Σχεδιασμός τροχιάς και αποφυγή εμποδίων με χρήση βαθιάς ενισχυτικής μάθησης

Πολυτεχνείο Κρήτης
Σχολή Μηχανικών Παραγωγής και Διοίκησης

Σκουλάξινος Παναγιώτης - Χαράλαμπος

A.M 2017010035

Δεκέμβριος 2022

Εξεταστική Επιτροπή

Δρ. Ελευθέριος Δοϊτσίδης, *Επίκουρος Καθηγητής*	Σχολή Μηχανικών Παραγωγής και Διοίκησης, Πολυτεχνείο Κρήτης
Δρ. Νικόλαος Τσουρβελούδης *Καθηγητής*	Σχολή Μηχανικών Παραγωγής και Διοίκησης, Πολυτεχνείο Κρήτης
Δρ. Σάββας Πιπερίδης, *Ε.ΔΙ.Π*	Σχολή Μηχανικών Παραγωγής και Διοίκησης, Πολυτεχνείο Κρήτης

Ευχαριστίες

Με την ολοκλήρωση αυτής της διπλωματικής εργασίας, κλείνει ένα σημαντικό κεφάλαιο της ζωής μου, διάρκειας πέντε ετών. Αρχικά, θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή μου, τον Δρ. Δοϊτσίδη Ελευθέριο για την καθοδήγηση και τις συμβουλές που μου προσέφερε και το Εργαστήριο Ευφυών Συστημάτων και Ρομποτικής για την παροχή του κατάλληλου εξοπλισμού και εργαλείων για την υλοποίηση αυτής της εργασίας.

Θα ήθελα επιπλέον να ευχαριστήσω την οικογένειά μου για τη στήριξη που μου προσέφεραν και τις θυσίες που έκαναν κατά τη διάρκεια των σπουδών μου στην πόλη των Χανίων.

Τέλος, ευχαριστώ τον παιδικό μου φίλο, συγκάτοικο και συνάδελφο Μπιρμπουτσάκη Χρήστο για την πολύτιμη βοήθειά του όλα αυτά τα χρόνια των σπουδών μας.

Περιεχόμενα

Περίληψη

Σκοπός της παρούσας εργασίας είναι η υλοποίηση μεθόδου σχεδιασμού τροχιάς και αποφυγής εμποδίων για το ρομποτικό όχημα Turtlebot 3 Burger, με χρήση βαθιάς ενισχυτικής μάθησης. Αναπτύχθηκε μια προσέγγιση 2 επιπέδων, όπου αρχικά χρησιμοποιείται ένα αλγόριθμος ευρετικής αναζήτησης για το σχεδιασμό του βέλτιστου μονοπατιού και στη συνέχεια χρησιμοποείται ένα κατάλληλο εκπαιδευμένο νευρωνικό δίκτυο προκειμένου να πλοηγήσει το ρομποτικό όχημα λαμβάνοντας υπόψιν την προκαθορισμένη τροχιά αποφεύγοντας παράλληλα τυχόν εμπόδια. Η υλοποίηση της συγκεκριμένης προσέγγισης πραγματοποιήθηκε στο περιβάλλον Gazebo με χρήση του Robotic Operating System (ROS), ενώ για την εκπαίδευση του νευρωνικού δικτύου χρησιμοποιήθηκε η μηχανικής μάθησης Keras.

Abstract

The purpose of this thesis is the development and implementation of a path planning and obstacle avoidance method for the Turtlebot 3 Burger robotic vehicle, using deep reinforcement learning, in the ROS environment. The goal is the development of a two stage approach, where a heuristic search algorithm is used for the design of the best path and the robotic vehicle is following the predefined path using a neural network which was trained, using the Keras machine learning library. The application and evaluation of the proposed approach was performed using the Gazebo robotic simulator environment.

Κεφάλαιο 1

Εισαγωγή

Ο τομέας της ρομποτικής τα τελευταία χρόνια εξελίσσεται με ταχείς ρυθμούς, και με την πρόοδο της τεχνολογίας, αυτή γίνεται ολοένα και πιο προσβάσιμη όχι μόνο σε επαγγελματικό αλλά και σε οικιακό επίπεδο, επιτρέποντας έτσι την πραγματοποίηση περισσοτέρων εργασιών που υπό άλλες συνθήκες τις πραγματοποιούσε ο άνθρωπος. Οι συνεχώς αυξανόμενες ανάγκες για τη χρήση αυτόνομων ρομποτικών οχημάτων, έκανε επιτακτική την ανάπτυξη της ευφυΐας αυτών των συστημάτων, ώστε να είναι ακόμη πιο αποτελεσματικά στις εργασίες τους, εξοικονομώντας κόπο, χρόνο και χρήμα από τον άνθρωπο που τα χειρίζεται.

Ένας από τους σημαντικότερους παράγοντες που καθορίζουν τον τρόπο λειτουργίας των αυτόνομων οχημάτων είναι η δυνατότητα να πλοηγηθούν αυτόνομα δίχως την παρέμβαση ανθρώπου χειριστή. Το ρομπότ πρέπει να διαθέτει την ικανότητα να κινείται εντός ενός περιβάλλοντος, γνωστού ή μη, αποφεύγοντας τυχόν εμπόδια που συναντάει στη διαδρομή του και να αφιχθεί με επιτυχία στη θέση στόχο του. Για να διεθέτει αυτή την ικανότητα, το σύστημα πρέπει να είναι σε θέση να γνωρίζει τη θέση του στο χώρο, τη θέση των περιβάλλοντων αντικειμένων και να διαθέτει την κατάλληλη δομή ώστε να μπορεί να κινηθεί με ικανοποιητικούς ρυθμούς στο περιβάλλον στο οποίο βρίσκεται.

Ένας επιπλέον παράγοντας πέραν της ικανότητας αποφυγής εμποδίων και της πλοήγησης προς ένα σημείο ενδιαφέροντος είναι ο σχεδιασμός της τροχιάς που καλείται να ακολουθήσει. Αυτή πρέπει να είναι τέτοια, ώστε να μην καθοδηγεί το ρομπότ σε κάποιο εμπόδιο, ενώ παράλληλα να είναι αποτελεσματική, δηλαδή να είναι όσο το δυνατόν πιο κοντά στην ελάχιστη δυνατή απόσταση που πρέπει να διανύσει το όχημα ώστε να φτάσει στη θέση στόχο του. Αυτό επιτυγχάνεται μέσω κατάλληλων αλγορίθμων, οι οποίοι λαμβάνουν πληροφορίες για το χώρο από το σύστημα και λαμβάνουν αποφάσεις αναλόγως.

Στόχος αυτής της διπλωματικής εργασίας είναι η υλοποίηση ενός ολοκληρωμένου συστήματος σχεδιασμού τροχιάς και αποφυγής εμποδίων για τη ρομποτική συσκευή Turtlebot 3 Burger. Προκειμένου να επιτευχθεί ο συγκεκριμένος στόχος, αναπτύχθηκε ένα σύστημα πλοήγησης με χρήση τεχνητών νευρωνικών δικτύων και βαθιάς ενισχυτικής μάθησης και συνδιάστηκε με ευρετικούς αλγορίθμους εύρεσης της βέλτιστης διαδρομής, προκειμένου η ρομποτική συσκευή να μπορεί να κατευθυνθεί στο επιθυμητό σημείο στόχο με βέλτιστο τρόπο. Η υλοποίηση έγινε με χρήση του Robot Operating System και οι προσομοιώσεις πραγματοποιήθηκαν στο περιβάλλον Gazebo.

Η εργασία απότελείται από 6 κεφάλαια, όπου στο δεύτερο κεφάλαιο πραγματοποιείται μια βιβλιογραφική επισκόπηση του θεωρητικού υποβάθρου, πάνω στο οποίο βασίζεται το αντικείμενο αυτής της εργασίας. Συγκεκριμένα, γίνονται αναφορές στους ευρετικούς αλγορίθμους εύρεσης συντομότερης διαδρομής (A*/CIA*) που χρησιμοποιούνται, στα τεχνητά νευρωνικά δίκτυα, στη βαθιά ενισχυτική μηχανική μάθηση και τους αλγορίθμους που την υλοποιούν. Επιπλέον,

περιγράφεται το κινηματικό μοντέλο διαφορικής οδήγησης στο οποίο βασίζεται η ρομποτική πλατφόρμα που χρησιμοποιείται, αναφέρεται ο τρόπος λειτουργίας του αισθητήρα λέιζερ για την ανίχνευση εμποδίων και η αρχή λειτουργίας της οδομετρίας. Τέλος, αναφέρονται κάποιες ανάλογες υλοποιήσεις παρόμοιων προβλημάτων στη ρομποτική.

Στο τρίτο κεφάλαιο, περιγράφονται τα εργαλεία που χρησιμοποιήθηκαν, και συγκεκριμένα η ρομποτική πλατφόρμα Turtlebot 3 Burger, ενώ από λογισμικά χρησιμοποιήθηκαν το Robot Operating System και το Keras, μέρος της βιβλιοθήκης TensorFlow.

Στο τέταρτο κεφάλαιο, περιγράφεται η μεθοδολογία υλοποίησης. Αναλύεται ο αλγόριθμος Deep Deterministic Policy Gradient (DDPG) που χρησιμοποιήθηκε και ο τρόπος με τον οποίο υλοποιήθηκε στη γλώσσα προγραμματισμού Python. Επίσης αναφέρεται τόσο η υλοποίηση των εν λόγω ευρετικών αλγορίθμων, αλλά και η μέθοδος με την οποία συνδέεται με το σύστημα αποφυγής εμποδίων βαθιάς ενισχυτικής μάθησης. Τέλος, αναφέρεται η υλοποίηση των εν λόγω αλγορίθμων ευρετικής αναζήτησης.

Στο πέμπτο κεφάλαιο πραγματοποιούνται όλες οι δοκιμές και τα πειράματα της υλοποίησης. Δοκιμάζεται το σύστημα σε 3 περιβάλλοντα, ένα γνωστό και δύο άγνωστα, και αναλύονται τα πειραματικά αποτελέσματα λαμβάνοντας υπόψιν παραμέτρους όπως ο χρόνος, η συνολική διανυθείσα απόσταση, οι συνολικές συγκρούσεις αλλά και η τροχιά του οχήματος. Τα δεδομένα συγκεντρώνονται και πραγματοποιούνται συγκρίσεις μεταξύ αυτών.

Στο έκτο κεφάλαιο πραγματοποιείται μια επισκόπηση των παραπάνω πληροφοριών, αξιολογείται η αποτελεσματικότητα της μεθόδου, εξάγονται συμπεράσματα και προτείνονται διορθώσεις αλλά και μελλοντικές επεκτάσεις.

Κεφάλαιο 2

Θεωρητικό Υπόβαθρο

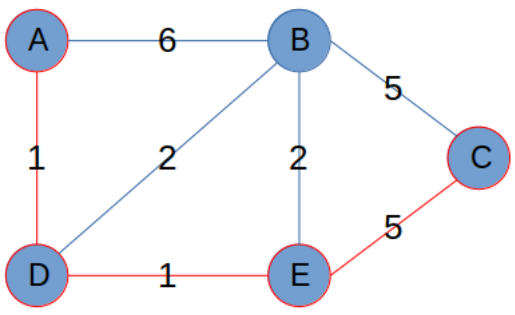
2.1 Αλγόριθμοι σχεδιασμού βέλτιστου μονοπατιού

Η εύρεση της βέλτιστης μονοπατιού (τροχιάς) είναι ιδιαίτερα σημαντική στη περίπτωση της αυτόνομης πλοήγησης, καθώς με χρήση των σχετικών μεθοδολογιών τα οχήματα έχουν τη δυνατότητα να κινηθούν προς ένα σημείο στόχο, βελτιστοποιώντας τη συμπεριφορά τους, λαμβάνοντας υπόψη κριτήρια όπως η κατανάλωση ενέργειας, το ελάχιστο μήκος της διαδρομής, η ελάχιστη χρονική διάρκεια κ.α. Προκειμένου να επιτευχθεί το παραπάνω αποτέλεσμα, έχουν υιοθετηθεί στη βιβλιογραφία μια σειρά από μεθοδολογίες. Ενδεικτικά θα αναφέρουμε τους αλγορίθμους Dijkstra, A* και CIA* που χρησιμοποιήθηκαν στην παρούσα εργασία.

2.1.1 Ο αλγόριθμος Dijkstra

Ο αλγόριθμος του Dijkstra είναι ένας αλγόριθμος εύρεσης της συντομότερης διαδρομής που συνδέει δύο κόμβους ενός γράφου [1]. Η βασική αρχή λειτουργίας του αλγορίθμου βασίζεται στην παρακάτω φιλοσοφία. Ξεκινώντας από τον αρχικό κόμβο, στον οποίο θέτουμε την απόσταση ίση με μηδέν, υπολογίζουμε την απόσταση κάθε γειτονικού του κόμβου και επιλέγουμε να μεταβούμε σε αυτόν με τη μικρότερη. Αν ο κόμβος έχει ήδη ορισμένη κάποια απόσταση, τότε η νέα απόσταση συγκρίνεται με την προηγούμενη και εφόσον είναι μικρότερη αντικαθίσταται. Η διαδικασία αυτή επαναλαμβάνεται μέχρις ότου να μεταβεί στον τελικό κόμβο, από τον οποίο προκύπτει και η συνολική απόσταση.

Στο Σχήμα 2.1 φαίνεται ένα παράδειγμα εφαρμογής αυτού του αλγορίθμου. Εφαρμόζεται σε



Κόμβος	Ελάχιστη απόσταση από Α	Προηγ. κόμβος
A	0	-
B	3	D
C	7	E
D	1	A
E	2	D

Σχήμα 2.1: Παράδειγμα εφαρμογής του αλγορίθμου του Dijkstra, σε ένα γράφημα 5 κόμβων

ένα γράφημα με 5 κόμβους και 7 ακμές, πάνω στις οποίες αναγράφεται η απόσταση μεταξύ των 2 κόμβων. Υπολογίζοντας πόσο απέχει ο κάθε κόμβος από τον κόμβο A και κρατώντας την ελάχιστη δυνατή απόσταση, με στόχο την αφίξη στον κόμβο C, ο οποίος είναι και ο τελικός. Η συντομότερη διαδρομή σε αυτό το παράδειγμα είναι η A - D - E - C με συνολική απόσταση 7.

2.1.2 Ο αλγόριθμος A*

Ο A* αποτελεί έναν αλγόριθμο ευρετικής αναζήτησης με στόχο την εύρεση της συντομότερης διαδρομής από έναν αρχικό κόμβο σε έναν τελικό. Πρόκειται για μια γενίκευση του αλγορίθμου του Dijkstra. Χρησιμοποιείται σε διάφορους τομείς, συμπεριλαμβανομένης και της ρομποτικής, όπου και χρησιμοποιήθηκε πρωταρχικά [2]. Χρησιμοποιεί μια συνάρτηση f, μέσω της οποίας επιλέγεται ο επόμενος κόμβος στον οποίο θα επεκταθεί, η οποία εκτιμάται ως εξής:

$$f(x) = g(x) + h(x), \tag{2.1}$$

όπου $g(x)$ το κόστος της συντομότερης διαδρομής από τον αρχικό κόμβο στον τρέχων κόμβο x και $h(x)$ το εκτιμώμενο κόστος της συντομότερης διαδρομής από το x στον τελικό κόμβο, το οποίο υπολογίζεται είτε με ευκλείδια απόσταση από τον τρέχων κόμβο μέχρι τον τελικό, είτε με απόσταση Manhattan. Μετά την εκτέλεση του αλγορίθμου, προκύπτει η τελική (βέλτιστη) συνάρτηση $f^*(x)$, η οποία υπολογίζεται ως εξής:

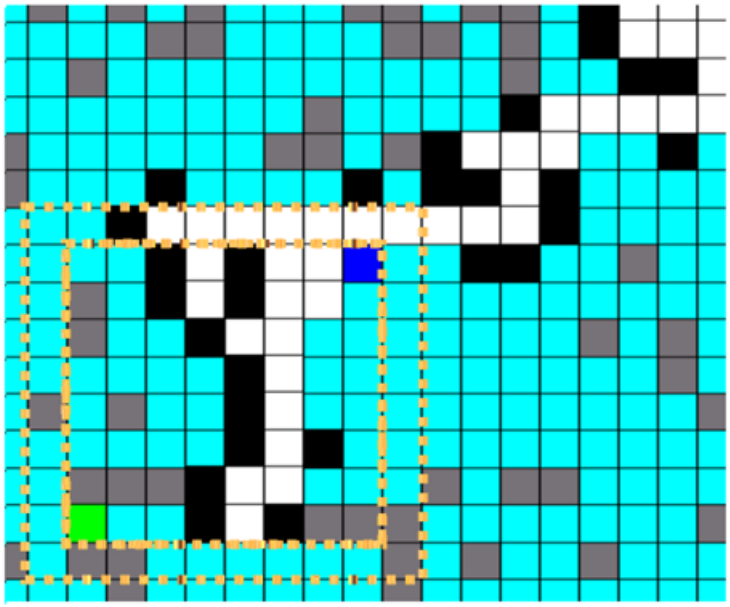
$$f^*(x) = g^*(x) + h^*(x). \tag{2.2}$$

Εφόσον ικανοποιείται η συνθήκη $h(x) < h^*(x)$ για όλους τους κόμβους x, αποδεικνύεται ότι η απόσταση που υπολογίστηκε πράγματι είναι η βέλτιστη. Πλέον των παραπάνω, η εφαρμογή του αλγορίθμου εκμεταλλεύεται και δύο λίστες, που ονομάζονται OPEN και CLOSED. Η πρώτη λίστα αποθηκεύει τους κόμβους που βρίσκονται στα σύνορα της αναζήτησης, δηλαδή τους υποψήφιους προς επίσκεψη κόμβους στο επόμενο βήμα, και η δεύτερη διατηρεί τους κόμβους που έχουν ήδη προσπελαστεί. Σε κάθε βήμα της επανάληψης αφαιρείται από την OPEN ο κόμβος με την μίκροτερη τιμή της συνάρτησης f και αυτός αποθηκεύεται στην λίστα CLOSED. Ο αλγόριθμος τερματίζεται είτε όταν αφαιρεθεί ο τελικός κόμβος - στόχος, είτε όταν δεν υπάρχουν άλλοι διαθέσιμοι κόμβοι στη λίστα OPEN. Στην πρώτη συνθήκη τερματισμού έχει επιτευχθεί η εύρεση της συντομότερης διαδρομής μεταξύ του αρχικού και του τελικού κόμβου, ενώ στη δεύτερη ο αλγόριθμος απέτυχε.

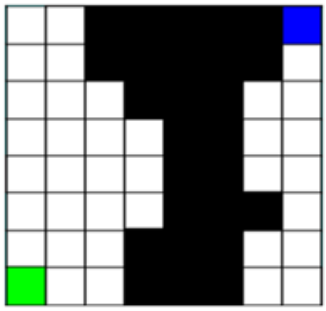
2.1.3 Ο αλγόριθμος CIA*

Ο CIA* αποτελεί μια βελτιωμένη εκδοχή του A*, προσφέροντας ανώτερη αλγοριθμική απόδοση, φτάνοντας σε συμπέρασμα με μικρότερη εξερεύνυση. Η ειδοποιός διαφορά του από τον πρωτότυπο A* είναι η αναθεώρηση του ευρετικού στελέχους του $h^*(x)$, το οποίο εκτιμά την απόσταση από το στόχο του.

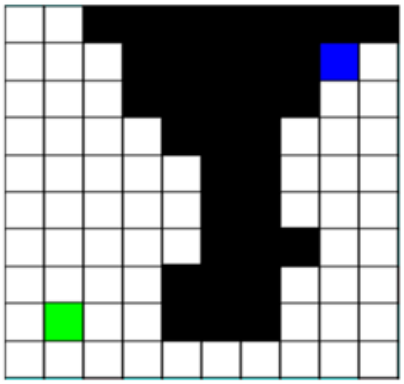
Αντί να χρησιμοποιεί μια σταθερή ευρετική συνάρτηση, όπως ο A*, εκμεταλλεύεται τη γνώση των θέσεων των εμποδίων και την αξιοποιεί για την ενημέρωση της απόστασης των κόμβων από το στόχο [3]. Συγκεκριμένα, εκτός των δύο συνόλων OPEN και CLOSED που ορίζονται στον A*, ορίζεται ένα επιπλέον σύνολο OBST, το οποίο διατηρεί την θέση τον εντοπισμένων εμποδίων στο χώρο κίνησης. Αυτό το σύνολο, σε συνδυασμό με το σύνολο CLOSED, ενώνονται και ορίζουν το σύνολο BLOCKED, με τη βοήθεια του οποίου ελέγχεται εάν εμποδίζεται η διέλευση μεταξύ του τρέχοντος κόμβου και του τελικού κόμβου. Για να επιτευχθεί αυτό, ορίζεται επίσης ένα ορθογώνιο μεταξύ του τρέχοντος κόμβου και του κόμβου στόχου, στο οποίο ελέγχεται αν υπάρχει ελεύθερη διαδρομή μεταξύ των δύο αυτών κόμβων και εφόσον δεν



(α′) Το συνολικό έδαφος, με τις γνωστές πληροφορίες στο διακεκκομένο πλαίσιο



(β′) Αναζήτηση με μηδενικό offset, δεν υπάρχει προσπελάσιμη διαδρομή



(γ′) Αναζήτηση με offset αυξημένο κατά μια μονάδα, οι κόμβοι είναι πλέον συνδεδεμένοι

Σχήμα 2.2: Ανανέωση της ευρετικής τιμής του αλγορίθμου CIA* [3]

υπάρχει επεκτείνεται μέχρι κάποιων ορίων ώστε να βρεθεί. Η ευρετική συνάρτηση ενημερώνεται ανάλογα με το πόσο επεκτάθηκε αυτό το ορθογώνιο.

Για την αποφυγή κινδύνων παραδοχής, ο αλγόριθμος επανεξετάζει και ενημερώνει τις τιμές του ευρετικού σκέλους του κόμβου με το ελάχιστο κόστος στο σύνολο OPEN. Αυτό συμβαίνει καθώς υπάρχουν μεταβολές στην τιμή της ευρετικής συνάρτησης με την ένταξη νέων εμποδίων, γεγονός το οποίο εάν δεν είχε προβλεφθεί θα επέφερε ένα ¨άδικο' πλεονέκτημα στους πρώτους κόμβους [3].

2.2 Τεχνητά νευρωνικά δίκτυα

Τα τεχνητά νευρωνικά δίκτυα αποτελούν δίκτυα νευρώνων τα οποία συνδέονται μεταξύ τους με ανάλογο τρόπο με αυτόν που συναντάται στους νευρώνες ενός βιολογικού εγκεφάλου. Στόχος τους είναι να διεγείρονται με κάποια δεδομένα και να εξάγουν κάποια πληροφορία σχετικά με αυτά, με στόχο την υποβοήθηση σύνθετων αλγορίθμων. Εφαρμόζονται συχνά στον τομέα της τεχνητής νοημοσύνης [4].

2.2.1 Βασικά χαρακτηριστικά νευρωνικών δικτύων

Νευρώνες

Κατ' αντιστοιχία με τους νευρώνες ενός βιολογικού εγκεφάλου, οι τεχνητοί νευρώνες δέχονται μια πληροφορία και την μεταδίδουν στον επόμενο. Αυτή η πληροφορία έχει συνήθως τη μορφή πραγματικών αριθμών. Επιπλέον, ο νευρώνας που λαμβάνει την πληροφορία μπορεί πριν την εξάγει να την επεξεργάζεται ή ακόμη και να εφαρμόζει κάποια συνάρτηση ενεργοποίησης.

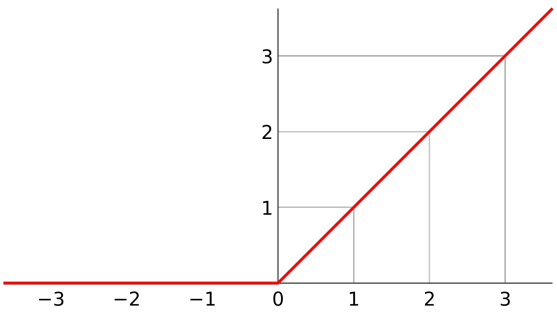
Συνάρτηση ενεργοποίησης

Η συνάρτηση ενεργοποίησης βρίσκει χρήση στη φραγή της τιμής εξόδου ενός τεχνητού νευρώνα, εντός ενός πεδίου τιμών και να προσφέρει την ομαλή μεταβολή της τιμής εξόδου, καθώς αυτή μεταβάλλεται. Μερικά παραδείγματα συχνά χρησιμοποιούμενων συναρτήσεων ενεργοποίησης είναι η βηματική, η γραμμική, η ReLU (Rectifier Linear Unit), η σιγμοειδής, η υπερβολική εφαπτομένη, η softmax, παραλλαγές αυτών αλλά και διάφορες προσαρμοσμένες στην εκάστοτε εφαρμογή.

Ενδεικτικά, η ReLU βρίσκει πολύ συχνά χρήση σε εφαρμογές νευρωνικών δικτύων και χρησιμοποιείται και στα πλαίσια αυτής της εργασίας, όπως περιγράφεται σε παρακάτω κεφάλαιο. Η μαθηματική της έκφραση είναι η εξής:

$$f(x) = x^+ = max(0, x) = \begin{cases} x, & x \geq 0 \\ 0, & x < 0 \end{cases} \tag{2.3}$$

Περιγραφικά, η ReLU διατηρεί την είσοδο x του νευρωνικού δικτύου, εφόσον αυτή είναι θετική. Γραφικά έχει τη μορφή συνάρτησης ράμπας, όπως φαίνεται στο Σχήμα 2.3. Αναλογικά, στην ηλεκτρονική μηχανική αυτή η συνάρτηση λέγεται ανορθωτής μισού κύματος, καθώς κατ' αυτόν τον τρόπο λειτουργεί η αντίστοιχη συσκευή, υπεύθυνη για τη μετατροπή εναλασσόμενου ρεύματος σε συνεχές. Υπάρχουν διάφορες παραλλαγές της ίδιας της ReLU, οι οποίες στοχεύουν στην βελτίωση κάποιων ελαττωμάτων αυτής ή σε εξομάλυνση των τιμών της.



Σχήμα 2.3: Η συνάρτηση ράμπας, ή ReLU σε εφαρμογές νευρωνικών δικτύων

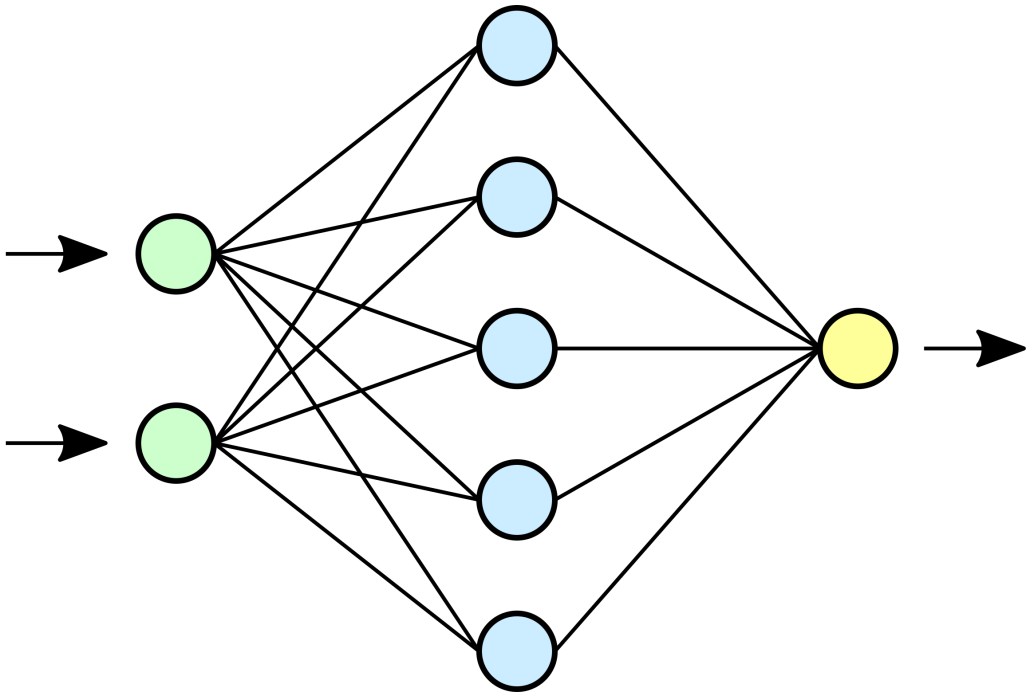
Συνδέσεις και Επίπεδα

Όπως προαναφέρθηκε, οι νευρώνες σε ένα νευρωνικό δίκτυο μπορούν να είναι συνδεδεμένοι μεταξύ τους, δηλαδή η είσοδος του επόμενου να είναι ίση με την έξοδο του προηγούμενου. Κάθε τέτοια σύνδεση χαρακτηρίζεται με έναν συντελεστή βάρους, ο οποίος καθορίζει τον βαθμό στον οποίο επιδρά ο προηγούμενος νευρώνα στον επόμενο. Αυτή η συνάρτηση επίδρασης ονομάζεται συνάρτηση διάδοσης και εκφράζεται ως εξής:

$$f(x) = Wx + b \tag{2.4}$$

όπου W το βάρος, x η είσοδος και b το bias.

Κάθε νευρωνικό δίκτυο οργανώνει τους νευρώνες του σε επίπεδα, τα οποία είναι συνδεδεμένα μεταξύ τους. Το αρχικό επίπεδο στο οποίο εισάγονται τα δεδομένα ονομάζεται επίπεδο εισόδου, ενώ το τελικό επίπεδο που παράγει και την τελική έξοδο του δικτύου ονομάζεται επίπεδο εξόδου. Όλα τα ενδιάμεσα επίπεδα ονομάζονται κρυφά επίπεδα. Η μορφή της σύνδεσης νευρώνων και κατά συνέπεια των επιπέδων μεταξύ τους χαρακτηρίζουν το είδος του νευρωνικού δικτύου. Για παράδειγμα, στην πλήρη σύνδεση (fully connected ή dense) κάθε νευρώνας ενός επιπέδου είναι συνδεδεμένος με όλους τους νευρώνες του επόμενου επιπέδου. Αν ένα επίπεδο συνδέεται μόνο με έναν νευρώνα του επόμενου επιπέδου, αυτό ονομάζεται pooling. Η μορφή ενός πλήρως συνδεδεμένου νευρωνικού δικτύου απεικονίζεται στο Σχήμα 2.4.



Σχήμα 2.4: Γραφική απεικόνιση ενός πλήρως συνδεδεμένου νευρωνικού δικτύου. Με πράσινο χρώμα συμβολίζονται τα επίπεδα εισόδου, με μπλέ χρώμα τα κρυφά επίπεδα και με κίτρινο το επίπεδο εξόδου.

Υπερπαράμετροι

Οι υπερπαράμετροι αποτελούν σταθερές παραμέτρους, οι τιμές των οποίων ορίζονται πρίν την εκκίνηση της διαδικασίας εκμάθησης του προβλήματος. Κάποιες από αυτές είναι οι εξής:

- ο ρυθμός εκμάθησης (learning rate), ο οποίος αφορά τον ρυθμό με τον οποίο το δίκτυο θα ενημερώνει τα βάρη, με στόχο να εξάγεται η επιθυμητή έξοδος που θα ελαχιστοποιεί το σφάλμα,
- το πλήθος των νευρώνων ανά επίπεδο,
- το μέγεθος του δείγματος κατά την οπισθοδρόμηση του σφάλματος (batch size).

Οπισθοδρόμηση του σφάλματος

Η ελαχιστοποίηση του σφάλματος από την επιθυμητή έξοδο επιτυγχάνεται με μια διαδικασία όπως την οπισθοδρόμηση του σφάλματος (backpropagation). Σε αυτή τη διαδικασία υπολογίζεται η παράγωγος της συνάρτησης απώλειας (loss function) σε σχέση με τα βάρη του δικτύου, μεταβάλλοντάς τα προς την κατεύθυνση της παραγώγου. Η διαδικασία αυτή επιτυγχάνεται διάφορες μεθόδους βελτιστοποίησης, όπως την gradient descent ή Adam.

2.3 Βαθιά ενισχυτική μάθηση

2.3.1 Γενικά

Η ενισχυτική μάθηση αποτελεί υποκατηγορία της μηχανικής μάθησης, μαζί με την μη επιβλεπόμενη και την επιβλεπόμενη μάθηση. Αφορά την εκπαίδευση ενός πράκτορα (agent), ώστε να λαμβάνει αποφάσεις (actions) σε μια κατάσταση (state) που θα τον οδηγήσουν στη μέγιστη ανταμοιβή (reward). Ο πράκτορας τοποθετείται εντός ενός περιβάλλοντος, με το οποίο αλλιλεπιδρά με συγκεκριμένο τρόπο και οδηγείται ανάλογα με τις αποφάσεις του σε πολλές διαφορετικές καταστάσεις. Ένα τέτοιο περιβάλλον περιγράφεται με χρήση της Μαρκοβιανής διαδικασίας λήψης αποφάσεων (Markov Decision Process - MDP).

Όσον αφορά τη βαθιά ενισχυτική μάθηση, αφορά την υλοποίηση των αλγορίθμων ενισχυτικής μάθησης με χρήση τεχνητών νευρωνικών δικτύων. Μια τέτοια έχει λιγότερες απαιτήσεις σε μνήμη ηλεκτρονικού υπολογιστή, σε σχέση με την παραδοσιακή υλοποίηση αυτών των αλγορίθμων με χρήση δυναμικού προγραμματισμού και παράλληλα είναι ιδανική υπο συνθήκες άγνωστου περιβάλλοντος, όπου ο χώρος καταστάσεων είναι συνεχής.

2.3.2 Μαρκοβιανή διαδικασία λήψης αποφάσεων

Ως μαρκοβιανή διαδικασία λήψης αποφάσεων (MDP) ορίζεται η πλειάδα (S, A, P_a, R_a), όπου:

- S, ο χώρος καταστάσεων,
- A, ο χώρος ενεργειών,
- $P_a(s_t, s_{t+1})$, η πιθανότητα μια ενέργεια α σε μια κατάσταση $s_t = s$ τη χρονική στιγμή t να οδηγήσει σε μια κατάσταση $s_{t+1} = s'$, τη χρονική στιγμή $t + 1$,
- $R_a(s_t, s_{t+1})$, η ανταμοιβή που δέχεται ο πράκτορας από την ενέργεια α για την κατάσταση $s_t = s$.

Η MDP χρησιμοποιείται σε συνδυασμό με κάποιον κατάλληλο αλγόριθμο για την εξαγωγή των ενεργειών του πράκτορα. Αυτές οι ενέργειες που ακολουθεί ο πράκτορας ονομάζονται και πολιτική (policy) $\pi(s)$. Αυτή η πολιτική συμπεριφέρεται ως μαρκοβιανή αλυσίδα, όπου κάθε προκυπτόμενη κατάσταση εξαρτάται από την προηγούμενη, αφού σε κάθε κατάσταση η ενέργεια εξαρτάται μόνο από την πολιτική. Στόχος είναι η βελτιστοποίηση μιας αθροιστικής συνάρτησης των μελλοντικών ανταμοιβών μέχρι την τελική κατάσταση του προβλήματος, η

οποία έστω ότι είναι τη χρονική στιγμή T. Αυτή η συνάρτηση εκφράζεται ως εξής:

$$\sum_{t=0}^{T-1} \gamma^t R_{\pi(s(t))}(s_t, s_{t+1}). \tag{2.5}$$

Η ποσότητα γ ονομάζεται ρυθμός έκπτωσης (discount factor) και είναι φραγμένη στο διάστημα $[0, 1]$, με τις τιμές τις να βρίσκονται πιο κοντά στη μονάδα. Εκφράζει την επιρροή που επιφέρει μια ενέργεια στις μελλοντικές ανταμοιβές που θα λάβει ο πράκτορας.

2.3.3 Deep Q-Learning

Το Q-Learning είναι ένας αλγόριθμος ενισχυτικής μάθησης, ο οποίος εκτιμάει την ποιότητα (quality) των ενεργειών, προσπαθώντας να προβλέψει τη μέγιστη μελλοντική ανταμοιβή και κατά συνέπεια τη βέλτιστη πολιτική του πράκτορα. Η ποιότητα μιας ενέργειας α σε μια κατάσταση s συμβολίζεται με $Q(s, a)$. Χρησιμοποιώντας την εξίσωση Bellman, εκφράζεται η συνάρτηση υπολογισμού της:

$$Q_t(s, a) = Q_{t-1}(s, a) + a(R(s, a) + \gamma(\max_{a'} Q(s', a') - Q_{t-1}(s, a)), \tag{2.6}$$

όπου α ο ρυθμός εκμάθησης, που δηλώνει πόσο γρήγορα η συνάρτηση προσαρμόζεται στα νέα δεδομένα. Το νευρωνικό δίκτυο που υλοποιεί μια τέτοια συνάρτηση ονομάζεται Deep Q-Network (DQN).

2.3.4 Actor - Critic

Το μοντέλο ενισχυτικής μάθησης Actor - Critic αποτελείται από τους δύο εκτιμητές, που περιγράφονται ως εξής:

- Actor, ο οποίος εξάγει τη βέλτιστη πολιτική, βάσει την εξόδο του critic, και
- Critic, ο οποίος εκτιμά την τιμή της μέγιστης μελλοντικής ανταμοιβής μιας ενέργειας στην κατάσταση s που θα οδηγήσει στην s'.

Ο λόγος ύπαρξης και χρήσης του μοντέλου critic είναι η μείωση της διακύμανσης της παραγώγου της βέλτιστης προκυπτόμενης πολιτικής, για τον λόγο ότι οι μεταβολές της ανταμοιβής κατά τη διάρκεια της εκπαίδευσης μπορεί να είναι μεγαλύτερες από τις μεταβολές της συνάρτησης εκτίμησής τους, της οποίας ο ρυθμός μάθησης ορίζεται αναλόγως.

2.3.5 Deep Deterministic Policy Gradient

Ο αλγόριθμος Deep Deterministic Policy Gradient (DDPG) συνδυάζει τις ιδέες των δύο παραπάνω μεθόδων, με στόχο την αποτελεσματική εκμάθηση συνεχών ενεργειών. Διατηρεί μνήμη των ενεργειών/εμπειρίες και 4 νευρωνικά δίκτυα στο σύνολο, εκ των οποίων τα 2 είναι τυπικά actor - critic και τα υπόλοιπα 2 είναι δίκτυα - στόχοι actor - critic, τα οποία ενημερώνονται με χαμηλότερους ρυθμούς. Ο λόγος υπάρξης των 2 δικτύων - στόχων είναι η αύξηση της σταθερότητας της εκπαίδευσης, αφού ο ρυθμός εκμάθησής τους είναι χαμηλότερος, διατηρώντας έτσι τους εκτιμώμενους στόχους σταθερότερους. Η μνήμη διατηρεί τις πλειάδες MDP από την αρχή της διαδικασίας εκπαίδευσης, μαθαίνοντας έτσι από ολόκληρη την εμπειρία μέχρι τη δεδομένη χρονική στιγμή, αντί μόνο από τις πρόσφατες εμπειρίες. Διαθέτει επίσης κάποια στοχαστική διαδικασία, η οποία χρησιμοποιείται ως θόρυβος για την έξοδο της ενέργειας, για λόγους αποτελεσματικότερης εξερεύνυσης του περιβάλλοντος.

