Apollo: Self-learning robots in medical centres

Tiago Fonseca
tiago.s.fonseca@tecnico.ulisboa.pt

Instituto Superior Técnico, Lisboa, Portugal

November 2023

Abstract

In recent years, Reinforcement Learning in Autonomous Mobile Robots has witnessed remarkable breakthroughs, opening new frontiers in various domains, particularly in healthcare. This work introduces Apollo, an open-source framework designed to empower the research community by providing a comprehensive platform for creating, training, and evaluating Reinforcement Learning methods for autonomous robots in healthcare. Apollo is a versatile framework that seamlessly integrates an autonomous mobile robot and a simulator, bridging the gap between the virtual and real worlds. It stands as a powerful tool, which researchers can use to address the complex challenges of navigation within the delicate medical environments. The framework leverages the potential of Transfer Learning, enabling the training of Reinforcement Learning models in simulated environments, followed by fine-tuning for real-world deployment. An important contribution of this work lies in the introduction of a Deep Reinforcement Learning method, centered on Double Deep Q-Network (DDQN), to address transportation tasks within healthcare environments. This method integrates Graph-based LiDAR Simultaneous Localization and Mapping (SLAM) with DDQNs, allowing Apollo to navigate and transport objects within sensitive medical environments. The Apollo framework and the DDQN approach mark significant milestones in the field of autonomous mobile robots in healthcare, holding immense promise for applications demanding adaptability and autonomous operation. Through this innovative framework and the proposed Deep Reinforcement Learning method, Apollo paves the way for new horizons in healthcare automation and sets the stage for further advancements in the field.
Keywords: Autonomous Mobile Robots, Reinforcement Learning, Deep Reinforcement Learning, Double Deep Q-Networks

1. Introduction

The good functioning of a hospital or medical centre is vital to the well being of the society. Keeping a short waiting time, availability of medical personnel and sufficient medical supplies is fundamental for assuring that patients receive proper quality treatment on time.

In recent years, technology advancements have been enabling major efficiency improvements in hospitals. The use of robots has shown relevant improvements in the well functioning of medical centres. Robots have been successfully used in healthcare [25], not only to improve core medical tasks, like surgery [12], but also to improve the logistics of transportation of medical equipment, such as the robot Tug[5].

Robots can assist with tasks that are repetitive or physically demanding, freeing up medical staff to focus on more complex tasks that require human judgment and expertise. They can help to reduce the risk of exposure to infectious diseases by performing tasks such as handling hazardous materials or cleaning patient rooms. Additionally, robots can provide a level of accuracy and consistency that may be difficult for humans to achieve, particularly in areas such as surgery. They can also help to improve the efficiency of medical centers by automating processes, reducing the time and resources needed to complete tasks. Overall, the use of robots in medical centers can help to improve the quality of care provided to patients and make the work of medical staff more efficient and effective.

The efficiency of transportation tasks in medical environments is equally important. A considerable amount of time is spent by the medical personnel doing tedious transportation tasks, such as transporting medicine, medical equipment, linens and other items. The use of robots to improve transportation services in hospitals is highly beneficial by offloading the medical personnel, allowing them to focus on more complex issues, being able to spend more time on the patient and the treatment, thereby improving the quality and efficiency of the

care [1].

Traditionally, designing a robot usually required the designer to have a deep prior understanding of the exact tasks to perform and environment in which the robot would operate to be able to define how the tasks would be carried out, the goals to be achieved and the control policy [26]. Our goal is to develop a robot that can perform various transportation tasks, such as accompanying patients and visitors, in a dynamic and constantly changing medical environment. In such an environment, patients and medical staff are in motion and there are non-static objects present. Therefore, we require a self-learning autonomous robot to meet these challenges.

The field of Reinforcement Learning (RL), a Machine Learning technique in which an agent learns its behaviour through trial-and-error interactions with a dynamic environment [13], has grown considerably in recent years.

Training a robot with RL presents a bigger challenge than most of the well studied RL benchmark problems. In Robotics it is unrealistic to assume that the true state of the environment is completely observable and noise free. Knowing precisely the current state of the environment is unlikely and different states might be perceived very differently. Furthermore, learning in the physical environment is expensive, takes a long time and is often hard to reproduce. Additionally, the robot itself can suffer damage due to extensive training in real world environments[15]. Transfer learning is often used to shorten the learning time in real world environments, minimizing these effects. Transfer learning (TL) is the paradigm of improving a learning agent from one domain by transferring information from an agent that learned on other related domain [37]. The latest advancements in the field of autonomous mobile robots propose the use of TL [41] to train the agent in a simulated environment to speed up and improve the training of a robot in the physical world [42]. However, real world experience cannot be replaced by learning in simulations alone [15].

In the last decades, the rise of Deep Learning has been playing major role in solving Machine Learning problems. Deep Reinforcement Learning (DRL) techniques have been not only been used to solve classical RL problems but also to solve control and navigation problems in robotics [16].

Mechanical hardware advancements in various types of mechanisms that enable robot movement, including legs and specialized wheels such as the Mecanum wheel [7], as well as different wheel configurations, have significantly enhanced the capabilities and adaptability of mobile robots to perform tasks related to specific fields with unique requirements [33].

This project aims to develop a self-learning mobile robot for dynamic medical environments, which has to coexist with medical professionals while autonomously navigating and mapping its surroundings. Simultaneously, a robust simulator was created for initial training and testing, and a novel RL method was designed and compared with existing approaches. Deploying both the robot and the simulator as an open-source framework is a crucial milestone, while real-world testing validates the methodology and its suitability for autonomous mobile robot navigation in healthcare settings.

The principal contribution of this research is a comprehensive framework, incorporating both a physical robot and a simulator. This framework serves as a versatile tool for the research community, enabling the creation, training, and evaluation of RL methods for autonomous mobile robots in medical environments. The physical robot, developed within the scope of this work, features an omnidirectional robotic platform equipped with Mecanum wheels and a 2D Light Detection and Ranging (LiDAR), and the simulator comprises a simulated version of the robot and 3D simulated environments, including a simulated hospital environment. Furthermore, this work introduces a promising modular DRL method that integrates Graph-based Simultaneous Localization and Mapping (SLAM) and Double Deep Q-Network (DDQN), utilizing the previously mentioned framework. The results demonstrate its capacity to overcome some of the intricate challenges of automated transportation tasks in medical centers.

