applied sciences

MDPI

Article

Autonomous Navigation of Robots: Optimization with DQN

Juan Escobar-Naranjo [1,†], Gustavo Caiza [2,†], Paulina Ayala [1,†], Edisson Jordan [1,†], Carlos A. Garcia [3,†] and Marcelo V. Garcia [1,*,†]

[1] Faculty of Systems, Electronics and Industrial Engineering, Universidad Tecnica de Ambato (UTA), Ambato 180206, Ecuador; jc.escobar@uta.edu.ec (J.E.-N.); ep.ayala@uta.edu.ec (P.A.); edissonpjordan@uta.edu.ec (E.J.)
[2] Electronics and Automation Department, Universidad Politecnica Salesiana (UPS), Quito 170146, Ecuador; gcaiza@ups.edu.ec
[3] Department of Systems Engineering, Automation and Industrial Informatics, Universitat Politecnica de Catalunya (UPC), 08034 Barcelona, Spain
* Correspondence: mv.garcia@uta.edu.ec; Tel.: +593-998-267-906
† These authors contributed equally to this work.

Featured Application: The application of "Autonomous Navigation of Robots: Optimization with DQN" involves using reinforcement learning techniques to optimize the navigation of autonomous robots. Specifically, the Deep Q-Network (DQN) algorithm is used to train a robot to make decisions based on sensory inputs and to learn optimal paths to reach a goal. This can have a wide range of potential applications, such as in manufacturing and logistics, where robots can autonomously navigate and transport materials within a warehouse; or in search and rescue missions, where robots can navigate through disaster-stricken areas to locate and rescue survivors. By optimizing the navigation process with DQN, these robots can operate more efficiently and safely in their respective environments.

Abstract: In the field of artificial intelligence, control systems for mobile robots have undergone significant advancements, particularly within the realm of autonomous learning. However, previous studies have primarily focused on predefined paths, neglecting real-time obstacle avoidance and trajectory reconfiguration. This research introduces a novel algorithm that integrates reinforcement learning with the Deep Q-Network (DQN) to empower an agent with the ability to execute actions, gather information from a simulated environment in Gazebo, and maximize rewards. Through a series of carefully designed experiments, the algorithm's parameters were meticulously configured, and its performance was rigorously validated. Unlike conventional navigation systems, our approach embraces the exploration of the environment, facilitating effective trajectory planning based on acquired knowledge. By leveraging randomized training conditions within a simulated environment, the DQN network exhibits superior capabilities in computing complex functions compared to traditional methods. This breakthrough underscores the potential of our algorithm to significantly enhance the autonomous learning capacities of mobile robots.

Keywords: autonomous navigation; optimization DQN (Deep Q-Network); robotics; artificial intelligence

Citation: Escobar-Naranjo, J.; Caiza, G.; Ayala, P.; Jordan, E.; Garcia, C.A.; Garcia, M.V. Autonomous Navigation of Robots: Optimization with DQN. *Appl. Sci.* **2023**, *13*, 7202. https://doi.org/10.3390/app13127202

Academic Editors: Benjamim Fonseca, Ivo Pereira and Ana Madureira

Received: 11 April 2023
Revised: 25 May 2023
Accepted: 7 June 2023
Published: 16 June 2023

1. Introduction

Industries in the present era are confronted with the challenge of staying abreast of technological progress by fostering innovation and investing in the advancement of efficient equipment and work methods. Over recent years, a notable transformation has taken place, driven by the adoption of artificial intelligence (AI) and the implementation of the DQN algorithm. This paradigm shift has given rise to the emergence of intelligent, adaptable, and dynamic manufacturing processes. By integrating AI and DQN into manufacturing operations, several benefits have been realized, including the automation of manufacturing

processes, the seamless exchange of real-time data leveraging cloud computing, and the establishment of interconnected cyberphysical systems. Consequently, this integration has paved the way for the development of control algorithms rooted in machine learning, thereby further augmenting the capabilities of manufacturing processes [1].

The implementation of AI- and DQN-powered autonomous systems, sensors, and actuators has revolutionized the manufacturing industry, enabling companies to achieve elevated quality standards, cost reductions, minimized downtime, and increased competitiveness. This integration has not only been successful, but also serves as a testament to the immense potential for further advancements in the creation of intelligent, efficient, and highly productive manufacturing systems in the future [2,3].

In this new era of the industrial revolution, we are witnessing the emergence of various transformative advancements. These include the deployment of autonomous production lines, the adoption of intelligent manufacturing practices, the implementation of efficient and secure data management systems, and the attainment of unparalleled process quality. These developments have given rise to a heightened sense of competitiveness among companies and have opened up new avenues for productivity across diverse sectors of the market. Notably, these advancements are fueled by innovative approaches and the integration of predictive tools that leverage data analysis to facilitate informed decision-making, thereby significantly reducing risks [4–7].

In this era of industrial transformation, we are currently witnessing a transformation in manufacturing practices driven by the integration of the DQN algorithm. This has led to the emergence of autonomous production lines, intelligent manufacturing practices, effective and protected data handling, and high-quality processes [8,9]. The use of the DQN algorithm for predictive modeling substantially reduces risks by providing accurate and reliable insights into the manufacturing process. By leveraging the power of AI and the DQN algorithm, manufacturing companies can optimize their operations, reduce costs, improve efficiency, and ultimately gain a competitive edge in the market [10,11].

Recently, mostly in globalized countries, statistical methods have been developed and applied for the analysis of data models, because prediction is important when making decisions, which can be risky and can represent a positive or negative change within a manufacturing process [12,13], and the utilization of machine learning systems has emerged as an innovative approach that utilizes statistical techniques to analyze and optimize algorithms. These algorithms were developed based on insights derived from prior outcomes, leading to the emergence of a regression-based learning methodology, whose application focuses on systems being reconfigured in real-time, and this being carried out in such a way that they automatically find the most optimal and efficient way to perform their respective operations [14–16].

In this context, the aim of this research is to present an autonomous model capable of controlling and reconfiguring the route of a mobile robot in real time by learning from its previous paths; an autonomous learning method based on reinforcements was implemented, which must learn to avoid fixed and mobile obstacles, in order to reach the target position and obtain a positive reward.

This study focuses on two main objectives. Firstly, it involves the calculation of Turtlebot3 Burger kinematics, which is crucial for determining the differential control of each wheel. This calculation is essential for accurately controlling the movement of the robot. The second objective of this research is to develop a control system based on DQN (Deep Q-Network) and reinforcement learning. The aim is to design a system that utilizes the most suitable optimization algorithm to minimize errors and enhance performance. Python was chosen as the programming language for this model due to its extensive documentation and support for the Keras library. Python and Keras provide a robust framework for network development and training. To facilitate communication between the simulated environment and the receiver and controller nodes, the ROS (Robot Operating System) API was utilized. This enables the exchange of position, odometry, and speed messages, ensuring seamless interoperability between components. The ROS API

offers a convenient interface for integrating different modules and simplifies the overall system implementation.

In the context of mobile robot control, the existing research has primarily focused on the use of predetermined paths for the robot's trajectory. While this approach has proved to be effective in many cases, it does not account for real-time methods for avoiding obstacles and reconfiguring the robot's route. This creates a significant research gap that needs to be addressed to develop more intelligent and efficient mobile robots. Therefore, this study aims to fill this research gap by presenting an autonomous learning method based on reinforcement learning and a control system based on the DQN algorithm. These components enable the mobile robot to learn from its previous paths, avoid obstacles, and reconfigure its route in real time, which leads to more efficient and intelligent mobile robots. The significance of this research lies in its contribution to the advancement of autonomous robotics by developing a more efficient and intelligent control system for mobile robots, which has implications for various industrial and service sectors.

The research contribution of this study can be summarized in two highlights. Firstly, it presents an autonomous learning method based on reinforcements, which enables a mobile robot to learn from its previous paths and reconfigure its route in real time while avoiding obstacles. This method is critical for the development of more intelligent and efficient mobile robots that can adapt to their environments and perform their tasks autonomously.

Secondly, this research introduces a control system that utilizes the DQN algorithm and reinforcement learning to enhance the robot's path planning by minimizing errors. The design of the control system is based on the most suitable optimization algorithm, and Python is chosen as the programming language for model development, including network training, due to its capabilities and the availability of extensive resources.

The communication between the simulated environment and the receiver and controller nodes is established using the ROS API. This integration allows for seamless interoperability by enabling the exchange of commands. As a result, the mobile robot can effectively transmit and receive position, odometry, and speed messages. This integration significantly improves the robot's efficiency and accuracy in performing its tasks. The outcomes of this study contribute to the advancement of autonomous robotics by providing a more intelligent and efficient control system for mobile robots.

The structure of this article is organized as follows: In Section 2, various relevant studies related to the research topic are presented. Section 3 provides a comprehensive overview of the fundamental concepts necessary for a deeper comprehension of the subsequent sections. The case study for the research methodology is presented in Section 4. Section 5 outlines the proposed implementation for developing a control algorithm for the mobile robot using reinforcement learning (RL). The results and findings are discussed in Section 6. Finally, Section 7 provides detailed conclusions and outlines potential avenues for future research.

2. Related Work

In this section, a comprehensive analysis is conducted on the most notable research and work pertaining to the design and implementation of control systems based on machine learning. The objective of this research proposal is further described from a technical standpoint, elucidating its potential applicability to diverse control systems and optimization algorithms. By exploring the existing body of knowledge and examining relevant studies, this section aims to establish a solid foundation for the proposed research, fostering a deeper understanding of the subject matter and its implications in the field of control system design.

Reinforcement learning (RL) is a type of machine learning that involves an agent interacting with an environment to learn the best actions to take in order to maximize a reward signal. RL has been successfully applied in robotics to solve a variety of tasks, including path determination [17–20]. Path determination in robotics involves finding a safe and efficient path for a robot to follow in order to complete a task. This can be a

challenging problem, as robots need to navigate through unknown environments while avoiding obstacles and minimizing the risk of collisions [21–28].

RL can be used to train a robot to determine the best path to take in order to achieve its goals [29–32]. The robot can be trained by interacting with its environment and receiving feedback in the form of rewards or penalties based on the actions it takes. For example, the robot may receive a reward for successfully navigating to a particular location, while receiving a penalty for colliding with an obstacle [33–37]. Over time, the robot can use the feedback it receives to learn the optimal path to take in different environments. This allows the robot to adapt to changes in its environment and make decisions in real time based on the current situation [38–41].

Due to technological advances, the research and implementation of robotic systems are in constant development, trying to optimize self-control, leading their system to be based on autonomous operations and intelligent decision making [42,43]. Especially for movement, different control methods have been designed that vary according to their field of application; however, the most used is the predictive model, which is based on generating a decision based on statistics, which in turn uses a large amount of data in industrial environments [44–49].

Various investigations [50,51] indicate that mobile robots are extensively utilized in diverse industrial and commercial settings, often replacing human labor. Instances can be found in warehouses and hospitals, where these robots are responsible for material transportation and assisting workers in repetitive tasks that may have adverse effects on their wellbeing [52,53]. Additionally, it is observed that in many domains involving mobile robots, the processing of a substantial amount of information poses a significant challenge. Within the context of machine learning, this refers to the learning process taking place within an environment that encompasses both fixed and mobile obstacles. The navigation tasks of mobile robots typically involve various optimization concepts, such as cost reduction, shorter trajectories, and minimized processing time. However, in complex and unpredictable industrial environments, adaptability to surroundings becomes essential [54–56].

The Deep Q-Network (DQN) algorithm is a popular variant of reinforcement learning that has been successfully applied to a wide range of problems, including path determination in robotics [57–59]. In DQN, the agent uses a neural network to approximate the optimal action-value function, which maps states to the expected long-term reward for each possible action. The network is trained using a variant of the Q-learning algorithm, where the agent learns from the transitions between states and the rewards received for each action [60–62].

To apply DQN to path determination in robotics, the agent must first be trained on a set of sample environments. During training, the agent explores the environment and learns to predict the optimal action to take in each state. The agent's performance is evaluated based on its ability to navigate to a specified target location while avoiding obstacles [63–66].

A novel work is presented in [54], where reinforcement learning is applied; it is mentioned that the robot will learn by training the environment in which it is located through a scoring system designed in a Deep Q network, which later will allow it to take the optimal action or strategy to move to its target position while avoiding obstacles.

Also in [67], it is mentioned that laser sensors are used for the autonomous navigation of mobile robots because they have a wide detection range and high precision; in addition to this research, it is detailed that robots need enough information to correctly detect obstacles and map the environment to subsequently carry out correct navigation.

Programming interfaces have been developed for the design and control of open-source applications, owing to the changes in computer systems and the incorporation of robotics into the industry [68,69]. ROS has the stability to program any type of robotic function, either for a manipulator or a mobile robot; in [70], this framework is used to control the entire flow of information through the tools and libraries included in its API, which contains functional packages that facilitate the creation and communication between nodes.

As mentioned in [71], a successful application developed between big data and machine learning can provide different solutions for solving problems in manufacturing industries, especially in the product life cycle and the entire supply chain. Also in this research, a model based on energy saving is presented using machine learning to determine the optimal trajectory for an industrial robot; the keys to this design are based on data collection and the control algorithm.

For mobile robots, autonomous navigation based on machine learning is the key to high-impact technological advancement; several works [17,72–88] indicate that industrial robots that have the appropriate sensors and the specific control algorithm can understand the environment in which they are located, to which humans generally do not have access, such as in war zones or nuclear plants.

The research by [89] shows three different control algorithms based on different techniques; in addition to the development of a simulation environment for the training of the mobile robot, a function focused on Q-learning was also introduced to return the reward value according to the records of executions and thus validate the performance of the robot. The Q-values are reported to be real, which turns the neural network into a regression task, allowing for optimization using the squared error loss function. Overall, the combination of reinforcement learning with the DQN algorithm has shown great promise in enabling robots to autonomously navigate through complex environments and find safe and efficient paths to their goals.

As can be seen in this section, progress in Industry 4.0 has led to the development of new technologies applicable to mobile robots, allowing for the design of control systems based on autonomous learning, resulting in more efficient devices due to the trajectory obtained by the optimized algorithm. This work proposes a mobile robotic system trained by means of simulation, capable of avoiding obstacles and being optimal in tracking trajectories.

3. State of Technology

In this section, all concepts related to the development of this research will be explained.

3.1. Turtlebot3 Burger

This is defined as a small mobile robot designed for the education and research of those who want to learn to program applications through the API of the Robot Operating System (ROS); this device is part of the new generation of robots whose hardware, including its sensors, is of a relatively low cost and its software is of the open-source type. The robot, depending on the application that is programmed, is capable of performing several tasks without external components, the most relevant example of which is its simultaneous location and mapping algorithm (SLAM), which allows it to communicate in real time and build the environment [90].

