UNIVERSIDADE FEDERAL DE VIÇOSA

Q-Learning-Based Unmanned Ground Vehicle Navigation in Warehouse-Like Environments

Hiago de Oliveira Braga Batista
Magister Scientiae

VIÇOSA - MINAS GERAIS
2025

HIAGO DE OLIVEIRA BRAGA BATISTA

Q-Learning-Based Unmanned Ground Vehicle Navigation in Warehouse-Like Environments

Dissertation submitted to the Computer Science Graduate Program of the Universidade Federal de Viçosa in partial fulfillment of the requirements for the degree of *Magister Scientiae*.

Adviser: Alexandre Santos Brandao

VIÇOSA - MINAS GERAIS
2025

Ficha catalográfica elaborada pela Biblioteca Central da Universidade Federal de Viçosa - Campus Viçosa

T
B333u
2025

Batista, Hiago de Oliveira Braga, 1999-
Q-learning-based unmanned ground vehicle navigation in warehouse-like environments / Hiago de Oliveira Braga Batista. – Viçosa, MG, 2025.
1 dissertação eletrônica (60 f.): il. (algumas color.).

Texto em inglês.
Orientador: Alexandre Santos Brandão.
Dissertação (mestrado) - Universidade Federal de Viçosa, Departamento de Informática, 2025.
Referências bibliográficas: f. 58-60.
DOI: https://doi.org/10.47328/ufvbbt.2025.488
Modo de acesso: World Wide Web.

1. Aprendizado do computador. 2. Robótica. 3. Bibliotecas - Automação. 4. Armazens gerais - Automação. I. Brandão, Alexandre Santos, 1982-. II. Universidade Federal de Viçosa. Departamento de Informática. Programa de Pós-Graduação em Ciência da Computação. III. Título.

CDD 22. ed. 006.31

Bibliotecário(a) responsável: Euzébio Luiz Pinto CRB-6/3317

HIAGO DE OLIVEIRA BRAGA BATISTA

Q-Learning-Based Unmanned Ground Vehicle Navigation in Warehouse-Like Environments

Dissertation submitted to the Computer Science Graduate Program of the Universidade Federal de Viçosa in partial fulfillment of the requirements for the degree of *Magister Scientiae*.

APPROVED: March 28, 2025.

Assent:

Hiago de Oliveira Braga Batista
Author

Alexandre Santos Brandao
Adviser

Dedico este trabalho aos meus pais, Oucione e Jairo, à minha avó Vera (in memoriam) e às minhas irmãs, Isabella e Iasmin, pelo amor incondicional que me deu forças para chegar até aqui. Aos meus amigos, em especial Celso, Werikson e Mateus, e aos colegas do NERo, pela jornada compartilhada. Aos meus orientadores, Alexandre e Kevin, pela confiança e pela orientação essencial. A todos vocês, minha eterna gratidão.

ACKNOWLEDGMENTS

Gostaria de agradecer a todas as pessoas que me apoiaram durante o processo de pesquisa e escrita deste trabalho. Em especial, quero agradecer aos meus pais - Oucione e Jairo - e minha falecida avó Vera, que foram incansáveis fontes de amor, incentivo e apoio. Obrigado por acreditarem em mim e por estarem sempre presentes para me dar forças quando mais precisei. Sem a ajuda de vocês, eu não teria conseguido completar este trabalho. Também sou grato às minhas irmãs, Isabella e Iasmin, por estarem presentes e proporcionarem bons momentos divertidos, alegres e que me motivam a nunca desistir. Vocês são as pessoas mais importantes da minha vida e sou eternamente grato por tudo que fazem por mim. Obrigado por tudo, Oucione, Jairo, Vera, Isabella e Iasmin. Eu amo vocês!

Gostaria de agradecer aos meus amigos - Guilherme, João, Thayron, Gabriel e Wallace - e membros do Núcleo de Especialização em Robótica (NERo) pelo apoio, amizade e colaboração durante todo o tempo em que estivemos juntos. Em especial, quero agradecer ao Celso, por toda essa jornada desde 2019, durante a graduação em Engenharia Elétrica, pelas noites em bares (Registrada, Murato e, mais recente, o Churrascar) onde tivemos momentos de diversão e até mesmo de tomadas de decisão na vida acadêmica, pessoal e profissional.

Ao Werikson, gostaria de agradecer pela amizade, desde 2018, entramos juntos na graduação e estamos saindo juntos do Mestrado em Ciência da Computação. Me lembro muito dos sábados de manhã jogando bola na quadra da UFV, depois nos projetos do pelu, society e por fim as quartas 18h30. Agradeço pelas nossas conversas nas 4 pilatras que de 5 minutos se transformaram em 4 horas.

Ao Mateus, gostaria de agradecer também por toda a amizade e parceria que tivemos juntos, seja no NERo, BDP, graduação, mestrado e até no futebol, na vitória e na derrota. Muito obrigado por fazer parte disso.

Obrigado por tudo, Celso, Werikson e Mateus. Vocês são pessoas maravilhosas, e espero que continuemos juntos nessa jornada de aprendizado e desenvolvimento profissional.

Aos meus orientadores, Alexandre e Kevin, sou grato pelo apoio, orientação e incentivos durante todo o processo de pesquisa e escrita deste trabalho. O conhecimento, a experiência e a dedicação de ambos foram fundamentais para o sucesso deste projeto. Agradeço também pelo tempo e atenção dedicados à revisão e orientação das minhas ideias e textos, pelas valiosas sugestões e críticas construtivas, e pela confiança depositada em mim. Obrigado por tudo, Alexandre e Kevin. Vocês são excelentes professores e profissionais, e tenho muito orgulho e gratidão por serem meus orientadores.

Gostaria de agradecer ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) pelo apoio financeiro concedido a este projeto de pesquisa. O financiamento do CNPq foi fundamental para que pudéssemos realizar as simulações e as análises necessárias para a conclusão deste trabalho. Agradeço também a toda a equipe do CNPq pelo profissionalismo e dedicação na análise e avaliação do projeto. Obrigado pelo valioso apoio ao desenvolvimento da ciência e da tecnologia no Brasil. Além disso, destaco que o presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Código de Financiamento 001 (88887.822750/2023-0).

This work has been sponsored by the following Brazilian research agencies: Coordination for the Improvement of Higher Education Personnel (CAPES; Financing code 001), Minas Gerais State Foundation for Research Aid (FAPEMIG) and National Council of Scientific and Technological Development (CNPq).

“If you don't know where you want to go, any road will get you there.”
— Lewis Carroll, Alice’s Adventures in Wonderland

ABSTRACT

BATISTA, Hiago de Oliveira Braga, M.Sc., Universidade Federal de Viçosa, March, 2025. **Q-Learning-Based Unmanned Ground Vehicle Navigation in Warehouse-Like Environments**. Adviser: Alexandre Santos Brandao.

This dissertation investigates robot navigation in logistics environments, focusing on libraries and warehouses, using the Q-learning method. To this end, three studies are presented, each applying reinforcement learning to optimize task performance and navigation efficiency. The first study employs Q-learning to enhance book organization in the library of the Federal University of Viçosa, reducing planning time and movements by 20% compared to a greedy method while achieving a 100% success rate in task completion. Meanwhile, the second study proposes an offline Q-learning approach for unmanned ground vehicles in warehouses, outperforming traditional algorithms such as Dijkstra, A-star, and Breadth-First Search, with planning speeds up to seven times faster and a reduction in turns of up to 41%. Finally, the third study extends Q-learning to multi-agent navigation in libraries, integrating transfer learning and curriculum learning. As a result, simulations indicated a 94% success rate with nine agents, along with a 73.36% reduction in task steps compared to scenarios with only one agent. Thus, this dissertation highlights the significant potential of reinforcement learning, particularly Q-learning, to enhance robotic navigation efficiency, reduce operational complexity, and optimize logistics processes in dynamic and complex environments.

Keywords: path Planning; reinforcement Learning; unmanned Ground Vehicles

RESUMO

BATISTA, Hiago de Oliveira Braga, M.Sc., Universidade Federal de Viçosa, março de 2025. **Navegação de Veículos Terrestres Não Tripulados com Base em Q-Learning em Ambientes Semelhantes a Armazéns**. Orientador: Alexandre Santos Brandao.

Esta dissertação investiga a navegação de robôs em ambientes logísticos, com foco em bibliotecas e armazéns, utilizando o método de Q-learning. Para isso, são apresentados três estudos que aplicam aprendizado por reforço visando otimizar o desempenho das tarefas e a eficiência na navegação. O primeiro utiliza Q-learning para aprimorar a organização de livros na biblioteca da Universidade Federal de Viçosa, reduzindo o tempo de planejamento e os movimentos em 20% em comparação a um método guloso, além de alcançar uma taxa de sucesso de 100% na conclusão das tarefas. Já o segundo estudo propõe uma abordagem offline de Q-learning para veículos terrestres não tripulados em armazéns, superando algoritmos tradicionais como Dijkstra, A-star e Busca em Largura, com velocidades de planejamento até sete vezes superiores e uma redução nas curvas de até 41%. Por fim, o terceiro estudo expande o Q-learning para a navegação multiagente em bibliotecas, integrando aprendizado por transferência e aprendizado curricular. Como resultado, as simulações indicaram uma taxa de sucesso de 94% com nove agentes, além de uma redução de 73,36% nas etapas das tarefas em relação a cenários com apenas um agente. Dessa forma, esta dissertação evidencia o potencial significativo do aprendizado por reforço, especialmente do Q-learning, para aumentar a eficiência da navegação robótica, reduzir a complexidade operacional e otimizar processos logísticos em ambientes dinâmicos e complexos.

Palavras-chave: planejamento de caminho; aprendizado por reforço; robótica terrestre

List of Figures

List of Tables

Contents

1 Introduction

Recent advancements in robotics and autonomous systems have spurred applications in diverse fields, from industrial automation (SANIUK; SANIUK; CAGÁŇOVÁ, 2021) to service robotics (BELANCHE et al., 2020). A key area of impact is logistics and warehouse management, where Unmanned Ground Vehicles (UGVs) are employed to optimize operations, reduce human workload, and enhance efficiency (JACOB et al., 2023). This work is specifically motivated by the challenges faced by the Central Library (BBT) at the Federal University of Viçosa (UFV), where an increasing demand for services coincides with a reduction in staff, creating a need for automated solutions to manage tasks like book retrieval and shelving. Such environments, with structured corridors and defined storage locations, serve as a practical model for broader "warehouse-like" scenarios.

Path planning is a fundamental challenge in mobile robotics. While traditional algorithms like Dijkstra's and A* search provide optimal solutions for finding the shortest path in static environments (XUE; SUN, 2018), they often lack the flexibility to handle multi-objective optimization criteria, such as minimizing path curvature to improve energy efficiency and mechanical stability. Furthermore, when a robot must visit multiple destinations, the problem evolves into a complex route optimization task, akin to the Traveling Salesman Problem (TSP) (FLOOD, 1956).

Reinforcement Learning (RL), particularly Q-Learning (QL) (WATKINS; DAYAN, 1992), offers a promising alternative. By learning through interaction, an RL agent can develop policies that balance multiple objectives, such as path length, smoothness, and task completion time, without an explicit model of the environment's dynamics. This adaptability makes it well-suited for the complexities of logistics operations.

This dissertation is presented as a compilation of three scientific articles that progressively explore the application of Q-learning to solve UGV navigation challenges. The research is focused on structured, indoor environments with dimensions and characteristics comparable to the UFV library, which serves as our primary case study. The goal is not to claim universal superiority over classic algorithms, but rather to investigate how RL-based methods can provide competitive and flexible solutions tailored to the specific demands of logistics tasks.

1.1 Theoretical Background

Reinforcement Learning (RL) is a machine learning paradigm that enables agents to learn optimal strategies through interaction with an environment, progressively improving their performance based on received rewards. This learning process is particularly

valuable in smart inventory management systems, where autonomous robots or unmanned ground vehicles (UGVs) must navigate warehouse environments efficiently, minimizing costs and maximizing task completion rates. In such scenarios, RL enables machines to adapt dynamically to logistical challenges, such as unexpected obstacles, varying demand locations, and multi-agent coordination.

At the core of RL lies the concept of decision-making through trial and error. An agent perceives the environment through a defined state space and selects actions that influence its surroundings. These actions yield numerical rewards that guide the learning process. The ultimate goal is to develop a policy that dictates the best action to take in each state to maximize long-term rewards. In the context of warehouse automation, this approach plays a crucial role in optimizing logistics operations, such as determining the most efficient routes, improving shelf organization, and streamlining order retrieval processes. By continuously refining decision-making strategies, RL-driven systems aim to enhance operational efficiency, reducing delays and resource wastage in smart inventory management.

The decision-making process in RL is often formulated as a Markov Decision Process (MDP), which provides a mathematical framework for modeling sequential decision-making problems. An MDP consists of a set of states, actions, transition probabilities, rewards, and a discount factor that determines the importance of future rewards. In a warehouse environment, the MDP framework can be used to model inventory management challenges, such as determining the optimal replenishment strategy. The state space represents the current inventory levels, order queue, and shelf organization, while the action space includes restocking decisions, item retrieval paths, and task assignments. Transition probabilities capture the likelihood of demand fluctuations, and the reward function incentivizes efficiency in fulfilling orders with minimal delays and reduced operational costs. In summary, by taking advantage of MDPs, inventory management systems can systematically optimize logistics operations, ensuring streamlined workflows and better use of resources. Thus, in this sense, this dissertation extends this approach to optimize the organization system of the UFV library, improving the placement or return of books through strategies based on reinforcement learning.

Q-learning is a specific RL algorithm that relies on value iteration to estimate the quality of state-action pairs. By maintaining a Q-table, the agent learns to associate each action with an expected cumulative reward. In a smart inventory context, this means that a robotic system can iteratively improve its decisions regarding which paths to take, when to pick up items, and how to avoid congestion. The update rule follows the Bellman equation:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[R + \gamma \max_{a'} Q(s',a') - Q(s,a) \right] \tag{1.1}$$

where α is the learning rate, γ is the discount factor, R is the immediate reward, and

$\max_{a'} Q(s', a')$ represents the best estimated future reward. In practical terms, this allows automated inventory robots to refine their movement and task execution strategies based on past experiences.

A crucial aspect of reinforcement learning is the balance between exploration and exploitation. Exploration enables the agent to discover new strategies, while exploitation takes advantage of the best-known policy to maximize immediate rewards. In an inventory management system, this trade-off is vital. If a robot only follows known routes (exploitation), it may never discover faster or safer paths. Conversely, excessive exploration could lead to inefficient operations and wasted energy. The ϵ-greedy strategy mitigates this by selecting random actions with probability ϵ and choosing the best-known action otherwise.

Smart inventory applications demand more than conventional Q-learning due to the complexity of multi-agent interactions and dynamic warehouse layouts. To enhance learning efficiency, Curriculum Learning (CL) can be introduced. This technique structures training by initially exposing agents to simpler tasks and gradually increasing the complexity. For instance, a warehouse robot may first learn to navigate empty aisles before being introduced to environments with obstacles, moving agents, and dynamically changing orders. This gradual increase in difficulty improves convergence speed and overall performance.

Curriculum Learning is particularly advantageous in collaborative warehouse environments where multiple autonomous agents operate simultaneously. Instead of training all robots in a complex warehouse setting from the beginning, they are first trained individually or in simplified versions of the real-world environment. This hierarchical learning structure allows agents to build foundational navigation and task execution skills before tackling full-scale operational challenges.

In broad terms, reinforcement learning, Q-learning, and curriculum learning form a structured hierarchy of learning strategies that enhance autonomous systems in inventory management. While reinforcement learning provides the foundational decision-making framework, Q-learning refines this process through iterative updates, and curriculum learning optimizes training by intelligently sequencing learning experiences. Their combined use ensures that smart inventory systems remain robust, scalable, and efficient in dynamic industrial environments.

1.2 Research Question and Hypotheses

The central research question guiding this work is:

How can Q-learning be effectively applied to improve the efficiency and adaptability of unmanned ground vehicles (UGVs) for path planning in warehouse-

like environments, considering multiple objectives such as path length, smoothness, and task completion time?

To address this question, the following hypotheses are proposed, framed to reflect a comparative and exploratory investigation:

- **H1:** A Q-learning approach can generate navigation paths that are competitive with those from traditional algorithms (e.g., Dijkstra), while offering greater flexibility to optimize for multiple objectives, such as reducing the number of turns.
- **H2:** The integration of Transfer Learning and Curriculum Learning with Q-learning can significantly improve training efficiency and the adaptability of UGVs in complex scenarios, leading to faster convergence and effective policy generation.
- **H3:** A multi-agent Q-learning framework can enhance the scalability and overall efficiency of logistics operations, though its effectiveness is dependent on the number of agents relative to the environment's capacity, avoiding congestion.