Μετά την αρχικοποίηση των δικτύων, της μνήμης αλλά και της διαδικασίας θορύβου, ένας εξωτερικός βρόχος επανάληψης ορίζει το πλήθος των επεισοδίων της εκπαίδευσης και ένας

εσωτερικός βρόχος τα βήματα που επιτρέπονται ανά επεισόδιο. Στον εξωτερικό βρόχο λαμβάνεται η αρχική παρατήρηση του χώρου κατάστασης, ενώ στον εσωτερικό βρόχο λαμβάνεται η κατάλληλη ενέργεια, παρατηρείται η ανταμοιβή και η νέα κατάσταση. Αυτά αποθηκεύονται ως πλειάδα MDP στη μνήμη, λαμβάνεται δείγμα συγκεκριμένου μεγέθους από αυτή και ενημερώνονται κατάλληλα τα βάρη των actor - critic, αλλά και των αντίστοιχων δικτύων - στόχων [5] [6]. Η διαδικασία περιγράφεται υπό τη μορφή ψευδοκώδικα και στο Σχήμα 2.5.

Algorithm 1 DDPG algorithm

Randomly initialize critic network $Q(s, a|\theta^Q)$ and actor $\mu(s|\theta^\mu)$ with weights θ^Q and θ^μ.
Initialize target network Q' and μ' with weights $\theta^{Q'} \leftarrow \theta^Q$, $\theta^{\mu'} \leftarrow \theta^\mu$
Initialize replay buffer R
for episode = 1, M **do**
  Initialize a random process $\mathcal{N}$ for action exploration
  Receive initial observation state s_1
  for t = 1, T **do**
    Select action $a_t = \mu(s_t|\theta^\mu) + \mathcal{N}_t$ according to the current policy and exploration noise
    Execute action a_t and observe reward r_t and observe new state s_{t+1}
    Store transition (s_t, a_t, r_t, s_{t+1}) in R
    Sample a random minibatch of N transitions (s_i, a_i, r_i, s_{i+1}) from R
    Set $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$
    Update critic by minimizing the loss: $L = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i|\theta^Q))^2$
    Update the actor policy using the sampled policy gradient:

$$\nabla_{\theta^\mu} J \approx \frac{1}{N} \sum_i \nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu} \mu(s|\theta^\mu)|_{s_i}$$

    Update the target networks:

$$\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$$
$$\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}$$

  end for
end for

Σχήμα 2.5: Ψευδοκώδικας που περιγράφει τον αλγόριθμο DDPG [5]

2.3.6 Εφαρμογές μεθόδου DDPG

Ο αλγόριθμος DDPG βρίσκει εφαρμογές σε διάφορους τομείς όπου μπορεί να συναντάται ανάγκη για μηχανική μάθηση, κυρίως σε περιβάλλοντα όπου οι παράμετροι δέχονται τιμές σε συνεχές πεδίο. Αναφέρονται μερικές εφαρμογές αυτής της μεθόδου από τη βιβλιογραφία.

Σταθεροποίηση αντιστρεπτού εκκρεμούς

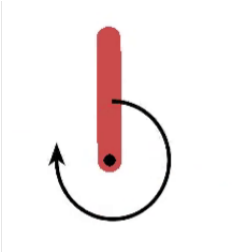
Ένα πολύ χαρακτηριστικό παράδειγμα εφαρμογής της μεθόδου αυτής είναι το παράδειγμα που συναντάται στις οδηγίες χρήσης του πακέτου μηχανικής μάθησης Keras [6], όπου πραγματοποιείται εκπαίδευση ενός πράκτορα, του οποίου ο στόχος είναι η σταθεροποίηση ενός αντιστρεπτού εκκρεμούς στην άνω κάθετη θέση. Το πεδίο των ενεργειών δέχεται τιμές από -2 εως 2 και εκφράζουν πόσο δεξιά ή αριστερά περιστρέφεται το εκκρεμές. Η μέθοδος λειτουργεί πολύ αποτελεσματικά καθώς σε λιγότερο από 100 επεισόδια ο πράκτορας είναι σε θέση να διατηρήσει το εκκρεμές στη ζητούμενη θέση.

Στα Σχήματα 2.6α′, 2.6β′, φαίνονται οι κινήσεις του εκκρεμούς, κατά τα αρχικά επεισόδια εκπαίδευσής του, όπου δεν διέθετε πολλές εμπειρίες, ενώ στο Σχήμα 2.7 φαίνεται το εκκρεμές στη άνω ακραία θέση του, με το σύστημα να το διατηρεί σε αυτή τη θέση.

(α΄) Αριστερή κίνηση αντιστρεπτού εκκρεμούς.

(β΄) Δεξιά κίνηση αντιστρεπτού εκκρεμούς.

Σχήμα 2.6: Διαδικασία εκπαίδευσης του αντεστραμένου εκκρεμούς.

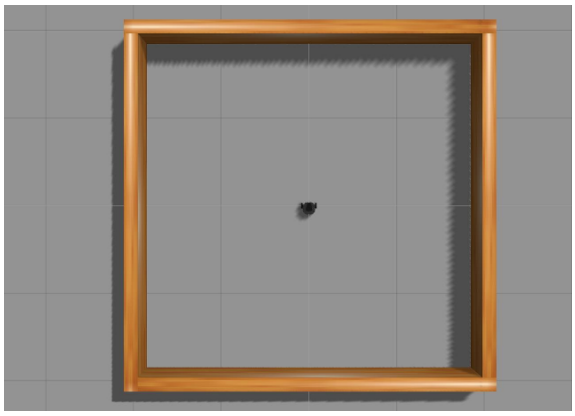


Σχήμα 2.7: Πλήρως εκπαιδευμένο αντιστρεπτό εκκρεμές, καταφέρνει και διατηρείται στην κάθετη θέση.

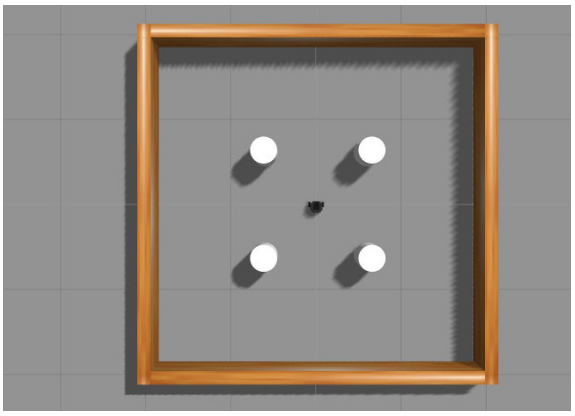
Σύστημα σχεδιασμού τροχιάς και αποφυγής εμποδίων για το ρομποτικό όχημα TurtleBot 3

Μια εφαρμογή με στόχο ανάλογο αυτόν της παρούσας εργασίας, είναι αυτή των Jesus *et.al* [7], οι οποίοι χρησιμοποίησαν τον αλγόριθμο DDPG για τη δημιουργία και εκπαίδευση ενός συστήματος που σχεδιάζει την τροχιά για το ρομποτικό όχημα TurtleBot 3 και παράλληλα αποφεύγει και εμπόδια στο περιβάλλον. Η εφαρμογή αυτή βασίζεται εξ ολοκλήρου στα ενσωματωμένα περιβάλλοντα του TurtleBot 3 στο Gazebo, που παρουσιάζονται στο Σχήμα 2.8.

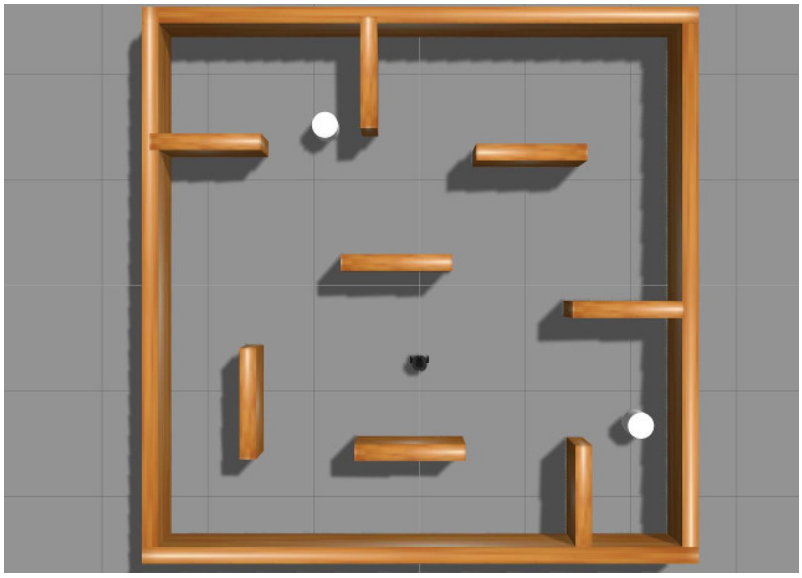
Τα αποτελέσματα ήταν ικανοποιητικά για τα πρώτα 2 περιβάλλοντα (Σχήμα 2.8α΄ και 2.8β΄), όμως ο πράκτορας συναντούσε δυσκολία στη μετάβαση στο τρίτο περιβάλλον (Σχήμα 2.8γ΄) λόγω της πολυπλοκότητας του. Αυτό οφείλεται στη συνάρτηση ανταμοιβής που χρησιμοποιήθηκε σε αυτή την εφαρμογή, η οποία υπολόγιζει αποκλειστικά την ανταμοιβή αναλόγως με το πόσο κοντά στο στόχο βρισκόταν το ρομπότ, παραλείποντας άλλες παραμέτρους όπως τη γωνία προσέγγισής του ή την απόστασή του απο εμπόδια.



(α′) Το πρώτο περιβάλλον εκπαίδευσης, με μοναδικό εμπόδιο η περίφραξη του χώρου $4m \times 4m$



(β′) Το δεύτερο περιβάλλον εκπαίδευσης, με 4 επιπλέον εμπόδια στο κέντρο



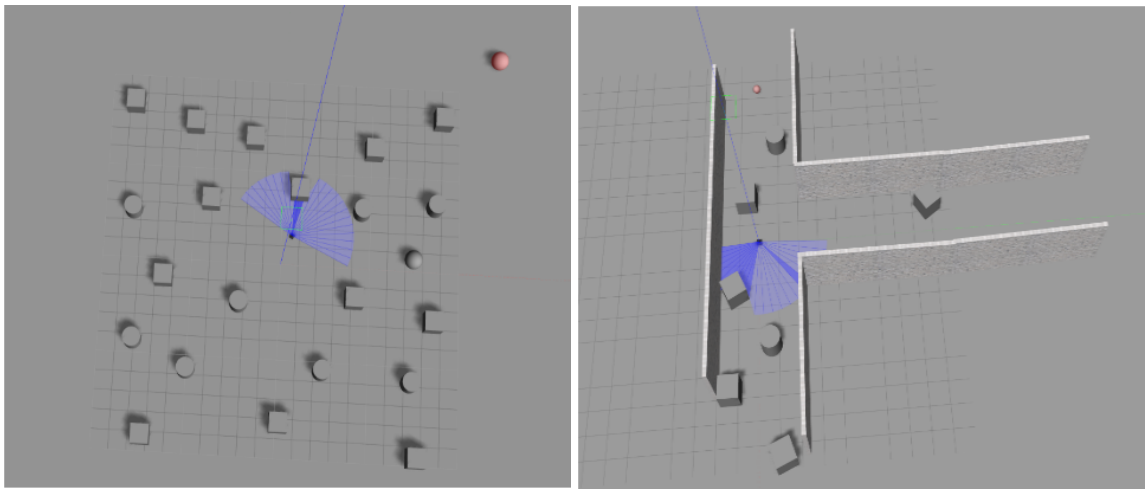
(γ′) Το τρίτο περιβάλλον εκπαίδευσης, με διάφορα μεγάλα εμπόδια

Σχήμα 2.8: Τα περιβάλλοντα εκπαίδευσης στο περιβάλλον Gazebo

Σύστημα σχεδιασμού τροχιάς και αποφυγής εμποδίων με εκπαίδευση από ανθρώπινο παίχτη

Αυτή η εφαρμογή των Niu *et.al*, [8] είναι ανάλογη σε φιλοσοφία με αυτήν των Jesus *et.al* [7], με τη διαφορά ότι αντί να χρησιμοποιείται στοχαστική διαδικασία θορύβου με στόχο την εξερεύνηση του περιβάλλοντος, εμπλέκεται ανθρώπινος παίχτης-χειριστής του ρομπότ, ο οποίος όχι μόνο βοηθά το δίκτυο να εκπαιδευτεί με διαφορετικά σενάρια, αλλά και του υποδεικνύει τις σωστές κινήσεις. Με αυτό τον τρόπο, το δίκτυο δέχεται ως εμπειρία τις υποδειγματικές κινήσεις του ανθρώπινου χειριστή και είναι σε θέση να τις αναπαραγάγει, ανάλογα το σενάριο στο οποίο βρίσκεται.

Η υλοποίηση αυτή ήταν σε θέση μετά από πολύ λίγες επαναλήψεις εκπαίδευσης να λειτουργεί αποτελεσματικά, χωρίς συγκρούσεις με το περιβάλλον. Συγκεκριμένα, όσον αφορά το πρώτο περιβάλλον, ο πράκτορας κατάφερε να μη συγκρούεται με το περιβάλλον του μετά από 20 επεισόδια και 25000 επαναλήψεις, με διάρκεια εκπαίδευσης κοντά στις 2 ώρες. Συγκρίνοντας αυτή τη μέθοδο με μια παραδοσιακή υλοποίηση DDPG, ο κλασσικός αλγόριθμος είχε κάνει 7 συγκρούσεις μετά από 60000 επαναλήψεις και 1 σύγκρουση μετά από 70000 επαναλήψεις. Όσον αφορά το δεύτερο περιβάλλον, το οποίο είναι και πιο απαιτητικό, ο πράκτορας κατάφερνε να μη συγκρούεται μετά από 20 επεισόδια και 35000 επαναλήψεις, με διάρκεια εκπαίδευσης 2



(α΄) Το πρώτο περιβάλλον δοκιμών, με διάφορα εμπόδια στο χώρο

(β΄) Το δεύτερο περιβάλλον δοκιμών, με στενούς διαδρόμους και εμπόδια

Σχήμα 2.9: Τα περιβάλλοντα δοκιμών στο Gazebo [8]

ώρες και 25 λεπτά. Συγκριτικά με τον κλασσικό αλγόριθμο, αυτός είχε κάνει 3 συγκρούσεις μετά από 80000 βήματα.

Σύμφωνα με τις δοκιμές που εκτελέστηκαν, παρατηρήθηκε όχι μόνο μεγαλύτερο ποσοστό επιτυχίας συγκριτικά με την κλασσική μέθοδο DDPG, αλλά και καλύτερες και πιο άμεσες τροχιές. Για παράδειγμα, υπήρχαν σενάρια τα οποία το ρομπότ έκανε διάφορες περιστροφές ή κύκλους, τα οποία δεν έκανε στην προτεινόμενη μέθοδο.

Συμπερασματικά, για την μέθοδο DDPG, φαίνεται πως είναι πιο αποτελεσματική η εκπαίδευση με ανθρώπινο χειριστή, εφόσον είναι εφικτό στο σενάριο για το οποίο εφαρμόζεται. Φέρει πλεονεκτήματα στο χρόνο εκπαίδευσης αλλά και στην ποιότητα του αποτελέσματος, όμως απαιτείται ο ανθρώπινος παράγοντας, συγκριτικά με την παραπάνω μέθοδο στην υποενότητα 2.3.6, στην οποία η διαδικασία της εκπαίδευσης είναι πλήρως αυτοματοποιημένη, αφού λαμβάνει θόρυβο για την εξερεύνηση πολλών πιθανών διαφορετικών σεναρίων.

2.4 Στοχαστική διαδικασία Ornstein–Uhlenbeck

2.4.1 Γενικά

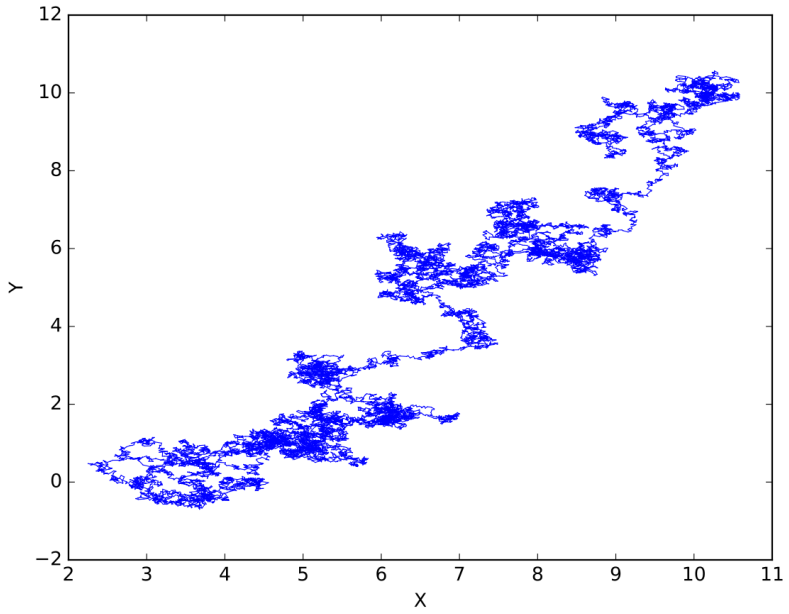
Η διαδικασία Ornstein–Uhlenbeck [9] είναι μια στοχαστική διαδικασία με διάφορες εφαρμογές στα χρηματοοικονομικά μαθηματικά και στις φυσικές επιστήμες. Χρησιμοποιείται για παραγωγή θορύβου σε συναρτήσεις. Η διαδικασία αυτή λαμβάνει δείγματα θορύβου από μια κανονική κατανομή. Πρόκειται για μια τροποποίηση του τυχαίου περίπατου σε συνεχή χρόνο, ή αλλιώς της διαδικασίας Wiener, ενώ της έχει δοθεί η ιδιότητα να έχει την τάση να επιστρέφει σε μια κεντρική θέση, με αυτή την τάση να γίνεται ολοένα και πιο ισχυρή όσο απομακρύνεται από αυτό το κέντρο.

Η διαδικασία OU x_t ορίζεται από την παρακάτω στοχαστική διαφορική εξίσωση:

$$dx_t = -\theta x_t dt + \sigma dW_t, \tag{2.7}$$

όπου $\theta > 0$ και $\sigma > 0$ οι παράμετροί της και W_t η στοχαστική διαδικασία Wiener. Συχνά παρουσιάζεται και με έναν επιπλέον σταθερό όρο - παράμετρο μ, ο οποίος ρυθμίζει την παρέκκλιση και εκφράζεται ως εξής:

$$dx_t = -\theta(\mu - x_t)dt + \sigma dW_t. \tag{2.8}$$



Σχήμα 2.10: Προσομοίωση της διαδικασίας OU με $\theta = 1.0, \sigma = 3.0, \mu = (0, 0)$, με αρχική θέση $(10, 10)$. Παρατηρείται η τάση των σημείων να επιστρέψουν στο κεντρικό σημείο μ.

2.5 Διαφορική οδήγηση

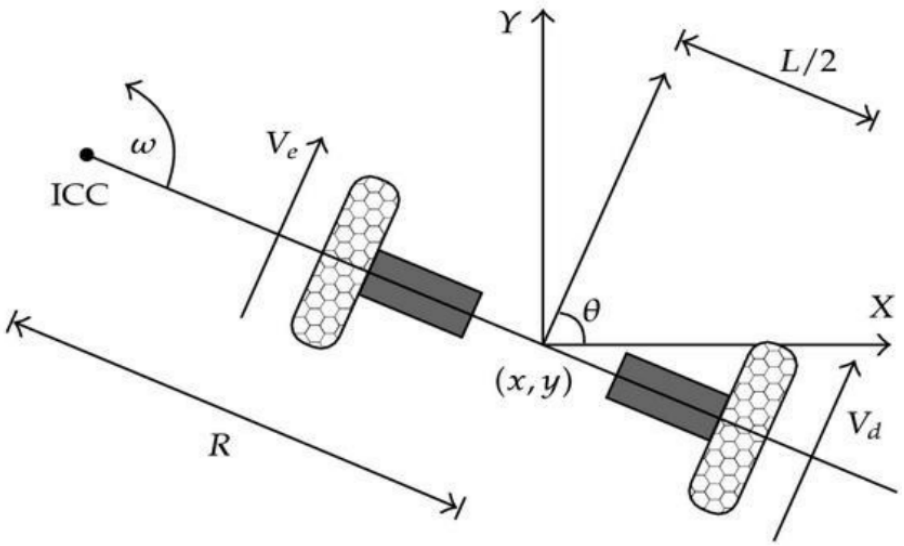
2.5.1 Κινηματικό μοντέλο

Το ρομποτικό όχημα στο οποίο βασίστηκε αυτή η εργασία, αλλά και ένα μεγάλο πλήθος ρομποτικών οχημάτων, επιτυγχάνει την κίνησή του με έναν μηχανισμό γνωστό ως διαφορική οδήγηση. Ο συγκεκριμένος μηχανισμός αποτελείται από δύο κινητήριους τροχούς τοποθετημένους σε κοινό άξονα και έχουν τη δυνατότητα να περιστραφούν ανεξάρτητα και προς τις δύο κατευθύνσεις. Τα οχήματα που εκμεταλλεύονται αυτόν τον κινητήριο μηχανισμό έχουν δύο βαθμούς ελευθερίας και η κίνησή τους εμπίπτει εντός του επιπέδου XY. Το ρομπότ, ανεξαρτήτως της ταχύτητας των δύο τροχών, περιστρέφεται πάντα γύρω από ένα σημείο, το οποίο βρίσκεται κατά μήκος του κοινού άξονα των δύο τροχών. Αυτό το σημείο ονομάζεται στιγμιαίο κέντρο στροφής (Instantaneous Center of Curvature - ICC) και φαίνεται στο Σχήμα 2.11.

Ένας σημαντικός περιορισμός της διαφορικής οδήγησης είναι πως το όχημα είναι μη ολονομικό, δηλαδή δεν μπορεί να κινηθεί προς όλες τις κατευθύνσεις. Συγκεκριμένα, δεν μπορεί να κινηθεί κατά τη διεύθυνση του άξονα των τροχών του, παραμόνο εμπρός, πίσω αλλά και υπο γωνία αυτού. Γι΄ αυτό το λόγο, απαιτείται συνδυασμός κινήσεων, τόσο γραμμικών όσο και περιστροφικών, προκειμένου να φτάσει στο στόχο του [10].

Εξηγούνται οι συμβολισμοί του Σχήματος 2.11:

- ICC, το στιγμιαίο κέντρο στροφής,
- V_e, V_d, η ταχύτητα του αριστερού και του δεξιού τροχού αντίστοιχα,
- ω, η γωνιακή ταχύτητα,
- R, η απόσταση του ICC από το μέσο της απόστασης μεταξύ των δύο τροχών,
- L, η απόσταση μεταξύ των δύο τροχών,



Σχήμα 2.11: Το κινηματικό μοντέλο διαφορικής κίνησης

- (x, y), η θέση του ρομπότ στο πλαίσιο συντεταγμένων αναφοράς,
- θ, η γωνία που έχει στραφεί το όχημα σε σχέση με το πλαίσιο αναφοράς.

Γνωρίζοντας ότι η γωνιακή ταχύτητα ω είναι όμοια και για τους δύο τροχούς και η σημειακή ταχύτητα υπολογίζεται σύμφωνα με τη σχέση $u = \omega \times r$, προκύπτουν οι εξής εξισώσεις για τις ταχύτητες των δύο τροχών:

$$\omega(R + \frac{L}{2}) = V_d \tag{2.9}$$

$$\omega(R - \frac{L}{2}) = V_e \tag{2.10}$$

Λύνοντας τις Εξισώσεις 2.9 και 2.10 ως πρός R και ω, προκύπτουν οι εξισώσεις υπολογισμού των κινήσεων του οχήματος, ως εξής:

$$R = \frac{L}{2}\frac{V_e + V_d}{V_d - V_e} \tag{2.11}$$

$$\omega = \frac{V_d + V_e}{L} \tag{2.12}$$

Συγκεκριμένα, εάν:

- $V_d = V_e$, τότε το όχημα κινείται είτε έμπροσθεν, είτε όπισθεν εάν είναι αρνητικές. Η απόσταση R τείνει στο άπειρο, ενώ η γωνιακή ταχύτητα ω είναι μηδενική, αφού δεν υπάρχει περιστροφή του οχήματος.
- $V_e = -V_d$, δηλαδή οι ταχύτητες είναι ίσες κατά μέτρο αλλά αντίθετες κατά κατεύθυνση, τότε το όχημα περιστρέφεται γύρω από τον εαυτό του, αφού η απόσταση R είναι μηδενική.
- $V_d = 0$ ή $V_e = 0$, δηλαδή μόνο ένας εκ των δύο τροχών κινείται, τότε το όχημα περιστρέφεται γύρω από τον ακίνητο τροχό.

Για τον υπολογισμό της νέας θέσης του οχήματος, διαθέτοντας ως δεδομένη τη γωνιακή του ταχύτητα, χρησιμοποιούνται οι σχέσεις που παρουσιάζονται στη συνέχεια. Αν θεωρηθεί ότι

τη χρονική στιγμή t το όχημα βρίσκεται στη θέση (x, y) με γωνία στροφής ω, βάσει του Σχήματος 2.11. Τότε, από τις Εξισώσεις 2.11 και 2.12 υπολογίζεται η θέση του ICC:

$$ICC = [x - Rsin\theta, y + Rcos\theta] \tag{2.13}$$

Τη χρονική στιγμή $t + \delta t$ το όχημα βρίσκεται στο σημείο (x', y') με γωνία ω' με διάνυσμα θέσης $[x', y', \theta']$ και κατεύθυνση η οποία υπολογίζεται ως εξής:

$$\begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} = \begin{bmatrix} cos(\omega \cdot \delta t) & -sin(\omega \cdot \delta t) & 0 \\ sin(\omega \cdot \delta t) & cos(\omega \cdot \delta t) & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x - ICC_x \\ y - ICC_y \\ \theta \end{bmatrix} \cdot \begin{bmatrix} ICC_x \\ ICC_y \\ \omega \cdot \delta t \end{bmatrix} \tag{2.14}$$

Στην περίπτωση που γνωρίζουμε τη νέα θέση του οχήματος μπορεί να υπολογιστεί η γωνιακή ταχύτητα για να φτάσει σε αυτή. Η θέση του οχήματος που κινείται με ταχύτητα $V(t)$ υπό γωνία θ, είναι:

$$x(t) = \int_0^t V(t)cos[\theta(t)]dt \tag{2.15}$$

$$y(t) = \int_0^t V(t)sin[\theta(t)]dt \tag{2.16}$$

$$\theta(t) = \int_0^t \omega(t)dt \tag{2.17}$$

Για την περίπτωση της διαφορικής κίνησης, οι παραπάνω Εξισώσεις 2.15, 2.16 και 2.17 μετασχηματίζονται κατ΄ αντιστοιχία ως εξής:

$$x(t) = \frac{1}{2}\int_0^t [V_e(t) + V_d(t)]cos[\theta(t)]dt \tag{2.18}$$

$$y(t) = \frac{1}{2}\int_0^t [V_e(t) + V_d(t)]sin[\theta(t)]dt \tag{2.19}$$

$$\theta(t) = \frac{1}{L}\int_0^t ([V_d(t) - V_e(t)])dt \tag{2.20}$$

Όσον αφορά την επιμέρους ανάλυση των αποκλειστικά γραμμικών ή περιστροφικών (γύρω από τον εαυτό του) κινήσεων, αυτές περιγράφονται ως εξής:

- $V_d = V_e = V$, για γραμμική κίνηση:

$$\begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} = \begin{bmatrix} x + Vcos(\theta)\delta t \\ y + Vsin(\theta)\delta t \\ \theta \end{bmatrix} \tag{2.21}$$

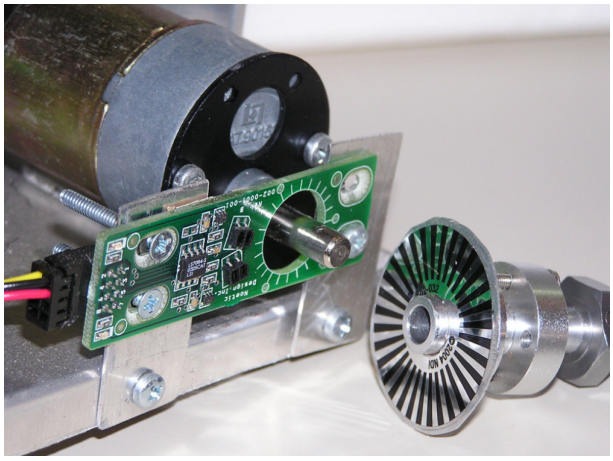
- $V_d = -V_e = V$, για περιστροφική κίνηση γύρω από τον εαυτό του:

$$\begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} = \begin{bmatrix} x \\ y \\ \theta + \frac{2V\delta t}{L} \end{bmatrix} \tag{2.22}$$

2.5.2 Οδομετρία

Οδομετρία είναι η χρήση δεδομένων από κατάλληλους αισθητήρες για την προσέγγιση μετατόπισης σε βάθος χρόνου. Χρησιμοποιείται ευρέως στον τομέα της ρομποτικής σε τροχούς ή αρθρώσεις για τον προσδιορισμό της θέσης τους σε σχέση με μια αρχική θέση, η οποία συνήθως ορίζεται από κάποιο πλαίσιο συντεταγμένων. Στην περίπτωση έντροχων ρομποτικών οχημάτων των οποίων το κινηματικό μοντέλο περιγράφεται με αυτό της διαφορικής κίνησης, προσαρτούνται περιστρεφόμενοι κωδικοποιητές στους δύο κινητήρες και όταν αυτοί περιστρέφονται, υπολογίζεται το πόσο περιστράφηκαν σε δεδομένο χρόνο, επιτυγχάνοντας έτσι την προσέγγιση της θέσης του οχήματος στο πλαίσιο συντεταγμένων του επιπέδου XY στο οποίο κινείται το όχημα.

Αυτή η μέθοδος είναι ιδιαίτερα ευαίσθητη σε σφάλματα, λόγω της ενσωμάτωσης υπολογισμού ταχύτητας σε βάθος χρόνου με στόχο την εξαγωγή προσεγγίσεων της θέσης. Παρατηρείται ότι το σφάλμα συσσωρεύεται και μεγαλώνει σε βάθος χρόνου, κατά τη διάρκεια εκτέλεσης κινήσεων. Με κατάλληλη βαθμονόμηση των μετρητικών οργάνων και γρήγορη και ακριβή συλλογή δεδομένων μειώνεται το σφάλμα στην οδομετρία, όμως δεν εξαλείφεται. Η μέθοδος αυτή χρησιμοποιείται αποτελεσματικότερα σε συνδυασμό με άλλα αισθητήρια, όπως έναν λέιζερ μετρητή αποστάσεων.



Σχήμα 2.12: Οδόμετρο από ρομποτικό όχημα του Εργαστηρίου Ευφυών Συστημάτων και Ρομποτικής του Πολυτεχνείου Κρήτης [11]

2.6 Μετρητής αποστάσεων λέιζερ

Ο μετρητής αποστάσεων λέιζερ, ή αλλιώς LiDAR, ακρωνύμιο για light detection and ranging, αποτελεί ένα αισθητήριο το οποίο υπολογίζει κάποια απόσταση εκπέμποντας μια ακτίνα λέιζερ στο σημείο που επιθυμεί να υπολογίσει την οποία λαμβάνει πίσω λόγω ανάκλασης και υπολογίζει το χρόνο που χρειάστηκε αυτή να επιστρέψει. Διαθέτει τη δυνατότητα μέσα από πολλαπλές μετρήσεις σε διάφορα σημεία να δημιουργήσει και εικόνα για το αντικείμενο στο οποίο εφαρμόζεται ο αισθητήρας.

Υπάρχουν αισθητήρες που λαμβάνουν δεδομένα μόνο σε ένα σημείο, ενώ υπάρχουν και περιστρεφόμενοι οι οποίοι λαμβάνουν δεδομένα σε διάφορα σημεία γύρω από τον εαυτό τους. Συνήθως πραγματοποιούν έναν πλήρη κύκλο, διαθέτοντας έτσι εικόνα για ολόκληρο το περιβάλλον γύρω τους. Ένας τυπικός αισθητήρας αυτού του τύπου που χρησιμοποείται σε εφαρμογές ρομποτικής παρουσιάζεται στο Σχήμα 2.13.

Το LiDAR βρίσκει εφαρμογές σε διάφορους τομείς όπου απαιτείται είτε ο απομακρυσμένος υπολογισμός αποστάσεων, είτε η ψηφιακή απεικόνιση κάποιας επιφάνειας. Χρησιμοποιείται

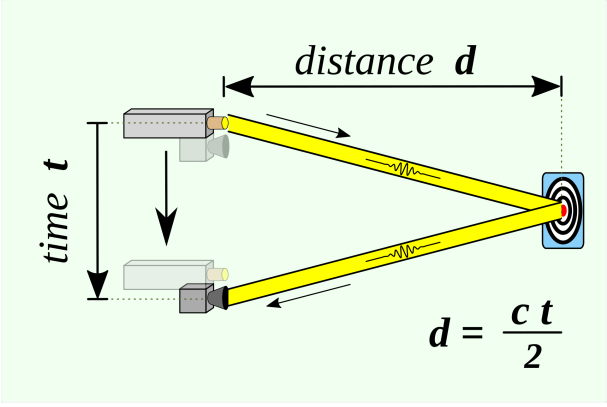
Σχήμα 2.13: Περιστρεφόμενος αισθητήρας LiDAR από την εταιρεία SICK

πολύ συχνά στη ρομποτική ως την κύρια πηγή δεδομένων πλοήγησης του ρομπότ για την απεικόνιση του περιβάλλοντος και την αποφυγή εμποδίων. Χρησιμοποιείται και ως μέθοδο τρισδιάστατης απεικόνισης αντικειμένων (3D scanning) και για μεθόδους επαυξημένης πραγματικότητας (AR). Βρίσκει επίσης εφαρμογές σε οχήματα, είτε για αυτόνομη οδήγηση είτε απλώς για συστήματα ασφαλείας όπως την αποφυγή συγκρούσεων και πεζών. Τέλος, χρησιμοποιείται επίσης για το σχηματισμό χαρτών υψηλής ευκρίνειας, από τομείς όπως τη γεωλογία, την γεωγραφία, την γεωμορφολογία, τη σεισμολογία, την αρχαιολογία κοκ. [12]

Η λειτουργία του LiDAR βασίζεται στο χρόνο πτήσης του φωτός. Συγκεκριμένα, απελευθερώνεται μια ακτίνα λέιζερ από τον αισθητήρα με τη λειτουργία του πομπού, ενώ ένας δέκτης δίπλα στον πομπό λαμβάνει την ανακλώμενη ακτίνα (Σχήμα 2.14). Μετράται ο χρόνος που χρειάστηκε ο δέκτης να λάβει την ακτίνα από τη στιγμή που η ακτίνα απελευθερώθηκε και η τελική απόσταση υπολογίζεται ως εξής:

$$d = \frac{ct}{2}$$

Τα μήκη κύματος των ακτίνων λέιζερ που χρησιμοποιούνται στα περισσότερα αισθητήρια λέιζερ της αγοράς κυμαίνονται στα $600 - 1000nm$, γεγονός το οποίο τα καθιστά ασφαλή προς το ανθρώπινο μάτι χωρίς προστασία, αλλά και αόρατο. Συνήθως χρησιμοποιούνται λέιζερ χαμηλής ισχύος, γεγονός το οποίο περιορίζει την απόσταση λειτουργίας (συνήθως δίνονται αξιόπιστα δεδομένα μέχρι αποστάσεις $3.5m$), όμως βελτιώνει την ακρίβεια. Εναλλακτικά, χρησιμοποιούνται λέιζερ μήκους κύματος $1500nm$ τα οποία είναι επίσης ασφαλή στο ανθρώπινο μάτι, με υψηλότερη ισχύ, άρα και μεγαλύτερη απόσταση αλλά χαμηλότερη ακρίβεια.



Σχήμα 2.14: Γραφική απεικόνιση της μεθόδου υπολογισμού του χρόνου πτήσης

Κεφάλαιο 3

Εργαλεία υλοποίησης

3.1 Robot Operating System

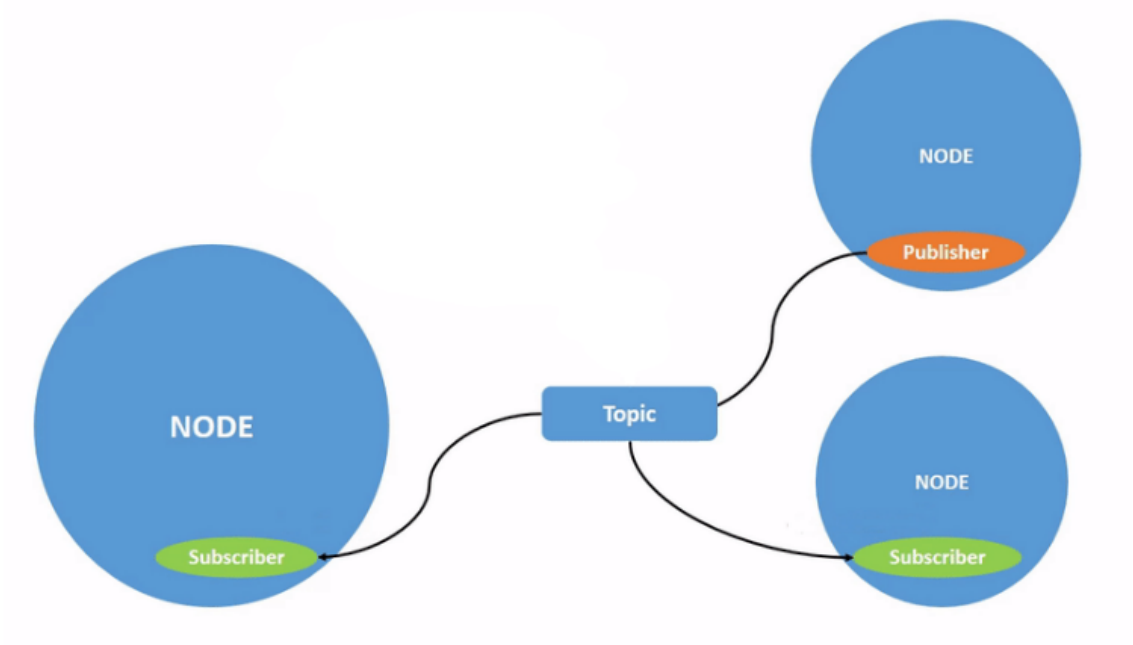
3.1.1 Γενικά

Το Robot Operating System (ROS) είναι ένα λογισμικό ανοιχτού κώδικα το οποίο παρέχει ένα σύνολο από ενδιάμεσες δομές για ανάπτυξη λογισμικού για ρομπότ. Παρέχει χαμηλού επιπέδου έλεγχο του υλικού, ένα σύστημα μετάδοσης μηνυμάτων μεταξύ των διεργασιών που εκτελούνται και χαμηλό χρόνο απόκρισης. Προσφέρει ευελιξία στη γλώσσα προγραμματισμού, καθώς διαθέτει βιβλιοθήκες για C++, Python, Lisp, MATLAB και Java. Έρχεται με διάφορα προεγκατεστημένα εργαλεία γραφικού περιβάλλοντος, όπως τον προσομοιωτή Gazebo, το εργαλείο διαχείρισης θεμάτων rqt και το εργαλείο οπτικοποίησης του χάρτη, των αισθητήρων και των κινητήρων rviz.

