



ΔΗΜΟΚΡΙΤΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΡΑΚΗΣ
ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
& ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΕΝΕΡΓΕΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

**Ανάπτυξη συστήματος πλοήγησης ρομποτικής
πλατφόρμας μέσω τεχνικών Μηχανικής Μάθησης**

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

**Όνοματεπώνυμο φοιτητή, ΑΕΜ:
ΜΑΖΗΣ ΑΡΙΣΤΕΙΔΗΣ, ΑΕΜ:58222**

**Επιβλέπων Καθηγητής:
ΛΟΥΚΑΣ ΜΠΑΜΠΗΣ, ΕΠ. ΚΑΘΗΓΗΤΗΣ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ & ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ,
Δ.Π.Θ.**

ΞΑΝΘΗ, 2025

ΔΗΜΟΚΡΙΤΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΡΑΚΗΣ
ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ & ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΕΝΕΡΓΕΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

Ανάπτυξη συστήματος πλοήγησης ρομποτικής
πλατφόρμας μέσω τεχνικών Μηχανικής Μάθησης

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Όνοματεπώνυμο φοιτητή, ΑΕΜ:
ΜΑΖΗΣ ΑΡΙΣΤΕΙΔΗΣ, ΑΕΜ:58222

ΕΞΕΤΑΣΤΙΚΗ ΕΠΙΤΡΟΠΗ

Επιβλέπων καθηγητής: ΛΟΥΚΑΣ ΜΠΑΜΠΗΣ, ΕΠ. ΚΑΘΗΓΗΤΗΣ

2ο Μέλος: ΝΙΚΟΛΑΟΣ ΠΑΠΑΝΙΚΟΛΑΟΥ, ΚΑΘΗΓΗΤΗΣ

3ο Μέλος: ΗΛΙΑΣ ΚΟΣΜΑΤΟΠΟΥΛΟΣ, ΚΑΘΗΓΗΤΗΣ

ΞΑΝΘΗ, 2025



DEMOCRITUS UNIVERSITY OF THRACE
SCHOOL OF ENGINEERING
DEPARTMENT OF ELECTRICAL AND
COMPUTER ENGINEERING
POWER SYSTEMS SECTOR

Robot navigation through Machine Learning

DIPLOMA THESIS

Full name of the author, Registration Number:
MAZIS ARISTEIDIS, AEM:58222

COMMITTEE OF EXAMINERS

Supervisor: LOUKAS BAMPIS, ASSISTANT PROFESSOR

Member 2: NIKOLAOS PAPANIKOLAOU, PROFESSOR

Member 3: ELIAS KOSMATOPOULOS, PROFESSOR

XANTHI, 2025



ΔΗΜΟΚΡΙΤΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΡΑΚΗΣ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ & ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΕΝΕΡΓΕΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ
ΕΡΓΑΣΤΗΡΙΟ ΜΗΧΑΝΟΤΡΟΝΙΚΗΣ ΚΑΙ ΑΥΤΟΜΑΤΙΣΜΩΝ Η-Μ ΣΥΣΤΗΜΑΤΩΝ

Ανάπτυξη συστήματος πλοήγησης ρομποτικής πλατφόρμας μέσω τεχνικών Μηχανικής Μάθησης

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΑΡΙΣΤΕΙΔΗΣ ΜΑΖΗΣ

58222

aristeidismazis@gmail.com

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ

ΔΡ. ΛΟΥΚΑΣ ΜΠΑΜΠΗΣ
lbampis@ee.duth.gr

ΞΑΝΘΗ, 2025



DEMOCRITUS UNIVERSITY OF THRACE
DEPARTMENT OF ELECTRICAL & COMPUTER ENGINEERING
ENERGY SYSTEMS SECTOR
MECHATRONICS & SYSTEMS AUTOMATION LABORATORY

Robot navigation through Machine Learning

DIPLOMA THESIS

ARISTEIDIS MAZIS

58222

aristeidismazis@gmail.com

SUPERVISOR PROFESSOR

Dr. LOUKAS BAMPIS
 lbampis@ee.duth.gr

XANTHI, 2025

© Copyright Authorship: Aristeidis Mazis
© Copyright Scientific Subject: Loukas Bampis
All rights reserved.

The approval of the thesis by the Department of Electrical and Computer Engineering of the Democritus University of Thrace does not necessarily indicate acceptance of the author's views by the Department.

© Copyright Συγγραφής: Αριστείδης Μάζης
© Copyright Θέματος: Λουκάς Μπάμπης
Με επιφύλαξη παντός δικαιώματος.
Η έγκριση της διπλωματικής εργασίας από το Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του Δημοκρίτειου Πανεπιστημίου Θράκης δεν υποδηλώνει απαραιτήτως και αποδοχή των απόψεων του συγγραφέα εκ μέρους του Τμήματος.

“Ουδείς ων ράθυμος ευκλεής ανήρ,
άλλ’ οι πόνοι τίκτουσι την ευδοξίαν.”

- Ευριπίδης (480-406 π.Χ.)

ΕΥΧΑΡΙΣΤΙΕΣ

Με το πέρας της παρούσας Διπλωματικής Εργασίας, θα ήθελα να εκφράσω τις θερμές μου ευχαριστίες στον επιβλέποντα καθηγητή μου, κ. Λουκά Μπάμπη, για την πολύτιμη καθοδήγηση, τον χρόνο και την εμπιστοσύνη που μου έδειξε σε όλη τη διάρκεια της εργασίας.

Επιπλέον, θα ήθελα να εκφράσω την βαθιά μου ευγνωμοσύνη στους γονείς μου, Παναγιώτη και Πολυτίμη, και στον αδερφό μου Γιώργο, για τη διαρκή στήριξη, την υπομονή και την ενθάρρυνσή τους σε όλη την διάρκεια της ζωής μου. Η ανιδιοτελής αγάπη και η πίστη τους στις δυνατότητές μου, αποτελούν τον σημαντικότερο παράγοντα που κρύβεται πίσω από κάθε προσπάθεια και επιτυχία μου.

Τέλος, ευχαριστώ θερμά τους φίλους μου για τη συμπαράσταση και την πολύτιμη βοήθειά τους σε κάθε πρόκληση και εμπόδιο αυτής της διαδικασίας.

ACKNOWLEDGEMENTS

Upon completion of this thesis, I would like to express my sincere gratitude to my supervisor, Mr. Loukas Bampis, for his valuable guidance, time, and trust throughout the course of my work.

In addition, I would like to express my deep gratitude to my parents, Panagiotis and Polytimi, and my brother George, for their constant support, patience, and encouragement throughout my whole life. Their selfless love and belief in my abilities are the most important factors in every success I have ever achieved.

Finally, I would like to express my warm gratitude to my friends for their support and precious help in overcoming every challenge and obstacle along the way.

ΠΡΟΛΟΓΟΣ

Η ραγδαία πρόοδος της Μηχανικής Μάθησης τα τελευταία χρόνια έχει αναδιαμορφώσει την αντίληψη και την κατανόηση του περιβάλλοντος στην πλοϊγηση ρομποτικών πλατφορμών. Μοντέλα μάθησης από κλασικές μεθόδους μέχρι βαθιά νευρωνικά δίκτυα, εξάγουν αυτόματα χρήσιμα χαρακτηριστικά από ακατέργαστα σήματα αισθητήρων (LiDAR, κάμερες, IMU), βελτιώνοντας την αναγνώριση εμποδίων, την εκτίμηση καταστάσεων και τη χαρτογράφηση. Ωστόσο, τα συγκεκριμένα μοντέλα στοχεύουν κυρίως στην αντίληψη: προβλέπουν καταστάσεις ή βοηθούν στην αντίληψη ύπαρξης εμποδίων στον χώρο, αλλά δεν αποφασίζουν μόνα τους ακολουθίες ενεργειών σε πραγματικό χρόνο. Παράλληλα, οι κλασικοί αλγόριθμοι πλοϊγησης, παρά την αξιόπιστη λειτουργία τους σε καλά ορισμένα, στατικά σενάρια, εξαρτώνται σημαντικά από ακριβείς αναπαραστάσεις του χώρου και δυσκολεύονται να προσαρμοστούν σε δυναμικές συνθήκες ή σε ελλιπή γνώση του περιβάλλοντος. Αυτό το κενό δημιουργεί την ανάγκη για μεθόδους που μαθαίνουν συμπεριφορές και όχι απλώς κανόνες.

Σε αυτή την ανάγκη ανταποκρίνεται η Ενισχυτική Μάθηση και ειδικότερα η Βαθιά Ενισχυτική Μάθηση η οποία μαθαίνει πολιτικές πλοϊγησης μέσω αλληλεπίδρασης με το περιβάλλον χρησιμοποιώντας ένα σύστημα ανταμοιβής σύμφωνα με το οποίο βελτιώνει προοδευτικά τη συμπεριφορά του. Στην παρούσα διπλωματική εργασία υλοποιήθηκαν και αξιολογήθηκαν οι αλγόριθμοι Βαθιάς Ενισχυτικής Μάθησης DDPG και TD3 με σκοπό την επίτευξη αυτόνομης πλοϊγησης μίας ρομποτικής πλατφόρμας σε προσομοιωμένα αλλά και πραγματικά περιβάλλοντα. Στο Κεφάλαιο 1 πραγματοποιείται μία πρώτη εισαγωγή στον σκοπό και την σημασία της παρούσας διπλωματικής. Το Κεφάλαιο 2 αφορά την ιστορική αναδρομή και θεωρητικό υπόβαθρο της πλοϊγησης ρομποτικών πλατφορμών. Το Κεφάλαιο 3 εισάγει τις βασικές έννοιες της Μηχανικής και Ενισχυτικής Μάθησης και αναδεικνύει τις εφαρμογές της Βαθιάς Ενισχυτικής Μάθησης στην ρομποτική πλοϊγηση. Στο Κεφάλαιο 4 περιγράφεται το πείραμα εφαρμογής των αλγορίθμων σε προσομοιωμένα περιβάλλοντα. Στο Κεφάλαιο 5 καταγράφονται και αναλύονται τα πειραματικά αποτελέσματα. Στο Κεφάλαιο 6 οι αλγόριθμοι εφαρμόζονται σε πραγματικό ρομπότ και αξιολογείται η απόδοσή τους. Στο Κεφάλαιο 7 συνοψίζονται τα προαναφερθέντα και εξάγονται συμπεράσματα, καθώς επίσης αναφέρονται προτάσεις για βελτιώσεις και μελλοντικές έρευνες.

ΑΡΙΣΤΕΙΔΗΣ ΜΑΖΗΣ

Ξάνθη, Νοέμβριος 2025

PROLOGUE

The rapid advancement of machine learning in recent years has reshaped the perception and understanding of the environment in robotic platform navigation. Learning models, ranging from classical methods to deep neural networks, automatically extract useful features from raw sensor signals (LiDAR, cameras, IMU), improving obstacle recognition, situation assessment, and mapping. However, these models mainly focus on perception: they predict situations or aid in the detection of obstacles in space, but they do not decide on sequences of actions in real time on their own. At the same time, classic navigation algorithms, despite their reliable operation in well-defined, static scenarios, depend heavily on accurate representations of space and have difficulty adapting to dynamic conditions or incomplete knowledge of the environment. This gap creates a need for methods that learn behaviors rather than just rules.

Reinforcement Learning and, in particular, Deep Reinforcement Learning, which learns navigation policies through interaction with the environment, using a reward system that progressively improves its behavior, aims to eliminate this gap. In this thesis, two Deep Reinforcement Learning algorithms, DDPG and TD3, were implemented and evaluated with respect to achieve an autonomous navigation system of a robotic platform in simulated and real environments. Chapter 1 provides an initial introduction to the purpose and significance of this thesis. Chapter 2 covers the historical background and theoretical basis of robotic platform navigation. Chapter 3 introduces the basic concepts of Machine Learning and Reinforcement Learning and highlights the applications of Deep Reinforcement Learning in robot navigation. Chapter 4 describes the experiment of implementing the algorithms in simulated environments. Chapter 5 records and analyzes the results of the experiment. In Chapter 6, the algorithms are implemented on a real robot and their performance is evaluated. Chapter 7 summarizes the above and draws conclusions, as well as suggesting improvements and future research directions.

ARISTEIDIS MAZIS

Xanthi, November 2025

ΠΕΡΙΛΗΨΗ

Η Βαθιά Μάθηση έχει γνωρίσει ραγδαία άνοδο τα τελευταία χρόνια στον τομέα της ρομποτικής, επιτρέποντας στις ρομποτικές πλατφόρμες να αντιλαμβάνονται και να αντιδρούν στο περιβάλλον το οποίο βρίσκονται με ακρίβεια και ευχέρεια, βελτιώνοντας την αποδοτικότητα και την αυτονομία τους. Η Βαθιά Ενισχυτική Μάθηση έχει σημαντική συνεισφορά στην πλοήγηση ρομποτικών πλατφορμών, καθώς επιτρέπει σε αυτές να βελτιώνουν την απόδοσή τους μέσω της αλληλεπίδρασής τους με το περιβάλλον. Αντί να απαιτούνται προκαθορισμένα δεδομένα ή σαφείς εντολές, τα ρομπότ μαθαίνουν να λαμβάνουν αποφάσεις με βάση την παρατήρηση των συνεπειών των ενεργειών τους. Με την ενίσχυση μέσω ανταμοιβών ή ποινών, οι ρομποτικές πλατφόρμες προσαρμόζονται και βελτιστοποιούν τις στρατηγικές πλοήγησης σε δυναμικά και άγνωστα περιβάλλοντα, καθιστώντας τις πιο ικανές στην αντιμετώπιση αβέβαιων ή μεταβαλλόμενων συνθηκών.

Η παρούσα διπλωματική εξετάζει την ανάπτυξη συστήματος πλοήγησης για ρομποτικές πλατφόρμες μέσω αλγορίθμων Βαθιάς Ενισχυτικής Μάθησης. Πιο συγκεκριμένα γίνεται χρήση δύο αλγορίθμων, του αλγορίθμου Deep Deterministic Policy Gradient (DDPG) και του αλγορίθμου Twin Delayed Deep Deterministic Policy Gradient (TD3), για την ανάπτυξη συστήματος πλοήγησης επίγειας ρομποτικής πλατφόρμας σε άγνωστα περιβάλλοντα προσομοίωσης. Οι αλγόριθμοι αυτοί αναλύονται, εκπαιδεύονται και συγκρίνονται σε διάφορα περιβάλλοντα, ώστε να αξιολογηθεί η αποδοτικότητά τους στην επίλυση προβλημάτων πλοήγησης και αναγνώρισης. Μέσω της εκπαίδευσης και της εφαρμογής των αλγορίθμων σε προσομοιωμένα περιβάλλοντα, γίνεται ανάλυση της απόδοσης τους σε όρους ακριβείας, ταχύτητας και αξιοπιστίας.

Λέξεις κλειδιά: Βαθιά Ενισχυτική Μάθηση, Πλοήγηση Ρομποτικής Πλατφόρμας, Τεχνητή Νοημοσύνη, Πλοήγηση σε Άγνωστο Περιβάλλον, Μηχανική Μάθηση, Βαθιά Μάθηση, Ενισχυτική Μάθηση, DDPG Αλγόριθμος, TD3 Αλγόριθμος

ABSTRACT

In recent years, Deep Learning has seen rapid advancements in the field of robotics, allowing robotic platforms to perceive and react with their environment with precision and agility, improving their efficiency and autonomy. Deep Reinforcement Learning has made a significant contribution in navigation of robotic platforms, as it allows them to improve their performance through their interaction with the environment. Instead of requiring predefined data or explicit commands, robots learn to make decisions by observing the consequences of their actions. Reinforced through rewards or penalties, robotic platforms adapt and optimize navigation strategies in dynamic and unknown environments, making them more capable of dealing with uncertain or changing conditions.

This thesis aims to examine the development of a navigation system for robotic platforms using Deep Reinforcement Learning algorithms. More specifically, two algorithms, Deep Deterministic Policy Gradient (DDPG) algorithm and Twin Delayed Deep Deterministic Policy Gradient (TD3) algorithm, were used to develop a ground-based robotic platform navigation system in unknown simulation environments. These algorithms were analyzed, trained and compared in different environments, to evaluate their efficiency in solving navigation and recognition problems. Through training and application of the algorithms in simulated environments, their performance was analyzed in terms of accuracy, speed and reliability.

Keywords: Deep Reinforcement Learning, Robot Navigation, Artificial Intelligence, Navigation in Unknown Environments, Machine Learning, Deep Learning, Reinforcement Learning, DDPG Algorithm, TD3 Algorithm

ΥΠΕΥΘΥΝΗ ΔΗΛΩΣΗ

Βεβαιώνω ότι είμαι συγγραφέας της παρούσας εργασίας και ότι έχω αναφέρει ή παραπέμψει σε αυτή, ρητά και συγκεκριμένα, όλες τις πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών, προτάσεων ή λέξεων, είτε αυτές μεταφέρονται επακριβώς (στο πρωτότυπο ή μεταφρασμένες) είτε παραφρασμένες. Επίσης βεβαιώνω ότι αυτή η εργασία προετοιμάστηκε από εμένα προσωπικά ειδικά για την συγκεκριμένη έρευνα, στο πλαίσιο εκπόνησης της διπλωματικής μου εργασίας.

STATEMENT OF RESPONSIBILITY

I certify that I am the author of this work and that I have expressly and specifically cited or referred to in it all sources from which I have used data, ideas, sentences or words, whether they are accurately conveyed (in the original or translated) or paraphrased. I also certify that this paper was prepared by me personally specifically for this research, in the context of my diploma thesis.

ΠΕΡΙΕΧΟΜΕΝΑ

ΕΥΧΑΡΙΣΤΙΕΣ	I
ACKNOWLEDGEMENTS	II
ΠΡΟΛΟΓΟΣ	III
PROLOGUE	IV
ΠΕΡΙΛΗΨΗ	V
ABSTRACT	VI
ΥΠΕΥΘΥΝΗ ΔΗΛΩΣΗ	VII
STATEMENT OF RESPONSIBILITY	VIII
ΠΕΡΙΕΧΟΜΕΝΑ	IX
1 Εισαγωγή	1
1.1 Αντικείμενο της Διπλωματικής	1
1.2 Στόχοι και Συνεισφορά	2
1.3 Δομή Κειμένου	2
2 Ιστορική Αναδρομή και Θεωρητικό Υπόβαθρο	5
2.1 Τα πρώτα αυτόνομα ρομπότ και οι βασικές αρχές	5
2.1.1 Σχεδιασμός Διαδρομής	6
2.2 Η εξέλιξη της ρομποτικής πλοήγησης (1980-2000)	7
2.3 Η Άνοδος της Τεχνητής Νοημοσύνης (2000-σήμερα)	9
2.4 Αισθητήρες στην Πλοήγηση	10
2.4.1 Αισθητήρες Αντίληψης Περιβάλλοντος	10
2.4.2 Αισθητήρες Προσδιορισμού Θέσης και Κίνησης	11
2.4.3 Συνδυασμός Αισθητήρων για Ακριβέστερη Πλοήγηση	12
3 Μηχανική Μάθηση	14
3.1 Ορισμός και Θεμελιώδη Χαρακτηριστικά	14
3.1.1 Τύποι Μηχανικής Μάθησης	14
3.1.2 Βαθιά Μάθηση	15
3.1.3 Νευρωνικά Δίκτυα	15
3.1.4 Συνάρτηση Κόστους	17
3.1.5 Συνάρτηση Ενεργοποίησης	19
3.1.6 Αλγόριθμοι Βελτιστοποίησης	20
3.1.7 Αλγόριθμος Backpropagation	22
3.1.8 Υπερπαράμετροι	23

3.1.9	Υπερπροσαρμογή/Υποπροσαρμογή	24
3.1.10	Ρύθμιση Υπερπαραμέτρων	24
3.2	Ενισχυτική Μάθηση	25
3.2.1	Τύποι Εργασιών στην Ενισχυτική Μάθηση	27
3.2.2	Το Δίλημμα Εξερεύνησης/Εκμετάλλευσης.....	28
3.2.3	Μέθοδοι Επίλυσης Προβλημάτων Ενισχυτικής Μάθησης	28
3.2.4	Online/Offline Εκμάθηση	31
3.2.5	Monte Carlo vs Temporal Difference Learning	32
3.2.6	Αλγόριθμος Q-Learning	32
3.3	Βαθιά Ενισχυτική Μάθηση.....	35
3.3.1	DRL στην πλοιήγηση ρομποτικών πλατφορμών	36
3.3.2	Αρχιτεκτονικές και μοντέλα DRL	36
3.3.2.1	Deep Q-Network.....	36
3.3.2.2	Deep Deterministic Policy Gradient	38
3.3.2.3	Twin Delayed Deep Deterministic Policy Gradient.....	40
3.3.2.4	Proximal Policy Optimization	43
4	Ανάλυση Πειράματος σε περιβάλλον Προσομοίωσης	46
4.1	Περιγραφή του Συστήματος.....	46
4.1.1	Το ρομπότ	46
4.1.2	Το περιβάλλον	48
4.1.3	Ο αλγόριθμος DRL	50
4.1.4	ROS	51
4.2	Εκπαίδευση Αλγορίθμων	52
5	Πειραματικά Αποτελέσματα	61
5.1	Αποτελέσματα Αλγορίθμου DDPG	61
5.2	Αποτελέσματα Αλγορίθμου TD3	68
5.3	Σύγκριση Μεθόδων	73
5.3.1	Ποσοστό Επιτυχίας και Ταχύτητα Σύγκλησης	73
5.3.2	Ρύθμιση Παραμέτρων	74
6	Εφαρμογή σε πραγματικό ρομπότ.....	77
6.1	Περιγραφή του Συστήματος.....	77
6.2	Εφαρμογή αλγορίθμων	82
7	Συμπεράσματα και Μελλοντικές Προεκτάσεις.....	89
7.1	Ανακεφαλαίωση των βασικών στοιχείων	89
7.2	Προτάσεις για μελλοντική έρευνα	90
	ΕΥΡΕΤΗΡΙΟ	93

ΒΙΒΛΙΟΓΡΑΦΙΑ	95
ΠΑΡΑΡΤΗΜΑ - ΚΩΔΙΚΑΣ	97

1 Εισαγωγή

1.1 Αντικείμενο της Διπλωματικής

Η πλοϊγηση ρομποτικών πλατφορμών αποτελεί έναν από τους πιο κρίσιμους τομείς της ρομποτικής, καθώς σχετίζεται άμεσα με την ικανότητα ενός ρομπότ να κινείται αυτόνομα σε άγνωστα περιβάλλοντα. Οι συμβατικοί-κλασικοί αλγόριθμοι πλοϊγησης βασίζονται συχνά σε προκαθορισμένα μοντέλα και κανόνες που απαιτούν εκ των προτέρων πλήρη γνώση του περιβάλλοντος ή της διαδρομής. Αν και είναι αποτελεσματικοί σε ελεγχόμενα και γνωστά περιβάλλοντα, συχνά αντιμετωπίζουν περιορισμούς όταν καλούνται να λειτουργήσουν σε άγνωστα και δυναμικά περιβάλλοντα. Οι κλασικοί αλγόριθμοι συχνά δυσκολεύονται να προσαρμοστούν σε αλλαγές στο περιβάλλον, όπως την κίνηση άλλων αντικειμένων ή την εμφάνιση νέων εμποδίων, και δεν μπορούν να μάθουν ή να βελτιώνονται δυναμικά μέσω εμπειρίας. Επιπλέον, απαιτούν τον καθορισμό μιας πλήρους και ακριβούς αναπαράστασης του περιβάλλοντος για να είναι αποτελεσματικοί. Αυτό μπορεί να είναι δύσκολο και χρονοβόρο, ειδικά όταν το περιβάλλον είναι μεγάλου μεγέθους ή μεταβαλλόμενο σε πραγματικό χρόνο. Παράλληλα, η διαδικασία της κατασκευής και διαχείρισης αυτών των αναπαραστάσεων απαιτεί σημαντικούς υπολογιστικούς πόρους, οι οποίοι μπορεί να περιορίσουν την αποτελεσματικότητα του συστήματος σε περιβάλλοντα με περιορισμένη υπολογιστική ισχύ ή σε πραγματικό χρόνο.

Αυτοί οι περιορισμοί καταδεικνύουν τη σημασία της ανάπτυξης νέων, πιο ευέλικτων μεθόδων πλοϊγησης που να βασίζονται σε δυναμική μάθηση και προσαρμογή. Αυτή την ανάγκη ήρθε να καλύψει η ανάπτυξη αλγορίθμων Βαθιάς Ενισχυτικής Μάθησης.

Η συγκεκριμένη διπλωματική εργασία εστιάζει στην ανάπτυξη και εφαρμογή τεχνικών Βαθιάς Ενισχυτικής Μάθησης για την πλοϊγηση αυτόνομων ρομποτικών πλατφορμών. Οι μέθοδοι Ενισχυτικής Μάθησης επιτρέπουν στα ρομπότ να μαθαίνουν βέλτιστες στρατηγικές κίνησης μέσα από αλληλεπίδραση με το περιβάλλον, χωρίς να απαιτείται χειροκίνητη προγραμματιστική παρέμβαση. Αντί να χρησιμοποιούν στατικά προκαθορισμένους κανόνες, τα συστήματα βασισμένα στην Ενισχυτική Μάθηση αξιοποιούν την εμπειρία που αποκτούν μέσω συνεχών δοκιμών και ενός συστήματος ανταμοιβών, βελτιστοποιώντας έτσι την πλοϊγηση του ρομπότ.

Η χρήση της Βαθιάς Ενισχυτικής Μάθησης, το οποίο συνδυάζει βαθιά νευρωνικά δίκτυα με μεθόδους Ενισχυτικής Μάθησης, επιτρέπει την αντιμετώπιση πιο περίπλοκων προβλημάτων πλοϊγησης, καθώς βοηθά στην εξαγωγή χαρακτηριστικών από δεδομένα αισθητήρων, όπως εικόνες από κάμερες ή σαρώσεις LiDAR.

1.2 Στόχοι και Συνεισφορά

Βασικός στόχος της παρούσας διπλωματικής εργασίας είναι η διερεύνηση της αποτελεσματικότητας τεχνικών Ενισχυτικής Μάθησης στην πλοϊγηση ρομποτικής πλατφόρμας. Συγκεκριμένα, επικεντρώνεται στην ανάλυση, εφαρμογή και συγκριτική αξιολόγηση των αλγορίθμων Deep Deterministic Policy Gradient (DDPG) και Twin Delayed Deep Deterministic Policy Gradient (TD3), οι οποίοι είναι σχεδιασμένοι για περιβάλλοντα με συνεχείς χώρους κατάστασης και δράσης.

Ένας από τους κεντρικούς στόχους της εργασίας είναι η κατανόηση του τρόπου με τον οποίο αυτοί οι δύο αλγόριθμοι μπορούν να μάθουν αποδοτικές στρατηγικές πλοϊγησης σε προσομοιωμένα περιβάλλοντα, μέσω επαναλαμβανόμενης αλληλεπίδρασης με το περιβάλλον και αξιολόγησης μέσω κατάλληλης συνάρτησης ανταμοιβής. Οι αλγόριθμοι εφαρμόστηκαν σε προσομοιωμένα περιβάλλοντα αυξανόμενης πολυπλοκότητας, τα οποία διέφεραν τόσο ως προς τη διαρρύθμιση του χώρου όσο και ως προς τον αριθμό των δυναμικών εμποδίων. Η δυσκολία κάθε περιβάλλοντος σχεδιάστηκε ώστε να ελέγχεται προοδευτικά η προσαρμοστικότητα και η ικανότητα γενίκευσης των αλγορίθμων, επιτρέποντας τη δοκιμή τους σε σενάρια που πλησιάζουν πραγματικές συνθήκες. Η μελέτη αποσκοπεί επίσης στο να καταδείξει τις διαφορές στην απόδοση μεταξύ των δύο αλγορίθμων, ως προς την ταχύτητα εκμάθησης, τη σταθερότητα, τη δυνατότητα γενίκευσης και τη διαχείριση πολύπλοκων ή απρόβλεπτων καταστάσεων.

Η συνεισφορά της παρούσας διπλωματικής επικεντρώνεται στην αξιολόγηση των αλγορίθμων DDPG και TD3, παρέχοντας συγκριτικά αποτελέσματα που μπορούν να χρησιμοποιηθούν ως βάση για μελλοντικές βελτιώσεις ή συνδυασμούς αλγορίθμων. Τα αποτελέσματα αυτής της έρευνας αναμένεται να συμβάλουν και στην κατανόηση της αποτελεσματικότητας της Ενισχυτικής Μάθησης στην ρομποτική πλοϊγηση, καθώς και στην ανάπτυξη πιο αποδοτικών και ευφυέστερων συστημάτων, που θα μπορούν να εφαρμόζονται σε πραγματικές συνθήκες.

1.3 Δομή Κειμένου

Στο **Κεφάλαιο 2 (Ιστορική Αναδρομή και Θεωρητικό Υπόβαθρο)** παρουσιάζεται το πλαίσιο της ρομποτικής πλοϊγησης: από τα πρώτα αυτόνομα ρομπότ και τις βασικές αρχές σχεδιασμού διαδρομής, έως την εξέλιξη των μεθόδων τη περίοδο 1980–2000. Ακολουθεί η άνοδος της Τεχνητής Νοημοσύνης από το 2000 μέχρι σήμερα, καθώς επίσης και μια επισκόπηση των αισθητήρων πλοϊγησης.

Στο **Κεφάλαιο 3 (Μηχανική Μάθηση)** εισάγονται οι βασικές έννοιες της Μηχανικής Μάθησης: ορισμοί, τύποι μάθησης, καθώς και τα δομικά στοιχεία της Βαθιάς Μάθησης, Νευρωνικά Δίκτυα, συναρτήσεις κόστους και ενεργοποίησης, αλγόριθμοι βελτιστοποίησης και backpropagation, υπερπαράμετροι και υπερ/υποπροσαρμογή. Η ενότητα συνεχίζει με ανάλυση του πεδίου Ενισχυτικής Μάθησης και ολοκληρώνεται με ανάπτυξη επί της Βαθιάς Ενισχυτικής Μάθησης, παρουσιάζοντας εφαρμογές της στην πλοϊγηση ρομποτικών πλατφορμών και αντιπροσωπευτικές αρχιτεκτονικές/μοντέλα.

Στο Κεφάλαιο 4 (Ανάλυση Πειράματος σε Προσομοίωση) περιγράφεται το πείραμα προσομοίωσης. Το υπό μελέτη σύστημα αποτελείται από: το ρομπότ, το περιβάλλον προσομοίωσης και τον αλγόριθμο Βαθιάς Ενισχυτικής Μάθησης. Καθορίζονται τα πειραματικά σενάρια, οι παράμετροι εκπαίδευσης και τα πρωτόκολλα αξιολόγησης, έτσι ώστε ο αναγνώστης να έχει πλήρη εικόνα για την αναπαραγωγή των εκπαιδεύσεων.

Στο Κεφάλαιο 5 (Πειραματικά Αποτελέσματα) παρουσιάζονται αναλυτικά τα ευρήματα για τον DDPG (ποσοστά επιτυχίας, καμπύλες μάθησης και γενικότερες παρατηρήσεις) και τον TD3 (αντίστοιχη ανάλυση). Ακολουθεί μία σύγκριση μεθόδων, όπου εξετάζονται συγκριτικά το ποσοστό επιτυχίας, η ταχύτητα σύγκλισης, οι επιδράσεις της ρύθμισης παραμέτρων και η γενικότερη συμπεριφορά των αλγορίθμων στα περιβάλλοντα προσομοίωσης.

Στο Κεφάλαιο 6 (Εφαρμογή σε πραγματικό ρομπότ) μεταφέρεται η μεθοδολογία από την προσομοίωση στον πραγματικό κόσμο. Παρουσιάζεται η υλικοτεχνική διάταξη, το λογισμικό, οι απαιτούμενες προσαρμογές (π.χ. αισθητήρες, επικοινωνία, περιορισμοί) και τα αποτελέσματα από την εφαρμογή των αλγορίθμων στο πραγματικό ρομπότ, αναδεικνύοντας την απόδοση και τα προβλήματα των αλγορίθμων κατά την μετάβαση.

Στο Κεφάλαιο 7 (Συμπεράσματα και Μελλοντικές Προεκτάσεις) συνοψίζονται τα βασικά συμπεράσματα της μελέτης ως προς την αποτελεσματικότητα των προσεγγίσεων Βαθιάς Ενισχυτικής Μάθησης στην ρομποτική πλοιόγηση. Έπειτα παρουσιάζονται προτάσεις για μελλοντική έρευνα, όπως βελτιώσεις σε αλγορίθμους και σε ρυθμίσεις υπερπαραμέτρων, ρεαλιστικότερα περιβάλλοντα προσομοίωσης, εμπλουτισμός αισθητήριων δεδομένων και εντατικότερη διεύρυνση εφαρμογών σε πραγματικά περιβάλλοντα.

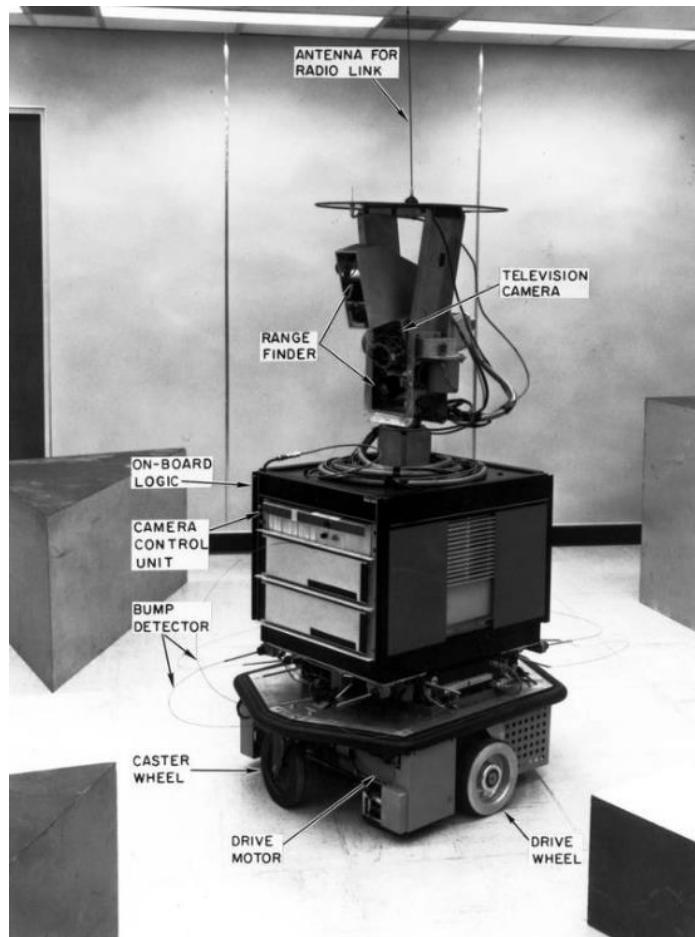
2 Ιστορική Αναδρομή και Θεωρητικό Υπόβαθρο

Η αυτόματη πλοιόγηση ρομποτικών πλατφορμών αποτελεί ένα βασικό πεδίο μελέτης στη ρομποτική, καθώς επιτρέπει στα ρομπότ να κινούνται αυτόνομα μέσα σε ένα περιβάλλον, να αποφεύγουν εμπόδια και να φτάνουν σε προκαθορισμένους στόχους. Η ανάγκη για ασφαλή, αποτελεσματική και έξυπνη πλοιόγηση έχει οδηγήσει στην ανάπτυξη μιας πληθώρας θεωρητικών προσεγγίσεων και τεχνολογιών. Από τις πρώτες προσπάθειες κινητικότητας μέσω απλών αισθητήρων μέχρι τη χρήση προηγμένων αλγορίθμων τεχνητής νοημοσύνης, το πεδίο έχει εξελιχθεί σημαντικά.

2.1 Τα πρώτα αυτόνομα ρομπότ και οι βασικές αρχές

Η ιστορική πορεία των αυτόνομων ρομπότ ξεκίνησε δυναμικά στα τέλη της δεκαετίας του 1960, σηματοδοτώντας τη γέννηση ενός νέου ερευνητικού πεδίου που συνδύαζε στοιχεία από τη ρομποτική και τα συστήματα ελέγχου. Ένα από τα πιο εμβληματικά παραδείγματα αυτής της πρώιμης περιόδου ήταν το Shakey the Robot (Εικόνα 1), το οποίο αναπτύχθηκε στο SRI International μεταξύ 1966 και 1972 από τους Charles Rosen και Nils J. Nilsson [1]. Σε αντίθεση με τα απλά τηλεκατευθυνόμενα ρομπότ της εποχής, το Shakey ήταν το πρώτο που μπορούσε να λάβει αυτόνομες αποφάσεις βάσει περιβαλλοντικών δεδομένων, να σχεδιάσει κινήσεις και να εκτελέσει εντολές χωρίς άμεση ανθρώπινη παρέμβαση. Το Shakey βασιζόταν σε έναν συνδυασμό λογικού προγραμματισμού και μηχανισμών αντίληψης, επιτρέποντας την κατασκευή ενός χάρτη του περιβάλλοντος μέσω επεξεργασίας εικόνας και δεδομένων από αισθητήρες απόστασης. Ο σχεδιασμός των ενεργειών του βασιζόταν στο σύστημα STRIPS (Stanford Research Institute Problem Solver) [2], το οποίο αποτελεί μέχρι και σήμερα σημείο αναφοράς στη συμβολική αναπαράσταση χώρου και στον σχεδιασμό ενεργειών [3].

Τα πρώτα αυτόνομα ρομπότ λειτουργούσαν κυρίως σε εργαστηριακά ή ελεγχόμενα περιβάλλοντα, όπου μπορούσαν να εκτελέσουν αποστολές πλοιόγησης, όπως μετακίνηση από σημείο σε σημείο και αποφυγή εμποδίων. Σε αυτά τα πρώιμα συστήματα καθιερώθηκαν οι βασικές αρχές της αυτόνομης πλοιόγησης: αντίληψη του περιβάλλοντος μέσω αισθητήρων, αναπαράσταση του χώρου (mapping), τοποθεσία (localization), σχεδιασμός διαδρομής και έλεγχος κινήσεων [3], [4].



