Guidance, Navigation, and Control Techniques for Service Robotics

Politecnico di Torino

ScuDo
Scuola di Dottorato ~ Doctoral School
WHAT YOU ARE, TAKES YOU FAR

Doctoral Dissertation
Doctoral Program in Computer and Control Engineering (37^{th}cycle)

Guidance, Navigation, and Control Techniques for Service Robotics

By

Romisaa Ali

******

Supervisor(s):
Prof. Marcello Chiaberge, Supervisor
Dr. Maurizio Griva, Co-Supervisor, Reply Concept Company

Doctoral Examination Committee:
Prof. Matteo Menolotto, Referee, University College Cork – Tyndall National Institute, Ireland
Prof. Francisco Barranco Expósito, Referee, Universidad de Granada, Spain

Politecnico di Torino
2026

Declaration

I hereby declare that, the contents and organization of this dissertation constitute my own original work and does not compromise in any way the rights of third parties, including those relating to the security of personal data.

Romisaa Ali
2026

* This dissertation is presented in partial fulfillment of the requirements for **Ph.D. degree** in the Graduate School of Politecnico di Torino (ScuDo).

To my mother, and to the loving memory of my grandmother.

Acknowledgements

Acknowledgment

I would like to express my sincere gratitude to my academic supervisor, Prof. Marcello Chiaberge, for his continuous guidance and support. I also thank the PhD program coordinator, Prof. Fabrizio Lamberti, and the SCUDO PhD program at Politecnico di Torino, along with the Department of Control and Computer Engineering (DAUIN), for providing an excellent research environment.

I am grateful to Reply – Concept for funding my PhD and to Mr. Maurizio Griva for his valuable industrial supervision. I also acknowledge the PIC4SeR center for supporting the development of the robotic systems used in this work.

A special thanks to Prof. Lino Marques and Dr. Sedat Dogru from the University of Coimbra for their support during my research stay in Portugal.

Finally, I deeply thank my family and friends for their constant encouragement throughout this journey.

Abstract

This thesis addresses a key challenge in service robotics: enabling autonomous mobile robots to navigate unpredictable environments with unexpected changes. Effective generalization is essential for advancing robot autonomy, as robots must be capable of handling unseen situations. Traditional approaches often focus on training with numerous fixed scenarios, but this research demonstrates that improving generalization through varied start and goal points can significantly improve adaptability. The primary objective of this research is to develop a deep reinforcement learning approach that enhances the generalization and transferability of mobile robot navigation in unseen environments. This is achieved by extending the Twin Delayed Deep Deterministic Policy Gradient (TD3) algorithm. The proposed solution integrates randomization at start and end points, and variable robot orientations to foster robust exploration and adaptability. The research methodology integrates the Noetic ROS framework with a skid-steered robot simulation in Gazebo and a custom OpenAI Gym environment. Training is carried out using the TD3 algorithm with LiDAR sensor data for decision-making. The model parameters are continuously updated on the basis of interactions, and extensive randomization is introduced during training to maximize generalization capabilities. The solution is tested in various custom environments designed with diverse object configurations. Experimental results demonstrate that the extended model significantly outperforms non-extended models in navigating complex and unfamiliar environments. Key metrics such as success rate, collision rate, and distance traveled reveal enhanced performance, with the extended model achieving superior efficiency and adaptability compared to models trained in more constrained scenarios. The findings contribute to the field of autonomous service robotics by providing a scalable approach to achieve robust generalization. This research highlights that training with diverse start and goal points, rather than simply increasing the number of static scenarios, is more effective for improving adaptability and flexibility.

List of Publications

Publications Related to This Thesis

- **Romisaa Ali**, Marcello Chiaberge, Sedat Dogru, Lino Marques. *Performance Evaluation of Move Base Integration with Deep Reinforcement Learning for Autonomous Robot Navigation.* European Conference on Mobile Robots (ECMR 2025), Padova, Italy, September 2025. IEEE Proceedings.

- **Romisaa Ali**, Marcello Chiaberge, Sedat Dogru, Lino Marques. *Adaptive Robot Navigation Using Randomized Goal Selection with Twin Delayed Deep Deterministic Policy Gradient.* Preprints.org, 2024. DOI: https://doi.org/10.20944/preprints202412.1357.v2

- **Romisaa Ali**. *Robot Exploration and Navigation in Unseen Environments Using Deep Reinforcement Learning.* International Conference on Robot Motion Control (ICRMC), Auckland, New Zealand, December 2023.

- **Romisaa Ali**. *Optimized GNC Techniques for Service Robotics.* AIxIA 2023 – 22nd International Conference of the Italian Association for Artificial Intelligence, Rome, Italy. CEUR-WS Proceedings.

- **Romisaa Ali**. *Robot Movement Using the Trust Region Policy Optimization.* International Conference on Robot Motion Control and Automation (ICRMCA), Rome, Italy, January 2023.

Other Publications

- **Romisaa Ali**. *Components Selection of ISRASAT Series Ground Station.* MAT Journals, 2020.
- **Romisaa Ali**. *Components Selection and Configuration of ISRAHAB1 Sensors.* MAT Journals, 2020.
- **Romisaa Ali**. *Modelling of Planetary Radar for Tracking Near-Earth Asteroid.* 5th International Conference on Aerospace Science & Engineering (ICASE), Islamabad, Pakistan, 2017. IEEE Xplore.
- **Romisaa Ali**. *Modeling of Planetary Radar for Ranging Near-Earth Asteroids.* Master Thesis, Sudan University of Science and Technology, 2018.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Autonomous robots, particularly service and skid-steered robots [1], are increasingly utilized in diverse real-world applications due to their durability and versatility. These systems are essential for safety and efficiency in transportation and service applications, particularly in dynamic and unpredictable environments. Reinforcement Learning (RL) [2], part of autonomous systems, has emerged as a powerful tool to develop adaptive control policies for navigating complex environments. The developments of Deep Reinforcement Learning (DRL) [3] have seen significant advancements with recent methods like opac and DreamerV2 (2023) [4] [5], showcasing considerable progress in the ability of agents to learn complex behaviors. However, the full integration of DRL into professional machine-learning tools is still evolving. Despite this, several DRL methods have become widely adopted for autonomous vehicle navigation, such as Twin Delayed Deep Deterministic Policy Gradient (TD3) [6], Soft Actor-Critic (SAC) [7], and Proximal Policy Optimization (PPO) [8]. These methods have shown significant improvements in handling the challenges of real-time decision-making and are more compatible with modern data science, machine learning, and analysis tools compared to older techniques like Deep Q-Network (DQN) [9] and Deep Deterministic Policy Gradient (DDPG) [10]. A crucial aspect of these systems is their ability to effectively navigate and interact with complex and dynamic environments. Skid-steered robots, for example, are frequently used in various robotic applications due to their versatility, robustness,

and maneuverability. Developing efficient navigation policies for these autonomous robots is a challenging task due to the variability of real-world conditions. The research presented in this thesis focuses on evaluating the TD3 algorithm in the context of autonomous robot navigation. While various DRL algorithms have been developed—such as SAC, introduced by Haarnoja et al. [7], this work emphasizes a comparative analysis between TD3 and more recent approaches to assess performance in complex environments. this work specifically investigates TD3 due to its promising performance in continuous control tasks. As demonstrated in Ali [11], both SAC and TD3 performed well in training autonomous navigation policies; however, TD3 proved more robust in terms of stability, time-efficiency, and generalization to unseen environments, aligning with prior experience in evaluating RL-based navigation models.

However, a direct comparison of their effectiveness in the specific domain of skid-steered robot navigation remains underexplored. Both TD3 and SAC are characterized by their robust exploration capabilities and adaptive learning, making them ideal candidates for environments that include both static and dynamic obstacles. the TD3 algorithm improves upon DDPG by addressing overestimation bias using dual critic networks and delayed updates. TD3 has been shown to be effective in enhancing the stability and robustness of learned policies in continuous control tasks. SAC introduces entropy regularization to encourage exploration and avoid premature convergence, making it particularly effective for navigating in unseen environments. SAC has been integrated widely in RL frameworks for autonomous robots due to its ability to achieve a balance between exploration and exploitation. In this context, DRL algorithms act as decision-making agents, determining appropriate control signals for navigation, which is critical for ensuring the robot can autonomously handle diverse and unpredictable situations. The performance of these trained policies is assessed based on several key metrics: success rate, collision rate, and average time to complete a path. These metrics reflect the effectiveness of each RL algorithm in navigating complex environments with minimal intervention.

1.2 Related Works

Recent DRL approaches for robot navigation differ mainly in training strategy, environmental assumptions, and generalization mechanisms. This section reviews

related work by grouping existing methods according to four key aspects: curriculum learning, map-based versus mapless navigation, imitation learning integration, and strategies for improving generalization through environment or goal variability. These distinctions help clarify the position of the proposed method.

Several studies improve robot navigation using curriculum learning, where training starts with simple scenarios and gradually increases in difficulty. Chen et al. [12] employed a Dueling Double DQN combined with curriculum learning, allowing the robot to first learn in simple simulated environments before moving to more complex real-world obstacle courses. Yu et al. [13] proposed PathRL, which uses PPO and SAC to generate navigation paths directly, supported by curriculum learning to accelerate training and improve transfer to real-world robots. Similarly, Taheri et al. [14] introduced ResBlock PPO, where curriculum-style training, together with changes in network structure and reward design, improved performance in increasingly complex environments. In contrast, the proposed work does not rely on staged training or predefined difficulty levels. Instead of gradually increasing task complexity, generalization is encouraged by continuously exposing the robot to diverse navigation conditions through randomized training configurations. This strategy focuses on breaking behavioral bias and reducing dependence on specific training patterns rather than guiding learning through curriculum stages.

A major line of research in DRL-based navigation distinguishes between map-based and mapless approaches. Surmann et al. [15], initially proposed by Babaeizadeh et al. [16], demonstrated effective mapless navigation using GA3C, relying on raw sensor inputs such as LiDAR and RGB-D data to achieve successful real-world deployment. Similarly, Xue et al. [17] focused on mapless navigation in crowded environments, combining multiple sensors and socially aware reward design to handle complex human-populated spaces. Oliveira et al. [18] further explored mapless navigation by comparing several DRL algorithms and proposing a hybrid imitation–reinforcement learning method, showing strong generalization in irregular environments without explicit global maps. In contrast, map-based approaches explicitly leverage spatial representations to guide navigation. Chen et al. [12] employed local occupancy maps together with curriculum learning to improve robustness and noise tolerance, while Yu et al. [13] generated full navigation paths using learned planners that implicitly assume structured spatial information. The proposed work follows a strictly mapless formulation and does not rely on explicit global or local maps. Instead, navigation decisions are learned directly from sensor

observations, with robustness encouraged through the same randomized training strategy rather than map construction or map-dependent planning.

Some navigation approaches combine RL with imitation or supervised learning to improve training stability and safety. Oliveira et al. [18] proposed a hybrid method that integrates behavior cloning with TD3, allowing the agent to learn from expert demonstrations before and during RL. This combination was shown to reduce collisions and improve performance in complex environments, but it relies on the availability of expert data and predefined demonstrations. Related ideas appear in the work of Akmandor et al. [19, 20] and Roth et al. [21, 22], where hierarchical structures and guided learning components are used together with PPO to improve safety, interpretability, and real-time adaptability. These methods often benefit from additional supervision, structured guidance, or explainable decision layers. In contrast, the proposed work follows a pure RL approach. No imitation learning, expert demonstrations, or supervised guidance are used during training. Instead, the policy is learned entirely through interaction with the environment, with robustness and generalization encouraged by environmental and task variability during training rather than external expert knowledge.

Generalization in robot navigation is often addressed by training policies across multiple static or semi-static environments with different layouts and obstacle configurations. Cimurs et al. [23, 24] trained TD3 agents in dynamic and randomized environments with varying obstacles and goals, showing effective transfer from simulation to real robots by combining global navigation with local obstacle avoidance. However, their work relied on randomized goal selection without explicitly analyzing the impact of start and goal randomization on learning efficiency and generalization. Similarly, Xu et al. [25] evaluated several DRL algorithms across static and dynamic environments, demonstrating that exposure to diverse environment layouts can improve robustness, while also revealing limitations in complex scenarios. Anas et al. [26] focused on crowded but predefined indoor environments, achieving high success rates in simulation but facing challenges when real-world dynamics differed from training conditions, and also trained a TD3-based model in an indoor scenario with multiple obstacles, achieving a high success rate in simulation but facing deployment challenges due to manually controlled obstacle placement, As Akmandor et al. [19, 20] incorporated randomized start and goal selection in PPO-based systems, they focused on occupancy-based representations rather than explicitly analyzing the impact of randomization. In contrast, the proposed work does not rely on a large set

of different maps or predefined environments. Instead, generalization is promoted by introducing systematic task variability within a limited set of environments during training. This approach increases task diversity without increasing the number of environments, forcing the policy to solve a wide range of navigation problems under changing goals rather than memorizing fixed paths. As a result, generalization is driven by goal variability and continuous exposure to new navigation situations, rather than by switching between many static worlds.

1.3 Motivation

Recent advancements in DRL have transformed robot navigation, offering powerful solutions for autonomous exploration in complex, real-world environments. However, a key challenge still remains: enabling robots to generalize effectively in unpredictable environments. This is especially important for practical applications, where a robot must be able to adapt to sudden changes in its starting position, goal location, or surrounding obstacles.