The robot chosen for the simulation of this research is the Turtlebot3 Burger, which is presented in Figure 1. It is mobile, small, and its programming interface is ROS; in addition, its hardware is composed of plastic plates, metal bars, two wheels, and fixing elements. The sensor integrated into the upper part of the robot is a 360-degree planar Lidar, which transmits a rotary laser that is reflected in the obstacles, for which the reflection time allows one to obtain the distance at which it is of the objects [91]. See Figure 1.

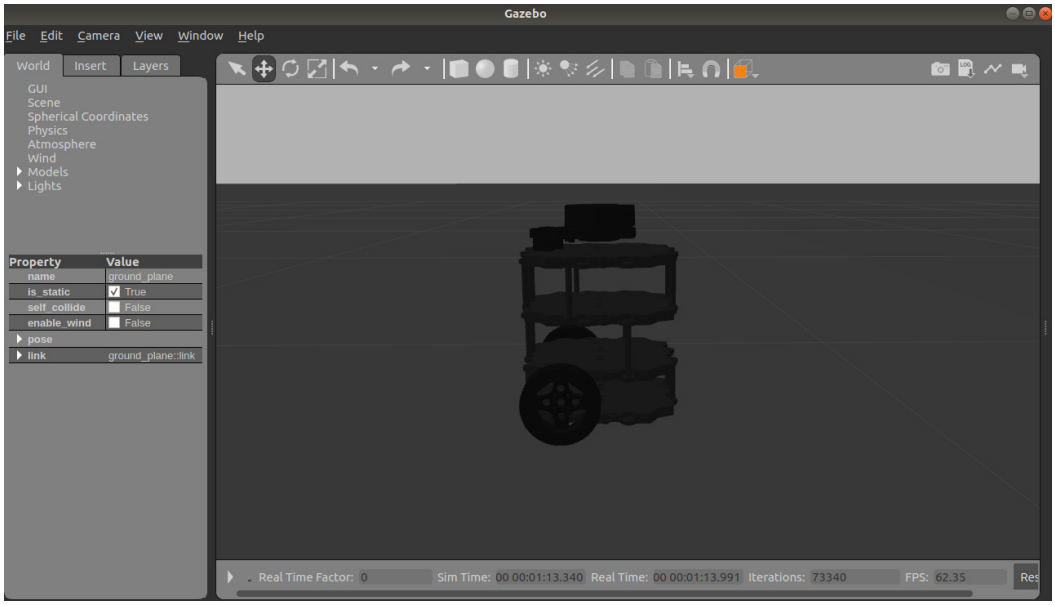

Figure 1. Turtlebot3 Burger in Gazebo.

The Turtlebot3 Burger has a Raspberry Pi 3 SBC type card, in charge of managing the sensor data; the hardware control has an Arduino-based card, which has an accelerometer, gyroscope, and magnetometer. In addition, the robot has a differential configuration on its wheels, which means that both motors are driven independently, in the same way, it has a rotating wheel that provides stability [92].

3.2. Kinematics

In this subsection, as part of the analysis of the mobile robot, the kinematics of the Turtlebot3 Burger is presented.

The kinematic analysis in mobile robots allows one to know the orientation and position in an environment without considering the forces that generated any change; from this information, it can be determined where the device is or where it will go. The main purpose is to represent the linear speed of the robot and the orientation angle based on the speeds of the wheels and the geometry of the robot [93].

Due to the fact that the robot moves only on the plane, the analysis is taken as two-dimensional. The differential drive of the wheels and the distance between them are also considered; the parameters to be determined are the orientation θ and the X_o, Y_o position respect to the absolute origin, all depending on the known variables.

The speed of the robot V_R is given as the average of the linear speeds of each wheel; they are presented as V_1 and V_2 in Figure 2. The calculation of V_R is shown in Equation (1).

$$V_R = R\frac{V_1 + V_2}{2} \tag{1}$$

The angular speed of the robot is described in Equation (2), while the components of the movement x', y' are described in (3), (4) and the new orientation as a function of time in (5).

$$\omega = R\frac{V_2 - V_1}{L} \tag{2}$$

$$x' = x_o + V_R \cos\theta \tag{3}$$

$$y' = y_o + V_R \sin\theta \tag{4}$$

$$\theta' = \theta + \omega_R t \tag{5}$$

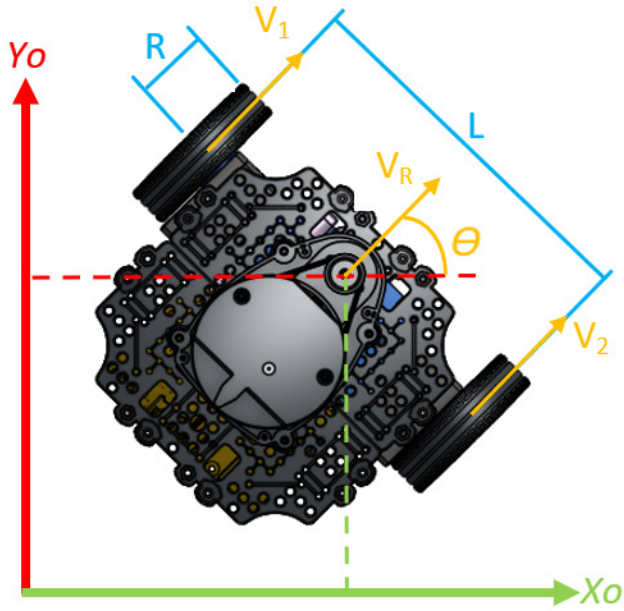

Figure 2. Turtlebot3 Burger in the plane, illustrating the robot's kinematic configuration and coordinate system.

After deriving Equations (3)–(5) as a function of time, they can be presented as a space of state in (6).

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} V_R \\ \omega_R \end{bmatrix} \tag{6}$$

Finally after replacing (1) and (2) in (6), Equation (7) is obtained as the matrix representation of the angular velocities of each tire of the robot.

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \frac{R\cos\theta}{2} & \frac{R\cos\theta}{2} \\ \frac{R\sin\theta}{2} & \frac{R\sin\theta}{2} \\ \frac{R}{L} & \frac{-R}{L} \end{bmatrix} \begin{bmatrix} \omega_2 \\ \omega_1 \end{bmatrix} \tag{7}$$

3.3. Machine Learning

One of the latest technological advances applicable to mobile robots is autonomous learning, which is defined by several authors as the ability to teach machines to handle large amounts of information efficiently, so today many industries apply machine learning to learn from data [94].

Different algorithms have been developed to solve data management, for which there is no single algorithm to solve a specific problem; however, the algorithm will depend on the problem to be solved, the number of variables, the model that best suits, and so on. For this article, the use of reinforcement learning with DQN will be described.

Reinforcement learning is one of the three main paradigms of machine learning with supervised learning and unsupervised learning; this deals with how (artificial) agents should perform actions in an environment in a way that maximizes a cumulative reward. In addition, the action changes the state of the environment in some way, and the agent receives this information and its corresponding reward, that is, based on the knowledge it acquires, it decides the next action, improving its strategy in order to increase its reward [95].

This type of learning allows the mobile robot to learn from the environment while it interacts with it; the objective of applying this type of training is that it allows the agent to find the most optimal strategy through reinforcement.

Within reinforcement learning is Q learning, which is based on a matrix of Q values that assigns the quality of the action when the environment is in a certain state; this Q value is refined through successive approximations. When the set of states and actions becomes so large, it becomes difficult to store the entire Q matrix, which is why neural networks are used to present it accurately. This is called deep Q learning [95].

The agent's action depends on the value of epsilon ϵ, which introduces randomness to the system. If this is zero, the algorithm will never perform random trajectories and will only be based on the stored knowledge; if it is only one, it will perform random actions. Usually, this value is set very close to zero [96].

As shown in Figure 3, the Turtlebot3 Burger agent during training takes actions (A_t) based on the current state (S_t); this gives rise to a new state called (S_{t+1}), and as a result of this iteration, obtains a reward (R_t). This process is repeated iteratively, looking for the accumulated rewards to increase over time.

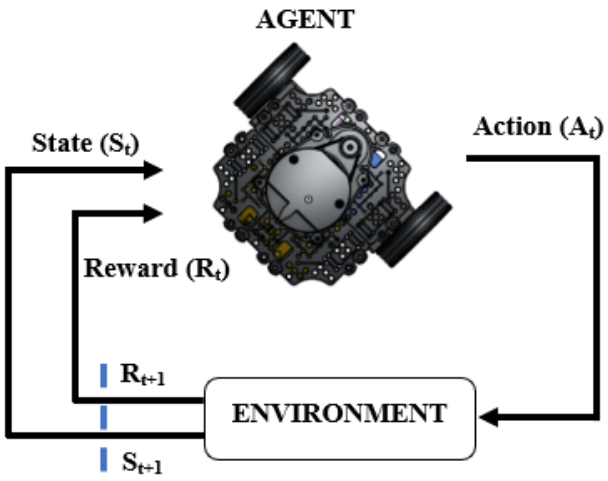

Figure 3. Illustration of the iterative reinforcement learning process.

3.4. Robot Operating System (ROS)

ROS is an open-source meta-operating system widely used in research and industrial robotics applications. In recent years, it has become a leader in the development of applications in mobile robots and manipulators due to the fact that it provides libraries and tools for localization, navigation, and communication. ROS, through its API, allows for the design of a publisher–subscriber interface, where those devices that communicate are called 'nodes', which can send and receive messages through 'topics', that is, a node can make certain information available in a topic in the form of a 'message' [70,92,97].

The programming can be described mainly in C++ and Python; it must be taken into account that in this architecture, there is always a master node, as shown in Figure 4.

3.5. Gazebo

This is defined as a three-dimensional simulator for open-source robotic applications. It is essential for the training of mobile robots and manipulators based on neural networks; it also allows for the testing of control algorithms, performing sensor readings and simulating complete systems in realistic environments. Gazebo allows for a correct simulation of the Turtlebot3 Burger mobile robot, allowing for correct control over collisions and the Lidar sensor. It is also compatible with ROS, which allows for the development of high-quality systems with a graphical interface that is very close to reality [90].

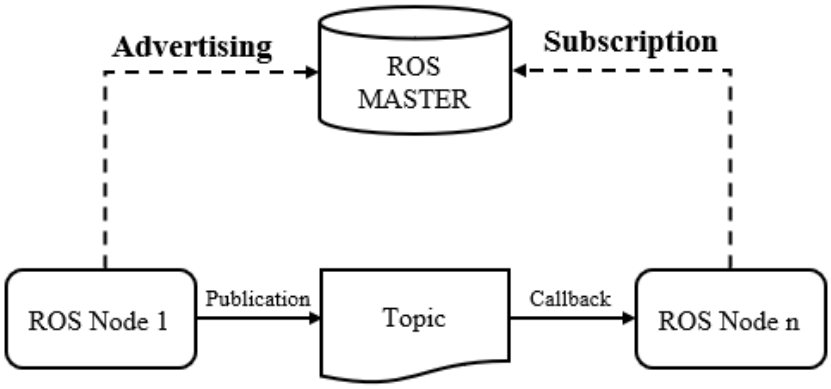

Figure 4. Node communication model in ROS.

4. Study Case

One of the primary tasks commonly performed by mobile robots involves obstacle avoidance, where optimization criteria such as minimizing cost and traversing the shortest distance within a limited timeframe are considered. Traditionally, this process required continuous monitoring and intervention from operators. However, advancements in technology have led to the automation of this process in recent years.

This study focuses on the development of a control system based on machine learning for the Turtlebot3 Burger mobile robot. Specifically, it aims to simulate the training process for trajectory planning, which includes obstacle avoidance and collision prevention. As depicted in Figure 5, the proposed architecture integrates multiplatform systems such as ROS, enabling seamless message transmission among different components of the system. Additionally, the Gazebo simulator and the Deep Q learning algorithm are employed to facilitate autonomous learning through reinforcement.

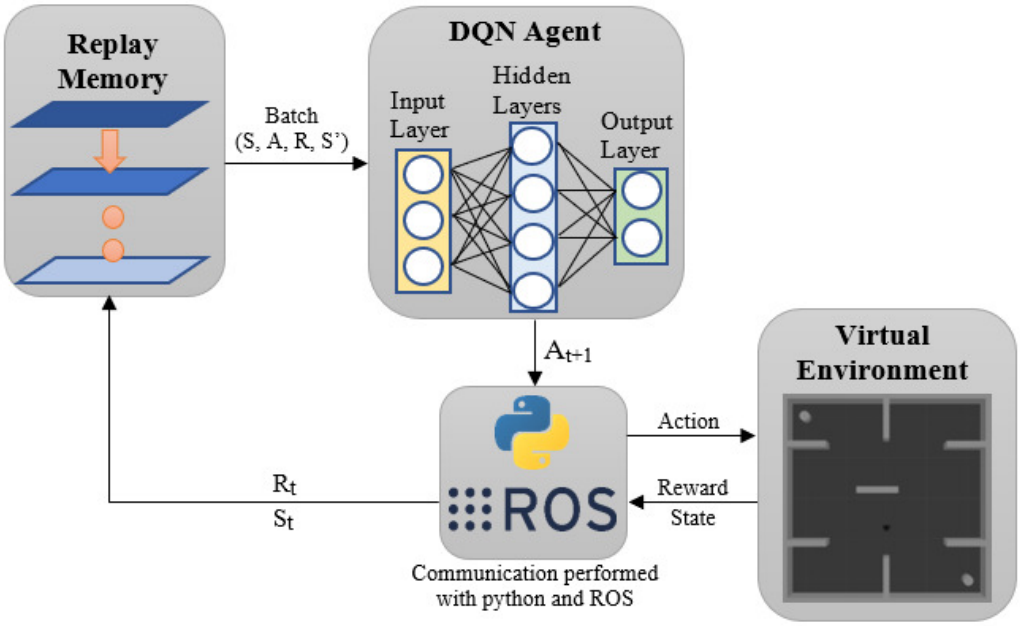

Figure 5. Study case architecture.

The DQN agent selects the action based on the maximum Q-value, and this is reflected in the environment. Then, the obtained reward is stored in the replay memory, which is used because the network should not be trained with the current value; instead, this information must be stored in a buffer that allows for the sampling of the batches of experience, either random or prioritized, so that the network can readjust itself. Finally, after there are enough examples in memory, the agent is updated randomly, thanks to

the previously performed sampling. ROS is used to launch the simulation and the DQN node, which allows for easy communication based on the publication and subscription of messages through topics, where the observed state S, the subsequent state S', the reward R and the action A are transmitted, which are part of the batch for the reinforcement learning algorithm.

