EasyChair Preprint

№ 13789

AI-Powered Mobile Robot Navigation System with Prioritized Double Q-Network (PDQN) and Multi-Objective Pumafish Optimization Algorithm (MOPFOA)

Kuldeep Singh

July 2, 2024

AI-Powered Mobile Robot Navigation System with Prioritized Double Q-Network (PDQN) and Multi-Objective Pumafish Optimization Algorithm (MOPFOA)

Abstract

Mobile robots are helping multiple sectors, including mining, health, space, the military, surveillance, and agriculture. Mobile robots (MR) depend mainly on complex algorithms for safe and efficient navigation. Perception, path planning, localization, and motion control are the four needs for mobile robot navigation. Although most mobile robots operate in dynamic situations, the number of algorithms able to navigate robots in such conditions is limited. This paper proposes novel reinforcement learning techniques and hybrid metaheuristic optimization for mobile robot navigation systems. Initially, the Multi-Objective Pumafish Optimization Algorithm (MOPFOA) is utilized for creating an efficient task schedule by minimizing total task completion time, path length, energy consumption, and robot idle time. After task scheduling, path planning occurs by training a deep reinforcement learning agent like a prioritized double Q-network (PDQN). This agent will plan collision-free paths, considering dynamic obstacles and optimizing multiple objectives. Additionally, novel techniques such as Multi-Agent Deep Deterministic Policy Gradient (MADDPG) are used for cooperative multi-robot navigation. After that, vision transformers (ViTs) are used for precise obstacle detection. Then the avoidance algorithm will use a hybrid of transformer-based detection and deep reinforcement learning to dynamically adjust the robot's path. Lastly, the system will guide the robot to a charging station when battery levels reach a threshold. The Python tool is used for implementing this work, and the energy consumption of the proposed work is 2.67.

Keywords: Deep reinforcement learning (DRL), Vision Transformer (ViT), Double Q-Network (DQN), Puma Optimizer (PO), Pufferfish Optimization Algorithm (POA).

1. Introduction

Robots serve multiple purposes in our daily lives, including medical assistance, cleaning, autonomous driving, and military operations. Mobile robots must navigate without competing with static or dynamic challenges in those mentioned applications. Navigation is the procedure through which an MR goes about the environment to complete a certain job [1, 2]. Autonomous navigation refers to a robot's ability to operate in a given environment without the help of an exterior controller, including a social. Autonomous navigation is an important area of study in

mobile robotics. Improvements in AI and computer vision have led to significant advancements in automated mobile robot navigation technology [3, 4]. Creating mobile robots autonomously in the real world remains a challenging task. Map-building navigation involves localization, path planning, and map construction, like simultaneous localization as well as mapping (SLAM). Utilizing map planning and control by transmitting high-dimensional observations, including camera images, into three-dimensional poses on the map [5-7].

It is easy to make sure that the global path is optimal; however, there are some limitations. Developing a complete environmental map is time-consuming and requires specialist knowledge. Updating and maintaining the location map over time can be expensive, especially with dynamic changes. The robot's control efficiency depends on its mathematical model, which is sometimes simplified or linearized, reducing the navigation system's accuracy [6, 8]. Mapless navigation eliminates the need for a map and enables straight mapping among sensory inputs as well as robot actions, making it a popular alternative to map-based navigation. Mobile robot navigational systems are at the top of robotics development and research, allowing robots to move easily while carrying out tasks in a variety of situations [9, 10]. These systems use algorithms, sensors, and operating mechanisms to help robots understand the environment, plan pathways, and perform movements. A navigation system's key components include localization, perception, path planning, mapping, and control of motion. Each component is essential for ensuring that the robot navigates correctly and securely, particularly in dynamic and unpredictable environments [11–13].