Εικόνα 1: *Shakey the Robot*

2.1.1 Σχεδιασμός Διαδρομής

Ο σχεδιασμός διαδρομής (path planning) αποτελεί μία από τις θεμελιώδης αρχές της ρομποτικής πλοήγησης. Αφορά τον υπολογισμό μιας βέλτιστης και αποδεκτής τροχιάς που επιτρέπει σε ένα ρομπότ να κινηθεί από ένα σημείο εκκίνησης προς έναν προκαθορισμένο στόχο, αποφεύγοντας παράλληλα στατικά-δυναμικά εμπόδια και περιορισμούς του περιβάλλοντος [5].

Το πρόβλημα χωρίζεται θεωρητικά σε δύο επίπεδα:

- **Παγκόσμιος Σχεδιασμός (Global Planning):** Απαιτεί πλήρη γνώση του περιβάλλοντος και σχεδιάζει ολόκληρη τη διαδρομή εκ των προτέρων.
- **Τοπικός Σχεδιασμός (Local Planning):** Αντιμετωπίζει δυναμικά περιβάλλοντα, παράγοντας δράσεις σε πραγματικό χρόνο με βάση δεδομένα από αισθητήρες.

Οι παραδοσιακές μέθοδοι βασίζονται σε αλγορίθμικές τεχνικές, οι οποίες προσεγγίζουν το πρόβλημα από οπτική λογικού σχεδιασμού. Οι πιο χαρακτηριστικοί αλγόριθμοι διακρίνονται σε αλγορίθμους αναζήτησης, αλγορίθμους πιθανοκρατικής

δειγματοληψίας και αλγορίθμους δυναμικού πεδίου. Μερικοί από τους πιο διαδεδομένους αλγορίθμους path planning είναι οι εξής:

Dijkstra

Η μέθοδος του Dijkstra αποτελεί έναν από τους πρώτους αλγορίθμους εύρεσης μικρότερης απόστασης σε γραφήματα. Παρουσιάζει αυξημένο υπολογιστικό κόστος σε μεγάλα περιβάλλοντα λόγω του ότι εξετάζει όλες τις δυνατές διαδρομές με βάση το συνολικό κόστος.

A*

Ο αλγόριθμος A* είναι ένας από τους πιο διαδεδομένους αλγορίθμους αναζήτησης βέλτιστης διαδρομής σε grid-based περιβάλλοντα. Χρησιμοποιεί μία ευρετική (heuristic) συνάρτηση κόστους $f(n) = g(n) + h(n)$, όπου $g(n)$ είναι το κόστος από την αρχή μέχρι τον κόμβο n και $h(n)$ η εκτιμώμενη απόσταση από τον n ως τον στόχο [6].

RRT (Rapidly-exploring Random Tree)

Ο RRT είναι ένας δειγματοληπτικός αλγόριθμος που δημιουργεί ένα δέντρο καταστάσεων μέσω τυχαίας δειγματοληψίας του χώρου κατάστασης. Είναι κατάλληλος για προβλήματα πολλών διαστάσεων με σύνθετα κινηματικά μοντέλα, όπως drones και αυτόνομα οχήματα. Δεν εγγυάται βελτιστοποίηση, αλλά επιλύει προβλήματα γρήγορα [7].

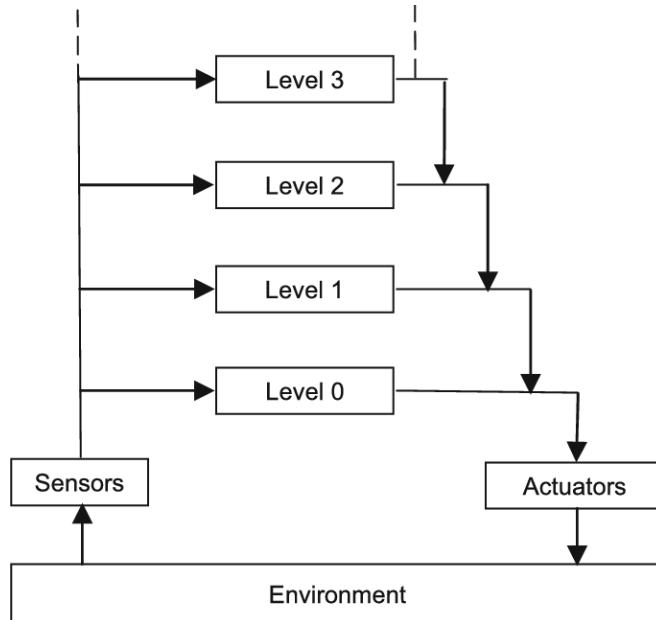
RRT*

Παραλλαγή του RRT που εγγυάται ασύμπτωτη βελτιστοποίηση. Χρησιμοποιείται για ασφαλέστερη και πιο αποδοτική πλοιόγηση. Καθώς αυξάνονται τα δείγματα, το δέντρο συγκλίνει προς τη βέλτιστη διαδρομή με πιθανότητα 1. Εξαιρετικά χρήσιμος σε path planning με χρονικούς ή υπολογιστικούς περιορισμούς.

2.2 Η εξέλιξη της ρομποτικής πλοιόγησης (1980-2000)

Η δεκαετία του 1980 αποτέλεσε σημείο καμπής για την πλοιόγηση αυτόνομων ρομπότ, με την επιστημονική κοινότητα να μετατοπίζει το ενδιαφέρον σε πιο ευέλικτες και ρεαλιστικές αρχιτεκτονικές ελέγχου. Το πιο χαρακτηριστικό παράδειγμα αυτής της αλλαγής είναι η αρχιτεκτονική Subsumption (Εικόνα 2) του Rodney Brooks, που προτάθηκε το 1986 ως εναλλακτική στο παραδοσιακό μοντέλο sense-plan-act [8]. Το ρομπότ Allen, που αναπτύχθηκε από τον Rodney Brooks στο MIT στα μέσα της δεκαετίας του 1980, αποτελεί μία από τις πρώτες υλοποιήσεις της αρχιτεκτονικής Subsumption. Αντί να βασίζεται στην αναπαράσταση του περιβάλλοντος και στον σχεδιασμό πορείας υψηλού επιπέδου, το Allen χρησιμοποιούσε τρία ιεραρχημένα επίπεδα χειρισμού, τα οποία ενεργοποιούνταν από ερεθίσματα αισθητήρων.

Η νέα αυτή προσέγγιση υποστήριζε την παράλληλη εκτέλεση διεργασιών, χωρίς την ανάγκη για σχεδιασμό πορείας ή πλήρη γνώση του περιβάλλοντος. Αυτή η φιλοσοφία έθεσε τις βάσεις για τα behavior-based ρομπότ, τα οποία έδειξαν ανθεκτικότητα και λειτουργικότητα σε άγνωστα και δυναμικά περιβάλλοντα [9].



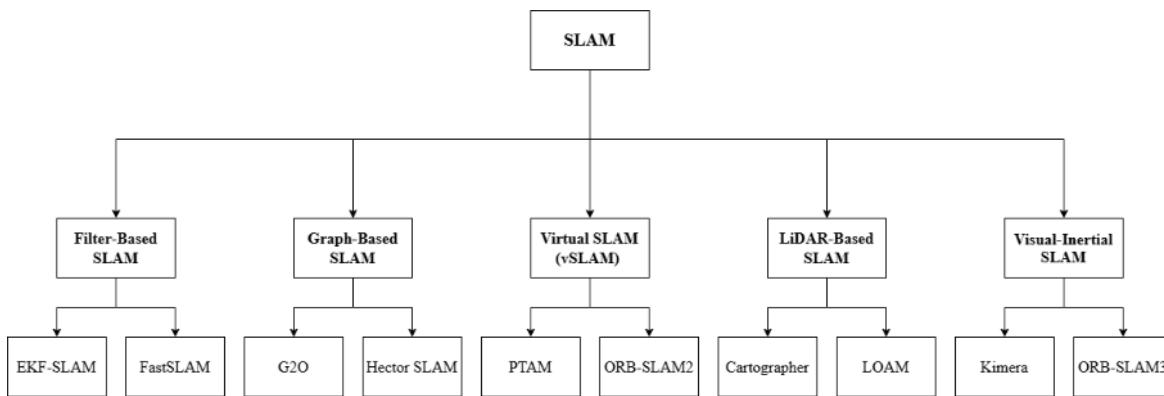
Εικόνα 2: Αρχιτεκτονική Subsumption

Ταυτόχρονος εντοπισμός και χαρτογράφηση (Simultaneous Localization and Mapping - SLAM)

Κατά τη δεκαετία του 1990 παρουσιάστηκε μία νέα προσέγγιση, η οποία αφορούσε την παράλληλη χαρτογράφηση και εντοπισμό θέσης σε πραγματικό χρόνο. Ο όρος SLAM άρχισε να διαμορφώνεται στα μέσα της δεκαετίας, με τις πρώτες ουσιαστικές υλοποιήσεις να εμφανίζονται σε εφαρμογές όπου το ρομπότ χρειαζόταν να κατασκευάσει έναν χάρτη του περιβάλλοντος, ενώ ταυτόχρονα εκτιμούσε τη θέση του μέσα σε αυτόν [10]. Ο αλγόριθμος SLAM επιτρέπει στα ρομπότ να δημιουργούν δυναμικά χάρτες του περιβάλλοντός τους ενώ ταυτόχρονα εντοπίζουν τη θέση τους σε αυτόν, χωρίς να χρειάζονται προ-προγραμματισμένα δεδομένα. Οι αλγόριθμοι SLAM χωρίζονται σε κατηγορίες ανάλογα με την μεθοδολογία που ακολουθούν για την εκτίμηση θέσης και χαρτογράφησης (Εικόνα 3).

- Filter-Based SLAM: Αλγόριθμοι οι οποίοι χρησιμοποιούν στοχαστικά φίλτρα (π.χ. Kalman ή Particle Filters) για εκτίμηση της θέσης και χαρτογράφησης (EKF-SLAM, FastSLAM).
- Graph-Based SLAM: Αλγόριθμοι οι οποίοι μοντελοποιούν το πρόβλημα SLAM ως γράφημα και το επιλύουν μέσω βελτιστοποίησης (G2O, Hector SLAM).

- Virtual SLAM (vSLAM): Αλγόριθμοι οι οποίοι αξιοποιούν κάμερες (monocular, stereo, RGB-D) για την εκτίμηση θέσης και χαρτογράφησης (PTAM, ORB-SLAM2).
- LiDAR-Based SLAM: Αλγόριθμοι που βασίζονται κυρίως σε σάρωση από LiDAR για την εκτίμηση θέσης και χαρτογράφηση (Cartographer, LOAM).
- Visual-Inertial SLAM: Αλγόριθμοι οι οποίοι συνδυάζουν κάμερες με IMU για καλύτερη εκτίμηση θέσης, ιδιαίτερα χρήσιμο σε δυναμικά περιβάλλοντα (Kimera, VINS-Fusion, ORB-SLAM3).



Εικόνα 3: Διάγραμμα Αλγορίθμων Πλοήγησης

2.3 Η Άνοδος της Τεχνητής Νοημοσύνης (2000-σήμερα)

Από τις αρχές της δεκαετίας του 2000, η πρόοδος στην Τεχνητή Νοημοσύνη (Artificial Intelligence - AI) και τη Μηχανική Μάθηση έφερε επανάσταση στη ρομποτική πλοήγηση. Η χρήση αισθητήρων όπως LiDAR, κάμερες βάθους και GNSS επέτρεψε στα ρομπότ να αποκτούν ακριβέστερη αντίληψη του χώρου.

Το επόμενο μεγάλο βήμα έγινε με τη Βαθιά Μάθηση (Deep Learning) και την Ενισχυτική Μάθηση (Reinforcement Learning). Οι τεχνικές Βαθιάς Ενισχυτικής Μάθησης (Deep Reinforcement Learning - DRL) επιτρέπουν πλέον στα ρομπότ να μαθαίνουν στρατηγικές πλοήγησης μέσω εμπειρίας και προσομοιώσεων.

Η εξέλιξη της πλοήγησης ρομποτικών πλατφορμών έχει ακολουθήσει μια εντυπωσιακή πορεία από τις πρώτες στατικές μεθόδους έως τα σύγχρονα έξυπνα συστήματα που βασίζονται στη Βαθιά Μάθηση. Μελλοντικές εξελίξεις αναμένεται να επικεντρωθούν στην προσαρμοστικότητα των αλγορίθμων σε άγνωστα περιβάλλοντα, στη βελτίωση της υπολογιστικής αποδοτικότητας και στην ενίσχυση της ασφάλειας, φέρνοντας την αυτόνομη πλοήγηση πιο κοντά στην πραγματική καθημερινή χρήση.

2.4 Αισθητήρες στην Πλοήγηση

Η πλοήγηση ρομποτικών πλατφορμών στηρίζεται στη δυνατότητα των ρομπότ να αντιλαμβάνονται το περιβάλλον, να σχεδιάζουν διαδρομές και να εκτελούν κινήσεις με ασφάλεια και ακρίβεια. Για να επιτευχθεί αυτό, χρησιμοποιούνται διάφοροι αισθητήρες που παρέχουν πληροφορίες σχετικά με τη θέση του ρομπότ, την παρουσία εμποδίων και τη δυναμική του περιβάλλοντος. Οι αισθητήρες αυτοί μπορούν να κατηγοριοποιηθούν ανάλογα με τη λειτουργία τους σε συστήματα αντίληψης του περιβάλλοντος, συστήματα προσδιορισμού θέσης και κίνησης, καθώς και ολοκληρωμένα συστήματα συγχώνευσης δεδομένων αισθητήρων (sensor fusion).

2.4.1 Αισθητήρες Αντίληψης Περιβάλλοντος

Οι αισθητήρες αντίληψης περιβάλλοντος επιτρέπουν στο ρομπότ να αναγνωρίζει τα αντικείμενα και τα εμπόδια που υπάρχουν στον χώρο. Η επιλογή του αισθητήρα που θα χρησιμοποιηθεί στην κάθε εφαρμογή γίνεται με βάση τις ανάγκες τις εκάστοτε εφαρμογής. Στον Πίνακα 1 αναφέρονται κάποια κριτήρια επιλογής αισθητήρων αντίληψης περιβάλλοντος.

LiDAR (Light Detection and Ranging)

Ο αισθητήρας LiDAR χρησιμοποιεί παλμούς λέιζερ για να υπολογίσει την απόσταση των αντικειμένων από το ρομπότ. Οι παλμοί εκπέμπονται προς διάφορες κατευθύνσεις και καταγράφεται ο χρόνος που χρειάζονται για να ανακλαστούν πίσω στον αισθητήρα. Μέσω αυτής της διαδικασίας δημιουργείται ένας τρισδιάστατος χάρτης του περιβάλλοντος, ο οποίος χρησιμοποιείται σε αλγορίθμους SLAM.

Η χρήση του LiDAR είναι ευρέως διαδεδομένη σε ρομποτικές πλατφόρμες και αυτόνομα οχήματα, καθώς επιτρέπει την ανίχνευση εμποδίων και τη δυναμική πλοήγηση σε εσωτερικά και εξωτερικά περιβάλλοντα. Τα κύρια πλεονεκτήματά του είναι η υψηλή ακρίβεια και η δυνατότητα δημιουργίας λεπτομερών χαρτών, ενώ τα βασικά του μειονεκτήματα περιλαμβάνουν το υψηλό κόστος και την ευαισθησία του σε επιφάνειες που δεν ανακλούν σωστά την ακτινοβολία λέιζερ.

Κάμερες RGB και RGB-D

Οι οπτικοί αισθητήρες, όπως οι κάμερες RGB και οι κάμερες βάθους (RGB-D), παρέχουν σημαντικές πληροφορίες σχετικά με το χρώμα, το σχήμα και τη δομή των αντικειμένων. Οι RGB κάμερες καταγράφουν οπτικά δεδομένα, ενώ οι RGB-D κάμερες συνδυάζουν την οπτική πληροφορία με δεδομένα βάθους που προέρχονται από τεχνικές όπως η στερεοσκοπική όραση ή η εκπομπή υπέρυθρων ακτινών.

Οι κάμερες χρησιμοποιούνται ευρέως για την αναγνώριση αντικειμένων και την κατανόηση της σκηνής μέσω αλγορίθμων υπολογιστικής όρασης και Βαθιάς Μάθησης.

Παρόλο που παρέχουν λεπτομερείς πληροφορίες για το περιβάλλον, η απόδοσή τους μπορεί να επηρεαστεί από συνθήκες χαμηλού φωτισμού ή γρήγορης κίνησης.

Ραντάρ (Radar)

Το ραντάρ λειτουργεί εκπέμποντας ραδιοκύματα και αναλύοντας την ανάκλασή τους, προκειμένου να υπολογίσει την απόσταση και την ταχύτητα των αντικειμένων. Σε αντίθεση με τις κάμερες και τους αισθητήρες LiDAR, το ραντάρ μπορεί να λειτουργεί αποτελεσματικά ακόμα και σε δυσμενείς καιρικές συνθήκες, όπως ομίχλη ή βροχή.

Η χρήση του ραντάρ στην πλοιόγηση ρομποτικών πλατφορμών είναι ιδιαίτερα σημαντική για την ανίχνευση κινούμενων αντικειμένων. Παρόλο που η ανάλυση των δεδομένων του ραντάρ είναι χαμηλότερη από αυτήν του LiDAR, η αξιοπιστία του σε δύσκολες συνθήκες καθιστά απαραίτητη τη χρήση του σε πολλές εφαρμογές.

Αισθητήρες Αντίληψης				
Αισθητήρας	Κόστος	Ακρίβεια Απόστασης	Αντίληψη Βάθους	Αντοχή
LiDAR	Πολύ Υψηλό	Υψηλή	Ναι	Μέτρια
RGB Camera	Χαμηλό	Χαμηλή	Όχι	Χαμηλή
RGB-D Camera	Μέτριο	Μέτρια	Ναι	Χαμηλή
Radar	Μέτριο	Μέτρια	Όχι	Υψηλή

Πίνακας 1: Αισθητήρες Αντίληψης

2.4.2 Αισθητήρες Προσδιορισμού Θέσης και Κίνησης

Η ακριβής γνώση της θέσης του ρομπότ είναι απαραίτητη για την αποτελεσματική πλοιόγησή του. Οι αισθητήρες θέσης και κίνησης χρησιμοποιούνται για την εκτίμηση του προσανατολισμού και της ταχύτητας της ρομποτικής πλατφόρμας. Στον Πίνακας 2 αναφέρονται κάποια κριτήρια επιλογής αισθητήρων προσδιορισμού θέσης και κίνησης.

GNSS (Global Navigation Satellite System)

Το GNSS χρησιμοποιεί δορυφορικά σήματα για τον προσδιορισμό της γεωγραφικής θέσης ενός ρομπότ ή οχήματος. Η τεχνολογία αυτή είναι ιδιαίτερα χρήσιμη για εφαρμογές σε εξωτερικούς χώρους. Ωστόσο, το GNSS παρουσιάζει περιορισμούς σε εσωτερικά περιβάλλοντα ή σε περιοχές με υψηλά κτίρια, όπου το σήμα μπορεί να εμποδίζεται.

Μονάδα Μέτρησης Αδράνειας (IMU - Inertial Measurement Unit)

Η IMU περιλαμβάνει ένα επιταχυνσιόμετρο και ένα γυροσκόπιο, επιτρέποντας την εκτίμηση της κίνησης του ρομπότ μέσω των αλλαγών στην ταχύτητα και την περιστροφή του. Η IMU χρησιμοποιείται συχνά σε περιβάλλοντα όπου το GNSS δεν είναι διαθέσιμο, όπως υπόγεια ή σήραγγες.

Αισθητήρες Προσδιορισμού Θέσης και Κίνησης			
Αισθητήρας	Κόστος	Ακρίβεια Θέσης	Αντοχή
GNSS	Χαμηλό	Μέτρια	Υψηλή
IMU	Χαμηλό	Υψηλή	Υψηλή

Πίνακας 2: Αισθητήρες Αντίληψης

2.4.3 Συνδυασμός Αισθητήρων για Ακριβέστερη Πλοϊγηση

Για να ξεπεραστούν οι περιορισμοί κάθε μεμονωμένου αισθητήρα, οι σύγχρονες ρομποτικές πλατφόρμες χρησιμοποιούν τεχνικές συγχώνευσης δεδομένων από αισθητήρες (sensor fusion). Ο συνδυασμός δεδομένων από LiDAR, κάμερες, ραντάρ, GNSS και IMU επιτρέπει τη δημιουργία ενός πιο ακριβούς και αξιόπιστου μοντέλου του περιβάλλοντος, βελτιώνοντας την ικανότητα του ρομπότ να πλοηγείται με ασφάλεια.

3 Μηχανική Μάθηση

3.1 Ορισμός και Θεμελιώδη Χαρακτηριστικά

Η Μηχανική Μάθηση (Machine Learning) είναι ένα υποσύνολο της AI που σχετίζεται με την ανάπτυξη αλγορίθμων και υπολογιστικών μοντέλων τα οποία επιτρέπουν σε υπολογιστικές μηχανές να μαθαίνουν από δεδομένα και να βελτιώνουν την απόδοσή τους με την εμπειρία.

Ο Arthur Samuel, ένας από τους πρωτοπόρους του πεδίου, περιέγραψε τη Μηχανική Μάθηση ως «Το πεδίο που δίνει στους υπολογιστές τη δυνατότητα να μαθαίνουν χωρίς να έχουν προγραμματιστεί ρητά» [11]. Μία πιο σύγχρονη προσέγγιση για την έννοια της Μηχανικής Μάθησης αναπτύχθηκε από τον Tom Mitchell, σύμφωνα με τον οποίο [12]:

«Ένα πρόγραμμα υπολογιστή μαθαίνει από εμπειρία E σε σχέση με κάποια τάξη εργασιών T και κάποιο μέτρο απόδοσης P , αν η απόδοσή του σε εργασίες της T , όπως μετριέται από το P , βελτιώνεται με την εμπειρία E .»

Με άλλα λόγια ένα σύστημα θεωρείται πως «μαθαίνει» όταν με την πάροδο του χρόνου και την επεξεργασία περισσότερων δεδομένων, βελτιώνει συστηματικά την ικανότητά του να επιλύει προβλήματα συγκεκριμένου τύπου.

3.1.1 Τύποι Μηχανικής Μάθησης

Η Μηχανική Μάθηση διακρίνεται σε τρεις βασικούς τύπους, ανάλογα με τον τρόπο που τα μοντέλα μαθαίνουν από τα δεδομένα και τον τύπο της πληροφορίας που τους παρέχεται κατά την εκπαίδευση. Κάθε τύπος έχει διαφορετικά χαρακτηριστικά και εφαρμογές.

Εποπτευόμενη Μάθηση (Supervised Learning)

Η Εποπτευόμενη Μάθηση, είναι ένας τύπος Μηχανικής Μάθησης με στόχο την εκμάθηση μιας συνάρτησης $f: X \rightarrow Y$, η οποία μπορεί να προβλέψει την τιμή $y \in Y$ για κάθε νέα είσοδο $x \in X$, ελαχιστοποιώντας μία συνάρτηση κόστους. Για την εκπαίδευση χρησιμοποιείται ένα σύνολο δεδομένων που περιλαμβάνει ετικέτες (labels), όπου κάθε δείγμα αποτελείται από ένα ζεύγος εισόδου-εξόδου. Τα πιο βασικά προβλήματα που υπάγονται στην Εποπτευόμενη Μάθηση είναι τα εξής:

- Ταξινόμηση (Classification):** Η ταξινόμηση είναι ένα πρόβλημα κατά το οποίο ο αλγόριθμος καλείται να αντιστοιχίσει κάθε είσοδο σε μία από προεπιλεγμένες κατηγορίες-ετικέτες.

- **Παλινδρόμηση (Regression):** Η παλινδρόμηση είναι ένα πρόβλημα πρόβλεψης συνεχών τιμών. Στόχος είναι η χάραξη μίας γραμμής ή καμπύλης μεταξύ των δεδομένων που αντιπροσωπεύει τη σχέση μεταξύ των ανεξάρτητων και των εξαρτημένων μεταβλητών.

Μη Εποπτευόμενη Μάθηση (Unsupervised Learning)

Η **Μη Εποπτευόμενη Μάθηση** αφορά περιπτώσεις όπου δεν υπάρχουν διαθέσιμες ετικέτες στα δεδομένα εκπαίδευσης. Το ζητούμενο είναι η αναγνώριση δομών (pattern recognition) και σχέσεων μεταξύ των δεδομένων χωρίς ετικέτες. Το πιο σύνηθες πρόβλημα Μη Εποπτευόμενης Μάθησης είναι η Ομαδοποίηση (Clustering) κατά την οποία ο αλγόριθμος στοχεύει στην ανάθεση των δειγμάτων σε ομάδες (clusters) με βάση την ομοιότητά που εμφανίζουν μεταξύ τους.

Ενισχυτική Μάθηση

Η **Ενισχυτική Μάθηση** διαφέρει ριζικά από τους παραπάνω τύπους Μηχανικής Μάθησης. Στην Ενισχυτική Μάθηση ένας πράκτορας (agent) μαθαίνει να λαμβάνει αποφάσεις μέσω αλληλεπίδρασης με ένα περιβάλλον, με σκοπό την εκτέλεση μίας λειτουργίας. Ο πράκτορας εκτελεί μία δράση (action) ενώ βρίσκεται σε μία κατάσταση (state) και λαμβάνει ανατροφοδότηση με τη μορφή θετικών ή αρνητικών ανταμοιβών. Ο πράκτορας προσπαθώντας να μεγιστοποιήσει το κέρδος που λαμβάνει μέσω δοκιμής και σφάλματος, μαθαίνει σταδιακά τις βέλτιστες δράσεις που πρέπει να λάβει σε κάθε κατάσταση που βρίσκεται έτσι ώστε να εκτελέσει σωστά την εργασία που του έχει ανατεθεί.

3.1.2 Βαθιά Μάθηση

Η **Βαθιά Μάθηση** αποτελεί υποσύνολο της Μηχανικής Μάθησης όπου η κύρια λειτουργία της βασίζεται στην χρήση πολυεπίπεδων νευρωνικών δικτύων (βαθιά δίκτυα) για την εκτέλεση εργασιών. Χρησιμοποιείται κυρίως σε προβλήματα μεγάλης πολυπλοκότητας με μεγάλο όγκο δεδομένων (big data), όπου τα δεδομένα είναι υψηλής διάστασης, μη γραμμικά και δύσκολο να επεξεργαστούν με παραδοσιακές μεθόδους Μηχανικής Μάθησης. Η όραση υπολογιστών, η αναγνώριση ήχου, η αυτόνομη οδήγηση και η ρομποτική είναι μερικές μόνο από τις περιπτώσεις όπου εφαρμόζεται η Βαθιά Μάθηση στην σήμερον ημέρα.

3.1.3 Νευρωνικά Δίκτυα

Τα **νευρωνικά νίκτυα** (neural networks) αποτελούν έναν βασικό υπολογιστικό μηχανισμό της Μηχανικής Μάθησης και ειδικότερα της Βαθιάς Μάθησης, εμπνευσμένο

από τη λειτουργία του ανθρώπινου εγκεφάλου. Πρόκειται για ένα σύνολο υπολογιστικών μονάδων, των λεγόμενων **τεχνητών νευρώνων** (neurons), οι οποίες συνεργάζονται μεταξύ τους σε διαδοχικά **στρώματα** (layers), με σκοπό την αναγνώριση προτύπων και τη μάθηση από δεδομένα. Μέσω αυτής της αρχιτεκτονικής, τα νευρωνικά δίκτυα έχουν την ικανότητα να προσεγγίζουν σύνθετες συναρτήσεις, να αναλύουν μεγάλης διάστασης δεδομένα και να επιλύουν προβλήματα πρόβλεψης, ταξινόμησης και λήψης αποφάσεων.

Ένα βασικό νευρωνικό δίκτυο (Εικόνα 4) αποτελείται από τρία κύρια μέρη: το **στρώμα εισόδου** (input layer), ένα ή περισσότερα **κρυφά στρώματα** (hidden layers) και το **στρώμα εξόδου** (output layer). Το κάθε κρυφό στρώμα αποτελείται από πολλαπλούς νευρώνες, οι οποίοι δέχονται σήματα από το προηγούμενο επίπεδο και παράγουν έξοδο μέσω μιας **συνάρτησης ενεργοποίησης**. Οι νευρώνες συνδέονται μεταξύ τους με **βάρη** (weights), τα οποία καθορίζουν τη σημασία-βαρύτητα του κάθε σήματος. Η εξίσωση η οποία περιγράφει την λειτουργεία ενός τεχνητού νευρώνα j , είναι η εξής:

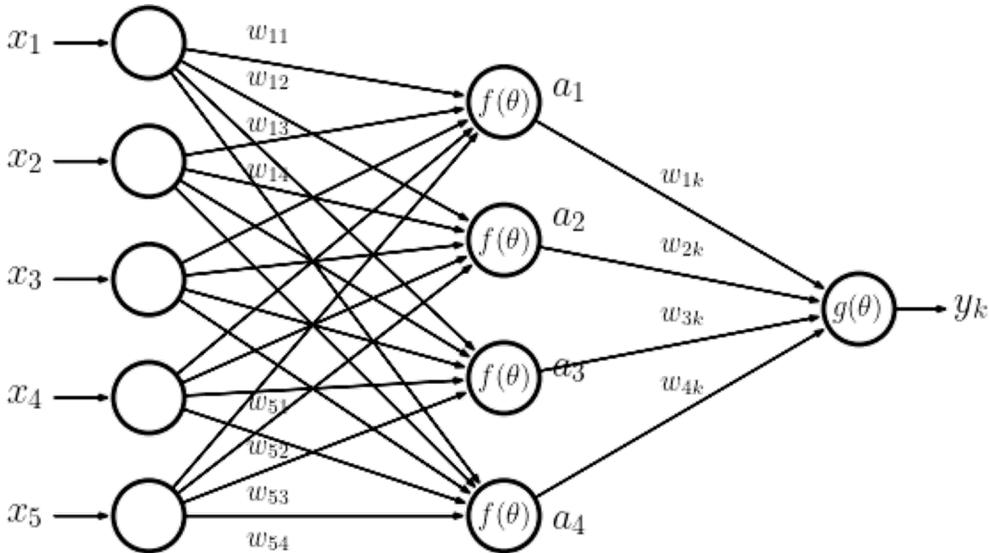
$$a_j = f \left(\sum_i w_{ij} x_i + b_j \right),$$

όπου:

- a_j η τελική έξοδος του νευρώνα,
- f η συνάρτηση ενεργοποίησης,
- x_i οι είσοδοι του νευρώνα,
- w_{ij} τα βάρη των συνδέσεων,
- b_j η προκατάληψη (bias).

Αφού υπολογιστούν όλες οι έξοδοι των νευρώνων των κρυφών στρωμάτων, στο τέλος υπολογίζεται η έξοδος y_k του τελικού νευρώνα του νευρωνικού δικτύου. Όπως και στα κρυφά στρώματα έτσι και στο στρώμα εξόδου χρησιμοποιείται μία συνάρτηση ενεργοποίησης $g(\theta)$. Η σχέση που περιγράφει τον νευρώνα εξόδου είναι η εξής:

$$y_k = g \left(\sum_j w_{jk} a_j + b_k \right),$$



Εικόνα 4: Τεχνητό Νευρωνικό Δίκτυο

3.1.4 Συνάρτηση Κόστους

Ο στόχος της εκπαίδευσης ενός νευρωνικού δικτύου είναι να κάνει το μοντέλο όσο το δυνατόν πιο ακριβές στις προβλέψεις του. Για την επίτευξη αυτού του στόχου χρησιμοποιείται μία συνάρτηση η οποία ονομάζεται **συνάρτηση κόστους (cost function)**. Η συνάρτηση αυτή καθορίζει πόσο "λάθος" είναι η πρόβλεψη ενός μοντέλου σε σχέση με την πραγματική τιμή (ground truth) για ένα δεδομένο δείγμα, μετρώντας την διαφορά τους. Σε κάθε επανάληψη της εκπαίδευσης, το δίκτυο κάνει μια πρόβλεψη για τα δεδομένα εισόδου, υπολογίζει το σφάλμα με βάση τη συνάρτηση κόστους, και στη συνέχεια τροποποιεί τα βάρη του ώστε να μειώσει αυτό το σφάλμα στο επόμενο βήμα. Έτσι, η συνάρτηση κόστους αποτελεί έναν μηχανισμό αξιολόγησης και καθοδήγησης [13].

Η κατάλληλη επιλογή συνάρτησης κόστους εξαρτάται από το είδος του προβλήματος. Παρακάτω παρουσιάζονται με λεπτομέρεια οι πιο διαδεδομένες συναρτήσεις κόστους:

Mean Squared Error (MSE)

Η πιο διαδεδομένη συνάρτηση κόστους για προβλήματα παλινδρόμησης. Υπολογίζει τη μέση τετραγωνική διαφορά μεταξύ των εκτιμώμενων τιμών και της πραγματικής τιμής. Λόγω της ύπαρξης του τετραγώνου, η συγκεκριμένη συνάρτηση επιπλήττει περισσότερο τα μεγάλα σφάλματα.

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

όπου:

- y_i η πραγματική τιμή,

- \hat{y}_i η εκτιμώμενη τιμή,
- n ο αριθμός δειγμάτων.

Mean Absolute Error (MAE)

Η MAE υπολογίζει τη μέση απόλυτη τιμή της διαφοράς μεταξύ πραγματικής και προβλεπόμενης τιμής. Σε αντίθεση με την MSE, η MAE είναι πιο ανθεκτική σε ακραίες τιμές.

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

Binary Cross-Entropy

Η συνάρτηση Binary Cross-Entropy χρησιμοποιείται σε δυαδικά προβλήματα ταξινόμησης, όπου η έξοδος είναι μία πιθανότητα.

$$BSE = -\frac{1}{n} \sum_{i=1}^n (y_i \cdot \log(\hat{y}_i) + (1 - y_i) \cdot \log(1 - \hat{y}_i)) ,$$

όπου:

- $y_i \in \{0,1\}$,
- $\hat{y}_i \in (0,1)$.

Categorical Cross-Entropy

Η συνάρτηση Categorical Cross-Entropy χρησιμοποιείται σε προβλήματα πολυταξινόμισης. Η έξοδος \hat{y} είναι διανυσματική και περιέχει τις πιθανότητες για κάθε κατηγορία, ενώ η πραγματική τιμή εξόδου y είναι one-hot encoded.

$$CCE = - \sum_{i=1}^c (y_i \cdot \log(\hat{y}_i)) ,$$

όπου:

- y_i η one-hot encoded πραγματική τιμή εξόδου,
- \hat{y}_i το διάνυσμα πιθανοτήτων για την κάθε κατηγορία,
- c ο αριθμός κατηγοριών.

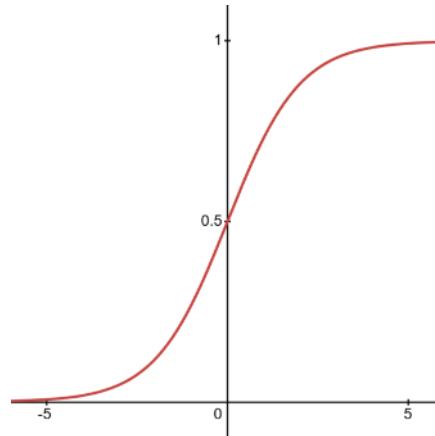
3.1.5 Συνάρτηση Ενεργοποίησης

Οι συναρτήσεις ενεργοποίησης (activation functions) είναι μη γραμμικές συναρτήσεις οι οποίες εφαρμόζονται στην έξοδο κάθε νευρώνα, επιτρέποντας στο δίκτυο να μάθει να λύνει πολύπλοκα προβλήματα. Χωρίς αυτές, το δίκτυο θα ισοδυναμούσε με μια γραμμική συνάρτηση και θα εμφάνιζε ανεπάρκεια στην επίλυση σύνθετων προβλημάτων. Οι πιο συνήθεις συναρτήσεις ενεργοποίησης είναι οι sigmoid, ReLU, tanh και softmax.

Sigmoid

Παράγει τιμές μεταξύ 0 και 1 και χρησιμοποιείται κυρίως σε προβλήματα δυαδικής ταξινόμησης (Εικόνα 5). Αν και είχε εκτενή χρήση στα πρώτα στάδια της Βαθιάς Μάθησης, σήμερα θεωρείται περιοριστική λόγω του προβλήματος vanishing gradient (οι παράμετροι του μοντέλου γίνονται σχεδόν αμελητέες με την πάροδο του χρόνου σε βαθιά δίκτυα), το οποίο εμποδίζει την αποτελεσματική εκπαίδευση, και έχει πλέον αντικατασταθεί σε μεγάλο βαθμό από πιο σύγχρονες συναρτήσεις.

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

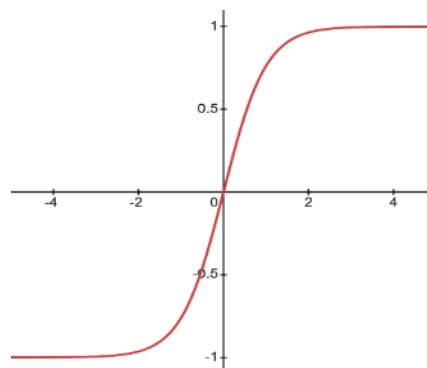


Εικόνα 5: Συνάρτηση Sigmoid

Hyperbolic Tangent (Tanh)

Παράγει τιμές στο διάστημα [-1, 1] (Εικόνα 6). Παρουσιάζει καλύτερη απόδοση σε σχέση με τη sigmoid, αλλά εξακολουθεί να υποφέρει από vanishing gradients.

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

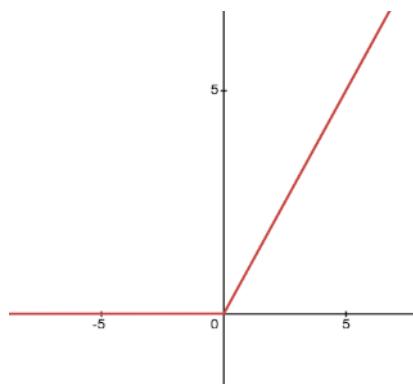


Εικόνα 6: Συνάρτηση Tanh

ReLU (Rectified Linear Unit)

Η πιο διαδεδομένη συνάρτηση ενεργοποίησης, λόγω της απλότητας και της αποδοτικότητάς της κατά την εκπαίδευση (Εικόνα 7). Επιτρέπει την ταχύτερη σύγκλιση και καταπολεμά αποτελεσματικά το πρόβλημα vanishing gradient.

