UNIVERSIDADE FEDERAL DE MINAS GERAIS
Escola de Engenharia
Programa de Pós-Gradução em Engenharia Elétrica

PhD DISSERTATION Nº 483

Reward Shaping For Goal-oriented Tasks Using Deep Reinforcement Learning

Victor Ricardo Fernandes Miranda

Belo Horizonte
2025

Victor Ricardo Fernandes Miranda

REWARD SHAPING FOR GOAL-ORIENTED TASKS USING DEEP REINFORCEMENT LEARNING.

Tese apresentada ao Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Minas Gerais, como requisito parcial à obtenção do título de Doutor em Engenharia Elétrica.

Orientador: Prof. Dr. Leonardo Amaral Mozelli

Coorientador: Prof. Dr. Armando Alves Neto
Coorientador: Prof. Dr. Gustavo Medeiros Freitas

Belo Horizonte
2025

M672r

Miranda, Victor Ricardo Fernandes.
Reward shaping for goal-oriented tasks using deep reinforcement learning [recurso eletrônico] / Victor Ricardo Fernandes Miranda. - 2025.
1 recurso online (99 f. : il., color.) : pdf.

Orientador: Leonardo Amaral Mozelli.
Coorientadores: Armando Alves Neto, Gustavo Medeiros Freitas.

Tese (doutorado) Universidade Federal de Minas Gerais, Escola de Engenharia.

Inclui bibliografia.

1. Engenharia elétrica - Teses. 2. Modelagem - Teses. 3. Navegação de robôs móveis - Teses. I. Mozelli, Leonardo Amaral. II. Alves Neto, Armando. III. Freitas, Gustavo Medeiros. IV. Universidade Federal de Minas Gerais. Escola de Engenharia. V. Título.

CDU: 621.3(043)

Ficha catalográfica elaborada pelo bibliotecário Reginaldo César Vital dos Santos CRB6/ 2165
Biblioteca Profº Mário Werneck, Escola de Engenharia da UFMG

UNIVERSIDADE FEDERAL DE MINAS GERAIS

Escola de Engenharia

COLEGIADO DO CURSO DE GRADUAÇÃO / PÓS-GRADUAÇÃO EM Engenharia Elétrica

FOLHA DE APROVAÇÃO

"Reward Shaping For Goal-oriented Tasks Using Deep Reinforcement Learning"

Victor Ricardo Fernandes Miranda

Tese de Doutorado submetida à Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em Engenharia Elétrica da Escola de Engenharia da Universidade Federal de Minas Gerais, como requisito para obtenção do grau de Doutor em Engenharia Elétrica.

Aprovada em 09 de maio de 2025.

Por:

Prof. Dr. Leonardo Amaral Mozelli
DELT (UFMG) - Orientador

Prof. Dr. Armando Alves Neto
(UFMG)

Prof. Dr. Gustavo Medeiros Freitas
DEE (UFMG)

Prof. Dr. Paulo Lilles Jorge Drews Jr
Centro de Ciências Computacionais (Universidade Federal do R

Prof. Dr. Marcos Ricardo Omena de Albuquerque Maximo
Laboratório de Sistemas Computacionais Autônomos (LAB-SCA) (

Prof. Dr. Luiz Chaimowicz
DCC (UFMG)

Prof. Dr. Frederico Gualberto Ferreira Coelho
DELT (UFMG)

Documento assinado eletronicamente por **Gustavo Medeiros Freitas**, **Professor do Magistério Superior**, em 09/05/2025, às 17:31, conforme horário oficial de Brasília, com fundamento no art. 5º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Frederico Gualberto Ferreira Coelho**, **Membro**, em 09/05/2025, às 17:36, conforme horário oficial de Brasília, com fundamento no art. 5º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Leonardo Amaral Mozelli**, **Professor do Magistério Superior**, em 09/05/2025, às 18:45, conforme horário oficial de Brasília, com fundamento no art. 5º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Armando Alves Neto**, **Professor do Magistério Superior**, em 09/05/2025, às 19:00, conforme horário oficial de Brasília, com fundamento no art. 5º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Luiz Chaimowicz**, **Professor do Magistério Superior**, em 15/05/2025, às 09:36, conforme horário oficial de Brasília, com fundamento no art. 5º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Rogerio Rodrigues Lima**, **Professor do Magistério Superior**, em 30/05/2025, às 13:24, conforme horário oficial de Brasília, com fundamento no art. 5º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Paulo Lilles Jorge Drews Junior**, **Usuário Externo**, em 30/05/2025, às 15:23, conforme horário oficial de Brasília, com fundamento no art. 5º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Marcos Ricardo Omena de Albuquerque Maximo**, **Usuário Externo**, em 17/06/2025, às 07:27, conforme horário oficial de Brasília, com fundamento no art. 5º do Decreto nº 10.543, de 13 de novembro de 2020.

A autenticidade deste documento pode ser conferida no site https://sei.ufmg.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **4186137** e o código CRC **0FDAD4EC**.

Referência: Processo nº 23072.227332/2025-58 SEI nº 4186137

RESUMO

Esta tese aborda o desafio de projetar funções de recompensa eficazes para o treinamento de agentes utilizando aprendizado por reforço profundo (DRL) em tarefas orientadas a objetivos. O principal objetivo é acelerar o aprendizado, minimizar a ocorrência de ótimos locais, aumentar a eficiência e garantir que o agente se alinhe corretamente aos objetivos definidos. A pesquisa explora como as funções de recompensa podem contribuir para a melhoria da generalização das políticas, permitindo um desempenho robusto em cenários novos e não treinados. Um dos focos centrais da tese é a redução da lacuna entre simulação e realidade, facilitando a transferência de políticas aprendidas em ambientes simulados para aplicações reais complexas e dinâmicas, sem perda de desempenho ou necessidade de treinamento adicional. Como solução para o problema analisado, propõe-se uma função de recompensa com *reward shaping* baseada no princípio do Potential-Based Reward Shaping (PBRS), em que a recompensa é definida pela diferença entre funções potenciais. Demonstra-se que a inclusão do *reward shaping* proposto à função de recompensa não altera a otimalidade da política, assegurando que o agente aprenda o comportamento desejado com os benefícios da nova formulação de recompensa. A recompensa proposta, empregada no treinamento de agentes para a navegação autônoma de robôs sem mapas, apresenta desempenho superior em relação a outras funções de *reward shaping* da literatura baseadas na distância até o alvo. A comparação com funções de recompensa existentes indica uma aceleração na convergência do treinamento e um aumento no número de tarefas concluídas em um ambiente de teste após o treinamento. Além disso, o método proposto demonstrou robustez e obteve resultados superiores em relação a abordagens da literatura ao operar em ambientes distintos daqueles utilizados para o treinamento. Resultados similares foram observados na transferência *zero-shot*, tanto em *sim-to-sim* quanto em *sim-to-real*, superando métodos existentes mesmo quando transferido para robôs com arquiteturas diferentes das utilizadas no treinamento. Dessa forma, a abordagem inovadora apresentada para o uso de *reward shaping* na navegação de robôs visa aprimorar a generalização e sua aplicação em cenários desconhecidos, com resultados promissores tanto em simulação quanto em cenários reais.

Palavras-chave: Reward Shaping; Aprendizado por Reforço Profundo; Generalização do Modelo; Ambientes Desordenados Desconhecidos; Navegação de Robôs Sem Mapa.

ABSTRACT

This thesis addresses the challenge of designing effective reward functions for training agents using deep reinforcement learning (DRL) in goal-oriented tasks. The main objective is to accelerate learning, minimize the occurrence of local optima, increase efficiency, and ensure that the agent correctly aligns with the defined objectives. The study explores how reward functions contribute to improving policy generalization, enabling robust performance in new and untrained scenarios. A key focus of this research is reducing the gap between simulation and reality, facilitating the transfer of policies learned in constrained simulated environments to complex and dynamic real-world applications without performance loss or the need for additional training. As a solution to the explored problem, we propose a reward function with *reward shaping* based on the *Potential-Based Reward Shaping* (PBRS) principle, where the reward is defined by the difference between potential functions. We demonstrate that adding the proposed *reward shaping* to the reward function does not interfere with policy optimality, ensuring that the agent learns the desired behavior while benefiting from the new reward function. The proposed reward function, applied to training agents in mapless autonomous robot navigation, achieves superior performance compared to other *reward shaping* functions in the literature that rely on distance to the target. Comparisons with existing reward functions indicate accelerated training convergence and an increase in the number of tasks completed in a test environment after training. Furthermore, the proposed method demonstrates robustness and achieves superior results compared to others in the literature when operating in environments different from those used for training. Similar results were obtained in *zero-shot* transfer, both in *sim-to-sim* and *sim-to-real* scenarios, outperforming existing methods even when transferred to robots with architectures different from those used during training. Thus, this innovative approach to *reward shaping* in robot navigation enhances generalization and its application in unknown scenarios, with promising results in both simulation and real-world environments.

Keywords: Reward Shaping; Deep Reinforcement Learning; Model Generalization; Unknown Cluttered Environments; Mapless Robot Navigation.

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

A2C	Advanced Actor-Critic
A3C	Asynchronous Advanced Actor-Critic
ANN	Artificial Neural Network
CNN	Convolutional Neural Networks
DDPG	Deep Deterministic Policy Gradient
DDQN	Double Deep Q-Network
DQN	Deep Q Network
DRL	Deep Reinforcement Learning
GPU	Graphics Processing Unit
IMU	Inertial Measurement Unit
LiDAR	Light Detection And Ranging
MCP	McCulloch and Pitts Artificial Model
MDP	Markov Decision Processes
PBRS	Potential-Based Reward Shaping
PFC	Potential Field Controller
PPO	Proximal Policy Optimization
PRM	Probabilistic Roadmaps
ReLU	Rectified Linear Unit
RL	Reinforcement Learning
ROS	Robot Operating System
SAC	Soft Actor-Critic
SLAM	Simultaneos Localization and Mapping
TD	Temporal Difference
TD3	Twin Delayed Deep Deterministic Policy Gradient
UAV	Unmanned Aerial Vehicle
UGV	Unmanned Ground Vehicle
USV	Unmanned Surface Vehicle
YOLO	You Only Look Once

CONTENTS

CHAPTER

1

INTRODUCTION

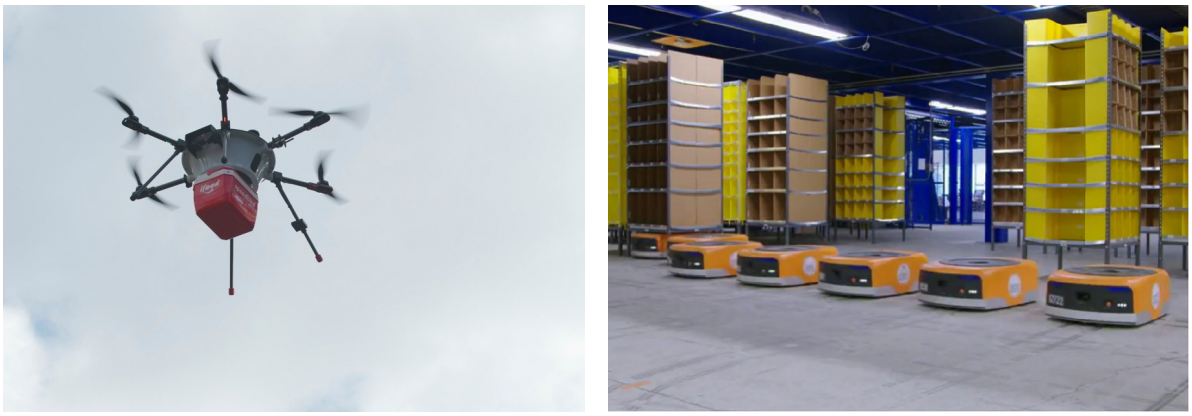
In the last decades, robotic platforms have been adopted in several industry sectors (such as load transportation, farming, and environment exploration, among others), in both civilian and military scenarios. The interest in such technologies is commonly motivated by the search for higher efficiency in production, cost reduction, and safety.

For example, companies in the mining and energy industry have adopted mobile robots in their activities, seeking to increase safety. Figure 1 presents a legged and a wheeled mobile robot used for inspection tasks in such companies. Similarly, boosted by the growth of e-commerce, some companies are using mobile robots for inventory control in warehouses and goods transportation, as shown in Figure 2.

(a) Spot - Boston Dynamics (b) EspeleoRobô - VALE

Figure 1 – UGV for autonomous inspection tasks.
Source (a): bostondynamics.com
Source (b): ITV.org

Autonomous navigation is a complex and interdisciplinary challenge in the mobile robots field. These tasks represent a union of sub-problems in the robotics areas, such as localization, mapping, planning, guidance, and control, aiming to move a robot safely to a new location. A



(a) Delivery Drone Ifood Concept - SpeedBird (b) Warehouse Control Robots - Amazon

Figure 2 – Mobile Robots used for e-commerce.
Source (a): economia.ig.com.br
Source (b): roboticsandautomationnews.com

common approach to the autonomous navigation problem consists of obtaining exteroceptive data of the environment to locate and generate a map using Simultaneos Localization and Mapping (SLAM) techniques, planning a path from the current position to a target point, avoiding obstacles, and guiding the robot by controlling its movements until reaching the final destination.

Although these methods perform well, those involving planning and SLAM techniques can impose a considerable computational cost and may underperform in dynamic environments. Other classical navigation strategies, such as potential fields, can be used as an alternative that does not rely on pre-planned paths. Instead, they define actions to guide navigation toward the target while avoiding locally detected obstacles [Choset et al., 2005; Machado et al., 2023]. However, these strategies may encounter issues, such as getting trapped in local minima, mainly in complex scenarios.

Deep Reinforcement Learning (DRL), a subfield of machine learning, has gained significant traction over the past decade, demonstrating its potential across various fields. The ability of DRL agents to learn complex behaviors through interactions with their environment, trial and error, offers versatile solutions for autonomous systems, particularly in robotics [Ibarz et al., 2021]. Robotic systems must make real-time decisions in dynamic and often unpredictable environments, rendering DRL an excellent candidate for autonomous navigation tasks in unmapped environments.

Figure 3 illustrates the DRL design, where the *environment* represents the physical world in which the agent operates. The *observed states* are the set of information collected from the environment and the agent's current situation. *Actions* refer to the ways the agent interacts with and modifies the environment, while the *reward* provides feedback on the executed action and its impact in the environment. Through this feedback, the agent refines its understanding of which actions are most beneficial in each state, updating its policy or value model to improve future decision-making. Moreover, in DRL, rewards can be delayed, meaning that an agent

may need to perform a series of actions before receiving feedback on their effectiveness. This characteristic promotes long-term planning, improved decision-making in sequential tasks, and learning from interaction with the environment, rather than relying on the labeled data. Compared to other machine learning techniques, DRL offers greater flexibility, as it can handle dynamic environments and stands out in situations where the optimal actions and outcomes are not immediately clear—such as in robotic navigation or game-playing tasks. By learning to maximize cumulative rewards through experience, the agent becomes well-equipped to solve complex, goal-oriented problems.

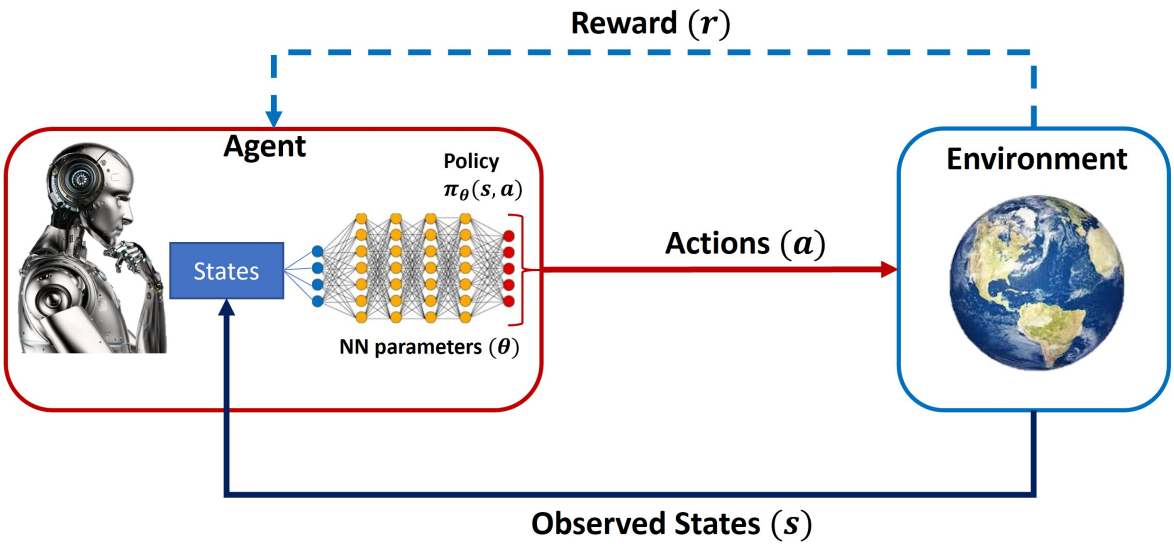


Figure 3 – Design for a Reinforcement Learning system.

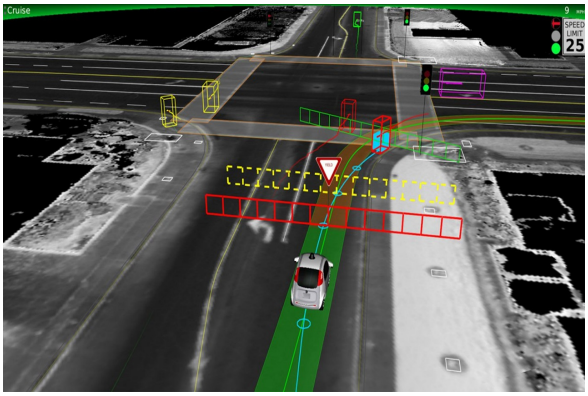
Companies in the chemical and energy industries are using DRL to optimize process control in refineries, adjusting operational parameters to maximize yield. They also have adopted autonomous mobile robots trained using DRL to navigate their facilities and collect necessary data, as presented in Figure 4. Transportation and logistics companies use DRL to optimize delivery routes, taking into account variables like traffic, weather conditions, and fuel costs. Similarly, boosted by the growth of e-commerce, some companies are using autonomous Unmanned Aerial Vehicle (UAV) controlled through DRL to optimize delivery routes and avoid obstacles in real-time, as shown in Figure 5(b). Other companies like Waymo apply DRL to improve autonomous driving, helping their vehicles make real-time decisions to avoid collisions, follow optimized routes, and handle unexpected road scenarios (Figure 5(a)).

A major advantage of DRL is that training can be conducted in simulated environments, which are much safer, faster, and cost-effective compared to real-world training. Simulation allows agents to explore various scenarios, learn optimal strategies, and fine-tune their policies without the risks involved in real-world trials. However, as compelling as these advantages are, transferring these learned policies from simulated to real-world environments remains a significant obstacle. This transfer, commonly referred to as *sim-to-real transfer*, often struggles

(a) ANYmal X - ANYbotics

(b) EXR - EXRobotics

Figure 4 – Autonomous Robots trained using DRL for inspection tasks.
Source (a): Anybotics.com
Source (b): shell.com



(a) Self-driving system - Waymo

(b) Prime Air UAV - Amazon

Figure 5 – Navigation system developed using DRL.
Source (a): dabrownstein.com
Source (b): ktla.com

due to the *reality gap*—discrepancies between the simulated environment and the real world [Boeing and Bräunl, 2012; Wada et al., 2022].

Replicating the real environment in simulation with high fidelity to minimize the *reality gap* is always a challenge, especially in robotics. Therefore, a common strategy for transferring a policy from the simulated (or training) environment to the real-world environment, and bridging the *reality gap*, involves conducting a small amount of additional training in the real environment to adapt to specific dynamics or sensory variations in a fine-tuning process [Chebotar et al., 2019]. However, as already mentioned, access to real platforms for training is not always possible and may result in higher costs. In this context, zero-shot policy transfer is a promising approach, where models are trained exclusively in a simulated environment and deployed directly to the real world without requiring additional interaction, extra training, or fine-tuning [Scarponi et al., 2024]. Nevertheless, the performance of the models transferred immediately from simulation to real-world applications is directly connected with the generalization of the policy to respond to untrained situations. Additionally, it is quite challenging to train *model-free* policies in growing

spaces, increasing the generalization with a large set of possible environment configurations.

Some solutions have been proposed in the literature to enhance generalization capacity, such as data augmentation [Shorten and Khoshgoftaar, 2019], domain randomization [Tobin et al., 2017], and regularization techniques [Wang et al., 2020]. However, these strategies alone require extensive data variation and numerous training steps to achieve significant performance. Moreover, the behavior encoded in a simulation may underperform or even fail when deployed in the real world. A better approach is to combine these methods with a reward function that incorporates knowledge of the environment and the task to be performed while simultaneously increasing the agent's exploration of actions during training and advancing states toward the goal.

The reward function is associated with the environment, and for Markov Decision Processes (MDP), it is typically a function that maps the current state, current action, and future state to a real value. This reward function can be either deterministic or stochastic. In a deterministic setting, for a given state, action, or future state, the function always returns the same value. In contrast, in a stochastic setting, the function's output is a random variable, meaning it can vary even for the same state-action pair. Since the agent's decisions are based on reward feedback, the definition of the reward is a crucial step in formulating a DRL training strategy. We can refer to a reward function as sparse or dense, depending on how it is defined. A sparse reward function provides little to no feedback on the agent's immediate actions. In most domains, the reward is typically set to zero, with non-zero rewards being granted only for a limited set of states, such as when a task or sub-task is completed [Dong et al., 2022]. This limitation makes the learning process more difficult, forcing it to explore the environment extensively before receiving any significant reward. On the other hand, a dense reward refers to a reward function that provides non-zero rewards frequently and consistently throughout the agent's interaction with the environment. In environments with dense rewards, the agent receives feedback after nearly every action or step, which helps it learn more quickly by allowing it to directly associate its actions with their outcomes.

Defining a dense reward function that represents the desired behavior with some flexibility to account for unmapped situations is essential for increasing generalization. Therefore, the reward must be well-defined according to the learning objective in the environment, and *reward shaping* methods can assist in improving the reward by including terms to accelerate training and enhance performance [Ng et al., 1999]. However, the definition of the shaping, as well as the reward function as a whole, must be carefully designed to avoid local optima.

Therefore, this dissertation proposes a novel *reward shaping* function to increase exploration, accelerate learning, and minimize the risks of local optima in mapless autonomous robot navigation applications. Within the DRL framework, we introduce a new state representation, continuous action space, and *reward shaping* function to improve sim-to-sim and sim-to-real transfer capability, both demonstrated through experiments. The proposed strategy enhances

action exploration during training, avoiding local minima in cluttered environments—i.e., actions that cause the robot to remain stuck or move around the same location—and ensures policy optimality. This approach endows the robot with capabilities for untrained scenarios, such as moving in obstacle-free directions and performing progressive movements (rather than remaining stationary). We achieve improved generalization by balancing behaviors that increase information in a local map with those that reduce the distance to the goal.

1.1 Problem Definition

The problem addressed in this dissertation is the challenge of designing an effective reward function for training DRL agents to bridge the reality gap between simulation and reality, enhance generalization, and prevent convergence to local optima in goal-oriented tasks, such as autonomous robot navigation.

The challenges become evident when focusing on the problem of mapless autonomous navigation, which is a goal-oriented task where a robot must move from an initial location to a target (goal) using only local navigation (i.e., without a prior map or path planning). The robot must avoid obstacles, localize itself, and make movement decisions based on progress in the environment and raw Light Detection And Ranging (LiDAR) data (Figure 6). Defining an effective reward function for such a high-complexity environment is particularly difficult, as it can lead to poor performance when transferring policies from the training environment to the real world.

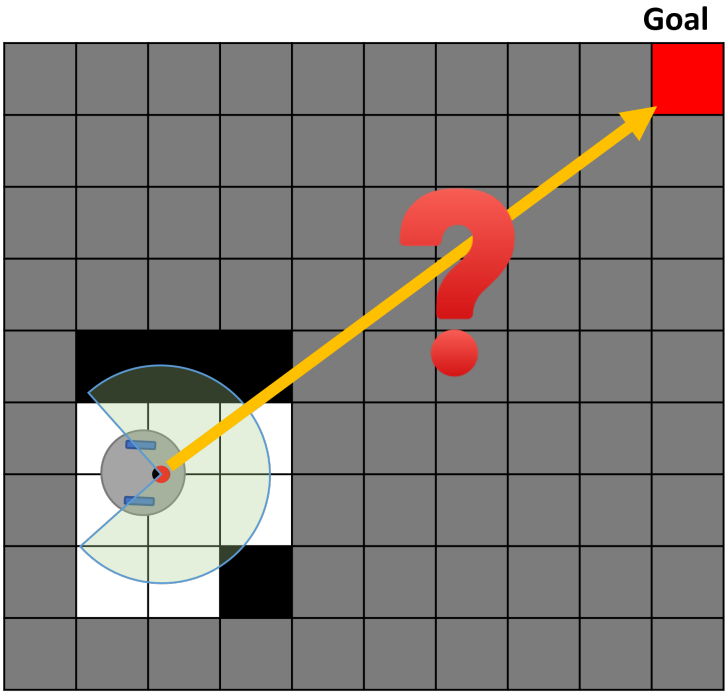


Figure 6 – Mapless autonomous navigation problem illustration.

Several studies in the literature on robot navigation using DRL employ simple and sparse reward structures for these applications. However, as demonstrated in this dissertation, such approaches are often ineffective and can lead the robot to become stuck in local minima, especially when deployed in untrained scenarios and unfamiliar environments. Moreover, even when techniques like reward shaping are applied, improper implementation can still result in convergence to local optima. To address these challenges, this dissertation introduces a novel reward shaping function within a DRL framework, aimed at enhancing exploration, accelerating learning, and improving generalization for autonomous robot navigation in both simulated and real-world environments.

1.2 Objectives

The main objective of this research is to propose a novel *reward shaping* function within the framework of DRL to increase exploration, accelerate learning, and minimize the risks of local optima in goal-oriented tasks, such as mapless autonomous robot navigation. The specific goals are:

1 - Propose a novel *reward shaping* function that incorporates exteroceptive information from the environment to enhance the agent's learning. This function aims to improve the agent's robustness to local optima, fostering exploration and accelerating the learning process.

2 - Perform a comparative analysis between the model trained with the proposed *reward shaping* function and those based on existing reward shaping strategies found in the literature. This comparison will highlight the improvements made by the novel approach in terms of agent performance and learning efficiency.

3 - Explore the impact of various training algorithms within the DRL framework to determine their effect on learning performance when combined with the proposed *reward shaping* function. The goal is to identify the most effective training strategies.

4 - Explore the combination of the proposed *reward shaping* function with other generalization techniques to enhance the agent's ability to generalize its policy to unseen environments and perform well in scenarios not included in the training process.

5 - Design and implement a series of sim-to-sim and sim-to-real experiments to demonstrate the performance and robustness of the proposed approach. These experiments provide evidence of the effectiveness of the *reward shaping* function in enhancing the zero-shot transfer method.

By achieving these objectives, this dissertation aims to contribute to the field by addressing the challenges of local minima in goal-oriented tasks and improving the transferability of policies trained in simulated environments to real-world applications.

1.3 Contributions

Compared to the existing literature, this research presents the following main contributions, highlighting the innovative approaches and advancements developed to tackle the challenges in autonomous robot navigation using DRL:

- a novel reward shaping to incorporate exteroceptive information (newer perception information from the environment map) and to improve the agent's robustness to the local minima of the environment;
- a comparative analysis among the model trained by the proposed reward shaping and others from the literature;
- a comparative analysis using different training algorithms;
- a set of *sim-to-sim* and *sim-to-real* experiments to illustrate the generalization of our proposed approach and enhancement of the zero-shot model transfer.