1.3 Contributions and Publications

The main contributions of this dissertation are encapsulated in three distinct but interconnected research articles.

1. **A Multi-Objective Q-Learning Path Planner:** We developed a Q-learning methodology for multi-goal path planning that optimizes for path length and smoothness. This work was validated through simulations modeled on the UFV library.
    - *Publication:* I. Batista et al., "Multi-Goal Robot Path Planning Based on Q-Learning for Library Logistics", presented at the Brazilian Congress of Automatica (CBA), 2022.
2. **Real-World Validation and Multi-Objective Analysis:** The methodology was implemented and validated on a physical Pioneer 3-DX UGV in a controlled laboratory environment. This study provided a comparative analysis against traditional grid-based search algorithms.
    - *Publication:* I. Batista et al., "Q-Learning-Based Multi-Objective Global Path Planning for UGV Navigation", presented at the Latin American Robotics Symposium (LARS), 2023.
3. **A Scalable Multi-Agent Framework with Curriculum and Transfer Learning:** We extended the approach to a multi-agent system, using Curriculum and

Transfer Learning to manage the immense state space and enable complex collaborative tasks across two floors of the simulated UFV library.

- *Status:* I. Batista et al., "Multi-Agent Path Planning Logistics Corporation Using Reinforcement Learning," under review for publication.

1.4 Dissertation Structure

This dissertation is based on a compilation of articles; this work is organized to provide context, present the core research, and offer extended discussion.

Chapter 1 (This Chapter) presents a unified and comprehensive theoretical foundation for the concepts employed throughout the subsequent chapters. It provides an in-depth examination of Reinforcement Learning, Markov Decision Processes (MDPs), Q-Learning, Curriculum Learning, and Transfer Learning, consolidating the fundamental knowledge required for the development of this work.

This chapter also defines the research problem, motivation, and scope of the study. Furthermore, it outlines the research question and hypotheses and establishes a clear correspondence between the proposed contributions and their respective publications.

Chapter 2: Multi-Goal and Multi-Objective Path Planning presents the work from the first two publications (CBA 2022 and LARS 2023). This chapter details the development of the single-agent Q-learning planner, from simulation to real-world validation with the Pioneer 3-DX robot.

Chapter 3: Multi-Agent Collaborative Navigation presents the third research article, focusing on the multi-agent system. It describes the use of Curriculum and Transfer Learning to address the scalability challenges of navigating a complex, two-floor environment.

Chapter 4: Discussions and Considerations offers a space for reflection and extended analysis beyond the scope of the individual articles. This chapter will address the nuances of the comparisons with traditional algorithms, discuss the limitations of the proposed methods (such as generalization), and elaborate on practical implementation challenges, providing a deeper context for the research.

Chapter 5: Concluding Remarks summarizes the findings of the dissertation, revisits the research hypotheses in light of the results, and proposes detailed directions for future work, building upon the lessons learned from this research.

2 Intelligent Path Planning for Autonomous Ground Vehicles and Library Logistics

This chapter is based on two studies published at the 2024 Latin American Robotics Symposium (LARS) (BATISTA et al., 2024b) and XXV Congresso Brasileiro de Automática (CBA) (BATISTA et al., 2024a), which explore intelligent path planning methodologies for autonomous ground vehicles within library logistics contexts. Traditional methods for robot navigation, such as Dijkstra's algorithm, A*, and Artificial Potential Fields (APF), have shown efficacy but often face challenges related to scalability, real-time performance, and adaptability in complex or dynamic environments. To address these limitations, the proposed research integrates reinforcement learning, specifically Q-Learning, to enable autonomous ground vehicles (UGVs) to efficiently navigate multi-goal scenarios within structured environments like libraries.

The primary goal of this work is to enhance autonomous navigation performance by applying machine learning techniques to the path planning problem, particularly focusing on library logistics involving multiple target destinations. The research specifically addresses challenges associated with navigating grid-based environments, where the vehicle must balance path length, energy efficiency, and maneuverability to deliver items effectively. This problem formulation considers practical constraints such as minimizing sharp turns and collisions, ensuring energy efficiency, and optimizing the robot's navigation policy to accommodate changing environmental conditions.

Experimental validations demonstrate that employing Q-Learning significantly improves the robot's ability to generate smoother and more efficient paths compared to traditional algorithms like Dijkstra and A*. The experiments confirm that the proposed Q-Learning-based navigation strategy achieves superior performance in terms of path smoothness, planning time, and real-time adaptability. Additionally, the proposed approach proves particularly advantageous for multi-objective tasks, demonstrating effectiveness in environments where conventional methods may experience difficulties due to computational overhead or lack of adaptability.

In summary, this chapter contributes to advancing autonomous navigation strategies for ground vehicles by demonstrating how Q-Learning-based methods can enhance operational efficiency and adaptability in library logistics and other structured multi-objective navigation scenarios.

2.1 Multi-Goal Robot Path Planning Based on Q-Learning

The path planning problem involves finding a collision-free route from the start to the end point (AGGARWAL; KUMAR, 2020). Various strategies have been employed in literature to address the path planning problem, including Dijkstra's Algorithm, A*, and Artificial Potential Fields (APF). First, Dijkstra's Algorithm, designed for graphs with positive weights, determines the shortest path between two nodes (DIJKSTRA, 2022). However, it requires non-negative weights to avoid path divergence. Second, the A* algorithm (HART; NILSSON; RAPHAEL, 1968) extends Dijkstra's approach by incorporating heuristic information, aiming to find the minimum-cost path in graphs. It optimizes the function $f(n) = g(n)+h(n)$, where n denotes the next node, $g(n)$ represents the cost from the start to n, and $h(n)$ is the heuristic function. Nevertheless, its effectiveness depends on the accuracy of the heuristic function in estimating the distance to the goal. Moreover, in multi-agent scenarios, A* may generate conflicting paths, potentially leading to agent collisions (FOEAD et al., 2021). Finally, potential fields path planning employs attractive and repulsive fields to guide robots to their destinations while avoiding collisions (KHATIB, 1986). While effective in many cases, this method struggles with U-shaped obstacles and narrow passages, as it can become trapped in local minima.

In the context of library scenarios, a gap exists in the state-of-the-art regarding path planning. However, various methodologies are available for warehouse path planning. For instance, Jang (1993) propose a two-stage optimization approach utilizing the A* algorithm, while Sharma e Doriya (2021) introduce an intelligent method for destination identification and collision-free path determination, demonstrating superior performance compared to both the A* algorithm and Integration Linear Programming. Although these methods address the path planning challenge, machine learning techniques have emerged as effective solutions due to their flexibility and generalization capabilities (ARINEZ et al., 2020; FAGUNDES-JUNIOR et al., 2024).

QL path planning, utilized in robotics and artificial intelligence (AI), determines optimal paths in obstacle-filled environments via reinforcement learning. The technique involves an agent navigating a grid-based environment, adjusting actions based on rewards and penalties. By iteratively updating Q-values, the agent learns to navigate efficiently, finding multiple feasible paths while avoiding obstacles to reach its destination (KHRIJI et al., 2011; HU; YANG; LOU, 2021).

Recent literature explores the use of QL for robot path planning. For instance, Carvalho et al. (2023) propose an online QL approach for Unmanned Aerial Vehicle (UAV) navigation in both static and dynamic environments. Their study demonstrates that QL outperforms traditional methods like Dijkstra's algorithm, yielding paths with fewer curves and increased distance from obstacles through iterative recomputations.

Q-learning has been effectively applied in UAV navigation, as demonstrated by

Sonny, Yeduri e Cenkeramaddi (2023). This study presents a Q-learning algorithm for UAV path planning, addressing both static and dynamic obstacle avoidance. The approach employs a grid graph-based method to optimize rewards based on the agent's behavior, demonstrating superior performance in minimizing UAV travel distance compared to existing methods.

While QL is effective for path planning, its applicability is restricted by the size of the state space. However, auxiliary techniques like Curriculum Learning can extend its use. In previous studies (OLIVEIRA; CARVALHO; BRANDãO, 2023), QL was employed for multi-agent path planning in warehouse scenarios. The study demonstrated that incorporating Transfer Learning and Curriculum Learning can achieve a success rate of up to 94% on the proposed map.

In summary, these studies demonstrate the superiority of reinforcement learning techniques, particularly QL, over traditional methods like A* and Dijkstra. Additionally, it highlights the effectiveness of auxiliary approaches such as transfer learning and curriculum learning in solving state space issues.

2.1.1 Problem Formulation

In this chapter, a Q-Learning-based approach is proposed to optimize ground robot paths for efficient book delivery on the second floor of the central library at the Federal University of Viçosa. The study area includes sections dedicated to exact and natural sciences, such as physics, mathematics, chemistry, and engineering, which are divided into five labeled areas: A, B, C, D, and E (Figure 1).

During high-demand periods, such as assessment weeks, book shortages occur due

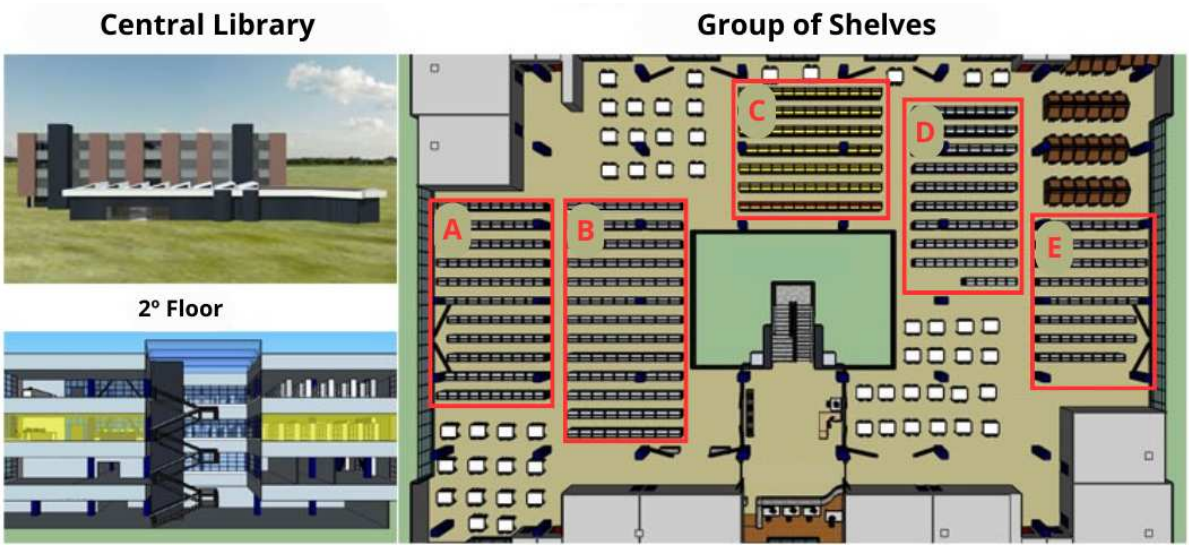


Figure 1 – Illustration of the second floor of UFV's central library with the marking of the five sets of shelves.

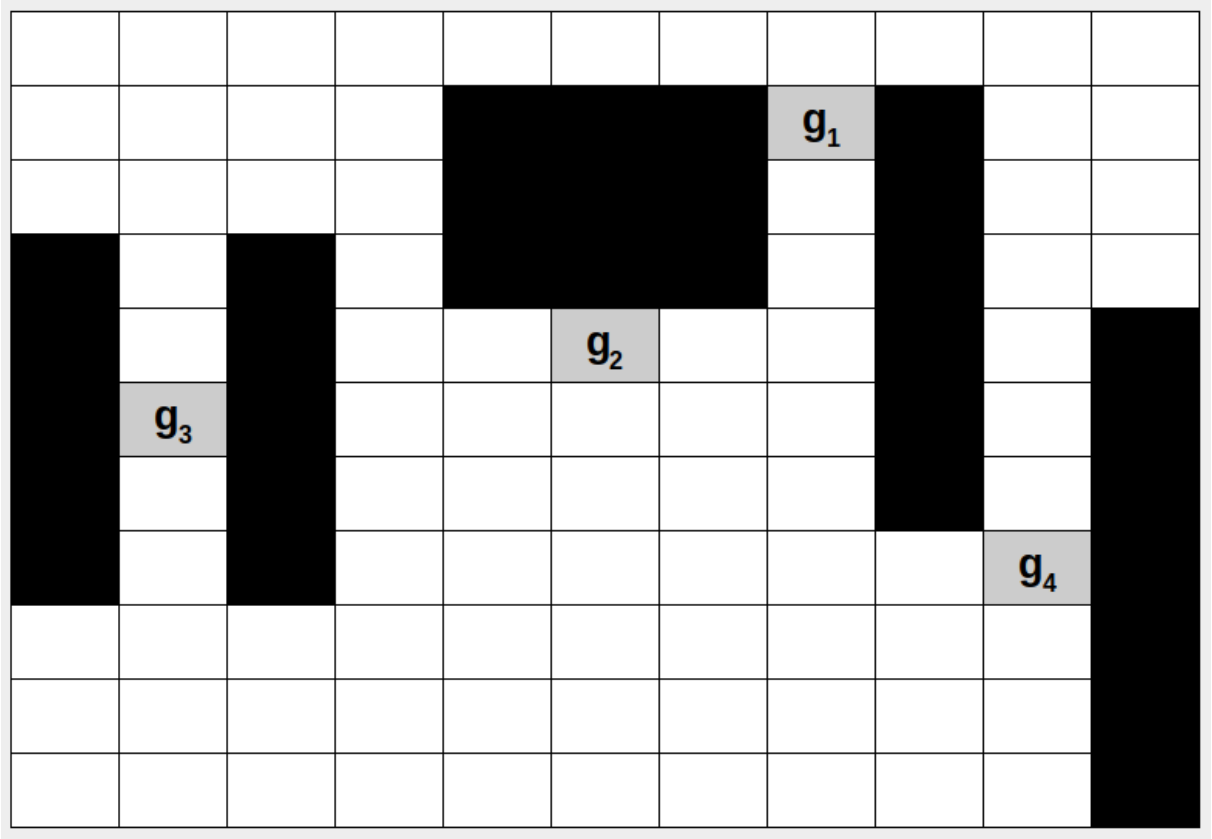


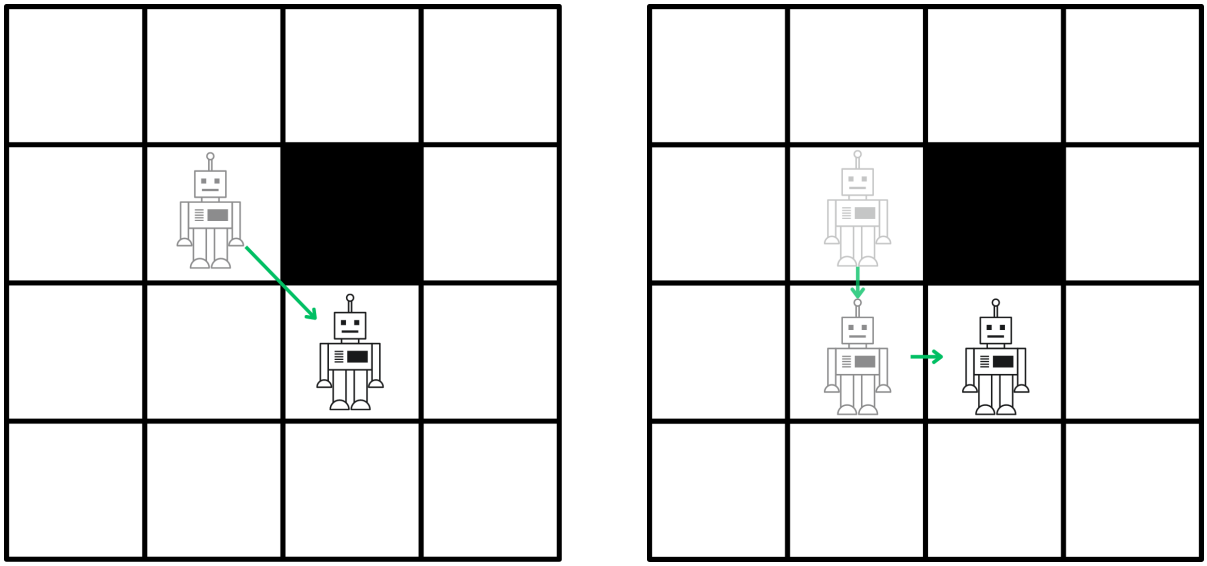
Figure 2 – Grid representation of the library's second floor. The shelves are in black, and the objectives in gray.

to staff overload in repositioning tasks. To mitigate this issue, terrestrial robotics is employed to transport books to critical areas of the library. A grid-based model of the library's second level is developed, identifying high-demand locations as g_1, g_2, g_3, and g_4 (Figure 2). The proposed solution involves staff placing books on the robot for automated delivery to these areas, enhancing efficiency in terms of time and resource utilization.

