AKADEMIA GÓRNICZO-HUTNICZA IM. STANISŁAWA STASZICA W KRAKOWIE

Nauki inżynieryjno-techniczne

Automatyka, Elektronika, Elektrotechnika i Technologie Kosmiczne

ROZPRAWA DOKTORSKA

Inteligentny system sterowania robotem mobilnym

Autor: Ravi Raj

Promotor rozprawy: Prof. dr hab. inż. Andrzej Kos

Praca wykonana: Akademia Górniczo-Hutnicza im. Stanisława Staszica w Krakowie
Wydział Informatyki, Elektroniki i Telekomunikacji

Kraków, 2024

AKADEMIA GÓRNICZO-HUTNICZA IM. STANISŁAWA STASZICA W KRAKOWIE

Engineering and technology

Automation, Electronics, Electrical Engineering and Space Technologies

DOCTORAL DISSERTATION

Intelligent Control System for Mobile Robot

Author: Ravi Raj

Supervisor: Prof. dr hab. inż. Andrzej Kos

Completed at: AGH University of Krakow
Faculty of Computer Science, Electronics and Telecommunications

Kraków, 2024

Acknowledgments

The support and direction of several individuals are crucial to the success of this project, aside from my endeavors. I would like to take this time to thank everyone who has helped me to make this project a triumph.

First and foremost, I offer my sincerest gratitude and would like to show my greatest appreciation to my supervisor, Prof. Andrzej Kos. During my doctoral studies, Prof. Kos has offered me amazing experiences along with his supervision, insightful advice, and unwavering support. Whenever I go to his meeting, I come away inspired and driven. Without his encouragement, guidance, and kind support, this project would not have materialized.

Finally, I wish to offer a special thanks for accepting my absence for four years of living abroad to my beloved wife, Kumkum, who has stood by me through all my travails, my absences, and my fits of pique and impatience. She gave me moral support. She also supported the family and took care of my son during much of my Ph.D. studies. Along with her, I want to acknowledge my beloved son, Deepesh. He is a good boy and a great source of love and relief from scholarly endeavors. I am particularly appreciative of both of my parents, Devendra Pandey and Malti Devi, for their love and emotional assistance. I have always known that you thought highly of me and wished for my success. I appreciate you educating me that the purpose of life is to study, to be content, and to become acquainted with oneself since it is only through doing so that I may know and comprehend others. To my entire family, thank you for your love and encouragement. During my dissertation, I pursued a prestigious degree in academia with their support and understanding.

Krakow, Poland *Ravi Raj*

May, 2024

"This work is dedicated to Prof. Andrzej Kos, who gave me an outstanding opportunity to learn and work under his supervision, and who was always supportive and present there for me, even on the tough days during the journey of my doctoral studies."

List of Contents

Abstract

Intelligent Control System for Mobile Robot

Ravi Raj

Modern mobile robots are developed to assist or substitute human personnel in complex control and planning operations and jobs, including object manipulation, expert assistance in a range of sectors, outdoor navigation, security surveillance, fire-fighting in unknown terrain exploration, and urban area driving. Even for those with specialized training in robot coding, developing a control structure for the robots that are used to carry out these activities is often a challenging approach, requiring the generation of a unique controller by hand for every specific operation. The developer must intentionally consider the wide variety of scenarios that the robot might experience in difficult situations. It might prove more beneficial for the robot to discover how to perform certain activities on its own rather than having to be already programmed for every activity. This dissertation examines the learning technique of robots as well as addresses difficulties related to the intelligent control system of mobile robots for autonomous navigation. We examine how the mobile robot acquires knowledge through expert presentations. This approach is based on the natural human tendency to imitate. When mobile robots are offered instances of conventional actions, they can gain information from this data and apply their knowledge to all possible scenarios that are not included in the instances. Using an artificial neural network, we integrated the inference function inside the robot controls. The mobile robot will acquire knowledge of how to navigate on its own after an appropriate number of instances.

We study the independent learning capability of mobile robots in this dissertation in the absence of trained demos for autonomous navigation. Modern reinforcement learning algorithms are employed in this study for training mobile robots through interactions

with mobile robots. We analyze a mobile robot through simulation that uses reinforcement learning to acquire information about possible rewards in different contexts. Additionally, an artificial neural network has been integrated to perform the quick generalization function. The robots need to try to comprehend the fundamental principles and rewards of the expert demos in this experiment, in addition to learning how to associate with states and activities. In comparison to conventional techniques, we are assisting the learning convergence in a much shorter quantity of episodes by using all the previous state-action pairings that were recorded through the interaction process to train the mobile robot. Based on this proposed technique, experimental findings demonstrated reliable and accurate effectiveness in autonomous navigation tasks for mobile robots. We therefore suggest that the advancement of mobile robot learning technology, compared to conventional robot programming, has a promising future ahead of it and will be beneficial and serve us more effectively.

Streszczenie

Inteligentny system sterowania robotem mobilnym

Ravi Raj

Nowoczesne roboty mobilne opracowano, aby pomagać lub zastępować personel ludzki w złożonych operacjach kontrolnych i planistycznych oraz zadaniach, w tym manipulacji obiektami, pomocy eksperckiej w różnych sektorach, nawigacji zewnętrznej, nadzorze bezpieczeństwa, gaszeniu pożarów eksploracji nieznanego terenu i prowadzeniu pojazdów po obszarach miejskich. Nawet dla osób posiadających specjalistyczne przeszkolenie w zakresie kodowania robotów opracowanie struktury sterującej dla robotów używanych do wykonywania tych czynności jest często trudnym podejściem, wymagającym ręcznego wygenerowania unikalnego sterownika dla każdej konkretnej operacji. Twórca musi celowo wziąć pod uwagę szeroką gamę scenariuszy, których robot może doświadczyć w trudnych sytuacjach. Dla robota korzystniejsze może okazać się odkrycie, jak samodzielnie wykonywać określone czynności, zamiast konieczności programowania go do każdej czynności. Niniejsza rozprawa doktorska bada technikę uczenia się robotów, a także uwypukla trudności związane z inteligentnym systemem sterowania robotami mobilnymi do autonomicznej nawigacji. Badamy, jak robot mobilny zdobywa wiedzę poprzez prezentacje eksperckie. Podejście to opiera się na naturalnej ludzkiej skłonności do naśladowania. Kiedy robotom mobilnym oferuje się przykłady konwencjonalnych działań, mogą one uzyskać informacje z tych danych i zastosować swoją wiedzę do wszystkich możliwych scenariuszy, które nie są uwzględnione w tych przypadkach. Korzystając ze sztucznej sieci neuronowej, zintegrowaliśmy funkcję wnioskowania ze sterownikami robota. Robot mobilny po odpowiedniej liczbie ćwiczeń nabędzie wiedzę dotyczącą samodzielnego poruszania się.

W tej rozprawie badamy zdolność niezależnego uczenia się robotów mobilnych przy braku przeszkolonych demonstracji autonomicznej nawigacji. W tym badaniu wykorzystano nowoczesne algorytmy uczenia się przez wzmacnianie do szkolenia robotów mobilnych poprzez interakcje z robotami mobilnymi. Analizujemy robota mobilnego poprzez symulację, która wykorzystuje uczenie się przez wzmacnianie w celu uzyskania informacji o możliwych osiągnięciach w różnych kontekstach. Dodatkowo zintegrowano sztuczną sieć neuronową realizującą funkcję szybkiej generalizacji. Roboty muszą zrozumieć podstawowe zasady i efekty wynikające z demonstracji ekspertów w tym eksperymencie, a także nauczyć się kojarzenia ze stanami i czynnościami. W porównaniu do technik konwencjonalnych, wspomagamy konwergencję uczenia się w znacznie krótszej liczbie odcinków, wykorzystując wszystkie poprzednie pary stanu i działania, które zostały zarejestrowane w procesie interakcji w celu szkolenia robota mobilnego. W oparciu o proponowaną technikę wyniki eksperymentów wykazały niezawodną i dokładną skuteczność w zadaniach autonomicznej nawigacji dla robotów mobilnych. Sugerujemy zatem, że rozwój technologii uczenia się robotów mobilnych, w porównaniu z konwencjonalnym programowaniem robotów, ma przed sobą obiecującą przyszłość, będzie korzystny i będzie nam służył efektywniej.

Abbreviations

ADAS- Advanced Driver Assistance System

AI- Artificial intelligence

AGV- Autonomous guided vehicle

AMR- Autonomous mobile robot

ANN- Artificial neural network

BPNN- Backpropagation neural network

CNN- Convolutional neural network

DL- Deep learning

DNN- Deep neural network

DP- Dynamic programming

DQN- Double Q-learning network

DRL- Deep reinforcement learning

FFNN- Feed-forward neural network

FOV- Field of view

GPI- Generalized policy iteration

GPS- Global positioning system

IR- Infrared

MDP- Markov decision process

ML- Machine learning

MR- Mobile robot

NN- Neural network

NNQL- Neural network-based Q-learning

PI- Policy iteration

POMDP- Partial observable Markov decision process

PSO- Particle Swarm Optimization

RL- Reinforcement learning

SGD- Stochastic gradient descent

SLAM- Simultaneous localization and mapping

TD- Temporal difference

UAV- Unmanned aerial vehicle

UGV- Unmanned ground vehicle

VI- Value iteration

WMR- Wheeled mobile robot

$\mathbb{N}$- Natural number

$\mathbb{R}$- Real number

Chapter 1
Introduction

1.1. Background and Motivation

1.1.1. Mobile Robots

A robot is a technological device, particularly one that can be programmed by a computer and is capable of performing a complex variety of tasks autonomously. A robot might be controlled by an internal control system or by an external supervision system. While some robots have been designed to mimic human shapes, the majority of robots are task-executing devices that prioritize utility over creative design. Robots are evolving quickly from industrial settings where they are physically confined to restricted workspaces to more sophisticated devices that can carry out difficult tasks in our everyday lives. These days, autonomous robots are becoming more useful in industrial and commercial scenarios. Conventional industrial robots, including manipulators and robotic arms, are mostly static and are utilized in production facilities with strictly regulated environments.

As artificial intelligence (AI) technology develops quickly, MRs are changing and getting better at jobs that were too difficult or complicated to complete in the past, such as security, heavy-weight transport, exploration of space, and several more. Mobile robots (MRs), on the other hand, are robotic systems designed to be able to perform tasks in uncontrolled surroundings and possess the capacity to navigate freely using devices, including wheels. With developments and investigations in the domains of electronics, computer programs, artificial intelligence (AI), computers, scientific study, and technological advances, modern robotics companies are developing increasingly sophisticated MRs [1]. An MR is a unique kind of

software-controlled device that can identify its surroundings and carry out a predetermined objective by using sensors and additional tools, including a camera, Lidar, and many more. To complete a predetermined job, an AMR usually comprises a total of three actions: perception (sensing), plan and inference (procedure), and mobility (act).

(a) MAARS robot: The military patrol robot [2]

(b) NASA Mars rover: The space exploration robot [3]

(c) Neerakshi: An AUV for mine detection [4]

(d) ABB YuMi robot: The medical assistance robot [5]

Fig. 1.1: Several applications of mobile robots in different fields

Excellent examples of the MRs are drones, humanoid robots, entertainment pets, unmanned rovers, and so on. The potential of the AMR to operate independently and their cognitive skills, which allow them to respond and make decisions depending on their perception of their surroundings, set them apart from other robots. MRs are frequently employed in many

different sectors, including military operations and surveillance (see Fig. 1.1 (a)), space exploration (see Fig. 1.1 (b)), underwater exploration (see Fig. 1.1 (c)), and medical assistance (see Fig. 1.1 (d)). Developments in AMRs additionally provide solutions for difficult jobs that were previously thought to be exclusively human-attainable.

The traditional MR navigational method lacks the capacity for independent learning. MRs need to have the skill competent to independently traverse unfamiliar terrain while avoiding stationary and dynamic obstacles. The variety of basic issues that MRs can address with ease ranges from challenging computational assignments to wonderfully logical ones. This wide range of issues is undoubtedly one of the most important and exciting aspects of MRs. Many heuristic and traditional techniques are applied to the development of MRs' operational methodology. The way MRs operate within tropical regions is, for academics, a particularly crucial aspect of developing them for practical applications. When situations get more complicated, traditional methods can get antiquated and might no longer yield the optimal outcomes. The MR usually proceeds via three stages to do a predefined task: mobility (activity), planning of paths and synthesizing (procedure), and perceptions (monitoring). An essential stage in MR travel is determining the most efficient paths to take in order to arrive at the destination while avoiding obstacles. To start experimental research on MR autonomous navigation, it is necessary to study the brief history of MR evolution up to the current state-of-the-art study.

1.1.1.1. History of Mobile Robots

Numerous studies have been carried out on the subject of robotics science throughout the past century. The Czech playwright Karel Capek's "Rossum's Universal Robots" from 1920 is credited with introducing the term "robot" to people worldwide [6]. An extensive study has been conducted on the subject of robotics technology over the past half-century. In 1942, author

Isaac Asimov introduced the term "robotics" in the small chapter "Runabout." The place of robots in human civilization was better articulated by Asimov. The operational laws of MRs have been outlined by Asimov. Asimov proposed three laws for robotics, and all of them remain relevant today [7].

1) **First Law:** A robot must never intentionally injure a human person or allow one to do so by remaining still.

2) **Second Law:** Except in situations where the first law is breached, robots ought to perform the tasks assigned to them by humans.

3) **Third Law:** Save for situations when the first and second rules collide with the circumstances, a robot should attempt to preserve itself.

The field of cybernetics was founded by American mathematician Norbert Wiener, who made substantial contributions to the advancement of MRs. The design and development of autonomous robots is significantly aided by cybernetics.

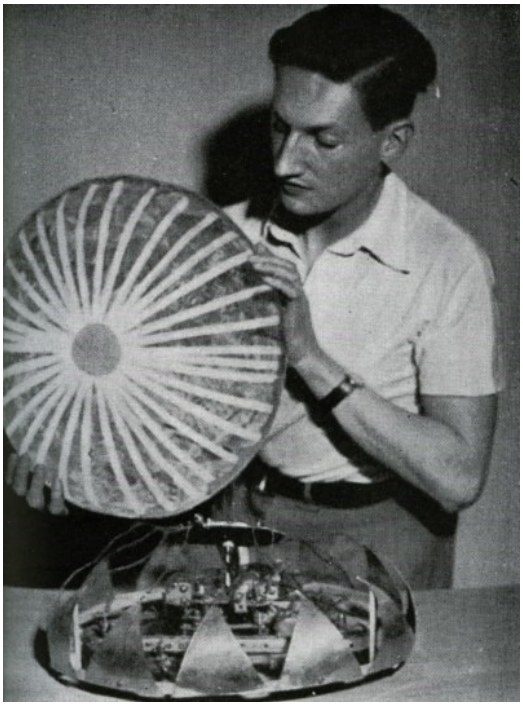

Fig. 1.2: A picture of Miso-1, Albert Ducrocq's robot, from the 1950s [8]

Information theory was originally developed by American electrical engineer Claude Elwood Shannon. In 1950, Shannon's mouse was invented. Shannon's mouse looked like a robotic mouse with an electromechanical relay network that allowed it to navigate through a labyrinth. This mouse is credited with being the first AI gadget. In the 1950s, the Ducrocq family designed a remote-controlled automobile and computerized animal, which is known as Miso. The cybernetic creature Miso evolved in five distinct versions, which are referred to as M1, M2, M3, M4, and M5, respectively [8]. The robot Miso-1 and its designer, Albert Ducrocq, are seen in Fig. 1.2, which was taken in the 1950s.

Fig. 1.3: Shakey the Robot and Charles Rosen in 1983 [9]

The applications of software technology and cybernetics within the domain of robotics science lead to the creation of autonomous and intelligent robots. Without including Shakey the robot, which originated in the late 1960s, it is impossible to depict the history of MRs. This was the first MR with the ability to perceive and influence its environment. The Defense Advanced Research Projects Agency (DARPA) provided funding to the engineers who worked at Stanford Research Institute (SRI) under the guidance of Charles Rosen to develop the Shakey robot [9]. Fig. 1.3 shows a picture of Charles Rosen and Robot the Shakey.

As research and development in the domains of electronics, software, computers, AI, and sciences continues, modern robotics enterprises are turning out increasingly sophisticated MRs. These days, AMRs are more useful in industrial and production settings. The academic field of AI was founded in 1956. Research in AI often aims to address the advancements in information technology, perception, acquisition, reasoning, participation, organizing, natural language processing, autonomous movement, and object manipulation [10]. AI and robotics, together, constitute an extremely effective combination for automating tasks both inside and outside of a commercial setting. The finest examples of contemporary robots that employ AI technology for how they operate are UAVs. Due to the advancements in AI technology, MRs have become increasingly sophisticated and are currently used in a wide range of applications, including mail delivery, medical treatment, pattern recognition, defense, freight, security, infrastructure assessment, passenger transit, and many more. Engineered Arts designed "Ameca," the most sophisticated and smart humanoid robot that debuted during the Consumer Electronics Show (CES) in January 2022 [11]. This robot incorporates cameras in the eyes and facial recognition technologies. Ameca is able to communicate in a language as well as comprehend it. The robot "Ameca" picture is displayed in Fig. 1.4.

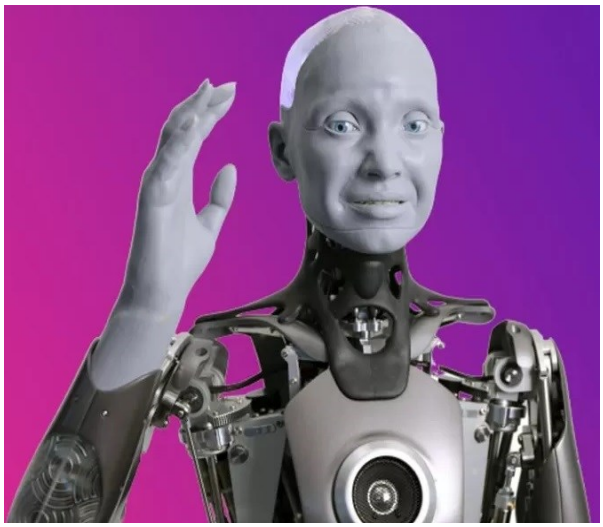

Fig. 1.4: Image of Ameca humanoid robot [11]

1.1.1.2. Types of Mobile Robots

The most widely recognized definition of a robot is: "A robot serves as a multipurpose, reprogrammable manipulator developed to maneuver substances, instruments, and particular devices or elements via movements incorporated with parameter programming in the execution of numerous jobs," according to the RIA's (Robot Institute of America's) [12]. The first problem with movement for MRs is locomotion. Most of the MRs generally work in well-known, regulated locations like factories, shopping malls, and other places, but there are times when they must move in hazardous, adverse, complex, and unknown settings, including space exploration, border surveillance, firefighting, and many more. A key component of an MR design is its locomotion system, which is dependent on a variety of technical factors, including terrain situations, maneuverability, controllability, stability, and so forth, in addition to the robot's intended mode of mobility (e.g., air, water, or land). Based on its locomotion, a robot can generally walk, roll over, skate, run, jump, slide, fly, and swim. MRs have been classified into the following main groups based on their locomotion [13]:

1.1.1.2.1. Stationary Mobile Robot

1.1.1.2.2. Land-based Mobile Robot

- **1.1.1.2.2.1.** Wheeled Mobile Robot
- **1.1.1.2.2.2.** Legged or walking Mobile Robot
- **1.1.1.2.2.3.** Tracked Mobile Robot

1.1.1.2.3. Air-based Mobile Robot

1.1.1.2.4. Water-based Mobile Robot

These above-mentioned MRs are discussed in more detail with their advantages and disadvantages as follows:

1.1.1.2.1. Stationary Mobile Robot

Flexible-arm robotics that does repeated stationary activities are referred to as stationary mobile robots. Since all of these robots work in three dimensions, their arm tooling must be located using a built-in computer that determines every arm joint's position. These robots perform well in settings where repetition is essential to the operation, and they frequently work in an efficient manner. Examples of this kind are industrial robots and manipulators. The robots have a stable foundation and are made up of a free kinematic chain that includes an end-effector that is primarily equipped with specialized tools that allow it to execute operations like assembling, painting, welding, machining, and other jobs in addition to handling things. This category of robots includes Comau, Wittman, Kawasaki, Fanuc, Kuka, Abb, and so on. Grasping equipment is another crucial type of stationary robotic system. Grasping is an essential component of handling, and initially, the primary purpose of grasping gadgets was to assist people in handling duties, these gadgets offer two types of solutions: instruments and prostheses. Nowadays, many industries, including manufacturing and agriculture, are now using grasping equipment, and as a result of this demand, various robotic arms and finger mechanisms have been invented [14].

1.1.1.2.2. Land-based Mobile Robot

Unmanned Ground Vehicles (UGVs) are the typical term used to describe land or house mobile robots. These robots can be further classified into the following categories:

1.1.1.2.2.1. Wheeled Mobile Robot

A WMR is a vehicle with wheels that can move on its own without the assistance of a human operator since it is outfitted with a motor powered by an embedded computer [15].

Wheels are among the most crucial sections for robot mobility, and AMRs are an essential part of a complex area of mobile robotics studies that depends on concepts including signal-image processing and pattern recognition. These will be essential for distribution, logistics, and transportation. For robots operating on smooth, non-rugged surfaces, the application of wheels becomes more simple compared to the use of legs or treads and is additionally relatively easy to develop, construct, and program. Compared to alternative options, wheel control is simpler and results in minimal surface deterioration when wheels are moved. An additional benefit is that, because the robot is generally in touch with the ground, WMRs do not pose a significant risk for balance problems.

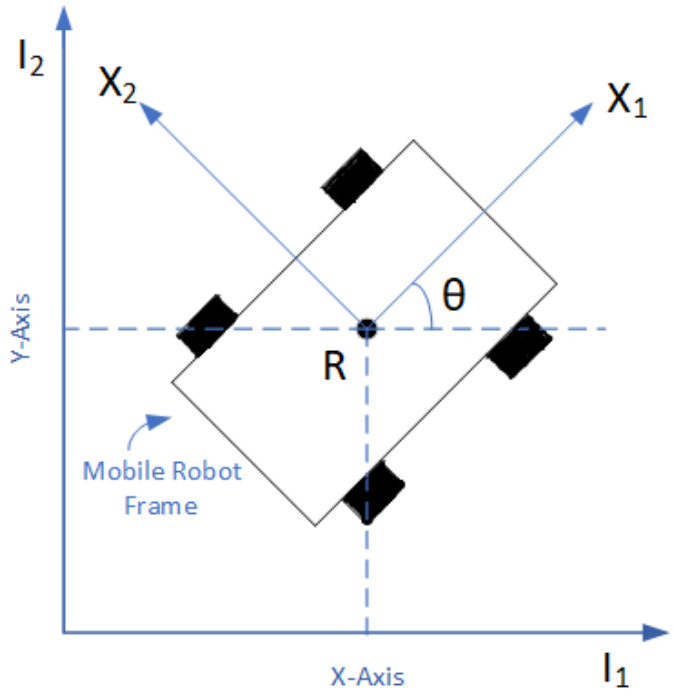

Fig. 1.5: Postural definition of wheeled mobile robot

The most basic state space model that may provide a comprehensive explanation for WMR is the postural kinematic framework. Fig. 1.5 describes where the MR is located on the XY plane. In a field of movement for the MR, a random orthonormal inertial standard $\{0, \vec{I}_1, \vec{I}_2\}$ is specified. Both a random base $\{\vec{X}_1, \vec{X}_2\}$ linked to the structure and an arbitrary point of

reference R within the framework are specified. After that, the three parameters X, Y, and Θ fully define the MR position (P) in equation (1.1) [15]:

$$P = \begin{pmatrix} X \\ Y \\ \theta \end{pmatrix} \tag{1.1}$$

1.1.1.2.2.2. Legged Mobile Robot

A legged mobile robot is a kind of robot that can navigate as a living creature, including an insect, quadruped (a creature having four legs), or biped (a creature that has two legs, like humans). Usually, these robots utilize control mechanisms, actuators, and sensors to help them navigate through different types of situations. Robots with legs can move through a variety of environments that wheeled robots might consider difficult, such as slippery surfaces, stairs, or impediments [16]. Robots with legs are more flexible when it comes to changing the terrain than those with wheels or tracks. They can adapt their stride length and posture to make their way through difficult situations. However, there are additional drawbacks to legged robots as well, including the consumption of energy, mechanical layout issues, and the complex nature of controlling mechanisms. Fig. 1.6 shows a robotic dog 'Spot' developed by Boston Dynamics that can walk autonomously.

Fig. 1.6: Four-legged robotics dog [17]

1.1.1.2.2.3. Tracked Mobile Robot

A tracked MR is a specific type of robot that runs without legs or wheels by using constant tracks, similar to those on tanks. Compared to wheeled MRs, these robot tracks' grip and stability enable the robot to traverse a variety of terrains—such as mud, snow, sand, and harsh terrain—more successfully, particularly in natural surroundings [18]. When compared with wheeled MRs of identical shape, tracked MRs can usually carry larger payloads, which renders them appropriate for operations involving the transportation of materials or apparatus. Numerous sectors and applications, including mining, exploration, agriculture, disaster relief, defense, and environmental monitoring, use tracked MRs. Fig. 1.7 illustrates an image of tracked MR known as XBOT, which can be applicable in inspection activities for all-terrain and support the Robotic Operating System (ROS).

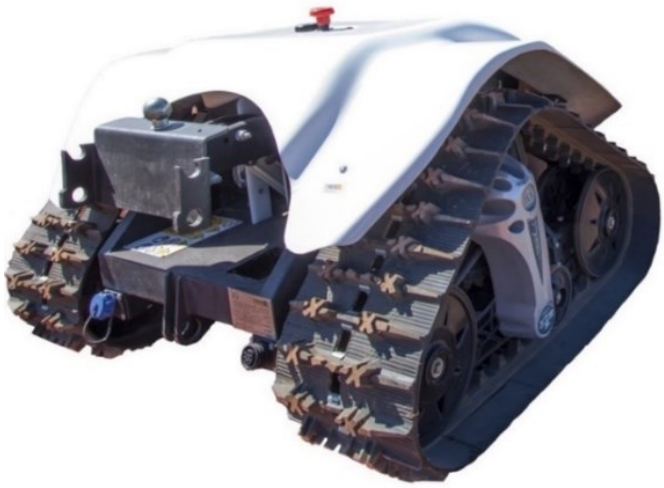

Fig. 1.7: Model of XBOT tracked mobile robot for all-terrain [19]

1.1.1.2.3. Air-based Mobile Robot

A flying robot, also known as an air-based mobile robot, is a kind of mechanical structure that uses air propulsion systems like propellers, rotors, or wings to move around and carry out activities in three dimensions. These autonomous machines come in a broad range of sizes,

shapes, and functions; they can be little drones or massive unmanned aerial vehicles (UAVs) utilized for freight shipment, monitoring, and research. Air-based mobile robots can pilot themselves independently or wirelessly thanks to technological breakthroughs in AI, computer vision, inertial sensors, GPS navigation, and computer vision [20]. This capability opens up a wide range of applications in many sectors including search and rescue missions, agricultural monitoring, environmental monitoring, infrastructure inspection, aerial photography, and videography. Air-based MRs are extremely useful instruments in situations where ground-based or piloted planes can be risky or unfeasible because of their adaptability and mobility. Research and development activities in this quickly developing industry are still motivated by obstacles including short battery life, legal limitations, and security and privacy issues. Fig. 1.8 illustrates a UAV, which is known as MQ-1C Gray Eagle. This UAV is manufactured by the General Atomics Aeronautical Systems for combat operations to support the United States Army.

Fig. 1.8: MQ-1C Gray Eagle is a UAV [21]

1.1.1.2.4. Water-based Mobile Robot

A specific kind of machine called a water-based mobile robot is made to function and travel in watery places, including rivers, lakes, and even small areas like containers and pipelines. These robots easily navigate over water by using a variety of propulsion techniques,

including fins, thrusters, and propellers. Miniature remotely operated vehicles (ROVs) for deep-sea exploration and observation and massive autonomous underwater vehicles, also known as AUVs, for scientific study, meteorological surveys, and underwater mapping are just a few examples of the many shapes and dimensions of water-based MRs. These robots collect information about the underwater environment, including depth, temperature, marine creatures, and water quality, using cameras, sensors, and other instruments [22]. These robots can accurately and autonomously traverse underwater settings thanks to sophisticated navigation systems, including inertial navigation and acoustic location. Numerous fields and businesses, including oceanography, maritime petroleum, and natural gas discovery, monitoring the environment, underwater archeology, search and rescue activities, and infrastructural surveillance, employ water-based MRs. These robots are essential instruments for exploring and comprehending the world's seas and rivers because of their capacity to reach underwater habitats in an effective and secure way. Fig. 1.9 represents an AUV which is known as Sentry. Sentry "flies" over the bottom to examine topographic characteristics at depths of 6,000 meters (3.7 miles) undersea, thanks to its integrated multibeam echo sounder, digital camera, and streamlined torpedo design.

Fig. 1.9: Image of an AUV 'Sentry' [23]

1.1.2. Autonomy of Mobile Robot

A completely autonomous robot is able to gather information about its surroundings, operate for lengthy periods of time without help from people, move any portion of itself across its operational area, and avoid hazardous situations for individuals or assets. Additionally, an autonomous robot is capable of learning new things, such as how to modify novel ways to carry out its job or react to ever-changing environments [24]. Therefore, it is necessary for MRs to possess autonomy and intelligence. Additionally, creating algorithms that enable the robots to operate independently in unorganized, dynamic, partly visible, and unpredictable settings presents a difficulty to scientists trying to address important issues including reliability, real-time action, and unpredictability (for both actions and sensing). Without navigational capabilities, mobility is nearly useless in any of these types of MR applications. While certain tasks like housekeeping or tracking can benefit from stochastic navigation, which does not require navigation, most research or commercial applications of mobile robots require the robots to be able to maneuver with a goal in mind. Therefore, autonomy in navigation is essential to the MR's performance and serves as a benchmark for the corresponding technology of autonomous MRs.

An MR navigation duty is to design and implement a course, taking into account the avoidance of obstacles, to a predetermined destination using sensor data. There are the following five connected competencies that are basically included in MR navigation:

1) Perception: To gather and analyze sensory data.

2) Mapping: To use the observed sensory data to build a virtual or environmental framework.

3) Localization: The method is used in tandem with navigation guidance to determine the MR's location inside the spatial mapping.

4) Path planning: The method to determine whether a route to a destination is ideal or not.

5) Exploration: The plan that directs the robot to choose its next course of action.

A robot needs a system that lets it navigate around freely in its surroundings; in other words, it has to be capable of recognizing and responding to its surroundings. Robotic sensors function as the equivalent of MR vision, allowing MR to understand its location, and exactly how it got there, and even make sense of its past movements. The sensors might be movable and adaptable, with the goal of monitoring the exterior environmental framework, inertial adjustments, and the distance that wheels traveled over the terrain. The sensors can be broadly categorized into two categories: exterior state sensors, including infrared, sonar, laser, and visual sensors, which offer outdoor knowledge of the surroundings, and inner state sensors, including accelerometers and gyroscopes, that give inside knowledge regarding the movement of MRs [25].

The robot's location in a two-dimensional (2D) space can be determined by analyzing information gathered from inbuilt state sensors. Exterior state data from sensors can be transformed into information for a surroundings map or utilized to immediately identify an object or circumstance [26]. Because of disruptions, readings from sensors are typically insufficient and unreliable. Thus, processing sensory information with disruptions is crucial for the navigation of MRs. Neural networks can offer some resilience or tolerance for errors for sensor analysis of data because of their large number of computing nodes, all of which largely have local links. In summary, the autonomous navigational challenge involves mobile robots being able to gather sufficient environmental information, analyze it, and take appropriate action to maneuver effectively within the environment, often following a predetermined course. An essential prerequisite for every automated framework is the capacity to perceive its surroundings. The robot bases all of its decision-making about actions on the sensory data it receives.