1.4 Text Structure

The document is organized as follows:

- **Chapter 2** provides a comprehensive review of the current state-of-the-art in DRL, focusing on reward shaping methods, generalization strategies for zero-shot transfer, and DRL applications for robot navigation.
- **Chapter 3** presents preliminary concepts, addressing the state-of-the-art in DRL, which are essential for understanding the content
- **Chapter 4** provides a detailed presentation of the proposed reward shaping, including the information gain-based reward, its integration with the learning algorithm, the role of domain randomization, and how all these elements combine to improve generalization.
- **Chapter 6** presents the experimental setup and results, including both sim-to-sim and sim-to-real zero-shot transfer performance, along with a comparison to other methods in the literature.
- **Chapter 7** discusses the results and conclusions, suggests future directions, and concludes the dissertation.

CHAPTER

2

LITERATURE REVIEW

This chapter presents a literature review of recent and significant papers related to Deep Learning and Reinforcement Learning, emphasizing reward shaping, model generalization, and sim-to-real transfer strategies, applied to robot navigation.

Reward shaping is a critical strategy in DRL that involves augmenting the reward signal to guide the agent toward desirable behaviors, improving the efficiency of policy learning. In robotic applications, reward shaping becomes essential for guiding agents toward desired behaviors, especially when handling complex tasks that may lead to suboptimal local behaviors. In the context of sim-to-real transfer, reward shaping plays an even more vital role as it aids the agent in generalizing learned behaviors across tasks with minimal or no additional training.

2.1 Reward Shaping

Reward shaping is commonly used to accelerate policy convergence by providing intermediate rewards that guide the agent toward optimal behavior, overcoming challenges due to sparse reward environments. Reward shaping can reduce training time and enhance generalization to novel environments or tasks without re-learning, essential for zero-shot transfer [Ng et al., 1999].

The capacity of reward shaping methods to address the scalability problems of Reinforcement Learning (RL) led researchers to further investigate the method, evaluating the influence of the function type and defined parameters [Grzes and Kudenko, 2008, 2009]. Even though reward shaping is generally effective, it comes with notable challenges. Creating well-designed reward functions can be complicated and may be vulnerable to exploitation by the agent. Therefore, Harutyunyan et al. [2015] addresses the efficacy of designing optimal potential functions that provide smooth guidance and transform available domain knowledge into a usable form. The authors introduce a framework to capture the desired shaping rewards based on arbitrary reward

functions while maintaining policy invariance. This method is evaluated through experiments on grid-world tasks and a cart-pole benchmark, demonstrating that the dynamic advice approach outperforms other methods that fail to maintain optimality under potential-based guidance. The authors conclude that this approach effectively integrates behavioral knowledge and enhances learning efficiency with minimal added complexity. However, only results from simulations were presented.

The significance of reward shaping continues to drive the research today. For example, Gupta et al. [2022] examines the role of reward shaping in RL, highlighting its importance for improving learning efficiency, especially for practical applications. The authors introduce the "UCBVI-Shaped" algorithm that integrates shaped rewards into standard RL methods, which leads to enhanced sample complexity by reducing the effective state space that the agent must explore. They provide a theoretical analysis showing how these shaped rewards can optimize exploration strategies and validate their findings through simulated experiments.

Hu et al. [2020b] presents a novel approach called BiPaRS (bi-level optimization of parameterized reward shaping) for adaptively utilizing shaping rewards in RL. The proposed method uses a bi-level optimization, where the lower level focuses on maximizing the policy using shaping rewards, while the upper level optimizes a shaping weight function to enhance true reward accumulation. The authors demonstrate through simulated experiments in cartpole and MuJoCo [1] environments that their approach can identify and exploit beneficial shaping rewards.

A concern in the reward shaping method is the risk of designing very complex reward functions, which may lead to suboptimal behaviors, as the agent may exploit unintended shortcuts in the environment. Moreover, manual reward shaping requires considerable domain knowledge, limiting its scalability. In this context, the authors in [Trott et al., 2019] discuss a common issue in RL where naive distance-to-goal reward shaping can lead agents to local optima. To address this, they propose a method called Sibling Rivalry, which enhances the traditional approach by introducing auxiliary distance-based rewards derived from sibling rollouts—pairs of trajectories initiated from the same state and goal. This method encourages exploration while simultaneously preventing policies from getting trapped in local optima, effectively balancing exploration and exploitation. It facilitates robust learning in goal-oriented tasks without requiring complex reward engineering or domain expertise, ultimately converging back to the sparse objective as the agent learns to achieve its goals. This is an alternative approach to the one proposed in this dissertation. However, it has a higher level of complexity and only simulated results.

Badnava et al. [2023] introduce a new approach that modifies the Potential-Based Reward Shaping (PBRS) function based on cumulative rewards from learning episodes, thereby reinforcing the reward signal, particularly in environments characterized by sparse rewards. This method proposes a reward function that adapts based on the agent's performance across episodes. The algorithm assesses both the maximum and minimum episode rewards obtained to shape the

[1] https://mujoco.org/

current reward signal. The potential function updates after each learning episode, allowing the RL agent to adjust its learning based on past performance. This modification encourages the agent to explore more when stuck in sub-optimal states by providing negative feedback for inaction and positive reinforcement for improvement. The proposed method was tested in the Arcade Learning Environment, demonstrating improvements in both single-task and multi-task scenarios. The simulated results indicate significant enhancements in learning performance compared to baseline methods.

In the case of multi-task environments, Zou et al. [2021] propose a meta-learning approach to automatically learn reward shaping for a distribution of tasks. The authors introduce a theoretical framework and a new algorithm based on Model-Agnostic Meta-Learning (MAML) to create a potential function that can be adapted to new tasks. The methodology involves meta-training and meta-testing phases, where the system learns an initial set of parameters for reward shaping that can be quickly adapted to new tasks, enhancing the agent's learning process across similar tasks. The framework's efficiency is validated through simulated experiments, showing it outperforms existing methods, including hand-designed shaping functions. However, meta-learning requires a larger amount of data for training to achieve effective model generalization and to reduce overfitting. As a result, it tends to be computationally intensive and highly sensitive to hyperparameter tuning.

Okudo and Yamada [2023] discusses advancements in reward shaping through the use of human knowledge via subgoals. Subgoal-based reward shaping is introduced as a technique that enriches environmental rewards with additional rewards from achieving subgoals, thereby accelerating the learning process. The authors validate their approach through simple simulated experiments, achieving superior performance compared to baseline algorithms and other reward shaping methods. This method requires a deeper understanding of the environment.

Recently, the study of reward shaping has also been intensified by applying RL in complex robotic systems. The authors in Shahid et al. [2022] and Onori et al. [2024] used the reward shaping strategy to position a grip's manipulator around a cube, grasp it, and lift it off, which can be considered a complex task concerning real-world robots. The first method used two dense reward functions switching according to the task phase, and the second included an adaptive parameter tuning for those rewards. The authors highlight that reward shaping involves iteratively tuning the parameters of the reward function based on observed performance during interactions with the learning environment.

2.2 Policy Transfer

In RL, training is usually performed in simulated environments, and transferring a trained agent from simulation to real-world applications is a critical challenge, particularly in robotics [Zhao et al., 2020]. Therefore, having a generalized model is essential for successfully

bridging the *sim-to-real* gap. To be effective, the agent must be capable of performing successful actions not only in situations where it has been trained but also in untrained contexts, avoiding the phenomenon of overfitting [Packer et al., 2018]. In this context, several studies discuss the capability of models to act in untrained situations and address methods to improve their efficiency [Kirk et al., 2021].

Some papers present forms of quantifying, measuring, and characterizing the generalization in DRL methods [Cobbe et al., 2019; Witty et al., 2021], and others focus on developing strategies to increase the generalization capability of the policies by modifying the learning algorithms. For instance, reward shaping can significantly enhance model generalization in reinforcement learning by providing more structured and informative reward signals during training. It guides the agent toward exploring behaviors that are generalizable across environments, reducing the chances of overfitting to specific solutions. Therefore, combining reward shaping with other model generalization strategies, as proposed in this dissertation, can significantly enhance sim-to-real transfer.

Some works address the generalization problem in DRL, proposing different solutions such as: increasing the similarity between training and execution environments with data augmentation [Shorten and Khoshgoftaar, 2019], domain randomization [Tobin et al., 2017; Wada et al., 2022], or generating different environments along training; and handling differences between environments with traditional regularization techniques [Wang et al., 2020].

Hansen and Wang [2021] presents a novel method called Soft Data Augmentation (SODA) to enhance the generalization ability of RL methods, especially when using visual observations. The method decouples data augmentation from policy learning by utilizing non-augmented data for training the RL policies while performing representation learning on augmented data to maximize the mutual information between these data types. The method was tested on tasks from the DeepMind Control suite and a robotic manipulation task. The simulated results showed significant improvements in sample efficiency, generalization, and stability compared to state-of-the-art vision-based RL methods.

The model generalization is also linked to the agent's exploration capability, i.e., varying the actions during learning to find useful behavior. Exploration is generally challenging in RL, but it is important for improving generalization and facilitating sim-to-real transfer. In Jiang et al. [2024b], the authors discuss the critical role of exploration strategies in enhancing generalization for RL and show that these strategies can help an agent gather more information about its environment, thereby improving its ability to generalize to new tasks or environments. The authors present Exploratory via Distributional Ensemble (EDE), a new method that leverages uncertainty-driven exploration to effectively gather knowledge around states with high epistemic uncertainty. Their experiments demonstrate the importance of exploration and show that EDE outperforms other methods. Ladosz et al. [2022] also presents other techniques to enhance exploration during training.

In RL applied to robotics, training through simulations allows for control over the environment, faster computation, and exploration of complex dynamics without risking damage to physical robots. The process of transferring learned policies from simulation to real-world applications typically requires fine-tuning, especially for tasks with complex dynamics such as friction and acceleration. In this context, Chebotar et al. [2019] presents a method for enhancing the sim-to-real transfer of policies by mixing simulated and real-world data for training. After some simulation training, they conduct a few real-world executions and use data from these executions to adapt the distributions of the simulation parameters based on observed discrepancies between real-world performance and simulated outcomes. The process is iterative, similar to a *incremental transfer*, with the authors repeating the cycle of simulation training, collecting real-world trials, and updating the simulation parameters multiple times. However, the method still requires some training using a real environment.

Access to real platforms for training is not always possible, and reducing the need for fine-tuning may be necessary. In zero-shot policy transfer, the agent can generalize and transfer knowledge from the learning phase (simulated) to act in a new environment (real world) without requiring any additional interaction, extra training, or fine-tuning [Scarponi et al., 2024]. The authors in Peng et al. [2018] present a method for transferring robotic control policies from simulation to real-world applications by utilizing dynamics randomization during training in a lower-fidelity simulation. They demonstrate that policies can adapt to varied real-world dynamics without requiring further training on physical systems. The main idea is to introduce random variations in the parameters of the training environment, exposing the agent to a wide variety of different scenarios and conditions during training. The effectiveness of this approach is validated through an object-pushing task using a robotic arm. The sim-to-real transfer was achieved with minimal calibration, resulting in performance similar to that observed in the simulations. Similarly, Baar et al. [2019] and Vacaro et al. [2019] propose varying the dynamics parameters through domain randomization during training in the simulated environment, allowing the agent to learn robust policies and facilitating the transfer to real-world applications.

The zero-shot policy transfer method is widely addressed for ground robot navigation. Tai et al. [2017] presents a mapless navigation policy trained in a simulated environment and applied to a real-world situation. However, the authors did not apply generalization strategies and used a simple distance-based reward, resulting in some performance issues in the real-world scenario. In a different approach, Xu et al. [2018] presents a combination of RL and a disturbance observer-based (DOB) robust tracking controller to mitigate the impact of discrepancies between training and deployment environments and improve zero-shot transfer. However, the zero-shot performance was evaluated only in a sim-to-sim transfer.

To improve sim-to-real transfer in robotics, incorporating generalization strategies and reward shaping is essential. Generalization techniques, like domain randomization and meta-learning, prepare agents to handle diverse real-world scenarios by exposing them to varied

training conditions. Meanwhile, reward shaping refines the learning process by adjusting reward structures to better align with real-world objectives, leading to more effective and adaptable policies. Together, these methods enhance the agent's ability to perform reliably when transitioning from simulation to real-world environments.

2.3 Deep Reinforcement Learning in Robot Navigation

As presented in the previous Sections, the versatility of neural networks, combined with the reinforcement learning strategy that relies on metrics and heuristics as rewards for actions taken, has attracted current research to adopt DRL techniques for complex problems such as robot navigation.

There are many strategies to safely (and/or efficiently) navigate a robot toward a target point in the environment, ranging from reactive to deliberative hierarchies. Classical planning methods generally provide collision-free, shortest distance or time-efficient paths [Chi et al., 2022; Rezende et al., 2020; Yang et al., 2019; Zhang et al., 2020], but they often require prior knowledge of the map and the obstacles' location. Alternative approaches were proposed to meet the requirement of navigating environments with obstacles. One such approach is to use bug algorithms that provide actions to bypass obstacles obstructing the path to the destination [S et al., 2021]. Another method uses potential field-based computation to derive robot velocities for collision-free paths [Sfeir et al., 2011]. However, these strategies may experience local minimal issues when operating in complex environments due to the nullification of antagonistic forces.

The utilization of DRL algorithms as the fundamental control [Carlucho et al., 2018; Jiang et al., 2024a], navigation [Kayakoku et al., 2021], localization [Gao et al., 2020], and planning [You et al., 2019] systems has gained notoriety recently. The adaptability of Artificial Neural Networks, combined with the ability to simulate and train, as well as employ them as end-to-end solutions, has caught the attention of the research community, compelling them to integrate these advancements into the realm of Robotics. Autonomous robot navigation is an area of interest for DRL surveys, which examine the effectiveness of policies trained through RL and the different algorithms specifically designed for this purpose [Jiang et al., 2020; Zhu and Zhang, 2021].

In the context of robot localization, Bohez et al. [2017] presents a sensor fusion algorithm using DRL to combine data from multiple sensors, incorporating a DropPath regularization method to mitigate issues caused by sensor failures. The method combines information from sensors embedded in the robot and others distributed throughout the environment, which communicate wirelessly with the localization system.

Considering path planning problems, the authors in Gao et al. [2020] use a deep reinforcement learning technique Twin Delayed Deep Deterministic Policy Gradient (TD3) together with the traditional Probabilistic Roadmaps (PRM) algorithm to plan and navigate in 3D environ-

ments. Similarly, You et al. [2019] presents two DRL methods for autonomous vehicles path planning problems. The first strategy learns by using a specific reward function for autonomous driving, while the second learns through the inverse reinforcement learning method, estimating the unknown reward function based on expert driver demonstration data.

Similar to the application presented in this dissertation, some papers explore the use of DRL to control a robot's actions within an environment, enabling autonomous navigation from a starting location to a goal. For instance, Ruan et al. [2019] used discrete actions with a Double Deep Q-Network (DDQN) algorithm for navigating an environment using only camera images, avoiding obstacles. Though effective in some situations, discrete actions can hinder performance in complex cluttered environments, and their reward function neglects local minimum problems. Similarly, Chen et al. [2021] also used a DDQN algorithm with discrete actions for navigation, relying on distance-based rewards for goal achievement and collision avoidance, but the method requires an occupancy grid map as input, which increases computational costs at runtime. Liu et al. [2020] proposed a navigation policy that uses discrete actions with the Asynchronous Advanced Actor-Critic (A3C) algorithm, allowing for global agent training from parallel agents. This approach facilitates generalization across a variety of maps and navigation scenarios.

Considering continuous actions, Grando et al. [2021] proposed a method for training a policy that controls a hybrid UAV to navigate towards a target point while avoiding obstacles using the Deep Deterministic Policy Gradient (DDPG) algorithm. The model employs the robot's states, LiDAR measurements, and distance to the target point, to select the best control actions for the UAV. Despite the qualitative results, the chosen reward function is too sparse and may result in poor performance in complex scenarios. In Wu et al. [2021], raw depth images captured at four steps are fed to a double Soft Actor-Critic (SAC) architecture, where a primary network trains the navigation policy and a secondary network deals with obstacle avoidance.

Similarly, Zhu and Hayashibe [2022] adopts a hierarchical approach, where a low-level policy is responsible for navigation towards the target, and a high-level policy ensures safety. This differs from the method proposed in this dissertation, in which only one policy is used to learn all tasks. Hierarchical networks generally help to improve learning efficiency and stability when the agent is facing complex tasks, but there are disadvantages when compared with single networks, such as an increase in the complexity of the algorithms, demand for more training, higher risk of overfitting, and more difficulty of hyperparameters tuning.

Zieliński and Markowska-Kaczmar [2021] discusses a DRL model developed for vision-based navigation of Autonomous Underwater Vehicles (AUVs). The authors focus on a practical implementation where the goal is to navigate the AUV from a starting point to a target object using visual and sensor data. The control model utilizes the Advanced Actor-Critic (A2C) strategy, which has been enhanced with Proximal Policy Optimization (PPO) for improved training stability. A vision module is also trained using a Convolutional Neural Networks (CNN) or a pre-trained Tiny You Only Look Once (YOLO) network [Redmon et al., 2016] to serve as

input to the control model. The authors propose and analyze several reward components, such as position, velocity, and angular velocity rewards, shaping the function to guide the agent in learning appropriate behavior for efficiently navigating toward a target.

In the map exploration context, Cimurs et al. [2021] and Hu et al. [2020a] present strategies to explore the space using a trained policy navigation system. The first one employs the TD3 algorithm and offers a reward function that observes only arrivals, collisions, and nonprogressive movement conditions. For the second one, the DDPG was used for training, and a reward that includes a safety clearance term, avoiding obstacles, and moving in the target direction was designed. Since the robot must navigate long distances, local minima issues are frequently seen in map exploration tasks. Additionally, policies trained with distance-based rewards are also subject to local optima. The authors of Cimurs et al. [2021] propose auxiliary algorithms to help the target destination avoid local minimum and increase the method generalization, and Hu et al. [2020a] uses a safety clearance reward to prevent a high attraction of the robot to the target. However, the algorithms used for training do not present high action exploration during agent learning, and the proposed reward function does not deal with this issue directly.

A different strategy for map exploration is adopted by Li et al. [2019] which uses DRL to define exploration goal points in an unknown environment, considering the robot's position and the occupancy grid map obtained from a SLAM algorithm as input data. Similarly, Niroui et al. [2019] presents an algorithm to define frontier points to be explored by analyzing the distance to the robot and map information gain. They use a DRL technique to select the goal point based on the robot position, occupancy grid map, and the candidate frontier points, with the last obtained from a clustering algorithm. Still focusing on exploration but considering multiple robots, Luo et al. [2019] propose a learning method based on graphs to divide the environment into regions and use these graphs as input to a CNN that defines an exploration goal point for each robot. However, all these methods require a grid map as input, which may lead to higher computational costs and increased training time.

Most DRL approaches for robot navigation focus on goal-oriented tasks to define the reward function for training. In their work, Jia et al. [2022] introduces a goal-oriented navigation network to control a robot through continuous actions of linear velocities in the direction of a defined goal position. The authors utilize the TD3 algorithm to train the model, implementing a straightforward sparse reward function that accounts for goal arrival, collision detection, and a component that promotes linear movement. In contrast to the method proposed in this dissertation, sparse rewards can lead to local optima and may prevent the robot from achieving the desired behavior in more complex environments. Similarly, Olayemi et al. [2023] propose a DRL method for goal-oriented navigation of a ground robot that focuses on avoiding static obstacles positioned in front of the robot. Aligning with the approach proposed in this dissertation, the authors reduce the state space by segmenting the LiDAR readings, thereby decreasing computational complexity. However, they employ a simple distance-based reward function, which may lead to local optima,

as discussed further in this manuscript.

Majid et al. [2024] present a DRL method for robot navigation to a defined goal position. In this approach, the authors allow the robot to move backward as an alternative means of avoiding dynamic obstacles. The reward components aim to reduce the distance to the goal, maintain the robot's orientation toward the goal, encourage forward movements, limit angular velocity, and avoid obstacles. Although a dense reward is proposed, the adopted terms are not designed to avoid local minima.

Alipanah and Moosavian [2022] propose an improved algorithm for mapless navigation using DRL with reward shaping. The authors' strategy focuses on creating a composite reward function for the mobile robot in mapless navigation, consisting of three key components: a goal-reaching reward; a collision penalty that imposes substantial negative rewards for close encounters with obstacles and a smaller penalty for being within double the collision threshold; and a performance reward that evaluates the robot's progress by rewarding it for reducing the distance to the goal and penalizing it for moving away, with additional measures to discourage excessive turning movements. This structured reward function effectively guides the robot to navigate safely and efficiently while minimizing unnecessary maneuvers. Although simulated experiments showed good results, this reward approach may lead to local minima in more complex environments, as discussed in this dissertation.

Addressing the same mapless navigation problem for mobile robots, the authors in [Xu et al., 2024] introduce the Potential Risk State Augmentation (PRSA) method, which aims to enhance generalization and decision-making under uncertainty by improving the processing of high-dimensional LiDAR data. The authors propose an adaptive safety optimization reward shaping method to mitigate issues associated with sparse rewards while promoting faster learning and effective obstacle avoidance. The adaptive reward shaping balances the achievement of navigation goals with obstacle avoidance based on collision risk. The components of the reward function are: a distance reward that incentivizes the agent to minimize the distance to the target; an obstacle reward that provides negative rewards when the agent approaches obstacles; and a Goal-oriented Adaptive Reward that encourages the agent to adapt its behavioral strategies in response to varying risk levels. However, the main issue with goal-based rewards is the risk of reaching local optima instead of the objective.

Also related to robot navigation, but employing strategies that differ slightly from those proposed in this dissertation, Walker et al. [2019] utilize DRL to address a layered problem by training two policies for UAV navigation. Their approach incorporates one policy for global planning and another for local planning. The global planner operates in a discrete environment defined by cells representing rooms within a larger space, using a sparse reward system to encourage the selection of unexplored cells. In contrast, the local planner is designed for a goal-oriented policy that directs the UAV to navigate toward targets, with rewards structured to avoid collisions and promote exploration of unexplored areas. The transition between the cells is

managed by another model responsible for aligning the UAV in the direction of the next cell and moving it in that direction. The simulated results presented were conducted in simple scenarios, and the proposed sparse reward may affect performance in more complex situations. In a similar approach, Kaufmann et al. [2023] presented a vision-based DRL navigation strategy for a drone racing application. In this approach, an observation policy was used for localization and a second control policy transformed the observation policy outputs into commands for the drone. The reward used is as simple as the task itself, where the policy maximizes progress toward the next point (the center of the racing gates), with a perception objective included to keep the next goal in the camera's field of view. In this environment, local optimality issues are not expected, and for this reason, the authors don't address this topic.

Other researchers have been influenced by the work presented in this dissertation to introduce methods for robot navigation, including Fu and Yao [2024], Wang et al. [2025], Montero et al. [2025], Li et al. [2025], Wu et al. [2024], Lou et al. [2024], Xu et al. [2024], and others. For instance, Lou et al. [2024] present a DRL method for collision avoidance in Unmanned Surface Vehicle (USV) that combines a collision-zone strategy with reward shaping. The reward function concerns the distance to the goal, proximity to obstacles, risk of collision, and the ability to predict the collision zone. Given that the navigation space for the USV is largely open, a distance-based reward shaping may provide adequate performance, which differs from the environment addressed in this dissertation.

Many works in the literature rely on sparse or distance-based rewards, which can lead to suboptimal convergence and slow learning. In contrast, we propose a novel PBRS-based reward shaping method that enhances learning efficiency and prevents local optima. Additionally, most papers present only simulated experiments, and those that include real-world experiments require additional training or adaptation when transferring policies. By using the proposed method, we demonstrate a successful zero-shot sim-to-real transfer without additional fine-tuning, even across different robot architectures. Finally, we integrate reward shaping with domain randomization during training and utilize a training algorithm that encourages exploration, all to improve zero-shot generalization to unknown scenarios. Most works do not explicitly focus on generalization, leading to performance degradation in untrained settings.

Table 2.1 presents a comparison of the differences between our approach and some related works mentioned in this Section.

Table 2.1 – Method comparison of related works.

Paper	Algorithm	Action Space	Reward Shaping	Navigation	Real-World Experiments	Sim-to-Real Transfer
Ruan et al. [2019]	DDQN	Discrete	Distance Based	Mapless	Only Simulation	N/A
Chen et al. [2021]	DDQN	Discrete	Distance Based	Mapped	Yes - Simple scenarios without local minima	zero-shot
Liu et al. [2020]	A3C	Discrete	Not included	Mapped	Yes - Simple scenario but includes dynamic obstacles	zero-shot
Jia et al. [2022]	TD3	Continuous	Not included	Mapless	Only Simulation	N/A
Olayemi et al. [2023]	TD3	Continuous	Distance Based and heading alignment	Mapless	Yes - Simple scenario without local minima	zero-shot
Majid et al. [2024]	TD3	Continuous	Distance Based and movement progress incentive	Mapless	Yes - closed room with obstacles	zero-shot
Xu et al. [2024]	SAC	Continuous	Distance Based and obstacle safe distance	Mapless	Yes - closed room with obstacles	zero-shot
Grando et al. [2021]	DDPG	Continuous	Not included	Mapless	Only Simulation	N/A
Cimurs et al. [2021]	TD3	Continuous	Distance Based	Mapped	Yes - only short distances using frontier exploration	zero-shot
Hu et al. [2020a]	DDPG	Continuous	Distance Based and safety clearance	Mapless	Yes - Simple scenarios in a multi-robot context	zero-shot
Proposed Method	**SAC**	**Continuous**	**Information from unexplored areas and distance**	**Mapless**	**Yes - Cluttered scenarios with local minima and different robot architectures**	**zero-shot**

CHAPTER

3

BACKGROUND AND THEORETICAL FOUNDATIONS

Machine learning is an evolving branch of computational algorithms designed to work like human intelligence, solving problems based on previous examples or learning from the surrounding environment [El Naqa and Murphy, 2015].

This chapter reviews key concepts in machine learning, artificial neural networks, deep learning, and reinforcement learning, essential for understanding the methodologies and approaches discussed in this dissertation.

3.1 Reinforcement Learning

Reinforcement Learning (RL) is a machine learning process focused on decision-making by an artificial agent based on observations of the environment, enabling it to perform actions to complete a specific task. Unlike supervised and unsupervised techniques, in RL, the agent learns to map situations to actions by observing only some parts of the environment and without the correct action information to compare. Therefore, the learner is not told which actions to take but must instead discover which actions yield the most reward by trying them out [Barto and Sutton, 1995; Sutton, 2018]. Figure 7 represents a basic design of RL, where an *environment* is a physical world in which the agent operates, *observation* is the information collected from the environment and/or current situation of the agent, and *reward* is feedback for the action executed.

RL is usually viewed as a general formalization of decision-making tasks and is deeply connected to dynamic programming and optimal control [Ivanov and D'yakonov, 2019; Sewak, 2019; Sutton, 2018]. Therefore, RL problems are closely related to optimal control problems or stochastic optimal control problems, which can be solved by dynamic programming or formulated as a MDP [Bellman, 1957b].

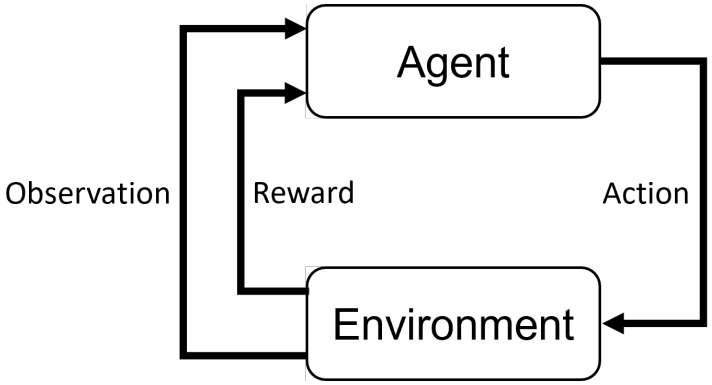


Figure 7 – Design for a Reinforcement Learning system.

3.1.1 Markov Decision Processes (MDP)

A Markov Decision Processes (MDP) is a mathematical framework that formalizes a sequential decision-making model, where actions influence not just immediate rewards, but also subsequent situations, or states, and through those future rewards. MDP is fundamental in reinforcement learning as it provides a formalism to describe the interaction between an agent and its environment over time [Sewak, 2019; Sutton, 2018].

