Διάγραμμα ανταμοιβών για alpha = 0.5.

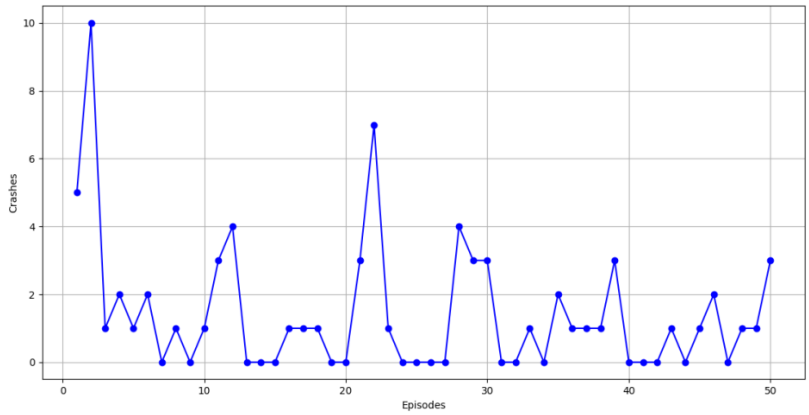


Σχήμα 7.2.6: Διάγραμμα τρακαρισμάτων για alpha = 0.5.

Πείραμα $\alpha 4$ με παραμέτρους epsilon = 1.0, alpha = 0.75, g = 0.95, episodes = 50, steps = 20. Το τελικό αποτέλεσμα είναι το ρομποτικό όχημα να έχει συγκεντρώσει συνολική ανταμοιβή -10543.0 και να έχει τρακάρει 72 φορές.

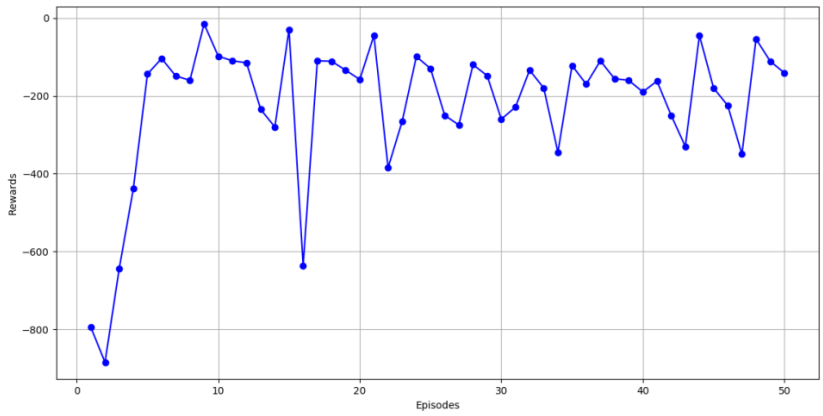


Σχήμα 7.2.7: Διάγραμμα ανταμοιβών για alpha = 0.75.

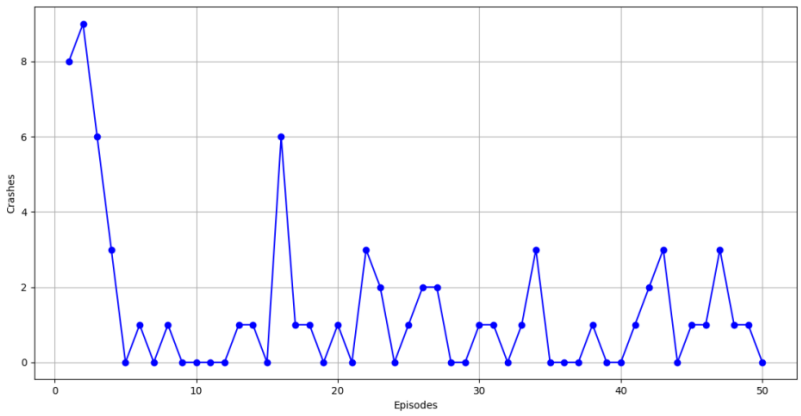


Σχήμα 7.2.8: Διάγραμμα τρακαρισμάτων για alpha = 0.75.

Πείραμα $\alpha 5$ με παραμέτρους epsilon = 1.0, alpha = 0.9, g = 0.95, episodes = 50, steps = 20. Το τελικό αποτέλεσμα είναι το ρομποτικό όχημα να έχει συγκεντρώσει συνολική ανταμοιβή -10974.0 και να έχει τρακάρει 69 φορές.

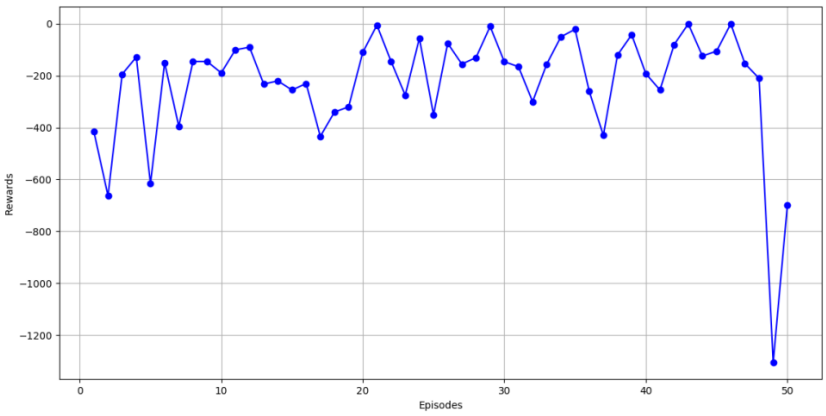


Σχήμα 7.2.9: Διάγραμμα ανταμοιβών για alpha = 0.9.

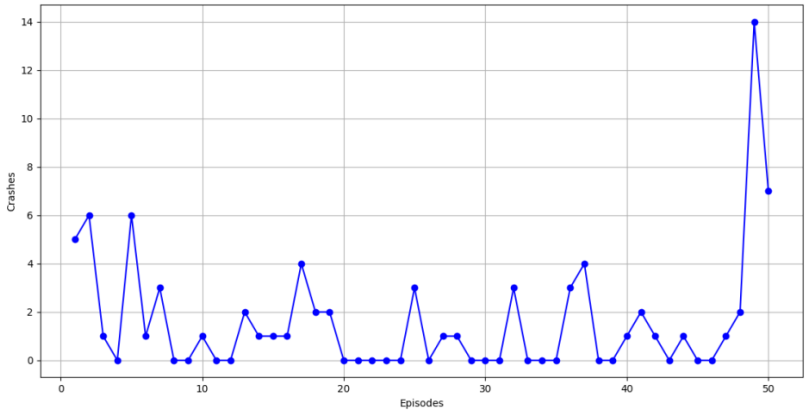


Σχήμα 7.2.10: Διάγραμμα τρακαρισμάτων για alpha = 0.9.

Πείραμα $\alpha 6$ με παραμέτρους epsilon = 1.0, alpha = 0.95, g = 0.95, episodes = 50, steps = 20. Το τελικό αποτέλεσμα είναι το ρομποτικό όχημα να έχει συγκεντρώσει συνολική ανταμοιβή -11385.0 και να έχει τρακάρει 80 φορές.



Σχήμα 7.2.11: Διάγραμμα ανταμοιβών για alpha = 0.95.

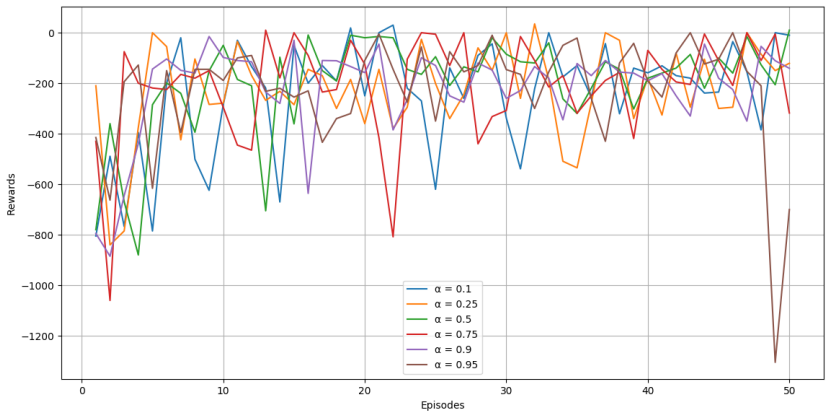


Σχήμα 7.2.12: Διάγραμμα τρακαρισμάτων για alpha = 0.95.

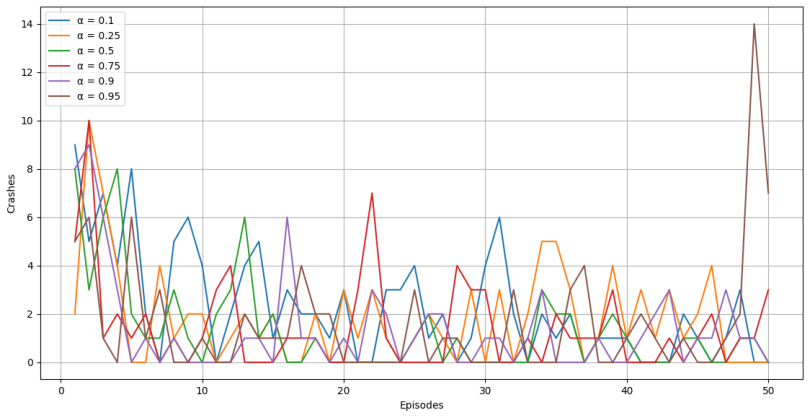
Από την σύγκριση των αποτελεσμάτων για κάθε τιμή alpha παρατηρείται ότι η τιμή 0.5 συγκεντρώνει την μεγαλύτερη ανταμοιβή και τα λιγότερα τρακαρίσματα. Επομένως, επιλέγεται η τιμή 0.5 για την παράμετρο alpha που θα χρησιμοποιηθεί για την εύρεση της τιμής της παραμέτρου g, αλλά και στην εκμάθηση της κίνησης του ρομποτικού οχήματος.

Πίνακας 7.2.1: Οι Ανταμοιβές και τα Τρακαρίσματα για κάθε Τιμή της Παραμέτρου alpha

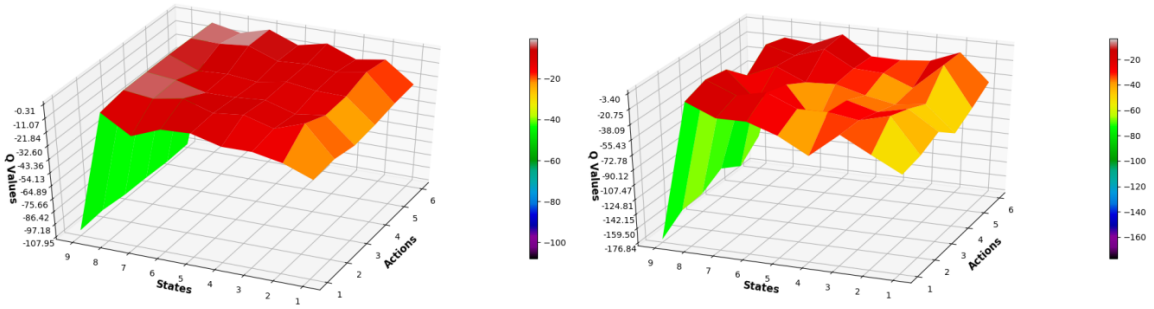
alpha	0.1	0.25	0.5	0.75	0.9	0.95
Rewards	-12200	-11388	-10013	-10543	-10974	-11385
Crashes	114	93	67	72	69	80



Σχήμα 7.2.13: Διάγραμμα ανταμοιβών για κάθε τιμή της παραμέτρου alpha.

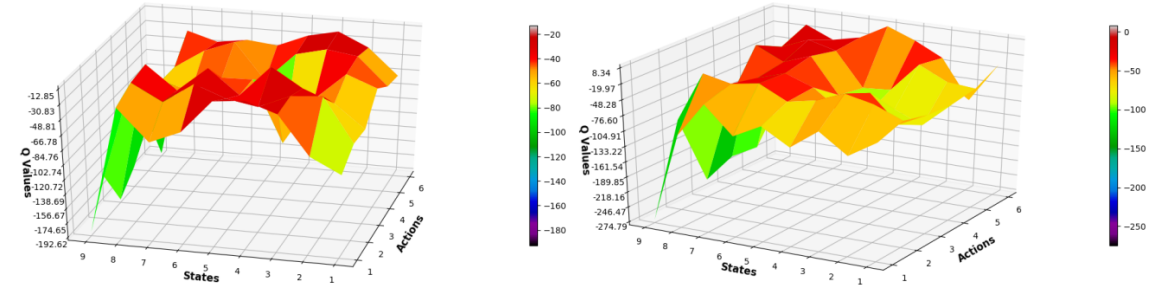


Σχήμα 7.2.14: Διάγραμμα τρακαρισμάτων για κάθε τιμή της παραμέτρου alpha.



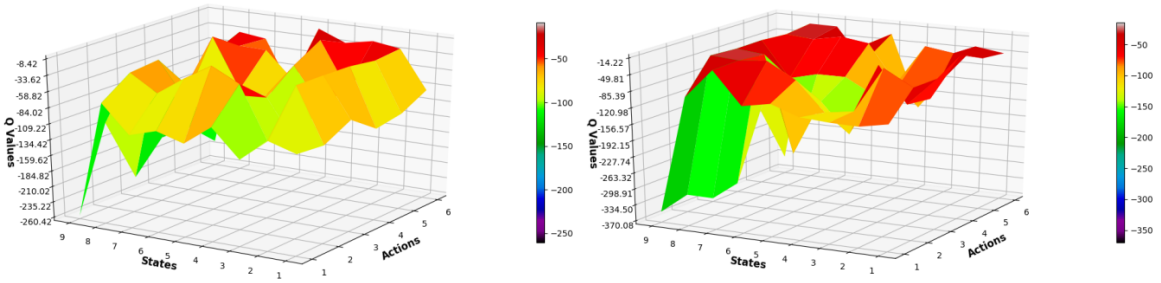
alpha = 0.1

alpha = 0.25



alpha = 0.5

alpha = 0.75



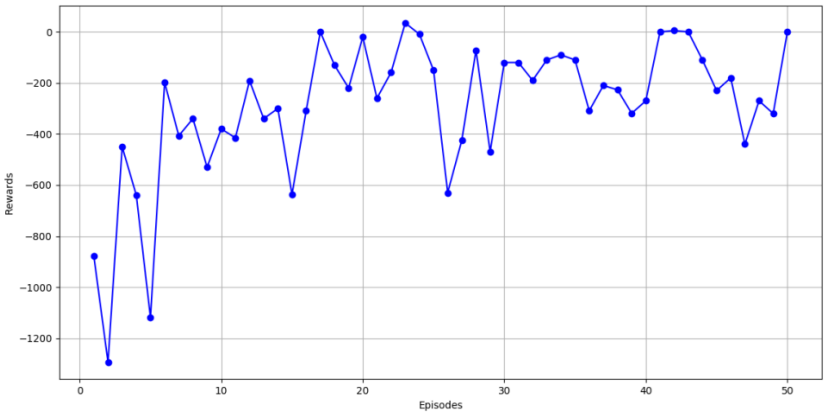
alpha = 0.9

alpha = 0.95

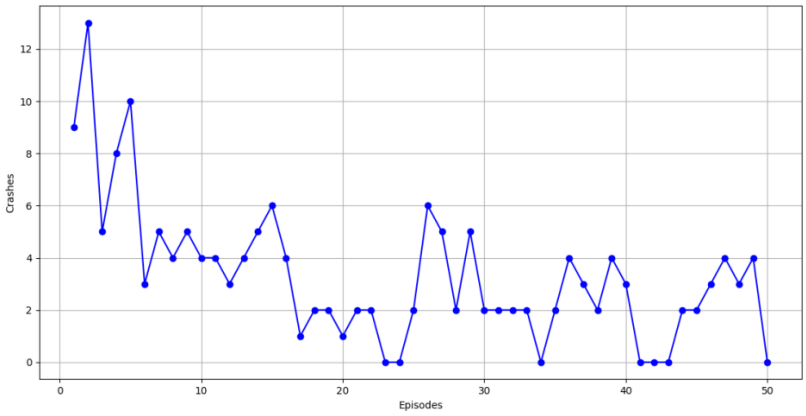
Σχήμα 7.2.15: Διαγράμματα τιμών q για κάθε τιμή της παραμέτρου alpha.

7.2.2 Πειράματα Παραμέτρου g

Πείραμα $\gamma 1$ με παραμέτρους epsilon = 1.0, alpha = 0.5, g = 0.1, episodes = 50, steps = 20. Το τελικό αποτέλεσμα είναι το ρομποτικό όχημα να έχει συγκεντρώσει συνολική ανταμοιβή -14557.0 και να έχει τρακάρει 166 φορές.

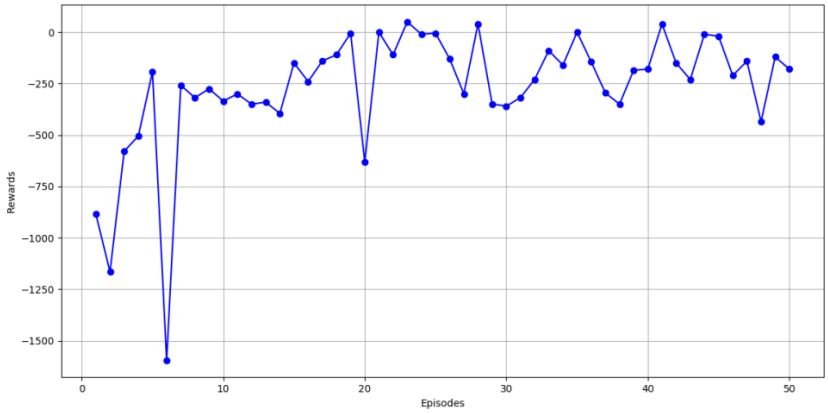


Σχήμα 7.2.16: Διάγραμμα ανταμοιβών για g = 0.1.