2. Background and Related Work

Autonomous mobile robots are self-navigating robots that perform tasks without human intervention. Various approaches are used to develop these robots, depending on factors like the environment, tasks, and desired autonomy level. These factors influence sensor choices, such as LiDAR, radar, wireless signals, and cameras, for obstacle detection, mapping, and navigation. Navigation methods include machine learning, preprogrammed paths, way point navigation, and map-based navigation.

Recent healthcare robotics developments aim to enhance efficiency, accuracy, patient safety, and satisfaction. Autonomous mobile robots offer healthcare transformation by enhancing patient care, reducing staff workload, and improving facility efficiency.In healthcare, autonomous mobile robots are utilized for tasks like medication delivery, patient transportation, and equipment delivery. Studies have explored their potential in healthcare. Ozkil et al. [22] proposed mobile robots to optimize

hospital transportation, improving pharmaceutical supplies, clinical waste, and patient meal transportation. The introduction of *HelpMate* [8] in 1992 marked the beginning of self-navigating hospital robots. Despite using sensor-based motion planning, it lacked adaptability to new environments. Takahashi et al. [31] developed an autonomous mobile robot with omni-directional wheels for hospital material transport. It employed virtual potential fields for obstacle collision avoidance, but relied on accurate state estimation and obstacle motion prediction. *Terapio* [32] aimed to relieve medical personnel by transporting supplies and tracking healthcare professionals. While it used advanced mobility mechanisms and obstacle detection, it required operator input. The latest development, *Pathfinder* [3], used LiDAR SLAM for navigation and mapping. Although it demonstrated effective navigation, manual guidance was required for object differentiation and environment understanding. These studies, while promising, lack the ability for independent task learning and environment adaptation, emphasizing the importance of SLAM, Active SLAM, and RL for autonomous learning in dynamic environments.

Mecanum wheels, characterized by their three degrees of freedom and omnidirectional movement capabilities, are well-suited for navigating complex indoor environments and tight spaces. Common configurations include three-wheeled robots with 90º angled rollers, offering three-directional movement, and four-wheeled robots with 45º angled rollers, providing increased stability, load-bearing capacity, and eight-directional movement, making them ideal for autonomous mobile robots in medical centers [7, 28, 23].

Reinforcement learning enables agents to learn optimal behavior without a world model, and it has found applications in developing autonomous mobile robots, allowing them to adapt to changing environments [15, 6, 21].

Deep Reinforcement Learning (DRL) combines deep learning and reinforcement learning, leading to remarkable advancements in various domains. Notably, Mnih et al. introduced the Deep Q-Network (DQN) in 2013, demonstrating superhuman capabilities in Atari 2600 games [19]. In 2016, Hasselt et al. presented the DDQN, a significant improvement that mitigated overestimation bias in DRL [35]. DRL is essential in robotics, enabling robots to learn from raw data and adapt to complex environments [11]. However, DRL poses challenges, requiring substantial data and computation power compared to traditional reinforcement learning (RL) [17, 20]. DRL models can be difficult to interpret, hindering the understanding of decision-making processes and model issues [4].

Wang et al. introduce a modular reinforcement learning architecture for complex dynamic environment navigation, consisting of three modules, including avoidance, navigation, and an action scheduler, which demonstrates efficient obstacle avoidance and navigation completion, though the modular approach may increase system complexity and computational demands [36].

Transfer learning (TL) involves reusing a model trained on a related problem to address new tasks when data or computational resources for training from scratch are limited. In the autonomous mobile robot context, TL often begins with simulated environment training, reducing cost and risk. Notably, Selfridge's pioneering 1985 study [27] demonstrated the benefits of transfer learning in reinforcement learning for physical robots. Barrett et al. [2] expanded on this, showing how Q-value reuse with transfer learning improves robot learning speed and performance. Recent efforts focus on addressing the gap between simulated and real environments, like Tzeng et al.'s method for adapting visual representations [34] and Peng et al.'s approach to training agents in simulation and deploying them in the real world [24]. Another promising approach introduced by Zhu et al. [41] targets robot navigation tasks using deep reinforcement learning, demonstrating good generalization performance. These studies indicate the potential of transfer learning for autonomous mobile robots, particularly in scenarios where real-world training is costly, slow, or risky.

Recent research in collision avoidance for hospital robots has explored various RL approaches. Manuelli et al. [18] compared RL methods like SARSA and Q-learning for autonomous car obstacle avoidance, finding traditional controllers outperformed RL in simulations with static obstacles. Kato et al. [14] developed a system that combines deep reinforcement learning for local navigation with topological maps for global navigation, enabling effective robot navigation in crowded environments. Han et al. [9] introduced a complex DRL framework that combines 2D LiDAR and monocular cameras to enhance collision avoidance in tall mobile robots, achieving substantial improvements in both simulated and real-world scenarios. Further evaluation is needed for practical viability in dynamic environments.

In recent years, there has been a surge of research in Simultaneous Localization and Mapping (SLAM). SLAM techniques can be categorized into two main approaches: visual-based and LiDAR-based. Visual-based SLAM relies on camera sensors, using stereo or monocular cameras for feature tracking and mapping. However, it can be sensitive to lighting conditions and movement speed.

LiDAR-based SLAM, on the other hand, utilizes LiDAR sensors to create high-precision 3D maps unaffected by lighting changes. WiFi-based SLAM has also been explored, allowing robots to triangulate their positions by measuring signal strengths from access points, but it relies on pre-existing access point locations. The two primary approaches for LiDAR-based SLAM are Graph-based SLAM, which uses graph representations to map the environment, and Filter-based SLAM, employing filters like Kalman or particle filters for localization and mapping. Among notable contributions, *Google Cartographer* by Hess et al. [10] introduced a Graph-based SLAM method using 2D LiDAR, providing efficient solutions for loop closure, a critical challenge in SLAM. Sun et al. [29] built upon Cartographer for Active SLAM, focusing on efficient frontier detection and path planning for autonomous mobile robots. SLAM is essential in addressing challenges in mobile robot navigation, and recent advancements in Deep Reinforcement Learning (DRL) have introduced new paradigms. Zhu et al. [40] explored the use of DRL frameworks for navigation, bypassing traditional SLAM approaches by learning navigation strategies directly from raw sensor inputs. While promising, DRL approaches require substantial training and pose challenges in monitoring and comprehension of the navigation process. Wen et al. [38] employed fully convolutional residual networks and the Dueling DQN algorithm for obstacle avoidance, while Tai et al. [30] presented a map-less motion planner for mobile robots based on asynchronous DRL, enabling efficient training with minimal data requirements. Yokoyama et al. [39] proposed a Double Deep Q-Network approach for autonomous mobile robot navigation using depth images, but it's essential to note that LiDAR remains preferable for detailed and accurate data in various lighting conditions [43].