2.1.1.1 State Space Modeling

To achieve the desired outcome, the problem is modeled as a Markov Decision Process, represented by a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, R, \gamma)$. In each episode, the agent begins in a random state $\mathbf{s_n} \in \mathcal{S}$ to encourage exploration. The agent's state, $\mathbf{s} \in \mathbb{N}^7$, comprises its pose in a grid world, denoted by row (r), column (c), and orientation (ψ), where $r, c \in \{1, \cdots, 11\}$ for an 11×11 map, and $\psi \in \{0, \frac{\pi}{2}, \pi, \frac{-\pi}{2}\}$ representing its four possible orientations. Additionally, the last four elements of $\mathbf{s}$ are binary values indicating whether the agent has reached destinations g_i, where $i \in \{1, 2, 3, 4\}$. This results in a total of 7,744 distinct states. Therefore, our state is defined as:

$$\mathbf{s} = [r \ c \ \psi \ g_1 \ g_2 \ g_3 \ g_4]. \tag{2.1}$$



(a) High probability of collision on the corners.

(b) Low probability of collision on the corners.

Figure 3 – Movement that may cause a collision, followed by the desired movement.

2.1.1.2 Actions

In this study, each state $\mathbf{s_n} \in \mathcal{S}$ allows the agent to take actions $a \in \mathcal{A}$. The model restricts the agent to either linear or rotational movements due to the gridworld simulation environment, where diagonal movements may cause collisions with corner objects (see Figure 3). Hence, available actions include moving forward, rotating 90°, and rotating −90°. Consequently, the state-action pair space comprises 23,232 elements.

2.1.1.3 Reward Function

The primary aim of this agent is to efficiently reach all destination cells while favoring linear paths over curves. Each action performed by the agent yields a reward $r_t \in \mathbb{R}$, contingent upon the action and resultant state. A universal penalty of -1 is applied to each step to expedite task completion, with an additional -2 penalty for rotation actions to promote linear movements. Considering the living penalty, rotation incurs a cost of -3, while forward movement incurs a cost of -1. Therefore, forward movement is three times less costly than rotation. Furthermore, the agent receives a $+10$ reward for each goal achieved but incurs a -10 penalty for colliding with obstacles. Thus, the reward function r_t is defined as follows:

$$r_t = -1 - 2H(a) - 10O(s) + 10G(s), \tag{2.2}$$

where the function $H(a)$ can be defined as a step function that produces a value of 1 in the case where the action a taken is to turn. Similarly, the function $O(s)$ is a step function that produces a value of 1 if the state s reached as a result of the action a corresponds

to a state containing an obstacle. Finally, the function $G(s)$ is also a step function that produces a value of 1 when the state s reached by the action a is one of the designated target states.

2.1.1.4 Training parameters

In this study, hyperparameters were determined empirically through experimentation. The learning rate (α) was set to 0.1 and the discount factor (γ) to 0.99. Exploration was addressed using a linearly decreasing variable parameter (ϵ), starting at 1 and ending at 0.1 as described in

$$\epsilon_t = 1 - \frac{0.9}{80,000} \cdot t, \tag{2.3}$$

where the symbol ϵ_t represents the randomization level for episode t. This randomness decreases over time to encourage early exploration and subsequent exploitation. The exploring starts strategy was employed during training, where an initial state is randomly chosen to boost exploration and refine the navigation policy. The value of eighty thousand in (2.7) is assigned based on the number of training episodes.

Furthermore, the policy $\pi(s)$ is updated for each action determined on the basis of its Q-values. The update is performed using the Bellman equation:

$$Q(s_t, a_t) = (1 - \alpha)Q(s_t, a_t) + \alpha(r_{s_t} + \gamma \underset{a}{max} Q(s_{t+1}, a)). \tag{2.4}$$

Given the Bellman equation and relevant parameter definitions the agent is trained using the algorithm outlined in Algorithm 1.

Algoritmo 1: Q-Learning Training

```
for i ← 1 to 80,000 do
    Define starting point according to exploring starts;
    Calculate ε_t via (2.7);
    while Episode not over do
        if Random(0,1) < ε then
            a_t ← RandomAction;
        else
            a_t ← π(s) ; // Optimal action
        Calculate r_t via (2.2);
        s_{t+1} ← Step(a_t) ; // Find next state
        Q(s_t, a_t) ← Update via (2.4);
        s_t ← s_{t+1} ; // Update current state
```

2.1.2 Results and Discussion

The evaluation of the proposed methodology consists of two phases: training and path quality assessment. The algorithm was implemented in Python and is publicly available in a GitHub repository[1]. Both training and evaluation were conducted on a computer equipped with 32 GB RAM, a Ryzen 5 5600G processor, and a GeForce RTX 3080 GPU.

2.1.2.1 Training Performance

Twenty training sessions were conducted to evaluate the agent's adaptability to the environment, recording its performance. Multiple sessions are necessary to mitigate the influence of stochastic elements on decision-making, which can affect the learning pace. On average, each training session consisting of 80,000 episodes requires an average of 8.6 minutes to complete.

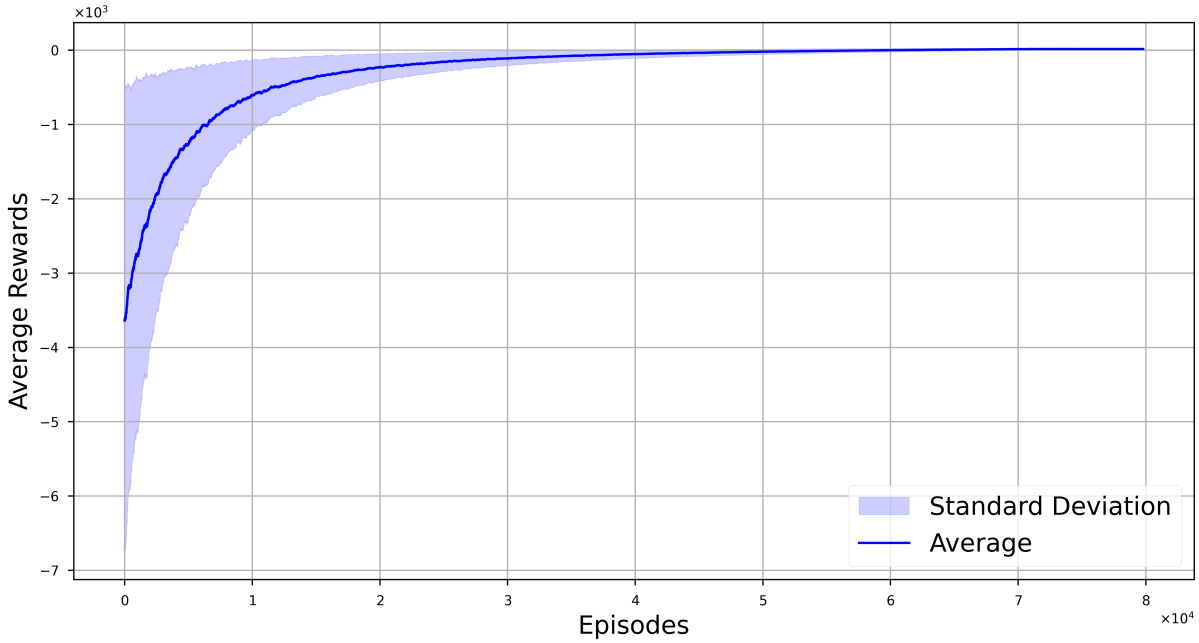


Figure 4 – Average training reward for 20 sections with 80,000 training episodes.

Throughout the training sessions, the agent demonstrated significant progress, depicted in Figure 4. Initially, it received minimal rewards due to its unfamiliarity with the environment and the randomness of its actions. However, over time, the average reward increased steadily, stabilizing at a consistent level. Notably, there was a high standard deviation at the outset of training, attributed to stochastic elements such as the use of exploring starts and the $\epsilon-$greedy strategy.

[1] <https://github.com/Hiago013/q-learning_navigation>

2.1.2.2 Path Evaluation

Path effectiveness was assessed using three metrics defined by Liu et al. (2023): path length, number of turns, and generation time. The results were compared with those obtained using Dijkstra's algorithm, which iteratively selects the nearest target cell until all targets are visited. The evaluation of the agent across these metrics was conducted for each free cell in Figure 2, assuming no prior visits to target cells.

The proposed approach yielded an average travel distance of 15.18 meters, compared to 13.63 meters using Dijkstra's technique, as shown in Figure 5a. It is a fact that Dijkstra's algorithm consistently identifies the shortest route connecting two nodes in a graph with positive edges. On average, the path produced by our approach shows an increase over Dijkstra's of 11.37%. However, practical scenarios demand consideration of factors beyond mere distance calculation, including path smoothness.

To assess path smoothness, we compared the number of turns generated by our proposed methodology with those using Dijkstra's strategy. Figure 5b illustrates this comparison. Our QL technique averages approximately 7.67 curves, while Dijkstra's method yields 9.24 curves. Furthermore, the third quartile for QL is around 8 curves, compared to around 10 curves for Dijkstra. Consequently, in approximately 75% of cases, our approach produces 20% fewer curves, enhancing vehicle maneuverability with a smoother path.

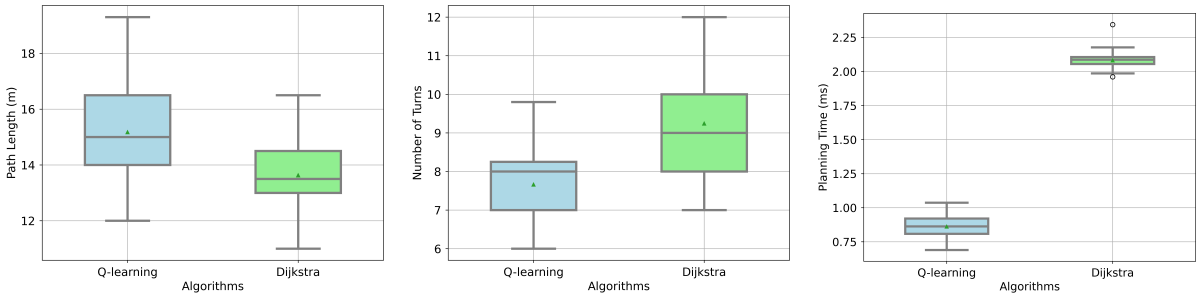
In practical scenarios, besides metrics like distance and curvature, assessing the algorithm's speed in generating obstruction-free paths is crucial. We record the time our algorithm takes to establish paths, as depicted in Figure 5c. Our method typically produces viable paths in 0.86 ms, compared to an average of 2.08 ms for the Dijkstra-based approach. This difference arises from our algorithm's $O(n)$ time complexity, whereas Dijkstra's algorithm operates at $O(n^2 log(n))$, requiring additional iterations until all destinations are covered.

QL demonstrates superior performance compared to the shortest path method in terms of turns and planning time, though it falls slightly behind in path length. Both methods achieve a 100% success rate in finding obstacle-free paths, as indicated in Table 1.

The length of the path does not always correlate with the time required for completion, considering the rotations made by the agent. For a demonstration of Dijkstra-based

Table 1 – Comparison of the average and standard deviation performance of the proposed approach (QL) against the baseline (Dijkstra) for metrics including path length, number of turns, planning time, and success rate.

Path	**Path Length (m)**	**# of Turns**	**Planning Time (ms)**	**Success Rate %**
QL	15.18 ± 1.54	7.67 ± 1.00	0.86 ± 0.07	100.00
Dijkstra	13.63 ± 1.08	9.25 ± 1.47	2.08 ± 0.04	100.00

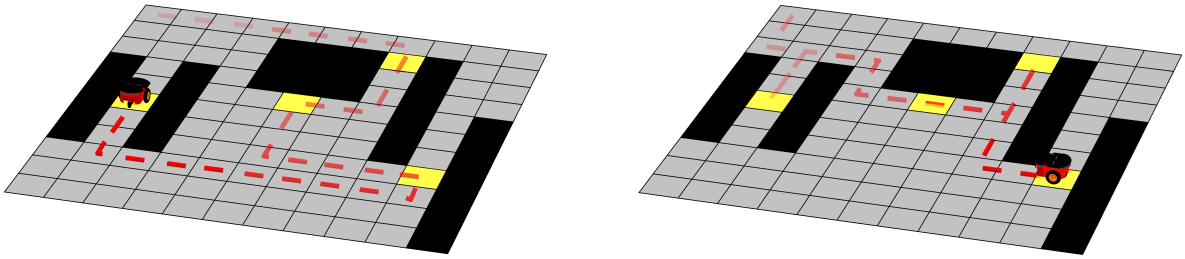


(a) Performance in terms of the path length generated.

(b) Performance in terms of the smoothness of the generated path.

(c) Performance in terms of the time needed to generate a path without obstacles.

Figure 5 – Comparison between QL and Dijkstra performance.



(a) Path executed by the robot using the q-learning approach.

(b) Path executed by the robot using the dijkstra-greedy approach.

Figure 6 – Paths executed by the robot using the approach based on q-learning and the dijkstra algorithm.

planning alongside our proposed strategy, visit: <youtu.be/hnEglYXulFE>. The video reveals that, despite QL producing a longer trajectory, it's faster due to fewer turns. Furthermore, the Dijkstra algorithm's trajectory includes 180° rotations, representing the most energy-intensive scenario.

Figure 6 illustrates the comparison. In Figure 6a, the robot follows a slightly longer path compared to Figure 6b. However, the former features more rectilinear movement with fewer curves, while the latter has numerous turns instead of long straight paths. Moreover, the path in Figure 6b includes two 180° turns, which are energetically more costly. In practical applications, paths with extended straight segments are preferable due to easier control and increased speed.

While both the proposed Q-learning-based approach and the Dijkstra algorithm achieved a 100% success rate in reaching target locations, the Q-learning method demonstrated notable advantages in reducing path curvature and improving planning efficiency. By minimizing the number of turns, the proposed approach enhances maneuverability, making it particularly suitable for real-world applications where smooth and time-efficient navigation is critical. Consequently, the Q-learning-based method is recommended for deployment in environments that prioritize operational speed, safety, and optimized path execution over purely shortest-distance criteria.

2.1.3 Concluding Remarks and Future Works

This section presents an offline path planning methodology for autonomous ground vehicle navigation using reinforcement learning, specifically QL. The primary objective is to address the challenge of efficiently visiting multiple shelves. The designed reward function is structured in such a way as to encourage the agent to select actions that facilitate the creation of paths with minimal curvature. This, in turn, results in paths characterized by predominantly linear segments rather than curved sections. The evaluation of the algorithm involved the use of various metrics, including path dimensions, frequency of curves, planning duration, and the rate of successful conclusions. Additionally, a comparative analysis was conducted between the proposed methodology and a greedy approach that incorporates Dijkstra's algorithm.