1.3.1 Research Gap

The core challenge in DRL-based robot navigation lies in enabling robots to generalize effectively across diverse and dynamic scenarios. Despite numerous research efforts, this challenge persists and poses a significant barrier to practical implementation. Most existing studies, including Xu et al.'s work, have achieved considerable success in static and controlled simulation environments. However, the ability to generalize effectively to environments with varying start and goal points remains insufficient. In Xu et al.'s study on Robot Navigation Using DRL Under the Robot Operating System (ROS), distributed rollouts and lightweight containerization (e.g., Singularity) were used to optimize resource utilization by running multiple models simultaneously. However, this training strategy relied on a fixed goal point, which limited the model's ability to adapt to changing start–goal configurations in real-world deployments. The low deployment success rate indicated that the model's generalization was biased towards static configurations, even when trained across numerous world scenarios. This limitation is not exclusive to Xu et al. — other recent studies also struggled with the challenge of generalization. For instance, Cimurs et

al. achieved strong performance in simulation and structured real-world tests, but their model showed reduced adaptability when confronted with manually controlled and unforeseen obstacle changes during deployment. Similarly, Akmandor et al. implemented a combination of global and local navigation strategies, which showed promising results in simulation environments for obstacle avoidance. However, their training did not include variability in start and goal configurations, which constrained the robot's robustness when operating in previously unseen environments. These gaps highlight a broader challenge in DRL-based robot navigation: the difficulty in achieving transferability from controlled training environments to unexpected, previously unseen scenarios. Addressing this challenge requires new approaches that emphasize adaptability to changing start and goal points as well as dynamic obstacles, allowing robots to navigate complex, unpredictable settings with greater reliability.

1.3.2 Proposed Contribution

To address these research gaps, this thesis extends the work of Xu et al. by enhancing the exploration capabilities of the TD3 model. The proposed extension includes the integration of randomly generated start and goal points, as well as varying robot orientations during training. Additionally, the inclusion of dynamic objects that appear and disappear at random intervals will help the robot adapt to unexpected changes within its environment, thereby fostering greater robustness during real-world navigation tasks. These modifications aim to improve the model's learning capacity by enabling the robot to explore a wider range of scenarios, significantly increasing its adaptability to dynamic and unpredictable environments.

Furthermore, by leveraging Xu et al.'s distributed rollout framework, This study bridges the gap between simulation-based training and the model's transferability to unexpected, previously unseen environments. The proposed enhancements ensure that the robot is capable of learning in diverse and dynamic scenarios, ultimately making it more robust when facing unexpected changes during real-world navigation tasks. This contribution advances the state of the art in DRL-based robot navigation by developing a more adaptable and scalable solution for real-world deployment, improving both robustness and efficiency in autonomous mobile robots.

1.4 Problem Statement

The objective of this research is to solve the navigation problem $\mathscr{T}$, where a mobile robot must navigate from a starting position $\mathscr{D}S = (X_0, Y_0)$ to a goal position $\mathscr{D}_G = (X_n, Y_n)$ in an environment E that contains obstacles $O = o_1, o_2, \ldots, o_m$, as illustrated in Figure 1.1.

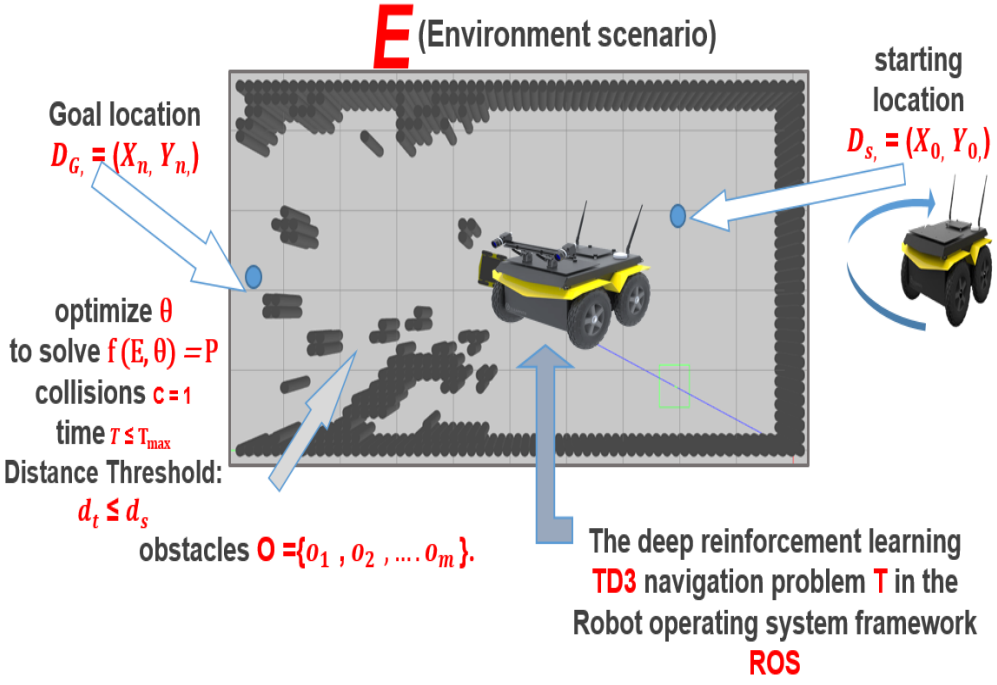


Fig. 1.1 Illustration of the navigation problem statement, showing the starting location $\mathscr{D}_S$, goal location $\mathscr{D}_G$ and obstacles O.

The robot must find a feasible path that reaches the goal while optimizing several key criteria. The optimization involves tuning the parameters of the navigation algorithm θ to achieve the following objectives:

- **Maximize the success rate of reaching the goal position.**
- **Avoid collisions**, enforcing a zero-collision constraint ($C = 0$).
- **Stay within a predefined time limit** $T \leq T_{\mathbf{max}}$, ensuring that the navigation is performed efficiently

- **Define goal achievement** based on the Euclidean distance between the current robot position and the goal position: the goal is considered reached if $d_t < 0.5\,\text{m}$, where $d_t = \|\mathscr{D}_t - \mathscr{D}_G\|$ denotes the distance between the robot's current position $\mathscr{D}_t$ and the goal position $\mathscr{D}_G$.

The goal is to maximize the success rate $P(\theta)$ subject to collision and time constraints, using a DRL algorithm. In particular, the TD3 algorithm.

The main components of the problem statement are summarized below:

- **Starting Location:** $\mathscr{D}_S = (X_0, Y_0)$
- **Goal Location:** $\mathscr{D}_G = (X_n, Y_n)$
- **Current Robot Position:** $\mathscr{D}_t = (x_t, y_t)$
- **Distance to Goal:** $d_t = \|\mathscr{D}_t - \mathscr{D}_G\|$
- **Obstacles:** $O = \{o_1, o_2, \ldots, o_m\}$
- **Optimize θ to maximize** $P(\theta)$ subject to $C = 0$ and $T \leq T_{\max}$.
- **Collision Constraint:** $C = 0$
- **Time Constraint:** $T \leq T_{\max}$
- **Goal Distance Threshold:** $d_t \leq d_s$, $d_s = 0.5\,\text{m}$, where d_s is the maximum allowable distance.

This navigation problem $\mathscr{T}$ is addressed within the ROS framework, utilizing a DRL approach to enhance the robot's adaptability and robustness in dynamic environments. The focus is on optimizing the TD3 algorithm to effectively navigate through environments containing various obstacles, while also ensuring that the robot can generalize to new, unseen scenarios by tuning the parameters θ for the given environment E.

The proposed solution involves using DRL in conjunction with ROS to handle real-time sensor data, efficiently calculate actions, and optimize the robot's path. By integrating TD3 into the ROS system, the aim is to create a robust navigation model that achieves high success rates while minimizing collisions and ensuring efficient time management.

1.5 Objectives

The primary objective of this thesis is to develop a DRL approach that significantly enhances the transferability and generalization of mobile robot navigation across dynamic and unfamiliar environments. Specifically, the focus is on extending the TD3 model to address the key challenges of navigating unknown environments with sudden changes, thereby improving the robot's overall adaptability and robustness.

The specific objectives of this research are:

1. **To enhance the TD3 model for improved transferability by implementing randomized start and goal points and varying robot orientations during training episodes**. This involves modifying the input observations of the TD3 model to include diverse initial conditions, thus allowing the robot to effectively explore and navigate a wider range of scenarios.

2. **To design and create custom training environments that integrate dynamic elements and complex static layouts.** The training environments consist of 16 unique static scenarios with distinct designs, increased dimensions (16m x 16m), and diverse object configurations. This allows for extensive randomization of start and goal points, enhancing exploration and promoting robust generalization across various obstacle setups.

3. **To evaluate the performance of the enhanced TD3 model using a custom evaluation metric along with a comprehensive set of existing metrics.** The custom evaluation metric considers penalties for collisions and deviations from optimal performance. Specifically, if the robot collides, A penalty is applied to the success score, and the score is also reduced for taking longer than the optimal time. or exceeding the optimal distance (without surpassing the maximum allowable distance or time). The maximum allowed number of collisions is set to 3, after which the success score is significantly impacted. The evaluation also includes transfer tests in unfamiliar and complex environments to assess the model's generalization capabilities.

4. **To provide a detailed analysis of the model's ability to generalize by testing its deployment in novel simulated environments with complex configurations.** Although validation on physical robots is not the focus at this

stage, the model's performance in simulation indicates its potential for future real-world deployment.

5. **To contribute to the research community by sharing the complete implementation, including source code and training environments, as an open-source package on GitHub**. This allows other researchers and practitioners to reproduce the results and further build upon the proposed approach.

These objectives are designed to systematically address the identified research gaps by enhancing the transferability and generalization of robot navigation models. By introducing new training methods, dynamic obstacles, and comprehensive metrics, this thesis aims to develop a more adaptable and scalable solution for wheeled robot navigation in dynamic and complex environments. The goal is to bridge the gap between simulation-based training and real-world applications, ultimately contributing to the broader field of RL and autonomous robotics.

1.6 Summary of Contributions and Experimental Scope.

This thesis presents the results and impacts of two experimental studies aimed at evaluating and improving the performance and generalization of TD3-based navigation models in static environments, as detailed in Chapter 4. Experiment 1, presented in Section 4.2.5, demonstrated that introducing randomized start and goal points during TD3 training significantly enhanced trajectory execution and robust navigation. The extended TD3 model showed improved adaptability in handling unpredictable goal locations and better generalization in previously unseen environments compared to the non-extended model, which performed well only in extremely narrow spaces but struggled with unpredictability. This helped reduce existing gaps in prior work by directly evaluating the impact of randomization on learning efficiency and generalization. Experiment 2, described in Section 4.3, examined the integration of global path-planning strategies with the extended TD3 model, comparing TD3 with and without MoveBase. The results revealed that TD3 without MoveBase achieved higher success rates and more efficient navigation, while the integrated approach, although effective in reducing collisions, introduced inefficiencies in route planning. These findings highlighted the limitations of global strategies when combined with DRL and demonstrated that the extended TD3 model

alone provided superior adaptability in unpredictable conditions. Collectively, both experiments emphasize that structured training diversity and reliance on DRL alone can significantly improve robustness, efficiency, and generalization in RL-based navigation systems.

1.7 Thesis Structure

This thesis is organized into six chapters, each guiding the reader step by step through the research journey. Chapter 1 introduces the topic clearly by presenting the motivation, the current research gaps, the specific objectives, contributions, and the problem statement related to improving robot navigation using DRL—particularly the TD3 algorithm. Next, Chapter 2 dives deeper into the TD3 algorithm itself, clearly explaining its architecture, robustness, and optimization methods for effective training. Chapter 3 then carefully outlines the methodology and simulation environment used to train and evaluate the TD3-based navigation model, highlighting details such as robot dynamics, randomized training environments, and monitoring techniques like TensorBoard. Following this methodological foundation, Chapter 4 presents the core experimental findings, thoroughly analyzing and comparing the performance and generalization ability of the extended and non-extended TD3 models, as well as examining how global path planning influences navigation efficiency. Finally, Chapter 5 wraps up the thesis by summarizing the key findings and conclusions, clearly demonstrating the effectiveness of training with randomized goals for better generalization, and concludes by recommending important directions for future research—including real-world experiments and integrating multiple training strategies to achieve robust navigation systems.

1.8 Notation and Definitions

Below is a list of the mathematical symbols and terms used in the (Section 1.4) :

$\mathscr{D}_S = (X_0, Y_0)$ Starting position of the robot.

$\mathscr{D}_G = (X_n, Y_n)$ Goal position in the environment.

$O = \{o_1, o_2, \ldots, o_n\}$ Set of static obstacles in the environment.

E The environment in which navigation occurs.

θ Parameters of the policy or navigation model.

$f(E, \theta)$ Function mapping environment and policy parameters to a success probability.

P Probability of successfully reaching the goal.

C Total number of collisions.

T Time taken to complete the navigation task.

$T_{\mathbf{max}}$ Maximum allowable time for reaching the goal.

d_t Distance from the robot's current position to the goal.

d_s Distance threshold below which the goal is considered reached.

Chapter 2

TD3 Robustness and Background

2.1 TD3 Robustness Overview

The TD3, introduced by Fujimoto et al. [6], was selected for this research due to its effectiveness in handling continuous action spaces and improving policy stability. TD3 addresses overestimation bias by employing dual Critic networks and enhances learning stability through delayed policy updates. Additionally, TD3 incorporates exploration noise via the Ornstein-Uhlenbeck process to encourage comprehensive exploration of the action space during training.

Empirical evaluations on various benchmark environments from OpenAI Gym demonstrate TD3's superiority over earlier methods, such as DDPG, particularly regarding sample efficiency and exploration capabilities [25]. TD3's robustness and consistent performance in dynamic path-planning tasks further highlight its suitability for complex real-world applications [27]. Moreover, recent studies, including Ali (2024), confirm TD3's robustness, emphasizing its stability, time-efficiency, and ability to generalize to unseen environments, aligning well with the practical objectives of this research.

This chapter focuses specifically on TD3, detailing its architecture, core mechanisms, and its role in ensuring efficient and robust policy learning.