The state of the art in Autonomous Mobile Robots for healthcare highlights the importance of sensor technology choices and Simultaneous Localization and Mapping (SLAM) methods. The choice between LiDAR and visual-based sensors depends on the trade-off between precision and robustness, with LiDAR being more reliable in indoor environments. Graph-based SLAM is favored for mapping and navigating complex indoor spaces, with Cartographer showing promise, but it may lack adaptability. To address these challenges, our proposed design integrates LiDAR for precision, Cartographer-based SLAM for robust mapping, and Transfer Learning for efficient task adaptation. Additionally, the healthcare environment's unique demands, such as precision, adaptability, and sensitivity to patient needs, are not adequately addressed in existing research. Our proposed design aims to provide a comprehensive solution to bridge this gap by balancing precision, adaptability, and stability, thus enabling efficient navigation in healthcare settings.

3. Methodology

This work introduces Apollo, an autonomous mobile robot that serves as both a prototype for training and evaluating Reinforcement Learning methods in real-world and simulated environments and a framework to explore RL techniques, providing an open-source platform for research in autonomous mobile robot navigation in healthcare settings. The approach we adopted is the development of an Autonomous Mobile Robot using the Reinforcement Learning framework. As previously mentioned, Apollo is not only a physical robot, but also an open-source framework that anyone, namely the RL research community can use to create, train, evaluate, and deploy Reinforcement Learning methods in dynamic and sensitive environments, namely in hospitals and medical centres. The Apollo framework is released under the MIT license. The Apollo framework encompasses Apollo, the physical robot and a simulator.

3.1. Robot

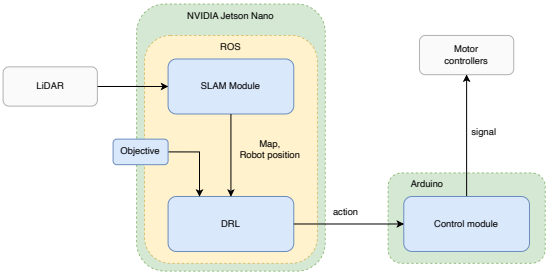

Figure 1: Robot architecture

Apollo's architecture is modular, featuring three key modules: the Simultaneous Localization and Mapping (SLAM) module, the Deep Reinforcement Learning (DRL) module, and the Control module. Each of these modules is designed to manage specific aspects of the robot's operation. The robot perceives the environment using LiDAR. Subsequently, this data serves as input for the SLAM module. The SLAM module simultaneously creates a 2D map of the robot's surroundings and identifies the robot's location within this map. Following this, the DRL module comes into play, receiving both the map and the robot's position within it. The DRL module can either receive an objective location for the robot to move to or, during training, randomly generate objectives. Then it calculates a reward and then proceeds to input the reward, the map, the robot's position, and the objective location into a DRL model. This model subsequently

generates actions that are executed by the Control module. This module translates these actions into physical movement by calculating the velocity and direction of each wheel. Finally, the Control module sends signals to the actuators, which, in this context, are the motor controllers for each wheel, to perform the movement corresponding to each action. The SLAM and DRL modules are specified to use the Robotic Operating System (ROS) framework, which operates on an NVIDIA Jetson Nano, while the control module is designed to run on an Arduino. The use of open-source technologies was a focus on this project to ease other researchers to use the Apollo framework for future research.

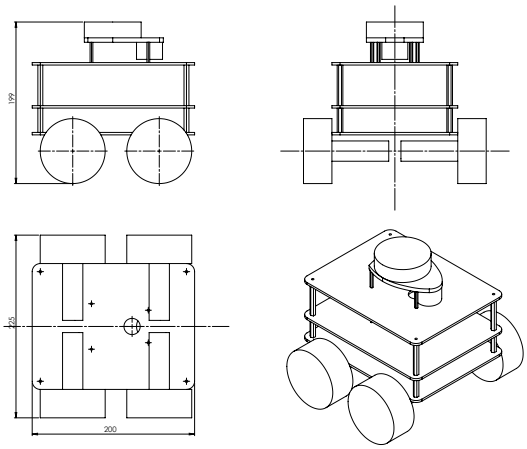

Figure 2: Technical drawing of the assembly of the robot

A configuration employing four Mecanum wheels was chosen for the robot. Mecanum wheels, well-suited for indoor environments like hospitals and medical centers due to their exceptional maneuverability on flat surfaces, were employed to enable precise navigation in confined spaces. This chosen configuration involves four Mecanum wheels, each equipped with rollers set at a 45° angle relative to the wheel's axis of rotation, facilitating versatile movement by directing forces in various orientations [7]. To facilitate effective environmental perception, a 360° 2D LiDAR sensor with a range from 0.15 meters to 15 meters was selected as the primary sensing technology, aligning with the robot's operation in indoor environments and minimizing data processing demands, particularly in SLAM applications. Furthermore, the chosen robotic platform for the Apollo framework, designed for research and development of RL models, features compact dimensions of approximately 20 cm in height, 20 cm in length, and 22.5 cm in width, although it lacks cargo transport capabilities, making it ideal for prototype and experimentation purposes [7].

The capability of graph-based techniques to construct comprehensive spatial graphs significantly enhances map accuracy, including loop closure, while also maintaining minimal error accumulation over time. Among graph-based SLAM techniques, Google Cartographer stands out as a robust open-source SLAM solution that can be integrated in the ROS framework, as demonstrated by its successful implementation by Hess et al. [10]. With its robust mapping capabilities and efficient loop closure handling, Cartographer is an excellent choice for Apollo, which is designed to operate in complex indoor environments like hospitals and medical centers.