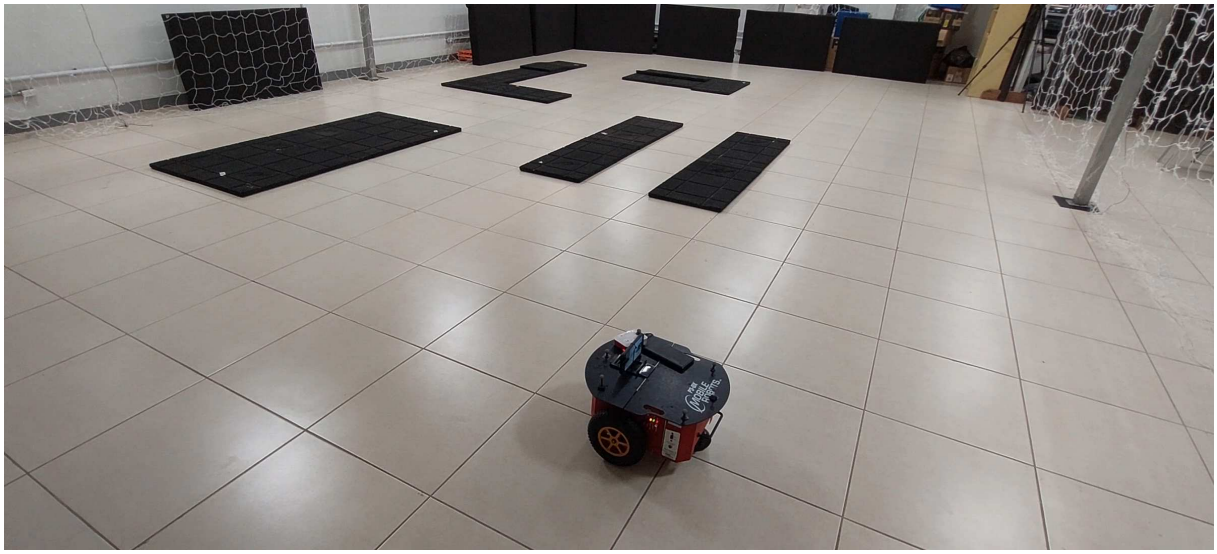
2.2 Q-Learning-Based Multi-Objective Global Path Planning

2.2.1 Problem Formulation

In this study, an offline navigation strategy for ground robots is proposed based on Q-learning. In warehouse-like scenarios, tasks such as picking up and delivering items to designated locations are common. The focus of this approach is to optimize the robot's path after item collection, ensuring efficient delivery. A Q-learning agent is trained to learn the most effective routes to visit the designated delivery points, minimizing travel time and unnecessary movements. This methodology is particularly relevant to structured environments like libraries and warehouses, where predefined delivery locations and high-demand zones require an optimized navigation strategy to improve workflow efficiency.

To validate the proposed approach, experiments were conducted in an 8×5 m workspace equipped with an Optitrack motion capture system. This system, consisting of 15 cameras, tracks the robot's position and orientation with millimeter precision. Additionally, the Pioneer 3DX ground robot was used, and two different scenarios were created, as shown in Figure 7. Both maps contain challenging obstacles, including U-shaped structures and narrow passageways. These obstacles introduce difficulties for mobile robotics, particularly for classical techniques such as Artificial Potential Fields or Tangential Escape (BRANDÃO; SARCINELLI-FILHO; CARELLI, 2013), which may fail to converge. Simulations were conducted using the Python programming language, with visualization performed in MATLAB.

To apply Q-learning to this problem, the environment was discretized into a grid. The arena, measuring $8 \times 5\,m$, was divided into cells of $50 \times 50\,cm$, resulting in a grid of 16×10 cells, totaling 160 cells. Figure 8 illustrates the discretized environment, where black cells represent obstacles, and green cells denote target locations.



(a) First map with obstacles and a robot.

(b) Second map with obstacles and a Pioneer 3-DX robot.

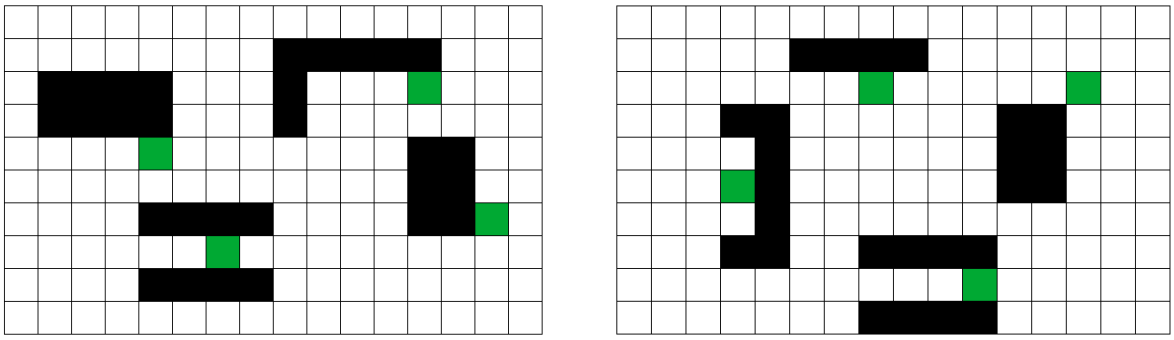
Figure 7 – Real scenarios monitored by the motion capture system with obstacles and a Pioneer 3-DX robot.

2.2.2 Q-Learning Algorithm for UGV Navigation

Q-Learning is a model-free reinforcement learning algorithm that estimates the value of actions in given states. Its goal is to determine the optimal policy $\pi(s)$ that maximizes accumulated rewards. This estimation, known as the Q-Value, is computed interactively for each state-action pair. Being model-free, it is suitable for environments with complex dynamics and offers more efficient use of computational resources compared to model-based approaches (ASADI, 2015).

The primary learning mechanism of this algorithm updates the Q-values using the Bellman equation:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right], \tag{2.5}$$



(a) First map discretized into 160 cells. (b) Second map discretized into 160 cells.

Figure 8 – Discretized maps into 50×50 centimeters cells. Black are obstacles and green are targets.

where: s and s' are the current and next states. a and a' are the current and next actions. r is the reward received after transitioning from s to s' due to a. α is the learning rate, which determines the extent of new information's impact. γ is the discount factor, balancing immediate and future rewards.

In UGV navigation, obstacle-free path planning involves representing states (s) by the vehicle's position and orientation within a discretized environment, typically a grid. Actions (a) correspond to possible movements. In this work the available actions are: moving forward, turning 90 degrees clockwise, or turning 90 degrees counterclockwise.

This section extends the state representation to track whether the agent has reached all four desired positions. The problem is modeled as a MDP, defined by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, R, \gamma)$. The agent starts randomly at any state $s_i \in \mathcal{S}$ to encourage exploration, a technique known as Exploring Starts (LIU, 2021).

The state $s \in \mathbb{N}^7$ includes the agent's current position and orientation in the grid world. The position is given by the row (r) and column (c), where $r \in \{1, 2, \ldots, 16\}$ and $c \in \{1, 2, \ldots, 10\}$ for a 16×10 map. The orientation (ψ) can be one of four values: $0, \frac{\pi}{2}, \pi, -\frac{\pi}{2}$. Binary values indicate whether the agent has reached each of the four desired positions (d_1, d_2, d_3, d_4). This results in a total of 10,240 states, $\mathbf{s} = [r\ c\ \psi\ d_1\ d_2\ d_3\ d_4]$.

The reward function for this agent is designed to save energy by prioritizing linear movement over rotational movement. Energy consumption is associated with the number of turns the robot makes. For each step, the robot receives a negative reward (r_l) of -1. If the robot performs a rotational action, it is penalized with a reward (r_t) of -4.55. Collisions with obstacles result in a reward (r_c) of -100. Each objective cell reached rewards the robot (r_g) with $+100$. Therefore, the reward function is described as:

$$r = -1 - r_t T(a) - r_c C(s) + r_g G(s), \tag{2.6}$$

where $T(a)$, $C(s)$, and $G(s)$ are step functions. $T(a)$ yields a high value if the action is a

turn, $C(s)$ yields a high value if the current state is an obstacle, and $G(s)$ yields a high value if the current state is a target state.

Here, it is assumed that increased curvature in the robot's path results in higher energy consumption. Therefore, this metric aims to minimize deviations from straight-line trajectories toward the goal, consequently optimizing energy efficiency.

It is important to highlight that, although the proposed task bears similarities to TSP, our scenario permits the robot to revisit the same destination. Unlike the TSP, where each location is visited only once, our objective accommodates multiple visits to the same location, introducing a distinct set of challenges and requiring alternative optimization strategies.

2.2.2.1 Agent's Hyperparameters

To train the agent, the hyperparameters and rewards were determined empirically through several simulations until optimal values were identified. The learning rate α was set to 0.1, and the discount factor γ was 0.99. To encourage exploration, an ϵ-greedy strategy was used, where the agent has a probability ϵ of choosing a random action. The value of ϵ follows a linear decay from 1 to 0.1 over 50,000 episodes, as shown in

$$\epsilon_t = 1 - \frac{0.9}{50\,000} \cdot t, \tag{2.7}$$

where t is the current episode number. Given these parameters, the agent is trained according to Algorithm 2.

Algoritmo 2: Q-Learning Training

```
for i ← 1 to 50,000 do
    if i mod 5000 = 0 then
        Get all the states that have not yet converged;
    Define starting point according to exploring starts;
    Calculate ε_t via (2.7);
    while episode not over do
        if Random(0,1) < ε then
            a_t ← RandomAction;
        else
            a_t ← π(s) ; // Optimal action
        Calculate r via (2.6);
        s' ← Step(a) ; // Find next state
        Q(s,a) ← Update via (2.5);
        s ← s' ; // Update current state
```

2.2.2.2 System Architecture

After training the agent, path planning becomes straightforward; however, deploying the system to a real robot is still required. The Robot Operating System (ROS) was employed for this purpose, utilizing the ROSARIA package[2] for robot communication. Path following was implemented using the Vector Field Control (VFC) method. The robot's speed modulus was set at $0.12\,m/s$, and the control point was positioned $10\,cm$ ahead of the robot's center. The robot was considered to have successfully reached the target point upon entering a $5\,cm$ radius around the desired location.

Figure 9 provides a high-level overview of the system architecture. Initially, the agent is trained, and the path is generated and sent as reference to the robot. The Optitrack system then tracks the robot's current pose. Using this pose information, the position error is computed and sent to the VFC controller, which generates the control signals (linear velocity v and angular velocity ω). These signals are transmitted to the robot via ROS. This loop continues until the robot has visited all points on the path.

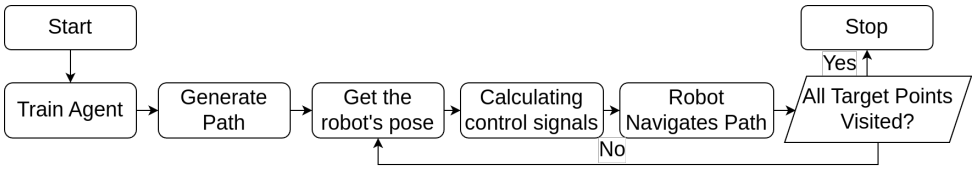


Figure 9 – Path-following workflow adopted in this work.

2.2.3 Results and Discussion

The proposed methodology was evaluated through two phases: training and path quality assessment. The algorithm was implemented in Python and is publicly available in a GitHub repository[3]. Both training and evaluation phases were executed on a computer equipped with 32 GB RAM, a Ryzen 5 5600G processor, and a GeForce RTX 3080 GPU.

2.2.3.1 Training Performance

To avoid inaccuracies in the training evaluation, 20 training sessions were conducted for each map. On average, the first map required 150 ± 3 seconds to train, while the second map required 174 ± 3 seconds. Both training sessions were successful, as indicated by the increasing accumulated rewards over episodes, as illustrated in Figure 10. The reward graph also highlights the impact of training on non-converged states. Specifically, at episodes 5000 and 10000, a slight drop in the average reward occurs because the agent is forced to start in non-converged states to promote learning. If the agent has converged in all states, it can start from any state until the maximum training episodes are reached.

[2] <https://wiki.ros.org/ROSARIA>
[3] <https://github.com/Hiago013/multi-goal-navigation>

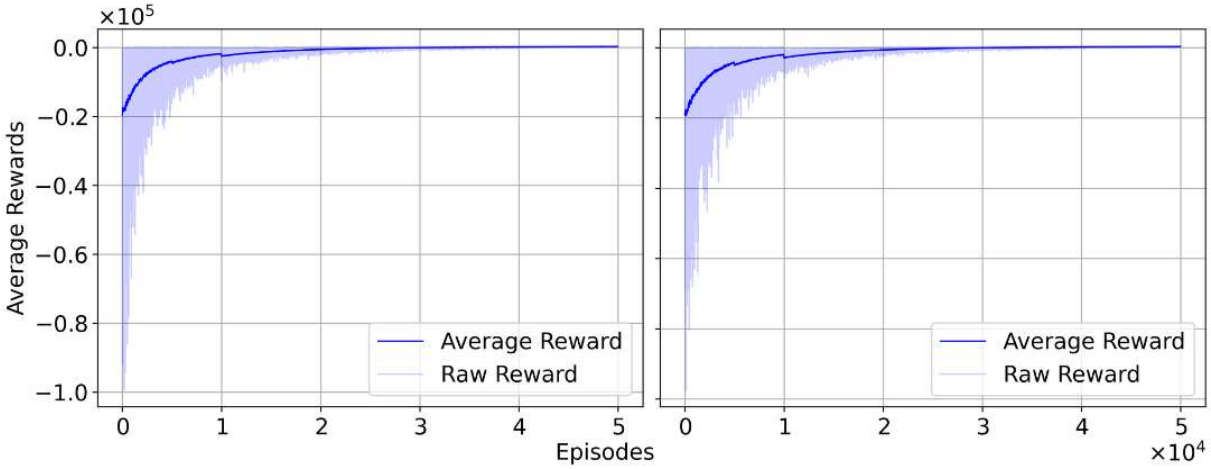


Figure 10 – The average reward from training the first map is shown on the left, and the reward from training the second map is shown on the right.

2.2.3.2 Path Evaluation

To assess path quality, the metrics described by Liu et al. (2023) were employed: path length, number of turns, and planning time. The planning time corresponds to the duration required to compute a feasible path, a calculation performed for each obstacle-free cell. The mean and standard deviation of these metrics were subsequently computed. For comparative analysis, four algorithms were evaluated: a greedy approach (which iteratively selects the nearest unvisited target), Dijkstra's algorithm, A*, and Breadth-First Search (BFS). The performance of each algorithm was assessed by conducting path planning from every obstacle-free cell in all four possible orientations.

As shown in Tables 2 and 3, the proposed approach outperforms other techniques in terms of path length. While algorithms like Dijkstra's always return the lowest cost path, which is the shortest path in this context, they are insufficient for multi-objective problems. In the second map (Figure 7b), the average path sizes for QL and other techniques were, respectively, 15.04 m and 15.44 m, showing no significant difference. However, in the first map (Figure 7a), QL achieved an average path length of 13.65 m, compared to 14.19m and 14.90m for the other techniques, resulting in up to an 8.39% reduction in path length.

The proposed approach demonstrates superior performance in terms of the number of curves. On the second map, it averaged 6.09 curves, compared to 9.19 for Dijkstra, 8.65 for A*, and 8.80 for BFS. This represents a reduction of up to 30.80%. On the first map, the results are even more impressive, with an average of 5.65 curves. In contrast, Dijkstra, A*, and BFS produced 9.65, 9.63, and 9.28 curves, respectively, with a standard deviation greater than 1.50. This indicates a reduction of up to 41.45%. These results demonstrate that the proposed algorithm effectively produces shorter paths with fewer curves.

Evaluating the time required to generate obstacle-free paths is crucial for real-

Table 2 – Comparison of the average and standard deviation performance of the proposed approach in the first map against the baseline for metrics including path length, number of turns, planning time, and success rate.

Path	**Path Length (m)**	**# of Turns**	**Planning Time (ms)**	**Success Rate %**
QL	13.65 ± 1.32	5.65 ± 0.99	0.15 ± 0.02	100.00
Dijkstra	14.19 ± 1.15	9.65 ± 1.89	0.41 ± 0.07	100.00
A*	14.90 ± 1.15	9.63 ± 2.46	1.16 ± 0.59	100.00
BFS	14.90 ± 1.15	9.28 ± 1.58	0.17 ± 0.02	100.00

Table 3 – Comparison of the average and standard deviation performance of the proposed approach in the second map against the baseline for metrics including path length, number of turns, planning time, and success rate.

Path	**Path Length (m)**	**# of Turns**	**Planning Time (ms)**	**Success Rate %**
QL	15.04 ± 1.36	6.09 ± 1.28	0.17 ± 0.02	100.00
Dijkstra	15.44 ± 1.35	9.19 ± 1.58	0.43 ± 0.04	100.00
A*	15.44 ± 1.35	8.65 ± 1.75	1.16 ± 0.21	100.00
BFS	15.44 ± 1.35	8.80 ± 1.85	0.19 ± 0.03	100.00

world applications. The proposed Q-learning (QL) technique demonstrated superior performance on both maps, achieving average planning times of $0.17\,ms$ and $0.15\,ms$. The Breadth-First Search (BFS) strategy was the closest competitor, with only a $0.02\,ms$ difference compared to QL. Conversely, the A* algorithm exhibited the slowest performance, averaging $1.16\,ms$, making QL approximately seven times faster. The significant speed advantage of the QL approach is attributed to the algorithm's nature; once trained, the learned policy $\pi(s)$ directly provides the subsequent state given the current state, resulting in a pathfinding complexity of $O(n)$.