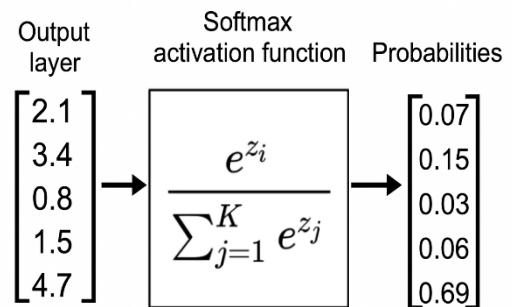
$$f(x) = \max(0, x).$$



Εικόνα 7: Συνάρτηση ReLU

Softmax

Η συνάρτηση softmax μετατρέπει ένα διάνυσμα πραγματικών αριθμών σε ένα διάνυσμα πιθανοτήτων (Εικόνα 8). Χρησιμοποιείται στο στρώμα εξόδου ενός νευρωνικού δικτύου και κυρίως σε προβλήματα ταξινόμησης, όπου απαιτείται κατανομή πιθανοτήτων στο διάνυσμα εξόδου.



$$\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^K e^{z_j}}.$$

Εικόνα 8: Συνάρτηση Softmax

3.1.6 Αλγόριθμοι Βελτιστοποίησης

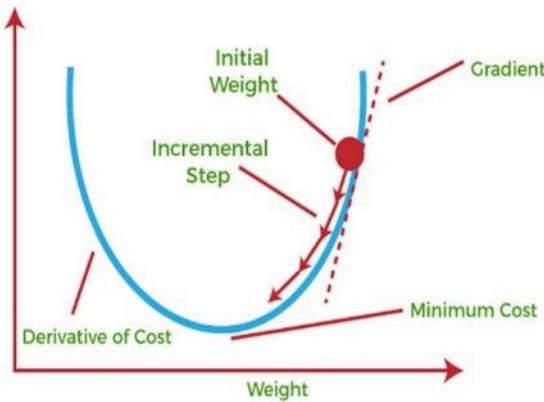
Οι Αλγόριθμοι Βελτιστοποίησης (Optimization Algorithms - Optimizers) είναι υπεύθυνοι για την ανανέωση των βαρών, καθορίζουν δηλαδή το πως πρέπει να αλλάξουν τα βάρη του δικτύου σε κάθε εποχή με σκοπό την σύγκληση της συνάρτησης κόστους σε ένα τοπικό ή ολικό ελάχιστο. Οι πιο γνωστοί αλγόριθμοι βελτιστοποίησης είναι οι Gradient Descent, Stochastic Gradient Descent και Adam [13].

Gradient Descent (GD)

Ο αλγόριθμος Gradient Descent υπολογίζει την κλήση (πρώτη παράγωγος) της συνάρτησης κόστους $J(\theta)$ και έπειτα τροποποιεί τα βάρη θ έτσι ώστε να κάνει ένα βήμα προς την αρνητική κατεύθυνση της κλήσης (Εικόνα 9). Το μέγεθος του βήματος καθορίζεται από μία ρυθμιστική παράμετρο η οποία ονομάζεται **Ρυθμός Μάθησης (Learning Rate)** και συμβολίζεται ως α . Με αυτό τον τρόπο οδηγείται σιγά σιγά προς ένα ελάχιστο της συνάρτησης.

$$\theta_j = \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta) .$$

Ο GD υπολογίζει την κλίση με βάση ολόκληρο το dataset σε κάθε επανάληψη. Είναι εξαιρετικά χρήσιμος σε μικρά datasets και σε κυρτές συναρτήσεις. Αντιθέτως είναι πολύ αργός όταν πρέπει να διαχειριστεί μεγάλα datasets και εμφανίζει προβλήματα σε πολύπλοκες συναρτήσεις, διότι συχνά εγκλωβίζεται σε κάποιο τοπικό ελάχιστο.



Εικόνα 9: Γραφική αναπαράσταση του Gradient Descent

Stochastic Gradient Descent (SGD)

Ο SGD είναι μία παραλλαγή του GD που προσφέρει μία αποδοτικότερη λύση σε προβλήματα με μεγάλα datasets. Αντί να χρησιμοποιεί ολόκληρο το dataset σε κάθε επανάληψη, χρησιμοποιεί μόνο ένα δείγμα ή ένα μικρό σύνολο δειγμάτων. Με αυτό τον τρόπο επιταχύνεται η διαδικασία υπολογισμού.

$$\theta_j = \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta; x_i, y_i) ,$$

όπου (x_i, y_i) ένα τυχαίο δείγμα του dataset.

Adaptive Moment Estimation (Adam)

Ο αλγόριθμος Adam είναι ο πλέον πιο χρησιμοποιημένος αλγόριθμος βελτιστοποίησης σε προβλήματα Βαθιάς Μάθησης. Είναι ένας συνδυασμός των αλγορίθμων Momentum και RMSProp, ιδιαίτερα αποτελεσματικός σε πολύπλοκα μοντέλα με μεγάλα datasets.

- **Momentum** - Χρησιμοποιείται για την επιτάχυνση της διαδικασίας gradient descent με την χρήση ενός κινητού μέσου όρου των κλήσεων.

$$\begin{aligned}\theta_{t+1} &= \theta_t - \alpha m_t, \\ m_t &= \beta m_{t-1} + (1 - \beta) \frac{\partial}{\partial \theta_j} J,\end{aligned}$$

όπου:

- m_t ο κινητός μέσος όρος των κλήσεων,
- β το decay rate.

- **RMSProp** - Χρησιμοποιεί τον κινητό μέσο όρο του τετραγώνου των κλήσεων.

$$\begin{aligned}\theta_{t+1} &= \theta_t - \frac{\alpha}{\sqrt{v_t + \epsilon}} \frac{\partial}{\partial \theta_j} J, \\ v_t &= \beta v_{t+1} + (1 - \beta) \left(\frac{\partial}{\partial \theta_j} J \right)^2,\end{aligned}$$

όπου:

- v_t ο κινητός μέσος όρος των τετραγώνων των κλήσεων,
- ϵ μία σταθερά (συνήθως $< 10^{-7}$).

Δεδομένου ότι οι μέσοι όροι αρχικοποιούνται με μηδενικές τιμές, είναι biased προς το μηδέν. Για να αποφευχθεί αυτό, υπολογίζονται οι διορθώσεις των μέσων όρων με βάση το bias ως εξής:

$$\hat{m}_t = \frac{m_t}{1 - \beta_1}, \hat{v}_t = \frac{v_t}{1 - \beta_2}.$$

Τέλος, τα τελικά βάρη ανανεώνονται σύμφωνα με την εξής σχέση:

$$\theta_{t+1} = \theta_t - \alpha \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}.$$

Ο αλγόριθμος Adam υπερτερεί έναντι των υπόλοιπων μεθόδων βελτιστοποίησης σε πολλές εφαρμογές. Η ικανότητά του προσαρμόζει τον ρυθμό μάθησης ξεχωριστά για κάθε παράμετρο μέσω των εκθετικών μέσων όρων τον καθιστά ιδιαίτερα αποτελεσματικό σε πιο πολύπλοκα μοντέλα. Η προσαρμογή αυτή πραγματοποιείται με τρόπο έτσι ώστε αν μία παράμετρος έχει μεγάλο ή ασταθές gradient, ο Adam αυτόματα προσαρμόζει το learning rate ώστε να κάνει μικρότερα ή πιο σταθερά βήματα. Αντίστοιχα, για παραμέτρους με μικρό gradient, ο αλγόριθμος επιτρέπει μεγαλύτερα βήματα ώστε να επιταχύνει τη μάθηση.

3.1.7 Αλγόριθμος Backpropagation

Ο αλγόριθμος Backpropagation είναι ένας αλγόριθμος εκπαίδευσης τεχνητών νευρωνικών δικτύων. Ο αλγόριθμος εισήχθη συστηματικά στο πεδίο των νευρωνικών δικτύων από τους Rumelhart, Hinton και Williams (1986), και αποτέλεσε το σημείο καμπής που

κατέστησε εφικτή την εκπαίδευση πολυεπίπεδων perceptrons, δηλαδή των πρώτων βαθιών δικτύων [14]. Η βασική ιδέα του backpropagation είναι ότι το σφάλμα που προκύπτει στην έξοδο ενός δικτύου “γυρνάει” προς τα πίσω, από το στρώμα εξόδου προς τα κρυφά στρώματα, με τρόπο που επιτρέπει τον υπολογισμό του βαθμού στον οποίο κάθε βάρος συνέβαλε στο συνολικό σφάλμα. Με αυτόν τον τρόπο, τα βάρη ενημερώνονται έτσι ώστε το δίκτυο να μειώνει προοδευτικά τη συνάρτηση κόστους.

Ο αλγόριθμος backpropagation περιλαμβάνει τέσσερα βασικά βήματα:

1. **Forward pass:** Υπολογίζεται η έξοδος του δικτύου για δεδομένες τιμές εισόδου και βαρών.
2. **Υπολογισμός σφάλματος:** Υπολογίζεται η διαφορά μεταξύ της πραγματικής εξόδου και της εξόδου του δικτύου μέσω μίας συνάρτησης κόστους.
3. **Backpropagation του σφάλματος:** Υπολογίζονται οι παράγωγοι της συνάρτησης κόστους ως προς κάθε βάρος, ξεκινώντας από την έξοδο και προχωρώντας προς τα πίσω μέσω του κανόνα της αλυσίδας (chain rule).
4. **Ενημέρωση βαρών:** Ενημερώνονται οι τιμές των βαρών σύμφωνα με τον αλγόριθμο βελτιστοποίησης.

3.1.8 Υπερπαράμετροι

Οι **Υπερπαράμετροι** (Hyperparameters) είναι ρυθμίσεις που ορίζονται πριν από την εκπαίδευση ενός μοντέλου και επηρεάζουν άμεσα τη συμπεριφορά, την ταχύτητα σύγκλισης και την απόδοση του μοντέλου. Σε αντίθεση με τις παραμέτρους του μοντέλου, όπως τα βάρη και τα bias, όπου τιμές τους αλλάζουν δυναμικά κατά την διάρκεια της εκπαίδευσης, οι υπερπαράμετροι ρυθμίζονται εξωτερικά από τον χρήστη πριν ξεκινήσει η διαδικασία της εκπαίδευσης.

Υπάρχουν πολλών ειδών υπερπαράμετροι ανάλογα με το είδος της μάθησης και τους αλγορίθμους που χρησιμοποιούνται. Παρακάτω αναφέρονται μερικές από τις σημαντικότερες και πιο συνήθεις υπερπαραμέτρους Μηχανικής Μάθησης.

- **Learning rate:**

Η τιμή του learning rate κυμαίνεται συνήθως μεταξύ 10^{-1} και 10^{-7} . Χαμηλές τιμές οδηγούν σε αργή αλλά σταθερή σύγκληση ενώ υψηλές τιμές μπορούν να οδηγήσουν σε γρήγορη σύγκληση με αντίκτυπο την αστάθεια.

- **Εποχές:**

Εποχή (epoch) ονομάζεται ένα πλήρες πέρασμα του dataset από την αλγόριθμο εκπαίδευσης. Ο αριθμός των εποχών σε κάθε εκπαίδευση μπορεί να επηρεάσει σημαντικά το αποτέλεσμα της εκπαίδευσης.

- **Batch size:**

Η τιμή του batch size καθορίζει το πλήθος των δειγμάτων που επεξεργάζεται το μοντέλο πριν ενημερώσει τα βάρη του. Επηρεάζει άμεσα την ταχύτητα εκπαίδευσης, την σύγκληση και την ακρίβεια του μοντέλου. Μικρό batch size απαιτεί λιγότερη μνήμη, αλλά επιβραδύνει την εκπαίδευση και προκαλεί αστάθεια στην σύγκληση. Αντιθέτως, μεγάλο batch size απαιτεί μεγαλύτερη επεξεργαστική ισχύ, προσδίδει πιο σταθερή σύγκληση αλλά μπορεί να εμφανίσει προβλήματα γενίκευσης και παγίδευσης σε τοπικά ελάχιστα.

Η ρύθμισή τους συχνά απαιτεί πειραματισμό, καθώς δεν υπάρχει καθολικά σωστή τιμή, εξαρτάται από τη φύση του προβλήματος και τα δεδομένα.

3.1.9 Υπερπροσαρμογή/Υποπροσαρμογή

Η υπερπροσαρμογή (overfitting) είναι η κατάσταση κατά την οποία ένα μοντέλο μαθαίνει τα δεδομένα εκπαίδευσης σε σημείο που προσαρμόζεται υπερβολικά σε αυτά. Το αποτέλεσμα είναι ότι, ενώ η απόδοση κατά την εκπαίδευση είναι εξαιρετική, η απόδοση σε νέα άγνωστα δεδομένα (test set) είναι χαμηλή. Για παράδειγμα ένα νευρωνικό δίκτυο που έχει εκπαιδευτεί σε πολύ λίγα δεδομένα ή για πάρα πολλές εποχές μπορεί να απομνημονεύσει πλήρως τα δεδομένα εκπαίδευσης και να προσαρμοστεί υπερβολικά σε αυτά, χωρίς να έχει μάθει τις γενικές ιδιότητες του προβλήματος.

Για να αποφευχθεί το overfitting χρησιμοποιούνται τεχνικές κανονικοποίησης, βελτιώνοντας την ικανότητα γενίκευσης του μοντέλου. Ενδεικτικά μερικές από τις πιο γνωστές τεχνικές κανονικοποίησης είναι η χρήση dropout layers, early stopping, data augmentation και batch normalization.

Η υποπροσαρμογή (underfitting) είναι το φαινόμενο κατά το οποίο ένα μοντέλο αποτυγχάνει να μάθει επαρκώς από τα δεδομένα εκπαίδευσης, με αποτέλεσμα να έχει χαμηλή απόδοση τόσο στο training όσο και στο test set. Πρόκειται για την αντίθετη κατάσταση από το overfitting και είναι εξίσου προβληματική, καθώς υποδηλώνει ότι το μοντέλο είναι υπερβολικά απλό για να μάθει σωστά τη δομή ή την πολυπλοκότητα των δεδομένων.

Η αντιμετώπιση του underfitting βασίζεται στην αύξηση της πολυπλοκότητας του μοντέλου, την μείωση της κανονικοποίησης και την εκπαίδευση για περισσότερες εποχές.

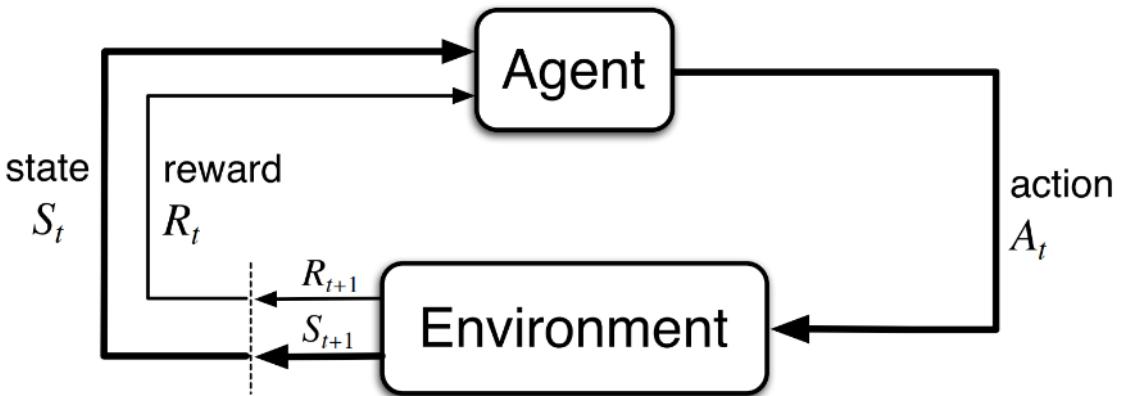
3.1.10 Ρύθμιση Υπερπαραμέτρων

Η ρύθμιση υπερπαραμέτρων (hyperparameter tuning) είναι η διαδικασία επιλογής των τιμών των υπερπαραμέτρων που ορίζονται για μία εκπαίδευση. Στόχος της είναι να βρεθεί το σύνολο τιμών των υπερπαραμέτρων που προσφέρει το καλύτερο αποτέλεσμα. Το

hyperparameter tuning είναι εξαιρετικά σημαντικό για την βελτιστοποίηση της αποτελεσματικότητας ενός αλγορίθμου Μηχανικής Μάθησης. Κακές υπερπαράμετροι οδηγούν σε αστάθεια, βραδεία ή εσφαλμένη σύγκλιση, overfitting ή underfitting, και χαμηλή αποδοτικότητα. Αντιθέτως, σωστά ρυθμισμένες τιμές βελτιώνουν την ακρίβεια, την ανθεκτικότητα σε θόρυβο, και μειώνουν τον χρόνο εκπαίδευσης.

3.2 Ενισχυτική Μάθηση

Τα τελευταία χρόνια, η **Ενισχυτική Μάθηση** έχει αναδειχθεί ως μία από τις πιο σημαντικές και δυναμικές τεχνολογίες στον χώρο της Υπολογιστικής Νοημοσύνης. Η Ενισχυτική Μάθηση αποτελεί ένα υπολογιστικό πλαίσιο μέσω του οποίου ένας **πράκτορας** (agent) μαθαίνει πώς να αλληλοεπιδρά με ένα **περιβάλλον** (environment), με στόχο τη μεγιστοποίηση της συνολικής ανταμοιβής που δέχεται με την πάροδο του χρόνου (Εικόνα 10). Η εκπαίδευση του πράκτορα βασίζεται στην εμπειρία και πραγματοποιείται μέσω δοκιμής και σφάλματος (trial-and-error). Ο πράκτορας δεν γνωρίζει εκ των προτέρων ποιες ενέργειες είναι οι σωστές, αντ' αυτού δοκιμάζει διάφορες επιλογές και παρατηρεί τις συνέπειες, ώστε να ανακαλύψει ποιες αποδίδουν τα καλύτερα αποτέλεσματα. Σε κάθε χρονικό βήμα, ο πράκτορας παρατηρεί την **κατάσταση** (state) του περιβάλλοντος, επιλέγει μια **δράση** (action) σύμφωνα με μια στρατηγική (πολιτική, ή policy), και ως αποτέλεσμα λαμβάνει μια **ανταμοιβή** (reward) μεταβαίνοντας σε μια νέα κατάσταση [15].



Εικόνα 10: Το πλαίσιο της Ενισχυτικής Μάθησης

Κατά την έναρξη της διαδικασίας, ο πράκτορας λαμβάνει από το περιβάλλον μια αρχική κατάσταση S_0 — για παράδειγμα, μία εικόνα από αισθητήρες. Με βάση την πληροφορία που περιέχει αυτή την κατάσταση, ο πράκτορας επιλέγει να εκτελέσει μία δράση A_0 — όπως για παράδειγμα, να κινηθεί προς τα δεξιά. Το περιβάλλον, μεταβαίνει σε μια νέα κατάσταση S_1 , την οποία επιστρέφει στον πράκτορα. Παράλληλα, το

περιβάλλον επιστρέφει μια ανταμοιβή R_1 στον πράκτορα — για παράδειγμα, θετική (+1) αν ο πράκτορας δεν έχει αποτύχει. Αυτή η αλληλουχία συνεχίζεται επαναληπτικά, δημιουργώντας μια σειρά καταστάσεων, δράσεων και ανταμοιβών, η οποία χρησιμοποιείται για την εκπαίδευση του πράκτορα με στόχο τη βελτιστοποίηση της πολιτικής του.

Μία αυτοτελής ακολουθία αλληλεπιδράσεων μεταξύ ενός πράκτορα και του περιβάλλοντος, με σαφές σημείο έναρξης και τερματισμού ονομάζεται **επεισόδιο** (episode) και αποτελεί μία από τις βασικότερες έννοιες της Ενισχυτικής Μάθησης. Ένα επεισόδιο ξεκινά όταν ο πράκτορας βρίσκεται σε μια αρχική κατάσταση και στη συνέχεια εκτελεί ενέργειες, λαμβάνει ανταμοιβές και μεταβαίνει από κατάσταση σε κατάσταση. Το επεισόδιο τελειώνει είτε όταν επιτευχθεί ο στόχος είτε όταν συμβεί κάποιο γεγονός τερματισμού (π.χ. αποτυχία, χρονικό όριο).

Στην Ενισχυτική Μάθηση, ο στόχος του πράκτορα είναι να μεγιστοποιήσει τη **σωρευτική αναμενόμενη ανταμοιβή** (expected cumulative reward), κάτι που θεμελιώνεται στην επονομαζόμενη **υπόθεση της ανταμοιβής** (reward hypothesis).

Το πρόβλημα της Ενισχυτικής Μάθησης διατυπώνεται με όρους που προέρχονται από τα δυναμικά συστήματα και τη θεωρία βέλτιστου ελέγχου, συγκεκριμένα μέσω του μαθηματικού μοντέλου **Markov Decision Process** (MDP). Το μοντέλο ορίζεται ως (S, A, P, r, γ) , όπου:

- **S**: είναι το σύνολο των πιθανών καταστάσεων του περιβάλλοντος,
- **A**: είναι το σύνολο των πιθανών ενεργειών που μπορεί να επιλέξει ο πράκτορας,
- **P(s' | s, a)**: είναι η πιθανότητα μετάβασης στην κατάσταση s' όταν εφαρμόζεται η ενέργεια a στην κατάσταση s ,
- **r(s, a)**: είναι η αναμενόμενη ανταμοιβή για την ενέργεια a στη κατάσταση s ,
- **$\gamma \in [0, 1]$** : είναι ο **συντελεστής απόσβεσης** (discount factor), που καθορίζει τη σημασία των μελλοντικών ανταμοιβών.

Ο στόχος του πράκτορα είναι να μάθει μια πολιτική, με σκοπό την μεγιστοποίηση της σωρευτικής αναμενόμενης ανταμοιβής ($R_{(\tau)}$, όπου τ η αλληλουχία καταστάσεων και ενεργειών), η οποία ορίζεται ως:

$$R_{(\tau)} = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}.$$

Η ανατροφοδότηση που λαμβάνει ο πράκτορας δεν είναι καθοδηγητική (όπως στην Εποπτευόμενη Μάθηση), αλλά ενισχυτική: ο πράκτορας δοκιμάζει, παρατηρεί τις συνέπειες και προσαρμόζει τη στρατηγική του με βάση τις ανταμοιβές, γεγονός που καθιστά την Ενισχυτική Μάθηση ιδιαίτερα κατάλληλη για εφαρμογές όπου δεν υπάρχει άμεσος τρόπος καθορισμού της “σωστής” ενέργειας σε κάθε βήμα.

3.2.1 Τύποι Εργασιών στην Ενισχυτική Μάθηση

Στην Ενισχυτική Μάθηση, ο όρος **εργασία** (task) αναφέρεται στο πρόβλημα ή σενάριο που αντιμετωπίζει ένας πράκτορας. Οι εργασίες κατηγοριοποιούνται με βάση τα χαρακτηριστικά και τη δομή τους σε τρεις κατηγορίες.

Επεισοδιακή Εργασία (Episodic Task)

Στην επεισοδιακή εργασία, η αλληλεπίδραση πράκτορα-περιβάλλοντος χωρίζεται σε επεισόδια με καθορισμένο σημείο έναρξης και λήξης. Κάθε επεισόδιο ξεκινά από μια αρχική κατάσταση και τελειώνει όταν επιτευχθεί ένας στόχος ή συμβεί μια αποτυχία.

Παράδειγμα: Ένα παιχνίδι σκακιού όπου κάθε παρτίδα αποτελεί ένα επεισόδιο. Ο πράκτορας (παίκτης) ξεκινά από την αρχική διάταξη και συνεχίζει μέχρι την κίνηση ματ ή την ισοπαλία.

Συνεχείς Εργασία (Continuous Task)

Στην συνεχή εργασία, η αλληλεπίδραση πράκτορα-περιβάλλοντος είναι αδιάκοπη χωρίς σαφή τελικά σημεία. Ο πράκτορας λειτουργεί συνεχώς, επιδιώκοντας τη μεγιστοποίηση της ανταμοιβής σε έναν απεριόριστο χρονικό ορίζοντα, μέχρι την διακοπή του από εξωγενή παράγοντα.

Παράδειγμα: Ένας αλγόριθμος Ενισχυτικής Μάθησης για επενδύσεις σε μετοχές που λειτουργεί καθημερινά, αναλύοντας την αγορά και επιλέγοντας σε ποια μετοχή να επενδύσει ή ποια να πουλήσει. Το περιβάλλον (χρηματιστήριο) εξελίσσεται συνεχώς, και ο πράκτορας δεν τερματίζει αλλά λειτουργεί σε διαρκή βάση, προσπαθώντας να μεγιστοποιήσει το συνολικό κέρδος μέσω συνεχόμενων αποφάσεων.

Προβλήματα Πολλαπλών Βραχιόνων (Multi-Armed Bandit Problems)

Σε αυτά τα προβλήματα, ο πράκτορας επιλέγει από ένα σύνολο ενεργειών χωρίς να επηρεάζει μελλοντικές καταστάσεις. Ο στόχος είναι η εξισορρόπηση μεταξύ εξερεύνησης (δοκιμή νέων ενεργειών) και εκμετάλλευσης (επιλογή της καλύτερης γνωστής ενέργειας) για τη μεγιστοποίηση της συνολικής ανταμοιβής.

Παράδειγμα: Ένας αλγόριθμος που επιλέγει ποια διαφήμιση να προβάλλει σε έναν ιστότοπο, βασισμένος σε προηγούμενα δεδομένα κλικ για να μεγιστοποιήσει την πιθανότητα αλληλεπίδρασης.

3.2.2 Το Δίλημμα Εξερεύνησης/Εκμετάλλευσης

Ένα από τα πιο σημαντικά θεωρητικά προβλήματα στην Ενισχυτική Μάθηση είναι η ισορροπία μεταξύ **εξερεύνησης** (exploration) και **εκμετάλλευσης** (exploitation). Ο πράκτορας, για να μεγιστοποιήσει τη συνολική ανταμοιβή του, πρέπει να αποφασίζει συνεχώς αν θα εκμεταλλευτεί τις ενέργειες που γνωρίζει ότι έχουν αποφέρει θετικά αποτελέσματα, ή αν θα εξερευνήσει καινούργιες ενέργειες που ενδέχεται να αποφέρουν ακόμη μεγαλύτερη απόδοση. Η εκμετάλλευση οδηγεί σε άμεσο κέρδος, καθώς βασίζεται στην υπάρχουσα γνώση, αλλά δεν επιτρέπει την ανακάλυψη πιθανώς καλύτερων επιλογών. Αντίθετα, η εξερεύνηση μπορεί αρχικά να αποδώσει χαμηλότερες ανταμοιβές, ωστόσο είναι απαραίτητη για την αποδοτικότερη εκμάθηση σε βάθος χρόνου.

Ένα απλό παράδειγμα εξερεύνησης/εκμετάλλευσης είναι και το δίλλημα επιλογής εστιατορίου. Έστω ότι ένας αλγόριθμος έχει εκπαιδευτεί για να προτείνει το καλύτερο εστιατόριο σε έναν χρήστη, με βάση τις προτιμήσεις του.

- **Εκμετάλλευση:** Εάν ο αλγόριθμος γνωρίζει ήδη ότι ένα εστιατόριο έχει συγκεντρώσει υψηλές βαθμολογίες στο παρελθόν για χρήστες με παρόμοιες προτιμήσεις, τότε θα συνεχίσει να το προτείνει. Αυτό μεγιστοποιεί τη βραχυπρόθεσμη ανταμοιβή.
- **Εξερεύνηση:** Ωστόσο, αν υπάρχει ένα νέο εστιατόριο που δεν έχει προταθεί αρκετές φορές, ο αλγόριθμος μπορεί να αποφασίσει να το προτείνει σε ορισμένους χρήστες ώστε να συλλέξει πληροφορίες. Η επιλογή αυτή ενδέχεται να μην οδηγήσει στην μεγαλύτερη βραχυπρόθεσμη ανταμοιβή, αλλά μπορεί μακροπρόθεσμα να αποδειχθεί πολύ καλύτερη πρόταση από την προηγούμενη.

Η μεγάλη πρόκληση είναι ότι καμία από τις δύο στρατηγικές δεν είναι επαρκής από μόνη της. Η εκμετάλλευση οδηγεί συχνά σε τοπικά βέλτιστα, ενώ η υπερβολική εξερεύνηση καθυστερεί την επίτευξη καλών αποτελεσμάτων. Επομένως, απαιτείται μια δυναμική ισορροπία, η οποία επιτρέπει στον πράκτορα να αξιοποιεί όσα γνωρίζει, ενώ παράλληλα αφήνει περιθώριο για συνεχή βελτίωση μέσω δοκιμών.

3.2.3 Μέθοδοι Επίλυσης Προβλημάτων Ενισχυτικής Μάθησης

Η επίλυση προβλημάτων στην Ενισχυτική Μάθηση βασίζεται στην αναζήτηση μιας πολιτικής π , η οποία μεγιστοποιεί τη σωρευτική ανταμοιβή.

Η Πολιτική π

Στην Ενισχυτική Μάθηση η πολιτική π , είναι η στρατηγική που ακολουθεί ο πράκτορας για να επιλέγει ενέργειες βάσει των καταστάσεων στις οποίες βρίσκεται. Η πολιτική π είναι ο πυρήνας της συμπεριφοράς του πράκτορα. Ορίζει το πώς να ενεργεί σε κάθε

δεδομένη στιγμή, με σκοπό τη μεγιστοποίηση της σωρευτικής αναμενόμενης ανταμοιβής. Η πολιτική μπορεί να είναι:

- **Ντετερμινιστική (deterministic):**

$$\pi(s) = a,$$

η πολιτική επιλέγει μία συγκεκριμένη ενέργεια a για κάθε κατάσταση s .

- **Στοχαστική (stochastic):**

$$\pi(a | s) = P(At = a | St = s),$$

η πολιτική καθορίζει την πιθανότητα να επιλεγεί η ενέργεια a όταν ο πράκτορας βρίσκεται στην κατάσταση s .

Για την εκπαίδευση του πράκτορα με σκοπό την εύρεση της βέλτιστης πολιτικής πέχουν αναπτυχθεί διάφορες μέθοδοι, καθεμία με τα δικά της πλεονεκτήματα και μειονεκτήματα. Οι κύριες προσεγγίσεις περιλαμβάνουν τις **value-based** μεθόδους, όπου ο πράκτορας μαθαίνει έμμεσα ποια ενέργεια να επιλέξει μέσω της εκτίμησης συναρτήσεων αξίας (value function), τις **policy-based** μεθόδους, που επικεντρώνονται στην άμεση εκμάθηση της συνάρτησης πολιτικής (policy function), και τις **actor-critic** μεθόδους, οι οποίες συνδυάζουν τα χαρακτηριστικά των δύο προηγούμενων προσεγγίσεων. Η επιλογή της κατάλληλης μεθόδου εξαρτάται από τη φύση του προβλήματος και το περιβάλλον δράσης.

Value-Based Μέθοδοι

Στις value-based μεθόδους, ο πράκτορας μαθαίνει μια **συνάρτηση αξίας** (value function), η οποία εκτιμά το πόσο ωφέλιμο είναι να βρίσκεται σε μια συγκεκριμένη κατάσταση ή να εκτελεί μια συγκεκριμένη δράση. Η συνάρτηση αξίας $V_\pi(s)$ βοηθά τον πράκτορα να εκτιμήσει ποια κατάσταση έχει μεγαλύτερη αξία, με βάση τις προσδοκώμενες ανταμοιβές που θα λάβει στο μέλλον αν ακολουθήσει μια συγκεκριμένη πολιτική π . Στις value-based μεθόδους, ο στόχος είναι να μάθουμε αυτές τις τιμές για όλες τις καταστάσεις, ώστε ο πράκτορας να επιλέγει τις ενέργειες που οδηγούν στις καταστάσεις με το μεγαλύτερο κέρδος. Ο ορισμός της συνάρτησης αξίας $V_\pi(s)$:

$$V_\pi(s) = E_\pi[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots | S_t = s],$$

όπου:

- $V_\pi(s)$: η συνάρτηση αξίας της κατάστασης s κάτω από την πολιτική π ,
- $E_\pi[*]$: η αναμενόμενη τιμή (expected value).

Policy-Based Μέθοδοι

Στις policy-based μεθόδους, ο πράκτορας μαθαίνει απευθείας την συνάρτηση πολιτικής (policy function) η οποία ορίζει ποια ενέργεια πρέπει να επιλεγεί σε κάθε δεδομένη κατάσταση. Σε αντίθεση με τις value-based μεθόδους, οι οποίες μαθαίνουν έμμεσα την πολιτική μέσω της εκτίμησης συναρτήσεων αξίας, οι policy-based μέθοδοι μοντελοποιούν και βελτιστοποιούν απευθείας την πολιτική.

Actor-Critic Μέθοδοι

Οι actor-critic μέθοδοι αποτελούν μια υβριδική προσέγγιση στην Ενισχυτική Μάθηση, συνδυάζοντας τα βασικά στοιχεία τόσο των policy-based όσο και των value-based αλγορίθμων. Ο βασικός στόχος τους είναι να επωφεληθούν από τα πλεονεκτήματα κάθε μεθόδου, ενώ παράλληλα μετριάζουν τα μειονεκτήματά τους.

Οι actor-critic μέθοδοι βασίζονται στην ύπαρξη δύο διαφορετικών λειτουργικών μονάδων [16]:

- **Actor:**
Είναι υπεύθυνος για την πολιτική $\pi(a | s)$, δηλαδή αποφασίζει ποια ενέργεια να εκτελεστεί σε κάθε κατάσταση. Συνήθως μοντελοποιείται μέσω ενός νευρωνικού δικτύου και εκπαιδεύεται ώστε να μεγιστοποιήσει την αναμενόμενη απόδοση.
- **Critic:**
Αξιολογεί τις ενέργειες του actor υπολογίζοντας τη συνάρτηση αξίας $V_\pi(s)$. Παρέχει ανατροφοδότηση στον actor σχετικά με το πόσο “κερδοφόρες” είναι οι ενέργειες που επιλέχθηκαν.

Η μέθοδος actor-critic προσφέρει έναν αποτελεσματικό και ευέλικτο μηχανισμό εκμάθησης και αποτελεί πλέον θεμελιώδης επιλογή σε σύγχρονες εφαρμογές Ενισχυτικής Μάθησης.

Η Ενισχυτική Μάθηση μπορεί να κατηγοριοποιηθεί όχι μόνο με βάση τι μαθαίνει ο πράκτορας, όπως στις value-based, policy-based και actor-critic μεθόδους, αλλά και με βάση το πώς μαθαίνει, δηλαδή αν αξιοποιεί ή όχι μοντέλο του περιβάλλοντος. Η διάκριση αυτή οδηγεί σε δύο θεμελιώδεις κατευθύνσεις: την **model-free** και την **model-based** Ενισχυτική Μάθηση.

Model-Free Ενισχυτική Μάθηση

Στην Model-Free Ενισχυτική Μάθηση, ο πράκτορας μαθαίνει απευθείας από την εμπειρία του, χωρίς να κατασκευάζει ή να χρησιμοποιεί ένα μοντέλο του περιβάλλοντος. Δηλαδή, δεν επιχειρεί να προβλέψει τις μελλοντικές καταστάσεις ή ανταμοιβές, αλλά βασίζεται αποκλειστικά στις παρατηρήσεις και τις ανταμοιβές που λαμβάνει μέσω της αλληλεπίδρασης με το περιβάλλον. Παρόλο που ο πράκτορας μαθαίνει μέσω δοκιμής και σφάλματος χωρίς να απαιτείται γνώση ή εκτίμηση μοντέλου του περιβάλλοντος, συχνά απαιτεί μεγάλο αριθμό δειγμάτων-επεισοδίων για αποτελεσματική μάθηση.

Model-Based Ενισχυτική Μάθηση

Στην Model-Based Ενισχυτική Μάθηση, ο πράκτορας κατασκευάζει ή χρησιμοποιεί ένα μοντέλο του περιβάλλοντος, το οποίο περιγράφει ουσιαστική το πως οι ενέργειες που εκτελεί ο πράκτορας επηρεάζουν τις μελλοντικές καταστάσεις και ανταμοιβές. Αυτό το μοντέλο μπορεί να είναι εκ των προτέρων γνωστό ή να μαθαίνεται κατά την διάρκεια της εκμάθησης. Ο πράκτορας χρησιμοποιεί το μοντέλο για να προσομοιώσει μελλοντικές καταστάσεις και να σχεδιάσει τον τρόπο με τον οποίο ενεργεί μέσα στο περιβάλλον.

3.2.4 Online/Offline Εκμάθηση

Η διάκριση μεταξύ **online** και **offline learning** αφορά τον τρόπο με τον οποίο συλλέγονται και χρησιμοποιούνται τα δεδομένα από τον πράκτορα κατά τη διαδικασία εκπαίδευσης. Κάθε μία από τις προσεγγίσεις μπορεί να υλοποιηθεί σε διαφορετικές εφαρμογές ανάλογα με το είδος και τις ανάγκες τους.

Online Learning

Στην online εκμάθηση, ο πράκτορας εκπαιδεύεται σε πραγματικό χρόνο κατά τη διάρκεια της αλληλεπίδρασής του με το περιβάλλον. Οι παρατηρήσεις, οι ενέργειες και οι ανταμοιβές που προκύπτουν χρησιμοποιούνται άμεσα για την ενημέρωση της πολιτικής ή των συναρτήσεων αξίας. Παρόλο που η συγκεκριμένη προσέγγιση μπορεί να προσφέρει υψηλή προσαρμοστικότητα σε δυναμικά περιβάλλοντα μέσω της άμεσης ανατροφοδότησης, σε πραγματικά συστήματα όπου δεν υπάρχει η δυνατότητα προεκπαίδευσης σε εικονικό περιβάλλον, ενδέχεται να αποβεί επικίνδυνη και κοστοβόρα.

Offline Learning (Batch Reinforcement Learning)

Η offline εκμάθηση ή αλλιώς batch reinforcement learning, βασίζεται σε προηγουμένως συλλεχθέντα δεδομένα. Ο πράκτορας εκπαιδεύεται χωρίς να έχει πρόσβαση σε πραγματικό χρόνο στο περιβάλλον, χρησιμοποιώντας ένα καταγεγραμμένο dataset με προκαθορισμένες καταστάσεις, ενέργειες και ανταμοιβές. Η offline εκμάθηση είναι κατάλληλη για επικίνδυνες, ευαίσθητες και δαπανηρές εφαρμογές καθώς προσφέρει ασφάλεια και υψηλή απόδοση σε ανάλογα περιβάλλοντα [17].