3.1.1 Simulation

The approach taken for Apollo is to have a modular simulator architecture, utilizing the SLAM module and the DRL module of the architecture of the robot. The simulator is built on the Gazebo platform, which enables the creation and importation of new environments and robot models with ease. Gazebo is a 3D physics-based simulator commonly integrated with ROS, providing a realistic simulation platform for various robotic applications. This integration simplifies its use for the research community and the incorporation of other ROS modules. The simulation is fully integrated into the ROS framework. The Gazebo module serves as the simulator, simulating a 3D physics-based environment. It can be loaded with various simulated world models; however, in the case of Apollo, it includes a default simulated hospital environment. The Gazebo ROS node simulates LiDAR data and publishes it. The SLAM module takes this data as input and generates a map of the robot's simulated surroundings, along with the simulated robot's position within the map (similar to what is described in Section 3.1). Subsequently, along with the reward and objective position, this data is fed into a DRL model in the DRL module. The DRL module then generates an action, which is published and executed by the simulator in the Gazebo module, thereby modifying the state of the simulated environment.

3.2. Reinforcement Learning method

Apollo is a framework designed for creating, training, and evaluating RL models through the interaction between the agent and the environment, as elaborated in Section 3.2.1.

This work introduces a Double Deep Q-Network (DDQN) based model. The decision to employ a DDQN model is based in the analysis conducted earlier, which highlighted the importance of addressing overestimation bias and optimizing learning efficiency in the context of autonomous mobile robots. In this section, the details of the our DRL approach are explained, outlining its key compo-

nents, training methodology, and the unique advantages it offers for the specific challenges posed by the dynamic hospital environment where Apollo operates.

3.2.1 Agent-Environment Interaction

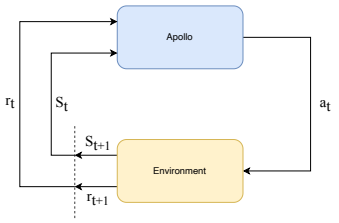

Figure 3: Interaction between Apollo and the environment.

In our system, the Agent-Environment interaction can be characterized using a Markov Decision Process (MDP). Apollo (the *agent*) has a series of interactions with its *environment* that occur at distinct and discrete time frames $t = 0, 1, 2....$

At each one of the time frames, the Apollo receives a representation of the current *state* of the environment, $S_t \in S$, where S is the set of all the possible states.

In this specific scenario, the state space S is characterized by the combination of three components: the map of environment, the current position of the robot within the map, and the target location. Formally, S can be expressed as $S = \{M, RP, OP\}$, where: M represents the set of possible maps; RP represents the set of possible robot positions within the map; OP represents the set of possible objective positions or target locations.

Therefore, any state $S_t \in S$ can be represented as a tuple $S_t = (M_t, RP_t, OP_t)$, $M_t \in M$, $RP_t \in RP$ and $OP_t \in OP$.

The map of the environment M_t is a matrix with variable size, outputted by the SLAM module as an image. Each position on the matrix represents a parcel of the map, where -1 represents unmapped/unknown areas, 0 represents the ground where the agent can navigate, and values between 1 and 50 represent walls.

The agent then selects an *action*, $a_t \in A$, where A is the set of actions. In this context, the available actions comprise the following options: moving forward, moving backward, rotating right, rotating left, and stopping. Formally, A can be expressed as $A = \{F, B, RR, RL, S\}$.

After taking an action, the agent receives a numerical *reward*, r_{t+1}, as a result, at the next step in time. The reward function is defined as: $R : S \times A \to \mathbb{R}$.

$R(S_t, a_t)$ and can be computed at each time frame t as: 0.5 if the robot reaches the objective, -0.00005 if the robot is less than 20 cm from an obstacle, -1 if the episode is terminated and no objectives are reached and 0.5 if the episode is terminated and at least one objective is reached. This reward function provides a balance of positive and negative reinforcement to guide the robot's learning process. It encourages successful task completion, discourages collisions, penalizes episode failures, and rewards partial achievements. This feedback mechanism is an integral part of training the robot to operate effectively and safely in dynamic and sensitive environments, such as hospitals and medical centers. Then the *agent* reaches the *new state*, S_{t+1}. These interactions are illustrated in Figure 3.

At every time frame, the agent follows a strategy that maps states to the likelihood of choosing each available action, known as the agent's policy, represented by π_t, where $\pi_t(a \mid S)$ maps pairs of *state-action* to probabilities. The objective of the agent is to establish a *policy* π that maximizes the expected cumulative reward.

3.2.2 Model

The algorithm proposed in this section is based on Double Deep Q-Network (DDQN). The algorithm is also based on the Agent-Environment interaction described in Section 3.2.1.

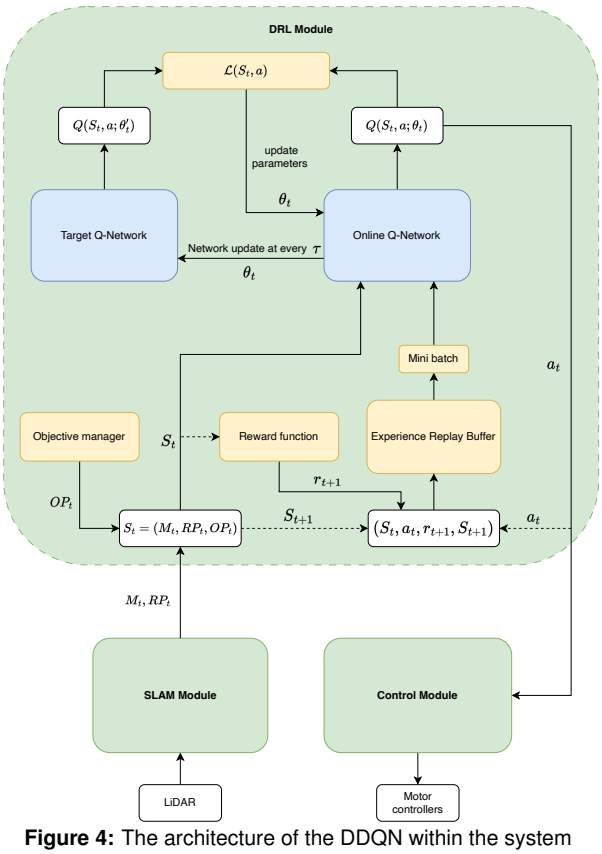

Figure 4: The architecture of the DDQN within the system

The architecture of the DRL module, based on the Double Deep Q-Network architecture, is depicted in Figure 4. The SLAM module outputs two components: the map of the environment M_t and the robot's position within that map RP_t. Additionally, within the DRL module, there is a sub-module known as *Objective Manager* responsible for generating an objective position for the robot to go, OP_t. The Objective Manager has the flexibility to generate a random objective or select an objective specified by the user. Subsequently, the DRL module combines these components to construct the state of the environment $S_t = (M_t, RP_t, OP_t)$.