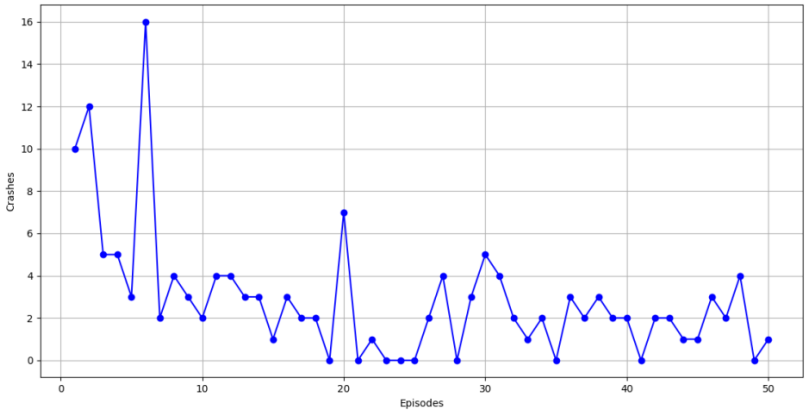


Σχήμα 7.2.17: Διάγραμμα τρακαρισμάτων για g = 0.1.

Πείραμα $\gamma 2$ με παραμέτρους epsilon = 1.0, alpha = 0.5, g = 0.25, episodes = 50, steps = 20. Το τελικό αποτέλεσμα είναι το ρομποτικό όχημα να έχει συγκεντρώσει συνολική ανταμοιβή -13352.0 και να έχει τρακάρει 143 φορές.

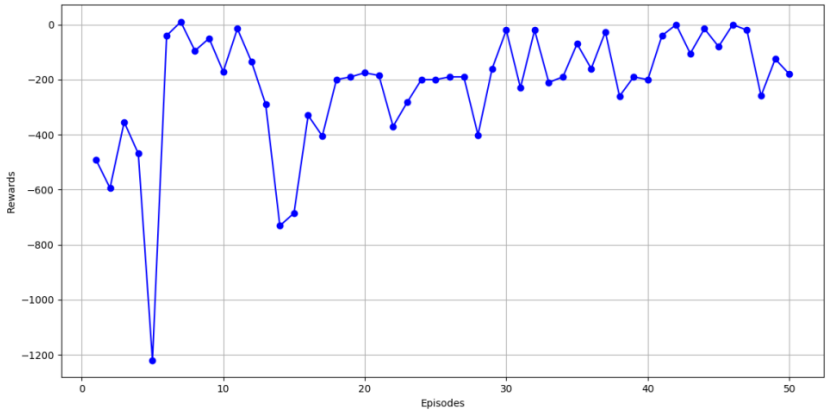


Σχήμα 7.2.18: Διάγραμμα ανταμοιβών για g = 0.25.

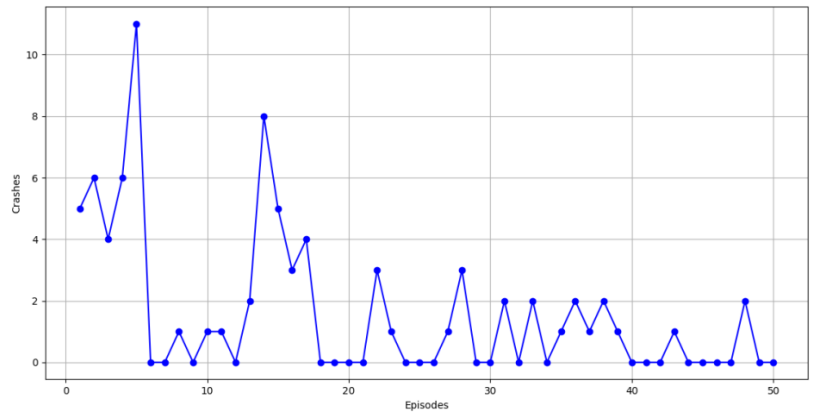


Σχήμα 7.2.19: Διάγραμμα τρακαρισμάτων για g = 0.25.

Πείραμα $\gamma 3$ με παραμέτρους epsilon = 1.0, alpha = 0.5, g = 0.5, episodes = 50, steps = 20. Το τελικό αποτέλεσμα είναι το ρομποτικό όχημα να έχει συγκεντρώσει συνολική ανταμοιβή -11207.0 και να έχει τρακάρει 79 φορές.

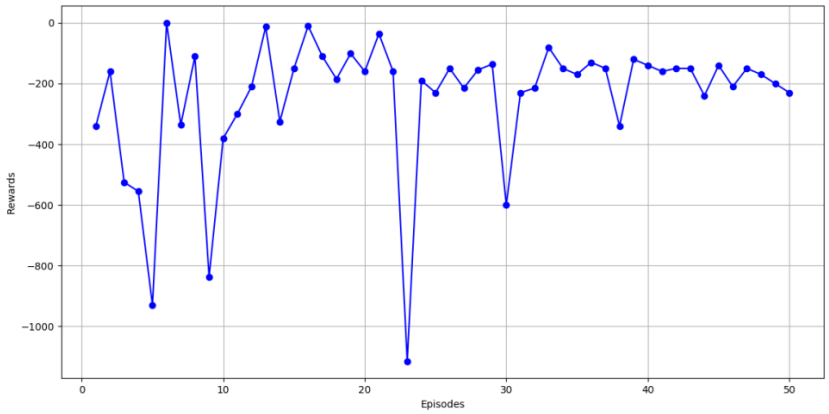


Σχήμα 7.2.20: Διάγραμμα ανταμοιβών για g = 0.5.

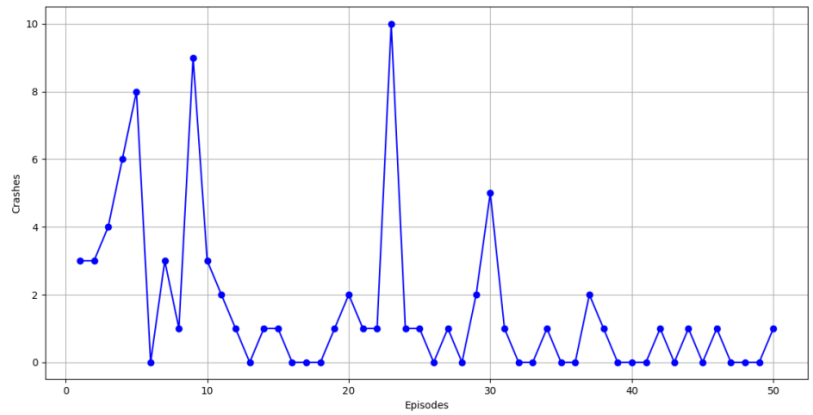


Σχήμα 7.2.21: Διάγραμμα τρακαρισμάτων για g = 0.5.

Πείραμα $\gamma 4$ με παραμέτρους epsilon = 1.0, alpha = 0.5, g = 0.75, episodes = 50, steps = 20. Το τελικό αποτέλεσμα είναι το ρομποτικό όχημα να έχει συγκεντρώσει συνολική ανταμοιβή -12250.0 και να έχει τρακάρει 79 φορές.

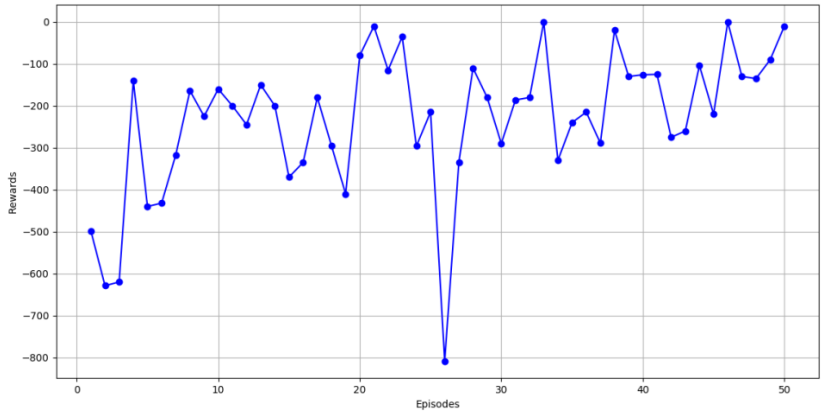


Σχήμα 7.2.22: Διάγραμμα ανταμοιβών για g = 0.75.

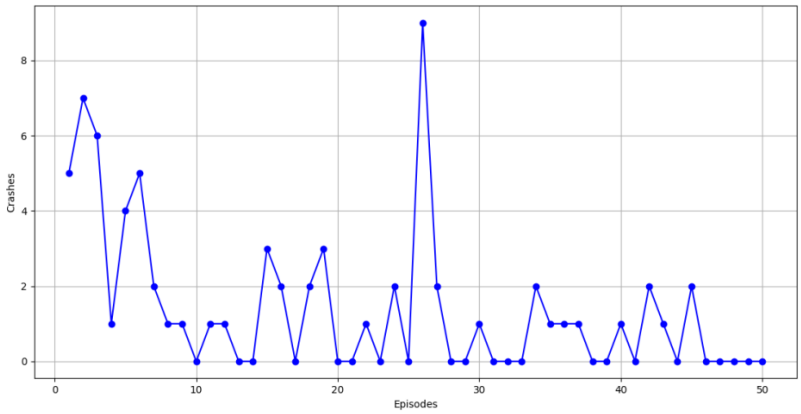


Σχήμα 7.2.23: Διάγραμμα τρακαρισμάτων για g = 0.75.

Πείραμα $\gamma 5$ με παραμέτρους epsilon = 1.0, alpha = 0.5, g = 0.9, episodes = 50, steps = 20. Το τελικό αποτέλεσμα είναι το ρομποτικό όχημα να έχει συγκεντρώσει συνολική ανταμοιβή -11550.0 και να έχει τρακάρει 70 φορές.

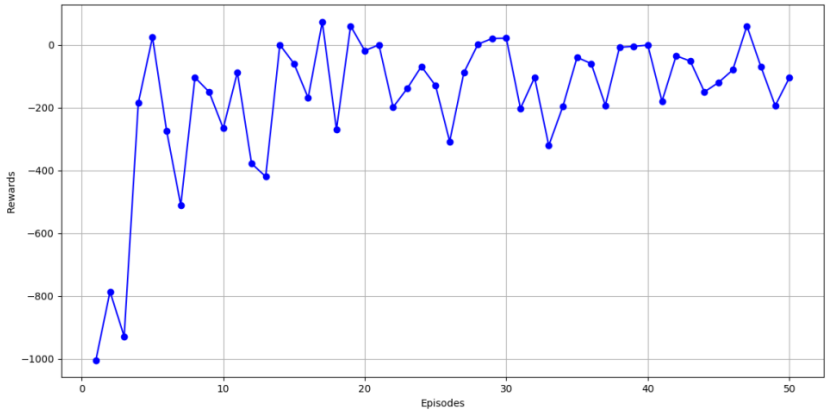


Σχήμα 7.2.24: Διάγραμμα ανταμοιβών για g = 0.9.

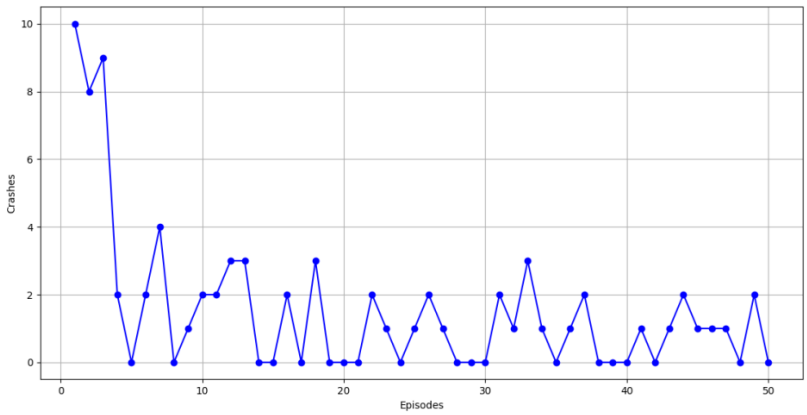


Σχήμα 7.2.25: Διάγραμμα τρακαρισμάτων για g = 0.9.

Πείραμα $\gamma 6$ με παραμέτρους epsilon = 1.0, alpha = 0.5, g = 0.95, episodes = 50, steps = 20. Το τελικό αποτέλεσμα είναι το ρομποτικό όχημα να έχει συγκεντρώσει συνολική ανταμοιβή -8406.0 και να έχει τρακάρει 77 φορές.



Σχήμα 7.2.26: Διάγραμμα ανταμοιβών για g = 0.95.

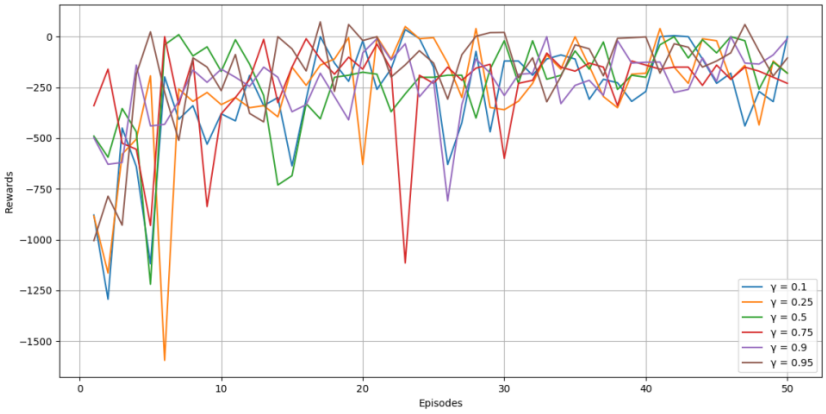


Σχήμα 7.2.27: Διάγραμμα τρακαρισμάτων για g = 0.95.

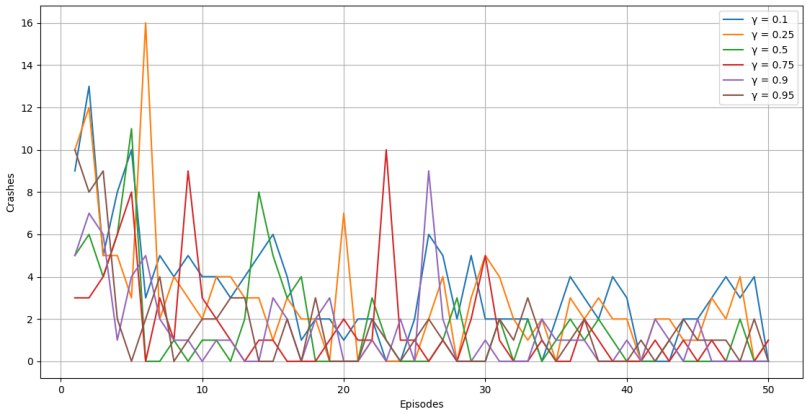
Από την σύγκριση των αποτελεσμάτων για κάθε τιμή g παρατηρείται ότι η τιμή 0.95 συγκεντρώνει την μεγαλύτερη ανταμοιβή και έχει σχετικά τα λιγότερα τρακαρίσματα. Επομένως, επιλέγεται η τιμή 0.95 για την παράμετρο g που θα χρησιμοποιηθεί στην εκμάθηση της κίνησης του ρομποτικού οχήματος.

Πίνακας 7.2.2: Οι Ανταμοιβές και τα Τρακαρίσματα για κάθε Τιμή της Παραμέτρου g

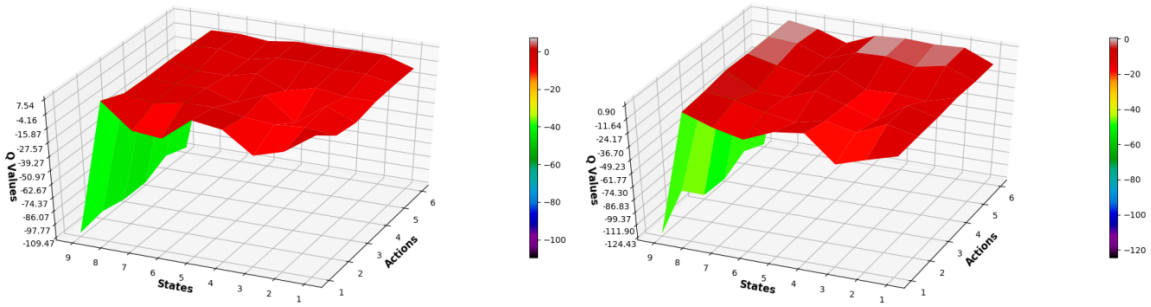
g	0.1	0.25	0.5	0.75	0.9	0.95
Rewards	-14557	-13352	-11207	-12250	-11550	-8406
Crashes	166	143	79	79	70	77



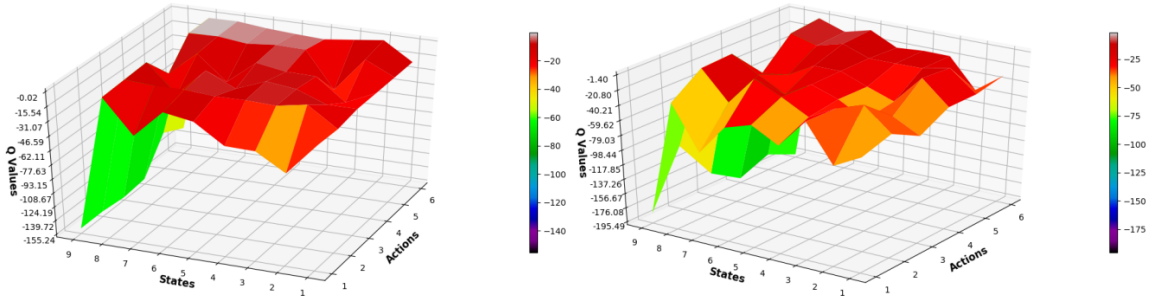
Σχήμα 7.2.28: Διάγραμμα ανταμοιβών για κάθε τιμή της παραμέτρου g.



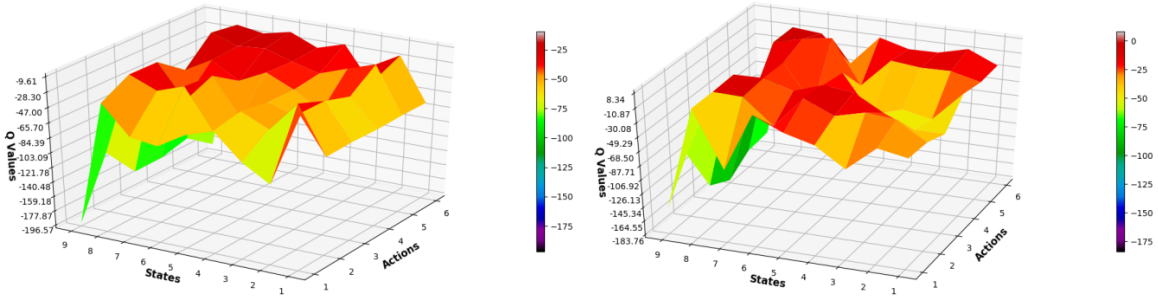
Σχήμα 7.2.29: Διάγραμμα τρακαρισμάτων για κάθε τιμή της παραμέτρου g.



g = 0.1 g = 0.25



g = 0.5 g = 0.75



g = 0.9 g = 0.95

Σχήμα 7.2.30: Διαγράμματα τιμών q για κάθε τιμή της παραμέτρου g.