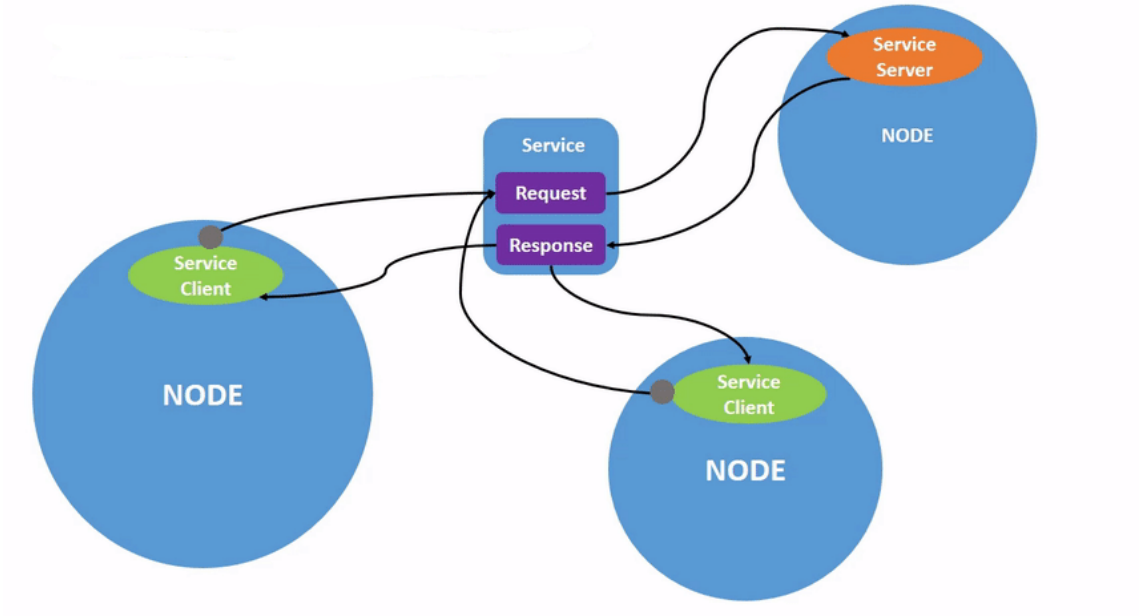
Διαθέτει ενσωματωμένο σύστημα δημιουργίας χαρτών και εύρεσης της τοποθεσίας του ρομπότ σε αυτόν (SLAM) και διαφόρων αλγορίθμων εύρεσης μονοπατιών (path planners), όπως τους carrot planner και dwa planner. Ένα ακόμη σημαντικό χαρακτηριστικό του είναι η ενσωματωμένη βιβλιοθήκη tf, η οποία αναπαριστά, παρακολουθεί και μετασχηματίζει πλαίσια συντεταγμένων.

Η φιλοσοφία λειτουργίας του ROS βασίζεται σε ένα είδους μοντέλου γραφήματος. Συγκεκριμένα, κάθε διεργασία που εκτελείται στο περιβάλλον αποτελεί έναν κόμβο (node). Οι κόμβοι συνδέονται μεταξύ τους με ακμές, οι οποίες ονομάζονται topics ή αλλιώς μηνύματα. Με αυτή τη λογική, μπορεί ο χρήστης να ρυθμίσει κατάλληλα το λογισμικό του ώστε να μπορεί να δημοσιεύει πληροφορίες υπό τη μορφή μηνυμάτων από έναν κόμβο και να λαμβάνονται από κάποιον άλλον. Η σχετική γραφική αναπαράσταση φαίνεται στο Σχήμα 3.1. Επιπλέον, υπάρχει και μια άλλη μορφή επικοινωνίας μεταξύ των κόμβων στο σύστημα, οι υπηρεσίες. Σε αντίθεση με τα μηνύματα, τα οποία αποστέλλονται συνεχώς, με μια ορισμένη συχνότητα, οι υπηρεσίες αποστέλλουν δεδομένα μόνο όταν κληθούν από κάποιον κόμβο-πελάτη της υπηρεσίας. Η σχετική γραφική αναπαράσταση φαίνεται στο Σχήμα 3.2. [11] [13]

Για παράδειγμα, στην περίπτωση του Turtlebot, το topic "/cmd_vel" διαθέτει την γραμμική και περιστροφική ταχύτητα του ρομπότ. Με αυτό τον τρόπο, μπορεί ο χρήστης να δημοσιεύσει την επιθυμητή ταχύτητα και το ρομπότ θα τροποποιήσει κατάλληλα, μέσω του υπεύθυνου node, την ταχύτητα των δύο κινητήρων. Άλλα αξιοσημείωτα topics αυτού του ρομπότ είναι το "/scan", που καταγράφει τις αποστάσεις που εξάγει ο αισθητήρας λέιζερ, το "/odom" που διατηρεί πληροφορίες για την οδομετρία και το "/tf" που παρακολουθεί όλα τα πλαίσια συντεταγμένων που αφορούν το ρομπότ. Στα Σχήματα 3.3 και 3.4 φαίνονται όλα τα μηνύματα που χρησιμοποιεί το εν λόγω ρομπότ και η γραφική απεικόνισή τους.



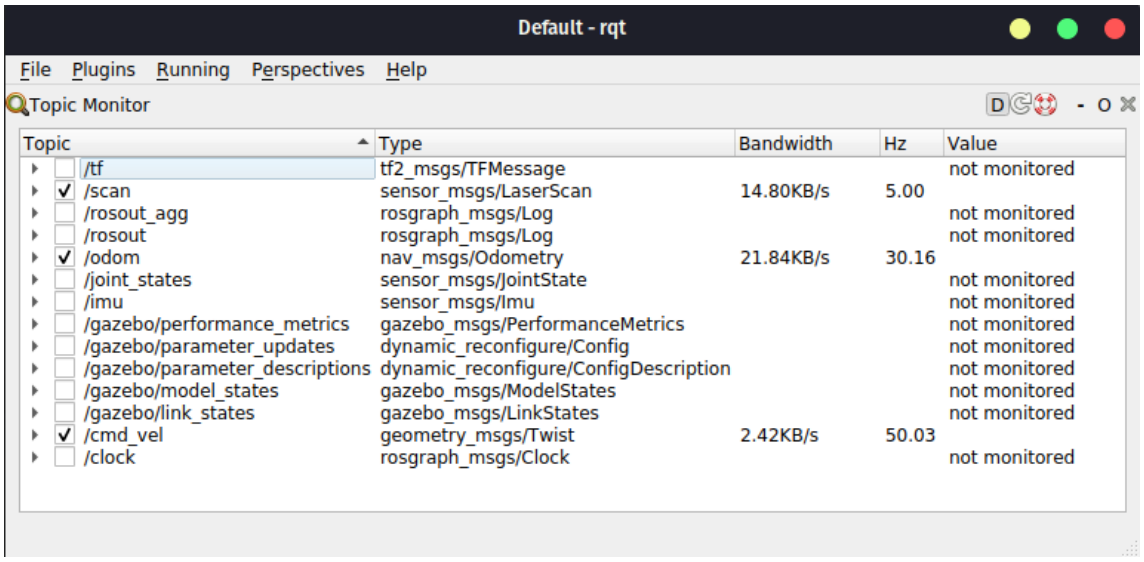
Σχήμα 3.1: Γραφική απεικόνιση ρομποτικής εφαρμογής με τρείς κόμβους. Ο ένας λειτουργεί ως αποστολέας (publisher) ενός μηνύματος με όνομα topic, ενώ οι άλλοι 2 είναι συνδρομητές/παραλήπτες (subscribers) σε αυτό το μήνυμα [13].



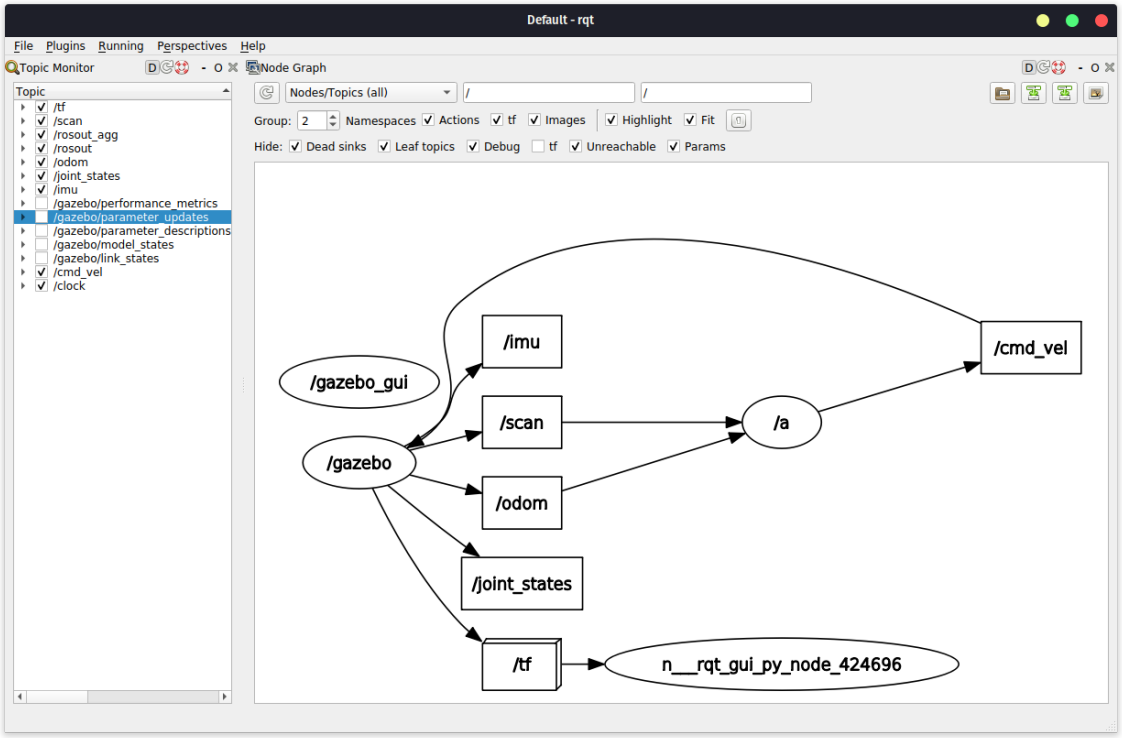
Σχήμα 3.2: Γραφική απεικόνιση ρομποτικής εφαρμογής με τρείς κόμβους. Οι 2 λειτουργούν ως πελάτες (clients) μιας υπηρεσίας (service), ενώ ο τρίτος λειτουργεί ως εξυπηρετητής (server) των κλήσεων που προέρχονται από τους πελάτες [13].

3.1.2 Gazebo

Το Gazebo είναι ένας προσομοιωτής ρομποτικής ανοιχτού κώδικα, ο οποίος έρχεται μαζί με το πακέτο του ROS από τότε που πρωτοκυκλοφόρησε. Διαθέτει μηχανή τρισδιάστατων γραφικών, η οποία παρέχει φωτορεαλιστική απεικόνιση, με υψηλής ποιότητας φωτισμό, σκιές και επιφάνειες. Προσφέρει πλήρη προσομοίωση φυσικής, μέσα από διάφορες εναλλακτικές μη-

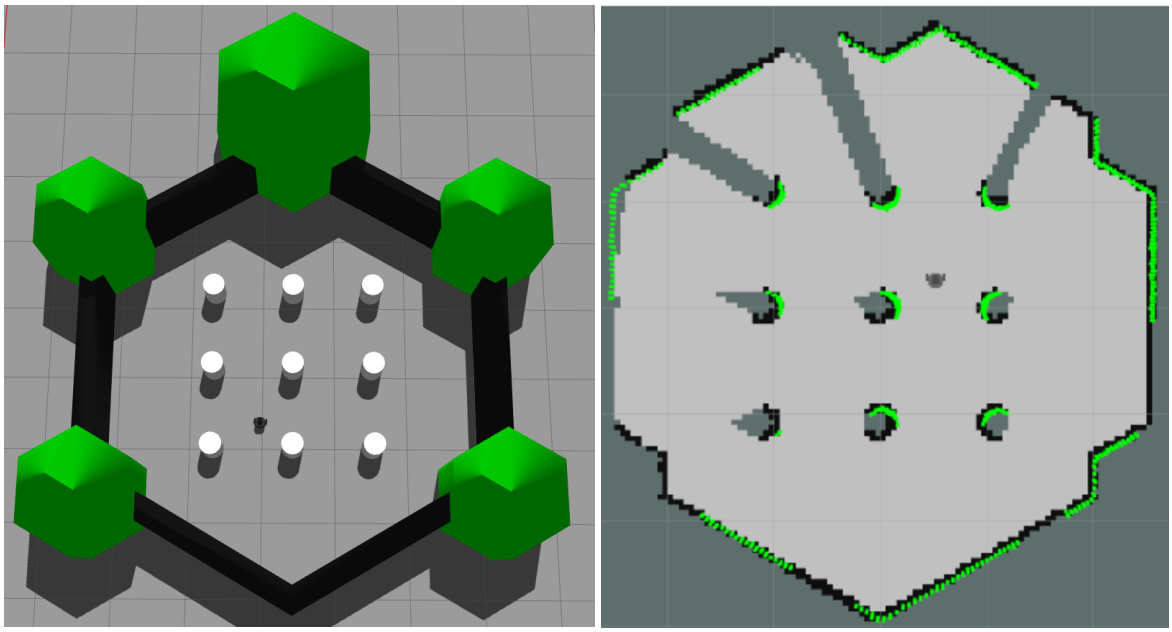


Σχήμα 3.3: rqt Topic Monitor για το Turtlebot



Σχήμα 3.4: rqt Node Graph, γραφική απεικόνιση δρομολόγησης μηνυμάτων σε εφαρμογή που εκτελείται στο Turtlebot, μέσα στο περιβάλλον του Gazebo.

χανές προσομοίωσης δυναμικής συμπεριφοράς στερεών και ρευστών, όπως οι ODE, Bullet, Simbody, DART. Διαθέτει υποστήριξη για προσομοίωση διφόρων αισθητηρίων που χρησιμοποιεί και το ROS, όπως LiDAR ή κάμερες τύπου Kinect και ενεργητών, προσομοιώνοντας παράλληλα και θόρυβο, διατηρώντας έτσι τις μετρήσεις ρεαλιστικές και πιο κοντά στην πραγματικότητα. Προσφέρει δυνατότητες προγραμματισμού των συμπεριφορών των αισθητηρίων, των ενεργοποιητών και των σωμάτων που συμμετέχουν στο περιβάλλον προσομοίωσης. Τέλος, διατίθενται προς χρήση διάφορα τρισδιάστατα μοντέλα, είτε από ρομπότ είτε από αντικείμενα, αλλά και ολόκληροι εσωτερικοί χώροι ή εξωτερικά περιβάλλοντα. [11]



(α′) Ο δοκιμαστικός κόσμος του Turtlebot 3, στον προσομοιωτή Gazebo

(β′) Γραφική απεικόνιση του χάρτη, από τον αισθητήρα λέιζερ, στο rviz

Σχήμα 3.5: Εκτέλεση χαρτογράφησης SLAM [14]

3.2 Keras/TensorFlow

Το TensorFlow είναι μια προγραμματιστική βιβλιοθήκη ανοιχτού κώδικα για τεχνητή νοημοσύνη και μηχανική μάθηση. Εστιάζει στη χρήση τανυστών (tensors) για την έκφραση και εκπαίδευση βαθιών νευρωνικών δικτύων. Αναπτύσσεται ενεργά από το 2015 από τη Google. Χρησιμοποιείται σε διάφορες γλώσσες προγραμματισμού, όπως Javascript, C++ και Java, αλλά πιο ευρεία χρήση βρίσκει με την Python. Είναι σε θέση να εκμεταλλευτεί επίσης συμπλέγματα υπολογιστικού υλικού, όπως κάρτες γραφικών (GPU) που υποστηρίζουν τις υπολογιστικές οδηγίες NVIDIA CUDA και ειδικούς υπολογιστές τανυστών Tensor Processing Units (TPU) εξειδικευμένοι για την εκτέλεση τέτοιων υπολογισμών.

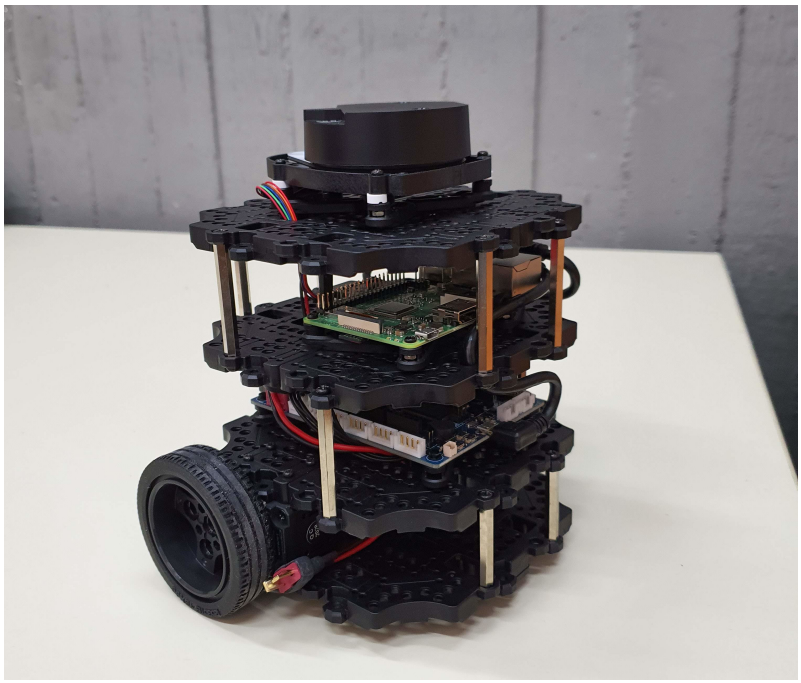
Λόγω της δυσκολίας χρήσης της βιβλιοθήκης TensorFlow, αναπτύχθηκε μια ακόμη βιβλιοθήκη ανοιχτού κώδικα, το Keras, το οποίο λειτουργεί ως διεπαφή για το TensorFlow στη γλώσσα Python. Διαθέτει έτοιμες υλοποιήσεις συχνά χρησιμοποιούμενων εργαλείων κατασκευής νευρωνικών δικτύων, όπως επίπεδα, συναρτήσεις ενεργοποίησης και βελτιστοποιητές. Διευκολύνει και επιταχύνει τη διαδικασία κατασκευής νευρωνικών δικτύων σε ένα μεγάλο βαθμό, γι΄ αυτό και πολύ συχνά οι δύο αυτές βιβλιοθήκες χρησιμοποιούνται μαζί. Πλέον το Keras αποτελεί ένα frontend του πακέτου του TensorFlow.

3.3 Turtlebot 3 Burger

3.3.1 Πλατφόρμα

Το Turtlebot 3 Burger είναι ένα έντροχο ρομποτικό όχημα του οποίου η κίνηση επιτυγχάνεται με διαφορική οδήγηση, δηλαδή από δύο τροχούς και κινητήρες οι οποίοι ελέγχονται ανεξάρτητα μεταξύ τους. Είναι μια έτοιμη ρομποτική πλατφόρμα για εκπαιδευτική ρομποτική αλλά και για πειραματικές δοκιμές. Σχεδιάστηκε από την ROBOTIS και το Open Source Robotics Foundation και είναι υλικό ανοικτού κώδικα. Διαθέτει λέιζερ αισθητήρα υπολογισμού αποστάσεων 360°(LiDAR). Ελέγχεται από την πλατφόρμα OpenCR, μια πλακέτα βασισμένη στον 32-bit μικροελεγκτή STM32, η οποία δέχεται εντολές από έναν υπολογιστή μονής πλακέτας

Raspberry Pi 3B, το οποίο εκτελεί το περιβάλλον ROS σε λειτουργικό σύστημα Ubuntu Linux [14].



Σχήμα 3.6: Το ρομποτικό όχημα Turtlebot 3 Burger του Εργαστηρίου Ευφυών Συστημάτων και Ρομποτικής του Πολυτεχνείου Κρήτης

Τα λειτουργικά χαρακτηριστικά του ρομποτικού οχήματος Turtlebot 3 Burger, παρουσιάζονται αναλυτικά στον Πίνακα 3.1.

Χαρακτηριστικά	**Τιμή**
Γραμμική ταχύτητα	$0.22m/s$
Περιστροφική ταχύτητα	$2.8rad/s$
Διαστάσεις	$138mm \times 178mm \times 192mm$
Βάρος	$1Kg$
Αισθητήρας λέιζερ	LDS-01
Ενεργητές	XL430-W250
IMU	Γυροσκόπιο και επιταχυνσιόμετρο 3 αξόνων

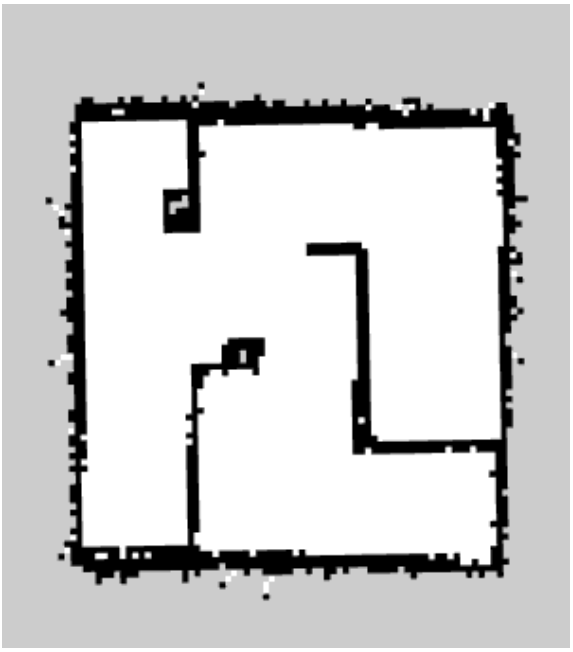
Πίνακας 3.1: Τεχνικά χαρακτηριστικά του Turtlebot 3 Burger

3.3.2 Ενωσμάτωση με ROS

Το Turtlebot 3 προορίζεται για χρήση αποκλειστικά με το ROS, καθώς διαθέτει έτοιμα πακέτα που το ενσωματώνουν πλήρως. Είναι έτοιμο για χρήση με προγράμματα για το ROS αμέσως, χωρίς επιπλέον ρυθμίσεις, όπως επίσης και πακέτα προσομοίωσης για το Gazebo. Διαθέτει επίσης διάφορα πακέτα που προσφέρουν έτοιμες λύσεις και σχετίζονται με Σιμυλτανεους Λοσαλιζατιον ανδ Μαππινγ (ΣΛΑΜ), μηχανική μάθηση (ΔΧΝ), αυτόνομη οδήγηση ή ακόμα και χειρισμό ρομποτικών αρπαγών που μπορούν να προσαρμοστούν στο έντροχο όχημα.

Συγκεκριμένα, η SLAM (Simultaneous Localization and Mapping) είναι μια μέθοδος χαρτογράφησης ενός χώρου, εκτιμώντας την τρέχουσα θέση στο χώρο συνδυάζοντας δεδομένα οδομετρίας και από τον λέιζερ μετρητή αποστάσεων. Αποτελεί μια πολύ γνωστή μέθοδο κατεύθυνσης ρομπότ σε γνωστά περιβάλλοντα, καθώς μετά την χαρτογράφηση είναι πολύ εύκολο

να δοθεί τοποθεσία - στόχος στο ρομπότ με τον ενσωματωμένο path planner του ROS και να κατευθυνθεί σε αυτή αποτελεσματικά και χωρίς συγκρούσεις.



Σχήμα 3.7: Χάρτης από τη μέθοδο SLAM [14]

Το Turtlebot 3 έρχεται επίσης με μια απλή υλοποίηση της μεθόδου ενισχυτικής μάθησης DQN. Έχει υλοποιηθεί με το Keras και το TensorFlow και παρέχει αρκετά καλά αποτελέσματα, αν και έχει πιο πολύ τη μορφή ενός proof of concept, καθώς το δίκτυο εξάγει δεδομένα μόνο για τη γωνιακή ταχύτητα, τα οποία είναι 4 διακριτές τιμές αυτής.

Διατίθενται επίσης προγράμματα που εκμεταλλεύονται κάμερα και υπολογιστική όραση, τα οποία επιτρέπουν στο ρομπότ να προσομοιώσουν αυτόνομη οδήγηση σε δημόσιους δρόμους από ένα αυτοκίνητο. Συγκεκριμένα, υπάρχουν υλοποιήσεις αναγνώρισης λωρίδων κυκλοφορίας, σημάνσεων και φαναριών. Επιπλέον υπάρχει υλοποίηση αυτόματης στάθμευσης και περιπολίας σε μια περίμετρο. Τέλος, υπάρχουν διάφορες υλοποίησεις για περισσότερα από ένα ρομπότ στην ίδια αρένα, τα οποία όλα επικοινωνούν μεταξύ τους και τους επιτρέπει για παράδειγμα να κινούνται το ένα πίσω από το άλλο.

Η OpenMANIPULATOR είναι μια αρπάγη ανοιχτού σχεδιασμού και μπορεί να ελεγχθει από ανοικτό κώδικα, και να προσαρμοσθεί στο Turtlebot 3 για την διεύρυνση των δυνατοτήτων του. Προορίζεται για την μεγαλύτερη έκδοση του ρομπότ Turtlebot 3 Waffle. Παρέχεται όχι μόνο έτοιμο υλισμικό για την υποστήριξη του επιπλέον υλικού, αλλά και ελεγκτής γραφικού περιβάλλοντος για χρήση της αρπάγης.

Κεφάλαιο 4

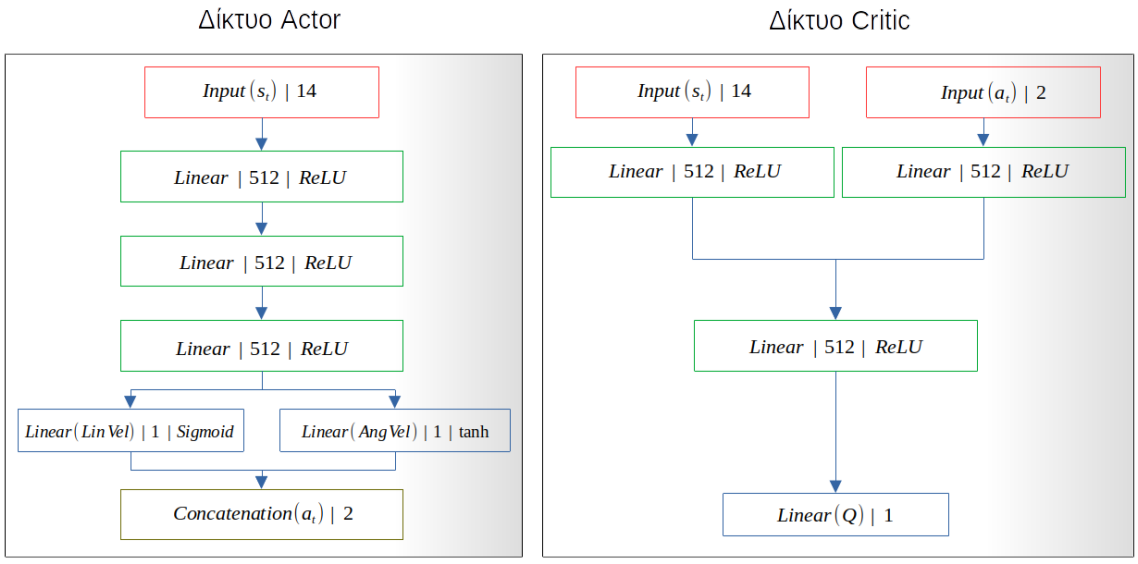
Ανάπτυξη μεθοδολογιών αυτόνομης πλοήγησης

4.1 Εισαγωγή

Σε αυτό το κεφάλαιο θα γίνει περιγραφή της μεθοδολογίας που ακολουθήθηκε για την υλοποίηση τόσο του αλγορίθμου βαθειάς ενισχυτικής μάθησης DDPG για την αυτόνομη πλοήγηση του οχήματος, όσο και των δύο ευρετικών αλγορίθμων εύρεσης μονοπατιού A*, CIA* αλλά και του συστήματος συνδυασμού αυτών των δύο. Για την υλοποίηση των παραπάνω προγραμμάτων χρησιμοποιήθηκε περιβάλλον Python και εκτελούνται ως κόμβοι στο σύστημα ROS. Για την υλοποίηση των νευρωνικών δικύων χρησιμοποιήθηκε η βιβλιοθήκη μηχανικής μάθησης Keras.

4.2 Deep Deterministic Policy Gradient

Το μοντέλο υλοποίησης αυτού του αλγορίθμου βασίζεται σε αυτά που παρουσιάστηκαν από τους H. Niu *et.al* [8] και C. Jesus *et.al* [7]. Συγκεκριμένα, βασίζεται σε 2 ζεύγη νευρωνικών δικτύων actor - critic, η δομή των οποίων απεικονίζεται στο Σχήμα 4.1.



Σχήμα 4.1: Η δομή του ζεύγους νευρωνικών δικτύων του αλγορίθμου DDPG [7]

Η κατάσταση (state) του ρομπότ αποτελείται από τις 10 μετρήσεις αποστάσεων που γίνονται από το λέιζερ, την τωρινή γραμμική και γωνιακή ταχύτητα του ρομπότ, την απόσταση από το στόχο του και τη γωνία απόκλισης από το στόχο του. Η δομή της κατάστασης του ρομπότ περιγράφεται στο Σχήμα 4.2.

Σχήμα 4.2: Η είσοδος της κατάστασης και η έξοδος των ταχυτήτων στο δίκτυο DDPG [7]

Όσον αφορά το δίκτυο actor, αυτό δέχεται ως είσοδο την κατάσταση του ρομπότ και την περνάει από 3 διαδοχικά πλήρως συνδεδεμένα κρυφά επίπεδα, το κάθε ένα με 512 νευρώνες. Την έξοδο του δικτύου αυτού την αναλαμβάνει μια ένωση 2 επιπέδων με έναν νευρώνα το κάθε ένα, με το πρώτο να έχει ως συνάρτηση ενεργοποίησης την υπερβολική εφαπτομένη ($tanh$), που εξάγει τη γωνιακή ταχύτητα αφού είναι φραγμένη στο διάστημα $[-1, 1]$ και το δεύτερο την σιγμοειδή ($sigmoid$), που εξάγει τη γραμμική ταχύτητα αφού είναι φραγμένη στο $[0, 1]$.

Το δίκτυο critic δέχεται και αυτό ως είσοδο την κατάσταση του ρομπότ, όμως δέχεται παράλληλα και ως επιπλέον είσοδο την έξοδο του δικτύου actor. Συγκεκριμένα, την κατάσταση την περνάει από ένα πλήρως συνδεδεμένο επίπεδο με 512 νευρώνες και η έξοδος αυτού ενώνεται με την έξοδο του δικτύου actor, αφού περαστεί και η ίδια από ένα πλήρως συνδεδεμένο επίπεδο με 512 νευρώνες. Οι έξοδοι από τα 2 αυτά επίπεδα περνιέται από ένα τρίτο ίδιο επίπεδο και εν τέλει, ως έξοδος υπάρχει ένα επίπεδο με μια γραμμική συνάρτηση ως συνάρτηση ενεργοποίησης, με 1 νευρώνα.

Η συνάρτηση επιβράβευσης του ρομπότ για τον αλγόριθμο DDPG αναλύθηκε σε 3 καταστάσεις, στις οποίες θα βρεθεί σίγουρα το ρομπότ κατά τη διάρκεια κίνησής του και αλλιλεπίδρασης με το περιβάλλον. Αυτές οι καταστάσεις είναι η επιτυχής εύρεση του στόχου του, η σύγκρουση με εμπόδιο και η μείωση ή αύξηση της απόστασής του από το στόχο του. Στην Εξίσωση 4.1 φαίνεται η μαθηματική έκφραση της συνάρτησης επιβράβευσης, όπου x_{min} η ελάχιστη απόσταση που εξάγεται από τον αισθητήρα αποστάσεων λέιζερ και d_t η απόσταση του ρομπότ από το στόχο του για το βήμα επανάληψης t. Η τιμή p_a είναι μια τιμωρία για πολύ υψηλές ταχύτητες περιστροφής, η τιμή p_l τιμωρεί τις πολύ χαμηλές γραμμικές ταχύτητες, η τιμή r_l αφαιρεί το άθροισμα των κανονικοποιημένων αποστάσεων λέιζερ (10 σε πλήθος), από το σύνολό τους, ενθαρρύνοντας έτσι το ρομπότ να κινείται όσο το δυνατόν πιο μακριά από εμπόδια και η τιμή της r_{obst} τιμωρεί το ρομπότ σε περίπτωση που πλησιάσει πολύ κοντά σε κάποιο εμπόδιο. Οι μαθηματικές εκφράσεις αυτών των ποσοτήτων εξηγούνται στις Εξισώσεις 4.2.

$$r = \begin{cases} 100, & d_t \leq 0.2 \\ -200, & x_{min} \leq 0.2 \\ \frac{40}{t}(d_{t-1} - d_t) + p_a + p_l + r_l + r_{obst}, & \text{αλλιώς} \end{cases} \tag{4.1}$$

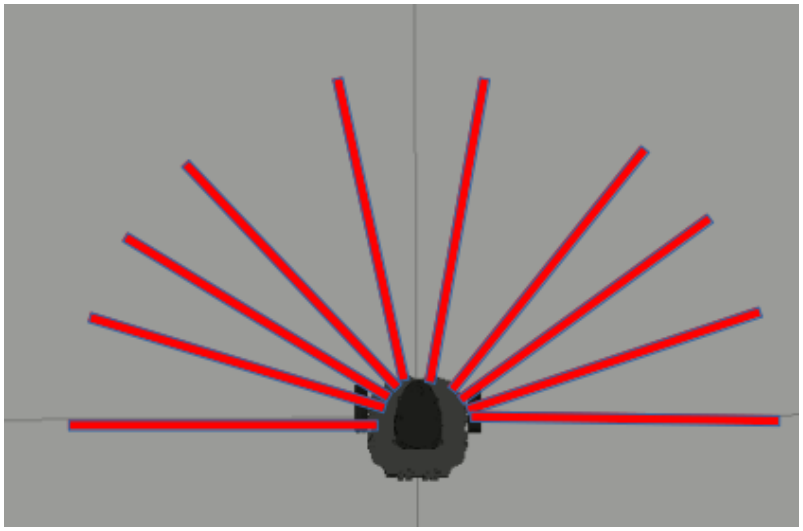
$$p_a = \begin{cases} -1, & \omega > 0.8 \text{ ή } \omega < -0.8 \\ 0, & \text{αλλιώς} \end{cases} \tag{4.2α′}$$

$$p_l = \begin{cases} -2, & v < 0.2 \\ 0, & \text{αλλιώς} \end{cases} \tag{4.2β′}$$

$$r_l = \sum_{i=0}^{9} (\frac{x_i}{3.5}) - 10 \tag{4.2γ′}$$

$$r_{obst} = \begin{cases} -80, & x_{min} < 0.3 \\ 0, & \text{αλλιώς} \end{cases} \tag{4.2δ′}$$

Όσον αφορά την εξαγωγή πληροφοριών για το περιβάλλον, αυτή πραγματοποιείται με χρήση του μετρητή αποστάσεων λέιζερ. Στη συγκεκριμένη υλοποίηση λαμβάνονται 10 ακτίνες-μετρήσεις, με την πρώτη να ξεκινάει από την αριστερή μεριά του ρομπότ και κάθετα σε αυτό και την τελευταία στην δεξιά μεριά του ρομπότ, όμοια με την πρώτη. Οι ενδιάμεσες ακτίνες τοποθετούνται η μια από την άλλη με διαφορά 18 μοιρών. Με αυτό τον τρόπο, λαμβάνονται δεδομένα για το περιβάλλον μόνο από την μπροστινή μεριά του ρομπότ, δηλαδή από αυτήν με την οποία είναι και η φορά της κίνησής του. Στο Σχήμα 4.3 φαίνεται η σχηματική απεικόνιση των ακτίνων λέιζερ από τις οποίες λαμβάνονται μετρήσεις.



Σχήμα 4.3: Οι μετρήσεις που λαμβάνονται από τον αισθητήρα αποστάσεων, για την απεικόνιση του περιβάλλοντος.

4.2.1 Υλοποίηση

Το μοντέλο αυτό υλοποιείται σε κώδικα Python, με χρήση της βιβλιοθήκης Keras. Αρχικά υλοποιείται κατάλληλη κλάση υπεύθυνη για τη διαχείριση της μνήμης, ονόματι Buffer. Μετά από κάθε βήμα προσομοίωσης, αυτή διατηρεί την MDP σε μια λίστα ορισμένου μεγέθους, ενώ όταν ζητηθεί για την εκπαίδευση λαμβάνει ένα δείγμα ορισμένου μεγέθους. Επιπλέον,

αυτή η κλάση αναλαμβάνει και την εκπαίδευση των δικτύων, η οποία επιτυγχάνεται όμοια με τις εξισώσεις στην Ενότητα 2.3.5. Για την οπισθοδρόμηση του σφάλματος χρησιμοποιείται ο βελτιστοποιητής Adam για όλα τα νευρωνικά δίκτυα.

Υλοποιείται επίσης κλάση για τον θόρυβο OU, όπως περιγράφεται στην Ενότητα 2.4.1. Συγκεκριμένα, υλοποιείται η παραλλαγή αυτής με τον παράγοντα απόκλισης μ. Επιπλέον, υλοποιούνται ρουτίνες για την συναρμολόγηση των νευρωνικών δικτύων Actor και Critic, όπως ακριβώς περιγράφεται στο Σχήμα 4.1. Αυτό επιτυγχάνεται με τη χρήση των έτοιμων επιπέδων του Keras, συγκεκριμένα τα Input, Dense, Concatenate αλλά και των έτοιμων συναρτήσεων ενεργοποίησης linear, sigmoid, tanh. Δημιουργείται ρουτίνα εξαγωγής της κατάστασης αλλά και της πολιτικής, εκ των οποίων η πρώτη λαμβάνει δεδομένα από τα αισθητήρια του ρομπότ μέσω των κατάλληλων μηνυμάτων στο ROS και η δεύτερη εξάγει βάσει της κατάστασης την κατάλληλη ενέργεια, συμπεριλαμβανομένου και του θορύβου.

Υλοποιήθηκαν επιπλέον βοηθητικές συναρτήσεις, όπως για τη συνάρτηση ανταμοιβής, συνάρτηση που ελέγχει αν το ρομπότ βρίσκεται στη θέση - στόχο που του έχει δοθεί και συνάρτηση για την καταγραφή των δεδομένων κατά τη διάρκεια της προσομοίωσης. Μετά κατασκευάζεται η βασική συνάρτηση εκτέλεσης της διαδικασίας εκπαίδευσης, που αρχικά εισάγει προηγούμενα βάρη εφόσον εντοπιστούν, πραγματοποιεί αρχικοποίηση για όλες τις απαραίτητες μεταβλητές και κλάσεις και εισέρχεται στον εξωτερικό βρόχο επανάληψης. Αυτός εκτελείται για το πλήθος των επεισοδίων που έχουν οριστεί. Στον εσωτερικό βρόχο επανάληψης, στον οποίο περιορίζονται τα βήματα επανάληψης και κατά συνέπεια η διάρκεια του επεισοδίου, λαμβάνεται η κατάσταση και εισάγεται στο δίκτυο actor, από το οποίο εξάγονται δεδομένα για την πολιτική του πράκτορα. Η πολιτική εκτελείται και λαμβάνεται η νέα κατάσταση και η ανταμοιβή βάσει της κατάστασης, καταγράφεται η MDP στη μνήμη και ενημερώνονται τα δίκτυα κατάλληλα, εφόσον έχουν συγκεντρωθεί αρκετές εμπειρίες, τουλάχιστον όσες και το μέγεθος της δειγματοληψίας. Εκτελούνται έλεγχοι για την εύρεση του στόχου, σύγκρουσης ή εάν το επεισόδιο βγήκε εκτός χρόνου. Τέλος, καταγράφεται η MDP μαζί με πληροφορίες για τη θέση του ρομπότ στο χώρο και το τρέχον επεισόδιο στο αρχείο καταγραφής.

Κατά την έξοδο αποθηκεύονται τα βάρη σε αρχεία, εκτυπώνεται πόσες φορές έφτασε το στόχο, συγκρούστηκε ή βγήκε εκτός χρόνου και εξάγεται γράφημα με τη μέση τιμή της ανταμοιβής ανά επεισόδιο.