3.2.5 Monte Carlo vs Temporal Difference Learning

Μέσω της αλληλεπίδρασης με το περιβάλλον ο πράκτορας ανανεώνει την συνάρτηση αξίας ή την πολιτική του. Δύο διαδεδομένες στρατηγικές που καθορίζουν το πως ανανεώνονται οι συναρτήσεις αξίας και πολιτικής είναι οι Monte Carlo και η Temporal Difference. Έστω ότι $V_\pi(s)$ μία συνάρτηση αξίας.

Monte Carlo

Η μέθοδος **Monte Carlo** ανανεώνει την συνάρτηση αξίας $V_\pi(s)$ στο τέλος του επεισοδίου, αφού πρώτα συλλέξει τις ανταμοιβές και υπολογίσει το συνολικό κέρδος G_t [18]. Ο πράκτορας εκτελεί την ίδια πολιτική μέχρι το πέρας ενός επεισοδίου και στη συνέχεια χρησιμοποιεί το συνολικό κέρδος (return) που έλαβε, για να εκτιμήσει τη συνάρτηση αξίας των καταστάσεων που επισκέφθηκε. Η νέα τιμή τη συνάρτησης αξίας $V_\pi(s)$ υπολογίζεται ως εξής:

$$V_\pi(s) \leftarrow V_\pi(s) + \alpha[G_t - V_\pi(s)],$$

όπου G_t το συνολικό κέρδος.

Temporal Difference

Η μέθοδος **Temporal Difference (TD)** ανανεώνει την συνάρτηση αξίας $V_\pi(s)$ σε κάθε χρονικό βήμα. Σε αυτή την προσέγγιση, η εκτίμηση της αξίας βασίζεται όχι στο συνολικό κέρδος, αλλά σε προβλέψεις της επόμενης κατάστασης και ανταμοιβής σε κάθε βήμα, οι οποίες χρησιμοποιούνται άμεσα για την ενημέρωση της συνάρτησης αξίας, χωρίς να χρειάζεται να ολοκληρωθεί το επεισόδιο [19]. Η νέα τιμή τη συνάρτησης αξίας $V_\pi(s)$ υπολογίζεται ως εξής:

$$V_\pi(s_t) \leftarrow V_\pi(s_t) + \alpha[R_{t+1} + \gamma V_\pi(s_{t+1}) - V_\pi(s_t)].$$

Το κύριο πρόβλημα της μεθόδου TD είναι ότι οι ενημερώσεις των βημάτων τους είναι biased από τις αρχικές συνθήκες των παραμέτρων μάθησης, καθώς στην αρχή της μάθησης αυτές οι εκτιμήσεις δεν περιέχουν καμία πληροφορία από πραγματικές ανταμοιβές ή μεταβάσεις καταστάσεων. Αντιθέτως, η μέθοδος Monte Carlo δεν επηρεάζεται από bias, καθώς κάθε ενημέρωση γίνεται χρησιμοποιώντας ένα δείγμα ενός ολόκληρου επεισοδίου. Ωστόσο, στην μέθοδο Monte Carlo συχνά προκύπτει το φαινόμενο υψηλή διακύμανση (variance), πράγμα που σημαίνει ότι απαιτούνται περισσότερα δείγματα για να επιτευχθεί υψηλή απόδοση σε σύγκριση με την μέθοδο TD.

3.2.6 Αλγόριθμος Q-Learning

Ο αλγόριθμος **Q-Learning** αποτελεί έναν από τους πιο θεμελιώδεις και ευρέως χρησιμοποιούμενους αλγορίθμους Ενισχυτικής Μάθησης. Εντάσσεται στην κατηγορία των model-free value-based αλγορίθμων, ενώ επίσης είναι και ένας από τους

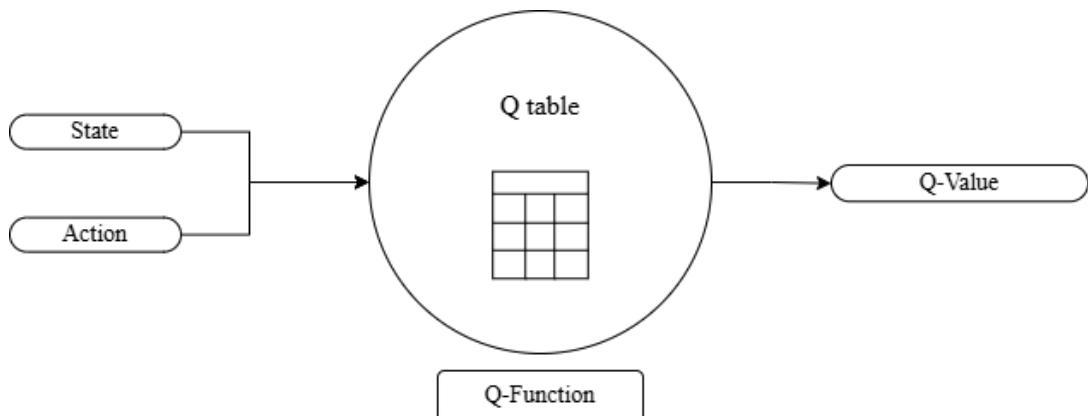
χαρακτηριστικότερους off-policy αλγορίθμους. Οι αλγόριθμοι μάθησης off-policy χρησιμοποιούν μία πολιτική ενημέρωσης (updating policy) η οποία είναι διαφορετική από την πολιτική που χρησιμοποιείται για την επιλογή των δράσεων του πράκτορα (action policy). Στον συγκεκριμένο αλγόριθμο ως πολιτική ενημέρωσης χρησιμοποιείται Greedy πολιτική ενώ ως πολιτική δράσης χρησιμοποιείται η Epsilon Greedy πολιτική (ϵ -greedy). Ο αλγόριθμος Q-Learning (Εικόνα 11) παρουσιάστηκε αρχικά από τον Christopher Watkins [20] το 1989 και στοχεύει στην εκμάθηση της βέλτιστης συνάρτησης αξίας δράσης (q -function) $Q(s, a)$, η οποία εκφράζει τη μέγιστη αναμενόμενη σωρευτική ανταμοιβή που μπορεί να επιτύχει ένας πράκτορας αν επιλέξει μια ενέργεια a σε μια κατάσταση s , και στη συνέχεια ακολουθήσει την καλύτερη δυνατή πολιτική. Ο αλγόριθμος Q-Learning χρησιμοποιεί τον ακόλουθο τύπο για να ενημερώσει την τιμή $Q(s_t, a_t)$ για ένα ζεύγος κατάστασης-δράσης:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[R_{t+1} + \gamma \max_a Q(s_{t+1}, a_t) - Q(s_t, a_t)],$$

όπου $\max_a Q(s_{t+1}, a_t)$ η Greedy πολιτική ενημέρωσης.

Ο πράκτορας οδηγείται στην βέλτιστη συνάρτηση αξίας δράσης ενημερώνοντας έναν πίνακα τιμών **Q-table**, όπου κάθε τιμή του είναι ένα ζεύγος δράσεων-καταστάσεων το οποίο αντιτροσωπεύει την αξία της εκτέλεσης μιας συγκεκριμένης ενέργειας σε μια κατάσταση. Χρησιμοποιείται στον αλγόριθμο Q-Learning για την αποθήκευση των τιμών εξόδου της συνάρτησης αξίας δράσης $Q(s_t, a_t)$. Πιο συγκεκριμένα, πρόκειται για έναν πίνακα δύο διαστάσεων όπου:

- Κάθε γραμμή αντιστοιχεί σε μια κατάσταση s_t ,
- Κάθε στήλη αντιστοιχεί σε μια δράση a_t ,



Εικόνα 11: Ο αλγόριθμος Q-Learning

Ακολουθεί ανάλυση των βημάτων του αλγορίθμου Q-Learning:

Βήμα 1º: Αρχικοποίηση του Q-table. Το πρώτο βήμα του αλγορίθμου είναι η αρχικοποίηση του πίνακα Q-table. Συνήθως χρησιμοποιούνται αυθαίρετες τιμές (π.χ. μηδενικά ή τυχαίοι αριθμοί).

Βήμα 2^ο: Παρατήρηση του περιβάλλοντος. Ο πράκτορας παρατηρεί την κατάσταση του περιβάλλοντος.

Βήμα 3^ο: Επιλογή δράσης. Η επιλογή της δράσης γίνεται με βάση την πολιτική δράσης. Χρησιμοποιείται η πολιτική ϵ -greedy, σύμφωνα με την οποία:

- Με πιθανότητα $1 - \epsilon$: γίνεται εκμετάλλευση (ο πράκτορας επιλέγει την ενέργεια με την υψηλότερη τιμή ζεύγους κατάστασης-δράσης).
- Με πιθανότητα ϵ : γίνεται εξερεύνηση (πραγματοποιείται τυχαία δράση).

Στην αρχή της εκπαίδευσης η τιμή του ϵ είναι ίση με 1, συνεπώς είναι πολύ μεγάλη η πιθανότητα εξερεύνησης. Με την πάροδο του χρόνου η τιμή του ϵ μειώνεται σταδιακά δίνοντας στον πράκτορα την δυνατότητα να εκμεταλλεύεται τις ενέργειες που γνωρίζει με βάση τον Q-table ότι οδηγούν στις μέγιστες ανταμοιβές.

Βήμα 4^ο: Εκτέλεση δράσης, ανταμοιβή, επόμενη κατάσταση. Ο πράκτορας εκτελεί την δράση που επέλεξε και παρατηρεί την ανταμοιβή που έλαβε από το περιβάλλον, καθώς προχωράει στην επόμενη κατάσταση.

Βήμα 5^ο: Ενημέρωση της q-function. Η συνάρτηση αξίας δράσης $Q(s_t, a_t)$ ενημερώνεται σύμφωνα με τον τύπο:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[R_{t+1} + \gamma \max_a Q(s_{t+1}, a_t) - Q(s_t, a_t)].$$

Όταν ολοκληρωθεί η ενημέρωση της τιμής Q στον πίνακα Q-table, ο πράκτορας ξεκινάει εκ νέου σε μια νέα κατάσταση, επαναλαμβάνει τα βήματα 2-5 μέχρι ο αλγόριθμος να φτάσει σε μία κατάσταση τερματισμού. Στην Εικόνα 12 παρουσιάζεται ο ψευδοκώδικας του αλγορίθμου Q-Learning.

Algorithm parameters: step size $\alpha \in (0,1]$, small $\epsilon > 0$

Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+$, $a \in \mathcal{A}(s)$, arbitrarily except that $Q(\text{terminal}, \cdot) = 0$

Loop for each episode:

 Initialize S

 Loop for each step of episode:

 Choose A from S using policy derived from Q (e.g., ϵ -greedy)

 Take action A , observe R, S'

$Q(S, A) \leftarrow Q(S, A) + \alpha[R + \gamma \max_a Q(S', a) - Q(S, A)]$

$S \leftarrow S'$

 Until S is terminal

Εικόνα 12: Ψευδοκώδικας αλγορίθμου Q-Learning

3.3 Βαθιά Ενισχυτική Μάθηση

Η DRL είναι ένα πεδίο της Ενισχυτικής Μάθησης το οποίο βρίσκεται στο επίκεντρο της ανάπτυξης τα τελευταία χρόνια. Αυτό οφείλεται στην ικανότητά της να αποδίδει εξαιρετικά σε προβλήματα που απασχολούν ολοένα και περισσότερο τον τομέα της τεχνολογίας. Πρόκειται στην ουσία για τον συνδυασμό της Ενισχυτικής Μάθησης με τη Βαθιά Μάθηση, συνδυασμός που έχει μεγάλες προοπτικές στην εκπαίδευση πρακτόρων όπου η παραδοσιακή Ενισχυτική Μάθηση δεν είναι εφικτό να αποδώσει ικανοποιητικά. Τέτοια παραδείγματα αφορούν εφαρμογές σε πολυδιάστατους χώρους εκπαιδεύσεως, όπου η πληροφορία προς επεξεργασία είναι πολυδιάστατη και πολύπλοκη, όπως εικόνες και δεδομένα από διάφορους αισθητήρες.

Η Βαθιά Μάθηση ενσαρκώνει τον ρόλο του εκτιμητή συναρτήσεων (function approximator) στην Ενισχυτική Μάθηση. Πιο συγκεκριμένα, σε value-based μεθόδους, ένα βαθύ νευρωνικό δίκτυο χρησιμοποιείται για να προσεγγίσει τη συνάρτηση $Q(s_t, a_t)$. Αυτό καθιστά δυνατή τη χρήση Ενισχυτικής Μάθησης σε εφαρμογές όπου ο πίνακας Q θα ήταν αδύνατο να αποθηκευτεί λόγω του τεράστιου χώρου καταστάσεων. Αντίστοιχα, στις policy-based ή actor-critic μεθόδους, το βαθύ δίκτυο χρησιμοποιείται για να μάθει απευθείας μία πολιτική πλοιόγησης $\pi(s)$, η οποία χαρτογραφεί καταστάσεις σε πιθανότητες επιλογής δράσεων. Αυτό είναι κρίσιμο για την πλοιόγηση σε συνεχείς χώρους δράσεων, όπου οι παραδοσιακοί πίνακες πολιτικής δεν επαρκούν.

Η χρήση νευρωνικών δικτύων στην ανάπτυξη αλγορίθμων Ενισχυτικής Μάθησης άρχισε να εμφανίζεται στις αρχές της δεκαετίας του 1990, αλλά οι τεχνικές δυσκολίες κατά την εκπαίδευση περιόρισαν την εφαρμογή τους. Η καθοριστική καμπή για την νέα αυτή τεχνολογία ήρθε το 2013, όταν η ερευνητική ομάδα της DeepMind πρότεινε τον πρώτο επιτυχημένο συνδυασμό νευρωνικών δικτύων και Ενισχυτικής Μάθησης, με τη μορφή του αλγορίθμου Deep Q-Network (**DQN**) [21]. Με την χρήση του DQN εκπαιδεύτηκε ένας πράκτορας να παίζει παιχνίδια Atari απευθείας μέσω της εικόνας που λαμβάνει από την οθόνη, επιτυγχάνοντας ανθρώπινο ή ακόμα και υπεράνθρωπο επίπεδο απόδοσης σε πολλές περιπτώσεις.

Η DRL βρίσκεται εφαρμογές σε πολλές νέες τεχνολογίες. Οι κυριότεροι τομείς εφαρμογής είναι οι εξής:

- Πλοιόγηση ρομποτικών πλατφορμών σε δυναμικά περιβάλλοντα,
- Έλεγχος βιομηχανικών διαδικασιών,
- Αυτόνομα οχήματα,
- Συστήματα ενεργειακής διαχείρισης,
- Παιχνίδια στρατηγικής.

Ωστόσο, η εφαρμογή της DRL δεν περιορίζεται μόνο στους συγκεκριμένους τομείς, καθώς τα τελευταία χρόνια γίνεται χρήση της εντατικά σε ιατρικές (robotic surgeons) και στρατιωτικές (military drones, autonomous missiles, κα.) εφαρμογές. Παρακάτω πραγματοποιείται ανάλυση της εφαρμογής DRL στην πλοιήγηση ρομποτικών πλατφορμών.

3.3.1 DRL στην πλοιήγηση ρομποτικών πλατφορμών

Η DRL έχει επιφέρει σημαντικές εξελίξεις στον τομέα της πλοιήγησης ρομποτικών συστημάτων τα τελευταία χρόνια, καθιστώντας δυνατή την αυτόνομη και ευέλικτη λήψη αποφάσεων σε πολύπλοκα και δυναμικά περιβάλλοντα. Η DRL επιτρέπει στα ρομπότ να εκπαιδεύονται απευθείας μέσω αλληλεπίδρασης με το περιβάλλον, αξιοποιώντας τον μεγάλο όγκο δεδομένων που λαμβάνει από διάφορους αισθητήρες. Η προσέγγιση αυτή υπερβαίνει τους περιορισμούς των κλασικών μεθόδων, όπως οι αλγόριθμοι SLAM, οι οποίοι βασίζονται στη χαρτογράφηση και προϋποθέτουν την επίγνωση του περιβάλλοντος δράσης.

Η DRL επιτρέπει στο ρομπότ να μαθαίνει από τις δικές του αλληλεπιδράσεις με το περιβάλλον, χωρίς να χρειάζεται εκ των προτέρων σχεδιασμένο μοντέλο ή γνώση του χώρου. Παράλληλα με την ενσωμάτωση νευρωνικών δικτύων, η DRL επιτρέπει ταχεία εκτίμηση δράσεων, μέσω της άμεσης και ευέλικτης διαχείρισης μεγάλων όγκων δεδομένων, κάνοντάς την κατάλληλη για πλοιήγηση σε πραγματικό χρόνο, ακόμα και σε περιπτώσεις με δυναμικά εμπόδια. Ταυτόχρονα, η ικανότητα των βαθιών δικτύων να αναγνωρίζουν μοτίβα και υψηλού επιπέδου χαρακτηριστικά, προσδίδει στους πράκτορες την δυνατότητα να μπορούν να προσαρμόζονται σε νέες καταστάσεις ή ακόμα και νέα περιβάλλοντα, χωρίς να έχουν εκπαιδευτεί εξειδικευμένα για κάθε μία ή κάθε ένα από αυτά. Τέλος, η DRL μπορεί να χρησιμοποιεί άμεσα δεδομένα εισόδου από κάμερες, LIDAR και άλλων αισθητήρων, χωρίς την ανάγκη για οποιαδήποτε είδους προεπεξεργασία, κάνοντας με αυτό τον τρόπο εφικτή την ανάπτυξη **end-to-end** λύσεων, ανεξάρτητων από την ανθρώπινη παρέμβαση.

3.3.2 Αρχιτεκτονικές και μοντέλα DRL

Στην συγκεκριμένη ενότητα θα γίνει λεπτομερής ανάλυση της αρχιτεκτονικής των σημαντικότερων και πιο διαδεδομένων αλγορίθμων DRL. Η ανάλυση θα περιλαμβάνει το θεωρητικό πλαίσιο, την αρχιτεκτονική του δικτύου, την δομή του αλγορίθμου καθώς και διάφορες τεχνικές που χρησιμοποιούνται στον κάθε αλγόριθμο.

3.3.2.1 Deep Q-Network

Ο αλγόριθμος **DQN** συνδυάζει τη λογική του Q-learning με την ισχύ των **συνελικτικών νευρωνικών δικτύων (CNNs)**. Ο στόχος είναι η εκμάθηση της συνάρτησης $Q(s, a)$, η

οποία εκτιμά την μέγιστη αναμενόμενη σωρευτική ανταμοιβή, και χρήση αυτής για την εξαγωγή της βέλτιστης πολιτικής:

$$\pi(s) = \operatorname{argmax} Q(s, a).$$

Η χρήση CNNs επιτρέπει στον πράκτορα να επεξεργάζεται τις εικόνες (π.χ. κάμερας ή οθόνης) και να αντιλαμβάνεται το περιβάλλον μέσα από αυτές. Η αρχιτεκτονική του αρχικού δικτύου DQN που χρησιμοποιήθηκε από την ομάδα της DeepMind για το παιχνίδι Atari είχε την εξής μορφή [21]:

- Είσοδος: Stack 4 εικόνων 84x84x4
- Συνελικτικά στρώματα:
 1. 32 φίλτρα 8x8, stride = 4, Activation Function = ReLU
 2. 64 φίλτρα 4x4, stride = 2, Activation Function = ReLU
 3. 64 φίλτρα 3x3, stride = 1, Activation Function = ReLU
- Πλήρως συνδεδεμένο στρώμα: 512 νευρώνες + Activation Function = ReLU
- Έξοδος: μία τιμή Q για κάθε πιθανή δράση

Στον αλγόριθμο DQN γίνεται χρήση μίας τεχνικής σταθεροποίησης της εκπαίδευσης η οποία ονομάζεται **Experience Replay Buffer**. Στην ουσία οι τιμές (s, a, r, s') αποθηκεύονται σε μία προσωρινή μνήμη και ανακτώνται τυχαία σε mini-batches, ώστε να μειωθεί η συσχέτιση των δειγμάτων και να ενισχυθεί η σταθερότητα της εκπαίδευσης. Ως Loss Function χρησιμοποιείται μία συνάρτηση βασισμένη στην MSE. Στην Εικόνα 13 παρουσιάζεται ο ψευδοκώδικας του αλγορίθμου DQN.

- Loss Function:

$$L_i(\theta_i) = \mathbb{E}_{s,a \sim p(\cdot)} \left[(r + \gamma \max_{a'} Q(s', a'; \theta_{i-1}) - Q(s, a; \theta_i))^2 \right].$$

```

Initialize replay memory  $\mathcal{D}$  to capacity  $N$ 
Initialize action-value function  $Q$  with random weights
for episode = 1,  $M$  do
    Initialise sequence  $s_1 = \{x_1\}$  and preprocessed sequenced  $\varphi_1 = \varphi(s_1)$ 
    for  $t = 1, T$  do
        With probability  $\epsilon$  select a random action  $a_t$ 
        otherwise select  $a_t = \max_a Q^*(\varphi(s_t), a; \theta)$ 
        Execute action  $a_t$  in emulator and observe reward  $r_t$  and image  $x_{t+1}$ 
        Set  $s_{t+1} = s_t, a_t, x_{t+1}$  and preprocess  $\varphi_{t+1} = \varphi(s_{t+1})$ 
        Store transition  $(\varphi_t, a_t, r_t, \varphi_{t+1})$  in  $\mathcal{D}$ 
        Sample random minibatch of transitions  $(\varphi_j, a_j, r_j, \varphi_{j+1})$  from  $\mathcal{D}$ 
        Set  $y_j = \begin{cases} r_j & \text{for terminal } \varphi_{j+1} \\ r_j + \gamma \max_{a'} Q(\varphi_{j+1}, a'; \theta) & \text{for non-terminal } \varphi_{j+1} \end{cases}$ 
        Perform a gradient descent step on  $(y_i - Q(\varphi_j, a_j; \theta))^2$  according to equation 3
    end for
end for

```

Εικόνα 13: Ψευδοκώδικας αλγορίθμου DQN

3.3.2.2 Deep Deterministic Policy Gradient

Ο **DDPG** [22] επεκτείνει την ιδέα του Deterministic Policy Gradient (DPG) [23] σε χώρους συνεχών δράσεων, υιοθετώντας την δομή actor–critic ενός off-policy αλγορίθμου, σε συνδυασμό με στοιχεία του DQN. Η αρχιτεκτονική του δικτύου που χρησιμοποιήθηκε έχει ως εξής:

Για το δίκτυο του actor:

- Είσοδος: κατάσταση s
- Layer 1: fully connected layer με 400 νευρώνες, Activation Function = ReLU
- Layer 2: fully connected layer με 400 νευρώνες, Activation Function = ReLU
- Έξοδος: χρήση της συνάρτησης tanh για τον περιορισμό των τιμών εξόδου που αντιστοιχούν στις δράσεις

Για το δίκτυο του critic:

- Είσοδος: συνδυασμός κατάστασης s και δράσης a
- Layer 1: fully connected layer με 400 νευρώνες, Activation Function = ReLU
- Layer 2: fully connected layer με $400 + (\text{action size})$ νευρώνες, Activation Function = ReLU
- Output Layer: γραμμική έξοδος για εκτίμηση της $Q(s, a)$.

Η μέθοδος DDPG χρησιμοποιεί μια συνάρτηση δράσης, η οποία συμβολίζεται ως $\mu(s|\theta^\mu)$ και ορίζει την τρέχουσα πολιτική, αναθέτοντας μια συγκεκριμένη δράση σε κάθε κατάσταση. Το δίκτυο του critic χρησιμοποιεί την εξίσωση Bellman, παρόμοια με την προσέγγιση που χρησιμοποιείται και στο Deep Q-Network (DQN).

Ο αλγόριθμος DQN προκαλεί συχνά αστάθεια, επειδή το ίδιο δίκτυο χρησιμοποιείται και για την ενημέρωση και για τον υπολογισμό των στόχων (targets), οδηγώντας το μοντέλο σε απόκλιση. Αυτό διορθώνεται με τη χρήση μιας “soft” μεθόδου ενημέρωσης **στόχων (soft updates)**, προσαρμόζοντας ξεχωριστά δίκτυα στόχου (target networks) αντίγραφα των actor ($\mu(s|\theta^\mu)$) και critic ($Q(s, a|\theta^Q)$) δικτύων, με αργό ρυθμό αντί για την άμεση αντιγραφή των βαρών. Αποτέλεσμα αυτού είναι η βελτίωση της σταθερότητας των actor-critic μοντέλων. Τα βάρη $\theta^{Q'}$ και $\theta^{\mu'}$ των target networks ενημερώνονται ως εξής:

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

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

όπου τ η παράμετρος ρύθμισης των soft updates των target networks.

Στον χώρο των συνεχών δράσεων δεν υπάρχουν διακριτά σύνολα ενεργειών ώστε να γίνει με εύκολο τρόπο η αξιολόγηση και η επιλογή τους. Οι off-policy μέθοδοι, όπως ο DDPG, προσφέρουν τη δυνατότητα διαχωρισμού της εξερεύνησης από τη διαδικασία της μάθησης, επιτρέποντας την προσθήκη θορύβου στην πολιτική του agent με τρόπο ανεξάρτητο από το μοντέλο μάθησης. Συγκεκριμένα, τροποποιείται η πολιτική δράσης με την προσθήκη στοχαστικού θορύβου από κάποια στοχαστική διεργασία \mathcal{N} , ώστε η νέα πολιτική να γίνεται: $\mu' = \mu + \mathcal{N}$. Στον DDPG, επιλέγεται η διαδικασία **Uhlenbeck-Ornstein** [24], κατά την οποία παράγεται θόρυβος με χρονική συσχέτιση. Η συγκεκριμένη μορφή θορύβου είναι κατάλληλη για φυσικά περιβάλλοντα με αδράνεια (inertia), όπως ρομποτικά συστήματα, διότι συνεισφέρει στην ομαλή και ρεαλιστική εξερεύνηση στον συνεχή χώρο δράσεων, βελτιώνοντας έτσι την απόδοση του πράκτορα σε δυναμικά προβλήματα.

Στην Εικόνα 14 παρουσιάζεται ο ψευδοκώδικας του αλγορίθμου DDPG.

```

Randomly initialize critic network  $Q(s, a | \theta^Q)$  and actor  $\mu(s | \theta^\mu)$  with weights  $\theta^Q$  and  $\theta^\mu$ .
Initialize target network  $Q'$  and  $\mu'$  with weights  $\theta^{Q'} \leftarrow \theta^Q, \theta^{\mu'} \leftarrow \theta^\mu$ 
Initialize replay buffer  $\mathcal{R}$ 
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  $a_t = \mu(s | \theta^\mu) + \mathcal{N}_t$  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  $\mathcal{R}$ 
        Sample a random minibatch of  $N$  transitions  $(s_t, a_t, r_t, s_{t+1})$  from  $\mathcal{R}$ 
        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

```

Εικόνα 14: Ψευδοκώδικας αλγορίθμου DDPG

3.3.2.3 Twin Delayed Deep Deterministic Policy Gradient

Ο TD3 αποτελεί εξέλιξη του DDPG με στόχο την αντιμετώπιση μιας βασικής αδυναμίας, την υπερεκτίμηση των Q-τιμών. Η μέθοδος συνδυάζει τρεις κρίσιμες τεχνικές [25]:

1. **Clipped Double Q-learning**, δηλαδή χρήση δύο critics και επιλογή της μικρότερης από τις δύο Q-τιμές. Ο αλγόριθμος Double Q-learning [26] εισήγαγε την χρήση του value network αντί του target network για την ανάπτυξη της πολιτικής.

$$y = r + \gamma Q_{\theta'}(s', \pi_\varphi(s')).$$

Ωστόσο παρατηρήθηκε ότι ο κριτής σε πολλές περιπτώσεις υπερεκτιμά την πραγματική αξία. Έτσι στον TD3 χρησιμοποιείται μία παραλλαγή της μεθόδου με

ένα ζεύγος από actors ($\pi_{\varphi_1}, \pi_{\varphi_2}$) και critics ($Q_{\theta_1}, Q_{\theta_2}$), όπου κάθε φορά επιλέγεται η μικρότερη τιμή εκ των δύο:

$$y_1 = r + \gamma Q_{\theta_1'}(s', \pi_{\varphi_1}(s')) ,$$

$$y_2 = r + \gamma Q_{\theta_2'}(s', \pi_{\varphi_2}(s')) ,$$

$$y \leftarrow r + \gamma \min_{i=1,2} Q_{\theta_i'}(s', \tilde{a}) .$$

2. **Delayed policy update**, όπου ο actor ενημερώνεται λιγότερο συχνά από τους critics, μειώνοντας τον αριθμό των λαθών σε κάθε ενημέρωση. Έτσι βελτιώνεται η σταθερότητα, ενώ παράλληλα μειώνεται η επίδραση των σφαλμάτων που προκαλούνται από τον critic.
3. **Target policy smoothing**, όπου προστίθεται μικρός, αποκομμένος (clipped) θόρυβος στις δράσεις του target actor, ώστε να ομαλοποιείται η συνάρτηση αξίας Q.

$$y = r + \gamma Q_{\theta'}(s', \pi_{\varphi'}(s') + \epsilon) ,$$

$$\epsilon \sim \text{clip}(\mathcal{N}(0, \sigma), -c, c) .$$

Ο θόρυβος ϵ αποκόπτεται έτσι ώστε να παραμείνουν οι τιμές κοντά στις αρχικές δράσεις.

Η βασική αρχιτεκτονική του δικτύου του αλγορίθμου TD3 είναι παρόμοια με αυτή του DDPG με την κύρια διαφορά μεταξύ τους να είναι η ύπαρξη δύο critic δικτύων στον TD3. Τα δύο critic δίκτυα είναι ανεξάρτητα μεταξύ τους. Σκοπός της χρήσης δύο critic δικτύων είναι η προσπάθεια αντιμετώπισης του overfitting επιλέγοντας την μικρότερη από τις τιμές Q-values όπως αναφέρθηκε και προηγουμένως.

Ο αλγόριθμος TD3 διακρίνεται στα εξής βήματα:

- Επιλογή δράσης:

$$a \sim \pi_{\varphi}(s) + \epsilon .$$

- Αποθήκευση των (s, a, r, s') στον Experienced Replay Buffer όπως αναφέρεται στην αρχιτεκτονική του DQN (3.3.2.1).
- Υπολογισμός Q-τιμών με τη χρήση Clipped Double Q-Learning:

$$\begin{aligned} \tilde{a} &\leftarrow \pi_{\varphi'}(s') + \epsilon , \quad \epsilon \sim \text{clip}(\mathcal{N}(0, \sigma), -c, c) , \\ y &\leftarrow r + \gamma \min_{i=1,2} Q_{\theta_i'}(s', \tilde{a}) . \end{aligned}$$

- Delayed policy update:

$$\nabla_{\varphi} J(\varphi) = N^{-1} \sum \nabla_a Q_{\theta_1}(s, a)|_{a=\pi_{\varphi}(s)} \nabla_{\varphi} \pi_{\varphi}(s).$$

Ο actor ενημερώνεται χρησιμοποιώντας τον μέσω όρο ενός mini-batch από τον Replay Buffer (άθροιση και διαίρεση διά το μέγεθος των δειγμάτων).

- Soft update των target networks όπως αναφέρεται στην αρχιτεκτονική του DDPG (3.3.2.2):

$$\begin{aligned}\theta'_i &\leftarrow \tau \theta_i + (1 - \tau) \theta'_i, \\ \varphi' &\leftarrow \tau \varphi + (1 - \tau) \varphi'.\end{aligned}$$

Στην Εικόνα 15 παρουσιάζεται ο ψευδοκώδικας του αλγορίθμου TD3.

```

Initialize critic networks  $Q_{\theta_1}, Q_{\theta_2}$ , and actor network  $\pi_{\varphi}$  with random parameters
 $\theta_1, \theta_2, \varphi$ .
Initialize target network  $\theta'_1 \leftarrow \theta_1, \theta'_2 \leftarrow \theta_2, \varphi' \leftarrow \varphi$ 
Initialize replay buffer  $\mathcal{B}$ 
for  $t = 1$  to  $T$  do
    Select action with exploration noise  $a \sim \pi_{\varphi}(s) + \epsilon$ ,
     $\epsilon \sim \mathcal{N}(0, \sigma)$  and observe reward  $r$  and new state  $s'$ 
    Store transition tuple  $(s, a, r, s')$  in  $\mathcal{B}$ 
    Sample mini-batch of  $N$  transitions  $(s, a, r, s')$  from  $\mathcal{B}$ 
     $\tilde{a} \leftarrow \pi_{\varphi}'(s') + \epsilon, \quad \epsilon \sim clip(\mathcal{N}(0, \sigma), -c, c)$ 
     $y \leftarrow r + \gamma min_{i=1,2} Q_{\theta_i}'(s', \tilde{a})$ 
    Update critics  $\theta_t \leftarrow argmin_{\theta_t} N^{-1} \sum (y - Q_{\theta_t}(s, a))^2$ 
    if  $t \bmod d$  then
        Update  $\varphi$  by the deterministic policy gradient:
         $\nabla_{\varphi} J(\varphi) = N^{-1} \sum \nabla_a Q_{\theta_1}(s, a)|_{a=\pi_{\varphi}(s)} \nabla_{\varphi} \pi_{\varphi}(s)$ 
        Update target networks:
         $\theta'_i \leftarrow \tau \theta_i + (1 - \tau) \theta'_i$ 
         $\varphi' \leftarrow \tau \varphi + (1 - \tau) \varphi'$ 
    end if
end for
```

Εικόνα 15: Ψευδοκώδικας αλγορίθμου TD3

3.3.2.4 Proximal Policy Optimization

Ο **Proximal Policy Optimization (PPO)** αναπτύχθηκε από ερευνητές της OpenAI με σκοπό να εξισορροπήσει την απόδοση και την ευρωστία κατά τη διαδικασία εκπαίδευσης πρακτόρων σε σύνθετα περιβάλλοντα [27].

Σε αντίθεση με τον DDPG και τον TD3, που ανήκουν στους off-policy αλγορίθμους, ο PPO είναι on-policy, πράγμα που σημαίνει ότι μαθαίνει από τα δεδομένα που προέρχονται μόνο από την τρέχουσα πολιτική του πράκτορα. Ο αλγόριθμος προέκυψε ως μία πιο σταθερή και απλή παραλλαγή του Trust Region Policy Optimization (TRPO) [28], ενός αλγορίθμου με εξαιρετικά αποτελέσματα σύγκλισης που παρ' όλα αυτά είναι πολύπλοκος στον υπολογισμό. Η βασική ιδέα του PPO είναι η εξής: κατά την ενημέρωση της πολιτικής, δεν επιτρέπουμε πολύ μεγάλες αλλαγές, ώστε να αποφύγουμε την καταστροφή της προηγούμενης γνώσης του πράκτορα. Αυτό υλοποιείται μέσω μίας συνάρτησης “**clipped surrogate objective function**”, η οποία περιορίζει το πόσο μπορεί να διαφέρει η νέα πολιτική από την προηγούμενη σε κάθε ενημέρωση. Η συγκεκριμένη προσέγγιση εγγυάται μεγαλύτερη σταθερότητα κατά την εκπαίδευση. Στον αλγόριθμο PPO σκοπός είναι η μεγιστοποίηση της συνάρτησης surrogate objective function $L(\theta)$:

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t [\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1 - \epsilon, 1 + \epsilon)\hat{A}_t)] ,$$

όπου:

- $r_t(\theta) = \frac{\pi_\theta(a_t | s_t)}{\pi_{\theta_{old}}(a_t | s_t)}$, είναι ο λόγος πιθανοτήτων της νέας πολιτικής προς την παλιά,
- ϵ είναι μια υπερπαράμετρος (συνήθως $\epsilon = 0.2$) που καθορίζει την επιτρεπόμενη απόκλιση της νέας πολιτικής σε σχέση με την προηγούμενη.

Με αυτό τον τρόπο οι ενημερώσεις διατηρούνται εντός ενός επιθυμητού εύρους αποφεύγοντας απότομες αλλαγές, ενώ παράλληλα ενισχύονται οι αφέλιμες μετατοπίσεις στην πολιτική. Σε πολλές περιπτώσεις, ο PPO συνδυάζεται με **Generalized Advantage Estimation (GAE)** [29] για την βελτίωση της ακρίβειας της εκτίμησης.

Η αρχιτεκτονική του PPO ενσωματώνει ένα actor-critic μοντέλο, το οποίο αποτελείται από δύο διακριτά αλλά συνεκπαιδευόμενα νευρωνικά δίκτυα: το δίκτυο του actor και του critic. Το actor δίκτυο μαθαίνει την στοχαστική πολιτική π_θ . Η είσοδος είναι η κατάσταση s , και οι ενδιάμεσες στρώσεις περιλαμβάνουν πλήρως συνδεδεμένα layers με ενεργοποιήσεις όπως ReLU ή tanh. Συνήθως εφαρμόζεται dropout ή layer normalization για σταθερότερη σύγκλιση. Για διακριτούς χώρους ενεργειών στην έξοδο χρησιμοποιείται η συνάρτηση softmax, ενώ για συνεχείς χώρους χρησιμοποιείται η MSE. Το critic δίκτυο έχει στόχο την εκτίμηση της συνάρτησης αξίας. Είναι ένα απλό regression network που παίρνει ως είσοδο την κατάσταση s και εξάγει έναν πραγματικό αριθμό.