While the proposed algorithm exhibits superior performance across various metrics, energy consumption remains a critical consideration. Paths characterized by straighter trajectories and fewer curves facilitate easier vehicle control, consequently reducing energy expenditure. Demonstrations available through the YouTube channel link[4] illustrate how the proposed algorithm can generate distinct paths from the same initial position, varying only in orientation. This behavior results from training the algorithm to prioritize longer straight-line segments. Figures 11 and 12 highlight these differences. Although all algorithms successfully guide the robot to its destination, the proposed QL-based method uniquely achieves paths dominated by straight-line segments with minimal curvature. In contrast, traditional techniques often produce paths involving sharp 180-degree turns, representing the worst-case scenario in terms of energy efficiency.

All the presented approaches achieve a 100% success rate. However, QL performs best across all metrics, generating paths with fewer turns. This makes QL the most suitable

[4] <https://youtu.be/P4kmoGEOCsI>

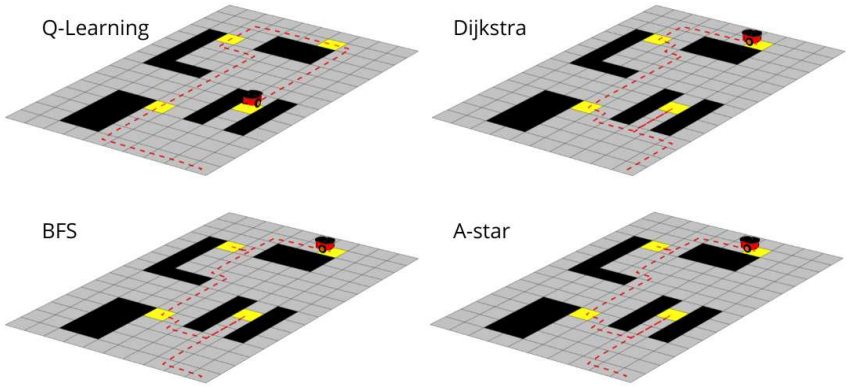


Figure 11 – First map paths using QL, Dijkstra, BFS, and A* techniques.

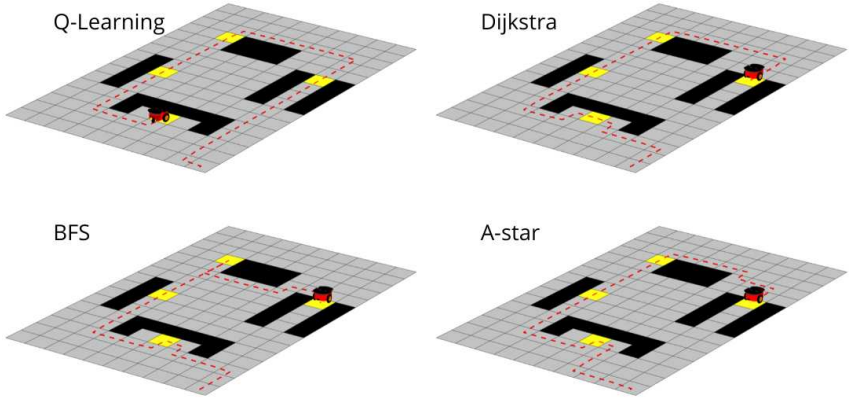


Figure 12 – Second map paths using QL, Dijkstra, BFS, and A* techniques.

for real scenarios, as shown in the video, where energy savings, planning speed, and reduced total path length are prioritized.

2.2.4 Concluding Remarks

This section presented an offline method for planning and navigating robots in structured environments using Q-learning (QL). The proposed approach was experimentally validated using a Pioneer 3-DX robot in conjunction with the Optitrack motion capture system. The primary goal was to achieve efficient multi-objective path planning by simultaneously optimizing the path length, the number of turns, and computational speed. Experimental results demonstrate that the proposed method outperforms comparative algorithms across all evaluated metrics, achieving a 100% success rate. Additionally, paths generated by this strategy are characterized by longer linear segments, contributing to reduced energy consumption and improved robot controllability.

In future work, this method will be extended to heterogeneous multi-agent systems, with an emphasis on optimizing collaboration among multiple robots. To examine the computational complexity, an NP-Hardness analysis will be conducted by varying map dimensions while maintaining a constant number of objectives. Conversely, with a fixed map size, the impact of increasing the number of objectives on the solution time will

also be assessed. These analyses aim to provide valuable insights into the scalability and complexity of multi-objective navigation problems. Furthermore, the relationship between agent quantity and collision probability will be explored, under the hypothesis that an increased number of agents may result in more frequent collisions, necessitating the development of advanced coordination strategies.

3 Multi-Agent Path Planning Logistics Operation using Reinforcement Learning

This chapter is based on the study in a major review at the journal IEEE Access, which investigates the application of RL techniques, specifically QL, in multi-agent path planning within a constrained library logistics scenario. Traditional multi-agent path planning methods, such as heuristic-based algorithms like Dijkstra's and A*, are effective in structured environments but suffer from significant computational overhead due to frequent path recalculations required for collision avoidance. Additionally, these methods lack flexibility when adapting to dynamic environmental changes.

The primary objective of this research is to enhance the efficiency and adaptability of navigation strategies for multi-agent systems operating in structured yet dynamic indoor environments, such as libraries. This study specifically addresses challenges related to computational inefficiency and rigid heuristics by integrating Q-learning with Transfer Learning (TL) and Curriculum Learning (CL). Taking advantage of TL and CL allows agents to progressively acquire and generalize navigation skills, making a smooth transition from simpler tasks to more complex scenarios. This approach significantly reduces training time and computational resources.

Through comprehensive simulation-based validation, the proposed Q-learning framework demonstrated superior navigation performance compared to traditional heuristic methods. The proposed approach effectively balances critical aspects such as collision avoidance and path efficiency. Moreover, the results underscore the ability of Q-learning algorithms to dynamically adjust to real-time interactions among multiple agents, facilitating efficient and decentralized task execution in complex, multi-agent scenarios.

Despite the notable success of RL in warehouse scenarios, limited research has addressed its application within library logistics, where structured layouts necessitate highly precise navigation strategies. This chapter addresses this gap by applying Q-learning in a constrained multi-agent setting, achieving efficient and collision-free navigation in a two-story library environment. Additionally, TL and CL are introduced to accelerate agent training and mitigate the state-space explosion issue. The effectiveness of this approach is demonstrated through comparative analyses against traditional path-planning algorithms, such as Dijkstra's algorithm and A*, in scenarios inspired by real-world library logistics tasks. In doing so, this chapter highlights RL's potential to outperform classical methodologies, laying the groundwork for future research directions in integrating deep reinforcement learning (DRL) techniques to further enhance operational adaptability and multi-agent collaboration in dynamic, structured indoor environments.

3.1 Background and Related Works

Multi-agent path planning plays an essential role in robotic navigation, particularly in logistics operations, including warehouse automation and industrial environments. Existing methods for addressing multi-agent path planning are typically classified into three main categories: heuristic-based methods (e.g., Dijkstra's algorithm and A*), potential field approaches, and machine learning-based solutions.

3.1.1 Heuristic-Based Methods

Traditional algorithms such as Dijkstra's Algorithm and A* are widely used for global path planning, ensuring optimal navigation in structured environments (DIJKSTRA, 2022; HART; NILSSON; RAPHAEL, 1968). However, these methods become computationally expensive in multi-agent settings due to the need for frequent re-computation to avoid collisions (FOEAD et al., 2021). Additionally, their ability to handle dynamic interactions is limited, requiring predefined heuristics for obstacle avoidance.

While heuristic-based methods such as A* and Dijkstra's Algorithm ensure optimal path planning in structured environments, they require extensive re-computation in dynamic settings (MEKALA, 2024), making them computationally expensive for multi-agent coordination. Moreover, these methods lack the flexibility to directly incorporate environmental changes into their decision-making process. In contrast, Q-learning (QL) offers several advantages for multi-agent path planning. First, once trained, QL performs action selection in O(n) complexity, significantly reducing computational overhead compared to heuristic approaches. Second, QL allows for dynamic obstacle adaptation by adjusting the reward function, eliminating the need for artificial obstacle inflation, a common limitation in deterministic methods. Additionally, unlike static heuristics, QL continuously improves its navigation strategy based on real-time interactions, making it more adaptable for decentralized multi-agent systems. By leveraging Transfer Learning (TL) and Curriculum Learning (CL), our approach further enhances QL's efficiency, reducing training time while maintaining high success rates in collaborative path planning. These advantages position QL as a strong alternative to traditional algorithms, particularly in scenarios requiring continuous learning and real-time adaptability.

3.1.2 Learning-Based Approaches in Multi-Agent Path Planning

Recent advances increasingly have taken advantage of machine learning techniques to improve adaptability in complex and dynamic environments. Reinforcement learning (RL), in particular, has emerged as a powerful approach, with methods such as Deep Q-Networks (DQN) demonstrating significant improvements in learning efficiency and speed compared to traditional Q (YANG; JUNTAO; LINGLING, 2020) learning algorithms. In

addition, alternative strategies involving dynamic traffic management have been explored to mitigate congestion problems in multi-agent navigation scenarios (CHUNG et al., 2019).

One of the most widely adopted RL techniques in path planning is Q-learning (QL). QL is a model-free algorithm that enables agents to learn an optimal policy through trial and error, using a value-based approach (WATKINS; DAYAN, 1992). It estimates the best action to take in a given state by iteratively updating Q-values based on a reward function and the Bellman equation. However, classical QL suffers from slow convergence in large state spaces due to its reliance on exhaustive exploration (LOW; ONG; CHEAH, 2019). To address these challenges, modified QL techniques, such as dynamic window-based search and obstacle-aware QL, have been proposed (CHANG et al., 2021; BUONO et al., 2023). These methods enhance QL's efficiency by incorporating environment constraints directly into the learning process.

Despite these improvements, applying QL in multi-agent environments remains challenging, primarily due to state-action space explosion and the need for coordination between multiple agents. To overcome these limitations, our work integrates Transfer Learning (TL) and Curriculum Learning (CL) to improve training efficiency and convergence rates. TL enables agents to transfer knowledge from simpler navigation tasks to more complex environments, reducing learning time (ZHU et al., 2023). Meanwhile, CL structures the training process into progressive difficulty levels, enhancing the agent's adaptability to increasingly complex scenarios (NARVEKAR et al., 2016).

Recent advancements in neural network-based control strategies have significantly contributed to improving the robustness and accuracy of multi-agent systems operating in uncertain environments. A particularly relevant study explores the problem of uncertain target enclosing control in multi-agent networks, leveraging neural network-based approximations to estimate dynamic uncertainties and disturbances in real-time. This approach eliminates the dependency on high-cost sensors for acquiring high-order target information such as velocity and acceleration, thereby improving the system's adaptability in dynamic environments (LI et al., 2025). The proposed method demonstrates enhanced robustness against matched and unmatched disturbances, which aligns with the challenges addressed in our work regarding decentralized and cooperative decision-making in reinforcement learning-based multi-agent navigation.

While other reinforcement learning techniques, such as Deep Q Networks (DQN) (ARULKUMARAN et al., 2017) and Proximal Policy Optimization (PPO) (RIO; JIMENEZ; SERRANO, 2024), could potentially enhance learning efficiency and decision-making, this study focuses on Q-learning due to its well-established reliability and interpretability in discrete state-action spaces. The integration of TL and CL further improves its convergence and performance in multi-agent environments. Future work could explore the incorporation of deep reinforcement learning methods to further refine navigation efficiency

and adaptability in dynamic environments.

In this chapter, reinforcement learning techniques have formed the core of the proposed path-planning framework, with neural network-based estimation methods serving as complementary tools, particularly valuable in scenarios requiring real-time adjustments due to environmental uncertainties. This integration highlights potential avenues for future research, notably the exploration of deep reinforcement learning (DRL) methodologies incorporating neural network-based estimation techniques to enhance multi-agent collaboration and adaptability.

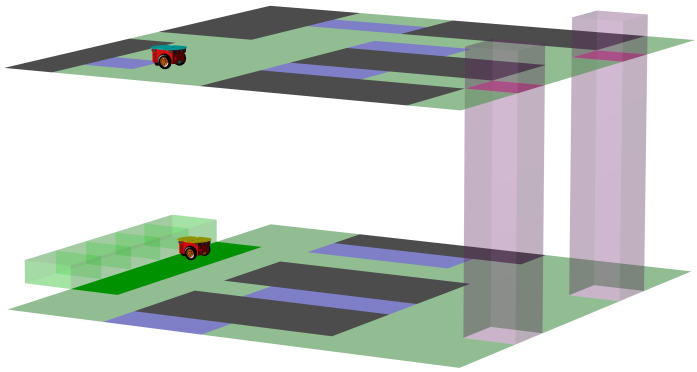


Figure 13 – Our 3D representation of the two-floors library, where the shelves are in gray, the elevators are in pink, the delivery locations are in purple, and the service bays are in dark green.

3.2 The problem Formulation

In this paper, TL and CL support a QL strategy to assist a multi-agent system in distributing items in a library. This section describes the implementation steps, and all the code is available in our repository[1].

The library used for testing is a 9×9 discrete, two-floor environment, as illustrated in Figure 14. In this grid-based map, the agents have 106 navigable positions. Each position has a state representation that encapsulates data about the agent itself, its adjacent agents, the pickup and delivery points, and information about the task's progress. Thus, the state representation consists of eight elements. At each position on the grid, each agent has five possible actions: moving forward, moving backward, moving left, moving right, or remaining stationary.

Since we are applying an artificial intelligence approach to the proposed application, the reward structure considers the shortest distance to the goal and the lowest risk of obstacle or agent collision. The state, action, and reward representations are detailed further in this section.

[1] https://github.com/Hiago013/rl-multiagent-transfer

Training and path planning are, a priori, done offline. However, if the robots are on a collision course, an online reaction occurs to generate a new route to avoid a possible collision. It is worth noting that one of the actions a robot can take in the event of a collision is to stop its movement and wait for the next one. Therefore, real-time performance is guaranteed, as there is an entity that observes the states of the robots. If they are on the verge of collision, a new plan is triggered for these robots. This hybrid approach ensures both precomputed efficiency through offline training and adaptive safety measures through online adjustments.

3.2.1 The Environment

- **Service Bay:** The location where agents pick up books. The task is complete when no books remain for delivery.
- **Delivery Shelves:** The location where agents deliver books. In this study, books are stored on two floors.
- **Access Channel:** The area where agents move between the two floors.
- **Bookshelves:** The final destination for books after delivery. In navigation, bookshelves serve as static obstacles, blocking agent movement in specific regions on the map.

Figure 14 illustrates the service bays (S), drop-off locations (D), and access channels (C). The bookshelves, shown in black, are static obstacles in the scene.

3.2.2 The States

Figure 14 illustrates all 162 positions on the grid map considered in this study. The simulated environment includes five service bays (S), fifteen delivery shelves (D), two interfloor access channels (C), and fifty-six bookshelves. Any cell not designated for these features is considered free space, allowing agents to traverse the map to complete their missions.

In this scenario, an agent's state is defined by its current position, pickup and delivery locations, current stage, and the presence of other agents in adjacent cells. This state is represented as:

$$s = (r, p, d, a, s_f, s_b, s_l, s_r), \tag{3.1}$$

where $r \in \mathbb{R}^{9\times9\times2}$ represents the robot's current position on the grid map, computationally expressed as a vector with 162 elements. Similarly, p and $d \in \mathbb{R}^{9\times9\times2}$ denote the pickup and delivery locations.