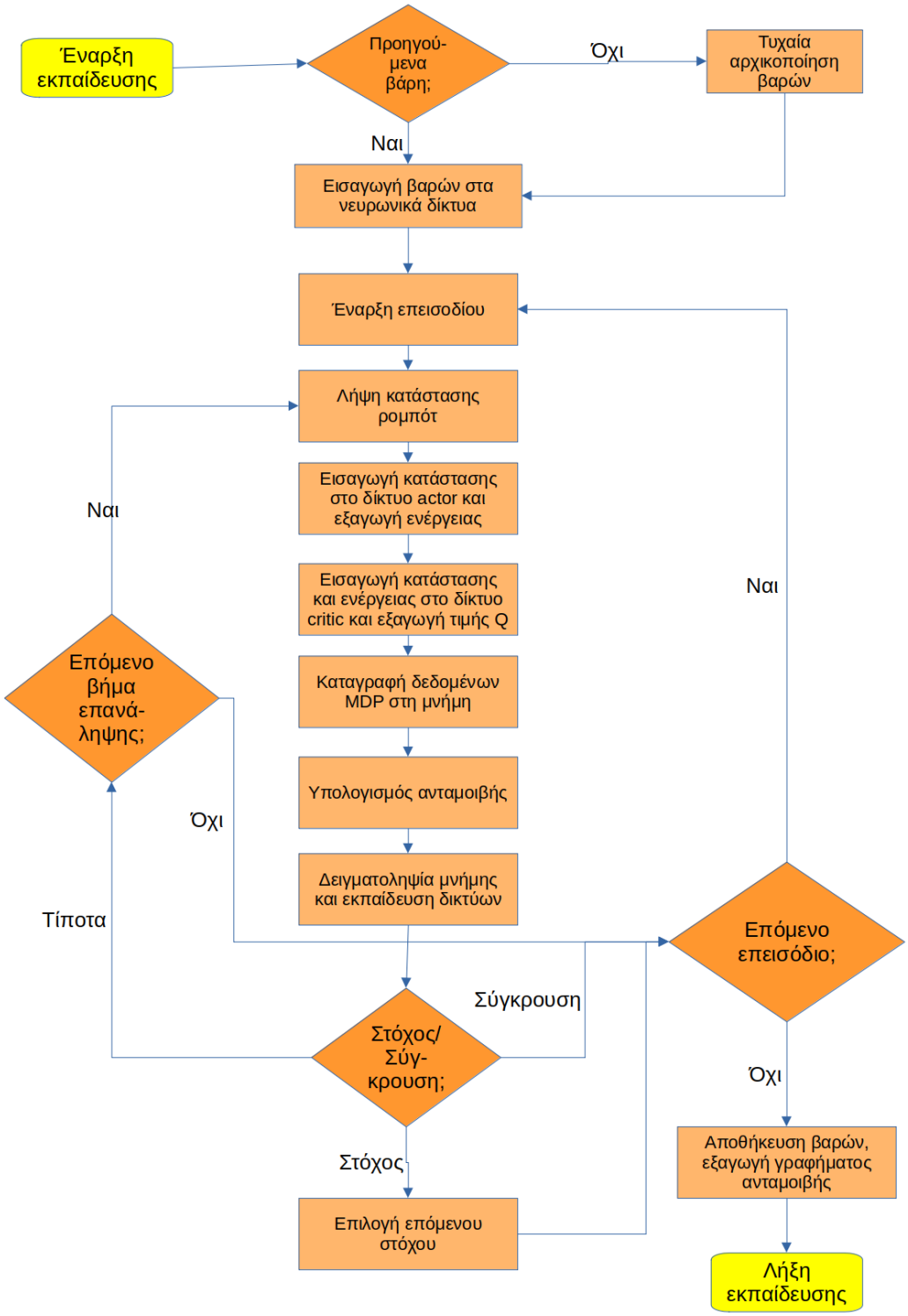
Στο Σχήμα 4.4 φαίνεται το διάγραμμα ροής που περιγράφει σχηματικά την παραπάνω διαδικασία.

4.2.2 Εκπαίδευση

Περιγραφή διαδικασίας

Η εκπαίδευση του νευρωνικού δικτύου αποτελεί ένα από τα σημαντικότερα βήματα για τη σωστή λειτουργία του αλγορίθμου. Αυτή επιτυγχάνεται σταδιακά, ξεκινώντας από ευκολότερα περιβάλλοντα, δηλαδή περιβάλλοντα χωρίς εμπόδια ή με λίγα, και στην πορεία προστίθενται επιπλέον εμπόδια ή και διάδρομοι, αυξάνοντας έτσι το βαθμό δυσκολίας του περιβάλλοντος. Χρησιμοποιήθηκαν τρία περιβάλλοντα, συγκεκριμένα τα δύο από τα τρία περιβάλλοντα που διατίθενται μαζί με το Turtlebot 3 για χρήση στις εφαρμογές μηχανικής μάθησης και ένα κατασκευασμένο στα πλαίσια αυτής της εργασίας.

Στην αρχή της διαδικασίας εκπαίδευσης, όπου πρακτικά το δίκτυο δεν διαθέτει καμία εμπειρία, το ρομπότ λαμβάνει τυχαίες τιμές ταχυτήτων, το οποίο το οδηγούν σε κάποια σύγκρουση με εμπόδιο. Μετά από μερικές εμπειρίες, παρατηρείται ότι το ρομπότ έχει την τάση να κινείται κυκλικά ή σε σπιράλ. Μόλις αποκτήσει μερικές εμπειρίες, με κάποιες από αυτές να είναι επιτυχείς, δηλαδή να φτάνει με επιτυχία στο στόχο του, παρατηρείται βελτίωση στη μετάβασή του από τη θέση εκκίνησης στη θέση στόχου.



Σχήμα 4.4: Διάγραμμα ροής που περιγράφει τη διαδικασία εκπαίδευσης των νευρωνικών δικτύων DDPG

Δίνεται στο ρομπότ μια αλληλουχία θέσεων - στόχων, στις οποίες πρέπει να μεταβεί με επιτυχία. Μόλις μεταφερθεί με επιτυχία στη θέση που του έχει οριστεί, τότε του δίνεται κάποια άλλη θέση - στόχος σε διαφορετικό σημείο του περιβάλλοντος. Αυτή η διαδικασία επαναλαμβάνεται μέχρις ότου να ολοκληρωθούν τα ορισμένα επεισόδια. Μετά από τη λήξη κάθε επεισοδίου, το ρομπότ μεταφέρεται σε μια θέση εκκίνησης εντός του περιβάλλοντος, η οποία αλλάζει μόνο σε περίπτωση επιτυχούς επεισοδίου. Παρατηρήθηκε ότι με τον εξαναγκασμό του ρομπότ να πρέπει να προσεγγίσει το στόχο του από διαφορετικό σημείο έναρξης κάθε φορά, βελτιώνεται σημαντικά τόσο η ταχύτητα όσο και η αποτελεσματικότητα εκπαίδευσης, σε σχέση με ένα σταθερό σημείο εκκίνησης. Με αυτή τη μέθοδο, το ρομπότ αποκτά περισσότερες εμπειρίες με αποτέλεσμα την πιο πλήρη εκπαίδευσή του. Η αλληλουχία θέσεων - στόχων ανακατεύεται τυχαία κάθε φορά που περνάει το ρομπότ με επιτυχία από όλες τους.

Στις αλλαγές περιβάλλοντος, πάντα λαμβάνονται τα βάρη από το προηγούμενο περιβάλλον. Αυτό επιταχύνει σημαντικά τη διαδικασία εκπαίδευσης, καθώς θα ήταν πολύ χρονοβόρο να εκκινούταν η εκπαίδευση σε κάθε περιβάλλον από την αρχή. Το ρομπότ στην αρχή δυσκολεύεται και η κινηματική του συμπεριφορά είναι απρόβλεπτη, καθώς με την αλλαγή του περιβάλλοντος τα δεδομένα κατάστασης διαφέρουν σημαντικά με αυτά στα οποία είχε εκπαιδευτεί πρίν. Όμως, μετά από αρκετές εμπειρίες, το πλήθος των οποίων ποικίλλει ανάλογα με την πολυπλοκότητα του περιβάλλοντος, μαθαίνει να αποφεύγει επιτυχώς τα νέα εμπόδια τα οποία του παρουσιάζονται στο νέο περιβάλλον.

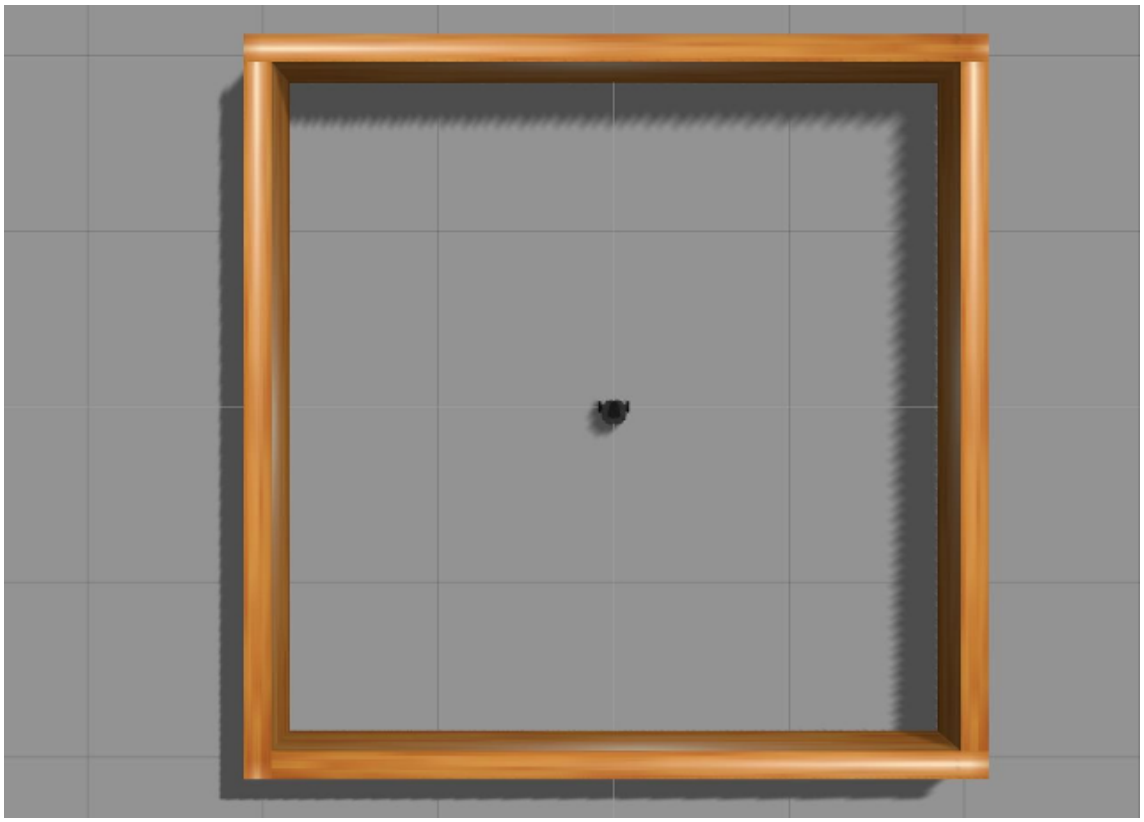
Όσον αφορά τις υπερπαραμέτρους της εκπαίδευσης του νευρωνικού δικτύου, αυτές βρέθηκαν με τη μέθοδο δοκιμής και σφάλματος. Συγκεκριμένα, εάν οι ρυθμοί εκμάθησης είναι πολύ μεγάλοι, τότε το νευρωνικό δίκτυο εξάγει δεδομένα ταχύτητας τα οποία δεν βοηθούν το ρομπότ να φτάσει στο στόχο του ή το οδηγούν συνεχώς προς συγκρούσεις. Άλλο ένα ανεπιθύμητο σενάριο είναι το ρομπότ να καταλήγει να κάνει κύκλους, είτε γύρω από τον εαυτό του είτε μικρής ακτίνας. Αντιθέτως, εάν οι ρυθμοί είναι πολύ μικροί, τότε η εκπαίδευση χρειάζεται πάρα πολύ μεγάλο χρόνο προσομοίωσης ώστε να διαθέτει μια ικανοποιητική εμπειρία. Στον Πίνακα 4.1 φαίνονται οι υπερπαράμετροι οι οποίοι χρησιμοποιήθηκαν στα πλαίσια αυτής της εργασίας για μια ικανοποιητική εκπαίδευση του νευρωνικού δικτύου.

Υπερπαράμετρος	**Τιμή**
Ρυθμός εκμάθησης δικτύου actor	0.0001
Ρυθμός εκμάθησης δικτύου critic	0.0001
τ	0.01
γ	0.90
Μέγεθος μνήμης	200000
Μέγεθος δείγματος	512
Μέγιστο πλήθος επαναλήψεων	700

Πίνακας 4.1: Υπερπαράμετροι εκπαίδευσης του νευρωνικού δικτύου

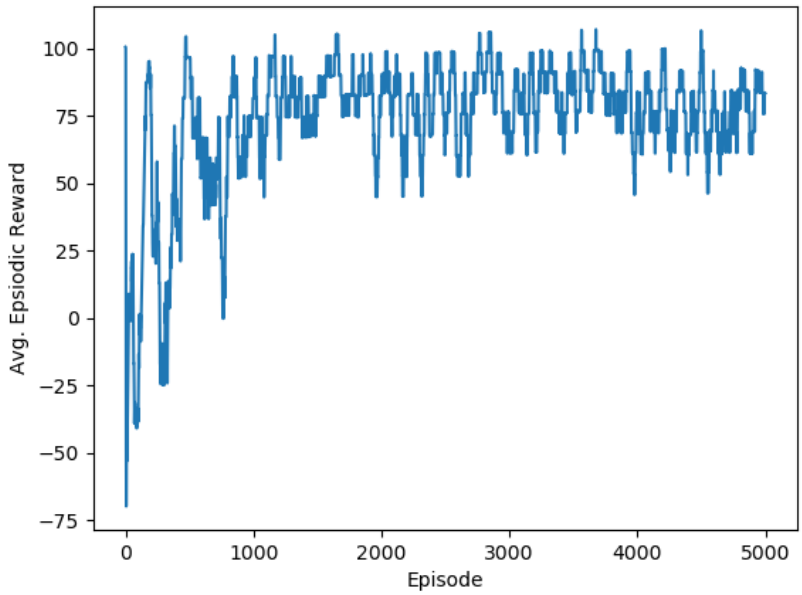
Πρώτο περιβάλλον εκπαίδευσης

Στο πρώτο περιβάλλον εκπαίδευσης, στόχος ήταν να εκπαιδευτεί ο πράκτορας κατάλληλα ώστε να μετακινείται με επιτυχία στο στόχο του, εντός του χρονικού ορίου και χωρίς να προσπαθήσει να βγει εκτός του περιβάλλοντος, δηλαδή χωρίς να συγκρούεται με την περίφραξη του χώρου (Σχήμα 4.5. Το ρομπότ τοποθετούταν στην θέση (0.5, 0.5) στην αρχή κάθε επεισοδίου και του δινόταν ένας στόχος στο χώρο. Για τα πρώτα 500-600 επεισόδια, ο πράκτορας μη έχοντας προηγούμενες εμπειρίες (τα δίκτυα αρχικοποιήθηκαν με τυχαία βάρη), αντιμετώπιζε δυσκολία στην πλοήγηση στο χώρο, με αποτέλεσμα είτε να συγκρούεται με την περίφραξη, είτε να εκτελεί περιστροφές γύρω από τον εαυτό του μέχρις ότου να λήξει ο χρόνος του επεισοδίου. Σε μικρό χρονικό διάστημα, όμως, κατάφερνε και προσέγγιζε τους στόχους του, ολοένα και καλύτερα με το πέρασμα των επεισοδίων και κατά συνέπεια την απόκτηση νέων εμπειριών. Υπήρξαν στόχοι τους οποίους προσέγγιζε απευθείας, υπήρξαν όμως και κάποιοι για τους οποίους χρειαζόταν περίπου 30 προσπάθειες για να τους καταφέρει. Οι στόχοι στους οποίους αντιμετώπιζε δυσκολία ήταν συνήθως αυτοί που ήταν πολύ κοντά στην περίφραξη (περίπου $0.5m$ από αυτή).



Σχήμα 4.5: Το πρώτο περιβάλλον εκπαίδευσης, με μοναδικό εμπόδιο η περίφραξη του χώρου $4m \times 4m$

Στο Σχήμα 4.6 φαίνεται το γράφημα που απεικονίζει τη μέση ανταμοιβή ανά επεισόδιο. Παρατηρείται η σταδιακή αύξηση της ανταμοιβής κατά τα πρώτα 500 επεισόδια και κατόπιν η σχετική σταθερότητα στις υψηλές τιμές ανταμοιβών. Οι αυξομειώσεις που παρατηρούνται στα 1000 - 5000 επεισόδια δικαιολογούνται με τις δυσκολίες που συναντούσε ο πράκτορας, που περιγράφηκαν παραπάνω. Μετά την εκτέλεση της εκπαίδευσης, ο πράκτορας είχε την ικανότητα να κατευθύνει το ρομπότ με επιτυχία εντός του άδειου περιβάλλοντος.

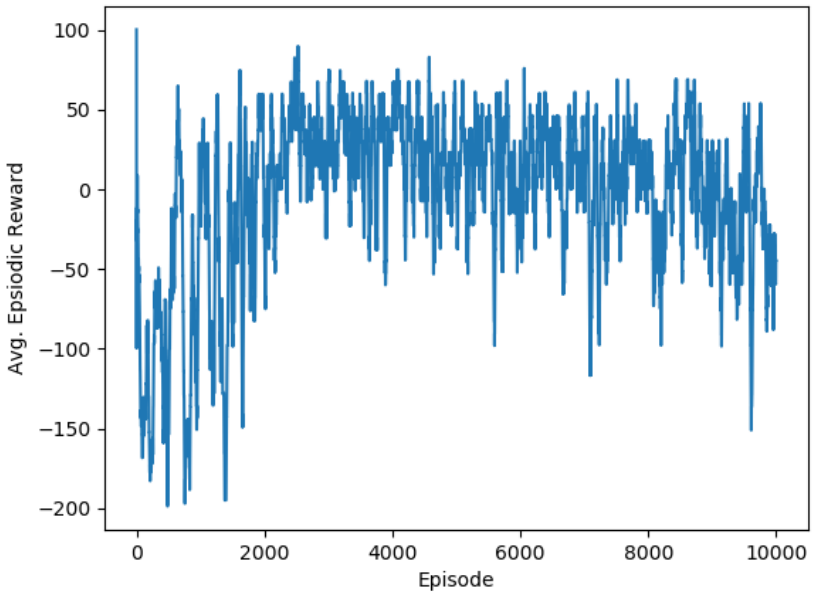


Σχήμα 4.6: Μέση τιμή ανταμοιβής ανά επεισόδιο, για το πρώτο περιβάλλον εκπαίδευσης

Δεύτερο περιβάλλον εκπαίδευσης

Στο δεύτερο περιβάλλον, τοποθετούνται στον ίδιο χώρο 4 κυλινδρικά εμπόδια στο κέντρο. Με αυτό τον τρόπο, ο πράκτορας καλείται να μάθει να αναγνωρίζει αυτά τα εμπόδια μέσω του λέιζερ και να κινείται γύρω τους, ώστε να φτάσει στο στόχο που του δίνεται. Η διαδικασία της εκπαίδευσης εκκινείται χρησιμοποιώντας τα βάρη για τα νευρωνικά δίκτυα που προέκυψαν από το προηγούμενο περιβάλλον εκπαίδευσης. Αυτό έχει ως αποτέλεσμα την επιτάχυνση της διαδικασίας, όμως παρατηρείται ότι ο πράκτορας αρχικά συγκρούεται συχνά με τα κυλινδρικά εμπόδια, καθώς από την εκπαίδευση από το προηγούμενο περιβάλλον ο πράκτορας γνωρίζει ότι δεν υπάρχουν εμπόδια σε εκείνα τα σημεία. Σύντομα, ο πράκτορας μαθαίνει να αποφεύγει τα εμπόδια, όμως υπάρχουν σενάρια που εξακολουθεί να συγκρούεται με τα εμπόδια. Αυτό οφείλεται στο συνδυασμό θέσης εκκίνησης με θέση στόχου, με τον οποίο μπορεί να μη διαθέτει τόσες πολλές εμπειρίες. Το κλειδί της επιτυχούς εκπαίδευσης σε αυτή την περίπτωση είναι να δοθούν στον πράκτορα όσοι περισσότεροι συνδυασμοί είναι δυνατοί, από αρκετές φορές τον καθέναν, με στόχο την πλήρη εκπαίδευσή του. Γι΄ αυτό το λόγο, χρειάστηκαν 10000 επεισόδια για να κατέχει ένα ικανοποιητικό επίπεδο εμπειριών.

Το γράφημα της μέσης ανταμοιβής ανά επεισόδιο φάινεται στο Σχήμα 4.7. Παρατηρείται μια ανοδική τάση του γραφήματος τμηματικά, με μερικές πτώσεις στα αρνητικά. Η ανοδική τάση δείχνει ότι το ρομπότ αποκτά εμπειρίες και αναπτύσσει την ικανότητα να κινείται στο περιβάλλον αποτελεσματικότερα. Οι πτώσεις στα αρνητικά αφορούν τις αλλαγές του συνδυασμού σημείου εκκίνησης και στόχου, ο οποίος κάποιες φορές δυσκόλευε ιδιαίτερα τον πράκτορα, με αποτέλεσμα να συγκρούεται πιο συχνά με εμπόδια, λόγω λανθασμένων αποφάσεων.

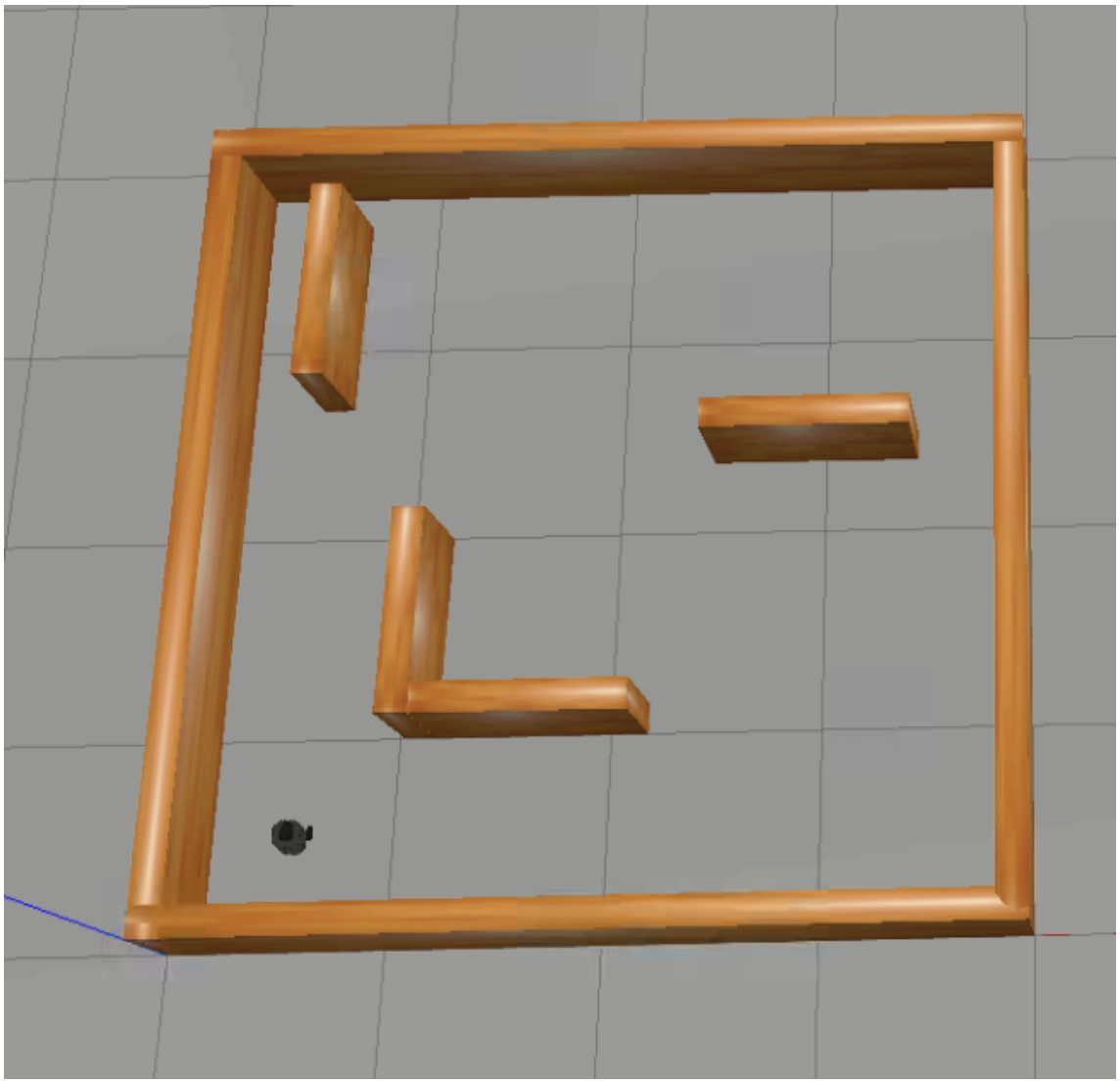


Σχήμα 4.7: Μέση τιμή ανταμοιβής ανά επεισόδιο, για το δεύτερο περιβάλλον εκπαίδευσης

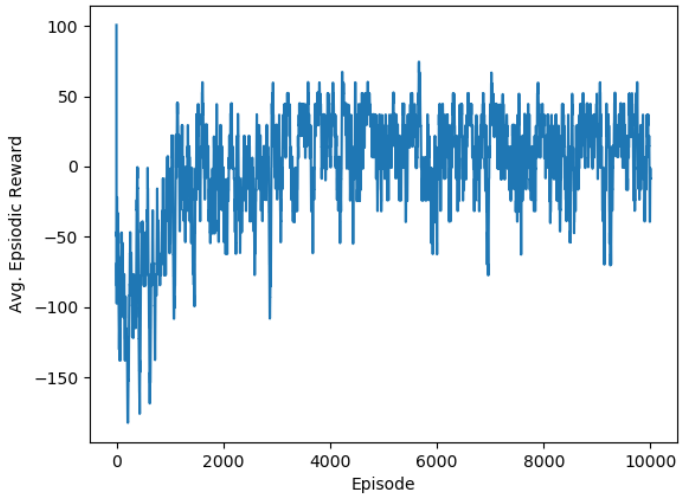
Τρίτο περιβάλλον εκπαίδευσης

Σε αυτό το περιβάλλον, τοποθετούνται τρία ογκώδη εμπόδια, υπό τη μορφή τοίχων, όπως παρουσιάζονται στο Σχήμα 4.8. Σκοπός του περιβάλλοντος είναι όχι μόνο ο πράκτορας να μάθει να ανταπεξέρχεται σε ογκώδη εμπόδια όπως αυτά, αλλά και να κινείται σε στενούς διαδρόμους, όπως αυτόν που σχηματίζεται στην κάτω αριστερή γωνία του περιβάλλοντος, μεταξύ περίφραξης και του εμποδίου που είναι ορθή γωνία. Όπως και πριν, η διαδικασία εκπαίδευσης γίνεται χρησιμοποιώντας τα βάρη που προέκυψαν από το προηγούμενο περιβάλλον, για την επιτάχυνση και την αποτελεσματικότερη λειτουργία της διαδικασίας.

Το γράφημα της μέσης τιμής ανταμοιβής ανά επεισόδιο για το τρίτο περιβάλλον, φαίνεται στο Σχήμα 4.9. Παρατηρούνται σημαντικές διαφορές συγκριτικά με τα προηγούμενα 2 γραφήματα (Σχήματα 4.6 και 4.7), συγκεκριμένα πολύ πιο έντονη αστάθεια και διαταρραχή στην τιμή της ανταμοιβής. Αυτό συμβαίνει επειδή η απαιτητικότητα του περιβάλλοντος αυξήθηκε απότομα, με αποτέλεσμα ο πράκτορας να αντιμετωπίζει ιδιαίτερη δυσκολία στην μετακίνησή του και πιο πολύ στην αποφυγή των εμποδίων του περιβάλλοντος. Το μεγαλύτερο πρόβλημα στον πράκτορα το δημιουργούσε το εμπόδιο σε σχήμα ορθής γωνίας στο κάτω αριστερά τμήμα του περιβάλλοντος, λόγω του στενού διαδρόμου που δημιουργούταν μεταξύ της περίφραξης και αυτού. Ο πράκτορας συγκρούστηκε πολλαπλές φορές με αυτό το εμπόδιο, όμως με τις πολλές εμπειρίες του ανέπτυξε μια συμπεριφορά όπου διατηρείται όσο πιο κεντρικά μπορεί στο διάδρομο και κινείται με πιο αργές από τις συνηθισμένες γραμμικές ταχύτητες. Μόλις εξέρχεται από αυτό το τμήμα, κινείται με πιο μεγάλες ταχύτητες.



Σχήμα 4.8: Το τρίτο περιβάλλον εκπαίδευσης, με διάφορα μεγάλα εμπόδια



Σχήμα 4.9: Μέση τιμή ανταμοιβής ανά επεισόδιο, για το τρίτο περιβάλλον εκπαίδευσης

4.3 Υλοποίηση αλγορίθμων συντομότερου μονοπατιού

4.3.1 A*

Η υλοποίηση του αλγορίθμου A* επιτυγχάνθηκε με τη βοήθεια του κώδικα [15]. Συγκεκριμένα, επειδή το περιβάλλον εκφράζεται ως γράφημα στα πλαίσια αυτού του αλγορίθμου, αρχικά υλοποιείται δομή δεδομένων που διατηρεί τους κόμβους του γραφήματος. Διατηρείται η θέση του κόμβου στο περιβάλλον, ο κόμβος - γονιός του, δηλαδή από ποιον κόμβο βρέθηκε το ρομπότ σε αυτόν και οι τιμές f, g, h του ευρετικού σκέλους του αλγορίθμου.

Η διαδικασία αναζήτησης, όπως περιγράφεται και στο Κεφάλαιο 2.1.2, υλοποιείται με χρήση των 2 δομών OPEN και CLOSED, υπό τη μορφή λιστών στη γλώσσα Python. Σε κάθε επανάληψη αφαιρείται ο κόμβος με την ελάχιστη απόσταση f από τη λίστα OPEN και τοποθετείται στην CLOSED, δίνεται εντολή στο ρομπότ να μετακινηθεί σε αυτόν και πραγματοποιείται αναζήτηση για τους επόμενους υποψήφιους κόμβους που τοποθετούνται στη λίστα OPEN, αφού υπολογισθούν οι ευρετικές τιμές αυτών. Η εύρεση των εμποδίων πραγματοποιείται με χρήση του αισθητήρα λέιζερ, συγκεκριμένα μετατρέποντας τα δεδομένα αποστάσεων σε σημεία στο χώρο, ή αλλιώς νέφος σημείων ή point cloud, και ελέγχοντας αν υπάρχει σημείο στην υποψήφια θέση. Ο αλγόριθμος τερματίζει όταν αδειάσει η λίστα OPEN, ή μετά από έναν συγκεκριμένο αριθμό επαναλήψεων, για να αποφύγονται ατέρμονοι βρόχοι. Στο τέλος δίνεται επιλογή στο χρήστη με ειδική συνάρτηση να εκτυπώσει και να αποθηκεύσει το συντομότερο μονοπάτι.

4.3.2 CIA*

Η υλοποίηση του CIA* ακολουθεί την φιλοσοφία που έχει προταθει στο [3] και είναι ανάλογη με την υλοποίηση του A*. Η βασική διαφορά είναι ότι υλοποιήθηκε συνάρτηση για το πληροφορούμενο ευρετικό σκέλος του αλγορίθμου, όπου υπολογίζεται το παραλληλόγραμμο γύρω από την περιοχή ενδιαφέροντος, όπως περιγράφεται στο [3]. Μια ακόμη διαφοροποίηση είναι η αποθήκευση των εμποδίων σε ξεχωριστή λίστα, η οποία ενώνεται με την λίστα CLOSED και προκύπτει η λίστα BLOCKED, η οποία εκφράζει τους κόμβους οι οποίοι είτε έχουν ήδη εξερευνυθεί, είτε διαθέτουν εμπόδιο και δεν μπορούν να εξερευνυθούν. Τέλος, υλοποιείται όπως περιγράφεται και στη θεωρία η υπορουτίνα επανυπολογισμού του ευρετικού σκέλους όλων των κόμβων στη λίστα OPEN αμέσως μετά την αφαίρεση κάποιου στοιχείου από αυτή, για την αποφυγή λαθών.

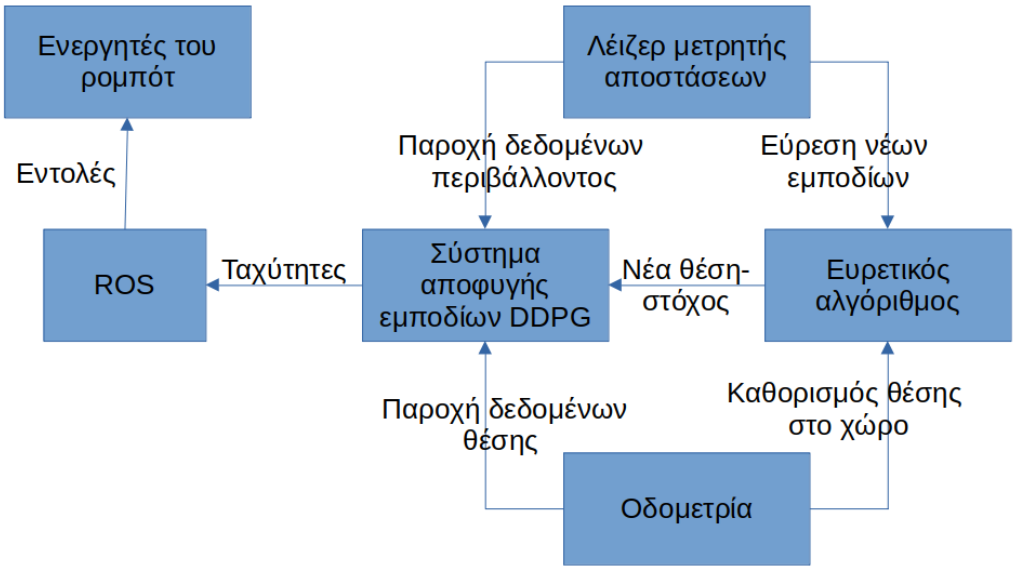
4.4 Σύστημα σχεδιασμού τροχιάς και αποφυγής εμποδίων

Το σύστημα αποφυγής εμποδίων υλοποιήθηκε για χρήση σε συνδυασμό με τους αλγορίθμους εύρεσης συντομότερης διαδρομής ως συνδετικός κρίκος μεταξύ των αλγορίθμων και του νευρωνικού δικτύου. Η κύρια λειτουργία του είναι τη μετάβαση του ρομπότ στη θέση - στόχο την οποία εξάγουν οι αλγόριθμοι, αποφεύγοντας τυχόν εμπόδια στο δρόμο και κατ΄ επέκταση συγκρούσεις. Εκμεταλλεύεται το εκπαιδευμένο νευρωνικό δίκτυο actor για την εξαγωγή δεδομένων για τις ταχύτητες του ρομπότ, βάσει την κατάσταση στην οποία βρίσκεται και το στόχο ο οποίος του έχει δοθεί.

Για την καλύτερη περιγραφή της διαδικασίας, στο Σχήμα 4.10 φαίνεται η ροή των δεδομένων μεταξύ των ευρετικών αλγορίθμων, του συστήματος αποφυγής εμποδίων DDPG και της οδομετρίας και του αισθητήρα λέιζερ. Ο ευρετικός αλγόριθμος αποφασίζει, μέσω του αισθητήρα λέιζερ, που δεν υπάρχει εμπόδιο ώστε να μεταβεί εκεί το ρομποτικό όχημα, δίνεται η νέα θέση στόχος στο σύστημα DDPG το οποίο αναλαμβάνει τη ρύθμιση της γραμμικής και της

γωνιακής ταχύτητας του ρομπότ με στόχο τη μετακίνηση του οχήματος στην εν λόγω θέση. Έπειτα, το ROS αναλαμβάνει τη μετάφραση των 2 αυτών τιμών σε εντολές κατάλληλες για τους ενεργητές του ρομπότ και έτσι επιτυγχάνεται η κίνηση αυτού. Μόλις πραγματοποιηθεί η μετάβαση, ο ευρετικός αλγόριθμος επαναλαμβάνει την ίδια διαδικασία, λαμβάντοντας παράλληλα και δεδομένα για την τρέχουσα θέση του ρομπότ μέσω της οδομετρίας, μέχρις ότου να αφιχθεί στη θέση στόχο του.

Λόγω του τρόπου με τον οποίο αυτοί οι αλγόριθμοι αντιλαμβάνονται το περιβάλλον τους, δηλαδή σε ένα πλέγμα συγκεκριμένων διαστάσεων (βλέπε Σχήμα 2.2), απαιτείται κάποια διακριτοποίηση του περιβάλλοντος, προκειμένου να επιτυγχάνεται σωστή επικοινωνία μεταξύ συστήματος σχεδιασμού τροχιάς και αυτών. Στα πλαίσια αυτής της εργασίας, η διακριτοποίηση γίνεται στα $0.5m$, πράγμα το οποίο σημαίνει ότι ο αλγόριθμος δίνει εντολές στο ρομπότ να μετακινείται ανά $0.5m$ τη φορά, είτε στον άξονα x ή στον άξονα y. Διαγώνιες κινήσεις, δηλαδή κινήσεις και στους δύο άξονες, δεν έχουν υλοποιηθεί. Με τις δοκιμές που εκτελέστηκαν σε συνδυασμό με το σύστημα, παρατηρήθηκε ότι δεν απαιτείται πιο αναλυτική διακριτοποίηση του περιβάλλοντος. Όμως, ενδέχεται σε πιο σύνθετα περιβάλλοντα να απαιτείται πιο αναλυτική διακριτοποίηση, σε πιο μικρές τιμές βήματος.



Σχήμα 4.10: Ροή δεδομένων μεταξύ ευρετικού αλγορίθμου, συστήματος αποφυγής εμποδίων, οδομετρίας και αισθητήρα λέιζερ.

4.5 Βοηθητικά εργαλεία

Πλέον των παραπάνω εργαλείων, υλοποιήθηκαν και βοηθητικά εργαλεία, για την υποστήριξη της λειτουργία των παραπάνω προγραμμάτων. Συγκεκριμένα, υλοποιήθηκε η κλάση simulation, η οποία είναι υπεύθυνη για τη διαχείριση του προσομοιωτή Gazebo και υποστηρίζει λειτουργίες όπως την κίνηση του ρομπότ σε συγκεκριμένο σημείο, την τυχαία επιλογή θέσης για το ρομπότ και την επαναφορά της αρένας στην αρχική της κατάσταση.

Η κλάση navigation είναι υπεύθυνη για την κίνηση του ρομπότ και την ανάγνωση δεδομένων από το περιβάλλον του. Συγκεκριμένα, διαθέτει συνάρτηση για την αποστολή μηνυμάτων γραμμικής και γωνιακής ταχύτητας στο αντίστοιχο θέμα του ρομπότ, την ανάγνωση και επιστροφή δεδομένων οδομετρίας, την μετατροπή δεδομένων αποστάσεων από τον αισθητήρα

λέιζερ σε νέφος σημείων και τον έλεγχο ύπαρξης εμποδίου στο χώρο σε συγκεκριμένες συντεταγμένες στο πλαίσιο συντεταγμένων του περιβάλλοντος με χρήση του νέφους.