1.1.3. Mobile Robot Learning

It is not generally feasible to train autonomous robots to carry out predetermined tasks because unexpected circumstances could arise that the robot is not expected to meet. Currently, nevertheless, almost all industrial robots have been pre-programmed and need a precise, regulated environment. Such robot reprogramming is frequently an expensive procedure that calls for a specialist. Installing robots and reprogramming work becomes easier by allowing them to learn activities either by themselves or with assistance from a human trainer. Meanwhile, among the most intriguing features of intelligence is the absence of MRs that are incapable of learning. Recent studies have indicated a trend toward AI techniques to enhance robot autonomy through experience. These techniques can also be technically less costly than traditional ones. ML techniques are frequently used to lessen the workload for system engineers. As an outcome, learning has taken center stage in contemporary robotics research. The study of how robots learn lies at the nexus of robotics and ML. It investigates methods that let a robot learn new abilities or adjust to its surroundings using learning techniques. The robot's physicality, embedded in an external setting, offers both possibilities to direct the learning procedure and unique challenges (such as high complexity and actual time limits for gathering information and learning). Although there are many other applications of robot learning, including perceiving, planning, and decision-making, our study focuses on teaching control in a simulated environment.

Robot learning includes the application of a wide range of ML techniques, most notably learning by imitation, inverse RL, RL, and regression techniques, that are adequately domain-adapted to enable learning for complicated robotic systems, including humanoid robots, legged robots, aircraft, and machines with shaking wings. While traditional AI-based robotics methods have frequently tried to manually establish a variety of principles and algorithms that enable

robotic structures to comprehend and participate in real life, the foundation of robot learning lies in the belief that it is unattainable that humans will be able to accurately predict every engaging real-world scenario. All things considered, learning control is the method of learning an activity and control technique via trial and error [27]. Two well-liked groups of techniques for learning policies for sequenced choice tasks are RL and learning from demonstration (LFD) [28]. RL methods are used to tackle Markov decision processes (MDPs), which are sequencing choice challenges. The agent learns a policy by experimenting with various behaviors in various scenarios and trying to optimize a minimal reward signal. RL has been used in many different situations with efficacy.

A technique for robot/agent learning known as "learning from demonstration (LFD)" builds activity or assignment frameworks by using human presentations as input. The field of LFD research encompasses a wide variety of methodologies [29]. The LfD method learns strategy mappings between states (input) and actions (output) according to the scenarios observed in the presentations. These demonstrations tend to be expressed as state-action pairs. Another strategy eliminates the requirement for a model or in-depth knowledge of the field by offering a mapping between sensory inputs and behaviors that quantitatively represent the important goals of behavior [30]. These approaches work effectively in fields where there are few resources available to draw lessons from past experiences and adjust to changing circumstances.

1.1.4. Human-Robot Interaction

A combination of AI in almost every system has contributed to a significant expansion of human capabilities in the past decade, including comprehension, awareness, learning, and activity. Artificial Intelligence's promising prospects are primarily attributed to human interaction with AI. Furthermore, an automated system or other equipment that is completely

automatically or manually operated has to collaborate with a human throughout a number of automating and support phases. Humans and robots can collaborate or communicate in a variety of ways, which is known as HRI. A unique form of human behavior prediction is human activity recognition (HAR), which represents a crucial method for HRI and MR movement that avoids human collisions [31]. Anticipating human patterns is given less consideration in certain contexts where supporting robots may become highly influential, such as offices, homes, hospitals, and assisted living facilities [32]. When comparing with urban settings, the primary variable agent is staying in designated lanes or following traffic laws. Additionally, the actual space is frequently shorter due to limited access points and an increased quantity of perceptive obstacles (such as inner walls or blind curves), that initially give the impression that the surroundings are nearer. Consequently, there will probably be a greater likelihood of a collision. It will become crucial to predict human mobility in dynamic contexts, such as homes and companies, for safe and dependable robot navigation. Thus, every assistive mobile robot must be built with HAR technologies. Predicting human trajectories is a challenging task; prior work has included various mobility situations, scene surroundings, and social interactions. HAR technology has been analyzed in [33].

1.2. Research Motivation

Modern autonomous vehicles rely heavily on their capacity for autonomous navigation. In 2003, the Defense Advanced Research Projects Agency (DARPA) of the United States government intended to accelerate the advancement of the techniques necessary to produce the first fully self-driving land vehicles that might complete a significant off-road track in a shorter span of duration. This initiative was known as the "Grand Challenge." Robotic cars were to complete the challenge in a maximum of 10 hours, covering a 142-mile distance over the Mojave desert. The inaugural tournament took place on March 13, 2004. However, no one of

the fifteen cars that are taking part has ever finished over five percent of the route. As a result, on October 8, 2005, the next DARPA Grand Challenge was launched. Ultimately, five of the twenty-three cars completed the race successfully [34]. An important step toward creating today's self-driving automobiles has been completed by this robotic vehicle. Two years later, on November 3, 2007, autonomous cars were required to complete the "DARPA Urban Challenge," which involved traveling 97 kilometers over a simulated urban setting in less than six hours while engaging with oncoming traffic, navigating obstacles, and adhering to each traffic law. "Boss" was pronounced the victorious car, while "Junior" secured the second position [35], [36]. These vehicles were also thought to be the first iteration of the Google autonomous vehicle prototype.

A real-life programmer would utilize their knowledge of the intended target in a standard programming instance, and they would have needed to make choices ahead for a while to design an MR controller that could react to any condition the MR would encounter, regardless of how improbable. Although this type of customized programming is quite effective, it is also costly and only works in the scenarios that the human operator has thought of. The entire expensive procedure could have to be duplicated if mistakes or unanticipated events occur once the robotic device is deployed. Although the aforementioned DARPA challenges were completed in unprepared programs, it is difficult to believe that any work that might be assigned could be preprogrammed. As a result, robots must possess the ability to acquire information either on their own or with assistance.

This dissertation, which is inspired by Google self-driving vehicles and DARPA challenges, focuses on enabling MRs to make thoughtful, logical judgments as well as being able to learn brand-new skills and enhance existing ones intelligently. Using this method, robots might be trained to deal with uncertain and unpredictable conditions by learning how to deal as best they can with uncertainties and unanticipated shifts.

The use of MRs is expanding across a number of industries, including production, logistics, the agricultural sector, and search and rescue. These robots require being highly autonomous and adaptive, with the goal of functioning well in dynamic and unpredictable surroundings. MR navigation in real-world contexts might require extremely complicated control challenges with dynamic obstacles, shifting topography, and erratic disruptions [37]. Without the requirement for specific programming or human involvement, RL is a viable method for giving robots the capacity to learn from historical experiences and modify their actions in response to environmental input.

The requirement for MRs to connect with humans in a way that is safe, effective, and simple is expanding as these robotic devices are incorporated more and more into human scenarios [38]. MRs might be taught socially conscious behaviors using RL-based control mechanisms. These behaviors include adhering to social standards, deciphering human gestures, and changing their actions in response to input from people. Furthermore, RL can help multiple robots working in groups along with people collaborate, resulting in more efficient and well-coordinated robotic systems. To sum up, the goal of studying RL-based intelligent control algorithms for MRs is to improve the adaptability, autonomy, effectiveness, and flexibility of robotic equipment in everyday situations. This is going to contribute to the development of robot technology and its possible uses in a variety of fields.

1.3. Thesis Contribution

The role of intelligent control in self-navigating systems through robot learning using sensory information is examined in this doctoral thesis. It is intended that people lacking programming skills would be able to more readily expand and customize robotic skills for new scenarios. We started our investigation by looking into three important learning algorithms: RL, learning without demonstrations, and NNQL. MRs can acquire complex control methods that

maximize goals like navigation performance, usage of energy, or job timeliness by utilizing techniques like DRL. The following is an overview of this thesis's significant contributions:

1. The drawback of conventional RL that has drawn the interest of several academics is its inability to generalize states that have not been explored yet. NNs are strong supervised learning algorithms that can help with this issue because of their strong adaptation capabilities. NNs were subsequently added to RL, which significantly enhanced the capacity for learning.
2. The term "imitation" immediately comes to mind as we consider how people pick up new abilities or knowledge when they are still very young. Robots ought to have the fundamental ability to mimic. This thesis led to the development of an effective policy learning technique that enables MRs to learn and adjust to their dynamic surroundings without demonstrations.
3. Autonomous learning is performed satisfactorily whenever specialists are not given ideal illustrations. However, non-optimal situations are provided almost all the time. Learning through demonstrations in these situations might result in inadequate performance. Learning the benefits in proven situations and then generalizing over all un-demonstrated contexts is a superior approach. We created a way by incorporating neural networks using RL, also known as training instruction, and we were able to produce a quick and reliable learning system.

These three suggested techniques have been used in research involving MR navigation. Based on the outcomes, we can confidently say that our techniques were successful in giving MRs the ability to learn on their own through experimentation and human demonstrations. Ultimately, by creating new algorithms, approaches, or strategies that increase the level of adaptability and autonomy of MRs, this thesis will enhance the latest developments in intelligent control mechanisms.

1.4. The Architecture of Thesis

This dissertation is further divided into six chapters, which are described below:

1. Chapter 2 begins by formalizing the theoretical aspects of the RL technique in the control system of MR. Further, we described AI, ML, DL, ANN, and DNN in detail. Lastly, we described the historical development of RL, the major applications of RL, and the role of RL in MR navigation approaches.
2. Chapter 3 described various pieces of research approaches performed in available literature related to the intelligent control system for MR.
3. Chapter 4 provides the partial observable Markov decision process (POMDP) and the Markov decision process (MDP). we further go over a few common dynamic programming, or MDP, solution algorithms.
4. Chapter 5 introduced autonomous MR's capacity for self-learning despite the absence of professional demonstrations. We provide a way to extrapolate and optimize the method of self-learning using an ANN within the context of RL. We extensively evaluate our approach in static as well as dynamic scenarios, using navigational autonomy challenges through simulation. The simulation presented in this approach illustrates an effective intelligent control technique for the navigation of MR in an environment containing different obstacles. The simulation presented in this approach illustrates an effective intelligent control technique for the navigation of MR in an environment containing different obstacles. The result and discussion provided in the dissertation give deeper insights into autonomous navigation in unknown environments using RL.
5. Chapter 6 provides a comparative discussion of different navigation strategies, including RL, PSO, and Thermal navigation.
6. Chapter 7 provides a conclusion and further research prospects in this field of study.

Chapter 2
Reinforcement Learning

2.1. Artificial Intelligence

Artificial intelligence (AI) is a technological power that is transforming economic growth, upending businesses, and drastically changing how humans interact with computers and technology. Fundamentally, AI aims to endow computers with human-like intelligence, allowing computers to sense their surroundings, reason, acquire knowledge, and reach conclusions on their own. AI's history began in the middle of the 20th century when scientist Alan Turing established the discipline's foundation [39]. AI has developed over the years from conceptual concepts to practical implementations, thanks to advances in algorithmic creativity, data accessibility, and processing capacity.

Autonomous vehicles, healthcare diagnosis, banking, and personal assistance are just a few of the fields where AI has advanced significantly in recent years. Researchers and practitioners are facing issues including interpretability, rights, and responsibility as AI develops. To guarantee that AI technologies maximize advantages for humanity as a whole and minimize risks, attempts to solve these issues are imperative. The social ramifications of AI, including how it might influence jobs, security, morality, and discrimination, have sparked both enthusiasm and worries in light of these breakthroughs [40]. Artificial neural networks (ANNs), machine learning (ML), deep learning (DL), deep neural networks (DNNs), natural language processing, image processing, robotics, and other subfields are among the many subfields that fall under the broad category of AI. Fig. 2.1 illustrates the relationship between AI, ML, and DL, which shows ML is the subset of AI and DL is the subset of ML.

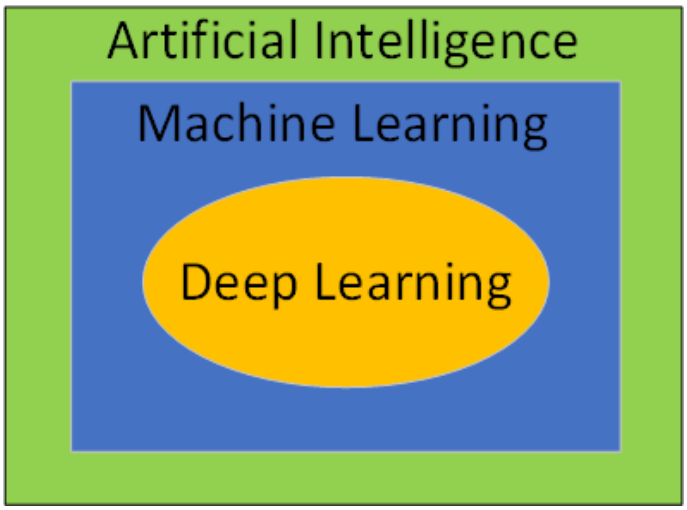

Fig. 2.1: Illustration of the relationship between AI, ML, and DL

2.1.1. Machine Learning

A branch of AI called machine learning (ML) focuses on creating algorithms and models of statistics that allow computers to carry out operations despite the need for explicit programming [41]. In contrast to conventional programming, which determines a computer's behavior by explicit instructions, ML systems learn from data by seeing trends and drawing inferences or projections from that information. Fundamentally, ML is about giving computers the ability to grow and learn from their experiences. Large volumes of information are needed for ML algorithms to acquire knowledge. A variety of information, including text, photos, statistics, and sensor assessments, might be included in this data.

ML algorithms are computational models that examine data in an attempt to find trends, correlations, or patterns [42]. These algorithms are split into several categories, depending on whether they are appropriate for a particular set of activities and data: unsupervised learning, semi-supervised learning, supervised learning, RL, and more. ML has applications in many different fields, including recommendation systems, computer vision, natural language processing, autonomous vehicles, healthcare diagnosis, and financial predictions. It is an

effective tool for handling a variety of complicated, unorganized collections of data and is versatile enough to solve many different kinds of real-world situations. Fig. 2.2 illustrates the relationship between ML and DL, which shows feature extraction and classification steps are considered two different steps in ML and a single step for DL.

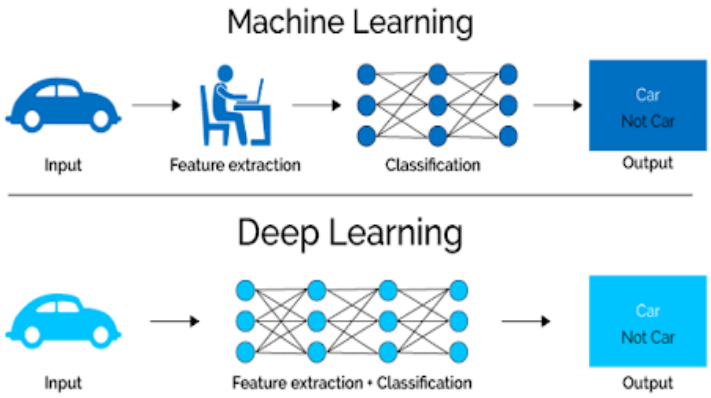

Fig. 2.2: Illustration of the relationship between ML and DL [43]

2.1.2. Deep Learning

DL is an ML approach that teaches computers to do tasks that come effortlessly to people: comprehend via illustration. In deep learning, a model created by computers is able to do tasks involving explicit classifications from text, audio, or pictures. Models created using DL have the capacity to attain greater accuracy, occasionally surpassing human accuracy. The secret to autonomous vehicles is DL technology, which enables them to accomplish tasks including stopping at a stoplight despite the need for humans.

DL constitutes an artificial neural network (ANN)-based ML hypothesis. The DL approach may be used in machine vision, information mining, supercomputers, detecting fraud, language processing, systems for managing customer relations, driverless cars, and HAR [44]. For sim-to-real transfer, which involves training robots in virtual settings before deploying them in real life, DL techniques are applied. Robots can implement their acquired rules in diverse contexts by using domain adaptation strategies, including transfer learning, competitive

training, and field variation, that help close the physical distinction between modeling and the actual world. DL can be applied for route planning, localization, and mapping challenges related to autonomous navigation.

2.1.3. Artificial Neural Networks

The use of an artificial neural network (ANN) can be found in data mining and ML. These kinds of networks constitute a class of biological neural network models that represent quantifiable brain learning. The neural network operating concept is similar to how the human brain's neural structure works. An ANN's structure resembles a network of linked neurons that communicate with each other. The most significant components of AI technologies, which are mostly found in control systems, are ANNs. ANNs are composed of uniform sequences of neurons, or units, linked through unique synapses, or weights [45]. Fig. 2.3 illustrates the basic structure of ANN, which contains input, hidden, and output layers, respectively.

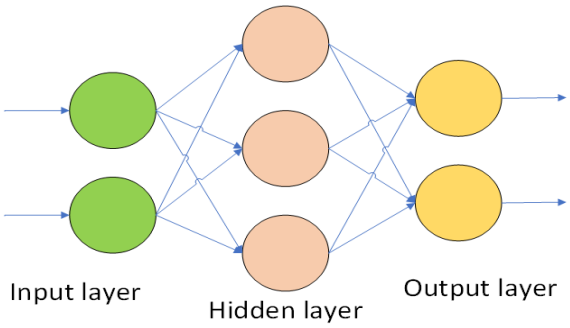

Fig. 2.3: Illustration of the basic structure of ANN

2.1.4. Deep Neural Networks

A set of ML models known as "deep neural networks" (DNNs) is motivated by the composition and operations of the brains of humans. DNN is a subset of ANNs, a larger class of computer models made up of linked nodes, or "neurons," arranged in layers [46]. Artificial neurons are the fundamental components that construct neural networks. After processing

signals from input through various computations, every neuron generates an output signal. Neural connections containing correlated weights link neurons in neighboring layers. Such weights, which are acquired throughout training, establish the link's durability.

Usually, an activation function is applied to every neuron based on the weighted total of its data [47]. By doing this, the network gains non-linearity, which enables it to recognize intricate patterns in the input. DNNs are trained by a technique known as backpropagation. This method involves comparing the predictions made by the network with the real targets and propagating the error back via the network. Next, using optimization procedures such as gradient descent, the weights are modified to reduce the error. DNNs play a major role in DL, a branch of ML that concentrates on learning data interpretations. Natural language processing, audio recognition, image recognition, and other applications have all seen impressive results with DNN. Fig. 2.4 shows the basic structure of DNN. Generally, DNN contains multiple hidden layers, which means it must have more than one hidden layer.

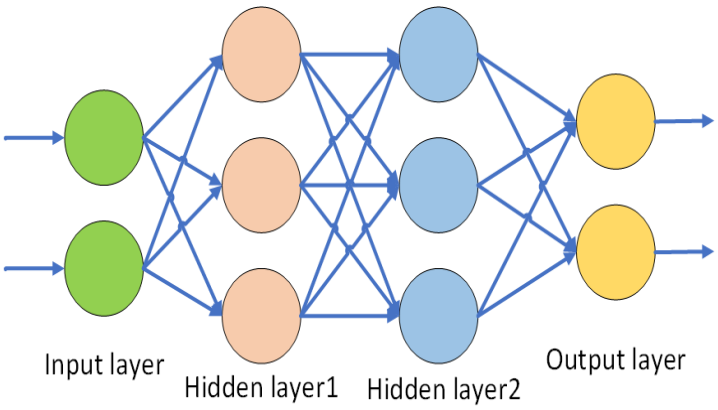

Fig. 2.4: Illustration of the basic structure of DNN

2.2. History of Reinforcement Learning

From fundamental concepts in psychology and control systems to real-world applications in AI and robotics, the long history of RL illustrates a path that continues to shape the

discipline's potential perspectives. RL is a branch of ML that studies how agents, or decision-makers, may pick up decision-making skills by interacting with their surroundings. There are two significant, rich, and lengthy strands for the origins of RL that were studied separately before coming together to form the current RL. One theme that emerged from the psychology of animal learning is trial-and-error learning. This line of reasoning permeates a portion of the first AI research and was responsible for the very first 1980s rise of RL [48]. The other part focuses on the optimum control issue and how function values and dynamic programming are used to solve it. This topic did not require learning in the majority of cases. The expectations center on a third, less distinguishable path that deals with temporal-difference techniques, like those employed in the tic-tac-toe instance in this section, even if the first two components have been mainly independent. The current area of RL was created in the second half of the 1980s when all three elements came together.

Temporal-difference learning's history in the context of RL is a major breakthrough. Unlike other learning techniques, temporal-difference learning is motivated by the distinction among spatially subsequent estimations of the same variable, such as the chance of victory during the tic-tac-toe case [49]. The concept of auxiliary reinforcements, which is central to learning in animal psychology, is where temporal-difference learning got its start. When an incentive is combined with an initial reinforcer, like a meal or discomfort, it becomes a second reinforcer and acquires comparable reinforcing qualities.

The current era of RL is characterized by a progression from conceptual foundations to real-world uses across a variety of fields, propelled by advancements in processing power, multidisciplinary teamwork, and algorithms. The comprehension and regulation of issues employing Markov Decision Processes (MDPs) witnessed significant advances in RL [50]. Deep Reinforcement Learning (DRL) emerged as a result of the confluence of DL and RL, garnering significant attention and propelling advancements across several fields. Research is

still being conducted to expand the capabilities of RL and tackle issues pertaining to its wider use in sophisticated real-world situations.

2.3. Major Applications of Reinforcement Learning

RL is a category of ML that focuses on control and decision-making problems. As a way to accomplish certain goals, an agent needs to communicate with its surroundings in order to learn how to make decisions [51]. Because RL can learn optimal activities via trial and error, it has gained widespread applications in many different disciplines. Fig. 2.5 describes that RL has a wide range of applications in different fields, including robotics, physics, automobiles, control systems, business, banking, game playing, and the health sector. These applications highlight RL's adaptability and strength within a range of sectors, rendering it a useful instrument for resolving difficult control and decision-making issues in real-world environments.

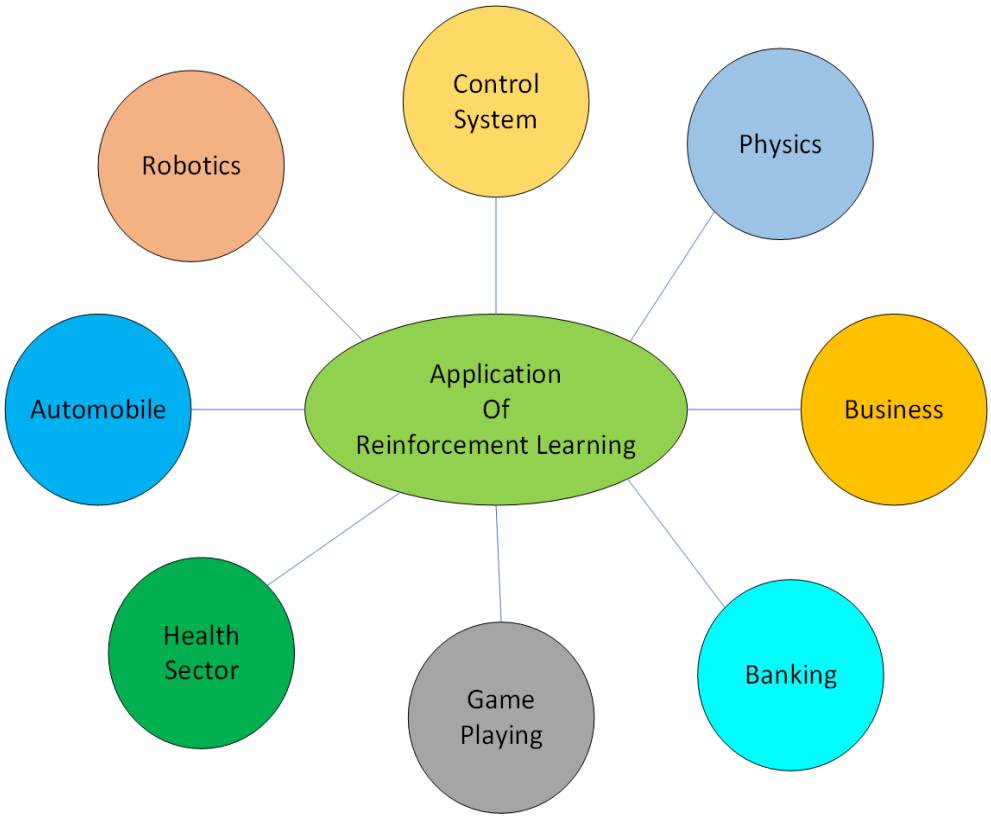

Fig. 2.5: Illustration of major applications of RL

2.4. Role of Reinforcement Learning in Navigation

Machine learning might be classified as reinforced, unsupervised, or supervised based on the type and quantity of data received on the assignment or process. Building an operational architecture that represents the acquired relationship between inputs, outcomes, and system variables represents the goal of supervised learning when algorithms for learning get feedback in the shape of a labeled data set. The goal of unsupervised learning is to categorize data sets into distinct categories according to the degree of familiarity among the input samples without giving the technique any feedback. Lastly, through interaction with the surroundings, an agent or decision maker can develop a policy to maximize a long-term reward through the use of RL, a focused target learning technology.

On-policy and off-policy learning control techniques are the two groups of algorithms for RL that are implemented for resolving optimal control issues [52]. On-policy techniques assess or enhance a similar policy that serves as the basis for decisions. In off-policy approaches, those two functions are divided. The behavior policy, which represents the policy that is implemented to produce data, cannot be linked to the estimate policy, also known as the objective policy, which is the strategy that is assessed and refined. The data utilized in this phase might be generated offline through implementing the behavioral policy to the variations within the framework, but the objective strategy learning process is conducted online. Since a collection of experience gained from implementing a behavior policy can be used to modify multiple value functions belonging to other estimating policies, the off-policy approaches are quick and data-efficient. Additionally, off-policy techniques consider the impact of probe noise required for exploration.

RL is a computer method for comprehending and automating decision-making and goal-directed learning. Its focus on an agent learning by interacting directly with its surroundings—

without the need for perfect models of the world or example supervision—sets it apart from other methods of computation. As the first area to take these computational challenges carefully, RL aims to attain long-term objectives by learning from contact with its surroundings. A scholastic structure that describes the state, action, and reward of the learning agent's relationship with its surroundings is used in RL. The goal of this structure is to provide a straightforward representation of the key elements of the AI challenge. Explicit aims, a feeling of ambiguity and unpredictability, and an awareness of cause and consequence are some of these characteristics [53]. RL is essential to MR navigation because it allows robots to interact with their surroundings and acquire nearly optimal or optimal navigational strategies. RL needs a well-specified state space that encompasses pertinent details regarding the robot's surroundings. Sensor inputs, robot locations, obstacle locations, and target locations are often included in navigation tasks.

The robot's potential actions, including forward motion, left and right turns, stopping, and so on, make up the action vector in RL. The actions used are determined by the particular navigational challenge and abilities of the robot. RL navigation requires the development of suitable reward functions. The robot receives information about its behaviors via the reward function, which helps it behave in the desired ways. Effective and secure navigation is encouraged with a well-thought-out rewards system.

Navigation policy learning might be accomplished through the use of RL techniques, including actor-critical methods, Q-learning, policy gradient methods, and deep Q networks (DQN) [54]. These techniques use data from rewards and changes in state during explorations to modify the robot's policies. Robots can adjust to diverse obstacles, variations in the environment, or human needs without having to be completely trained through the continuous learning techniques used in RL frameworks. In situations where many robots are moving in a single environment, multi-agent RL approaches facilitate collaboration, the avoidance of

collisions, and the effective usage of resources. Through the utilization of such RL features, MRs can acquire the ability to independently traverse sophisticated surroundings, adjust to varying situations, and accomplish navigation objectives effectively while maintaining security and stability.

When it comes to MR navigation, RL is a robust ML technique that has earned a lot of curiosity. Robots can adjust and enhance their navigational behaviors using RL, in contrast to conventional rule-based or programmed techniques, by utilizing input from their surroundings. Determining the state space, developing suitable action spaces, generating reward functions to encourage desired actions, and using learning algorithms for modifying the robot's navigational strategy in response to observations are important elements of RL in MR navigation [55]. MRs can traverse dynamic and complicated surroundings, adjust to shifting situations, avoid obstacles, accomplish goals, and improve navigation courses by utilizing RL approaches. These are critical characteristics for autonomous robots in a range of practical uses. To be able to provide effective and secure navigation, the robot will ultimately develop a policy using RL—a mapping between states and actions—that optimizes accumulated rewards over time.

Chapter 3
Literature Survey

In recent years, many researchers have been working in the field of autonomous navigation to enhance the capabilities of MR, and many of them are using RL as a primary technique to do this work. Alitappeh *et al.* [56] describe a navigational robotic system that is intended to perform the dual duties of avoiding obstacles and line following in partially known surroundings that contain obstacles. This system performs robustly by using distance sensors led by a CNN model to avoid obstacles and a strategically located camera for an accurate line following a model of long short-term memory (LSTM). The results of the experiments show a significant improvement in performance, proving that the suggested method is effective in producing better results in difficult robotic navigation situations. Tsuruta *et al*. [57] emphasize MRs' autonomous navigation, utilizing DRL for interpreting photographs from monocular cameras. Since semantic division can minimize the disparities between surroundings by breaking down complicated RGB pictures into segmented images, it is used to address real-world recorded image analysis. In this work, a semantic segmentation system with an RL model is obtained for MR navigation. These models are implemented in an intelligent navigation system built on a robot operating system (ROS). Experiments conducted in real-life contexts validate the theoretical models' effective applicability.

Saxena *et al.* [58] provide a time-dependent policy that will enable robust signal temporal logic (STL) description fulfillment in continuous state space using a feasible RL method, using the idea for funnel functions within this article. This study uses several contexts to illustrate the usefulness of the suggested technique on many STL tasks. Jiang *et al.* [59] propose a method to overcome low navigation efficiency by utilizing a unique type of graph convolutional network

(GCN) called message-passing GCN (MP-GCN) to encode either human-human interaction or HRI. The suggested approach, called MP-GatedGCN-RL, represents asymmetrical interactions by combining a unique message-passing function with edgewise gating processes in comparison with current approaches that represent interaction among humans and robots equally. This suggested method is evaluated using ETH/UCY pedestrian datasets that mimic various scenarios, such as avoiding collisions, forming groups, crossing, diverging, and more. Kumaar *et al.* [60] developed a route plan in a dynamic environment using RL. The recommended approach makes use of an interactive topographical mapping of the surroundings to acquire the very first pathways based on the deep Q-learning method. Depending on how similar the prior and novel settings are, this technique switches between experience-based training and exploration-based training. The Turtlebot3 MR within the Gazebo simulator is used to evaluate the system, which is built using the ROS platform. Based on the original topographical map of various service contexts, the results of the experiment demonstrate that the RL method of learning knows all paths with an accuracy rate of more than 98%.

Tongloy *et al.* [61] describe an asynchronous DRL approach that has been modified for MR navigation using supervised auxiliary duties. In this work, the TensorFlow-based hybrid Asynchronous Advantage Actor-Critic (A3C) method is implemented on the CPU or GPU. Robot location assessment and depth estimation are examples of supervised auxiliary duties. The simulated MR demonstrates the capacity to identify locations on a map and learn to travel using just input from unprocessed RGB images. Williams *et al.* [62] provide a model predictive control (MPC) method based on information theory that can handle generic nonlinear behavior and complicated cost constraints. Multi-layer neural networks are frequently used as dynamical models because of the strategy's universality, and this MPC technique incorporates them to tackle model-based RL challenges. This technique is evaluated both on real hardware in intense driving difficulties and in simulation using a cart-pole swinging up and quadrotor navigating

task. Experimental findings show that the algorithm can operate at the highest level in this study using the information gathered from the system. Liu *et al.* [63] show the challenge of figuring out how to combine and transfer robot expertise so that they are easily adapted to novel settings and make effective utilization of existing expertise. This work presents Lifelong Federated RL (LFRL), a learning framework for navigational instruction for cloud systems of robotics, as a solution to the issue. An information-fusing approach is presented in this research to upgrade a cloud-deployed shared model. Subsequently, efficient techniques for transfer learning in LFRL are presented. Tests demonstrate that LFRL significantly boosts RL's effectiveness for robot navigation.