2.2 TD3 Architecture and Principles

TD3 consists of a policy network (actor), two critic networks, and corresponding target networks. These components collectively enhance policy optimization by leveraging multiple Q-value estimates. TD3 follows an actor–critic architecture in which the actor network selects actions to maximize the expected return, while the critic networks evaluate the quality of these actions through Q-value estimation. To reduce overestimation bias caused by function approximation errors, TD3 employs two critic networks and computes target values using the minimum of their Q-value estimates. Policy updates are intentionally delayed by updating the actor network less frequently than the critics. In addition, the overall structure of TD3 is illustrated in Figure 2.1.

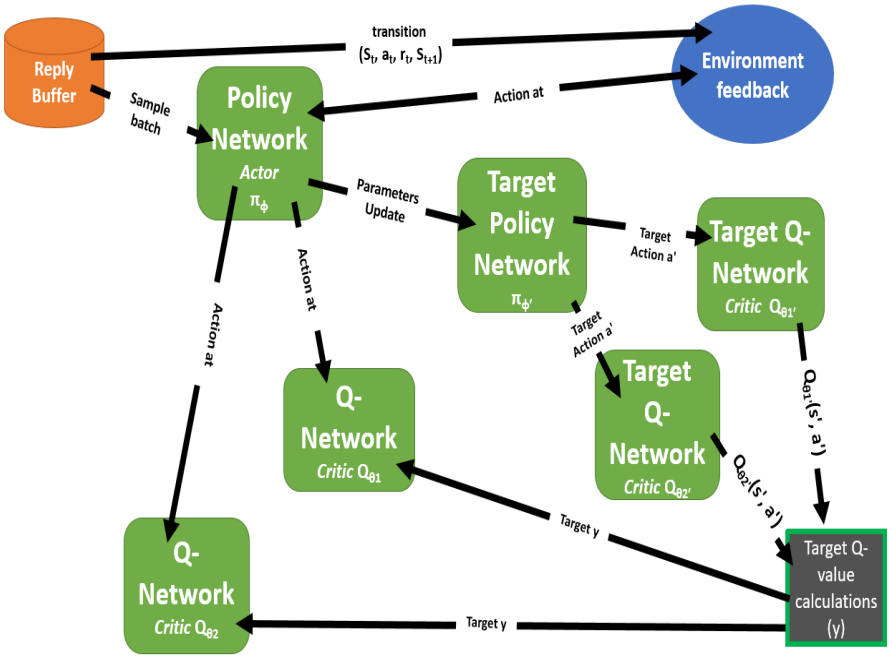


Fig. 2.1 TD3 algorithm architecture illustrating the connections between the policy network (actor), the twin critic networks, and the target networks.

2.3 Policy Network Updates

The policy network update in TD3 follows the deterministic policy gradient theorem, aiming to maximize the expected return. The mathematical formulation of this optimization is provided in Equation 2.1:

$$J(\phi) = \mathbb{E}_{s\sim p^{\pi}}\left[Q^{\pi}(s,\pi_{\phi}(s))\right] \tag{2.1}$$

The policy gradient is computed based on the gradient of the Q-value function with respect to the action and the gradient of the policy function itself. The corresponding formulation is given in Equation 2.2:

$$\nabla_{\phi} J(\phi) = \mathbb{E}_{s\sim p^{\pi}}\left[\nabla_{a} Q^{\pi}(s,a)\big|_{a=\pi_{\phi}(s)} \nabla_{\phi}\pi_{\phi}(s)\right] \tag{2.2}$$

To refine the policy network parameters, TD3 updates them using a mini-batch learning approach. The parameter update rule is provided in Equation 2.3:

$$\phi \leftarrow \phi + \alpha \cdot \frac{1}{N}\sum_{i=1}^{N}\left[\nabla_{a} Q_{\theta_1}(s_i,a_i)\big|_{a_i=\pi_{\phi}(s_i)} \cdot \nabla_{\phi}\pi_{\phi}(s_i)\right] \tag{2.3}$$

2.4 Q-Network Updates

TD3 mitigates overestimation bias in Q-learning by utilizing two Q-networks, each providing an independent estimate of the state-action value. The use of clipped double Q-learning ensures that the minimum of the two Q-values is used for policy evaluation, reducing upward bias in Q-value estimation.

The target Q-value computation is detailed in Equation 2.4:

$$y = r + \gamma \min(Q_{\theta_1'}(s',\tilde{a}), Q_{\theta_2'}(s',\tilde{a})) \tag{2.4}$$

While the Q-network update mechanism is formulated in Equations 2.5 and 2.6:

$$\theta_1 \leftarrow \arg\min_{\theta_1} \frac{1}{N}\sum_{j=1}^{N}\left(y_j - Q_{\theta_1}(s_j,a_j)\right)^2 \tag{2.5}$$

$$\theta_2 \leftarrow \arg\min_{\theta_2} \frac{1}{N} \sum_{j=1}^{N} \left(y_j - Q_{\theta_2}(s_j, a_j)\right)^2 \tag{2.6}$$

2.4.1 Actor Target Network Update

The actor target network, denoted as $\pi_{\phi'}$, is a delayed copy of the actor network and is used to generate stable action targets for the critic updates. The actor target network changes gradually over time, ensuring that the target actions used in critic updates remain consistent and stable. This stability is critical for reducing the variance in the Q-value targets.

At initialization, the actor target network parameters are set equal to those of the actor network:

$$\phi' \leftarrow \phi. \tag{2.7}$$

During learning, the actor target network is used to compute the target action for the next state, which is required for calculating the target Q-value in the critic update:

$$\tilde{a} = \pi_{\phi'}(s') + \varepsilon, \quad \varepsilon \sim \text{clip}(\mathcal{N}(0, \tilde{\sigma}), -c, c). \tag{2.8}$$

The Gaussian noise term ε is added to the target action to regularize the critic update and to ensure smoother and more stable target Q-value estimation during training. The actor target network parameters are subsequently updated using a soft update mechanism that slowly tracks the actor network:

$$\phi' \leftarrow \tau\phi + (1-\tau)\phi'. \tag{2.9}$$

By updating the actor target network more slowly than the actor network, TD3 ensures smooth policy evolution and more reliable critic target estimation.

2.4.2 Overestimation Bias and Mitigation Approaches

In standard Q-learning, overestimation bias occurs due to function approximation errors when selecting the maximum Q-value. This leads to an overestimation of the Q-function, which results in learning a suboptimal policy. To mitigate this, Double

Q-learning introduces two independent Q-value estimators, and TD3 further refines this technique by computing the minimum Q-value between two estimators.

The TD3 approach improves upon Double Q-learning by incorporating a delayed policy update strategy, preventing premature policy updates based on biased Q-values.

2.4.3 TD3 Algorithm Q-Network Updates

The TD3 algorithm maintains two Q-networks along with their target networks. During training, a mini-batch of transitions is sampled from the replay buffer, and the target action is computed using the target policy network with noise regularization. The formulation for this process is presented in Equation 2.4.

TD3 updates the Q-networks by minimizing the mean squared error between the estimated Q-values and the target Q-values. The optimization formulation is provided in Equations 2.5 and 2.6.

2.5 Target Networks Updates

To stabilize learning, TD3 employs target networks that provide slowly varying targets for critic updates. At initialization, the target networks are set equal to their corresponding online networks:

$$\theta_1' \leftarrow \theta_1, \quad \theta_2' \leftarrow \theta_2. \tag{2.10}$$

During training, the target critic networks are updated using a soft update mechanism based on an exponential moving average:

$$\theta_i' \leftarrow \tau\theta_i + (1-\tau)\theta_i', \quad i \in \{1,2\}. \tag{2.11}$$

This slow update strategy ensures smooth evolution of the target Q-values and improves the stability of critic learning.

2.6 Conclusion

TD3 represents a significant improvement in DRL by effectively mitigating overestimation bias while ensuring stable policy optimization. Its architecture, policy updates, Q-network updates, and target network mechanisms contribute to its success in continuous control tasks.

This chapter provides a conceptual overview of TD3, focusing on its robustness and optimization mechanisms. The following chapters will discuss its integration with ROS-based robotic systems and evaluate its performance in realistic simulation environments.

2.7 TD3 Notation and Parameters

Below is a list of the mathematical symbols and terms used throughout this chapter:

α Learning rate of the policy network.

γ Discount factor in reinforcement learning.

τ Target network soft update rate in TD3.

N Mini-batch size for training updates.

r Immediate reward at each transition.

p^{π} State distribution under policy π.

ϕ Parameters of the policy (actor) network.

ϕ' Parameters of the target policy network.

θ_1, θ_2 Parameters of the twin Q-networks (critics).

θ_1', θ_2' Parameters of the target Q-networks.

$Q_{\theta_1}, Q_{\theta_2}$ Q-functions representing value estimators in TD3.

$Q_{\theta_1'}, Q_{\theta_2'}$ Target Q-functions used for bootstrapping.

π_ϕ Policy function parameterized by ϕ.

$\pi_{\phi'}$ Target policy function parameterized by ϕ'.

$J(\phi)$ Expected return under policy π_ϕ.

$\nabla_\phi J(\phi)$ Gradient of the expected return with respect to the policy parameters.

$\nabla_a Q^\pi(s,a)\big|_{a=\pi_\phi(s)}$ Gradient of Q-value w.r.t. action.

$\nabla_\phi \pi_\phi(s)$ Gradient of the policy function with respect to ϕ.

y Target Q-value calculated for the critic update.

$\tilde{a}$ Noisy action computed using the target policy with added noise.

ε Gaussian noise added to the target action.

$\mathcal{N}(0,\tilde{\sigma})$ Zero-mean Gaussian noise with standard deviation $\tilde{\sigma}$.

$\textbf{clip}(\mathcal{N}(0,\tilde{\sigma}),-c,c)$ Clipped noise for bounded action smoothing.

Chapter 3

Research Methodology and Simulation Framework

The research methodology presented in this chapter is built upon a comprehensive simulation infrastructure, which integrates a Singularity container [28], the ROS Noetic framework, the Jackal robot model, and a specifically designed OpenAI Gym environment. These core components collaborate closely with DRL, which is implemented using PyTorch version 1.10.1, developed by the PyTorch Foundation under the Linux Foundation [29], forming a robust setup for training the robot on navigation tasks.

All simulations are carried out within the Gazebo simulator, specifically using the customized MotionControlContinuousLaser environment tailored for the Jackal robot. This environment provides the robot with a continuous action space defined by linear and angular velocities, allowing for fluid and precise movements. Additionally, the environment processes sensory data collected via LiDAR sensors, which form part of the observation space accessible to the reinforcement learning agent. By incorporating real-time sensor processing with continuous motion control, this setup creates a realistic and challenging environment suitable for evaluating and training advanced DRL algorithms. Training data generated through interactive simulation sessions are saved locally within a replay buffer, from which random sampling is conducted during the iterative training phases. Continuous updates of model parameters and policy improvements occur iteratively until the model achieves convergence.

The computational setup facilitating the extensive training sessions relies on an AMD Ryzen 9 3900X CPU (12 cores and 24 threads), an NVIDIA GeForce GTX 1660 Ti GPU, and 32 GB of RAM. Despite the availability of GPU hardware, the entire training process utilized only CPU resources, simultaneously running six training models within the Singularity container. The complete training process lasted approximately five days.

3.1 Jackal Robot Dynamics and Sensors

The Jackal robot is equipped with two principal sensors, each directly interfaced with the DRL algorithm and the robot's control framework:

- **Laser Sensor**: This LiDAR sensor scans the robot's surroundings to detect and measure distances to obstacles, generating a detailed 720-dimensional laser scan. The scan data directly feed into the DRL algorithm, playing a crucial role in the robot's obstacle avoidance and path selection strategies. Within the Gazebo simulator, the sensor accurately simulates real-world operational characteristics, ensuring robust training in complex navigation scenarios.
- **Velocity Sensor**: This sensor records linear and angular velocity commands issued by the DRL module. The sensor feedback verifies that the robot's movements align with the intended commands, helping maintain stable navigation and detect any deviations that could affect overall performance.

3.1.1 Training and Evaluation Scenarios

In this research, the training methodology for the TD3 algorithm was significantly enhanced through extensive modifications of the environmental configuration codes, specifically aiming at boosting the adaptability and generalization of the robot in diverse navigation tasks. This methodological improvement builds upon the foundational studies previously conducted by Xu et al. [25, 30], which originally focused on scenarios featuring fixed start and goal positions. Unlike the earlier work, this research introduces notable adjustments such as randomizing the start and goal locations, varying object placements, and diversifying the paths available to the robot. These extensive changes, further described in Section 3.1.1, present the DRL model

with numerous and unpredictable navigational conditions, thereby mitigating the risk of the model memorizing fixed trajectories and becoming overly specialized.

Because the actor network is inherently reliant on real-time observations to select suitable actions, these newly introduced randomized conditions compel it to consistently generate optimal solutions dynamically, rather than falling back on memorized shortcuts. Similarly, the critic networks, which estimate the Q-values that inform the action selection, also benefit greatly from this enhanced variety in the environment. They receive a broader, more representative set of state-action pairs, thereby enhancing their accuracy in estimating Q-values across different scenarios. By expanding the diversity of experiences available during training, the TD3 model becomes more proficient in making optimal action choices, increasing both its learning effectiveness and its capacity to generalize learned skills to new situations. Ultimately, this comprehensive training approach aids the model in uncovering the most effective strategies in complex and highly unpredictable navigation environments.

The core contribution and primary advancement introduced by this research is precisely the expansion of the training framework, achieved by integrating numerous scenarios that feature randomized starting and goal positions. This randomization markedly increases the complexity and variety of the training conditions, distinguishing it significantly from the original scenarios characterized by fixed points. The expanded scenario framework promotes improved exploratory behaviors, enhances the robustness of the learned policies, and ensures greater adaptability in managing uncertain or previously unseen paths.