As illustrated in Figure 7, in an MDP, the *agent* continually interacts with the *environment* through *actions*, and the *environment* responds by presenting new *states* for the agent's observation and a *reward* for the action taken. In this process, the main objective of the agent is to maximize the *reward* over time by choosing appropriate *actions*.

An MDP can be defined as a set of five tuples comprising of $\mathscr{M} = (S, A, T, R, \gamma)$, where:

- **States (S)**: The set of all possible states the agent can be in. Each state $s \in S$ represents a unique environment configuration.
- **Actions (A)**: The set of all possible actions the agent can take. In each state s, the agent selects an action $a \in A(s)$.
- **Transition Function (T)**: This is the probability of transitioning from one state to another, given an action. It is often denoted as $T = p(s'|s,a) \rightarrow [0,1]$, representing the probability of ending up in state s' after taking action a in state s.
- **Reward Function (R)**: The reward function assigns a scalar value $r = R(s,a,s')$ to each state-action transition, determining the immediate reward of moving from state s to state s' by taking action a.
- **Discount Factor (γ)**: A value $\gamma \in [0,1]$ that represents the degree to which future rewards are considered.

In MDP and RL, the agent's goal is to maximize the cumulative reward, also known as the expected *return*, it receives in the long run. Therefore, an agent learns an optimal strategy,

called *policy* $\pi(a|s)$, to decide which actions to take in each state by interacting with a defined environment [Sewak, 2019]. The return G_t is defined as the cumulative sum of rewards received over time:

$$G_t = R_{t+1} + R_{t+2} + R_{t+3} + \cdots = \sum_{k=0}^{\infty} R_{t+k+1}. \tag{3.1}$$

If there exists a policy where the agent receives a constant positive reward at every time step, i.e. $R_t = c \; \forall \; t$ where $c > 0$, then the return can diverge to infinity:

$$G_t = c + c + c + \cdots = \sum_{k=0}^{\infty} c \rightarrow \infty. \tag{3.2}$$

Therefore, the return is typically expressed as the sum of discounted rewards. The discount factor γ ensures that future rewards are weighted less, preventing infinite returns and making the value function converge:

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}, \; 0 \leq \gamma < 1, \tag{3.3}$$

where t is the time step, R_{t+1} is the reward received after taking action at time step t, and R_{t+k+1} is the reward received k steps into the future from time step t.

3.1.2 Policies and Value Functions

In general, the policy π is a mapping from states to a probability distribution over actions $\pi : S \rightarrow p(A = a|S)$. The policy $\pi(A|S)$ is the probability that $A_t = a$ if $S_t = s$ [Sutton, 2018].

The *state-value function* of a state s under a policy π, denoted $v_\pi(s)$, is the expected return when starting in the state s and following the policy π:

$$V_\pi(s) = \mathbb{E}_\pi \left[G_t | S_t = s \right] = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s \right], \forall s \in S, \tag{3.4}$$

where $\mathbb{E}_\pi$ denotes the expected value under the policy π, $\gamma \in [0, 1]$ is the discount factor, and t is any time step.

A fundamental property of value functions in RL is that they satisfy recursive relationships. The recursive form of $V_\pi(s)$ is called Bellman Equation [Bellman, 1957a,b] and is defined as follows:

$$\begin{aligned} V_\pi(s) &= \mathbb{E}_\pi \left[R_{t+1} + \gamma V_\pi(s') \right], \\ &= \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a) \left[r + \gamma V_\pi(s') \right], \forall s \in S. \end{aligned} \tag{3.5}$$

Similarly, the *action-value function* of taking an action a in the state s under a policy π is the expected return when starting in the state s, taking the action a and following the policy π:

$$Q_\pi(s, a) = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s, A_t = a \right]. \tag{3.6}$$

The goal of RL is to find an optimal policy π^* that achieves the maximum expected return from all states [Arulkumaran et al., 2017]:

$$\pi^* = \underset{\pi}{\operatorname{argmax}}\, \mathbb{E}[R|\pi]. \tag{3.7}$$

A policy is considered better than another policy, $\pi \geq \pi'$, if the value function of the first policy is greater than or equal to the value function of the second policy, $V_\pi(s) \geq V_{\pi'}(s)$, for all $s \in S$. That is, a policy is considered optimal, π^*, if its *state-value function* is also optimal. The optimal state-value function, $V^*(s)$, can be defined as follows:

$$V^*(s) = \max_\pi V_\pi(s), \forall s \in S. \tag{3.8}$$

Optimal policies also share the same optimal *action-value function*, defined as:

$$\begin{aligned} Q^*(s,a) &= \max_\pi Q_\pi(s,a), \forall s \in S \text{ and } a \in A, \\ &= \mathbb{E}\left[R_{t+1} + \gamma V^*(s) | S_t = s, A_t = a\right]. \end{aligned} \tag{3.9}$$

Following the *Bellman optimality* principle, the value of a state under an optimal policy must equal the expected return for the best action from that state [Sutton, 2018]

$$\begin{aligned} V^*(s) &= \max_{a \in A} Q^*_\pi(s,a), \forall s \in S, \\ &= \max_a \sum_{s',r} p(s',r|s,a)\left[r + \gamma V^*_\pi(s')\right], \\ &= \max_a \sum_{s',r} p(s',r|s,a)\left[r + \gamma \max_{a'} Q^*_\pi(s',a')\right]. \end{aligned} \tag{3.10}$$

Finally, the optimal policy $\pi^*(s)$ can be defined as:

$$\begin{aligned} \pi^*(s) &= \underset{a \in A}{\operatorname{argmax}}\, Q^*(s,a), \\ \pi^*(s) &= \underset{a \in A}{\operatorname{argmax}} \left[R(s,a) + \gamma \sum_{s'} p(s' \mid s,a) V^*(s')\right]. \end{aligned} \tag{3.11}$$

3.1.3 Temporal Difference (TD) Learning

Temporal Difference (TD) learning as presented in Sutton [1988], combines ideas from both dynamic programming and Monte Carlo methods to estimate value functions. Similar to Monte Carlo methods, TD methods can learn directly from raw experience without requiring a model of the environment's dynamics. Like dynamic programming (DP), TD methods update their estimates based partially on other learned estimates, rather than waiting for the final result.

A key benefit of TD methods is their suitability for online learning and continuous reinforcement learning tasks. In TD methods, the policy is updated at each step, without needing to wait until the episode concludes, different from Monte Carlo methods.

TD methods use the difference between the predicted reward and the updated prediction of the future reward, or *TD error*, to adjust the value function toward more accurate estimates. The TD error is defined as:

$$\delta_t = R_{t+1} + \gamma V(S_{t+1}) - V(S_t), \tag{3.12}$$

where δ_t is the TD error available at step $t+1$.

This TD error is then used to update the value function as follows:

$$V(S_t) \leftarrow V(S_t) + \alpha \delta_t, \tag{3.13}$$

where α is the learning rate, controlling the magnitude of the update at each step.

The simplest version of TD learning is the TD(0), which updates the value function using only the next immediate time step. This one-step prediction is computationally simple and is the foundation for more advanced RL methods like Q-learning [Watkins and Dayan, 1992] and SARSA [Rummery and Niranjan, 1994]. TD(0) assumes that, with enough exploration, the value function will eventually converge to an optimal solution. There are other variants of TD Learning as well, like the TD(1), and the more generalist TD(λ), which extends TD(0) to an extra step of multiple steps through a parameter λ.

TD(λ) combines information from multiple future steps, providing a more robust estimate of the value function by averaging predictions over several time steps. This allows for a smoother convergence and a better trade-off between bias and variance. The update rule for TD(λ) is:

$$V(S_t) \leftarrow V(S_t) + \alpha \sum_{k=0}^{\infty} (\lambda\gamma)^k \delta_{t+k}. \tag{3.14}$$

3.1.3.1 On-policy vs Off-policy

In RL, we have algorithms categorized as on-policy and off-policy. On-policy algorithms, such as SARSA (State-Action-Reward-State-Action) [Rummery and Niranjan, 1994; Sutton, 2018], the agent evaluates and improves the same policy that it uses to make decisions. This allows the agent to learn in a way directly influenced by its actions and experiences. SARSA updates the Q-value for a specific (s,a) pair by incorporating the immediate rewards the agent receives at each step, along with the Q-value of the subsequent state-action pair, (s',a'), where $s' = s_{t+1}$ and $a' = a_{t+1}$. This update is done after every transition from a nonterminal state as per the following equation:

$$Q(S_t,A_t) \leftarrow Q(S_t,A_t) + \alpha \left[R_{t+1} + \gamma Q(S_{t+1},A_{t+1}) - Q(S_t,A_t)\right]. \tag{3.15}$$

In contrast, off-policy algorithms, like Q-learning [Watkins and Dayan, 1992], learn from actions taken by a different policy, which can be either a random policy or a more exploratory one. This enables off-policy methods to benefit from experiences generated by other agents or previous versions of the policy, allowing for more flexible learning and the ability to improve

upon existing policies without the need for direct interaction with the environment. In Q-learning, the Q-value is updated as follows:

$$Q(S_t,A_t) \leftarrow Q(S_t,A_t) + \alpha \left[R_{t+1} + \gamma \max_a Q(S_{t+1},a) - Q(S_t,A_t) \right]. \tag{3.16}$$

Despite its efficiency in some applications, the main practical issue with the TD learning algorithms is scalability [Ivanov and D'yakonov, 2019]. In traditional TD methods, such as Q-learning, a table stores value estimates for every possible state-action set. The size of the Q-table increases exponentially following the complexity of the environment, leading to issues with memory requirements and inefficient updates. In continuous state spaces, this problem is worse, as the Q-table would need to store an infinite number of values, making it infeasible.

3.2 Exploration in Reinforcement Learning

Training a model to optimize a policy while avoiding overfitting is always challenging in Machine Learning. Overfitting occurs when the model fits the training data "too well," capturing undesirable patterns that degrade performance in test environments or real-world situations [Goodfellow et al., 2016], as shown in Figure 8. The main objective in training a model is to achieve high performance on unseen data, i.e., to develop a generalized model. Generalization refers to the model's ability to learn beyond the specifics of the training environment and perform effectively in untrained situations.

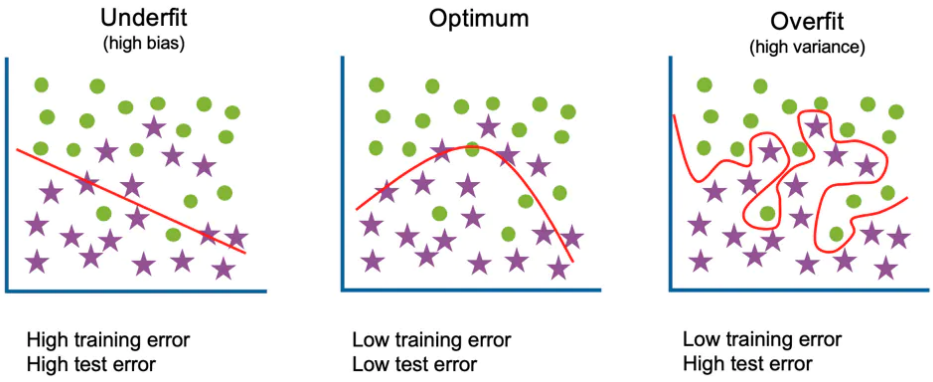


Figure 8 – Example of Underfitting, Optimal fitting and Overfitting.
Source: IBM Cloud Education - Overfitting

In RL, maximizing the reward according to the objective may seem sufficient to obtain a good model, potentially giving the impression that overfitting is part of the process. However, in certain applications, such as robot navigation, the environment can change dynamically, introducing situations not encountered during training. Therefore, RL algorithms must incorporate robustness to environmental variations and, consequently, adapt to unexpected changes in the model's input to handle unseen (but similar) situations effectively [Kirk et al., 2021].

Many solutions can be implemented to enhance model generalization during training and improve the agent's performance, including data augmentation, domain randomization, and traditional regularization techniques. Additionally, increasing the exploration capacity, i.e., varying actions during training to discover useful learning opportunities rather than focusing solely on reward maximization (exploitation) can positively impact model generalization [Ladosz et al., 2022]

The trade-off between exploration and exploitation is one of the most important aspects in reinforcement learning methods and presents a fundamental dilemma [Arulkumaran et al., 2017]. Moreover, understanding the problem and the application environment aids in defining and interpreting the impact of this balance on training [Gupta et al., 2006]. This dilemma determines whether the agent should choose a known action that yields a good reward (exploitation) or explore new actions in pursuit of a higher reward (exploration).

Many algorithms emerged in the last years seeking to improve the optimization methods to increase the generalization of the model and the reward at the same time. In terms of increasing the exploration capacity, a commonly adopted strategy consists of introducing noise to the action, forcing the model to explore, or introducing noise to the policy parameters [Plappert et al., 2017].

Off-policy algorithms, such as the Deep Q Network (DQN) (Section 3.5.1.1), use the ε-greedy strategy to add randomness to the selection of actions and encourage exploration by the agent.

3.2.1 ε-Greedy Strategy

The ε-greedy is a simple but effective method adopted by some algorithms to increase the exploration during the learning process while maximizing the reward [Sutton, 2018].

The method consists of introducing randomness into the action selection process by defining a probability $\varepsilon \in [0,1]$, which represents the likelihood of choosing a random action (exploration) versus selecting the action with the highest estimated value (exploitation). At each step, there is a probability ε that the agent will select a random action and a probability of $(1-\varepsilon)$ that it will choose the action that maximizes the estimated value (Q-function).

Additionally, it is common to dynamically adjust the ε value to improve efficiency by decreasing its magnitude over time following the agent's progress toward exploitation (maximum reward) [Arulkumaran et al., 2017]. This strategy, known as ε decay, allows the agent to shift from exploration to exploitation behavior. Different decay methods, such as linear, exponential, or adaptive can be used.

3.3 Artificial Neural Networks

The human brain's information processing capacity intrigues researchers today. Several studies emerged in an attempt to find a representative model to represent the organ's physiological structure and capable of performing tasks similar to the human brain, resulting in the Artificial Neural Networks (ANNs) models.

The initial research rush about artificial neural networks started in 1943 with the publication of McCulloch and Pitts [1943], which presents the first artificial model of a biological neuron, illustrated in Figure 9.

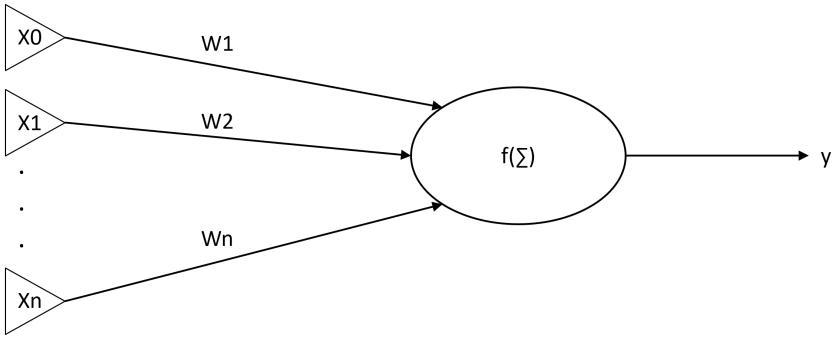


Figure 9 – Schematic of the McCulloch and Pitts Artificial Model (MCP).

The artificial neuron, illustrated in Figure 9, is formed by the model inputs $x_0, x_1, ..., x_n$, the weights $w_1, w_2, ..., w_n$ and a node that computes some mathematical function. This function maps the inputs given each respective weight to an output. The combination of multiple artificial neurons forms a neural network.

Another milestone in the ANNs research is the appearance of the Perceptron, which is a model based on the McCulloch and Pitts Artificial Model (MCP), that represents a network with multiple layered neurons presented by Rosenblatt [1958] and the back-propagation training algorithms presented by Rumelhart et al. [1986]. These discoveries enabled the creation of a training procedure that adjusts the weights of the Neural Network such that the output approximates a mathematical function.

The following equation represents the weight update during the training of a single-layer neural network:

$$w(k) = w(k-1) + \eta \left(y_d(i) - y(i)\right) x(i), \tag{3.17}$$

where $w(k)$ is the actual weights vector, $w(k-1)$ the previous weights vector, $y_d(i)$ desired output, $y(i)$ network output, $x(i)$ vector of inputs, and η is the learning rate.

A simple artificial neural network with only one layer produces a binary output and can classify only linear patterns. However, according to the Universal Approximation Theorem, networks with more than one layer can approximate any mathematical function [Cybenko, 1989]. For instance, a multi-layer neural network, as shown in Figure 10, combined with

backpropagation training algorithms to adjust the weights, can be applied to solve nonlinear classification problems.

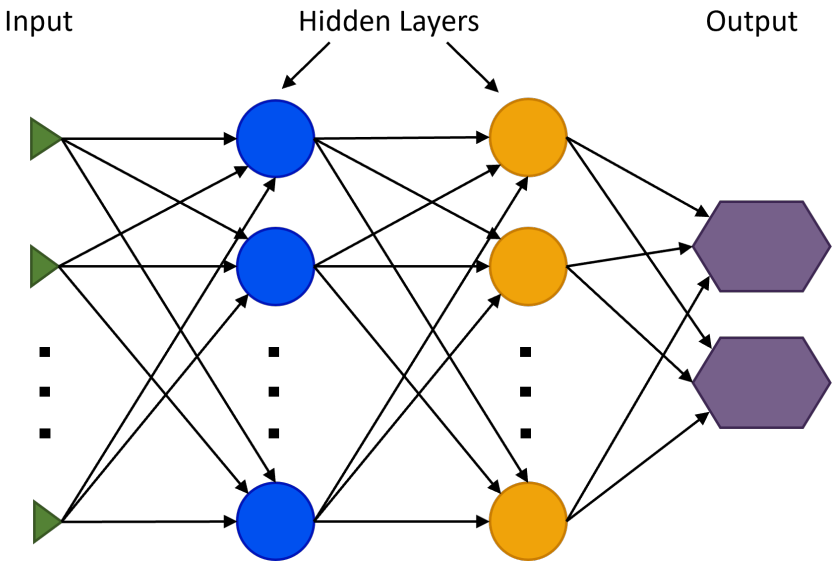


Figure 10 – Schematic of a fully-connected multi-layer Neural Network.

3.3.1 Learning Process

ANNs are widely used for several machine learning tasks, such as classification and regression of data. However, according to the needs of the target task, their application requires a learning stage to adjust the parameters of the neural network.

There are three main types of learning methods in Machine Learning. *Supervised learning* is widely used in classification and regression tasks and involves training models using labeled data, where the correct outputs are known, helping the model to learn the relationship between the input and output. The *unsupervised learning*, on the other hand, the data used for training are not labeled and the model is tasked with identifying patterns or structures within the data, such as clustering or dimensionality reduction [Berry et al., 2019; Bishop and Nasrabadi, 2006; Goodfellow et al., 2016]. Finally, *Reinforcement Learning (RL)* is a decision-making approach where an agent interacts with an environment and learns to take actions that maximize a cumulative reward, usually applied to robotics and game-playing AI systems. Each of these methods has unique applications and advantages depending on the nature of the task at hand [Sutton, 2018].

Regardless of whether the learning process is supervised or unsupervised, neural networks rely on iterative optimization techniques to update their parameters. This is typically done through gradient-based optimization, where the gradient of the loss function concerning the network's weights is computed using backpropagation [Rumelhart et al., 1986]. The objective is to minimize a predefined loss function, also called the cost function, by adjusting the parameters to reduce the error.

Backpropagation is an efficient supervised algorithm used to train neural networks by computing the gradients of the loss function concerning the network weights. The procedure consists of two main steps:

- Forward step: To calculate the output, input data is sent throughout the network in this phase. To generate its output, each neuron applies an activation function after applying a weighted sum of its inputs. Layer by layer, this process is carried out until the desired result is achieved.
- Backward step: After computing the output in the previous step, the algorithm calculates the gradient of the loss function for each weight. This is done in a reverse way, starting from the output layer and moving back to the input layer. Based on the gradient each weight is adjusted to minimize the loss.

The expression for the weight update in backpropagation is given by:

$$\Delta w_{ij} = -\eta \frac{\partial L}{\partial w_{ij}}, \tag{3.18}$$

where Δw_{ij} is the change in weight between neurons i and j, η is the learning rate, and $\frac{\partial L}{\partial w_{ij}}$ is the gradient of the loss concerning the weight w_{ij}. The learning rate η is a hyperparameter that determines the size of the step taken during weight updates in the optimization process [Murphy, 2012].

3.3.2 Activation Functions

Activation functions are crucial components of neural networks that introduce non-linearity into the model, enabling it to learn complex patterns and relationships in data. Without activation functions, a neural network will be limited to linear behavior [Anderson, 1995].

The activation function is applied to the output of each layer, including the hidden layers, of the neural network, as illustrated in Figure 11. After the weights are multiplied by the inputs and the biases are added, the activation function transforms this output by introducing non-linearity. The most common activation functions are shown in Figure 12.

3.4 Deep Learning

Following the technological evolution, new research possibilities appear in Machine Learning and artificial neural networks, approaching different applications. In a historical context, the third wave of study and research about ANNs started in 2006 with the Deep Learning present by Hinton et al. [2006].

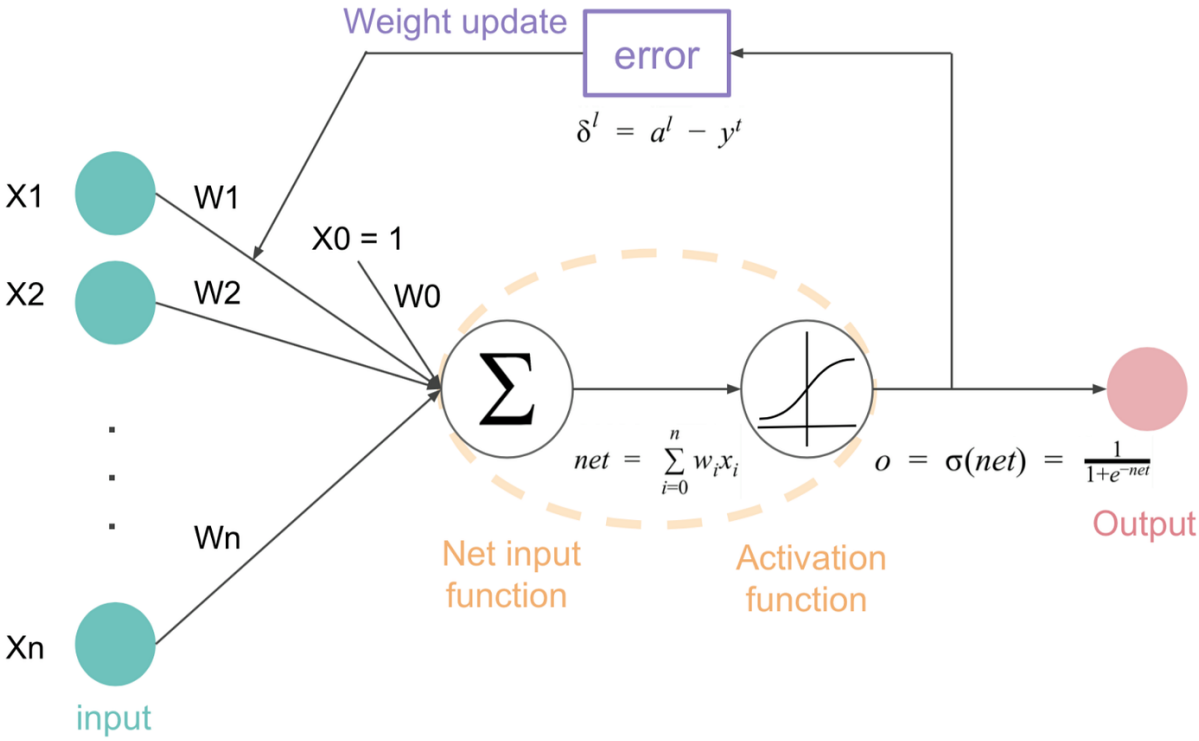


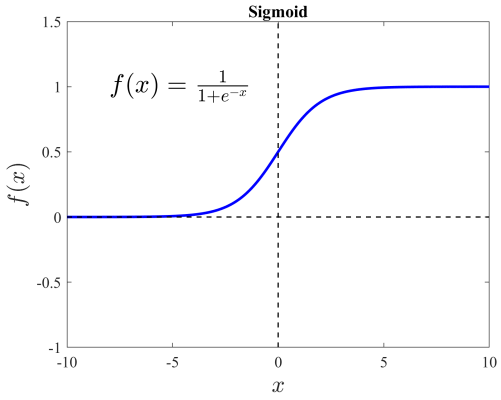
Figure 11 – Example of an ANN with an activation function.
Source: Medium - Deep Learning Activation Functions

The concept of "deep" refers to the depth of these layers, which allows the network to model intricate relationships in data that simpler models may not capture. The key principle of deep learning is to automatically learn hierarchical representations of data through layers of increasingly abstract features. In a deep neural network, each layer learns different levels of abstraction: the first layers may detect simple features like edges or colors, while deeper layers capture more complex patterns such as shapes or object parts.

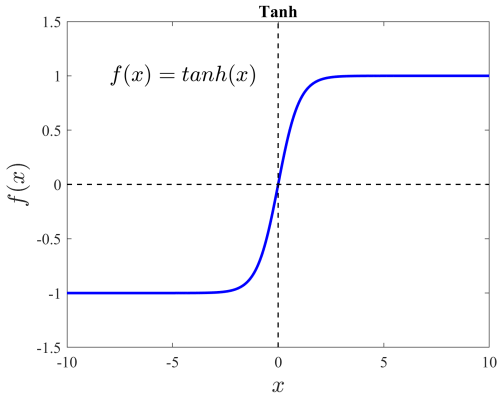
Deep learning techniques have proven to be more advanced than others due to their capacity to solve problems in image classification and speech recognition [LeCun et al., 2015]. Deep learning emerged from the possibility of using Graphics Processing Unit (GPU) for training acceleration, as well as advances in unsupervised learning, which is used in the feature extraction step.

In the beginning of 1990, several works of image recognition and object detection emerged using CNN, a new neural architecture proposed in [LeCun et al., 1998] for character recognition. A CNN is a popular deep network widely used for image interpretation problems due to its capacity for unsupervised feature extraction. Unlike traditional neural networks, CNNs are specifically designed to handle grid-like data structures such as images, where spatial relationships are crucial. This neural network uses convolution instead of general matrix multiplication in at least one of its layers [Goodfellow et al., 2016].

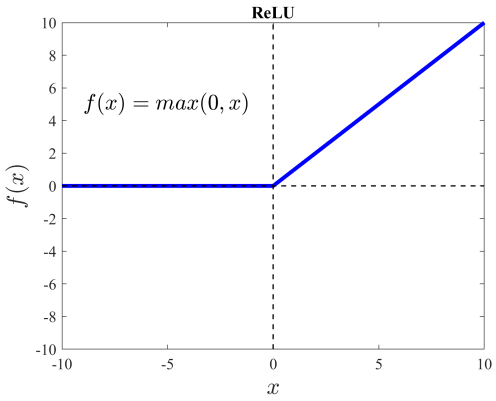
Figure 13 presents an example of a deep network to classify handwritten digits. In this example, the network architecture has two convolutional steps followed by a pooling layer, and



(a) Sigmoid function



(b) Hyperbolic Tangent function



(c) ReLU function

Figure 12 – Commonly adopted activation functions.

finally, a fully connected neural network that classifies the number in the image input.

The convolutional step detects meaningful features using kernels that occupy only tens or hundreds of pixels instead of thousands or millions of pixels of the input image. The algorithm convolves the input with a matrix filter kernel to obtain a small-size matrix containing only important pixels. The max-pooling layer stage is used to reduce the spatial volume of the image after convolution by selecting the maximum values in the matrix according to the filter parameter, as shown in Figure 14. Finally, the matrix obtained in the pooling step is flattened into a vector to be the input of the dense neural network [Goodfellow et al., 2016].

3.5 Deep Reinforcement Learning

In more complex real-world problems, traditional RL approaches become inefficient. Classical RL methods can be infeasible due to the complexity of the environment, particularly when the state space becomes large or continuous.