Cang Ye *et al.* [64] suggest a neural-fuzzy system that combines stages of both fine and coarse learning. During the initial stage, the member functions of both inputs and outputs are concurrently found using a supervised learning technique. Once there has been enough training, the RL algorithm is used for finer learning, which adjusts the membership function in the output variables. A novel approach to learning that strengthens exploration and makes use of a variation of Sutton and Barto's framework. This two-step tuning method allows the MR to navigate without collisions. Thananjeyan *et al.* [65] offer a method for safe RL that incorporates learned recovery zones. One major barrier keeping RL from being widely used in real life is safety. Learning new activities in unknown situations necessitates significant exploration, while safety dictates restricting exploration. The Recovery RL algorithm manages this trade-off by dividing the objectives of increasing task effectiveness and constraint fulfillment into two regulations: an activity regulation that just enhances the assignment reward and a recovery regulation that directs an agent towards security when constraints are probable. Recovery RL uses offline information to acquire knowledge about constraint-breaching regions prior to regulation learning. Based on the results, it appears that Recovery RL trades off assignment

accomplishments and violating constraints 2–20 times more effectively in simulation environments and three times more effectively in real-world tests.

Jiang *et al.* [66] present a 3D visual navigation system using neuromorphic RL targeting micro-aerial vehicles (MAVs) equipped with depth cameras. Using a previous map as a guide, conventional visual navigation techniques for MAVs often compute a feasible route that meets the criteria. Although these approaches suffer from a number of problems, including an excessive reliance on computer power and a lack of resilience in novel situations. This study presents a neuromorphic RL technique (Neuro-Planner) that integrates DRL and spiking neural networks (SNN) to enable MAV 3D visual navigation using a depth camera in an effort to address these issues. According to the paper, this is a pioneering effort to combine DRL with neuromorphic computation to solve the MAV 3D visual navigation challenge. Wu *et al.* [67] provide an information-theoretic normalization factor in the RL to improve target-driven visual exploration using the DRL's cross-target and cross-scene adaptations. The regularization enhances the collaboration shared by an agent's perception of visual transformations and navigation behaviors, leading to better navigation decision-making. Using this manner, the agent learns a variational generative framework to represent the interaction between action and observation. The agent uses its present perception and navigational objective to create (visualize) the future perception determined by the model. Through the process of analyzing the present and anticipated future observations, the agent gains the ability to comprehend the link of causality among navigation acts and variations in its assessments, allowing it to anticipate the subsequent navigational action.

Hu *et al.* [68] suggest a unique collaborative exploration approach with multiple MRs that, when compared to traditional techniques, lowers the total mission execution time and energy expenses. With the goal of effectively steering the networked robots throughout the cooperative activities, a low-level goal monitoring level and a higher-level decision-making

level are combined to create a hierarchical control framework. By allocating distinct goal sites to each robot, the stochastic Voronoi partitions used in the development of the collaborative research strategy limit the number of exploration regions that are repeated. A combined collision-avoiding method based on DRL is subsequently suggested to cope with unexpected obstacles in unexplored surroundings. This approach allows the control strategy to acquire knowledge from human presentation information, thereby enhancing its efficiency and learning speed. Zhang *et al.* [69] provide a policy gradient potential (PGP) method that learns the best collaborative approach with the most global reward by using PGP as a data resource for the approach updates instead of the gradient themselves. Although the matrix of payoffs and the integrated approach are frequently unattainable for learning agents in real life, the efficiency metric in this study is the likelihood of receiving the optimal reward. The crucial locations related to every perfect coordinated action are asymptotically steady when every element activity of each ideal collaborative action is special, according to a theoretical study using the PGP algorithm for a continuous framework containing a similar desired recurring event. Multiagent reinforcement learning (MARL) has made substantial use of the gradient-based technique. Based on the accumulated reward for the quantity of time spent during a single episode, the PGP algorithm works better than the other algorithms, according to the data.

Wang *et al.* [70] offer a unique hybrid obstacle avoidance controller technique to provide an adaptively secure movement trajectory for autonomous systems. Initially, a barrier function is incorporated inside the cost function, allowing the barrier-avoidance issue to be effectively represented through an infinite-horizon maximum control issue. This allows system security for clarification utilizing forward invariance. Subsequently, a combination of state-following-based estimation and model-based policy repetition is offered as a robust RL paradigm. This learning architecture is accomplished using the actor-critic framework, which is based on current time information and extended experienced information. In this study, actor networks

generate adaptable control strategies through gradient presentation, while critic networks are modified using gradient-descent adaptation. The Lyapunov approach is then used to conceptually investigate weight convergence and system stability. Lastly, a nonlinear unicycle dynamical structure and a 2D singular integration scheme are used to show the suggested learning-based controller. Yin *et al.* [71] suggest a distributed multi-robot route selection technique for DRL using multi-critic twin delayed deep deterministic policy gradient (AMC-TD3). Using an asynchronous training process and a multi-critic system, this technique improves upon the primary GRU-Attention-Driven TD3 approach. This technique is very flexible for different contexts since it can acquire an end-to-end navigational strategy without depending on precise maps or any localized data. Simulation findings show that this suggested method outperforms standards in various contexts with varying robot populations and degrees of complexity.

Nan *et al.* [72] suggest using an integration of a depth estimation model and a monocular camera in place of a cheaper 2D LiDAR and provide a modified version of the Common Encoder Self-Attention Soft Actor Critic (SESA-SAC) method enabling MRs to navigate indoor without collision. To enhance robot learning performance in crowded scenarios, this approach gathers 200 episodes' worth of expert information and stores it inside a loop buffer. Without any prior training, this work uses a random selection from both expert and exploration data to perform training. This research introduces a channel-wise self-attention framework and layers of standardization in the network to acquire greater characteristics and improve the efficiency of training. Antonelo *et al.* [73] present an extensive framework for reservoir computing (RC) learning that might be utilized to instruct mobile robots to navigate in both easy and complicated, partly apparent unknown situations. By allowing the recurring portion of the network—known as the reservoir—to remain constant and just train the linear reading output layer, RC offers an effective method for training recurring neural networks. The idea of

a navigational attractor, or activity that might be learned and then implanted in the reservoir's highly dimensional environment, is the foundation for the RC architecture. The stochastic robot actions, which consist of a sensory-motor pattern, might be linearly differentiated through the highly dimensional nonlinear domain of the dynamic's reservoir, allowing for the learning of numerous actions.

Xue *et al.* [74] present a DRL technique based on the double Q-learning network (DQN) to help MRs acquire autonomous navigation and collision avoidance skills. Goal location, obstacle dimension, and location are examples of inputs, while the robot's trajectory path is an example of an output. For global navigation, classical MRs often need real-time, rapid, precise simultaneous localization and mapping (SLAM) systems. This research aims to address the possibility that, once the first globally viable route has been determined, the route might be divided into finite parts with sub-goals. The suggested approach relies on applying DRL to guide the robots as they sequentially reach the subgoals. Studies demonstrate that the suggested approach can guide the MRs to the intended destination without hitting any obstacles or other MRs, and the strategy has been effectively implemented on a real robot platform. Li *et al.* [75] introduce SARL*, a more sophisticated iteration of the socially attentive reinforcement learning (SARL) technique, to enable human-aware indoor navigation. DRL has made significant progress in producing human-aware navigation strategies recently. Despite this, there are several drawbacks to the practical applications: the environment's simplicity ignores obstacles besides people, and the learned navigational strategies are restricted to specific training-related distances. To attempt to address the issues, this study enhances the existing SARL technique by adding a dynamic regional setting targets system and a map-based secure activity region.

Pambudi *et al.* [76] suggest a fuzzy approach to state minimization in RL for MRs that avoids obstacles. The challenge in solving and avoiding obstacles using RL is determining the optimal number of states. RL becomes challenging to implement when situations arise that the

MR might not have imagined since there are an infinite number of states. It is necessary to apply fuzzy to lower processor performance, generalize the situation, and remove the amount of state trouble. Simulation results indicate that the highest level of performance is produced by the reinforcement spot that uses a change of states and five angle zones of the sensor. Liu *et al.* [77] created a 3D simulation setting to allow DRL-based robot navigation in a crowded pedestrian area. This work incorporates Gazebo, the Social Force Pedestrian Simulator, stable baselines, and the ROS navigation stack into its simulation environment. This simulation environment relies on the Gazebo modeling framework to gather the extensive environmental data surrounding the robot. This paper describes the ROS navigating framework to enable the usage of conventional path-planning techniques. This study provides Steady baselines, a set of enhanced adaptations of RL algorithms that utilize Open AI Standards, to facilitate the calling of the existing popular RL algorithms.

Cui *et al.* [78] propose an efficient map representation that offers adequate structural data within a suitable receptive space, and the path-extended graph is proposed. It can subsequently be included in a hierarchy-based policy for increased adaptability and performance. The path-extended graph avoids the burden of redundant data by including the compact shape of the threshold design and environmental architecture enabling a large-scale perception. Using a lower-level motion controller which manages the planning of paths and avoiding collisions and a higher-level endpoint decision policy that leverages DRL, this hierarchical policy resolves long-range decision-making. Experiments conducted in real-life situations and simulators show that this strategy works better than competing strategies in minimizing duplicate movement and achieving efficient goal-attaining, particularly in complicated contexts. Huang *et al.* [79] emphasize techniques of planning paths using sampling with inadequate supervision. This work uses DL to execute variable sampling on sample-based approaches with an emphasis on locations where optimum pathways are more probable to exist. The goal is to improve both path

reliability and computing efficiency. In particular, the challenge of semantic division is considered to be the creation of diverse sample zones. Here, diverse sample zones are predicted using a variety of map information. This paper provides an attention-guided approach for non-uniform sample planning of paths, which draws inspiration from DL attention processes. Online modeling estimation, offline dataset creation, and training of models make up the learning-driven planning of paths procedure. On the other hand, creating offline datasets frequently requires a lot of effort and resources. This paper suggests a weakly supervised approach to overcome this difficulty, which calls for the creation of just one truth route for every situation in semantic division training.

Sun *et al.* [80] utilize multi-head localized awareness modules to enhance the traditional actor-critic system and gather local data at an entity level. In this situation, avoiding collision mechanisms might concentrate on important environmental elements to function more effectively and react to environmental variations faster. Ultimately, the proposed framework is expanded to include policy delay (PD) and policy entropy (PE) with the objective to improve policy analysis and strengthen policy. Comprehensive test outcomes demonstrate that this technique may provide collision-free, quicker guide pathways to avoid collisions in extremely dynamic situations. Xu *et al.* [81] provide a general Multi-Reward Architecture (MRA) strategy that might be used to enhance state-of-the-art MRA techniques because automating the time-dependent priority task for branches is therefore essential. First, in order to enable MRA to acquire knowledge from every sub-reward section while retaining the expertise offered by the primary reward, this research introduced a policy section that matched the primary reward function. The Asynchronous Advantage Actor Critic (A3C) method is subsequently employed in this study to determine time-sensitive weights for each policy section. After that, such weights are formed with an instantaneous vector to choose the right policy section that will result in a decision. Multiple evaluations have shown that, for four assignments, the

recommended approach significantly outperforms three conventional MRA techniques in terms of episode time, episode reward, score difference, and success rate.

Huang *et al.* [82] propose a new goal-guided Transformer-enabled reinforcement learning (GTRL) technique in order to lead the visual depiction to interact with the target data and realize effective autonomous travel, where the real-world target states as a source of data have been taken. To be more precise, this study proposes employing the Goal-guided Transformer (GoT), a unique variation within the Visual Transformer, as the foundation for a perception system and pre-training using specialist prior knowledge to increase the data effectiveness. The decision-making system subsequently implements an RL algorithm, which generates decision instructions based on the goal-oriented image description provided by GoT. Consequently, this method encourages the scene description to focus primarily on attributes related to the objective, significantly improving the information effectiveness in the DRL learning procedure and improving navigational accuracy. Bai *et al.* [83] examine an adaptive RL controlling issue for a certain type of discrete-time, no-strict feedback system. Secondly, an RL-based govern adapting control technique is implemented using a backstepping approach to accomplish optimum control, as well as the multigradient recursive (MGR) technique is used for estimating the weight vectors. This is achieved by describing a compensating term for adjusting the controller along with using the feature from the radial-basis-function neural network (RBFNN). Lastly, the Lyapunov theory ensures the reliability of the control systems and establishes the semi-global uniformly ultimately bounded (SGUUB) nature of every signal in the system with closed loops. Two simulation scenarios that use the maritime vessel's course-keeping system demonstrate the reliability of this technique.

Chapter 4

Markov Decision Processes for Robotics

4.1. Introduction

Creating completely self-aware agents that communicate with their surroundings to acquire optimal behaviors while becoming better over time by using trial and error represents one of the main objectives of the area of AI. From robots that can perceive and respond to their environment to solely based on software agents that can communicate with media and natural language, creating AI systems that are sensitive and capable of learning remains a problem. RL provides an analytical mathematical foundation of experience-driven self-learning [84]. The use of ML approaches to robot control issues has advanced rapidly in the last several years. ML enables an agent to resolve an issue by learning from sample data or previous experience. Under supervised learning, the environment instructs the learner on what action to take for each and every input, giving the learner a clarified objective for each and every one of them.

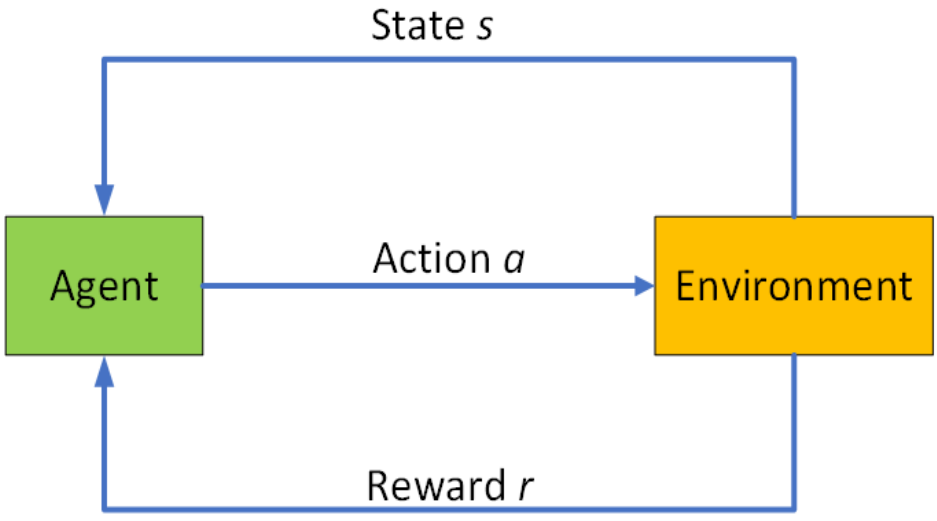

Fig. 4.1: Illustration of the Reinforcement Learning technique describes how a learning agent interacts with the environment.

As seen in Figure 4.1, RL usually takes place in an interactive environment. As a learning agent interacts within an originally unfamiliar environment, it gets instant feedback in the form of a reward and an interpretation of the state. After that, it computes a course of action and finally takes it. The environment changes towards an alternate state as a result of this activity. The procedure is then repeated when the agent obtains the latest presentation and the associated reward. The environment in RL is usually designed as a Markov Decision Process (MDP), with the prospective purpose being to optimize the overall reward by learning an approach to control. The learner in RL receives only partial feedback on their decisions. As a result, in the context of RL, the learner functions as a decision-making agent who acts in the surroundings and is rewarded (or penalized) for their efforts to resolve an issue. It must eventually figure out the optimal course of action, or the collection of steps that optimizes the overall reward, through a series of trial and error sessions [85]. Lastly, an agent or decision maker can develop a strategy to maximize an extended reward by interaction with the surroundings through the use of RL, a focused on objectives learning technology. The fundamental foundation of the MDP and RL in robotics is covered in this chapter.

4.2. Markov Decision Processes

The definition of MDP is a probabilistic method of decision-making that models the selection process of an unpredictable system using a framework of mathematics in situations where the outcomes are either arbitrary or under the control of a decision-maker who takes successive decisions over a period of time. MDPs use several factors, including the environment, agent behavior, and rewards, to determine the best course of action for the system to take next. A serial decision-making issue where an agent needs to choose a set of activities that optimizes reward-based criteria is described by a MDP [85], [86]. Initially, an MDP is abbreviated as equation (4.1):

$$M = \{S, A, T, r, y\} \tag{4.1}$$

Where,

- $S = \{s_1, \dots\dots, s_N\}$, is a finite number of state N which symbolises the changing surroundings.
- $A = \{a_1, \dots\dots, a_k\}$, is a collection of k possible actions that an agent might carry out.
- $T: S \times A \times S \rightarrow [0,1]$, is a transition model or function of transition probability, where the probability of state transition is denoted by $T(s, a, s')$ whenever utilizing action $a \in A$ for state $s \in S$ directing towards state in state $s' \in S$, which means $T(s, a, s') = P(s'|s, a)$.
- The discount factor is denoted by $y \in [0, 1]$.
- $r: S \times A \rightarrow \mathbb{R}$, is a reward function having a finite overall value by $R_m; r(s, a)$ represents the instant reward received when action $a \in A$ is implemented in state $s \in S$.

The agent-environment interaction shown in Figure 4.1, given an MDP M, proceeds as follows: Let, the current time is denoted by $t \in \mathbb{N}$, the environment random state is denoted by $S_t \in S$, and the action taken by agent time t is denoted by $A_t \in A$. Upon selection, the action is transmitted to the system, causing a transition as mentioned inequation (4.2) [85], [87]:

$$(S_{t+1}, R_{t+1}) \sim P(. | S_t, A_t). | \tag{4.2}$$

In specific, S_{t+1} is arbitrary and $P(S_{t+1} = s' | S_t = s, A_t = a) = T(s, a, s')$ keeps true for any $s, s' \in S$, $a \in A$. Moreover, $E[R_{t+1} | S_t, A_t] = r(S_t, A_t)$. The agent subsequently selects another action $A_{t+1} \in A$, analyzes the subsequent state S_{t+1} and reward R_{t+1}, and repeats the

procedure. The transition model *T* is specified by the series of state-action pairings, according to the Markovian hypothesis by equation (4.3) [85]:

$$P(S_{t+1}|S_t, A_t, \dots\dots, S_0, A_0) = P(S_{t+1}|S_t, A_t). \quad (4.3)$$

Transitions between states might be probabilistic or predictable. A particular action for an existing state usually leads to a similar subsequent state for a predictable situation; in the uncertain case, however, the subsequent state is random. Finding a theory for selecting actions that optimize the projected total reduced reward becomes a learning agent's objective as equation (4.4) [85]:

$$R = \sum_{t=0}^{\infty} \gamma^t R_{t+1} \quad (4.4)$$

Rewards obtained later on are significantly less deserving than those obtained at the beginning if γ is smaller than 1.

4.2.1. Policies and Value Functions

The agent chooses its course of action based on a unique feature known as policy. Policies can be defined as a mapping $\pi: S \times A \rightarrow [0, 1]$ that designates to every $s \in S$ a distribution $\pi(s, .)$ over *A* satisfying $\sum_{a \in A} \pi(a|s) = 1, \forall_s \in S$.

For any $s \in S$, there exists a deterministic constant policy, $\pi(.|s)$ is focused on an individual action, that means at any time $t \in \mathbb{N}$, $A_t = \pi(S_t)$. A function that converts each condition into a distribution of probabilities over the range of feasible actions $A_t \sim \pi(.|S_t)$ is known as a stochastic static policy. Π represents a category for all stochastic static policies.

The following procedure is followed when implementing a policy: Initially, a state known as S_0 is formed. Subsequently, an action $A_0 = \pi(S_0)$ is proposed by policy π and executed. With the help of reward function r and transition function *T,* a transition is generated to state S_1,

containing possibilities $T(S_0, A_0, S_1)$ and received a reward $R = r(S_0, A_0, S_1)$. This transition process continues, generating a sequence $S_0, A_0, S_1, S_1, A_1, S_2, S_2, A_2, S_3$,....., which is shown in Fig. 4.2.

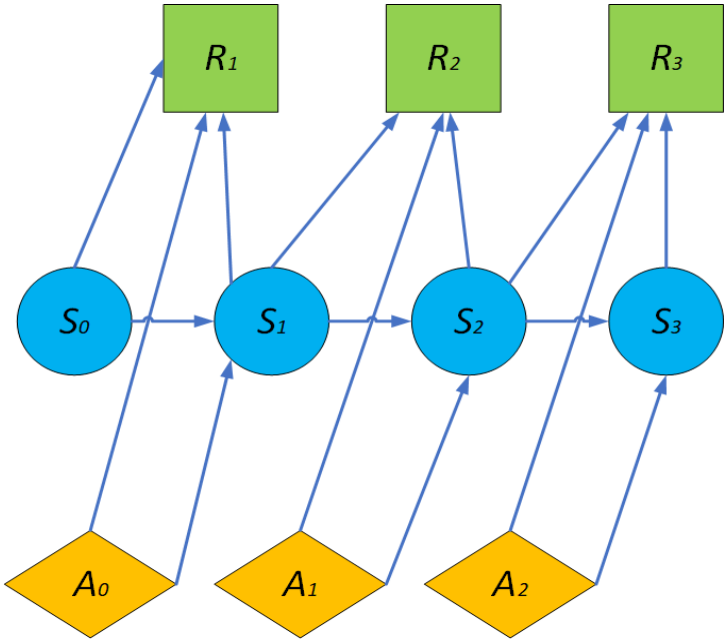

Fig. 4.2: Illustration of finite MDP decision network.

Value functions quantify the degree to which an agent is optimally situated in a particular state by evaluating states (or state-action sequences). Value functions assess the agent's perceived utility in a certain state by evaluating states or state-action sequences. For the purposes of this discussion, "how good" is interpreted as the projected future rewards—more specifically, the expected return. Naturally, the agent's potential rewards are contingent upon the behaviors it undertakes today. Value functions are therefore specified in relation to specific policies. In the context of a π policy, a value function is expressed by a function $V^{\pi}: S \to \mathbb{R}$, which assigns to every state the anticipated total of rewards that the agent is going to get up on initiating policy π execution via the state shown in equation (4.5) [85], [87]:

$$V^{\pi}(s) \to \mathrm{E}_{\pi}[\sum_{t=0}^{\infty} \gamma^{t} r(S_t, A_t) | S_0 = s], \quad \forall_s \in S \tag{4.5}$$

The unknown variable S_t denotes the state during time *t*, while the unknown variable A_t corresponds with the action performed at the same point within time and is like this, $P(A_t = a|S_t = s) = \pi(s, a).(S_t, A_t)_{t\geq 0}$ is the series of arbitrary state-action pairings produced when the policy π is put into practice. A static policy's value function might alternatively be sequentially expressed as equation (4.6) [85], [87]:

$$\begin{aligned} V^{\pi}(s) &= \mathrm{E}_{\pi}[\textstyle\sum_{t=0}^{\infty} \gamma^t\, r(S_t, A_t)|S_0 = s] \\ &= \mathrm{E}_{\pi}[r(S_0, A_0) + \textstyle\sum_{t=1}^{\infty} \gamma^t\, r(S_t, A_t)|S_0 = s] \\ &= r(s, \pi(s)) + \mathrm{E}_{\pi}[\textstyle\sum_{t=1}^{\infty} \gamma^t\, r(S_t, A_t)|S_0 = s] \\ &= r\big(s, \pi(s)\big) + \gamma \mathrm{E}_{\pi}[\textstyle\sum_{t=0}^{\infty} \gamma^t\, r(S_t, A_t)|S_0 \sim T(s, \pi(s), .)] \\ &= r\big(s, \pi(s)\big) + \gamma \textstyle\sum_{s' \in S} T\big(s, \pi(s), s'\big) V^{\pi}(s'), \end{aligned} \tag{4.6}$$

Where state *s* containing an action $\pi(s)$.

When considering the unpredictability of a stochastic strategy π(s), V π(s) can alternatively be expressed explicitly as equation (4.7) [85], [87]:

$$V^{\pi}(s) = \sum_{a \in A(s)} \pi(s, a)(r(s, a) + \gamma \sum_{s' \in S} T(s, a, s') V^{\pi}(s')) \tag{4.7}$$

Similarly, the basic action-value function of a policy π, $Q^{\pi}: S \times A \to \mathbb{R}$, is described as equation (4.8) [85], [87]:

$$Q^{\pi}(s, a) = E_{\pi}[\textstyle\sum_{t=0}^{\infty} \gamma^t\, r(S_t, A_t)|S_0 = s, A_0 = a], \tag{4.8}$$

Thus, for any $t > 0$, St will be distributed in accordance with $\pi(S_t, .)$. In conclusion, we identified the benefit function linked to π as equation (4.9):

$$A^{\pi} = Q^{\pi}(s,a) - V^{\pi}(s) \tag{4.9}$$

The optimal policy, represented by the symbol π^*, is one that optimizes the predicted total discounting reward across every state. There exists a single optimal policy with each finite MDP. The definitions of the optimum action-value function Q^* and the optimum value function V^* are shown in equations (4.10) and (4.11) [87]:

$$V^*(s) = \sup_{\pi} V^{\pi}(s), \quad s \in S \tag{4.10}$$

$$Q^*(s,a) = \sup_{\pi} Q^{\pi}(s,a), \quad s \in S, a \in A \tag{4.11}$$

Furthermore, the subsequent equations (4.12) and (4.13), relate the best value- and action-value functions.

$$V^*(s) = \sup_{a \in A} Q^*(s,a), \quad s \in S \tag{4.12}$$

$$Q^*(s) = r(s,a) + \gamma \sum_{s' \in S} P(s'|s,a) V^*(s'), \quad s \in S, a \in A \tag{4.13}$$

It is well known that the Bellman optimality equations are satisfied by V^* and Q^*, which are abbreviated as in equations (4.14) and (4.15) [87], [88]:

$$Q^*(s,a) = r(s,a) + \gamma \sum_{s' \in S} P(s'|s,a) \operatorname*{Max}_{b \in A} Q^*(s',b), \tag{4.14}$$

$$V^*(s) = r(s,a) + \gamma \operatorname*{Max}_{a \in A} r(s,a) + V^*(s'), \tag{4.15}$$

A policy that complies with $\sum_{a \in A} \pi(a|s) Q(s,a) = \operatorname*{Max}_{a \in A} Q(s,a)$ at every state $s \in S$ insatiable while comparing with Q function. It is widely recognized that all insatiable policies with respect to $Q^*$are ideal, which means these policies might be leveraged for generating all

static optimal policies. Here, we offer a few significant findings on MDP as abbreviated in the below-given theorems [85]:

Theorem 4.1 (*Bellman equations*): Lets consider $M = \{S, A, T, r, \gamma\}$ is an MDP and a policy is given as $\pi: S \times A \rightarrow [0,1]$, then, $\forall_s \in S, a \in A, V^{\pi}, and\ Q^{\pi}$ satisfies as following given equations (4.16) and (4.17) [85], [87]:

$$V^{\pi}(s) = r(s, \pi(s)) + \gamma \sum_{s' \in S} T(s, \pi(s), s')\, V^{\pi}(s'), \tag{4.16}$$

$$Q^{\pi}(s, a) = r(s, a) + \gamma \sum_{s' \in S} T(s, a, s')\, V^{\pi}(s'), \tag{4.17}$$

Theorem 4.2 (*Bellman optimality*): Let us consider $M = \{S, A, T, r, \gamma\}$ is an MDP and a policy is given as $\pi: S \times A \rightarrow [0,1]$, then, π is the optimal course of action with *M* if and only if, $\forall_s \in S$, which is described in the following given equation (4.18) [85], [87]:

$$\pi(s) = arg \operatorname*{Max}_{a \in A} Q^{\pi}(s, a) \tag{4.18}$$

Where the probability of transition $T(s, a, s') = P(s'|s, a)$.

4.2.2. Partially Observable Markov Decision Processes

In an MDP setup, the agent needs to be able to acquire accurate environmental state details at all times. However, given that the agent sees the environment through inadequate and restricted sensors and that data gathered in perception alone is insufficient to determine the state, this hypothesis might not prove true in real life. To deal with these situations, partially observable Markov decision processes (POMDPs) [89], were developed. A POMDP represents technically a tuple $\{S, A, T, O, Z, R\}$, where an MDP is $\{S, A, T, R\}$, the set of perceptions or observations is abbreviated as O, and the observation function is represented by Z (where,

$Z(o, s, a)$ represents the possibility of observing $o \in O$, while the structure has entered state s and the activity which brought it into such state will be a). An MDP's extension is called POMDP. An agent cannot explicitly view the fundamental state, yet an MDP is supposed to govern its dynamics in a POMDP framework for an agent decision procedure. Numerous real-world sequencing decision-making steps might be modeled by the POMDP system because of its sufficiently wide design. Applications cover servicing machines, robot navigation issues, and overall planning under ambiguity. The observations might be stochastic, meaning that various observations might be made in a similar state, or aliased, meaning that a single observation might be made in many states. As such, it is not possible to infer the system's state using the data. Alternatively, an observation might be regarded as proof of the state. Having faith state as shown in equation (4.19) [87], [90], which represents the agent's perception of the embedded state, is a distribution of probabilities over all feasible states.

$$f_t = [P_r(s_t = x^0), P_r(s_t = x^1), \dots\dots, P_r\left(s_t = x^{|s|-1}\right)]^T \tag{4.19}$$

The agent begins with a starting faith state, f_0, and employs Bayes' Rule as shown in equation (4.20) [87], [90] to modify its faith state f_t whenever an action takes place or an observation is obtained at o_{t+1}.

$$f_{t+1}(x) = P_r(s_{t+1} = s | f_t, a_t, o_{t+1})$$

$$= \frac{P_r(s_{t+1} = s, o_{t+1} | f_t, a_t)}{P_r(o_{t+1} | f_t, a_t)}$$

$$= \frac{\sum_{s' \in S} f_t(s')\, T(s, a_t, s')\, Z(o_{t+1}, s, a_t)}{\sum_{s' \in S} \sum_{s'' \in S} f_t(s')\, T(s, a_t, s'')\, Z(o_{t+1}, s'', a_t)} \tag{4.20}$$

As a result, the faith state f_{t+1} is an uncertain function of the action a_t, prior faith state f_t and observation o_{t+1}, which is shown in equation (4.21) [87], [90]:

$$f_{t+1} = T(f_t, a_t, o_{t+1}) \tag{4.21}$$

The likelihood for each observation o for the moment in time $t + 1$ might be determined using a faith state f_t at time t as expressed in equation (4.22) [87], [90]:

$$P_r(o_{t+1} = o|f_t, a_t) = \sum_{s\in S} \sum_{s'\in S} f_t(s)\, T(s, a_t, s')\, Z(o, s', a_t) \tag{4.22}$$

For a faith state f_t, the anticipated reward to carrying out action a is provided as abbreviated in equation (4.23) [87], [90]:

$$r(a|f_t) = \sum_{s\in S} f_t(s) R(s, a) \tag{4.23}$$

Each faith is a state in a Markovian faith state, which makes it possible to describe a POMDP as an MDP. Due to the fact that there are an endless number of faiths for every POMDP, the resultant Faith MDP will be specified in an infinite state space [91]. A tuple containing the faith MDP can be described as $\langle f, A, \tau, r, \gamma\rangle$, where the array of faith states for all POMDP states is denoted by f, A is the identical group of steps as described in POMDP's initial version, the faith state transition function is denoted by τ, the way in which a faith states are rewarded is denoted by $r: f \times A \to \mathbb{R}$, and the γ for initial POMDP is equivalent for the discount factor γ.

In faith, MDP, τ, and r must be obtained from the first POMDP. In equations (4.24) and (4.25), transitions and rewards derived consecutively for every $f, f' \in f, a \in A$ [87], [90]:

$$\tau(f, a, f') = \sum_{o\in O} P_r(f'|f, a, o) P_r(o|a, f) \tag{4.24}$$

$$r(f, a) = \sum_{x\in S} f(s) R(s, a) \tag{4.25}$$

Fig. 4.3. shows the faith states' temporal sequence. The Markov feature suggests that all of the data about the present state and, by extension, all of the prospective occurrences, is included identically in a similar way in the faith state and the system's whole experience (a series of observations and actions).