Η κλάση framework λειτουργεί ως συνδετικός κρίκος μεταξύ των αλγορίθμων συντομότερης διαδρομής και του συστήματος αποφυγής εμποδίων. Εκτελεί τον αλγόριθμο εύρεσης συντομότερης διαδρομής, εξάγει τα δεδομένα θέσης από αυτόν και τα εισάγει στο σύστημα αποφυγής εμποδίων, το οποίο αναλαμβάνει την κατεύθυνση του ρομπότ στη ζητούμενη θέση. Επιπλέον, αυτή καταγράφει και εξάγει κινηματικά δεδομένα αλλά και δεδομένα περιβάλλοντος από τη διαδικασία, με στόχο την οπτικοποίηση αυτών στα πλαίσια αυτής της εργασίας.

Τέλος, υλοποιήθηκε και μια κλάση για την καταγραφή των δεδομένων από τις MDP κατά τη διάρκεια της προσομοίωσης και την εξαγωγή και αποθήκευση αυτών σε αρχείο κειμένου τύπου csv.

Κεφάλαιο 5

Πειραματικά Αποτελέσματα

5.1 Μεθοδολογία

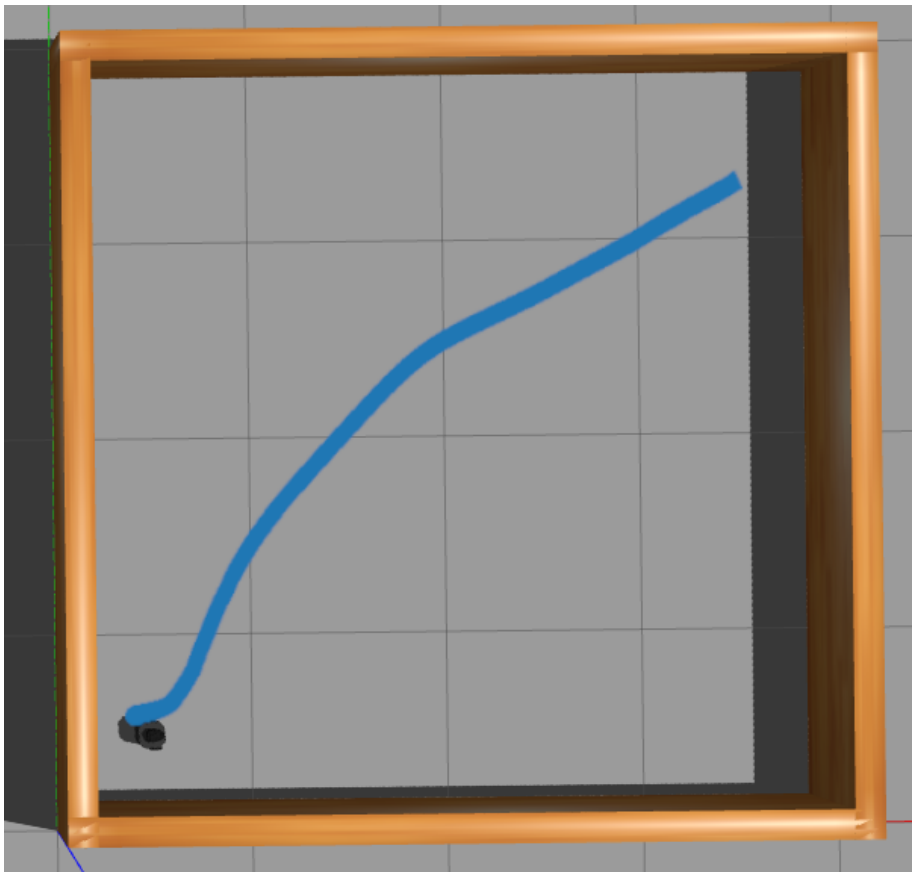
Προκειμένου να εξακριβωθεί η ορθή λειτουργία τόσο του συστήματος αποφυγής εμποδίων και εύρεσης του σημείου στόχου, έγινε σειρά πειραματών με προκαθορισμέμους στόχους εξωτερικά από τον χρήστη. Για τα παραπάνω πειράμτα χρησιμοποιήθηκαν τα 3 περιβάλλοντα που περιγράφηκαν αναλυτικά στην προηγούμενη ενότητα.

Έπειτα, έχοντας βεβαιωθεί ότι τα νευρωνικά δίκτυα έχουν εκπαιδευθεί κατάληλλα, μπορούμε να εκμεταλλευτούμε τη λειτουργία τους σε συνδυασμό με τους ευρετικούς αλγορίθμους εύρεσης της συντομότερης διαδρομής σε ένα περιβάλλον με εμπόδια. Τελικός στόχος είναι η εξερεύνυση και η πλοήγηση σε ένα άγνωστο περιβάλλον, όπου ο ευρετικός αλγόριθμος, θα εξάγει θέσεις στόχους στο χώρο στις οποίες θα πρέπει να μεταβεί το ρομπότ και το ρομπότ θα κινείται προς αυτές με τη βοήθεια του πράκτορα αποφυγής εμποδίων. Μετά τη λήξη των δοκιμών, θα είναι γνωστό το συντομότερο μονοπάτι στο συγκεκριμένο περιβάλλον κι έτσι το ρομπότ θα είναι σε θέση να μεταβεί με τον βέλτιστο τρόπο από το σημείο έναρξης ως το σημείο τερματισμού, πάντα χωρίς να συγκρούεται με τα εμπόδια στο περιβάλλον.

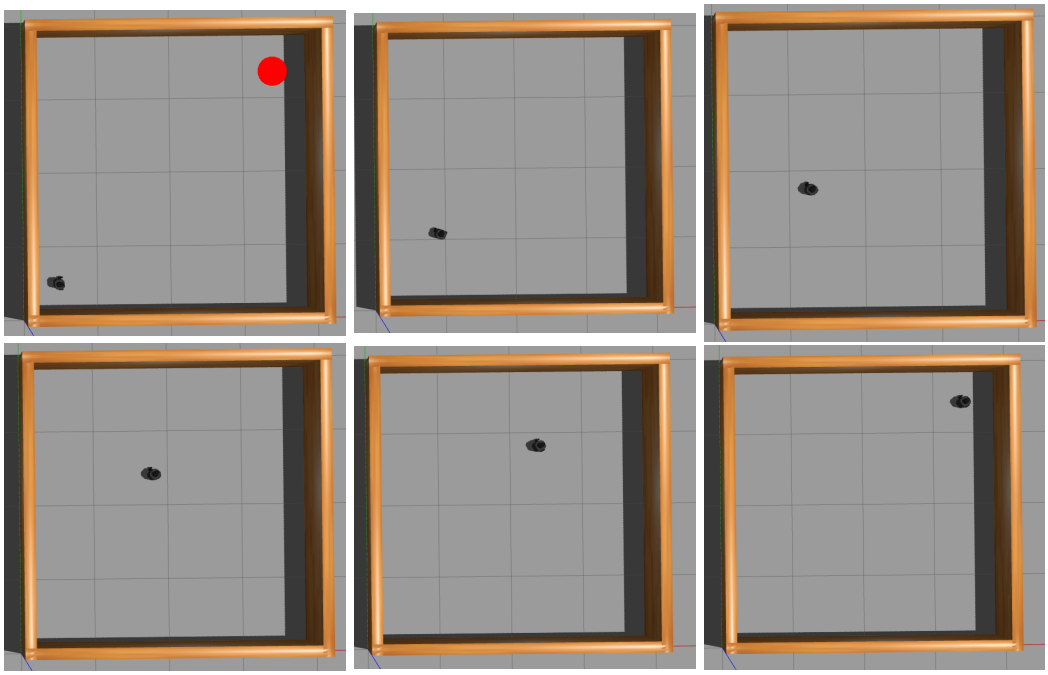
5.2 Δοκιμές εύρεσης στόχου με χρήση του εκπαιδευμένου συστήματος

5.2.1 Πρώτο περιβάλλον εκπαίδευσης

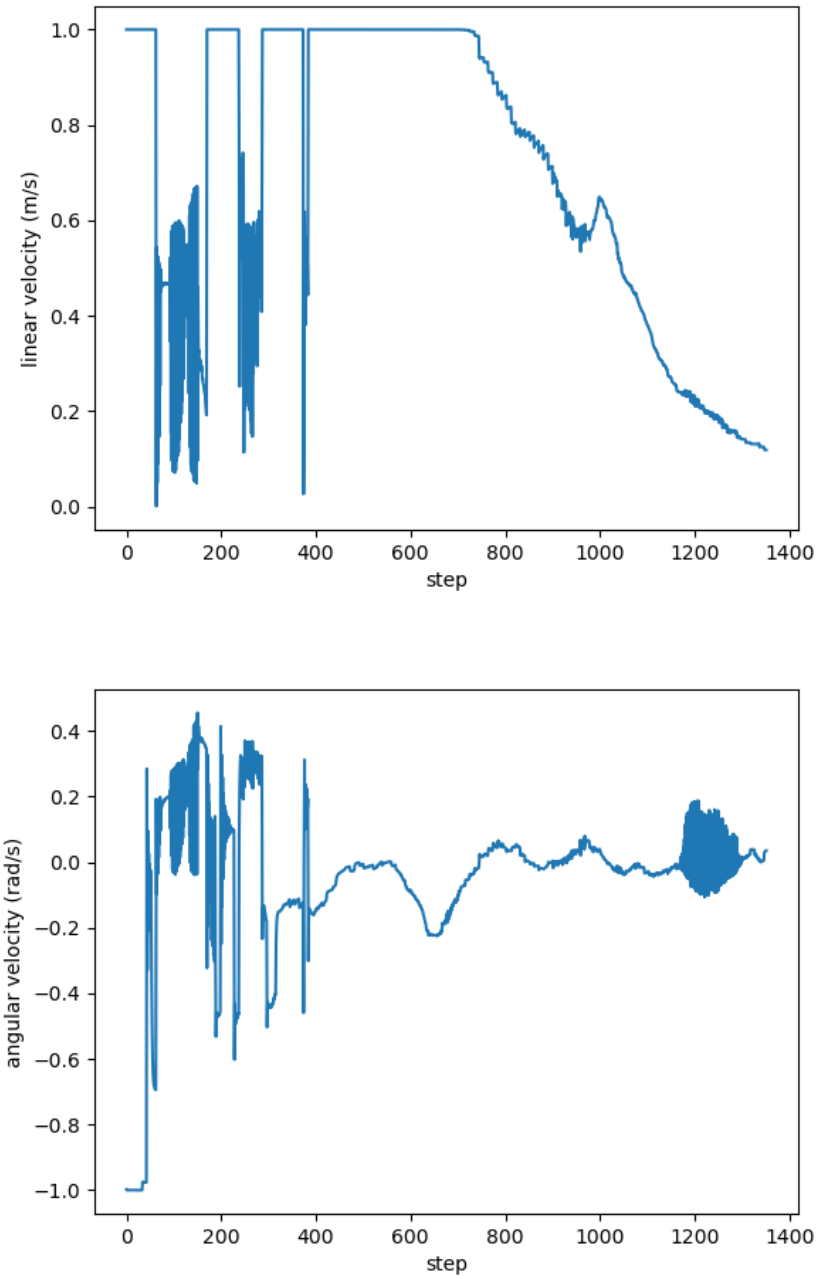
Το πρώτο δοκιμαστικό λειτουργίας του συστήματος σχεδιασμού τροχιάς και αποφυγής εμποδίων DDPG πραγματοποιείται σε άδειο περιβάλλον.Δίνεται στον πράκτορα η εντολή να μετακινήσει το ρομπότ από τη θέση $(0.5, 0.5)$ στη θέση $(3.5, 3.5)$. Παρατηρείται ότι ο πράκτορας εντοπίζει αποτελεσματικά την έλλειψη εμποδίων και κινεί το ρομπότ σε σχεδόν τέλεια ευθεία προς το στόχο του, μέχρι τη στιγμή που τον φτάνει.



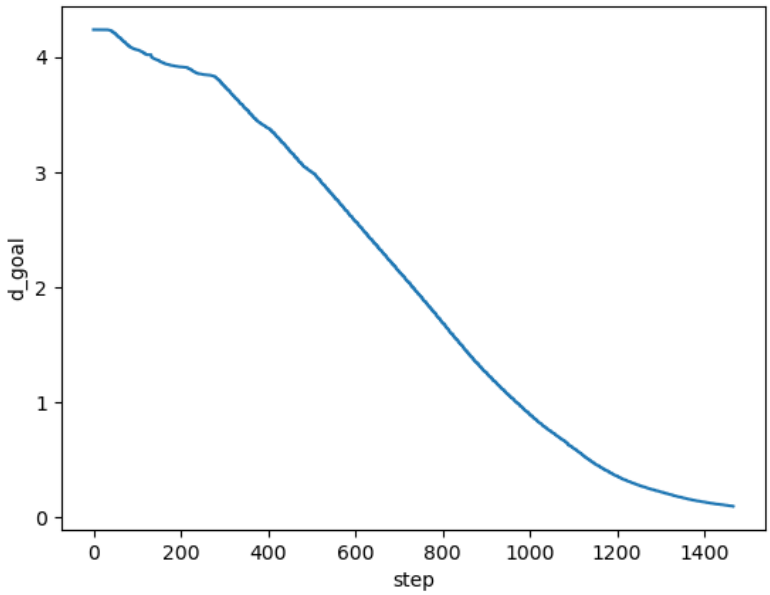
Σχήμα 5.1: Η τροχιά που ακολούθησε το ρομπότ, στο πρώτο δοκιμαστικό περιβάλλον.



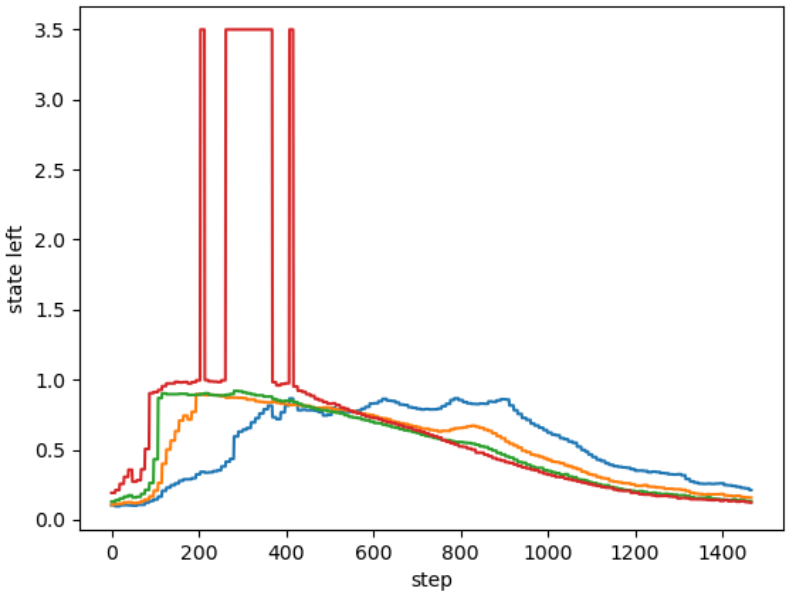
Σχήμα 5.2: Η αλληλουχία κινήσεων του δοκιμαστικού σχεδιασμού τροχιάς, από τη θέση $(0.5, 0.5)$ προς την $(3.5, 3.5)$ για το πρώτο δοκιμαστικό περιβάλλον. Στην πρώτη εικόνα φαίνεται η θέση εκκίνησης (το ρομπότ) και τερματισμού (κόκκινος κύκλος).



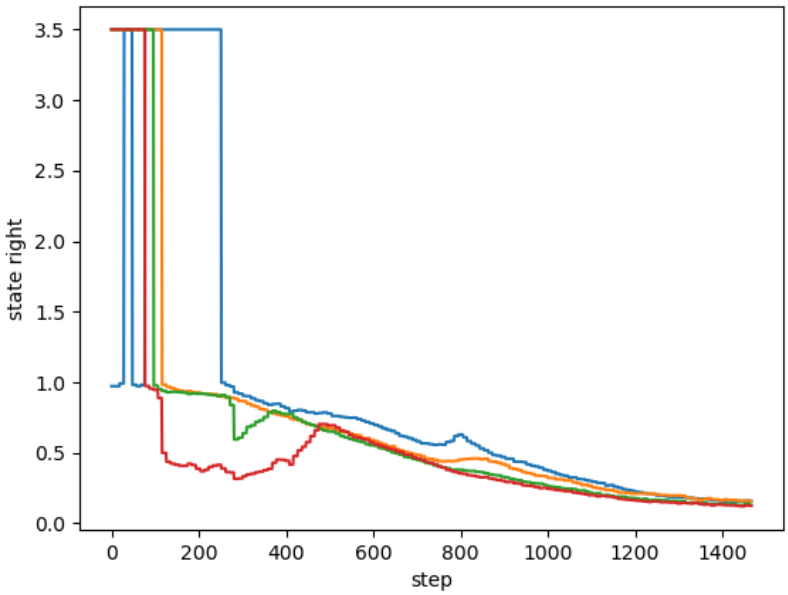
Σχήμα 5.3: Γραφήματα μεταβολής των ταχυτήτων κατά τη διάρκεια εκτέλεσης των δοκιμών στο πρώτο δοκιμαστικό περιβάλλον.



Σχήμα 5.4: Γράφημα μεταβολής της απευθείας απόστασης από τη θέση στόχο, σε m, για το πρώτο δοκιμαστικό περιβάλλον.



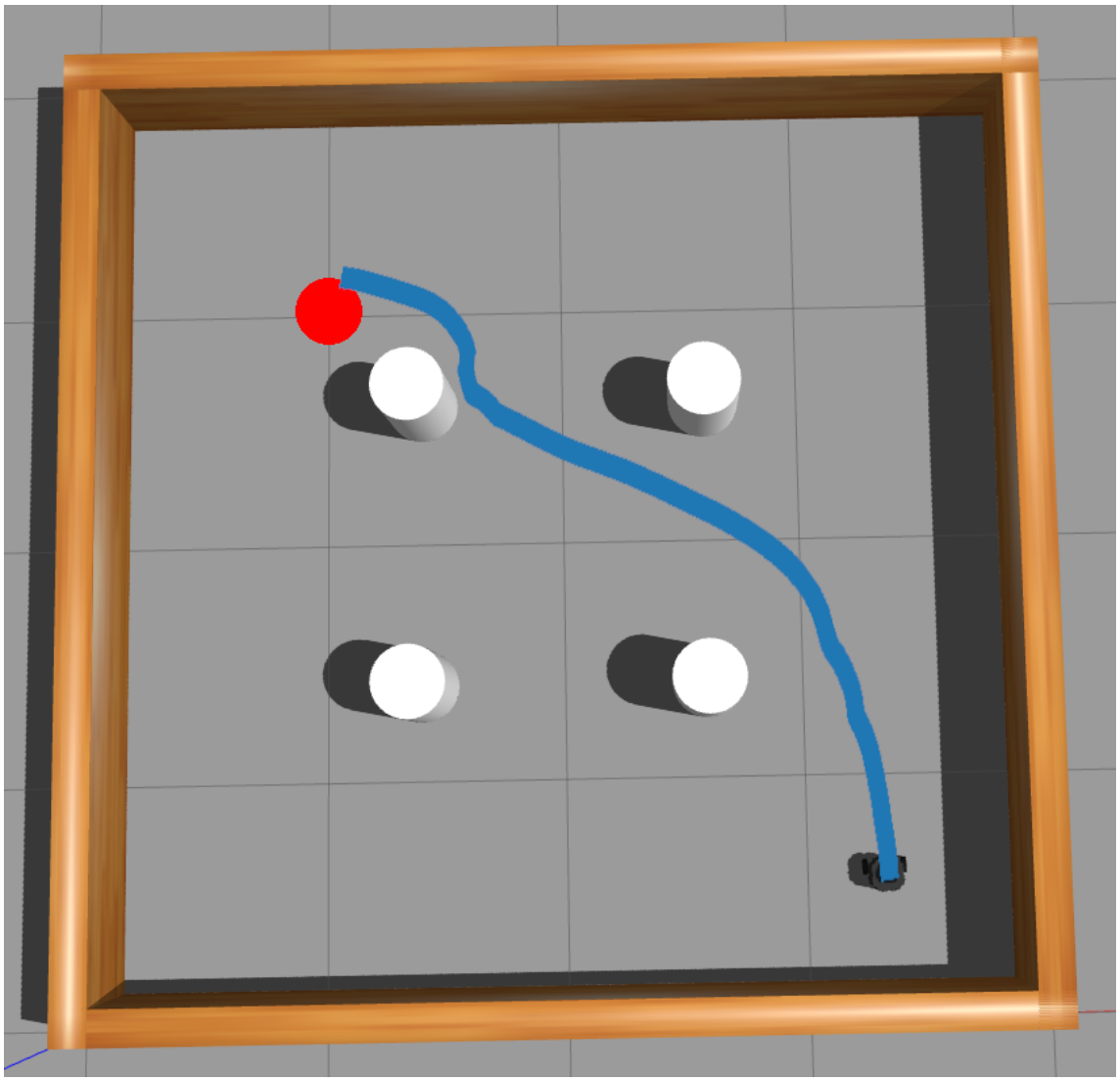
Σχήμα 5.5: Γράφημα μεταβολής των δεδομένων του αισθητήρα λέιζερ, για τις 5 αριστερές ακτίνες, για το πρώτο δοκιμαστικό περιβάλλον. Η απόσταση μετράται σε m.



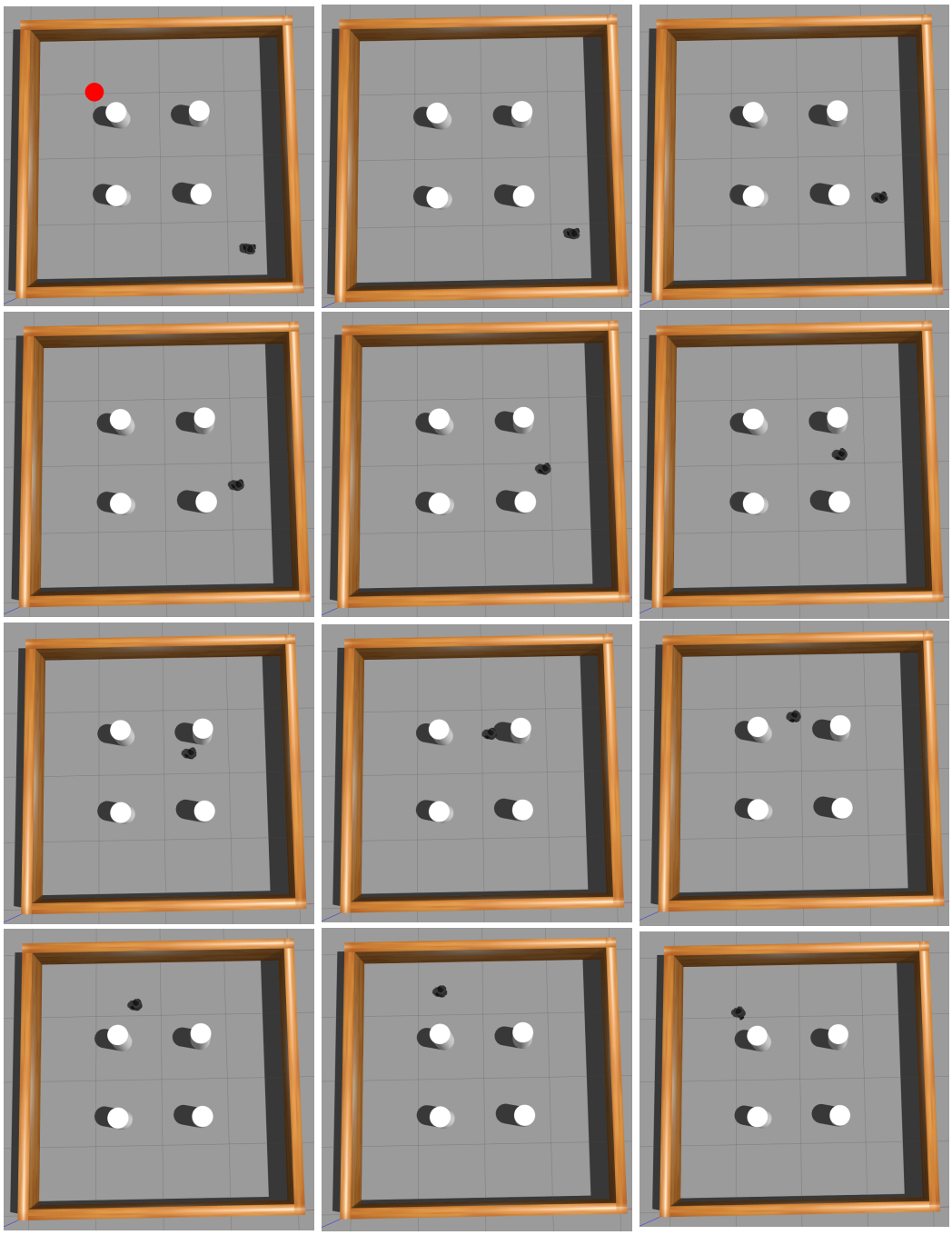
Σχήμα 5.6: Γράφημα μεταβολής των δεδομένων του αισθητήρα λέιζερ, για τις 5 δεξιές ακτίνες, για το πρώτο δοκιμαστικό περιβάλλον. Η απόσταση μετράται σε m.

5.2.2 Δεύτερο περιβάλλον εκπαίδευσης

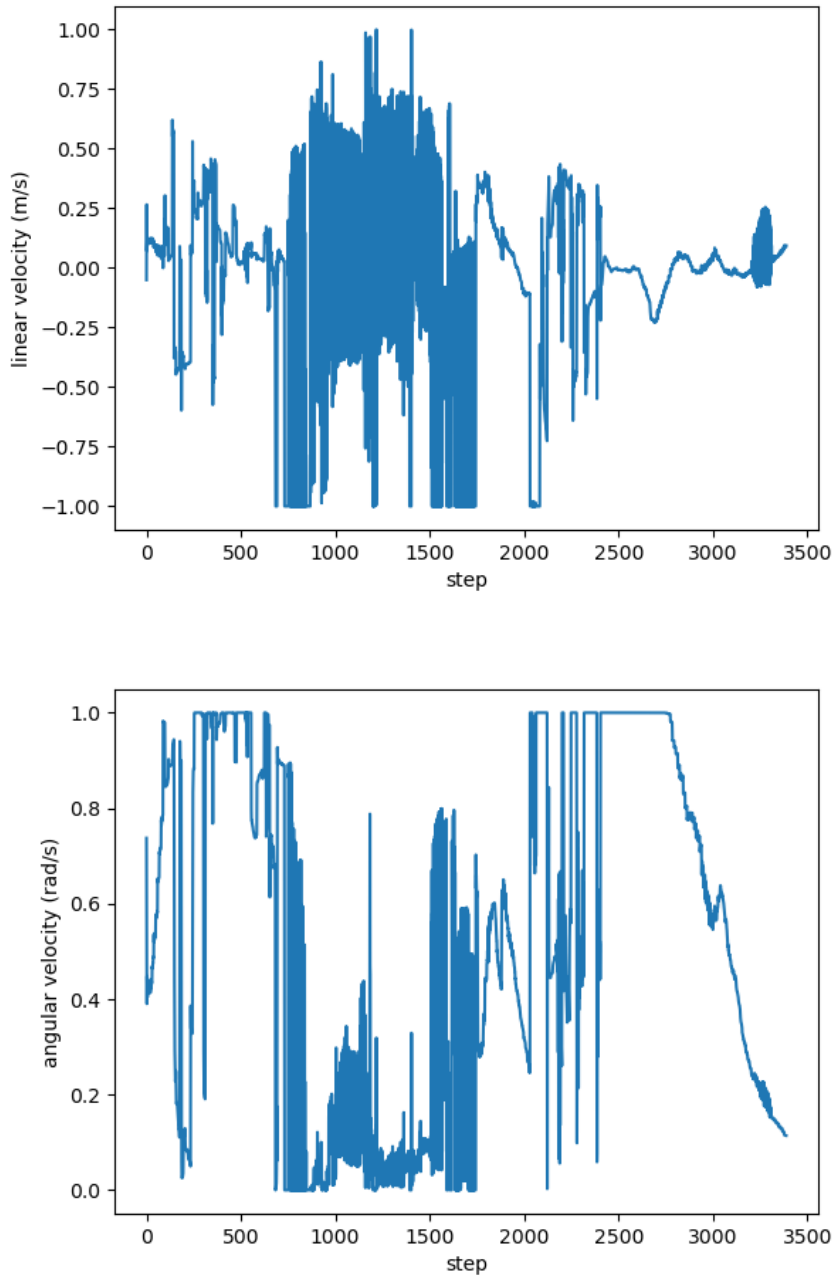
Έχοντας βεβαιωθεί ότι το σύστημα είναι ικανό να μετακινήσει το ρομπότ σε άδειο περιβάλλον, εκτελείται το επόμενο δοκιμαστικό σε ένα περιβάλλον με 4 κυλινδρικά εμπόδια στο κέντρου του, ίδιο με το περιβάλλον στο Σχήμα 5.7. Το ρομπότ εκκινεί τη μετακίνησή του από το σημείο $(3.5, 0.5)$ προς το $(1, 3)$, όπως φαίνεται στην πρώτη εικόνα του Σχήματος 5.8. Με κόκκινη κουκίδα συμβολίζεται το σημείο στόχος, ενώ το ρομπότ φαίνεται με το κατάλληλο γραφικό του. Όπως παρατηρείται σε αυτό το σχήμα, τα διαδοχικά στιγμιότυπα με τη θέση του ρομπότ στο χώρο υποδεικνύουν ότι το ρομπότ διαθέτει την ικανότητα να μετακινείται επιτυχώς και σε περιβάλλον με μερικά εμπόδια.



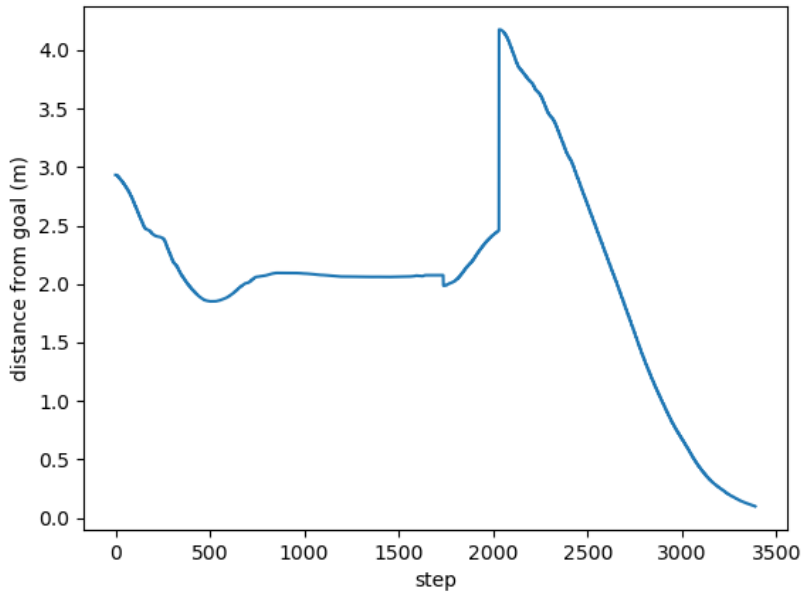
Σχήμα 5.7: Η τροχιά που ακολούθησε το ρομπότ, στο δεύτερο δοκιμαστικό περιβάλλον.



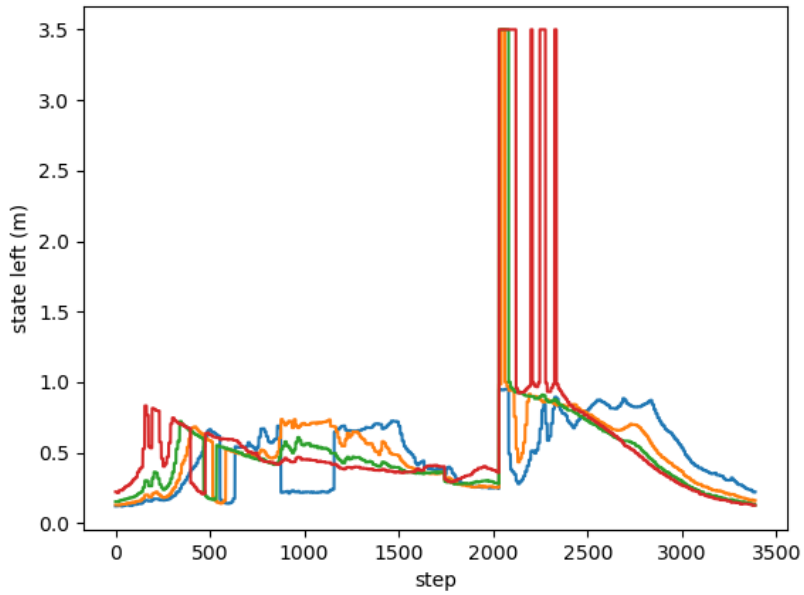
Σχήμα 5.8: Η αλληλουχία κινήσεων του δοκιμαστικού σχεδιασμού τροχιάς, από τη θέση $(3.5, 0.5)$ προς την $(1, 3)$ για το δεύτερο δοκιμαστικό περιβάλλον. Στην πρώτη εικόνα φαίνεται η θέση εκκίνησης (το ρομπότ) και τερματισμού (κόκκινος κύκλος).



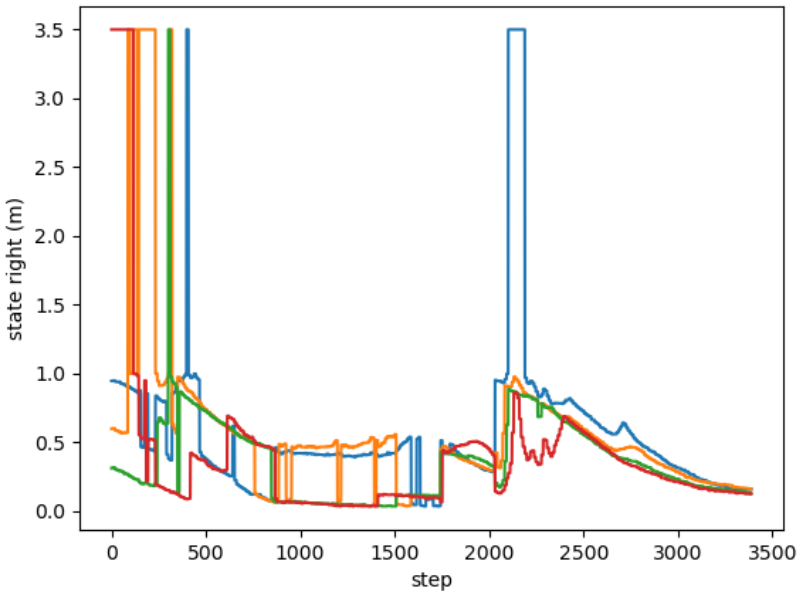
Σχήμα 5.9: Γράφημα μεταβολής των ταχυτήτων κατά τη διάρκεια εκτέλεσης των δοκιμών στο δεύτερο δοκιμαστικό περιβάλλον.



Σχήμα 5.10: Γράφημα μεταβολής της απευθείας απόστασης από τη θέση στόχο, σε m, για το δεύτερο δοκιμαστικό περιβάλλον.



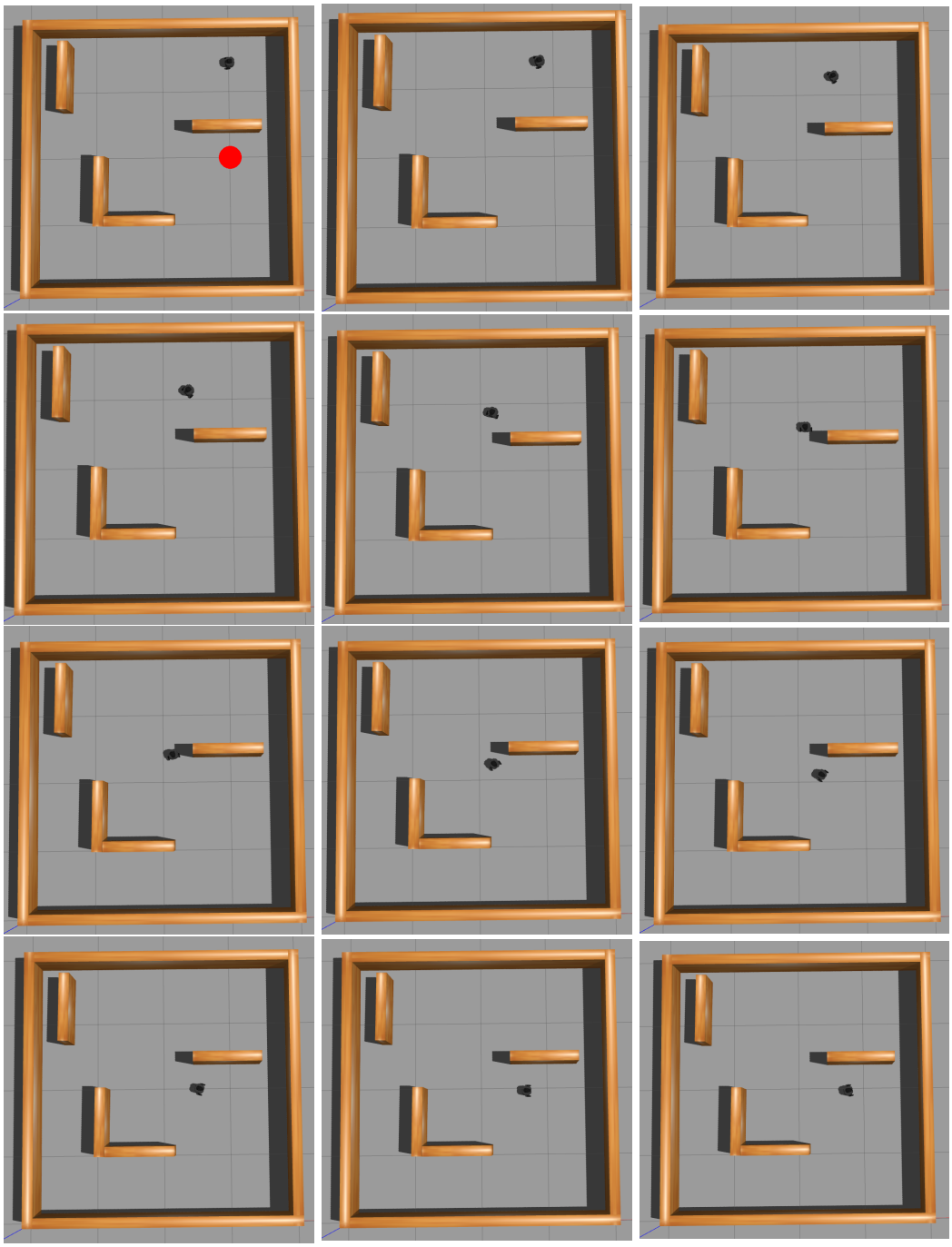
Σχήμα 5.11: Γράφημα μεταβολής των δεδομένων του αισθητήρα λέιζερ, για τις 5 αριστερές ακτίνες, για το δεύτερο δοκιμαστικό περιβάλλον. Η απόσταση μετράται σε m.