To address this issue, neural networks can be used as function approximators, offering a scalable alternative for high-dimensional or continuous state spaces. This integration of RL

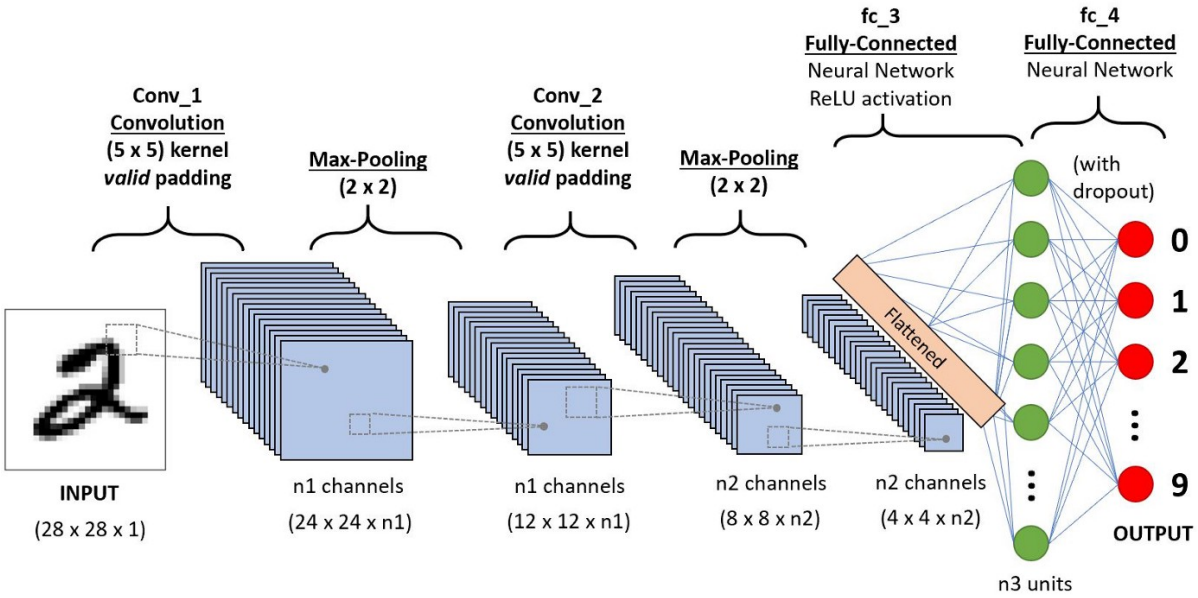


Figure 13 – A CNN sequence to classify handwritten digits.
Source: towardsdatascience - A Comprehensive Guide to CNN

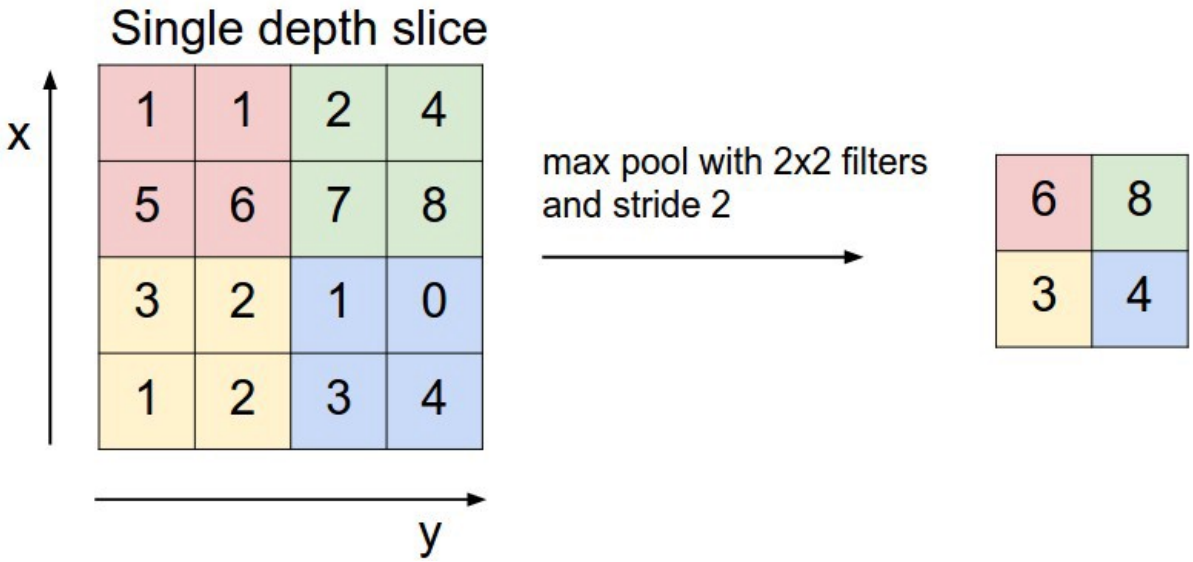


Figure 14 – A CNN sequence to classify handwritten digits.
Source: towardsdatascience - CNN

with deep learning, known as Deep Reinforcement Learning (DRL), allows agents to generalize across large or infinite state spaces by learning complex representations, thereby overcoming the limitations of classical methods. DRL has shown great success in tackling a wide range of complex tasks where traditional RL would otherwise struggle due to the curse of dimensionality. The generality of DRL framework allows its application in both discrete and continuous domains to solve tasks in robotics and simulated environments [Lillicrap et al., 2015].

3.5.1 Value-based Methods

In Reinforcement Learning (RL), value-based methods aim to optimize decision-making by estimating value functions that quantify the expected cumulative rewards for given states or state-action pairs. These methods are fundamental for determining the optimal policy, where the agent seeks to maximize long-term returns through the actions it selects. While traditional value-based methods, such as SARSA and Q-learning (Section 3.1.3), rely on tabular representations, their scalability to high-dimensional and continuous spaces is limited. This challenge has been effectively addressed by integrating deep learning techniques, leading to the development of value-based DRL methods.

3.5.1.1 Deep Q-Network (DQN)

The team of DeepMind introduced the DQN in [Mnih et al., 2013] and demonstrated its effectiveness later on in [Mnih et al., 2015], showing impressive results of the algorithm achieving human-level performance on various Atari games by learning directly from raw pixel inputs. DQN extends the traditional Q-learning algorithm by using deep neural networks to approximate the Q-value function. In this way, the algorithm can learn in environments with large or continuous state spaces. Figure 15 presents the DQN architecture used by Mnih et al. [2015].

The core idea behind DQN is to approximate the optimal action-value function, $Q(s,a,\theta)$, where θ represents the network's parameters. The neural network provides as output a probability distribution across all potential actions to identify the optimal action to take. Additionally, DQN introduced two key innovations to overcome the instability problem of value-based methods due to the constantly shifting Q-values during training: *experienced replay* and *target network* [Sewak, 2019]

- *Experience Replay*: To break the correlation between consecutive experiences during training, the agent stores its experiences $(S_t, A_t, R_{t+1}, S_{t+1})$ in a replay buffer. In each training step, the algorithm samples randomly mini-batches of experiences from this buffer and fed to the network. This process enhances convergence and reduces overfitting.
- *Target Network*: In standard Q-learning, the Q-values are updated using estimates from the same network, which can cause instability. DQN addresses this by employing a separate

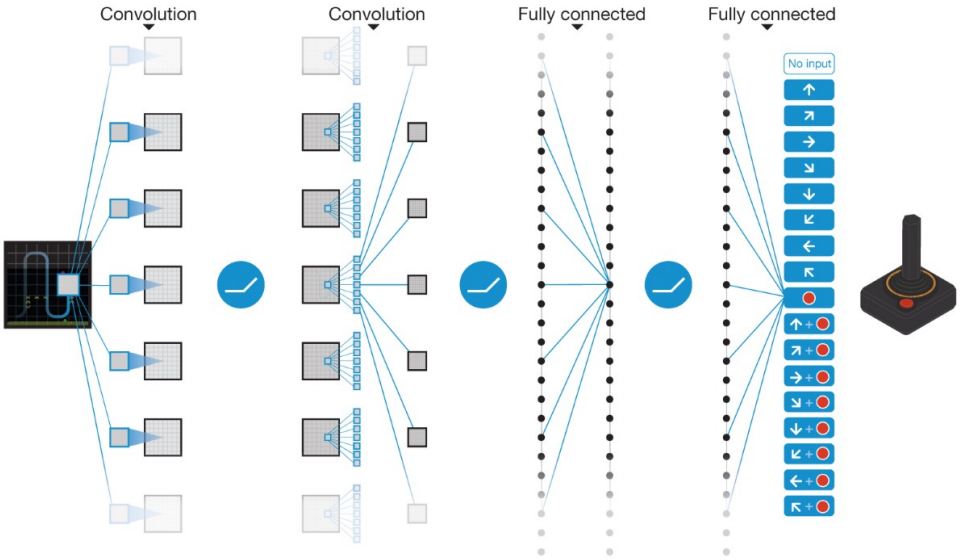


Figure 15 – Schematic illustration of the DQN.
Source: Mnih et al. [2015]

target network, which is updated at slower intervals.

The learning process in DQN involves minimizing the following loss function:

$$L_i(\theta_i) = \mathbb{E}_t \left[\left(R_{t+1} + \gamma \max_a Q(S_{t+1}, a; \theta_i^-) - Q(S_t, A_t; \theta_i) \right)^2 \right], \tag{3.19}$$

where θ^- represents the parameters of a target network, which is a separate copy of the Q-network that is updated less frequently to improve stability in training.

Following the presentation of the successful DQN approach, further investigations arose, proposing variations of DQN, such as Double Deep Q-Network (DDQN) and Dueling DQN. Additionally, Hessel et al. [2018] and Jäger et al. [2021] discuss and present an algorithm that combines all the mapped improvements.

3.5.1.2 Double DQN

One limitation of the original DQN algorithm is the tendency to overestimate action values, which can lead to suboptimal policies. This issue occurs because in DQN the same Q-value is used to select the action and evaluate the action. DDQN, proposed by Van Hasselt et al. [2016], addresses this issue by decoupling the action selection from the action evaluation, i.e, using two networks.

In DDQN, the action that maximizes the Q-value is selected using the online network θ, but the Q-value itself is evaluated using the target network θ^-. The loss function becomes:

$$L_i(\theta_i) = \mathbb{E}_t \left[\left(R_{t+1} + \gamma Q(S_{t+1}, \operatorname*{argmax}_a Q(S_{t+1}, a; \theta_i); \theta_i^-) - Q(S_t, A_t; \theta_i) \right)^2 \right]. \tag{3.20}$$

3.5.1.3 Dueling DQN

Dueling DQN [Wang et al., 2016] is an architectural improvement on the standard DQN, introduced to better capture the value of being in a particular state, regardless of the specific action taken. The dueling architecture splits the Q-function into two separate streams: one to estimate the value $V(s)$ of being in a state s, and the other to estimate the advantage of each action in that state $Adv(s,a)$, relative to the other actions.

By learning separate estimates of state value and action advantage, Dueling DQN enables the agent to more effectively learn which states are valuable and which actions are critical in those states. These two streams are then combined to form the Q-value as follows:

$$Q(s,a;\theta,\alpha,\beta) = V(s;\theta,\beta) + Adv(s,a;\theta,\alpha), \tag{3.21}$$

where θ denotes the parameters of the convolutional layers, while α and β are the parameters of the two streams of fully-connected layers.

Despite the success of the value-based methods, several challenges remain. One key limitation is their reliance on discrete action spaces. While algorithms such as DQN and its variants are effective in environments with a limited set of discrete actions, they struggle with continuous action spaces, which are common in robotics. Additionally, value-based methods often require extensive exploration to discover optimal policies, and balancing exploration with exploitation is a challenge.

3.5.2 Policy Gradient Methods

Policy Gradient methods present a different approach from traditional value-based methods such as Q-learning and its variants. Instead of estimating value functions to derive optimal policies, Policy Gradient methods optimize the policy directly. While a value function is not required for action selection in these methods, it can still assist in learning the policy parameters [Sewak, 2019; Sutton, 2018]. This advantage is particularly relevant in high-dimensional state-action spaces, such as robotic control tasks, where continuous action spaces provide more precise control over the robot's movements. Unlike discrete actions, which are limited to a predefined set of choices, continuous actions allow for smoother, more adaptable behaviors, making them more suitable for real-world applications and influencing the choice of the DRL method.

The policy $\pi(a|s,\theta)$ is typically stochastic, parameterized by θ, and denotes the probability that action a is taken at step t given that the environment is in state s at step t with parameter θ. The objective is to optimize the parameters θ such that the expected cumulative reward is maximized. The objective function can be formalized as [Sewak, 2019; Sutton, 2018]:

$$J(\theta) = \mathbb{E}\sum_{t=0}^{N}\left[\gamma^t r_t|\pi_\theta\right]. \tag{3.22}$$

Therefore, the optimal θ^* maximizes the expected reward following this policy, such as

$$\theta^* = \underset{\theta}{\operatorname{argmax}} J(\theta). \tag{3.23}$$

The parameters θ are updated via gradient ascent, which is the inverse of gradient descent. The gradient ascent updates the parameters θ in the positive direction of the gradient of the objective function $\nabla_\theta J(\theta_t)$ as per the following equation, where α is the learning rate.

$$\theta_{t+1} \leftarrow \theta_t + \alpha \nabla J(\theta_t). \tag{3.24}$$

• REINFORCE: Monte Carlo Policy Gradient

REINFORCE is one of the earliest and most straightforward policy gradient algorithms introduced by Williams [1992]. It works by using samples of complete episodes (trajectories of states, actions, and rewards) to compute the policy gradient. The gradient of the expected return $J(\theta)$ is estimated as:

$$\nabla J(\theta) = \mathbb{E}_\pi \left[G_t \nabla \ln \pi(A_t | S_t; \theta) \right], \tag{3.25}$$

where $G_t = \sum_{k=t}^{T} \gamma^k R_{t+k+1}$.

3.5.3 Actor-Critic Methods

Actor-Critic methods are considered a hybrid approach that combines both policy gradient and value-based methods in reinforcement learning, where an actor (the policy) and a critic (the value function) are learned simultaneously. This combination allows this method to leverage the benefits of policy search methods with learned value functions, which are able to learn from full returns and/or TD errors [Belousov et al., 2021; Sutton, 2018].

Figure 16 shows the learning architecture of the actor-critic methods. The actor is responsible for learning the policy and deciding which actions to take in a given state. This aligns with the policy gradient methods discussed earlier, where the policy is updated directly to maximize cumulative rewards. The critic, on the other hand, evaluates the actions taken by the actor by estimating its value through the value function. This component draws from value-based methods, and this evaluation is the TD error (Eq 3.12). Additionally, the actor uses the Critic's output to update its policy [Konda and Tsitsiklis, 1999].

Actor-Critic methods, such as DDPG, TD3, and SAC, are particularly well-suited for continuous action spaces, making them ideal for applications like robot navigation.

3.5.3.1 Deep Deterministic Policy Gradient (DDPG)

Deep Deterministic Policy Gradient (DDPG) is a model-free actor-critic algorithm presented in Lillicrap et al. [2015] that extends deterministic policy gradients [Silver et al., 2014] to work with deep neural networks, enabling it to handle continuous action spaces effectively.

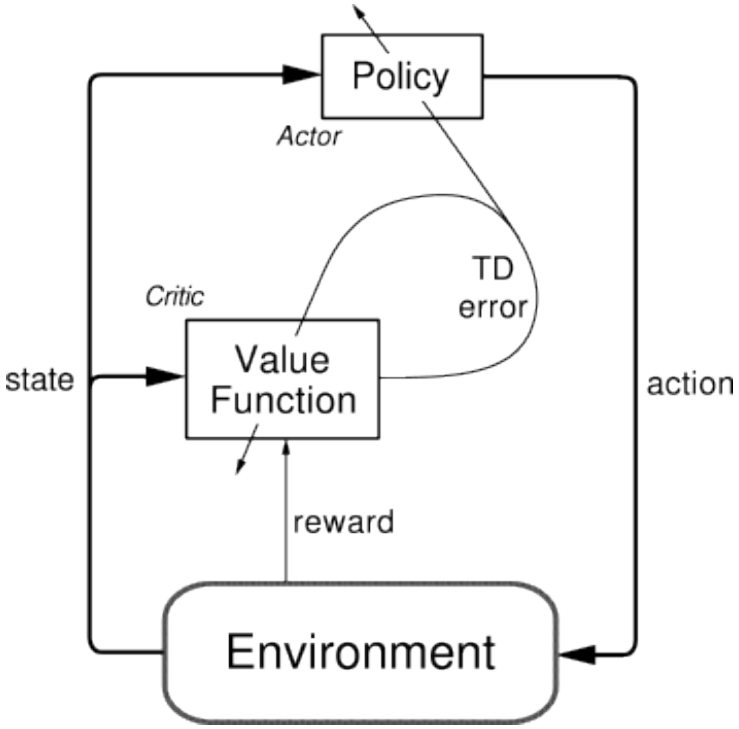


Figure 16 – Actor-Critic Architecture.

Source: Sutton [2018].

DDPG addresses the limitations of traditional methods such as DQN and its variants, which are designed for discrete action spaces, making them unsuitable for complex tasks like robotic control or autonomous vehicles, where continuous and high-dimensional action spaces are crucial.

In policy gradient methods, the policy is usually stochastic, which is suitable for exploration in discrete action spaces. However, in continuous action spaces, sampling from a stochastic policy can become inefficient. Therefore, in DDPG the Actor learns from a deterministic policy $\mu(s|\theta)$, which directly maps a state s to an action a, avoiding the need to sample actions stochastically [Silver et al., 2014].

The Critic is learned using the Bellman equation as in Q-Learning (Eq 3.16), and the deterministic policy gradient is given by:

$$\nabla_{\theta^\mu} J(\theta) \approx \mathbb{E}_{s_t \sim \rho^\beta} \left[\nabla_a Q(s,a|\theta^Q)\big|_{s=s_t, a=\mu(s_t)} \nabla_{\theta^\mu} \mu(s|\theta^\mu)\big|_{s=s_t} \right], \tag{3.26}$$

where $\nabla_{\theta^\mu} J(\theta)$ represents the gradient of the policy's objective function with respect to the actor's parameters θ^μ; $Q(s,a|\theta^Q)$ is the critic's estimate of the action-value function; and ρ^β is the replay buffer.

A major challenge of learning in continuous action spaces is exploration. To increase the exploration, the authors propose to introduce a time-correlated noise $\mathcal{N}$ in the actions during training. Therefore, the problem of exploration is addressed independently from the learning

algorithm, as follows:

$$\mu'(s_t) = \mu(s_t|\theta_t^{\mu}) + \mathcal{N}. \tag{3.27}$$

3.5.3.2 Twin Delayed Deep Deterministic Policy Gradient (TD3)

Twin Delayed Deep Deterministic Policy Gradient (TD3) is a DDPG successor *off-policy* learning algorithm designed for continuous control [Fujimoto et al., 2018]. The method was developed to solve critical issues of overestimation bias associated with value-based methods, such as DDQN, which can result in suboptimal policies and divergent behavior.

TD3 address the overestimation issue by introducing three critical innovations to improve the DDPG method:

- Clipped Double Q-Learning: Different from the original DDQN, which optimizes a pair of actor-critics, this method propose to take the minimum value between the two estimates to the target update. The target function is given by:

$$y = r + \gamma \min_{i=1,2} Q_{\theta_i'}(s', \pi_{\phi_1}(s')), \tag{3.28}$$

  where s' represents the state in the step $t+1$, and r is the reward.

- Delayed Policy Updates: TD3 include a delayed update for the actor (policy), being less frequently than the critic (Q-function). This approach guarantees that the actor is trained based on more accurate Q-value estimates.

- Target Policy Smoothing Regularization: Seeking to avoid overfitting and increase the exploration, TD3 introduces uncorrelated mean-zero clipped Gaussian noise (ε) to the actions during learning. Therefore, the target update is modified as follows:

$$y = r + \gamma \min_{i=1,2} Q_{\theta_i'}(s', \pi_{\phi_1}(s') + \varepsilon), \tag{3.29}$$

$$\varepsilon \approx clip(N(0, \sigma), -c, c). \tag{3.30}$$

3.5.3.3 Soft Actor Critic (SAC)

Soft Actor-Critic (SAC) is an off-policy algorithm, which learn from past samples using experience replay buffers [Haarnoja et al., 2018a,b]. Despite being derived from DDPG [Lillicrap et al., 2015] and being closest to TD3 [Fujimoto et al., 2018] with the clipped double-Q trick, SAC seeks to optimize a stochastic policy with an entropy regularization. This method trains the policy to maximize a trade-off between the expected reward and entropy, promoting more efficient exploration and robust learning, while minimizing the need for fine-tuning hyperparameters during training, which is a great advance for real-world applications.

The regularization strategy, including the entropy term, encourages the agent's exploration by favoring policies that maintain a certain degree of randomness in their actions, helping

to prevent convergence to suboptimal policies. The maximum entropy objective [Ziebart et al., 2008] generalizes the standard objective by augmenting it with an entropy term, such that the optimal policy additionally aims to maximize its entropy at each visited state [Haarnoja et al., 2018b]:

$$\pi^* = \underset{\pi}{\operatorname{argmax}}\, \mathbb{E}_{(s_t,a_t)\sim\rho_\pi}\left[\sum_{t=0}^{\infty}\gamma^t\left(R(s_t,a_t,s_{t+1}) + \alpha H(\pi(.|s_t))\right)\right] \tag{3.31}$$

where $\alpha > 0$ is the temperature parameter that determines the relative importance of the entropy term versus the reward, and thus controls the stochasticity of the optimal policy π^*. The entropy term is given by:

$$H(\pi(.|s_t)) = \log(\pi(a_t|s_t)). \tag{3.32}$$

From Eq. 3.31 it is possible to derive the state-value function $V_\pi(s)$ and the action-value function $Q_\pi(s,a)$ as follows:

$$V_\pi(s) = \mathbb{E}\left[\sum_{t=0}^{\infty}\gamma^t\left(R(s_t,a_t,s_{t+1}) + \alpha H(\pi(.|s_t))\right) \mid s_0 = s\right], \tag{3.33}$$

$$Q_\pi(s,a) = \mathbb{E}\left[\sum_{t=0}^{\infty}\gamma^t R(s_t,a_t,s_{t+1}) + \alpha\sum_{t=1}^{\infty}\gamma^t H(\pi(.|s_t)) \mid s_0 = s, a_0 = a\right]. \tag{3.34}$$

Additionally, it is possible to define the relation between $V_\pi(s)$ and $Q_\pi(s,a)$ and substitute the term in Eq. 3.32:

$$V_\pi(s) = \mathbb{E}_{a_t\sim\pi}\left[Q_\pi(s,a) - \alpha\log(\pi(a_t|s_t))\right]. \tag{3.35}$$

Similar to TD3, SAC uses the minimum value between the two Q-functions for policy updates, reducing the risk of overestimation. Therefore, the Q-function update is given by [Haarnoja et al., 2018a]:

$$Q(s_t,a_t) = R(s_t,a_t) + \gamma\mathbb{E}_{s_{t+1}\sim\pi}\left(\min_{i=1,2} Q_i(s_{t+1},a_{t+1}) - \alpha\log\pi(a_{t+1}|s_{t+1})\right). \tag{3.36}$$

Finally, the policy update is given by [Haarnoja et al., 2018a]:

$$J_\pi(\theta) = \mathbb{E}_{s_t\sim D,\varepsilon_t\sim\mathcal{N}}\left[log\pi_\theta(a_t|s_t) - Q(s_t,a_t)\right], \tag{3.37}$$

where ε_t is an input noise vector, sampled from some fixed distribution, such as a spherical Gaussian.

3.5.4 Reward Shaping

In reinforcement learning studies, the optimal policy is determined based on a reward function. A good representation of the reward function is crucial because it defines the agent's objective and directly influences the type of behavior that will be learned. In this context,

techniques such as reward shaping have emerged to enhance the reward function, allowing for the scaling up of learning to deal with complex problems.

Skinner [1965] introduced the term *shaping* to describe the effective training of an animal by reinforcing successive approximations of desired behavior [Sutton, 2018]. Later, Ng et al. [1999] formalized the term *reward shaping* in reinforcement learning as the practice of modifying the reward function to guide an agent to learn faster. By giving additional or modified feedback, *reward shaping* can increase learning efficiency, aid in the elimination of sparse rewards, and promote more robust exploration, particularly in complex and high-dimensional environments.

Reward shaping can be applied by modifying the reward function of the original MDP as follows:

$$R'(s,a,s') = R(s,a,s') + F(s,a,s'), \tag{3.38}$$

where s is the current state, a represents the actions, and s' is the transition states upon taking the action a. $R(s,a,s')$ is the reward function of the original MDP and $F(s,a,s')$ is the shaping reward function. We will hide the step term t in the equations for practicality. Using SARSA algorithm [Sutton, 2018] to represent the concept of *reward shaping* the Q-value update presented in Eq. 3.15 can be modified as:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[R(s,a,s') + F(s,a,s') + \gamma Q(s,a) - Q(s,a)\right], \tag{3.39}$$

where $F(s,a,s')$ is the *reward shaping*.

The concept of modifying reward functions to accelerate learning and enhance performance in environments with multiple goals and complex behaviors was early introduced in reinforcement learning by Gullapalli and Barto [1992], Randløv and Alstrøm [1998], and Mataric [1994]. However, the concept lacked theoretical analysis and formalism, and some attempts did not work as expected, failing to achieve the learning objective.

One challenge in *reward shaping* is that altering the reward function effectively transforms the original problem M into a new problem M'. This adjustment does not guarantee that the solutions or policies optimal in M' are also valid for the original problem M. Ideally, the *reward shaping* method should preserve the optimal policies, ensuring that high-quality policies learned with the modified rewards remain effective for the original problem.

In 1999, Andrew Y. Ng introduced the *reward shaping* in Ng et al. [1999], providing a detailed formalization of the method. The authors presented the Potential-Based Reward Shaping (PBRS) method, which involves adding a shaping reward derived from a potential function that depends on the state. PBRS formalization became a fundamental concept in RL, providing both theoretical insights and practical applications for improving learning efficiency without compromising the final outcome.

Ng et al. [1999] prove that if the *reward shaping* F is a difference of potential function,

i.e., if there exists a real-valued function $\phi : S \longrightarrow \mathbb{R}$ such that for all $s \in S - s_0$, $a \in A$, $s' \in S$:

$$F(s,a,s') = \gamma\phi(s') - \phi(s), \tag{3.40}$$

the optimal policy in the new MDP ($M' = (S,A,T,\gamma,R+F)$) remains optimal for the original MDP ($M = (S,A,T,\gamma,R)$), where T are the next-state transition probabilities, and $\gamma \in (0,1)$ is the discount factor.

Following the method proposed by Ng, undesirable behavior can be discouraged, such as in Randløv and Alstrøm [1998], where the reward was modified to include an extra term encouraging proximity to the goal in a bicycle-riding task aimed at reaching a target location. As a result, the agent learned to ride in cycles without moving towards the goal.

Reward shaping plays a pivotal role in improving the learning efficiency and performance of the RL agents, primarily in goal-oriented tasks. By carefully designing the reward structure, we can guide the agent's exploration and encourage behaviors that lead to more efficient policy learning. This is particularly important in complex environments, such as mapless robot navigation.

CHAPTER

4

METHODOLOGY

As discussed in Section 1.1, this dissertation aims to propose an autonomous agent design for mapless robot navigation. This problem can be considered a goal-oriented task and defined as the modification of a given state ($\vec{s}_k$) by specifying actions ($\vec{a}_k$):

$$\vec{s}_{k+1} = f(\vec{s}_k, \vec{a}_k).$$

The autonomous agent needs to determine control actions based on a set of information and a state space. Considering robotic systems, we can define the states $\vec{s}$ and actions $\vec{a}$ as follows:

$$\vec{s} = \begin{bmatrix} \vec{q} \\ \dot{\vec{q}} \\ \vec{l} \end{bmatrix}, \quad \vec{a} = \tau. \tag{4.1}$$

For simplicity, we omit the time index k in $\vec{s}$ and $\vec{a}$, assuming that states and actions are implicitly indexed at each time step. Here, $\vec{q}$ is the configuration vector of the robot, which may represent position, orientation, or joint angles, depending on the robot type. $\dot{\vec{q}}$ is the time derivative of the configuration vector and typically represents velocities. $\vec{l}$ denotes the exteroceptive sensor measurements (e.g. LiDAR). The control actions τ represent commanded velocities or torques.