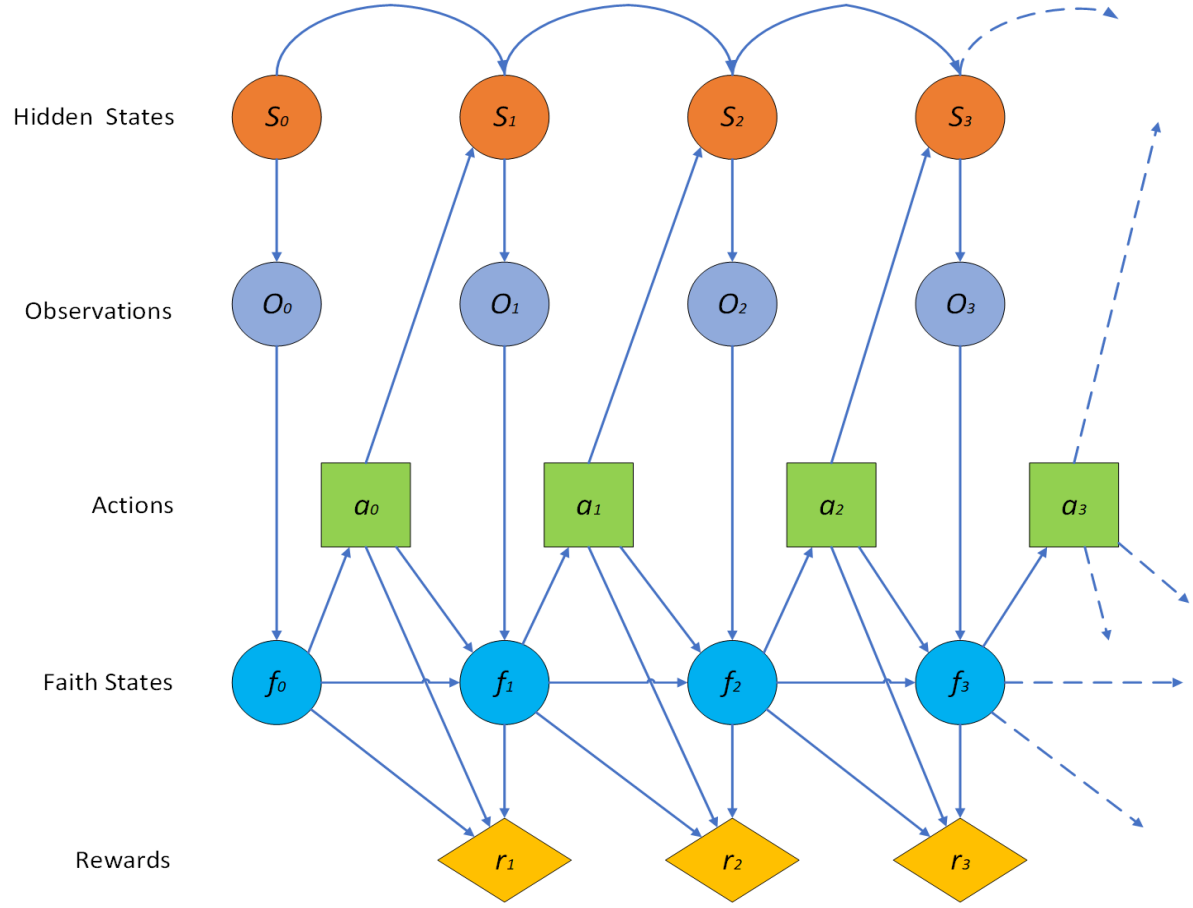

Fig. 4.3: Illustration of POMDP decision network.

4.3. Dynamic Programming Approach: Model-Based Techniques

A technique called dynamic programming (DP) might be used to find the best possible policy π^* for solving a specific MDP. Dynamic programming makes the assumption that the MDP, comprising the reward function and environmental transitional dynamics, is well understood [92]. As a result, these belong within the category of model-based learning

techniques. Model-free learning techniques, on the other hand, are going to be covered subsequent to this chapter and require only the perfect design of surroundings.

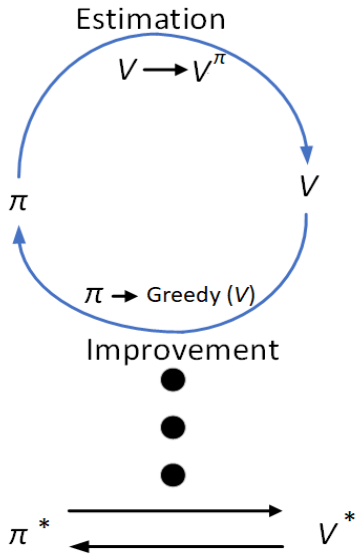

Fig. 4.4 (a): Relationship between improvement procedures and policy evaluation [85].

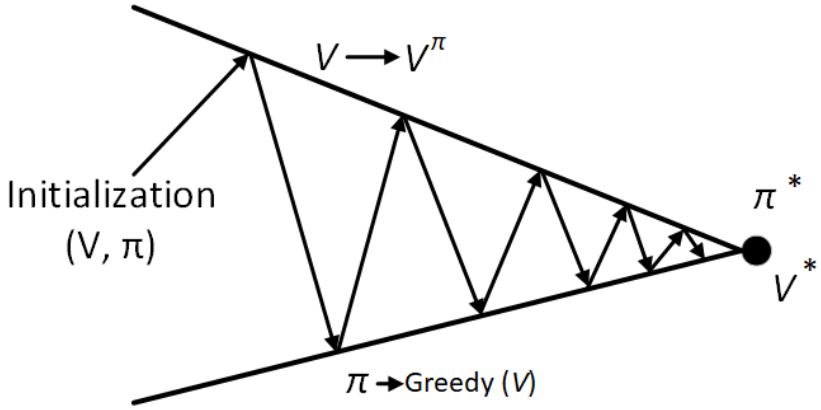

Fig. 4.4 (b): The policies and value function's progression towards their optimals [85].

There are two categories of dynamic programming techniques for MDP solutions policy iteration (PI) and value iteration (VI) [82]. Fig. 4.4 (a) and Fig. 4.4 (b) illustrate the generalized policy iteration (GPI) concept, which is the fundamental technique shared by both of these methods [85]. There are two interacting mechanisms in this principle. In the first phase, policy assessments, the value, V^{π} is computed, which evaluates the benefit of the present policy π. Information on the policy is gathered in the current phase with the aim of computing the

subsequent step, which is the policy enhancement phase. The actions' values are assessed in each state within this phase to identify potential enhancements or alternative actions in specific states that might be preferable to the ones that the existing policy recommends. Using the data in V^{π}, this stage determines an enhanced policy π' from the present policy π. The primary objective is to converge on a best-value function and an ideal policy, provided that both procedures keep updating every state.

4.3.1. Policy Iteration

Between the two GPI procedures, policy iteration iterates. Until an ideal policy of action is reached, the process is repeated. In Algorithm 4.1, this procedure is illustrated. It begins (from steps 1 to 3) using a randomly generated policy, π_t, and arbitrarily initializes the associated value function, V_k, assuming $k = 0$ and $t = 0$, and performing the processes for enhancing policies and evaluating policies repeatedly. The process (from steps 5 to 8) of evaluating a policy involves figuring out its action value (π_{t+1}) for each state $s \in S$ using computing equation (4.17). One effective method for solving this problem iteratively is to use the value function V_k from the preceding policy for initializing the value function for π_{t+1}. subsequently repeating the procedure in equation (4.26) [85], [87]:

$$\forall_s \in S: V_{k+1}(s) = r\big(s, \pi_t(s)\big) + \gamma \sum_{s' \in S} T(s, \pi_t(s), s') V_k(s') \tag{4.26}$$

Up to $\forall_s \in S: |V_k(s) - V_{k-1}(s)| < \epsilon$, over a certain error criterion ϵ. Determining (from steps 9 to 10) a greedy policy π_{t+1} provided the value function V_k represents the goal of policy enhancement in equation (4.27) [85], [87]:

$$\forall_s \in S: \pi_{t+1}(s) = \arg\max_{a \in A} [r(s, a) + \gamma \sum_{s' \in S} T(s, a, s') V_k(s')] \tag{4.27}$$

The procedure comes to an end when $\pi_t = \pi_{t-1}$, at which point π_t is the optimal policy, or $\pi^* = \pi_t$. All things considered, policy iteration produces a straight line of alternated policies and value functions in equation (4.28) [85], [87]:

$$\pi_O \to V^{\pi_O} \to \pi_1 \to V^{\pi_1} \to \cdots \ldots \ldots \to \pi^* \to V^0 \to \pi^* \tag{4.28}$$

The transitions from $\pi_O \to V^{\pi_t}$ include the policy assessment procedures, whereas the $V^{\pi_t} \to \pi_{t+1}$ transformations are accomplished by the policy enhancement procedures.

ALGORITHM 4.1: ALGORITHM FOR POLICY ITERATION

Algorithm 4.1: Policy Iteration

Input: $M = \{S, A, T, r, \gamma\}$, is an MDP model;
/* Starting */
$k = 0, t = 0$;
$\forall_s \in S$: start $\pi_t(s)$ with a random action;
$\forall_s \in S$: start $V_k(s)$ with a random value;
Repeat
/* Starting */
  Repeat
    $\forall_s \in S$: $V_{k+1}(s) = r\big(s, \pi_t(s)\big) + \gamma \sum_{s' \in S} T(s, \pi_t(s), s') V_k(s')$;
    $k \leftarrow k + 1$;
  Until $\forall_s \in S$: $|V_k(s) - V_{k-1}(s)| < \epsilon$;
/* Policy enhancement */
  $\forall_s \in S$: $\pi_{t+1}(s) = \arg\max_{a \in A} [r(s, a) + \gamma \sum_{s' \in S} T(s, a, s') V_k(s')]$;
  $t \leftarrow t + 1$;
Until $\pi_t = \pi_{t+1}$;
$\pi^* = \pi_t$;
Output: π^*, which is an optimal policy.

4.3.2. Value Iteration

One major disadvantage of policy iteration lies in the fact every iteration requires a thorough policy assessment. The technique of value iteration involves overlapping procedures of enhancement and assessment. The value iteration strategy ends the assessment procedure with just a single iteration, as opposed to totally dividing the procedures of enhancement and

assessment. As a matter of fact, it incorporates the policy enhancement stage right away into its iterations, allowing it to concentrate just on value function estimation. Value iteration, which is shown in Algorithm 4.2, might be expressed as a straightforward backup procedure as given in equation (4.29).

$$\forall_s \in S: V_{k+1}(s) = \max_{a \in A} [r(s,a) + \gamma \sum_{s' \in S} T(s,a,s') V_k(s')] \tag{4.29}$$

This process is continued within step 3 to step 6 till $\forall_s \in S: |V_k(s) - V_{k-1}(s)| < \epsilon$. At that point, the greedy policy in reference to the value function V_k is the optimal policy (Step 7). The value functions that value iteration generates are as follows given in equation (4.30) [82]:

$$V_0 \rightarrow V_1 \rightarrow V_2 \rightarrow V_3 \rightarrow V_4 \rightarrow V_5 \rightarrow \cdots \rightarrow \pi^* \tag{4.30}$$

ALGORITHM 4.2: ALGORITHM FOR VALUE ITERATION

Algorithm 4.2: Value Iteration

Input: $M = \{S, A, T, r, \gamma\}$, is an MDP model;
1: $k = 0$;
2: $\forall_s \in S$: start $V_k(s)$ with a random value;
3: **Repeat**
4: $\forall_s \in S: V_{k+1}(s) = \max_{a \in A} [r(s,a) + \gamma \sum_{s' \in S} T(s,a,s') V_k(s')]$;
5: $k \leftarrow k + 1$;
6: Until $\forall_s \in S: |V_k(s) - V_{k-1}(s)| < \epsilon$;
7: $\forall_s \in S: \pi^*(s) = \arg\max_{a \in A} [r(s,a) + \gamma \sum_{s' \in S} T(s,a,s') V_k(s')]$;
Output: π^*, which is an optimal policy.

4.4. Reinforcement Learning: Model-Free Techniques

One ML approach for handling sequential decision problems that might be represented as MDPs is called RL [93]. The theoretical framework of RL, which lies between unsupervised and supervised learning, addresses acquisition in sequential decision-making scenarios involving little or no feedback. A wide family of ML algorithms known as RL seeks to enable

an agent to understand how to act in a situation when the sole form of response is a scalar reward output. RL is better understood as an issue of learning or a framework compared to being defined by certain types of learning techniques [94]. The agent's objective is to do activities that will ultimately optimize the reward signal. In contrast to dynamic computing, which relies on the existence of a perfect simulation for its surroundings, RL focuses on determining the best policy of action in situations where an ideal system is unavailable. Reinforcement learning is therefore model-free [95]. Moreover, RL emphasizes estimation and partial data, as well as the necessity of analyzing and investigating to get a statistical understanding of this unidentified framework, in addition to MDPs.

A continuous or discrete collection, as well as multiple dimensional behaviors, might be represented as the agent and its surroundings in an RL issue to appear within a state $s \in S$ and capable of doing activities $a \in A$. In order to predict prospective states, a state s has the entire required information regarding the existing circumstance. The method employed to regulate the condition of the system is an action a. The agent additionally receives a reward R, after each step; this reward is a scalar number that is thought to depend upon the observation and state. It can also be represented as an arbitrary variable that just relies on such variables. A potential reward for completing the navigational goal might be created using the energy costs associated with the activities performed and the goals attained. Using RL, a policy π linking state and action is found which selects an action a for an individual state which optimizes the average anticipated reward. The probabilistic or stochastic nature of the policy π is observed. Every time a state is encountered, the previous one always performs an identical action in an expression of $a = \pi(s)$, whereas the latter selects an example using the variation across actions, i.e., $a \sim \pi(s, a) = P(a|s)$. The relationship among state, action, and reward must be discovered by the RL agent. Hence investigation is necessary that might either be explicitly incorporated into the policy or undertaken independently and solely as an element of the process

of learning. Various kinds of reward functions can be employed, comprising rewards dependent just on the present state $R = R(s)$, rewards dependent on the present state along with action $R = R(s, a)$, with rewards incorporating transitioning $R = R(s', a, s)$.

Robot learning is aided by the RL framework. Through contact with its surroundings, an AI-enabled robot can gain and learn fresh information. This aids in the creation of an autonomous robot that can learn on its own [96]. It enhances the robot's overall effectiveness in completing operations.

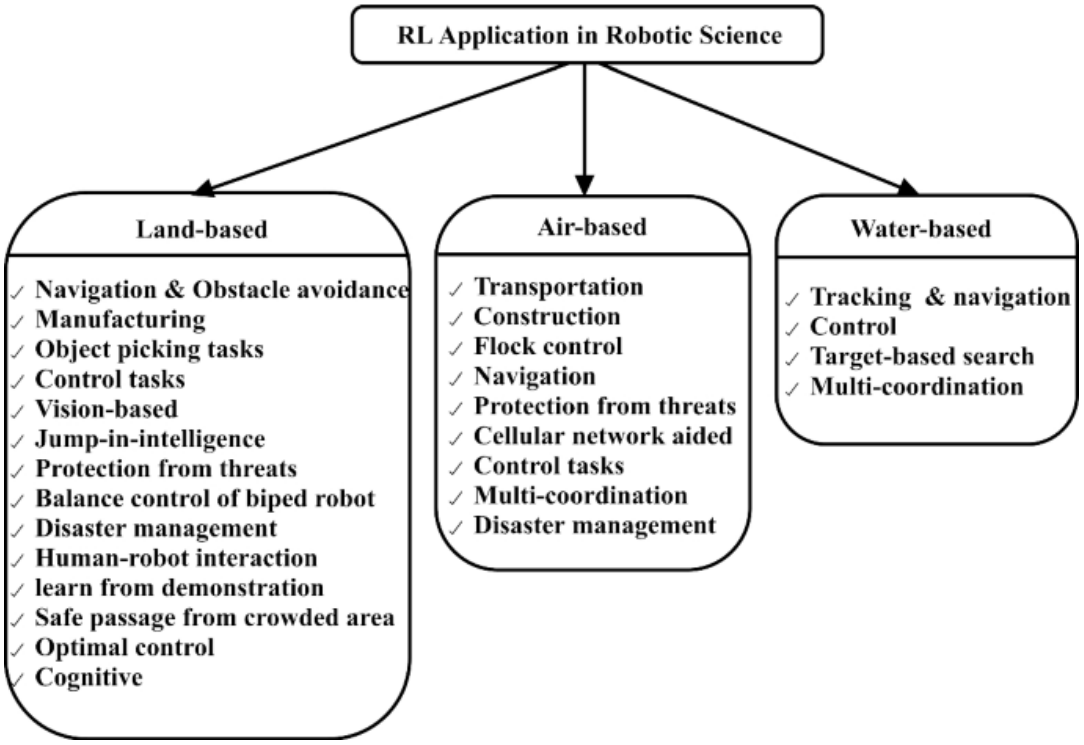

Fig. 4.5: List of applications of RL in robotics sciences [96].

The RL architecture facilitates agent learning using environmental interaction. An agent's first policy choice at the start of the procedure of learning will instruct the agent to proceed within its present state [96]. A complete list of applications of RL in robotics sciences is shown in Fig. 4.5. The agent transitions into its subsequent state and receives a reward message from the agent-environment relationship. In this case, the field specialist has already created a reward signal. In essence, a reward signal measures the quality of the activity in that particular state.

The acquired reward signals are applied to modify the policy. The present state's path, actions taken in that state, reward signals received, agent transitions into the subsequent state, and policy updates are all produced by this agent-environment relationship. A thorough overview of RL in robotics can be found in [97].

4.4.1. Objectives of Reinforcement Learning

The main objective of RL is the find the best policy $\pi *$ that links perceptions or state to action for the purpose of optimizing the anticipated return J, which is equivalent to the average anticipated reward. A finite-horizon framework will only look to optimize the anticipated reward during the subsequent H (time-)steps h, or the horizon H is abbreviated in equation (4.31) [85], [87]:

$$J = E\{\sum_{h=0}^{H} R_h\} \tag{4.31}$$

Model issues where we know the number of steps left might also employ this parameter. An alternate method is to apply a discount factor γ (where $0 \leq \gamma < 1$) on potential rewards is denoted in equation (4.32) [85], [87]:

$$J = E\{\sum_{h=0}^{\infty} \gamma^h R_h\} \tag{4.32}$$

For the learner, two aims naturally emerge. In the initial step, we try to determine the optimal plan of action after the conclusion of an interaction or training period. In the second scenario, the objective is to optimize the reward during the robot's complete interaction with the environment. The learner must initially explore their surroundings and is not informed of the optimal course of action, in comparison with supervised learning. With the goal of learning

more regarding the incentives and system dynamics, the agent must investigate by taking into account actions that it has not utilized before or activities that it is unsure about. It must choose between being cautious and sticking to tried-and-true activities that have (slightly) high rewards or taking a risk and attempting something new with the aim of finding new methods that have greater rewards. The term "exploration-exploitation trade-off" refers to this issue. Requiring communication between the learning agent and the surroundings, RL follows the same procedure as shown in Fig. 4.1 can be explained in the following ways:

- Discrete-time steps are used by a learning agent to interact with its surroundings;
- The agent monitors the surroundings at every step t, and it is rewarded with r_t with an illustration of the state s_t;
- An action a_t afterward performed in the surroundings becomes apparent by the agent;
- After seeing the novel surroundings, an agent is granted an additional state illustration ($s_t + 1$) along with a corresponding reward (r_t+1).

Off-policy versus on-policy approaches to RL might be differentiated according to the manner in which the agent selects an action [98]. Off-policy methods may develop without regard to the specific policy that is being used; therefore, they can use an exploratory approach throughout the process of learning which differs compared to the intended end policy. On-policy methods use the existing policy to gather a sample of environmental data. Therefore, research has to be included within the policy and dictates how quickly modifications can be made.

4.4.2. Monte Carlo Method

The practice of sampling is used in Monte Carlo techniques to determine value functions and identify the optimal policy [85]. The policy assessment phase within the dynamic programming-based approaches described above might be substituted with this process. Monte

Carlo approaches do not presume an exhaustive understanding of the surroundings, in contrast to dynamic programming (DP). Because Monte Carlo techniques do not require a specific transitional function, these processes are model-free [99]. All they need is expertise, which consists of sampling states, actions, and rewards via simulation or internet-based interaction. One can however achieve optimal conduct by learning via online interactions without any prior understanding of the nature of surroundings.

A model is necessary for learning by simulation background, but DP techniques need the framework to produce the whole distributions of probability for every conceivable transition, instead of just sample ones. By aggregating sample returns, Monte Carlo techniques are used to tackle issues related to RL. They operate on policy, meaning that they carry out implementations by putting the existing policy into practice in the system. Value function estimations are formed by tracking the frequency of rewards and transitions. For example, within an episode situation, one can determine the state-action quantity for a particular state-action pairing by aggregating all of the values obtained whenever one begins at that point.

Monte Carlo localization is well recognized as a very dependable technique for estimating a mobile robot's position. By analyzing the ranging data obtained from the laser scan to reference information generated from the provided environmental map, Monte Carlo Localization calculates the robot position [100]. The process of creating a global topology mapping involves using the thinning method on the provided global grid mapping, and this is often done offline. The Monte Carlo approach is used for robot navigation tasks whenever using RL for agent learning from historical data.

4.4.3. Temporal Difference Method

DP and Monte Carlo techniques are used to create Temporal Difference (TD) techniques[85]. TD learning techniques, in contrast to Monte Carlo approaches, are not required

to modify the value function until a prediction of a return is obtained, which might mean waiting till the conclusion of an episode. Rather, they wait until the following time step by using temporal errors. The variation between the initial estimation and the latest estimation of the value function—which accounts for the reward obtained in the most recent sample—is known as the temporal error. Prospective predictions serve as the training signals for forecasting during TD Learning. Unlike DP techniques, these iterative modifications only consider the samples of subsequent states, not the full distribution across the resulting states. Similar to conventional Monte Carlo approaches, such techniques are modeling-free because they can be learned directly through expertise despite the need for an explanation of the behavior of the surroundings. This is because it is not dependent on a modeling of the function of transition for determining the value function [101]. In this case, the sampling transitions within the MDP must be employed for estimating the value function because it is not possible to calculate computationally.

Watkins *et al.* (1989) introduced Q-Learning, a model-free, off-policy RL method [102]. The transition samples are processed gradually. The Q-value is repeatedly modified using equation (4.33) [102]:

$$Q'(s,a) \leftarrow Q(s,a) + \alpha(r(s,a) + \gamma \max_{b \in A} Q(s',b) - \mathrm{Q}(s,a)) \tag{4.33}$$

SARSA is a model-free, representative on-policy RL method [103]. Unlike Q-learning, which estimates potential rewards using $\max_{b \in A} Q(s',b)$, SARSA employs $Q(s', a')$ for a' the action carried out in s' under the present policy that creates the change in the sample (s,a,r,s',a'). In terms of the mathematical estimation process, the update rule is abbreviated in equation (4.34) [103]:

$$Q'(s,a) \leftarrow Q(s,a) + \alpha(r(s,a) + \gamma Q(s', a') - \mathrm{Q}(s,a)) \tag{4.33}$$

The Q-values will come closer to having probability 1 towards the optimized Q^* if every action is carried out in every state regardless of the number of repetitions and the learning rates α is degraded suitably for every state-activity set (s, a) [104]. A more rigorous condition for the investigation of every state and action is described in [105], which provides an identical assurance of converging for SARSA.

4.5. Model of Mobile Robot

All experimental studies are carried out in a simulation environment with MATLAB 2023a software in this thesis. As seen in Fig. 4.6, the MR simulation utilized in this dissertation is constructed in the simulation environment under the guidelines of the Festo Didactic MR system Robotino.

Fig. 4.6: Robotino: a MR platform for education and research [106].

Robotino has an omnidirectional motor, sensors, user interfaces, and adaptable expansions that allow for flexible application. Different programming environments enable the creation of unique usages. Every distance sensor for the Robotino reads out a level of voltage and this value is dependent on its distance from the reflected item. The robot has nine infrared vision sensors, positioned across the foundation with an angle of 40° by each other. Because of

its omnidirectional drive component foundation, the system can rotate about freely. As a result, it is presumed that our MR design has three wheels—two in the rear along one in front. Nine distance sensors are included inside the robot to help it measure the separations between objects in its immediate environment.

As seen in Fig. 4.7, the robot's initial point, goal point, and obstacle position make up the navigational environment. The robot has predetermined starting and goal positions; regardless of stationary or dynamic obstacles it will attempt to reach the destination via a collision-free route.

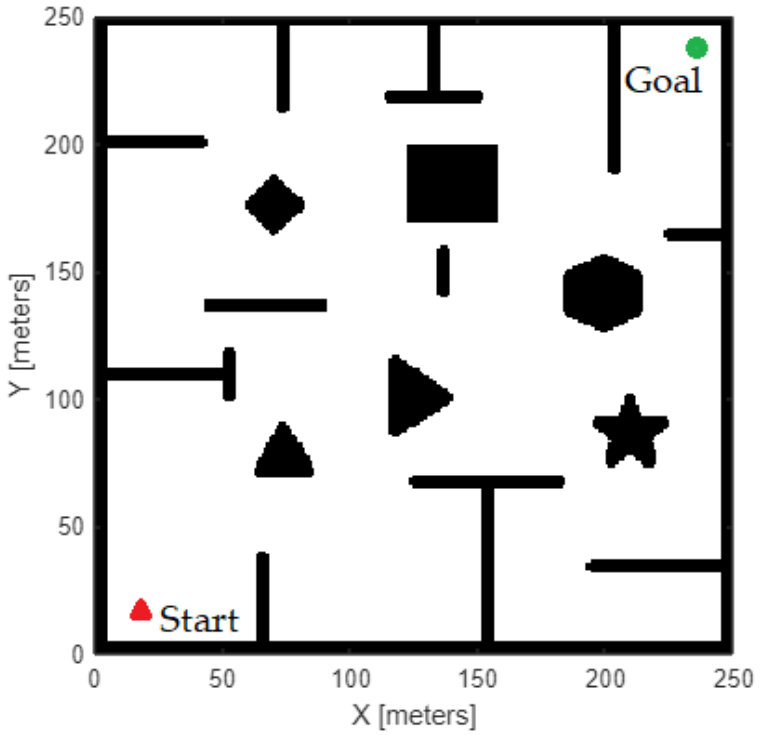

Fig. 4.7: Map of binary occupancy for the navigational task [107].

4.5.1. Unpredictability in Mobile Robot

There will always be unpredictability in MR navigation. Obstacles move randomly, noise might hinder sensors, and wheels might skid. MRs or decision-making agents that are automated must be able to respond to these kinds of setbacks and unforeseen circumstances,

which are often regarded as uncertainties. Although there are many other kinds of uncertainty, we focus on two in particular for the sake of clarity and conciseness, which are mentioned below [108].

- Uncertainty in prediction arises when the results of activities are not entirely anticipated. One way to conceptualize this is as a future state of uncertainty.
- Uncertainty in the present moment defines what is intended by sensing ambiguity. This happens, for instance, in robotics with inadequate or insufficient perceptions. We also acknowledge the scenario in which robots are completely blind.

Because the knowledge expressed in the state cannot always accurately predict the effects of the behaviors, MRs can be unpredictable and attempt to perform actions in certain states of the surroundings that can sometimes result in the anticipated outcome.

Chapter 5

Intelligent Navigation in Unknown and Complex Environment Using Reinforcement Learning

5.1. Introduction

These days, heuristic methodologies are increasingly used by academics due to their ability to mimic how people learn behavior [109]. RL is one of the most popular heuristic techniques which is widely applied for MR navigation. By using the RL technique, MR can determine the path by analyzing previous behavior. The MR sometimes referred to as an agent, perceives its surroundings, renders a decision, and receives a reward or a penalty based on the decision in response to its surroundings, enabling the effective application of the RL method. Once the MR eventually gets a bigger reward, it then adjusts its tactics, and the agent may learn different behavioral patterns by creating a large number of rules [110]. Researchers are interested in RL due to its robust visualization and ability to acquire knowledge from experience, which might help it solve the basic navigational issues experienced by MRs.

Developing human-like capabilities to enable robotics to do particular tasks seamlessly and naturally is a crucial objective in the field of robotics. We examined MDP's function in MR navigational instruction in the previous chapter. The robot can gain knowledge from demonstrations, and our learning system is able to apply it to the states. However, in specific situations, including investigating dangerous areas, the MRs are lacking access to past knowledge or examples, and we continue to require them to acquire skills in how to act in new environments. This is the primary issue that this dissertation addresses.

The essential technology that allows us to build robots that, like humans, are capable of self-learning new abilities is RL. Through interaction with the environment, RL is made

possible. The learners in RL are decision-making agents who act in their surroundings and get rewarded for their efforts to complete a task [111]. The agent aims to learn how to choose a series of acts, or a policy, that optimizes the entire cumulative reward over time. The signal, additionally referred to as reward (or penalize), determines the result of an action. RL might be estimated as a MDP. The transition function of state $T(s, a, s')$ must be known in order to implement model-based RL techniques. As an alternative, we concentrate increasingly on model-free RL when the surrounding models T are unknown, particularly in our research on autonomous navigation problems. The entire learning process consists of making trials and errors run. Our approach to solving the exploration versus exploitation compromise challenge involves using a Boltzmann probability distribution. This means that we must decide whether to explore unfamiliar and probably more rewarding states or to take advantage of previous experiences and choose actions that are, as soon as we can tell, helpful. Policies are stochastic in the given conditions.

Typically, classical RL methods are used on narrow arrays of states and actions; in other words, throughout trial-and-error runs, an agent is permitted to explore a fraction of the states. Large-scale state spaces are seen in real-world applications, which presents adaptation and the wrath of dimensionality issues. We estimate and extrapolate the value of each state by utilizing the framework of neural networks. Initially, after gathering the entire state-action pairings, we effectively apply neural networks to build an offline control strategy for MRs. Finally, we employ the neural network to perform continuous online learning without deliberately obtaining the output labeling. We provide an RL-based autonomous learning technique for situations in which learning agents must interact with their surroundings to determine a viable policy in the absence of specialist demonstrations. Autonomous navigation challenges involving MRs are tested in experimental simulation.

5.2. Model-Free Reinforcement Learning Techniques

Our research focuses on challenges related to RL and robot control, wherein an MR operates in an uncertain environment by making successive decisions about actions during a series of time steps, with the ultimate goal of optimizing the overall reward. The challenge is modeled as a MDP, with the following components: an action space A, a state space S, a reward function $r : S \times A \rightarrow \mathbb{R}$, and a transition patterns distribution $P(s_{t+1} \mid s_t, a_t)$ that satisfy the Markov characteristic $P(s_{t+1} \mid s_1, a_1, \ldots, s_t, s_t) = P(s_{t+1} \mid s_t, a_t)$, with any path $s_1, a_1, s_2, a_2 \ldots, s_T, s_T$ for state-action area. A random probability distribution policy $\pi(s_t, a_t) = P(a_t \mid s_t)$ is generated to choose actions and create a path of state, action, and reward $s_1, a_1, r_1 s_2, a_2, r_1, \ldots, s_T, s_T, r_T$ for $S, A, \mathbb{R}$.

Using an on-policy approach, one can determine the worth of the policy that guides decision-making. Value functions are modified with the outcomes of actions that are carried out in accordance with a policy. Off-policy techniques provide the ability to determine the optimal policy values without relying upon the agent's behaviors. By utilizing hypothetical activities that haven't been attempted yet, it modifies the projected value functions. It impacts anticipated value functions through the use of hypothetical, untested actions. Two popular model-free RL algorithms are Sarsa [103] and Q-learning [104].

5.2.1. Q-Learning

Q-learning determines the most effective policy for every discrete MDP by optimizing the anticipated value of the overall reward across every step, beginning at the present state. It can address issues with probabilistic transition and reward despite the need for changes, and it does not need system modeling. At its most basic, Q-learning uses tables for preserving data. This method breaks down as the variety of states and actions expands because there will be a

decreasing chance that the agent will arrive at a given state and carry out a given action. Function estimation might be used in conjunction with Q-learning. Because of this, the approach might be used to solve bigger issues even in cases where the state region is indefinite. In chaotic settings, Q-learning might overestimate the action metrics, which might slow down learning, due to the subsequent highest estimated action value being assessed by applying an identical Q functionality used for the present decision-making strategy.