7.3 Εκπαίδευση Αυτόνομου Ρομποτικού Οχήματος

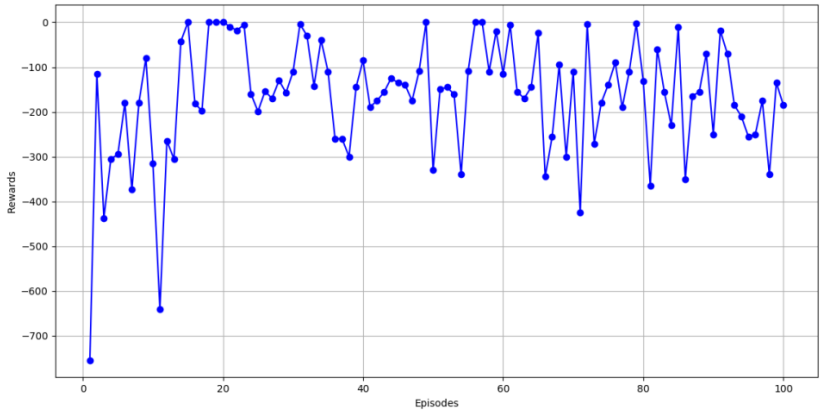
Στην διαδικασία της εκπαίδευσης του ρομποτικού οχήματος για την εκμάθηση της κίνησης, χρησιμοποιούνται οι παράμετροι alpha = 0.5 και g = 0.95 που βρέθηκαν από τα προηγούμενα πειράματα.

Συνολικά διεξάγονται πέντε πειράματα (εκπαιδεύσεις) όπου στο ρομποτικό όχημα εκτελείται το Πρόγραμμα Εκπαίδευσης Q-Learning. Σε κάθε εκπαίδευση το ρομποτικό όχημα μαθαίνει να κινείται στο περιβάλλον του χρησιμοποιώντας την εμπειρία που έχει αποκτήσει από προηγούμενες εκπαιδεύσεις και μειώνοντας κάθε φορά την τιμή της παραμέτρου epsilon. Η εμπειρία προέρχεται από τον πίνακα qtable που σε κάθε εκπαίδευση αλλάζει, ενώ η

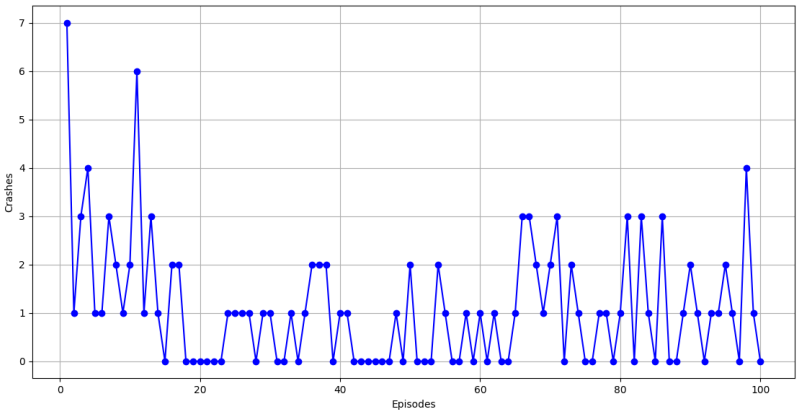
παράμετρος epsilon που μειώνεται σε κάθε εκπαίδευση παίρνει τις τιμές 1, 0.75, 0.5, 0.25 και 0.1.

Σε όλες τις εκπαιδεύσεις η τιμή των παραμέτρων alpha = 0.5, g = 0.95, episodes = 100 και steps = 20 παραμένει σταθερή. Ο συνολικός χρόνος κάθε πειράματος είναι 00:55:23 όπου στο τέλος τα δεδομένα που εξάγονται είναι η ανταμοιβή και τα τρακαρίσματα για κάθε επεισόδιο σε δύο διαφορετικούς πίνακες, αλλά και ο πίνακας qtable.

Πείραμα $T1$ με παραμέτρους epsilon = 1, alpha = 0.5, g = 0.95, episodes = 100, steps = 20. Το τελικό αποτέλεσμα είναι το ρομποτικό όχημα να έχει συγκεντρώσει συνολική ανταμοιβή -16619 και να έχει τρακάρει 110 φορές.

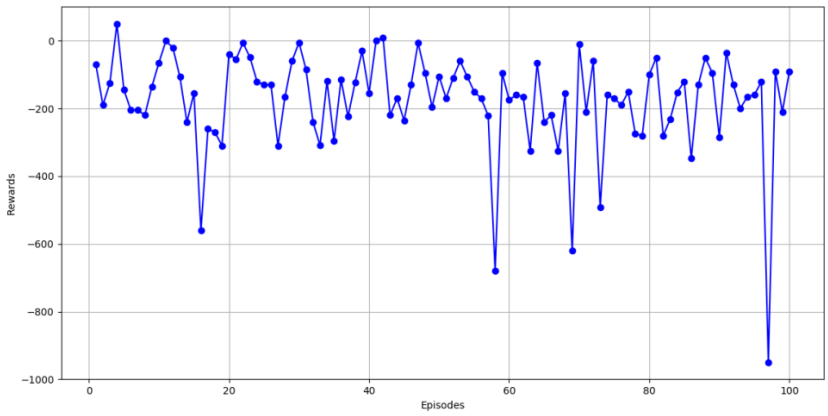


Σχήμα 7.3.1: Διάγραμμα ανταμοιβών πρώτης εκπαίδευσης.

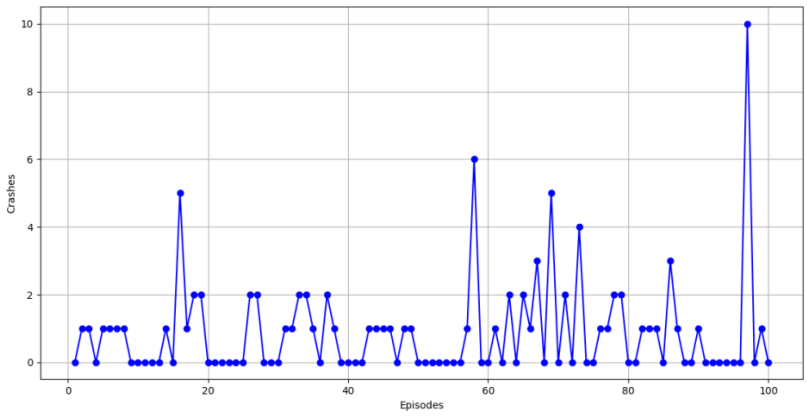


Σχήμα 7.3.2: Διάγραμμα τρακαρισμάτων πρώτης εκπαίδευσης.

Πείραμα $T2$ με παραμέτρους epsilon = 0.75, alpha = 0.5, g = 0.95, episodes = 100, steps = 20. Το τελικό αποτέλεσμα είναι το ρομποτικό όχημα να έχει συγκεντρώσει συνολική ανταμοιβή -17345 και να έχει τρακάρει 89 φορές.

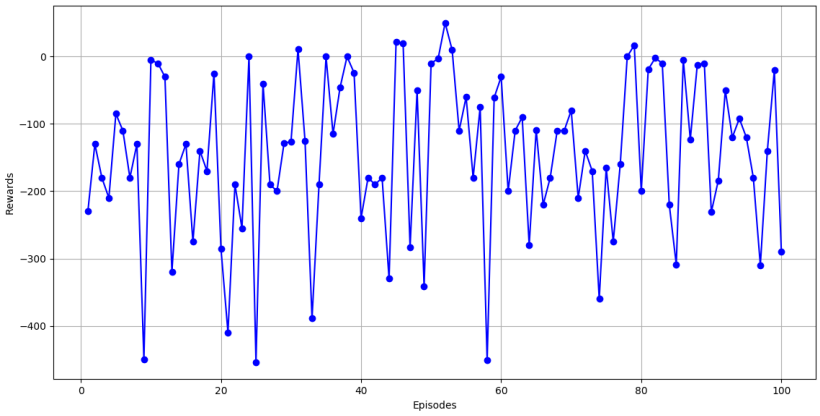


Σχήμα 7.3.3: Διάγραμμα ανταμοιβών δεύτερης εκπαίδευσης.

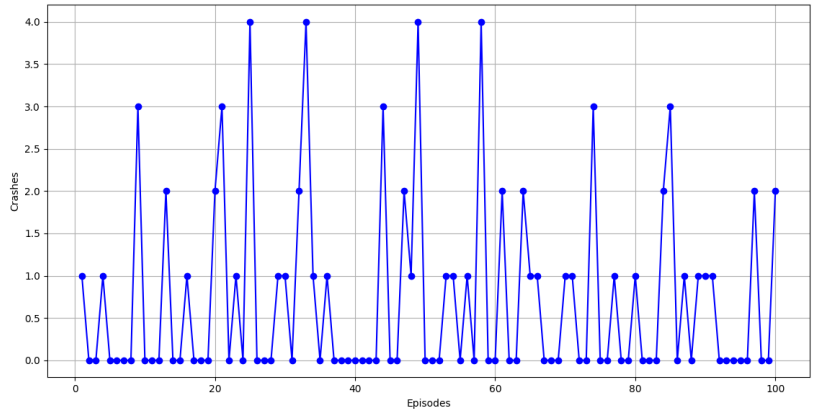


Σχήμα 7.3.4: Διάγραμμα τρακαρισμάτων δεύτερης εκπαίδευσης.

Πείραμα $T3$ με παραμέτρους epsilon = 0.5, alpha = 0.5, g = 0.95, episodes = 100, steps = 20. Το τελικό αποτέλεσμα είναι το ρομποτικό όχημα να έχει συγκεντρώσει συνολική ανταμοιβή -14398 και να έχει τρακάρει 71 φορές.

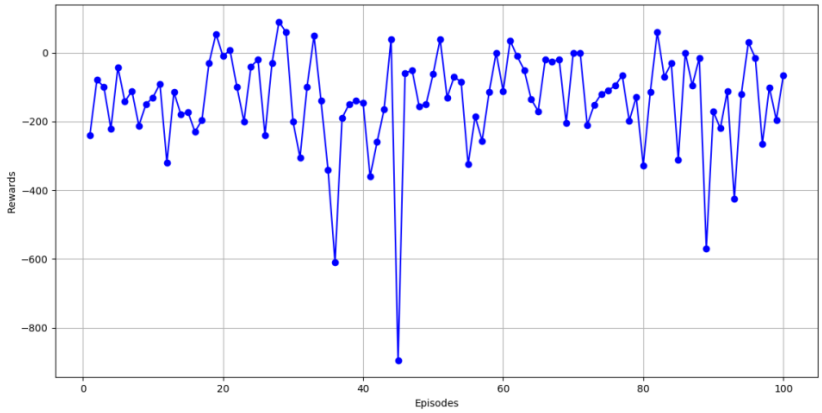


Σχήμα 7.3.5: Διάγραμμα ανταμοιβών τρίτης εκπαίδευσης.

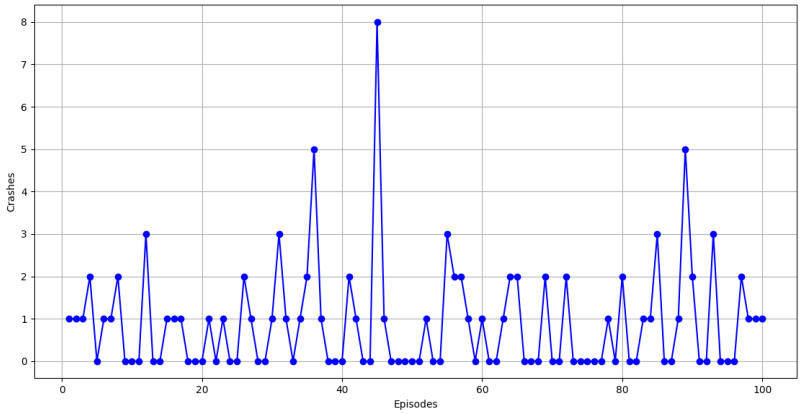


Σχήμα 7.3.6: Διάγραμμα τρακαρισμάτων τρίτης εκπαίδευσης.

Πείραμα $T4$ με παραμέτρους epsilon = 0.25, alpha = 0.5, g = 0.95, episodes = 100, steps = 20. Το τελικό αποτέλεσμα είναι το ρομποτικό όχημα να έχει συγκεντρώσει συνολική ανταμοιβή -13624 και να έχει τρακάρει 89 φορές.

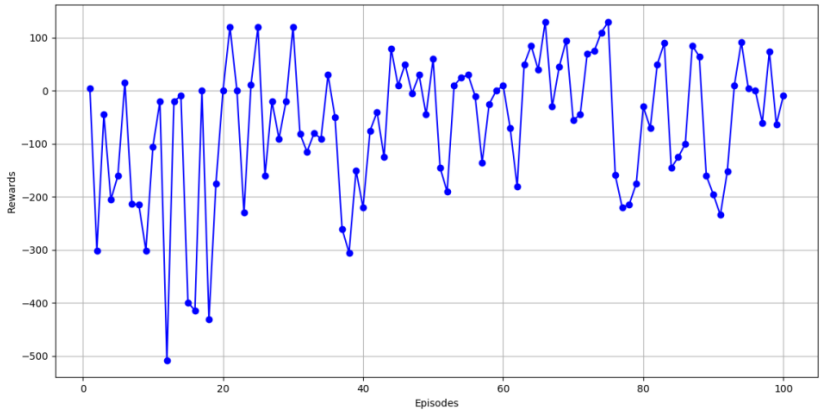


Σχήμα 7.3.7: Διάγραμμα ανταμοιβών τέταρτης εκπαίδευσης.

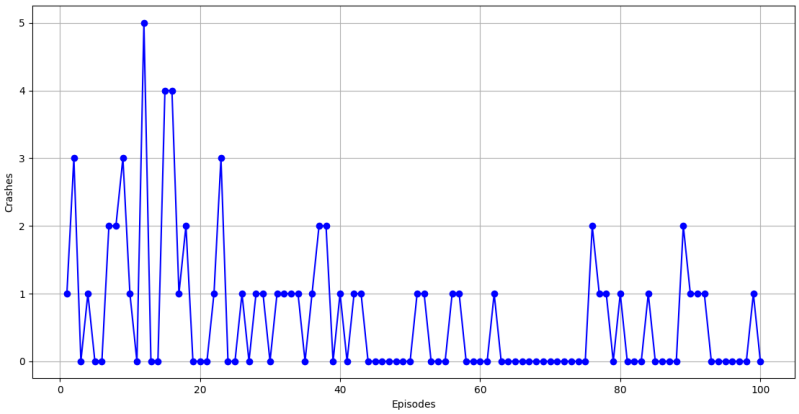


Σχήμα 7.3.8: Διάγραμμα τρακαρισμάτων τέταρτης εκπαίδευσης.

Πείραμα $T5$ με παραμέτρους epsilon = 0.1, alpha = 0.5, g = 0.95, episodes = 100, steps = 20. Το τελικό αποτέλεσμα είναι το ρομποτικό όχημα να έχει συγκεντρώσει συνολική ανταμοιβή -6651 και να έχει τρακάρει 65 φορές.



Σχήμα 7.3.9: Διάγραμμα ανταμοιβών πέμπτης εκπαίδευσης.

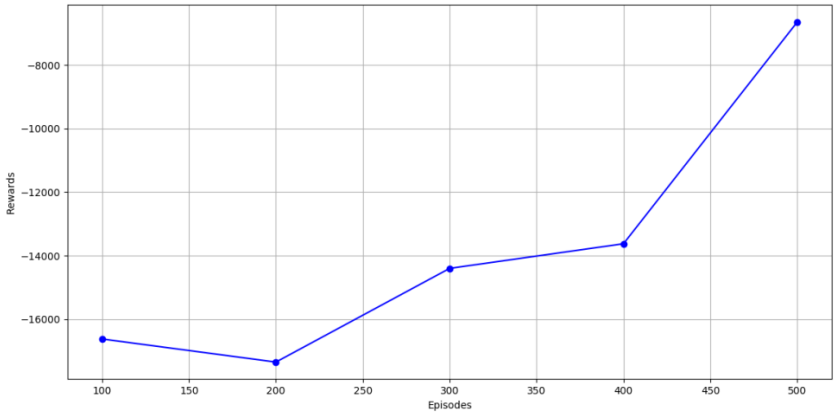


Σχήμα 7.3.10: Διάγραμμα τρακαρισμάτων πέμπτης εκπαίδευσης.

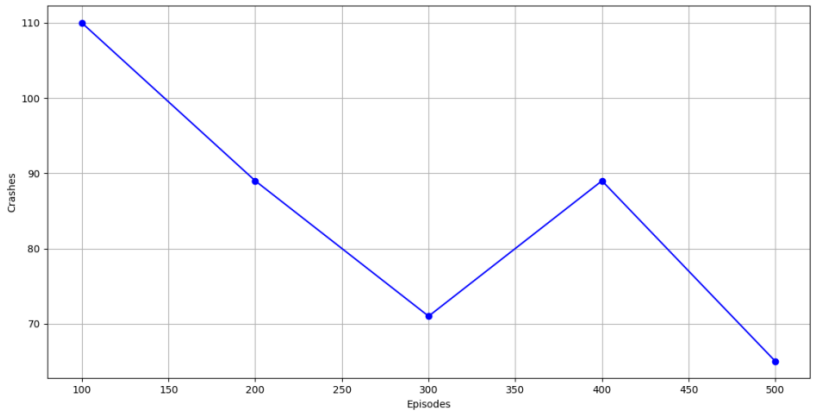
Από την σύγκριση των αποτελεσμάτων η συνολική εικόνα που παρατηρείται από τις εκπαιδεύσεις είναι ότι το ρομποτικό όχημα συγκεντρώνει όλο και μεγαλύτερη ανταμοιβή και μειώνονται τα τρακαρίσματα σε κάθε νέα εκπαίδευση.