This model directly maps raw sensor data to velocities in the robot workspace. We do not constraint the robot's kinematics and assume that the control frequency is fast enough so that the robot can plan the next step as new observations arrive. Figure 17 presents a diagram illustrating this model.

Therefore, using DRL the autonomous agent will learn an optimal policy to determine navigation actions. In this framework, the navigation problem can be formulated as a MDP, defined by $M = (S, A, T, \gamma, R)$, where (S) represents the states, (A) the actions, (T) the transition function, (R) the reward function, and (γ) the discount factor. This formulation allows the agent to iteratively refine its policy through continuous interaction with the environment, improving its decision-making over time.

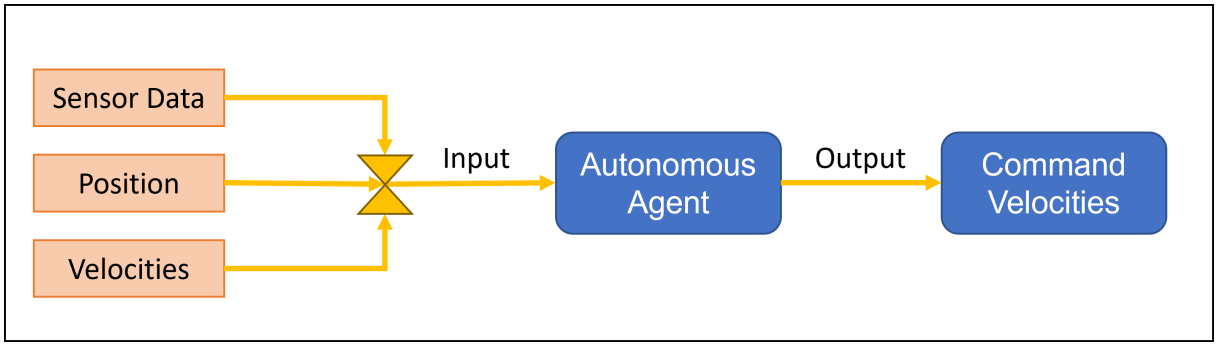


Figure 17 – Model operation diagram.

The optimal policy, obtained by solving the MDP, provides the agent with the best strategy for acting in a given environment. During learning, the reward serves as feedback from the environment, evaluating the agent's actions in each state. Therefore, defining the reward function is a crucial step in formulating a DRL problem, as it directly influences the agent's behavior and learning efficiency.

In goal-oriented problems, the most straightforward reward function involves providing a positive reward to the agent upon successfully reaching the goal state, as represented below:

$$R(s,a,s') = \begin{cases} r_a & \text{if } s' = s_{goal}, r_a > 0, \\ 0 & \text{otherwise}, \end{cases} \tag{4.2}$$

where s is the current state, a the action chosen, s' the state resulting from the action taken, and s_{goal} the goal state.

To consider a mapless navigation scenario, this reward can be rewritten to achieve the desired collision avoidance behavior by including a negative reward to the agent in case of any collision. Additionally, the reward can be adjusted to accelerate learning by setting a timeout to reach the goal. The resultant reward can be:

$$R(s,a,s') = \begin{cases} r_a & \text{if } s' = s_{goal}, \ r_a > 0, \\ r_c & \text{if } s' = s_{obstacle}, \ r_c < 0, \\ r_T & \text{if } EP_{dt} \geq \text{Timeout}, \ r_T < 0, \\ 0 & \text{otherwise}, \end{cases} \tag{4.3}$$

where $s_{obstacle} \in \mathcal{O}$ denotes the state in which an obstacle from a set of obstacles is located within the environment, EP_{dt} is the episode duration, and $Timeout$ the limit established for the episode duration.

4.1 Reward Shaping

Many works have addressed the reward shaping, aiming to obtain better learning results and thereby improve the agent's behavior in the proposed tasks [Alipanah and Moosavian, 2022;

Randløv and Alstrøm, 1998]. However, this reward modification must be applied carefully to avoid unexpected behaviors or convergence to suboptimal solutions. Ng et al. [1999] presented the importance of policy invariance in reward shaping to avoid "positive reward cycles", where the agent learns a suboptimal policy, leading to being stuck in a sub-task loop (rather than the goal), hindering progress towards the intended goal.

Although the reward function presented in Eq. 4.3 essentially captures the expected behavior of the agent for the mapless navigation problem [Cimurs et al., 2021; Grando et al., 2021], this simple and sparse reward structure results in poor learning performance. It leads to slow training, limited generalization, and hinders zero-shot sim-to-real transfer. Moreover, it is not ideal for application in more complex and structured environments. To enhance the reward function, reward shaping can be applied by incorporating a distance-based reward. This encourages the agent to move directly toward the goal, accelerating learning while preserving the original behavior defined by the initial reward.

Therefore, a new reward can be defined by building on the initial reward and adding a shaping function that accounts for the distance from the current state to the goal, as shown below:

$$R'(s,a,s') = R(s,a,s') + F(s,a,s'), \tag{4.4}$$

where $R'(s,a,s')$ is the new reward, $R(s,a,s')$ is represented by Eq. 4.3 and $F(s,a,s')$ is the distance-based shaping function that can be written as below:

$$\begin{aligned} F(s,a,s') &= \gamma\phi(s') - \phi(s), \\ \phi(s) &= \frac{1}{D(s,s_{goal})}, \end{aligned} \tag{4.5}$$

where $D(s,s_{goal})$ is the Euclidean distance between the current state s and the goal state s_{goal}, and $\gamma \in (0,1)$ is the discount factor.

Most papers in the literature have successfully adopted this distance-based reward or its variations, including in real-world applications [Cimurs et al., 2021; Hu et al., 2020a; Xu et al., 2024]. However, even with this reward shaping, the agent can still become trapped in suboptimal solutions in more complex scenarios, achieving a false maximum reward. Although the dissertation is primarily focused on reward design and its parameters, it is important to note that the distance-based approach also relies on precise global localization, which can often be challenging to achieve in real-world scenarios and may mislead the agent on its way to the goal. These issues are particularly evident in mapless robot navigation, where the robot can become stuck in a local optimum, ultimately failing to reach the final goal, as illustrated in Figure 18.

To better illustrate this suboptimal convergence, Figure 19 shows a heatmap representing the reward distribution in a navigation task, where brighter areas indicate higher rewards and darker regions correspond to lower values. The concave region in the center creates a local optimum, where the reward is higher than its immediate surroundings but significantly lower than

the global maximum near the target. An agent navigating into this region may become trapped, as moving toward the true goal initially results in a temporary reward decrease, discouraging escape. This demonstrates a key challenge in reinforcement learning, where agents relying solely on distance-based rewards, such as Eq. (4.5), may fail to reach the optimal solution. Trott et al. [2019] also illustrate this behavior through a simple toy task where the agent needs to reach the goal by controlling its position along a warped circular track, but the distance-based reward creates a local optimum.

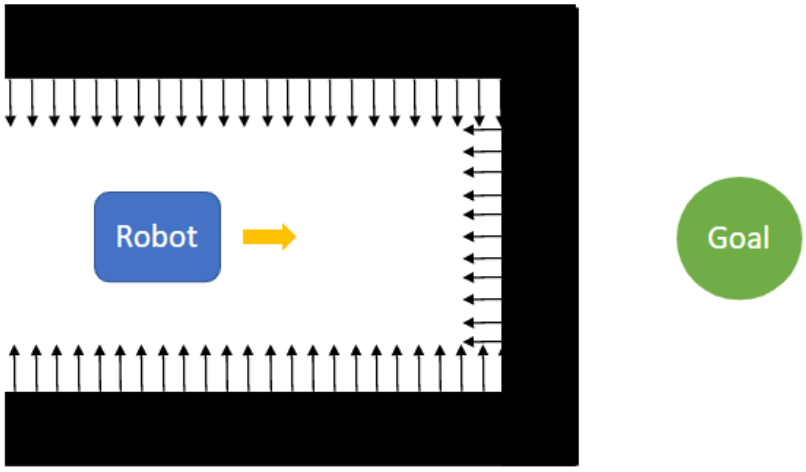


Figure 18 – Local optimum within the concavity: The robot moves into a concave area and receives a positive reward from the distance-based function (local optimum). It gets stuck when it encounters an obstacle blocking the goal. Moving back to finding an alternative path results in a reduced reward.

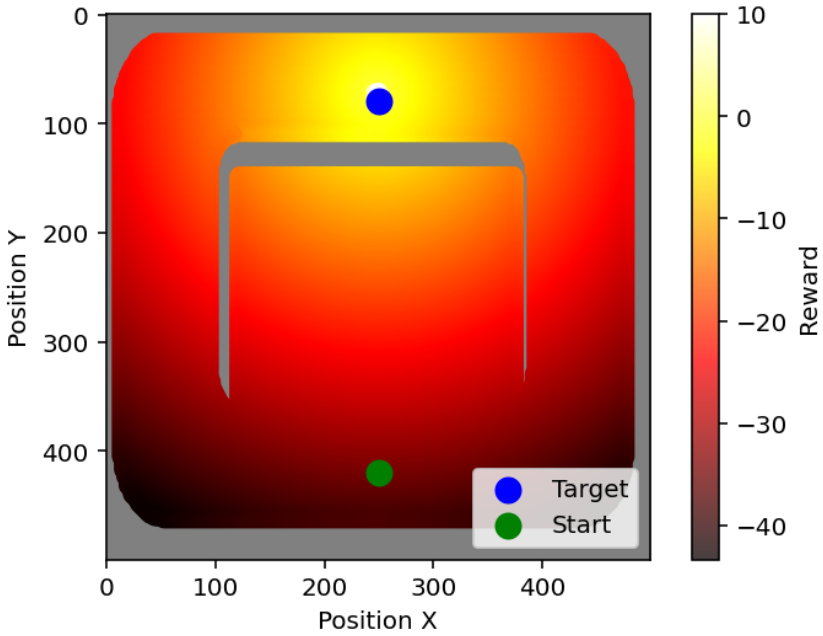


Figure 19 – Reward heatmap for a navigation task, where the color scale indicates the reward values. The yellow regions correspond to higher rewards, while darker red and black regions represent lower rewards. The start position is marked in green, and the target is in blue.

4.2 A novel reward shaping function for mapless navigation

Considering the issues presented in the previous section, we propose a reward-shaping function that, when combined with the original sparse reward function (Eq. 4.3), encourages the agent to take actions that lead it toward the target, avoid obstacles, and steer clear of local minima. The shaping function ($\phi(s_t)$) is derived from the environment information ($G(s_t)$) obtained at each step through the exploration of unexplored areas. In robot navigation applications, this environment information can be defined by probabilistic maps generated using LiDAR data, camera images, or any exteroceptive sensor.

This approach reinforces the agent exclusively when it navigates to novel locations. In this dissertation, the environment information $G(s_t)$ is computed as the difference between the sum of all occupied (Oc) and free (Fc) cells obtained from a occupancy grid map, generated only for training, at the current step t and the one from the previous step $t-1$, as follows:

$$G(s_t) = \sum(Oc_t + Fc_t) - \sum(Oc_{t-1} + Fc_{t-1}). \tag{4.6}$$

Therefore, the proposed shaping function can be defined as follows:

$$\phi(s_t) = K\left(1 - \frac{G(s_{t-1}) + \alpha}{G(s_t) + \alpha}\right), \tag{4.7}$$

where $G(s_{t-1})$ is the environment information captured in the previous step $t-1$, $G(s_t)$ represents the environment information acquired in the current step t as a result of the action taken, and α and K are arbitrary constants, where $K > 0$.

Analysing the Eq. 4.6, $G(s_t) \geq 0 \; \forall s \in S$. Additionally, $G(s_t) = 0$ if s_t leads to an already visited location where $Oc_t = Oc_{t+1}$ and $Fc_t = Fc_{t-1}$. Considering Eq. 4.7, the potential increases as the agent explores new locations, which encourages the exploration of undiscovered areas, avoiding getting stuck in local minima. Conversely, the potential decreases when the agent navigates through known areas. Thus, Eq. 4.7 represents a potential function and is therefore conservative, as demonstrated below.

Theorem 1. Let $\phi(s)$ be a potential shaping function. The function $\phi(s)$ is conservative if, for any closed trajectory, where the next state is equal to the starting state, the total shaping reward sums to zero.

Proof.
Let $G(s_t) = G(s_{t-1}) = x | x \in \mathbb{R}_0^+$, thus:

$$\begin{aligned} &\phi(s_t) = K\left(1 - \frac{x+\alpha}{x+\alpha}\right), \\ &\phi(s_t) = 0, \qquad\qquad \forall K, \in \mathbb{R}^+, \text{ and } \alpha \in \mathbb{R}. \end{aligned} \tag{4.8}$$

Now, let $G(s_t) > G(s_{t-1})$, thus:

$$\begin{aligned} &\phi(s_t) = K\left(1 - \frac{x+\alpha}{x+\varepsilon+\alpha}\right), \\ &\phi(s_t) > 0, \qquad\qquad \forall K, \varepsilon \in \mathbb{R}^+, \text{ and } \alpha \in \mathbb{R}. \end{aligned} \tag{4.9}$$

Based on the previous cases (Eq. 4.8 and 4.9), we can conclude that $\phi(s_t)$ is a conservative potential function.

□

As discussed by [Ng et al., 1999], in PBRS methods, the reward shaping function must be formulated as the difference of potential functions to maintain the optimality of the original MDP. This ensures that the agent will keep learning the correct behavior while benefiting from the additional guidance. Therefore, considering the function $\phi(s_t)$ in Eq. 4.7 and following the same structure of Eq. 4.4 we can define the following reward shaping function:

$$\begin{aligned} R'(s_t, a_t, s_{t+1}) &= R(s_t, a_t, s_{t+1}) + F(s_t, a_t, s_{t+1}), \\ F(s_t, a_t, s_{t+1}) &= \gamma\phi(s_{t+1}) - \phi(s_t), \end{aligned} \tag{4.10}$$

where $\gamma \in (0,1)$ is the discount factor.

To prove that the optimal policy remains unchanged under PBRS, it is necessary to show that adding the shaping reward does not modify the optimal policy. The following analysis evaluates the influence of the shaping reward on the SAC off-policy method.

Theorem 2. Let $F : S \times A \times S \to \mathbb{R}$ be a shaping reward function. The optimal policy in the new MDP ($\mathscr{M}' = (S, A, T, \gamma, R+F)$) remains optimal for the old MDP ($\mathscr{M} = (S, A, T, \gamma, R)$) if F is a PBRS function.

Proof. Consider the following SAC objective function that maximizes the entropy:

$$J(\pi) = \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{\infty} \gamma^t \left(R(s_t, a_t, s_{t+1}) + \alpha\mathscr{H}(\pi(\cdot|s_t))\right)\right], \tag{4.11}$$

where τ represents trajectories, γ is the discount factor, $R(s_t, a_t, s_{t+1})$ is the reward, $\alpha > 0$ is trade-off coefficient, and $\mathscr{H}(\pi(\cdot|s_t))$ is the entropy of the policy.

Also, consider the SAC optimal value function:

$$V^*(s) = \max_{\pi} \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \left(R(s_t, a_t, s_{t+1}) + \alpha\mathscr{H}(\pi(\cdot|s_t))\right) \mid s_0 = s, \pi\right]. \tag{4.12}$$

Substituting the original reward with the potential-based reward shaping results in:

$$V'^*(s) = \max_{\pi} \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \left(R(s_t, a_t, s_{t+1}) + \gamma\phi(s_{t+1}) - \phi(s_t) + \alpha\mathscr{H}(\pi(\cdot|s_t))\right) \mid s_0 = s, \pi\right]. \tag{4.13}$$

Separating the potential terms:

$$\begin{aligned} V'^*(s) = \max_{\pi} \Bigg(& \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t, s_{t+1}) + \alpha \mathscr{H}(\pi(\cdot|s_t)) \mid s_0 = s, \pi\right] \\ & + \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^{t+1} \Phi(s_{t+1}) \mid s_0 = s, \pi\right] - \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \Phi(s_t) \mid s_0 = s, \pi\right] \Bigg), \end{aligned} \tag{4.14}$$

$$V'^*(s) = \max_{\pi} \left(V^*(s) + \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^{t+1} \Phi(s_{t+1}) - \sum_{t=0}^{\infty} \gamma^t \Phi(s_t) \mid s_0 = s, \pi\right] \right). \tag{4.15}$$

Notice that the potential term is a telescope sum and can be simplified as:

$$\begin{aligned} &\sum_{t=0}^{\infty} \gamma^{t+1} \Phi(s_{t+1}) - \sum_{t=0}^{\infty} \gamma^t \Phi(s_t) = \gamma\Phi(s_1) - \Phi(s_0) + \gamma^2 \Phi(s_2) - \gamma\Phi(s_1) + \gamma^3 \Phi(s_3) - \gamma^2 \Phi(s_2) + \cdots \\ &\sum_{t=0}^{\infty} \gamma^{t+1} \Phi(s_{t+1}) - \sum_{t=0}^{\infty} \gamma^t \Phi(s_t) = -\Phi(s_0) + \lim_{T \to \infty} \gamma^T \Phi(s_T). \end{aligned} \tag{4.16}$$

Since $\Phi(s)$ is conservative, $\gamma^T \Phi(s_T) \to 0$ with $T \to \infty$. Thus we have:

$$V'^*(s) = V^*(s) - \Phi(s). \tag{4.17}$$

The same approach can be used for the SAC Q-function. Consider the following Q-function:

$$Q^*(s,a) = \mathbb{E}\left[R(s,a,s') + \gamma V^*(s') \mid s, a\right]. \tag{4.18}$$

Substituting the original reward with the potential-based reward shaping results in:

$$Q'^*(s,a) = \mathbb{E}\left[R'(s,a,s') + \gamma V'^*(s') \mid s, a\right]. \tag{4.19}$$

Substituting $R'(s,a,s')$ and $V'^*(s')$, we have:

$$Q'^*(s,a) = \mathbb{E}\left[R(s,a,s') + \gamma\Phi(s') - \Phi(s) + \gamma(V^*(s') - \Phi(s')) \mid s, a\right]. \tag{4.20}$$

Simplifying:

$$Q'^*(s,a) = \mathbb{E}\left[R(s,a,s') + \gamma V^*(s') - \Phi(s) \mid s, a\right]. \tag{4.21}$$

Finally:

$$Q'^*(s,a) = Q^*(s,a) - \Phi(s). \tag{4.22}$$

Since $\Phi(s)$ does not depend on the actions, it can be considered a constant concerning a. Therefore, the SAC optimal policy is:

$$\begin{aligned}
\pi^*(s) &= \arg\max_a Q'^*(s,a), \\
\pi^*(s) &= \arg\max_a \left(Q^*(s,a) - \Phi(s)\right), \\
\pi^*(s) &= \arg\max_a \left(Q^*(s,a) + \text{constant}\right), \\
\pi^*(s) &= \arg\max_a Q^*(s,a).
\end{aligned} \tag{4.23}$$

Therefore, the optimal policy $\pi^*(s)$ under the reward shaping is also the optimal policy for the original reward, proving that the optimality remains unchanged with the PBRS function for both the value function and the Q-function.

□

Combining distance-based reward shaping with the proposed approach can significantly enhance policy training by balancing the exploratory behavior of the reward function proposed in Eq. 4.7 with the distance-based reward function presented in Eq. 4.5. This combination allows the policy to leverage the strengths of both methods: the proposed reward encourages exploration by adjusting incentives to promote ongoing exploration throughout the environment, while the distance-based reward provides a clearer reward signal for actions that move to the goal. This balancing act is crucial for guiding the agent more efficiently toward the goal, facilitating the discovery of more effective paths, and avoiding stagnation in suboptimal solutions.

Combining both conservative potential functions (Eq. 4.7 and Eq. 4.5) preserves the PBRS without loss of generality. Therefore, the final proposed reward can be defined as follows:

$$R'(s,a,s') = \begin{cases} r_a & \text{if } s' = s_{goal}, \ r_a > 0, \\ r_c & \text{if } s' = s_{obstacle}, \ r_c < 0, \\ r_T & \text{if } EP_{dt} \geq \text{Timeout}, \ r_T < 0, \\ r_{GD} & \text{otherwise}, \end{cases} \tag{4.24}$$

where r_{GD} is the shaping term given by the following equation:

$$r_{GD} = \frac{K\left(1 - \frac{G(s)+\alpha}{G(s')+\alpha}\right)}{D(s, s_{goal})}. \tag{4.25}$$

4.3 Enhancing Generalization on Training

In RL, generalization refers to an agent's capacity to apply knowledge gained from training in one set of conditions to novel, unseen situations [Kirk et al., 2021]. This capability is particularly significant in real-world applications where environments are often complex and

dynamic. Effective generalization ensures that the learned policies are not overly specialized to the training environment but can instead perform well in diverse scenarios. This is essential for applying the zero-shot transfer strategy when performing sim-to-real transfer.

As previously discussed, reward shaping is effective in enhancing learning efficiency by accelerating learning and improving the agent's performance. Additionally, well-defined rewards ensure that the agent understands which actions lead to desired outcomes, facilitating generalization to new scenarios that share characteristics with those seen during training.

However, reward shaping alone may not be sufficient to ensure robust generalization. To address this limitation, domain randomization can be employed as a complementary strategy. Domain randomization involves varying the parameters and conditions of the training environment to expose the agent to a broader range of scenarios [Wada et al., 2022]. This technique aims to create a training regime where the agent encounters a diverse set of possibilities, enhancing its adaptability to real-world variations.

Additionally, encouraging the exploratory capacity of the agent is another crucial strategy for effective learning and generalization. Exploratory behavior allows the agent to discover new strategies and adapt to unforeseen conditions, improving its overall performance and avoiding local optimum.

Therefore, we propose the use of the off-policy learning algorithm SAC, which is designed for continuous and high-dimensional action spaces, aiming to promote both efficiency and exploration by maximizing not only the expected reward but also the policy's entropy, as outlined in Haarnoja et al. [2018a]. Other reinforcement learning algorithms often face a trade-off between exploration and exploitation. SAC tackles this trade-off by explicitly encouraging exploration through entropy regularization, which helps prevent premature convergence to suboptimal deterministic policies. The entropy term encourages stochastic policies that maintain diversity in actions, which is particularly beneficial in environments with sparse rewards or deceptive local optima.

Additionally, we implement a domain randomization strategy, where the initial state is randomly modified in each episode, and the environment configuration is varied throughout training. This technique exposes the agent to a wide range of scenarios during training, encouraging it to learn policies that generalize beyond the specific conditions encountered during simulation. By randomizing aspects of the scenarios, the agent becomes less sensitive to exact environment specifications and more reliant on invariant, transferable features. Domain randomization is particularly effective in sim-to-real transfer tasks, aiming to reduce the "reality gap". By training the agent in a distribution of environments rather than a fixed one, the learned policy is more likely to perform well when deployed in previously unseen, real-world scenarios.

By integrating the proposed reward shaping, domain randomization, and exploration strategies with SAC, we establish a comprehensive approach to enhancing both training efficiency

and generalization capabilities. This combination is responsible for enhancing the zero-shot transfer capacity.

CHAPTER

5

RESULTS

This chapter provides a performance analysis of the reward shaping and generalization strategies introduced in Chapter 4. The evaluation focuses on the benefits of the proposed reward shaping method, which utilizes environment information as described in Eq. 4.7. We compared the proposed reward againts the sparse reward approach (Eq. 4.3) and the distance-based shaping method (Eq. 4.5).

This evaluation involved analyzing the progression of rewards throughout the training phase of an agent within the same environment setup. To achieve this, a simple scenario was implemented in a basic simulator to represent a differential robot navigating a mapless navigation problem. The simulator proposed by [Surmann et al., 2020] and shown in Fig. 20 simulates a robot with a differential architecture, equipped with a configurable LiDAR. It provides the robot's pose data, LiDAR readings, and receives linear and angular velocities as input.

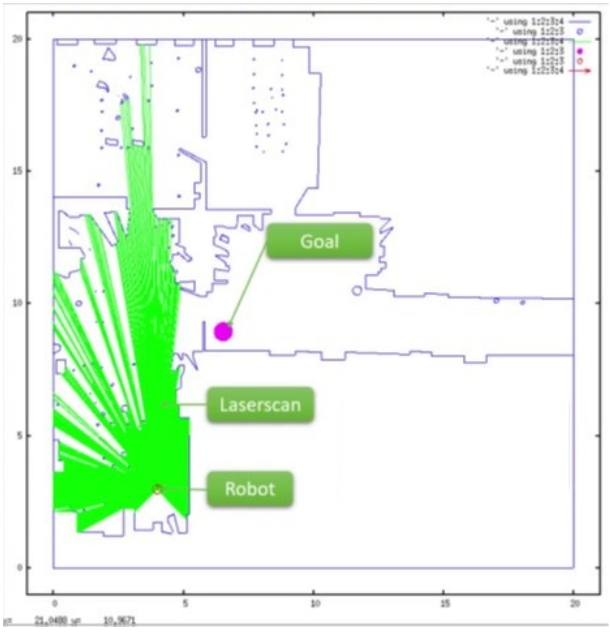


Figure 20 – 2D stand-alone simulator of a simple differential robot equipped with LiDAR.

5.1 Training setup

The training process was performed on a laptop with Ubuntu 20.04, equipped with an NVIDIA GTX 3060 graphics card, 16 GB of RAM, and Intel Core i7-11800H CPU. The learning environment is represented by a differential wheeled robot equipped with a planar LiDAR and navigating in limited workspaces, all running in the simulator presented in Fig. 20.

Fig. 21 illustrates the navigation system adopted. First, sensors collect data from the robot's states and the environment, and a goal point is defined. In the sequence, the system processes this data in order to create the observation state for the navigation policy model. Finally, the system defines the robot's velocity actions based on the observed states.

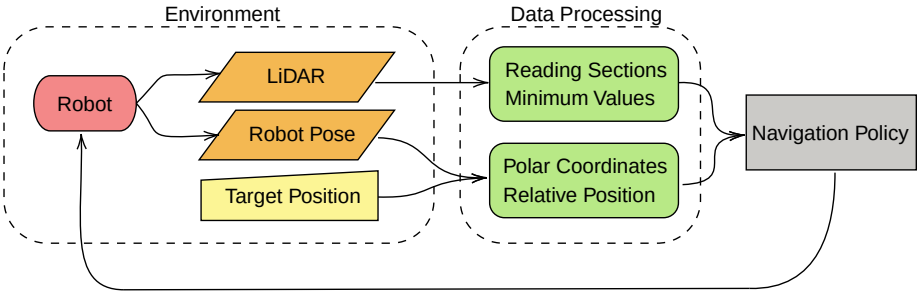


Figure 21 – Robot navigation system overview.