Στην Εικόνα 16 παρουσιάζεται ο ψευδοκώδικας του αλγορίθμου PPO.

```
for iteration = 1,2, ... do
    for actor = 1,2, ..., N do
        Run policy  $\pi_{\theta_{old}}$  in environment for  $T$  timesteps
        Compute advantage estimates  $\hat{A}_1, \dots, \hat{A}_t$ 
        Optimize surrogate  $L$  wrt  $\theta$ , with  $K$  epochs and minibatch size  $M \leq NT$ 
         $\theta_{old} \leftarrow \theta$ 
    end for
end for
```

Εικόνα 16: Ψευδοκώδικας αλγορίθμου PPO

4 Ανάλυση Πειράματος σε περιβάλλον Προσομοίωσης

Η πειραματική διαδικασία αφορά την εφαρμογή των αλγορίθμων DDPG και TD3 για την εκπαίδευση μίας επίγειας ρομποτικής πλατφόρμας να πλοηγείται αυτόνομα και να αποφεύγει στατικά και δυναμικά εμπόδια μέσα σε ένα προσομοιωμένο περιβάλλον. Οι συγκεκριμένοι αλγόριθμοι επιλέχθηκαν διότι προσφέρουν γρήγορη εκπαίδευση με αξιόλογες επιδόσεις σε δυναμικά περιβάλλοντα, σε αντίθεση με τον PPO ο οποίος μπορεί να προσφέρει μεγαλύτερα ποσοστά επιτυχίας, αλλά χρειάζεται πολύ περισσότερο χρόνο εκπαίδευσης. Για την υλοποίηση του πειράματος χρησιμοποιήθηκε το ανοιχτού κώδικα αποθετήριο (open source repository) turtlebot3_drlnav [30]. Το συγκεκριμένο repository περιέχει ένα ολοκληρωμένο πλαίσιο (framework) το οποίο συνδυάζει το μεσολογισμικό ρομποτικών εφαρμογών ROS και την βιβλιοθήκη PyTorch της γλώσσας προγραμματισμού Python, με σκοπό την ανάπτυξη και εφαρμογή αλγορίθμων Ενισχυτικής Μάθησης για αυτόνομη πλοήγηση ρομποτικών πλατφορμών. Περισσότερες πληροφορίες για την PyTorch και για το ROS αναφέρονται στα κεφάλαια 4.1.3 και 4.1.4 αντίστοιχα. Στην παρούσα ενότητα περιγράφονται αναλυτικά τα τεχνικά χαρακτηριστικά του συστήματος προσομοίωσης, τα περιβάλλοντα προσομοίωσης και η διαδικασία εκπαίδευσης των μοντέλων.

4.1 Περιγραφή του Συστήματος

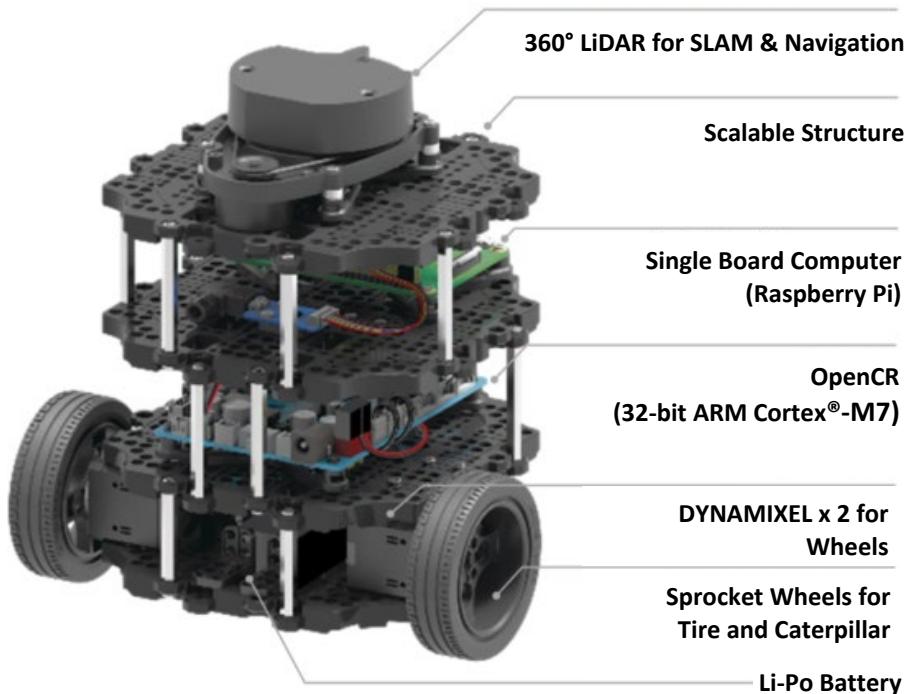
Το σύστημα αποτελείται από τρεις συνιστώσες, το ρομπότ, το περιβάλλον προσομοίωση και τον αλγόριθμο DRL.

4.1.1 Το ρομπότ

Για τις ανάγκες της προσομοίωσης, το ρομπότ το οποίο χρησιμοποιήθηκε είναι το Turtlebot3 και πιο συγκεκριμένα το μοντέλο Turtlebot3 Burger. Το TurtleBot3 είναι μια ευρέως διαδεδομένη ρομποτική πλατφόρμα, σχεδιασμένη ειδικά για πειραματισμό σε εφαρμογές ρομποτικής πλοήγησης. Χρησιμοποιείται κατά κόρων στην ακαδημαϊκή και ερευνητική κοινότητα λόγω της συμβατότητας και της προσαρμοστικότητας την οποία προσφέρει σε μία πληθώρα εφαρμογών, ενώ ταυτόχρονα η εκτεταμένη χρήση του ανά τα χρόνια έχει οδηγήσει στην ανάπτυξη αξιόλογης διαδικτυακής υποστήριξης. Το TurtleBot3 Burger διαθέτει διαφορική κίνηση με δύο τροχούς, αισθητήρα LiDAR για χαρτογράφηση και αποφυγή εμποδίων, καθώς και IMU για ανίχνευση επιτάχυνσης και προσανατολισμού. Η συμβατότητα του λογισμικού του με το λογισμικό όλων των επιμέρους συστημάτων που χρησιμοποιήθηκαν στην συγκεκριμένη υλοποίηση και σε

συνδυασμό με όλα τα πλεονεκτήματα που προαναφέρθηκαν, καθιστούν το Turtlebot3 ιδανικό για την παρούσα εφαρμογή. Στην Εικόνα 17 φαίνονται τα εξαρτήματα από τα οποία αποτελείται το μοντέλο που χρησιμοποιήθηκε και στον Πίνακας 3 αναγράφονται μερικές από τις σημαντικότερες προδιαγραφές του hardware του.

TurtleBot3 Burger



Εικόνα 17: Εξαρτήματα του Turtlebot3 Burger

Στοιχείο	Τιμή
Μέγιστη γραμμική ταχύτητα	0.22 m/s
Μέγιστη περιστροφική ταχύτητα	2.84 rad/s (162.72 deg/s)
Μέγιστο φορτίο	15kg
Διαστάσεις (Μ x Π x Γ)	138mm x 178mm x 192mm
Βάρος (+ SBC + Μπαταρία + Αισθητήρες)	1kg
Αναμενόμενος χρόνος λειτουργίας	2h 30m
Αναμενόμενος χρόνος για πλήρη φόρτιση	2h 30m
SBC (Single Board Computer)	Raspberry Pi 4
MCU	32-bit ARM Cortex®-M7 with FPU (216 MHz, 462 DMIPS)
LDS (Laser Distance Sensor)	360 Laser Distance Sensor LDS-02
IMU	Gyroscope 3 Axis Accelerometer 3 Axis

Πίνακας 3: Προδιαγραφές Hardware του Turtlebot3 Burger

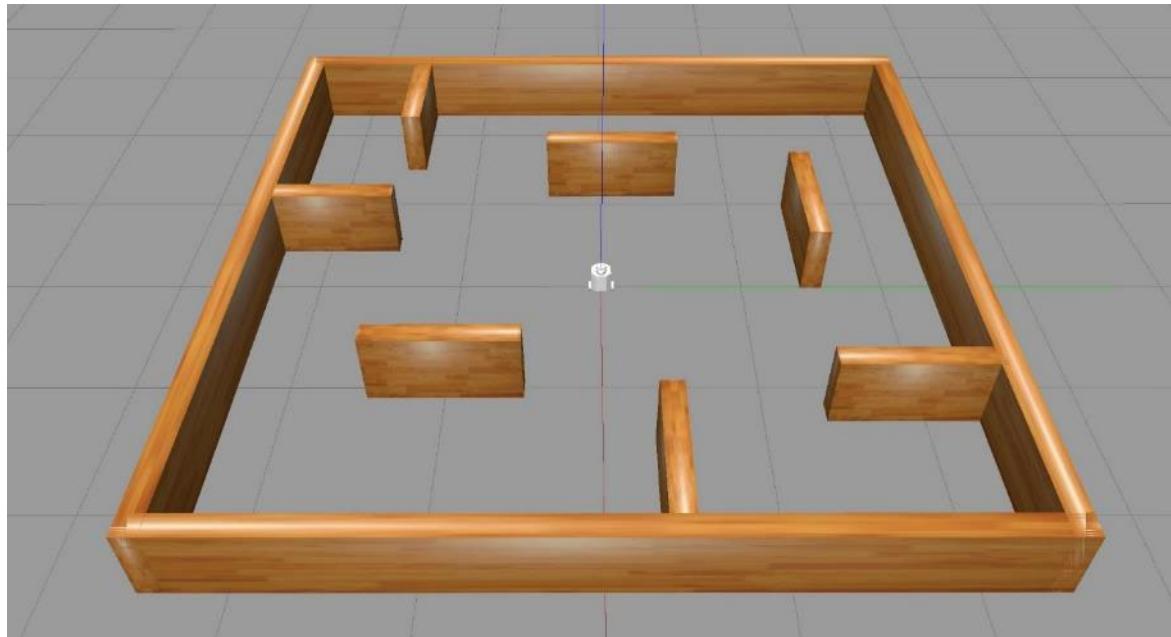
4.1.2 Το περιβάλλον

Η χρήση περιβαλλόντων προσομοίωσης για την εκπαίδευση αλγορίθμων αποτελεί βασική τακτική στον χώρο της ρομποτικής και της τεχνητής νοημοσύνης. Οι αλγόριθμοι DRL πιο συγκεκριμένα βασίζονται σε δοκιμή και σφάλμα (trial-and-error), γεγονός που σημαίνει ότι το ρομπότ πρέπει να εκτελέσει χλιάδες ενέργειες, πολλές από τις οποίες είναι λανθασμένες ή και καταστροφικές. Η απευθείας εκπαίδευση σε φυσικό περιβάλλον θα οδηγούσε σε φθορά εξοπλισμού με μεγάλο οικονομικό κόστος, καθώς επίσης και κινδύνους ασφαλείας. Αντιθέτως, σε ένα εικονικό περιβάλλον το ρομπότ μπορεί να εκπαιδευτεί με ασφάλεια, χωρίς να υπάρχει πιθανότητα υλικής ζημιάς. Τα προσομοιωμένα περιβάλλοντα παρέχουν πλήρη έλεγχο, επιτρέποντας την ακριβή μέτρηση μεγεθών όπως η θέση, η ταχύτητα και οι επιδόσεις του πράκτορα, ενώ παράλληλα προσφέρουν ευκολία επανάληψης, σημαντικό πλεονέκτημα σε σχέση με τον πραγματικό κόσμο όπου οι συνθήκες είναι δύσκολο να επαναληφθούν με ακρίβεια. Έτσι μπορεί κανείς να ισχυριστεί ότι τα προσομοιωμένα περιβάλλοντα λειτουργούν ως σημείο εκκίνησης για την ανάπτυξη αλγορίθμων πριν μεταφερθούν στο πραγματικό ρομπότ. Παρόλο που η προσομοίωση δεν αντικαθιστά πλήρως την πραγματική δοκιμή, αποτελεί αναγκαίο στάδιο για την επιτάχυνση, βελτιστοποίηση και ασφαλή ανάπτυξη σύγχρονων συστημάτων πλοήγησης που βασίζονται στη DRL.

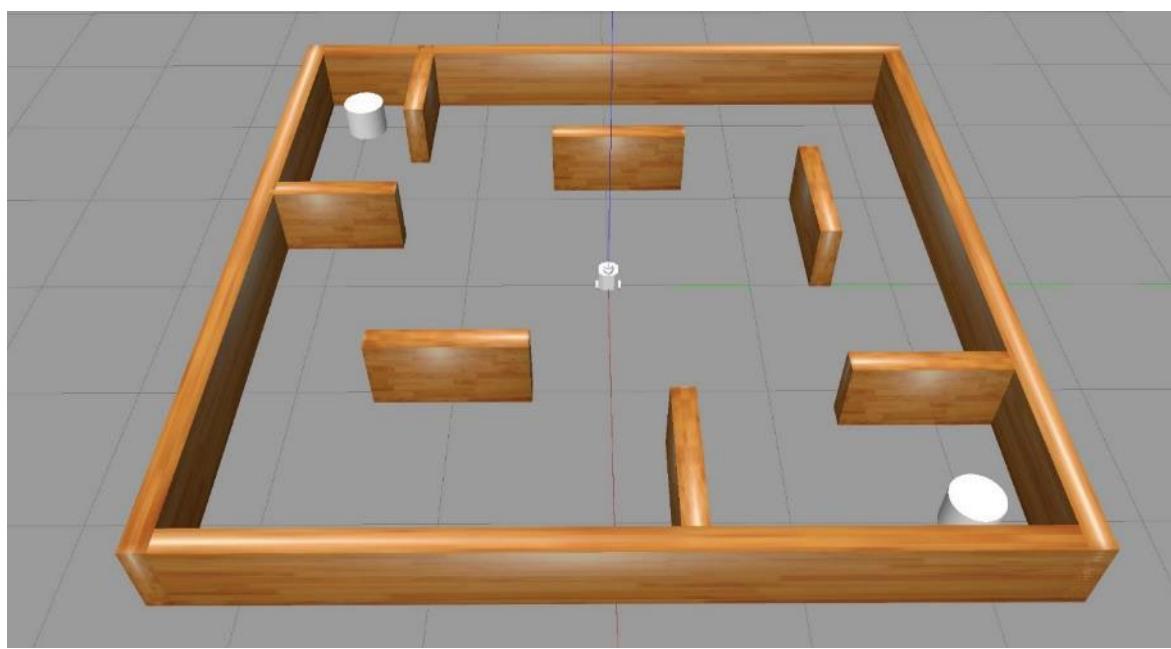
Για την εκπαίδευση και αξιολόγηση των αλγορίθμων DRL που αναπτύχθηκαν στην παρούσα εργασία, χρησιμοποιήθηκε το περιβάλλον προσομοίωσης **Gazebo**. Το Gazebo είναι ένα ανοιχτού κώδικα περιβάλλον προσομοίωσης σχεδιασμένο ειδικά για την ανάπτυξη και δοκιμή ρομποτικών συστημάτων. Χάρη σε μηχανές φυσικής όπως η DART (Dynamic Animation and Robotics Toolkit) το Gazebo υποστηρίζει βαρυτικές δυνάμεις, τριβή και σύγκρουση, παρέχοντας με αυτό τον τρόπο ρεαλιστική απεικόνιση βάση νόμων της φυσικής. Παράλληλα κάνει δυνατή την ρεαλιστική προσομοίωση πολλών αισθητήρων όπως LiDAR, κάμερες RGB/D, IMU και GNSS, οι οποίοι είναι απαραίτητα στοιχεία για την ανάπτυξη path planning αλγορίθμων.

Οι αλγόριθμοι εκπαιδεύτηκαν σε τρία διαφορετικά περιβάλλοντα αυξανόμενης δυσκολίας ανάλογα με το πλήθος των κινητών εμποδίων που περιέχουν. Σκοπός είναι η αξιολόγηση των επιδόσεων του κάθε αλγορίθμου τόσο σε περιβάλλοντα με στατικά εμπόδια (τοίχους) όσο και σε δυναμικά περιβάλλοντα με κινητά εμπόδια. Το πρώτο περιβάλλον (Εικόνα 18) είναι ένας 10x10 χώρος ο οποίος περικλείεται από τέσσερεις τοίχους. Στο εσωτερικό αυτού έχουν προσαρμοστεί μικρά τοιχία τα οποία αποτελούν τα εσωτερικά στατικά εμπόδια, σε μία προσπάθεια να προσομοιωθούν φυσικά εμπόδια όπως τραπέζια, ντουλάπες και εσωτερικοί τοίχοι ενός κτηρίου. Το δεύτερο περιβάλλον (Εικόνα 19) είναι ίδιο με το πρώτο με μόνη διαφορά την προσθήκη δύο δυναμικών εμποδίων. Ως δυναμικά εμπόδια χρησιμοποιήθηκαν κυλινδρικά συμπαγή σώματα. Η κυλινδρική δομή επιλέχθηκε με σκοπό την όσο γίνεται πιο προσεγγιστική απεικόνιση κινητών εμποδίων όπως για παράδειγμα ενός ανθρώπου ή άλλων ρομπότ. Το τρίτο και πιο δύσκολο περιβάλλον (Εικόνα 20) είναι εξίσου ίδιο με τα προηγούμενα δύο ως προς το στατικό μέρος, περιέχει όμως και έξι δυναμικά εμπόδια ίδιου σχήματος με αυτά του

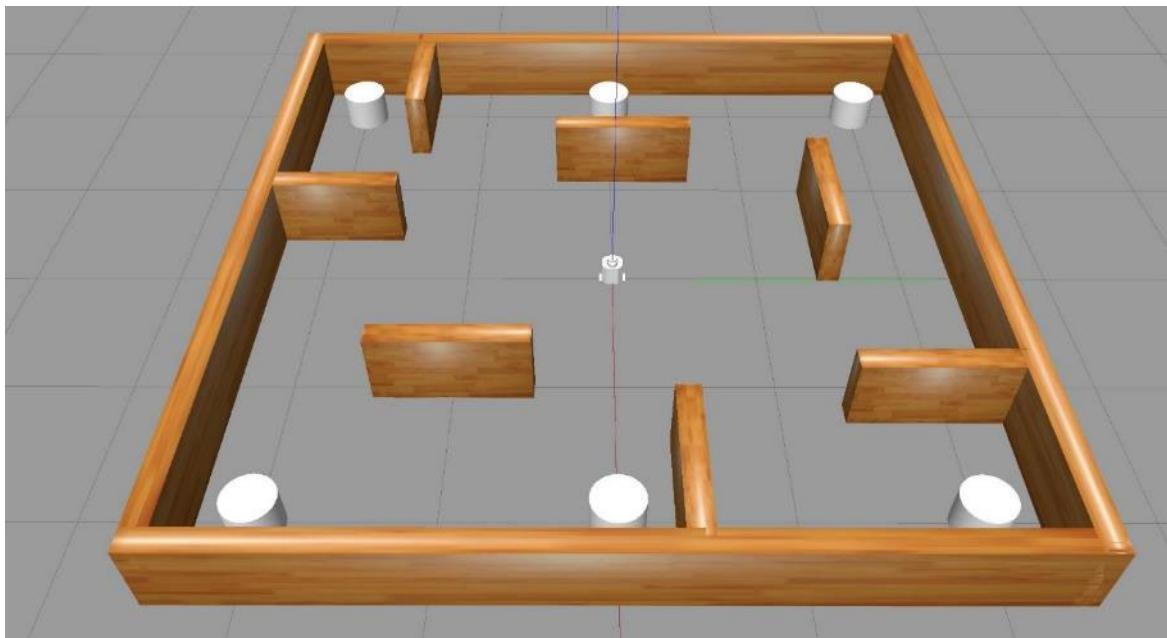
δεύτερου περιβάλλοντος. Τα συγκεκριμένα περιβάλλοντα μπορούν να βρεθούν στο repository `turtlebot3_drlnav`.



Εικόνα 18: Περιβάλλον Δυσκολίας 1



Εικόνα 19: Περιβάλλον Δυσκολίας 2



Εικόνα 20: Περιβάλλον Δυσκολίας 3

4.1.3 Ο αλγόριθμος DRL

Η τρίτη συνιστώσα του συστήματος είναι ο αλγόριθμος DRL, ο οποίος αποτελεί τον πράκτορα μάθησης και λήψης αποφάσεων. Όπως προαναφέρθηκε, η εκπαίδευση του ρομπότ πραγματοποιήθηκε με την χρήση δύο αλγορίθμων DRL, των TD3 και DDPG. Η επιλογή των αλγορίθμων αυτών βασίστηκε στην ικανότητά τους να χειρίζονται προβλήματα ελέγχου με συνεχείς χώρους δράσης, κάτι που είναι σύνηθες σε ρομποτικές εφαρμογές πλοήγησης. Σε αντίθεση με άλλες τεχνικές που περιορίζονται σε διακριτούς χώρους δράσης, οι DDPG και TD3 χρησιμοποιούν έναν actor-critic μηχανισμό για την εκμάθηση πολιτικών, γεγονός που διευκολύνει την επιλογή βέλτιστων κινήσεων μέσω του χειρισμού συνιστωσών όπως η ταχύτητα και η κατεύθυνση.

Ο πράκτορας δέχεται ως είσοδο ένα διάνυσμα σαρανταεσσάρων (44) τιμών, το οποίο αποτελείται από:

- Δείγματα από τον αισθητήρα LiDAR: Η αρχική έξοδος του LiDAR είναι 1080 ακτίνες, αλλά για λόγους υπολογιστικής αποδοτικότητας επαναδειγματοληπτούνται (resampled) σε 40 κανονικοποιημένες τιμές (από 0 έως 1)
- Τέσσερεις επιπλέων τιμές οι οποίες βοηθούν τον πράκτορα να εκτιμήσει πού βρίσκεται, πώς κινείται και πώς να κατευθυνθεί προς τον στόχο.
 1. Κανονικοποιημένη απόσταση προς τον στόχο
 2. Κανονικοποιημένη γωνιακή διαφορά από τον στόχο
 3. Γραμμική ταχύτητα του ρομπότ
 4. Γωνιακή ταχύτητα του ρομπότ

Στην έξοδό του ο πράκτορας παράγει δύο διακριτές τιμές οι οποίες αντιστοιχούν στη γραμμική και στη γωνιακή ταχύτητα του ρομπότ. Οι δύο αυτές τιμές αποτελούν την δράση του πράκτορα.

Στον κώδικα των νευρωνικών δικτύων για την κάθε υλοποίηση χρησιμοποιείται μία βιβλιοθήκη της Python, η PyTorch. Η PyTorch είναι ένα πλαίσιο Μηχανικής Μάθησης (machine learning framework), σχεδιασμένο για την ανάπτυξη και εκπαίδευση νευρωνικών δικτύων. Δημιουργήθηκε από την Facebook's AI Research lab (FAIR) και έχει γίνει ένα από τα πιο διαδεδομένα εργαλεία στον χώρο της τεχνητής νοημοσύνης, ειδικά για εφαρμογές Βαθιάς Μάθησης, λόγω της δυνατότητάς της να συνδυάζει ευχρηστία, ταχύτητα και αποδοτικότητα [31].

4.1.4 ROS

Η καθεμία από τις συνιστώσες αυτές επιτελεί τον δικό της σκοπό στο ευρύτερο σύστημα. Ωστόσο, αν και ξεχωριστές οντότητες υπάρχει άμεση εξάρτηση μεταξύ αυτών, καθώς η κάθε μία χρησιμοποιεί δεδομένα τα οποία προκύπτουν από τις άλλες. Για την ανταλλαγή δεδομένων και την ομαλή λειτουργεία του συστήματος συνεπώς, είναι αναγκαία η δημιουργία επικοινωνίας μεταξύ τους. Ο δίαυλος επικοινωνίας μεταξύ αυτών είναι το **ROS (Robotics Operation System)**. Το ROS είναι ένα framework σχεδιασμένο για την ανάπτυξη εφαρμογών ρομποτικής. Αν και ονομάζεται "λειτουργικό σύστημα", στην πράξη λειτουργεί ως middleware, δηλαδή ως ενδιάμεσο επίπεδο που επιτρέπει σε διάφορα κομμάτια λογισμικού να επικοινωνούν μεταξύ τους εύκολα και αποτελεσματικά. Η εξέλιξη της τεχνολογίας στον τομέα της ρομποτικής ανά τα χρόνια οδήγησε στην δημιουργία του ROS2. Το ROS2 σε αντίθεση με το ROS1 σχεδιάστηκε για real-time εφαρμογές, καθιστώντας το πιο κατάλληλο για χρήση σε βιομηχανικά ρομπότ. Παρακάτω αναλύονται μερικές από τις βασικότερες έννοιες του ROS:

- **Node**

Ένας node είναι η βασική μονάδα εκτέλεσης σε ένα σύστημα ROS που επιτελεί μια συγκεκριμένη λειτουργία. Κάθε κόμβος εκτελεί μία συγκεκριμένη διεργασία: για παράδειγμα, μπορεί να διαβάζει δεδομένα από έναν αισθητήρα, να ελέγχει ένα μοτέρ, να επεξεργάζεται εικόνες κ.ά.

- **Topic**

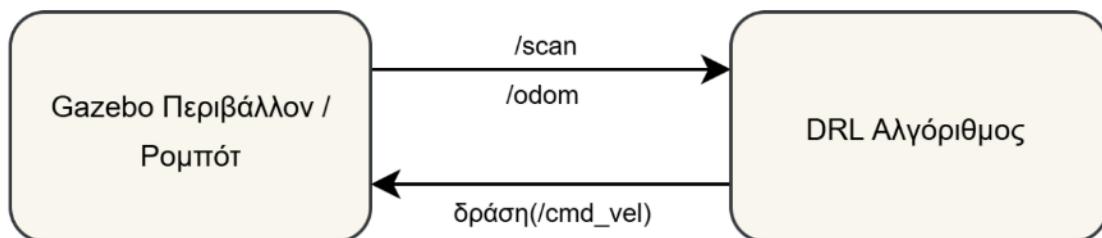
Τα topics είναι κανάλια επικοινωνίας στα οποία δημοσιεύονται ή λαμβάνονται (publish/subscribe) μηνύματα μεταξύ των nodes. Ένας node (publisher) δημοσιεύει μηνύματα (messages σε ένα topic ενώ παράλληλα ένας άλλος (subscriber) εγγράφεται στο ίδιο topic για να λαμβάνει αυτά τα μηνύματα.

- **Μήνυμα (Message)**

Τα messages είναι δεδομένα που μεταφέρονται από ένα node σε ένα άλλο. Αποτελούν την πληροφορία που μεταδίδεται μέσα από τα topics.

Στο πλαίσιο της παρούσας εργασίας, το ROS2 λειτουργεί ως middleware που διαχειρίζεται την επικοινωνία και τον συγχρονισμό μεταξύ των επιμέρους στοιχείων του συστήματος, όπως είναι το ρομπότ (TurtleBot3), το περιβάλλον προσομοίωσης (Gazebo) και οι αλγόριθμοι DRL. Συγκεκριμένα, το ROS2 αναλαμβάνει την ανταλλαγή πληροφοριών μέσω topics, καθώς και τη διαχείριση των δεδομένων κατάστασης του ρομπότ, των εντολών ελέγχου και των δεδομένων αισθητήρων (όπως από LiDAR ή IMU). Οι πράκτορες υλοποιούνται ως ROS2 nodes, οι οποίοι λαμβάνουν δεδομένα κατάστασης από το περιβάλλον μέσω topics όπως /scan (LiDAR scans) και /odom (οδομετρία), επεξεργάζονται την πληροφορία και έπειτα αποστέλλουν τις κατάλληλες εντολές δράσης (γραμμική και γωνιακή ταχύτητα) στο ρομπότ μέσω του topic /cmd_vel. Το ολοκληρωμένο σύστημα πλοήγησης παρουσιάζεται σε μορφή γραφήματος στην Εικόνα 21.

Σύστημα Πλοήγησης



Εικόνα 21: Το Σύστημα Πλοήγησης

4.2 Εκπαίδευση Αλγορίθμων

Ο Στόχος

Ο στόχος (goal) στην εκπαίδευση του αλγορίθμου πλοήγησης DRL ορίζεται ως ένα συγκεκριμένο σημείο στον χώρο στο οποίο πρέπει να φτάσει το ρομπότ. καθορίζεται από τις συντεταγμένες (x_{goal}, y_{goal}) . Το ρομπότ ξεκινά από μια τυχαία θέση στον χώρο και μαθαίνει να κινείται έτσι ώστε να ελαχιστοποιεί την απόσταση του από αυτόν τον στόχο.

Η απόσταση από τον στόχο υπολογίζεται χρησιμοποιώντας την Ευκλείδεια απόσταση:

$$goal_distance = \sqrt{(x - x_{goal})^2 + (y - y_{goal})^2},$$

όπου (x, y) είναι η τρέχουσα θέση του ρομπότ.

Επειδή είναι σχεδόν αδύνατο το ρομπότ να μπορέσει να μάθει να πηγαίνει σε ένα συγκεκριμένο σημείο του χώρου με την έννοια του σημείου (δηλαδή ακριβείς

συντεταγμένες), δημιουργήθηκε η έννοια του στόχου ως κυκλική περιοχή. Η ακτίνα του στόχου (goal radius) είναι το μέγιστο επιτρεπόμενο σφάλμα θέσης που θεωρείται αποδεκτό ώστε να θεωρηθεί ότι το ρομπότ "πέτυχε" τον στόχο, δηλαδή εάν η απόσταση από τον στόχο είναι μικρότερη από την τιμή της ακτίνας τότε το επεισόδιο τερματίζει επιτυχώς. Η επιλογή της ακτίνας είναι κρίσιμη, καθώς αν είναι πολύ μικρή μπορεί το ρομπότ να γυρίζει γύρω από τον στόχο και να μην τον πετυχαίνει ποτέ, ενώ αν είναι πολύ μεγάλη το επεισόδιο θα τερματίζεται πρόωρα, με το ρομπότ να είναι ακόμη αρκετά μακριά από τον στόχο. Η τιμή που χρησιμοποιήθηκε στην συγκεκριμένη υλοποίηση είναι 0.1 μέτρα.

Η Συνάρτηση Ανταμοιβής

Η ανταμοιβή του πράκτορα υπολογίζεται μέσα από μία συνάρτηση ανταμοιβής (reward function). Οι παράμετροι εισόδου της συνάρτησης αυτής είναι οι εξής:

- *succeed*: κατάσταση τερματισμού του επεισοδίου. Οι κατηγορίες τερματισμού του επεισοδίου είναι οι εξής:
 1. Επιτυχία (success)
 2. Σύγκρουση με τοίχο (collision_wall)
 3. Σύγκρουση με δυναμικό εμπόδιο (collision_obstacle)
 4. Λήξη χρόνου (timeout)
 5. Ανεξήγητος τερματισμός (tumble)
- *action_linear*: γραμμική ταχύτητα που επέλεξε ο πράκτορας στο βήμα.
- *action_angular*: γωνιακή ταχύτητα (στροφή).
- *goal_dist*: η απόσταση του ρομπότ από τον στόχο τη δεδομένη στιγμή.
- *goal_angle*: η γωνιακή απόκλιση του ρομπότ από την κατεύθυνση προς τον στόχο (το πόσο "στρίβει" ή πόσο στραβή είναι η κατεύθυνση).
- *min_obstacle_dist*: η μικρότερη απόσταση μεταξύ του ρομπότ και εμποδίου.

Η συνολική ανταμοιβή του βήματος υπολογίζεται ως το άθροισμα κάποιων επιμέρους συνιστώσων. Οι συνιστώσες μπορεί να είναι είτε ανταμοιβές είτε ποινές:

Γωνιακή Ποινή: Μείωση της ανταμοιβής ανάλογα με τη γωνιακή απόκλιση του ρομπότ με τον στόχο. Έτσι ενθαρρύνεται το ρομπότ να στρίβει προς τον στόχο. Παίρνει τιμές στο διάστημα [3.14, 0].

$$r_{yaw} = - \text{abs}(goal_angle) .$$

Ανταμοιβή απόστασης από τον στόχο: Όσο μικρότερη η απόσταση από τον στόχο σε σχέση με την αρχική θέση του ρομπότ (*goal_dist_initial*), τόσο μεγαλύτερη η ανταμοιβή. Ο πράκτορας ανταμείβεται σταδιακά όσο πλησιάζει στον στόχο. Παίρνει τιμές μεταξύ -1 και 1.

$$r_distance = (2 * goal_dist_initial) / (goal_dist_initial + goal_dist) - 1.$$

Ποινή για κοντινό εμπόδιο: Ο πράκτορας τιμωρείται όταν έρθει πολύ κοντά σε κάποιο εμπόδιο. Εάν η απόσταση από το κοντινότερο εμπόδιο είναι μικρότερη από 0.22 μέτρα ($min_obstacle_dist < 0.22$), τότε ο πράκτορας τιμωρείται με $r_obstacle = -20$.

Ποινή για γραμμική ταχύτητα: Η συγκεκριμένη ποινή είναι υπεύθυνη για να παροτρύνει τον πράκτορα να κινείται με την μέγιστη γραμμική ταχύτητα. Επιπλήττει λοιπόν τον πράκτορα όλο και περισσότερο για μικρότερες τιμές γραμμικής ταχύτητας. Παίρνει τιμές στο διάστημα [-4.84, 0].

$$r_vlinear = -(((0.22 - action_linear) * 10)^2).$$

Ποινή για γωνιακή ταχύτητα: Όσο μεγαλύτερη είναι η γωνιακή ταχύτητα τόσο μεγαλύτερη και η ποινή. Με αυτό τον τρόπο το ρομπότ μαθαίνει να αποφεύγει απότομες περιστροφικές κινήσεις που ενδέχεται να οδηγήσουν σε αστάθεια. Παίρνει τιμές στο διάστημα [-4, 0].

$$r_vangular = -(action_angular^2).$$

Συνεπώς, η συνολική ανταμοιβή ενός βήματος υπολογίζεται ως εξής:

$$reward = r_yaw + r_distance + r_obstacle + r_vlinear + r_vangular - 1.$$

Επιπλέον ανταμοιβές/ποινές στο τέλος ενός επεισοδίου: Αν το ρομπότ φτάσει στον στόχο τότε έχουμε μια επιτυχή προσπάθεια και ο πράκτορας επιβραβεύεται με αύξηση της συνολικής ανταμοιβής κατά 2500. Αντιθέτως, αν το ρομπότ συγκρουστεί με κάποιο εμπόδιο τότε μειώνεται η συνολική ανταμοιβή του πράκτορα κατά 2000. Παρατηρείται ότι υπάρχει μεγάλη διαφορά μεταξύ αυτών των τιμών και των τιμών των επιμέρους συνιστώσων. Αυτό οφείλεται στο γεγονός ότι η κατάσταση τερματισμού έχει μεγάλο βάρος, είναι στην ουσία το αποτέλεσμα του επεισοδίου και είναι αυτή που πρέπει να διαμορφώνει την τελική τιμή της συνολικής ανταμοιβής, κάνοντας τον πράκτορα να αντιληφθεί ποιος είναι ο πραγματικός σκοπός του.

Τα στάδια εκπαίδευσης

Κατά την έναρξη της εκπαίδευσης ενός αλγορίθμου Ενισχυτικής Μάθησης, ειδικά σε περιπτώσεις όπου χρησιμοποιούνται off-policy μέθοδοι όπως οι DDPG και TD3, είναι σύνηθες να προηγείται μία **φάση προθέρμανσης (warm-up phase)** [8]. Στην περίοδο αυτή, ο πράκτορας δεν βασίζεται στο νευρωνικό δίκτυο για την επιλογή δράσεων, αλλά επιλέγει τυχαίες ενέργειες από έναν στοχαστικό θόρυβο, ώστε να γεμίσει αρχικά τη μνήμη replay buffer με έναν επαρκή αριθμό μεταβάσεων (state, action, reward, next state). Η προσέγγιση αυτή είναι κρίσιμη, διότι η έναρξη της μάθησης χωρίς προηγούμενη πληροφορία μπορεί να οδηγήσει σε overfitting. Η συλλογή εμπειριών μέσω τυχαίας εξερεύνησης βοηθά στην εκκίνηση της διαδικασίας εκπαίδευσης με πιο σταθερές βάσεις. Η διαδικασία αυτή αποτελεί το πρώτο στάδιο της εκπαίδευσης.

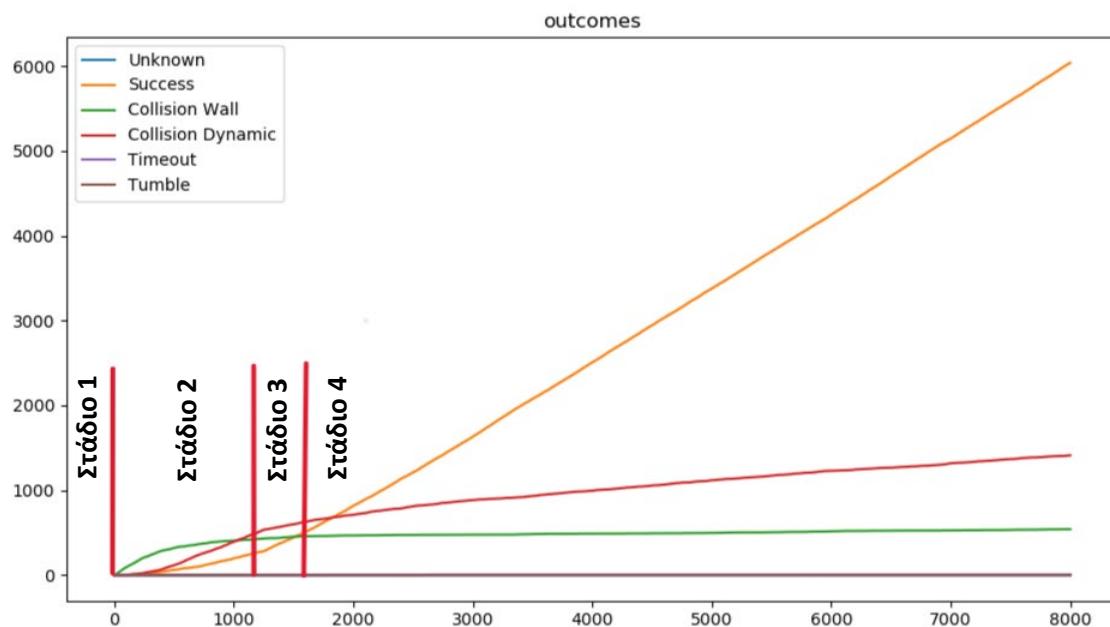
Μετά την ολοκλήρωση του προκαθορισμένου αριθμού χρονικών βημάτων (π.χ. 10.000), ο πράκτορας αρχίζει να αξιοποιεί την πολιτική του αλγορίθμου για την επιλογή ενέργειας και να ενημερώνει το δίκτυο βάσει των δειγμάτων που έχουν αποθηκευτεί. Στις πρώτες εποχές (~500 – 1000 ανάλογα με το πόσο γρήγορα συγκλίνει το μοντέλο) παρατηρείται μεγάλος αριθμός συγκρούσεων σε στατικά και δυναμικά εμπόδια. Αυτό είναι απόλυτα λογικό από την άποψη ότι ο πράκτορας δεν έχει “μάθει” ακόμη το πως να χρησιμοποιεί τα δεδομένα του για να αποφύγει την σύγκρουση με κάποιο εμπόδιο και να αυξήσει την σωρευτική ανταμοιβή που λαμβάνει. Η φάση αυτή είναι το δεύτερο στάδιο της εκπαίδευσης.