Πίνακας 7.3.1: Οι Ανταμοιβές και τα Τρακαρίσματα σε κάθε Εκπαίδευση

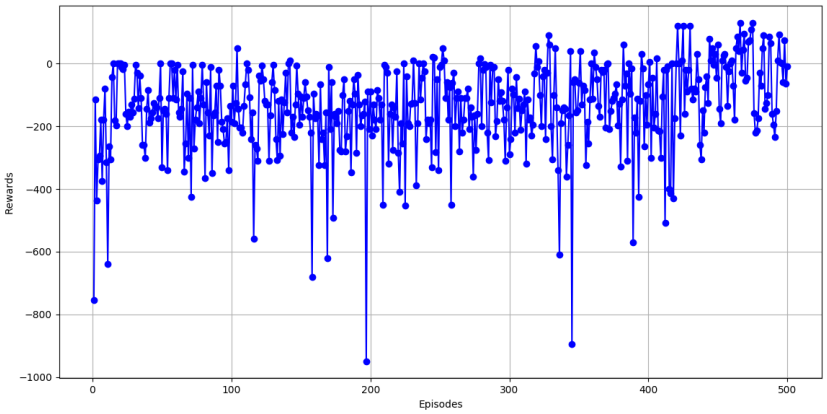
Training	1	2	3	4	5
Rewards	-16619	-17345	-14398	-13624	-6651
Crashes	110	89	71	89	65



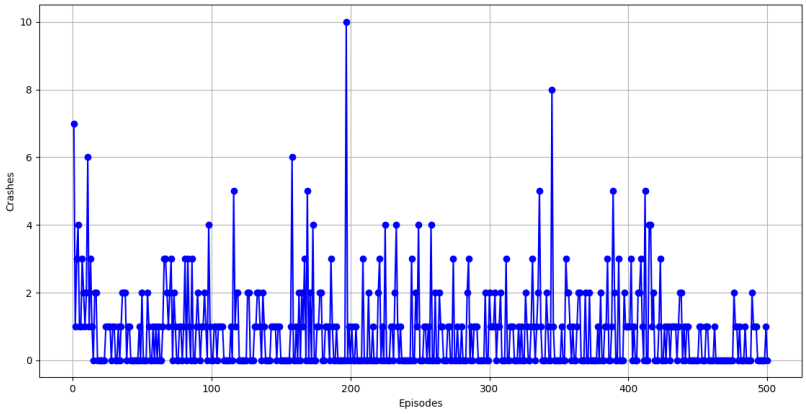
Σχήμα 7.3.11: Διάγραμμα ανταμοιβών πέντε εκπαιδεύσεων.



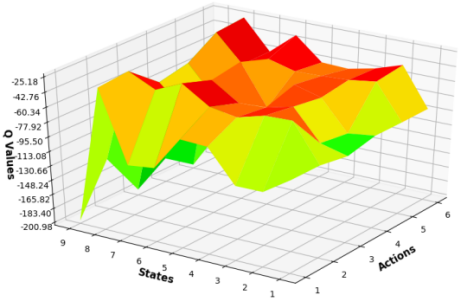
Σχήμα 7.3.12: Διάγραμμα τρακαρισμάτων πέντε εκπαιδεύσεων.



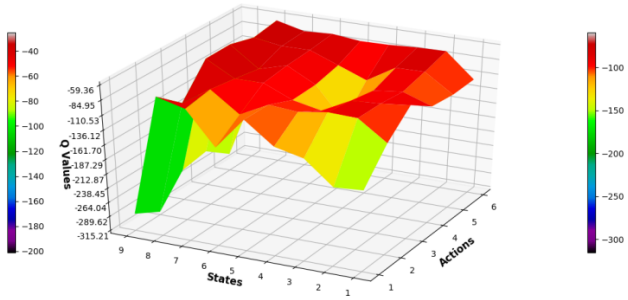
Σχήμα 7.3.13: Διάγραμμα ανταμοιβών για κάθε επεισόδιο.



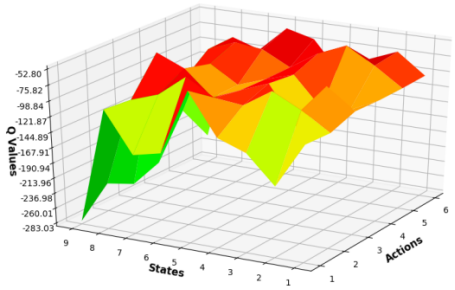
Σχήμα 7.3.14: Διάγραμμα τρακαρισμάτων για κάθε επεισόδιο.



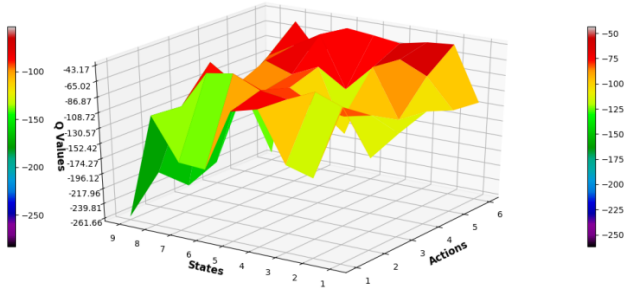
Training 1



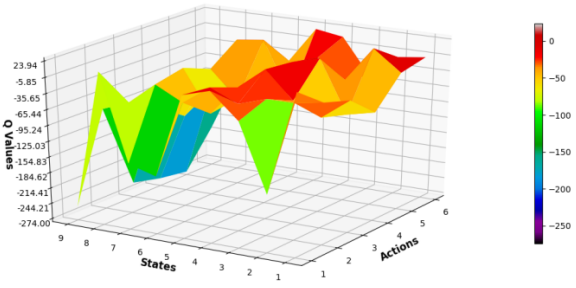
Training 2



Training 3



Training 4



Training 5

Σχήμα 7.3.15: Διαγράμματα τιμών q για πέντε εκπαιδεύσεις.

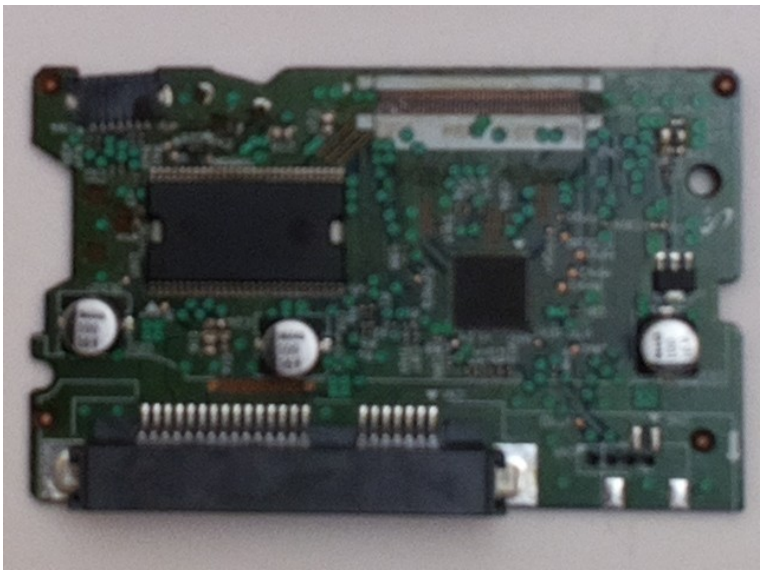
Στο Σχήμα 7.3.15 παρατηρείται η διαδικασία της εκπαίδευσης με το ρομποτικό όχημα να ανανεώνει τις τιμές του πίνακα qtable μέσω της αλληλεπίδρασής του με το περιβάλλον. Επίσης, όσο περισσότερες εκπαιδεύσεις κάνει το ρομποτικό όχημα, τόσο καλύτερη εμπειρία αποκτάει για την κίνησή του στο περιβάλλον.

7.4 Εντοπισμός Αντικειμένου

Για τον εντοπισμό της πλακέτας από το ρομποτικό όχημα τα αρχικά πειράματα διεξάγονται σε ένα νέο περιβάλλον με διάφορα αντικείμενα όπως φαίνεται στην εικόνα 7.4.1. Το συγκεκριμένο περιβάλλον δίνει την δυνατότητα να επιλεγούν οι αποτελεσματικοί παράμετροι για την δημιουργία ενός ισχυρού προγράμματος εντοπισμού.

Εικόνα 7.4.1: Περιβάλλον πειραμάτων για τον εντοπισμό της πλακέτας.

Σε όλα τα πειράματα εκτελείται το Πρόγραμμα Εντοπισμού Αντικειμένου και χρησιμοποιείται η φωτογραφία της πλακέτας όπου η λήψη της έχει γίνει από την Raspberry Pi Camera V2. Έχει γίνει περικοπή της φωτογραφίας, ώστε να περιέχει μόνο την πλακέτα και να έχει ανάλυση 640 × 480 pixels.



Εικόνα 7.4.2: Η φωτογραφία της πλακέτας που χρησιμοποιείται στο πρόγραμμα για τον εντοπισμό.

7.4.1 Παράμετροι Εντοπισμού Αντικειμένου

Η ανάλυση της κάμερας έχει ρυθμιστεί στα 640 × 480 pixels και ο ρυθμός καρέ στα 50. Οι παράμετροι που ρυθμίζονται αφορούν τον αλγόριθμο ORB, την μέθοδο FLANN και την Homography.

Στον ORB οι τιμές των εννέα παραμέτρων που έχουν επιλεγεί είναι nfeatures = 1500, scaleFactor = 1.2, nlevels = 18, edgeThreshold = 10, firstLevel = 0, WTA_K = 2, scoreType = 0, patchSize = 17 και fastThreshold = 40.

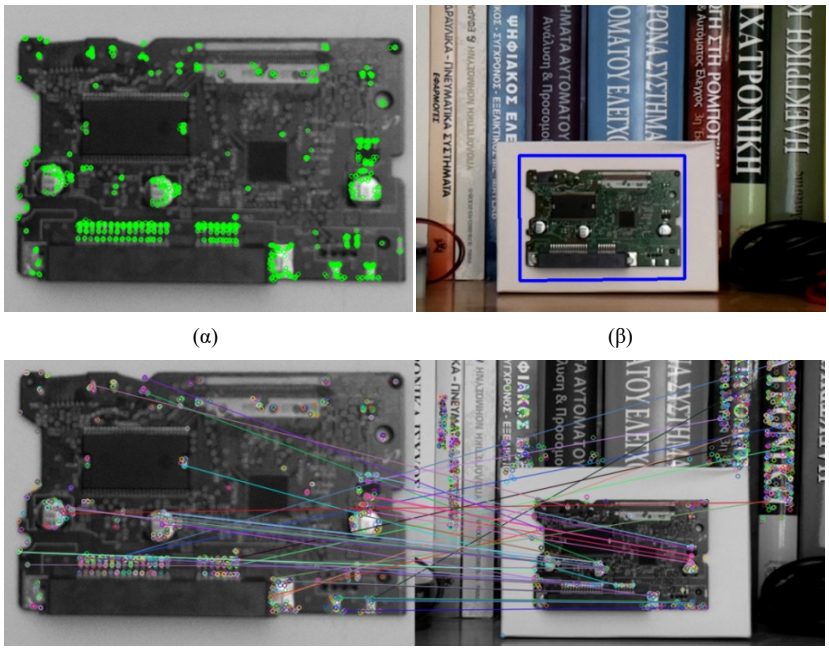
Στον FLANN οι τιμές των παραμέτρων που έχουν επιλεγεί είναι FLANN_INDEX_LSH = 5, index_params= dict(algorithm = FLANN_INDEX_LSH, table_number = 8, key_size = 16, multi_probe_level = 1) και search_params = dict(checks = 60).

Οι τιμές των παραμέτρων που έχουν επιλεγεί στην Homography εάν έχουν εντοπιστεί πάνω από 30 όμοια σημεία είναι query_pts, train_pts, cv2.RANSAC και το ransacReprojThreshold που είναι 4.0.

7.4.2 Πειράματα Εντοπισμού Αντικειμένου

Τα πειράματα διεξάχθηκαν αρχικά στο νέο περιβάλλον και μετά στο περιβάλλον του ρομποτικού οχήματος. Τα αποτελέσματα έδειξαν ότι η μέγιστη απόσταση εντοπισμού της πλακέτας από το ρομποτικό όχημα είναι 75 cm. Σε μεγάλες διακυμάνσεις φωτισμού του περιβάλλοντος παρατηρείται εσφαλμένος εντοπισμός ή δεν εντοπίζεται η πλακέτα.

Παρακάτω παρουσιάζονται εικόνες από τα πειράματα στα δύο περιβάλλοντα με σωστούς και εσφαλμένους εντοπισμούς της πλακέτας.

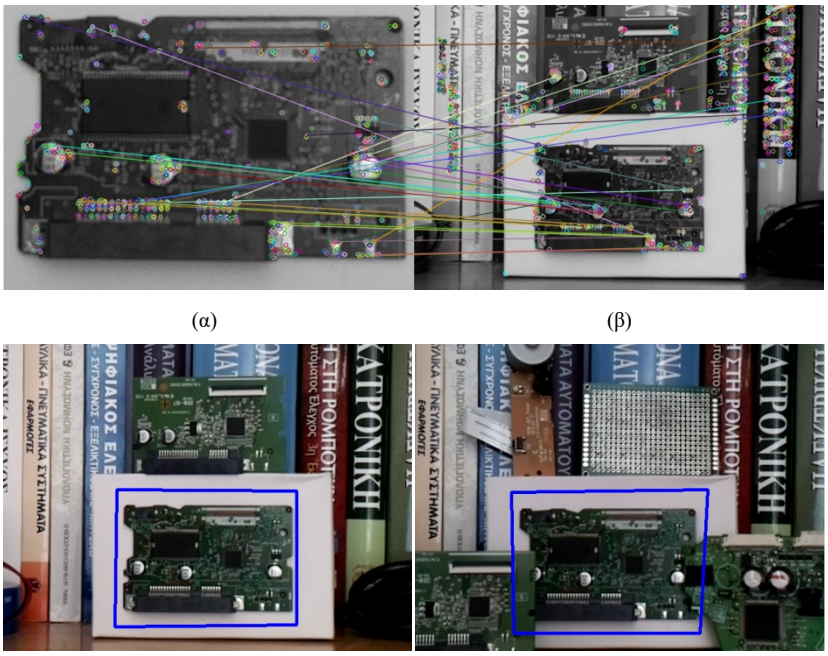


(α) (β)

(γ) (δ)

Σχήμα 7.4.1: Εντοπισμός της πλακέτας από το ρομποτικό όχημα σε απόσταση 40 cm.

Στο σχήμα 7.4.1 παρουσιάζεται η εικόνα (α) στην οποία έχουν βρεθεί τα τοπικά χαρακτηριστικά της φωτογραφίας της πλακέτας και στην εικόνα (β) φαίνεται ότι έχει εντοπιστεί η πλακέτα με το περίγραμμα γύρω της. Στις εικόνες (γ) και (δ) παρουσιάζεται η αντιστοίχηση των τοπικών χαρακτηριστικών που έχουν εντοπιστεί στην φωτογραφία της πλακέτας και στην συνεχόμενη ροή εικόνας.

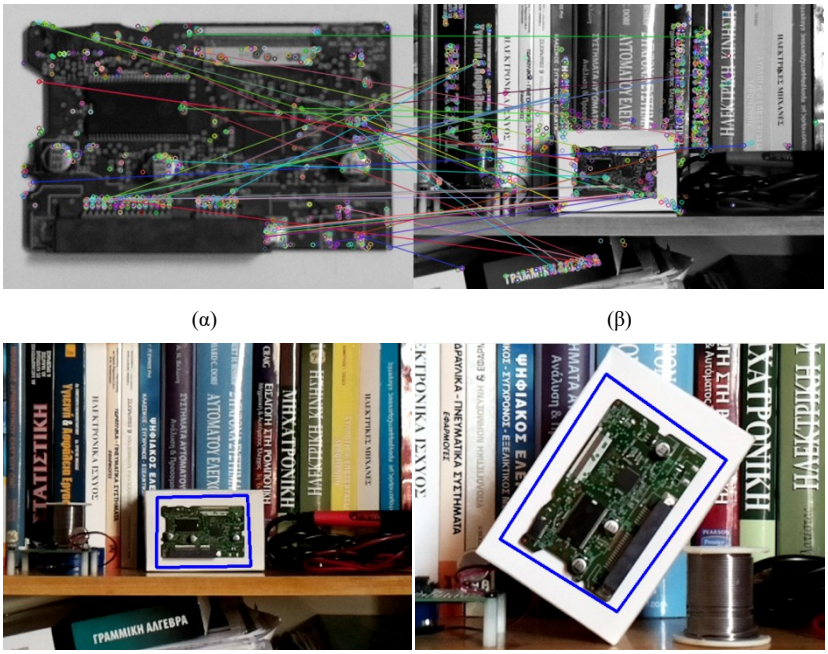


(α) (β)

(γ) (δ)

Σχήμα 7.4.2: Εντοπισμός της πλακέτας από το ρομποτικό όχημα σε απόσταση 40cm με διαφορετικές πλακέτες στο περιβάλλον.

Στο σχήμα 7.4.2 παρουσιάζεται η αντιστοίχηση των τοπικών χαρακτηριστικών που έχουν εντοπιστεί στην φωτογραφία της πλακέτας (α) και στην συνεχόμενη ροή εικόνας (β). Στις εικόνες (γ) και (δ) εντοπίζεται η σωστή πλακέτα, αν και γύρω από αυτή υπάρχουν άλλες σχεδόν όμοιες πλακέτες.