To clearly illustrate and validate the effectiveness of this expanded training methodology, the TensorBoard visualization tool was employed as a graphical interface for continuous monitoring. This enabled tracking the convergence behavior of the model, observing stability in the training progress, and assessing critical performance metrics. Such real-time visualization provided detailed insights into the algorithm's learning trajectories, clearly demonstrating how the introduced randomization impacts the robot's navigational capabilities and generalization potential.

Further comprehensive details about the specific implementation of these environmental adjustments, as well as the exact training configuration settings utilized, can be found in [31].

Static Box Environments

Initially, training was conducted within static box environments measuring 4.5m x 4.5m, replicating conditions previously established in the work by Xu et al. and Daffan et al. [25, 30]. These environments comprised stationary obstacles arranged systematically, offering a controlled context for assessing navigation performance under consistent conditions. The configuration of these environments is detailed in Figure 3.1.

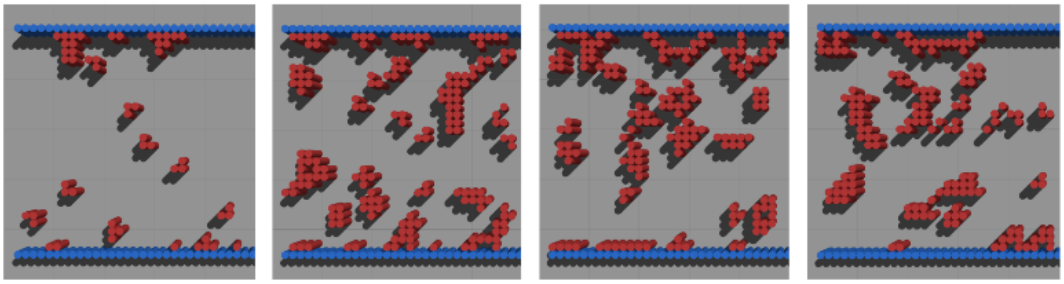


Fig. 3.1 Original Training Setup: Layout of four different Static Box Environments (4.5m x 4.5m) for training.

Custom Static Environments with Randomized Start and Goal Points

This research utilizes specially designed environments consisting of 16 distinct scenarios, each characterized by unique configurations and varying positions of obstacles, offering a wide spectrum of navigational challenges for the robot. Unlike previous setups that employed relatively small 4.5m x 4.5m static box environments, these custom-designed environments have been significantly enlarged to dimensions of 16m x 16m. This increase in spatial dimensions allows for substantial flexibility, enabling extensive randomization of the start and goal points during each training episode. By providing a larger and more open navigational area, these environments significantly enhance the robot's ability to explore diverse pathways and interactions, thereby improving its generalization capabilities when encountering unfamiliar obstacles and path configurations. Each scenario within this set of environments was explicitly crafted to present distinct layouts, varying degrees of navigational complexity, and diverse arrangements of static obstacles, all intended to reinforce the robot's robustness, adaptability, and effectiveness when navigating complex, real-world-like conditions.

The scenarios developed for this research incorporate two specific types of static objects: boxes and cylinders, each with their own well-defined dimensions and functional purposes within the environment. Box-shaped obstacles consistently maintain uniform dimensions of 1 meter in length, width, and height, and they serve as standard impediments strategically placed throughout the navigation area to introduce variability into the environment layouts. Cylindrical obstacles, however, vary more significantly in dimensions, with five clearly defined types: Cylinder 1, measuring 0.14 meters in radius and 1 meter in height; Cylinder 2, measuring 0.22 meters in radius and 1 meter in height; Cylinder 3, measuring 0.30 meters in radius and 2.19 meters in height; Cylinder 4, measuring 0.13 meters in radius and 1 meter in height; and Cylinder 5, measuring 0.30 meters in radius and 2.19 meters in height. Cylinder 5, specifically, is used not only within the navigation area but also for constructing the outer boundary walls of each environment. All these obstacles remain static and fully collidable, meaning that any contact or collision involving the robot during its navigation task is recorded and included in the performance evaluation metrics. The deliberate distribution of these obstacles across the 16 custom scenarios ensures a rich variety of obstacle densities, path complexity, and navigational routes, intentionally challenging the robot and consequently enhancing its ability to generalize across varying environmental contexts.

To ensure a consistent level of navigational challenge and feasibility, both the start and goal positions within these environments are randomly generated according to clearly defined validation criteria. The randomization methodology is constrained such that the distance between any randomly generated start and goal point must be at least 8 meters but not exceed 11 meters. This restriction ensures that the generated scenarios provide an adequately balanced level of complexity and navigational difficulty. Additionally, both start and goal points are required to maintain a safe margin of at least 1.7 meters from any obstacle. This criterion prevents the robot from initiating its navigation in areas too close to obstacles or generating unreachable targets. Points failing to meet these requirements are discarded, prompting the system to repeatedly generate new random points until valid coordinates are identified.

Furthermore, the initial orientation (heading) of the robot is randomized at the start of every navigational attempt. This randomization occurs concurrently with the selection of new start and goal points. By doing so, the robot is consistently exposed to varied initial directions and paths, thus enhancing its adaptability and improving its competence in handling unpredictable or novel navigation challenges.

This systematic randomization procedure is implemented not only at the initiation of each new navigation episode but also following any unsuccessful attempt, thereby ensuring each navigational task commences with newly randomized parameters, maintaining freshness and variability.

The identical randomization and validation process described above is consistently employed during the evaluation phases as well, ensuring that both the extended and baseline (non-extended) models undergo testing under identical and previously unseen environmental conditions. Representative examples of these custom-designed environments are illustrated clearly in Figure 3.2.

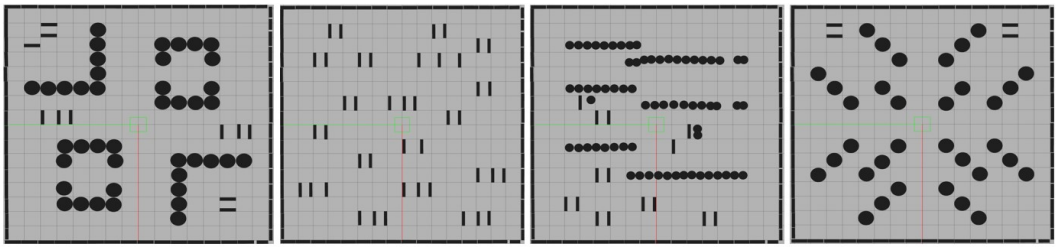


Fig. 3.2 Layout of the four different 16m x 16m custom static environments for training.

3.2 Graphical Monitoring of Training Process

To ensure thorough monitoring of the training process and effectively track the performance metrics, TensorBoard was adopted. This tool provided continuous tracking of key indicators such as the success rate and collision frequency throughout the training duration. The training activities were comprehensively documented through event logs generated during model execution, as detailed explicitly in [31]. By utilizing these logs, it became possible to visualize the evolution of critical performance measures, specifically the success and collision rates, as the training progressed. These visual representations are demonstrated clearly in Figures 3.3 and 3.4. These figures highlight precisely how the implementation of randomized start and goal points affected the robot's learning progression. The graphical monitoring thus provided an effective and detailed assessment of the training dynamics, enabling clear observation of model convergence and overall stability during training.

Fig. 3.3 Success rates over training steps for the extended model (blue) and the non-extended model (orange), obtained from a single training run using a fixed random seed (Seed 14). The curves illustrate qualitative learning trends rather than statistical averages across multiple seeds.

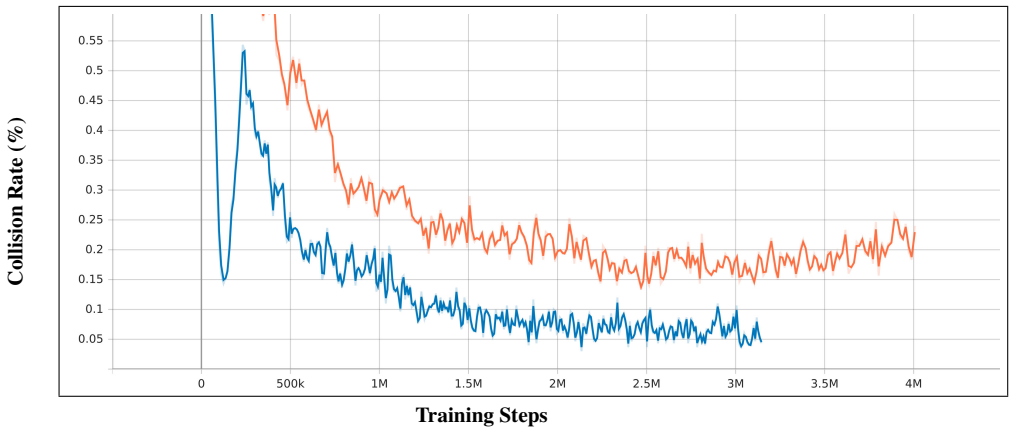


Fig. 3.4 Collision rates over training steps for the extended model (blue) and the non-extended model (orange), obtained from a single training run using a fixed random seed (Seed 14). The curves are reported to illustrate training behavior trends rather than confidence intervals over multiple seeds.

3.3 TD3 Training Framework

3.3.1 Training Architecture

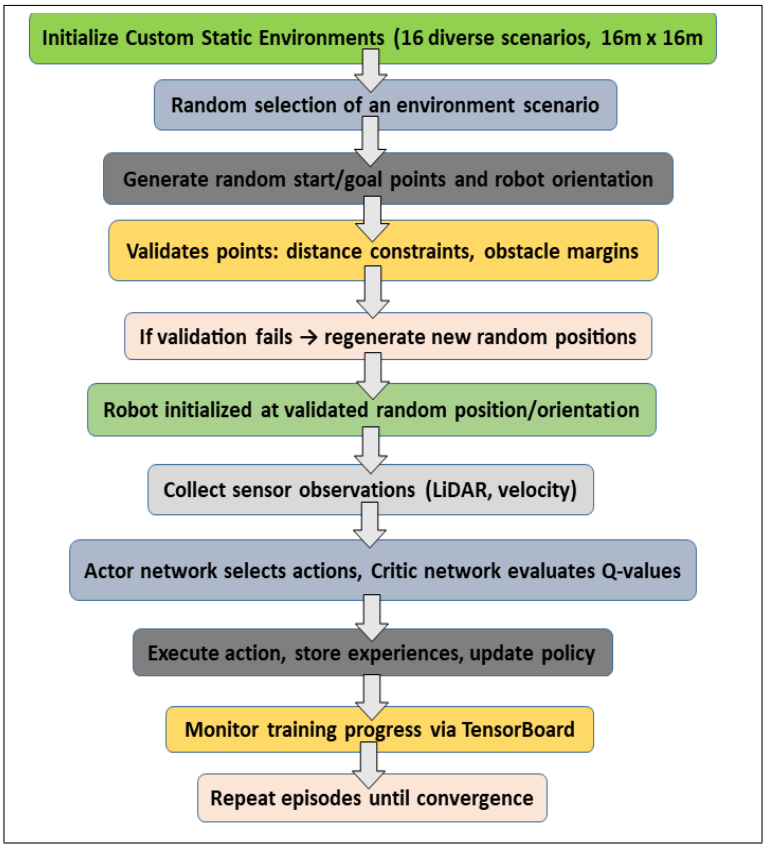


Fig. 3.5 Block diagram of the expanded TD3 training methodology with randomized start and goal points.

3.3.2 Training Process Flowchart

The TD3 training protocol commences with environment initialization and data acquisition via robot sensors. The actor network selects actions according to the policy, which are subsequently evaluated by the critic networks, generating Q-values. The robot executes the chosen actions, records environmental feedback, and logs experiences in a replay buffer. The training cycle continues iteratively, updating both the actor and critic networks until convergence. Finally, the trained model is deployed for practical use, as depicted in the detailed flowchart provided in Figure 3.6.

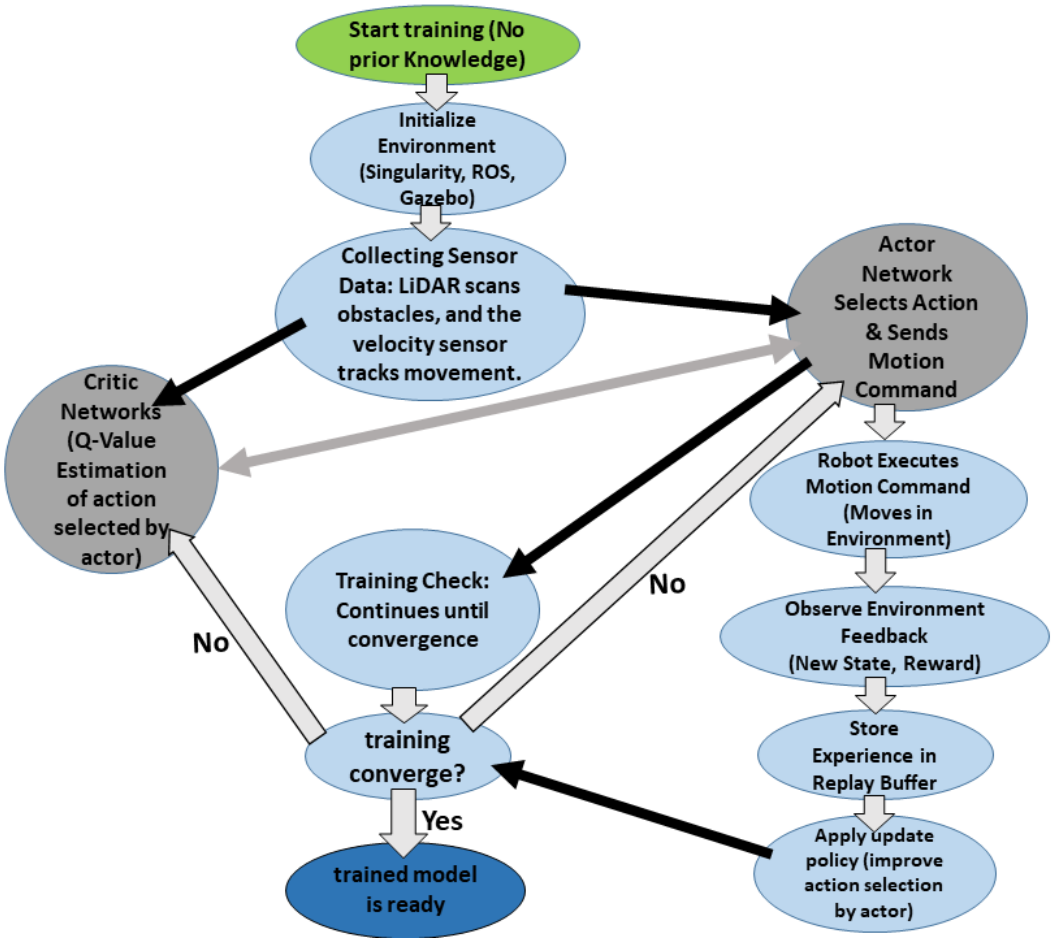


Fig. 3.6 Flowchart of the TD3 training process.