The objective of this work is to use machine learning in a mobile robot within a simulated environment to solve mobile and fixed obstacle avoidance through a multiplatform and open-source system. The control algorithm is developed as a node and includes some layers within its programming as a neural network, which allows it to obtain the state from the environment, calculate the reward, and establish an action based on the stored results.

5. Design and Execution Strategy for the Control System: A Comprehensive Overview

This section serves as a comprehensive guide to the planned actions involved in the development of the control system for mobile robot obstacle avoidance. The primary objective is to delve into the intricate details of the proposed implementation, enabling the thorough identification, establishment, and analysis of the available information. By providing a comprehensive understanding of the control system's design and methodology, readers will have the necessary tools to critically evaluate the validity and effectiveness of the study.

In order to ensure a robust and reliable control system, it is essential to address key aspects such as sensor capabilities, state representation, action selection, and reward mechanisms. By delving into these crucial components, this section aims to provide a solid foundation for the subsequent sections, facilitating a deeper comprehension of the control system's intricacies and functionality.

Furthermore, by providing an extensive description of the planned actions, readers will gain insights into the technical decisions made during the development process. This transparency enhances the study's reproducibility and allows for a more comprehensive evaluation of the control system's performance and potential implications.

Ultimately, through the detailed exposition of the planned actions, this section aims to establish a robust foundation for the subsequent sections, laying the groundwork for a comprehensive analysis and evaluation of the proposed control system for mobile robot obstacle avoidance.

5.1. Understanding the Robot's Environment: Utilizing Lidar Sensor for State Perception

The robot state plays a crucial role in the control system, as it encompasses the observations made during the interactions between the Turtlebot3 Burger and its environment. By capturing and analyzing the state information, the control system can make informed decisions for effective robot behavior.

To enable state perception, the Turtlebot3 Burger is equipped with a 2D laser distance sensor known as Lidar or LDS. This sensor provides a comprehensive view of the robot's surroundings, allowing it to sample the environment in a 360-degree range and detect objects located between 120 mm and 3500 mm. The Lidar is mounted on a rotating head, which offers the flexibility to adjust the sample size based on the specific requirements of the application.

In the simulation environment, Gazebo, a carefully chosen configuration of 11 Lidar points is utilized. This choice strikes a balance between the utilization of computational resources and the inputs required for the control system. By carefully selecting the number of Lidar points, the control system can effectively capture relevant information about the environment while maintaining computational efficiency.

With this comprehensive perception of the robot's surroundings, the control system can leverage the state information to analyze the environment, identify obstacles, and plan optimal trajectories for obstacle avoidance. The accurate and detailed perception obtained through the Lidar sensor greatly enhances the overall capabilities of the robot and facilitates its safe and efficient navigation in complex environments.

By leveraging the state perception capabilities, the control system can ensure that the robot interacts with its surroundings in a way that aligns with its objectives, such as avoiding obstacles and efficiently navigating through its environment. The precise representation of the robot state, facilitated by the Lidar sensor, lays the foundation for effective decision making and control actions to be taken by the robot.

$$State = LDS(11values) + Distance(1) + Angle(1) \quad (8)$$

The 11 sensor values obtained from the Lidar sensor are crucial for capturing the robot's perception of its environment. These sensor values are transmitted through the Robot Operating System (ROS) framework, where they are published in a designated topic by a specific node. This allows other components of the system to access and utilize this valuable information.

In addition to the sensor values, the state representation includes two additional components. These components are derived from the angle between the rotation of the robot and the distance to the target. The calculation of this angle relies on the availability of odometry information, which provides the position of the robot with respect to its reference frame. The odometry data are published in the topic called "/odom," allowing the control system to access and incorporate it into the state representation.

By combining the 11 sensor values with the angle and odometry information, the state representation comprises a total of 13 components. This extended state representation provides a more comprehensive understanding of the robot's surroundings and its position relative to the target. Figures 6 and 7 visually illustrate the relationship between the sensor values, angle, and odometry in the state representation. Equation (8) represents the mathematical formulation of the state, highlighting the individual components that contribute to its definition.

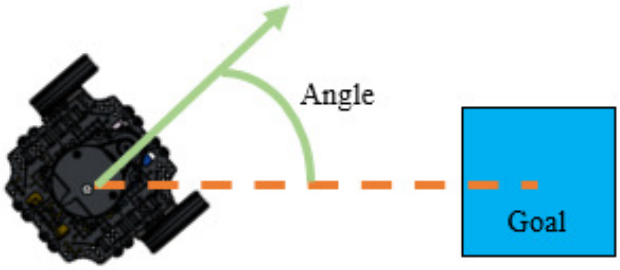

Figure 6. Illustrating angle calculation: sensor values, odometry, and spatial orientation.

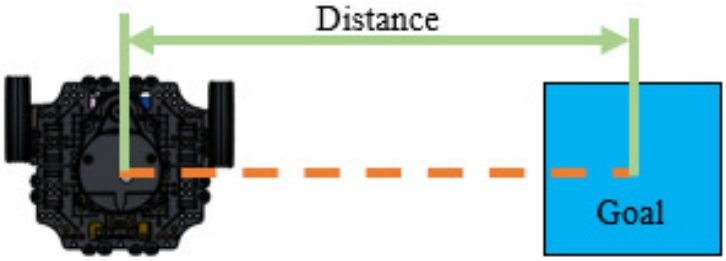

Figure 7. Analyzing distance calculation: sensor readings, odometry, and spatial relationship.

The integration of sensor values, angle, and odometry information into the state representation enables the control system to make informed decisions based on a more comprehensive understanding of the robot's current state and its relation to the target. This enhanced state representation is vital for effective path planning, obstacle avoidance, and overall navigation performance of the Turtlebot3 Burger in its environment.

5.2. Robot's Interaction with the Environment: Actions of the DQN Agent

The DQN agent's actions refer to the movements performed by the Turtlebot3 Burger as it interacts with the environment. These actions are determined based on the output produced by the last layer of the neural network. The Turtlebot3 Burger is designed to have a set of five actions that dictate its movement, depending on the observed state. A visual representation of these actions can be found in Figure 8.

By leveraging the power of the neural network, the DQN agent is able to make informed decisions regarding its movement, taking into account the information it has gathered from the environment. These actions play a crucial role in navigating obstacles and accomplishing tasks effectively and efficiently.

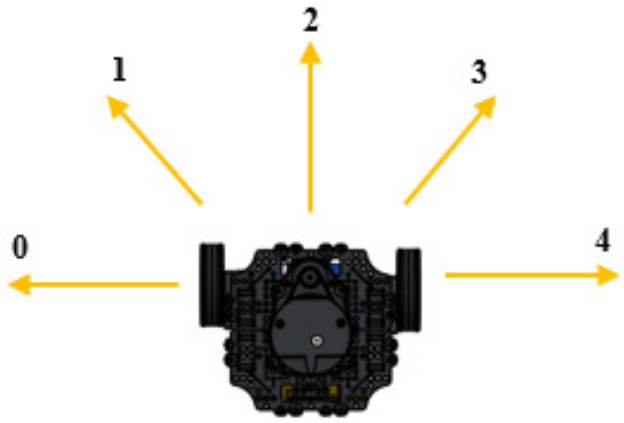

Figure 8. Visualizing Turtlebot3 Burger's movement actions for environment interaction.

Additionally, the Turtlebot3 robot maintains a constant linear speed of 0.15 m/s. However, its angular speed varies depending on the action taken by the robot, as outlined in Table 1.

Table 1 provides the corresponding angular velocities for each action performed by Turtlebot3. These angular velocities, measured in radians per second, determine the rotational movement of the robot. The values in the table allow the robot to effectively navigate its environment by adjusting its orientation based on the desired action.

By associating specific angular velocities with each action, the control system ensures that the Turtlebot3 responds appropriately to achieve its intended trajectory and successfully avoid obstacles.

Table 1. Relationship between actions and angular velocities.

Actions	Angular Velocity (rad/s)	Additional Information
0	−1.5	High left turn
1	−0.75	Moderate left turn
2	0	No rotation (straight)
3	0.75	Moderate right turn
4	1.5	High right turn

5.3. Reward System for the DQN Agent

The reward mechanism plays a crucial role in shaping the behavior of the DQN agent by providing feedback from the environment after each action. In supervised learning, the objective is typically to maximize the reward; however, depending on the specific action performed, the reward can also be negative.

Various alternatives exist for designing the reward function, each with its own criteria and considerations. In this study, the reward function is tailored to incentivize the Turtlebot3 Burger mobile robot to achieve its goals while avoiding collisions with obstacles.

Upon successfully reaching the goal, the robot receives the maximum positive reward, while colliding with obstacles incurs the maximum negative reward.

The overall reward R for the Turtlebot3 Burger's actions is determined by multiplying the reward for distance R_d and the reward for orientation R_θ, as depicted in Equation (9). This composite reward encourages the robot to optimize both its trajectory towards the goal and its alignment with the desired orientation, leading to effective and efficient navigation in the environment.

$$R = R_d \cdot R_\theta \tag{9}$$

5.3.1. Reward Based on Heading Alignment

The orientation-based reward mechanism focuses on the alignment between the robot and its goal, as illustrated in Figure 9. The heading, representing the angle between the robot's current orientation and the desired goal direction, plays a crucial role in determining the reward. A small heading value indicates that the robot is closely aligned with the goal orientation, resulting in a positive reward for the agent. Conversely, as the heading value increases, indicating a deviation from the desired direction, a negative reward is assigned to discourage the robot from moving further away from the intended path.

By incorporating this orientation-based reward into the reinforcement learning framework, the agent is encouraged to continually adjust its heading towards the goal, improving its ability to navigate effectively and converge to the desired orientation.

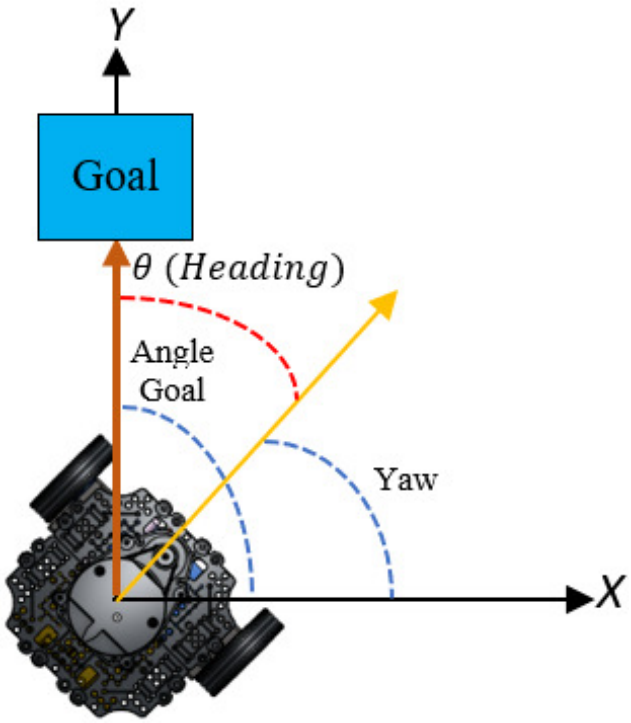

Figure 9. Visual depiction of heading relationship between robot and goal.

The heading reward plays a crucial role in evaluating the alignment between the mobile robot and its intended goal. It is an essential component of the reinforcement learning framework employed by the agent. The reward calculation takes into account two significant factors: the angle between the robot's orientation and the desired heading, and the number of available actions that the robot can perform.

The primary purpose of the heading reward is to guide the agent towards making correct decisions and actions. By considering the angle in radians, the reward function assigns a higher value when the robot's orientation is closer to the desired heading (near zero degrees). In such cases, the agent receives a positive reward, reinforcing its successful

alignment with the goal. Conversely, as the angle increases, indicating a deviation from the desired direction, the reward decreases proportionally. This negative reward signals to the agent that it needs to adjust its heading to achieve better alignment.

By incorporating the angle parameter and accounting for the number of actions, the heading reward provides an effective mechanism for training the mobile robot to navigate towards its goal with improved accuracy and precision.

$$R_\theta = 5/\theta \tag{10}$$

For the calculation of the heading reward, the restrictions shown in (11)–(13) are defined.

$$if\ -\frac{1}{2}\pi < \theta < \frac{1}{2}\pi, \tag{11}$$

$$R_\theta \geq 0 \tag{12}$$

$$else\ R_\theta < 0 \tag{13}$$

where θ represents the heading; if this value is within the 180 degrees involved within the 5 actions established in Figure 8, the reward will be greater than or equal to zero, otherwise it would mean that the robot is moving away, so the reward would be negative or less than zero.

5.3.2. Quantifying Spatial Proximity: The Role of Linear Distance Reward

The linear distance reward plays a crucial role in assessing the proximity of the mobile robot to its goal. It involves calculating the distance rate D_R by considering both the current distance D_c and the absolute distance D_a.

The current distance refers to the real-time spatial separation between the robot and the goal. As the robot moves and the goal's position changes throughout the training process, this distance continuously varies. To calculate the current distance, trigonometric analysis is required, taking into account the distances of the robot and the goal with respect to the main plane of Gazebo. Figure 10 illustrates this geometric relationship.

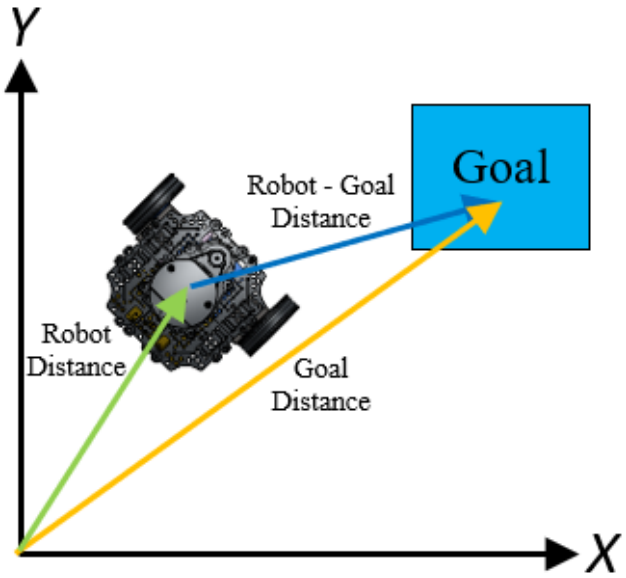

Figure 10. Real-time geometric analysis: calculating distances of robot and goal.

On the other hand, the absolute distance represents the spatial span between the starting point of the Gazebo reference frame and the goal, specifically at the initiation of a

training scenario. It serves as a fixed reference for assessing the robot's progress toward the goal.