Το τρίτο στάδιο της εκπαίδευσης είναι ένα μεταβατικό στάδιο κατά το οποίο ο πράκτορας αρχίζει να αντιλαμβάνεται ότι η αποφυγή συγκρούσεων οδηγεί σε μεγαλύτερη ανταμοιβή. Έτσι παρατηρούμε το ρομπότ να προσπαθεί να αποφύγει τα εμπόδια πραγματοποιώντας παράλληλα ένα είδος εξερεύνησης, η οποία το οδηγεί σιγά σιγά προς τον στόχο. Όπως θα φανεί και γράφημα της Εικόνα 22, το μεταβατικό αυτό στάδιο είναι χαρακτηριστικό διότι σταδιακά μειώνονται οι συγκρούσεις και αυξάνονται οι επιτυχίες.

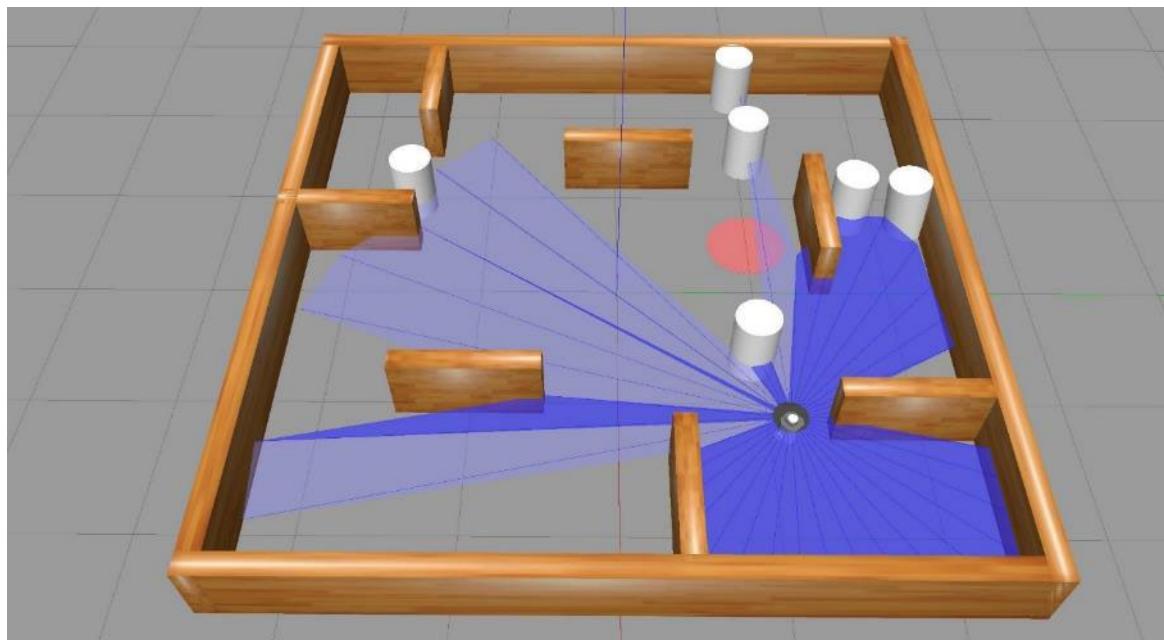
Στην συνέχεια ο πράκτορας μεταβαίνει στο τέταρτο και τελικό στάδιο της εκπαίδευσης όπου πλέον έχει μάθει να αντιλαμβάνεται τον χώρο και χρησιμοποιεί τα δεδομένα του για να οδηγηθεί στον στόχο όσο το δυνατόν πιο σύντομα, χωρίς να συγκρουστεί με κάποιο εμπόδιο.

Κατά την εκπαίδευση δημιουργείται και ένα γράφημα το οποίο περιλαμβάνει τις κατηγορίες τερματισμού ενός επεισοδίου (επιτυχία, σύγκρουση με τοίχο ή δυναμικό εμπόδιο, λήξη χρόνου, ανεξήγητος τερματισμός) και παρουσιάζει το πόσες φορές προέκυψε το συγκεκριμένο αποτέλεσμα σε βάθος εποχών. Το συγκεκριμένο γράφημα θα αναλυθεί περεταίρω μαζί και με άλλα γραφήματα στην επόμενη ενότητα. Στην Εικόνα 22 χρησιμοποιείται ένα τυχαίο γράφημα κατηγοριών τερματισμού για την απεικόνιση των σταδίων της εκπαίδευσης.

Η Εικόνα 23 είναι ένα στιγμιότυπο της εκπαίδευσης στο οποίο φαίνεται το ρομπότ μέσα στο περιβάλλον και οι μπλε ακτίνες οι οποίες οπτικοποιούν τις ακτίνες του LiDAR. Ο κόκκινος κύκλος αντιπροσωπεύει τον στόχο της πλοιόγησης.



Εικόνα 22: Γράφημα Καταστάσεων Τερματισμού



Εικόνα 23: Στιγμιότυπο εκπαίδευσης

Η εκπαίδευση ήταν πολύωρη, περίπου 36-60 ώρες για κάθε υλοποίηση, με τους αλγόριθμους να εκπαιδεύονται για 6000 και άνω επεισόδια έκαστος. Το ρομπότ εκπαιδεύτηκε αρχικά με τις προεπιλεγμένες ρυθμίσεις που προτείνονται στο repository. Έπειτα πραγματοποιήθηκε ρύθμιση των παραμέτρων των αλγορίθμων με σκοπό την βελτίωση της απόδοσής τους.

Ρύθμιση Παραμέτρων

Η πρώτες εκπαιδεύσεις σε κάθε περιβάλλον πραγματοποιήθηκαν με τις προκαθορισμένες ρυθμίσεις παραμέτρων του repository. Με σκοπό την βελτιστοποίηση του αλγορίθμου έγιναν αλλαγές σε έξι από αυτές τις παραμέτρους οι οποίες είναι σημειωμένες στην Εικόνα 24 με κόκκινο περίγραμμα. Η ρύθμισή τους έγινε σταδιακά έτσι ώστε να υπάρχει ελεγχόμενη προσαρμογή. Οι αλλαγές γίνονταν με γνώμονα την απόδοση και τον χρόνο σύγκλησης του αλγορίθμου.

```
state_size = 44
action_size = 2
hidden_size = 512
input_size = 44
batch_size = 128
buffer_size = 1000000
discount_factor = 0.99
learning_rate = 0.003
tau = 0.003
step_time = 0.01
loss_function = <function smooth_l1_loss at 0x74a189712c20>
epsilon = 1.0
epsilon_decay = 0.9995
epsilon_minimum = 0.05
reward_function = A
backward_enabled = False
stacking_enabled = False
stack_depth = 3
```

Εικόνα 24: Αρχικές Τιμές Παραμέτρων

Learning Rate

Η τιμή του learning rate επηρεάζει τον χρόνο σύγκλησης και την ευστάθεια του αλγορίθμου. Πραγματοποιήθηκαν δοκιμές για τιμές από 10^{-4} έως $3 \cdot 10^{-3}$.

Παράμετρος τ

Είναι η παράμετρος ρύθμισης των soft updates των target networks (πχ. στον DDPG: $\theta^0' \leftarrow \tau\theta^0 + (1 - \tau)\theta^0'$). Μικρές τιμές του οδηγούν σε σταθερή αλλά αργή προσαρμογή των target networks, ενώ αντιθέτως μεγάλες τιμές οδηγούν σε ταχύτερη προσαρμογή αλλά ενδέχεται να προκαλέσουν θόρυβο-αστάθεια. Έγιναν δοκιμές για εύρος τιμών από 10^{-4} έως $3 \cdot 10^{-3}$.

Hidden Size

Το μέγεθος των κρυφών στρωμάτων του δικτύου. Όσο περισσότερα κρυφά στρώματα έχει ένα δίκτυο τόσο πιο “βαθύ” είναι. Δοκιμάστηκαν εμπειρικά και σύμφωνα με την βιβλιογραφία οι τιμές 256, 512 και 1024.

Batch Size

Μικρό batch size απαιτεί λιγότερη μνήμη, αλλά προκαλεί προβλήματα στην ευστάθεια και τον χρόνο σύγκλησης, ενώ μεγάλο batch size απαιτεί μεγαλύτερη επεξεργαστική ισχύ αλλά το μοντέλο συγκλίνει αμεσότερα και με ευστάθεια. Πολύ μεγάλο batch size ενδέχεται να προκαλέσει overfitting και πρέπει να αποφεύγεται. Χρησιμοποιήθηκαν οι τιμές 128, 256 και 512.

Backward Enabled

Ενεργοποιώντας την συγκεκριμένη ρύθμιση επιτρέπουμε στο ρομπότ να κινηθεί και προς τα πίσω, με την έξοδο του αλγορίθμου να παράγει πλέον και αρνητικές τιμές. Η συγκεκριμένη ρύθμιση μπορεί να μην είναι ιδιαίτερα χρήσιμη κατά την εκπαίδευση του ρομπότ σε ένα περιβάλλον που αποτελείται μόνο από στατικά εμπόδια, όταν όμως υπάρχουν δυναμικά εμπόδια στον χώρο η κίνηση προς τα πίσω είναι εξαιρετικά σημαντική. Για παράδειγμα έστω ότι το ρομπότ κινείται προς μία κατεύθυνση και ένα δυναμικό εμπόδιο οδεύει προς τα πάνω του. Εάν δεν είναι ενεργοποιημένη η ρύθμιση για οπίσθια κίνηση τότε το ρομπότ προκειμένου να αποφύγει το δυναμικό εμπόδιο θα πρέπει να εκτελέσει πρώτα περιστροφική κίνηση και έπειτα να προχωρήσει προς τα εμπρός. Αυτή η διαδικασία απαιτεί έναν συγκεκριμένο χρόνο για να εκτελεστεί που είναι κρίσμας για το αν θα προκύψει σύγκρουση με το εμπόδιο. Σε πολλές περιπτώσεις λοιπόν, η αδυναμία του ρομπότ να κινηθεί προς τα πίσω σε δυναμικά περιβάλλοντα οδηγεί σε αυξημένο αριθμό συγκρούσεων. Η συγκεκριμένη ρύθμιση ενεργοποιήθηκε στα Περιβάλλοντα 2 και 3 όπου υπάρχουν δυναμικά εμπόδια.

Stacking Enabled

Ενεργοποιώντας την ρύθμιση stacking αλλάζουμε τα δεδομένα εισόδου του πράκτορα. Η είσοδός του δεν αποτελείται πλέον από 44 τιμές που περιέχει ένα στιγμιότυπο (frame), αλλά αντιθέτως δημιουργείται ως είσοδος μία στοίβα από τρία frames της εκπαίδευσης. Σκοπός αυτού είναι να μάθει ο πράκτορας να χρησιμοποιεί και τα δεδομένα προηγούμενων βημάτων, έτσι ώστε να μπορέσει να προβλέψει την κίνηση δυναμικών εμποδίων. Όπως γίνεται εύκολα αντιληπτό είναι σχεδόν αδύνατο ακόμα και για ένα άνθρωπο να μπορέσει να προβλέψει την κίνηση ενός κινητού εμποδίου βλέποντας μόνο ένα στιγμιότυπο της κίνησης. Αν όμως έχει δεδομένα και από προηγούμενα στιγμιότυπα τότε γίνεται εφικτή η πρόβλεψη της κίνησης. Το ίδιο συμβαίνει και στην περίπτωση του πράκτορα. Η συγκεκριμένη ρύθμιση, όπως και η κίνηση προς τα πίσω, ενεργοποιήθηκε στα δύο περιβάλλοντα που περιέχουν και δυναμικά εμπόδια με σκοπό την ελάττωση των συγκρούσεων με αυτά.

Οι ρυθμίσεις Backward Enabled και Stacking Enabled δεν χρησιμοποιήθηκαν στο Περιβάλλον Δυσκολίας 1, καθώς θα αύξαναν την υπολογιστική πολυπλοκότητα χωρίς να προσφέρουν κάτι αποτελεσματικό στην εκπαίδευση του αλγορίθμου. Τα αποτελέσματα

και ο τρόπος που επηρέασε η κάθε αλλαγή των παραμέτρων θα αναλυθούν στην Ενότητα 5.

5 Πειραματικά Αποτελέσματα

Στην εν λόγω ενότητα παρουσιάζονται και αναλύονται τα πειραματικά αποτελέσματα που προέκυψαν από την εφαρμογή και εκπαίδευση των αλγορίθμων Ενισχυτικής Μάθησης. Στόχος της ανάλυσης είναι να διερευνηθεί η απόδοση του κάθε αλγορίθμου ως προς την ταχύτητα και τη σταθερότητα εκμάθησης, το ποσοστό επιτυχίας και γενικότερα την αποτελεσματικότητά του. Παράλληλα θα γίνει ανάλυση του τρόπου με τον οποίο επηρεάζει την απόδοσή των αλγορίθμων η ρύθμιση των παραμέτρων. Τέλος, θα πραγματοποιηθεί σύγκριση μεταξύ των αλγορίθμων, καθώς επίσης θα γίνει αναφορά σε προβλήματα και περιορισμούς που προέκυψαν κατά την εκπαίδευση. Τα αποτελέσματα τα οποία αναλύονται αφορούν κάποιες μετρικές οι οποίες εξάγονται κατά την διάρκεια της εκπαίδευσης. Οι μετρικές αυτές περιλαμβάνουν τη μέση σωρευτική ανταμοιβή ανά 10 επεισόδια, τη μέση απώλεια του actor και του critic δικτύου, καθώς και τις κατηγορίες τερματισμού (επιτυχία, σύγκρουση με τοίχο ή δυναμικό εμπόδιο, λήξη χρόνου, ανεξήγητος τερματισμός).

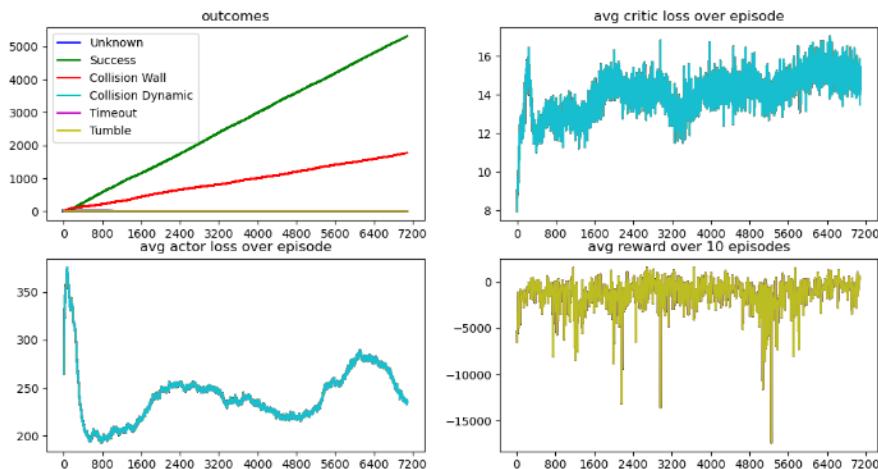
5.1 Αποτελέσματα Αλγορίθμου DDPG

Στους παρακάτω πίνακες παρουσιάζονται τα αποτελέσματα κάθε εκπαίδευσης του αλγορίθμου DDPG στα τρία διαφορετικά περιβάλλοντα. Κάτω από κάθε πίνακα, θα αναλυθεί ο τρόπος με τον οποίο η ρύθμιση των διαφόρων παραμέτρων οδήγησε στην σταδιακή βελτίωση της αποτελεσματικότητας του αλγορίθμου. Στις Εικόνες 20-24 παρουσιάζονται δειγματοληπτικά κάποια διαγράμματα που προκύπτουν από την εκάστοτε εκπαίδευση, βάση των οποίων προκύπτουν διάφορα συμπεράσματα για την σταδιακή βελτίωση των μοντέλων. Το διάγραμμα κάθε εκπαίδευσης αποτελείται από επιμέρους διαγράμματα τα οποία αφορούν τις καταστάσεις τερματισμού (επάνω αριστερά), το critic loss σε κάθε επεισόδιο (επάνω δεξιά), το actor loss σε κάθε επεισόδιο (κάτω αριστερά) και την μέση ανταμοιβή σε κάθε 10 επεισόδια (κάτω δεξιά).

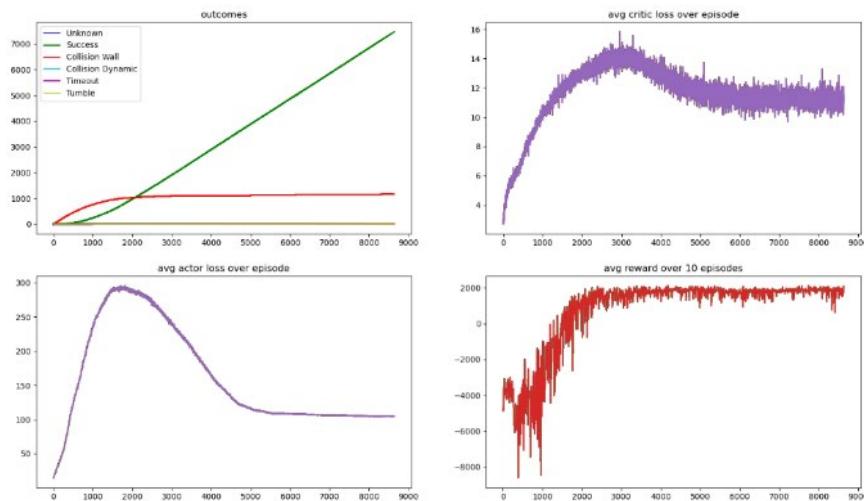
DDPG – Περιβάλλον Δυσκολίας 1								
	Learning Rate	τ	Hidden Size	Batch Size	Backward Enabled	Stack Enabled	Success Rate	Conversion Time (~episode)
Εκπαίδευση 1	0.003	0.003	512	128	False	False	70%	160
Εκπαίδευση 2	10^{-3}	10^{-3}	256	512	False	False	95%	600
Εκπαίδευση 3	10^{-4}	10^{-4}	512	512	False	False	100%	1600

Πίνακας 4: Αποτελέσματα DDPG - Περιβάλλον Δυσκολίας 1

Στο Περιβάλλον Δυσκολίας 1 παρατηρείται μεγάλο ποσοστό επιτυχίας (70%) από την πρώτη κιόλας εκπαίδευση με τις αρχικές ρυθμίσεις (Εικόνα 25(α)). Αυτό συμβαίνει διότι λόγω της ευκολίας του περιβάλλοντος, ακόμα και με μεγάλες τιμές στις υπερπαραμέτρους learning rate και τ , ο πράκτορας καταφέρνει να μάθει τον τρόπο ώστε να πλοηγείται στο περιβάλλον με σχετική επιτυχία. Μειώνοντας τις τιμές των υπερπαραμέτρων σε 10^{-3} έκαστος, σε συνδυασμό με αύξηση του batch size σε 512 και μείωση των κρυφών στρωμάτων από 512 σε 256, επιτυγχάνεται ραγδαία αύξηση του ποσοστού επιτυχίας σε 95%. Το μοντέλο συγκλίνει μετά την πάροδο περισσότερων επεισοδίων σε σχέση με την Εκπαίδευση 1, λόγω της μείωσης του learning rate και του τ , ωστόσο η σύγκλιση είναι σταθερότερη και αποτελεσματικότερη. Στην Εκπαίδευση 3 μειώνοντας εκ νέου τις τιμές των υπερπαραμέτρων σε 10^{-4} και διπλασιάζοντας τα κρυφά στρώματα ($256 \rightarrow 512$), επιτεύχθηκε το απόλυτο 100% ποσοστό επιτυχίας (Εικόνα 25(β)). Βραδύτερη σύγκλιση με βελτιωμένη σταθερότητα οδήγησε τον πράκτορα στο να μάθει πλήρως το πως πρέπει να κινείται στον χώρο έτσι ώστε να αποφύγει με επιτυχία τα στατικά εμπόδια και να πλοηγηθεί προς τον στόχο. Οι ρυθμίσεις Backward Enabled και Stack Enabled δεν χρειάστηκε να ενεργοποιηθούν στο Περιβάλλον Δυσκολίας 1 διότι δεν υπάρχουν δυναμικά εμπόδια και το μοντέλο μπορεί να ανταπεξέλθει εξίσου καλά στις απαιτήσεις του περιβάλλοντος χωρίς την επιβάρυνση στην υπολογιστική πολυπλοκότητα που θα προκαλούσαν.



(α): Εκπαίδευση 1



(β): Εκπαίδευση 3

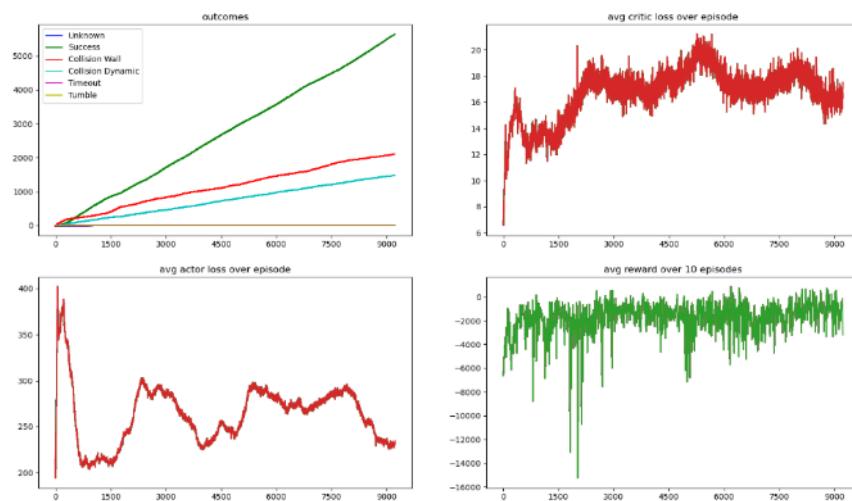
Εικόνα 25: Διαγράμματα Εκπαίδευσεων DDPG – Περιβάλλον Δυσκολίας 1

DDPG – Περιβάλλον Δυσκολίας 2								
	Learning Rate	τ	Hidden Size	Batch Size	Backward Enabled	Stack Enabled	Success Rate	Conversion Time (~episode)
Εκπαίδευση 1	0.003	0.003	512	128	False	False	63%	300
Εκπαίδευση 2	10^{-4}	10^{-4}	256	256	True	True	77%	1400
Εκπαίδευση 3	10^{-4}	10^{-3}	512	512	True	True	83%	1200
Εκπαίδευση 4	10^{-4}	10^{-3}	256	512	True	False	88%	700

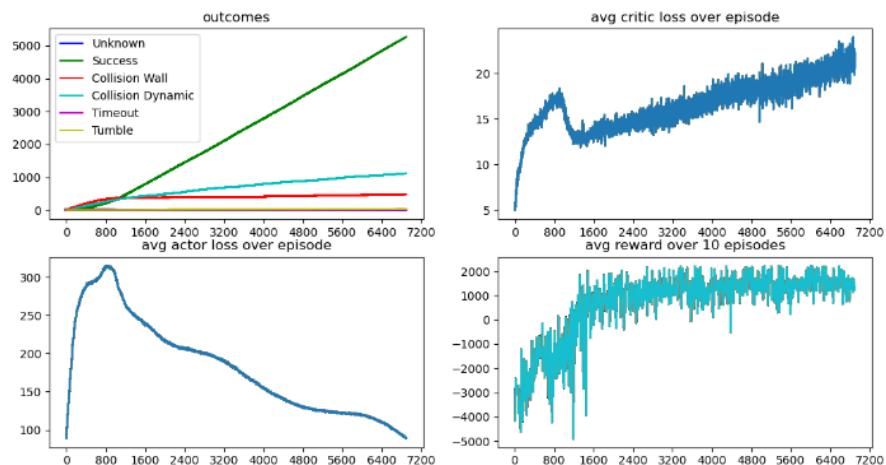
Πίνακας 5: Αποτελέσματα DDPG - Περιβάλλον Δυσκολίας 2

Όπως φαίνεται στον πίνακα 5 ο αλγόριθμος DDPG πέτυχε 63% ποσοστό επιτυχίας χρησιμοποιώντας τις προκαθορισμένες ρυθμίσεις (Εικόνα 26(α)). Στις επόμενες δύο εκπαίδευσεις που πραγματοποιήθηκαν ενεργοποιήσαμε τις ρυθμίσεις Backward Enabled και Stack Enabled. Τα ποσοστά επιτυχίας σε αυτές τις προσπάθειες αυξήθηκαν σε σχέση με την Εκπαίδευση 1. Στην Εκπαίδευση 2, με learning rate και τ να παίρνουν τιμή 10^{-4} και hidden και batch size ως 256 ο πράκτορας κατάφερε να πλοηγηθεί εντός του Περιβάλλοντος Δυσκολίας 2 με 77% επιτυχία. Οι αρκετά χαμηλές τιμές των υπερπαραμέτρων οδήγησαν το μοντέλο σε πιο αργή αλλά σταθερή σύγκλιση και η ενεργοποίηση των ρυθμίσεων Backward Enabled και Stack Enabled συνέβαλαν στην μείωση συγκρούσεων του πράκτορα με δυναμικά εμπόδια. Στην συνέχεια, στην Εκπαίδευση 3 διπλασιάστηκαν οι τιμές των hidden και batch size, αυξήθηκε το τ σε 10^{-3} ενώ το learning rate παρέμεινε σταθερό. Οι αλλαγές αυτές οδήγησαν σε μεγαλύτερο

ποσοστό επιτυχίας (83%), ενώ η σύγκληση του μοντέλου επιταχύνθηκε κατά 200 εποχές. Η μέγιστη απόδοση στο Περιβάλλον Δυσκολίας 2 επετεύχθη απενεργοποιώντας την ρύθμη Stack Enabled στην Εκπαίδευση 4 (88%). Το συγκεκριμένο περιβάλλον περιέχει μόνο δύο δυναμικά εμπόδια, με αποτέλεσμα ο πράκτορας να έρχεται σχετικά σπάνια κοντά σε κάποιο από αυτά. Η απενεργοποίηση της ρύθμισης Stack Enabled σε συνδιασμό με hidden size 256, μείωσε την πολυπλοκότητα του μοντέλου επιτρέποντας σε αυτό να συγκλίνει πιο γρήγορα και πιο ουσιαστικά, χωρίς να μειώσει την απόδοση του πράκτορα στην αποφυγή δυναμικών εμποδίων (Εικόνα 26(β)). Συμπερασματικά, ο πράκτορας στο Περιβάλλον Δυσκολίας 2 φαίνεται να ανταποκρίνεται καλύτερα με την ενεργοποίηση μόνο της ρύθμισης Backward Enabled.



(α): Εκπαίδευση 1



(β): Εκπαίδευση 4

Εικόνα 26: Διαγράμματα Εκπαίδευσεων DDPG – Περιβάλλον Δυσκολίας 2

DDPG – Περιβάλλον Δυσκολίας 3

	Learning Rate	τ	Hidden Size	Batch Size	Backward Enabled	Stack Enabled	Success Rate	Conversion Time (~episode)
Εκπαίδευση 1	0.003	0.003	512	128	False	False	28%	300
Εκπαίδευση 2	10^{-4}	10^{-4}	256	256	True	True	68%	1700
Εκπαίδευση 3	10^{-4}	10^{-4}	256	512	True	True	72%	2200
Εκπαίδευση 4	10^{-4}	10^{-4}	512	512	True	True	76%	2700
Εκπαίδευση 5	10^{-4}	10^{-3}	512	512	True	True	79%	500
Εκπαίδευση 6	10^{-4}	10^{-3}	1024	512	True	True	68%	900
Εκπαίδευση 7	10^{-3}	10^{-3}	1024	512	True	True	55%	350

Πίνακας 6: Αποτελέσματα DDPG - Περιβάλλον Δυσκολίας 3

Σύμφωνα με τον Πίνακα 6 παρατηρείται σταδιακή βελτίωση μέχρι το μοντέλο της 5^{ης} Εκπαίδευσης, όπου έχουμε και την μεγαλύτερη τιμή ποσοστού επιτυχίας. Ας εξετάσουμε την κάθε παράμετρο και το πως επιδρά στην εκπαίδευση ξεχωριστά:

Learning Rate:

Παρατηρούμε ότι υψηλές τιμές LR (Εκπ.1, Εκπ.7) συμβάλλουν στο να συγκλίνει γρηγορότερα το μοντέλο (300-350 επεισόδια), αλλά σε κάποιο τοπικό ελάχιστο και συνάμα σε μία υποδεέστερη πολιτική με σχετικά χαμηλό ποσοστό επιτυχίας (Εικόνα 27(α), (δ)). Αντιθέτως, μία μικρότερη τιμή (10^{-4}) επιτρέπει στον πράκτορα να μάθει σε βάθος χρόνου καλύτερες πολιτικές αυξάνοντας σημαντικά το ποσοστό επιτυχίας (Εκπ.2 – Εκπ.6) (Εικόνα 27(β), (γ)).

Παράμετρος τ :

Λειτουργεί ως σταθεροποιητής. Μικρές τιμές (10^{-4}) σε συνδυασμό με μικρό LR προσδίδει πολύ αργή αλλά σταθερή σύγκλιση, καθώς τα target networks αργούν να προσαρμοστούν (Εικόνα 27(β)). Αυξάνοντας την τιμή του σε (10^{-3}) κρατώντας ταυτόχρονα χαμηλά την τιμή του LR, παρατηρούμε ταχύτερη σύγκλιση και οδηγούμαστε σε ακόμα ακριβέστερα μοντέλα (Εκπ.5) (Εικόνα 27(γ)).

Hidden size:

Από τα αποτελέσματα πηγάζει ότι η τιμή 512 είναι η επικρατέστερη και ταιριάζει περισσότερο στην πολυπλοκότητα του συστήματός μας. Μικρότερες τιμές δεν αποδίδουν

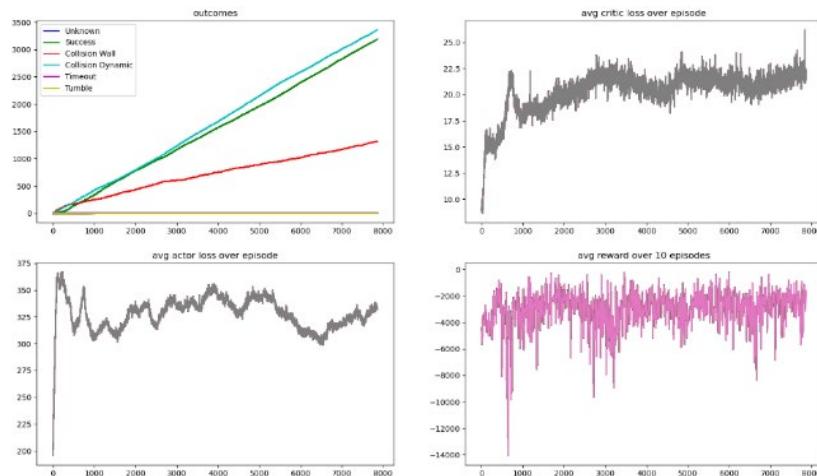
εξίσου καλά λόγω της έλλειψης βάθους του δικτύου για τα δεδομένα τα οποία έχουμε. Αντιθέτως μεγαλύτερες τιμές σε συνδυασμό με υψηλό LR και τ , αυξάνουν την υπολογιστική πολυπλοκότητα και παρ'όλο που το μοντέλο φαίνεται να αποδίδει καλύτερα κατά την εκπαίδευση, στην πραγματικότητα το ποσοστό επιτυχίας μειώνεται ($79\% \rightarrow 68\%$) λόγω overfitting. Παρατηρούμε στην Εικόνα 27(δ) ότι η τιμή actor loss παίρνει πολύ αρνητικές τιμές (≈ -1200), το οποία αποτελεί επίσης δείγμα overfitting. Επίσης το average reward είναι θετικό αλλά έχει πολύ μεγάλη διακύμανση.

Batch size:

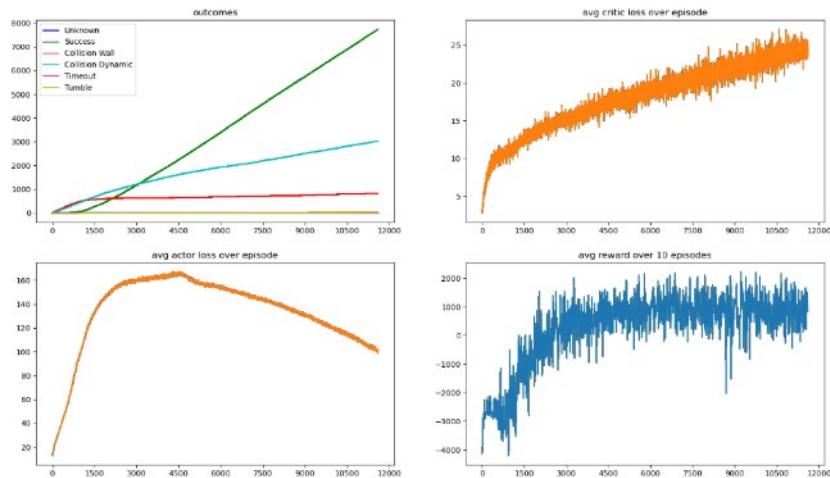
Η μετάβαση από 256 σε 512 βελτίωσε τη σταθερότητα και το ποσοστό επιτυχίας ($68\% \rightarrow 72\%$) αυξάνοντας σε επιτρεπτά πλαίσια το υπολογιστικό κόστος. Αυτό έχει ως αποτέλεσμα πιο αξιόπιστες ενημερώσεις κλίσεων και συνάμα πιο σταθερή σύγκλιση.

Backward και Stacking:

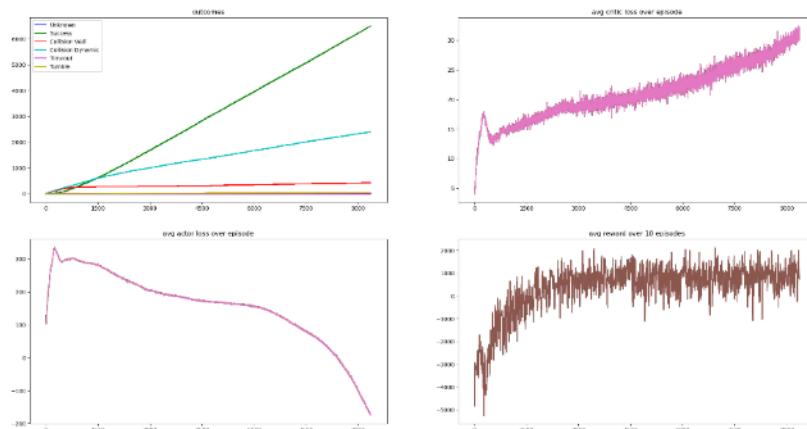
Η ενεργοποίησή τους, όπως ήταν αναμενόμενο, σε συνδυασμό και με ρύθμιση των υπόλοιπων παραμέτρων εκτοξεύει το ποσοστό επιτυχίας του πράκτορα εντός του Περιβάλλοντος Δυσκολίας 3 από ένα αρκετά χαμηλό ποσοστό 28% μέχρι και το βέλτιστο 79%.



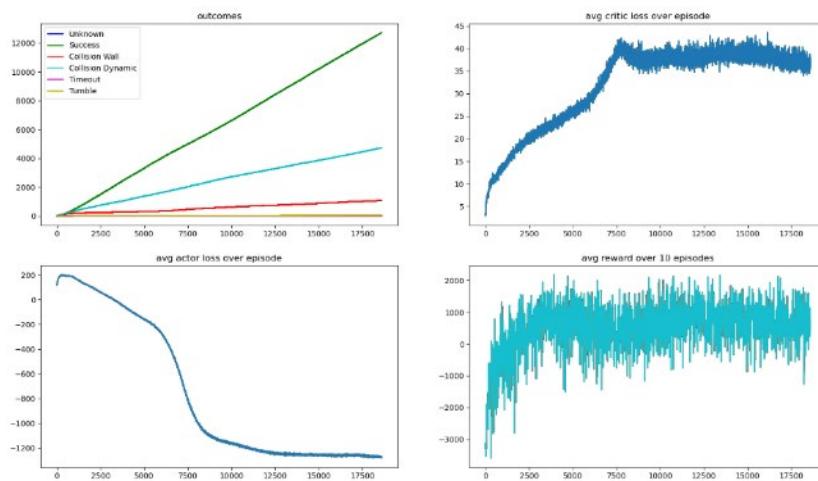
(α): Εκπαίδευση 1



(β): Εκπαίδευση 4



(γ): Εκπαίδευση 5



(δ): Εκπαίδευση 7

Εικόνα 27: Διαγράμματα Εκπαίδευσεων DDPG – Περιβάλλον Δυσκολίας 3

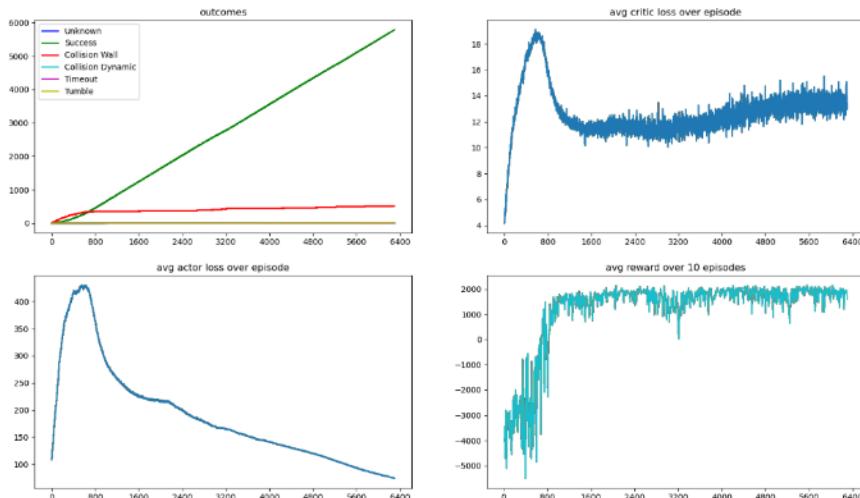
5.2 Αποτελέσματα Αλγορίθμου TD3

Στους παρακάτω πίνακες παρουσιάζονται τα αποτελέσματα κάθε εκπαίδευσης του αλγορίθμου TD3 στα τρία διαφορετικά περιβάλλοντα.