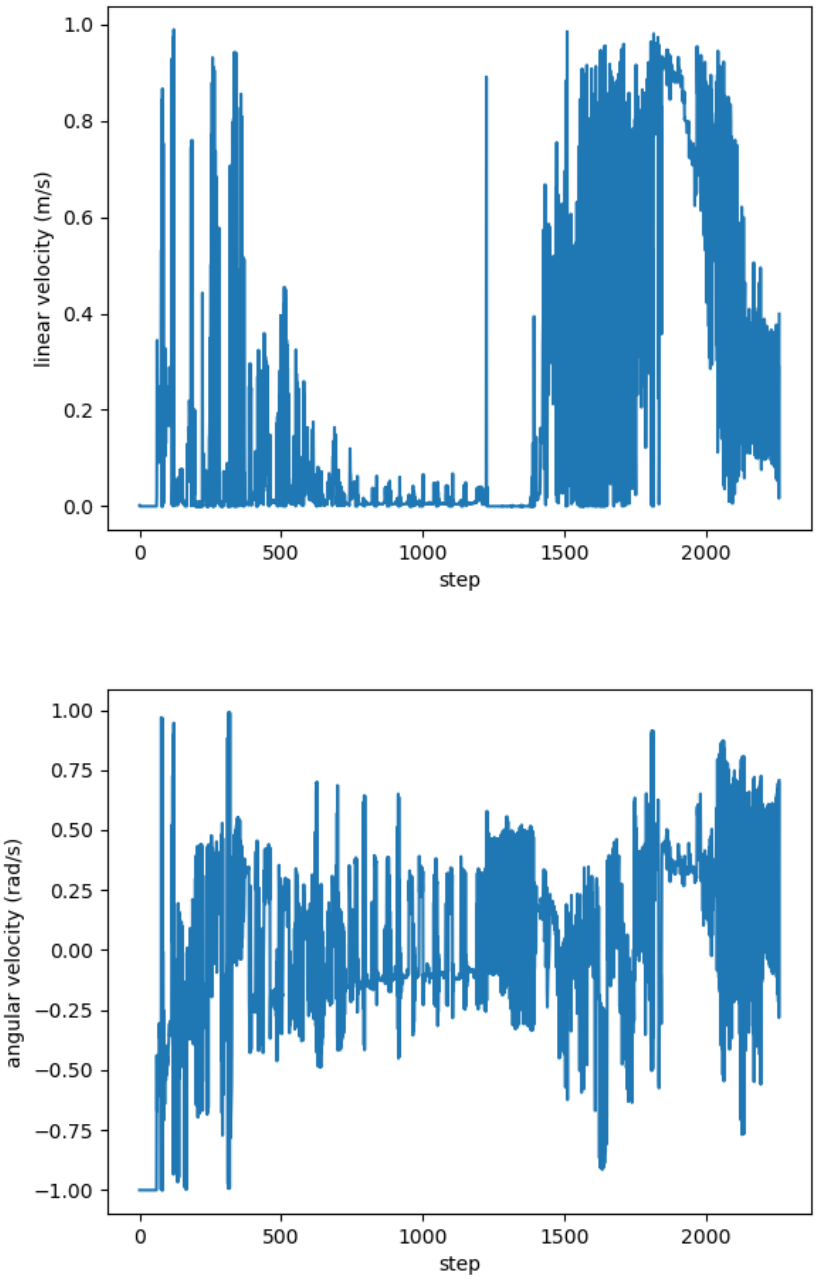
Σχήμα 5.12: Γράφημα μεταβολής των δεδομένων του αισθητήρα λέιζερ, για τις 5 δεξιές ακτίνες, για το δεύτερο δοκιμαστικό περιβάλλον. Η απόσταση μετράται σε m.

5.2.3 Τρίτο περιβάλλον εκπαίδευσης

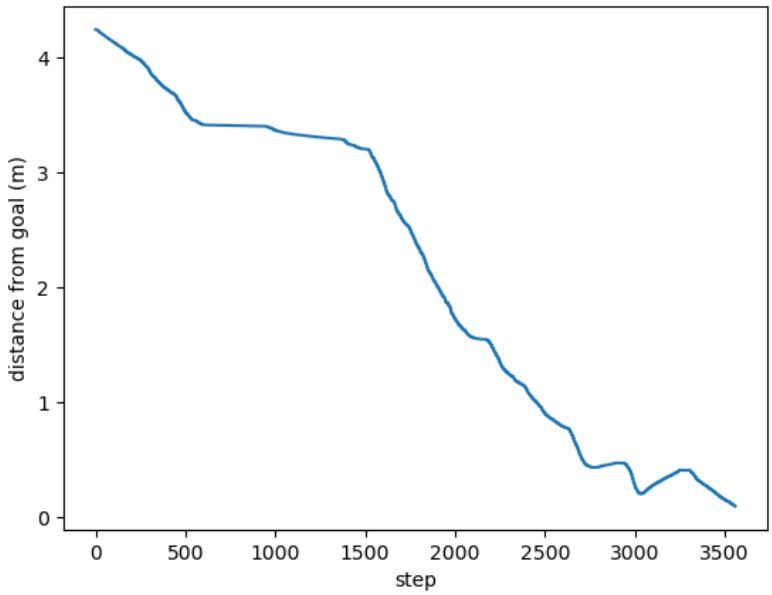
Εφόσον ελέγχθηκε το γεγονός ότι το σύστημα είναι ικανό να ανταπεξέλθει και σε εμπόδια, τοποθετείται σε ένα σύνθετο περιβάλλον, με διάφορα εμπόδια μικρού και μεγάλου μεγέθους. Το περιβάλλον είναι το ίδιο με το Σχήμα 4.8. Δίνεται εντολή στο σύστημα να πλοηγήσει το ρομπότ από τη θέση $(3, 3.5)$ προς την $(3, 2)$, γύρω από ένα μεγάλο εμπόδιο. Παρατηρείται πώς το ρομπότ αποφεύγει με επιτυχία το εμπόδιο και φθάνει στο στόχο του.



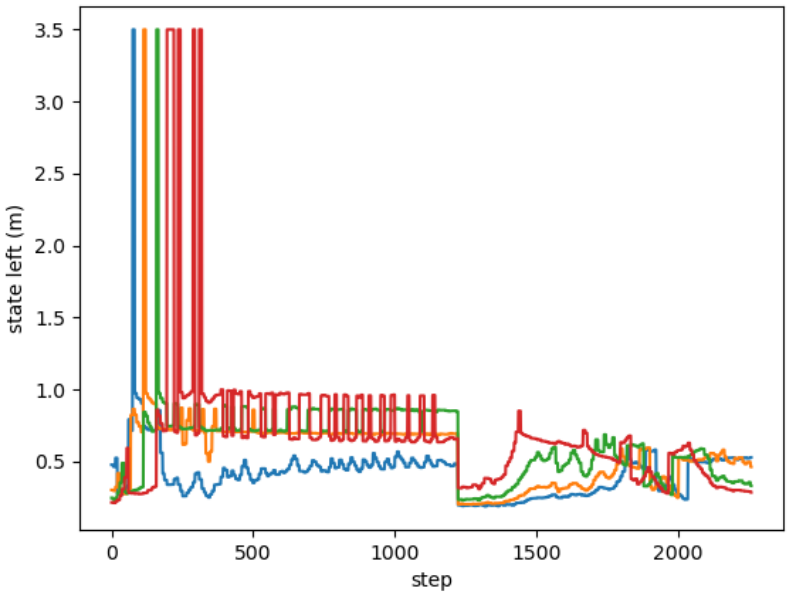
Σχήμα 5.13: Η αλληλουχία κινήσεων του δοκιμαστικού σχεδιασμού τροχιάς, από τη θέση $(3, 3.5)$ προς την $(3, 2)$ για το τρίτο δοκιμαστικό περιβάλλον. Στην πρώτη εικόνα φαίνεται η θέση εκκίνησης (το ρομπότ) και τερματισμού (κόκκινος κύκλος).



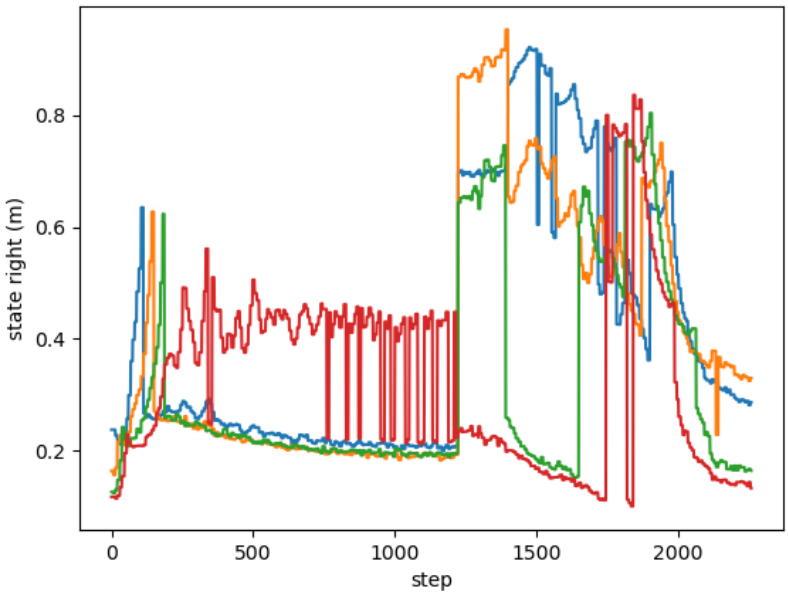
Σχήμα 5.14: Γράφημα μεταβολής των ταχυτήτων κατά τη διάρκεια εκτέλεσης των δοκιμών στο τρίτο δοκιμαστικό περιβάλλον.



Σχήμα 5.15: Γράφημα μεταβολής της απευθείας απόστασης από τη θέση στόχο, σε m, για το τρίτο δοκιμαστικό περιβάλλον.



Σχήμα 5.16: Γράφημα μεταβολής των δεδομένων του αισθητήρα λέιζερ, για τις 5 αριστερές ακτίνες, για το τρίτο δοκιμαστικό περιβάλλον. Η απόσταση μετράται σε m.



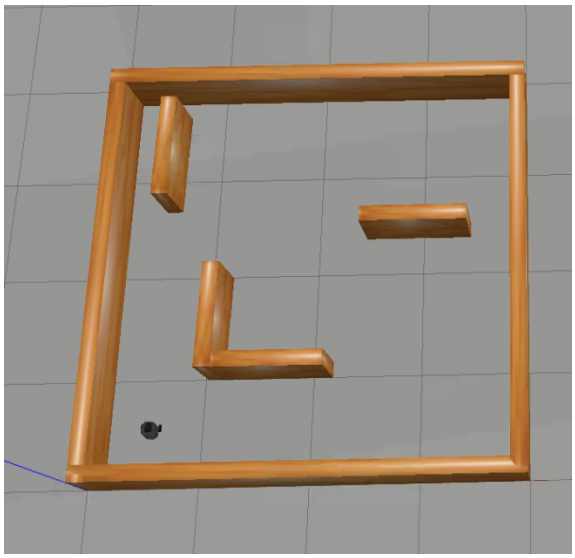
Σχήμα 5.17: Γράφημα μεταβολής των δεδομένων του αισθητήρα λέιζερ, για τις 5 δεξιές ακτίνες, για το τρίτο δοκιμαστικό περιβάλλον. Η απόσταση μετράται σε m.

Παρατηρώντας τα παραπάνω πειραματικά αποτελέσματα, διαπιστώνεται ότι ο πράκτορας του συστήματος σχεδιασμού τροχιάς και αποφυγής εμποδίων είναι πράγματι σε θέση να υπαγορεύσει εντολές κίνησης στο ρομπότ ώστε να το καθοδηγήσει προς το στόχο του, αποφεύγοντας παράλληλα τυχόν εμπόδια που μπορεί να συναντήσει στη διαδρομή του προς αυτόν. Αυτό δεν γίνεται με βέλτιστο τρόπο όσον αφορά διανυθείσα απόσταση ή χρόνο, αλλά διασφαλίζει ότι θα φτάσει στο στόχο του, χωρίς να πλησιάσει επικίνδυνα κάποιο εμπόδιο ή συγκρουστεί με αυτό.

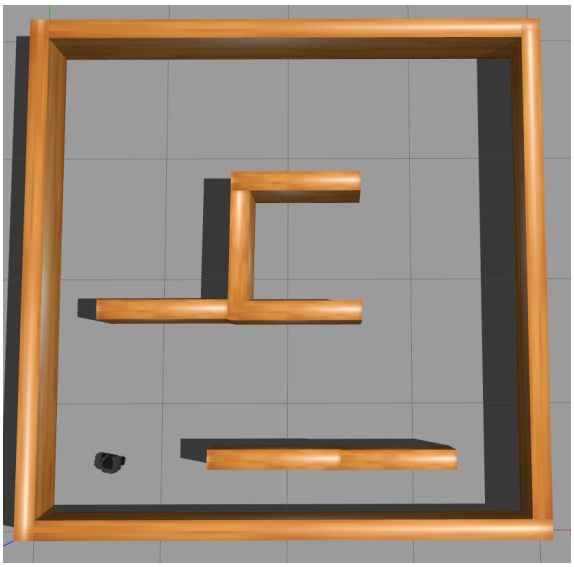
Σε συγκεκριμένες πειραματικές δοκιμές το ρομποτικό όχημα καθυστερούσε να προσεγγίσει το στόχο του. Αυτό συνέβησε κυρίως όσες φορές πλησίασε πολύ κάποιο εμπόδιο. Λόγω της συνάρτησης ανταμοιβής που επιλέχθηκε στα πλαίσια της εκπαίδευσης των νευρωνικών δικτύων, ο πράκτορας δεχόταν αρνητική ανταμοιβή όταν πλησίαζε επικίνδυνα κάποιο εμπόδιο. Έτσι, αν τύχει για κάποιο λόγο στην πράξη το ρομπότ να πλησιάσει πολύ κάποιο εμπόδιο, προκειμένου να αποφευχθεί η σύγκρουση, ο πράκτορας ανέπτυξε μόνος του μια συμπεριφορά στην οποία κινείται πάρα πολυ αργά, με πολύ μικρές μεταβολές στη γραμμική και γωνιακή του ταχύτητα, μέχρις ότου να απομακρυνθεί αρκετά από το εμπόδιο, όπου και επέστρεφε στην κανονική του συμπεριφορά και κινούταν πάλι με κανονικές ταχύτητες. Αυτό μπορεί να παρατηρηθεί στα γραφήματα των ταχυτήτων για το τρίτο περιβάλλον (Γράφημα 5.14, όπου για ένα μεγάλο χρονικό διάστημα οι ταχύτητες κινούνται πολύ κοντά στο μηδέν.

5.3 Πειραματικές δοκιμές εύρεσης βέλτιστης τροχιάς με ταυτόχρονη αποφυγή εμποδίων

Πραγματοποιήθηκαν πειραματικές δοκιμές των ευρετικών αλγορίθμων σε 3 διαφορετικά περιβάλλοντα, ένα γνωστό και δύο άγνωστα. Το γνωστό είναι το τρίτο περιβάλλον από τη διαδικασία εκπαίδευσης, δηλαδή αυτό στο Σχήμα 5.18α΄. Το πρώτο άγνωστο περιβάλλον αποτελεί ένα περιβάλλον παρόμοιο με το γνωστό, όμως με διάφορες διαφορές και λίγα παραπάνω εμπόδια. Το γνωστό περιβάλλον και το πρώτο άγνωστο παρουσιάζονται στο Σχήμα 5.18.



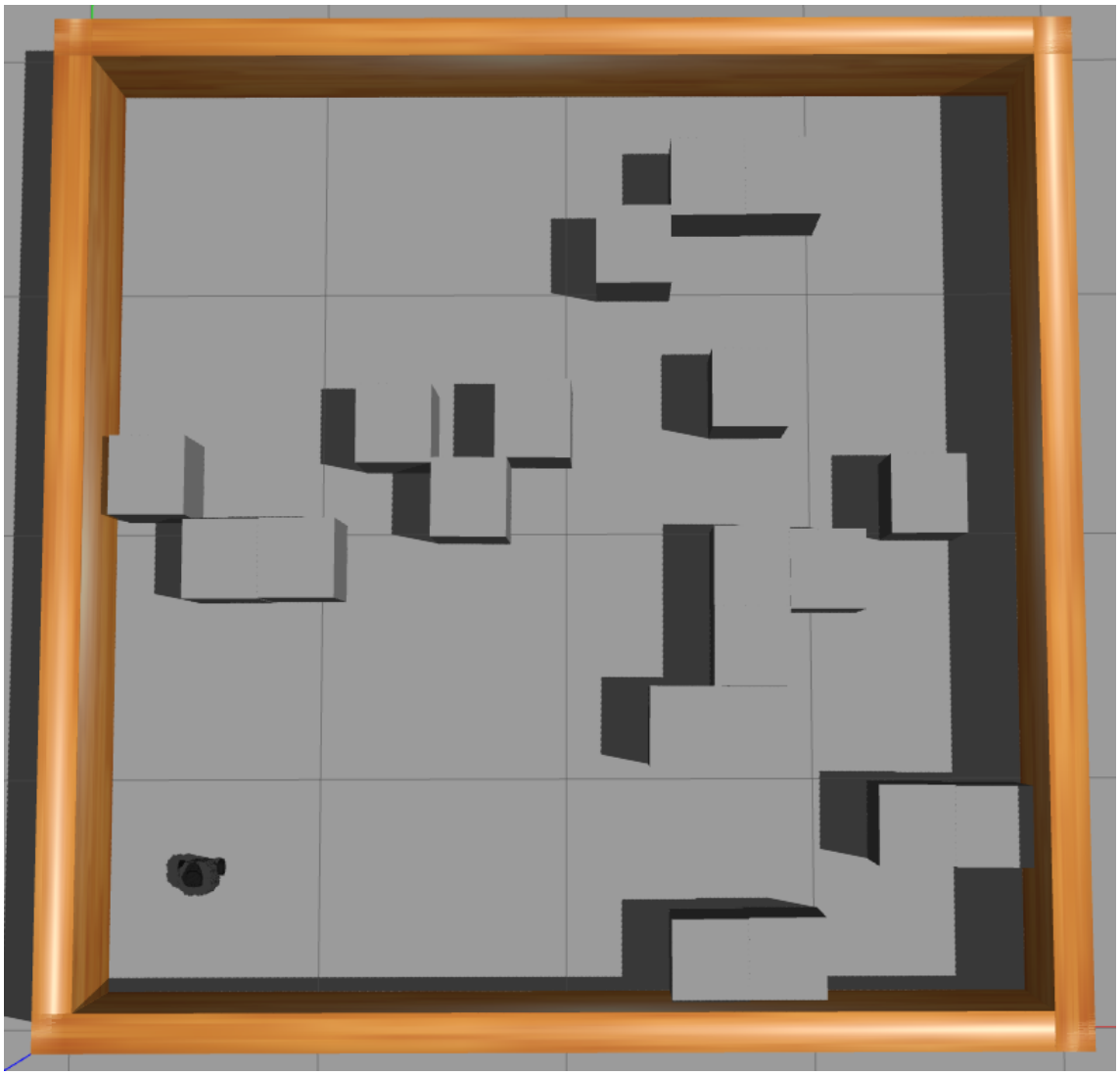
(α΄) Το γνωστό περιβάλλον δοκιμών, στον προσομοιωτή Gazebo. Είναι ταυτόσημο με αυτό στο Σχήμα 4.8.



(β΄) Το άγνωστο περιβάλλον δοκιμών, στον προσομοιωτή Gazebo.

Σχήμα 5.18: Τα περιβάλλοντα δοκιμών, στα οποία δοκιμάζονται οι ευρετικοί αλγόριθμοι.

Το δεύτερο άγνωστο περιβάλλον είναι πολύ πιο απαιτητικό από τα προηγούμενα δύο, καθώς διαθέτει αρκετά μεγαλύτερο ποσοστό εμποδίων, μικρά σε μέγεθος. Αυτό θέτει έναν βαθμό δυσκολίας στο σύστημα αποφυγής εμποδίων, επειδή στα προηγούμενα περιβάλλοντα είχε εκπαιδευτεί σε μεγάλα και ογκώδη εμπόδια, τα οποία όμως είναι λίγα σε πλήθος στο περιβάλλον, ενώ τώρα ισχύει το αντίθετο. Το περιβάλλον αυτό φαίνεται στο Σχήμα 5.19.



Σχήμα 5.19: Το δεύτερο άγνωστο περιβάλλον δοκιμών, στον προσομοιωτή Gazebo. Παρατηρούνται τα πολλαπλά μικρά εμπόδια κατανεημημένα στο χώρο του περιβάλλοντος.

5.3.1 Πειραματικά αποτελέσματα της εφαρμογής του αλγορίθμου A*

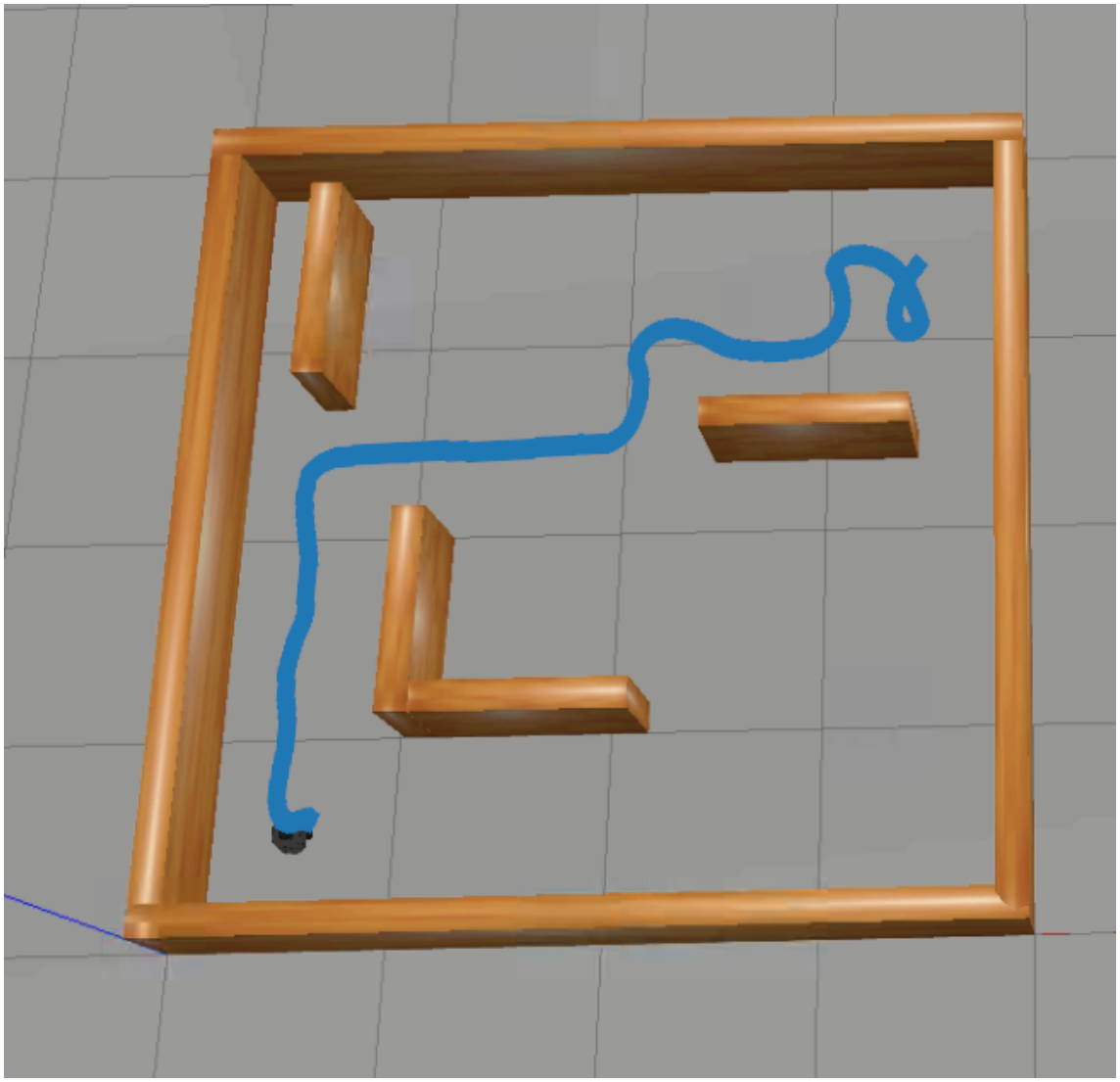
Γνωστό περιβάλλον

Αρχικά το σύστημα δοκιμάστηκε σε γνωστό περιβάλλον, στο οποίο διαθέτει πολλές προηγούμενες εμπειρίες, όσον αφορά τον σχεδιασμό τροχιάς και την αποφυγή εμποδίων σε αυτό. Το ρομπότ εκκινεί από τη θέση $(0.5, 0.5)$ με στόχο τη θέση $(3.5, 3.5)$. Εκτελώντας τον αλγόριθμο, αυτός παρέχει στο σύστημα σχεδιασμού τροχιάς και αποφυγής εμποδίων την επόμενη θέση στόχο, που προκύπτει από τον ευρετικό αλγόριθμο συντομότερης διαδρομής A*. Ο κόμβος επιλέγεται ανάλογα με την απόστασή του από την τελική θέση στόχο, ερώτημα το οποίο απαντά το ευρετικό σκέλος του αλγορίθμου, αλλά και ανάλογα με ύπαρξη εμποδίων στον εν λόγω κόμβο. Σε αυτή την περίπωση, ο κόμβος με το εμπόδιο τοποθετείται στη λίστα με τα εμπόδια και δεν πρόκειται να ζητηθεί από το ρομπότ να κινηθεί σε αυτόν, αποφεύγοντας έτσι συγκρούσεις με το περιβάλλον.

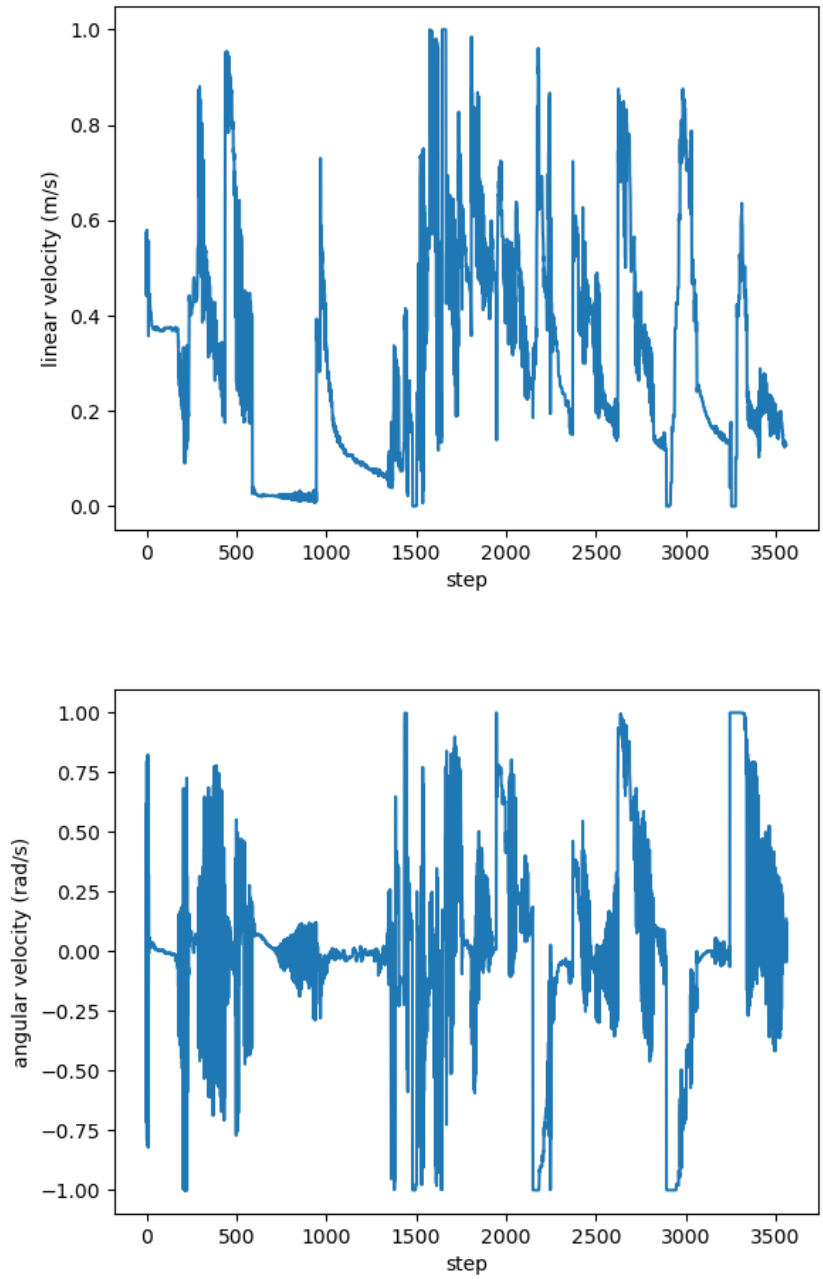
Στο Σχήμα 5.25 φαίνεται η τροχιά που ακολούθησε το ρομποτικό όχημα εντός του δοκιμαστικού περιβάλλοντος. Παρατηρείται ομαλότητα σε αυτή, ενώ αποφεύγει με επιτυχία τα εμπόδια,

διατηρώντας μια ικανοποιητική απόσταση ασφαλείας από αυτά. Στο Σχήμα 5.21 απεικονίζεται η γραμμική και γωνιακή ταχύτητα που δίνεται στο ρομπότ από τον πράκτορα DDPG, στο Σχήμα 5.22 φαίνεται η απόσταση από την θέση στόχο του και στα Σχήματα 5.23 και 5.24 φαίνονται τα δεδομένα από το μετρητή αποστάσεων λέιζερ, στην αριστερή και δεξιά μεριά του ρομπότ αντίστοιχα, όπως παρουσιάζονται στο Σχήμα 4.3.

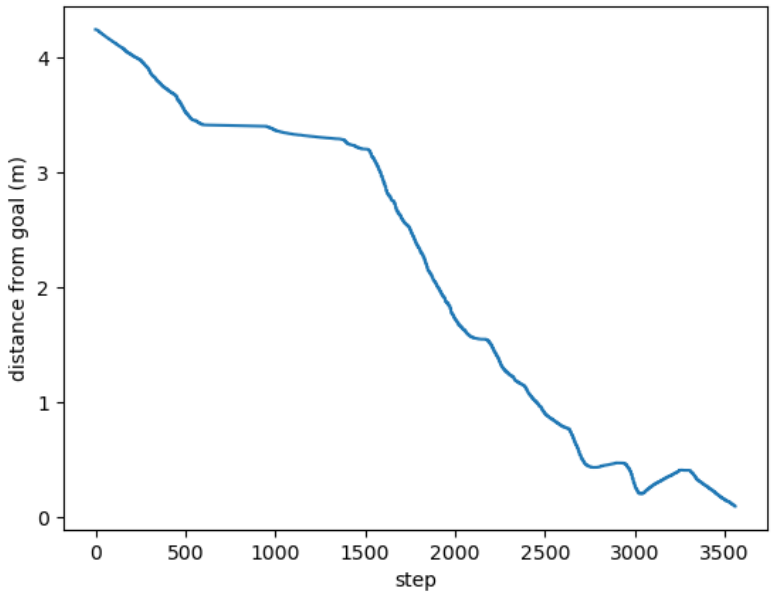
Ο χρόνος που χρειάστηκε το ρομπότ να μετακινηθεί από τη θέση εκκίνησης στη θέση τερματισμού ήταν 38.44 δευτερόλεπτα, ενώ η διανυθείσα απόσταση ανέρχεται στα $6.7m$.



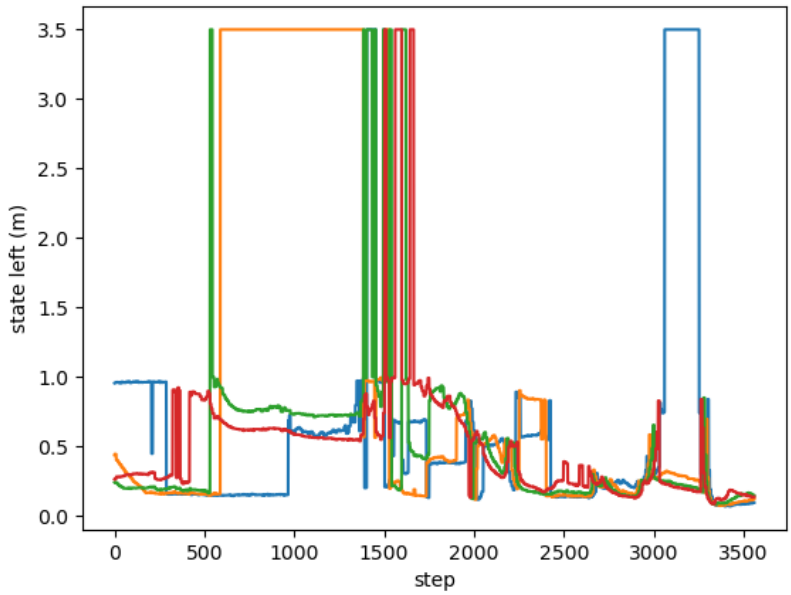
Σχήμα 5.20: Η τροχιά που ακολούθησε το ρομποτικό όχημα στο γνωστό περιβάλλον, σύμφωνα με τους κόμβους που του υπαγόρευε ο αλγόριθμος A*.



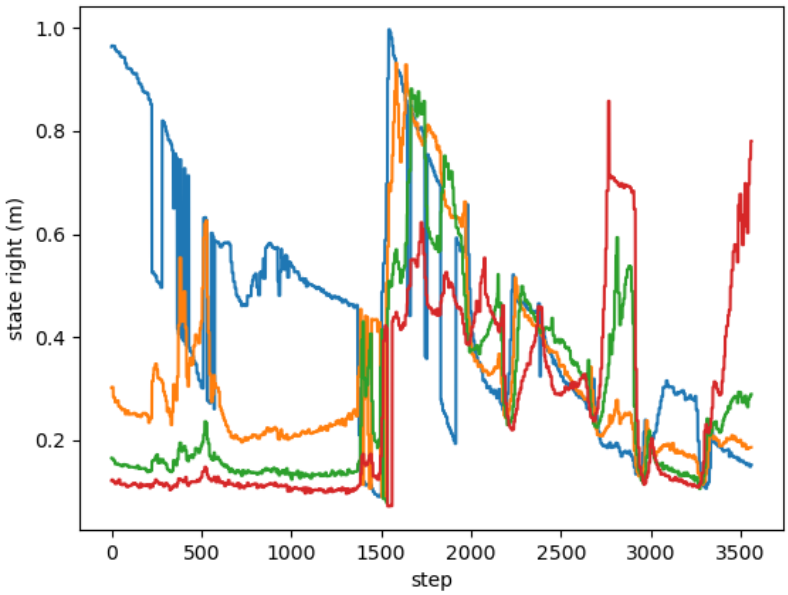
Σχήμα 5.21: Γράφημα μεταβολής των ταχυτήτων κατά τη διάρκεια εκτέλεσης του αλγορίθμου A* στο γνωστό περιβάλλον.



Σχήμα 5.22: Γράφημα μεταβολής της απευθείας απόστασης από τη θέση στόχο, σε m, κατά τη διάρκεια εκτέλεσης του αλγορίθμου A* στο γνωστό περιβάλλον.



Σχήμα 5.23: Γράφημα μεταβολής των δεδομένων του αισθητήρα λέιζερ, για τις 5 αριστερές ακτίνες, κατά τη διάρκεια εκτέλεσης του αλγορίθμου A* στο γνωστό περιβάλλον. Η απόσταση μετράται σε m.



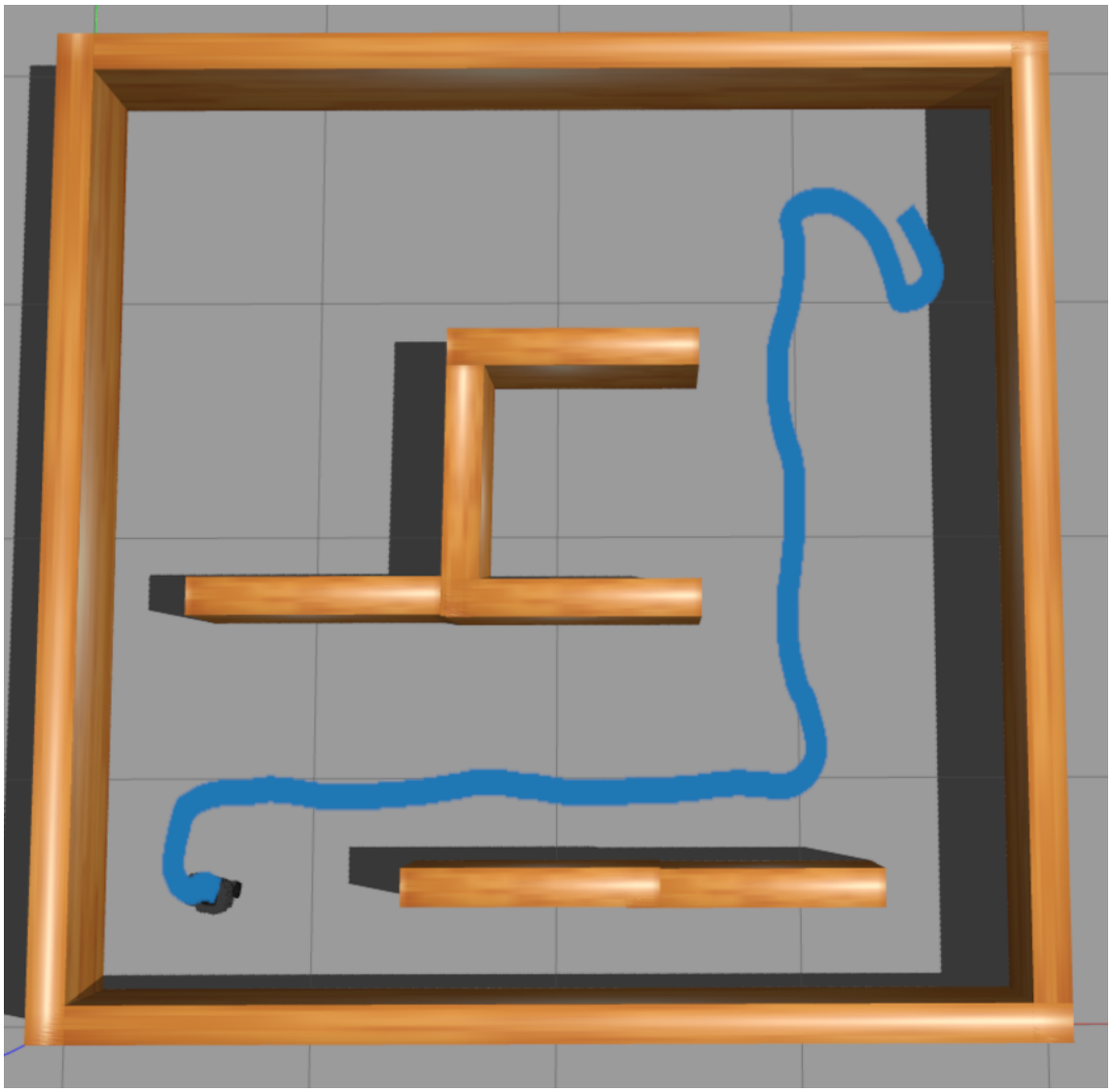
Σχήμα 5.24: Γράφημα μεταβολής των δεδομένων του αισθητήρα λέιζερ, για τις 5 δεξιές ακτίνες, κατά τη διάρκεια εκτέλεσης του αλγορίθμου A* στο γνωστό περιβάλλον. Η απόσταση μετράται σε m.