By considering both the current distance and the absolute distance, the linear distance reward enables the agent to effectively gauge its proximity to the goal. This reward component provides valuable feedback for the agent's decision-making process, and aids in optimizing its navigation strategy toward achieving the desired objective.

The distance rate is a crucial factor in determining the reward for the robot's progress towards the goal. This rate is calculated using the equation presented in Equation (14). The purpose of this calculation is to ensure that as the robot approaches the goal, the reward value increases, indicating successful progress. However, if the robot gets too far away from the goal, the reward score is reduced to one, reflecting the need to prioritize proximity to the target. This approach encourages the robot to navigate efficiently and prioritize reaching the goal within a reasonable distance.

$$R_d = 2^{(D_a/D_c)} \tag{14}$$

The constraints in Equations (15)–(17) are established for the calculation of the reward based on the distance from the robot to the goal.

$$if\ D_a > D_c, \tag{15}$$

$$R_d > 2 \tag{16}$$

$$else\ 1 < R_d \leq 2 \tag{17}$$

If the absolute or initial distance from the goal is greater than the current distance, the value of the reward will always be greater than 2, because this is the base; however, the further the robot moves, the reward will be between 1 and 2.

Finally, the reward function for the network is presented in (18); three conditions are defined for the system, which will allow for adequate automatic control of the robot's route to the goal.

$$R = \begin{cases} 200, & \text{if goal is reached} \\ -100, & \text{if collision occurs} \\ R_d \cdot R_\theta, & else \end{cases} \tag{18}$$

5.4. Hyperparameters for the Simulation

Hyperparameters are those configurable elements that control the behavior of the neural network; in Table 2, the name, value, and a brief description of the defined parameters that the agent must follow are presented.

Table 2. Hyperparameters.

Hyperparameter	Value	Description
memory	1,000,000	Size of the replay memory
train start	64	When the memory is greater than 64 the training begins
batch size	64	Training sample set
epsilon min	0.05	Minimum value of epsilon
epsilon decay	0.99	Epsilon reduction rate at the end of an episode
epsilon	1.0	Probability of choosing a random action
learning rate	0.00025	Learning speed
discount factor	0.99	Future event value reduction factor based on distance
target update	2000	Update rate of target network
episode step	6000	Duration of an episode

5.5. ROS System

For the proposed implementation of the case study, a four-node ROS system was developed, as shown in Figure 11; by employing a reinforcement learning system utilizing Deep Q-Network (DQN), it becomes possible to control the simulated Turtlebot3 Burger mobile robot. The communication between nodes in this system relies on the exchange of messages through publishing and subscribing to various topics. This mechanism facilitates the seamless flow of information and enables effective coordination and control of the robot's actions within the simulated environment. In order for the neural network designed in the controller node to receive the necessary information from the environment and be able to send the speed commands to the robot, the ROS API describes how the messages will be published and read by the agent, which is the core of the control system; however, the ROS master is initialized by the simulated environment.

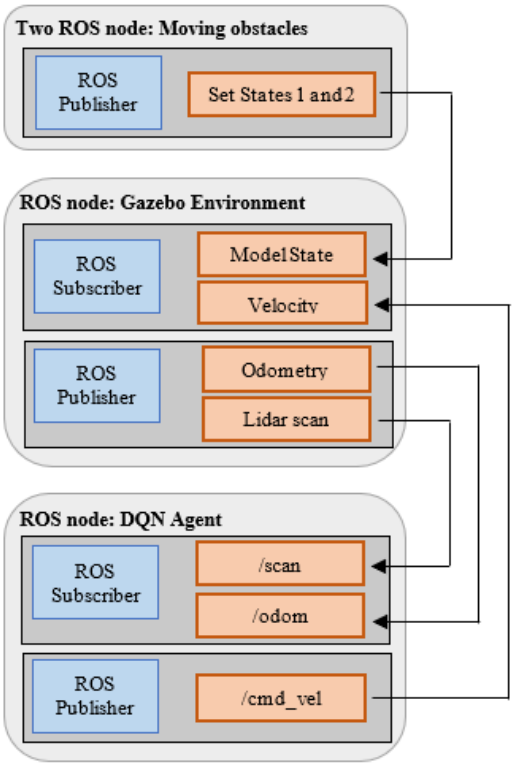

Figure 11. ROS architecture.

In addition to executing the DQN neural network node with its respective hyperparameters, it transmits the robot's speed commands to another node, which contains the simulated environment; the latter also receives information from two more nodes about the combination of movements for the moving obstacles, and sends the parameters of odometry and scanning of the Lidar sensor to the network.

The ROS rqt graph tool is used to obtain a diagram of the state of communication in real time between the nodes of the system and how they send and receive information about the topics through the messages; it allows one to graphically observe the flow of information. Figure 12 shows the graph obtained as a result of the system proposed as a case study.

The Turtlebot3 Burger receives orders through the topic /cmd_vel, and according to its operation, it is made up of two vectors: one of linear velocity and the other of angular, where one indicates how fast it should move forward and the other how fast it should turn.

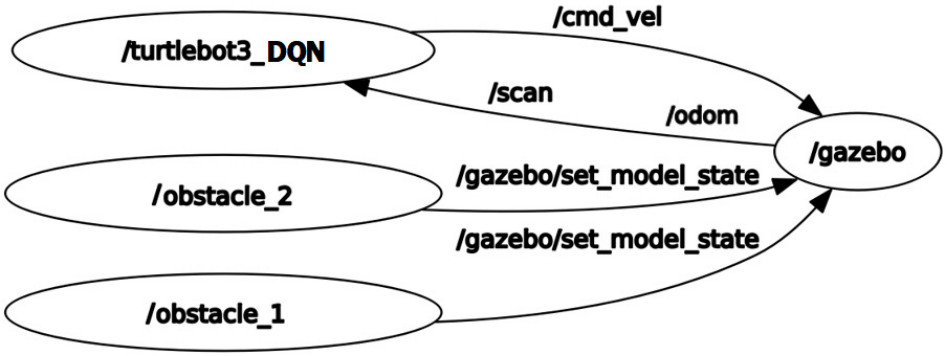

Figure 12. Information flow analysis: visualizing node communication with Rqt graph.

5.6. Network Structure

For reinforcement learning based on DQN, the Tensorflow library was used, because it is an open-source platform that provides flexible tools for machine learning; its API provides an adaptable architecture for the development of models and applications. For the programming of the network layers, Keras was used, which minimizes the amount of code used for the development of control systems.

The architecture of the network model employed in the system is designed in a sequential manner. The first layer is a convolutional layer, which serves the purpose of extracting relevant information from the input data. The subsequent layers are fully connected, denoted as Dense, indicating that the neurons are interconnected. To mitigate issues related to saturation of negative values or overly long positive values, the rectified linear unit (ReLU) activation function is applied to both the hidden layers and the initial layer.

To prevent overfitting, a Dropout layer was incorporated into the model. This layer randomly deactivates 20% of the input units during each training step, thereby enhancing the generalization capability of the network. Finally, the last layer employs a linear activation function. This network architecture, depicted in Figure 13, enables effective information processing and decision making within the reinforcement learning framework.

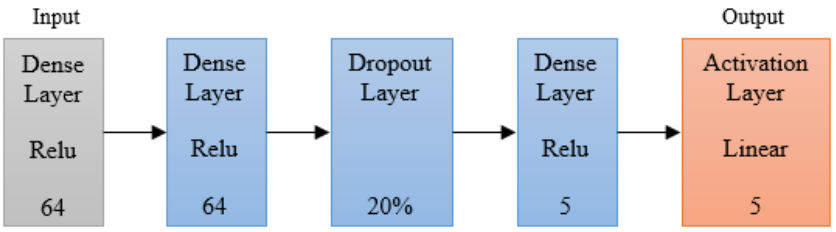

Figure 13. Enhancing decision making in RL: architecture of the Q network model.

The input layer consists of 64 neurons and 13 inputs corresponding to the size of the state, previously defined in Section 5.1, while the output layer has 5 neurons due to the number of actions established in Section 5.2. In addition, the activation of this is linear because it is a regression and not a classification.

The model is developed by three main classes (see Figure 14). (i) The first class is called Components, and within it are the layers of the neural network and its compiler; this class corresponds to the global operation of the model and contains all the programming in python. (ii) The second class is called Parameters, and within it, all those variables that are related within the model are defined, such as the Qvalue, state, action, and reward. Finally, (iii) the third class is called Monitoring, and it allows the user to see graphically the operation of the network through real-time diagrams of the Qvalue and the reward.

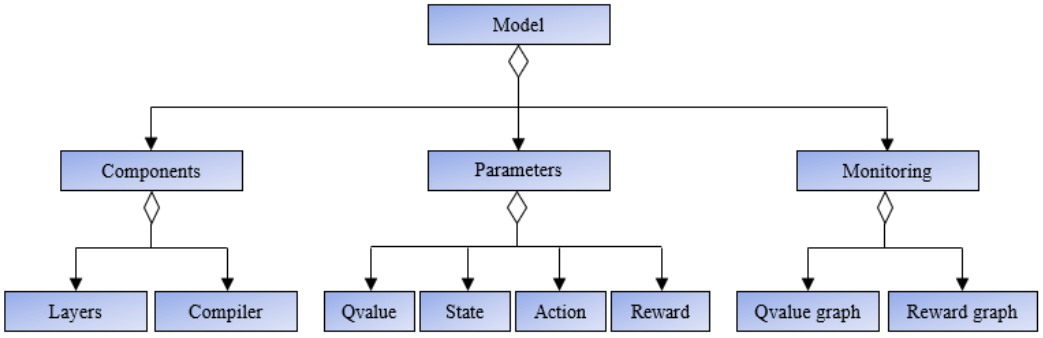

Figure 14. Comprehensive model structure: UML class diagram visualizing neural network layers and monitoring capabilities.

5.7. Experiment Configuration of the DQN Parameters

In the experimental setup, the MSE (mean square error) loss function was chosen based on the available data and the type of network utilized. This loss function is commonly employed in both simulated and real systems, particularly in regression tasks where the descending gradient method outperforms the least-squares approach, providing more efficient and optimal solutions in less time.

The MSE loss function is well suited for various models, ranging from simple to complex ones. It penalizes predictions that deviate significantly from the actual values, emphasizing the importance of minimizing large deviations.

The experiment was configured to run for 400 episodes, allowing for a comprehensive evaluation of the system's performance. Additionally, three different optimization algorithms were defined, taking into account the previously established loss function and the calculation of inferred gradients. These optimization algorithms play a crucial role in adjusting the model's parameters to minimize the MSE loss and enhance overall learning performance. To evaluate the algorithm that works best on the data, the one that obtains the best results in the same amount of time will be determined, because the correct choice of the algorithm will determine if the model works accurately in a few hours or in several.

In Figure 15, the resulting diagram of the experiment configured with the MSE loss function and Adam optimizer is presented. A large variation of the reward between 800 and −400 is observed; in addition, a positive trend can be observed. However, the result is poor, because the robot after the total of episodes still collides and vaguely searches for the goal.

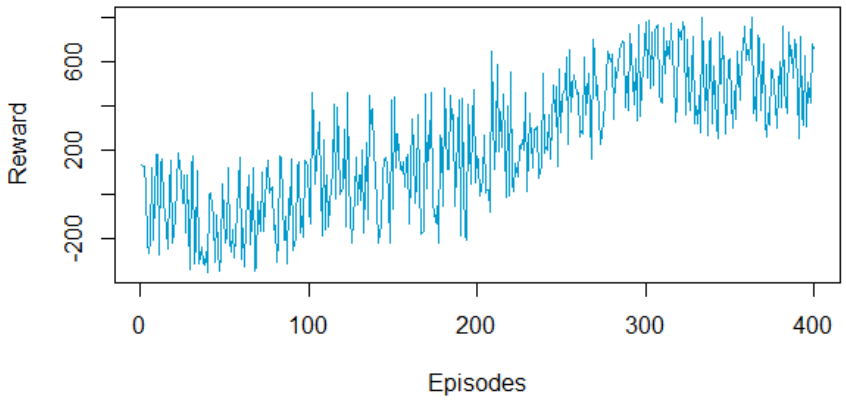

Figure 15. MSE function and Adam optimizer.

In Figure 16, the resulting graph of the experiment configured with the MSE loss function and RMSprop optimizer is shown. It is observed that the variation of the reward is reduced compared to the previous case and is between 6000 and −400, this means that the robot positions itself correctly and finds the goal. This method is usually the most used, because the predictions that are too far away are reduced by the calculation.

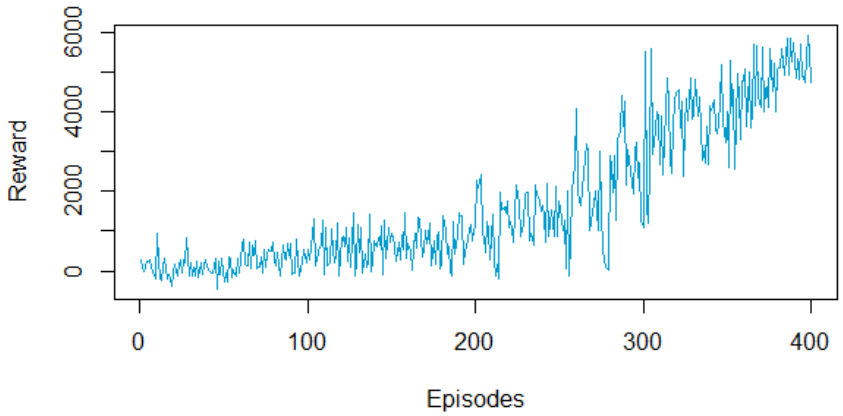

Figure 16. MSE function and RMSprop optimizer.

In Figure 17, the resulting graph of the experiment configured with the MSE loss function and SGD optimizer is presented. It is observed that the variation of the reward is the smallest of the three experiments; however, it is between 2000 and -3000, which means that the robot in its first training episodes collides and moves too far from the goal. Although a growing trend has been observed, this method is usually the least used due to its low precision and the need for a greater number of samples to obtain an adequate result.

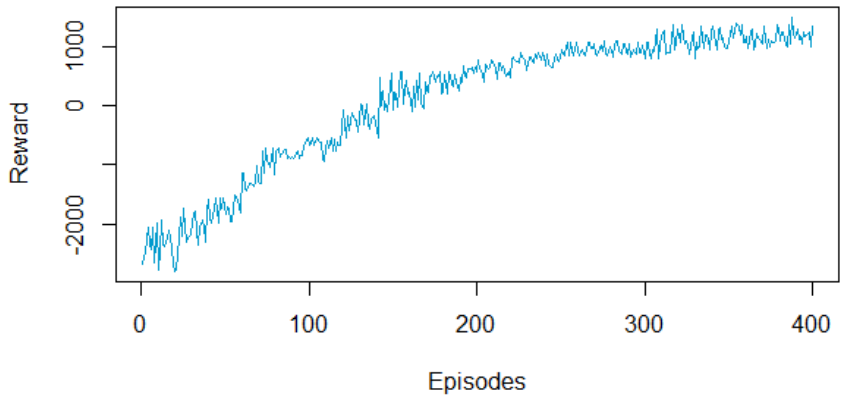

Figure 17. MSE function and SGD optimizer.