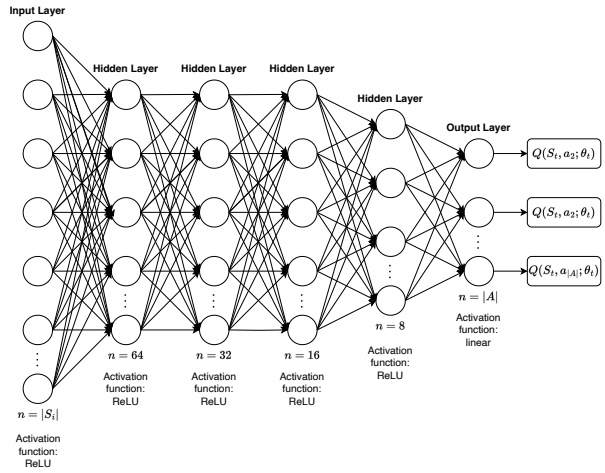

Figure 5: The architecture of the Deep Q-Network

DDQNs have two separate Artificial Neural Q-Networks with the same network architecture, depicted in Figure 5. The online network $Q(S_t, a_t; \theta_t)$ selects actions, while the target network $Q(S_t, a_t; \theta'_t)$ is used to estimate target action-values. During training, the online network $Q(S_t, a_t; \theta_t)$ is periodically synchronized with the target network $Q(S_t, a_t; \theta'_t)$ at every τ steps, ensuring that the value estimates are based on a more reliable set of parameters θ. $Q(S_t, a_t; \theta_t)$ and $Q(S_t, a_t; \theta'_t)$ are used to compute the loss.

The objective of this model is to update the parameters θ of the Q-Network to approximate the optimal action-value function $Q^*(S_t, a_t; \theta_t)$. To achieve the optimal parameters θ, the model is trained using the Algorithm 2. The algorithm runs for a series of episodes until the maximum number of episodes N_{episodes} is reached. Each episode has a series of interactions between the agent and the environment that happen at each time step t, until a maximum number of time steps T_{max} is reached, as described in section 3.2.1. Y_i^Q can be interpreted as the target.

To balance *exploration* and *exploitation*, the model features a ϵ-greedy policy, in which the agent either chooses an action randomly (exploration), or chooses the action which has the highest $Q(S_t, a_t; \theta_t)$ (exploitation). The ϵ-greedy algorithm works as follows:

if $random(0, 1) < \epsilon$ **then**
  $r \leftarrow random(0, |A|)$;
  $a \leftarrow a_r$, $(a_r \in A)$;
end
else
  $a \leftarrow max(Q(S_t, a; \theta_t))$;
end
return a

Algorithm 1: Epsilon-Greedy Policy

Input: Replay buffer β, target network update frequency τ, learning rate α, discount factor γ, exploration rate ϵ, maximum number of episodes N_{episodes}, maximum number of time steps per episode T_{max}
Output: Learned Q-network parameters θ
Initialize the online $Q(S_0, a; \theta)$ and the target $Q(S_0, a; \theta')$ Q-networks
Initialize replay buffer β
$\theta' \leftarrow \theta$ (Update target network)
for *episode* $= 1$ **to** $N_{episodes}$ **do**
  Initialize S_0
  for $t = 0$ **to** T_{max} **do**
    Select action a_t using the ϵ-greedy policy based on $Q(S_t, a_t; \theta)$ (Algorithm 1)
    Execute action a_t, observe reward r_{t+1} and next state S_{t+1}
    Store $(S_t, a_t, r_{t+1}, S_{t+1})$ in β
    if $|\beta| >$ *batch_size* **then**
      Sample a mini-batch of experiences from β
      for $(S_i, a_i, r_{i+1}, S_{i+1}) \in$ *mini-batch* **do**
        $Y_i^Q \leftarrow r_{i+1} + \gamma\, Q(S_{i+1}, max_a\ Q(S_{i+1}, a; \theta); \theta')$
        $\theta \leftarrow \theta + \alpha[Y_i^Q - Q(S_i, a_i; \theta)]\ \nabla_{\theta_i} Q(S_i, a_i; \theta)$
      end
    end
    if t *mod* $\tau = 0$ **then**
      $\theta' \leftarrow \theta$ (Update target network)
    end
    $S_t \leftarrow S_{t+1}$
  end
end

Algorithm 2: Double Deep Q-Network used in Apollo

3.2.3 Pre-processing

The environment map generated by the SLAM module is inherently discrete, however it does not have a consistent standard size. For compatibility with a DDQN model, it is essential to have a standardized input size. To achieve this, the map,

which can be originally of varying dimensions, often in a rectangular form with differing width and height, is reshaped into a square configuration by augmenting rows or columns to match the shorter side. Subsequently, the map is scaled using spline interpolation to fit a standardized size, in this case set at 200 by 200 pixels. The positions of both the robot and the objective are then encoded using a one-hot encoding scheme within a matrix, mirroring the dimensions of the standardized map size.

3.3. Implementation

3.3.1 Hardware

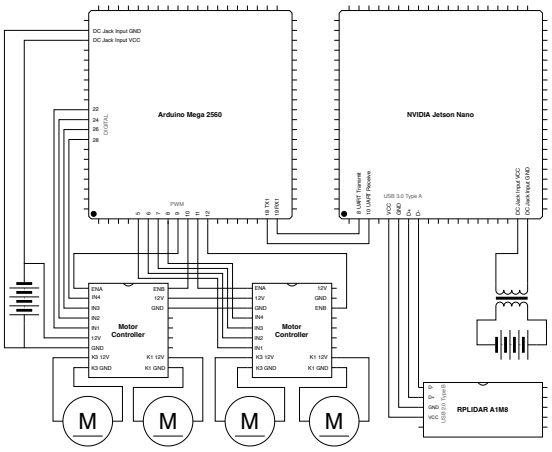

Figure 6: The robot circuit diagram

As shown in the robot's circuit diagram (Figure 6), NVIDIA Jetson Nano was chosen as the onboard computer due to its optimal balance between the ability to run deep learning algorithms effectively and power efficiency. The robot is equipped with a set of rechargeable lithium-ion batteries, providing approximately three hours of autonomy while training. The LiDAR sensor used is the RP LiDAR A1M8, a 2D 360º LiDAR with a 13m range, connected to the Jetson Nano via USB. The NVIDIA Jetson Nano is connected to the Arduino Mega 2560 via Universal Asynchronous Receiver/Transmitter (UART) protocol, using a serial connection between the RX and TX ports on the Arduino and the 40-pin expansion header on the Jetson Nano. The Arduino interfaces with the motor controllers as illustrated in Figure 6.