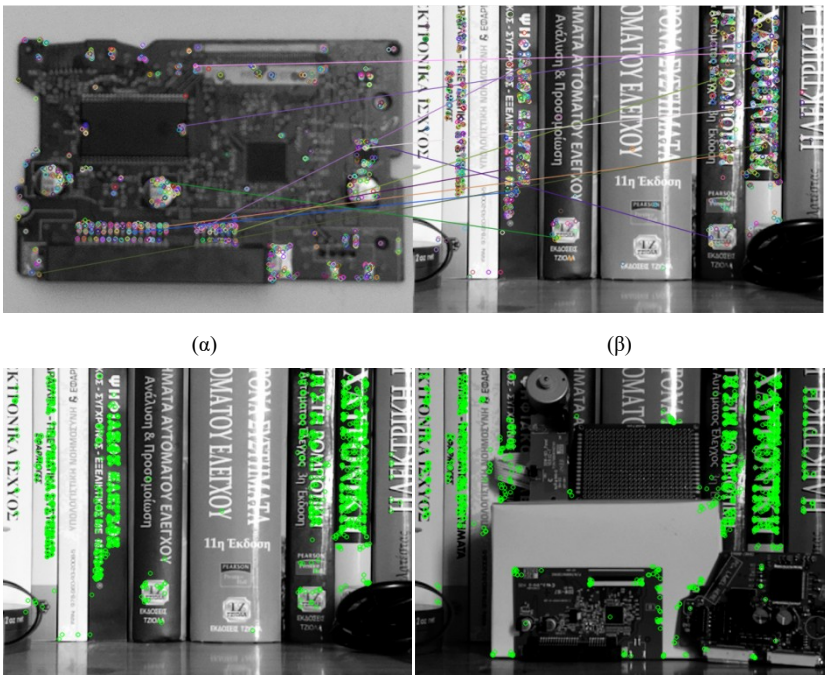


(α) (β)

(γ) (δ)

Σχήμα 7.4.3: Εντοπισμός της πλακέτας από το ρομποτικό όχημα σε απόσταση 40 cm και 75 cm.

Στο σχήμα 7.4.3 παρουσιάζεται η αντιστοίχηση των τοπικών χαρακτηριστικών που έχουν εντοπιστεί στην φωτογραφία της πλακέτας (α) και στην συνεχόμενη ροή εικόνας (β) σε απόσταση 75 cm. Στην εικόνα (γ) εντοπίζεται η πλακέτα σε απόσταση 75 cm, ενώ στην εικόνα (δ) εντοπίζεται η πλακέτα σε απόσταση 40 cm όταν έχει περιστραφεί.

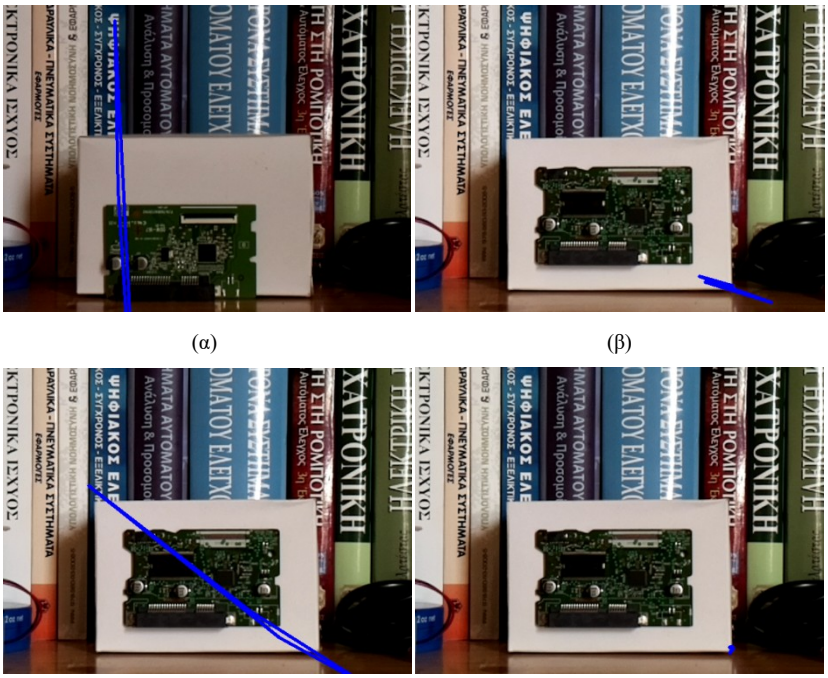


(α) (β)

(γ) (δ)

Σχήμα 7.4.4: Η πλακέτα δεν υπάρχει στο περιβάλλον και δεν εντοπίζεται από το ρομποτικό όχημα.

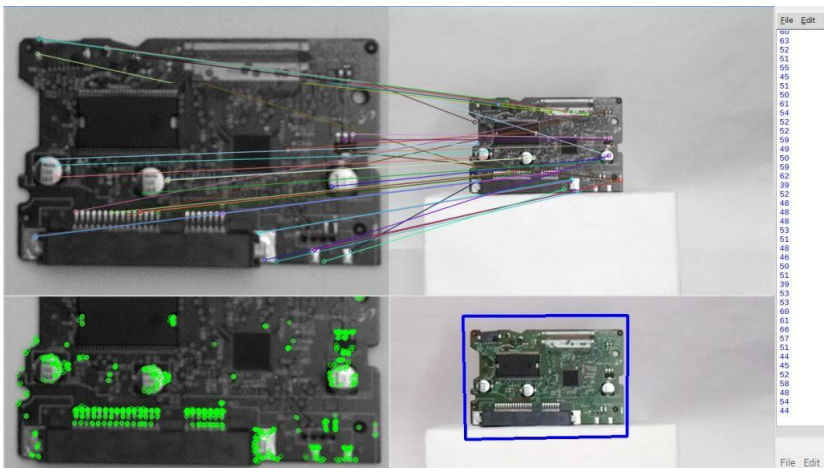
Στο σχήμα 7.4.4 παρουσιάζεται η αντιστοίχηση μερικών τοπικών χαρακτηριστικών που έχουν εντοπιστεί στην φωτογραφία της πλακέτας (α) και στην συνεχόμενη ροή εικόνας (β) σε απόσταση 40 cm. Στις εικόνες (γ) και (δ) δεν εντοπίζεται η πλακέτα και εμφανίζονται μόνο τα τοπικά χαρακτηριστικά από την συνεχόμενη ροή εικόνας σε απόσταση 40 cm.



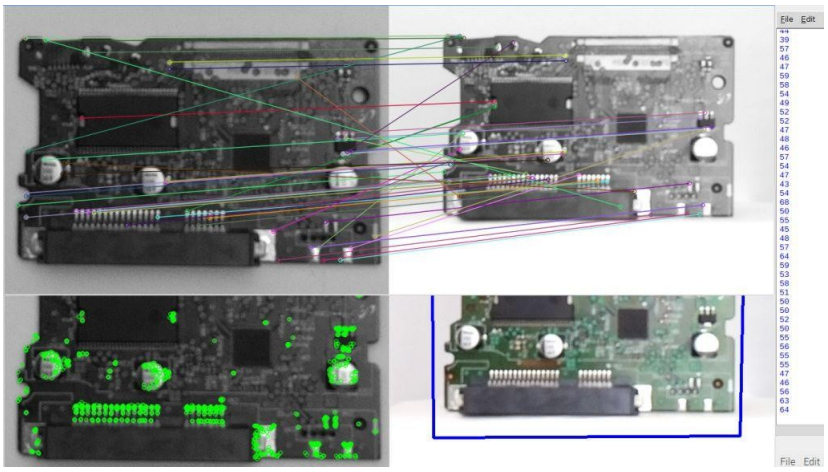
(α) (β)

(γ) (δ)

Σχήμα 7.4.5: Εσφαλμένος εντοπισμός της πλακέτας από το ρομποτικό όχημα σε απόσταση 40 cm όταν έχουν εντοπιστεί πάνω από 19 όμοια σημεία.



Σχήμα 7.4.6: Εντοπισμός της πλακέτας από το ρομποτικό όχημα στο περιβάλλον του σε απόσταση 35 cm όταν έχουν εντοπιστεί πάνω από 40 όμοια σημεία.

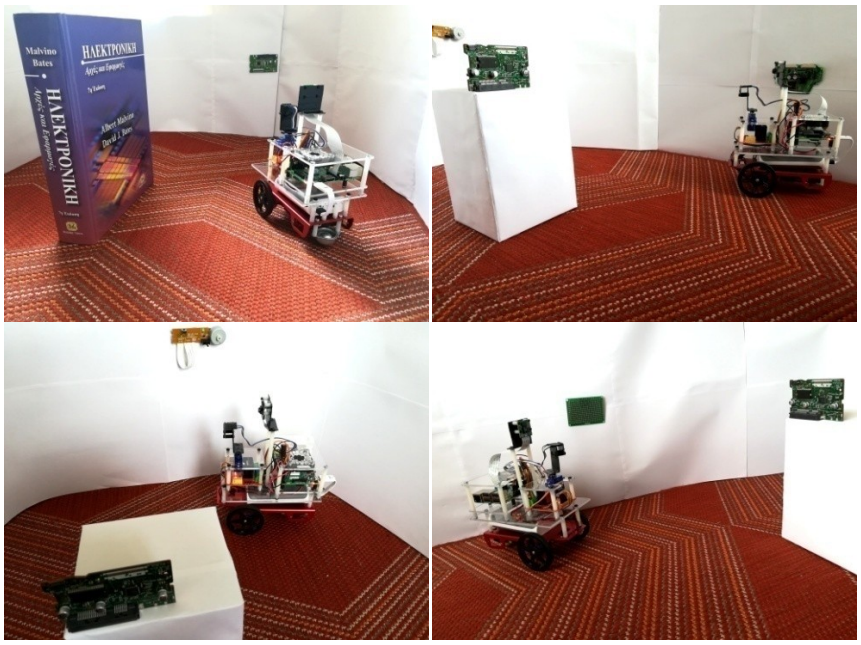


Σχήμα 7.4.7: Εντοπισμός της πλακέτας από το ρομποτικό όχημα στο περιβάλλον του σε απόσταση 15 cm όταν έχουν εντοπιστεί πάνω από 50 όμοια σημεία.

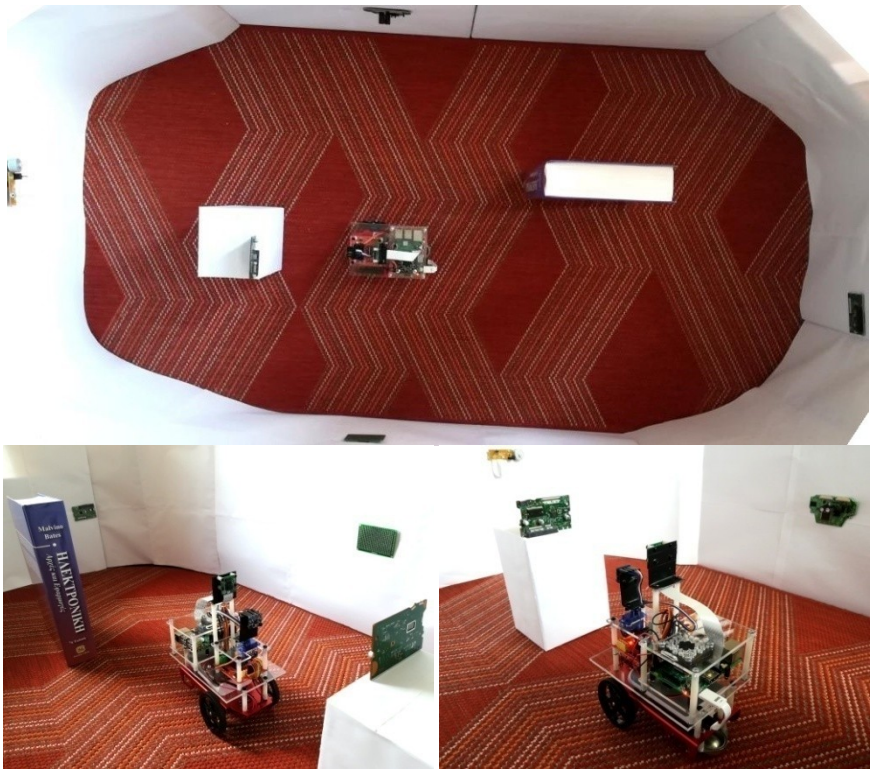
7.5 Τελική Εργασία Αυτόνομου Ρομποτικού Οχήματος

Στο τελικό πείραμα το ρομποτικό όχημα τοποθετείται στο περιβάλλον του και αξιολογείται για την τελική εργασία του εκτελώντας το Τελικό Πρόγραμμα. Το ρομποτικό όχημα χρησιμοποιεί την τελευταία εκπαίδευση για την εκμάθηση της κίνησης, ώστε να πλοηγηθεί στο περιβάλλον του για να εντοπίσει και να προσεγγίσει την πλακέτα.

Το αποτέλεσμα του πειράματος έδειξε ότι η πλοήγηση του ρομποτικού οχήματος στο περιβάλλον του είναι αρκετά ικανοποιητική. Ο εντοπισμός της πλακέτας πραγματοποιείται χωρίς προβλήματα όταν δεν υπάρχουν μεγάλες διακυμάνσεις φωτισμού του περιβάλλοντος. Η προσέγγιση και το σταμάτημα μπροστά από την πλακέτα πραγματοποιείται ομαλά και χωρίς προβλήματα. Τα εμπόδια και οι διάφορες πλακέτες που υπάρχουν στο περιβάλλον του ρομποτικού οχήματος δεν το επηρεάζουν στον εντοπισμό της πλακέτας. Επίσης, σε συνθήκες χαμηλού φωτισμού ο εντοπισμός της πλακέτας δεν πραγματοποιείται και το ρομποτικό όχημα ενώ είναι κοντά στην πλακέτα την αποφεύγει σαν να είναι εμπόδιο.



Σχήμα 7.5.1: Πλοήγηση αυτόνομου ρομποτικού οχήματος.



Σχήμα 7.5.2: Εντοπισμός και προσέγγιση της πλακέτας από το αυτόνομο ρομποτικό όχημα.

8. Συμπεράσματα και Μελλοντικές Επεκτάσεις

Στην εργασία αυτή σχεδιάστηκε και υλοποιήθηκε ένα αυτόνομο ρομποτικό όχημα χρησιμοποιώντας Μηχανική Μάθηση, αλλά και Μηχανική Όραση. Στόχος του ρομποτικού οχήματος ήταν να μάθει να κινείται στο περιβάλλον του για να καταφέρει να εντοπίσει και να προσεγγίσει ένα συγκεκριμένο αντικείμενο. Η συνολική εικόνα που παρατηρήθηκε από την εκτέλεση της εργασίας του ρομποτικού οχήματος ήταν ικανοποιητική και αρκετά ενθαρρυντική για μελλοντικές επεκτάσεις.

Στην εκμάθηση της κίνησης χρησιμοποιήθηκε ο αλγόριθμος Q-Learning όπου με την επιλογή των κατάλληλων παραμέτρων το ρομποτικό όχημα εκπαιδεύτηκε και κατάφερε να κινηθεί αρκετά ικανοποιητικά στο περιβάλλον του. Ενώ στον εντοπισμό του αντικειμένου χρησιμοποιήθηκε ο αλγόριθμος ORB, η μέθοδος FLANN και η Homography. Παρατηρήθηκε ότι ο εντοπισμός και η προσέγγιση στο αντικείμενο ήταν ομαλή και χωρίς προβλήματα όταν δεν υπήρχαν μεγάλες διακυμάνσεις φωτισμού του περιβάλλοντος.

Διάφορες μελλοντικές επεκτάσεις που μπορούν να υλοποιηθούν είναι να γίνουν περισσότερες εκπαιδεύσεις και να προστεθεί αισθητήρας στο πίσω μέρος του ρομποτικού οχήματος για να ανιχνεύονται και σε αυτό το σημείο τα τρακαρίσματα. Με τις περισσότερες εκπαιδεύσεις και την προσθήκη του αισθητήρα το ρομποτικό όχημα θα αποκτήσει περισσότερη εμπειρία και θα βελτιωθεί η κίνησή του.

Επίσης, μπορεί να υλοποιηθεί ένα περιβάλλον προσομοίωσης στον υπολογιστή που να περιλαμβάνει το ρομποτικό όχημα και το περιβάλλον του για την εκμάθηση της κίνησης. Με την προσομοίωση μπορούν να πραγματοποιηθούν πιο πολλές εκπαιδεύσεις γρήγορα και οικονομικά, όπου στο τέλος όλων των εκπαιδεύσεων η αποκτηθείσα εμπειρία για την κίνηση θα μπορούσε να χρησιμοποιηθεί στο πραγματικό ρομποτικό όχημα.

Για τον εντοπισμό του αντικειμένου μπορούν να γίνουν βελτιώσεις και τροποποιήσεις, ώστε να πραγματοποιείται ο εντοπισμός ακόμα και σε μεγάλες διακυμάνσεις φωτισμού του περιβάλλοντος. Ακόμη, μπορεί να υλοποιηθεί ασύρματη σύνδεση του ρομποτικού οχήματος με τον υπολογιστή, ώστε να δοθεί η δυνατότητα σε έναν απομακρυσμένο χρήστη να παρακολουθεί την όλη διαδικασία του εντοπισμού και της προσέγγισης του αντικειμένου, αλλά και να παρέμβει εάν χρειαστεί για να ελέγξει το ρομποτικό όχημα.

Βιβλιογραφία

[1] Alismail, H., Browning, B., & Lucey, S. (2016). Robust Tracking in Low Light and Sudden Illumination Changes. *4th International Conference on 3D Vision, 3DV 2016*, (pp. 389–398). Stanford, CA.

[2] Alpaydın, E. (2010). *Introduction to Machine Learning* (2nd ed.). Cambridge, MA: The MIT Press.

[3] Amayri, M., & Ploix, S. (2015). *Machine Learning with Python/Scikit-Learn-Application to the Estimation of Occupancy and Human Activities* -. Ανάκτηση από http://ibpsa.fr/jdownloads/Simurex/2015/Presentations/30_03_atelierdatamining.pdf

[4] Baggio, D. L., Emami, S., Escrivá, D. M., Ievgen, K., Mahmood, N., Saragih, J., & Shilkrot, R. (2012). *Mastering OpenCV with Practical Computer Vision Projects.* Birmingham: Packt Publishing.

[5] Barber, D. (2017). *Bayesian Reasoning and Machine Learning.* Ανάκτηση από http://web4.cs.ucl.ac.uk/staff/D.Barber/textbook/091117.pdf

[6] Beyeler, M. (2015). *OpenCV with Python Blueprints.* Birmingham: Packt Publishing.

[7] Beyeler, M. (2017). *Machine Learning for OpenCV.* Birmingham: Packt Publishing.

[8] Beyerer, J., León, F. P., & C. F. (2016). *Machine Vision: Automated Visual Inspection: Theory, Practice and Applications.* (J. Meyer, Trans.) Berlin: Springer.

[9] Bradski, G., & Kaehler, A. (2008). *Learning OpenCV.* Sebastopol, CA: O'Reilly Media.

[10] Brahmbhatt, S. (2013). *Practical OpenCV.* New York, NY: Apress Media.