6. Discussion of Results

The control algorithm was evaluated in a three-dimensional environment using the Turtlebot3 Burger mobile robot. The training process involved 400 iterations where the robot learned to navigate towards a goal while avoiding obstacles. The level of reward obtained for each executed action was measured, reflecting the effectiveness of the trained network.

The rewards obtained with different optimizers were collected from the DQN network based on reinforcement learning. Figure 18 illustrates the reward trends, which were represented using a smoothing technique incorporating weighted adjustments derived from the least-squares method.

Among the optimization algorithms tested, RMSprop demonstrated the best results. When utilizing RMSprop, the robot exhibited a higher success rate in reaching the goal and effectively avoided obstacles. The reward values steadily increased over time, indicating that the network was functioning optimally and the algorithm was learning from its experiences.

Throughout the training process, the goal positions were randomly generated within the arena, ensuring the robot encountered different scenarios. The linear speed of the robot remained constant at $v = 0.15, \mathrm{m/s}$, while the angular speed varied based on the chosen action, as presented in Table 1. Initially, the robot frequently collided with obstacles and struggled to reach the goal. However, as the training progressed through multiple episodes,

the agent learned how to interact with the environment more effectively. It autonomously determined its trajectory, avoiding collisions, and achieved the goal by attaining positive scores.

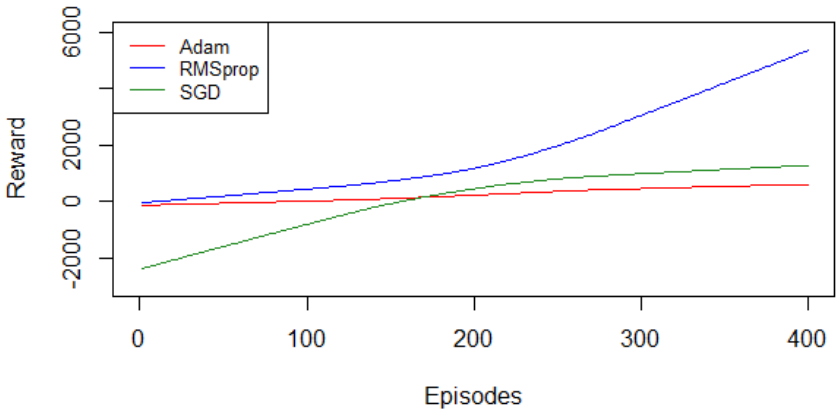

Figure 18. Comparison between the optimizers Adam, RMSprop, and SGD.

The experimentation conducted served to validate the effectiveness of the model. By successfully training the robot to navigate and make decisions based on reinforcement learning, the study demonstrated the potential and applicability of the developed approach.

The performance of the simulation is an important factor during the training of the model, because if the hardware has adequate resources, then less time will be needed to reach the number of episodes proposed; in turn, obtaining information from the environment will be optimal, allowing the algorithm to perform an adequate control.

6.1. DQN Scenarios Validation

To evaluate the effectiveness of the DQN algorithm for path determination, a series of experiments were conducted using a Turtlebot3 Burger in a Gazebo simulation environment. In each scenario, the robot was placed in the same starting position, while a cube represented the goal to be reached. The Lidar sensor obtained information about the surrounding obstacles and walls.

The first scenario was a simple one without any obstacles. The DQN algorithm was run for a total of 200 episodes, during which the robot was able to reach the goal successfully. Initially, the robot collided with the environment, resulting in a negative reward. However, as the algorithm learned, the reward level increased to almost 800 points, showing a clear increasing trend. This indicates that the DQN algorithm is capable of learning from past experiences, resulting in a robot that can autonomously navigate toward the goal while avoiding collisions. See Figure 19.

The success of the DQN algorithm in the first scenario highlights its ability to learn from past experiences to improve future decision making. During each episode, the DQN algorithm collects sensory inputs from the environment, such as Lidar scans, and generates actions to control the robot's movements towards the goal. As the algorithm interacts with the environment, it receives rewards that are used to update the Q-value function, which is used to select the best actions in a given state.

In the case of the first scenario, the negative reward signal received from the robot's initial collision with the environment helped the algorithm learn to avoid collisions in the future shown in Figure 20. The increasing trend in the reward level throughout the episodes shows that the algorithm was able to learn an effective policy for navigating toward the goal while avoiding collisions.

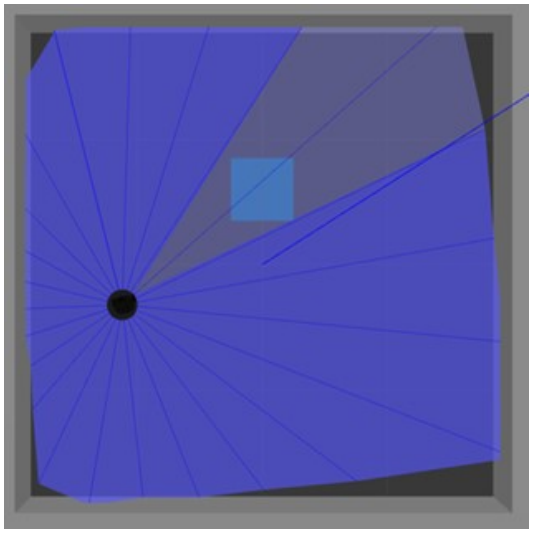

Figure 19. Learning and performance improvement in the initial scenario: validation results.

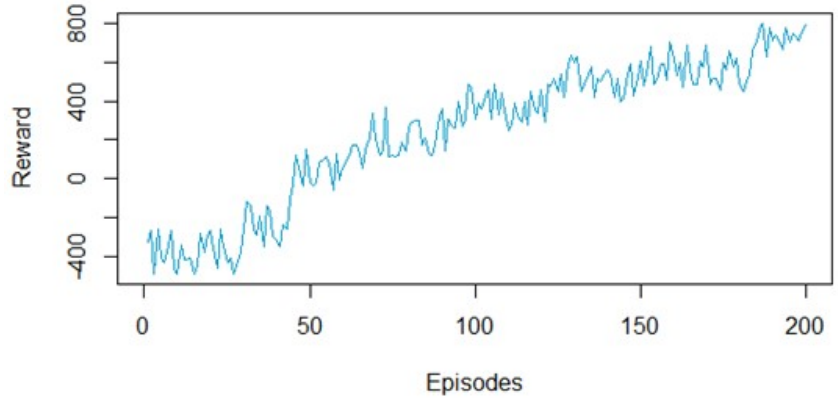

Figure 20. DQN algorithm reward evolution for the first validation scenario.

In the second scenario (See Figure 21), the environment was made more complex by adding four static obstacles, which made navigation for the robot more challenging. However, after 300 episodes, it can be concluded that the DQN algorithm enables the robot to learn effectively. Throughout the training process, the robot learned to avoid collisions and choose the correct path toward the goal.

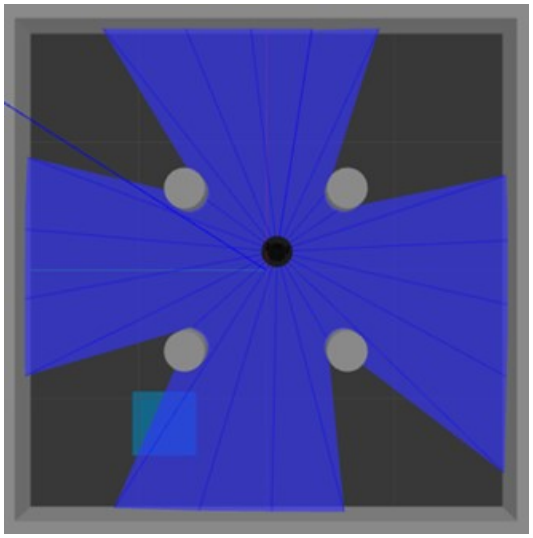

Figure 21. Effective learning in the second scenario: validation results with obstacles.

As with any training process, the robot initially received negative rewards when it chose the wrong path or collided with an obstacle. However, the trend was positive, and the robot's score increased to nearly 1500 positive points. This validates the successful performance of the DQN algorithm in the second scenario shown in Figure 22.

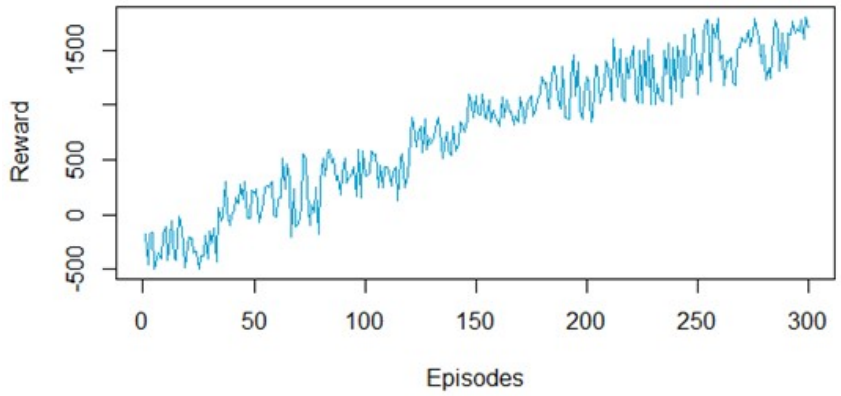

Figure 22. DQN algorithm reward evolution for second validation scenario.

In the third scenario, shown in Figure 23, the same obstacles as in the previous scenario were added, but now they are mobile and follow a circular trajectory around the center of the environment. These conditions make it more challenging for the robot to learn, as there is a higher probability of collision, and it must choose the correct path to avoid low scores.

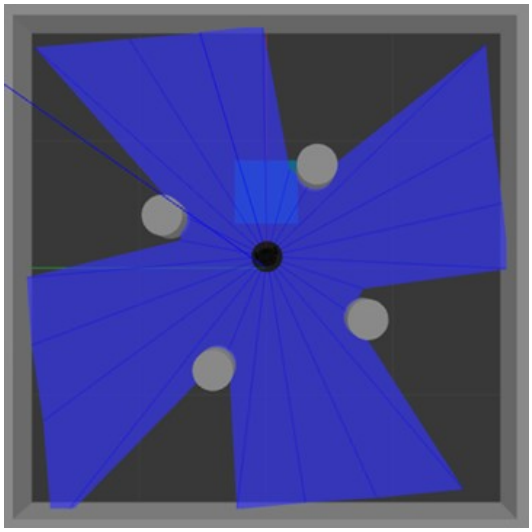

Figure 23. Navigating obstacles in the third scenario: evaluating robot's adaptability.

After 400 episodes, it can be concluded that the DQN algorithm allowed the robot to learn effectively in this scenario as well. The robot was able to avoid collisions, resulting in a score of over 3000 points, and the trend was positive throughout the training process. Initially, the robot received negative rewards (See Figure 24), indicating collisions in the early stages of training. However, as the robot learned, the trend became increasingly positive. The DQN algorithm was still able to learn from past experiences and adapt to the changing environment, resulting in successful path determination. The fact that the robot was able to achieve a higher score in the third scenario, despite the added difficulty, shows the potential of the DQN algorithm to handle complex and dynamic environments.

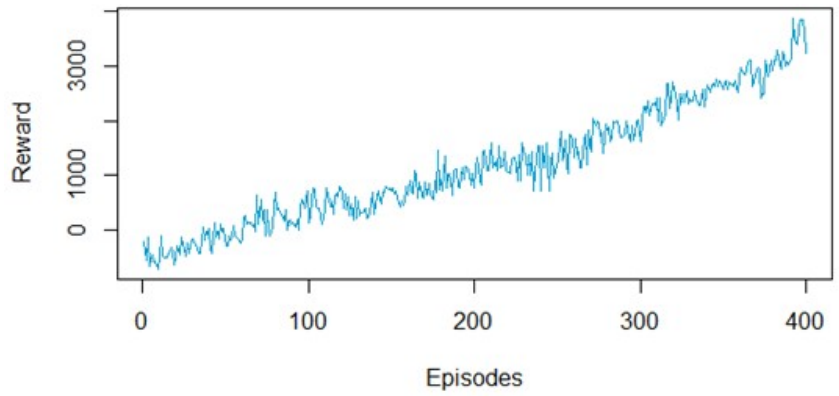

Figure 24. DQN algorithm reward evolution for third validation scenario.

In the fourth and final scenario, a combination of static and mobile obstacles was introduced, making the path determination task much more challenging for the robot. Despite this added difficulty, the DQN algorithm was able to learn and adapt to the changing environment, resulting in a high reward score of almost 6000 points. This indicates that the DQN algorithm is capable of handling complex and dynamic environments in autonomous navigation tasks. See Figure 25.

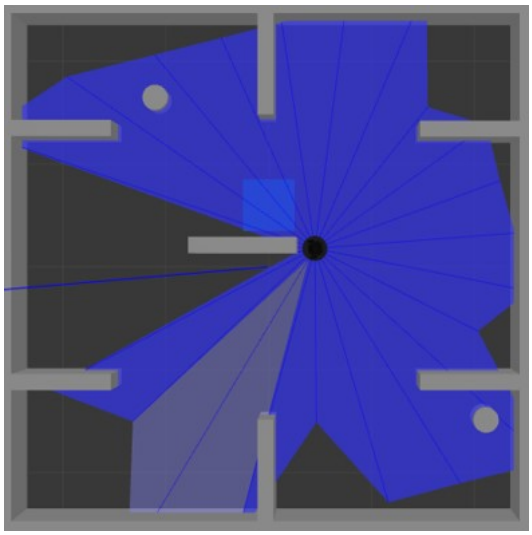

Figure 25. Dynamic and static obstacles: evaluating robot's adaptability in Scenario 4.

Based on the results of all four scenarios, it can be concluded that the DQN algorithm is a promising approach for autonomous path determination in robotics. The algorithm demonstrated the ability to learn from past experiences and adapt to changing environments, while successfully avoiding obstacles and reaching the goal. However, it is worth noting that the algorithm's performance may vary depending on the complexity of the environment and the number and type of obstacles present. Therefore, further experimentation and optimization are necessary to fully assess the capabilities of the DQN algorithm in autonomous navigation tasks.