3.3.2 Software

The implementation of the SLAM and DRL modules was achieved through the utilization of the ROS framework. Within the context of the Apollo framework, a dedicated ROS package housing these modules was created, which encompasses the robot configuration in Unified Robot Description Format (URDF), the capability to load diverse Gazebo environments, configuration files for Cartographer, and the launch configuration for the Gazebo simulation. Furthermore, our ROS package incorporates the DDQN model and the Cartographer node. Notably, this ROS package extends its functionality by providing a real-time visualization map of the environment generated by Cartographer, as well as information about the robot's current position within the map and its trajectory. When operating on the physical robot, this package is executed on the NVIDIA Jetson Nano, with the Gazebo functionality deactivated. In contrast, in a simulated environment, this package launches the Gazebo simulation alongside the core SLAM and DRL functionalities.

The control module is implemented using Arduino. This module contains a script that receives commands from the DRL module in the form of text commands and adjusts the speed and direction of each wheel to achieve the intended movements. The script incorporates pulse-width modulation to enable the robot to move at the desired speed. The DRL module is implemented using TensorFlow and Keras. The neural network model is structured as an artificial neural network, which is depicted in Figure 5, and comprises several fully connected layers: The input layer's size aligns with the state size $|S|$, representing the current state S_t flattened; The network has four hidden layers, containing 64, 32, 16, and 8 units, respectively employing the Rectified Linear Unit (ReLU) as activation function; The output layer matches the number of possible actions $|A|$, providing Q-values $Q(S_t, a; \theta_t)$ for each potential action a. This model employs the Mean Squared Error (Mean Squared Error (MSE)) loss function and the Adam optimizer, which maintains a fixed learning rate of $\alpha = 0.00025$. The discount factor γ for future rewards is set to $\gamma = 0.98$. This means that the agent is forward-looking and prefers to make decisions that not only maximize immediate rewards but also take into account the potential cumulative rewards it can receive in the long run. A higher discount factor implies a greater emphasis on the long-term consequences of actions. This decision was made taking into account the structure of the reward function. During the training process, an ϵ-greedy policy is employed for exploration, with the epsilon value initially set to $\epsilon = 0.3$ and gradually reduced as training progresses. Additionally, experience replay is facilitated using a batch size of 64, since using mini-batches contributes to the model's stability. The network's weights are initialized using a random uniform distribution, ensuring diverse starting points for learning, and the weights of the target network are synchronized with the online network. To enhance versatility and allow for different

Metric	Average Value
Number of collisions per episode	110.55
Path efficiency ratio	73.71
Number of objectives reached per episode	0.53

Table 1: Summary of the average value of each metric

model configurations, the implementation includes features to save and load models as needed.

4. Results

4.1. Experimental setup

An experiment was conducted to assess the effectiveness of the DRL method introduced in Section **??**. This experiment not only aimed to evaluate the proposed DRL approach but also to assess the real-world applicability of the Apollo framework in the development, training and evaluation of RL methods.

In practice, this experiment involves placing the robot in an unknown real-world environment, which comprised a controlled setting with static obstacles in a physical room. The original furniture of the room and cardboard boxes were arranged to create a testing scenario in which the robot had to navigate around obstacles. The glass windows were covered with white sheets of paper at the height of the LiDAR sensor, in order to minimize the problems caused by transparent surfaces being invisible for the LiDAR. It is important to note that during this experiment, no dynamic obstacles were utilized, and no individuals entered the room. Furthermore, there was no interference from any other external factors.

The objective of the experiment was to run the robot for 100 episodes, each comprising 2000 steps. In each episode, the robot's goal is to reach as many objectives as possible. At the start of each episode, a completely random objective position is generated, and the robot's task is to reach it. If, after 2000 steps, the robot fails to reach the objective, the episode ends, and the next episode begins. However, if the robot successfully reaches the objective, another objective position is generated, and this process continues until the maximum number of steps was reached.

4.2. Experimental Results

The experimental results demonstrate the performance of the robot in completing the tasks of the experiment.

The summary of the average value of each metric, as presented in Table 1, reveals that the model continues to struggle with collisions. Additionally, the model's success rate in accomplishing tasks remains low, with an average of 0.53 objectives reached per episode. The plot depicting the number of collisions and the number of objectives reached per episode, as shown in Figure 7, illustrates that, although not in direct correlation, collisions significantly impact the robot's ability to fulfill its tasks.

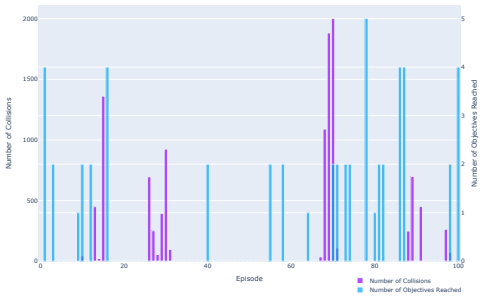

Figure 7: Number of Collisions and number of objectives reached per episode

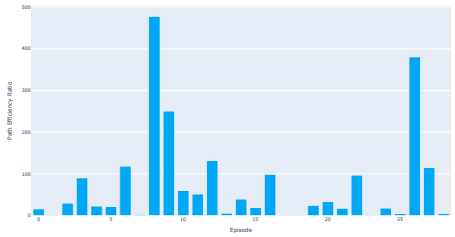

Figure 8: Path Efficiency Ratio (PER) per objective that was reached

When analyzing the average Path Efficiency Ratio (PER) value, as presented in Table 1, it becomes evident that the model encounters challenges in consistently selecting the most efficient actions to construct the shortest path between the robot's initial position (RP) and the objective position (OP). However, a more detailed examination of PER, as depicted in Figure 8, unveils interesting nuances. It reveals that in specific scenarios, the robot successfully chooses the shortest path, indicating its potential for efficient task completion. Nevertheless, this observation also highlights that in other situations, the robot struggles to generate optimal paths, resulting in inefficiency.