[11] Calonder, M., Lepetit, V., Strecha, C., & Fua, P. (2010). BRIEF: Binary Robust Independent Elementary Features. *Computer Vision-ECCV 2010, 11th European Conference on Computer Vision: Part IV*, (pp. 778–792). Heraklion, Crete.

[12] Davies, E. R. (2012). *Computer and Machine Vision:Theory, Algorithms, Practicalities* (4th ed.). Waltham: Elsevier.

[13] Dearden, R., Friedman, N., & Russell, S. J. (1998). Bayesian Q-Learning. *Proceedings of the 15th National Conference on Artificial Intelligence and 10th Innovative Applications of Artificial Intelligence Conference, AAAI 98, IAAI 98*, (pp. 761–768).

[14] Engelbrecht, A. P. (2002). *Computational Intelligence: An Introduction.* Chichester, WS: John Wiley & Sons.

[15] Fischler, M. A., & Bolles, R. C. (1981). Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Communications of the ACM*, *24*, pp. 381–395.

[16] Goodfellow, I., Bengio, Y., & Courville, A. (2017). *Deep Learning.* Cambridge, MA: The MIT Press.

[17] Harris, C. G., & Stephens, M. (1988). A Combined Corner and Edge Detector. *Proceedings of the 4th Alvey Vision Conference, AVC 1988*, (pp. 147–151). Manchester, UK.

[18] Hartley, R., & Zisserman, A. (2004). *Multiple View Geometry in Computer Vision* (2nd ed.). Cambridge: Cambridge University Press.

[19] Haykin, S. S. (2010). *Νευρωνικά Δίκτυα και Μηχανική Μάθηση* (3η εκδ.). (Ε. Γκαγκάτσιου, Επιμ., & Ε. Γκαγκάτσιου, Μεταφρ.) Αθήνα: Παπασωτηρίου.

[20] Howse, J., Joshi, P., & Beyeler, M. (2016). *OpenCV: Computer Vision Projects with Python.* Birmingham: Packt Publishing.

[21] Hwangbo, J., Lee, J., Dosovitskiy, A., Bellicoso, D., Tsounis, V., Koltun, V., & Hutter, M. (2019). Learning Agile and Dynamic Motor Skills for Legged Robots. *Science Robotics*, *4.*

[22] Jähne, B., Haußecker, H., & Geißler, P. (1999). *Handbook of Computer Vision and Applications.* San Diego, CA: Academic Press.

[23] Julian, D. (2016). *Designing Machine Learning Systems with Python.* Birmingham: Packt Publishing.

[24] Laganiere, R. (2017). *OpenCV 3 Computer Vision Application Programming Cookbook* (3rd ed.). Birmingham: Packt Publishing.

[25] Liu, Y., Pan, Z., Stirling, D., & Naghdy, F. (2009). Q-Learning for Navigation Control of an Autonomous Blimp. *Australasian Conference on Robotics and Automation (ACRA).* Sydney, Australia.

[26] Machado, M. C., Bellemare, M. G., Talvitie, E., Veness, J., Hausknecht, M. J., & Bowling, M. (2018). Revisiting the Arcade Learning Environment: Evaluation Protocols and Open Problems for General Agents. *Proceedings of the 27th International Joint Conference on Artificial Intelligence, IJCAI-18*, (pp. 5573–5577). Stockholm, Sweden.

[27] Manuelli, L., & Florence, P. R. (2015). *Reinforcement Learning for Autonomous Driving Obstacle Avoidance using LIDAR.* Technical Report, Massachusetts Institute of Technology.

[28] Mayor, A. (2018). *Gods and Robots: Myths, Machines, And Ancient Dreams Of Technology.* Princeton, NJ: Princeton University Press.

[29] Meuleau, N., Peshkin, L., & Kim, K.-E. (2001). *Exploration in Gradient-Based Reinforcement Learning.* Technical Report, Massachusetts Institute of Technology, Artificial Intelligence Laboratory, Cambridge, MA.

[30] Michie, D., Spiegelhalter, D. J., & Taylor, C. C. (1994). *Machine Learning, Neural and Statistical Classification.* Hemel Hempstead, Hertfordshire, England: Ellis Horwood.

[31] Mohri, M., Rostamizadeh, A., & Talwalkar, A. (2012). *Foundations of Machine Learning.* Cambridge, MA: The MIT Press.

[32] Muja, M., & Lowe, D. G. (2009). Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration. *Proceedings of the 4th International Conference on Computer Vision Theory and Applications, VISAPP 2009*, *1*, pp. 331–340. Lisboa, Portugal.

[33] Ng, A. Y., & Russell, S. J. (2000). Algorithms for Inverse Reinforcement Learning. *Proceedings of the 17th International Conference on Machine Learning, ICML-00*, (pp. 663–670). Stanford, CA.

[34] Nilsson, N. J. (1998). *Introduction to Machine Learning. An early draft of a proposed textbook.* Stanford, CA.

[35] Nilsson, N. J. (2010). *The Quest for Artificial Intelligence.* Cambridge: Cambridge University Press.

[36] Nixon, M. S., & Aguado, A. S. (2002). *Feature Extraction and Image Processing.* Oxford: Newnes.

[37] Pajankar, A. (2015). *Raspberry Pi Computer Vision Programming.* Birmingham: Packt Publishing.

[38] Parker, J. R. (2011). *Algorithms for Image Processing and Computer Vision* (2nd ed.). Indianapolis, IN: Wiley Publishing.

[39] Prescott, T. J., & Mayhew, J. E. (1991). Obstacle Avoidance through Reinforcement Learning. *Advances in Neural Information Processing Systems 4, NIPS Conference*, (pp. 523–530). Denver, CO.

[40] Ramakrishna, M., & Shylaja, S. (2014). Is ORB Efficient Over SURF for Object Recognition ? *International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)*, *3*, pp. 2783–2788.

[41] Raschka, S. (2015). *Python Machine Learning.* Birmingham: Packt Publishing.

[42] Rosin, P. L. (1999). Measuring Corner Properties. *Computer Vision and Image Understanding*, *73*, pp. 291–307.

[43] Rosten, E., & Drummond, T. (2005). Fusing Points and Lines for High Performance Tracking. *10th IEEE International Conference on Computer Vision, ICCV05*, *2*, pp. 1508–1515. Beijing, China.

[44] Rosten, E., & Drummond, T. (2006). Machine Learning for High-Speed Corner Detection. *Computer Vision-ECCV 2006, 9th European Conference on Computer Vision*, *1*, pp. 430–443. Graz, Austria.

[45] Rosten, E., Porter, R. B., & Drummond, T. (2010). Faster and Better: A Machine Learning Approach to Corner Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *32*, pp. 105–119.

[46] Rublee, E., Rabaud, V., Konolige, K., & Bradski, G. R. (2011). ORB: An efficient alternative to SIFT or SURF. *IEEE International Conference on Computer Vision, ICCV 2011*, (pp. 2564–2571). Barcelona, Spain.

[47] Rummery, G. A., & Niranjan, M. (1994). *On-line Q-learning Using Connectionist Systems.* Technical Report, Cambridge University, Engineering Department, Cambridge.

[48] Russell, S. J., & Norvig, P. (2005). *Τεχνητή Νοημοσύνη: Μια Σύγχρονη Προσέγγιση* (2η εκδ.). (Γ. Ρεφανίδης, Επιμ.) Αθήνα: Κλειδάριθμος.

[49] Russell, S. J., & Norvig, P. (2010). *Artificial Intelligence: A Modern Approach* (3rd ed.). Upper Saddle River, NJ: Prentice Hall.

[50] Si, J., Barto, A. G., Powell, W. B., & Wunsch II, D. (2004). *Handbook of Learning and Approximate Dynamic Programming.* Piscataway, NJ: IEEE Press.

[51] Siciliano, B., Sciavicco, L., Villani, L., & Oriolo, G. (2010). *Robotics: Modelling, Planning and Control.* London: Springer.

[52] Smart, W. D., & Kaelbling, L. P. (2002). Effective Reinforcement Learning for Mobile Robots. *Proceedings of the 2002 IEEE International Conference on Robotics and Automation, ICRA 2002*, *4*, pp. 3404–3410. Washington, DC.

[53] Smola, A., & Vishwanathan, S. (2008). *Introduction to Machine Learning.* Cambridge: Cambridge University Press.

[54] Solem, J. E. (2012). *Programming Computer Vision.* Sebastopol, CA: O'Reilly Media.

[55] Sutton, R. S., & Barto, A. G. (2012). *Reinforcement Learning: An Introduction* (2nd ed.). Cambridge, MA: The MIT Press.

[56] Sutton, R. S., & Barto, A. G. (2018). *Reinforcement Learning: An Introduction* (2nd ed.). Cambridge, MA: The MIT Press.

[57] Szeliski, R. (2010). *Computer Vision: Algorithms and Applications.* Springer.

[58] Tai, L., Paolo, G., & Liu, M. (2017). Virtual-to-real Deep Reinforcement Learning: Continuous Control of Mobile Robots for Mapless Navigation. *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, (pp. 31–36).

[59] Tan, J., Zhang, T., Coumans, E., Iscen, A., Bai, Y., Hafner, D., . . . Vanhoucke, V. (2018). Sim-to-Real: Learning Agile Locomotion For Quadruped Robots. *Proceedings of Robotics: Science and Systems (RSS).* Pittsburgh, PA.

[60] Taylor, M. E., Kulis, B., & Sha, F. (2011). Metric Learning for Reinforcement Learning Agents. *10th International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2011*, (pp. 777–784). Taipei, Taiwan.

[61] Tom, M. M. (1997). *Machine Learning.* New York, NY: McGraw-Hill.

[62] Watkins, C. J. (1989). *Learning from Delayed Rewards.* Ph.D Thesis, King's College, Cambridge, UK.

[63] Watkins, C. J., & Dayan, P. (1992). Technical Note: Q-Learning. *Machine Learning*, *8*, pp. 279–292.

[64] Wicaksono, H. (2011). Q Learning Behavior on Autonomous Navigation of Physical Robot. *8th International Conference on Ubiquitous Robots and Ambient Intelligence, URAI 2011*, (pp. 50–54). Incheon, Korea (South).

[65] Yang, J., Shi, Y., & Rong, H.-J. (2016). Random neural Q-learning for obstacle avoidance of a mobile robot in unknown environments. *Advances in Mechanical Engineering*, *8*, pp. 1–15.

[66] Zamora, I., Lopez, N. G., Vilches, V. M., & Cordero, A. H. (2016). Extending the OpenAI Gym for robotics: a toolkit for reinforcement learning using ROS and Gazebo. *CoRR*, *abs/1608.05742.*

[67] Κόντος, Ι. (1996). *Τεχνητή Νοημοσύνη και Λογομηχανική.* Αθήνα: Ευγ. Μπένου.

[68] Μάθηση. (1997). *Εγκυκλοπαίδεια Δομή: Από Γενιά σε Γενιά* (Τόμ. 19). Αθήνα: Δομή.

[69] Μπούταλης, Γ., & Συρακούλης, Γ. (2010). *Υπολογιστική Νοημοσύνη & Εφαρμογές.* Ξάνθη.

Διαδίκτυο

[70] *Adafruit VL53L0X Time of Flight Micro-LIDAR Distance Sensor*. (n.d.). Ανάκτηση Οκτώβριος 23, 2019, από https://learn.adafruit.com/adafruit-vl53l0x-micro-lidar-distance-sensor-breakout/pinouts

[71] *Computer Vision: Algorithms and Applications*. (n.d.). Ανάκτηση Οκτώβριος 17, 2019, από http://szeliski.org/Book/

[72] *Deep reinforcement learning on GCP: using hyperparameter tuning and Cloud ML Engine to best OpenAI Gym games*. (n.d.). Ανάκτηση Σεπτέμβριος 11, 2019, από https://cloud.google.com/blog/products/ai-machine-learning/deep-reinforcement-learning-on-gcp-using-hyperparameters-and-cloud-ml-engine-to-best-openai-gym-games

[73] *FAST Corner Detection – Edward Rosten*. (n.d.). Ανάκτηση Οκτώβριος 9, 2019, από https://www.edwardrosten.com/work/fast.html

[74] *FEETECH RC Model Co.,Ltd.* (n.d.). Ανάκτηση Οκτώβριος 24, 2019, από http://www.feetechrc.com/

[75] *GP Batteries International*. (n.d.). Ανάκτηση Οκτώβριος 28, 2019, από https://www.gpbatteries.com/

[76] *GP ReCyko 2600mAh AA*. (n.d.). Ανάκτηση Οκτώβριος 28, 2019, από https://www.gpbatteries.com/int_en/batteries/rechargeable-batteries/2600-aa-2

[77] *IIPImage " Images*. (n.d.). Ανάκτηση Οκτώβριος 9, 2019, από https://iipimage.sourceforge.io/documentation/images/

[78] *johnbryanmoore/VL53L0X_rasp_python*. (n.d.). Ανάκτηση Οκτώβριος 30, 2019, από https://github.com/johnbryanmoore/VL53L0X_rasp_python

[79] *L298*. (n.d.). Ανάκτηση Οκτώβριος 27, 2019, από https://www.st.com/content/st_com/en/products/motor-drivers/brushed-dc-motor-drivers/l298.html#overview

[80] *Matplotlib*. (n.d.). Ανάκτηση Οκτώβριος 30, 2019, από https://matplotlib.org/

[81] *Micro 1.8kg.cm Metal Gears Analog Servo FS90MG*. (n.d.). Ανάκτηση Οκτώβριος 24, 2019, από FEETECH RC Model Co.,Ltd.: http://www.feetechrc.com/product/analog-servo/micro-1-8kg-cm-metal-gears-analog-servo-fs90mg/

[82] *Multiple View Geometry in Computer Vision Second Edition*. (n.d.). Ανάκτηση Οκτώβριος 15, 2019, από https://www.robots.ox.ac.uk/~vgg/hzbook/

[83] *NumPy*. (n.d.). Ανάκτηση Οκτώβριος 30, 2019, από https://numpy.org/

[84] *OpenCV*. (n.d.). Ανάκτηση Οκτώβριος 29, 2019, από https://opencv.org/

[85] *Power Bank 6000*. (n.d.). Ανάκτηση Οκτώβριος 28, 2019, από https://www.varta-consumer.com/en/products/portable-power/overview/details/power-bank-6000

[86] *Python.org*. (n.d.). Ανάκτηση Οκτώβριος 29, 2019, από https://www.python.org/

[87] *Raspberry Pi*. (n.d.). Ανάκτηση Οκτώβριος 24, 2019, από https://www.raspberrypi.org/

[88] *Raspberry Pi 3 Model B GPIO 40 Pin Block Pinout*. (n.d.). Ανάκτηση Οκτώβριος 21, 2019, από https://www.element14.com/community/docs/DOC-73950/l/raspberry-pi-3-model-b-gpio-40-pin-block-pinout

[89] *Reinforcement Learning - Algorithms*. (n.d.). Ανάκτηση Σεπτέμβριος 21, 2019, από https://www.cse.unsw.edu.au/~cs9417ml/RL1/algorithms.html

[90] *SciPy library*. (n.d.). Ανάκτηση Οκτώβριος 30, 2019, από https://www.scipy.org/scipylib/index.html

[91] *SciPy.org*. (n.d.). Ανάκτηση Οκτώβριος 30, 2019, από https://www.scipy.org/

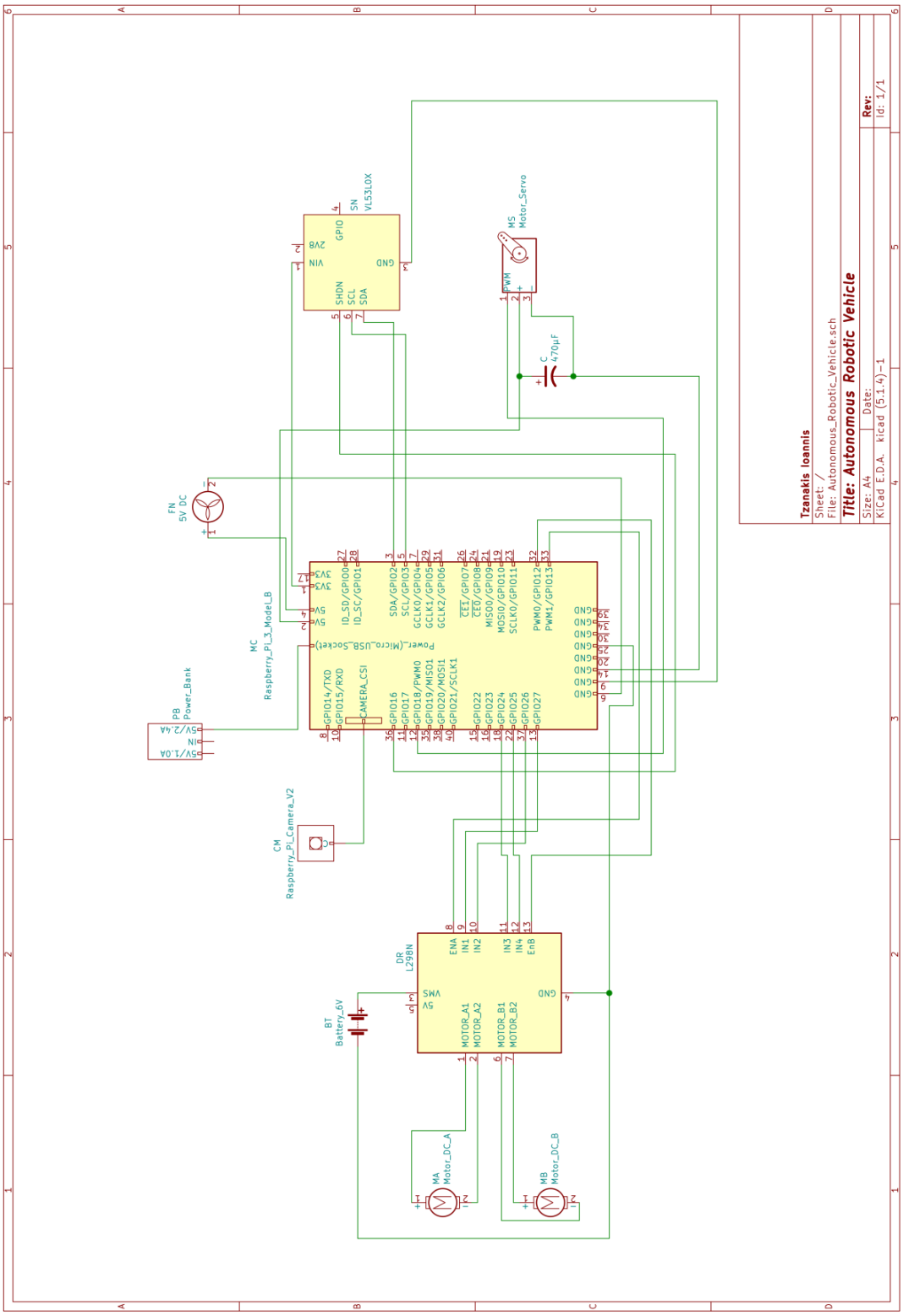
[92] *STMicroelectronics*. (n.d.). Ανάκτηση Οκτώβριος 27, 2019, από https://www.st.com/content/st_com/en.html

[93] *The pigpio library*. (n.d.). Ανάκτηση Οκτώβριος 30, 2019, από http://abyz.me.uk/rpi/pigpio/

[94] *VARTA Consumer Batteries*. (n.d.). Ανάκτηση Οκτώβριος 28, 2019, από https://www.varta-consumer.com/en

[95] *VL53L0X*. (n.d.). Ανάκτηση Οκτώβριος 23, 2019, από https://www.st.com/en/imaging-and-photonics-solutions/vl53l0x.html

Παράρτημα Α



Σχήμα Α.1: Διάγραμμα κυκλώματος αυτόνομου ρομποτικού οχήματος.