For temporal difference learning, the Q-Learning algorithm acts as a crucial off-policy model-free RL technique. RL aims to teach the agent ways to behave within an unfamiliar environment by determining the optimal Q-value function which yields optimal results in any situation. It is demonstrated that the method convergence occurs with probability 1 to attain a near approximate to the action-value functions with every goal policy, providing adequate training for every ϵ-soft policy. In particular, when actions are chosen based on a more experimental or perhaps arbitrary policy, Q-Learning will be able to identify the optimal policy. In Q-learning, the updating of state-action variables is determined by equation (5.1) [87], [104]:

$$Q(s_t, a_t) \coloneqq Q(s_t, a_t) + \beta[r_{t+1} + \delta \max_a Q(s_{t+1}, a) - Q(s_t, a_t)] \tag{5.1}$$

The following variables are utilized for updating the Q-value:

- The learning rate is denoted by β, which lies between 0 and 1. When it has the value 0, no updates are made to the Q-values, hence no information is obtained. A high score, such as 0.8, indicates that learning might occur rapidly.
- The discount factor is denoted by δ, which lies between 0 and 1. This simulates the idea that rewards in subsequent years are not as valuable as those in the present. In mathematical terms, the method can only converge if the discount factor is smaller than 0.

In this instance, regardless of the policy that is followed, the learned action-value function, Q, precisely simulates Q^*, the optimum action-value function. This makes the process of algorithm assessment simpler and makes early convergence evidence possible. The policy remains to have an impact since it controls the state action pairings that have been modified and visited. But the only thing needed for proper convergence involves for every pair to keep getting updated. It can be demonstrated that Q_t will converge to Q^* at probability 1 with this hypothesis and a variation of the standard probabilistic estimation requirements on a series of step-size variables.

The algorithm 5.1 for Q-learning is presented below.

ALGORITHM 5.1: ALGORITHM FOR Q-LEARNING

Algorithm 5.1: One-step algorithm of Q-learning

Start $Q(s, a)$ randomly;
Repeat (for every episode):
    Start s;
    Repeat (for every step of the episode):
        Select a from s utilizing the Q-derived policy;
        Perform action a, monitor r, s';
        $Q(s, a) \leftarrow Q(s, a) + \beta[r + \delta max_{a'} Q(s', a') - Q(s, a)]$;
        $s \leftarrow s'$;
    Until the terminal is s
Until every episode finished.

5.2.2. SARSA

An on-policy method for TD learning is called the SARSA algorithm. The primary distinction among it and Q-Learning is that the Q-values are frequently not updated using the optimum reward in the subsequent state. As an alternative, another action is picked along with its reward according to a similar policy that chose the initial action. The quintuple $Q(s, a, r, s', a')$ is used to do the updating process, which is where the term SARSA originates. where r is the reward seen during the subsequent state, s, a are the latest state-action set, and

s, a represent the initial state and action. The definition of the Q-value updating rule is shown in equation (5.2) [87], [104]:

$$Q(s_t, a_t) \coloneqq Q(s_t, a_t) + \beta[r_{t+1} + \delta\, Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)] \tag{5.2}$$

Designing an on-policy control mechanism depending on the SARSA predictive approach is simple. Similar to other on-policy techniques, we continuously estimate Q^{π} for the action of policy π while also shifting π in the direction of greediness relative to Q^{π}.

The following is the standard SARSA algorithm 5.2:

ALGORITHM 5.2: ALGORITHM FOR THE SARSA ALGORITHM

Algorithm 5.2: On-policy SARSA algorithm

Start $Q(s, a)$ randomly;
Repeat (for every episode):
    Start s;
    Select a from s utilizing the Q-derived policy;
    Repeat (for every step of the episode):
        Perform action a, monitor r, s';
        Select a' from s' utilizing the Q-derived policy;
        $Q(s, a) \leftarrow Q(s, a) + \beta[r + \delta Q(s', a') - Q(s, a)]$;
        $s \leftarrow s'$; $a \leftarrow a'$;
    Until the terminal is s
Until every episode finished.

5.2.3. Approximating Functions using Feature-Based Illustrations

When dealing with large dimensionality issues, the quantity of states increases dramatically, rendering the storage of Q-value databases impractical. In order to apply Q-value approximations in practice, features, or the extrapolation of states, must be used. Features are functions that associate states to actual values and record significant state attributes. We can generate a Q-function, also known as a value function, for every state by utilizing characteristic representations and some weights in equations (5.3) and (5.4) [85], [87]:

$$V(s) = w_1 f_1(s) + w_2 f_2(s) + w_3 f_3(s) + \cdots + w_n f_n(s) \tag{5.3}$$

$$Q(s,a) = w_1 f_1(s,a) + w_2 f_2(s,a) + w_3 f_3(s,a) + \cdots + w_n f_n(s,a) \tag{5.4}$$

This representation allows us to summarize our experience into an attractive set of statistics. States might have similar characteristics, yet their corresponding values can vary greatly. In robotics, handling substantial state-action regions typically requires the application of either non-linear or linear depictions of features.

The Q-function for SARSA might be roughly represented by a linear function of characteristics. Although the agent's interactions sampled the reward using the policy that the agent is currently following along, compared to sampling the optimal policy, this approach employs the on-policy technique SARSA. There are several methods for obtaining a feature-based Q-function representation. In this part, we offer characteristics to the linear function using characteristics derived from the present state and the action. Assume that the state and actions have mathematical properties $\{f_1, f_2, \dots \dots, f_n\}$.Therefore, with state s and action a, $f_i(s,a)$ gives a value of the i-th attribute. These attributes might refer to different mathematical features, however, these are usually binary having domain $[0,1]$. The weights, $W = [W_0, W_1, \dots, W_n]$ for an array of values. Assume that W_0 is not required to qualify as a particular instance since there is a further characteristic f_0 and its value is constantly 1. The Q-function shown in equation (5.5) can be represented by these characteristics [85], [87]:

$$Q_W(s,a) = W_0 + W_1 f_1(s,a) + \cdots + W_n f_n(s,a) \tag{5.5}$$

To update $Q(s,a)$, a SARSA experience of the type $\{s,a,r,s',a'\}$ yields an updated estimation of $r + \delta\, Q(s',a')$. For the purposes of linear regression, this experience can potentially be utilized as data. Let $\varphi = r + \delta\, Q(s',a') - Q(s,a)$. W_i- weight can be updated by equation (5.6) [85], [87]:

$$W_i \leftarrow W_i + \beta\varphi f_i(s,a) \tag{5.6}$$

After that, the SARSA might utilize this update, making the algorithm 5.3.

ALGORITHM 5.3: APPROXIMATION OF THE LINEAR FUNCTION FOR SARSA

Algorithm 5.3: Approximation of linear function for the SARSA algorithm

Input:
$f = \{f_1, f_2, \ldots\ldots, f_n\}$: an array of attributes;
$\delta \in [0,1]$: discount factor;
$\beta > 0$: the gradient descent steps size.
1: Start weights $W = [W_0, W_1, \ldots, W_n]$;
2: Perceive present state s;
3: Choose action a;
4: **Repeat**
5: Perform action a;
6: Perceive state s' and get reward r;
7: Select action a' utilizing the Q_W-derived policy;
8: Let $\varphi = r + \delta\, Q(s', a') - Q(s, a)$;
9: **For** $i = 0$ to n **do**
10: Updating weights: $W_i \leftarrow W_i + \beta\varphi f_i(s,a)$;
11: **End for**
12: $s \leftarrow s'$; $a \leftarrow a'$;
13: **Until** termination.
Output: W^*, which is an optimal weight.

5.3. Q-Learning Technique Based on Neural Network

The addition we introduced to Q-learning by adding a neural network is presented in this part. The ability of Q-learning to analyze the predicted usefulness of various actions without having an environmental model is one of its advantages. Stochastic rewards and transitions are an issue that Q-learning might solve. RL becomes unpredictable or volatile whenever Q is represented by an equation with a nonlinear estimator, such as ANN. The associations in the observed series, indicate that modest changes to Q might drastically change the agent's policy and the overall distribution of information, and the relationships among Q and the goal values are the sources of this unpredictability.

5.3.1. Neural Network

Layers comprise an artificial neural network (ANN), where every layer is made up of many "neuron" nodes. As seen in Fig. 5.1, a neuron acts as a computing unit that is able to comprehend inputs, interpret them, and provide an output. Numerous neurons are connected to form the entire network. Each circle within this figure corresponds to one neuron. The network's leftmost component is known as the input layer, and its rightmost component is the output layer. As the values for the intermediate layer of nodes will not be visible within the training array, it is known as the hidden layer. Both the inputs and the outputs of the ANN are represented by both input and output layers, respectively. We refer to the neurons with the label "+1" as bias elements. A bias unit continuously produces +1 and contains nothing to input. The ANN has 4 input units (not including its bias component), 4 hidden units (not including the bias component), and 1 output unit.

We designate each layer l as L_l, with the value n_l to indicate the number of layers. The input layer is the layer L_l and the output layer is the layer L_{n_l} in Fig. 5.1, where $n_l = 4$. Weights are assigned to indicate the linkages that join two neurons, signifying the power of the link between the neurons. The weight linked to the link among element j in layer l and element i in layer $l + 1$ is represented by the notation $w_{ij}^{(l)}$, and the variables used within the ANN are $(w, b) = (w^{(1)}, b^{(1)}, w^{(2)}, b^{(2)}, w^{(3)}, b^{(3)})$. Furthermore, the bias connected to unit i within layer $l + 1$ is denoted as $b_i^{(l)}$. As a result, $w^{(1)} \in \mathbb{R}^{(4\times4)}$and $w^{(2)} \in \mathbb{R}^{(1\times4)}$. For the purpose of controlling their output, every neuron in the neural system has an activation function. The activation is $a_i^{(l)}$ for unit i within l-layer. In the layer of input L_1, $a_i^{(l)} = x_i$, the input for i-th of the entire ANN. Also, in other layers, $a_i^{(l)} = f(z_i^{(l)})$. In this case, $z_i^{(l)}$ represents the entire sum weighed of all inputs, comprising the biased factor, to unit i within layer l. The ANN produces

a real number, denoted as the hypotheses $H_{w,b}(x)$, provided the set value of its variables (w, b). Particularly, the estimation that this ANN shows is provided in equation (5.7) [85]:

$$\left.\begin{aligned} a_1^{(2)} &= f\left(w_{11}^{(1)}x_1 + w_{12}^{(1)}x_2 + w_{13}^{(1)}x_3 + w_{14}^{(1)}x_4 + b_1^{(1)}\right) \\ a_2^{(2)} &= f\left(w_{21}^{(1)}x_1 + w_{22}^{(1)}x_2 + w_{23}^{(1)}x_3 + w_{24}^{(1)}x_4 + b_2^{(1)}\right) \\ a_3^{(2)} &= f\left(w_{31}^{(1)}x_1 + w_{32}^{(1)}x_2 + w_{33}^{(1)}x_3 + w_{34}^{(1)}x_4 + b_3^{(1)}\right) \\ a_4^{(2)} &= f\left(w_{41}^{(1)}x_1 + w_{42}^{(1)}x_2 + w_{43}^{(1)}x_3 + w_{44}^{(1)}x_4 + b_4^{(1)}\right) \\ H_{w,b}(x) = a_1^{(3)} &= f\left(w_{11}^{(2)}a_1^{(2)} + w_{12}^{(2)}a_2^{(2)} + w_{13}^{(2)}a_3^{(2)} + w_{14}^{(2)}a_4^{(2)} + b_1^{(2)}\right) \end{aligned}\right\} \quad (5.7)$$

We might use the activation function $f(\cdot)$ to be applied to vectors element-wise in an additional concise expression: $f([z_1, z_2, z_3, z_4]) = [f(z_1), f(z_2), f(z_3), f(z_4)]$. Thus, equation (6.7) can be rewritten in the following way in equation (5.8) [85]:

$$\left.\begin{aligned} a^{(1)} &= x \\ z^{(2)} &= w^{(1)}a^{(1)} + b^{(1)} \\ a^{(2)} &= f(z^{(2)}) \\ z^{(3)} &= w^{(2)}a^{(2)} + b^{(2)} \\ a^{(3)} &= f(z^{(3)}) \\ z^{(4)} &= w^{(3)}a^{(3)} + b^{(3)} \\ H_{w,b}(x) = a^{(4)} &= f(z^{(4)}) \end{aligned}\right\} \quad (5.8)$$

The vector of the input layer values is $x = [x_1, x_2, x_3, x_4]^T$. Forward propagation is the term used for the computing procedure that goes from input to output. Furthermore, we might determine the activation value $a^{(l+1)}$ of the following layer $l + 1$ as follows using the activation $a^{(l)}$ for each layer, l is shown in equation (5.9) [85]:

$$\left.\begin{aligned} z^{(l+1)} &= w^{(l)}a^{(l)} + b^{(l)} \\ a^{(l+1)} &= f(z^{(l+1)}) \end{aligned}\right\} \quad (5.9)$$

We shall select $f(\cdot)$ to represent the sigmoid function $f: \mathbb{R} \to]-1, +1[$ in this thesis as shown in equation (5.10) [85]:

$$f(z) = \frac{1}{1 + \exp(-z)} \quad (5.10)$$

The derivative of equation (6.10) is given below in equation (5.11) [85]:

$$f'(z) = f(z)\big(1 - f(z)\big) \quad (5.11)$$

Utilizing matrix-vector computations allows us to significantly increase computation speed, which is one benefit of organizing every parameter and variable within matrices.

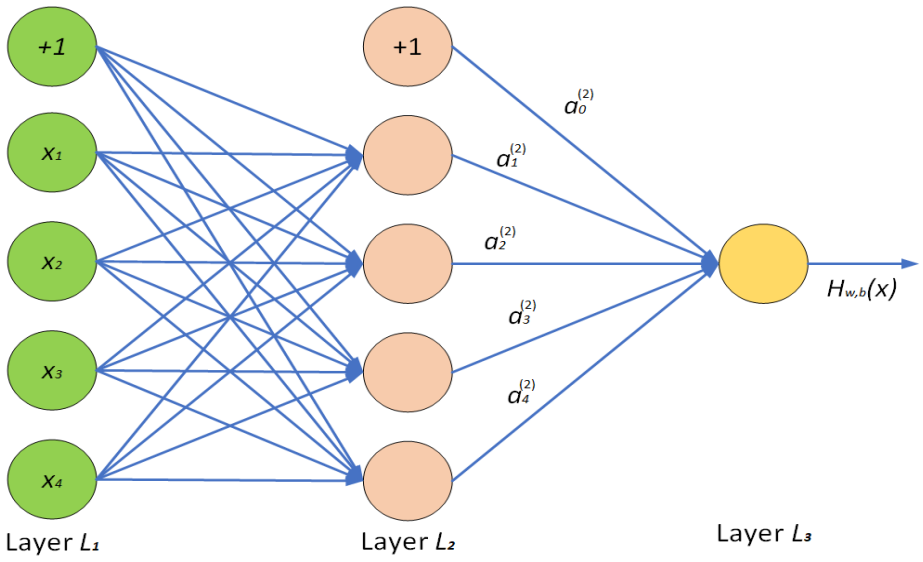

Fig. 5.1: An example of ANN used in this simulation.

5.3.2. State and Action Spaces

We must first establish the state spaces S and action spaces A, because RL methods might be represented as an MDP. The goal of Q-learning is to acquire an ability to map the input of the state to the output of action. With a navigation challenge, the robot uses its sensors to sense its surroundings and reasoning abilities to decide what to do in each state depending on the present state in the environment [112]. There are eight distinct robot actions that constitute the action space A. Seven of these are fundamental movement actions: go forward (F), make a left turn at 30^0 (L30), a left turn at 45^0 (L45), a left turn at 60^0 (L60), a right turn at 30^0 (R30), a right turn at 45^0 (R45), a right turn at 60^0 (R60), and take emergency steps in order to go backward (B). The orientation of the robot's trajectory determines the eight actions. Since an actual vehicle is greatly endangered by a sudden turn, a turn at 90^0 is not specified. Reversing direction is considered an emergency maneuver that is only carried out when a robot has no alternate route to follow and is unable to turn around to prevent obstacles. As a sort of "instinct" action for MRs, this activity has not been taken seriously in the autonomous learning processes. In summary, the action space will be abbreviated in equation (5.12):

$$A = \{F, L30, L45, L60, R30, R45, R60, B\} \tag{5.12}$$

A substantial but limited number of states make up the state space $S = \{S_1, S_2, \ldots, S_N\}$, and every state S_t is specified using the 11-dimension representation of states. We consider 9 sensors in the MR system. Thus, the multidimensional computational complexities are represented by $D_t = [d_1, d_2, d_3, d_4, d_5, d_6, d_7, d_8, d_9,]^T \in \mathbb{R}^{9\times1}$. A robot's job is to determine if a barrier detected by a sensor is a wall that might not be avoided or a substance that might be avoided. Therefore, on each of the nine sensors within D_t, we add a vector $U_t \in \mathbb{R}^{9\times1}$ that indicates whether an object obstacle ($U = 0$) or an immovable barrier or wall ($U = 1$) is

identified. As a result, a state is represented using equation (5.13) [85] and might be characterized by four sets of attributes. The target region is $V_t \in \{1,2,\dots,9\}$ and the indicator is $I_t \in \{0,1\}$ that estimates if the MR has recognized the goal.

$$S_t = \begin{bmatrix} D_t \\ V_t \\ I_t \\ U_t \end{bmatrix} \in \mathbb{R}^{16\times 1} \tag{5.13}$$

5.3.3. Reward Function

The instantaneous feedback for an activity performed in a specific state is measured by the reward function. It assesses how well or poorly the action was carried out. Prior to providing the reward function, every environmental state is divided into four properties, referred to by the term the state property p_t as shown in equation (5.14) [85], at every point in time.

$$p_t = \begin{cases} SS, & d_{bou} \le d_{r-0} \\ CS, & d_{cozy} \le d_{r-0} \le d_{bou} \\ DS, & d_{col} \le d_{r-0} \le d_{cozy} \\ WS, & d_{r-t} \le d_{win} \\ FS, & d_{r-0} \le d_{col} \end{cases} \tag{5.14}$$

Where,

- SS (Safe State), is a state in which no detectable outside obstacles are present.
- CS (Cozy State), is a condition when the robot's chances of colliding against nearby barriers are minimal or nonexistent.
- DS (Dangerous State), is a condition where there is a significant possibility that a robot might crash into certain environmental barriers.
- WS (Winning State), is a final stage in which the robot arrives at its objective.
- FS (Failure State), is a terminating condition in which the robot crashes into obstacles.

The robot's distances to obstacles and its target are defined by d_{r-0} and d_{r-t}. The border width (sensor sensing range) between SS and any other states is specified by d_{bou}. The width of the area of collision surrounding the obstacle is defined by d_{col}. The length of the winning zone surrounding the goal is defined by d_{win}. The comfy length is defined by d_{cozy}. Table 5.1 defines the reward function $r(t)$ by considering the state characteristics. The minimal distances among the MR and nearby obstacles at time $t-1$ and time t are d_{min}^{t-1} and d_{min}^{t}, respectively. The warning distance, or d_{warn}, indicates when a robot is getting extremely close to a certain obstacle.

TABLE 5.1: REWARD FUNCTION [85]

State Transition	Extra Criteria	r
Winning State ← Other State		1
Cozy State ← Safe State		0
Safe State ← Cozy State		0
Cozy State ← Dangerous State		0.5
Dangerous State ← Cozy State		0
Failure State ← Dangerous State		-1
Dangerous State ← Dangerous State (approaching obstacles)	$d_{warn} \leq d_{min}^{t} \leq d_{min}^{t-1} - 2$	-0.2
	$d_{min}^{t} = d_{min}^{t-1} - 1 \leq d_{warn}$	-0.2
	$d_{min}^{t} = d_{min}^{t-1} - 2 \leq d_{warn}$	-0.5
	$d_{min}^{t} \leq d_{min}^{t-1} - 3$	-1
Unsafe State ← Unsafe state (evading obstacles)	$d_{warn} \leq d_{min}^{t}$	0.6
	$d_{warn} \leq d_{min}^{t-1}$	0.4

5.3.4. Policy for the Stochastic Control

As opposed to receiving explicit instruction, an RL agent chooses its actions based on both recent and prior decisions, as well as the outcomes of previous acts. We might use the MC techniques, for instance, to determine the state values when we have access to every state action (s, a) in an adequate number of instances. Nonetheless, this approach is unrealistic, and, worse still, it is unlikely to visit several state-action pairings again. Handling the trade-off between exploration and extraction is crucial. In our approach, a probabilistic control policy can be expressed by transferring a Boltzmann distribution. Depending on its Q-values, a learning agent attempts different behaviors within a probabilistic way. A specific state, denoted by s, yields an action, a, having probability shown in equation (5.15) [85]:

$$\pi(s, a) = P(a|s) = \frac{e^{\frac{Q(s,a)}{T}}}{\sum_{b \in A} e^{\frac{Q(s,b)}{T}}} \tag{5.15}$$

whereby the choice of action uncertainty is controlled by a temperature, T. When T is extreme the agent selects an unanticipated action since the entire action Q-values appear to be identical. When T is inadequate the actions have different Q-values, and choosing actions containing the greatest Q-value is preferable. Since stochastic exploring requires too long to concentrate on the optimal actions during the complete self-learning procedure, we avoid setting the temperature as fixed.

Since every $Q(s, a)$ is initially produced incorrectly, a large T is chosen to ensure that every action has an approximately equal probability of getting chosen. A great deal of unplanned investigation has been done over time, and the agent might ultimately take advantage of the information it has been gathering. As a result, the agent reduces T and increases the likelihood of selecting actions based on greater Q-values. Ultimately, T gets closer to 0 (purely

exploited) as we suppose Q is convergent to Q^*, and we usually choose an action using the greatest Q-value as shown in equation (5.16) [85]:

$$P(a|s) = \begin{cases} 1, & if\ Q(s,a) = max_{b \in A} Q(s,b) \\ 0, & otherwise \end{cases} \quad (5.16)$$

To summarize, the agent begins with a substantial amount of exploration and gradually shifts to exploitation, until eventually, we are only investigating (s, a) which have performed less efficiently in the past.

5.3.5. Iteration of State-Action Value

The mapping policy between the observed environmental condition and the action that will be executed is expressed by the Q-value function. A single action for a given state and a single Q-value $Q(s_t, a_t)$ relate to each other. Our major areas of interest for research include autonomous navigation challenges. They feature extensive state and action spaces, similar to various real-world robot applications. Typically, a Q-table contains every one of its state or action values. For large-scale issues, however, it becomes operationally costly and impractical. In our technique, we recommend applying a three-layer ANN to forecast every state's Q-values. The state aspects that MR observes in its surroundings serve as its inputs, and each of its action's Q-values are its results.

There are 16 neurons within the input layers and 5 within the output layers of the neural network. Additionally, the hidden layers have eight neurons. Using the weights from the neural network as a guide, the iteration of action values is accomplished. The goal value in neural network training for Q-learning is the optimum one at the end of every value iteration phase. The network's weights do not operate linearly in Q-values of action. The NN weights are updated to achieve the iteration of the action value.

Using the weights of the NN as a guide, iteration of the action value is accomplished. The activations in the output and hidden layers are computed using the sigmoid function. The Q-value of adopting at for s_t is specified using $Q(s_t,\ a_t)$, which we represent as a vector representing every action value for the state s_t. Therefore equation (5.17) [85]:

$$Q(s_t) = \begin{bmatrix} Q(s_t, a_1) \\ Q(s_t, a_2) \\ Q(s_t, a_3) \\ Q(s_t, a_4) \\ Q(s_t, a_5) \end{bmatrix} \tag{5.17}$$

The NN is used for supervised learning, with specific label provision during every training state-action combination. On the other hand, NN utilized for RL lacks label outputs. The goal value for NN training in Q-learning comprises the optimum one at the end of every value iteration phase. This update rule is abbreviated in equation (5.18) [85]:

$$Q_{k+1}(s_t, a_t) \coloneqq\ Q_{k+1}(s_t, a_t) + \beta \max_{a \in A} [r_t + \delta\ Q_k(s_{t+1}, a) - Q_k(s_t, a_t)] \tag{5.18}$$

where state-action combinations are created at random between 0 and 1, and the starting Q_0 action values for all. For the $(k+1)^{th}$ iteration, $Q_{k+1}(s_t, a_t)$ is regarded as the goal value of the real value $Q_k(s_t, a_t)$.

5.3.6. Algorithms

A scenario with an agent that explores all unfamiliar surroundings makes up the Q-learning algorithms. Finding the best action-selection strategy in every MDP constitutes the aim of Q-learning. A policy is an agent's set of behaviors that maximizes the overall anticipated reward for the current condition. Using NNQL, an independent navigation problem might be split into two steps. The initial step is the training phase, which gives the robot the power to

acquire knowledge on its own, and the subsequent one includes the navigating procedure, which utilizes the learned policy to carry out an autonomous navigational mission [113].

5.3.6.1. NNQL Training Procedure

The MR is trained by subjecting it to several learning sessions, all of which have a unique setting. The robot has predetermined start locations as well. The diversity enables the robot to experience a wider range of scenarios, perhaps quickening its pace of learning. Every episode begins with an assessment of the surrounding conditions. With the use of its sensors, the robot locates nearby obstacles, and it is given an approximate objective zone. After verifying whether its present state is sufficiently secure, the robot performs a focused-on-target activity in which it shifts its orientation in the direction of the target location and advances a single step in an attempt to reach nearer. In the event that its present state is not secure, the robot transmits the characteristics of the present state via the FFNN and generates every conceivable Q-value. The robot selects an action and adjusts states in accordance with the action-selecting tactics, which is illustrated using a probabilistic control policy. After that, the robot verifies the newly created state, gets the reward right away, and modifies the Q-values appropriately. The NN uses the backpropagation technique to modify the network's weights after receiving the revised Q-values.

There are a set number of movement steps for every episode. The robot must complete the stages in order to arrive at its goal. An episode ends and a fresh one begins whenever a robot runs over the steps and misses the objective, or if it encounters an obstacle and still manages to obtain the goal. The utilization of the collected pattern of state-action pairings and related Q-values is crucial for training effectiveness. Updating a single Q-value during a moment is possible with single-step Q-learning [114], [115]. The prior values of actions are deleted but only the present Q-value is revised whenever the robot reaches its new state. Others employed

batch learning, which, after each of the Q-values is gathered, modifies every Q-value [116]. There are certain benefits to this as well. We are unable to verify that the obtained Q-values represent the optimum goal values in the absence of an online update. Furthermore, awaiting all values to be acquired is a waste of time. In addition to updating the Q-values operation, we recommend gathering earlier values for combined training. Algorithm 5.4 provides the training algorithm.

ALGORITHM 5.4: NNQL TRAINING ALGORITHM

Algorithm 5.4: NNQL Training Algorithm

1: Randomly start the NN weights $W^{(1)} and\ W^{(2)}$;
2: **For** every episode **do**
3: Set the goal position and create obstacles randomly;
4: Set the initial position and orientation of MR at $[x_0, y_0]$ and θ_0;
5: Observe the present state and property of the state as s_1 and p_1 respectively;
6: Input ≔ [], goal≔ [];
7: $t \leftarrow 1$;
8: **For** every movement step **do**
9: Estimate every action value $\{Q(s_t, a_i)\}_i$ for state s_t using NN;
10: Choose only one action a_t based on the probabilistic policy $\pi(s, a)$ as stated in (6.15) and then proceed;
11: Observe the novel state and property of the state as s_{t+1} and p_{t+1} respectively;
12: Acquire the instant reward (r_t);
13: Change the Q-value functions using (6.18) through $Q(s_t, a_t)$ to $\acute{Q}(s_t, a_t)$;
14: Scale the $\acute{Q}$ characteristic to the interval $[0, 1]$;
15: Insert s_t in the input and $\acute{Q}(s_t, a)$ in the goal;
16: Utilize SGD for updating both weights $W^{(1)} and\ W^{(2)}$, and to train (input, goal);
17: **If** s_{t+1} represents a winning state or a failing state **then**
18: Initialize a fresh episode;
19: **End if**
20: $t \leftarrow t + 1$;
21: **End for**
22: **End for**

5.3.6.2. NNQL-based Robot Navigation

The robot uses its trained strategy, which is still probabilistic but almost predictable, to navigate through different scenarios in the future. The robot first determines its present

state before beginning its journey around its surroundings. In a secure state, the MR just needs to shift its position in the direction of the goal and take a single step forward, bypassing the requirement to adhere to the policy. The robot keeps going as long as it reaches a non-safe area, at which point it must implement its trained control policy. The robot generates every conceivable state-action Q-value using the FFNN. The action with the highest Q-value is taken by the MR in a greedy manner. The robot then determines its latest state and continues the action-choice procedure until it either crosses an obstacle or achieves its goal. Algorithm 5.5 illustrates the navigation algorithm.

ALGORITHM 5.5: NNQL-BASED ROBOT NAVIGATION ALGORITHM

Algorithm 5.5: NNQL-Based Robot Navigation Algorithm

1: Enter the pre-trained NN weights $W^{(1)} and\ W^{(2)}$;
2: Provide the goal and randomly generate obstacles;
3: Set the initial position and orientation of MR at $[x_0, y_0]$ and θ_0;
4: $t \leftarrow 1$;
5: **For** every mobile step **do**
6: Observe the present state and property of the state as s_t and p_t respectively;
7: **If** s_t represents a winning state or a failing state **then**
8: Stop the navigation;
9: **End if**
10: Estimate every action value $\{Q(s_t, a_i)\}_i$ for state s_t using NN;
11: Choose only one action a_t based on greedy policy and then proceed;
12: **End for**

5.4. Experimental and Simulation Analysis

The targeted navigation problem for this simulation is described as follows: Everything that is required to effectively navigate a previously unknown environment with widely spread terrain, including long halls and blind edges, is an MR that contains dispersed local range sensors. With investigation, the MR will generate an autonomous environmental system since at first it has little knowledge of its surroundings. The complexities of environments with volatile obstacles are greatly increased by their dynamic and unexpected character. We

implemented action a local plan that utilizes RL that translates visible perceptions into actions for secure exploration and obstacle avoidance in dynamic, unfamiliar environments. To enhance the fundamental RL techniques and enable mobility within complex scenarios, we use many common patterns. We utilize the MR model that is explained in Chapter 4.5 to perform autonomous navigation within unfamiliar environments.

The size of the environment map is $250 \times 250m^2$ for the simulation. The obstacles are dispersed arbitrarily around the surroundings, and the robot is not aware of their quantities, dimensions, or locations beforehand. On the terrain map, the MR's beginning location position (10, 10) is displayed as a red-colored triangle, while its target position (240, 240) is displayed as a green-colored circle. The MR job is to commence at the beginning location and select the optimal path to travel in order to get to the destination avoiding striking into any obstacles and return back from the goal position to the starting point avoiding striking into any obstacles. The complete set of hyperparameters and their corresponding descriptions that are utilized in the present research are shown in Table 5.2.

TABLE 5.2: LIST OF HYPERPARAMETERS

Configuration of System	Description/Version
Version of Python	2.9.1
Version of MatLab	2023a
Version of Keras	2.15.0
Version of NumPy	1.24.3
Version of TensorFlow	2.9.1
Version of Matplotlib	3.7.0
RAM	8GB
Processor	Intel(R) core (TM) i3-4005U

Given a partial awareness of the environment and a target destination or group of destinations, intelligent navigation refers to the robot's ability to arrive at decisions according

to cognitive abilities and information collected from sensors with the purpose of reaching its target destinations as rapidly and efficiently as possible. A simulated MR that performs the selected behaviors in the environment of simulation at each episode's phase. The NN learning procedure can also be extended to a database of available examples for training, which allows the gathering of workable solutions for similar scenarios that have not been previously seen in experiments. Four totally different navigation operations utilizing similar weights were displayed in Fig. 5.2.