The experimental results highlight that the trained model is still in its learning phase. While these results do not meet the desired benchmarks for the immediate application of this method in autonomous mobile robots in healthcare, they still provide valuable insights into how the robot is learning, showing promise for further improvements.

Furthermore, we recognize that while we may not have achieved the desired endpoint, the knowledge gained from these experiences will contribute to the ongoing advancement of this field. The lessons learned, the issues uncovered, and the opportunities for improvement are all part of the narrative that propels us forward.

4.3. Impact of the Apollo Framework

The Apollo framework represents a significant contribution to the fields of robotics and RL. It streamlines the complex process of robot engineering, simulator development, and software infrastructure by providing a unified platform for research and experimentation. By encapsulating key components and reducing barriers to entry for new researchers, Apollo enables a focus on the development of RL algorithms, accelerating the development cycle and providing a cost-effective solution. While Apollo offers substantial advantages, it comes with certain limitations, including hardware constraints, computational limitations, and the static nature of simulated environments. These limitations are important to consider when deploying models trained in simulation to real-world healthcare environments. The Apollo framework democratizes research in robotics and RL, making it accessible to a broader community of researchers and developers. It paves the way for innovation in the field and offers a promising path towards the development of more robust and capable autonomous mobile robots.

5. Conclusions

Autonomous mobile robots hold great potential to revolutionize healthcare. The application of reinforcement learning techniques in such complex medical environments has emerged as a promising approach. To facilitate research and innovation in this domain, we have introduced the Apollo framework, an open-source tool designed to empower the research community in developing reinforcement learning methods for autonomous mobile robots in healthcare. Autonomous robots have the potential to enhance patient care and improve hospital logistics, making them a crucial asset in the medical field. However, deploying such robots in healthcare settings presents a unique set of challenges, including navigating complex and dynamic environments, ensuring safety, and adapting to unpredictable situations. Reinforcement learning, as demonstrated in this work, offers a viable approach to address these challenges. The proposed deep reinforcement learning method combines graph-based SLAM with DDQNs and Transfer Learning. The introduction of the Apollo framework is a significant contribution. This open-source framework simplifies the process of developing and evaluating reinforcement learning methods for autonomous mobile robots. It provides a unified platform that encapsulates hardware, simulation, and ROS integration. With Apollo, researchers can focus on algorithm development and training, accelerating progress in the field. However, it is crucial to acknowledge that the proposed framework and the deep reinforcement learning method have their limitations. Hardware and computational constraints, challenges to be addressed regarding the performance of the model, and the reality gap are some of the limitations to overcome. Additionally, the simulated environment may lack the dynamism and complexity of real-world healthcare settings. The results of this work, while not fully meeting the desired benchmarks for immediate application, provide valuable insights into the potential of the proposed approach. It serves as a foundational step toward more robust and capable autonomous mobile robots in healthcare. The lessons learned, challenges encountered, and opportunities for improvement are part of the ongoing narrative that propels this research field forward. In conclusion, the work presented represents a significant contribution to the advancement of autonomous mobile robots in healthcare and the application of reinforcement learning in this domain. The Apollo framework empowers researchers to drive innovation in the field, and the proposed deep reinforcement learning method, despite its current limitations, paves the way for future research. The road to fully realizing the potential of autonomous mobile robots in healthcare is an ongoing journey, and this work marks a crucial step along that path.

5.1. Future work

Future research in this domain must build upon the foundational progress achieved in integrating autonomous mobile robots into healthcare settings. Crucial areas for exploration involve extended real-world training of the proposed model to enhance its adaptability and performance within dynamic healthcare environments. Varied and extensive training within different medical facilities, each with distinct layouts and conditions, is vital for comprehending the robot's versatility and effectiveness. Additionally, simulator development should include the ability to reset the simulation environment for research and testing purposes, enabling researchers to reinitialize it at each episode. Furthermore, the simulator's capabilities should be expanded to simulate dynamic environments with moving obstacles and dynamic elements, reflecting the dynamic nature of healthcare settings. To gain deeper insights into the proposed deep reinforcement learning (DRL) algorithm, future research should conduct comparative analyses with other state-of-the-art DRL algorithms. By focusing on these aspects, the field of autonomous mobile robots in healthcare will continue to evolve, facilitating the development of more robust, adaptable, and capable robots to enhance healthcare support and assistance significantly.

References

[1] H. S. Ahn, M. H. Lee, and B. A. MacDonald. Healthcare robot systems for a hospital environment: Carebot and receptionbot. In *2015 24th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, pages 571–576, 2015.

[2] S. Barrett, M. E. Taylor, and P. Stone. Transfer learning for reinforcement learning on a physical robot. 1, 2010.

[3] J. BačÍK, F. ĎUrovský, M. Biroš, K. Kyslan, D. Perduková, and S. Padmanaban. Pathfinder–development of automated guided vehicle for hospital logistics. *IEEE Access*, 5:26892–26900, 2017.

[4] Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

[5] R. Bloss. Mobile hospital robots cure numerous logistic needs. *Industrial Robot: An International Journal*, 38(6):567–571, 2011.

[6] P. Dayan and Y. Niv. Reinforcement learning: The good, the bad and the ugly. *Current Opinion in Neurobiology*, 18(2):185–196, 2008. Cognitive neuroscience.

[7] I. Doroftei, V. Grosu, and V. Spinu. *Omnidirectional mobile robot-design and implementation*. INTECH Open Access Publisher London, UK, 2007.

[8] J. Evans, B. Krishnamurthy, B. Barrows, T. Skewis, and V. Lumelsky. Handling real-world motion planning: a hospital transport robot. *IEEE Control Systems Magazine*, 12(1):15–19, 1992.

[9] Y. Han, I. H. Zhan, W. Zhao, J. Pan, Z. Zhang, Y. Wang, and Y.-J. Liu. Deep reinforcement learning for robot collision avoidance with self-state-attention and sensor fusion. *IEEE Robotics and Automation Letters*, 7(3):6886–6893, 2022.