Πρόγραμμα DC Κινητήρες και Actions

```
1   ################################
2   # Title: DC Motors and Actions #
3   # Author: Tzanakis Ioannis     #
4   ################################
5
6   import RPi.GPIO as GPIO
7   import time
8   from time import sleep
9
10  # Pins L298N-RPI
11  MotorENA = 13
12  MotorIN1 = 27
13  MotorIN2 = 26
14
15  MotorENB = 12
16  MotorIN3 = 24
17  MotorIN4 = 25
18
19  GPIO.setmode(GPIO.BCM)
20  GPIO.setup(MotorENA,GPIO.OUT)
21  GPIO.setup(MotorIN1,GPIO.OUT)
22  GPIO.setup(MotorIN2,GPIO.OUT)
23  pwm_motor1=GPIO.PWM(MotorENA,50) # 35Hz
24  pwm_motor1.start(0)
25
26  GPIO.setup(MotorENB,GPIO.OUT)
27  GPIO.setup(MotorIN3,GPIO.OUT)
28  GPIO.setup(MotorIN4,GPIO.OUT)
29  pwm_motor2=GPIO.PWM(MotorENB,50) # 35Hz
30  pwm_motor2.start(0)
31
32  actions = ['FORWARD','BACKWARD', 'FRIGHT', 'FLEFT', 'BRIGHT', 'BLEFT']
33
34  # opou '#' me mpataries kai '##' me trofodotiko
35
36  # Empros
37  def FORWARD():
38      GPIO.output(MotorIN1,False)
39      GPIO.output(MotorIN2,True)
40      pwm_motor1.ChangeDutyCycle(25) #25 ##44
41
42      GPIO.output(MotorIN3,False)
43      GPIO.output(MotorIN4,True)
44      pwm_motor2.ChangeDutyCycle(20) #20 ##40
45
46   # Pisw
47  def BACKWARD():
48      GPIO.output(MotorIN1,True)
49      GPIO.output(MotorIN2,False)
50      pwm_motor1.ChangeDutyCycle(22) #22 ##34
51
52      GPIO.output(MotorIN3,True)
53      GPIO.output(MotorIN4,False)
54      pwm_motor2.ChangeDutyCycle(20) #20 ##30
55
56  # Empros Deksia
57  def FRIGHT():
58      GPIO.output(MotorIN1,False)
```

```
59      GPIO.output(MotorIN2,True)
60      pwm_motor1.ChangeDutyCycle(0)
61
62      GPIO.output(MotorIN3,False)
63      GPIO.output(MotorIN4,True)
64      pwm_motor2.ChangeDutyCycle(28) #28 ##45
65
66  # Empros Aristera
67  def FLEFT():
68      GPIO.output(MotorIN1,False)
69      GPIO.output(MotorIN2,True)
70      pwm_motor1.ChangeDutyCycle(33) #33 ##50
71
72      GPIO.output(MotorIN3,True)
73      GPIO.output(MotorIN4,False)
74      pwm_motor2.ChangeDutyCycle(0)
75
76  # Pisw deksia
77  def BRIGHT():
78      GPIO.output(MotorIN1,True)
79      GPIO.output(MotorIN2,False)
80      pwm_motor1.ChangeDutyCycle(33) #33 ##50
81
82      GPIO.output(MotorIN3,True)
83      GPIO.output(MotorIN4,False)
84      pwm_motor2.ChangeDutyCycle(0)
85
86  # Pisw aristera
87  def BLEFT():
88      GPIO.output(MotorIN1,False)
89      GPIO.output(MotorIN2,True)
90      pwm_motor1.ChangeDutyCycle(0)
91
92      GPIO.output(MotorIN3,True)
93      GPIO.output(MotorIN4,False)
94      pwm_motor2.ChangeDutyCycle(28) #28 ##45
95
96  def STOP():
97      GPIO.output(MotorENA,GPIO.LOW)
98      GPIO.output(MotorENB,GPIO.LOW)
99      GPIO.cleanup()
100
101
102 def FORWARD2():
103     GPIO.output(MotorIN1,False)
104     GPIO.output(MotorIN2,True)
105     pwm_motor1.ChangeDutyCycle(22)
106
107     GPIO.output(MotorIN3,False)
108     GPIO.output(MotorIN4,True)
109     pwm_motor2.ChangeDutyCycle(20)
110
111 def STOP2():
112     GPIO.output(MotorIN1,False)
113     GPIO.output(MotorIN2,True)
114     pwm_motor1.ChangeDutyCycle(0)
115
116     GPIO.output(MotorIN3,False)
117     GPIO.output(MotorIN4,True)
118     pwm_motor2.ChangeDutyCycle(0)
```

Πρόγραμμα Μέτρησης Απόστασης και States

```
1  ##############################
2  # Title: Distance and States #
3  # Author: Tzanakis Ioannis   #
4  ##############################
5
6  from os import system
7  system("sudo pigpiod")
8  import pigpio
9  import time
10 import VL53L0X
11
12 pi = pigpio.pi()
13 pi.set_mode(18, pigpio.OUTPUT)
14
15 tof = VL53L0X.VL53L0X()
16 tof.start_ranging(VL53L0X.VL53L0X_HIGH_SPEED_MODE)
17 timing = tof.get_timing()
18
19 def centre():
20     pi.set_servo_pulsewidth(18, 1355)
21     #print("centre position")
22     time.sleep(0.5)
23     C = tof.get_distance()/10
24     #print ("%d C" % (C))
25     time.sleep(timing/1000000.00)
26     if C > 120:
27         C = 120
28
29     return C
30
31 def right():
32     pi.set_servo_pulsewidth(18, 900)
33     #print("right position")
34     time.sleep(0.5)
35     R = tof.get_distance()/10
36     #print ("%d R" % (R))
37     time.sleep(timing/1000000.00)
38     if R > 120:
39         R = 120
40
41     return R
42
43 def left():
44     pi.set_servo_pulsewidth(18, 2100)
45     #print("left position")
46     time.sleep(0.5)
47     L = tof.get_distance()/10
48     #print ("%d L" % (L))
49     time.sleep(timing/1000000.00)
50     if L > 120:
51         L = 120
52
53     return L
54
55 def servo_stop():
56     pi.set_servo_pulsewidth(18, 1355)
57     time.sleep(0.5)
58     pi.set_servo_pulsewidth(18, 0)
```

```
    pi.stop()
    tof.stop_ranging()


def states():

    R = right()
    C = centre()
    L = left()

    global state

    if (R >= 20 and C >= 20 and L >= 20):
        state = 0

    elif (R >= 20 and 5 < C < 20 and 5 < L < 20):
        state = 1

    elif (5 < R < 20 and 5 < C < 20 and L >= 20):
        state = 2

    elif (R >= 20 and C >= 20 and 5 < L < 20):
        state = 3

    elif (5 < R < 20 and C >= 20 and L >= 20):
        state = 4

    elif (5 < R < 20 and 5 < C < 20 and 5 < L < 20):
        state = 5

    elif (R >= 20 and 5 < C < 20 and L >= 20):
        state = 6

    elif (5 < R < 20 and C >= 20 and 5 < L < 20):
        state = 7

    elif (R <= 5 or C <= 5 or L <= 5):
        state = 8

    return state
```

Πρόγραμμα Εκπαίδευσης Q-Learning

```
##############################
# Title: Training Q-Learning #
# Author: Tzanakis Ioannis   #
##############################

from DistStates import*
from DCMotActions import*
import numpy as np
import time
import datetime
from pathlib import Path

R = right()
L = left()
C = centre()
time.sleep(10)
```

```
17
18  def move(action):
19
20      if action == 0:
21          FORWARD()
22
23      elif action == 1:
24          BACKWARD()
25
26      elif action == 2:
27          FRIGHT()
28
29      elif action == 3:
30          FLEFT()
31
32      elif action == 4:
33          BRIGHT()
34
35      elif action == 5:
36          BLEFT()
37
38  def R():
39
40      r = np.array([[10, -10, -10, -10, -10, -10],
41                    [-10, -5, 1, -10, 0, 0],
42                    [-10, -5, -10, 1, 0, 0],
43                    [10, -5, 0, -10, 0, 0],
44                    [10, -5, -10, 0, 0, 0],
45                    [-10, 0, 0, 0, 0, 0],
46                    [-10, -5, 0, 0, 0, 0],
47                    [10, -5, -1, -1, -1, -1],
48                    [-100, -100, -100, -100, -100, -100]])
49
50      return r
51
52
53  saved_table = Path('q.npy')
54  if saved_table.is_file():
55      qtable = np.load('q.npy')
56  else:
57      s = 9
58      a = len(actions)
59      qtable = np.zeros((s, a))
60
61  print(qtable)
62
63
64  epsilon = 0.1
65  alpha = 0.5
66  g = 0.95
67  episodes = 100
68  steps = 20
69  #total_rewards = 0
70  #reward = 0
71  ter = np.array([]).reshape(0,1) # total episode rewards
72  crsh = np.array([]).reshape(0,1) # crash
73
74
75  try:
76      start = datetime.datetime.now()
77      for ep in range (episodes):
```

```
78          step = 0
79
80          total_rewards = 0
81          total_crashes = 0
82
83          state = states()
84
85          while step < steps:
86
87              if np.random.rand() < epsilon:
88                  action = np.random.choice(len (actions))
89                  if epsilon >  0.1:
90                      epsilon *= 0.9
91              else:
92                  action = np.argmax(qtable[state,:])
93
94              take_act = move(action)
95              reward = R()
96              r = reward[state, action]
97              state_new = states()
98              '''print('----')
99              print(r)'''
100
101             qtable[state, action] = qtable[state, action] + alpha * (r
+ g * np.max(qtable[state_new, :]) - qtable[state, action])
102
103             total_rewards += r
104
105             state = state_new
106             if state == 8:
107                 total_crashes += 1
108                 #print('crashhh')
109
110             step +=1
111         '''print('****************************')
112         print(ep)
113         print('****************************')'''
114
115         ter = np.vstack([ter, [total_rewards]])
116         np.save('reward', ter)
117
118         crsh = np.vstack([crsh, [total_crashes]])
119         np.save('crash', crsh)
120         ##print(ter)
121         #print(total_crashes)
122         ##print(crsh)
123
124         #print(ep)
125
126         #print(total_rewards)
127     else:
128         end = datetime.datetime.now()
129         print(end - start)
130         np.save('q', qtable)
131         print(qtable)
132         print('End Training')
133
134         ##print(step)
135
136         print(total_rewards)
137         servo_stop()
```

```
138         STOP()
139
140 except KeyboardInterrupt:
141     print ('Cleaning up')
142     servo_stop()
143     STOP()
```

Πρόγραμμα Εντοπισμού Αντικειμένου

```
1   ############################
2   # Title: Object Detection  #
3   # Author: Tzanakis Ioannis #
4   ############################
5
6   import picamera.array
7   import picamera
8   import time
9   import cv2
10  import numpy as np
11
12  camera = picamera.PiCamera()
13
14  camera.resolution = (640, 480)
15  camera.framerate = 50
16  camera.exposure_mode = 'antishake'
17  camera.meter_mode = 'spot'
18  output = picamera.array.PiRGBArray(camera, size=(640, 480))
19
20  time.sleep(0.5)
21
22  img = cv2.imread("capture7_14.jpg", cv2.IMREAD_GRAYSCALE) # fortwsh
fwtografias antikeimenou
23
24  # Features
25  orb = cv2.ORB_create(nfeatures = 1500, scaleFactor = 1.2, nlevels = 18,
edgeThreshold = 10, firstLevel = 0, WTA_K = 2, scoreType = 0, patchSize =
17, fastThreshold = 40)
26
27  print("-------------------------")
28  retval2 = orb.getScaleFactor()
29  print(retval2)
30  print("-------------------------")
31
32  kp_image, desc_image = orb.detectAndCompute(img, None)
33  desc_image = np.asarray(desc_image,np.float32)
34
35  imgpoints = cv2.drawKeypoints(img, kp_image, None, color=(0,255,0),
flags=0)
36  cv2.imshow("p1", imgpoints)
37
38  # Feature matching
39  # FLANN parameters
40  FLANN_INDEX_LSH = 5
41  index_params= dict(algorithm = FLANN_INDEX_LSH,
42  table_number = 8,
43  key_size = 16,
44  multi_probe_level = 1)
45  search_params = dict(checks = 60)
46  flann = cv2.FlannBasedMatcher(index_params, search_params)
```

```
47
48
49  while True:
50      camframe = camera.capture(output, 'bgr', use_video_port=True)
51      frame = output.array
52      grayframe = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY) # frame image
53
54      kp_grayframe, desc_grayframe = orb.detectAndCompute(grayframe,
None)
55      desc_grayframe = np.asarray(desc_grayframe,np.float32)
56      matches = flann.knnMatch(desc_image, desc_grayframe, k=2)
57
58      good_points = []
59      for m,n in matches:
60          if m.distance < 0.67*n.distance:
61              good_points.append(m)
62      print (len(good_points))
63
64      #img3 = cv2.drawMatches(img, kp_image, grayframe, kp_grayframe,
good_points, grayframe)
65      #cv2.imshow("img3", img3)
66
67      # Homography
68      if len(good_points) > 30:
69          query_pts = np.float32([kp_image[m.queryIdx].pt for m in
good_points]).reshape(-1, 1, 2)
70          train_pts = np.float32([kp_grayframe[m.trainIdx].pt for m in
good_points]).reshape(-1, 1, 2)
71
72          matrix, mask = cv2.findHomography(query_pts, train_pts,
cv2.RANSAC, 4.0)
73          matches_mask = mask.ravel().tolist()
74
75          # Perspective transform
76          h, w = img.shape
77          pts = np.float32([[0, 0], [0, h], [w, h], [w, 0]]).reshape(-1,
1, 2)
78          dst = cv2.perspectiveTransform(pts, matrix)
79
80          homography = cv2.polylines(frame, [np.int32(dst)], True, (255,
0, 0), 3)
81          cv2.imshow("Homography1", homography)
82
83          '''res = cv2.drawMatches(img, kp_image, grayframe,
kp_grayframe, good_points[:40],None,flags=2)
84          cv2.imshow("Homography2", res)'''
85
86      else:
87          gray_points = cv2.drawKeypoints(grayframe, kp_grayframe, None,
color=(0,255,0), flags=0)
88          cv2.imshow("Homography1", gray_points)
89
90      #cv2.imshow("Image", img)
91      #cv2.imshow("grayFrame", grayframe)
92      #cv2.imshow("img3", img3)
93
94      if cv2.waitKey(1) & 0xFF == ord('q'): # ean paththei to plhktro 'q'
termatizei to programma
95          break
96      output.seek(0)
97      output.truncate()
```

```
98
99  cv2.destroyAllWindows()
```