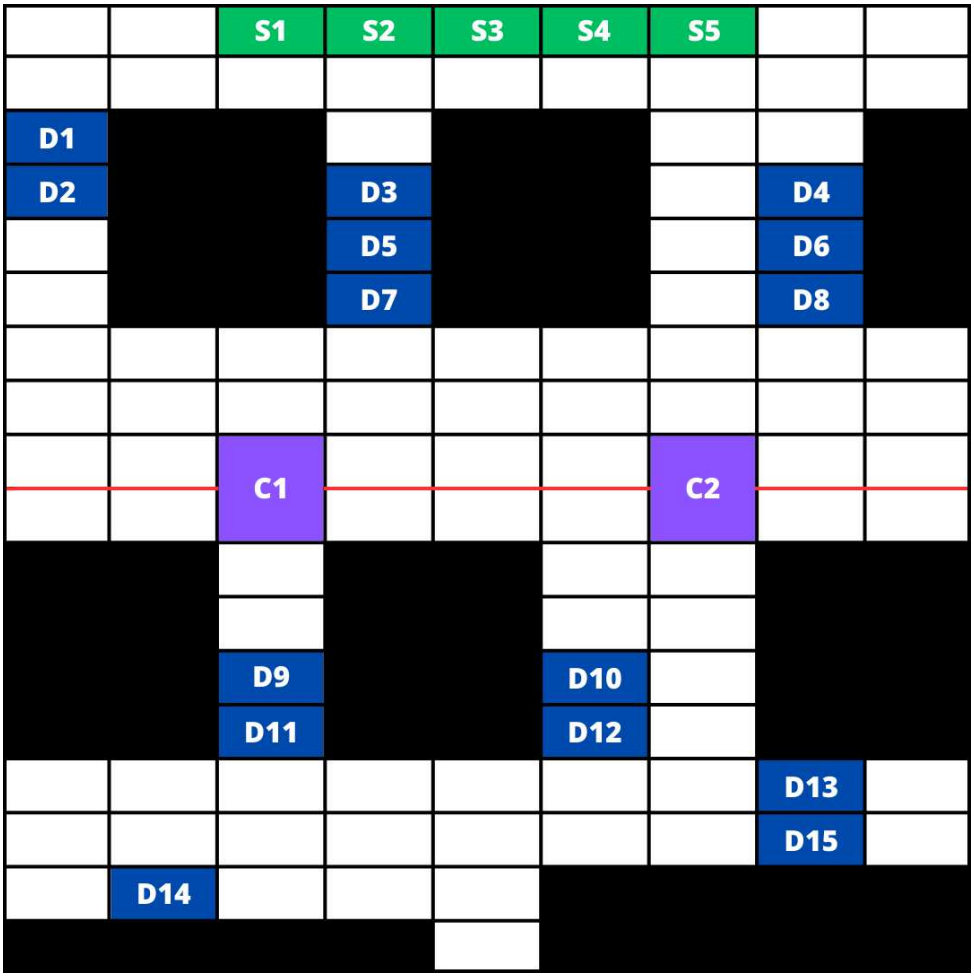


Figure 14 – The proposed environment uses the following color scheme: green represents the service bay, blue the delivery shelves, purple the access channel, and black the bookshelves.

Additionally, s_f, s_b, s_l, and s_r are Boolean variables indicating whether the cells in the forward, backward, left, and right directions within the agent's 4-neighborhood are occupied. With this formulation, the problem encompasses $381,000$ possible states.

3.2.3 The Actions

In a grid environment, an agent typically has five possible actions: moving forwards, backwards, left, right, or staying still. However, the agent's movement is constrained by the environment, such as static bookshelves or the grid boundaries. Specifically, when the agent is positioned at the edges of the grid, denoted by $r_{(\{1,9\},j,k)}$ or $c_{(i,\{1,9\},k)}$ where $i, j = \{1, 2, ..., 9\}$ and $k = \{1, 2\}$, it cannot perform actions that would move it beyond these limits. For this study, the agent considered is a differential-drive robot (Pioneer 3DX).

3.2.4 The Reward Function

The proposed reward function aims to increase agent engagement by promoting efficient task execution while preventing collisions. It includes three positive rewards to incentivize task completion and four negative rewards to discourage undesirable behavior.

- **Collection Reward:** The reward r_p is received when the agent picks up a book. Here, $r_p = 50$.
- **Delivery Reward:** The reward r_d is received after completing a delivery task. Assuming the agent can carry only one book, we set $r_d = 100$.
- **Accomplishment Reward:** The reward r_a is received when the agent completes all tasks and returns to the service bay. In this case, $r_a = 150$.
- **Delay Penalty:** A constant negative reward is applied at each step to encourage faster task completion.

$$\text{For each step, } r_s = -1. \tag{3.2}$$

- **Distance Penalty:** A negative reward that decreases as the agent nears its goal, thereby enhancing performance. Its calculation is based on the Manhattan distance (D_M) between the current and goal positions:

$$D_M = |x_a - x_g| + |y_a - y_g| \tag{3.3}$$

$$\text{For each step, } r_g = -\frac{D_M}{2N_{cells}}. \tag{3.4}$$

- **Waiting Penalty:** A negative reward is applied to discourage excessive inactivity. We set $r_w = -6$.
- **Collision Penalty:** A negative reward is assigned to mitigate agent collisions. A constant penalty of $r_c = -150$ is applied whenever an agent collides with another.

Thus, after defining these rewards and penalties, the total reward for each step is given by

$$r_t = r_p + r_d + r_a + r_s + r_g + r_w + r_c. \tag{3.5}$$

3.3 The Training Process

The training phases were designed to facilitate learning and transfer the agent's behaviors to other tasks. Four distinct learning stages were defined: delivery, pickup, return, and multi-agent coordination.

During the delivery stage, the agent moves the book to the appropriate shelf after retrieval. In the pickup stage, the agent learns to navigate to service bays and select the correct book. The return stage occurs after all books have been delivered, requiring the agent to return to the service bays. In the multi-agent phase, multiple agents are progressively introduced to learn cooperative interactions.

The first three stages include Recognition and Augmentation phases. Recognition exposes the agent to specific combinations of service bays and delivery locations, gradually expanding its access across the entire map. Augmentation enhances and transfers knowledge across a broader range of states, encompassing additional bays and delivery locations.

The training employed Q-learning (QL) with an ϵ-greedy strategy to address the exploration-exploitation trade-off. A curriculum strategy based on agent-goal distance was utilized, featuring six difficulty levels across four training stages. This method accelerates learning by progressively increasing task complexity. Transfer learning facilitates the propagation of agent behaviors across different training levels. The multi-agent system operates in a decentralized, cooperative manner, enabling information sharing while allowing individual agents to make decisions and collaborate on book deliveries.

The learning process consists of two phases, each comprising six curricular levels organized by proximity to the goal, with the nearest level preceding the farthest. Training durations vary across levels: the Recognition phase involves 50, 100, 150, 200, 250, and 300 epochs for each successive level. Additionally, the maximum value of $\epsilon_{\max}$ decreases linearly across stages, starting at 1 for the first stage and decreasing by 0.1 per stage down to 0.5. Therefore, the value of ϵ for each episode is computed as follows:

$$\epsilon = \epsilon_{\max} + \frac{0.1 - \epsilon_{\max}}{\text{episode}_{\max}} \cdot \text{episode}_{\text{current}}. \tag{3.6}$$

In both the Augmentation and Recognition phases, the value of ϵ remains consistent. However, the number of epochs varies across different levels: 200 for level 1, 300 for level 2, 400 for level 3, 500 for level 4, 600 for level 5, and 700 for level 6. The learning rate α is set to 0.2 and remains unchanged throughout all stages. Table 4 summarizes the hyperparameters employed in this study.

3.3.1 First Stage: Delivery operation

In this stage, the agent learns to deliver books to shelves on two floors. Initially, the agent is trained to deliver books exclusively on the first floor after collecting them from the central service bay. Subsequently, the agent is trained to deliver books to shelves on the second floor. Figures 15a and 15b illustrate the implemented training methodology. Training progresses through curricula categorized by goal proximity. The agent starts at

Table 4 – Hyperparameters used in each curricular level: Number of epochs in the Recognition phase, Number of epochs in the Augmentation phase, epsilon (ϵ), learning rate (α), and discount factor (γ).

Level	$\text{Epochs}_{\text{REC.}}$	$\text{Epochs}_{\text{AUG.}}$	$\epsilon_{\max}$	α	γ
1	50	200	1.0	0.2	0.9
2	100	300	0.9	0.2	0.9
3	150	400	0.8	0.2	0.9
4	200	500	0.7	0.2	0.9
5	250	600	0.6	0.2	0.9
6	300	700	0.5	0.2	0.9

the closest level to its destination, the first floor. After 50 episodes, it advances to the next level, using only cells not utilized in previous levels to increase difficulty. Training epochs for each curriculum level are specified in Table 4.

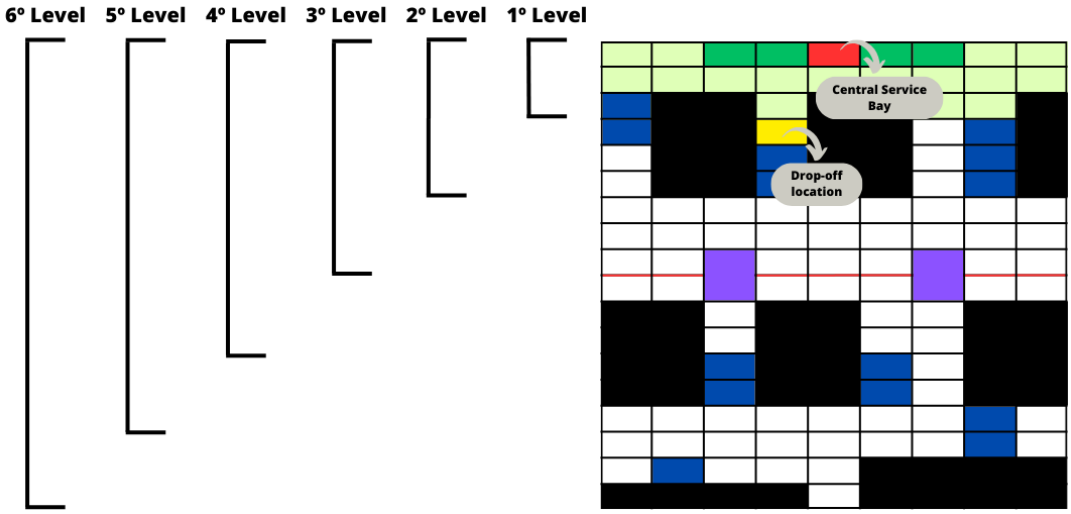
Figures 16a and 16b show the Recognition and Augmentation phases. In the Recognition phase, the agent aims to deliver a book collected from the central service bay to a designated location. This acquired behavior is then transferred to other delivery locations on the first and second floors. After this transfer, training continues in a structured curriculum, as detailed in Table 4 for the Augmentation phase. By the end of this training, the agent is adept at delivering books to any designated location, starting from the central service bay. This knowledge is then transferred to ensure the same behavior when collecting books from any other service bay.

3.3.2 Second Stage: Pickup operation

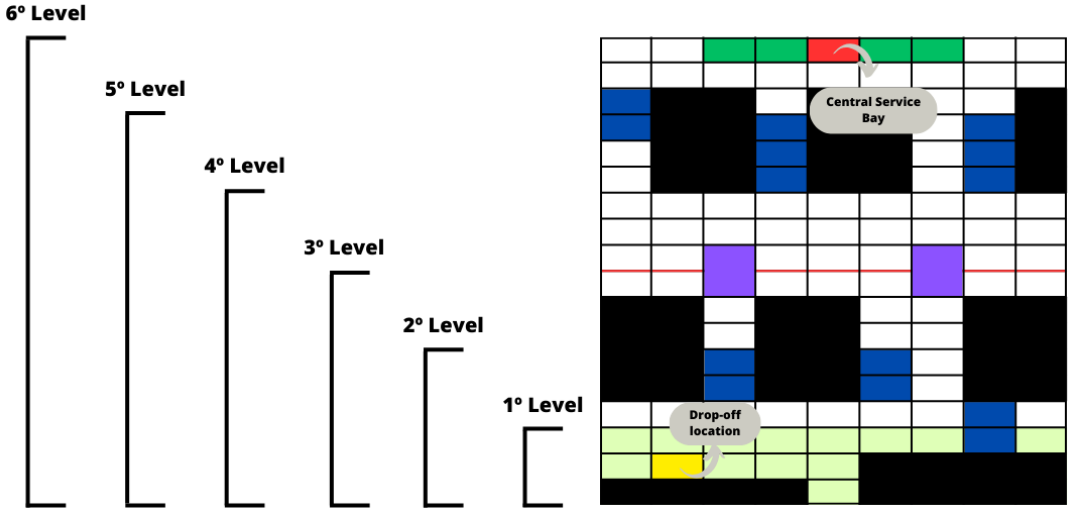
During this stage, the agent's objective is to learn how to collect books from service bays for delivery to a predetermined location. This approach is necessary because the agent's state formulation, shown in Equation (3.1), requires information about the intended delivery point.

Figure 17a illustrates the agent's training process, which follows a curriculum-based learning approach. Training begins at the simplest level and progresses sequentially through increasingly difficult levels, with the final level being the most challenging due to its greater distance from the goal. The number of training epochs for this stage is detailed in Table 4. Figure 17b depicts the Recognition and Augmentation phases of training. In the Recognition phase, the agent learns to pick up a book from the central service bay for delivery to a specific map location. This phase focuses solely on the pick-up action, without actual delivery. Knowledge acquired here is then extended to other bays by transferring weighted Q-values, $Q(s, a)$, from source states to destination states.

Subsequently, training follows a structured curriculum, as indicated by the number of episodes in Table 4. Upon completion, the agent can pick up books from any service



(a) Training levels to place the books on the first floor.



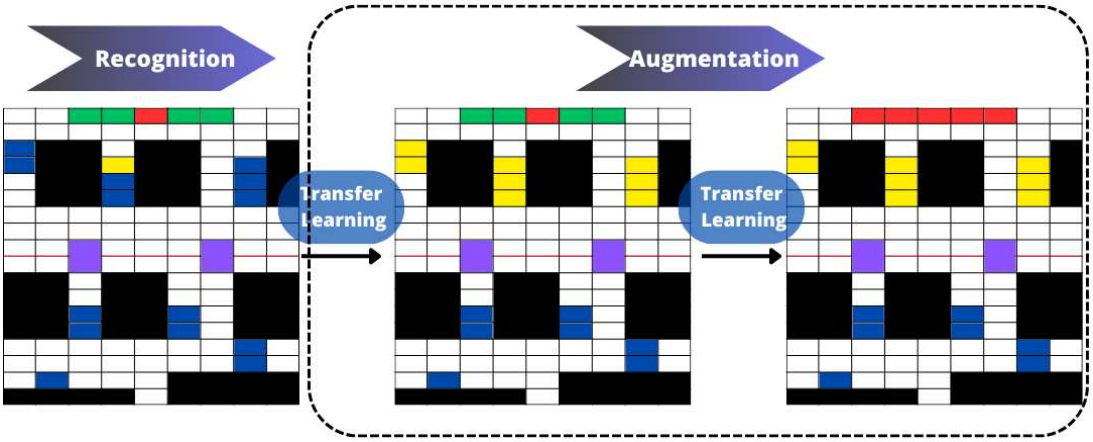
(b) Training levels to place the books on the second floor.

Figure 15 – Illustration of the training levels to teach the agent to place the books on the correct shelves. Figure (a) shows the proposed curriculum for the agent to learn to deliver to the first floor of the scenario, while the Figure (b) teaches the agent to deliver to the second floor.

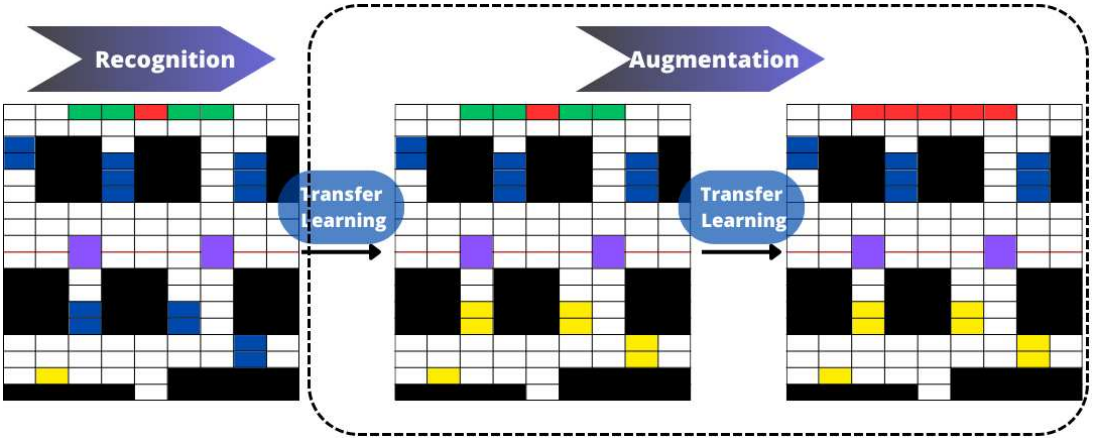
bay while maintaining consistent behavior regardless of the pick-up location, since the fixed delivery location does not influence the pick-up action.

3.3.3 Third Stage: Return to bays

Returning to the service bays signifies the completion of the task. The agent returns to any service bay once all books have been delivered. Due to the similarity between actions in the Pickup and Return stages, the Recognition phase is unnecessary for the latter. Knowledge gained during the Pickup stage provides a solid foundation for task completion, allowing training in the Return stage to begin directly in the Augmentation phase.