The main challenge of scenario 4 is that it involves a combination of both static and dynamic obstacles. This makes the path to the goal much more complex for the robot and requires it to navigate through a more cluttered environment. As a result, the robot frequently collides with obstacles at the beginning of the training, leading to negative rewards. However, as the robot continues to interact with the environment, it learns to navigate more efficiently towards the goal, resulting in a significant increase in reward points. See Figure 26.

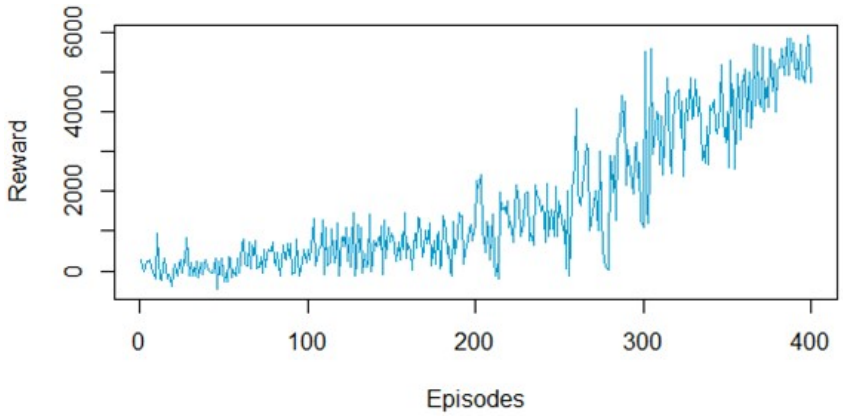

Figure 26. DQN Algorithm Reward Evolution for fourth validation scenario.

In the fourth scenario previously described, the environment posed a greater challenge for the robot due to a combination of static and dynamic obstacles. To overcome this challenge, the use of the Lidar sensor was implemented to obtain information about the surroundings of the robot. As shown in the Figure 27, the Lidar sensor emitted a beam of light that was reflected on the walls, allowing the robot to map the entire environment. This map was stored and used for later visualization in Rviz, which is a tool used for graphing two sources of information: odometry readings of the encoders connected to the wheels and information from sensors such as Lidar or Kinect. By using this environment recognition method, the control algorithm based on autonomous learning was able to locate itself in the environment and identify the obstacles present in order to avoid collisions.

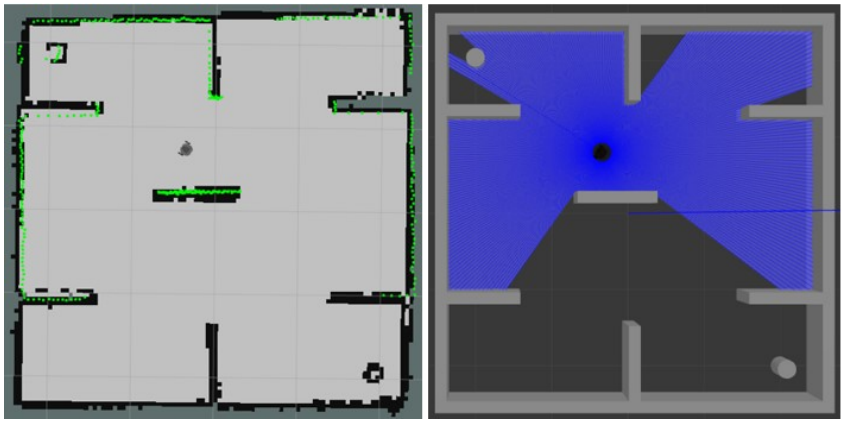

Figure 27. Spatial mapping for obstacle avoidance: Lidar-based environment perception in Scenario 4.

6.2. Discussion

In Figure 28, we present the performance of the Gazebo real-time factor during the training period, which was set to a duration of 8 h. The real-time factor is a measure of how close the simulation runs to real time. In our experiments, we observed that the real-time factor varied between 80% and 100%. Although this range represents a reasonably adequate simulation, it is important to note that it is not optimal.

It is worth emphasizing that the quality of the communication carried out by the Robot Operating System (ROS) remained unaffected throughout the training process. ROS is known for its robust and reliable communication infrastructure, ensuring efficient data exchange between nodes. However, it is important to highlight that the high consumption of resources, particularly due to the large number of nodes involved and the processing requirements of the DQN network, had an impact on the real-time factor.

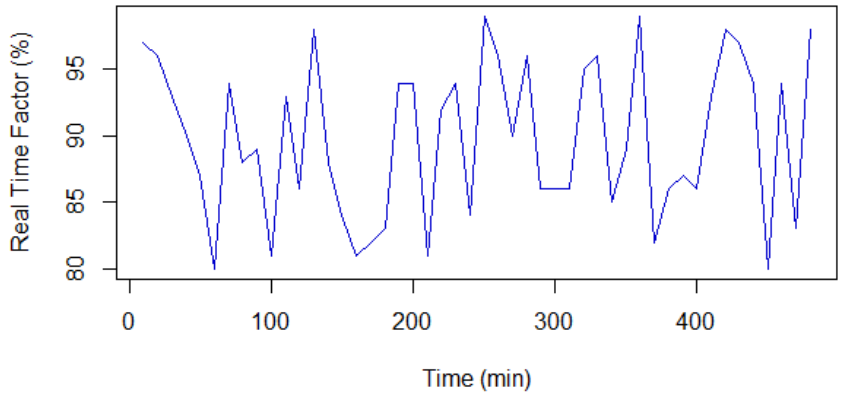

Figure 28. Gazebo real-time factor.

The variations in the real-time factor can be attributed to the computational demands of the simulation environment, the complexity of the neural network, and the overall system performance. As the number of nodes and the computational load of the DQN network increase, the resources required for processing also increase. This, in turn, can affect the real-time factor of the simulation.

While achieving a higher real-time factor would be ideal for a more seamless and responsive simulation, it is important to consider the trade-offs between computational resources, network complexity, and simulation performance. In our study, we aimed to strike a balance between these factors to ensure meaningful results without compromising the overall functionality of the system.

By presenting the performance of the real-time factor, we provide insights into the computational demands and resource consumption associated with our experimental setup. This information adds an important perspective to the evaluation of our approach and highlights the considerations and limitations of our simulation environment.

Overall, understanding the impact of resource consumption on the real-time factor enhances our understanding of the system's behavior and performance, and it allows us to make informed decisions regarding the scalability and efficiency of our proposed approach.

As presented in [98], the structure of the network is composed of three convolutional layers due to the nature of the input information. In this investigation, the processing of four image frames is used for the sampling of the environment and five types of actions are used because the work is performed by the same robot as in this case of study. The design of a convolutional neural network (CNN) demands a large amount of resources due to the use of filters and the size of the kernel; in the same way, the use of a double DQN dueling network is used for the estimation of the state and the actions, which represents two separate layers of 512 nodes each, unlike the current work, which uses two separate layers of 64 nodes in its DQN network due to the size of the input data. However, despite the different conditions within each system, the results of the simulation are positive in both works, and indicate that the network has the learning capacity, since it is mentioned that at the beginning, the robot collides with the obstacles, and with the passing of the training, it learns to avoid them and to go more precisely to the target autonomously. The most notable differences between both control systems are the computational requirements necessary for the network, the amount of programming for a correct operation, and the simulation time demanded. The present proposal stands out for being a lighter control and adaptable to machines not so powerful.

In [99], a system based on DQN and reinforcement learning is designed, for which a two-dimensional simulated environment is used, unlike the present case study, which uses a three-dimensional environment. Both works use relative distances and angles for the location of the robot, and in the same way, the objective of the two works is that the robot reaches the goal. A simultaneous localization and modeling (SLAM) system is added in the comparative investigation, which allows for the estimation of the position of the robot based on the mapping of the environment. In addition, a penalty for localization

is introduced to the reward, so that the behavior of the agent can be regulated to control that the robot does not enter areas with unobserved characteristics. In the implementation proposal, SLAM is not considered, for which the robot does not use the mapping of the area but learns to avoid obstacles through the relative distance between both and the negative reward in collision. The present work is not affected by sudden changes in the obstacles in the area; however, in the previously mentioned research, a change would lead to changing the sampling conditions defined in the first training scenarios.

In [54,100], a DQN network is established for the approximation of the values of the state and the action of a mobile robot. The system is composed of three phases: initially, the acquisition and processing of the image are carried out, for which two convolutional layers are defined, then the model is programmed to maximize the Q value in each training, and finally the DQN network selects the best possible action using two fully connected layers. The proposal of this work is based on developing a path planning system, for which an SGD optimization algorithm is used. The results of the experiment mention that the system is successful in reaching the goal because the score grows with each training scenario; however, it does not operate properly in the presence of mobile obstacles. The case study of our work is based on exploration, so that the DQN network learns to react to possible unexpected events and select based on the results above the best route to reach the goal. The most optimal solution can be determined through exploration; however, exceeding it could reduce the performance in the path planning system and affect the learning speed. This does not occur in the implementation proposal, where the greater the amount of training, the better the balance of the system.

In recent years, there has been a growing interest in using reinforcement learning (RL) techniques to train robots to perform complex tasks such as path planning and navigation. One popular RL algorithm for these tasks is Deep Q-Networks (DQN), which is a deep neural network that uses a Q-learning algorithm to learn an optimal policy for the robot to follow.

The main findings of the present study suggest that the DQN algorithm can successfully be used for path detection in robotics platforms. The algorithm was trained on a dataset of path sequences and tested on both seen and unseen environments, achieving high accuracy rates in both cases. Moreover, the algorithm demonstrated its ability to adapt to changing environmental conditions and to generalize across different robotic platforms.

The comparison of our findings with other studies in the field suggests that the DQN algorithm can be a competitive approach for path detection in robotics platforms. Several previous studies have used different algorithms and techniques for path detection, such as SLAM and reinforcement learning, and achieved varying degrees of success. However, our study demonstrates that the DQN algorithm can achieve high accuracy rates in path detection while being computationally efficient and adaptable to changing conditions.

The implications of our findings are significant for the field of robotics, as accurate path detection is essential for the development of autonomous robotic systems. The DQN algorithm provides a promising approach for path detection, as it can be trained on a dataset of path sequences and can generalize to different environments and robotic platforms. Moreover, the computational efficiency of the algorithm makes it a practical option for real-world applications, where real-time processing is crucial. However, further research is needed to optimize the algorithm for complex environments and to investigate its performance in real-world scenarios.

The strengths of the DQN algorithm for path detection in robotics platforms include its ability to generalize across different environments and robotic platforms, its computational efficiency, and its ease of implementation. However, the limitations of the algorithm include the need for a large dataset of path sequences for training, the risk of overfitting to specific environments or paths, and the need for optimization for complex environments. Additionally, the performance of the algorithm may be affected by the quality of the sensor data and the accuracy of the robot's control system. Despite these limitations, the DQN

algorithm provides a promising approach for path detection in robotics platforms, and can be further improved through ongoing research and development.

7. Conclusions and Future Work

The article presents a novel control algorithm based on a DQN network, which has been designed as a node through the ROS API. The architecture proposed for this case study integrates different open source systems that are represented as the agent and the environment, and they are interrelated through states, actions, and rewards to learn through reinforcement. The experimentation conducted in this research allows for the optimization of the functional structure of the neural network by correcting some of its parameters. It is determined that the RMSprop optimizer and MSE loss function are the most appropriate for the calculation of regression, which helps the robot to find the goal more accurately and avoid collisions better, leading to better performance at the reward level.

The findings of the present study contribute to the development of robotics platforms by proposing a new control algorithm that shows better performance compared to other existing approaches. The proposed DQN-based algorithm provides a promising solution for the control of robotic platforms by learning from interactions with the environment. Collisions are better, thus giving better performance at the reward level.

While DQN algorithms have shown promising results for path navigation and obstacle avoidance in robotic platforms, there are still some limitations and gaps in their use. One of the main challenges is the high computational resources required for training DQN networks, which limits their real-time application on resource-limited robotic platforms. Additionally, DQN algorithms require a significant amount of training data, which can be time-consuming and expensive to collect in real-world scenarios.

Another challenge is the sensitivity of DQN networks to changes in the environment or task, which can lead to overfitting or poor generalization. This can make it difficult to transfer the learned policy to different environments or tasks, which limits the versatility of the approach.

Furthermore, DQN algorithms often require a carefully selected set of hyperparameters, and tuning these parameters can be a challenging task. Moreover, the optimal configuration of hyperparameters for a specific task may not generalize well to other tasks, which makes it difficult to use DQN algorithms for a wide range of applications. Finally, while DQN algorithms have shown impressive results for some tasks, there is still a need for more comparative studies with other reinforcement learning algorithms to determine their performance advantages and limitations in different applications.

Further research should focus on exploring the scalability and computational efficiency of the proposed algorithm in more complex environments and compare its performance with other approaches to identify the most effective solution for path detection in robotics platforms.

Author Contributions: Conceptualization, J.E.-N. and M.V.G.; methodology, P.A.; software, J.E.-N., G.C. and C.A.G.; validation, M.V.G.; investigation, J.E.-N., E.J. and M.V.G.; writing–original draft preparation, J.E.-N. and M.V.G.; writing–review and editing, J.E.-N., C.A.G. and M.V.G.; visualization, P.A.; supervision, M.V.G.; funding acquisition, G.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Not applicable.