The observation state vector $\vec{s}$ includes the robot's relative position to the target region $\vec{p} \in \mathbb{R}^2$ in polar coordinates, absolute linear v and angular ω velocities, and the information data provided by the robot's planar LiDAR $\vec{l} \in \mathbb{R}^l$, with l being the number of laser beams. Formally, it can be represented as:

$$\vec{s} = \begin{bmatrix} \vec{p} \\ v \\ \omega \\ \vec{l} \end{bmatrix}, \tag{5.1}$$

where $\vec{p} = [d \ \alpha]^T$ and $\vec{l} = [l_1 \ l_2 \ l_3 \ \dots \ l_n]^T$. Figure 22 illustrates the elements of the observation space. On the other hand, the continuous action vector includes the commanded linear velocity $\bar{v} \in \mathbb{R}_+$ and commanded angular velocity $\bar{\omega} \in \mathbb{R}$, such that it can be represented by:

$$\vec{a} = \begin{bmatrix} \bar{v} \\ \bar{\omega} \end{bmatrix}. \tag{5.2}$$

All models used in this chapter were defined according to the actor-critic deep networks illustrated in Fig. 23. The actor network consists of an observation-state input layer, followed by three hidden dense layers with 512 nodes each, utilizing the Rectified Linear Unit (ReLU) activation function. The action output is generated by combining values from a linear layer using a sigmoid activation function for linear velocity and a hyperbolic tangent function for angular velocity. In the critic networks, the actor-generated action is combined with the observation state

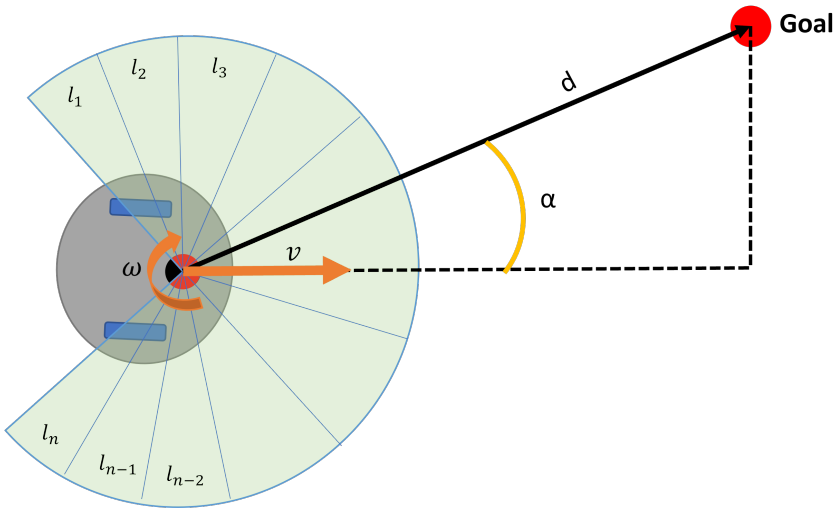


Figure 22 – Observation state space illustration.

as input, followed by three dense ReLU layers with 512 nodes each. The Q-value is computed using a linear activation function.

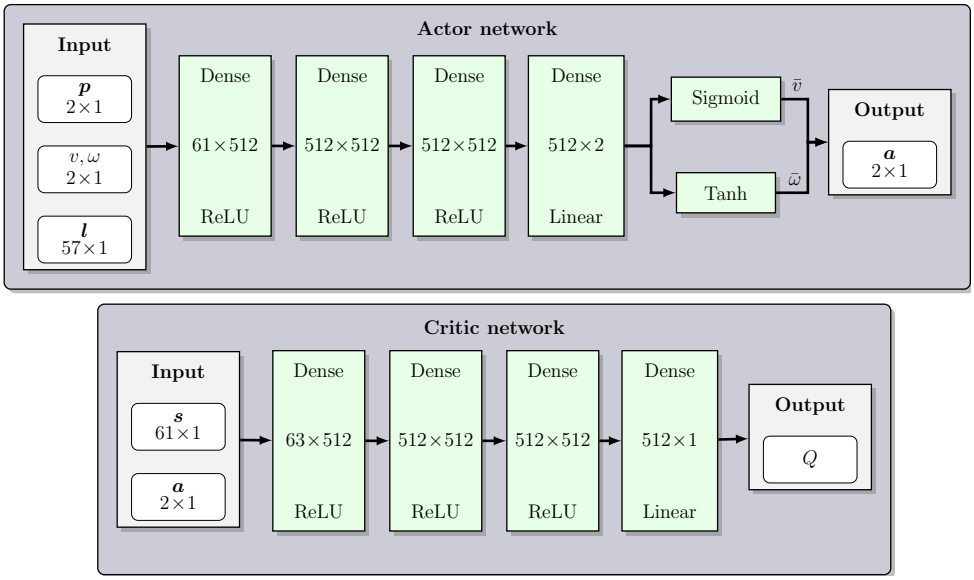


Figure 23 – Actor network: input layer formed by the observation state space, followed by three dense ReLU layers of 512 nodes, and the output action generated by merging values from a linear layer with a sigmoid and hyperbolic tangent activation functions. Critic network: input layer formed by the observation state space merged with the action space, followed by three dense ReLU layers of 512 nodes and the output Q-value generated by a dense linear layer.

In the defined environment, each training episode ends when the robot reaches the target region, collides with some obstacle, or violates the Timeout. The actions were limited to $v \in [0, 0.5]$ m/s, by diving the sigmoid output by 2, and $\omega \in [-1, 1]$ rad/s. The LiDAR has readings between $-135°$ to $135°$ with 684 distance measurements. Seeking to reduce the state space for the training and avoid overfitting due to repetitive obstacles, we have divided these 684 values into 57 groups of 12 measures and selected the 57 minimum distances of each group

[Choi et al., 2020]. Figure 22 shows the segmentation of the LiDAR readings, with $l_1, l_2, ..., l_n$ representing the set of n minimum distance of each group of 12 measurements, where $n = 57$.

5.2 Performance Evaluation

To evaluate the performance of the proposed strategy, we compared training results using the sparse reward approach (Eq. 4.3) and the distance-based shaping (Eq. 4.5) against the proposed reward shaping presented below.

$$R'(s,a,s') = \begin{cases} r_a & \text{if } s' = s_{goal}, \ r_a > 0, \\ r_c & \text{if } s' = s_{obstacle}, \ r_c < 0, \\ r_T & \text{if } EP_{dt} \geq \text{Timeout}, \ r_T < 0, \\ r_G & \text{otherwise}, \end{cases} \tag{5.3}$$

where r_G is the shaping term given by Eq.(4.7) as follows:

$$r_G = K\left(1 - \frac{G(s) + \alpha}{G(s') + \alpha}\right). \tag{5.4}$$

In addition, we have set the constants as presented in Table 5.1. These values were defined empirically based on preliminary experiments and domain knowledge. The chosen rewards reflect the expected behavior of the agent and were adjusted to ensure stable learning while encouraging the agent to pursue the desired objectives.

Table 5.1 – Constant parameters defined for the reward shaping function calculation.

K	α	r_a	r_c	r_t	l_1	l_2	d_{min}	Timeout
1	1	100	-200	-200	336	348	0.5	500

First, we analyzed the rewards progression results throughout the training phase of an agent within the same environment setup. To achieve this, a simple scenario was implemented in the simulator to represent a mapless navigation problem.

Fig. 24 presents the reward progression over 10,000 episodes using each reward function during training with the SAC algorithm in a unique map presented in Figure 26(a). It shows that the proposed reward shaping accelerates the agent's learning, reaching the optimal reward after just 1,000 episodes. In contrast, the distance-based reward method achieves optimal performance only after 4,000 episodes, while the sparse reward method performs the worst, reaching the optimal reward only after 5,500 episodes. This demonstrates the importance of reward shaping and highlights the advantages of the proposed method compared to more commonly used approaches. It is important to note that we are not focusing on the magnitude of the reward but rather on the speed of convergence during training.

Moreover, when comparing the proposed reward shaping (Eq. (5.3)) with the final shaping combination presented in Eq. (4.24) by adding the distance-based term, it can be observed that in

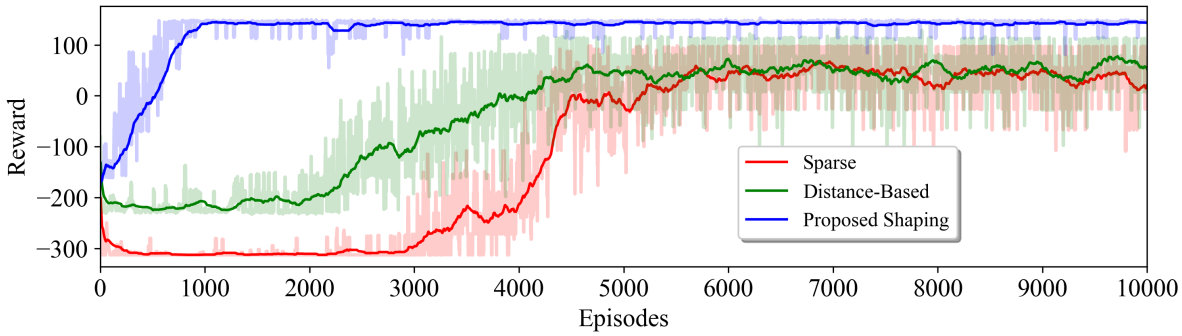


Figure 24 – Comparison of training reward progression using different reward strategies: sparse reward, distance-based shaping, and the proposed reward shaping method. The raw reward data are shown as shaded areas, while the smoothed lines are obtained by applying a moving average with a window size of 200 episodes.

both cases, the optimal reward is reached at around 1,000 episodes, as shown in Fig. 25. However, the combined reward can accelerate training in larger environments by rewarding exploration toward the goal instead of other areas.

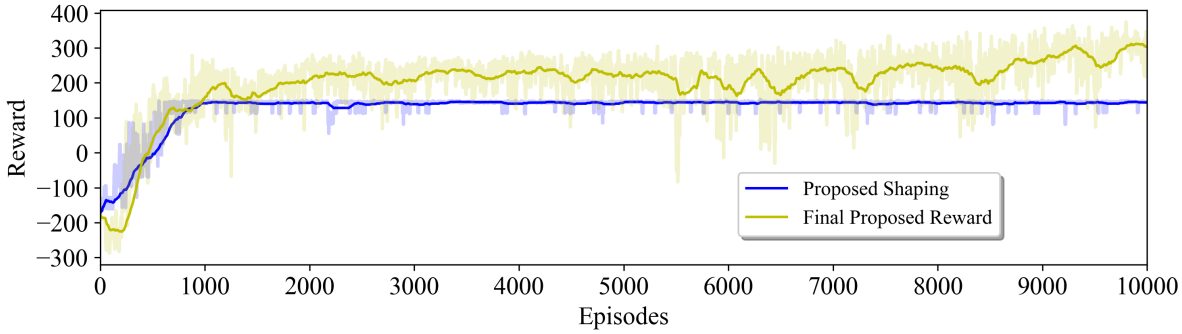
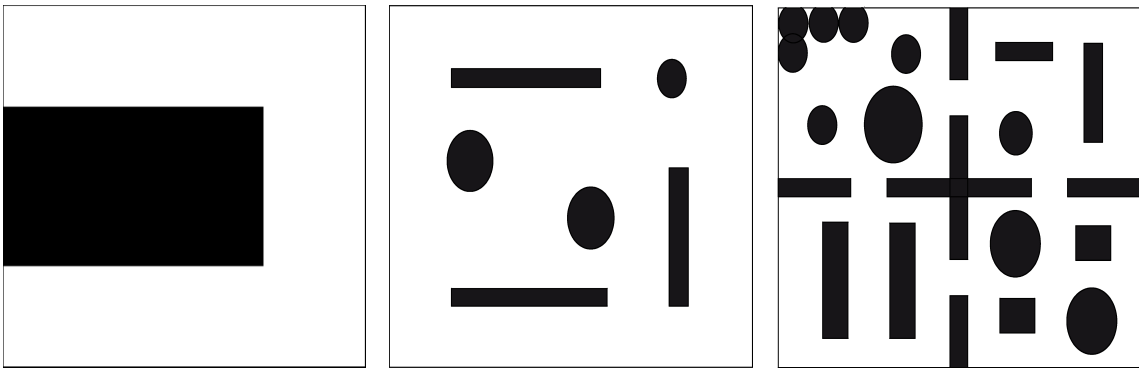


Figure 25 – Comparison of training reward progression using the proposed reward and the final reward, which combines the proposed gain of information with the distance-based approach. The raw reward data are shown as shaded areas, while the smoothed lines are obtained by applying a moving average with a window size of 200 episodes.

To assess the benefits of the proposed domain randomization, we compared the performance of the policies trained using the same rewards evaluated before, with and without domain randomization. Each policy was trained over 10,000 episodes on a unique map presented in Fig 26(b) (i.e., no domain randomization applied) and alternating every 500 episodes among three selected maps shown in Figure 26 (i.e., including domain randomization). The training maps vary in difficulty depending on the obstacle density, and all of them are bounded by obstacles. Additionally, certain areas within the maps were designed to induce local minima. For the performance comparison, we used the office-like map shown in Figure 27. This map includes obstacles and dead-end rooms designed to create local minima.

Table 5.2 demonstrates that the model trained with domain randomization outperforms the model trained on a single map when applied to a different scenario than it was trained on. This highlights the importance of generalization strategies for effective model transfer. Additionally,



(a) 10×10 m (b) 12×12 m (c) 20×20 m

Figure 26 – Simulated scenarios used for training in the learning step.

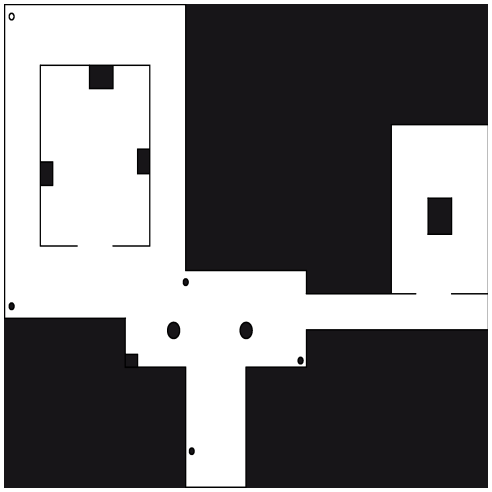


Figure 27 – Validation scenario used to compare the performance of learning strategies after training.

the model trained using the final proposed reward outperforms the others in any case. The impact of the SAC algorithm and the reward function will be further evaluated in Chapter 6.

Table 5.2 – Performance Comparison Between Models Trained With and Without Domain Randomization.

Reward \Gen. Strategy	**No Domain Randomization**	**Domain Randomization**
Final Proposed Reward	73%	**95%**
Distance-Based	53%	**61%**
Sparse	38%	**51%**

CHAPTER

6

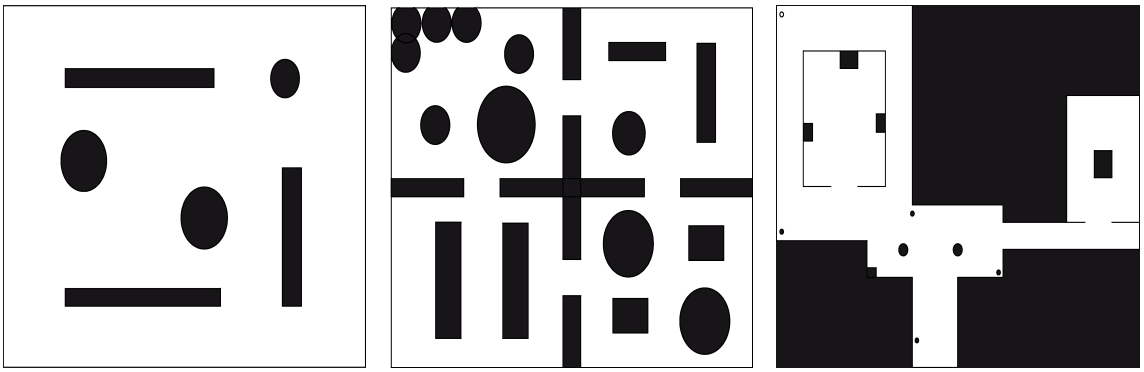
CASE STUDY

This chapter presents a case study for performance analysis of the reward shaping and generalization strategies discussed in this dissertation, focusing on enhancing training efficiency and zero-shot policy transfer. The evaluation focuses on autonomous robot mapless navigation, comparing the proposed method's performance with existing approaches in the literature.

Two main aspects have been analyzed: *i*) the impact of the exploration in the context of generalization employing TD3 or SAC algorithms for safe local navigation for mobile robots, and *ii*) the effect of using our novel reward function instead of others. To do so, we have first trained our policies in different simulated cluttered (sparse and complex) scenarios. Then, concerning the success rate (the number of times the robot reaches the goal region without collision in a limited time), we compare our method with other approaches in environments distinct from those used in the training stage. Next, to advance the generalization analysis of the solution and the zero-shot capacity, we added comparative trials in a second simulator (sim-to-sim analysis). Finally, we demonstrate the effectiveness of our proposal in a real-world cluttered scenario (sim-to-real analysis).

6.1 Training setup

The training process was set to run for 20,000 episodes, using the same simulator and training configuration presented in 5. The three maps illustrated in Fig. 28 have been employed at the training, whose dimensions are 12×12 m, 20×20 m, and 40×40 m, respectively. The maps used for training were carefully designed to challenge the agent's ability to navigate without relying on pre-existing maps or global localization. These maps vary in complexity, incorporating elements such as obstacles, narrow and open spaces, dead-end rooms, and regions with high similarity. All the maps are relatively simple but simulate real-world scenarios that a mobile robot may encounter. During the training, the map changes randomly every 500 episodes to enhance the agent's generalization ability and prevent overfitting to a specific environment.



(a) 12×12 m (b) 20×20 m (c) 40×40 m

Figure 28 – Simulated scenarios used for training in learning step.

6.2 Reward function

To evaluate the performance of the proposed strategy described in Chapter 4, we have defined the following reward function based on Eq. 4.24 and including other elements to improve the learning in the complex task of mapless autonomous navigation, as demonstrated in Miranda et al. [2024]. The reward function used in the experiments is defined as follows:

$$r = \begin{cases} r_a & \text{if } d \leq d_{min}, \\ r_c & \text{if collision}, \\ r_t & \text{if steps} \geq \text{Timeout}, \\ r_{GD} + r_l + r_v & \text{otherwise}, \end{cases} \tag{6.1}$$

where r_a is a positive reward assigned when the Euclidean distance d between the robot and the goal is less than or equal to a minimal threshold $d_{min} > 0$, indicating that the robot has reached the goal. r_c is a negative reward given in the event of a collision, and r_T is a negative reward given if the number of steps exceeds a predefined *Timeout* limit. Additionally, r_{GD} is the PBRS term defined in Eq. 4.25.

The other two terms in Eq. 6.1, r_l and r_v, which constitute the dense part of the reward, were included to encourage actions moving into free spaces (r_l) and prevent movements without progression (r_v) [Cimurs et al., 2021]. However, their conservative behavior was not demonstrated to qualify as a PBRS. r_l is the minimum value measured by the LiDAR within an interval between l_1 and l_2, representing a read in the robot's front side, and r_v is a reward based on the linear and angular velocities, as described below:

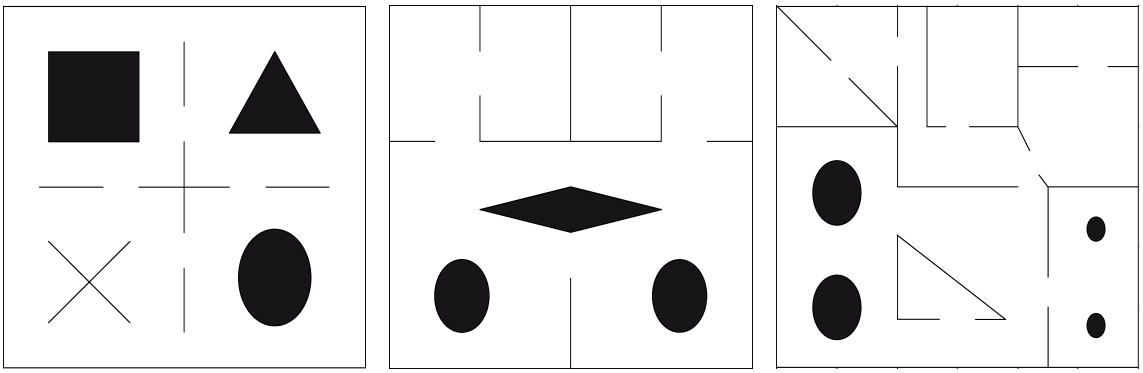
$$r_l = \min(\text{lidar}|_{l_1}^{l_2}), \tag{6.2}$$

$$r_v = v - |\omega|, \tag{6.3}$$

The constants were defined with the same values presented in Table 5.1.

6.3 Comparative Analysis

In order to evaluate the performance of the proposed reward function and training approach compared to other strategies from the literature, a sequence of tests was conducted in scenarios different from those previously trained. Fig. 29 illustrates the tested maps, relatively larger than those of Fig. 28, with dimensions 40×40 m, 40×40 m, and 60×60 m, respectively. The three maps represent cluttered environments with challenging elements for mapless navigation, such as corridors, dead-end rooms, and high similarity between locations. They were designed to introduce different levels of challenge in the tests, with Map 1 being the easiest, Map 2 of medium difficulty, and Map 3 the most challenging.



(a) Map 1: 40×40 m (b) Map 2: 40×40 m (c) Map 3: 60×60 m

Figure 29 – Simulated scenarios used for testing and evaluating the algorithms learning performance.

In our comparative analysis, we have used three different reward functions from the state-of-the-art literature. In Cimurs et al. [2021], the reward is based on the distance to the goal, collision information, and shaped based only on the robot's linear and rotational speeds ($v - |\omega|$). In Hu et al. [2020a], the authors also used the robot's speeds and distance to the goal, but they incorporated exteroceptive (LiDAR) data to compute a safety clearance reward term for collision avoidance purposes. Finally, in Grando et al. [2021], the reward is the simplest one, using only collision information and the arrival at the goal.

Our trials were separated into two groups. First, for all rewards functions (including ours), we have applied the TD3 algorithm proposed by Fujimoto et al. [2018] to train them. Here it is important to highlight that Cimurs et al. [2021] have used the TD3 to learn the navigation policy, while Hu et al. [2020a] and Grando et al. [2021] used the DDPG [Lillicrap et al., 2015], a predecessor of the TD3 and SAC. Second, for the same reward functions, we have replaced TD3 by the SAC algorithm. These algorithms were selected to compare their performance in learning policies for the robot navigation task, allowing for a comprehensive evaluation of their effectiveness in both the training environment and their generalization capacity.

Table 6.1 describes the learning parameters used for training all structures. However, as we have considered different reward functions for comparison, the reward constants received

values according to the author's proposal of each paper in analysis, as shown in Table 6.2. In the specific case of TD3, the delayed rewards were updated over the last 10 steps.

Table 6.1 – Learning parameters.

Learning Rate	Batch Size	Discount Factor	Update Rate	Policy Noise
0.0003	256	0.99	0.005	0.2

Table 6.2 – Policy reward constant values used by literature papers considered for comparing.

	Strategy		
Reward/Reference	**Cimurs et al. [2021]**	**Hu et al. [2020a]**	**Grando et al. [2021]**
Collision	$r_c = -100$	-	$r_c = -10$
Arrival	$r_a = 80$	$r_a = 40$	$r_a = 100$
Others	-	$r_{cp} = r_{cpo} = r_{av} = r_{lv} = -1$	-

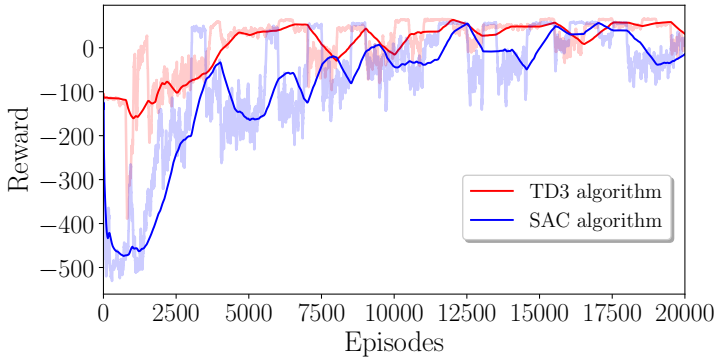
Fig. 30 presents the moving average evolution for all aforementioned reward functions, each one trained with both, TD3 and SAC algorithms. In most cases, rewards evolve side by side (with some variations) throughout training. The periodic oscillations that appear in the graphs refer to each change in the training map, where the value decreases for a while and increases according to the train evolution in the new map.

Table 6.3 compiles the results obtained for the first set of simulations using the TD3. The percentage value for each entry *scenario/method* was calculated by counting the number of trials in which the robot was capable of reaching the goal position before the timeout of 1500 steps or colliding with an obstacle after 500 attempts. For each attempt, we have set the starting and goal locations following 15 pairs of positions that were repeated during all experiments. The first observation is that our reward function outperforms all others in terms of completed missions. The results were significantly superior, although Hu et al. [2020a] also stood out from the other two. It can possibly be explained by the fact that, with the use of a safety clearance reward, the agent learns better how to avoid collisions and local minima. Additionally, the use of perception information about the increasing of the map (exteroceptive information) helped to deal with local minima problems.

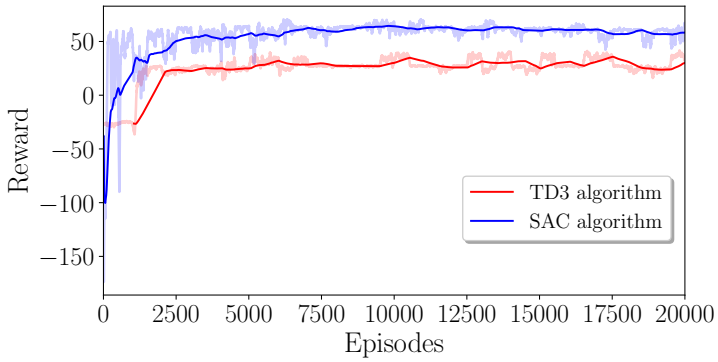
Table 6.3 – Performance comparison among our proposed reward and others from the literature using the TD3 algorithm on untrained scenarios.

Scenario/Method	Cimurs et al. [2021]	Hu et al. [2020a]	Grando et al. [2021]	Ours
Map 1 – Fig.29(a)	63.4%	81.6%	56.2%	**86.4%**
Map 2 – Fig.29(b)	51.4%	54.2%	36.6%	**57.2%**
Map 3 – Fig.29(c)	18.2%	38.4%	33.8%	**42.0%**

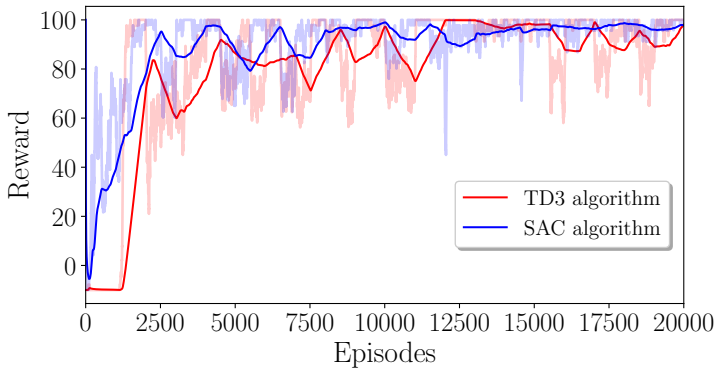
The results also allow us to conclude that Map 1 is the least complex environment, while Map 3 is the most complex regarding obstacles and local minima situations, once all reward functions follow this pattern. Still evaluating the reward functions, an outlier appears in scenario 3 when solely comparing the performance obtained by the rewards presented in [Cimurs et al.,



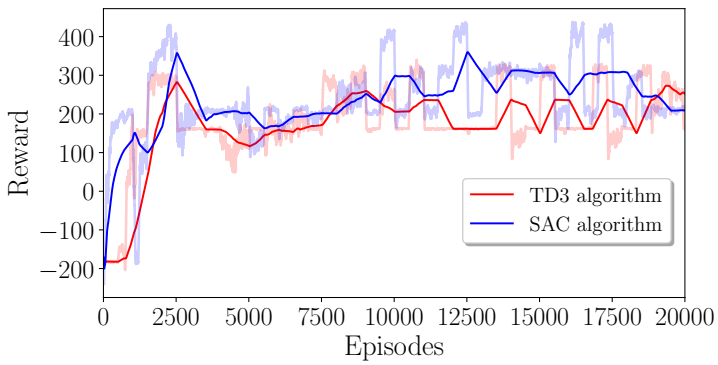
(a) Reward function in Cimurs et al. [2021].



(b) Reward function in Hu et al. [2020a].



(c) Reward function in Grando et al. [2021].



(d) Our reward function.

Figure 30 – Moving average of all compared reward functions for the TD3 (in red) and the SAC (in blue) algorithms.

2021] and [Grando et al., 2021]. Although the method presented by Cimurs et al. [2021] is better in Maps 1 and 2, the performance deteriorates sharply in Map 3, even with a simpler reward function such as the one proposed in [Grando et al., 2021].

Table 6.4 contains the results obtained from tests performed using SAC, and following the same test parameters used for TD3. Results show an increase in the number of successfully completed trials when using SAC for training compared to TD3. This performance shows that, for the robot navigation policies, increasing the exploration of actions during training helps the agent to select better actions in untrained scenarios, i.e., increasing the generalization of the trained network. Therefore, adopting exploration techniques as including the entropy maximization on training, which is done in the SAC algorithm, helps to increase the generalization.