Chapter 4

Experimental Results and Analysis

4.1 Experimental Comparison of TD3 Models

The objective of this experiment is to evaluate the performance of an improved TD3 model, trained using 16 specifically designed static environments as described in Chapter 3, against a baseline model developed by Xu et al. [30], which was trained on 300 static environments from their original experimental configuration. Through this comparative analysis, the study investigates how variations in environmental diversity and training approaches influence overall performance outcomes.

4.1.1 Training and Evaluation Metrics

To systematically evaluate the extended TD3 model, an extensive range of performance metrics was implemented, providing detailed insights into the model's navigation performance within previously unseen environments. These metrics include:

- **Success Rate**: Defined as the proportion of episodes in which the robot successfully reaches its goal without collisions.
- **Collision Rate**: Defined as the proportion of episodes during which the robot collides with obstacles.

- **Episode Length**: The average number of steps taken by the robot to complete each navigation task.
- **Average Return**: The cumulative rewards collected by the robot during each episode, averaged across all episodes, serve as a measure of the learning efficacy.
- **Time-Averaged Number of Steps**: The mean number of steps the robot requires to complete an episode, giving insight into the efficiency of its navigation approach.
- **Total Distance Traveled**: Recorded during the testing phase, this metric measures the cumulative distance covered by the robot per episode in meters, providing an additional evaluation of the efficiency of its path-planning strategy.

4.1.2 Baseline Comparison in Static Environments

To fairly evaluate the effectiveness of the proposed training strategy—particularly the randomized selection of starting and goal positions—a baseline comparison was performed in a simulated static environment setting, specifically considering only DRL-based navigation strategies for consistency. Cimurs et al. [23] performed several simulation evaluations, but only their static environment results were used here, employing TD3 with a global navigation strategy (GDAE) as a baseline for comparison. Likewise, Akmandor et al. [19] evaluated methods across static and dynamic scenarios, but their static environment outcomes were exclusively included to maintain fairness. Two models (Tentabot FC and Tentabot 1DCNN FC, both trained via PPO without global strategies) from Akmandor et al. were averaged into a single metric for clarity. The proposed TD3 approach with a global strategy was assessed within a maze-like static scenario characterized by long walls, thereby allowing direct comparison to other DRL methodologies tested within static environments, thereby allowing direct comparison to other DRL methodologies tested within static environments (Table 4.1).

Table 4.1 Comparison of Simulation Results in Static Environments. **Bold** values indicate better performance for the corresponding metric.

Author	Method Used	Obstacle Type & Difficulty	Success Rate (%)	Time (s)	Distance (m)
Cimurs et al. [23]	TD3 (With Global Strategy)	Static environment with smooth walls and multiple local optima	**100**	88.03	41.42
Akmandor et al. [19]	PPO (Without Global Strategy)	Static environment with distinct structured obstacles	55	-	-
Proposed Method	TD3 (With Global Strategy)	Maze-like static environment with long walls	80.0	**16.36**	**22.87**

According to results reported by Cimurs et al. [23], TD3 with a global strategy achieved a 100% success rate in structured static scenarios, underscoring its effectiveness, although at the expense of increased travel time. Conversely, the PPO approach used by Akmandor et al. [19], without a global guidance strategy, reached only 55% success, emphasizing limitations of local-only navigation strategies. The method proposed in this study, employing TD3 with a global navigation strategy, attained an 80% success rate within a notably more complex maze-like static environment and demonstrated superior navigation efficiency with significantly reduced completion time and path length compared to the approach by Cimurs et al.

4.2 Experiment 1: Performance Evaluation of the Extended and Non-Extended Models

This section presents a comparative assessment of the extended model (shown in blue in the plots) versus the non-extended model (shown in orange in the plots). The focus is on key metrics such as success rate, collision rate, and average return.

4.2.1 Reward Calculation and Success Rate Comparison

The reward function employed in this study follows the formulation presented by Xu et al. [25, 30]. It is composed of a slack reward, a progress reward, a collision penalty, a success reward, and a failure penalty. This formulation ensures an appropriate balance between progress toward the goal, avoidance of collisions, and successful goal completion. Given that the primary objective here is to enhance the model's

capacity for generalization by randomizing both start and goal positions, the existing reward function design was retained without modification.

At each timestep in an episode, the immediate reward is computed based on several contributing factors. The total reward for a single episode is then defined as the sum of these step-wise rewards, as indicated in equation (4.1).

$$R = \sum \Big(-0.1 + (\text{goal\_reward} \cdot \text{progress}) - 1.0 \cdot \mathbf{1}_{\text{collision}} + 20.0 \cdot \mathbf{1}_{\text{success}} - 10.0 \cdot \mathbf{1}_{\text{failure}} \Big) \tag{4.1}$$

where:

- **Slack reward:** A penalty of -0.1 is applied at every timestep.
- **Progress reward:** A reward of $+1.0$ is granted for every meter moved closer to the goal.
- **Collision penalty:** A penalty of -1.0 is applied in the event of a collision.
- **Success reward:** A reward of $+20.0$ is given upon successful arrival at the goal.
- **Failure penalty:** A penalty of -10.0 is imposed if the episode concludes unsuccessfully.

When multiple episodes are considered, the overall cumulative reward is the total of all episode-level rewards within the current buffer. For instance, for a buffer containing up to 300 episodes, the total cumulative reward can be expressed as (4.2):

$$R_{\text{total}} = R_{\text{Episode 1}} + R_{\text{Episode 2}} + \cdots + R_{\text{Episode 300}} \tag{4.2}$$

Thus, each episode's overall reward (summed over its steps) contributes to this total. As new episodes are added and older ones removed, the cumulative reward reflects up to the most recent 300 episodes. Higher cumulative rewards correlate with stronger overall performance, characterized by higher success rates and more effective navigation.

Figure 3.3 illustrates that the extended model reached a 95% success rate and converged prior to 2 million training steps, whereas the non-extended model attained

roughly 70% success only after 3 million steps. These findings indicate that the extended model not only learns more quickly but also operates more efficiently, a result attributed to the randomization in start and goal positions.

4.2.2 Collision Rate Analysis

As depicted in Figure 3.4, the extended model eventually settled at a collision rate of approximately 4%, whereas the non-extended model ended at around 24%. Both models showed early-stage improvements, but after 3.5 million steps, the non-extended model's collision rate began to climb again, possibly signaling overtraining. By contrast, the extended model maintained a steady decline in collision frequency.

4.2.3 Training Efficiency and Convergence

The extended model not only reached its peak performance more quickly but also demonstrated superior efficiency in learning, requiring fewer steps to converge. Randomly varying the start and goal positions allowed the extended model to encounter a wider range of scenarios, which in turn led to higher success rates. Conversely, the non-extended model—trained with consistent start and goal positions—improved at a slower rate and achieved a lower final success ratio. Furthermore, some of the training environments for the non-extended model were extremely narrow, which may have restricted the robot's potential for a higher success rate.

4.2.4 Transfer to Test Environments Using TD3 and Global Path Planning

To assess how well the extended model generalized, a set of transfer tests was carried out through the Competition Package [32]. Originally created by previous investigators for testing purposes, this package was used in its existing form, without alterations or extensions, during the study. However, for the sake of evaluation, the Race World environment [33]—which was not included by default—was integrated into the package. Race World features a maze-like setup with extensive wall segments, creating a demanding setting for navigation. Both the extended model and the non-extended model were validated within this augmented environment.

All methodological improvements were confined to the training phase in a separate package, while the Competition Package itself (with the added Race World) served as a neutral testing ground to gauge the models' ability to operate effectively in intricate environments.

During these tests, the MoveBase package [34] was used to produce a global path, whereas the principal navigation remained driven by TD3. Specifically, MoveBase generated a global route from the starting point to the target, serving as a reference path. Concurrently, the TD3-based controller handled local decision-making in real time for obstacle avoidance and general movement. This integrated approach provided broad guidance through global path planning, while the TD3 controller's flexibility enabled instantaneous adjustments to challenging local conditions. It is noteworthy that the MoveBase global planner in this study relied exclusively on odometry and real-time LiDAR data, devoid of any preloaded map or global localization. This underscores the local adaptation capacity of the DRL-based controller in uncharted environments.

Evaluation Results

Tables 4.2 and 4.3 display the respective outcomes for the extended and non-extended models, presenting data such as total travel distance, collision frequency, and success indicators for each of the ten test paths. The paths used in these experiments are illustrated in Figure 4.1. Consistently, the extended model delivered superior performance: it attained higher success rates with fewer collisions and frequently matched near-optimal path lengths. In contrast, the non-extended model struggled on more complicated routes, often failing to complete the path or incurring heavier collision penalties.

Table 4.2 Evaluation results for the extended model. **Bold** entries highlight superior performance relative to the non-extended model on distance, collisions, goal reached, and time.

Path	Distance (m)	Collisions	Goal Reached	Time (s)
Path 1	**18.72**	**0**	**Yes**	**9.77**
Path 2	**47.16**	1	**Yes**	**31.66**
Path 3	**13.81**	**0**	**Yes**	**8.28**
Path 4	**26.41**	4	No	**21.79**
Path 5	**15.51**	**0**	**Yes**	**9.26**
Path 6	8.12	**0**	**Yes**	**4.19**
Path 7	**15.04**	3	**Yes**	9.87
Path 8	37.74	**0**	No	**35.49**
Path 9	**15.66**	1	**Yes**	**9.98**
Path 10	**30.54**	**0**	**Yes**	**23.33**

Table 4.3 Evaluation results for the non-extended model. **Bold** entries denote superior performance compared with the extended model for distance, collisions, goal reached, and time.

Path	Distance (m)	Collisions	Goal Reached	Time (s)
Path 1	33.53	4	No	33.79
Path 2	41.97	**0**	No	80.00
Path 3	10.65	4	No	14.88
Path 4	39.40	**3**	No	67.59
Path 5	32.09	**0**	**Yes**	42.20
Path 6	8.12	**0**	**Yes**	4.24
Path 7	15.22	**3**	**Yes**	**9.55**
Path 8	**37.75**	**0**	No	39.67
Path 9	44.95	**0**	No	56.42
Path 10	58.16	**1**	No	68.19

Analysis of Results

Success Rate and Distance Traveled: In terms of both success rate and distance traveled, the extended model outperformed the non-extended model. For example, on Path 1, the extended model navigated 18.72 m without collisions, yielding a perfect (100%) success metric. The non-extended model, however, failed on this path, traveling 33.53 m with four collisions and registering a 0% success outcome. This underscores the extended model's ability to adhere more closely to an optimal route, highlighting robust generalization and effective navigation decisions. Meanwhile, the non-extended model faced considerable hurdles in unfamiliar scenarios.

Collision Rate and Penalties: Across nearly all paths, the extended model showed notably fewer collisions—often zero—while the non-extended model recorded multiple collisions more frequently. On Path 4, for instance, the extended model failed the task after four collisions, leading to 0% success, while the non-extended model also failed, with three collisions and an even more substantial distance penalty. By contrast, whenever the extended model entirely avoided collisions on other paths, it routinely reached 100% success, highlighting strong collision avoidance and well-optimized navigation.

Time and Distance Efficiency: In every path, the extended model showed faster completion and covered shorter routes overall. Path 5 serves as an illustrative example: the extended model finished in 9.26 s, covering 15.51 m, whereas the non-extended model took 42.20 s to traverse 32.09 m, which deviated more significantly from an optimal path. These contrasts underscore the extended model's precision in real-time navigation decisions and its capacity to finish tasks expediently without veering off the ideal trajectory.

Generalization and Adaptability: The extended model's robust generalization to unencountered scenarios was evident in its successful traversal of paths deemed highly challenging, maintaining few collisions and limiting travel distance (see Tables 4.2 and 4.3). Figures 4.2 and 4.3, depicting representative runs on Path 1 and Path 2, visually illustrate the difference: the extended model followed a smooth, efficient route, whereas the non-extended model's path exhibited significant diversions and demonstrated difficulty adapting to the environment.

Challenges with Complex Goal Configurations: The non-extended model encountered substantial difficulties in paths that were not straightforward or required

more nuanced turns, suggesting a deficit in its capacity to handle tricky layouts. Because it was trained on static start and goal points, the non-extended model mostly gained proficiency in simpler navigation tasks. This shortcoming became obvious with routes featuring multiple turns and lengthy walls (for example, Path 2 or Path 8). On Path 2, the extended model finished successfully with a 57.60% success rate and did not time out, while the non-extended model traveled 41.97 m and timed out before reaching the goal.