Acknowledgments: This work was supported by Universidad Tecnica de Ambato (UTA) and their Research and Development Department (DIDE) under project PFISEI32.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Zong, L.; Yu, Y.; Wang, J.; Liu, P.; Feng, W.; Dai, X.; Chen, L.; Gunawan, C.; Jimmy Yun, S.; Amal, R.; et al. Oxygen-vacancy-rich molybdenum carbide MXene nanonetworks for ultrasound-triggered and capturing-enhanced sonocatalytic bacteria eradication. *Biomaterials* **2023**, *296*, 122074. [CrossRef] [PubMed]
2. Hamid, M.S.R.A.; Masrom, N.R.; Mazlan, N.A.B. The key factors of the industrial revolution 4.0 in the Malaysian smart manufacturing context. *Int. J. Asian Bus. Inf. Manag.* **2022**, *13*, 1–19. [CrossRef]
3. Zou, T.; Situ, W.; Yang, W.; Zeng, W.; Wang, Y. A Method for Long-Term Target Anti-Interference Tracking Combining Deep Learning and CKF for LARS Tracking and Capturing. *Remote Sens.* **2023**, *15*, 748. [CrossRef]
4. Ochoa-Zezzatti, A.; Oliva, D. Impact of Industry 4.0: Improving Hybrid Laser-Arc Welding with Big Data for Subsequent Functionality in Underwater Welding. In *Studies in Systems, Decision and Control*; Springer: Berlin/Heidelberg, Germany, 2022; Volume 347, pp. 87–94. [CrossRef]
5. de Castro, G.; Pinto, M.; Biundini, I.; Melo, A.; Marcato, A.; Haddad, D. Dynamic Path Planning Based on Neural Networks for Aerial Inspection. *J. Control. Autom. Electr. Syst.* **2023**, *34*, 85–105. [CrossRef]
6. Mizoguchi, Y.; Hamada, D.; Fukuda, R.; Inniyaka, I.; Kuwata, K.; Nishimuta, K.; Sugino, A.; Tanaka, R.; Yoshiki, T.; Nishida, Y.; et al. *Image-based navigation of Small-size Autonomous Underwater Vehicle "Kyubic" in International Underwater Robot Competition*; ALife Robotics Corporation Ltd.: Oita, Japan, 2023; pp. 473–476.
7. Pavel, M.D.; Roșioru, S.; Arghira, N.; Stamatescu, G. Control ofÂ Open Mobile Robotic Platform Using Deep Reinforcement Learning. In *Service Oriented, Holonic and Multi-Agent Manufacturing Systems for Industry of the Future: Proceedings of SOHOMA 2022*; Springer: Berlin/Heidelberg, Germany, 2023; Volume 1083, pp. 368–379. [CrossRef]
8. Cerquitelli, T.; Ventura, F.; Apiletti, D.; Baralis, E.; Macii, E.; Poncino, M. Enhancing manufacturing intelligence through an unsupervised data-driven methodology for cyclic industrial processes. *Expert Syst. Appl.* **2021**, *182*, 115269. [CrossRef]
9. Cruz Ulloa, C.; Garcia, M.; del Cerro, J.; Barrientos, A. Deep Learning for Victims Detection from Virtual and Real Search and Rescue Environments. In *ROBOT2022: Fifth Iberian Robotics Conference: Advances in Robotics*; Springer: Berlin/Heidelberg, Germany, 2023; Volume 590, pp. 3–13. [CrossRef]
10. Cordeiro, A.; Rocha, L.; Costa, C.; Silva, M. Object Segmentation for Bin Picking Using Deep Learning. In *ROBOT2022: Fifth Iberian Robotics Conference: Advances in Robotics*; Lecture Notes in Networks and Systems; Springer: Berlin/Heidelberg, Germany, 2023; Volume 590, pp. 53–66. [CrossRef]
11. Rodrigues, N.; Sousa, A.; Reis, L.; Coelho, A. Intelligent Wheelchairs Rolling in Pairs Using Reinforcement Learning. In *ROBOT2022: Fifth Iberian Robotics Conference: Advances in Robotics*; Lecture Notes in Networks and Systems; Springer: Berlin/Heidelberg, Germany, 2023; Volume 590, pp. 274–285. [CrossRef]
12. Crawford, B.; Sourki, R.; Khayyam, H.; Milani, A.S. A machine learning framework with dataset-knowledgeability pre-assessment and a local decision-boundary crispness score: An industry 4.0-based case study on composite autoclave manufacturing. *Comput. Ind.* **2021**, *132*, 103510. [CrossRef]
13. Vidal-Soroa, D.; Furelos, P.; Bellas, F.; Becerra, J. An Approach to 3D Object Detection in Real-Time for Cognitive Robotics Experiments. In *ROBOT2022: Fifth Iberian Robotics Conference: Advances in Robotics*; Lecture Notes in Networks and Systems; Springer: Berlin/Heidelberg, Germany, 2023, Volume 589, pp. 283–294. [CrossRef]
14. Lu, H.; Liu, J.; Luo, Y.; Hua, Y.; Qiu, S.; Huang, Y. An autonomous learning mobile robot using biological reward modulate STDP. *Neurocomputing* **2021**, *458*, 308–318. [CrossRef]
15. Sivaranjani, A.; Vinod, B. Artificial Potential Field Incorporated Deep-Q-Network Algorithm for Mobile Robot Path Prediction. *Intell. Autom. Soft Comput.* **2023**, *35*, 1135–1150. [CrossRef]
16. Raja, V.; Talwar, D.; Manchikanti, A.; Jha, S. Autonomous Navigation for Mobile Robots with Sensor Fusion Technology. In *Industry 4.0 and Advanced Manufacturing: Proceedings of I-4AM 2022*; Lecture Notes in Mechanical Engineering; Springer: Berlin/Heidelberg, Germany, 2023; pp. 13–23. [CrossRef]
17. Cruz Ulloa, C.; Krus, A.; Barrientos, A.; Cerro, J.; Valero, C. Robotic Fertilization in Strip Cropping using a CNN Vegetables Detection-Characterization Method. *Comput. Electron. Agric.* **2022**, *193*, 106684. [CrossRef]
18. Herr, G.; Weerakoon, L.; Yu, M.; Chopra, N. *Cardynet: Deep Learning Based Navigation for Car-Like Robots in Dynamic Environments*; American Society of Mechanical Engineers (ASME): New York, NY, USA, 2022; Volume 5. [CrossRef]
19. Jaiswal, A.; Ashutosh, K.; Rousseau, J.; Peng, Y.; Wang, Z.; Ding, Y. *RoS-KD: A Robust Stochastic Knowledge Distillation Approach for Noisy Medical Imaging*; Institute of Electrical and Electronics Engineers Inc.: Piscataway, NJ, USA, 2022; pp. 981–986. [CrossRef]
20. Chen, C.W.; Tsai, A.C.; Zhang, Y.H.; Wang, J.F. *3D Object Detection Combined with Inverse Kinematics to Achieve Robotic Arm Grasping*; Institute of Electrical and Electronics Engineers Inc.: Piscataway, NJ, USA, 2022. [CrossRef]
21. Kulkarni, J.; Pantawane, P. *Person Following Robot Based on Real Time Single Object Tracking and RGB-D Image*; Institute of Electrical and Electronics Engineers Inc.: Piscataway, NJ, USA, 2022. [CrossRef]
22. M'Sila, C.; Ayad, R.; Ait-Oufroukh, N. *Automated Foreign Object Debris Detection System Based on UAV*; Institute of Electrical and Electronics Engineers Inc.: Piscataway, NJ, USA, 2022. [CrossRef]
23. Balachandran, A.; Lal S, A.; Sreedharan, P. *Autonomous Navigation of an AMR Using Deep Reinforcement Learning in a Warehouse Environment*; Institute of Electrical and Electronics Engineers Inc.: Piscataway, NJ, USA, 2022. [CrossRef]
24. Ghodake, A.; Uttam, P.; Ahuja, B. *Accurate 6-DOF Grasp Pose Detection in Cluttered Environments Using Deep Learning*; Institute of Electrical and Electronics Engineers Inc.: Piscataway, NJ, USA, 2022. [CrossRef]

25. Zhang, J.; Xu, Z.; Wu, J.; Chen, Q.; Wang, F. *Lightweight Intelligent Autonomous Unmanned Vehicle Based on Deep Neural Network in ROS System*; Institute of Electrical and Electronics Engineers Inc.: Piscataway, NJ, USA, 2022; pp. 679–684. [CrossRef]
26. Miyama, M. Robust inference of multi-task convolutional neural network for advanced driving assistance by embedding coordinates. In Proceedings of the 8th World Congress on Electrical Engineering and Computer Systems and Science, EECSS 2022, Prague, Czech Republic, 28–30 July 2022; pp. 105-1–105-9.
27. Jebbar, M.; Maizate, A.; Ait Abdelouahid, R. *Moroccan's Arabic Speech Training And Deploying Machine Learning Models with Teachable Machine*; Elsevier: Amsterdam, The Netherlands, 2022; Volume 203, pp. 801–806. [CrossRef]
28. Copot, C.; Shi, L.; Smet, E.; Ionescu, C.; Vanlanduit, S. *Comparison of Deep Learning Models in Position Based Visual Servoing*; Institute of Electrical and Electronics Engineers Inc.: Piscataway, NJ, USA, 2022; Volume 2022. [CrossRef]
29. Liu, J.; Rangwala, M.; Ahluwalia, K.; Ghajar, S.; Dhami, H.; Tokekar, P.; Tracy, B.; Williams, R. Intermittent Deployment for Large-Scale Multi-Robot Forage Perception: Data Synthesis, Prediction, and Planning. *IEEE Trans. Autom. Sci. Eng.* **2022**, 1–21. [CrossRef]
30. Lai, J.; Ramli, H.; Ismail, L.; Hasan, W. Real-Time Detection of Ripe Oil Palm Fresh Fruit Bunch Based on YOLOv4. *IEEE Access* **2022**, *10*, 95763–95770. [CrossRef]
31. Lin, H.Z.; Chen, H.H.; Choophutthakan, K.; Li, C.H. *Autonomous Mobile Robot as a Cyber-Physical System Featuring Networked Deep Learning and Control*; Institute of Electrical and Electronics Engineers Inc.: Piscataway, NJ, USA, 2022; Volume 2022, pp. 268–274. [CrossRef]
32. Mandel, N.; Sandino, J.; Galvez-Serna, J.; Vanegas, F.; Milford, M.; Gonzalez, F. *Resolution-adaptive Quadtrees for Semantic Segmentation Mapping in UAV Applications*; IEEE Computer Society: Piscataway, NJ, USA, 2022; Volume 2022. [CrossRef]
33. Chen, Y.; Li, D.; Zhong, H.; Zhao, R. The Method for Automatic Adjustment of AGV's PID Based on Deep Reinforcement Learning. *Inst. Phys.* **2022**, *2320*, 012008. [CrossRef]
34. Chen, Y.; Li, D.; Zhong, H.; Zhu, O.; Zhao, Z. The Determination of Reward Function in AGV Motion Control Based on DQN. *Inst. Phys.* **2022**, *2320*, 012002. [CrossRef]
35. Chavez-Galaviz, J.; Mahmoudian, N. *Underwater Dock Detection through Convolutional Neural Networks Trained with Artificial Image Generation*; Institute of Electrical and Electronics Engineers Inc.: Piscataway, NJ, USA, 2022; pp. 4621–4627. [CrossRef]
36. Carvalho, E.; Susbielle, P.; Hably, A.; Dibangoye, J.; Marchand, N. *Neural Enhanced Control for Quadrotor Linear Behavior Fitting*; Institute of Electrical and Electronics Engineers Inc.: Piscataway, NJ, USA, 2022, pp. 378–385. [CrossRef]
37. Liu, K.; Zhou, X.; Zhao, B.; Ou, H.; Chen, B. An Integrated Visual System for Unmanned Aerial Vehicles Following Ground Vehicles: Simulations and Experiments. In Proceedings of the 2022 IEEE 17th International Conference on Control & Automation (ICCA), 2022 IEEE 17th International Conference on Control & Automation (ICCA), Naples, Italy, 27–30 June 2022; pp. 593–598. [CrossRef]
38. Yun, J.; Jiang, D.; Sun, Y.; Huang, L.; Tao, B.; Jiang, G.; Kong, J.; Weng, Y.; Li, G.; Fang, Z. Grasping Pose Detection for Loose Stacked Object Based on Convolutional Neural Network with Multiple Self-Powered Sensors Information. *IEEE Sens. J.* **2022**. [CrossRef]
39. Zhu, C.; Chen, L.; Cai, Y.; Wang, H.; Li, Y. Vehicle-Mounted Multi-Object Tracking Based on Self-Query. In Proceedings of the International Conference on Advanced Algorithms and Neural Networks (AANN 2022), Zhuhai, China, 25–27 February 2022; Volume 12285. [CrossRef]
40. Nawabi, A.; Jinfang, S.; Abbasi, R.; Iqbal, M.; Heyat, M.; Akhtar, F.; Wu, K.; Twumasi, B. Segmentation of Drug-Treated Cell Image and Mitochondrial-Oxidative Stress Using Deep Convolutional Neural Network. *Oxidative Med. Cell. Longev.* **2022**, *2022*, 5641727. [CrossRef]
41. Saripuddin, M.; Suliman, A.; Sameon, S. *Impact of Resampling and Deep Learning to Detect Anomaly in Imbalance Time-Series Data*; Institute of Electrical and Electronics Engineers Inc.: Piscataway, NJ, USA, 2022, pp. 37–41. [CrossRef]
42. Yu, Y.; Zhang, J.Z.; Cao, Y.; Kazancoglu, Y. Intelligent transformation of the manufacturing industry for Industry 4.0: Seizing financial benefits from supply chain relationship capital through enterprise green management. *Technol. Forecast. Soc. Chang.* **2021**, *172*, 120999. [CrossRef]
43. Santoso, F.; Finn, A. A Data-Driven Cyber-Physical System Using Deep-Learning Convolutional Neural Networks: Study on False-Data Injection Attacks in an Unmanned Ground Vehicle Under Fault-Tolerant Conditions. *IEEE Trans. Syst. Man, Cybern. Syst.* **2023**, *53*, 346–356. [CrossRef]
44. Sinulingga, H.R.; Munir, R. Road Recognition System with Heuristic Method and Machine Learning. In Proceedings of the 2020 7th International Conference on Advance Informatics: Concepts, Theory and Applications (ICAICTA), Tokoname, Japan, 8–9 September 2020; pp. 1–6. [CrossRef]
45. Bhattacharya, S.; Dutta, S.; Maiti, T.K.; Miura-Mattausch, M.; Navarro, D.; Mattausch, H.J. Machine learning algorithm for autonomous control of walking robot. In Proceedings of the 2018 International Symposium on Devices, Circuits and Systems (ISDCS), Howrah, India, 29–31 March 2018; pp. 1–4. [CrossRef]
46. Mishra, P.; Jain, U.; Choudhury, S.; Singh, S.; Pandey, A.; Sharma, A.; Singh, R.; Pathak, V.; Saxena, K.; Gehlot, A. Footstep planning of humanoid robot in ROS environment using Generative Adversarial Networks (GANs) deep learning. *Robot. Auton. Syst.* **2022**, *158*, 104269. [CrossRef]
47. Mahmeen, M.; Sanchez, R.; Friebe, M.; Pech, M.; Haider, S. Collision Avoidance Route Planning for Autonomous Medical Devices Using Multiple Depth Cameras. *IEEE Access* **2022**, *10*, 29903–29915. [CrossRef]