(a) Two training phases: Recognition and Augmentation for the first training stage on the first floor.



(b) Two training phases: Recognition and Augmentation for the first training stage on the second floor.

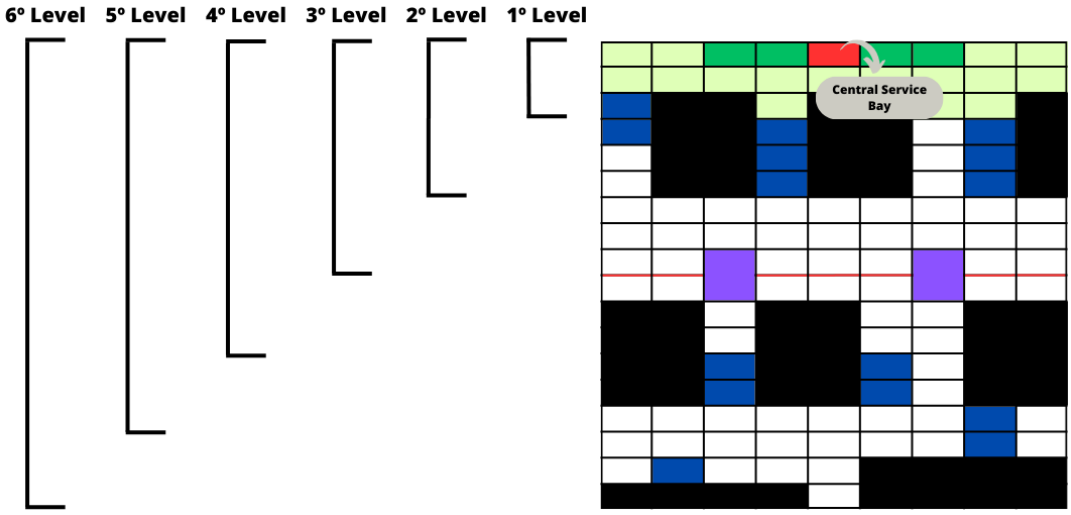
Figure 16 – Agent learning diagram on the first and second floor for the first stage.

In the Augmentation phase, transfer learning helps the agent understand how to finalize actions at the central service bay, building on behaviors learned in the Pickup stage. Predetermined pickup and delivery locations, embedded in the agent's state representation, enable learning to incorporate any service bay for action completion. Transfer learning is applied again at the final curriculum level, facilitating knowledge transfer about returning to any service bay for various pickup and drop-off combinations.

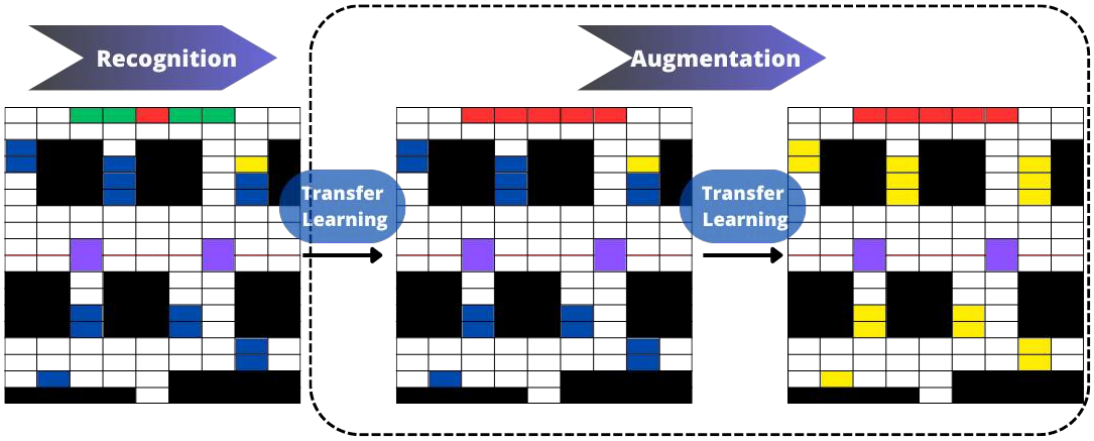
3.3.4 Fourth Stage: Multi-agent collaboration

In this phase, knowledge from states $(r, p, d, a, 0, 0, 0, 0)$ is transferred to states $(r, p, d, a, s_f, s_b, s_l, s_r)$. The agent uses a task policy learned in a solo environment. The transfer follows the initial steps, where learning begins with the state elements set to zero. In the source domain, states do not include other agents, whereas in the target domain, states incorporate all agent combinations.

Q-values are evaluated, and if a collision risk is detected from previously learned



(a) Training levels to pickup the books on the bays.



(b) Two training phases: Recognition and Augmentation for the second stage.

Figure 17 – Illustration of the second stage of training. Figure (a) shows the learning levels of the curriculum, while Figure (b) shows the Recognition and Augmentation phase.

Table 5 – Number of agents, books and epochs used in each curriculum level.

	Agents	Books	Epochs
Curriculum 1	2	3	5000
Curriculum 2	3	4	3000
Curriculum 3	4	5	2000

actions, the Q-value is replaced with a constant negative value -100 to indicate unsatisfactory actions. Subsequently, the agent undergoes structured training based on the number of agents, the total number of books to be delivered, and the number of training epochs, as detailed in Table 5.

3.3.4.1 How to Transfer?

Transferring behavior via the Q-table involves initially acquiring action values for a trained state. Subsequently, 80% of these values are transferred to the target state if no

collision is detected. In the event of a collision, a value of -100 is assigned to discourage the agent from taking that action.

For instance, consider a state s_1 with $Q(s_1, A) = [10, 15, 20, 35, -5]$. The optimal action in state s_1 corresponds to the value 35, indicating a right turn. If a new state s_2 emerges with an adjacent agent on the right, its initial values will be $Q(s_2, A) = [8, 12, 16, -100, -4]$.

3.3.4.2 Evaluating Effectiveness

The effectiveness of a multi-agent system is evaluated based on its success rate, the average reduction in steps per agent, and the total number of steps taken by all agents. The system's efficiency, E_i, is defined as:

$$E_i = S_i \cdot \frac{R_i}{N_i}, \tag{3.7}$$

where i represents the number of agents, S_i is the success rate, R_i is the average reduction in steps compared to a single-agent system, and N_i is the total number of steps taken by the system with i agents.

3.4 Results and Discussion

To evaluate the effectiveness of the proposed strategy, we considered two key aspects: training time and success rate in task execution. Traditional Q-learning was used as a baseline for comparison. The training, simulations, and visualizations were performed using Python on a computer equipped with 16 GB of RAM, a Ryzen 5 5600G processor, and a GeForce RTX 3080 GPU.

The performance of the proposed method was evaluated by initializing all agents with the same Q-table for optimal decision-making. The algorithm's effectiveness was assessed through 100 tests, each involving the dispatch of 10 items to randomly generated locations. Additionally, the system's scalability was analyzed by gradually increasing the number of agents and comparing the results to a scenario with only one agent.

Table 6 – Comparison of Training Time between Q-learning and our Proposed Algorithm Even Third Stage.

Method	# Epochs	Time (h)	**Success Rate %**
QL pure	13,950	12.38	85
QL pure	27,900	23.98	93
Proposed	13,950	1.16	100

3.4.1 Training Time Evaluation

The study's simulations also emphasized the importance of efficient agent training. It was found that using only QL without additional techniques resulted in prohibitively long training times, making multi-agent training impractical. Consequently, training was halted at the third stage. Table 6 presents data on the training process, including episodes, duration, and success rates for book deliveries.

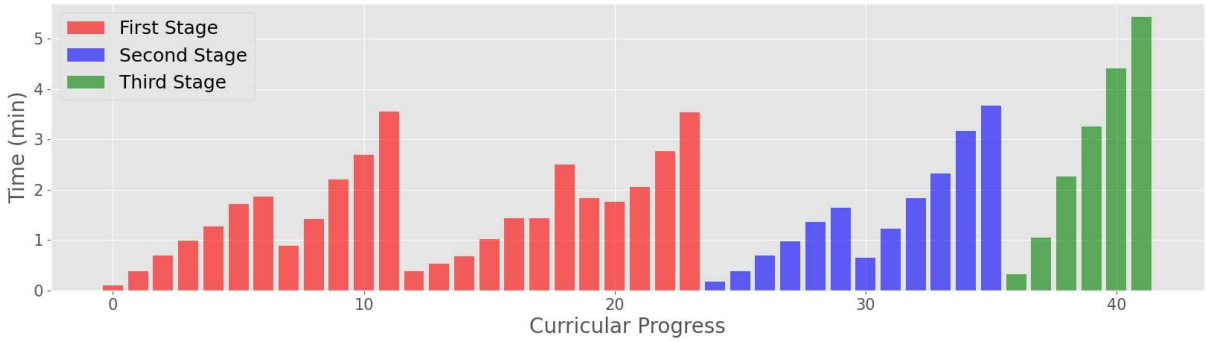


Figure 18 – Time in minutes for each curriculum level up to Stage Three

The proposed algorithm demonstrated superior efficiency, achieving a 100% success rate in approximately 70 minutes, compared to 12.38 hours for traditional QL, which only achieved an 85% success rate. Doubling the episode count for QL increased the training time to 23.98 hours, resulting in a 93% success rate. Thus, the proposed approach is over 20 times faster than traditional QL and over 10 times faster when considering equivalent episode counts.

Figure 18 illustrates the training time of agents under the proposed approach. The TL stage is conducted once at the end of each curriculum level within the first three stages, totaling nine behavior transfers and 45 curriculum levels. For comparison, to train an agent to collect a book, it was necessary to visit 530 states (as shown in Table 7), which took approximately 16 minutes. This corresponds to 6.67% of the total possible state combinations for a single agent to collect a book. Similar efficiency was observed in other stages; for instance, during the first stage of training, the agent visited 1,590 out of 7,950 possible state combinations, representing 20% of the total.

These results highlight the efficiency of the proposed method. In traditional QL approaches, an agent would need to visit all states to explore and discover the optimal policy. Therefore, it can be concluded that the approach based on TL and CL significantly optimizes the agent's learning process, offering greater efficiency and faster training times.

In the final stage, training multiple agents takes 11.67 hours due to the increased complexity of the task. This complexity requires the agents to deliver more books, collaborate with each other, and actively avoid collisions.

Table 7 – The Recognition column represents the states the agent visited during learning. The Augmentation column includes these states and those in the target domain.

	Recognition	Augmentation	
Pickup Operation	212	1590	7950
Delivery Operation	106	530	7950
Return to Bays	-	106	7950
Multi-Agent Collab.	-	-	381,600

Table 8 – Metrics recorded during the task of delivering 10 books over 100 iterations. Columns show the number of agents, average steps per iteration, total steps, and success rate. QL 1 was trained with 13,500 episode and QL 2 with 27,900 episodes.

Method	**# of Agents**	**Steps**	**Total Steps**	**Success Rate (%)**
Ours	1	236.08	236.08	100.00
Ours	2	129.15	258.30	100.00
Ours	3	92.73	278.19	99.00
Ours	4	74.74	298.96	99.00
Ours	5	63.34	316.70	99.00
Ours	6	58.33	349.98	96.00
Ours	7	58.22	407.54	95.00
Ours	8	49.26	394.08	97.00
Ours	9	48.72	438.48	94.00
QL 1	1	246.00	246.00	85.00
QL 2	1	238.00	238.00	85.00

3.4.2 Success Rate Evaluation

Table 8 presents the average step count per agent in the task of distributing 10 books. When a single agent operates independently, the average step count is 236. In contrast, with two collaborating agents, the step count decreases to 129, reflecting a reduction of approximately 45.33%. This reduction follows an inverse correlation with the number of agents involved. The most significant decrease occurs with nine agents, reducing the total step count by approximately 73.36% compared to the single-agent scenario.

As the number of agents increases, the average number of actions per agent decreases, but the total number of actions across all agents increases. For instance, with one agent, there are 236 actions; with two agents, there are 258 actions; and with nine agents, there are 439 actions. This increase occurs because a higher agent density elevates the risk of collisions, requiring agents to prioritize collision avoidance over task completion objectives.

Table 8 presents the average number of actions taken by individual agents to deliver 10 books. The results demonstrate that our proposed algorithm significantly accelerates task completion; by up to 4.84 times. However, increasing the number of agents in the environment does not consistently improve efficiency.

Specifically, introducing more than five agents does not notably reduce the required number of actions. Figure 19 illustrates that efficiency peaks with up to five agents, speeding up task completion by 3.73 times. Beyond this threshold, efficiency diminishes due to increased congestion, which diverts agent focus toward collision avoidance rather than task completion.

A video demonstrating our algorithm performing the book delivery task with two agents is available on our YouTube channel[2]. Figure 13 shows a screenshot from the video, depicting robots engaged in collecting and delivering 10 books within a library environment. Red service bays indicate available books for pickup, while bays matching the robots' color reflect their current activity; either collecting or delivering. Green bays signify locations where no books are available for pickup.

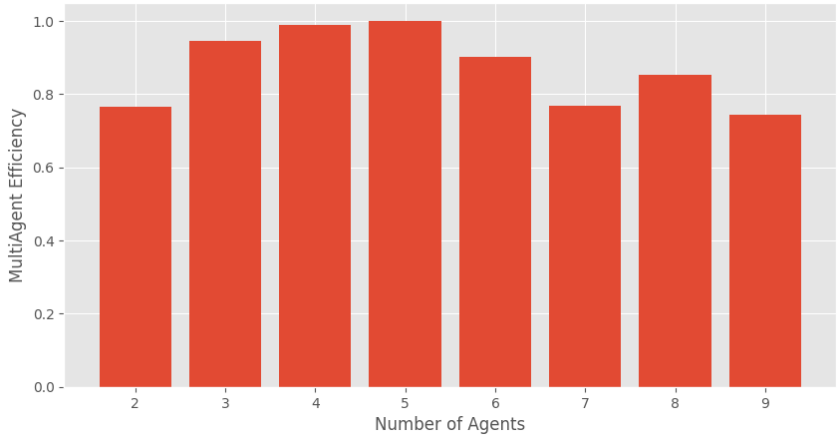


Figure 19 – Efficiency of the multi-agent system based on (3.7) normalized by the maximum value.

In this simulation, each grid cell measures 70×70 units, while the robots occupy 50×50 units. Movement between cells is divided into 50 segments, with 20% allocated to turning and the remaining 80% dedicated to linear movement. This segmentation ensures collision-free navigation within the environment.

At the timestamp 0:33 in the video, the blue agent attempts to reach a cell occupied by another agent, assuming the other robot will vacate the space. Despite their proximity, a collision is avoided due to the size of the cells and the faster rotation speed compared to linear movement. At 2:10, the blue robot picks up a book but initially moves away from its delivery destination instead of toward it. Despite this initial deviation, it corrects its path in the subsequent state and progresses toward its intended destination. This observation indicates that while our approach is successful in scenarios involving two agents, there is still room for further improvement and refinement.

At 2:48, the agents were on a collision course, although our method detects only nearby agents within a 4-neighborhood radius. Upon detecting the yellow agent, the sys-

[2] https://youtu.be/QZkJQLRYqt8

tem reverses its route, giving priority to the blue robot. This sequence begins with the yellow robot making a choice, followed by the blue robot's decision. The collision is successfully avoided at 3:14, but the blue robot remains close behind the yellow one. Aware of this, the yellow robot adjusts its route to proceed toward the goal. By 4:14, the agents had completed 70% of their mission. With no remaining books in the fourth service bay, it turned green. By 5:30, the final agent had completed the task, delivering five out of ten books.

In summary, our proposed algorithm enhances the efficiency of multi-agent systems responsible for book delivery in enclosed spaces. Our study underscores the algorithm's effectiveness, achieving a notable 4.84-fold reduction in the average number of actions required per agent to deliver 10 books. Despite these promising results, including a 94% success rate with nine agents and a 73.36% decrease in task completion steps, further research and refinement in more intricate scenarios are warranted. Nevertheless, our algorithm demonstrates significant potential to enhance navigation and task efficiency in multi-agent logistics, particularly in smart library environments.

3.5 Concluding Remarks

This chapter presented an RL approach for navigating multiple agents in a library environment. The methodology takes advantage of QL, along with TL and CL, to improve performance. The reward function effectively guides agents towards the desired behaviors, reducing collisions with other agents and with bookshelves, thus facilitating task success. The proposed framework incorporates TL by initially training agents on a simplified subset of states and then transferring acquired knowledge to more extensive state spaces, effectively optimizing computational resources. Experimental results demonstrated consistently high success rates exceeding 94%, validating the algorithm's efficiency.