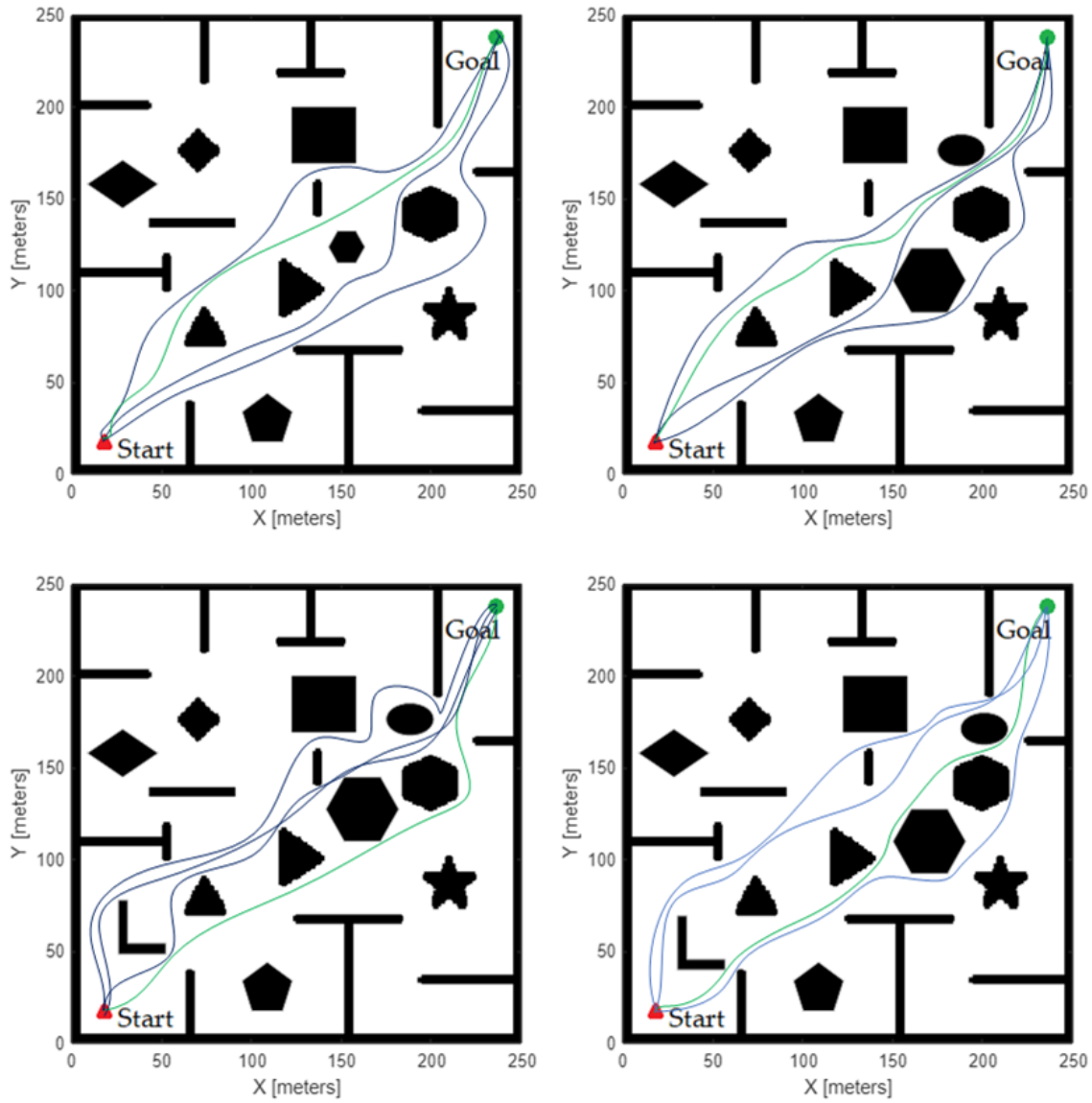

Fig. 5.2: An illustration of NNQL-based autonomous navigation outcomes across various environments.

From one starting point to a specific goal location, the robot attempted to attain with different path in Fig. 5.2. In each of the four scenarios, the robot managed to reach the desired location without running into any obstacles and maintained a secure distance from them. Because the robot did not have a thorough understanding of the environment's map, it is noticed that the pathways it selected may not have been the best ones, but even in such complex surroundings, they are perfectly acceptable and fulfill our expectations. The blue-colored paths are considered as less optimal paths w. r. t. green-colored path because the green-colored path is the shortest and consumes less energy thus it is the optimal path when compared with blue-colored paths. Simulation experiments are carried out in order to evaluate the suggested strategy. The robot is dispatched to complete a mission that involves navigating a new and strange area. The main objective of the simulation aims to generate an optimal Q-value. Table 5.3 shows the list of Q-learning parameters and their values used for the experimental analysis.

TABLE 5.3: Q-LEARNING PARAMETERS

List of Parameters	Values
Gamma	0.99
Epsilon	1.0
Epsilon decay	0.995
Epsilon min	0.01
Learning rate	0.001

Unlike the trials, the robot does not immediately know how many, how big, or where the obstacles are. It also becomes apparent that, despite the fact that the robot was not completely acquainted with its surroundings, the paths it chose might not have proven to be the most favorable, but they were nonetheless entirely suitable and satisfied standards references. The algorithm used to navigate MR using NNQL is given in Algorithm 5.5.

The robot is able to embark on new challenges and avoid being stuck in an endless cycle once it becomes used to its surroundings, thanks to learning in a variety of contexts. As we observed, the robot experimented with a variety of activities, and the rewards assessed its choice of action. This information will enable the robot to make better decisions about actions going forward. Once the learning phase is over, the weights will be proficient and ready to go straight into the robot navigation procedure. Given that the proposed approach uses trial and error, it makes sense that many episodes will not succeed as a result of a collision. An episode where the robot successfully reaches the goal location is considered an effective learning episode. Fig. 5.3 displays the number of effective learning episodes for each 50 episodes. Less than 50 of the robots in the initial 100 episodes achieved their goals, but more than 70 of the final 50 episodes showed the robot reaching its destination. This rapidly growing trend proved that the robot's intelligence was enhanced by the suggested NNQL algorithm.

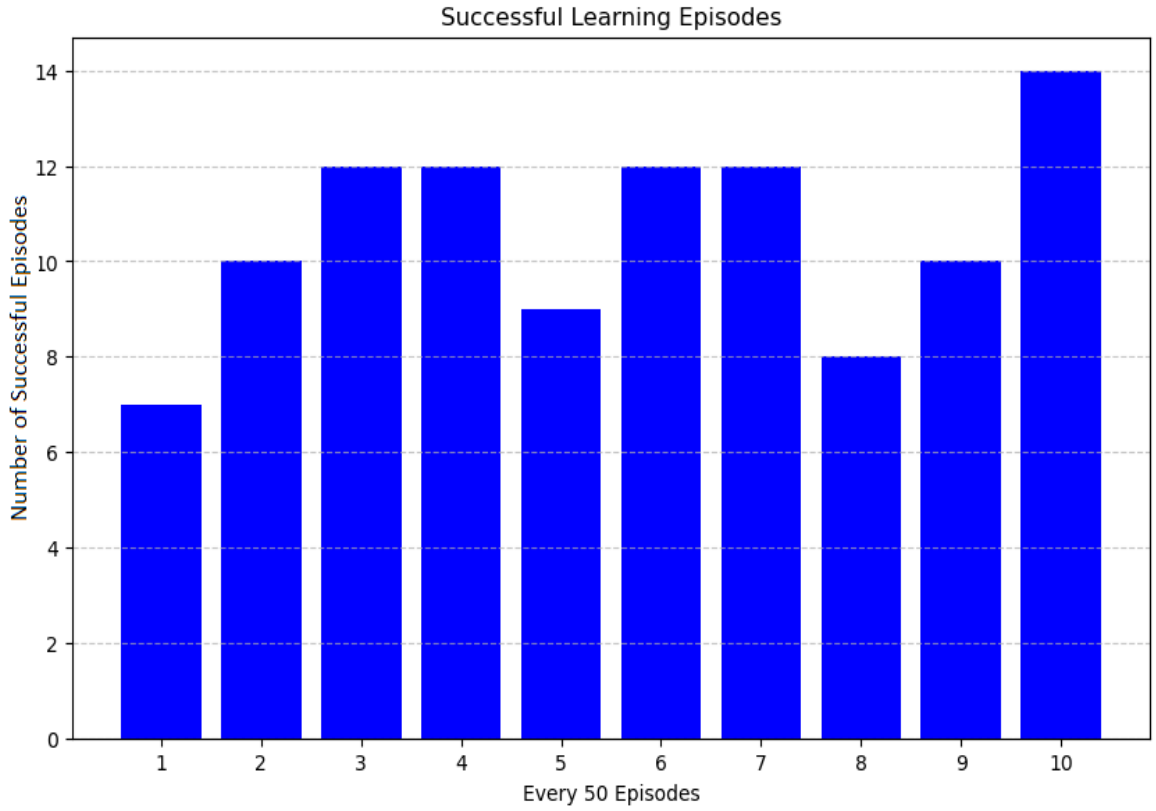

Fig. 5.3: An illustration of successful learning episodes during simulation

This information will enable the robot to make better decisions about actions going forward. In other words, the robot has progressively picked up skills to deal with its surroundings. The robot has picked up guidelines for how to act around obstacles. The robot basically just adopts the FFNN and selects the optimal course of action to select the optimal path to reach the desired goal location. Sometimes robots choose different paths to reach the target location. Since these aren't among the most efficient paths, we're looking at how they vary from each other. Then we select the shortest path. The proposed system creates a state collection with flexible variables whereby, after navigating a particular scenario properly, it can be useful in another one, regardless of the location where each obstacle and objective is located. Defining the reward function, or how the environment responds to the activities of the behavioral agent, constitutes additional criteria for decision-making. The simulation employs the epsilon-greedy algorithm to analyze autonomous navigation behavior. The simulation-time reward and epsilon variation curves are shown in Fig. 5.4.

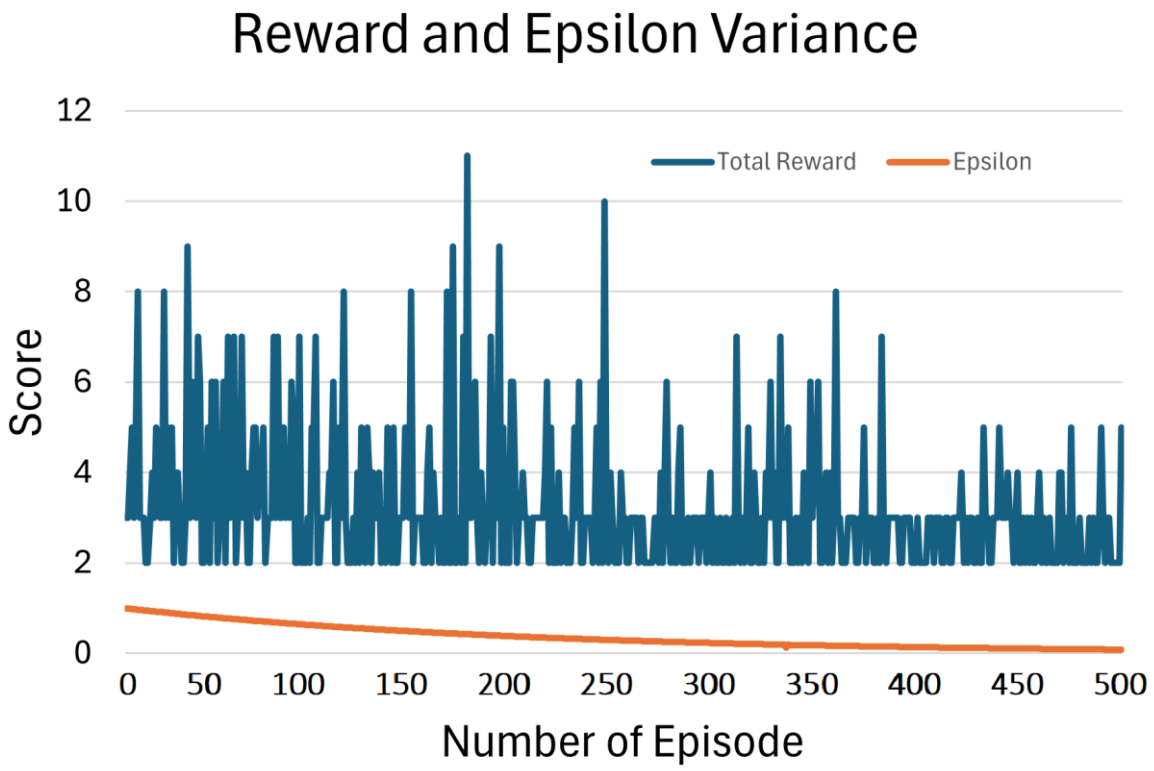

Fig. 5.4: Reward and Epsilon variance scores for each episode in the simulation.

5.5. Discussion

According to the above trial findings, NNQL has shown itself to be capable of performing autonomous navigation activities well and consistently despite the need for explicit robot behavior programming. The robot is capable of autonomous navigation with nearly 100% accuracy after just 500 episodes. Two well-known model-free algorithms were compared, and NNQL appeared better. It is reasonable to conclude that NNQL has effectively equipped an MR using robust self-learning capabilities for duties involving autonomous navigation. One drawback of probabilistic strategies is that the robot might not always pick the best course, and it can go lost for a time before discovering a route out. The robot needs to explore its surroundings for the purpose of learning a limited portion of its surroundings because it lacks a comprehensive perspective of it. Using simultaneous localization and mapping (SLAM) methods represents a possible remedy. An extensive portion of traditional robotics concentrated on sensor interpreting, thinking, and optimum control provided representations regarding the robot and surroundings. Although this method works well for a lot of commercial uses, it is not as ambitious as using robots as a testing ground for AI.

The robot self-learning technique under specific input was introduced within this chapter for those lacking any prior expertise. We investigated the challenge of MR navigation through the integration of NNs and RL. Q-learning is used to improve an MR's capacity for self-learning using trial-and-error encounters via a new environment. In order to preserve and educate substantial Q-values and to extend the acquired effectiveness to substantial state and action spaces, we created unique reward expressions and implemented NN architecture. The findings from the experiment demonstrate the reliability and effectiveness of the presented approach. By securely completing navigation objectives in an unpredictable and unpredictable setting, the robot develops into a highly intelligent system capable of powerful self-learning and adaptation.

Chapter 6

A Comparative Discussion: Reinforcement Learning Vs Particle Swarm Optimization and Reinforcement Learning Vs Thermal Navigation

6.1. Introduction

Only a few extremely stochastic prospective, and dynamic environments have been found that make traditional strategies for sophisticated system formation unsuitable [117]. These innovative requirements are not sufficiently addressed by the technology and approaches that are currently in use. Therefore, by giving autonomous devices the ability to adapt and make decisions as well as acquire knowledge, we can provide these systems with the knowledge and skills including RL-based navigation technique, which is necessary to identify and resolve these kinds of problems. Because of the frequently huge range of variables and crowded indicators of performance, designing efficient robotic systems is a classic instance of costly optimization in times of uncertainty. A control technique, as used for the framework of MRs, is a method or mechanism that interprets sensor input and produces actuator instructions to direct the motion of the robot along an intended action or objective. The key component within the MR that belongs to the control mechanism is the control techniques including RL, Particle Swarm Optimization (PSO), which enables more precise and efficient navigation. The control module continually modifies the actuator orders to preserve an intended trajectory or complete a specified job based on input given by the MR sensor. The problem of determining how much torque and force the robot's controllers must produce in order to allow MR to proceed in the

right direction, maintain the desired path, and, typically, finish an activity with the appropriate performance criteria is addressed by MR control techniques [118].

It cannot be an easy process to create high-performing robotic processors by hand for various reasons. First important all, even the most basic contemporary robots contain a lot of actuators and sensors, which indicates that there are a lot of control factors to be optimized. Furthermore, the presence of discontinuity and irregularities within real-world systems can make it challenging to use widely recognized linear control strategies. Controlling challenges in MRs (dynamic and static) is more difficult than usual because of inertial factors, linked reaction behaviors, and gravitational influences. The control module continually modifies the actuator orders to preserve an intended trajectory or complete a specified job based on input given by the MR sensor. The use and intended functionality of the MR will determine the exact kind of control technique that is employed. In an MR, a navigational or control technique's main objective is to provide the robots the ability to move around and carry out tasks on their own in unpredictable and complex environments.

Mobile robots are capable of using a variety of control techniques, including thermal, PSO, and RL controller-based navigation approaches. In this chapter, we are going to study a comparative analysis between various navigational approaches including, RL, PSO, and thermal or infrared assistive navigation. RL approach has been already discussed and analyzed in previous chapters. Thus we are going to discuss more details about PSO and thermal assistive navigation approach before starting a comparative study in this chapter.

6.2. Particle Swarm Optimization

When designing online controllers, a variety of evaluating methods can be used. When designing online controllers, a variety of evaluating methods can be used. For online controller structure, a variety of evaluating methods are available. PSO is an approach applied for the

optimization of non-linear continuous functions. PSO can find ideal or near-optimum techniques and is easy to utilize [119]. The information transmission between every particle in the swarm is a particularly important component of the PSO approach. Each of the particles will be examined with the goal of finding the optimal answer within its searching region. The searching process is guided by both collective and individual data across the entire swarm of particles. One particular kind of iterative algorithm is known as the PSO approach. In every subsequent iteration, every particle attempts to get closer to the optimal result. At the conclusion of every iteration, an observation addressing the swarm's best selection from the previous possibilities is made. In order to find the shortest route between every point of a predetermined environment that is accessible within the observation region, the PSO provides an effective approach for planning the path.

By PSO, every particle finds a response for the issue by integrating its specific past searches together with the experiences of the other particles. After going through several iterations, one might finally determine whether the technique is optimum or inadequate for a given problem [120]. For initial PSO, particles inside the resulting zone are first generated randomly. Equations (6.1) and (6.2) represent how the locations and velocities developed under the PSO approach [121].

$$v_i^d = v_i^d + \mathrm{c}_1\, r_1^d\left(\, pbest_i^d - y_i^d\right) + \mathrm{c}_2\, r_2^d\left(gbest^d - y_i^d\right) \tag{6.1}$$

$$y_i^d = y_i^d + \, v_i^d \tag{6.2}$$

Where the velocity of ith particle is given by $V_i = (v_i^1, v_i^2, v_i^3, v_i^4, \dots\dots, v_i^D)$, the position of ith particle is given by $Y_i = (y_i^1, y_i^2, y_i^3, y_i^4, \dots\dots, y_i^D)$, the acceleration coefficient is c_1 and c_2, two values, r_1^d and r_2^d, were chosen arbitrarily from the interval $[0, 1]$, the place which is greatest recognized by the swarm is $gbest = (gbest^1, gbest^2, \dots\dots, gbest^d)$, a personal best

place recognized by ith particle is $pbest_i = (pbest_i^1, pbest_i^2, pbest_i^3, pbest_i^4, \dots\dots, pbest_i^D)$, D is the dimension of the problem associated with optimization, and $d \in \{1, 2, \dots\dots, D\}$.

6.3. Thermal Navigation

There are many methods for providing safe and efficient autonomous navigation for mobile robots in an unknown environment. The thermal navigation approach is one of the most important strategies for autonomous navigation of MRs that utilizes an infrared sensor and GPS technique. The infrared (IR) sensor represents a specific type of discrete sensor that has been emphasized as a way to conduct landmark or item recognition in an unfamiliar environment. This sensor, which uses the technique of optical triangulation concept, provides superior resolution with a faster reaction time, high accuracy, extended range, and obstacle-avoiding capabilities [122]. The orientation and proximity of an MR with respect to a chosen component, including a building or roadway edge, might be ascertained using the thermal navigation system. When compared to the precision provided by widely used terrestrial satellite systems, this technique allows for a substantial improvement in the reliability of establishing an MR location. Instead of generating more signals, the system takes advantage of the thermal energy that has been collected in the surrounding area [123]. A basic infrared matrix, a computer for processing data, and an electronic armband are the core components of the thermal navigation system. In this approach, the utilization of renewable energy for information collection and computation allows the mobile navigation system to function throughout extended periods of duration despite the needed charging of the batteries. The system's primary job is to determine if the vehicle has crossed across to wrong edge of the roadway or has entered the space between the traffic lanes.

This sensor's length measuring capability has the benefit of allowing it to calculate the length from the object's observed terrain, which is helpful for 3D extrapolation that mimics the

object's appearance. We can identify an adequate path for a robot to navigate, for instance, relative to building walls or road edges, through the use of matrix infrared sensors mounted on the exterior surface of an MR model. Finding out if the MR is headed closer or farther from the object of choice requires evaluating the image captured by the infrared sensor. Precise location determination, such as when a robot is down a route or a route, is made possible by the integration of GPS navigation with infrared sensors.

A thermal navigation system explains in further detail the way a thermal camera might signal a robot to the approaching end of a structure when it detects an edge within its walls [124]. The building's floor and walls are composed of two distinct materials with various colors and patterns. Because every material possesses unique transparency, the thermographic picture might show the varying temperatures of the floor and the exterior wall. The approach using the thermographic cameras measures the length L within an MR and an interior wall can be abbreviated as equation (6.3) [124], [125].

$$L = \frac{h(\frac{1}{tg(\theta_1 + \theta_2)} + x)}{1 + \frac{x}{tg(90 - \theta_1)}} \tag{6.3}$$

Where the sensor height is h, the thermographic camera reading is x, the field of view (FOV) angle for the infrared sensor is θ_2, and the degree of inclination of the sensors relative to the floor is θ_1.

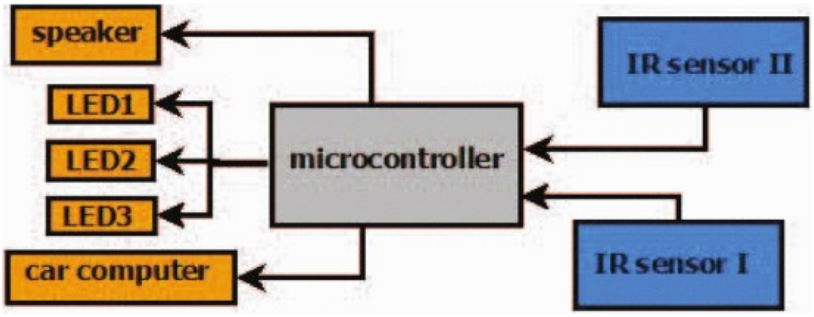

Fig. 6.1: The infrared thermal line sensing techniques schematic diagram [126].

One of the most important applications of the thermal navigation approach is in the autonomous driving assistance system. Fig. 6.1 displays an assembly architecture for the infrared thermal line tracking method. Advanced driver support system development, assessment, and deployment are presently in progress. The term Advanced Driver Assistance System (ADAS) refers to the grouping of security and driver support technologies. The automobile's bumper in the front has an infrared sensor that detects if the vehicle is getting closer to the border dividing the lanes of traffic or on the roadside. Having road monitoring is essential for autonomous vehicle operation. Despite this, the monitoring system has to deal with two primary limitations: limited processing capacity and real-time calculation. Detecting a road path or the border of the roadway and repositioning the automobile in case of an unexpected deviation from the intended track is one of its primary duties [127]. These days, this technology is often seen in automobiles; it works by analyzing images captured by cameras mounted on front bumpers or windshields. When lanes are well-defined on the roadway, this approach performs effectively.

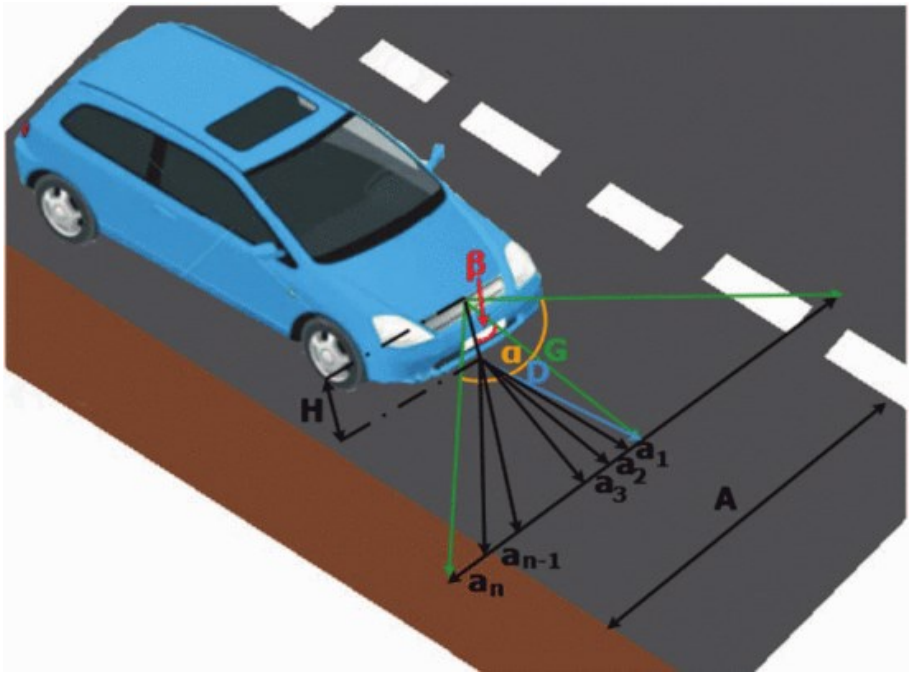

Fig. 6.2: The infrared sensor installation and the positioning of the automobile with relative to the road layer [126].

The thermal navigation approach uses the thermal emission factor, which varies depending on the color and substance that an item is constructed. During this approach, the temperature for the portrayed white roadway segments, the internal temperature for the sidewalk, and the temperatures of the darkest or black roadway are all measured in the situation under assessment. These items vary in both color and substance composition due to their varying emission parameters. This image is captured using a thermographic camera on the left and a video camera on the right while an automobile is passing by roadway lines and road edges as shown in Fig.6.3. The heat produced by the vehicle's engine and radiated off the road surface is used by the infrared sensors to identify the location of the vehicle when compared to the road.

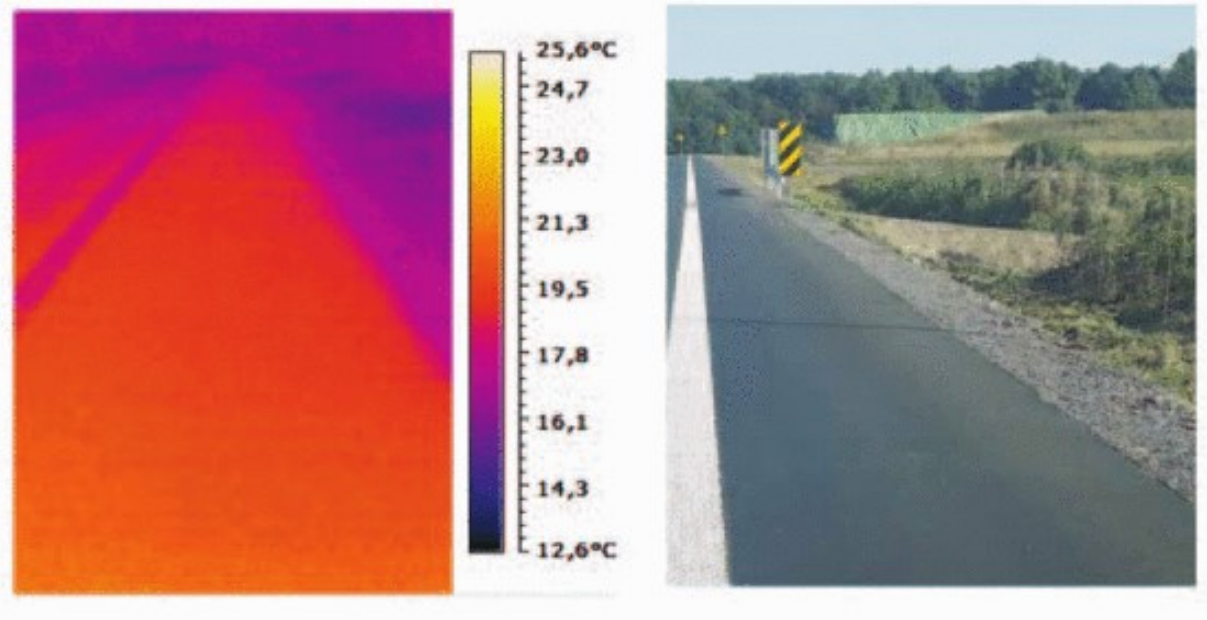

Fig. 6.3: Illustration of the thermographic image on the left taken by the thermographic camera and the right side image taken by normal video camera [126].

The basic idea of the infrared thermal lines tracking system is to identify instances that involve the car drifting from its intended lane and notify the driver about the issue. Two thermographic cameras are used as heat sensors with this framework. The infrared sensor exploits the phenomena of variable emissivity and temperatures of the substances that make up the roadway pavement and the sidewalk to calculate the location of the automobile with respect

to the roadway. As seen in Fig. 6.2, this approach can figure out if the automobile is on the right route using the thermographic images if we understand the location of the thermal sensor with respect to the floor (tilt angle β, mounted heights for the sensor H), and the dimension of the roadway lanes (width A). The floor width a_n for the hypothetical measurement point might be determined by applying equation (6.4) [123], which depends on the height H and the degree of inclination β.

$$a_n = Hcos(\beta)\left(tg\left(\frac{\alpha}{N}n\right) - tg\left(\frac{\alpha}{N}(n-1)\right)\right) \tag{6.4}$$

Where, the sensor's horizontal resolution is N, the number of the measurement point is n, and α is the viewing of the FOV angle.

Equation (6.5) [126] might be used to determine the length of the roadway A in regard to height H and angle of inclination β.

$$A = 2 \times \sum_{n=1}^{n=\frac{N}{2}} \left(Hcos(\beta)\left(tg\left(\frac{\alpha}{N}n\right) - tg\left(\frac{\alpha}{N}(n-1)\right)\right)\right) \tag{6.5}$$

The observed variance in temperature between the roadside and the road top layer is typically caused by variations in the emissivity for a particular surface or variations in the temperature of the air, which contribute to distinct surfaces becoming warmer or cooler at distinct times owing to various thermal resistances and capacitances. The phenomena of heat released by the automobile engine reflecting off the roadway's surface and the sidewalk across resulting in the automobile traveling based on thermal navigation can be employed in this approach to identify the border of the roadway. In automobiles, choices are frequently made based on various sources of data to allow the system to function perfectly and consistently. The automobile may thus respond without mistake if the information is known from many sources.

Alongside other techniques, the program makes use of the reversed perspective shift to turn the visual through an aerial perspective and the graph split classification approach. The thermal reflecting coefficient, which varies according to various materials and colors, determines how much heating radiates off the road surfaces. It is feasible to maintain the vehicle in the proper lane and alert the driver in the case of an intended unexpected directional shift or an unexpected effort to move the vehicle from the side of the roadway to the sideline since the navigation system can identify road lanes as well as road edges. The working algorithm of the infrared thermal line detecting method is displayed in Fig. 6.4.

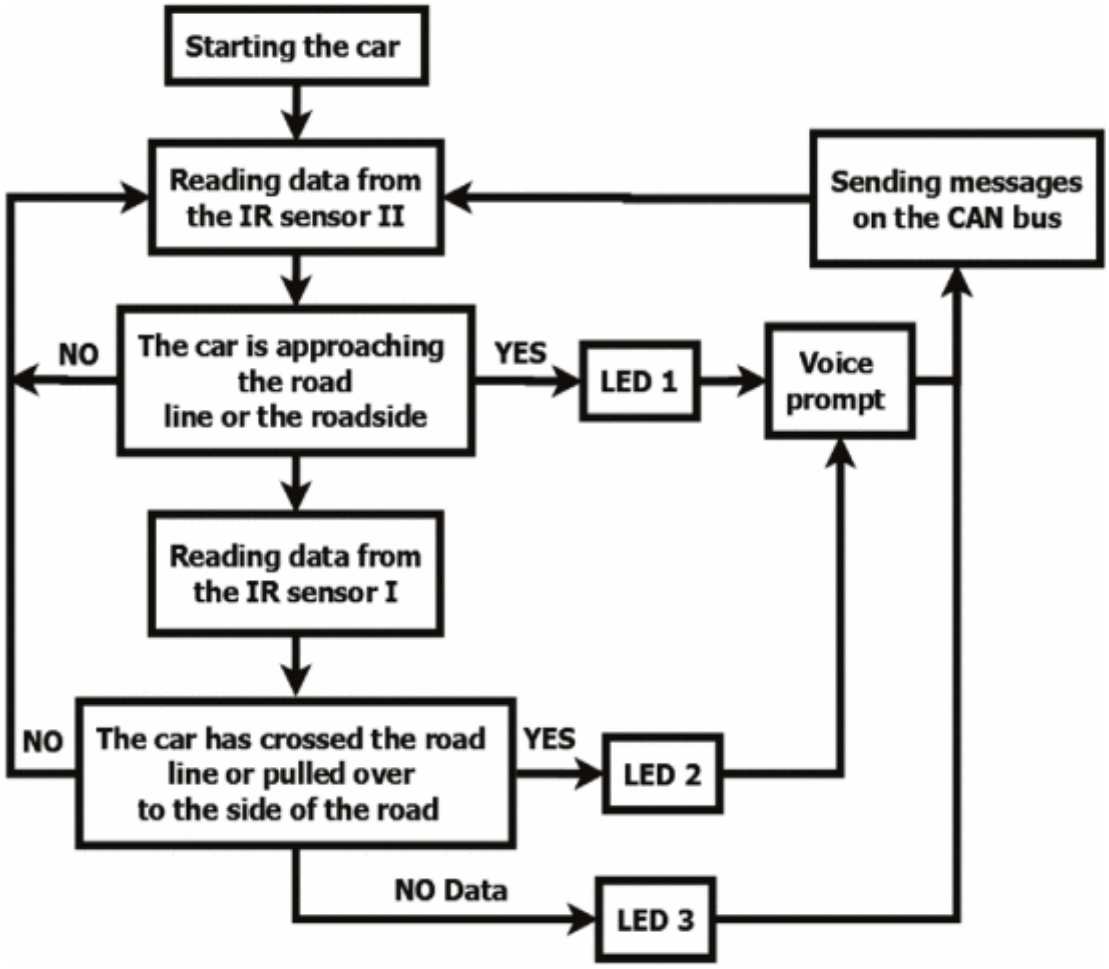

Fig. 6.4: The system's operational algorithm for detecting infrared thermal lines [126].