Πρώτο άγνωστο περιβάλλον

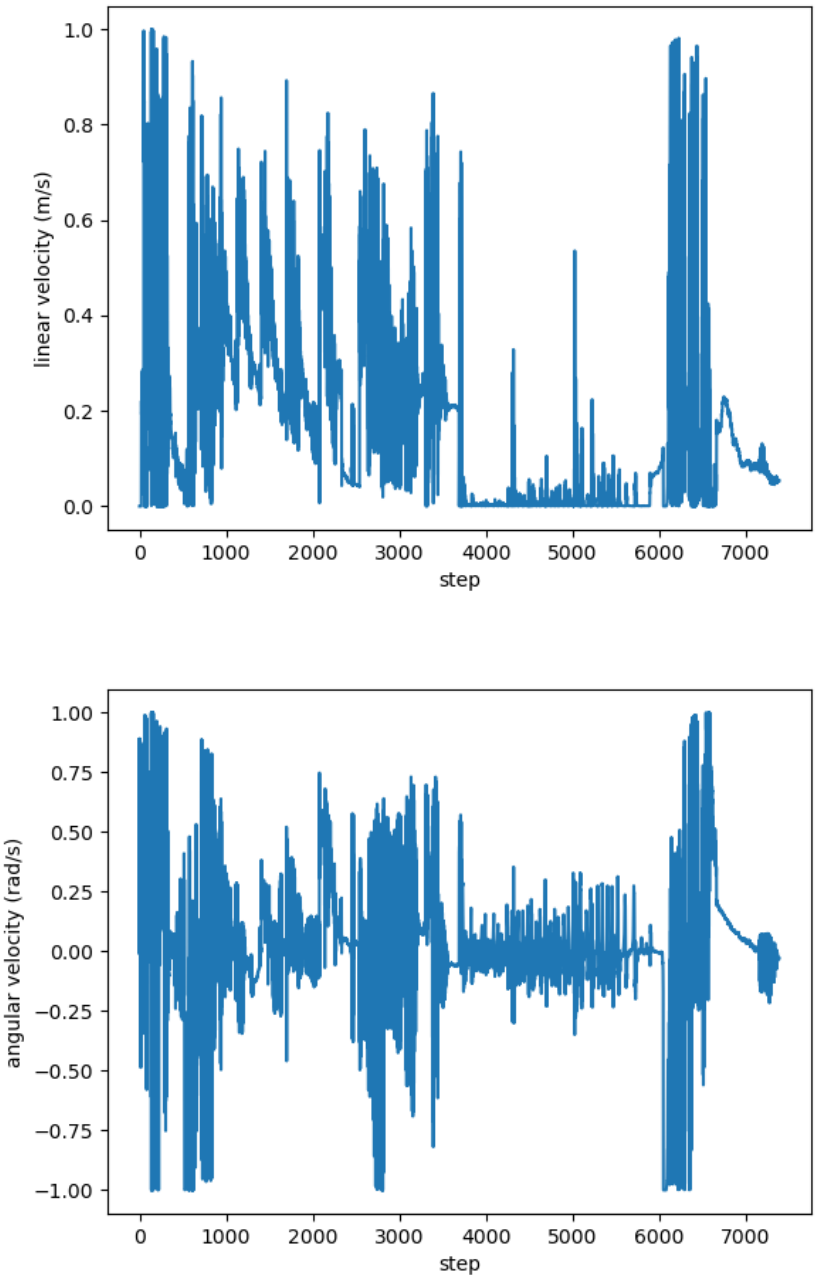
Όμοια με την παραπάνω δοκιμή, επαναλαμβάνεται η ίδια διαδικασία σε νέο περιβάλλον, το οποίο όμως αυτή τη φορά είναι άγνωστο. Ο πράκτορας, δηλαδή, δε διαθέτει καμία εμπειρία πάνω σε αυτό. Το ρομπότ εκκινεί και σε αυτή την περίπτωση από τη θέση $(0.5, 0.5)$ με θέση στόχο την $(3.5, 3.5)$. Παρατηρείται ότι, παρόλο που ο πράκτορας δε γνωρίζει το περιβάλλον αυτό, καταφέρνει με μεγάλη επιτυχία να μετακινήσει το ρομπότ σε αυτό το περιβάλλον χωρίς προβλήματα ή λανθασμένες κινήσεις και μάλιστα με πολύ καλής ποιότητας τροχιά.

Στο Σχήμα 5.25 φαίνεται η τροχιά που ακολούθησε το ρομποτικό όχημα εντός του δοκιμαστικού περιβάλλοντος. Στο Σχήμα 5.21 απεικονίζεται η γραμμική και γωνιακή ταχύτητα που δίνεται στο ρομπότ από τον πράκτορα DDPG, στο Σχήμα 5.22 φαίνεται η απόσταση από την θέση στόχο του και στα Σχήματα 5.23 και 5.24 φαίνονται τα δεδομένα από το μετρητή αποστάσεων λέιζερ, στην αριστερή και δεξιά μεριά του ρομπότ.

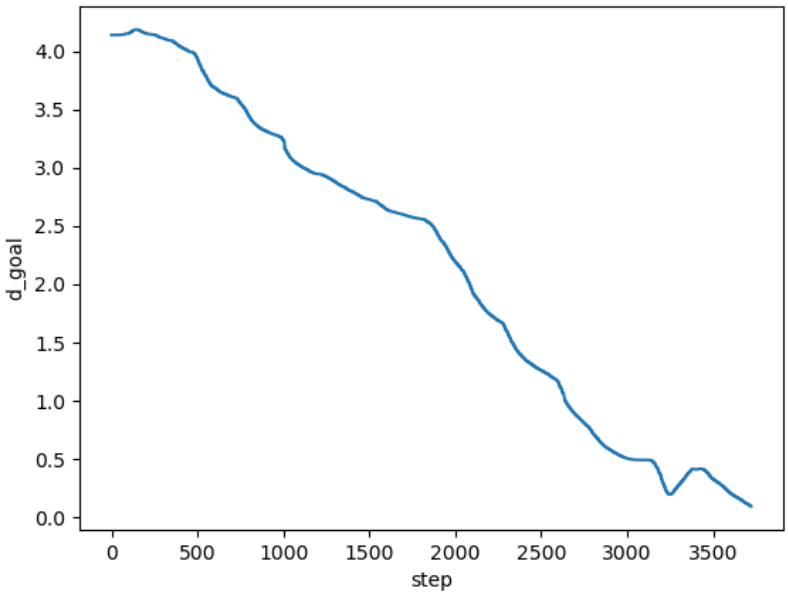
Ο χρόνος που χρειάστηκε στο ρομπότ να κινηθεί από τη θέση εκκίνησης στη θέση τερματισμού ήταν 38.23 δευτερόλεπτα, ενώ η διανυθείσα απόσταση ανέρχεται στα $6.8m$.



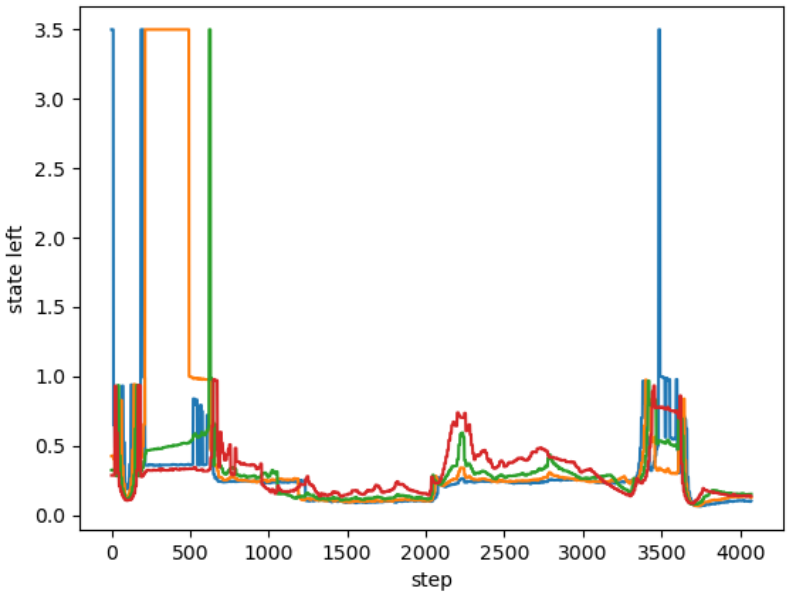
Σχήμα 5.25: Η τροχιά που ακολούθησε το ρομποτικό όχημα στο άγνωστο περιβάλλον, σύμφωνα με τους κόμβους που του υπαγόρευε ο αλγόριθμος A*



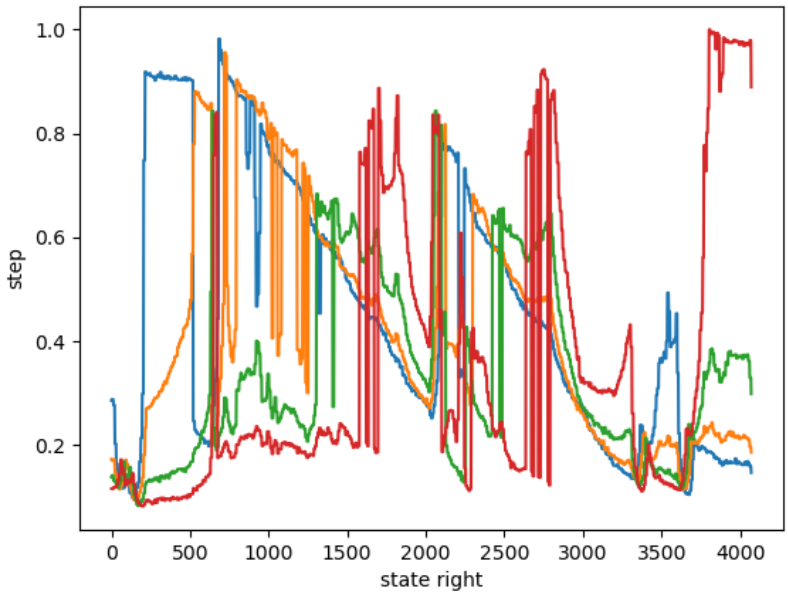
Σχήμα 5.26: Γράφημα μεταβολής των ταχυτήτων κατά τη διάρκεια εκτέλεσης του αλγορίθμου A* στο άγνωστο περιβάλλον.



Σχήμα 5.27: Γράφημα μεταβολής της απευθείας απόστασης από τη θέση στόχο, σε m, κατά τη διάρκεια εκτέλεσης του αλγορίθμου A* στο άγνωστο περιβάλλον.



Σχήμα 5.28: Γράφημα μεταβολής των δεδομένων του αισθητήρα λέιζερ, για τις 5 αριστερές ακτίνες, κατά τη διάρκεια εκτέλεσης του αλγορίθμου A* στο άγνωστο περιβάλλον. Η απόσταση μετράται σε m.



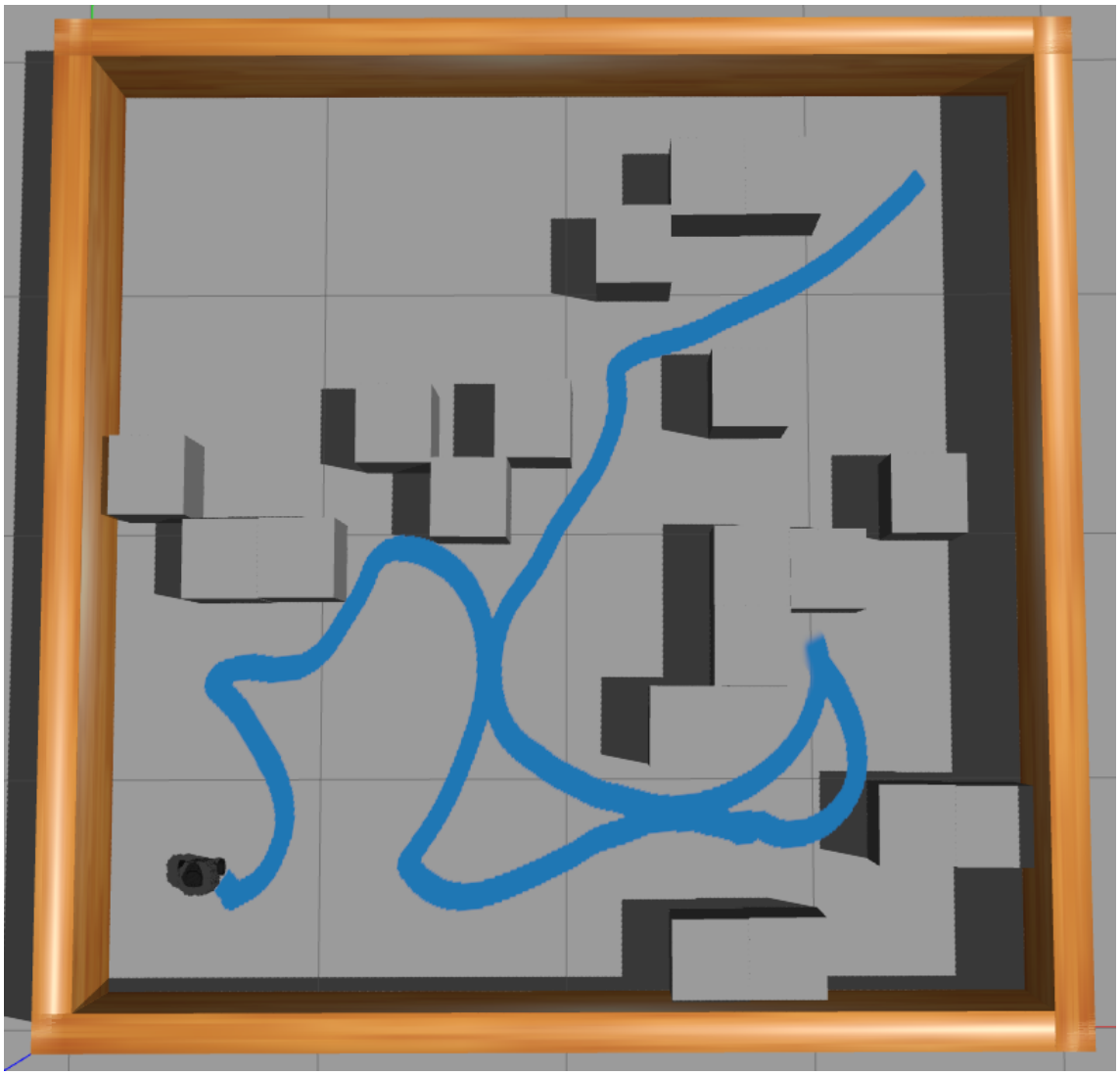
Σχήμα 5.29: Γράφημα μεταβολής των δεδομένων του αισθητήρα λέιζερ, για τις 5 δεξιές ακτίνες, κατά τη διάρκεια εκτέλεσης του αλγορίθμου A* στο άγνωστο περιβάλλον. Η απόσταση μετράται σε m.

Δεύτερο άγνωστο περιβάλλον

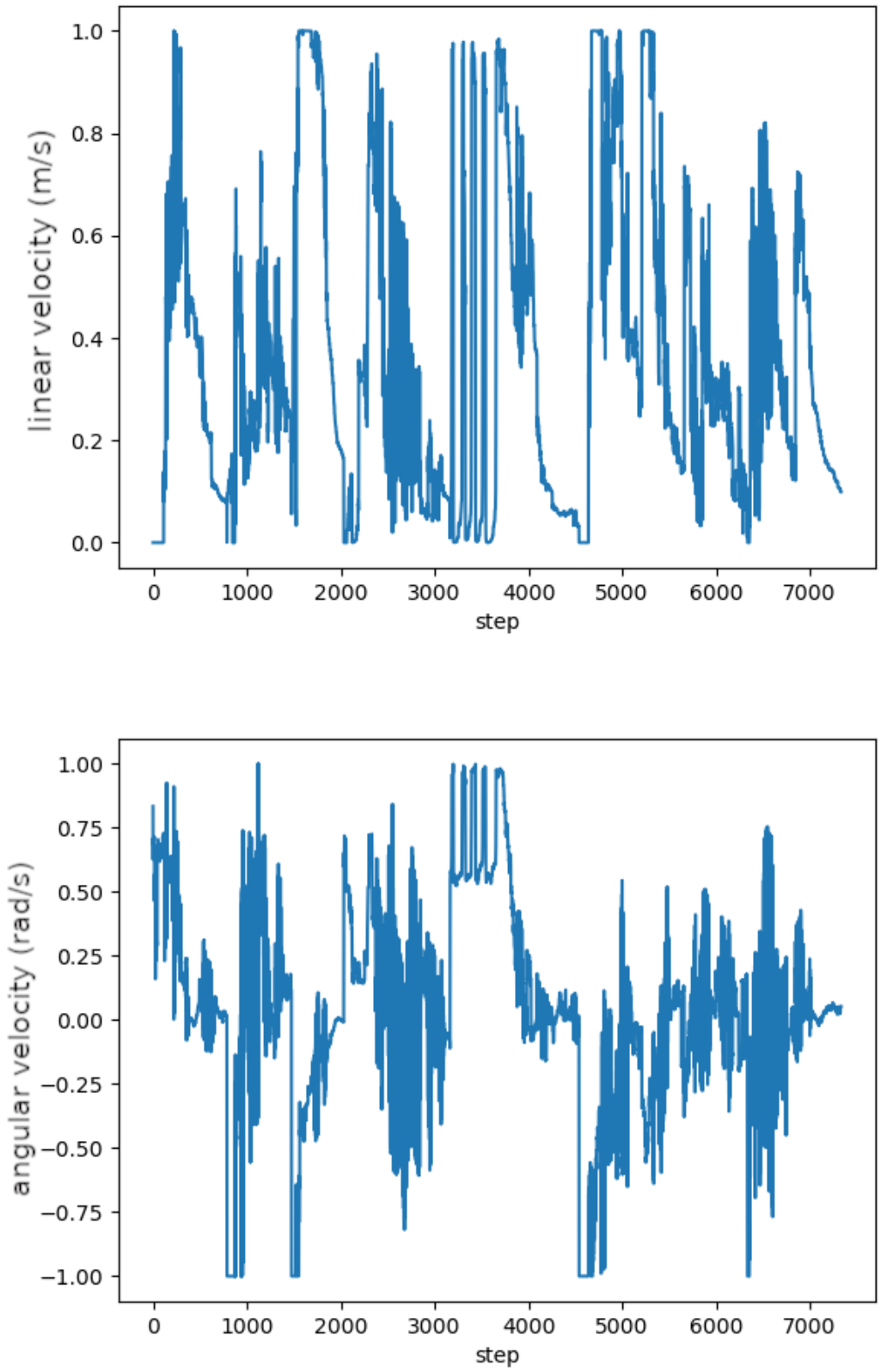
Σε αυτό το περιβάλλον, όπως και πρίν, το ρομπότ εκκινεί από τη θέση $(0.5, 0.5)$ με θέση στόχο την $(3.5, 3.5)$ και εκτελείται ξανά όμοια δοκιμή του αλγορίθμου A*. Όπως αναφέρθηκε και νωρίτερα, σε αυτό το περιβάλλον το επίπεδο συνθετότητας αυξάνεται πολύ συγκριτικά με πριν. Γι΄ αυτό το λόγο και μεταβάλλεται και η συμπεριφορά του αλγορίθμου.

Στο Σχήμα 5.30 φαίνεται η τροχιά που ακολούθησε το ρομποτικό όχημα εντός του δοκιμαστικού περιβάλλοντος. Στο Σχήμα 5.31 απεικονίζεται η γραμμική και γωνιακή ταχύτητα που δίνεται στο ρομπότ από τον πράκτορα DDPG, στο Σχήμα 5.32 φαίνεται η απόσταση από την θέση στόχο του και στο Σχήμα 5.33 φαίνονται τα δεδομένα από το μετρητή αποστάσεων λέιζερ, στην αριστερή και δεξιά μεριά του ρομπότ.

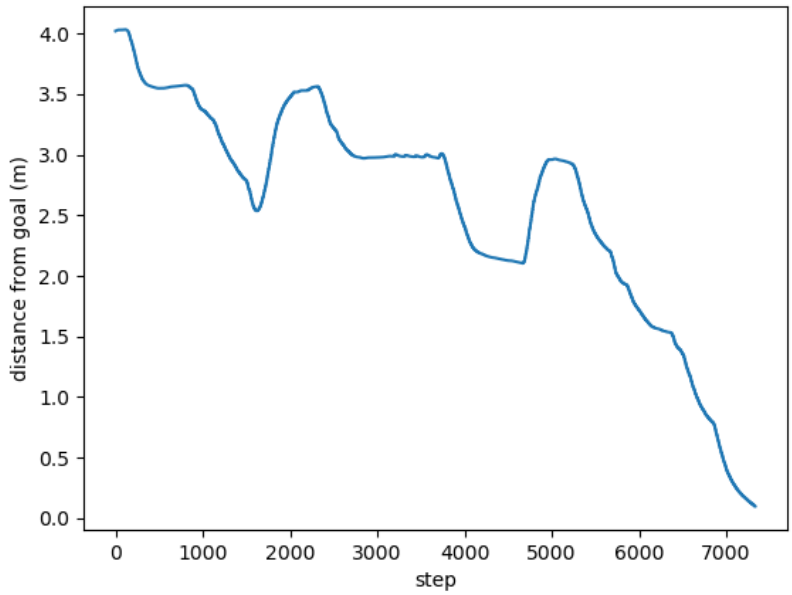
Ο χρόνος που χρειάστηκε στο ρομπότ να κινηθεί από τη θέση εκκίνησης στη θέση τερματισμού ήταν 70.49 δευτερόλεπτα, ενώ η διανυθείσα απόσταση ανέρχεται στα $11.6m$.



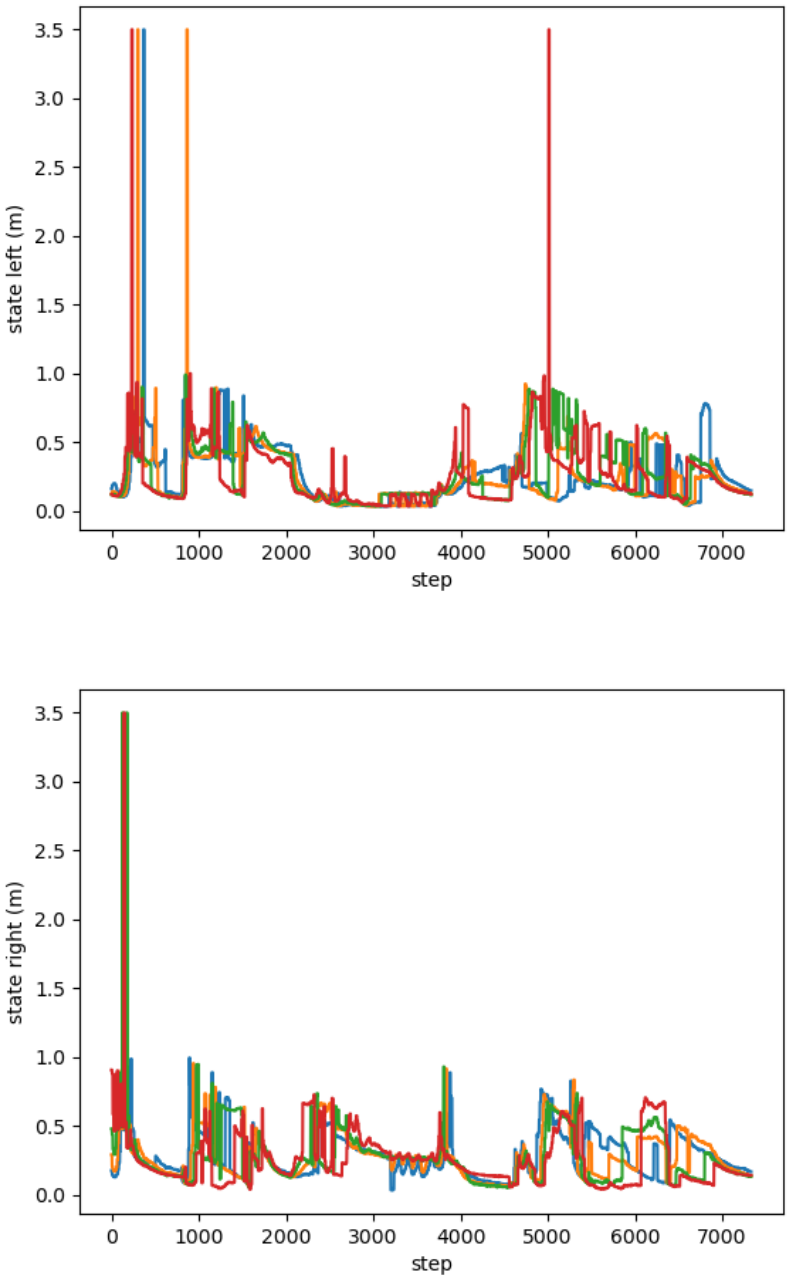
Σχήμα 5.30: Η τροχιά που ακολούθησε το ρομποτικό όχημα στο δεύτερο άγνωστο περιβάλλον, σύμφωνα με τους κόμβους που του υπαγόρευε ο αλγόριθμος A*.



Σχήμα 5.31: Γραφήματα μεταβολής των ταχυτήτων κατά τη διάρκεια εκτέλεσης του αλγορίθμου A* στο δεύτερο άγνωστο περιβάλλον.



Σχήμα 5.32: Γράφημα μεταβολής της απευθείας απόστασης από τη θέση στόχο, σε m, κατά τη διάρκεια εκτέλεσης του αλγορίθμου A* στο δεύτερο άγνωστο περιβάλλον.



Σχήμα 5.33: Γράφημα μεταβολής των δεδομένων του αισθητήρα λέιζερ, για τις 5 αριστερές ακτίνες και τις 5 δεξιές ακτίνες, κατά τη διάρκεια εκτέλεσης του αλγορίθμου A* στο δεύτερο άγνωστο περιβάλλον. Η απόσταση μετράται σε m.

5.3.2 Εκτέλεση CIA*

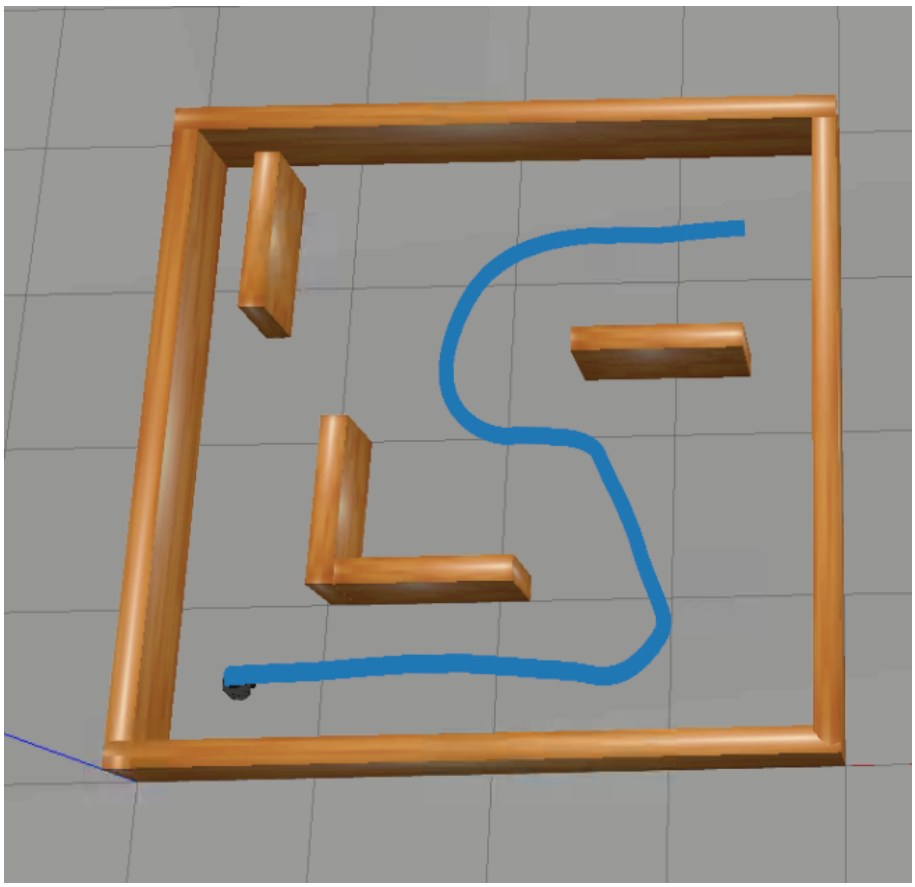
Όμοια με πρίν, εκτελείται στα ίδια περιβάλλοντα με ίδιες συνθήκες έναρξης και τερματισμού ο αλγόριθμος CIA*.

Γνωστό περιβάλλον

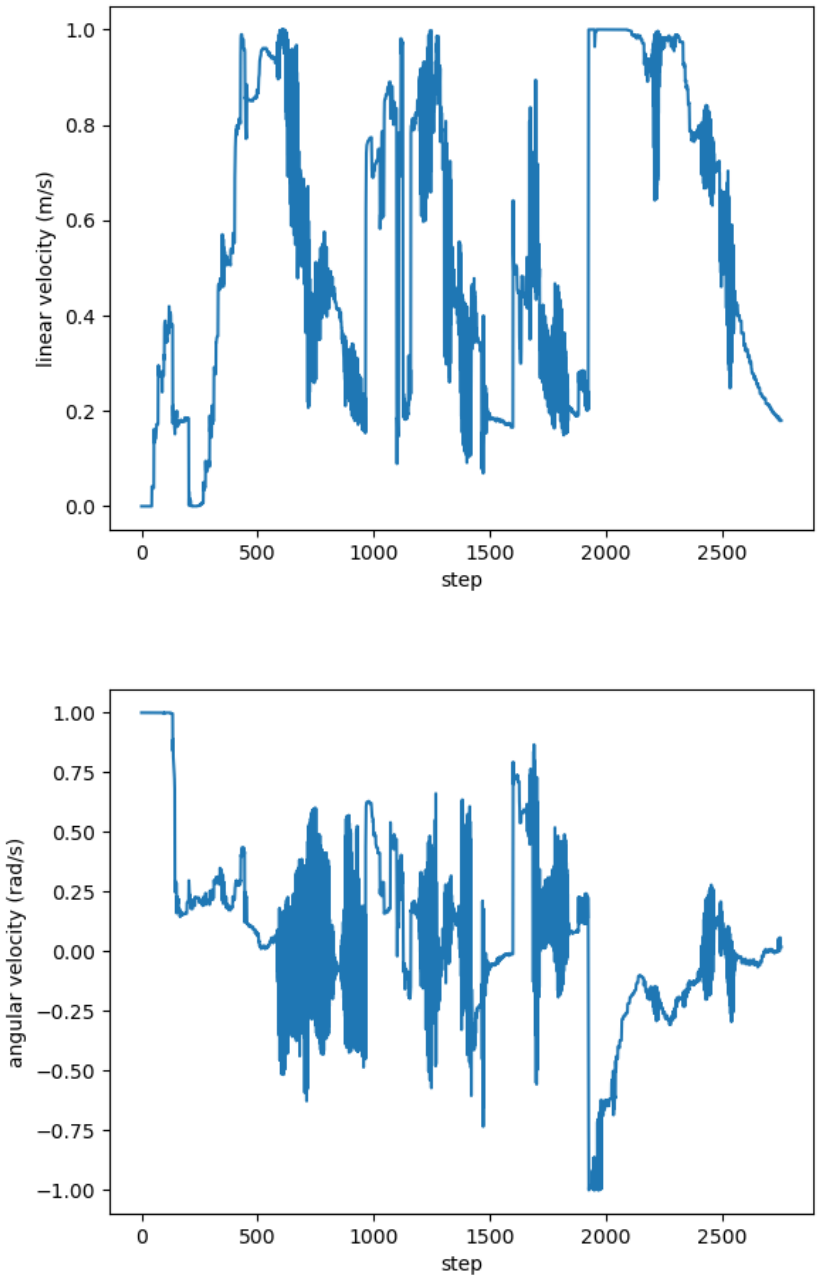
Εκτελείται η ίδια δοκιμή στο γνωστό περιβάλλον, αυτή τη φορά με χρήση του αλγορίθμου CIA* ως τον αλγόριθμο εύρεσης συντομότερης διαδρομής. Το ρομπότ εκκινεί από τη θέση $(0.5, 0.5)$ με θέση στόχο την $(3.5, 3.5)$. Το ρομπότ εκτελεί μια ελαφρώς διαφοροποιημένη διαδρομή σε σχέση με τον κανονικό A*, επιλέγοντας να μετακινηθεί ανάμεσα από τα 2 δεξιά εμπόδια, αντί ανάμεσα από τα 2 αριστερά.

Στο Σχήμα 5.34 φαίνεται η τροχιά που ακολούθησε το ρομποτικό όχημα εντός του δοκιμαστικού περιβάλλοντος. Στο Σχήμα 5.35 απεικονίζεται η γραμμική και γωνιακή ταχύτητα που έχει το ρομπότ, στο Σχήμα 5.36 φαίνεται η απόσταση από την θέση στόχο του και στα Σχήματα 5.37 και 5.38 φαίνονται τα δεδομένα από το μετρητή αποστάσεων λέιζερ, στην αριστερή και δεξιά μεριά του ρομπότ.

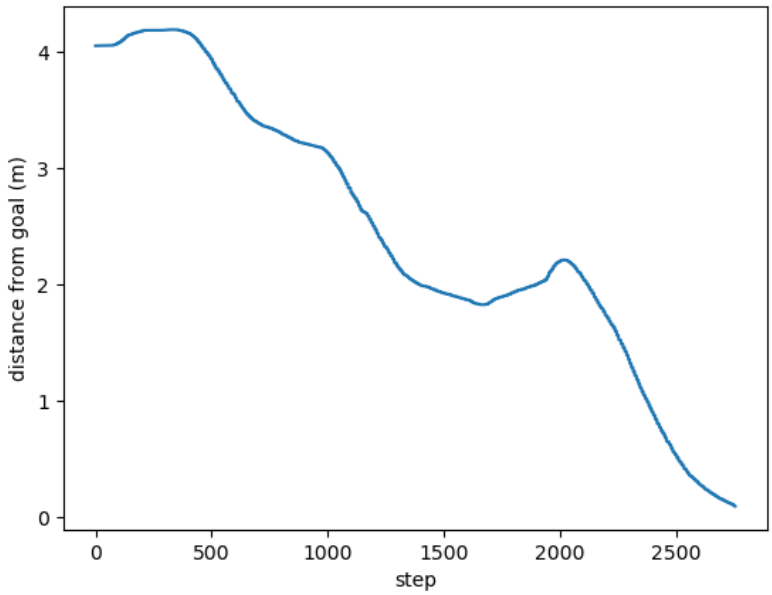
Ο χρόνος που χρειάστηκε στο ρομπότ να κινηθεί από τη θέση εκκίνησης στη θέση τερματισμού ήταν 32.94 δευτερόλεπτα, ενώ η διανυθείσα απόσταση ανέρχεται στα $6.6m$.



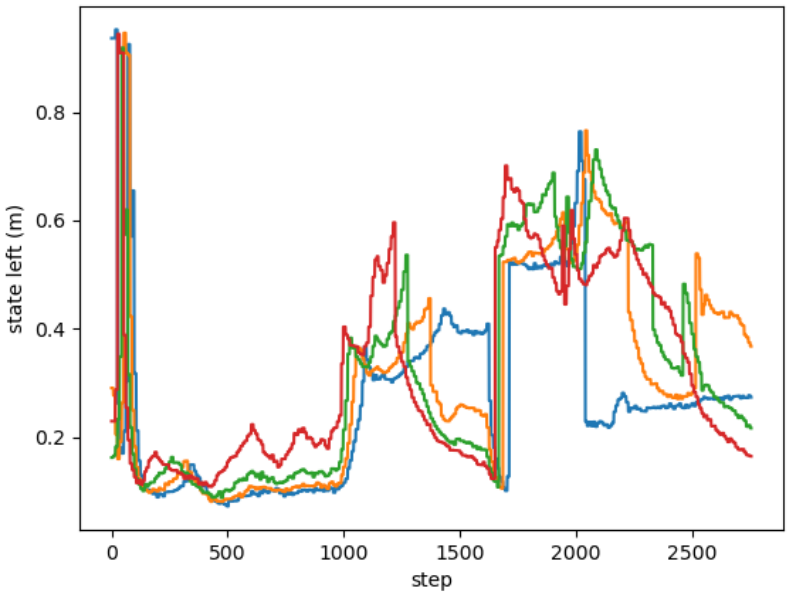
Σχήμα 5.34: Η τροχιά που ακολούθησε το ρομποτικό όχημα στο γνωστό περιβάλλον, σύμφωνα με τους κόμβους που του υπαγόρευε ο αλγόριθμος CIA*.



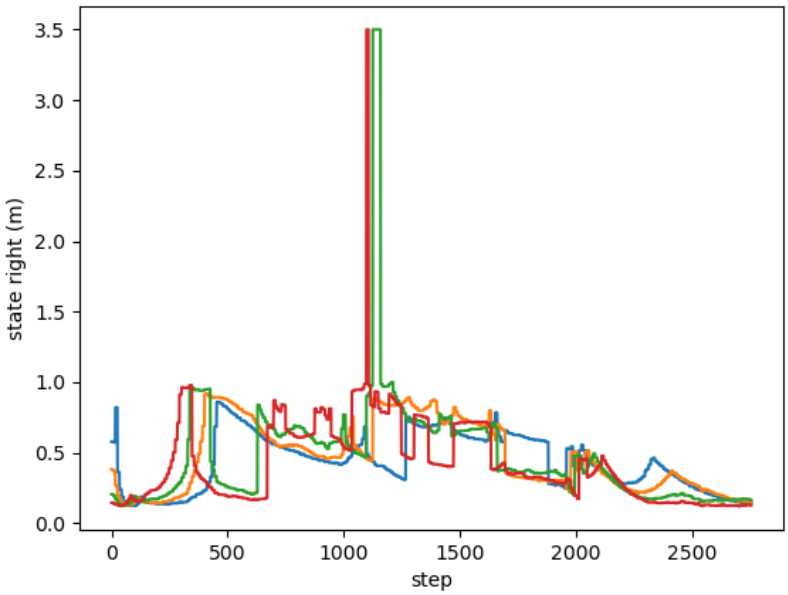
Σχήμα 5.35: Γράφημα μεταβολής των ταχυτήτων κατά τη διάρκεια εκτέλεσης του αλγορίθμου CIA* στο γνωστό περιβάλλον. Με πορτοκαλί χρώμα συμβολίζεται η γραμμική ταχύτητα σε m/s και με μπλέ χρώμα η γωνιακή ταχύτητα σε rad/s.



Σχήμα 5.36: Γράφημα μεταβολής της απευθείας απόστασης από τη θέση στόχο, σε m, κατά τη διάρκεια εκτέλεσης του αλγορίθμου CIA* στο γνωστό περιβάλλον.



Σχήμα 5.37: Γράφημα μεταβολής των δεδομένων του αισθητήρα λέιζερ, για τις 5 αριστερές ακτίνες, κατά τη διάρκεια εκτέλεσης του αλγορίθμου CIA* στο γνωστό περιβάλλον. Η απόσταση μετράται σε m.