Despite these advantages, performance tends to degrade when scaling to large datasets or highly varied state spaces. Thus, careful consideration of computational limitations is essential when applying this method in broader or more complex contexts. One practical solution is selecting environments and scenarios strategically, such as focusing on after-hours operations in libraries, where human interaction is minimized, and environmental conditions are relatively stable.

Future research directions include exploring the integration of deep reinforcement learning (DRL) methodologies, which utilize neural network-based function approximation to enhance performance and adaptability. Additionally, developing automated curriculum generation methods could facilitate scalability to more extensive and diverse environments. Furthermore, transitioning from a decentralized to a centralized multi-agent system could significantly improve efficiency and coordination among agents.

4 Discussions and Considerations

This dissertation presented a progressive exploration of the application of Q-learning for UGV path planning in logistics environments. Although the preceding chapters detailed the methodologies and results from each publication, this chapter provides a space for a more in-depth analysis, addressing the nuances, limitations, and broader context of the research, in line with the discussions raised during the evaluation of this work.

4.1 Comparative Analysis and the Notion of "Optimality"

A central question raised was the fairness and interpretation of the comparisons between Q-learning and classical algorithms such as Dijkstra's and A*. It is essential to clarify that the objective was never to prove the superiority of Q-learning in finding the geometrically shortest path—a task for which Dijkstra, by its deterministic nature, guarantees an optimal solution. The intention was, in fact, to demonstrate the flexibility of Q-learning in solving a *multi-objective* optimization problem.

Our reward function was designed to penalize not only the path length (via the *living penalty*) but also the number of turns. This approach aims to generate smoother trajectories, which are more energy-efficient and less mechanically strenuous for a differential-drive robot. Standard implementations of Dijkstra's and A* algorithms optimize for a single cost: distance. Therefore, the comparison is not between two methods solving the same problem, but rather between a multi-objective method (Q-learning) and a shortest-path method (Dijkstra). The result—a path with fewer turns but slightly longer—should be interpreted from this perspective: the agent learned to sacrifice minimal distance in favor of smoothness, as instructed by the reward function.

Additionally, in the multi-target scenario, the approach used for the classical algorithms was that of a greedy heuristic: visiting the nearest destination first. As was rightly pointed out, this is not a solution to the Traveling Salesman Problem (TSP). Q-learning, by learning a policy over a state space that includes the already visited targets, was implicitly optimizing the visitation order along with the path. A more direct comparison would require combining Dijkstra's algorithm with a TSP solver, which would constitute a hybrid architecture and a distinct research scope.

4.2 Scope, Generalization, and Limitations of the Approach

It is crucial to acknowledge the inherent limitations of scope and generalization in the methods developed herein.

The Q-learning models presented were trained and validated for the specific layouts studied, primarily based on the UFV library. The tabular Q-learning approach used does not possess generalization capabilities; a change in the map, such as the addition or removal of an obstacle (e.g., a shelf), would require a complete retraining of the agent.

This is a known limitation of tabular methods. Application in dynamic environments or across a variety of different layouts would necessitate a transition to function approximation approaches, such as Deep Q-Networks (DQN), where the agent learns from environmental features rather than memorizing the value of each discrete cell.

The decision to restrict the agent's movements to orthogonal actions (forward, backward, left, right) and in-place rotations was a deliberate simplification to make training feasible. As discussed, the inclusion of diagonal movements would double the action space, exponentially increasing the training complexity. Furthermore, in narrow corridors like those in the library, diagonal movements could increase the risk of collision, depending on the robot's exact position within the grid cell.

However, we recognize that in open areas, the absence of diagonal movements prevents the robot from following the shortest Euclidean path. The investigation of an expanded action space, perhaps with a reward function that penalizes diagonal movements near obstacles, is a clear opportunity for future work.

4.3 Multi-Agent Scenario and Practical Application

The third article of this dissertation, focusing on a multi-agent system, represents the most ambitious step and, consequently, the one furthest from immediate practical implementation.

The coordination strategy used, based on a shared Q-table and the detection of imminent collisions, is a fundamental approach. A robust fleet management system, such as the *Open-RMF* framework, utilizes far more complex architectures. These include, for example, task allocation via auction (where robots "bid" to perform a task based on their estimated cost) and negotiation protocols to resolve conflicts at intersections. Our approach should be seen as a proof-of-concept for the feasibility of reinforcement learning for coordinated navigation, rather than a complete fleet management solution.

The simulation of the library environment inevitably contains simplifications. The "teleportation" of the robot via an elevator is the most obvious one. In a real-world scenario, the elevator would be an agent or a resource to be managed. The robot would

need to communicate with the elevator's system to call it, enter, request the destination floor, and exit. As was pertinently pointed out, a critical challenge would be the potential loss of wireless communication inside the elevator car (a Faraday cage), which could disrupt the entire coordination system and would require the robot to have the autonomy to complete the transport cycle and re-establish communication.

The results showing a decline in the success rate as the number of agents increases (from 100% with two agents to 94% with nine) are of utmost importance. They reveal a saturation point: adding more robots to a fixed-size environment increases congestion to a point where agents spend more time avoiding collisions or stuck in "traffic jams" than performing tasks. This demonstrates that the solution to increasing productivity is not simply to add more units, but rather to optimize the existing fleet and, possibly, redesign the workflows in the environment. The performance reduction with an increased fleet size is a realistic and valuable finding for planning any automated logistics operation.

4.4 Insights and Perspectives

Currently, the energy expenditure is estimated based on the number of 90-degree turns executed by the robot. However, this approach only provides a partial view of the system's dynamics. A more comprehensive metric could be derived by integrating the individual wheel motions to represent the overall movement of the UGV.

In this regard, the norm of the individual wheel velocities can serve as an energy-related measure. The linear and angular velocities can be converted into the respective wheel velocities, which are then squared, summed, and square-rooted to yield a single scalar value. By evaluating this metric over time, it becomes possible to compare the energy demand of different controllers and identify which strategies result in smoother or more efficient navigation.

Such an analysis would provide valuable insights into the relationship between control performance and energy consumption, supporting the design of controllers that optimize both accuracy and efficiency in mobile robot navigation. Moreover, this metric would allow for fairer comparisons among different path-planning strategies such as the proposed method, Dijkstra, A*, and BFS. By considering not only trajectory optimality and execution time but also the energy required to complete the same navigation task, future studies could establish a more balanced evaluation that integrates kinematic efficiency with energetic performance.

5 Concluding Remarks

This dissertation investigates the effectiveness of Q-learning-based path planning methodologies within logistics environments, particularly focusing on libraries and warehouse scenarios. The primary objective was to explore how reinforcement learning techniques, especially Q-learning, can enhance the efficiency and adaptability of UGVs compared to traditional path planning algorithms such as Dijkstra's and A*. To address the formulated research question, three main hypotheses were evaluated:

First, the hypothesis (**H1**), stating that Q-learning-based path planning generates more efficient navigation paths was confirmed. Experimental results showed that Q-learning consistently produced smoother, shorter, and less complex navigation paths with fewer turns, significantly outperforming conventional methods. Although traditional algorithms like Dijkstra's are effective at computing the shortest paths, they lack flexibility in dynamic or multi-goal scenarios. The Q-learning approach demonstrated a clear advantage in optimizing paths for practical logistics tasks, balancing distance, energy consumption, and vehicle maneuverability.

Second, the integration of Transfer Learning and Curriculum Learning (**H2**) was found to markedly improve the adaptability and convergence speed of Q-learning in dynamic environments. By systematically structuring learning tasks from simpler to more complex scenarios, these auxiliary learning techniques facilitated quicker convergence and enhanced the UGV's capability to adapt to changing environmental conditions. This advancement allowed the agents to achieve higher success rates, faster planning times, and optimal routing compared to baseline reinforcement learning approaches without these techniques.

Third, the hypothesis concerning the scalability and efficiency of multi-agent Q-learning (**H3**) was validated. The multi-agent setup demonstrated significant improvements in overall task completion time, operational efficiency, and navigation success rates. By coordinating multiple autonomous vehicles simultaneously, the proposed multi-agent Q-learning framework substantially reduced bottlenecks and optimized resource allocation, making it highly suitable for large-scale logistics applications.

The main contributions of this dissertation can be summarized as follows. First, it presents the development of a robust Q-learning-based methodology for multi-objective path planning in logistics scenarios. Second, it introduces significant enhancements through the integration of Transfer Learning and Curriculum Learning, which improve performance and adaptability in complex and dynamic environments. Third, it provides a comprehensive comparative analysis showing that Q-learning-based methods perform as well

as traditional navigation algorithms while offering greater flexibility and learning capabilities. Finally, it includes real-world validation of the proposed approach using autonomous ground vehicles, such as the Pioneer 3-DX.

Overall, this dissertation highlights the substantial benefits of adopting reinforcement learning, specifically Q-learning, to overcome traditional limitations in path planning for logistics environments. The proposed methods effectively address key challenges such as collision avoidance, multi-goal optimization, and efficient navigation, providing a robust and adaptable framework suitable for practical implementations in libraries, warehouses, and similar structured environments. Future research should explore further enhancements in multi-agent coordination strategies and real-time adaptability, extending these findings to more complex and larger-scale applications.

Bibliography

AGGARWAL, S.; KUMAR, N. Path planning techniques for unmanned aerial vehicles: A review, solutions, and challenges. **Computer Communications**, Elsevier, v. 149, p. 270–299, 2020.

ARINEZ, J. F. et al. Artificial intelligence in advanced manufacturing: Current status and future outlook. **J. of Manufacturing Science and Engineering**, American Society of Mechanical Engineers Digital Collection, v. 142, n. 11, 2020.

ARULKUMARAN, K. et al. Deep reinforcement learning: A brief survey. **IEEE Signal Processing Magazine**, IEEE, v. 34, n. 6, p. 26–38, 2017.

ASADI, K. Strengths, weaknesses, and combinations of model-based and model-free reinforcement learning. **Department of Computing Science University of Alberta**, 2015.

BATISTA, H. O. B. et al. Multi-goal robot path planning based on q-learning for library logistics. In: SOCIEDADE BRASILEIRA DE AUTOMAçãO. **2024 XXV Congresso Brasileiro de Automática (CBA)**. Rio de Janeiro, Brazil, 2024.

BATISTA, H. O. B. et al. Q-learning-based multi-objective global path planning for ugv navigation*. In: IEEE. **2024 Latin American Robotics Symposium (LARS)**. Arequipa, Peru, 2024. p. 1–6.

BELANCHE, D. et al. Service robot implementation: a theoretical framework and research agenda. **The Service Industries Journal**, Taylor & Francis, v. 40, n. 3-4, p. 203–225, 2020.

BRANDÃO, A. S.; SARCINELLI-FILHO, M.; CARELLI, R. An analytical approach to avoid obstacles in mobile robot navigation. **International Journal of Advanced Robotic Systems**, SAGE Publications Sage UK: London, England, v. 10, n. 6, p. 278, 2013.

BUONO, A. et al. Modified q-learning algorithm for mobile robot real-time path planning using reduced states. **Jurnal RESTI (Rekayasa Sistem dan Teknologi Informasi)**, v. 7, n. 3, p. 628–636, 2023.

CARVALHO, K. B. de et al. Q-learning based local path planning for uavs with different priorities. In: IEEE. **2023 Latin American Robotics Symposium, 2023 Brazilian Symposium on Robotics, and 2023 Workshop on Robotics in Education**. Salvador, Brazil, 2023. p. 89–94.

CHANG, L. et al. Reinforcement based mobile robot path planning with improved dynamic window approach in unknown environment. **Autonomous Robots**, Springer, v. 45, p. 51–76, 2021.

CHUNG, J. J. et al. A multiagent framework for learning dynamic traffic management strategies. **Autonomous Robots**, Springer, v. 43, p. 1375–1391, 2019.

DIJKSTRA, E. W. **A note on two problems in connexion with graphs**. 2022. 287–290 p.

FAGUNDES-JUNIOR, L. A. et al. Machine learning for unmanned aerial vehicles navigation: An overview. **SN Computer Science**, Springer, v. 5, n. 2, p. 1–26, 2024.

FLOOD, M. M. The traveling-salesman problem. **Operations research**, INFORMS, v. 4, n. 1, p. 61–75, 1956.

FOEAD, D. et al. A systematic literature review of a* pathfinding. **Procedia Computer Science**, v. 179, p. 507–514, 2021. ISSN 1877-0509. 5th International Conference on Computer Science and Computational Intelligence 2020.

HART, P. E.; NILSSON, N. J.; RAPHAEL, B. A formal basis for the heuristic determination of minimum cost paths. **IEEE Transactions on Systems Science and Cybernetics**, v. 4, n. 2, p. 100–107, 1968.

HU, Y.; YANG, L.; LOU, Y. Path planning with q-learning. **Journal of Physics: Conf. Series**, IOP Publishing, v. 1948, n. 1, p. 012038, jun 2021.

JACOB, F. et al. Picking with a robot colleague: A systematic literature review and evaluation of technology acceptance in human–robot collaborative warehouses. **Computers & Industrial Engineering**, v. 180, p. 109262, 2023. ISSN 0360-8352.

JANG, J.-S. Anfis: adaptive-network-based fuzzy inference system. **IEEE transactions on systems, man, and cybernetics**, IEEE, v. 23, n. 3, p. 665–685, 1993.

KHATIB, O. Real-time obstacle avoidance for manipulators and mobile robots. **The international journal of robotics research**, Sage Publications Sage CA: Thousand Oaks, CA, v. 5, n. 1, p. 90–98, 1986.

KHRIJI, L. et al. Mobile robot navigation based on q-learning technique. **Int. Journal of Advanced Robotic Systems**, v. 8, n. 1, p. 4, 2011.

LI, W. et al. Neural network-based dynamic target enclosing control for uncertain nonlinear multi-agent systems over signed networks. **Neural Networks**, Elsevier, v. 184, p. 107057, 2025.

LIU, J. On the convergence of reinforcement learning with monte carlo exploring starts. **Automatica**, Elsevier, v. 129, p. 109693, 2021.

LIU, L. et al. Path planning techniques for mobile robots: Review and prospect. **Expert Systems with Applications**, v. 227, p. 120254, 2023. ISSN 0957-4174.

LOW, E. S.; ONG, P.; CHEAH, K. C. Solving the optimal path planning of a mobile robot using improved q-learning. **Robotics and Autonomous Systems**, v. 115, p. 143–161, 2019. ISSN 0921-8890.

MEKALA, S. Comparing pathfinding agents: Heuristic vs. reinforcement learning. **Reinforcement Learning (August 01, 2024)**, 2024.

NARVEKAR, S. et al. **Source task creation for curriculum learning**. 2016. 566–574 p.

OLIVEIRA, I. R. L. D.; CARVALHO, K. B. D.; BRANDãO, A. S. Curriculum-based reinforcement learning for an effective multi-agent path planning algorithm in warehouse scenarios. In: IEEE. **2023 15th IEEE Int. Conf. on Industry Applications**. São Bernardo do Campo, Brazil, 2023. p. 471–477.

RIO, A. D.; JIMENEZ, D.; SERRANO, J. Comparative analysis of a3c and ppo algorithms in reinforcement learning: A survey on general environments. **IEEE Access**, IEEE, 2024.

SANIUK, S.; SANIUK, A.; CAGÁŇOVÁ, D. Cyber industry networks as an environment of the industry 4.0 implementation. **Wireless Networks**, Springer, v. 27, p. 1649–1655, 2021.

SHARMA, K.; DORIYA, R. Coordination of multi-robot path planning for warehouse application using smart approach for identifying destinations. **Intelligent Service Robotics**, Springer, v. 14, p. 313–325, 2021.

SONNY, A.; YEDURI, S. R.; CENKERAMADDI, L. R. Q-learning-based unmanned aerial vehicle path planning with dynamic obstacle avoidance. **Applied Soft Computing**, v. 147, p. 110773, 2023. ISSN 1568-4946.

WATKINS, C. J.; DAYAN, P. Q-learning. **Machine learning**, Springer, v. 8, p. 279–292, 1992.

XUE, Y.; SUN, J.-Q. Solving the path planning problem in mobile robotics with the multi-objective evolutionary algorithm. **Applied Sciences**, MDPI, v. 8, n. 9, p. 1425, 2018.

YANG, Y.; JUNTAO, L.; LINGLING, P. Multi-robot path planning based on a deep reinforcement learning dqn algorithm. **CAAI Transactions on Intelligence Technology**, Wiley Online Library, v. 5, n. 3, p. 177–183, 2020.

ZHU, Z. et al. Transfer learning in deep reinforcement learning: A survey. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 45, n. 11, p. 13344–13362, 2023.