Although most of the results present this performance increase comparing the model trained using SAC instead of TD3, the proposed reward functions presented in [Hu et al., 2020a] and [Grando et al., 2021] decreased only in Map 3. However, these results can be observed as outliers since, in all other experiments, the SAC results were better than TD3, and also maintaining the performance results observed in the different reward functions.

Table 6.4 – Performance comparison among our proposed reward and others from the literature using the SAC algorithm on untrained scenarios.

Scenario/Method	**Cimurs et al. [2021]**	**Hu et al. [2020a]**	**Grando et al. [2021]**	**Ours**
Map 1 – Fig.29(a)	80.2%	88.8%	64.8%	**95.2%**
Map 2 – Fig.29(b)	61.0%	61.6%	45.2%	**83.0%**
Map 3 – Fig.29(c)	35.6%	34.2%	31.6%	**59.4%**

Regarding the proposed reward function, some benefits can be commented on. Even when using the TD3 algorithm, our approach outperforms SAC-Cimurs et al. [2021] and SAC-Grando et al. [2021], for Map 1; SAC-Grando et al. [2021] for Map 2; and all methods for Map 3. It is slightly worse than SAC-Hu et al. [2020a], for Maps 1 and 2 and (under 4.5% difference), and than SAC-Cimurs et al. [2021], for Map 2 (under 4% difference).

When the best combinations of reward and training algorithms are compared, our approach outperforms all competitors.

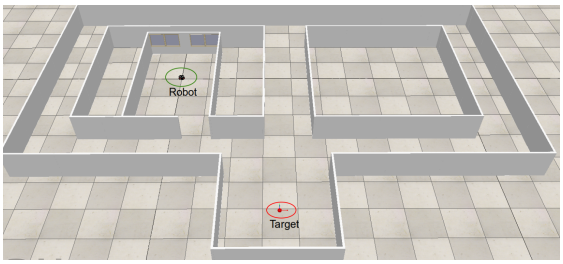
6.4 Sim-to-sim analysis

Sim-to-sim analysis is often an intermediate step to sim-to-real transfer tests since, in Robotics, collecting and validating data in the real world is more costly [James et al., 2019]. Although we have demonstrated in the previous section that our approach is more generalizable than others in the current literature in the context of robot navigation in cluttered environments, it has been done using the standalone simulator presented in Surmann et al. [2020], which is more simple in terms of dynamics, friction effects, and disturbances in general. Our choice was

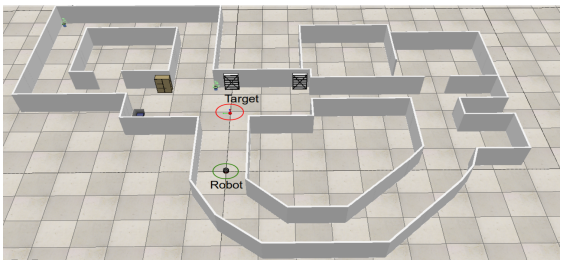
based on the fact that a simpler simulator tends to be more efficient in the training stage, allowing greater exploration of the environment in a shorter time.

Therefore, in this section, we present a comparative analysis concerning a more realistic simulator, the *CoppeliaSim*[2], integrated with Robot Operating System (ROS) Noetic in a laptop with Ubuntu 20.04, NVIDIA GTX 3060 graphics card, 16 GB of RAM, and Intel Core i7-11800H CPU. The simulations rely on a Pioneer P3DX equipped with a planar LiDAR. The 2D scans also have readings between $-135°$ to $135°$ with 684 distance measurements and decimated in 57 values of minimum distances detected.

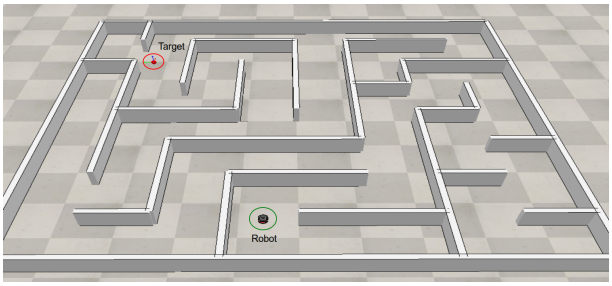
Here, the set of trials was performed in the three environments illustrated in Fig. 31, two with dimensions 40×30 m and another maze style with 24×24. In the last section, the reward function in Hu et al. [2020a] shows to be the second most efficient, then we incorporate it in this new test in two versions, one with the TD3 algorithm and another with the SAC. Also, we have limited our proposed method to the SAC method, which presented a better performance in the previous section.



(a) 40×30 m



(b) 40×30 m



(c) 24×24 m

Figure 31 – Simulated scenarios used for the test in CoppeliaSim.

Table 6.5 presents the success ratio, median of steps, and the first and third percentiles for the simulations in the CoppeliaSim using the same policies trained in the previous section. The Steps metric refers to the number of interactions with the environment performed by the agent before achieving the goal. For each method and scenario, the median, 25th percentile, and 75th percentile are presented, providing insight into the variability and efficiency of the learned behavior

[2] https://www.coppeliarobotics.com/

This time, due to the higher cost of running the simulator, we executed 200 trials for each table entry and increased the Timeout to 4000 steps due to the size and complexity of the scenarios. As seen, our method also outperforms Hu et al. [2020a], both with the TD3 and the SAC, in the two specific scenarios.

Table 6.5 – Performance comparison between the proposed learning structure and the literature best results obtained using SAC and TD3 in CoppeliaSim simulator scenarios.

Scenario	Method	Success Ratio	Steps		
			Median	25th	75th
Map 1 – Fig.31(a)	SAC-Ours	**91.0%**	554.0	515.0	818.0
	SAC-Hu et al. [2020a]	83.5%	2306	1953	2390
	TD3-Hu et al. [2020a]	75.0%	3101	2196	4000
Map 2 – Fig.31(b)	SAC-Ours	**92.5%**	629.5	374.0	1099
	SAC-Hu et al. [2020a]	76.5%	1597	1344	2196
	TD3-Hu et al. [2020a]	72.5%	2250	1925	4000
Map 3 – Fig.31(c)	SAC-Ours	**61.0%**	894.5	593.0	3812.5
	SAC-Hu et al. [2020a]	25.0%	4000	-	-
	TD3-Hu et al. [2020a]	20.0%	4000	-	-

Concerning the binomial probability of the success rate for Map 1 the SAC-Hu et al. [2020a] presents 90% confidence interval of approximately [78%, 87%], while our method falls between [87%, 94%]. Results were even better for Map 2 where the 99% confidence interval is [67%, 83%] for the SAC-Hu et al. [2020a], and [88%, 94%] for our method. Finally, for Map 3, the 99% confidence interval is [17%, 33%] for the SAC-Hu et al. [2020a], and [67%, 83%] for our method.

Table 6.5 also shows that our approach presents the lowest expected values for the number of steps until the target, demonstrating that it is more efficient than others. At the same time, as the maps are more complex, they presented a more significant variation in the distribution. This may reflect that the proposed approach actively tries to explore the environment when faced with the local minimum traps, leading to longer paths to the goal. The other methods accommodate this adversity since most of the missions fail because of timeout. The main conclusion at this point is that our proposed approach was capable of dealing better with the differences between the previous simulator (in which all policies have been trained) and the CoppeliaSim, at least for the used maps.

6.5 Sim-to-real experiments

Finally, to evaluate our proposed method, we have applied it to navigate a robot in the real world. Here, the main idea is to qualitatively evaluate the *sim-to-real* transfer capacity, which are concrete instances of the generalization problem [Kirk et al., 2021]. It is important to highlight that our evaluation is performed in a *sim-to-real zero-shot* fashion since no fine-tuning

adjustment is realized over our trained policy [Peng et al., 2018]. Two types of experiments were carried out according to the following description:

- **Comparative Analysis:** Compare the performance of the proposed method with SAC-Hu et al. [2020a] (the best-performing method from others in the previous analysis) and with a classical navigation strategy, using a zero-shot transfer to a differential robot for navigation in an office-like environment.
- **Direct Analysis:** Evaluate the performance of the proposed method in navigating an office-like environment by utilizing a zero-shot transfer to a robot with a different architecture than the one used in training.

6.5.1 Comparative Analysis: Zero-shot to Differential robot

The experiments in this section involve navigating an office-like environment, with multiple rooms and heterogeneous obstacles, to some target points. The robot selected for the task was an autonomous vacuum cleaner *Roomba Irobot Create 1* equipped with a Hokuyo URG-04LX-UG01, and a laptop with Ubuntu 20.04 and ROS Noetic, fitted with an NVIDIA GTX 3060, 16GB of RAM, and an Intel Core i7-11800H CPU.

In order to keep evaluating the comparison made in the *sim-to-sim* analysis, we performed the same *zero-shot* experiment for the method SAC-Hu et al. [2020a]. We also repeated the experiment using the Potential Field Controller (PFC) as a baseline method, a classical navigation strategy that seeks to reach goals using attractive potential fields and avoiding obstacles using repulsive fields.

Fig. 32 presents the performed experiment in a sequence of frames. Fig. 33 shows the path performed by the robot in three experiments, starting on the blue points and with the target defined to the red points. The experiments were conducted at the same place and with the same target, modifying only the control strategy and the start position. Additionally, a map was created simultaneously using the software Gmapping[3] only for visual purposes. In the experiment in Fig. 33, the robot navigates to the goals, avoiding obstacles (black points on the map) and local minima. Both goals were selected to emulate local minima issues, but the robot does not get stuck when running with the proposed method. However, the results in Fig. 33 show that the PFC method has difficulty avoiding local minima. We modified the test's start point with the PFC to make navigation easier and avoid starting in local minima.

Additionally, in the third experiment using SAC-Hu et al. [2020a], although the robot moved in the target direction, the algorithm presented difficulty in avoiding obstacles using the *zero-shot* transfer method, colliding when trying to reach the target. This behavior should be related to the generalization of the model and performance in a *zero-shot sim-to-real* transfer.

[3] Gmapping ROS - `http://wiki.ros.org/gmapping`

Figure 32 – Robot performing the real-world experiment.

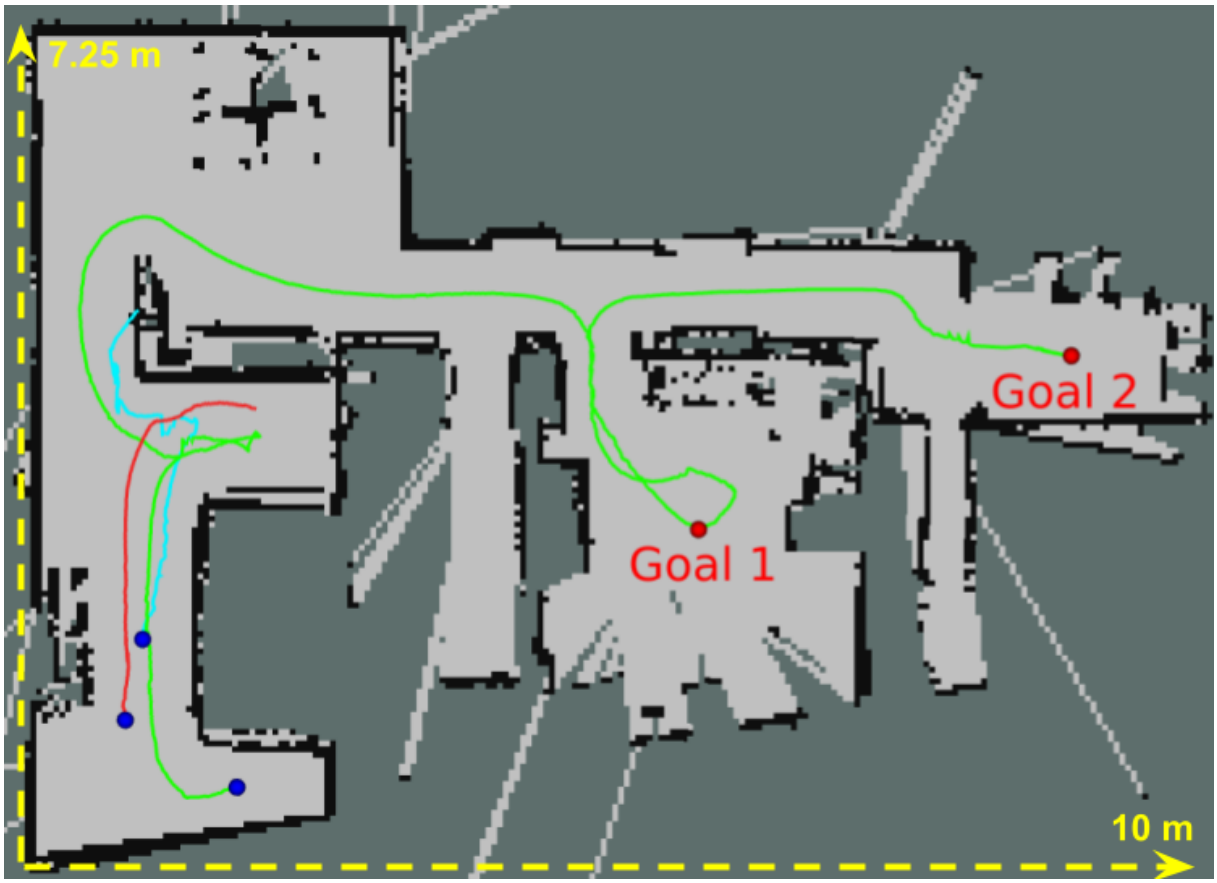


Figure 33 – Sim-to-real results of the experiments using our proposed navigation police (green path), using SAC from Hu et al. [2020a] (cyan path), and using PFC (red path) for the robot navigating in an office-like environment. The green line represents the robot's path from the starting point (blue) to the goal points (red). The grey area is the free space on the map, black points are obstacles, and dark green is the unknown area or places to be discovered.

In the first experiment, the robot navigates a distance of 21.54 m in 120 sec, the second run 3.75 m in 30 sec and gets stuck, and in the third run 6.34 m in 33 sec and finished colliding with an obstacle.

Using the proposed local navigation policy, the robot could successfully navigate toward the goals region without knowledge of the map of this indoor environment. A video illustrating the training, sim-to-sim, and sim-to-real experiments can be seen in https://youtu.be/2t_n66_uCSc.

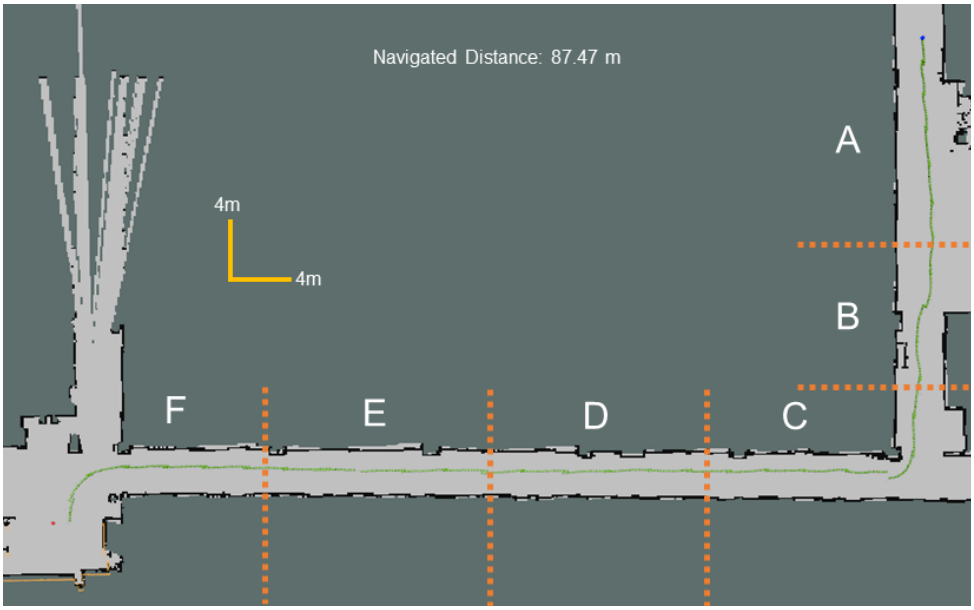
6.5.2 Direct Analysis: Zero-shot to Skid Steering robot

The experiments in this section consist of navigating through an environment formed by corridors to a target point, avoiding obstacles. The robot selected for the task was a *Pioneer 3at* equipped with a *Hokuyo UTM-30LX-EW* LiDAR and a *Jetson TX2* NVIDIA Pascal GPU architecture with 256 CUDA cores, Ubuntu 18.04 and ROS Melodic.

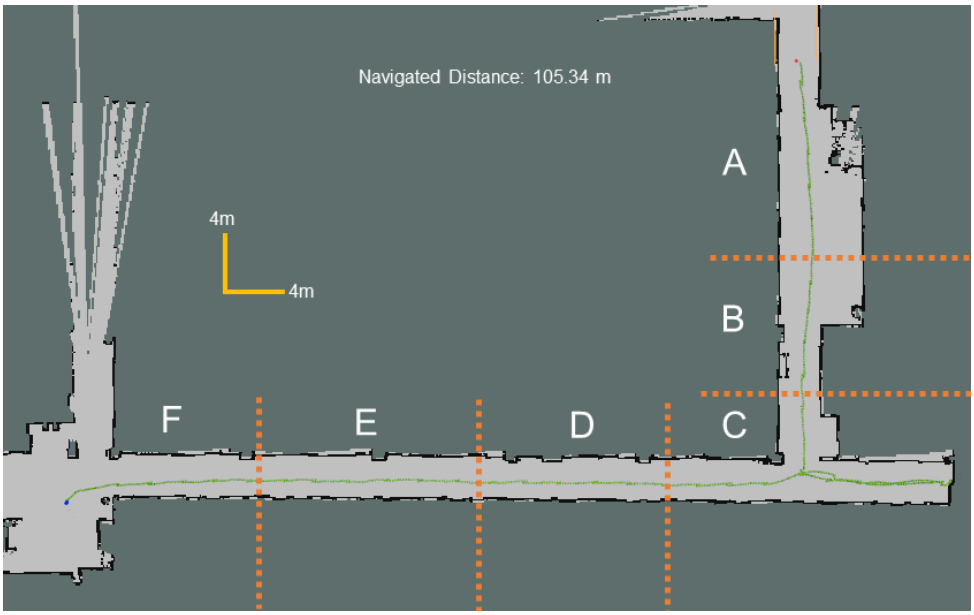
Fig. 34 presents the performed experiment in a sequence of frames. Fig. 35 shows the path performed by the robot in two experiments illustrated by the green line, starting on the blue point and finishing on the red point. The experiments were conducted at the same place, where the first simulates a going trip and the second a returning, but not departing from the exact same previous arrival point. In the experiment in Fig. 35(a), the robot goes directly to the target, avoiding obstacles (black points on the map). The results in Fig. 35(b) show that, in the first moment, the trained policy missed the passage on the left for the target and passed through straight. This failure can be due to the large curve performed by the robot to access that corridor, which may have induced the system to consider the walls as nearby obstacles and dodge them. However, the robot does not get stuck in a local minima, returning and arriving at the target.

In the first experiment, the robot navigates a distance of 87.47 m, and in the second 105.34 m. Using the proposed local navigation policy, the robot was capable of successfully navigating toward the goal region in unknown cluttered environments, avoiding collisions and local minima. A video illustrating this experiment can be seen in https://youtu.be/gufpYGcpLZA.

Figure 34 – Frame sequence of the real-world experiment.



(a) First Experiment - Going



(b) Second Experiment - Returning

Figure 35 – Sim-to-real results of the experiments using our proposed navigation police for the robot navigating in the hallways of the UFMG School of Engineering. The green line represents the robot's path from the starting point (blue) to the target point (red). The grey area is the free space on the map, black points are obstacles, and dark green is the unknown area or places to be discovered. The letters indicate the correspondent place labeled in Fig. 34, limited by the orange dash lines.

CHAPTER

7

CONCLUSIONS

This dissertation focused on the challenge of solving goal-oriented tasks for autonomous agents while effectively reducing the occurrence of local optima and providing precise guidance. The proposed approach was based on the DRL framework, where an intelligent agent is trained to take the most advantageous action to achieve the goal, guided by a reward function that represents the desired behavior. The main contribution of this research is a reward shaping function based on the PBRS principle, which modifies the reward to improve performance without affecting the policy optimality. This approach accelerates the learning process, enhances efficiency, and ensures alignment with the defined task objectives. Furthermore, the research explores how these reward functions contribute to improving the generalization capabilities of policies, enabling them to perform robustly in untrained and novel scenarios. An aspect of this methodology involves minimizing the reality gap and facilitating the transfer of policies learned through DRL from simulated environments to complex and highly dynamic real-world applications without loss of performance and extra training.

To do so, the problem was validated on robot mapless autonomous navigation, where the robot must move from an initial location to a goal using local navigation, without relying on a pre-existing map or pre-planned path. The robot is equipped with a simple planar LiDAR, and its data and the robot's position were included as inputs to the DRL framework, while the output consisted of a continuous action space defined by the robot's velocities. Considering this scenario, we proposed a novel reward shaping function designed to minimize local optimum during training, enhance exploration, accelerate learning, and improve generalization for autonomous robot navigation in both simulation and real-world environments.

The presented method utilizes exteroceptive information from LiDAR to compute the gain of environmental information during navigation, incorporating it into the reward function to encourage the agent to explore the environment while simultaneously progressing toward the goal. This approach minimizes the likelihood of the robot becoming stuck in local minima within the environment. Additionally, the novel reward function guides the agent to select actions that

ensure progressive navigation toward the goal while seeking clear paths and avoiding obstacles.

The reward shaping was formulated based on the PBRS principle, where the shaping function is defined as the difference between potential functions. We demonstrated that adding the reward shaping to the reward function does not alter the optimal policy. The proposed reward function was compared with others from the literature and showed superior performance by accelerating training convergence and increasing the number of tasks completed in a test environment after training.

Additionally, we used the SAC algorithm, which incorporates entropy regularization during the training phase to enable the agent to balance exploration and exploitation, thereby enhancing generalization. Furthermore, domain randomization was employed during training, and its impact was evaluated by comparing the performance of models trained with and without this method.

The performance analysis of the proposed method was extended by comparing the DRL model trained with our reward function to three reference methods from the literature developed for robot mapless navigation. The results demonstrate that utilizing local information to make the reward less sparse significantly improved the success rate compared to other approaches. This reward shaping strategy utilizes the map information gained during navigation to encourage exploration of new areas, while the distance to the target prevents navigation away from it. Additionally, our method required significantly fewer steps to reach the goal, highlighting its efficiency. Furthermore, when tested in untrained scenarios, the proposed strategy proved to be more effective in completing missions. Another key finding is that the model trained using SAC consistently outperformed those trained with TD3 across almost all reward functions, including both the proposed shaping strategy and those from prior studies. This supports the adoption of learning algorithms that incorporate techniques to enhance action exploration.

To further evaluate generalization capabilities, we conducted a comparative experiment using a simulator different from the one used during training. In this experiment, the model trained with our reward function was compared exclusively with the best-performing networks from the previous analysis. The results reinforced earlier findings, confirming that the proposed method outperforms the others in untrained environments, thereby demonstrating sim-to-sim generalization.

To validate the improvement in zero-shot sim-to-real transfer, we conducted real-world experiments to compare our approach with reference methods from the literature. The results confirmed the robustness of the proposed method in performing navigation tasks in untrained scenarios, even when trained in a simulated and simplified environment. Our method consistently outperformed other navigation approaches.

Finally, our method demonstrated robustness by achieving strong performance when transferred to a different robot architecture in a real-world scenario through zero-shot learning.

7.1 Future Works

Future work will focus on studying the use of asynchronous learning algorithms with the proposed reward function to optimize the training process. This includes evaluating its impact across a larger number of scenarios and assessing its influence on generalization performance.

Additionally, future research could also explore alternative methods to obtain environmental information, such as cameras, depth sensors, or Inertial Measurement Unit (IMU). Moreover, learning-based scene understanding methods, such as semantic mapping or occupancy grid prediction, could enable the agent to extract more meaningful features from its surroundings, potentially leading to more effective reward formulations.

During the experiments, whether during training or testing of the trained models, hesitant behavior was observed as the agent approached the target, leading the robot to reduce its velocity. This behavior was not addressed here and will be investigated in future work.

Another important research direction is to evaluate the proposed reward shaping function in more complex and dynamic real-world environments. Future studies could investigate its applicability in outdoor and highly dynamic scenarios, such as urban environments with moving obstacles or varying terrain conditions. Additionally, testing the method on a diverse range of robotic platforms would help assess its adaptability across different hardware configurations.

A promising extension of this work is its application to multi-agent reinforcement learning (MARL). Many real-world applications require collaboration or competition among multiple agents, such as swarm robotics or convoy navigation. Investigating how reward shaping can enhance coordination and learning efficiency in multi-agent systems could lead to more robust and scalable navigation strategies.

Finally, further efforts could be directed toward combining reward shaping with representation learning. Integrating techniques such as contrastive learning and unsupervised feature extraction could accelerate learning, improve generalization, and reduce sample inefficiency, enhancing the robustness of policies in previously unseen environments.

7.2 Publications

This section presents the papers published during the development of this Ph.D dissertation. Initially, the research focused on robot autonomous navigation, exploring techniques such as SLAM, path planning, and control. This phase of research led to the publication of [Miranda et al., 2022a] and [Miranda et al., 2022b]. The paper [Miranda et al., 2022b] received the SBA-JCAE Award in 2024, recognizing its outstanding contributions to the field. Subsequently, the research shifted to the exploration of DRL methods for mapless autonomous navigation. In this area of study, we identified the challenges associated with defining reward functions to train agents for goal-oriented tasks. This led us to investigate reward shaping and generalization

strategies, resulting in the publication of [Miranda et al., 2024], which presents the findings of this research applied to robotic mapless navigation.

Other publications were also influenced by the research conducted during the development of this dissertation. In [Rezende et al., 2020], for example, we applied Deep Learning techniques to the localization system of an autonomous navigation system for drone racing. A list of these publications, along with other related works, is presented below.

Journals

- [Miranda et al., 2024] **Victor R. F. Miranda**, Armando A. Neto, Gustavo M. Freitas, Leonardo A. Mozelli. *Generalization in Deep Reinforcement Learning for Robotic Navigation by Reward Shaping*, in **IEEE Transactions on Industrial Electronics (TIE)**, vol. 71, no. 6, pp. 6013-6020, June 2024, DOI: 10.1109/TIE.2023.3290244.

- [Miranda et al., 2022b] **Victor R. F. Miranda**, Adriano M. C. Rezende, Thiago L. Rocha, Héctor Azpúrua, Luciano C. A. Pimenta, Gustavo M. Freitas. *Autonomous Navigation System for a Delivery Drone*, in **Journal of Control, Automation and Electrical Systems**, vol. 33, pp. 141-155, 2022, DOI: 10.1007/s40313-021-00828-4.

- [Júnior et al., 2022] Gilmar P. C. Junior, Adriano M. C. Rezende, **Victor R. F. Miranda**, Rafael Fernandes, Héctor Azpúrua, Armando A. Neto, Gustavo Pessin, Gustavo M. Freitas. *EKF-LOAM: An Adaptive Fusion of LiDAR SLAM With Wheel Odometry and Inertial Data for Confined Spaces With Few Geometric Features*, in **IEEE Transactions On Automation Science And Engineering**, vol. 19, no. 3, pp. 1458-1471, 2022, DOI: 10.1109/TASE.2022.3169442 2022.

- [Rezende et al., 2020] Adriano M. C. Rezende, **Victor R. F. Miranda**, Paulo A. F. Rezeck, Héctor Azpúrua, Elerson R. S. Santos, Vinicius M. Gonçalves, Douglas G. Macharet, Gustavo M. Freitas. *An integrated solution for an autonomous drone racing in indoor environments*, in **Journal of Intelligent Service Robotics**, vol. 14, pp. 641-661, 2021, DOI: 10.1007/s11370-021-00385-4

Conferences

- [Miranda et al., 2022a] **Victor R. F. Miranda**, Armando A. Neto, Gustavo M. Freitas, Israel Amaral, Luciano C. A. Pimenta, Leonardo A. Mozelli. *Exploração Autônoma com Múltiplos Robôs Baseada em Fronteiras e Planejamento RRT**, in **Congresso Brasileiro de Automática (CBA)**, vol. 3, no. 1, 2022, DOI: 10.20906/CBA2022/3265.