48. Isaac, M.M., Wilscy, M., Aji, S. A Skip-Connected CNN and Residual Image-Based Deep Network for Image Splicing Localization. In *Pervasive Computing and Social Networking*; Ranganathan, G., Bestak, R., Palanisamy, R., Rocha, Á., Eds.; Lecture Notes in Networks and Systems; Springer: Singapore, 2022; Volume 317. [CrossRef]
49. Domingo, J.; GÃ³mez-GarcÃa-Bermejo, J.; Zalama, E. Optimization and improvement of a robotics gaze control system using LSTM networks. *Multimed. Tools Appl.* **2022**, *81*, 3351–3368. [CrossRef]
50. Caesarendra, W.; Wijaya, T.; Pappachan, B.K.; Tjahjowidodo, T. Adaptation to industry 4.0 using machine learning and cloud computing to improve the conventional method of deburring in aerospace manufacturing industry. In Proceedings of the 2019 International Conference on Information and Communication Technology and Systems, ICTS 2019, Surabaya, Indonesia, 18- July 2019; pp. 120–124. [CrossRef]
51. Gonzalez, A.G.C.; Alves, M.V.S.; Viana, G.S.; Carvalho, L.K.; Basilio, J.C. Supervisory Control-Based Navigation Architecture: A New Framework for Autonomous Robots in Industry 4.0 Environments. *IEEE Trans. Ind. Inform.* **2018**, *14*, 1732–1743. [CrossRef]
52. Mayer, C.; Ofek, E.; Fridrich, D.; Molchanov, Y.; Yacobi, R.; Gazy, I.; Hayun, I.; Zalach, J.; Paz-Yaacov, N.; Barshack, I. Direct identification of ALK and ROS1 fusions in non-small cell lung cancer from hematoxylin and eosin-stained slides using deep learning algorithms. *Mod. Pathol.* **2022**, *35*, 1882–1887. [CrossRef] [PubMed]
53. Ma, C.Y.; Zhou, J.Y.; Xu, X.T.; Qin, S.B.; Han, M.F.; Cao, X.H.; Gao, Y.Z.; Xu, L.; Zhou, J.J.; Zhang, W.; et al. Clinical evaluation of deep learning–based clinical target volume three-channel auto-segmentation algorithm for adaptive radiotherapy in cervical cancer. *BMC Med. Imaging* **2022**, *22*, 123. [CrossRef]
54. Xin, J.; Zhao, H.; Liu, D.; Li, M. Application of deep reinforcement learning in mobile robot path planning. In Proceedings of the Proceedings—2017 Chinese Automation Congress, CAC 2017, Jinan, China, 20–22 October 2017; Volume 2017, pp. 7112–7116. [CrossRef]
55. Gattu, S.; Penumacha, K. Autonomous Navigation and Obstacle Avoidance using Self-Guided and Self-Regularized Actor-Critic. In Proceedings of the 8th International Conference on Robotics and Artificial Intelligence, Singapore, 14–16 September 2022; pp. 52–58. [CrossRef]
56. Xie, L.; Shen, Y.; Zhang, M.; Zhong, Y.; Lu, Y.; Yang, L.; Li, Z. Single-model multi-tasks deep learning network for recognition and quantitation of surface-enhanced Raman spectroscopy. *Opt. Express* **2022**, *30*, 41580–41589. [CrossRef] [PubMed]
57. Bravo-Arrabal, J.; Toscano-Moreno, M.; Fernandez-Lozano, J.; Mandow, A.; Gomez-Ruiz, J.; GarcÃa-Cerezo, A. The internet of cooperative agents architecture (X-ioca) for robots, hybrid sensor networks, and mec centers in complex environments: A search and rescue case study. *Sensors* **2021**, *21*, 7843. [CrossRef] [PubMed]
58. Liu, Z.; Shi, Y.; Chen, H.; Qin, T.; Zhou, X.; Huo, J.; Dong, H.; Yang, X.; Zhu, X.; Chen, X.; et al. Machine learning on properties of multiscale multisource hydroxyapatite nanoparticles datasets with different morphologies and sizes. *NPJ Comput. Mater.* **2021**, *7*, 142. [CrossRef]
59. Bae, S.Y.; Lee, J.; Jeong, J.; Lim, C.; Choi, J. Effective data-balancing methods for class-imbalanced genotoxicity datasets using machine learning algorithms and molecular fingerprints. *Comput. Toxicol.* **2021**, *20*, 100178. [CrossRef]
60. Choi, S.; Park, S. Development of Smart Mobile Manipulator Controlled by a Single Windows PC Equipped with Real-Time Control Software. *Int. J. Precis. Eng. Manuf.* **2021**, *22*, 1707–1717. [CrossRef]
61. Saeedvand, S.; Mandala, H.; Baltes, J. Hierarchical deep reinforcement learning to drag heavy objects by adult-sized humanoid robot. *Appl. Soft Comput.* **2021**, *110*, 107601. [CrossRef]
62. Chen, K.; Liang, Y.; Jha, N.; Ichnowski, J.; Danielczuk, M.; Gonzalez, J.; Kubiatowicz, J.; Goldberg, K. FogROS: An Adaptive Framework for Automating Fog Robotics Deployment. In Proceedings of the 2021 IEEE 17th International Conference on Automation Science and Engineering (CASE), Lyon, France, 23–27 August 2021; pp. 2035–2042. [CrossRef]
63. Ou, J.; Guo, X.; Lou, W.; Zhu, M. *Learning the Spatial Perception and Obstacle Avoidance with the Monocular Vision on a Quadrotor*; Institute of Electrical and Electronics Engineers Inc.: Piscataway, NJ, USA, 2021; pp. 582–587. [CrossRef]
64. Jembre, Y.; Nugroho, Y.; Khan, M.; Attique, M.; Paul, R.; Shah, S.; Kim, B. Evaluation of reinforcement and deep learning algorithms in controlling unmanned aerial vehicles. *Appl. Sci.* **2021**, *11*, 7240. [CrossRef]
65. Yuhas, M.; Feng, Y.; Ng, D.; Rahiminasab, Z.; Easwaran, A. *Embedded Out-of-Distribution Detection on an Autonomous Robot Platform*; Association for Computing Machinery, Inc.: New York, NY, USA, 2021; pp. 13–18. [CrossRef]
66. Timmis, I.; Paul, N.; Chung, C.J. Teaching Vehicles to Steer Themselves with Deep Learning. In Proceedings of the 2021 IEEE International Conference on Electro Information Technology (EIT), Mt. Pleasant, MI, USA, 14–15 May 2021; pp. 419–421. [CrossRef]
67. Sui, J.; Yang, L.; Zhang, X.; Zhang, X. Laser measurement key technologies and application in robot autonomous navigation. *Int. J. Pattern Recognit. Artif. Intell.* **2011**, *25*, 1127–1146. [CrossRef]
68. Parikh, A.; Karamchandani, S.; Lalani, A.; Bhavsar, D.; Kamat, G. Unmanned Terrestrial Deep Stereo ConvNet Gofer Embedded with CNN Architecture. *Int. J. Mech. Eng. Robot. Res.* **2022**, *11*, 807–819. [CrossRef]
69. Aslan, M.; Durdu, A.; Yusefi, A.; Yilmaz, A. HVIOnet: A deep learning based hybrid visual–inertial odometry approach for unmanned aerial system position estimation. *Neural Netw.* **2022**, *155*, 461–474. [CrossRef] [PubMed]
70. Zhi, L.; Xuesong, M. Navigation and Control System of Mobile Robot Based on ROS. In Proceedings of the 2018 IEEE 3rd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC), Chongqing, China, 12–14 October 2018; pp. 368–372. [CrossRef]

71. Yin, S.; Ji, W.; Wang, L. A machine learning based energy efficient trajectory planning approach for industrial robots. *Procedia CIRP* **2019**, *81*, 429–434. [CrossRef]
72. Wang, L. Automatic control of mobile robot based on autonomous navigation algorithm. *Artif. Life Robot.* **2019**, *24*, 494–498. [CrossRef]
73. Yuan, M.; Shan, J.; Mi, K. Deep Reinforcement Learning Based Game-Theoretic Decision-Making for Autonomous Vehicles. *IEEE Robot. Autom. Lett.* **2022**, *7*, 818–825. [CrossRef]
74. Guan, W.; Guo, Y. A Visual Learning based Robotic Grasping System. In Proceedings of the 2022 The 6th International Conference on Advances in Artificial Intelligence, Birmingham, UK, 22–24 October 2022; pp. 22–28. [CrossRef]
75. Liu, G.; Sun, W.; Xie, W.; Xu, Y. Learning visual path–following skills for industrial robot using deep reinforcement learning. *Int. J. Adv. Manuf. Technol.* **2022**, *122*, 1099–1111. [CrossRef]
76. Su, L.; Hua, Y.; Dong, X.; Ren, Z. Human-UAV swarm multi-modal intelligent interaction methods. *Hangkong Xuebao/Acta Aeronaut. Astronaut. Sin.* **2022**, *43*. [CrossRef]
77. Tsai, J.; Chang, C.C.; Ou, Y.C.; Sieh, B.H.; Ooi, Y.M. Autonomous Driving Control Based on the Perception of a Lidar Sensor and Odometer. *Appl. Sci.* **2022**, *12*, 7775. [CrossRef]
78. Yinka-Banjo, C.; Ugot, O.; Ehiorobo, E. Object Detection for Robot Coordination in Robotics Soccer. *Niger. J. Technol. Dev.* **2022**, *19*, 136–142. [CrossRef]
79. Khalifa, A.; Abdelrahman, A.; Strazdas, D.; Hintz, J.; Hempel, T.; Al-Hamadi, A. Face Recognition and Tracking Framework for Humanâ€"Robot Interaction. *Appl. Sci.* **2022**, *12*, 5568. [CrossRef]
80. Ravi, N.; El-Sharkawy, M. Real-Time Embedded Implementation of Improved Object Detector for Resource-Constrained Devices. *J. Low Power Electron. Appl.* **2022**, *12*, 21. [CrossRef]
81. Pu, L.; Zhang, X. Deep learning based UAV vision object detection and tracking. *Beijing Hangkong Hangtian Daxue Xuebao/J. Beijing Univ. Aeronaut. Astronaut.* **2022**, *48*, 872–880. [CrossRef]
82. Gong, H.; Wang, P.; Ni, C.; Cheng, N. Efficient Path Planning for Mobile Robot Based on Deep Deterministic Policy Gradient. *Sensors* **2022**, *22*, 3579. [CrossRef] [PubMed]
83. Baek, E.T.; Im, D.Y. ROS-Based Unmanned Mobile Robot Platform for Agriculture. *Appl. Sci.* **2022**, *12*, 4335. [CrossRef]
84. Zhu, X.; Li, N.; Wang, Y. Software change-proneness prediction based on deep learning. *J. Software: Evol. Process.* **2022**, *34*, e2434. [CrossRef]
85. Bui, K.; Truong, G.; Ngoc, D. GCTD3: Modeling of Bipedal Locomotion by Combination of TD3 Algorithms and Graph Convolutional Network. *Appl. Sci.* **2022**, *12*, 2948. [CrossRef]
86. Escobar-Naranjo, J.; Garcia, M.V. Self-supervised Learning Approach to Local Trajectory Planning for Mobile Robots Using Optimization of Trajectories. *Lect. Notes Netw. Syst.* **2023**, *578*, 741–748. [CrossRef]
87. Montalvo, W.; Garcia, C.A.; Naranjo, J.E.; Ortiz, A.; Garcia, M.V. Tele-operation system for mobile robots using in oil & gas industry; [Sistema de tele-operación para robots móviles en la industria del petróleo y gas]. *Risti - Rev. Iber. Sist. Tecnol. Inf.* **2020**, *2020*, 351–365.
88. Caiza, G.; Garcia, C.A.; Naranjo, J.E.; Garcia, M.V. Flexible robotic teleoperation architecture for intelligent oil fields. *Heliyon* **2020**, *6*, e03833. [CrossRef]
89. Mohanty, P.K.; Sah, A.K.; Kumar, V.; Kundu, S. Application of deep Q-learning for wheel mobile robot navigation. In Proceedings of the 2017 3rd International Conference on Computational Intelligence and Networks (CINE), Odisha, India, 28–28 October 2017; pp. 88–93. [CrossRef]
90. de Assis Brasil, P.M.; Pereira, F.U.; de Souza Leite Cuadros, M.A.; Cukla, A.R.; Tello Gamarra, D.F. A study on global path planners algorithms for the simulated turtlebot 3 robot in ros. In Proceedings of the 2020 Latin American Robotics Symposium (LARS), 2020 Brazilian Symposium on Robotics (SBR) and 2020 Workshop on Robotics in Education (WRE), Natal, Brazil, 9–13 November 2020; pp. 1–6. [CrossRef]
91. Guizzo, E.; Ackerman, E. The TurtleBot3 Teacher Resources Hands On. *IEEE Spectr.* **2017**, *54*, 19–20. [CrossRef]
92. Amsters, R.; Slaets, P. *Turtlebot 3 as a Robotics Education Platform*; Springer: Berlin/Heidelberg, Germany, 2020; pp. 170–181. [CrossRef]
93. Gurgoze, G.; Turkoglu, I. Analysis of the Performance According to Object Density in Static Environments of GA and PSO Algorithms Used in Mobile Robot Path Planning. *Turk. J. Sci. Technol.* **2021**, *16*, 205–214.
94. Bhagwat, H.; Gogate, U. Reviewing Machine Learning Algorithms in the Domain of Healthcare. *Int. J. Eng. Res. Technol.* **2021**, *10*, 381–386.
95. Giuseppe, C.; Ignacio, C.; Kyle, C.; Laurent, D.; Maria, S.; Naftali, T.; Leslie, V.M.; Lenka, Z. Machine learning and the physical sciences. *Rev. Mod. Phys.* **2019**, *91*, 1–39. [CrossRef]
96. Liu, Y.; Cao, B.; Li, H. Improving ant colony optimization algorithm with epsilon greedy and Levy flight. *Complex Intell. Syst.* **2021**, *7*, 1711–1722. [CrossRef]
97. Montalvo, W.; Escobar-Naranjo, J.; Garcia, C.A.; Garcia, M.V. Low-Cost Automation for Gravity Compensation of Robotic Arm. *Appl. Sci.* **2020**, *10*, 3823. [CrossRef]
98. Ruan, X.; Ren, D.; Zhu, X.; Huang, J. Mobile robot navigation based on deep reinforcement learning. In Proceedings of the 2019 Chinese Control And Decision Conference (CCDC), Nanchang, China, 3–5 June 2019; pp. 6174–6178. [CrossRef]

99. Lin, F.; Ji, Z.; Wei, C.; Niu, H. *Reinforcement Learning-Based Mapless Navigation with Fail-Safe Localisation*; Springer: Berlin/Heidelberg, Germany, 2021; pp. 100–111. [CrossRef]
100. Tian, S.; Lei, S.; Huang, Q.; Huang, A. The application of path planning algorithm based on deep reinforcement learning for mobile robots. In Proceedings of the 2022 International Conference on Culture-Oriented Science and Technology (CoST), Lanzhou, China, 18–21 August 2022; pp. 381–384. [CrossRef]