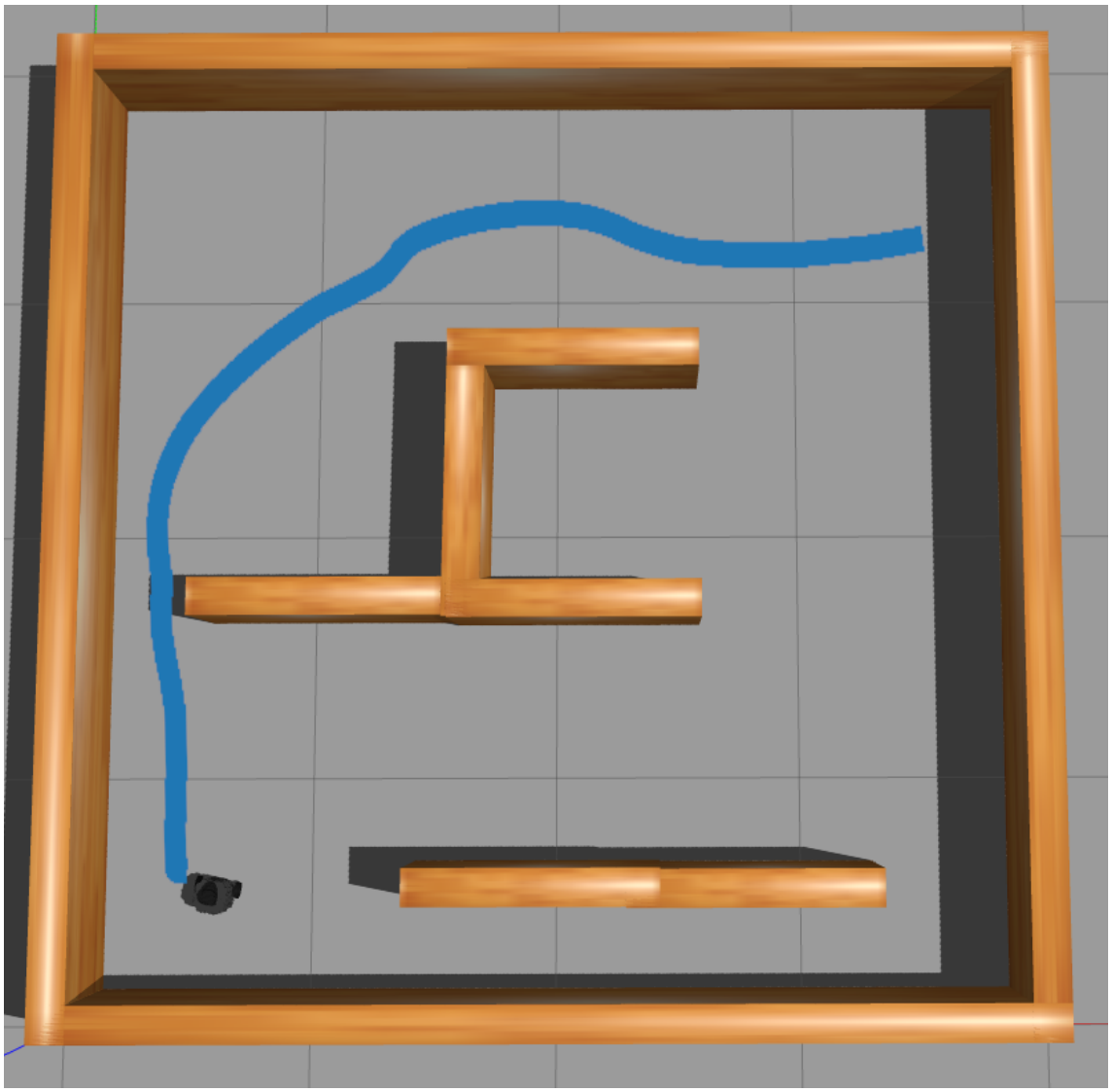
Σχήμα 5.38: Γράφημα μεταβολής των δεδομένων του αισθητήρα λέιζερ, για τις 5 δεξιές ακτίνες, κατά τη διάρκεια εκτέλεσης του αλγορίθμου CIA* στο γνωστό περιβάλλον. Η απόσταση μετράται σε m.

Πρώτο άγνωστο περιβάλλον

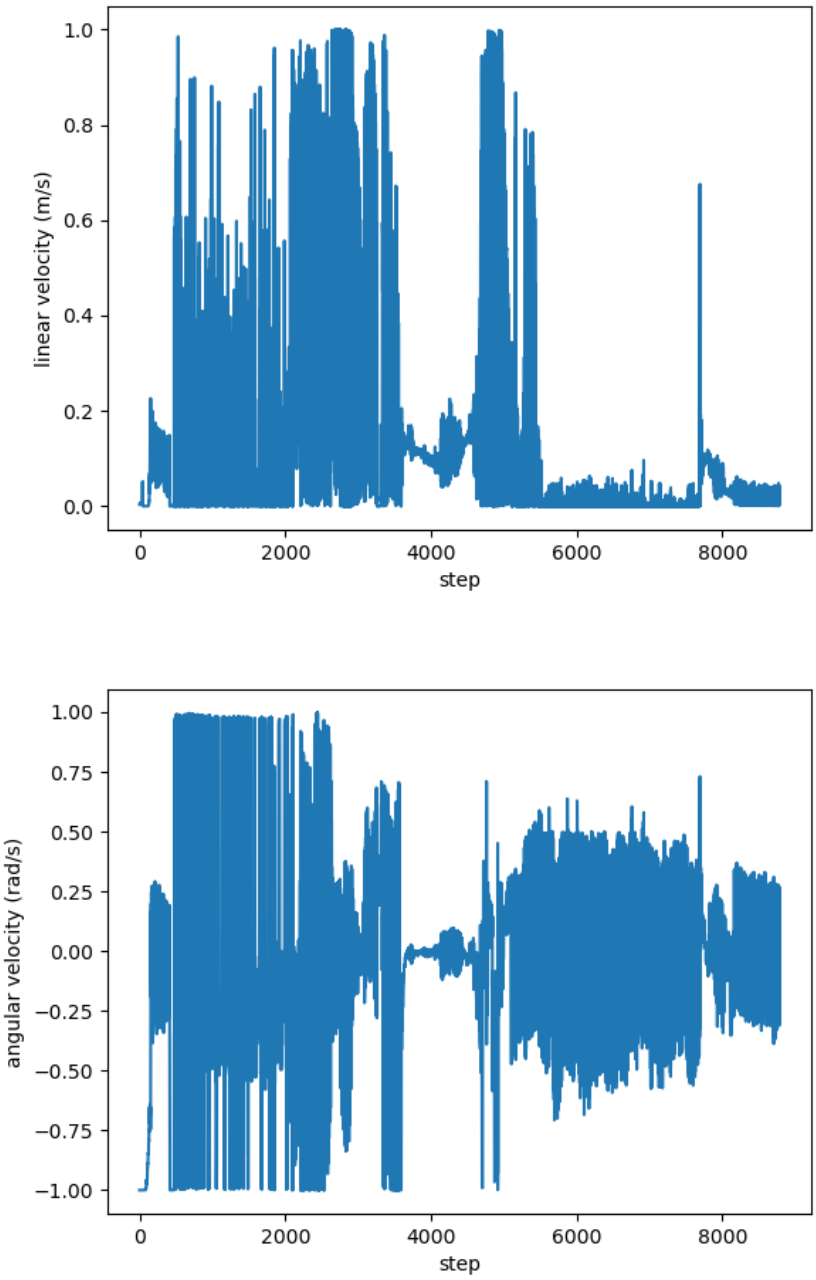
Όμοίως, εκτελείται ο αλγόριθμος CIA* για το πρώτο άγνωστο περιβάλλον, με ίδιες ακριβώς συνθήκες εκκίνησης και τερματισμού. Το ρομπότ ακολουθεί μια διαφορετική διαδρομή σε σχέση με τον απλό A*, επιλέγοντας να μετακινηθεί στην αριστερή μεριά του περιβάλλοντος αντι στη δεξιά.

Στο Σχήμα 5.39 φαίνεται η τροχιά που ακολούθησε το ρομποτικό όχημα εντός του δοκιμαστικού περιβάλλοντος. Στο Σχήμα 5.40 απεικονίζεται η γραμμική και γωνιακή ταχύτητα που έχει το ρομπότ, στο Σχήμα 5.41 φαίνεται η απόσταση από την θέση στόχο του και στα Σχήματα 5.42 και 5.43 φαίνονται τα δεδομένα από το μετρητή αποστάσεων λέιζερ, στην αριστερή και δεξιά μεριά του ρομπότ.

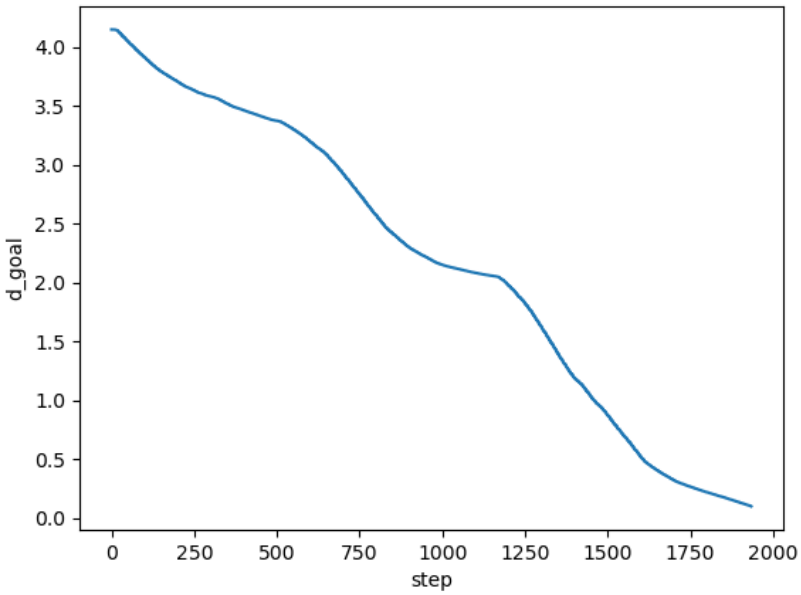
Ο χρόνος που χρειάστηκε στο ρομπότ να κινηθεί από τη θέση εκκίνησης στη θέση τερματισμού ήταν 29.68 δευτερόλεπτα, ενώ η διανυθείσα απόσταση ανέρχεται στα $5.1m$.



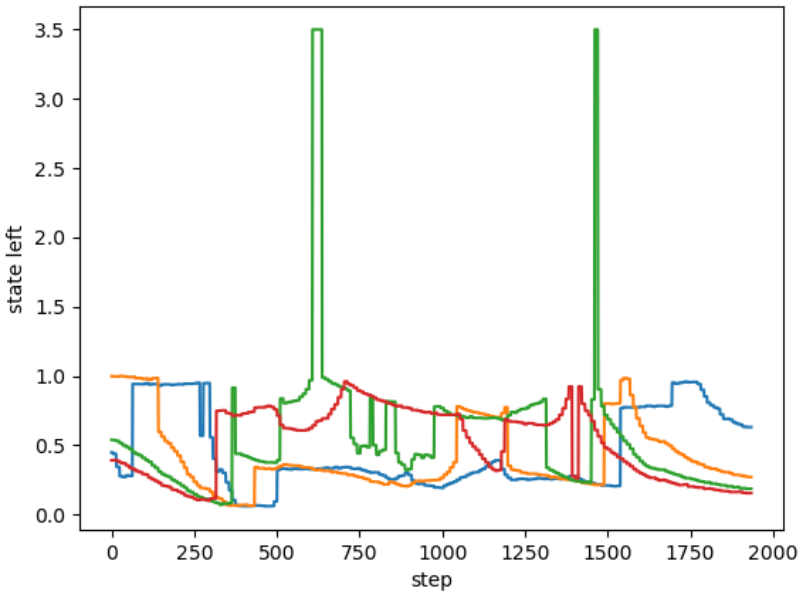
Σχήμα 5.39: Η τροχιά που ακολούθησε το ρομποτικό όχημα στο άγνωστο περιβάλλον, σύμφωνα με τους κόμβους που του υπαγόρευε ο αλγόριθμος CIA*.



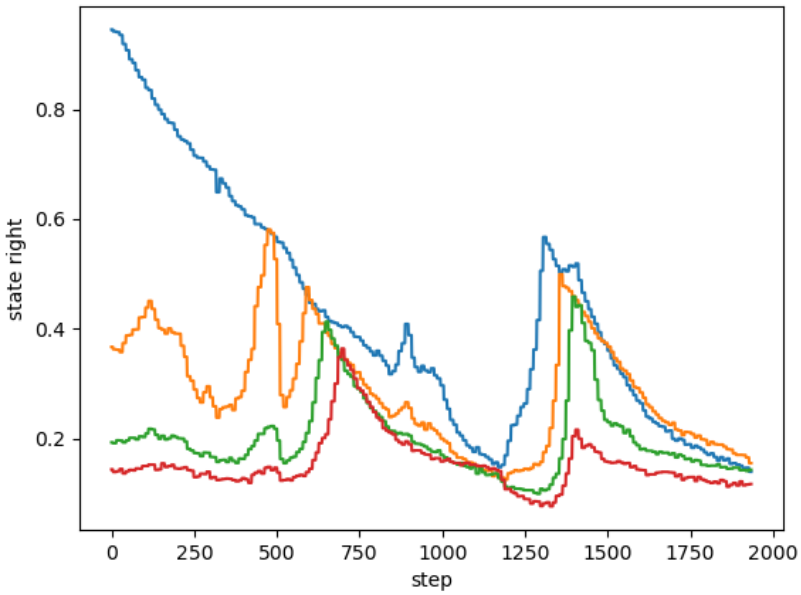
Σχήμα 5.40: Γράφημα μεταβολής των ταχυτήτων κατά τη διάρκεια εκτέλεσης του αλγορίθμου CIA* στο άγνωστο περιβάλλον.



Σχήμα 5.41: Γράφημα μεταβολής της απευθείας απόστασης από τη θέση στόχο, σε m, κατά τη διάρκεια εκτέλεσης του αλγορίθμου CIA* στο άγνωστο περιβάλλον.



Σχήμα 5.42: Γράφημα μεταβολής των δεδομένων του αισθητήρα λέιζερ, για τις 5 αριστερές ακτίνες, κατά τη διάρκεια εκτέλεσης του αλγορίθμου CIA* στο άγνωστο περιβάλλον. Η απόσταση μετράται σε m.



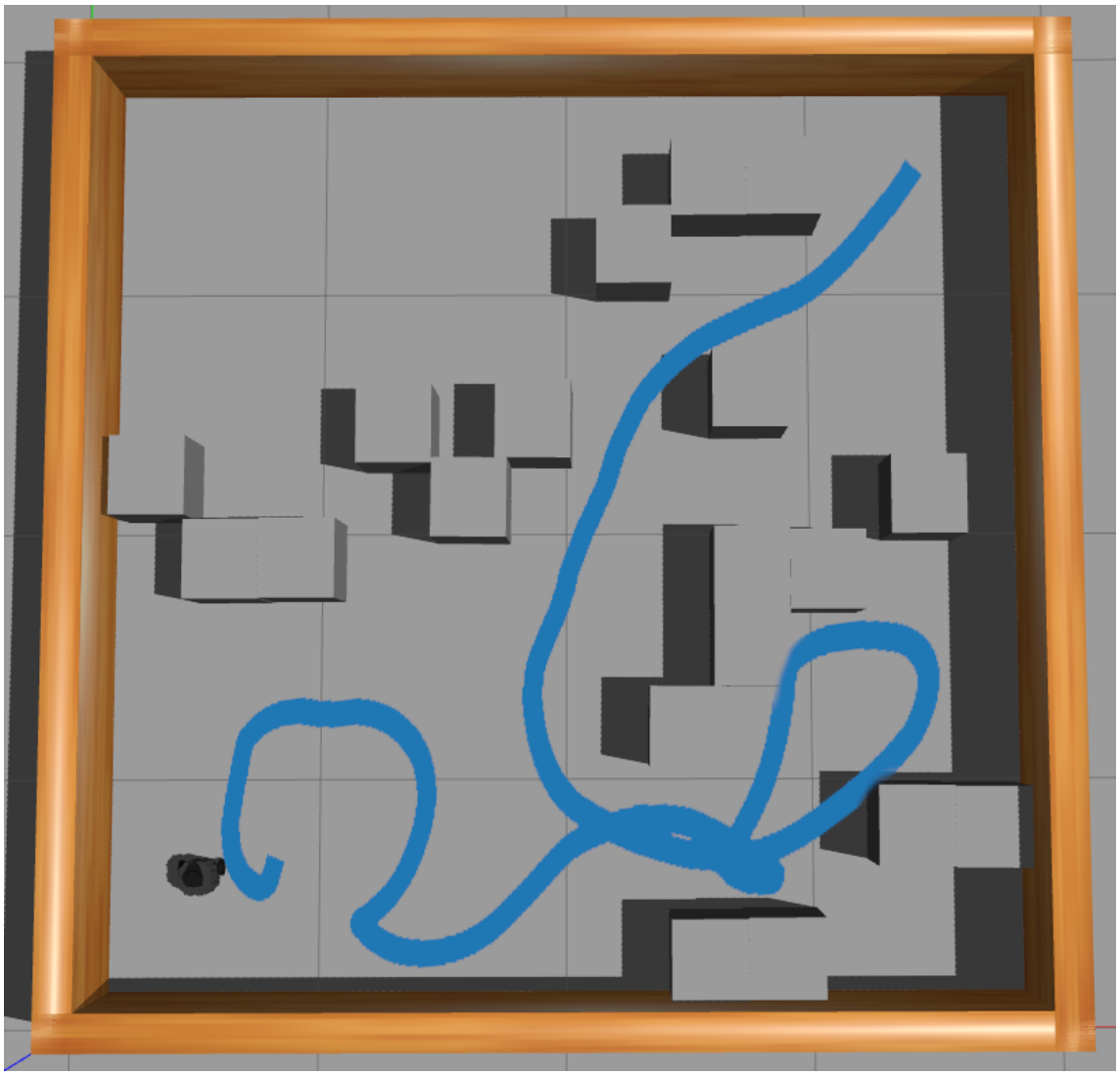
Σχήμα 5.43: Γράφημα μεταβολής των δεδομένων του αισθητήρα λέιζερ, για τις 5 δεξιές ακτίνες, κατά τη διάρκεια εκτέλεσης του αλγορίθμου CIA* στο άγνωστο περιβάλλον. Η απόσταση μετράται σε m.

Δεύτερο άγνωστο περιβάλλον

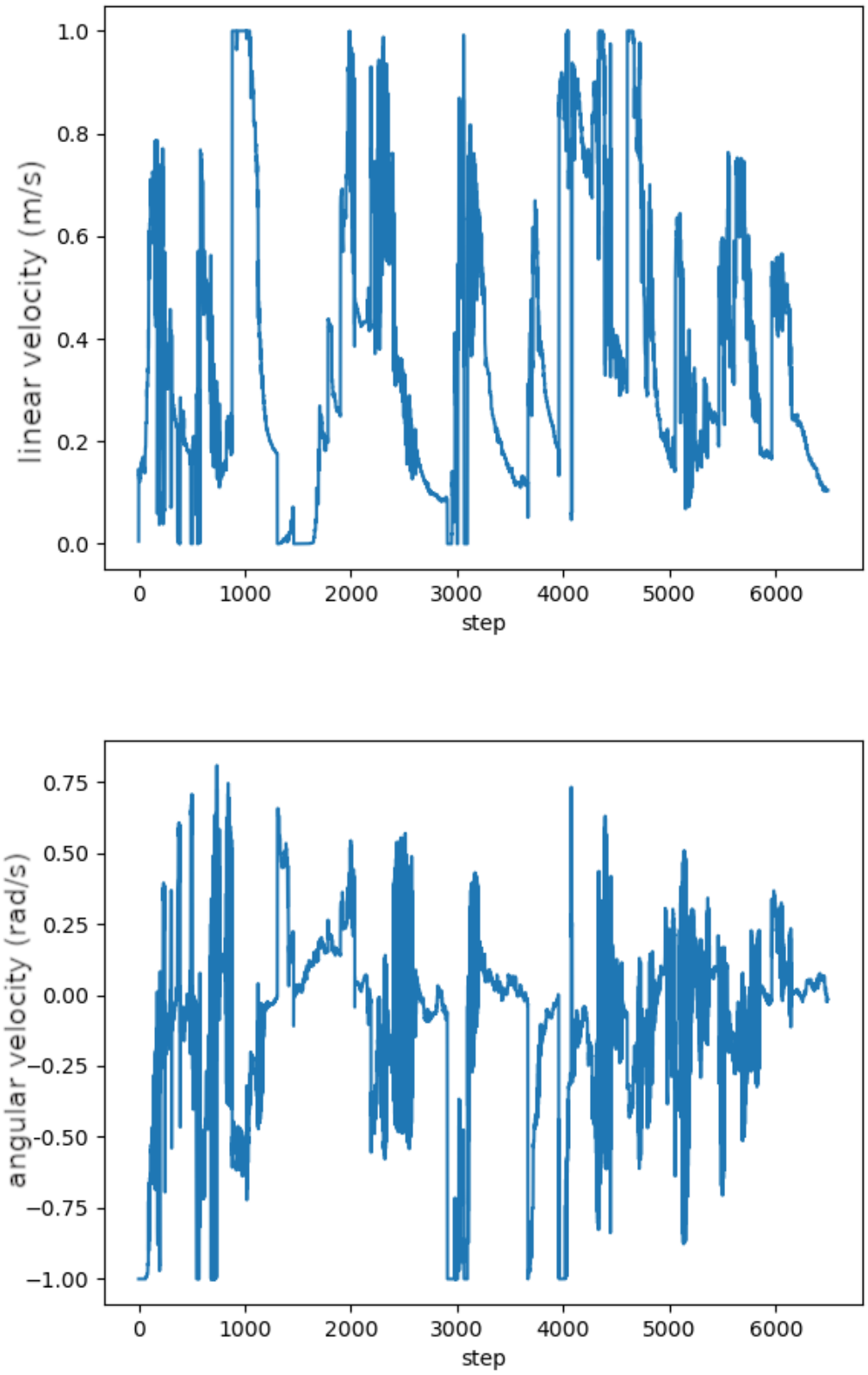
Με αντίστοιχη φιλοσοφία εκτελείται ξανά η δοκιμή του CIA* και στο δεύτερο άγνωστο περιβάλλον, με το ρομπότ να ξεκινάει από τη θέση $(0.5, 0.5)$ με θέση στόχο την $(3.5, 3.5)$.

Στο Σχήμα 5.44 φαίνεται η τροχιά που ακολούθησε το ρομποτικό όχημα εντός του δοκιμαστικού περιβάλλοντος. Στο Σχήμα 5.45 απεικονίζεται η γραμμική και γωνιακή ταχύτητα που έχει το ρομπότ, στο Σχήμα 5.46 φαίνεται η απόσταση από την θέση στόχο του και στο Σχήμα 5.47 φαίνονται τα δεδομένα από το μετρητή αποστάσεων λέιζερ, στην αριστερή και δεξιά μεριά του ρομπότ.

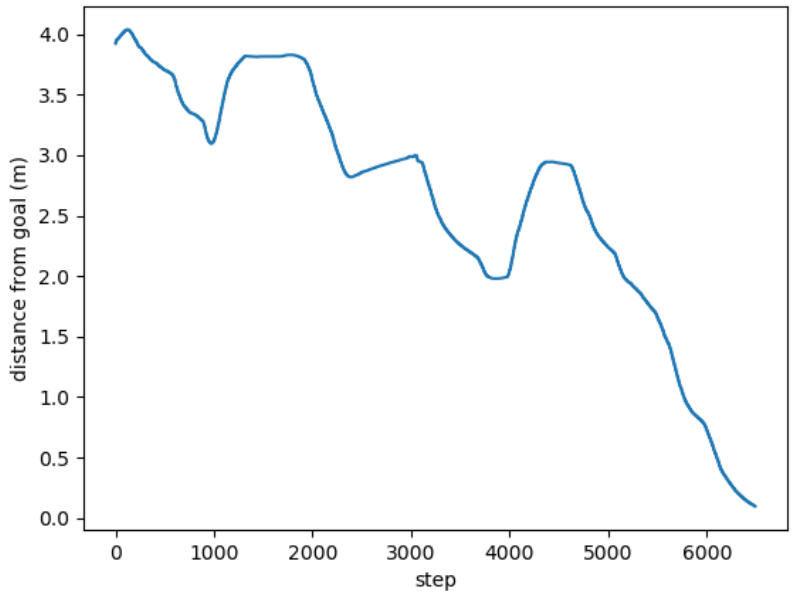
Ο χρόνος που χρειάστηκε στο ρομπότ να κινηθεί από τη θέση εκκίνησης στη θέση τερματισμού, χρησιμοποιώντας τον αλγόριθμο CIA* για την υπαγόρευση των κινήσεων στα νευρωνικά δίκτυα DDPG ήταν 58.94 δευτερόλεπτα, ενώ η διανυθείσα απόσταση ανέρχεται στα $9.8m$.



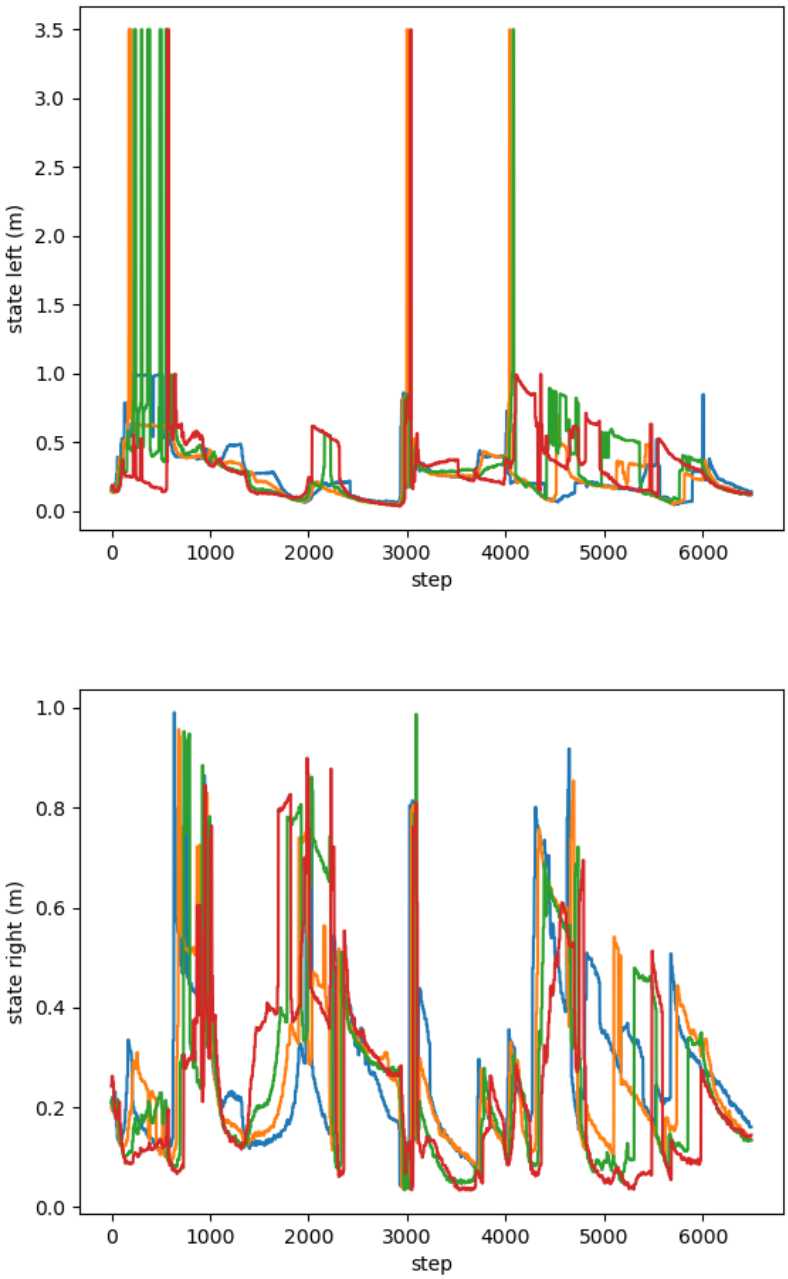
Σχήμα 5.44: Η τροχιά που ακολούθησε το ρομποτικό όχημα στο δεύτερο άγνωστο περιβάλλον, σύμφωνα με τους κόμβους που του υπαγόρευε ο αλγόριθμος CIA*.



Σχήμα 5.45: Γραφήματα μεταβολής των ταχυτήτων κατά τη διάρκεια εκτέλεσης του αλγορίθμου CIA* στο δεύτερο άγνωστο περιβάλλον.



Σχήμα 5.46: Γράφημα μεταβολής της απευθείας απόστασης από τη θέση στόχο, σε m, κατά τη διάρκεια εκτέλεσης του αλγορίθμου CIA* στο δεύτερο άγνωστο περιβάλλον.



Σχήμα 5.47: Γράφημα μεταβολής των δεδομένων του αισθητήρα λέιζερ, για τις 5 αριστερές ακτίνες και τις 5 δεξιές ακτίνες, κατά τη διάρκεια εκτέλεσης του αλγορίθμου CIA* στο δεύτερο άγνωστο περιβάλλον. Η απόσταση μετράται σε m.

5.3.3 Σύγκριση αλγορίθμων

Στα αποτελέσματα των πειραματικών δοκιμών,στα δύο πρώτα περιβάλλοντα, το γνωστό περιβάλλον αλλά και το πρώτο άγνωστο, οι δύο αλγόριθμοι είναι πάρα πολύ κοντά σε ποιότητα. Ναι μεν και στις 2 περιπτώσεις επέλεξαν διαφορετικές διαδρομές, όμως σε ποιότητα, δηλαδή τόσο χρόνο αναζήτησης όσο και απόσταση που δυήνησαν, ήταν τόσο κοντά, που δεν μπορεί να αποφανθεί αν ο ένας είναι καλύτερος από τον άλλον. Συμπερασματικά, παρόλο που ο CIA* είναι αδιαμφισβήτητα ένας πιο προηγμένος και εξελιγμένος αλγόριθμος σε σχέση με τον A*, σε μικρά περιβάλλοντα με εμπόδια τέτοιου τύπου, όπως αυτά στα πρώτα 2 περιβάλλοντα, δεν

υπάρχει πλεονέκτημα.

Όμως, τα πειραματικά αποτελέσματα αλλάζουν ραγδαία σε ένα σημαντικά πιο σύνθετο περιβάλλον, όπως αυτό του τρίτου. Εκεί, ο A* ακολούθησε μια σημαντικά μεγαλύτερη σε απόσταση διαδρομή σε σχέση με τον CIA*, κάνοντας πολλαπλές αναστροφές επειδή η τροχιά που επέλεξε να κινηθεί δεν ήταν κοντά στη βέλτιστη. Ο CIA* ήταν σημαντικά πιο αποτελεσματικός τόσο σε χρόνο όσο και σε διανυθείσα απόσταση, καθιστώντας τον γενικά έναν πιο αποτελεσματικό αλγόριθμο. Είναι έυκολο να προβλεφθεί από τα εν λόγω πειραματικά αποτελέσματα ότι σε ένα ακόμη πιο σύνθετο περιβάλλον κίνησης του ρομπότ, ο αλγόριθμος CIA* θα είναι πολύ πιο αποτελεσματικός συγκριτικά με τον παραδοσιακό A*. Αυτό οφείλεται στο πληροφορούμενο ευρετικό σκέλος αυτού του αλγορίθμου, το οποίο έχει μια πιο εποπτική εικόνα για το περιβάλλοον του, σε σχέση με ένα ευρετικό σκέλος που βασίζεται αποκλειστικά στην ευκλείδια ή τη Manhattan απόσταση μεταξύ του τρέχοντος κόμβου και του κόμβου - στόχου.

Στον Πίνακα 5.1 παρουσιάζονται οι αριθμητικές συγκρίσεις των πειραματικών αποτελεσμάτων για τα 3 παραπάνω πειράματα. Για τα πρώτα 2 πειράματα, τα αριθμητικά αποτελέσματα ήταν τόσο κοντά που δεν μπορεί να ληφθεί ξεκάθαρη απόφαση για το ποιος αλγόριθμος είναι αποτελεσματικότερος, όμως στο τρίτο περιβάλλον η διαφορά είναι μετρήσιμη και ξεκάθαρη ότι ο αλγόριθμος CIA* λειτουργεί αποτελεσματικότερα.

Περιβάλλον	A*		CIA*	
	Απόσταση (m)	Χρόνος (sec)	Απόσταση (m)	Χρόνος (sec)
Γνωστό περιβάλλον	6.7	38.44	6.6	32.94
Πρώτο άγνωστο περιβάλλον	6.8	38.23	5.1	29.68
Δεύτερο άγνωστο περιβάλλον	11.6	70.49	9.8	58.94

Πίνακας 5.1: Συγκριτικό πειραματικών αποτελεσμάτων, από τα 3 περιβάλλοντα και τις 2 εκτελέσεις των αλγορίθμων σε κάθε περιβάλλον.

Κεφάλαιο 6

Συμπεράσματα και μελλοντικές εφαρμογές

Στα πλαίσια αυτής της διπλωματικής εργασίας, αναπτύχθηκε σύστημα σχεδιασμού τροχιάς και αποφυγής εμποδίων για έντροχα ρομποτικά οχήματα που κινούνται βάσει της αρχής της διαφορικής κίνησης. Το σύστημα αυτό λειτουργεί με τον αλγόριθμο DDPG, ο οποίος χρησιμοποιήθηκε τόσο για την εκπαίδευση των νευρωνικών δικτύων που απαρτίζουν το εν λόγω σύστημα, αλλά και για την χρήση του σε συνδυασμό με τους ευρετικούς αλγορίθμους εύρεσης συντομότερης διαδρομής, A* και CIA*. Οι αλγόριθμοι υπαγόρευαν στο σύστημα τις κινήσεις του ρομπότ που εξήγαγαν, και το σύστημα καθοδηγούσε το ρομπότ ελέγχοντας τις παραμέτρους γραμμικής και γωνιακής ταχύτητας, με στόχο την άφιξή του στη θέση στόχο που του έχει οριστεί.

Αρχικά, δοκιμάστηκε το σύστημα σχεδιασμού τροχιάς μόνο του, σε 3 γνωστά περιβάλλοντα, προκειμένου να δοκιμαστεί η ορθή λειτουργία του. Πραγματοποιήθηκαν πειράματα και διαπιστώθηκε ότι το σύστημα είχε εκπαιδευτεί κατάλληλα, εφόσον ήταν σε θέση να μετακινήσει το ρομπότ με επιτυχία, αποφεύγοντας όλα τα εμπόδια στο δρόμο του.

Σε επόμενη φάση, το σύστημα δοκιμάστηκε σε συνδυασμό με τους ευρετικούς αλγορίθμους εύρεσης συντομότερης διαδρομής σε 3 περιβάλλοντα, ένα γνωστό περιβάλλον, ένα εύκολο άγνωστο και ένα δύσκολο άγνωστο. Πραγματοποιήθηκαν δοκιμές και για τους 2 ευρετικούς αλγορίθμους, A* και CIA*, και στα 3 περιβάλλοντα. Εξάχθηκαν δεδομένα για τον αισθητήρα λέιζερ, αλλά και τις ταχύτητες του ρομπότ.

Σε όλες αυτές τις δοκιμές παρατηρήθηκε ότι ο πράκτορας υπαγόρευε εντολές στο ρομπότ με επιτυχία, κινώντας το μέσα στο περιβάλλον δίχως να συγκρούεται με αυτό. Παράλληλα, οι ευρετικοί αλγόριθμοι επέλεγαν την επόμενη κίνηση του ρομπότ, βάσει του περιβάλλοντος και των εμποδίων του. Στόχος των αλγορίθμων ήταν η κατάλληλη καθοδήγηση του πράκτορα, ώστε να κινείται με βέλτιστο τρόπο, τόσο χρονικό όσο και αποστασιακό. Αυτό χρειάζεται επειδή ο πράκτορας μηχανικής μάθησης δε διαθέτει το βέλτιστο στη φιλοσοφία του, ο στόχος του είναι απλώς η καθοδήγηση του ρομπότ στη θέση στόχο.

Το σύστημα σχεδιασμού τροχιάς σε συνδυασμό με τους ευρετικούς αλγορίθμους θα μπορούσε να χρησιμοποιηθεί σε διάφορες εφαρμογές που διαθέτουν κινητά ρομπότ σε άγνωστα ή γνωστά περιβάλλοντα. Για παράδειγμα σε ρομπότ που χρησιμοποιούνται σε έρευνα και διάσωση ή ρομπότ αναζήτησης, ακόμη και σε οικιακά ρομπότ όπως για παράδειγμα αυτό που παρουσιάζεται στο Σχήμα 6.1.

Επιπλέον, θα μπορούσε να βρεί εφαρμογές και το σύστημα αποφυγής εμποδίων μόνο του σε εφαρμογές που δεν είναι απαραίτητη η εύρεση της συντομότερης διαδρομής, αλλά το περιβάλλον

Σχήμα 6.1: Ρομποτικό όχημα οικιακής καθαριότητας, από την εταιρεία iRobot

είναι άγνωστο επειδή μεταβάλλεται συνεχώς. Ένα τέτοιο περιβάλλον θα μπορούσε να είναι μια μονάδα παραγωγής, όπου είναι πιθανό να τοποθετηθούν αντικείμενα εμπόδια σε διάφορα σημεία που δεν μπορούν να προβλευθούν, αλλά και γενικά σε διάφορες βιομηχανικές τοποθεσίες, όπου συναντούνται αντίστοιχα προβλήματα.

Το σύστημα αυτό μπορεί, μετά από μερικές κατάλληλες τροποποιήσεις, να εφαρμοσθεί και σε άλλες ρομποτικές πλατφόρμες, όπως για παράδειμα κάποια εναέρια ή ακόμη και κάποια άλλου είδους έντροχη ρομποτική πλατφόρμα. Θα μπορούσε ακόμη να εφαρμοστεί και σε περιπτώσεις αυτόνομης οδήγησης, εφόσον πρώτον τροποποιηθεί κατάλληλα το μοντέλο και δεύτερον, πέρα από τον αισθητήρα μέτρησης αποστάσεων λέιζερ υλοποιηθεί και παροχή δεδομένων από κάμερα για την αναγνώριση των σημάνσεων κυκλοφορίας και τη θέση του οχήματος στο δρόμο.

Bibliography

[1] E.W. Dijkstra. ≪A note on two problems in connexion with graphs.≫ In: (1959). URL: https://doi.org/10.1007/BF01386390.

[2] Peter Hart, Nils Nilsson, and Bertram Raphael. ≪A Formal Basis for the Heuristic Determination of Minimum Cost Paths≫. In: *IEEE Transactions on Systems Science and Cybernetics* 4.2 (1968), pp. 100–107. DOI: 10.1109/tssc.1968.300136. URL: https://doi.org/10.1109/tssc.1968.300136.

[3] Athanasios Ch. Kapoutsis et al. ≪Continuously informed heuristic A-optimal path retrieval inside an unknown environment≫. In: *2017 IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR)*. 2017, pp. 216–222.

[4] Τιριτίρης Παύλος. ≪Βαθιά ενισχυτική μάθηση για ενεργή ρομποτική αντίληψη≫. Σχολή Θετικών Επιστημών, Τμήμα Πληροφορικής, Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης, 2020.

[5] Timothy P. Lillicrap and Jonathan J. Hunt and Alexander Pritzel and Nicolas Heess and Tom Erez and Yuval Tassa and David Silver and Daan Wierstra. ≪Continuous control with deep reinforcement learning≫. In: (Sept. 2015). URL: http://arxiv.org/abs/1509.02971.

[6] Keras Manual. *Deep Deterministic Policy Gradient (DDPG)*. July 2022. URL: https://keras.io/examples/rl/ddpg_pendulum/.

[7] Junior C Jesus and Jair A Bottega and Marco A S L Cuadros and Daniel F T Gamarra. *Deep Deterministic Policy Gradient for Navigation of Mobile Robots in Simulated Environments*. 2019. ISBN: 9781728124674.

[8] Hanlin Niu et al. ≪Accelerated Sim-to-Real Deep Reinforcement Learning: Learning Collision Avoidance from Human Player≫. In: *2021 IEEE/SICE International Symposium on System Integration (SII)*. 2021, pp. 144–149.

[9] G E Uhlenbeck and L S Ornstein. ≪On the theory of the brownian motion≫. In: (1930).

[10] Ιωάννης Ντίζος. ≪Σχεδιασμός τροχιάς εντρόχων ρομποτικών οχημάτων με χρήση ευρετικών αλγορίθμων≫. Διπλωματική Εργασία, Σχολή Μηχανικών Παραγωγής και Διοίκησης, Πολυτεχνείο Κρήτης, 2021.

[11] Πιπερίδης Σάββας και Δοϊτσίδης Ελευθέριος. ≪Σημειώσεις εργαστηρίου ρομποτικής≫. Σχολή Μηχανικών Παραγωγής και Διοίκησης, Πολυτεχνείο Κρήτης, 2021.

[12] National Ocean Service US. *What is lidar?* July 2022. URL: https://oceanservice.noaa.gov/facts/lidar.html.

[13] ROS. *Documentation*. July 2022. URL: https://docs.ros.org/.

[14] ROBOTIS. *Turtlebot 3 Manual*. July 2022. URL: https://emanual.robotis.com/docs/en/platform/turtlebot3/overview/#overview.

[15] Baijayanta Roy. *A-Star algorithm in Python*. July 2022. URL: https://github.com/BaijayantaRoy/Medium-Article/blob/master/A_Star.ipynb.