[10] W. Hess, D. Kohler, H. Rapp, and D. Andor. Real-time loop closure in 2d lidar slam. pages 1271–1278, 2016.

[11] J. Hua, L. Zeng, G. Li, and Z. Ju. Learning for a robot: Deep reinforcement learning, imitation learning, transfer learning. *Sensors*, 21(4):1278, 2021.

[12] C. Jiahai, L. You, G. Jianping, and W. Yakun. Application of da vinci surgical robotic system in hepatobiliary surgery. *International Journal of Surgery and Medicine*, 4:1, 01 2017.

[13] L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285, 1996.

[14] Y. Kato, K. Kamiyama, and K. Morioka. Autonomous robot navigation system with learning based on deep q-network and topological maps. In *2017 IEEE/SICE International Symposium on System Integration (SII)*, pages 1040–1046, 2017.

[15] J. Kober, J. A. Bagnell, and J. Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274, 2013.

[16] S. Kuutti, R. Bowden, Y. Jin, P. Barber, and S. Fallah. A survey of deep learning applications to autonomous vehicle control. *IEEE Transactions on Intelligent Transportation Systems*, 22(2):712–733, 2021.

[17] S. Levine, C. Finn, T. Darrell, and P. Abbeel. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016.

[18] L. Manuelli and P. Florence. Reinforcement learning for autonomous driving obstacle avoidance using lidar. 2017.

[19] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

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

[21] H. Nguyen and H. La. Review of deep reinforcement learning for robot manipulation. pages 590–595, 2019.

[22] A. G. Ozkil, Z. Fan, S. Dawids, H. Aanes, J. K. Kristensen, and K. H. Christensen. Service robots for hospitals: A case study of transportation tasks in a hospital. pages 289–294, 2009.

[23] T. Peng, J. Qian, B. Zi, J. Liu, and X. Wang. Mechanical design and control system of an

omni-directional mobile robot for material conveying. *Procedia CIRP*, 56:412–415, 2016. The 9th International Conference on Digital Enterprise Technology – Intelligent Manufacturing in the Knowledge Economy Era.

[24] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. pages 3803–3810, 2018.

[25] J. A. Pepito and R. Locsin. Can nurses remain relevant in a technologically advanced future? *International Journal of Nursing Sciences*, 6(1):106–110, 2019.

[26] A. Romero, F. Bellas, J. A. Becerra, and R. J. Duro. Motivation as a tool for designing lifelong learning robots. *Integrated Computer-Aided Engineering*, 27(4):353–372, 2020.

[27] O. G. Selfridge, R. S. Sutton, and A. G. Barto. Training and tracking in robotics. pages 670–672, 1985.

[28] M. A. Sharbafi, C. Lucas, and R. Daneshvar. Motion control of omni-directional three-wheel robots by brain-emotional-learning-based intelligent controller. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 40(6):630–638, 2010.

[29] Z. Sun, B. Wu, C.-Z. Xu, S. E. Sarma, J. Yang, and H. Kong. Frontier detection and reachability analysis for efficient 2d graph-slam based active exploration. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2051–2058. IEEE, 2020.

[30] L. Tai, G. Paolo, and M. Liu. Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation. pages 31–36, 2017.

[31] M. Takahashi, T. Suzuki, H. Shitamoto, T. Moriguchi, and K. Yoshida. Developing a mobile robot for transport applications in the hospital domain. *Robotics and Autonomous Systems*, 58(7):889–899, 2010.

[32] R. Tasaki, M. Kitazaki, J. Miura, and K. Terashima. Prototype design of medical round supporting robot “terapio”. pages 829–834, 2015.

[33] S. G. Tzafestas. *Introduction to mobile robot control*. Elsevier, 2013.

[34] E. Tzeng, C. Devin, J. Hoffman, C. Finn, P. Abbeel, S. Levine, K. Saenko, and T. Darrell. Adapting deep visuomotor representations with weak pairwise constraints. pages 688–703, 2020.

[35] H. Van Hasselt, A. Guez, and D. Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.

[36] Y. Wang, H. He, and C. Sun. Learning to navigate through complex dynamic environment with modular deep reinforcement learning. *IEEE Transactions on Games*, 10(4):400–412, 2018.

[37] K. Weiss, T. M. Khoshgoftaar, and D. Wang. A survey of transfer learning. *Journal of Big data*, 3(1):1–40, 2016.

[38] S. Wen, Y. Zhao, X. Yuan, Z. Wang, D. Zhang, and L. Manfredi. Path planning for active slam based on deep reinforcement learning under unknown environments. *Intelligent Service Robotics*, 13(2):263–272, 2020.

[39] K. Yokoyama and K. Morioka. Autonomous mobile robot with simple navigation system based on deep reinforcement learning and a monocular camera. pages 525–530, 2020.

[40] K. Zhu and T. Zhang. Deep reinforcement learning based mobile robot navigation: A review. *Tsinghua Science and Technology*, 26(5):674–691, 2021.

[41] Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, L. Fei-Fei, and A. Farhadi. Target-driven visual navigation in indoor scenes using deep reinforcement learning. pages 3357–3364, 2017.

[42] Z. Zhu, K. Lin, and J. Zhou. Transfer learning in deep reinforcement learning: A survey. *arXiv*, abs/2009.07888, 2020.

[43] Q. Zou, Q. Sun, L. Chen, B. Nie, and Q. Li. A comparative analysis of lidar slam-based indoor navigation for autonomous vehicles. *IEEE Transactions on Intelligent Transportation Systems*, 23(7):6907–6921, 2022.

RL	Reinforcement Learning
DRL	Deep Reinforcement Learning
SLAM	Simultaneous Localization and Mapping
TL	Transfer learning
DL	Deep Learning
ANN	Artificial Neural Networks
MLP	Multi-layer Perceptron
MSE	Mean Squared Error
MDP	Markov Decision Process
UCB	Upper Confidence Bound
DQL	Deep Q-learning
DQN	Deep Q-Network
DDQN	Double Deep Q-Network
LiDAR	Light Detection and Ranging
LSTM	Long Short-term Memory
SARSA	State–action–reward–state–action
BOM	Bill of Materials
ROS	Robotic Operating System
UART	Universal Asynchronous Receiver/Transmitter
URDF	Unified Robot Description Format
ReLU	Rectified Linear Unit