The extended model's capacity to handle more convoluted end goals is attributed to its training protocol, which randomized start and goal positions. This broadened exploration of possible configurations and bolstered adaptability to unfamiliar settings, leading to higher completion rates and fewer penalties. Additional navigation demonstrations can be viewed at [35].

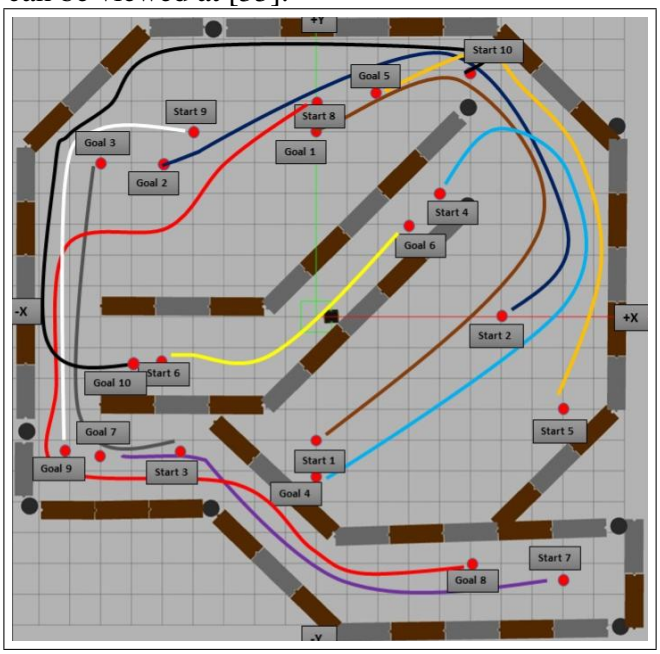


Fig. 4.1 The environment and the ten start–goal pairs used during evaluation. The depicted lines simply illustrate how each start point corresponds to its goal for both extended and non-extended models. They are not the actual routes followed in practice. Each model generated distinct navigation trajectories based on its trained policy.

4.2.5 Performance Evaluation of the Extended and Non-Extended Models

In real-world navigation tasks, assessing robot performance requires more than a simple binary outcome of success or failure. To more accurately represent the robot's behavior, a customized navigation performance scoring metric is introduced in this study. Unlike conventional binary evaluation methods, this score assigns a percentage value ranging from 0% to 100%, enabling a more refined assessment of navigation quality. The metric is designed to evaluate a robot's overall navigation performance based on three essential criteria: total distance traveled, time required to complete the task, and the number of collisions encountered.

This performance score enables partial credit in scenarios where the robot successfully reaches the goal but does so with certain inefficiencies—such as taking a longer path or experiencing minor collisions. Such graded assessment is particularly valuable in real-world contexts, where some level of deviation or minor impact is tolerable. By integrating these performance aspects, the proposed metric offers a more realistic evaluation of the robot's navigation strategy, highlighting its adaptability, efficiency, and safety.

The scoring process begins with an initial perfect score of 100%, from which penalties are subtracted depending on deviations in performance. The overall navigation performance score is defined in Equation (4.3):

$$\text{Score} = 100 - \text{total\_penalty} \tag{4.3}$$

The total penalty applied to the score, denoted as total_penalty, is the sum of three individual penalty components, as shown in Equation (4.4):

$$\text{total\_penalty} = P_d + P_t + P_c \tag{4.4}$$

In the implementation, each penalty is applied only when its corresponding threshold is exceeded; otherwise, the penalty is set to zero. This conditional evaluation prevents negative penalties and ensures that the final navigation score is bounded within $[0, 100]$. The constants used in the penalty formulation are chosen to keep the navigation score simple, balanced, and easy to interpret. The maximum

possible penalty is 100%, which is divided approximately equally among the three evaluation criteria: distance traveled, navigation time, and collisions. For this reason, the distance and time penalties are each limited to a maximum of 33%. The collision penalty is defined as 11% per collision and is limited to a maximum of three collisions, which also results in a maximum collision penalty of 33%. The time threshold of 40 s is selected as a nominal reference duration, corresponding to half of the maximum allowed navigation time, beyond which slower navigation performance is progressively penalized.

The maximum allowed distance (max_allowed_distance) is defined as four times the optimal distance and represents an upper bound beyond which navigation is considered inefficient or unsuccessful. The actual navigation time (actual_time) is measured from the moment the robot starts moving until it reaches the goal or a termination condition is triggered. The reference time of 40 s is a design choice corresponding to half of the maximum allowed navigation duration of 80 s used in the experiments and serves as a nominal completion time for penalizing delayed navigation behavior.

Each of the components—distance penalty P_d, time penalty P_t, and collision penalty P_c—is calculated as follows:

Distance Penalty (P_d): This penalty is imposed when the robot's traveled path exceeds twice the optimal path length. The excess distance is scaled in proportion to the allowable maximum, and the resulting penalty is capped at a maximum of 33%. The mathematical expression used to compute this component is provided in Equation (4.5):

$$P_d = \max\left(0,\ \frac{\text{path\_length} - 2 \times \text{optimal\_distance}}{\text{max\_allowed\_distance} - 2 \times \text{optimal\_distance}}\right) \times 33 \qquad (4.5)$$

In this context, the value of max_allowed_distance is defined as four times the optimal distance. Any penalty computed above 33% is clipped to this maximum value.

Time Penalty (P_t): This component penalizes navigation attempts that exceed a duration of 40 seconds. The longer the robot takes beyond this threshold, the higher the penalty—up to a maximum of 33%. The calculation is formalized in Equation (4.6):

$$P_t = \max\left(0,\ \frac{\text{actual\_time} - 40}{40}\right) \times 33 \tag{4.6}$$

As with the distance penalty, if the calculated time penalty surpasses 33%, it is capped at that value.

Collision Penalty (P_c): This penalty is applied according to the number of collisions incurred during the navigation task. A fixed penalty of 11% is applied per collision, with the total capped at three collisions—resulting in a maximum penalty of 33%. The formula used to determine this value is provided in Equation (4.7):

$$P_c = \text{collision\_count} \times 11 \tag{4.7}$$

If the computed penalty exceeds 33%, it is limited accordingly. It is important to note that a score of 0% is automatically assigned in any of the following conditions: if the robot fails to reach the goal, if the distance traveled exceeds four times the optimal path length, or if more than three collisions occur.

Tables 4.4 and 4.5 summarize the calculated navigation scores for both the extended and non-extended models. The extended model consistently yielded higher scores, reflecting its superior ability to navigate efficiently, with minimal collisions and better adaptability to the environment.

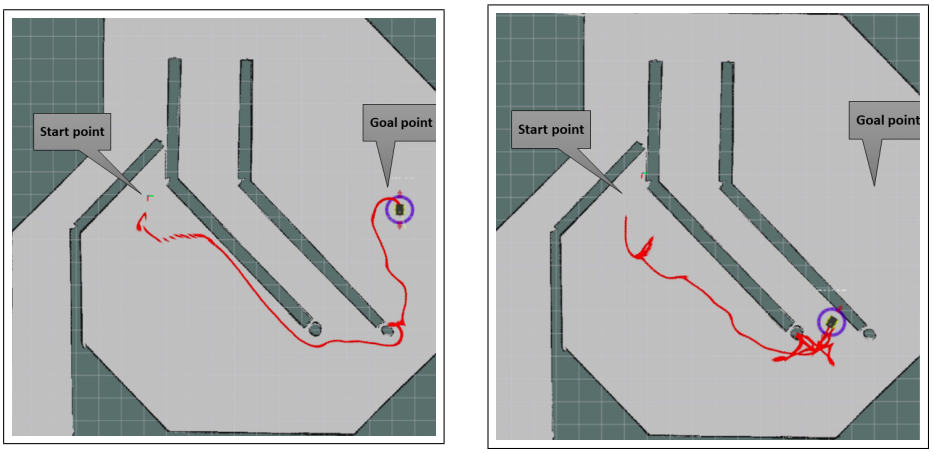
Table 4.4 Navigation Performance Assessment Metric Scores for the Extended Model.

Path	**Metric (%)**	**Description**
Path 1	100.00	No collisions; completed within maximum allowed distance
Path 2	57.60	One collision; exceeded optimal path length
Path 3	100.00	No collisions; followed optimal path
Path 4	0.00	Four collisions; surpassed maximum distance allowed
Path 5	100.00	No collisions; completed within optimal range
Path 6	100.00	No collisions; optimal path followed
Path 7	67.00	Three collisions; slight distance exceedance
Path 8	0.00	Navigation failed; exceeded maximum allowed distance
Path 9	70.00	One collision; slight performance penalty
Path 10	98.33	No collisions; minor deviation in path distance

Table 4.5 Navigation Performance Assessment Metric Scores for the Non-Extended Model.

Path	**Metric (%)**	**Description**
Path 1	0.00	Four collisions; maximum distance exceeded
Path 2	0.00	Timeout; did not complete within allowed limits
Path 3	0.00	Four collisions; navigation failed
Path 4	0.00	Three collisions; exceeded distance threshold
Path 5	85.77	No collisions; slight penalties for time and distance
Path 6	100.00	No collisions; followed optimal trajectory
Path 7	10.00	Three collisions; resulted in major penalty
Path 8	0.00	Maximum distance exceeded; navigation failed
Path 9	0.00	Timeout; exceeded allowed distance
Path 10	0.00	One collision; distance exceeded threshold

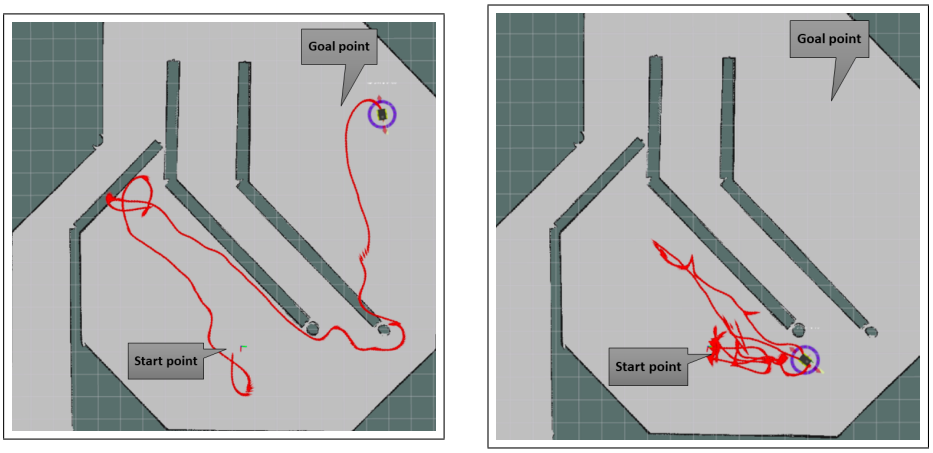
These quantitative results are further supported by the qualitative visualizations shown in Figures 4.2, 4.2a, 4.2b, 4.3, 4.3a, and 4.3b. These figures compare the navigation paths taken by both models on Paths 1 and 2 using RViz. They clearly illustrate the extended model's advantage—displaying smoother trajectories, fewer deviations, and better collision avoidance—while the non-extended model demonstrates significant navigation inefficiencies and higher collision rates.



(a) Extended Model

(b) Non-Extended Model

Fig. 4.2 Comparison of paths taken by both models on Path 1, as described in Tables 4.4 and 4.5. Trajectories are shown in red. The extended model (left) demonstrates efficient navigation with no collisions. The non-extended model (right) exhibits significant deviations and fails to reach the goal due to excessive collisions. Visualizations are for analysis only and were not used by the MoveBase or DRL system for navigation.



(a) Extended Model

(b) Non-Extended Model

Fig. 4.3 Path 2 navigation comparison using RViz. The extended model (left) shows smooth, collision-free navigation with successful goal completion. The non-extended model (right) deviates significantly, fails to reach the goal, and experiences a timeout. Apparent overlaps with walls in RViz are due to rendering artifacts and do not represent real collisions. Maps shown are for visualization only and were not used during navigation.

4.3 Experiment 2: Evaluation of Extended Model with and without Global Path Planning

Building upon the same extended model described in the previous experiment (Section 4.2.5), this second experiment evaluates the model's performance when integrated with MoveBase, as opposed to using only the extended model on its own. Key performance indicators—including execution time, distance traveled, collisions, and an overall navigation score—are measured. Both approaches are tested in the same environment shown in Figure 4.1, maintaining consistency with the earlier experimental setup.

In this experiment, MoveBase was used without a map, as the objective was to evaluate navigation without prior environment knowledge. The global planner was based on the default MoveBase configuration using the NavfnROS planner, together with the standard local planner. All planner and costmap parameters were kept at their default values, and no manual tuning was performed.

The scoring approach remains based on the 0%–100% metric established in prior work [36], which reflects both efficiency and safety by considering the distance covered, time taken, and the number of collisions. Below are the revised scoring parameters for the current experiment.

Updated Scoring Parameters for the Current Experiment

- **Success Definition:** A trial is deemed successful if the robot reaches the goal while remaining within six times the optimal distance, with no more than three collisions, and in under 100 seconds.
- **Distance Penalty Threshold:** Travel distances beyond three times the optimal path incur a distance penalty.
- **Time Penalty Threshold:** Any navigation lasting longer than 50 seconds incurs a time-based penalty.
- **Collision Penalty:** Each collision deducts 11% from the final score, up to three collisions (33% total).
- **Failure Conditions:** The run is considered unsuccessful if it exceeds six times the optimal distance, experiences more than three collisions, or goes beyond 100 seconds.

Tables and figures in the upcoming sections highlight the results under these updated guidelines.

4.3.1 Navigation Performance in the Environment from Figure 4.1

This subsection contrasts how the same extended model, with and without MoveBase, navigates the environment depicted in Figure 4.1. The most recent outcomes are summarized in Tables 4.6, 4.7, and 4.8.