Another method that uses far fewer computing resources and is intended for tiny devices is to recognize a roadway path by detecting changes in the surface of the photo that is captured

by the camera [128]. The visual appearance and layout of the outer layer, at which the thermal energy produced by the engine of the automobile emanates, account for the perceived variance in temperature. The road and the pavement radiate the thermal radiation produced by the vehicle's engine. The infrared photographs show that additional heat is radiated from the road surfaces because it is better than the sidewalk. An example image of an automobile traversing a road lane might exist viewed in Fig. 6.5. The image captured from a thermographic camera on the left-hand side shows the thinner region separating the automobile's wheels. This is caused by heat from the engine of the automobile radiating off the white roadway lines.

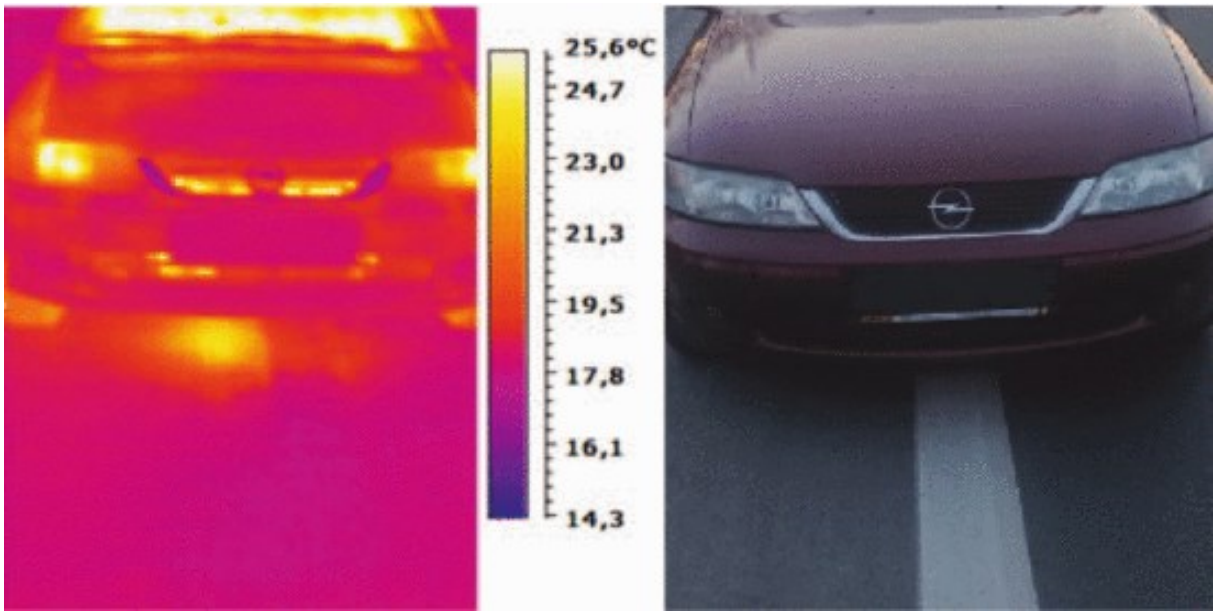

Fig. 6.5: An image captured by a thermographic camera (on the left side) and a video recording camera (on the right side) [126].

By using infrared sensors to identify heat emissions from individuals and objects, these devices provide vital perception of space using non-visual sensory methods including vibration or auditory signals. Using infrared sensor technology to develop a thermal guide for the blind is a creative way to improve the mobility and freedom of those with visual impairments. The infrared sensors are very useful to support visually impaired people by determining accurate distance from other objects to avoid collision because, a visually challenged individual finds it

challenging to navigate inside environments due to extreme precision demands, poor or nonexistent GPS signal within houses, and other factors. An extensive range of commercial solutions are available that aid blind individuals in navigating. Thermal navigation strategy is one of the famous approaches nowadays among researchers to develop the best assistive system to provide aid to blind or visually impaired people. While infrared sensors might be employed to establish a person's location on the roadway, the accuracy is rather low. These sensors make it possible to measure the gap between a visually impaired individual and different objects or architecture to provide aid in their movement [129].

The data gathered using infrared sensors is processed by a microcontroller. It uses the sensor information to interpret them in order to calculate the obstacles' location and orientation. The schematic layout of the navigation system is presented in Fig. 6.6. There are two primary segments in the framework. The modules communicate with each other over wireless lower energies, a power-efficient interface. The sensor segment, which is worn on the upper arm of the blind individual, collects data from their environment, and the notice section, which is worn on the blind person's hand, transmits details to people.

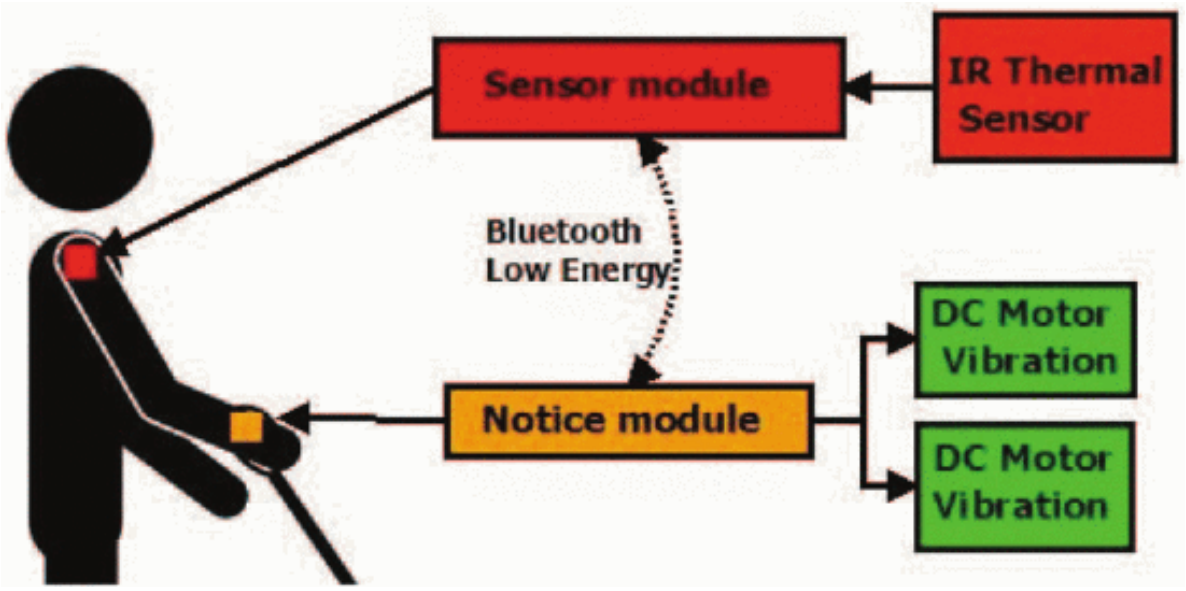

Fig. 6.6: The thermal navigation system layout for those with vision difficulties [124].

The separation between the blind person and a chosen item is computed using the thermal sensor data, which is sent to the sensor unit. A blind person can be placed in relation to an item, such as an obstacle, by measuring the variations in emissivity and temperature between both of them. Notifications are created according to the information that is gathered through the Sensor unit and then delivered to the Notice unit using the wireless Bluetooth connector. Vibration sensors are worn on a visually impaired individual's hand and provide the proper vibrating patterns that the notice unit translates from the received signals. As shown in Fig. 6.7, the employment of a narrow component, such as a tape composed of a substance possessing a large thermal reflection factor, mounted on an interior wall in a structure at a specific height h above the floor is the approach suggested by [124]. Due to the tape's large thermal reflecting factor, any heat produced by an individual standing close to the building's surface can be reflected off of it and then measured using a thermal sensor that is attached to the person's wrist.

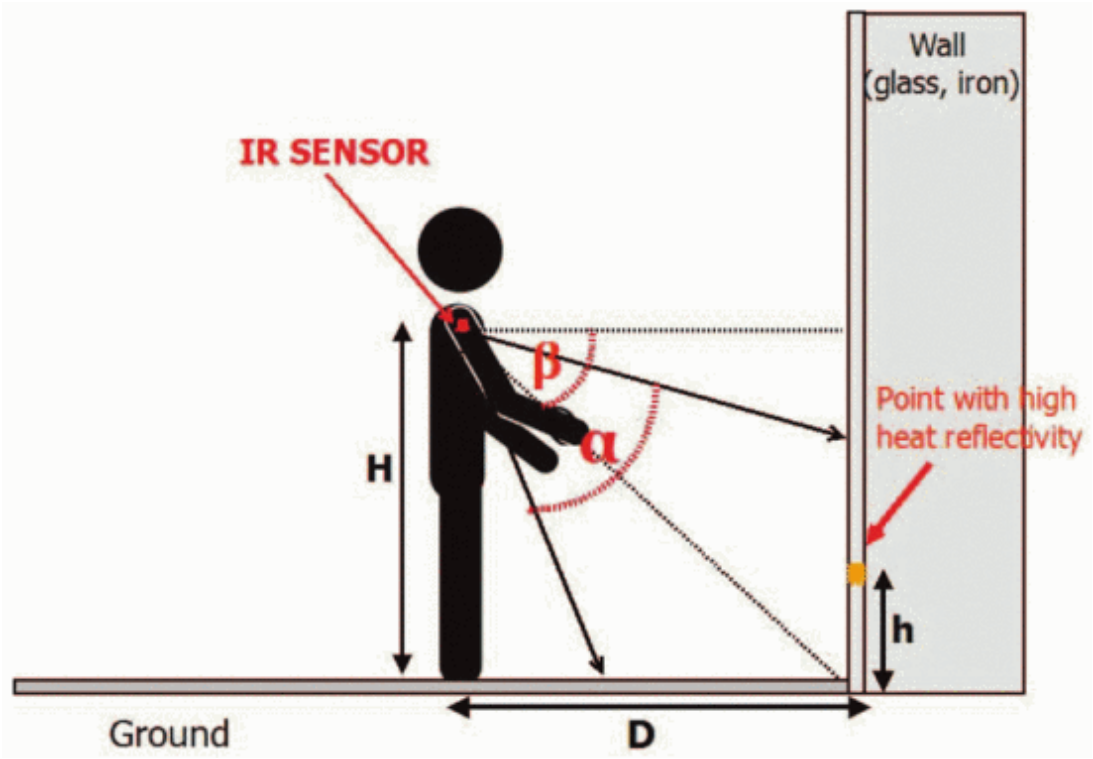

Fig. 6.7: The technique for measuring the distance using the component with a large coefficient of reflections [124].

We can determine the blind person's location relative to the wall by applying equation (6.6) [124].

$$D = \frac{tg\left(\pi - \frac{1}{2}\alpha - \beta\right) HZ - Zh - h}{tg\left(\beta - \frac{1}{2}\alpha\right) + Z} \tag{6.6}$$

Where the tilt angle is β for the infrared sensor, the static angle for the FOV sensor is α, the height of the sensor from the floor is H, h is the height of point with high reflectivity, and the quantity of pixels above and beneath the plane displayed in the thermographic picture, respectively, as determined by reading the value using the picture produced by a thermographic camera is Z. This value is equal to the ratio of pixels C to pixels B.

Thus, infrared-based thermal navigation systems provide a promising technological advancement for enhancing the freedom and standard of living of those with visual impairments. With continued advancements, technology has the ability to transform into a navigational and safety essential.

6.4. Comparative Discussion

There may be an unanticipated decrease in efficiency when implementing a specified controller on actual robots for a variety of reasons, including manufacturing flaws, environmental changes, or inaccurate modeling. As we discussed above about the metaheuristic (PSO) approach and a modern technique such as thermal navigation for the navigation of robots in a complex environment scenario. These two techniques are also well known among robotics researchers for finding an optimal route within complex and unknown environments for the navigation of MR when they are compared with DL-based robot navigation. The thermal navigation approach and PSO technique are well known for obstacle avoidance tasks during the

implementation of MR. Now, we further discuss the comparative analysis based on major differences between these two techniques and RL based navigation approach.

6.4.1. Reinforcement Learning Vs PSO

In place of human-guided approach ML approaches such as RL offer a solution to the aforementioned problems. Specifically, evaluation techniques will automatically generate autonomous controls in enormous exploration spaces, handling discontinuity and irregularities. and discovering novel solutions that human creators might not have considered. Moreover, an entirely on-board implementation of the learning procedure allows for automatic adaptability to the surroundings and supporting electronics [130]. Although the primary disadvantage of utilizing an online, evaluating approach is the significant time investment required to describe the efficiency of potential controller implementations. Assessment of performance noises can originate from multiple sources of unpredictability, including production tolerances, sensor noise, and loose collaboration in multi-robot environments. Two categories of evaluation techniques that have previously been used in robotic controller development are the subject of this section: A population-centered metaheuristic called Particle Swarm Optimization with a technique and an RL approach called Q-learning.

PSO has been employed to execute adaptation procedures for a distributed method for multi-robot setups. This approach optimizes the evaluation procedure and decreases the needed analysis time, but it necessitates the examination of several different options. However, Q-learning has the ability to iteratively improve one policy in particular, perhaps resulting in a shorter assessment time [131]. In contrast to the RL-based technique, which has a higher rate of convergence and greater efficiency w. r. t. PSO algorithm. PSO has relied on a population-centered optimization technique While the RL-based navigational methodology relies on previous experiences. Whereas the PSO algorithms perform better when handling jobs requiring

multi-objective optimization, the RL approach performs better when dealing with single-objective problems. When PSO is compared with Q-learning, which operates with discontinuous actions and states and hence necessitates the separation of proximity sensor data and wheel velocity results, the PSO approach is used to distinguish the influence of differentiation from learning. Within the fields of optimization and ML, RL and PSO represent two different methodologies, which are compared based on their aspects in Table 6.1.

TABLE 6.1: COMPARISON BETWEEN REINFORCEMENT LEARNING AND PSO

Sr. No.	Reinforcement Learning	PSO
01.	Generally required huge data and careful tunning process for implementation.	Easy to implement and does not require huge data.
02.	It can be slow in operations related to complex tasks.	It is very fast for most of the optimization problems.
03.	Handeled by the exploration techniques such as epsilon-greedy and Q- learning algorithm.	Generally handled by the parameters of particle dynamics.
04.	It solves sequential decision-making tasks.	It solves generally fixed optimization tasks.
05.	Learning by feedback and interactions.	Optimization by swarm intelligence.
06.	It is generally applied for tasks related to robotics, healthcare, autonomous systems, and gameplay.	It is generally applied for tasks related to ANN training, function optimization, and engineering design.

6.4.2. Reinforcement Learning Vs Thermal Navigation

Thermal navigation is also a practical and simple way to utilize because of the advancements in infrared devices that have made it possible to use inexpensive infrared cameras. RL and Thermal Navigation represent two discrete ideas that are applied in various settings. The technique of employing infrared sensors to help robots travel through unfamiliar

environments by utilizing temperature gradients is known as thermal navigation [132]. On the other hand, RL is a kind of ML approach in which an agent gains decision-making skills by interacting with its surroundings with the objective of optimizing cumulative rewards [133]. Thermal navigation is dependent on physical and social responses to variations in temperature instead of a method of learning, whereas RL consists of trial and error methods, during which the agent evaluates alternative behaviors and learns via its results. For RL, when an agent acts in a situation, it obtains information in the way of rewards and modifies its strategy to enhance subsequent actions, while for thermal navigation, infrared sensors sense variations in temperature within their surroundings and navigate away from regions that have more temperatures. RL-based navigation and thermal navigation perform differently and have distinct goals, which are explained in Table 6.2:

TABLE 6.2: COMPARISON BETWEEN RL-BASED NAVIGATION AND THERMAL NAVIGATION

Sr. No.	Reinforcement Learning	Thermal Navigation
01.	The process of learning depends upon a trial-and-error technique with rewards feedback.	Based on infrared sensors that have inherent sensitivity to heat-related inputs.
02.	This method follows the interactions with the environment to take optimal policy.	This approach tracks movement by looking for temperature gradients.
03.	This method is flexible enough to work in dynamic, and complicated environments.	This technique is constrained to navigation based on temperature.
04.	This approach requires a lot of processing and sophisticated algorithms.	This approach is based on temperature change and is very easy to compute.
06.	This technique is in the nature of AI and ML.	This technique is inspired by biological behavior.

Chapter 7
Conclusion and Future Research Perspectives

This chapter will serve as the dissertation conclusion, providing a summary of the most important findings and offering some suggestions for further study in the future to explore and enhance navigation strategy.

7.1. Conclusion

The need for mobile robots to navigate intelligently within dynamic environments is essential, as these machines are becoming more and more prevalent throughout the modern world. Robust and dependable controllers that can handle the unpredictability of the outside world have been established as a result of the necessity for the robot to possess the capacity to respond to variations in external circumstances. A pre-programmed MR is unable to meet potential demands, particularly where human collaboration is required because it is improbable that humans can accurately anticipate every possible real-world scenario. Sampling all potential solutions is not realistic for continuous-state issues, therefore RL algorithms become the recommended approach in supervised learning within the DL field when there is no instructor. In this method, the MR uses the RL to obtain prior information regarding the environment using the sensor data for the purpose of performing its collision-free movement. The suggested method uses a very basic but efficient simulation platform that shows fast and efficient training processes.

This dissertation provides a better theoretical concept of MR, a historical background of MR, types of MR, and a robot learning process in the first chapter. Chapter 2 provides a broad range of information about AI, DL, ML, ANN, DNN, and RL. Chapter 3 gives a wide range of

surveys of related work. Chapter four describes MDP and POMDP broadly, and finally, this dissertation analyzes the role of RL in robot navigation within complex environments through simulation. Chapter six widely describes the concept of PSO and thermal navigation techniques. Also, the application of the thermal navigation approach by using infrared sensors for assistance to visually impaired people has been discussed in chapter six. Evaluating all potential solutions for continuous-state tasks is not realistic, therefore RL techniques serve as the recommended way to conduct supervised learning throughout the DL field when there is no trainer present. The results showed an apparent enhancement in reliability and safety when navigating in complex environments, but at an additional cost of enhanced energy requirements because of a longer exploration period. Simulations proved the efficacy of our whole learning strategy as there is a high probability that the robot will be allowed to travel autonomously in an unknown environment.

This dissertation focuses on enabling MRs with the potential to acquire knowledge intelligently and adapt to dynamic environments. In this thesis, we presented a robot learning approach that uses neural network-based Q-learning models for self-learning during intelligent navigation in a variety of scenarios. It needs to be feasible for an intelligent robot to acquire knowledge to render decisions despite the absence of demonstrations. Interaction with the task environment makes it possible. We presented a Q-learning approach in this thesis using neural networks based on these presumptions. We enhanced the quality of the Q-learning method by adding this representation because we have demonstrated that an ANN might serve as a suitable policy formulation.

The proposed robot learning technique uses model-free algorithms, which are better suited for practical use as it is hard to build a model of a new environment. The recommended approach in this dissertation uses a relatively simple but efficient simulation system that shows a fast and efficient learning rate in addition to the capability of sending the models to real robots.

The thermal navigation approach based on infrared cameras has been discussed broadly in this thesis. Thermal navigation techniques can provide assistance in the navigation of automobiles, mobile robots, and visually impaired movement. Thermal navigation technique is broadly described. Lastly, we have comparatively analyzed the relationship between RL and other navigation strategies including PSO and thermal navigation.

7.2. Future Research Perspectives

There are many future research possibilities still available for researchers to focus on RL technique regarding robotics application, which are described in detail as follows:

Researchers can focus on facilitating the transmission of RL policy using methods such as field variation and adjustment from simulation to real-world circumstances. Firstly, this dissertation has established a conceptual framework for mobile robot learning and produced trustworthy simulation results. A step advance in robot learning technique might be achieved by conducting trials on actual vehicles within outside environments. In subsequent research, the reward function might be updated to improve the method of learning and optimize the accuracy rate of entirely autonomous navigation within an unknown environment. Secondly, a variety of sensors that record ambient data are essential to an intelligent robot control framework. Robotics having sensors of their own need to be enabled to acquire methods of control using a wide range of sensory data, particularly audio and visual ones. The suggested method has shown encouraging results, and it can potentially be enhanced to boost both learning and performance rates. The capacity to acquire knowledge from sophisticated sensor data might not simply confer intelligence into the robot, but it also improves its communication with people.

Thirdly, for multi-robot frameworks, robot learning has allowed for further enhancement. As a result of a growing quantity of laborious tasks being performed by a team of robots, it will become vital to investigate robot learning approaches for the purpose of anticipating behavior

and improving interactions among other robots. Using machine vision and combination of sensors coupled with RL to enhance contextual awareness and decision-making. The sector is projected to experience notable progress in coming years with the incorporation of diverse AI technologies including DL and DRL, higher durability and security, and better communication between robotics and their surroundings. Finally, using a convolutional neural network (CNN), the deep reinforcement learning approach has been used recently for robotics navigation. In the near future research can be focused on using self-supervised methods to lessen dependency on data labeled by having robots develop independent data to train using explorations. DRL study is expected to provide significant advancements in robotics, since CNN, a member of the DL family, has demonstrated its efficacy in recognizing images and machine vision.

Further developments in AI and sensor technologies have the potential to improve thermal navigation systems' reliability and effectiveness. More thorough environmental modeling and constraint recognition might be achieved by utilizing ML methods and combining extra sensors (such as LiDAR and ultrasonic ones).

List of Figures

List of Tables

List of Algorithms

List of Published Research Articles by Author of this Thesis in Scientific Journals and Conference Proceedings

• JCR listed articles

1. R. Raj, and A. Kos, "**An improved human activity recognition technique based on convolutional neural network**," *Scientific Reports*, vol. 13, pp. 22581, Dec. 2023. https://doi.org/10.1038/s41598-023-49739-1 (Published), (Q1-Scopus), [Ministry Point-140].
2. R. Raj, and A. Kos, "**Intelligent Mobile Robot Navigation in Unknown and Complex Environment Using Reinforcement Learning Technique**," *Scientific Reports*, (Submitted on March 16, 2024), (Under revision since May 21, 2024), (Q1-Scopus), [Ministry Point-140].
3. R. Raj, and A. Kos, "**An Effective Estimation of Human Pose by Digital Image Processing for the Optimization of Human-Robot Interactions**," *Optics Express*, (Submitted on May 8, 2024), (Under review since May 9, 2024), (Q1-Scopus), [Ministry Point-140].
4. R. Raj and A. Kos, "**A Comprehensive Study of Mobile Robot: History, Developments, Applications, and Future Research Perspectives**," *Applied Sciences*, vol. 12, no. 14, p. 6951, Jul. 2022. https://doi.org/10.3390/app12146951, (Published), (Q1-Scopus), [Ministry Point-100].
5. R. Raj and A. Kos, "**An Optimized Energy and Time Constraints-Based Path Planning for the Navigation of Mobile Robots Using an Intelligent Particle Swarm Optimization Technique**," *Applied Sciences*, vol. 13, no. 17, p. 9667, Aug. 2023. https://doi.org/10.3390/app13179667, (Published), (Q1-Scopus), [Ministry Point-100].
6. R. Raj and A. Kos, " **Discussion on different controllers used for the navigation of mobile robot**," *International Journal of Electronics and Telecommunications*, vol. 70, no. 1, pp. 229-239, March 2024. https://doi.org/10.24425/ijet.2024.149535, (Published), (Q3-Scopus), [Ministry Point-70].
7. R. Raj, and A. Kos, "**Artificial Intelligence: Evolution, Developments, Applications, and Future Scope**," *Przegląd Elektrotechniczny*, vol. 2023, no. 2, pp. 1–13, Feb. 2023. https://doi.org/10.15199/48.2023.02.01, (Published), (Q4-Scopus), [Ministry Point-70].
8. R. Raj, and A. Kos, "**Study and Analysis of Discrete Event-Driven Autonomous System with a Case Study for a Robotics Task**," *Przegląd Elektrotechniczny*, vol. 2023, no. 9, pp. 50-56, Feb. 2023. https://doi.org/10.15199/48.2023.02.01,(Published), (Q4-Scopus), [Ministry Point-70].
9. R. Raj, and A. Kos, "**A Novel Method of Islanding Detection in a Distributed Power Generation System Integrated with Photovoltaic-Array**," *Przegląd Elektrotechniczny*, vol. 2022, no. 7, pp. 88–94, July 2022. https://doi.org/10.15199/48.2022.07.15, (Published), (Q4-Scopus), [Ministry Point-70].

• Non- JCR listed articles

10. R. Raj and A. Kos, "**Dynamic Obstacle Avoidance Technique for Mobile Robot Navigation Using Deep Reinforcement Learning**," *International Journal of Emerging Trends in Engineering Research*, vol. 11, no. 9, pp. 307–314, Sep. 2023. https://doi.org/10.30534/ijeter/2023/031192023, (Published), [Ministry Point-20].

11. R. Raj, and A. Kos, "**Designing and Analysis of an Architecture of Motion Control for a Mobile Robot Using Simulation**," *International Journal of Advanced Trends in Computer Science and Engineering*, vol. 12, no. 4, pp. 172–177, Aug. 2023. https://doi.org/10.30534/ijatcse/2023/051242023, (Published), [Ministry Point-20].

12. R. Raj, and A. Kos, " **Recognition of Hindi Character Using OCR-Technology: A Review**," *International Journal of Advanced Trends in Computer Science and Engineering*, vol. 12, no. 4, pp. 196–201, Aug. 2023. https://doi.org/10.30534/ijatcse/2023/071242023, (Published), [Ministry Point-20].

13. R. Jakhar, and R. Raj, " **A Review on the Impacts of Climate Change on the Power Systems***," International Journal of Innovative Technology and Exploring Engineering (IJITEE),* vol. 12, no. 6, pp. 56-61, May 2023. https://doi.org/10.35940/ijitee.F9572.0512623, (Published), [Ministry Point-20].

• List of Articles in Conference Proceedings

14. R. Raj and A. Kos, "**A Comprehensive Study of Optical Character Recognition**," *2022 29th International Conference on Mixed Design of Integrated Circuits and System (MIXDES), Wrocław, Poland*, 2022, pp. 151-154. https://doi.org/10.23919/MIXDES55591.2022.9837974, (Published), [Ministry Point-80].

15. R. Raj and A. Kos, "**Different Techniques for Human Activity Recognition**," *2022 29th International Conference on Mixed Design of Integrated Circuits and System (MIXDES), Wrocław, Poland*, 2022, pp. 171-176. https://doi.org/10.23919/MIXDES55591.2022.9838050, (Published), [Ministry Point-80].

16. R. Raj and A. Kos, "**Learning the Dynamics of Human Patterns for Autonomous Navigation**," *2024 IEEE 18th International Conference on Compatibility, Power Electronics and Power Engineering (CPE-POWERENG), Gdynia, Poland*, 2024. (Accepted for Conference Proceeding Between 24th to 26th June, 2024)

References

1. R. Raj and A. Kos, "A Comprehensive Study of Mobile Robot: History, Developments, Applications, and Future Research Perspectives," *Applied Sciences*, vol. 12, no. 14, p. 6951, July 2022. https://doi.org/10.3390/app12146951
2. "MAARS Mission: The Military Patrol Robot," by *Forces News*, July 5, 2016. Available online: https://www.forces.net/services/tri-service/maars-mission-military-patrol-robot (accessed on December 6, 2023).
3. "Mars Science Laboratory with Power Source and Extended Arm, Artist's Concept," by *NASA/JPL-Caltech*, February 21, 2007. Available online: https://mars.nasa.gov/resources/3454/mars-science-laboratory-with-power-source-and-extended-arm-artists-concept/ (accessed on December 6, 2023).
4. "Mine detector AUV launched," by *Press Trust of India*, July 28, 2023. Available online: https://www.ptinews.com/news/national/mine-detector-auv-launched/618679.html (accessed on December 6, 2023).
5. "ABB demonstrates concept of mobile laboratory robot for Hospital of the Future," by *ABB Press Release, Houston, Texas, USA*, October 9, 2019. Available online: https://new.abb.com/news/detail/37279/hospital-of-the-future (accessed on December 6, 2023).
6. The Editors of the Encyclopedia Britannica, "R.U.R.", by *Encyclopedia Britannica*, November 20, 2014. Available online: https://www.britannica.com/topic/RUR (accessed on December 6, 2023).
7. Robotics: A Brief History. Available online: https://cs.stanford.edu/people/eroberts/courses/soco/projects/1998-99/robotics/history.html (accessed on December 12, 2023).
8. J. Rekveld, "PHILIDOG, MISO AND MORE VEHICLES," Available online: http://www.joostrekveld.net/?p=321 (accessed on December 7, 2023).
9. Abby, "Shakey the Robot Explained: Everything You Need to Know," *History Computer Staff*, January 2022. Available online: https://history-computer.com/shakey-the-robot (accessed on December 7, 2023).
10. R. Raj, and A. Kos, "Artificial Intelligence: Evolution, Developments, Applications, and Future Scope," *PRZEGLĄD ELEKTROTECHNICZNY*, vol. 1, no. 2, pp. 1–13, February 2023. https://doi.org/10.15199/48.2023.02.01
11. "CES 2022: The humanoid robot, Ameca, revealed at CES show," *British Broadcasting Corporation (BBC)*. Available online: 710 https://www.bbc.co.uk/newsround/59909789 (accessed on December 7, 2023).

12. D. M. Considine and G. D. Considine, "Robot Technology Fundamentals," In Standard Handbook of *Industrial Automation—Chapman and Hall Advanced Industrial Technology Series*, D. M. Considine, G. D. Considine, Eds., Springer, Boston, MA, USA, 1986, pp. 262–320. https://doi.org/10.1007/978-1-4613-1963-4_17

13. F. Rubio, F. Valero, and C. Llopis-Albert, "A review of mobile robots: Concepts, methods, theoretical framework, and applications," *International Journal of Advanced Robotic Systems*, vol. 16, no. 2, p. 172988141983959, Mar. 2019. https://doi.org/10.1177/1729881419839596

14. M. Ceccarelli, "Notes for a History of Grasping Devices," In: *Grasping in Robotics, Mechanisms and Machine Science*, G. Carbone, eds, Springer, London, vol. 10, 2013. https://doi.org/10.1007/978-1-4471-4664-3_1

15. G. Campion, G. Bastin and B. Dandrea-Novel, "Structural properties and classification of kinematic and dynamic models of wheeled mobile robots," in *IEEE Transactions on Robotics and Automation*, vol. 12, no. 1, pp. 47-62, Feb. 1996. https://doi.org/10.1109/70.481750

16. Y. Gong et al., "Legged robots for object manipulation: A review," *Frontiers in Mechanical Engineering*, vol. 9, Apr. 2023. https://doi.org/10.3389/fmech.2023.1142421

17. Spot by Boston Dynamics. Available online: https://bostondynamics.com/products/spot (accessed on February 19, 2024).

18. C. Zong, Z. Ji, and H. Yu, "Dynamic stability analysis of a tracked mobile robot based on human–robot interaction," *Assembly Automation*, vol. 40, no. 1, pp. 143–154, Mar. 2019. https://doi.org/10.1108/AA-10-2018-0156

19. "XBOT ALL TERRAIN TRACKED MOBILE ROBOT," by *Horus Dynamics*. Available online: https://robot.horusdy.namics.com/robots/s/xbot-all-terrain-tracked-mobile-robot (accessed on February 24, 2024).