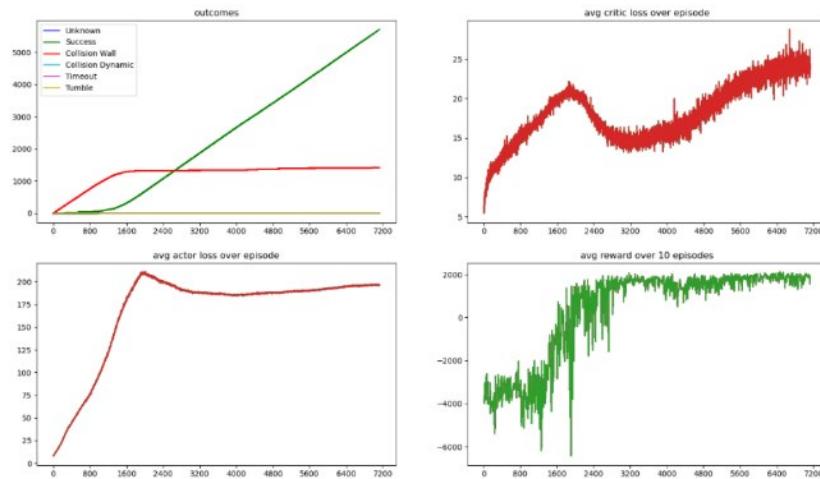
TD3 – Περιβάλλον Δυσκολίας 1								
	Learning Rate	τ	Hidden Size	Batch Size	Backward Enabled	Stack Enabled	Success Rate	Conversion Time (~episode)
Εκπαίδευση 1	0.003	0.003	512	128	False	False	94%	400
Εκπαίδευση 2	10^{-4}	10^{-3}	512	512	False	False	100%	1400

Πίνακας 7: Αποτελέσματα TD3 – Περιβάλλον Δυσκολίας 1

Ο αλγόριθμος TD3 δείχνει να αποδίδει εξαιρετικά ακόμα και με τις προκαθορισμένες ρυθμίσεις στο Περιβάλλον Δυσκολίας 1, με ένα εμβληματικό ποσοστό επιτυχίας 94% σε σχέση με οποιαδήποτε άλλη εκπαίδευση που πραγματοποιήθηκε με τις ρυθμίσεις αυτές. Στην Εκπαίδευση 2, μειώνοντας learning rate και τ σε 10^{-4} και 10^{-3} έκαστος, επιτεύχθηκε το απόλυτο ποσοστό επιτυχίας. Οι τιμές για hidden και batch size που χρησιμοποιήθηκαν είναι 512 και η συμβολή τους σε συνδυασμό με hyperparameter tuning ήταν καθοριστική στην επίτευξη 100% επιτυχίας, οδηγώντας το μοντέλο σε σταθερή, γρήγορη και βέλτιστη σύγκλιση.



(α): Εκπαίδευση 1



(β): Εκπαίδευση 2

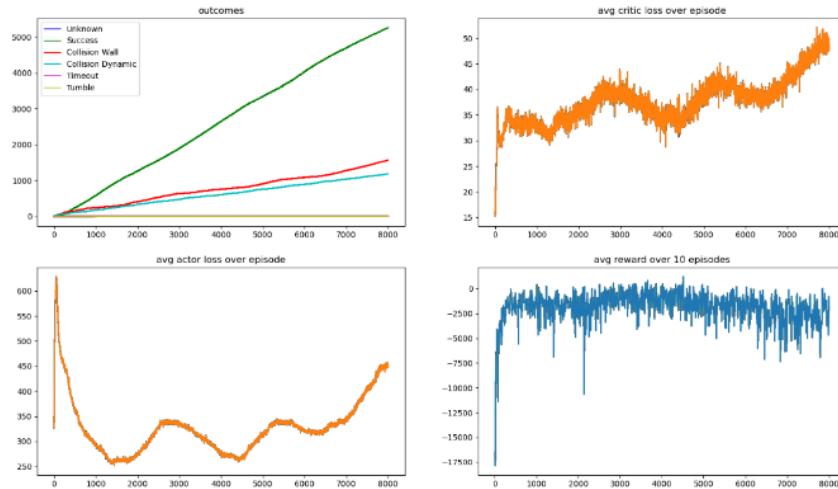
Εικόνα 28: Διαγράμματα Εκπαίδευσεων TD3 – Περιβάλλον Δυσκολίας 1

TD3 – Περιβάλλον Δυσκολίας 2

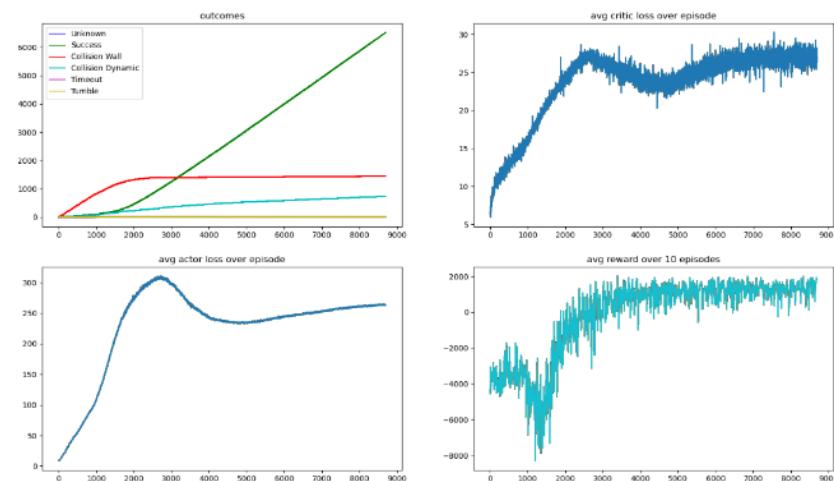
	Learning Rate	τ	Hidden Size	Batch Size	Backward Enabled	Stack Enabled	Success Rate	Conversion Time (~episode)
Εκπαίδευση 1	0.003	0.003	512	128	False	False	43%	150
Εκπαίδευση 2	10^{-4}	10^{-4}	512	512	True	False	96%	1800
Εκπαίδευση 3	10^{-4}	10^{-3}	512	512	True	False	92%	900
Εκπαίδευση 4	10^{-4}	10^{-3}	512	512	True	True	87%	1100

Πίνακας 8: Αποτελέσματα TD3 – Περιβάλλον Δυσκολίας 2

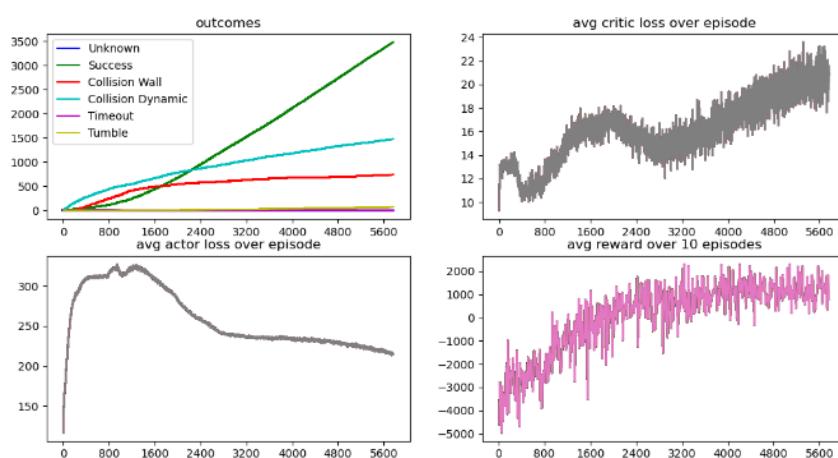
Ο αλγόριθμος TD3 με τις αρχικές ρυθμίσεις απέδωσε 43% ποσοστό επιτυχίας. Οι υψηλές τιμές των ρυθμίσεων learning rate και το οδήγησαν σε άμεση σύγκληση, ωστόσο η πολιτική στην οποία σύγκλινε το μοντέλο δεν ήταν η βέλτιστη δυνατή. Στην Εκπαίδευση 2 παρατηρήθηκε εξαιρετική ανταπόκριση στις συγκεκριμένες ρυθμίσεις. Το μοντέλο κατάφερε να φτάσει σε επιδώσεις σχεδόν άριστες, με το ποσοστό επιτυχίας να ανέρχεται σε 96%. Όπως φαίνεται και στα διαγράμματα τις Εικόνας 29(β), το μοντέλο συγκλίνει περίπου στα 1800 επεισόδια και οι απώλειες των actor και critic δικτύων σταθεροποιούνται περίπου στα 4000 επεισόδια. Η μέση ανταμοιβή στο τέλος της εκπαίδευσης έχει μικρές διακυμάνσεις σε μεγάλες τιμές, αντιπροσωπευτική ένδειξη του εξαιρετικού ποσοστού επιτυχίας του μοντέλου. Στις επόμενες δύο εκπαίδευσεις δοκιμάστηκε η μείωση της υπερπαραμέτρου τ από 10^{-4} σε 10^{-3} και η ενεργοποίηση της ρύθμισης Stack Enabled. Αποτέλεσμα αυτού ήταν η μείωση του χρόνου σύγκλισης, μειώνοντας όμως ταυτόχρονα και το ποσοστό επιτυχίας ($96\% \rightarrow 92\% \rightarrow 87\%$).



(α): Εκπαίδευση 1



(β): Εκπαίδευση 2



(γ): Εκπαίδευση 4

Εικόνα 29: Διαγράμματα Εκπαίδευσεων TD3 – Περιβάλλον Δυσκολίας 2

TD3 – Περιβάλλον Δυσκολίας 3

	Learning Rate	τ	Hidden Size	Batch Size	Backward Enabled	Stack Enabled	Success Rate	Conversion Time
Εκπαίδευση 1	0.003	0.003	512	128	False	False	16%	350
Εκπαίδευση 2	10^{-4}	10^{-4}	256	256	True	True	71%	1950*
Εκπαίδευση 3	10^{-4}	10^{-4}	512	512	True	True	73%	2400
Εκπαίδευση 4	10^{-4}	10^{-3}	512	512	True	True	73%	1650*
Εκπαίδευση 5	10^{-4}	10^{-3}	1024	512	True	True	80%	2800
Εκπαίδευση 6	10^{-3}	10^{-3}	1024	512	True	True	63%	800

Πίνακας 9: Αποτελέσματα TD3 – Περιβάλλον Δυσκολίας 3

Στον Πίνακα 9 αναγράφονται οι ρυθμίσεις και τα αποτελέσματα των εκπαιδεύσεων του αλγορίθμου TD3 στο Περιβάλλον Δυσκολίας 3. Ας εξετάσουμε την κάθε παράμετρο και το πως επιδρά στην εκπαίδευση ξεχωριστά:

Learning Rate:

Υψηλές τιμές LR (Εκπ.1, Εκπ.6) οδηγούν σε πιο άμεση σύγκλιση (350/800 επεισόδια), ωστόσο εγκλωβίζονται σε κάποιο τοπικό ελάχιστο και συνάμα σε μία υποδεέστερη πολιτική με σχετικά χαμηλό ποσοστό επιτυχίας (Εικόνα 30(α), (δ)). Αντιθέτως, η τιμή 10^{-4} συμβάλει στην αύξηση του ποσοστού επιτυχίας των μοντέλων, καθώς επιτρέπει στον πράκτορα να μάθει καλύτερες πολιτικές κάνοντας μικρότερα βήματα σύγκλησης (Εκπ.2 – Εκπ.5) (Εικόνα 30(β), (γ)).

Παράμετρος τ :

Μικρές τιμές (10^{-4}) σε συνδυασμό με μικρό LR επιτρέπουν σταθερή σύγκλιση. Τα target networks αργούν να προσαρμοστούν με αποτέλεσμα της επιβράδυνση της σύγκλησης (Εικόνα 30(β)). Αυξάνοντας την τιμή του σε (10^{-3}) παρατηρούμε ταχύτερη σύγκλιση ενώ αυξάνεται ταυτόχρονα και το ποσοστό επιτυχίας του μοντέλου (Εκπ.4, Εκπ.5).

Hidden size:

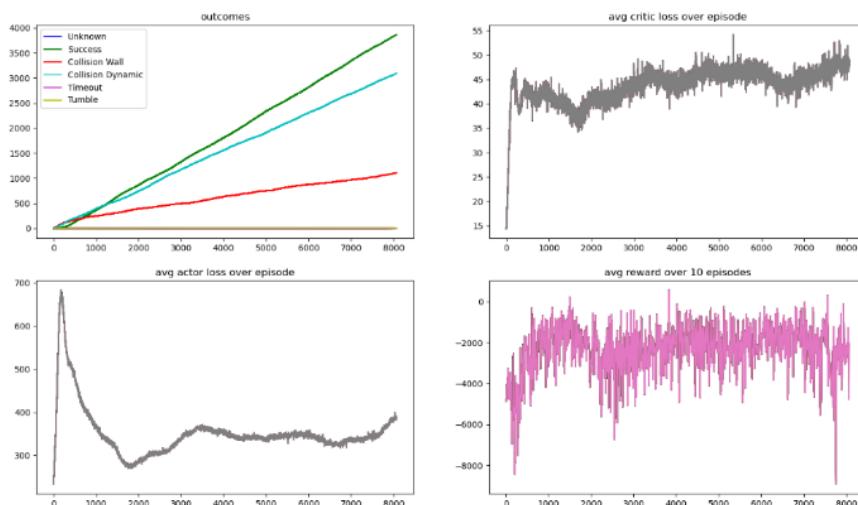
Στον αλγόριθμο TD3 η τιμή 1024 φαίνεται να είναι η βέλτιστη στο Περιβάλλον Δυσκολίας 3. Στην Εκπαίδευση 5, όπου επιτυγχάνεται και το μεγαλύτερο ποσοστό επιτυχίας (80%), παρατηρείται ότι η τιμή 1024 παρ' όλο' που επιβραδύνει την σύγκληση λόγω αύξησης του βάθους του μοντέλου, οδηγεί σε βελτιωμένα αποτελέσματα. Μικρότερες τιμές δεν επιτρέπουν στον πράκτορα να διαχειριστεί κατάλληλα τον όγκο δεδομένων που λαμβάνει με αποτέλεσμα να έχει μειωμένη απόδοση (Εικόνα 30(δ)).

Batch size:

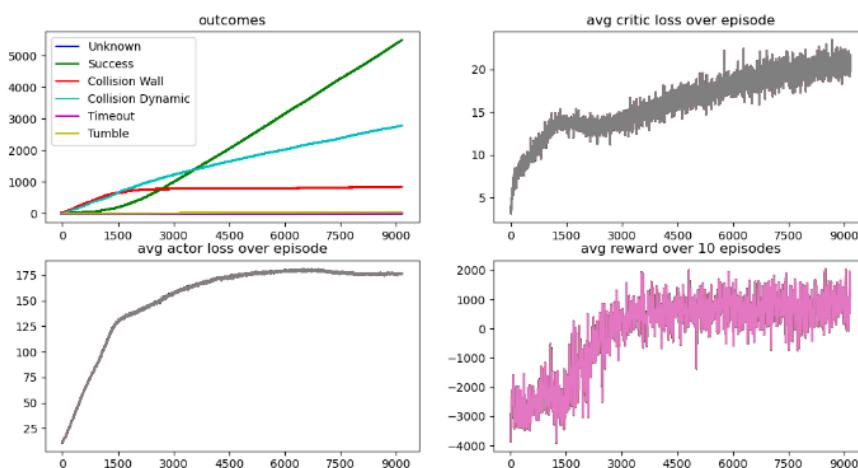
Η σταδιακή αύξηση του batch size βελτίωσε τη σταθερότητα των μοντέλων και αποτέλεσε κύριο παράγοντα στις εκπαιδεύσεις με μεγάλα ποσοστά επιτυχίας, προσφέροντας σταθερότερη και ουσιαστικότερη σύγκληση. Ιδανική αποδείχθηκε η τιμή 512, η οποία οδήγησε το μοντέλο στα μεγαλύτερα ποσοστά επιτυχίας (Εκπ.5).

Backward και Stacking:

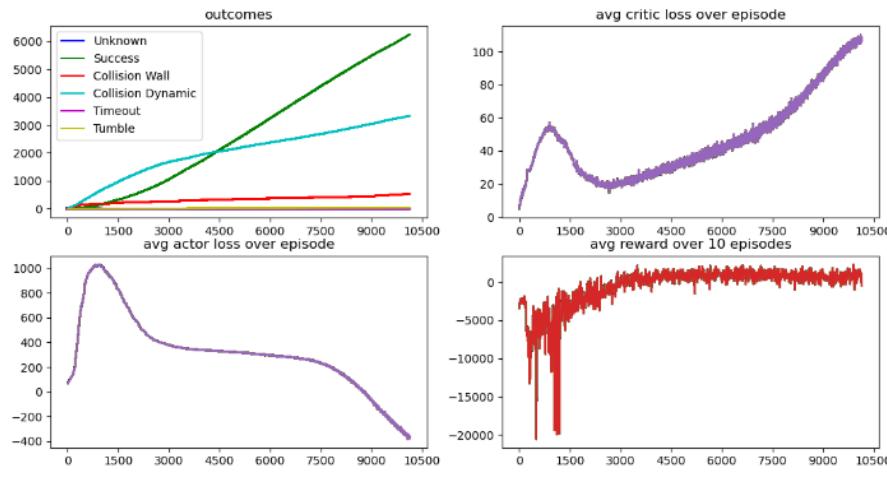
Η ενεργοποίησή των ρυθμίσεων αυτών, σε συνδυασμό και με ρύθμιση των υπόλοιπων παραμέτρων έπαιξε καθοριστικό ρόλο στην ραγδαία αύξηση των ποσοστών επιτυχίας του αλγορίθμου TD3 στο Περιβάλλοντος Δυσκολίας 3. Με απενεργοποιημένες τις ρυθμίσεις το μοντέλο είχε μόλις 16% επιτυχία (Εκπ.1) και η ενεργοποίησή του οδήγησε σε άμεση αύξηση αυτού του ποσοστού στις υπόλοιπες εκπαιδεύσεις, όπως διαφαίνεται μέσα από του Πίνακα 9



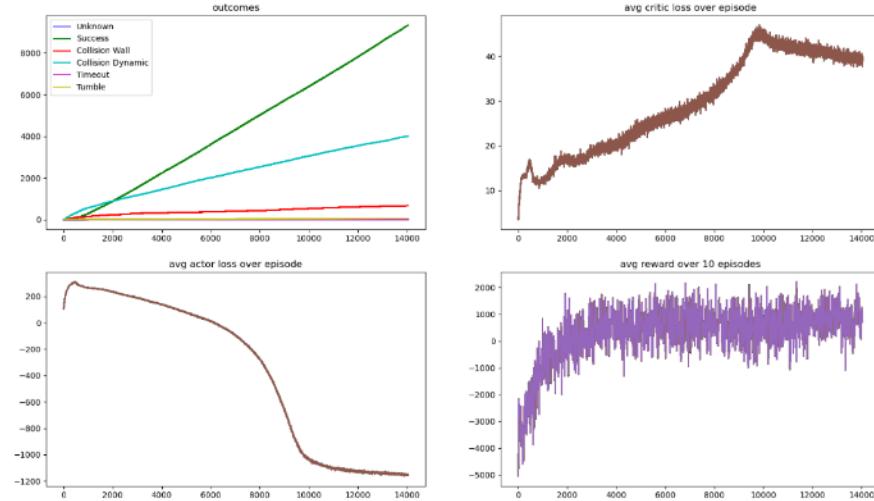
(α): Εκπαίδευση 1



(β): Εκπαίδευση 3



(γ): Εκπαίδευση 5



(δ): Εκπαίδευση 6

Εικόνα 30: Διαγράμματα Εκπαίδευσεων TD3 – Περιβάλλον Δυσκολίας 3

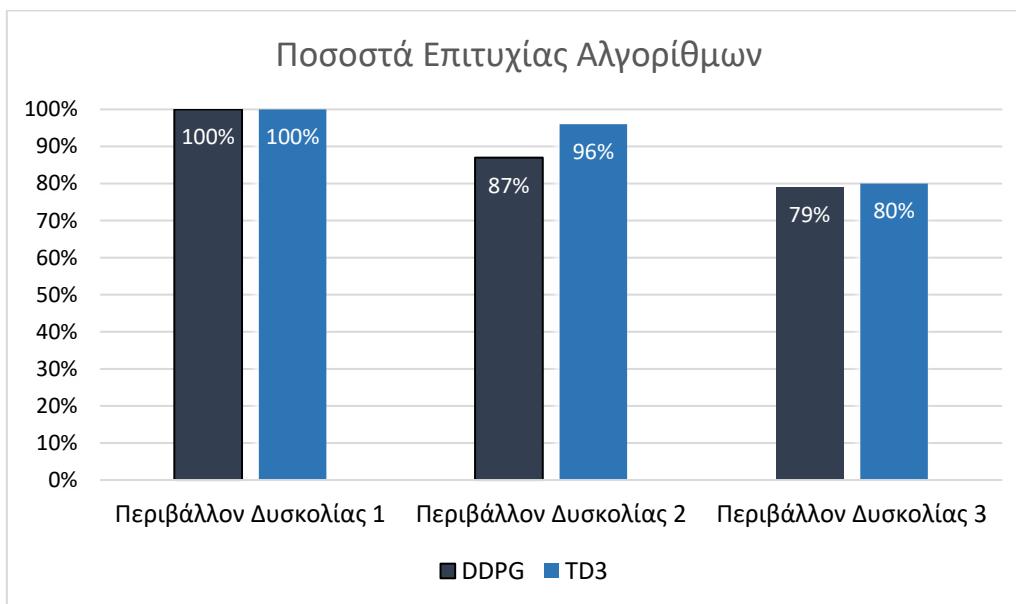
5.3 Σύγκριση Μεθόδων

Στην συγκεκριμένη υποενότητα θα πραγματοποιηθεί σύγκριση των δύο αλγορίθμων DDPG και TD3 ως προς το ποσοστό επιτυχίας, την ταχύτητα σύγκλησης και την απόδοσή τους κατά την προσαρμογή των παραμέτρων, ερμηνεύοντας παράλληλα την συμπεριφορά τους σε κάθε περιβάλλον.

5.3.1 Ποσοστό Επιτυχίας και Ταχύτητα Σύγκλησης

Τόσο ο αλγόριθμος DDPG όσο και ο TD3 ανταποκρίθηκαν με επιτυχία στις απαιτήσεις του πειράματος. Στο Περιβάλλον Δυσκολίας 1, οι δύο αλγόριθμοι πέτυχαν το απόλυτο ποσοστό επιτυχίας έπειτα από ρύθμιση των παραμέτρων τους. Λόγω ρυθμίσεων της παραμέτρου τ , το μοντέλο του TD3 συγκλίνει πιο γρήγορα από αυτό του DDPG (1400

έναντι 1600 επεισοδίων). Στο Περιβάλλον Δυσκολίας 2, το μέγιστο ποσοστό επιτυχίας που επιτυγχάνεται με την χρήση του αλγορίθμου TD3 είναι 96%, έναντι 88% που επιτυγχάνεται με την χρήση του DDPG. Τα ποσοστά αυτά προκύπτουν από δύο μοντέλα τα οποία συγκλίνουν με διαφορετικό τρόπο το ένα με το άλλο. Το μοντέλο του αλγορίθμου DDPG συγκλίνει έπειτα από περίπου 700 επεισόδια, ενώ το μοντέλο του TD3 χρειάζεται περισσότερο χρόνο για να οδηγηθεί σε σύγκληση (περίπου 1800 επεισόδια). Τέλος, οι δύο αλγόριθμοι είχαν παρόμοια ποσοστά επιτυχίας στο Περιβάλλον Δυσκολίας 3. Η εκπαίδευση του πράκτορα με την χρήση του αλγορίθμου DDPG σύγκλινε ταχύτατα (500 επεισόδια) έπειτα από ρύθμιση των παραμέτρων, σε τελική μέγιστη απόδοση 79%. Αντιθέτως, ο αλγόριθμος TD3 με σκοπό να πετύχει ποσοστό απόδοσης 80%, χρησιμοποιήθηκαν τιμές παραμέτρων οι οποίες οδήγησαν σε πολύ αργή σύγκληση μετά από σχεδόν 2800 επεισόδια. Στην Εικόνα 31 αναγράφονται τα ποσοστά επιτυχίας των αλγορίθμων σε κάθε περιβάλλον.

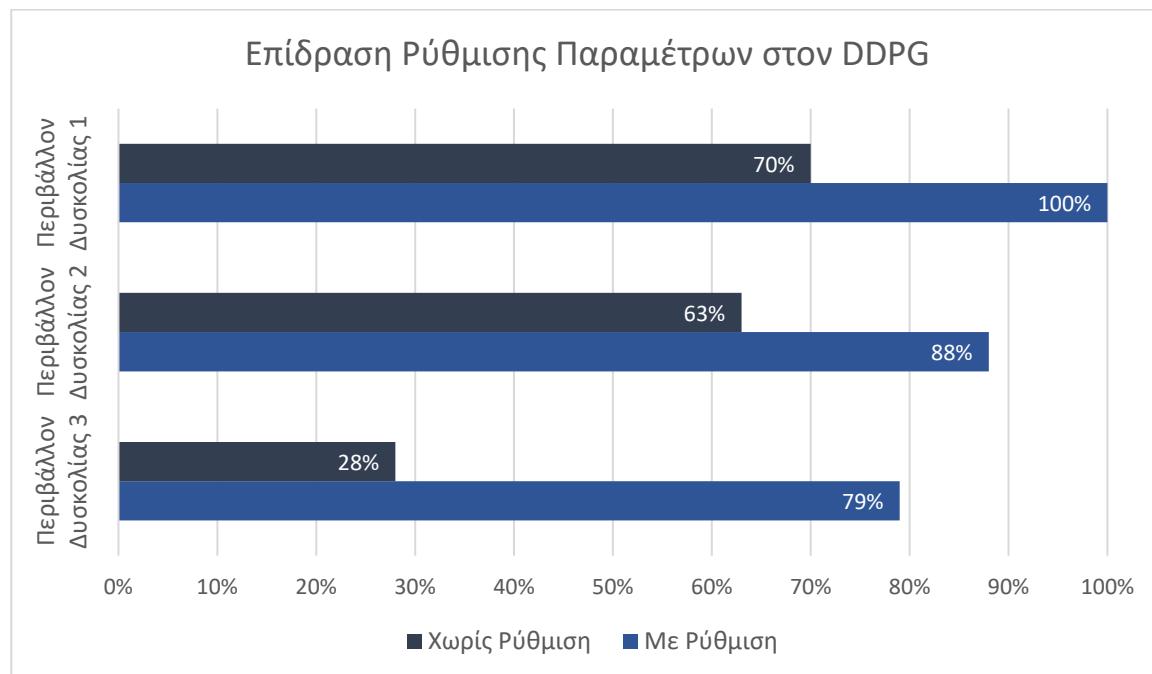


Εικόνα 31: Ποσοστά Επιτυχίας Αλγορίθμων DDPG και TD3

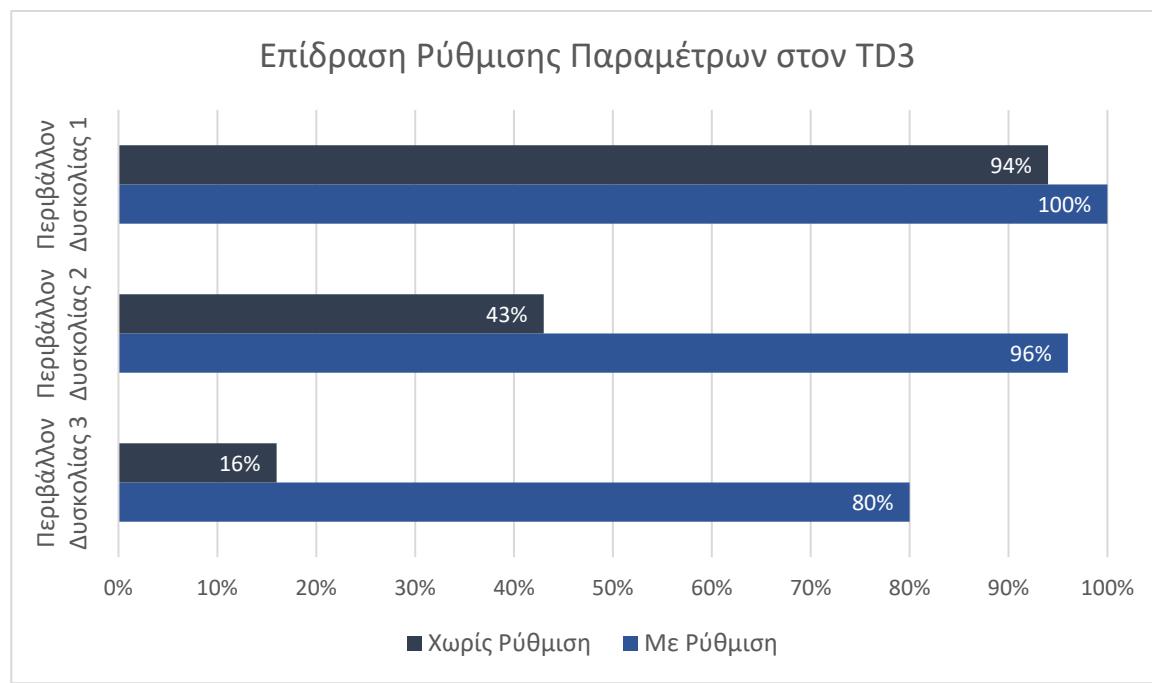
5.3.2 Ρύθμιση Παραμέτρων

Ο κάθε αλγόριθμος ανταποκρίθηκε διαφορετικά στην εφαρμογή ρύθμισης των παραμέτρων του. Τόσο αλγόριθμος DDPG όσο και ο TD3 φαίνεται να επωφελούνται σημαντικά από αλλαγές στις ρυθμίσεις των παραμέτρων. Όπως παρατηρήθηκε, οι αλγόριθμοι με τις αρχικές ρυθμίσεις είχαν χαμηλά ποσοστά επιτυχίας, ειδικά στα περιβάλλοντα με δυναμικά εμπόδια, ενώ έπειτα από αλλαγές των τιμών τους τα μοντέλα κατάφεραν να αγγίξουν μεγάλες επιδώσεις. Ιδιαίτερης σημασίας αποδείχθηκαν οι ρυθμίσεις Backward Enabled και Stack Enabled, οι οποίες σε συνδυασμό με μικροαλλαγές στις υπόλοιπες βοήθησαν τους πράκτορες να αποφύγουν σε μεγάλο ποσοστό τα δυναμικά εμπόδια στα Περιβάλλοντα Δυσκολίας 2 και 3 και να μπορέσουν να πλοηγηθούν με επιτυχία σε αυτά. Στις Εικόνες 32 και 33 αναγράφονται τα ποσοστά

επιτυχίας των αλγορίθμων σε κάθε περιβάλλον πρωτού και έπειτα από ρύθμιση των παραμέτρων τους.



Εικόνα 32: Ποσοστά Επιτυχίας Αλγορίθμου DDPG πριν και μετά την Ρύθμιση Παραμέτρων



Εικόνα 33: Ποσοστά Επιτυχίας Αλγορίθμου TD3 πριν και μετά την Ρύθμιση Παραμέτρων

6 Εφαρμογή σε πραγματικό ρομπότ

Αφού αξιολογήθηκε η απόδοση των υπό μελέτη αλγορίθμων DRL σε περιβάλλοντα προσομοίωσης, κρίθηκε αναγκαίο να εξεταστεί η λειτουργικότητά τους σε ένα πραγματικό περιβάλλον. Η μετάβαση από ένα σύστημα προσομοίωσης σε ένα πραγματικό σύστημα (sim-to-real) επιτρέπει τη δοκιμή της ανθεκτικότητας των μοντέλων σε απρόβλεπτους παράγοντες του πραγματικού κόσμου, όπως είναι ο θόρυβος των αισθητήρων, οι μη ιδανικές συνθήκες ελέγχου και η διαφορετικότητα-ιδιαιτερότητα του κάθε φυσικού περιβάλλοντος. Οι προεκπαιδευμένοι αλγόριθμοι εφαρμόστηκαν χωρίς επανεκπαίδευση, με στόχο την απευθείας αξιολόγηση της αποδοτικότητάς τους σε πραγματικές συνθήκες.

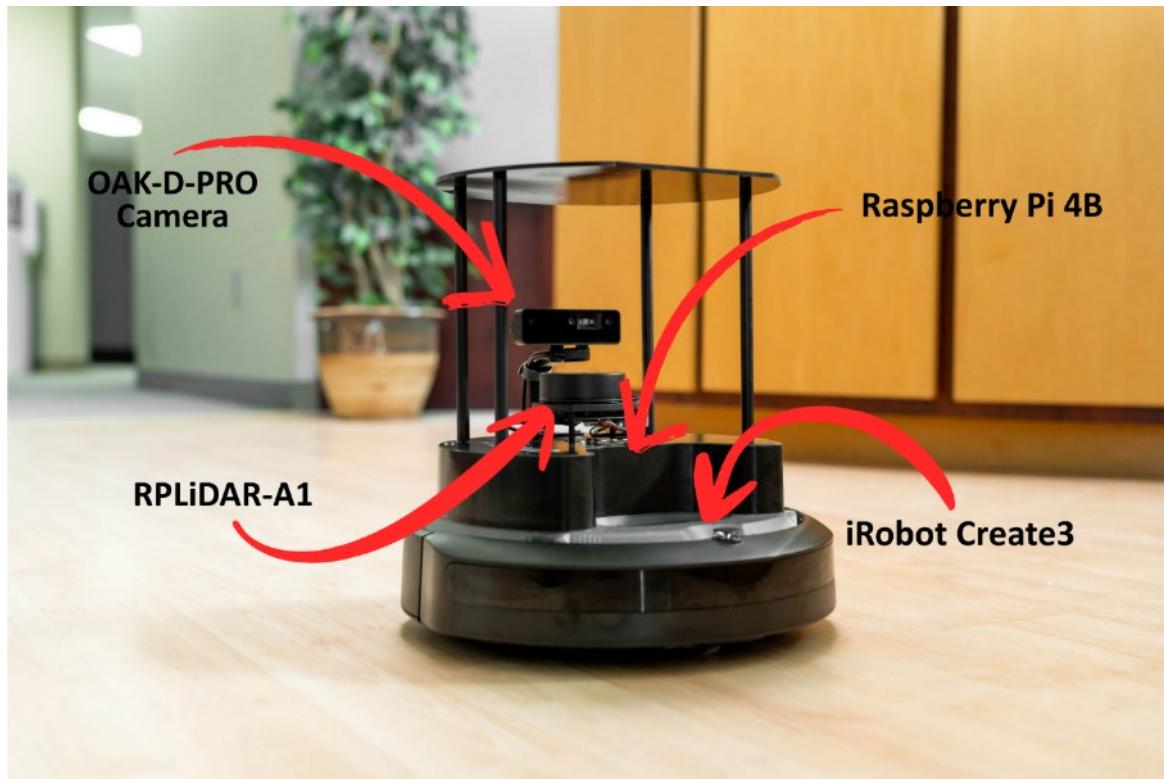
6.1 Περιγραφή του Συστήματος

Το Πραγματικό Ρομπότ

Για την υλοποίηση των αλγορίθμων σε πραγματικό περιβάλλον χρησιμοποιήθηκε το ρομπότ του εργαστηρίου, το TurtleBot4 [31]. Το TurtleBot4 είναι το τελευταίο μοντέλο της σειράς των TurtleBot το οποίο αναπτύχθηκε από την εταιρία Clearpath Robotics. Είναι μία ρομποτική πλατφόρμα ανοιχτού κώδικα με αναβαθμισμένους αισθητήρες και ενισχυμένη υπολογιστική ισχύ, η οποία διακρίνεται για την ευχρηστία της και την ολοκληρωμένη εμπειρία χρήστη που προσφέρει σε συνδυασμό με το χαμηλό της κόστος. Μία plug-and-play λύση πλήρως συμβατή με το ROS2, η οποία προορίζεται για εκπαίδευτική και ερευνητική χρήση στον χώρο της ρομποτικής.

Στην Εικόνα 34 παρουσιάζεται το ρομπότ του εργαστηρίου και στον Πίνακα 10 αναγράφονται μερικές από τις σημαντικότερες προδιαγραφές του hardware του. Η φυσική κατασκευή του TurtleBot4 αποτελείται από την βάση, την υπολογιστική μονάδα και τους αισθητήρες.

- Βάση:** Το TurtleBot4 χρησιμοποιεί ως βάση το iRobot Create3 μία ρομποτική πλατφόρμα συμβατή με ROS2, με τροχούς διαφορικής οδήγησης, αισθητήρες πρόσκρουσης και ακριβή εντοπισμού θέσης. Υποστηρίζει βάρος φορτίου από 9 έως 15 κιλά και έχει μέγιστη ταχύτητα 0,31 m/s.
- Υπολογιστική μονάδα:** Το TurtleBot4 περιλαμβάνει ένα Raspberry Pi 4B (4 GB ή 8 GB RAM), το οποίο λειτουργεί ως onboard υπολογιστής με προεγκατεστημένο λειτουργικό σύστημα Ubuntu 22.04 και ROS 2 Humble.
- Αισθητήρες:**
 - LiDAR:** 2D 360° RPLIDAR-A1,
 - Κάμερα:** OAK-D-PRO (4K RGB auto focus camera).