Results Without MoveBase

As presented in Table 4.6, the extended model, operating independently of MoveBase, achieves a 100% navigation score on the majority of paths. The one exception is

Path 2, scoring 87.99% due to a longer-than-optimal trajectory. Path 4 ends with 0% because it struggles to navigate around a wall situated directly behind the goal. Averaging across all runs, the navigation metric is about 88.799%, reflecting shorter distances and quicker travel times overall.

On Path 4, the robot repeatedly attempts to circumvent the wall but surpasses the 59.09-meter threshold without colliding, thus failing to reach the goal. This behavior arises because the extended model was never trained with an obstacle arrangement completely blocking the goal location. Consequently, the model fails to correctly interpret whether to route on the left or the right side of the wall.

From these data, one can infer that in relatively simple settings, the extended model alone performs effectively. However, in more intricate layouts (e.g., continuous walls or maze-like structures), its capacity to chart a path without a global planner can be constrained.

Table 4.6 Navigation Results Without MoveBase.

Path	**Time (s)**	**Distance (m)**	**Collisions**	**Metric (%)**	**Reason for Score**
1	11.168	19.7273	0	100.00	Clear path, no collisions.
2	32.313	49.4413	0	87.99	Longer route than optimal.
3	12.501	15.4900	0	100.00	Short, direct route.
4	21.520	30.3458	4	0.00	Path estimation error behind a wall.
5	7.521	13.8702	0	100.00	Optimal route with no collisions.
6	4.349	8.3963	0	100.00	Short, efficient path.
7	8.596	15.5965	0	100.00	Clear navigation path.
8	8.524	16.6024	0	100.00	Smooth route without obstacles.
9	9.607	15.9270	0	100.00	Direct and efficient route.
10	24.174	30.0191	0	100.00	Longer route but successful navigation.

Results With MoveBase

Table 4.7 reports navigation outcomes for the extended model enhanced by MoveBase. Except for Path 2—where the score is 76.87% due to a more meandering path—and Path 4—ending in 0% from the same behind-the-wall issue—the robot achieves a 100% metric on all other paths. The average navigation score stands at roughly 87.687%. The inclusion of global waypoints leads to marginally longer distances and times in certain instances. Sample performance illustrations can be seen in [35].

Table 4.7 Navigation Results with MoveBase.

Path	Time (s)	Distance (m)	Collisions	Metric (%)	Reason for Score
1	10.804	19.0100	0	100.00	Clear, efficient route.
2	37.610	61.6545	0	76.87	Longer route due to waypoint adjustments.
3	8.886	14.6418	0	100.00	Short and optimal path.
4	50.478	59.1000	0	0.00	Failed path estimation behind a wall.
5	8.831	15.1413	0	100.00	Efficient route without collisions.
6	4.339	8.3672	0	100.00	Short, smooth path.
7	12.364	17.5683	0	100.00	Clear route with no interference.
8	8.478	16.4355	0	100.00	Efficient and obstacle-free route.
9	7.023	13.2094	0	100.00	Straight and clear path.
10	15.656	23.4948	0	100.00	Longer but successful route.

Summary and Comparison

Table 4.8 provides a concise comparison of these two modes. Running only the extended model (i.e., without MoveBase) generally reduces travel times and total

distances, although collisions do occur in isolated cases. On the other hand, integrating MoveBase eliminates collisions almost entirely, but tends to lengthen routes and operating time.

Table 4.8 Summary Comparison of Navigation Metrics (Mean $\pm$ SD).

Metric	**Without MoveBase (Mean $\pm$ SD)**	**With MoveBase (Mean $\pm$ SD)**	**Analysis**
Average Time (s)	14.03 ± 8.94	16.45 ± 15.16	Slightly longer with MoveBase due to waypoint adjustments.
Average Distance (m)	21.54 ± 11.97	24.86 ± 19.13	Longer with MoveBase due to global path adjustments.
Average Collisions	0.40 ± 1.26	0.00 ± 0.00	No collisions with MoveBase.
Average Navigation Metric (%)	88.80 ± 31.25	87.69 ± 20.41	Slightly higher without MoveBase due to direct paths.

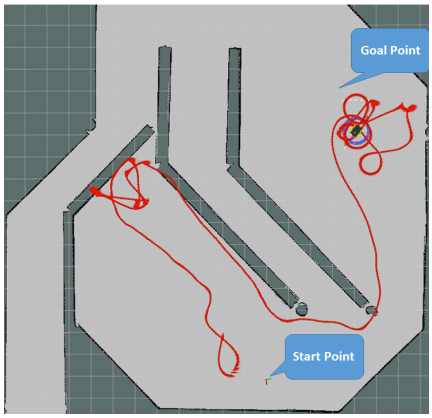
4.3.2 Selected Paths for Detailed Analysis

In order to better understand the navigation dynamics, Paths 2, 4, and 5 were examined using RViz visualization. Figures 4.4, 4.5, and 4.6 illustrate performance with and without MoveBase.

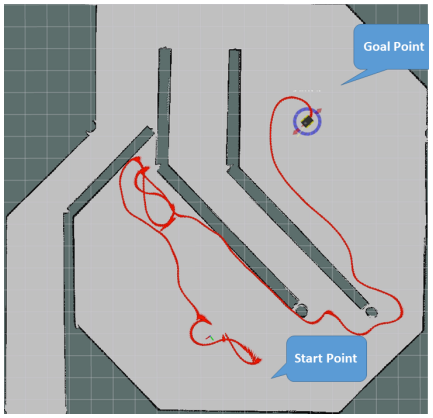
4.3.3 Paths 2, 4, and 5: In-Depth Comparison

Focusing on the same three paths, Figures 4.4, 4.5, and 4.6 show that both approaches succeed on Path 2, though MoveBase yields a 76.87% score due to a roundabout route. Without MoveBase, the robot obtains 87.99%, reflecting a more direct course.

Regarding Path 4, neither setup succeeds, as the continuous wall behind the goal was not part of the model's training data. MoveBase causes the robot to traverse a significant distance before reaching the limit, while the standalone extended model collides several times over a shorter distance, still failing to reach the goal.



(a) With MoveBase - Path 2. The robot reached the goal but traveled a longer route due to waypoint adjustments from the global planner.



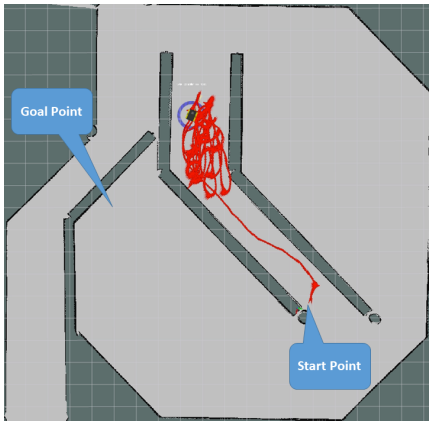
(b) Without MoveBase - Path 2. The robot reached the goal with a shorter, more direct path, indicating efficient navigation without waypoint adjustments.

Fig. 4.4 Comparison of Path 2 trajectories with (a) and without (b) MoveBase. With MoveBase, the robot followed a longer path due to waypoint adjustments. Without MoveBase, it navigated efficiently using a shorter route.

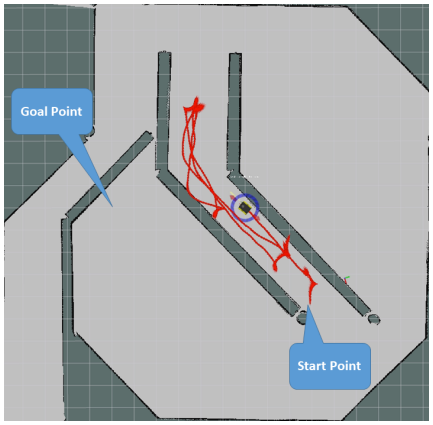
Finally, Path 5 is completed successfully under both approaches, with each run reporting minimal or no collisions. Although MoveBase introduces a slightly extended path, both runs finish with a 100% navigation score. Collectively, these observations confirm that MoveBase increases path length and duration in certain situations, whereas the extended model alone is sufficient for less complex routes.

4.4 Threats to Validity and Sim-to-Real Considerations

All experiments in this thesis were conducted entirely in ROS/Gazebo simulation, which introduces limitations for real-world transfer. The standard Clearpath Jackal model provided in the official ROS Noetic documentation was used without modification, with default physics parameters All training and evaluation environments were fully static, consisting of fixed layouts with no dynamic obstacles. LiDAR resolution and range were fixed across all experiments, no explicit sensor noise was injected, Odometry was obtained from simulation ground truth, which is noise-free and more accurate than real robot odometry. As a result, the learned policies were trained and evaluated under idealized sensing conditions and do not capture real-world noise,

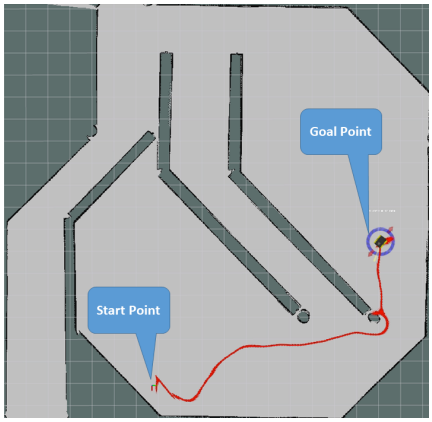


(a) With MoveBase - Path 4. The robot navigates around the wall but fails to reach the goal.

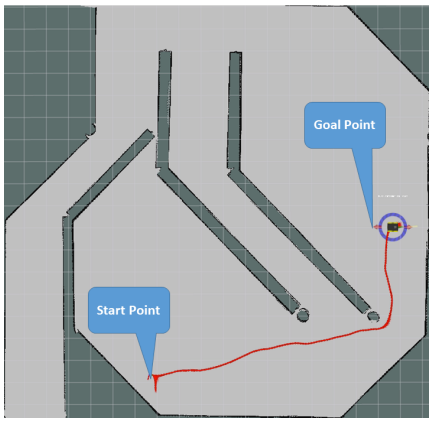


(b) Without MoveBase - Path 4. The robot gets trapped, looping without progress.

Fig. 4.5 Comparison of Path 4 trajectories with (a) and without (b) MoveBase. Both approaches fail to resolve the obstruction, demonstrating a shared limitation in navigating around long walls.



(a) With MoveBase - Path 5. The robot completes the route without collisions, following an efficient trajectory.



(b) Without MoveBase - Path 5. The robot also navigates efficiently without collisions, following a smooth route.

Fig. 4.6 Comparison of Path 5 trajectories with (a) and without (b) MoveBase. Both approaches demonstrate efficient navigation without collisions, indicating that MoveBase did not provide additional benefits for this path.

drift, or sensing imperfections, which represents a threat to validity when considering real-world transfer.

4.5 Notation and Definitions

Below is a list of the mathematical symbols and terms used throughout this chapter:

α Learning rate of the policy network.

γ Discount factor in reinforcement learning.

τ Target network soft update rate in TD3.

N Mini-batch size for training updates.

r Immediate reward at each transition.

P_d Distance-based penalty.

P_t Time-based penalty.

P_c Collision-based penalty.

R Total episode reward.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

Core Contributions and Findings:

This thesis demonstrates that training a DRL-based navigation policy with randomized start and goal positions significantly improves generalization to unseen environments compared to fixed-goal training. The extended TD3-based model showed higher adaptability and robustness in diverse and unpredictable scenarios, while the fixed-goal model showed strong performance only in narrowly constrained environments. Furthermore, integrating a global planner (MoveBase) with the DRL policy reduced collision occurrences but did not lead to improved overall navigation performance, with the purely DRL-based approach achieving superior success rates in most test environments. These results highlight the importance of training diversity over dependence on global planning for robust autonomous navigation.

This work presents a detailed study on enhancing robot navigation through DRL, aiming to enhance the robot's capability to adapt and effectively navigate through unfamiliar environments. Two closely related experiments were conducted. The second experiment aimed to evaluate the impact of applying a global planning strategy to DRL by using the same enhanced model trained in the first experiment. Both experiments relied on an improved DRL model developed within the scope of this thesis.

In the initial experiment, two distinct training methods were compared to understand which approach allowed robots to better generalize their navigation skills. One approach involved fixed start and goal points (the non-extended model), while the other utilized randomized start and goal points (the extended model). Through careful observation and analysis, a clear outcome was identified: training with diverse, randomly placed start and goal points greatly improved the robot's ability to handle new and previously unseen environments. The extended model demonstrated strong adaptability to unexpected scenarios, highlighting its robustness. In contrast, the non-extended model was particularly effective at navigating extremely narrow spaces due to its specialized and focused training; however, it struggled considerably when faced with unpredictably placed goals, even in simpler setups. These results indicate that each model possesses distinct strengths that can potentially complement one another.

To address the challenge of long training times, retraining an already trained model is suggested as a more efficient alternative to training from scratch. Although this approach was not applied in the conducted experiments, it represents a promising direction for reducing the time required to achieve high performance. The experimental results showed that the extended model achieved higher success rates in a shorter time compared to the non-extended model. However, this difference cannot be fully attributed to the extended training strategy alone, as the environments used for training the two models were substantially different. Therefore, future studies are encouraged to evaluate the extended model in narrow and constrained environments similar to those used for the non-extended model to enable a fair comparison.

As the study progressed, it became evident that evaluating the proposed model in isolation was insufficient; performance also needed to be assessed relative to existing approaches. This comparison was presented in detail in Chapter 4, specifically in Subsection 4.1.2, where several DRL-based navigation methods were analyzed in static environments. The analysis revealed clear performance differences across the evaluated methods, particularly as environment complexity increased. The method proposed in this thesis exhibited strong adaptability in more complex environments, particularly those featuring long walls and maze-like layouts, where a balance between navigation efficiency and success was achieved. These results emphasize that the design of the training environment plays a critical role in determining the generalization capability of DRL models. Comprehensive evaluation, therefore, requires testing across a wide range of environmental configurations.