Τελικό Πρόγραμμα

```
1   ####################################
2   # Title: Autonomous Robotic Vehicle #
3   # Author: Tzanakis Ioannis          #
4   ####################################
5
6   import picamera.array
7   import picamera
8   import time
9   import cv2
10  import numpy as np
11  from DistStates import*
12  from DCMotActions import*
13  from pathlib import Path
14
15  R = right()
16  L = left()
17  C = centre()
18  time.sleep(10) # Meta apo 10sec ksekinaei h leitourgia tou robotikou
oxhmatos
19
20  def move(action):
21
22      if action == 0:
23          FORWARD()
24
25      elif action == 1:
26          BACKWARD()
27
28      elif action == 2:
29          FRIGHT()
30
31      elif action == 3:
32          FLEFT()
33
34      elif action == 4:
35          BRIGHT()
36
37      elif action == 5:
38          BLEFT()
39
40  saved_table = Path('q.npy')
41  if saved_table.is_file():
42      qtable = np.load('q.npy')
43  else:
44      print('error')
45
46  print(qtable)
47
48  camera = picamera.PiCamera()
49
50  camera.resolution = (640, 480)
51  camera.framerate = 50
52  camera.exposure_mode = 'antishake'
53  camera.meter_mode = 'spot'
```

```
54  output = picamera.array.PiRGBArray(camera, size=(640, 480))
55  time.sleep(0.5)
56
57  img = cv2.imread("capture7_14.jpg", cv2.IMREAD_GRAYSCALE) # fortwsh
fwtografias antikeimenou
58
59  # Features
60  orb = cv2.ORB_create(nfeatures = 1500, scaleFactor = 1.2, nlevels = 18,
edgeThreshold = 10, firstLevel = 0, WTA_K = 2, scoreType = 0, patchSize =
17, fastThreshold = 40)
61
62  print("-------------------------")
63  retval2 = orb.getScaleFactor()
64  print(retval2)
65  print("-------------------------")
66
67  kp_image, desc_image = orb.detectAndCompute(img, None)
68  desc_image = np.asarray(desc_image,np.float32)
69
70  imgpoints = cv2.drawKeypoints(img, kp_image, None, color=(0,255,0),
flags=0)
71  cv2.imshow("p1", imgpoints)
72
73  # Feature matching
74  # FLANN parameters
75  FLANN_INDEX_LSH = 5
76  index_params= dict(algorithm = FLANN_INDEX_LSH,
77  table_number = 8,
78  key_size = 16,
79  multi_probe_level = 1)
80  search_params = dict(checks = 60)
81  flann = cv2.FlannBasedMatcher(index_params, search_params)
82
83
84  while True:
85      camframe = camera.capture(output, 'bgr', use_video_port=True)
86      frame = output.array
87      grayframe = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY) # frame image
88
89      kp_grayframe, desc_grayframe = orb.detectAndCompute(grayframe,
None)
90      desc_grayframe = np.asarray(desc_grayframe,np.float32)
91      matches = flann.knnMatch(desc_image, desc_grayframe, k=2)
92
93      good_points = []
94      for m,n in matches:
95          if m.distance < 0.67*n.distance:
96              good_points.append(m)
97      #print (len(good_points))
98
99      #img3 = cv2.drawMatches(img, kp_image, grayframe, kp_grayframe,
good_points, grayframe)
100     #cv2.imshow("img3", img3)
101
102     # Homography
103     if len(good_points) > 30:
104         C=centre()
105         query_pts = np.float32([kp_image[m.queryIdx].pt for m in
good_points]).reshape(-1, 1, 2)
106         train_pts = np.float32([kp_grayframe[m.trainIdx].pt for m in
good_points]).reshape(-1, 1, 2)
```

```
107         #print('@@@@@@@@@@@@@')
108         #print(query_pts)
109         '''print('--------')
110         print(train_pts)
111         print('----------')'''
112 
113         matrix, mask = cv2.findHomography(query_pts, train_pts,
cv2.RANSAC, 4.0)
114         matches_mask = mask.ravel().tolist()
115 
116         # Perspective transform
117         h, w = img.shape
118         pts = np.float32([[0, 0], [0, h], [w, h], [w, 0]]).reshape(-1,
1, 2)
119         dst = cv2.perspectiveTransform(pts, matrix)
120         '''print('$$$$$$$')
121         print(pts)'''
122         '''print('-------')
123         print(dst)
124         print('$$$$$$$')'''
125 
126         homography = cv2.polylines(frame, [np.int32(dst)], True, (255,
0, 0), 3)
127         cv2.imshow("Homography1", homography)
128 
129         '''res = cv2.drawMatches(img, kp_image, grayframe,
kp_grayframe, good_points[:40],None,flags=2)
130         cv2.imshow("Homography2", res)'''
131 
132         if C > 15:
133             FORWARD2()
134         else:
135             STOP2()
136     else:
137         state = states()
138         action = np.argmax(qtable[state,:])
139         take_act = move(action)
140 
141         '''gray_points = cv2.drawKeypoints(grayframe, kp_grayframe,
None, color=(0,255,0), flags=0)
142         cv2.imshow("Homography1", gray_points)'''
143 
144     #cv2.imshow("Image", img)
145     #cv2.imshow("grayFrame", grayframe)
146     #cv2.imshow("img3", img3)
147 
148     if cv2.waitKey(1) & 0xFF == ord('q'): # ean paththei to plhktro 'q'
termatizei to programma
149         servo_stop()
150         STOP()
151         break
152 
153     output.seek(0)
154     output.truncate()
155 
156 cv2.destroyAllWindows()
```

Πρόγραμμα VL53L0X

Πρόγραμμα από το πακέτο VL53L0X_rasp_python που χρησιμοποιείται στο αυτόνομο ρομποτικό όχημα.

```
1   #!/usr/bin/python
2
3   # MIT License
4   #
5   # Copyright (c) 2017 John Bryan Moore
6   #
7   # Permission is hereby granted, free of charge, to any person obtaining
a copy
8   # of this software and associated documentation files (the "Software"),
to deal
9   # in the Software without restriction, including without limitation the
rights
10  # to use, copy, modify, merge, publish, distribute, sublicense, and/or
sell
11  # copies of the Software, and to permit persons to whom the Software is
12  # furnished to do so, subject to the following conditions:
13  #
14  # The above copyright notice and this permission notice shall be
included in all
15  # copies or substantial portions of the Software.
16  #
17  # THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR
18  # IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY,
19  # FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT
SHALL THE
20  # AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR
OTHER
21  # LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE,
ARISING FROM,
22  # OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER
DEALINGS IN THE
23  # SOFTWARE.
24
25  import time
26  from ctypes import *
27  import smbus
28
29  VL53L0X_GOOD_ACCURACY_MODE      = 0   # Good Accuracy mode
30  VL53L0X_BETTER_ACCURACY_MODE    = 1   # Better Accuracy mode
31  VL53L0X_BEST_ACCURACY_MODE      = 2   # Best Accuracy mode
32  VL53L0X_LONG_RANGE_MODE         = 3   # Longe Range mode
33  VL53L0X_HIGH_SPEED_MODE         = 4   # High Speed mode
34
35  i2cbus = smbus.SMBus(1)
36
37  # i2c bus read callback
38  def i2c_read(address, reg, data_p, length):
39      ret_val = 0;
40      result = []
41
42      try:
43          result = i2cbus.read_i2c_block_data(address, reg, length)
```

```
44          except IOError:
45              ret_val = -1;
46
47          if (ret_val == 0):
48              for index in range(length):
49                  data_p[index] = result[index]
50
51          return ret_val
52
53      # i2c bus write callback
54      def i2c_write(address, reg, data_p, length):
55          ret_val = 0;
56          data = []
57
58          for index in range(length):
59              data.append(data_p[index])
60          try:
61              i2cbus.write_i2c_block_data(address, reg, data)
62          except IOError:
63              ret_val = -1;
64
65          return ret_val
66
67      # Load VL53L0X shared lib
68      tof_lib = CDLL("../bin/vl53l0x_python.so")
69
70      # Create read function pointer
71      READFUNC = CFUNCTYPE(c_int, c_ubyte, c_ubyte, POINTER(c_ubyte),
c_ubyte)
72      read_func = READFUNC(i2c_read)
73
74      # Create write function pointer
75      WRITEFUNC = CFUNCTYPE(c_int, c_ubyte, c_ubyte, POINTER(c_ubyte),
c_ubyte)
76      write_func = WRITEFUNC(i2c_write)
77
78      # pass i2c read and write function pointers to VL53L0X library
79      tof_lib.VL53L0X_set_i2c(read_func, write_func)
80
81      class VL53L0X(object):
82          """VL53L0X ToF."""
83
84          object_number = 0
85
86          def __init__(self, address=0x29, TCA9548A_Num=255, TCA9548A_Addr=0,
**kwargs):
87              """Initialize the VL53L0X ToF Sensor from ST"""
88              self.device_address = address
89              self.TCA9548A_Device = TCA9548A_Num
90              self.TCA9548A_Address = TCA9548A_Addr
91              self.my_object_number = VL53L0X.object_number
92              VL53L0X.object_number += 1
93
94          def start_ranging(self, mode = VL53L0X_GOOD_ACCURACY_MODE):
95              """Start VL53L0X ToF Sensor Ranging"""
96              tof_lib.startRanging(self.my_object_number, mode,
self.device_address, self.TCA9548A_Device, self.TCA9548A_Address)
97
98          def stop_ranging(self):
99              """Stop VL53L0X ToF Sensor Ranging"""
100             tof_lib.stopRanging(self.my_object_number)
```

```
101
102     def get_distance(self):
103         """Get distance from VL53L0X ToF Sensor"""
104         return tof_lib.getDistance(self.my_object_number)
105
106     # This function included to show how to access the ST library
directly
107     # from python instead of through the simplified interface
108     def get_timing(self):
109         Dev = POINTER(c_void_p)
110         Dev = tof_lib.getDev(self.my_object_number)
111         budget = c_uint(0)
112         budget_p = pointer(budget)
113         Status =
tof_lib.VL53L0X_GetMeasurementTimingBudgetMicroSeconds(Dev, budget_p)
114         if (Status == 0):
115             return (budget.value + 1000)
116         else:
117             return 0
```

Πρόγραμμα Διαγράμματος

Παράδειγμα προγράμματος για την δημιουργία διαγράμματος ανταμοιβών.

```
1  ############################
2  # Title: Plot of Rewards   #
3  # Author: Tzanakis Ioannis #
4  ############################
5
6  import numpy as np
7  import matplotlib.pyplot as plt
8
9  x = np.load('numbers.npy') # pinakas me times 1-100
10 y = np.load('rewardt5.npy') # pinakas me ta reward se kathe episode
11
12 plt.grid(True)
13 plt.xlabel('Episodes')
14 plt.ylabel('Rewards')
15 plt.plot(x, y, 'bo-')
16
17 # ektypwsh ths grafikhs
18 plt.show()
19 print(y)
20
21 # ektypwsh tou pinaka me ta reward kai sum twn timwn
22 b = np.sum(y,axis=0)
23 print(b)
24
25 # ektypwsh sum
26 c = np.sum(y)
27 print(c)
```

Παράρτημα Β

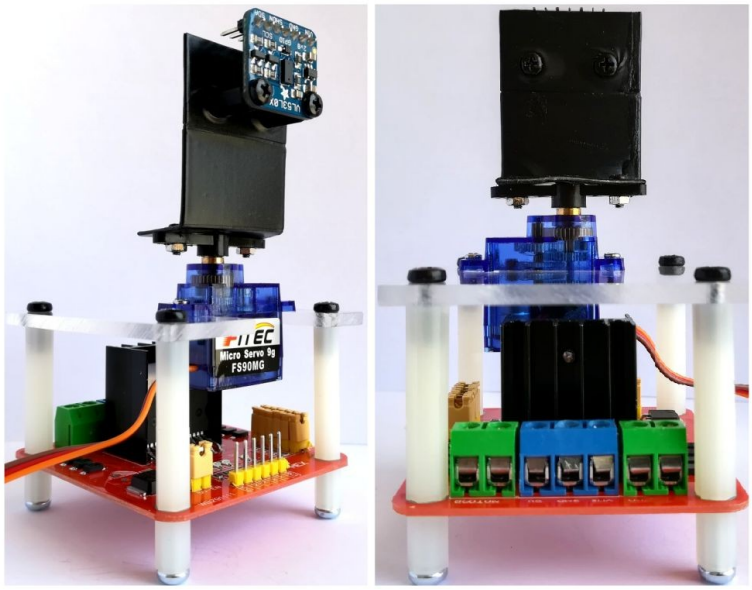
Πίνακας Β.1: Εξαρτήματα Αυτόνομου Ρομποτικού Οχήματος

Εξάρτημα	Ποσότητα
Σασί	1
DC Motors	2
Τροχοί	2
Ball Caster	1
Μπαταριοθήκη 4χ AA	1
Μπαταριοθήκη 1χ AA	1
Επαναφορτιζόμενη Μπαταρία AA	5
Power Bank	1
Καλώδιο USB - Micro USB	1
Σερβοκινητήρας	1
Servo arm	1
Επιφάνεια Plexiglass 3mm	4
Raspberry Pi 3 Model B	1
Raspberry Pi Camera V2	1
VL53L0X	1
Πλακέτας Οδήγησης L298N	1
Ανεμιστήρας 5V DC	1
Ψύκτρες	2
Πυκνωτής 470μF 25V	1
Micro SD 32 GB	1
Βάση Στήριξης VL53L0X	1
Βάση Στήριξης Raspberry Pi Camera V2	1
Καλώδια	• 11× F-F 200 mm • 6 × M-M 100 mm • 1 × F-M 200 mm • 1 × 15 way Flexible Flat Cable (*FFC*) 150 mm
Αποστάτες	• 12 × M3 Πλαστικός F/F L30 mm • 4 × M3 Πλαστικός F/F L20 mm • 2 × M3 Πλαστικός F/F L10 mm • 6 × M3 Πλαστικός M/F L30 mm • 12 × M3 Πλαστικός M/F L10 mm

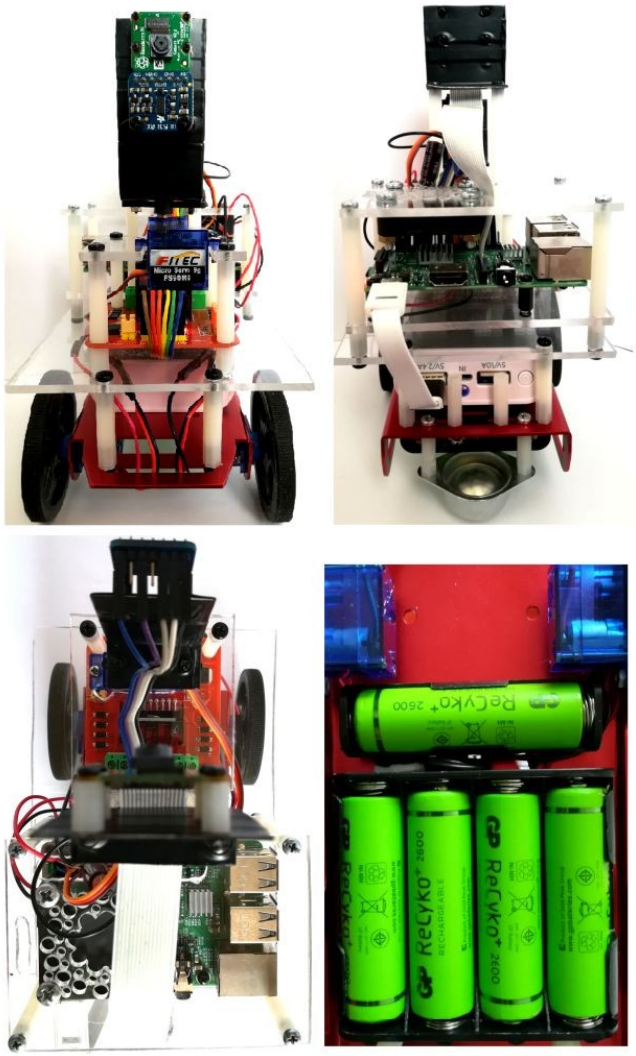
	• 4 × M2.5 Πλαστικός F/F L10 mm • 2 × M2.5 Πλαστικός F/F L5 mm • 4 × M2 Πλαστικός F/F L8 mm
Βίδες	• 4 × M4 L20 mm • 18 × M3 L10 mm • 12 × M3 L8 mm • 4 × M3 L5 mm • 4 × M2.5 L10 mm • 8 × M2.5 L5 mm • 1 × M2.5 L4 mm • 2 × M2 L10 mm • 4 × M2 L8mm • 10 × M2 L5 mm • 2 × PB 2.0 L8 mm
Παξιμάδια	• 8 × M2 Μεταλλικό H1.6 mm • 4 × M4 Μεταλλικό

Σχήμα Β.1: Διάφορα εξαρτήματα του ρομποτικού οχήματος.

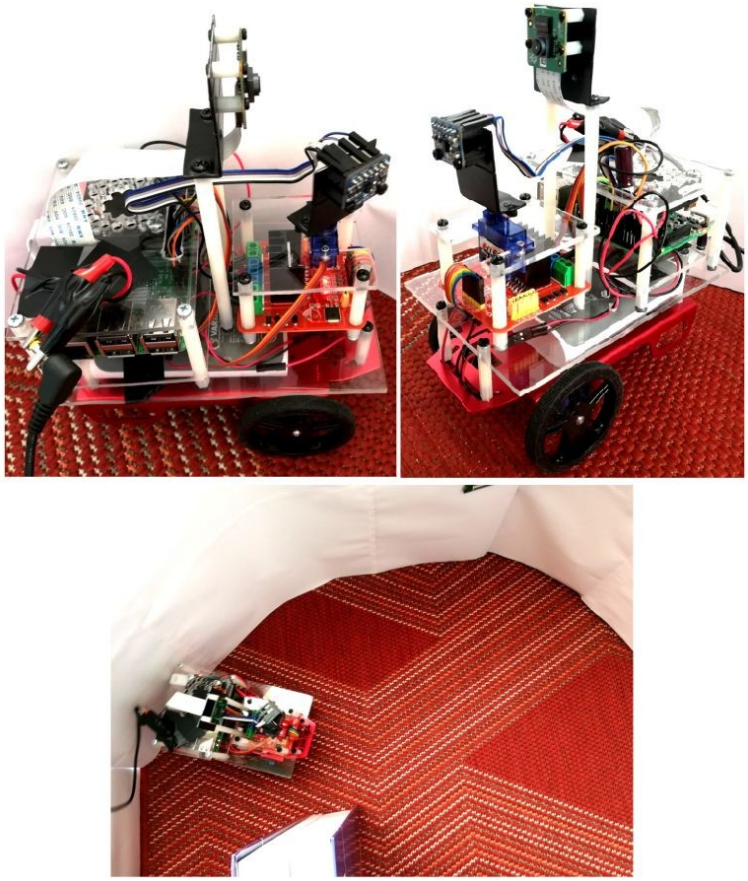
Σχήμα Β.2: Η θήκη με το Raspberry Pi.



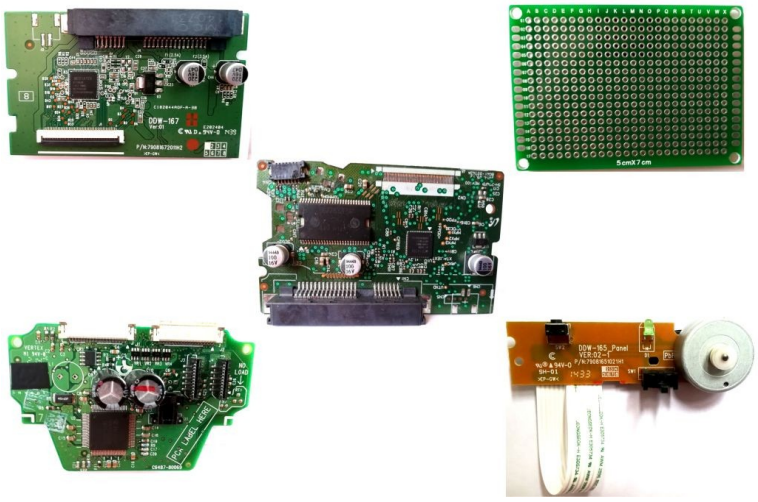
Σχήμα Β.3: Συναρμολόγηση εξαρτημάτων.



Σχήμα Β.4: Ολοκλήρωση της συναρμολόγησης του αυτόνομου ρομποτικού οχήματος.



Σχήμα Β.5: Εκπαίδευση του ρομποτικού οχήματος με την τροφοδοσία των DC κινητήρων να προέρχεται από τροφοδοτικό 5V DC 2000mA.



Σχήμα Β.6: Οι πλακέτες που χρησιμοποιούνται στα πειράματα.