Εικόνα 34: TurtleBot 4

Στοιχείο	Τιμή
Μέγιστη γραμμική ταχύτητα	0.31 m/s
Μέγιστη περιστροφική ταχύτητα	1.9 rad/s (108.86 deg/s)
Μέγιστο φορτίο	9 kg
Μέγιστο φορτίο (προσαρμοσμένη διαμόρφωση)	15 kg
Διαστάσεις (M x Π x Y)	341 mm x 339 mm x 351 mm
Βάρος	3.9 kg
Αναμενόμενος χρόνος λειτουργίας (εξαρτάται από το φορτίο)	2.5 – 4.0 hrs
Αναμενόμενος χρόνος για πλήρη φόρτιση	2.5 hrs
Υπολογιστική Μονάδα	Raspberry Pi 4B (4GB)
Βάση	iRobot Create3
LDS (Laser Distance Sensor)	2D 360° RPLIDAR-A1
Κάμερα	OAK-D-PRO (4K RGB auto focus camera)
IMU	Gyroscope 3 Axis Accelerometer 3 Axis

Πίνακας 10: Προδιαγραφές Hardware του Turtlebot4

Επαναδειγματοληψία

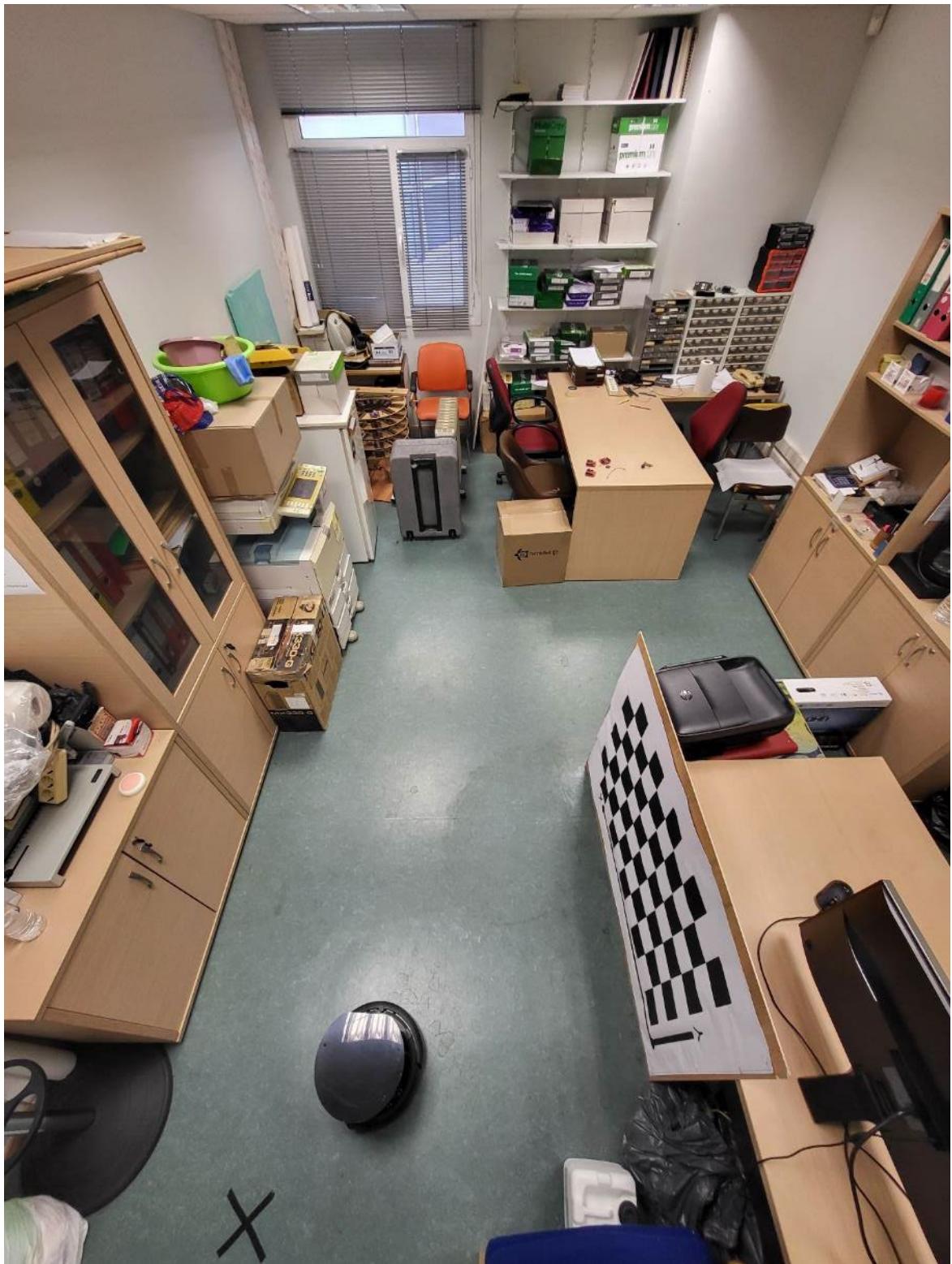
Κατά την εκπαίδευση στο εικονικό περιβάλλον χρησιμοποιήθηκαν 40 επαναδειγματοληπτούμενες τιμές LiDAR του TurtleBot3. Η ίδια διαδικασία χρειάζεται να γίνει και στην πραγματική υλοποίηση, δηλαδή πρέπει να πραγματοποιηθεί δειγματοληψία των LiDAR τιμών του πραγματικού ρομπότ σε 40 τιμές, έτσι ώστε να μπορέσουν να εισαχθούν στην είσοδο του μοντέλου. Για αυτό τον σκοπό δημιουργήθηκε ένα python (.py) αρχείο το οποίο παίρνει τις τιμές του LiDAR από το topic /scan, τις επαναδειγματοληπτεί σε 40 ισοκατανεμημένες τιμές και έπειτα τις κάνει publish σε ένα νέο topic /scan_resampled. Είναι σημαντικό να γίνει ίση κατανομή μεταξύ των τιμών, έτσι ώστε οι 40 τελικές τιμές να προσδίδουν απεικόνιση 360 μοιρών. Ο κώδικας του αρχείου laser_scan_resampler.py δίνεται στο Παράρτημα – Κώδικας (Κώδικας 1), το οποίο βρίσκεται στο τέλος του εγγράφου.

Το Πραγματικό Περιβάλλον

Η υλοποίηση σε πραγματικό περιβάλλον πραγματοποιήθηκε στον χώρο του εργαστηρίου του τμήματος Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του Δημοκριτείου Πανεπιστημίου Θράκης. Εντός του εργαστηρίου δημιουργήθηκε ένας χώρος με στατικά και δυναμικά εμπόδια χρησιμοποιώντας συμπαγή αντικείμενα όπως χαρτόνια, κουτιά γραφεία και ντουλάπες. Ο λόγος που χρησιμοποιήθηκαν τα συγκεκριμένα αντικείμενα είναι η δημιουργία ενός περιβάλλοντος το οποίο να μοιάζει με το προσομοιωμένο περιβάλλον από την οπτική του φυσικού ρομπότ, δηλαδή ενός περιβάλλοντος όπου τα δεδομένα του αισθητήρα του ρομπότ θα μοιάζουν ποιοτικά με αυτά της προσομοίωσης. Αυτό είναι σημαντικό καθώς εάν τα δεδομένα τα οποία εισάγονται στο δίκτυο δεν είναι παρόμοια με αυτά πάνω στα οποία εκπαιδεύτηκε, το αποτέλεσμα θα είναι μια μη επιθυμητή έξοδος. Το περιβάλλον αυτό παρουσιάζεται στην Εικόνα 35.



(α): Το Πραγματικό Περιβάλλον



(β): Το Πραγματικό Περιβάλλον

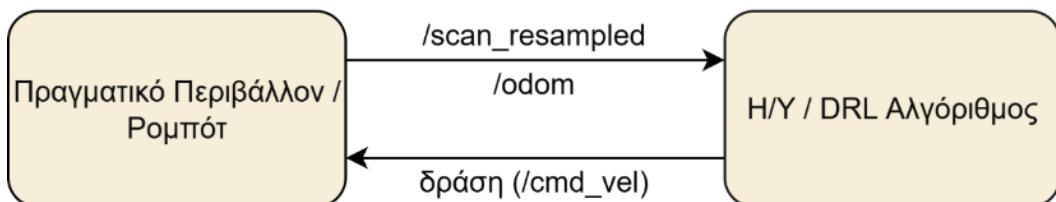
Εικόνα 35: Το Πραγματικό Περιβάλλον

Ο πράκτορας DRL

Έχοντας πλέον εκπαιδεύσει και έπειτα αξιολογήσει μερικές από τις καλύτερες υλοποιήσεις του αλγορίθμου, οι πράκτορες θα αξιολογηθούν και σε πραγματικό περιβάλλον. Ο αλγόριθμος DRL έτρεξε σε έναν ξεχωριστό σταθερό υπολογιστή. Ο υπολογιστής αυτός εγκαταστάθηκε στον ίδιο χώρο όπου πραγματοποιήθηκε και η πραγματική εκτέλεση του πειράματος, έτσι ώστε να υπάρχει άμεση επικοινωνία με το ρομπότ. Η διαδικασία αυτή επιτρέπει πλήρη διαχωρισμό του λογισμικού του πράκτορα από την υπολογιστική μονάδα του ρομπότ, κάνοντας την υλοποίηση πιο ευέλικτη και αποδοτική. Για λόγους ευχρηστίας θα ονομάσουμε αυτόν τον υπολογιστή H/Y.

Γίνεται λοιπόν αντιληπτό ότι και στην πραγματική εφαρμογή υπάρχουν τρεις ξεχωριστές οντότητες: το ρομπότ, το φυσικό περιβάλλον, και ο πράκτορας που τρέχει στον H/Y (Εικόνα 36). Ως δίαυλος επικοινωνίας μεταξύ αυτών χρησιμοποιήθηκε το ROS2. Οι οντότητες αυτές αλληλοεπιδρούν μεταξύ τους την κάθε δεδομένη χρονική στιγμή σχηματίζοντας έναν βρόγχο αντίληψης–απόφασης–δράσης σε πραγματικό χρόνο. Το ρομπότ λαμβάνει πληροφορία από το περιβάλλον μέσω των αισθητήρων του (LiDAR, IMU) και τη δημοσιεύει σε ROS 2 topics μέσω του μηχανισμού publish/subscribe. Τα μηνύματα αυτά μεταφέρονται δικτυακά στον H/Y, ο οποίος τα χρησιμοποιεί ως εισόδους του αλγορίθμου DRL. Ο αλγόριθμος υπολογίζει τις εντολές ελέγχου (γραμμική και γωνιακή ταχύτητα) και τις δημοσιεύει σε αντίστοιχα ROS 2 topics. Στη συνέχεια, ο ελεγκτής του ρομπότ εγγράφεται στα topics αυτά, λαμβάνει τις εντολές και τις εφαρμόζει στους ενεργοποιητές, υλοποιώντας την κίνηση στο επόμενο χρονικό βήμα. Έτσι κλείνει ο βρόγχος αντίληψης–απόφασης–δράσης σε πραγματικό χρόνο.

Πραγματικό Σύστημα Πλοήγησης



Εικόνα 36: Το Σύστημα Πλοήγησης

6.2 Εφαρμογή αλγορίθμων

Αρχικά θα πρέπει να επισημανθεί ότι στην πραγματική εφαρμογή δεν επιτρέπεται στο ρομπότ να κινείται με την όπισθεν. Ενώ υπάρχει η δυνατότητα οπίσθιας κίνησης, η προεπιλεγμένη ρύθμιση του TurtleBot4 δεν επιτρέπει στο ρομπότ να κινείται προς τα

πίσω για λόγους ασφαλείας. Συνεπώς, επιλέξαμε να διατηρήσουμε την συγκεκριμένη ρύθμιση ούτως ώστε να εξασφαλίσουμε την ακεραιότητά του. Αυτό περιορίζει τις επιλογές των μοντέλων που μπορούμε να εφαρμόσουμε, με μόνα επιτρεπτά αυτά τα οποία η ρύθμιση backward enabled είναι απενεργοποιημένη.

Τα μοντέλα τα οποία επιλέχθηκαν για να εφαρμοστούν στο πραγματικό πείραμα είναι τα δύο μοντέλα των αλγορίθμων DDPG και TD3 τα οποία έχουν απενεργοποιημένη την ρύθμιση για οπίσθια κίνηση και εκπαιδεύτηκαν στο Περιβάλλον Δυσκολίας 2.

Αρχικά εφαρμόστηκε το μοντέλο του αλγορίθμου DDPG. Το ρομπότ ξεκίνησε να πλοηγείται στον χώρο αποφεύγοντας σε αρκετές περιπτώσεις με επιτυχία τα στατικά εμπόδια, όπως φαίνεται στα στιγμιότυπα της Εικόνα 37. Στην συνέχεια πλησίασε σε δυναμικά εμπόδια (άνθρωπος, κινούμενα κουτιά). Από τον τρόπο που κινείται φαίνεται να αντιλαμβάνεται την ύπαρξη των εμποδίων, τόσο στις επαναλήψεις που το ρομπότ απέφυγε κάποιο δυναμικό εμπόδιο όσο και στις επαναλήψεις που το ρομπότ συγκρούστηκε με αυτό. Στην περίπτωση σύγκρουσης παρ' όλο που το εμπόδιο έγινε αντιληπτό, ο πράκτορας δεν πρόλαβε να αντιδράσει εγκαίρως έτσι ώστε να καταφέρει να το αποφύγει. Το ρομπότ φαίνεται επίσης να προσπαθεί να κατευθυνθεί προς τον στόχο, χωρίς όμως να οδηγείται σε κάποια επιτυχημένη προσπάθεια.

Στην συνέχεια εφαρμόστηκε το μοντέλο του αλγορίθμου TD3. Η συμπεριφορά του είναι παρόμοια με αυτή του DDPG, δηλαδή ο πράκτορας πλοηγείται στον χώρο, αποφεύγει στατικά εμπόδια αλλά δυσκολεύεται να αποφύγει δυναμικά εμπόδια παρ' όλο που φαίνεται να αντιλαμβάνεται την ύπαρξή τους. Αν και η απόδοση του αλγορίθμου TD3 στο περιβάλλον προσομοίωσης είναι χαμηλότερη από αυτή του DDPG, τα αποτελέσματα στο πραγματικό περιβάλλον φαίνεται να είναι παρόμοια. Αυτό οφείλεται κυρίως στο ότι και τα δύο ποσοστά επιτυχίας είναι σχετικά χαμηλά, με αποτέλεσμα να μην φαίνεται η διαφορά αυτή για ένα μικρό αριθμό εκτελέσεων που πραγματοποιήθηκαν στην πραγματική υλοποίηση.

Σημαντικό ρόλο στην αναποτελεσματικότητα του πράκτορα παίζει το ότι δεν επιτρέπεται η κίνηση προς τα πίσω. Τα μοντέλα με απενεργοποιημένη την ρύθμιση αυτή εμφανίζουν όπως είδαμε στο Κεφάλαιο 5 χαμηλά ποσοστά επιτυχίας. Είναι φυσικό επακόλουθο λοιπόν, το ποσοστό επιτυχίας να πέφτει ακόμα περισσότερο στην πραγματική υλοποίηση όπου υπάρχουν και επιπλέον παράγοντες δυσκολίας. Στις παρακάτω εικόνες παρουσιάζονται φωτογραφίες από τις εφαρμογές των αλγορίθμων που πραγματοποιήθηκαν. Οι φωτογραφίες αποτελούν στιγμιότυπα από τα βίντεο που δημιουργήθηκαν.



(α): Στιγμιότυπο 1



(β): Στιγμιότυπο 2



(γ): Στιγμιότυπο 3



(δ): Στιγμιότυπο 4

Εικόνα 37: Στιγμιότυπα από την εφαρμογή των Αλγορίθμων σε Πραγματικό Περιβάλλον

7 Συμπεράσματα και Μελλοντικές Προεκτάσεις

Στην παρούσα διπλωματική εργασία εξετάστηκε αναλυτικά η πλοϊγηση ρομποτικών πλατφορμών με την χρήση τεχνικών DRL. Το πρώτο μέρος της περιλαμβάνει το θεωρητικό υπόβαθρο που απαιτείται για να μπορέσει ο αναγνώστης να κατανοήσει τις βάσεις της πλοϊγησης ρομποτικών πλατφορμών, καθώς και το πως συνδυάζεται με ένα τύπο Μηχανικής Μάθησης, την Ενισχυτική Μάθηση. Έπειτα, στο δεύτερο μέρος της διπλωματικής αναλύθηκε η πειραματική διαδικασία κατά την οποία εκπαιδεύτηκε, χρησιμοποιώντας αλγορίθμους DRL, μία επίγεια ρομποτική πλατφόρμα σε εικονικό περιβάλλον, με σκοπό να εκτελεί αυτόνομη πλοϊγηση σε άγνωστο χάρτη, αποφεύγοντας παράλληλα στατικά και δυναμικά εμπόδια.

7.1 Ανακεφαλαίωση των βασικών στοιχείων

Η εφαρμογή αλγορίθμων Ενισχυτικής Μάθησης στην πλοϊγηση ρομποτικών πλατφορμών απασχολεί έντονα την επιστημονική κοινότητα τα τελευταία χρόνια. Τα πειραματικά αποτελέσματα της παρούσας εργασίας επικυρώνουν το αυξημένο ενδιαφέρων που εμφανίζεται γύρω από την συγκεκριμένη τεχνική ρομποτικής πλοϊγησης.

Οι δύο αλγόριθμοι (DDPG και TD3) που εφαρμόστηκαν λειτούργησαν αποτελεσματικά, επιτρέποντας στο ρομπότ να μάθει τεχνικές πλοϊγησης μέσα από την εμπειρία που απέκτησε σταδιακά κατά την διάρκεια της εκπαίδευσης. Οι πολύωρες εκπαίδευσεις απέδωσαν καρπούς, σημειώνοντας ικανοποιητικά αποτελέσματα. Ο πράκτορας έμαθε να πλοηγείται με υψηλά ποσοστά επιτυχίας τόσο σε περιβάλλοντα με στατικά εμπόδια όσο και σε περιβάλλοντα με δυναμικά εμπόδια.

Πιο συγκεκριμένα σε περιβάλλον όπου δεν υπάρχουν δυναμικά εμπόδια ο αλγόριθμος DDPG κατάφερε απόδοση 100% ίδια με αυτή του αλγορίθμου TD3. Σε περιβάλλοντα με δυναμικά εμπόδια, ανάλογα με τον βαθμό δυσκολίας του περιβάλλοντος, τα βέλτιστα ποσοστά επιτυχίας των δύο αλγορίθμων κυμάνθηκαν μεταξύ 79% και 96%. Τα συγκεκριμένα αποτελέσματα είναι αρκετά ικανοποιητικά τόσο για τον αλγόριθμο DDPG όσο και για την βασική υλοποίηση του TD3 αλγορίθμου που χρησιμοποιήθηκε.

Εξαιρετικά σημαντική ήταν η συνεισφορά του hyperparameter tuning στην βελτίωση των αλγορίθμων. Όπως αναλύθηκε και στο Κεφάλαιο 5, η χρήση hyperparameter tuning συνείσφερε καταλυτικά στην αύξηση των ποσοστών επιτυχίας των εκπαίδευσεων, ειδικότερα στις εκπαίδευσεις που πραγματοποιήθηκαν στα περιβάλλοντα με δυναμικά εμπόδια. Είναι πολύ σημαντικό λοιπόν να βρεθούν οι κατάλληλες ρυθμίσεις για τον

εκάστοτε αλγόριθμο, έτσι ώστε να επιτευχθεί η βέλτιστη απόδοσή τους, πράγμα το οποίο ισχύει και γενικότερα στον κόσμο της Μηχανικής Μάθησης.

Όσον αφορά την υλοποίηση των αλγορίθμων σε πραγματικό περιβάλλον γίνεται εύκολα αντιληπτό ότι αυξάνεται η δυσκολία και η πολυπλοκότητα του συστήματος. Παρόλο που τα αποτελέσματα δεν ήταν στο επίπεδο της προσομοίωσης, οι αλγόριθμοι εφαρμόστηκαν με επιτυχία στο πραγματικό ρομπότ και αυτό πλοηγήθηκε μέσα στον χώρο αποφεύγοντας στατικά και δυναμικά εμπόδια. Ο σημαντικότερος παράγοντας που έπαιξε ρόλο στα χαμηλά ποσοστά επιτυχίας του πραγματικού πειράματος είναι ο περιορισμός χρήσης αλγορίθμων, καθώς για λόγους ασφαλείας οι μόνοι αλγόριθμοι που χρησιμοποιήθηκαν δεν ήταν αποτελεσματικοί. Ένας δεύτερος παράγοντας είναι η διαφορά μεταξύ εκπαίδευσης και πραγματικού ρομπότ. Παρ' όλο που μέσω ρυθμίσεων επιχειρήθηκε να εξαλειφθεί, η διαφορά τόσο στην δομή όσο και στις αποκλείσεις του πραγματικού μέσου επιβαρύνει τον βαθμό δυσκολίας.

7.2 Προτάσεις για μελλοντική έρευνα

Ο κόσμος της Ενισχυτικής Μάθησης βρίσκεται ακόμα σε πολύ πρώιμα στάδια και οι δυνατότητές της είναι άνευ ορίων. Στην παρούσα διπλωματική εργασία εξετάστηκε μόνο ένα πολύ μικρό κομμάτι των δυνατοτήτων της χρήσης DRL για πλοήγηση ρομποτικών πλατφορμών, και φυσικά υπάρχουν πολλές προτάσεις για μελλοντική ανάπτυξη και βελτιστοποίηση πρακτικών. Παρακάτω αναφέρονται μερικές από αυτές:

- **Περεταίρω βελτίωση των υπαρχόντων μοντέλων**

Ένας τρόπος για περεταίρω βελτίωση των μοντέλων είναι η δοκιμή μεγαλύτερου εύρους τιμών κατά την ρύθμιση των παραμέτρων. Η ανάπτυξη μίας βελτιωμένης συνάρτησης ανταμοιβών μπορεί επίσης να οδηγήσει σε μεγαλύτερα ποσοστά επιτυχίας.

- **Εκπαίδευση σε πιο ρεαλιστικά εικονικά περιβάλλοντα**

Ένας τρόπος γενίκευσης για διατήρηση της αποδοτικότητας του μοντέλου όταν πραγματοποιείται μεταφορά από την προσομοίωση στην πραγματικότητα, είναι η εκπαίδευση των πρακτόρων σε περιβάλλοντα που πλησιάζουν όσο το δυνατόν περισσότερο γίνεται στον πραγματικό κόσμο. Αναφορικά, δύο εξαιρετικά περιβάλλοντα προσομοίωσης που θα μπορούσαν να συμβάλλουν στην βελτίωση των αλγορίθμων είναι τα εξής:

1. **NVIDIA Isaac Sim (Omniverse)** – Προσομοιωτής σχεδιασμένος ιδανικά για ρομποτικές προσομοιώσεις με χρήση Τεχνητής Νοημοσύνης. Το Isaac Sim παρέχει ρεαλιστικό σύστημα φυσικής και αισθητήρων (LiDAR, κάμερες), φωτορεαλισμό, καθώς και εσωτερικά σχεδιασμένο ROS2 bridge το οποίο επιτρέπει την επικοινωνία μέσω ROS. Υποστηρίζει domain randomization και παραγωγή συνθετικών δεδομένων, σημαντικές προσθήκες για την μείωση του sim-to-real gap και την εκπαίδευση πιο ανθεκτικών πολιτικών απέναντι σε θόρυβο, μεταβολές φωτισμού και διαφοροποιήσεις τριβών.

2. Unity – Εξαιρετικά ρεαλιστικό περιβάλλον προσομοίωσης. Υποστηρίζει ROS2 σύνδεση μέσω του πακέτου ROS-TCP-Connector και εκπαίδευση αλγορίθμων Ενισχυτικής Μάθησης μέσω του ML-Agents Toolkit.

Η έρευνα για την βελτίωση της απόδοσης δεν πρέπει να περιορίζεται στο αλγορίθμικό κομμάτι. Δεν πρέπει να αγνοείται η σημαντικότητα της επιλογής κατάλληλου περιβάλλοντος προσομοίωσης, έτσι ώστε κατά την μετάβαση sim-to-real να μην υπάρχει απόκλιση αποδόσεων.

- **Βελτιώσεις στην υλοποίηση σε πραγματικό περιβάλλον**

Πέρα από την μέριμνα για βελτιστοποίηση της εκπαίδευσης στο προσομοιωμένο περιβάλλον, για να πραγματοποιηθεί με επιτυχία και ακεραιότητα η μετάβαση στον πραγματικό κόσμο, θα πρέπει και το πραγματικό σύστημα να είναι σωστά δομημένο. Βελτιώσεις στο πραγματικό σύστημα οι οποίες ενδέχεται να οδηγούσαν σε σαφέστατα καλύτερα αποτελέσματα είναι οι εξής:

1. Ενεργοποίηση της ρύθμισης που επιτρέπει στο ρομπότ την κίνηση προς τα πίσω. Η συγκεκριμένη ρύθμιση θα μας επέτρεπε να χρησιμοποιήσουμε μοντέλα με πολύ υψηλότερα ποσοστά επιτυχίας κατά την εκπαίδευση.
2. Χρήση του ίδιου ρομπότ που χρησιμοποιήθηκε και κατά την εκπαίδευση. Έστω και μικρές διαφορές στην δομή προκαλούν αποκλίσεις που μειώνουν την αποδοτικότητα κατά την υλοποίηση σε πραγματικό περιβάλλον.

- **Προϋποθέσεις για υλοποιήσεις μεγαλύτερης κλίμακας**

Η εφαρμογή Ενισχυτικής Μάθησης στην πλοήγηση ρομποτικών συστημάτων δεν περιορίζεται μόνο στην εκπαίδευση ενός πράκτορα έτσι ώστε να πλοηγηθεί με επιτυχία σε έναν χώρο δωματίου με λίγα δυναμικά εμπόδια. Γενικότερος σκοπός είναι η εκπαίδευση πρακτόρων χωρίς περιορισμούς, που είναι ικανοί να πλοηγηθούν σε οποιοδήποτε περιβάλλον. Μπορεί να φαντάζει εξωπραγματικό και ακατόρθωτο, ωστόσο η θεωρητική ικανότητα της Τεχνητής Νοημοσύνης επιβεβαιώνει την δυνατότητα να επιτευχθεί στο μέλλον. Απαραίτητες προϋποθέσεις είναι η χρήση ισχυρότερων δικτύων, ισχυρότερων υπολογιστικών μονάδων και εκπαίδευση σε ρεαλιστικά προσομοιωμένα περιβάλλοντα μεγαλύτερης κλίμακας.

ΕΥΡΕΤΗΡΙΟ

Κατάλογος Εικόνων

Εικόνα 1: Shakey the Robot. Πηγή: https://www.computerhistory.org/revolution/artificial-intelligence-robotics/13/289/1241	6
Εικόνα 2: Αρχιτεκτονική Subsumption. Πηγή: https://www.researchgate.net/figure/The-subsumption-architecture-adapted-from-Brooks-1999_fig5_225257926	8
Εικόνα 3: Διάγραμμα Αλγορίθμων Πλοϊόγησης	9
Εικόνα 4: Τεχνητό Νευρωνικό Δίκτυο. Πηγή: https://math.uni.lu/eml/assets/reports/2020/DiasMoreira.pdf	17
Εικόνα 5: Συνάρτηση Sigmoid	19
Εικόνα 6: Συνάρτηση Tanh	19
Εικόνα 7: Συνάρτηση ReLU.....	20
Εικόνα 8: Συνάρτηση Softmax.....	20
Εικόνα 9: Γραφική αναπαράσταση του Gradient Descent. Πηγή: https://pmc.ncbi.nlm.nih.gov/articles/PMC10426722/figure/j_biol-2022-0665_fig_002/	21
Εικόνα 10: Το πλαίσιο της Ενισχυτικής Μάθησης. Πηγή: R. S. . Sutton and A. G. . Barto, Reinforcement learning : an introduction. The MIT Press, 2020	25
Εικόνα 11: Ο αλγόριθμος Q-Learning	33
Εικόνα 12: Ψευδοκώδικας αλγορίθμου Q-Learning	34
Εικόνα 13: Ψευδοκώδικας αλγορίθμου DQN	38
Εικόνα 14: Ψευδοκώδικας αλγορίθμου DDPG	40
Εικόνα 15: Ψευδοκώδικας αλγορίθμου TD3	42
Εικόνα 16: Ψευδοκώδικας αλγορίθμου PPO	44
Εικόνα 17: Εξαρτήματα του Turtlebot3 Burger. https://emanual.robotis.com/assets/images/platform/turtlebot3/hardware_setup/turtlebot3_burger_components.png	47
Εικόνα 18: Περιβάλλον Δυσκολίας 1	49
Εικόνα 19: Περιβάλλον Δυσκολίας 2	49
Εικόνα 20: Περιβάλλον Δυσκολίας 3	50
Εικόνα 21: Το Σύστημα Πλοϊόγησης	52
Εικόνα 22: Γράφημα Καταστάσεων Τερματισμού	56
Εικόνα 23: Στιγμιότυπο εκπαίδευσης	56
Εικόνα 24: Αρχικές Τιμές Παραμέτρων	57
Εικόνα 25: Διαγράμματα Εκπαιδεύσεων DDPG – Περιβάλλον Δυσκολίας 1	63
Εικόνα 26: Διαγράμματα Εκπαιδεύσεων DDPG – Περιβάλλον Δυσκολίας 2	64
Εικόνα 27: Διαγράμματα Εκπαιδεύσεων DDPG – Περιβάλλον Δυσκολίας 3	67
Εικόνα 28: Διαγράμματα Εκπαιδεύσεων TD3 – Περιβάλλον Δυσκολίας 1	69
Εικόνα 29: Διαγράμματα Εκπαιδεύσεων TD3 – Περιβάλλον Δυσκολίας 2	70
Εικόνα 30: Διαγράμματα Εκπαιδεύσεων TD3 – Περιβάλλον Δυσκολίας 3	73
Εικόνα 31: Ποσοστά Επιτυχίας Αλγορίθμων DDPG και TD3	74
Εικόνα 32: Ποσοστά Επιτυχίας Αλγορίθμου DDPG πριν και μετά την Ρύθμιση Παραμέτρων.....	75
Εικόνα 33: Ποσοστά Επιτυχίας Αλγορίθμου TD3 πριν και μετά την Ρύθμιση Παραμέτρων.....	75

Εικόνα 34: TurtleBot 4. Πηγή: https://clearpathrobotics.com/turtlebot-4/	78
Εικόνα 35: Το Πραγματικό Περιβάλλον	81
Εικόνα 36: Το Σύστημα Πλοήγησης	82
Εικόνα 37: Στιγμιότυπα από την εφαρμογή των Αλγορίθμων σε Πραγματικό Περιβάλλον	87

Κατάλογος Πινάκων

Πίνακας 1: Αισθητήρες Αντίληψης	11
Πίνακας 2: Αισθητήρες Αντίληψης	12
Πίνακας 3: Προδιαγραφές Hardware του Turtlebot3 Burger	47
Πίνακας 4: Αποτελέσματα DDPG - Περιβάλλον Δυσκολίας 1	61
Πίνακας 5: Αποτελέσματα DDPG - Περιβάλλον Δυσκολίας 2	63
Πίνακας 6: Αποτελέσματα DDPG - Περιβάλλον Δυσκολίας 3	65
Πίνακας 7: Αποτελέσματα TD3 – Περιβάλλον Δυσκολίας 1	68
Πίνακας 8: Αποτελέσματα TD3 – Περιβάλλον Δυσκολίας 2	69
Πίνακας 9: Αποτελέσματα TD3 – Περιβάλλον Δυσκολίας 3	71
Πίνακας 10: Προδιαγραφές Hardware του Turtlebot4	78

ΒΙΒΛΙΟΓΡΑΦΙΑ

- [1] N. J. Nilsson and D. L. Nielson, “SRI International ® SHAKEY THE ROBOT,” 1984.
- [2] R. E. Fikes and N. J. Nilsson, “Strips: A new approach to the application of theorem proving to problem solving,” *Artif Intell*, vol. 2, no. 3–4, pp. 189–208, Dec. 1971, doi: 10.1016/0004-3702(71)90010-5.
- [3] Roland Siegwart and Illah R. Nourbakhsh, *Introduction to Autonomous Mobile Robots*. 2004.
- [4] S. Thrun, “Robotic Mapping: A Survey,” 2002.
- [5] S. M. LaValle, “PLANNING ALGORITHMS”, Accessed: Sep. 25, 2025. [Online]. Available: <http://planning.cs.uiuc.edu/>
- [6] P. E. Hart, N. J. Nilsson, and B. Raphael, “A Formal Basis for the Heuristic Determination of Minimum Cost Paths,” *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968, doi: 10.1109/TSSC.1968.300136.
- [7] S. LaValle and J. Kuffner, “Rapidly-Exploring Random Trees: Progress and Prospects,” *Algorithmic and computational robotics: New directions*, Jan. 2000.
- [8] R. A. Brooks, “A Robust Layered Control System For A Mobile Robot,” *IEEE Journal on Robotics and Automation*, vol. 2, no. 1, pp. 14–23, 1986, doi: 10.1109/JRA.1986.1087032.
- [9] W. Yun, “Behavior-Based Robotics: Ronald C. Arkin; The MIT Press, Cambridge, MA, 1998,” *Automatica*, vol. 38, no. 12, pp. 2193–2196, Dec. 2002, doi: 10.1016/S0005-1098(02)00169-3.
- [10] H. Durrant-Whyte and T. Bailey, “Simultaneous localization and mapping: Part I,” *IEEE Robot Autom Mag*, vol. 13, no. 2, pp. 99–108, Jun. 2006, doi: 10.1109/MRA.2006.1638022.
- [11] A. L. Samuel, “Some studies in machine learning using the game of checkers (Reprinted from Journal of Research and Development, vol 3, 1959),” *IBM J Res Dev*, vol. 44, pp. 207–226, Jan. 2000.
- [12] T. M. Mitchell, *Machine learning*, vol. 1, no. 9. McGraw-hill New York, 1997.
- [13] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.
- [14] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning representations by back-propagating errors,” *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.
- [15] R. S. . Sutton and A. G. . Barto, *Reinforcement learning : an introduction*. The MIT Press, 2020.
- [16] V. Konda and J. Tsitsiklis, “Actor-Critic Algorithms,” in *Advances in Neural Information Processing Systems*, S. Solla, T. Leen, and K. Müller, Eds., MIT Press, 1999.

- [17] P. Swazinna, S. Udluft, and T. Runkler, "Overcoming Model Bias for Robust Offline Deep Reinforcement Learning," *Eng Appl Artif Intell*, vol. 104, Aug. 2020, doi: 10.1016/j.engappai.2021.104366.
- [18] T. Jaakkola, S. Singh, and M. Jordan, "Reinforcement Learning Algorithm for Partially Observable Markov Decision Problems," *Adv Neural Inf Process Syst*, vol. 7, May 1999.
- [19] G. Tesauro, "Practical issues in temporal difference learning," *Machine Learning* 1992 8:3, vol. 8, no. 3, pp. 257–277, May 1992, doi: 10.1007/BF00992697.
- [20] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Machine Learning* 1992 8:3, vol. 8, no. 3, pp. 279–292, May 1992, doi: 10.1007/BF00992698.
- [21] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–533, May 2015, doi: 10.1038/nature14236.
- [22] T. P. Lillicrap *et al.*, "Continuous control with deep reinforcement learning," *4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings*, Sep. 2015.
- [23] D. Silver, G. Lever, N. M. O. Heess, T. Degris, D. Wierstra, and M. A. Riedmiller, "Deterministic Policy Gradient Algorithms," in *International Conference on Machine Learning*, 2014.
- [24] G. E. Uhlenbeck and L. S. Ornstein, "On the Theory of the Brownian Motion," *Physical Review*, vol. 36, no. 5, pp. 823–841, Sep. 1930, doi: 10.1103/PhysRev.36.823.
- [25] S. Fujimoto, H. Van Hoof, and D. Meger, "Addressing Function Approximation Error in Actor-Critic Methods," 2018.
- [26] H. Van Hasselt, "Double Q-learning".
- [27] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. K. Openai, "Proximal Policy Optimization Algorithms".
- [28] J. Schulman, S. Levine, P. Moritz, M. Jordan, and P. Abbeel, "Trust Region Policy Optimization".
- [29] J. Schulman, P. Moritz, S. Levine, M. I. Jordan, and P. Abbeel, "HIGH-DIMENSIONAL CONTINUOUS CONTROL USING GENERALIZED ADVANTAGE ESTIMATION".
- [30] "GitHub - tomasvr/turtlebot3_drlnav: A ROS2-based framework for TurtleBot3 DRL autonomous navigation.".
- [31] A. Paszke *et al.*, "PyTorch: An Imperative Style, High-Performance Deep Learning Library," 2019.

ΠΑΡΑΡΤΗΜΑ - ΚΩΔΙΚΑΣ

Κώδικας 1: *laser_scan_resampler.py*

```
#!/usr/bin/env python3

import rclpy
from rclpy.node import Node
import numpy as np
from sensor_msgs.msg import LaserScan

class LaserScanResampler(Node):
    def __init__(self):
        super().__init__('laser_scan_resampler')
        self.subscription = self.create_subscription(
            LaserScan,
            '/scan',
            self.scan_callback,
            10
        )
        self.publisher_ = self.create_publisher(
            LaserScan,
            '/scan_resampled',
            10
        )
        self.target_samples = 40
        self.get_logger().info('LaserScanResampler initialized. Waiting for /scan data...')

    def scan_callback(self, msg):
        original_ranges = np.array(msg.ranges)
        original_len = len(original_ranges)
        if original_len < self.target_samples:
            self.get_logger().warn(f"Not enough samples to resample: got {original_len}")
            return

        # Compute equally spaced indices for downsampling
        indices = np.linspace(0, original_len - 1, self.target_samples).astype(int)
        resampled_ranges = original_ranges[indices]

        # Create a new LaserScan message
        resampled_msg = LaserScan()
        resampled_msg.header = msg.header
        resampled_msg.angle_min = msg.angle_min
        resampled_msg.angle_max = msg.angle_max
        resampled_msg.angle_increment = (msg.angle_max - msg.angle_min)/(self.target_samples-1)
        resampled_msg.time_increment = msg.time_increment * (original_len/self.target_samples)
        resampled_msg.scan_time = msg.scan_time
        resampled_msg.range_min = msg.range_min
        resampled_msg.range_max = msg.range_max
        resampled_msg.ranges = resampled_ranges.tolist()
        resampled_msg.intensities = []
        self.publisher_.publish(resampled_msg)
        self.get_logger().info("Published resampled scan with 40 samples.")

    def main():
        rclpy.init()
        node = LaserScanResampler()
        rclpy.spin(node)
        node.destroy_node()
        rclpy.shutdown()

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