In the second experiment, the effect of integrating MoveBase, a global path planning strategy, with a DRL-based navigation system was evaluated. The initial assumption was that MoveBase would enhance navigation performance by providing reliable global paths through complex environments. Although a reduction in collision events was observed—consistent with previous findings reported by R. Ali et al. [36]—no clear improvement in overall navigation performance was achieved. In most scenarios, the DRL model operating without a global planner outperformed the hybrid approach. The model demonstrated effective navigation without reliance on prior maps or global planning, highlighting the flexibility of DRL-based navigation when appropriately trained and its ability to cope with unpredictable or unfamiliar environments.

Comparison with related studies revealed both similarities and differences that help contextualize these findings. Akmandor et al. [19, 20], Roth et al. [21, 22], and Anas et al. [26, 37] all report improved navigation performance and generalization using DRL-based strategies, which is consistent with the results obtained in this thesis. Similarly, Cimurs et al. [23, 24] showed that reliance on global planning alone is insufficient for robust navigation, supporting the observed limitations of purely global planning approaches.

The proposed DRL model, operating without reliance on global planning, demonstrated strong adaptability across the conducted experiments. This adaptability can be attributed to improvements introduced in earlier work [36], where the model was trained using randomized start and goal positions. Although the model was not explicitly trained with dynamic obstacles, it exhibited an inherent ability to avoid moving objects, such as the moving Jackal robot used during evaluation. Nevertheless, a notable limitation was observed in scenarios involving long walls obstructing the direct path to the goal. In such cases, the robot struggled to consistently select the correct path around the obstacle, likely due to limited exposure to similar structures during training. This highlights a key open challenge in DRL-based navigation: improving the transfer of learned behaviors to structurally different and more complex environments.

Both experiments were conducted using simulation environments provided by ROS and Gazebo, offering controlled and reproducible testing conditions. However, simulation environments cannot fully capture the complexity of real-world conditions, including sensor noise, dynamic obstacles, and unmodeled environmental variations.

Consequently, validating the proposed DRL models through real-world experiments remains an essential step for confirming their robustness and practical applicability.

5.2 Recommendations for Future Work

Based on the findings and insights gained throughout this thesis, several important recommendations can help guide future research in DRL-based robot navigation. First, combining the strengths of both training approaches used in this work is highly recommended. The extended model offered flexibility by using randomized start and goal positions, while the non-extended model performed well in tight and narrow spaces due to its specialized training. A future model that brings these two approaches together can become more adaptable and capable of handling a wide variety of navigation tasks. Second, instead of training new models from scratch, it is recommended to retrain existing pre-trained models. This approach can save time and reduce the effort needed to reach high performance, especially when introducing new or more complex training environments. Third, future work should focus on integrating both global and local navigation strategies. Using global waypoint planning together with local DRL-based decision-making can lead to better and more reliable navigation, especially in unfamiliar or dynamic environments. It is also important to train navigation models in more challenging and diverse environments. Including complex layouts such as maze-like paths, long continuous walls, and narrow corridors will help improve the model's ability to make good decisions and adapt to unfamiliar situations. Future work should study how different sources of variability, such as the magnitude of environment randomization, the inclusion of dynamic obstacles, and changes in sensor resolution, contribute to improved generalization performance. In addition, hybrid reward systems should be considered. These systems can give the model feedback not only for reaching the goal but also for following the planned path more accurately. This can improve the model's navigation precision and overall performance. It is also suggested to include dynamic obstacles in the training process. Since the current model already showed a natural ability to handle moving objects, training with such obstacles can further improve its performance in real-world conditions. Finally, it is essential to test the trained models outside of simulation. Future research should include real-world experiments using physical robots in structured indoor areas. This type of testing

will help evaluate how the model deals with challenges like sensor noise, unexpected obstacles, and environmental changes—factors that cannot be fully simulated. In addition, a sim-to-real evaluation plan should be considered, including controlled tests on a physical robot (e.g., low speed in a single environment) and analysis of practical risks such as sensor noise models, system latency, odometry drift, and safety constraints. Any physical test would require extra safety tools—such a remote emergency stop button, and software monitors—to keep the robot safe. Following these recommendations can lead to the development of more powerful, adaptable, and reliable DRL-based navigation systems, better prepared for real-world applications.

References

[1] Y. Chen, C. Rastogi, and W. R. Norris. A cnn based vision-proprioception fusion method for robust ugv terrain classification. *IEEE Robotics Autom. Lett.*, 6(4):7965–7972, 2021.

[2] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, USA, 2 edition, 2018.

[3] Maxim Lapan. *Deep Reinforcement Learning Hands-On: Apply modern RL methods, with deep Q-networks, value iteration, policy gradients, TRPO, AlphaGo Zero and more*. Packt Publishing, 2018.

[4] Danijar Hafner, Timothy Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models. *arXiv preprint*, 2020.

[5] Saurabh Roy, Shubham Bakshi, and Tristan Maharaj. Opac: Opportunistic actor-critic. *arXiv preprint*, 2020.

[6] S. Fujimoto, H. van Hoof, and D. Meger. Addressing function approximation error in actor-critic methods. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018*, volume 80, pages 1582–1591, 2018.

[7] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. Pmlr, 2018.

[8] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint*, 2017.

[9] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint*, 2013.

[10] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2016.

[11] Romisaa Ali. Robot exploration and navigation in unseen environments using deep reinforcement learning. *International Journal of Computer and Systems Engineering*, 18(9):619–625, 2024. Open Science Index 213, 2024.

[12] Guangda Chen, Lifan Pan, Yu'an Chen, Pei Xu, Zhiqiang Wang, Peichen Wu, Jianmin Ji, and Xiaoping Chen. Robot navigation with map-based deep reinforcement learning. In *Proceedings of the 2020 International Conference on Networking, Sensing and Control (ICNSC)*, pages 1–6, October 2020.

[13] Wenhao Yu, Jie Peng, Quecheng Qiu, Hanyu Wang, Lu Zhang, and Jianmin Ji. Pathrl. *arXiv preprint*, 2023.

[14] Hamid Taheri, Seyed Rasoul Hosseini, and Mohammad Ali Nekoui. Deep reinforcement learning with enhanced ppo for safe mobile robot navigation. *arXiv preprint*, 2024.

[15] Hartmut Surmann, Christian Jestel, Robin Marchel, Franziska Musberg, Houssem Elhadj, and Mahbube Ardani. Deep reinforcement learning for real autonomous mobile robot navigation in indoor environments. *arXiv*, May 2020.

[16] Mohammad Babaeizadeh, Iuri Frosio, Stephen Tyree, Jason Clemons, and Jan Kautz. Ga3c: Gpu-based a3c for deep reinforcement learning. *arXiv preprint*, 2016.

[17] Bingxin Xue, Fengyu Zhou, Chaoqun Wang, Ming Gao, and Lei Yin. Robot mapless navigation in vuca environments via deep reinforcement learning. *IEEE Transactions on Industrial Electronics*, PP:1–11, January 2024.

[18] Iure Oliveira and Alexandre Santos Brandão. Deep reinforcement learning for mapless robot navigation systems. In *Proceedings of the Latin American Robotics Symposium (LARS), Brazilian Robotics Symposium (SBR), and Workshop on Robotics in Education (WRE)*, October 2023.

[19] Neşet Ünver Akmandor, Hongyu Li, Gary Lvov, Eric Dusel, and Taşkin Padir. Deep reinforcement learning based robot navigation in dynamic environments using occupancy values of motion primitives. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, page 9982133, Kyoto, Japan, 2022. IEEE.

[20] A. M. Akmandor, M. Kocamaz, and A. Yıldız. Tentabot navigation framework. GitHub repository, 2022. See Appendix B for repository details.

[21] Aaron M. Roth, Jing Liang, and Dinesh Manocha. Xai-n: Sensor-based robot navigation using expert policies and decision trees. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021.

[22] E. Roth, C. Paxton, and D. Gupta. Jackal crowd navigation environment. GitHub repository, 2019. See Appendix B for repository details.

[23] R. Cimurs, I. H. Suh, and J. H. Lee. Goal-driven autonomous exploration through deep reinforcement learning. *IEEE Robotics and Automation Letters*, 7(2):730–737, 2022. [Accessed: October 2024].

[24] R. Cimurs. Drl robot navigation. GitHub repository, 2024. See Appendix B for repository details.

[25] Z. Xu, B. Liu, X. Xiao, A. Nair, and P. Stone. Benchmarking reinforcement learning techniques for autonomous navigation. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 9224–9230, 2023.

[26] A. Anas and colleagues. Crowd navigation with deep reinforcement learning, 2023. See Appendix B for implementation details.

[27] Y. Tan, Y. Lin, T. Liu, and H. Min. Pl-td3: A dynamic path planning algorithm of mobile robot. In *IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 3040–3045, Prague, Czech Republic, Oct. 9-12 2022.

[28] Sylabs. Installing singularityce, 2024. See Appendix B for software environment setup and reproducibility.

[29] A. Paszke et al. Pytorch, 2019. See Appendix B for software framework.

[30] Daffan. Ros jackal navigation environment, 2021. See Appendix B for repository details.

[31] R. Ali. Extended-ros-jackal-environment, 2024. See Appendix B for repository details.

[32] F. Daffan. Ros jackal: Competition package. GitHub repository, 2023. [Online access: 30 March 2025].

[33] Clearpath Robotics. Jackal gazebo simulation, 2024. See Appendix B for simulation framework.

[34] ROS Navigation Stack. Move base package, 2020. See Appendix B for software details.

[35] Zenodo. Robot navigation using td3 with movebase integration in env2, 2025. See Appendix B for archived experimental artifacts.

[36] R. Ali, S. Dogru, L. Marques, and M. Chiaberge. Adaptive robot navigation using randomized goal selection with twin delayed deep deterministic policy gradient. *Preprints*, Jan 2025.

[37] Zerosansan. Td3, ddpg, sac, dqn mobile robot navigation. GitHub repository, 2024. See Appendix B for repository details.

Appendix A

List of Abbreviations

Table A.1 List of Abbreviations Used in the Thesis

Abbreviation	Definition
Reinforcement Learning Algorithms	
TD3	Twin Delayed Deep Deterministic Policy Gradient
PPO	Proximal Policy Optimization
SAC	Soft Actor-Critic
DDPG	Deep Deterministic Policy Gradient
DQN	Deep Q-Network
General Terms and Systems	
RL	Reinforcement Learning
DRL	Deep Reinforcement Learning
ROS	Robot Operating System
Architectures and Models	
COA	Crowds and Obstacles Avoidance with Deep Reinforcement Learning
GDAE	Goal-Driven Autonomous Exploration
Tentabot FC	Fully Connected Model in Tentabot Framework
Tentabot 1DCNN FC	1D Convolutional Neural Network with FC Layers in Tentabot
OPAC	Opportunistic Actor-Critic

Appendix B

Code, Software, and Experimental Artifacts

This appendix provides consolidated access to all code repositories, software frameworks, and experimental artifacts referenced in this thesis. The resources listed here support reproducibility, transparency, and comparison with related works discussed throughout the manuscript.

B.1 Proposed Method Implementation

The complete implementation of the proposed navigation framework, including environment design, training setup, and evaluation scripts, is available at:

- **Extended-ROS-Jackal-Environment** (GitHub):
  https://github.com/Romisaa-Ali/Extended-ROS-Jackal-Environment

This repository contains the custom simulation environments, randomized training configurations, and the integration of the DRL controller within the ROS navigation stack.

B.2 Containerized Training Environment

SingularityCE was used to containerize the training environment in order to run multiple deep reinforcement learning models in parallel while maintaining isolated and independent execution contexts.

The container setup followed the official SingularityCE documentation, accessible at:

- https://docs.sylabs.io/guides/latest/admin-guide/installation.html

B.3 Simulation and Robot Frameworks Used in This Work

The simulation and robotic framework adopted in this study is based on the ROS Jackal platform and its official Gazebo simulation tools. The following repositories constitute the core environment used and extended in this work:

- **ROS Jackal Navigation Environment (Daffan et al.)**: https://github.com/Daffan/ros_jackal
- **ROS Jackal Competition Environment (Daffan et al.)**: https://github.com/Daffan/ros_jackal/tree/competition
- **Clearpath Jackal Gazebo Simulation**: https://docs.clearpathrobotics.com/docs/ros1noetic/robots/outdoor_robots/jackal/tutorials_jackal/#simulating-jackal

These frameworks provide the mobile robot model, sensor interfaces, and simulation infrastructure that were directly used and extended to implement the proposed navigation and training environments.

B.4 External Repositories Referenced for Context and Comparison

In addition to the main simulation framework, several publicly available repositories related to deep reinforcement learning-based robot navigation were consulted for contextual understanding and comparison with recent studies:

- **DRL Robot Navigation (Cimurs et al.)**:
  https://github.com/reiniscimurs/DRL-robot-navigation
- **Tentabot Navigation Framework (Akmandor et al.)**:
  https://github.com/RIVeR-Lab/tentabot/tree/master
- **Reinforcement Learning Navigation Benchmark (Xu et al.)**:
  https://cs.gmu.edu/~xiao/Research/RLNavBenchmark/
- **Jackal Crowd Navigation Environment (Roth et al.)**:
  https://github.com/AMR-/JackalCrowdEnv
- **Mobile Robot Navigation Comparison (Zerosansan)**:
  https://github.com/zerosansan/td3_ddpg_sac_dqn_qlearning_sarsa_mobile_robot_navigation

These resources were referenced to understand common benchmarking practices, problem formulations, and architectural trends in recent navigation studies. They were not directly integrated into the experimental pipeline of this work.