BIBLIOGRAPHY

Alipanah, A. and Moosavian, S. A. A. (2022). An improvement on mapless navigation with deep reinforcement learning: A reward shaping approach. In *2022 10th RSI International Conference on Robotics and Mechatronics (ICRoM)*, pages 261–266. Cited 2 times on pages 30 and 56.

Anderson, J. (1995). *An Introduction to Neural Networks*. The MIT Press. Cited on page 42.

Arulkumaran, K., Deisenroth, M. P., Brundage, M., and Bharath, A. A. (2017). Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine*, 34(6):26–38. Cited 2 times on pages 36 and 39.

Baar, J. v., Sullivan, A., Cordorel, R., Jha, D., Romeres, D., and Nikovski, D. (2019). Sim-to-real transfer learning using robustified controllers in robotic tasks involving complex dynamics. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 6001–6007. Cited on page 26.

Badnava, B., Esmaeili, M., Mozayani, N., and Zarkesh-Ha, P. (2023). A new potential-based reward shaping for reinforcement learning agent. In *2023 IEEE 13th Annual Computing and Communication Workshop and Conference (CCWC)*, pages 01–06. Cited on page 23.

Barto, A. G. and Sutton, R. S. (1995). Reinforcement learning. *Handbook of brain theory and neural networks*, pages 804–809. Cited on page 33.

Bellman, R. (1957a). Dynamic programming princeton university press princeton. *New Jersey Google Scholar*. Cited on page 35.

Bellman, R. (1957b). A markovian decision process. *Journal of mathematics and mechanics*, pages 679–684. Cited 2 times on pages 33 and 35.

Belousov, B., Abdulsamad, H., Klink, P., Parisi, S., and Peters, J. (2021). Reinforcement learning algorithms: analysis and applications. Cited on page 49.

Berry, M. W., Mohamed, A., and Yap, B. W. (2019). *Supervised and unsupervised learning for data science*. Springer. Cited on page 41.

Bishop, C. M. and Nasrabadi, N. M. (2006). *Pattern recognition and machine learning*, volume 4. Springer. Cited on page 41.

Boeing, A. and Bräunl, T. (2012). Leveraging multiple simulators for crossing the reality gap. In *2012 12th International Conference on Control Automation Robotics Vision (ICARCV)*, pages 1113–1119. Cited on page 17.

Bohez, S., Verbelen, T., De Coninck, E., Vankeirsbilck, B., Simoens, P., and Dhoedt, B. (2017). Sensor fusion for robot control through Deep Reinforcement Learning. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2365–2370. Cited on page 27.

Carlucho, I., De Paula, M., Wang, S., Petillot, Y., and Acosta, G. G. (2018). Adaptive low-level control of autonomous underwater vehicles using Deep Reinforcement Learning. *Robotics and Autonomous Systems*, 107:71–86. Cited on page 27.

Chebotar, Y., Handa, A., Makoviychuk, V., Macklin, M., Issac, J., Ratliff, N., and Fox, D. (2019). Closing the sim-to-real loop: Adapting simulation randomization with real world experience. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8973–8979. Cited 2 times on pages 17 and 26.

Chen, G., Pan, L., Chen, Y., Xu, P., Wang, Z., Wu, P., Ji, J., and Chen, X. (2021). Deep Reinforcement Learning of map-based obstacle avoidance for mobile robot navigation. *SN Computer Science*, 2(6):1–14. Cited 2 times on pages 28 and 32.

Chi, W., Ding, Z., Wang, J., Chen, G., and Sun, L. (2022). A generalized Voronoi diagram-based efficient heuristic path planning method for RRTs in mobile robots. *IEEE Trans. Ind. Electron.*, 69(5). Cited on page 27.

Choi, J., Dance, C., Kim, J.-e., Park, K.-s., Han, J., Seo, J., and Kim, M. (2020). Fast adaptation of Deep Reinforcement Learning-based navigation skills to human preference. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 3363–3370. Cited on page 68.

Choset, H., Lynch, K. M., Hutchinson, S., Kantor, G. A., and Burgard, W. (2005). *Principles of robot motion: theory, algorithms, and implementations*. MIT press. Cited on page 15.

Cimurs, R., Suh, I. H., and Lee, J. H. (2021). Goal-driven autonomous exploration through deep reinforcement learning. *IEEE Robotics and Automation Letters*, 7(2):730–737. Cited 8 times on pages 29, 32, 57, 72, 73, 74, 75, and 76.

Cobbe, K., Klimov, O., Hesse, C., Kim, T., and Schulman, J. (2019). Quantifying generalization in Reinforcement Learning. In Chaudhuri, K. and Salakhutdinov, R., editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 1282–1289. PMLR. Cited on page 25.

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314. Cited on page 40.

Dong, K., Luo, Y., Cheng, E., Sun, Z., Zhao, L., Zhang, Q., Zhou, C., and Song, B. (2022). Balance between efficient and effective learning: Dense2sparse reward shaping for robot manipulation with environment uncertainty. In *2022 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, pages 1192–1198. Cited on page 18.

El Naqa, I. and Murphy, M. J. (2015). What is machine learning? In *machine learning in radiation oncology*, pages 3–11. Springer. Cited on page 33.

Fu, B. and Yao, X. (2024). Research and application of an improved TD3 algorithm in mobile robot environment perception and autonomous navigation. In *2024 3rd International Conference on Robotics, Artificial Intelligence and Intelligent Control (RAIIC)*, pages 158–162. Cited on page 31.

Fujimoto, S., van Hoof, H., and Meger, D. (2018). Addressing function approximation error in actor-critic methods. In Dy, J. and Krause, A., editors, *Int. Conf. on Machine Learning*, volume 80 of *Machine Learning Research*, pages 1587–1596. PMLR. Cited 2 times on pages 51 and 73.

Gao, X., Luo, H., Ning, B., Zhao, F., Bao, L., Gong, Y., Xiao, Y., and Jiang, J. (2020). RL-AKF: An adaptive Kalman Filter navigation algorithm based on Reinforcement Learning for ground vehicles. *Remote Sensing*, 12(11). Cited on page 27.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep learning*. MIT press. Cited 4 times on pages 38, 41, 43, and 44.

Grando, R. B., de Jesus, J. C., Kich, V. A., Kolling, A. H., Bortoluzzi, N. P., Pinheiro, P. M., Neto, A. A., and Drews, P. L. (2021). Deep Reinforcement Learning for mapless navigation of a hybrid aerial underwater vehicle with medium transition. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1088–1094. IEEE. Cited 7 times on pages 28, 32, 57, 73, 74, 75, and 76.

Grzes, M. and Kudenko, D. (2008). Plan-based reward shaping for reinforcement learning. In *2008 4th International IEEE Conference Intelligent Systems*, volume 2, pages 10–22–10–29. Cited on page 22.

Grzes, M. and Kudenko, D. (2009). Theoretical and empirical analysis of reward shaping in reinforcement learning. In *2009 International Conference on Machine Learning and Applications*, pages 337–344. Cited on page 22.

Gullapalli, V. and Barto, A. G. (1992). Shaping as a method for accelerating reinforcement learning. In *Proceedings of the 1992 IEEE international symposium on intelligent control*, pages 554–559. IEEE. Cited on page 53.

Gupta, A., Pacchiano, A., Zhai, Y., Kakade, S., and Levine, S. (2022). Unpacking reward shaping: Understanding the benefits of reward engineering on sample complexity. *Advances in Neural Information Processing Systems*, 35:15281–15295. Cited on page 23.

Gupta, A. K., Smith, K. G., and Shalley, C. E. (2006). The interplay between exploration and exploitation. *Academy of Management Journal*, 49(4):693–706. Cited on page 39.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. (2018a). Soft Actor-Critic: Off-policy maximum entropy Deep Reinforcement Learning with a stochastic actor. In Dy, J. and Krause, A., editors, *Int. Conf. on Machine Learning*, volume 80 of *Machine Learning Research*, pages 1861–1870. PMLR. Cited 3 times on pages 51, 52, and 63.

Haarnoja, T., Zhou, A., Hartikainen, K., Tucker, G., Ha, S., Tan, J., Kumar, V., Zhu, H., Gupta, A., Abbeel, P., et al. (2018b). Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*. Cited 2 times on pages 51 and 52.

Hansen, N. and Wang, X. (2021). Generalization in reinforcement learning by soft data augmentation. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 13611–13617. Cited on page 25.

Harutyunyan, A., Devlin, S., Vrancx, P., and Nowé, A. (2015). Expressing arbitrary reward functions as potential-based advice. In *Proceedings of the AAAI conference on artificial intelligence*, volume 29. Cited on page 22.

Hessel, M., Modayil, J., Van Hasselt, H., Schaul, T., Ostrovski, G., Dabney, W., Horgan, D., Piot, B., Azar, M., and Silver, D. (2018). Rainbow: Combining improvements in deep reinforcement learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32. Cited on page 47.

Hinton, G. E., Osindero, S., and Teh, Y.-W. (2006). A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554. Cited on page 42.

Hu, J., Niu, H., Carrasco, J., Lennox, B., and Arvin, F. (2020a). Voronoi-based multi-robot autonomous exploration in unknown environments via deep reinforcement learning. *IEEE Transactions on Vehicular Technology*, 69(12):14413–14423. Cited 10 times on pages 29, 32, 57, 73, 74, 75, 76, 77, 78, and 79.

Hu, Y., Wang, W., Jia, H., Wang, Y., Chen, Y., Hao, J., Wu, F., and Fan, C. (2020b). Learning to utilize shaping rewards: A new approach of reward shaping. *Advances in Neural Information Processing Systems*, 33:15931–15941. Cited on page 23.

Ibarz, J., Tan, J., Finn, C., Kalakrishnan, M., Pastor, P., and Levine, S. (2021). How to train your robot with Deep Reinforcement Learning: lessons we have learned. *The International Journal of Robotics Research*, 40(4-5):698–721. Cited on page 15.

Ivanov, S. and D'yakonov, A. (2019). Modern Deep Reinforcement Learning algorithms. *arXiv preprint arXiv:1906.10025*. Cited 2 times on pages 33 and 38.

Jäger, J., Helfenstein, F., and Scharf, F. (2021). Bring color to deep Q-Networks: Limitations and improvements of DQN leading to rainbow DQN. In *Reinforcement Learning Algorithms: Analysis and Applications*, pages 135–149. Springer. Cited on page 47.

James, S., Wohlhart, P., Kalakrishnan, M., Kalashnikov, D., Irpan, A., Ibarz, J., Levine, S., Hadsell, R., and Bousmalis, K. (2019). Sim-to-Real via Sim-to-Sim: Data-efficient robotic grasping via randomized-to-canonical adaptation networks. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12619–12629. Cited on page 76.

Jia, J., Xing, X., and Chang, D. E. (2022). Gru-attention based TD3 network for mobile robot navigation. In *2022 22nd International Conference on Control, Automation and Systems (ICCAS)*, pages 1642–1647. Cited 2 times on pages 29 and 32.

Jiang, H., Wang, H., Yau, W.-Y., and Wan, K.-W. (2020). A brief survey: Deep Reinforcement Learning in mobile robot navigation. In *2020 15th IEEE Conference on Industrial Electronics and Applications (ICIEA)*, pages 592–597. Cited on page 27.

Jiang, R., Cheng, X., Sang, H., Wang, Z., Zhou, Y., and He, B. (2024a). GTHSL: A goal-task-driven hierarchical sharing learning method to learn long-horizon tasks autonomously. *IEEE Transactions on Industrial Electronics*, pages 1–12. Cited on page 27.

Jiang, Y., Kolter, J. Z., and Raileanu, R. (2024b). On the importance of exploration for generalization in reinforcement learning. *Advances in Neural Information Processing Systems*, 36. Cited on page 25.

Júnior, G. P. C., Rezende, A. M. C., Miranda, V. R. F., Fernandes, R., Azpúrua, H., Neto, A. A., Pessin, G., and Freitas, G. M. (2022). EKF-LOAM: An adaptive fusion of LiDAR SLAM with wheel odometry and inertial data for confined spaces with few geometric features. *IEEE Transactions on Automation Science and Engineering*, 19:1458–1471. Cited on page 87.

Kaufmann, E., Bauersfeld, L., Loquercio, A., Müller, M., Koltun, V., and Scaramuzza, D. (2023). Champion-level drone racing using deep reinforcement learning. *Nature*, 620(7976):982–987. Cited on page 31.

Kayakoku, H., Guzel, M. S., Bostanci, E., Medeni, I. T., and Mishra, D. (2021). A novel behavioral strategy for RoboCode platform based on deep q-learning. *Complexity*, 2021. Cited on page 27.

Kirk, R., Zhang, A., Grefenstette, E., and Rocktäschel, T. (2021). A survey of generalisation in Deep Reinforcement Learning. *arXiv preprint arXiv:2111.09794*. Cited 4 times on pages 25, 38, 62, and 78.

Konda, V. and Tsitsiklis, J. (1999). Actor-critic algorithms. *Advances in neural information processing systems*, 12. Cited on page 49.

Ladosz, P., Weng, L., Kim, M., and Oh, H. (2022). Exploration in Deep Reinforcement Learning: A survey. *Information Fusion*, 85:1–22. Cited 2 times on pages 25 and 39.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *nature*, 521(7553):436–444. Cited on page 43.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324. Cited on page 43.

Li, H., Zhang, Q., and Zhao, D. (2019). Deep Reinforcement Learning-based automatic exploration for navigation in unknown environment. *IEEE transactions on neural networks and learning systems*, 31(6):2064–2076. Cited on page 29.

Li, Z., Li, X., Hu, J., and Liu, X. (2025). Mapless autonomous navigation for UGV in cluttered off-road environment with the guidance of wayshowers using deep reinforcement learning. *Applied Intelligence*, 55(3):1–25. Cited on page 31.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. (2015). Continuous control with Deep Reinforcement Learning. *arXiv preprint arXiv:1509.02971*. Cited 4 times on pages 46, 49, 51, and 73.

Liu, L., Dugas, D., Cesari, G., Siegwart, R., and Dubé, R. (2020). Robot navigation in crowded environments using Deep Reinforcement Learning. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5671–5677. Cited 2 times on pages 28 and 32.

Lou, M., Yang, X., Hu, J., Shen, H., Xu, B., Xiang, Z., and Zhang, B. (2024). Design and field test of collision avoidance method with prediction for usvs: A deep deterministic policy gradient approach. *IEEE Internet of Things Journal*, pages 1–1. Cited on page 31.

Luo, T., Subagdja, B., Wang, D., and Tan, A.-H. (2019). Multi-agent collaborative exploration through graph-based Deep Reinforcement Learning. In *2019 IEEE International Conference on Agents (ICA)*, pages 2–7. IEEE. Cited on page 29.

Machado, R. I., Machado, M. D. S., and Da Costa Botelho, S. S. (2023). Enhancing crane handling safety: A deep deterministic policy gradient approach to collision-free path planning. In *2023 IEEE 21st International Conference on Industrial Informatics (INDIN)*, pages 1–6. Cited on page 15.

Majid, A. Y., van Rietbergen, T., and Prasad, R. V. (2024). Challenging conventions towards reliable robot navigation using deep reinforcement learning. *Computing&AI Connect*, 1(1):1–10. Cited 2 times on pages 30 and 32.

Mataric, M. J. (1994). Reward functions for accelerated learning. In *Machine learning proceedings 1994*, pages 181–189. Elsevier. Cited on page 53.

McCulloch, W. S. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133. Cited on page 40.

Miranda, V. R., Neto, A. A., Freitas, G. M., Amaral, I., Pimenta, L. C., and Mozelli, L. A. (2022a). Exploração autônoma com múltiplos robôs baseada em fronteiras e planejamento RRT. In *Congresso Brasileiro de Automática-CBA*, volume 3. Cited 2 times on pages 86 and 87.

Miranda, V. R., Rezende, A. M., Rocha, T. L., Azpúrua, H., Pimenta, L. C., and Freitas, G. M. (2022b). Autonomous navigation system for a delivery drone. *Journal of Control, Automation and Electrical Systems*, 33. Cited 2 times on pages 86 and 87.

Miranda, V. R. F., Neto, A. A., Freitas, G. M., and Mozelli, L. A. (2024). Generalization in deep reinforcement learning for robotic navigation by reward shaping. *IEEE Transactions on Industrial Electronics*, 71(6). Cited 2 times on pages 72 and 87.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. (2013). Playing ATARI with Deep Reinforcement Learning. *NIPS*. Cited on page 46.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. (2015). Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533. Cited on page 46.

Montero, E., Pico, N., Ghergherehchi, M., and Song, H. S. (2025). Memory-driven deep-reinforcement learning for autonomous robot navigation in partially observable environments. *Engineering Science and Technology, an International Journal*, 62:101942. Cited on page 31.

Murphy, K. P. (2012). *Machine learning: a probabilistic perspective*. MIT press. Cited on page 42.

Ng, A. Y., Harada, D., and Russell, S. (1999). Policy invariance under reward transformations: Theory and application to reward shaping. In *Icml*, volume 99, pages 278–287. Citeseer. Cited 5 times on pages 18, 22, 53, 57, and 60.

Niroui, F., Zhang, K., Kashino, Z., and Nejat, G. (2019). Deep reinforcement learning robot for search and rescue applications: Exploration in unknown cluttered environments. *IEEE Robotics and Automation Letters*, 4(2):610–617. Cited on page 29.

Okudo, T. and Yamada, S. (2023). Learning potential in subgoal-based reward shaping. *IEEE Access*, 11:17116–17137. Cited on page 24.

Olayemi, K. B., Van, M., McLoone, S., McIlvanna, S., Sun, Y., Close, J., and Nguyen, N. M. (2023). The impact of lidar configuration on goal-based navigation within a deep reinforcement learning framework. *Sensors*, 23(24):9732. Cited 2 times on pages 29 and 32.

Onori, G., Shahid, A. A., Braghin, F., and Roveda, L. (2024). Adaptive optimization of hyper-parameters for robotic manipulation through evolutionary reinforcement learning. *Journal of Intelligent & Robotic Systems*, 110(3):108. Cited on page 24.

Packer, C., Gao, K., Kos, J., Krähenbühl, P., Koltun, V., and Song, D. (2018). Assessing generalization in Deep Reinforcement Learning. *arXiv preprint arXiv:1810.12282*. Cited on page 25.

Peng, X. B., Andrychowicz, M., Zaremba, W., and Abbeel, P. (2018). Sim-to-Real transfer of robotic control with dynamics randomization. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3803–3810. Cited 2 times on pages 26 and 79.

Plappert, M., Houthooft, R., Dhariwal, P., Sidor, S., Chen, R. Y., Chen, X., Asfour, T., Abbeel, P., and Andrychowicz, M. (2017). Parameter space noise for exploration. *arXiv preprint arXiv:1706.01905*. Cited on page 39.

Randløv, J. and Alstrøm, P. (1998). Learning to drive a bicycle using reinforcement learning and shaping. In *ICML*, volume 98, pages 463–471. Cited 3 times on pages 53, 54, and 57.

Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2016). You Only Look Once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Cited on page 28.

Rezende, A. M. C., Júnior, G. P. C., Fernandes, R., Miranda, V. R. F., Azpúrua, H., Pessin, G., and Freitas, G. M. (2020). Indoor localization and navigation control strategies for a mobile robot designed to inspect confined environments. In *IEEE Int. Conf. Autom. Sci. and Eng. (CASE)*. Cited 2 times on pages 27 and 87.

Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386. Cited on page 40.

Ruan, X., Ren, D., Zhu, X., and Huang, J. (2019). Mobile robot navigation based on Deep Reinforcement Learning. In *2019 Chinese Control And Decision Conference (CCDC)*, pages 6174–6178. Cited 2 times on pages 28 and 32.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *nature*, 323(6088):533–536. Cited 2 times on pages 40 and 41.

Rummery, G. A. and Niranjan, M. (1994). *On-line Q-learning using connectionist systems*, volume 37. University of Cambridge, Department of Engineering Cambridge, UK. Cited on page 37.

S, S., Nandesh, D. A., Raman K, R., K, G., and R, R. (2021). An investigation of bug algorithms for mobile robot navigation and obstacle avoidance in two-dimensional unknown static environments. In *Int. Conf. Commun. Inf. and Comput. Technol. (ICCICT)*. Cited on page 27.

Scarponi, V., Duprez, M., Nageotte, F., and Cotin, S. (2024). A zero-shot reinforcement learning strategy for autonomous guidewire navigation. *International Journal of Computer Assisted Radiology and Surgery*, pages 1–8. Cited 2 times on pages 17 and 26.

Sewak, M. (2019). *Deep reinforcement learning*. Springer. Cited 5 times on pages 33, 34, 35, 46, and 48.

Sfeir, J., Saad, M., and Saliah-Hassane, H. (2011). An improved artificial potential field approach to real-time mobile robot path planning in an unknown environment. In *IEEE Int. Symp. Robot. and Sens. Environ. (ROSE)*. Cited on page 27.

Shahid, A. A., Piga, D., Braghin, F., and Roveda, L. (2022). Continuous control actions learning and adaptation for robotic manipulation through reinforcement learning. *Auton. Robots*, 46(3). Cited on page 24.

Shorten, C. and Khoshgoftaar, T. M. (2019). A survey on image data augmentation for Deep Learning. *Journal of big data*, 6(1):1–48. Cited 2 times on pages 18 and 25.

Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., and Riedmiller, M. (2014). Deterministic policy gradient algorithms. In *International conference on machine learning*, pages 387–395. Pmlr. Cited 2 times on pages 49 and 50.

Skinner, B. F. (1965). *Science and human behavior*. Number 92904. Simon and Schuster. Cited on page 53.

Surmann, H., Jestel, C., Marchel, R., Musberg, F., Elhadj, H., and Ardani, M. (2020). Deep reinforcement learning for real autonomous mobile robot navigation in indoor environments. *arXiv preprint arXiv:2005.13857*. Cited 2 times on pages 65 and 76.

Sutton, R. S. (1988). Learning to predict by the methods of temporal differences. *Machine learning*, 3:9–44. Cited on page 36.

Sutton, R. S. (2018). Reinforcement learning: An introduction. *A Bradford Book*. Cited 11 times on pages 33, 34, 35, 36, 37, 39, 41, 48, 49, 50, and 53.

Tai, L., Paolo, G., and Liu, M. (2017). Virtual-to-real Deep Reinforcement Learning: Continuous control of mobile robots for mapless navigation. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 31–36. Cited on page 26.

Tobin, J., Fong, R., Ray, A., Schneider, J., Zaremba, W., and Abbeel, P. (2017). Domain randomization for transferring deep neural networks from simulation to the real world. In

2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 23–30. Cited 2 times on pages 18 and 25.

Trott, A., Zheng, S., Xiong, C., and Socher, R. (2019). Keeping your distance: Solving sparse reward tasks using self-balancing shaped rewards. *Advances in Neural Information Processing Systems*, 32. Cited 2 times on pages 23 and 58.

Vacaro, J., Marques, G., Oliveira, B., Paz, G., Paula, T., Staehler, W., and Murphy, D. (2019). Sim-to-real in reinforcement learning for everyone. In *2019 Latin American Robotics Symposium (LARS)*, pages 305–310. Cited on page 26.

Van Hasselt, H., Guez, A., and Silver, D. (2016). Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30. Cited on page 47.

Wada, D., Araujo-Estrada, S., and Windsor, S. (2022). Sim-to-real transfer for fixed-wing uncrewed aerial vehicle: Pitch control by high-fidelity modelling and domain randomization. *IEEE Robotics and Automation Letters*, 7(4):11735–11742. Cited 3 times on pages 17, 25, and 63.

Walker, O., Vanegas, F., Gonzalez, F., and Koenig, S. (2019). A deep reinforcement learning framework for uav navigation in indoor environments. In *2019 IEEE Aerospace Conference*, pages 1–14. Cited on page 30.

Wang, K., Kang, B., Shao, J., and Feng, J. (2020). Improving generalization in reinforcement learning with mixture regularization. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H., editors, *Advances in Neural Information Processing Systems*, volume 33, pages 7968–7978. Curran Associates, Inc. Cited 2 times on pages 18 and 25.

Wang, Y., Xie, Y., Xu, D., Shi, J., Fang, S., and Gui, W. (2025). Heuristic dense reward shaping for learning-based map-free navigation of industrial automatic mobile robots. *ISA Transactions*, 156:579–596. Cited on page 31.

Wang, Z., Schaul, T., Hessel, M., Hasselt, H., Lanctot, M., and Freitas, N. (2016). Dueling network architectures for deep reinforcement learning. In *International conference on machine learning*, pages 1995–2003. PMLR. Cited on page 48.

Watkins, C. J. and Dayan, P. (1992). Q-learning. *Machine learning*, 8:279–292. Cited on page 37.

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8:229–256. Cited on page 49.

Witty, S., Lee, J. K., Tosch, E., Atrey, A., Clary, K., Littman, M. L., and Jensen, D. (2021). Measuring and characterizing generalization in Deep Reinforcement Learning. *Applied AI Letters*, 2(4):e45. Cited on page 25.

Wu, K., Wang, H., Esfahani, M. A., and Yuan, S. (2021). Learn to navigate autonomously through deep reinforcement learning. *IEEE Trans. Ind. Electron.*, 69(5). Cited on page 28.

Wu, S., Tian, D., Duan, X., Zhou, J., Zhao, D., and Cao, D. (2024). Continuous decision-making in lane changing and overtaking maneuvers for unmanned vehicles: A risk-aware reinforcement learning approach with task decomposition. *IEEE Transactions on Intelligent Vehicles*, 9(4):4657–4674. Cited on page 31.

Xu, D., Chen, P., Zhou, X., Wang, Y., and Tan, G. (2024). Deep reinforcement learning based mapless navigation for industrial amrs: advancements in generalization via potential risk state augmentation. *Applied Intelligence*, pages 1–18. Cited 4 times on pages 30, 31, 32, and 57.

Xu, Z., Tang, C., and Tomizuka, M. (2018). Zero-shot deep reinforcement learning driving policy transfer for autonomous vehicles based on robust control. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 2865–2871. Cited on page 26.

Yang, H., Qi, J., Miao, Y., Sun, H., and Li, J. (2019). A new robot navigation algorithm based on a double-layer ant algorithm and trajectory optimization. *IEEE Trans. Ind. Electron.*, 66(11). Cited on page 27.

You, C., Lu, J., Filev, D., and Tsiotras, P. (2019). Advanced planning for autonomous vehicles using reinforcement learning and deep inverse Reinforcement Learning. *Robotics and Autonomous Systems*, 114:1–18. Cited 2 times on pages 27 and 28.

Zhang, L., Lin, Z., Wang, J., and He, B. (2020). Rapidly-exploring random trees multi-robot map exploration under optimization framework. *Robot. Auton. Syst.*, 131. Cited on page 27.

Zhao, W., Queralta, J. P., and Westerlund, T. (2020). Sim-to-real transfer in deep reinforcement learning for robotics: a survey. In *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 737–744. Cited on page 24.

Zhu, K. and Zhang, T. (2021). Deep Reinforcement Learning based mobile robot navigation: A review. *Tsinghua Science and Technology*, 26(5):674–691. Cited on page 27.

Zhu, W. and Hayashibe, M. (2022). A hierarchical deep reinforcement learning framework with high efficiency and generalization for fast and safe navigation. *IEEE Trans. Ind. Electron.* Cited on page 28.

Ziebart, B. D., Maas, A. L., Bagnell, J. A., Dey, A. K., et al. (2008). Maximum entropy inverse reinforcement learning. In *Aaai*, volume 8, pages 1433–1438. Chicago, IL, USA. Cited on page 52.

Zieliński, P. and Markowska-Kaczmar, U. (2021). 3D robotic navigation using a vision-based deep reinforcement learning model. *Applied Soft Computing*, 110:107602. Cited on page 28.

Zou, H., Ren, T., Yan, D., Su, H., and Zhu, J. (2021). Learning task-distribution reward shaping with meta-learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 11210–11218. Cited on page 24.