20. F. Ahmed, J. C. Mohanta, A. Keshari, and P. S. Yadav, "Recent Advances in Unmanned Aerial Vehicles: A Review," *Arabian Journal for Science and Engineering*, vol. 47, no. 7, pp. 7963–7984, Apr. 2022. https://doi.org/10.1007/s13369-022-06738-0

21. A. Dangwal, "Twice As Lethal As Bayraktars, US To Sell The 'Most Advanced UAV' Ever Operated By Ukraine To Battle Russia," by *The Eur Asian Times*, June 2, 2022. Available online: https://www.eurasiantimes.com/twice-as-lethal-as-bayraktars-us-to-sell-the-most-advanced-uav-ukraine/ (accessed on February 24, 2024).

22. Z. Cui, L. Li, Y. Wang, Z. Zhong, and J. Li, "Review of research and control technology of underwater bionic robots," *Intelligent Marine Technology and Systems*, vol. 1, no. 1, Oct. 2023. https://doi.org/10.1007/s44295-023-00010-3

23. "How AUVs Are Used To Spot Oil Plumes After A Spill," by *gCaptain*, February 12, 2022. Available online: https://gcaptain.com/how-auvs-spot-oil-plumes-after-a-spill/ (accessed on February 24, 2024).

24. M. B. Alatise, and G. P. Hancke, "A Review on Challenges of Autonomous Mobile Robot and Sensor Fusion Methods," in *IEEE Access*, vol. 8, pp. 39830-39846, 2020. https://doi.org/10.1109/ACCESS.2020.2975643

25. M. Knudson and K. Tumer, "Adaptive navigation for autonomous robots," *Robotics and Autonomous Systems*, vol. 59, no. 6, pp. 410–420, Jun. 2011. https://doi.org/10.1016/j.robot.2011.02.004

26. F. Gul, W. Rahiman, and S. S. Nazli Alhady, "A comprehensive study for robot navigation techniques," *Cogent Engineering*, vol. 6, no. 1, Jan. 2019. https://doi.org/10.1080/23311916.2019.1632046

27. S. Schaal and C. G. Atkeson, "Learning Control in Robotics," in *IEEE Robotics & Automation Magazine*, vol. 17, no. 2, pp. 20–29, June 2010. https://doi.org/10.1109/MRA.2010.936957

28. L. C. Cobo, K. Subramanian, C. L. Isbell, A. D. Lanterman, and A. L. Thomaz, "Abstraction from demonstration for efficient reinforcement learning in high-dimensional domains," *Artificial Intelligence*, vol. 216, pp. 103–128, Nov. 2014. https://doi.org/10.1016/j.artint.2014.07.003

29. B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robotics and Autonomous Systems*, vol. 57, no. 5, pp. 469–483, May 2009. https://doi.org/10.1016/j.robot.2008.10.024

30. M. Cummins and P. Newman, "Probabilistic Appearance Based Navigation and Loop Closing," *Proceedings 2007 IEEE International Conference on Robotics and Automation, Rome, Italy*, 2007, pp. 2042-2048. https://doi.org/10.1109/ROBOT.2007.363622

31. R. Raj and A. Kos, "Different Techniques for Human Activity Recognition," *2022 29th International Conference on Mixed Design of Integrated Circuits and System (MIXDES), Wrocław, Poland*, 2022, pp. 171-176. https://doi.org/10.23919/MIXDES55591.2022.9838050

32. R. Raj and A. Kos, "Learning the Dynamics of Human Patterns for Autonomous Navigation," *2024 IEEE 18th International Conference on Compatibility, Power Electronics and Power Engineering (CPE-POWERENG), Gdynia, Poland*, 2024. (Accepted for Conference Proceeding Between 24th to 26th June, 2024)

33. R. Raj, and A. Kos, "An improved human activity recognition technique based on convolutional neural network," *Scientific Reports*, vol. 13, pp. 22581, Dec. 2023. https://doi.org/10.1038/s41598-023-49739-1

34. S. Thrun, et al. Stanley: The Robot That Won the DARPA Grand Challenge. In: Buehler, M., Iagnemma, K., Singh, S. (eds), *The 2005 DARPA Grand Challenge. Springer Tracts in Advanced Robotics*, vol. 36. Springer, Berlin, Heidelberg, 2007. https://doi.org/10.1007/978-3-540-73429-1_1

35. C. Urmson, et al., Autonomous Driving in Urban Environments: Boss and the Urban Challenge. In: Buehler, M., Iagnemma, K., Singh, S. (eds), *The DARPA Urban Challenge. Springer Tracts in*

Advanced Robotics, vol 56. Springer, Berlin, Heidelberg, 2009. https://doi.org/10.1007/978-3-642-03991-1_1

36. M. Montemerlo, et al., Junior: The Stanford Entry in the Urban Challenge. In: Buehler, M., Iagnemma, K., Singh, S. (eds), *The DARPA Urban Challenge. Springer Tracts in Advanced Robotics*, vol. 56. Springer, Berlin, Heidelberg, 2009. https://doi.org/10.1007/978-3-642-03991-1_3

37. T. Brogårdh, "Present and future robot control development—An industrial perspective," *Annual Reviews in Control*, vol. 31, no. 1, pp. 69–79, Jan. 2007. https://doi.org/10.1016/j.arcontrol.2007.01.002

38. S. Panagou, W. P. Neumann, and F. Fruggiero, "A scoping review of human robot interaction research towards Industry 5.0 human-centric workplaces," *International Journal of Production Research*, vol. 62, no. 3, pp. 974–990, Feb. 2023. https://doi.org/10.1080/00207543.2023.2172473

39. A. M. Turing, "Computing Machinery and Intelligence," *Mind*, vol. LIX, Issue 236, Oct. 1950, pp. 433–460. https://doi.org/10.1093/mind/LIX.236.433

40. A. Przegalinska, "State of the art and future of artificial intelligence," *Policy Department for Economic, Scientific, and Quality of Life Policies (European Parliament)*, Feb. 2019, pp. PE631.051. https://doi.org/10.2861/613707

41. M. I. Jordan, T. M. Mitchell, "Machine Learning: Trends, Perspectives, and Prospectives," *Science*, vol. 349, issue 6245, pp. 255–260, July 2015. https://doi.org/10.1126/science.aaa8415

42. T. G. Dietterich, "Machine-learning Research," *AI Magazine*, Vol. 18, no. 4, 1997. https://doi.org/10.1609/aimag.v18i4.1324

43. A. Oppermann, "What is Deep Learning and How Does It Work?" *Built In*, December 12, 2023. https://builtin.com/machine-learning/deep-learning (accessed on March 7, 2024).

44. L. Alzubaidi et al., "Review of deep learning: concepts, CNN architectures, challenges, applications, future directions," *Journal of Big Data*, vol. 8, no. 1, Mar. 2021. https://doi.org/10.1186/s40537-021-00444-8

45. B. D. B. F. Filho, E. L. L. Cabral and A. J. Soares, "A new approach to artificial neural networks," in *IEEE Transactions on Neural Networks*, vol. 9, no. 6, pp. 1167–1179, Nov. 1998. https://doi.org/10.1109/72.728360

46. J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Networks*, vol. 61, pp. 85–117, Jan. 2015. https://doi.org/10.1016/j.neunet.2014.09.003

47. F. Stelzer, A. Röhm, R. Vicente, I. Fischer, and S. Yanchuk, "Deep neural networks using a single neuron: folded-in-time architecture using feedback-modulated delay loops," *Nature Communications*, vol. 12, no. 1, Aug. 2021. https://doi.org/10.1038/s41467-021-25427-4

48. R. S. Sutton and A. G. Barto, Reinforcement Learning, *second edition. MIT Press*. Available online: https://web.stanford.edu/class/psych209/Readings/SuttonBartoIPRLBook2ndEd.pdf (accessed on March 18, 2024).

49. G. Tesauro, "Practical issues in temporal difference learning," Machine Learning, vol. 8, no. 3–4, pp. 257–277, May 1992. https://doi.org/10.1007/BF00992697

50. S. D. Whitehead and L.-J. Lin, "Reinforcement learning of non-Markov decision processes," Artificial Intelligence, vol. 73, no. 1–2, pp. 271–306, Feb. 1995. https://doi.org/10.1016/0004-3702(94)00012-P

51. K. Sivamayil, E. Rajasekar, B. Aljafari, S. Nikolovski, S. Vairavasundaram, and I. Vairavasundaram, "A Systematic Study on Reinforcement Learning Based Applications," *Energies*, vol. 16, no. 3, p. 1512, Feb. 2023. https://doi.org/10.3390/en16031512

52. R. S. Sutton and A. G. Barto, Reinforcement Learning: An Introduction, vol. 1, *2nd ed. Cambridge, MA, USA: MIT Press*, 2017.

53. S. Sehad, "Neural Reinforcement Learning for Robot Navigation," *In: L.C. Jain, T. Fukuda (eds), Soft Computing for Intelligent Robotic Systems. Studies in Fuzziness and Soft Computing*, vol. 21. Physica, Heidelberg, 1998. https://doi.org/10.1007/978-3-7908-1882-6_7

54. B. Zuo, J. Chen, L. Wang, and Y. Wang, "A reinforcement learning based robotic navigation system," *2014 IEEE International Conference on Systems, Man, and Cybernetics (SMC), San Diego, CA, USA*, 2014, pp. 3452–3457. https://doi.org/10.1109/SMC.2014.6974463

55. Mu-Chun Su, De-Yuan Huang, Chien-Hsing Chou, and Chen-Chiung Hsieh, "A reinforcement-learning approach to robot navigation," *IEEE International Conference on Networking, Sensing, and Control, 2004, Taipei, Taiwan*, 2004, pp. 665–669, Vol. 1. https://doi.org/10.1109/ICNSC.2004.1297519

56. R. J. Alitappeh, N. Mahmoudi, M. R. Jafari and A. Foladi, "Autonomous Robot Navigation: Deep Learning Approaches for Line Following and Obstacle Avoidance," *2024 20th CSI International Symposium on Artificial Intelligence and Signal Processing (AISP), Babol, Iran, Islamic Republic of*, 2024, pp. 1-6, https://doi.org/10.1109/AISP61396.2024.10475313

57. R. Tsuruta and K. Morioka, "Autonomous Navigation of a Mobile Robot with a Monocular Camera Using Deep Reinforcement Learning and Semantic Image Segmentation," *2024 IEEE/SICE International Symposium on System Integration (SII), Ha Long, Vietnam*, 2024, pp. 1107-1112. https://doi.org/10.1109/SII58957.2024.10417188

58. N. Saxena, S. Gorantla and P. Jagtap, "Funnel-Based Reward Shaping for Signal Temporal Logic Tasks in Reinforcement Learning," in *IEEE Robotics and Automation Letters*, vol. 9, no. 2, pp. 1373-1379, Feb. 2024. https://doi.org/10.1109/LRA.2023.3341775

59. H. Jiang et al., "Learning Relation in Crowd Using Gated Graph Convolutional Networks for DRL-Based Robot Navigation," in *IEEE Transactions on Intelligent Transportation Systems*. https://doi.org/10.1109/TITS.2023.3343923

60. A. A. N. Kumaar and S. Kochuvila, "Mobile Service Robot Path Planning Using Deep Reinforcement Learning," in *IEEE Access*, vol. 11, pp. 100083-100096, 2023. https://doi.org/10.1109/ACCESS.2023.3311519

61. T. Tongloy, S. Chuwongin, K. Jaksukam, C. Chousangsuntorn and S. Boonsang, "Asynchronous deep reinforcement learning for the mobile robot navigation with supervised auxiliary tasks," *2017 2nd International Conference on Robotics and Automation Engineering (ICRAE), Shanghai, China*, 2017, pp. 68-72. https://doi.org/10.1109/ICRAE.2017.8291355

62. G. Williams et al., "Information theoretic MPC for model-based reinforcement learning," *2017 IEEE International Conference on Robotics and Automation (ICRA), Singapore*, 2017, pp. 1714-1721. https://doi.org/10.1109/ICRA.2017.7989202

63. B. Liu, L. Wang, and M. Liu, "Lifelong Federated Reinforcement Learning: A Learning Architecture for Navigation in Cloud Robotic Systems," in *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 4555-4562, Oct. 2019. https://doi.org/10.1109/LRA.2019.2931179

64. Cang Ye, N. H. C. Yung and Danwei Wang, "A fuzzy controller with supervised learning assisted reinforcement learning algorithm for obstacle avoidance," in *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 33, no. 1, pp. 17-27, Feb. 2003, https://doi.org/10.1109/TSMCB.2003.808179

65. B. Thananjeyan et al., "Recovery RL: Safe Reinforcement Learning With Learned Recovery Zones," *in IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 4915-4922, July 2021. https://doi.org/10.1109/LRA.2021.3070252

66. J. Jiang, D. Kong, K. Hou, X. Huang, H. Zhuang, and Z. Fang, "Neuro-Planner: A 3D Visual Navigation Method for MAV With Depth Camera Based on Neuromorphic Reinforcement Learning," in *IEEE Transactions on Vehicular Technology*, vol. 72, no. 10, pp. 12697-12712, Oct. 2023. https://doi.org/10.1109/TVT.2023.3278097

67. Q. Wu, K. Xu, J. Wang, M. Xu, X. Gong, and D. Manocha, "Reinforcement Learning-Based Visual Navigation With Information-Theoretic Regularization," in *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 731-738, April 2021. https://doi.org/10.1109/LRA.2020.3048668

68. J. Hu, H. Niu, J. Carrasco, B. Lennox, and F. Arvin, "Voronoi-Based Multi-Robot Autonomous Exploration in Unknown Environments via Deep Reinforcement Learning," in *IEEE Transactions on Vehicular Technology*, vol. 69, no. 12, pp. 14413-14423, Dec. 2020. https://doi.org/10.1109/TVT.2020.3034800

69. Z. Zhang, Y. -S. Ong, D. Wang, and B. Xue, "A Collaborative Multiagent Reinforcement Learning Method Based on Policy Gradient Potential," in *IEEE Transactions on Cybernetics*, vol. 51, no. 2, pp. 1015-1027, Feb. 2021. https://doi.org/10.1109/TCYB.2019.2932203

70. K. Wang, C. Mu, Z. Ni, and D. Liu, "Safe Reinforcement Learning and Adaptive Optimal Control With Applications to Obstacle Avoidance Problem," in *IEEE Transactions on Automation Science and Engineering*. https://doi.org/10.1109/TASE.2023.3299275

71. H. Yin, C. Wang, C. Yan, X. Xiang, B. Cai, and C. Wei, "Deep Reinforcement Learning with Multi-Critic TD3 for Decentralized Multi-Robot Path Planning," in *IEEE Transactions on Cognitive and Developmental Systems*. https://doi.org/10.1109/TCDS.2024.3368055

72. Z. Nan and H. Nam, "Visual-Based Deep Reinforcement Learning for Mobile Robot Obstacle Avoidance Navigation," *2023 14th International Conference on Information and Communication Technology Convergence (ICTC), Jeju Island, Republic of Korea*, 2023, pp. 440-445. https://doi.org/10.1109/ICTC58733.2023.10393297

73. E. A. Antonelo and B. Schrauwen, "On Learning Navigation Behaviors for Small Mobile Robots With Reservoir Computing Architectures," in *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, no. 4, pp. 763-780, April 2015. https://doi.org/10.1109/TNNLS.2014.2323247

74. X. Xue, Z. Li, D. Zhang, and Y. Yan, "A Deep Reinforcement Learning Method for Mobile Robot Collision Avoidance based on Double DQN," *2019 IEEE 28th International Symposium on Industrial Electronics (ISIE), Vancouver, BC, Canada*, 2019, pp. 2131-2136, https://doi.org/10.1109/ISIE.2019.8781522

75. K. Li, Y. Xu, J. Wang, and M. Q. -H. Meng, "SARL: Deep Reinforcement Learning based Human-Aware Navigation for Mobile Robot in Indoor Environments," *2019 IEEE International Conference on Robotics and Biomimetics (ROBIO), Dali, China*, 2019, pp. 688-694. https://doi.org/10.1109/ROBIO49542.2019.8961764

76. A. D. Pambudi, T. Agustinah, and R. Effendi, "Reinforcement Point and Fuzzy Input Design of Fuzzy Q-Learning for Mobile Robot Navigation System," *2019 International Conference of Artificial Intelligence and Information Technology (ICAIIT), Yogyakarta, Indonesia,* 2019, pp. 186-191. https://doi.org/10.1109/ICAIIT.2019.8834601

77. Q. Liu, Y. Li, and L. Liu, "A 3D Simulation Environment and Navigation Approach for Robot Navigation via Deep Reinforcement Learning in Dense Pedestrian Environment," *2020 IEEE 16th International Conference on Automation Science and Engineering (CASE), Hong Kong, China*, 2020, pp. 1514-1519. https://doi.org/10.1109/CASE48305.2020.9217023

78. Y. Cui et al., "Learning Hierarchical Graph-Based Policy for Goal-Reaching in Unknown Environments," in *IEEE Robotics and Automation Letters*. https://doi.org/10.1109/LRA.2024.3396059

79. L. Huang, R. Chai, K. Chen, J. Zhang, S. Chai and Y. Xia, "Navigating Partially Unknown Environments: A Weakly Supervised Learning Approach to Path Planning," in *IEEE Transactions on Intelligent Vehicles*. https://doi.org/10.1109/TIV.2024.3393068

80. C. Sun, X. Wu, Y. Wang, and C. Sun, "Attention-based Value Classification Reinforcement Learning for Collision-free Robot Navigation," in *IEEE Transactions on Intelligent Vehicles*. https://doi.org/10.1109/TIV.2024.3391007

81. M. Xu, X. Chen, Y. She, Y. Jin and J. Wang, "Time-Varying Weights in Multi-Reward Architecture for Deep Reinforcement Learning," in *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 8, no. 2, pp. 1865-1881, April 2024. https://doi.org/10.1109/TETCI.2024.3359039

82. W. Huang, Y. Zhou, X. He and C. Lv, "Goal-Guided Transformer-Enabled Reinforcement Learning for Efficient Autonomous Navigation," in *IEEE Transactions on Intelligent Transportation Systems*, vol. 25, no. 2, pp. 1832-1845, Feb. 2024. https://doi.org/10.1109/TITS.2023.3312453

83. W. Bai et al., "A Novel Adaptive Control Design for a Class of Nonstrict-Feedback Discrete-Time Systems via Reinforcement Learning," in *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 54, no. 2, pp. 1250-1262, Feb. 2024, https://doi.org/10.1109/TSMC.2023.3326466

84. S. Sukhbaatar, A. Szlam, and R. Fergus, "Learning multiagent communication with backpropagation," in *Proc. Neural Information Processing Systems*, 2016, pp. 2244–2252. https://doi.org/10.48550/arXiv.1605.07736

85. R.S. Sutton, and A.G. Barto, "Reinforcement learning: An introduction," *MIT press*, 1998. Available online: https://mitpress.mit.edu/9780262039246/reinforcement-learning/ (accessed on April 7, 2024).

86. M. L. Puterman, "Chapter 8 Markov decision processes," *Handbooks in Operations Research and Management Science, Elsevier*, Vol. 2, pp. 331-434, 1990, ISSN 0927-0507, ISBN 9780444874733. https://doi.org/10.1016/S0927-0507(05)80172-0

87. C. Xia, "Intelligent Mobile Robot Learning in Autonomous Navigation," Automatic Control Engineering, Ecole Centrale de Lille, 2015. Available online: https://theses.hal.science/tel-01298608/document (accessed on April 14, 2024).

88. M. L. Puterman, "Markov Decision Processes: Discrete Stochastic Dynamic Programming," *John Wiley & Sons, Inc., New York, NY, USA, 1st ed. 1994.*

89. R. D. Smallwood and E. J. Sondik, "The optimal control of partially observable Markov processes over a finite horizon," Operations Research, vol. 21, pp. 1071–1088, 1973. Available online: http://www.jstor.org/stable/168926 (Accessed April 14, 2024).

90. X. Xiang and S. Foo, "Recent Advances in Deep Reinforcement Learning Applications for Solving Partially Observable Markov Decision Processes (POMDP) Problems: Part 1—Fundamentals and Applications in Games, Robotics and Natural Language Processing," *Machine Learning Knowledge Extraction*, vol. 3, pp. 554-581, 2021. https://doi.org/10.3390/make3030029

91. L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, "Planning and acting in partially observable stochastic domains," *Artificial Intelligence*, vol. 101, Issues 1-2, Pp. 99-134, May 1998. https://doi.org/10.1016/S0004-3702(98)00023-X

92. D. P. Bertsekas, "Dynamic programming and optimal control", *Athena Scientific Belmont, Massachusetts*, vol. 1, 1996.

93. L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," Journal of Artificial Intelligence Research, vol. 4, issue 1, pp. 237–285, 1996.

94. M. van Otterlo, and M. Wiering, "Reinforcement Learning and Markov Decision Processes," In: M. Wiering, M. van Otterlo, (eds) *Reinforcement Learning. Adaptation, Learning, and Optimization, vol 12. Springer, Berlin, Heidelberg,* 2012. https://doi.org/10.1007/978-3-642-27645-3_1

95. E. M. Hahn, M. Perez, S. Schewe, F. Somenzi, A. Trivedi, and D. Wojtczak, "Model-Free Reinforcement Learning for Branching Markov Decision Processes," In: A. Silva, K. R. M. Leino, (eds) *Computer Aided Verification. CAV 2021. Lecture Notes in Computer Science (),* vol 12760. *Springer, Cham*, 2021. https://doi.org/10.1007/978-3-030-81688-9_30

96. B. Singh, R. Kumar, and V. P. Singh, "Reinforcement learning in robotic applications: a comprehensive survey," *Artificial Intelligence Review*, vol. 55, pp. 945-990, April 2021. https://doi.org/10.1007/s10462-021-09997-9

97. J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *The International Journal of Robotics Research*, 0278364913495721, 2013.

98. N. Hammami and K. K. Nguyen, "On-Policy vs. Off-Policy Deep Reinforcement Learning for Resource Allocation in Open Radio Access Network," *2022 IEEE Wireless Communications and Networking Conference (WCNC), Austin, TX, USA*, 2022, pp. 1461-1466. https://doi.org/10.1109/WCNC51071.2022.9771605

99. S. Mo, X. Pei, and C. Wu, "Safe Reinforcement Learning for Autonomous Vehicle Using Monte Carlo Tree Search," in *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 7, pp. 6766-6773, July 2022. https://doi.org/10.1109/TITS.2021.3061627

100. T. -b. Kwon, J. -h. Yang, J. -b. Song and W. Chung, "Efficiency Improvement in Monte Carlo Localization through Topological Information," *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems, Beijing, China*, 2006, pp. 424-429. https://doi.org/10.1109/IROS.2006.281962

101. G. Tesauro, "Practical issues in temporal difference learning," *Machine Learning*, vol. 8, pp. 257–277, 1992. https://doi.org/10.1007/BF00992697

102. C. J. C. H. Watkins, "Learning from delayed rewards," *Ph.D. thesis, University of Cambridge England*, 1989.

103. G. A. Rummery, and M. Niranjan, "On-line q-learning using connectionist systems," 1994. Available online: https://mi.eng.cam.ac.uk/reports/svr-ftp/auto-pdf/rummery_tr166.pdf (accessed on May 2, 2024).

104. C. J. C. H. Watkins, and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, pp. 279–292, 1992. https://doi.org/10.1007/BF00992698

105. S. Singh, T. Jaakkola, M. L. Littman, and C. Szepesvári, "Convergence Results for Single-Step On-Policy Reinforcement-Learning Algorithms," *Machine Learning*, vol. 38, pp. 287–308, 2000. https://doi.org/10.1023/A:1007678930559

106. "Robotino," by The Festo Didactic. Available online: https://ip.festo-didactic.com/InfoPortal/Robotino/Overview/EN/index.html (accessed on May 2, 2024).

107. R. Raj and A. Kos, "Intelligent Mobile Robot Navigation in Unknown and Complex Environment Using Reinforcement Learning Technique," *Scientific Reports* (Under review), (Submitted on March 16, 2024).

108. J. M. O'kane, B. Tovar, P. Cheng, and S. M. LaValle, "Algorithms for planning under uncertainty in prediction and sensing," *Autonomous Mobile Robots: Sensing, Control, Decision-Making, and Applications, Series in Control Engineering, CRC Press*, pp. 501–547, 2006.

109. M. J. M, R. Mathew, and S. S. Hiremath, "Deep reinforcement learning Based Approach For Mobile Robot Navigation," *2019 International Conference on Computational Intelligence and Knowledge Economy (ICCIKE), Dubai, United Arab Emirates*, 2019, pp. 523-526. https://doi.org/10.1109/ICCIKE47802.2019.9004256

110. R. Raj and A. Kos, "Dynamic Obstacle Avoidance Technique for Mobile Robot Navigation Using Deep reinforcement learning," *International Journal of Emerging Trends in Engineering Research*, vol. 11, no. 9, pp. 307–314, Sep. 2023. https://doi.org/10.30534/ijeter/2023/031192023

111. Guo-Sheng Yang, Er-Kui Chen and Cheng-Wan An, "Mobile robot navigation using neural Q-learning," *Proceedings of 2004 International Conference on Machine Learning and Cybernetics (IEEE Cat. No.04EX826), Shanghai, China*, vol.1, pp. 48-52, 2004. https://doi.org/10.1109/ICMLC.2004.1380601

112. D. L. Poole, and A. K. Mackworth, "Artificial Intelligence: foundations of computational agents," *Cambridge University Press*, 2010.

113. C. XIA, A. EL KAMEL, "A Reinforcement Learning Method of Obstacle Avoidance for Industrial Mobile Vehicles in Unknown Environments Using Neural Network," *In: Qi, E., Shen, J., Dou, R. (eds) Proceedings of the 21st International Conference on Industrial Engineering and Engineering Management 2014. Proceedings of the International Conference on Industrial Engineering and Engineering Management. Atlantis Press, Paris*, 2015. https://doi.org/10.2991/978-94-6239-102-4_136

114. M. A. K. Jaradat, M. Al-Rousan, and L. Quadan, "Reinforcement based mobile robot navigation in dynamic environment," *Robotics and Computer-Integrated Manufacturing*, vol. 27, pp. 135–149, 2011. https://doi.org/10.1016/j.rcim.2010.06.019

115. J. Qiao, Z. Hou, and X. Ruan, "Application of reinforcement learning based on neural network to dynamic obstacle avoidance," *2008 International Conference on Information and Automation, Changsha, China*, 2008, pp. 784-788. https://doi.org/10.1109/ICINFA.2008.4608104

116. M. Riedmiller, "Neural Fitted Q Iteration – First Experiences with a Data Efficient Neural Reinforcement Learning Method," *In: Gama, J., Camacho, R., Brazdil, P.B., Jorge, A.M., Torgo, L. (eds) Machine Learning: ECML 2005. ECML 2005, Lecture Notes in Computer Science ()*, vol 3720. *Springer, Berlin, Heidelberg*, pp 317–328, 2005. https://doi.org/10.1007/1156409632

117. R. Raj, and A. Kos, "Study and Analysis of Discrete Event-Driven Autonomous System with a Case Study for a Robotics Task," *Przegląd Elektrotechniczny*, vol. 2023, no. 9, pp. 50-56, Feb. 2023. https://doi.org/10.15199/48.2023.02.01

118. R. Raj and A. Kos, "Discussion on different controllers used for the navigation of mobile robot," *International Journal of Electronics and Telecommunications*, vol. 70, no. 1, pp. 229-239, March 2024. https://doi.org/10.24425/ijet.2024.149535

119. R. Raj and A. Kos, "An Optimized Energy and Time Constraints-Based Path Planning for the Navigation of Mobile Robots Using an Intelligent Particle Swarm Optimization Technique," *Applied Sciences*, vol. 13, no. 17, p. 9667, Aug. 2023. https://doi.org/10.3390/app13179667

120. P. Bhattacharya and M. L. Gavrilova, "Roadmap-Based Path Planning - Using the Voronoi Diagram for a Clearance-Based Shortest Path," in *IEEE Robotics & Automation Magazine*, vol. 15, no. 2, pp. 58-66, June 2008. https://doi.org/10.1109/MRA.2008.921540

121. J. Zhang, Y. Lu, L. Che, and M. Zhou, "Moving-Distance-Minimized PSO for Mobile Robot Swarm," in *IEEE Transactions on Cybernetics*, vol. 52, no. 9, pp. 9871-9881, Sept. 2022. https://doi.org/10.1109/TCYB.2021.3079346

122. S. Adarsh, S. M. Kaleemuddin, D. Bose, and K. I. Ramachandran, "Performance comparison of Infrared and Ultrasonic sensors for obstacles of different materials in vehicle/ robot navigation applications," *IOP Conference Series: Materials Science and Engineering*, vol. 149, p. 012141, Sep. 2016. https://doi.org/10.1088/1757-899x/149/1/012141

123. P. Marzec, and A. Kos, "Thermal navigation for blind people*," Bulletin of The Polish Academy of Sciences Technical Sciences*, Vol. 69(1), 2021, Article number: e136038. https://doi.org/10.24425/bpasts.2021.136038

124. P. Marzec and A. Kos, "Indoor Precise Infrared Navigation," *2020 27th International Conference on Mixed Design of Integrated Circuits and System (MIXDES), Lodz, Poland*, 2020, pp. 249-254. https://doi.org/10.23919/MIXDES49814.2020.9155998

125. S. Blessenohl, C. Morrison, A. Criminisi and J. Shotton, "Improving Indoor Mobility of the Visually Impaired with Depth-Based Spatial Sound," *2015 IEEE International Conference on Computer Vision Workshop (ICCVW), Santiago, Chile*, 2015, pp. 418-426. https://doi.org/10.1109/ICCVW.2015.62

126. P. Marzec and A. Kos, "Road Line Detection by Reflected Heat Assistant system for car navigation," *2022 29th International Conference on Mixed Design of Integrated Circuits and*

System (MIXDES), Wrocław, Poland, 2022, pp. 213-217. https://doi.org/10.23919/MIXDES55591.2022.9838012

127. M. F. Norazman, and N. M. Thamrin, "Landmark scanning by using infrared sensor for simultaneous localization and mapping application," *2018 IEEE 14th International Colloquium on Signal Processing & Its Applications (CSPA), Penang, Malaysia*, 2018, pp. 145-149. https://doi.org/10.1109/CSPA.2018.8368702

128. A. Muis and J. Mae, "Simple road side detection and tracking on embedded PC," 2018 IEEE 15th International Workshop on Advanced Motion Control (AMC), Tokyo, Japan, 2018, pp. 301-305, https://doi.org/10.1109/AMC.2019.8371107

129. P. Marzec and A. Kos, "Low Energy Precise Navigation System for the Blind with Infrared Sensors," *2019 MIXDES - 26th International Conference "Mixed Design of Integrated Circuits and Systems", Rzeszow, Poland*, 2019, pp. 394-397, https://doi.org/10.23919/MIXDES.2019.8787093

130. E. Di Mario, Z. Talebpour and A. Martinoli, "A comparison of PSO and Reinforcement Learning for multi-robot obstacle avoidance," *2013 IEEE Congress on Evolutionary Computation, Cancun, Mexico*, 2013, pp. 149-156. https://doi.org/10.1109/CEC.2013.6557565

131. J. Wu, C. Song, J. Ma, J. Wu, and G. Han, "Reinforcement Learning and Particle Swarm Optimization Supporting Real-Time Rescue Assignments for Multiple Autonomous Underwater Vehicles," in *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 7, pp. 6807-6820, July 2022. https://doi.org/10.1109/TITS.2021.3062500

132. I. Papagianopoulos, G. De Mey, A. Kos, B. Wiecek, and V. Chatziathasiou, "Obstacle Detection in Infrared Navigation for Blind People and Mobile Robots," *Sensors*, vol. 23, no. 16, p. 7198, Aug. 2023. https://doi.org/10.3390/s23167198

133. W. Qiang and Z. Zhongli, "Reinforcement learning model, algorithms and its application," *2011 International Conference on Mechatronic Science, Electric Engineering and Computer (MEC), Jilin, China*, 2011, pp. 1143-1146. https://doi.org/10.1109/MEC.2011.6025669