Mapless navigation is commonly used for jobs without stated destinations, such as collision avoidance, which is well known in its local coordinate border. Here, mapless navigation has equivalent behavior-based navigation, as it requires high-level reasoning based on prior environmental knowledge. Deep reinforcement learning (DRL) has become more popular over the past four years. Two successful examples of merging RL and DNNs yielded impressive as well as interesting outcomes [14–16]. The combined use of AI deep learning with optimization approaches has provided more powerful mobile robot navigation applications. For example, with autonomous delivery networks, robots can navigate difficult metropolitan environments while avoiding people and traffic and delivering products quickly. In industrial environments, mobile robots can carry materials and goods autonomously, optimizing routes for rapid delivery and security [17–19]. The integration of such technologies improves robot capabilities in rescue and search operations, allowing them to navigate hazardous settings and identify

survivors with great precision. With these advanced technologies, they promise to increase the potential uses of mobile robots across multiple sectors and industries, making them important tools in our more automated world [20].

1.1 Motivation

Mobile robots have gained popularity in a variety of sectors, like healthcare, manufacturing, logistics, and self-driving cars. These robots have the capacity to explore dynamic situations with their effectiveness and dependability. Traditional navigation systems frequently struggle with immediate decision-making and adaptation in a wide range of unexpected circumstances. Existing mobile robot navigation systems typically utilize static path planning algorithms and traditional RL techniques, which might not be adequate for dealing with the complexities of dynamic settings. To overcome these problems, advanced AI-powered navigation systems are introduced to adapt to changing situations while optimizing many objectives at the same time. The combination of powerful deep reinforcement learning methods and robust optimization algorithms represents a promising solution for improving mobile robot navigation skills. The primary contribution of this research is to develop and evaluate an AI-powered MR navigation scheme for dynamic decluttering applications. This includes:

- Implementing sophisticated multi-objective optimization techniques for task scheduling and path planning, considering decluttering efficiency, path length, and energy consumption.
- Training for Deep Reinforcement Learning agent to plan collision-free paths in dynamic environments, adapting to unforeseen obstacles.
- Integrating an advanced motion control algorithm for smooth and accurate robot movements.
- Incorporating a robust recharging strategy within the task schedule.

1.2 Paper Organization

The remaining section of this study is represented in a paper organization. Section 2 contains related work; section 3 contains system design; section 4 has proposed methodology; section 5 contains results as well as a discussion part; and section 6 contains a conclusion and future study.

2. Related Works

Surmann et al. [21] suggested a DRL that allows autonomous MR navigation within interior settings. This study provides a modern concept for unsupervised self-learning robots navigating within an unfamiliar area without a map or plan. The robot receives input through a 2D laser scanner and an RGB-D camera, along with orientation to the target. The asynchronous Advanced Actor-Critic network generates angular and linear speeds on the robot. To accelerate learning, the navigator network was trained inside a self-implemented simulation environment before being attached to the actual robot. To prevent overfitting, train minor networks as well as incorporate Gaussian noise into the input laser information.

For the navigation of mobile robots, Lee et al. [22] introduced deep reinforcement learning. This research proposes two DQL agents, DQN and DDQN, for helping mobile robots study avoiding collisions as well as navigation within new environments. To navigate autonomously in an unknown environment, a DNN is used to detect the target item, followed by the DQN or DDQN algorithm.

Li et al. [23] suggested a deep learning-based robot vision navigation system in an edge computing environment. The method employs a cascaded DCN as well as hybrid expanded convolution fusion for processing images from a vision scheme. Then the path of the needed images was extracted using the enhanced Hough transform technique. The position of farming robots was changed to provide autonomous navigation. This existing technique is verified using both non-interference and noisy experimental settings.

For mobile robot navigation, Liu et al. [24] introduced a lifelong learning method. This letter suggests and executes the initial self-supervised Lifelong Learning for Navigation framework (LLNf). The robot uses a static sampling method for predictive control, which doesn’t expand with experience. It can identify inadequate actions, which contain similar scenarios with good actions, learn, and continuously improve its navigation. Furthermore, within a multi-environment context, LLfN can adapt to new surroundings while not overlooking the old ones. Replicated trials evaluate LLfN's ability to learn in and across environments.

For the autonomous navigation of mobile robots, Sadeghi Esfahlani et al. [25] introduced deep convolutional neural networks. This framework combines a 2D laser digital scanner, an RGB-D MYNTEYE photographic camera, and inertial measurement units (IMU) into an embedded scheme for DL. Real-time decision-making as well as tests were shown using an integrated image capture as well as a signal processing scheme for continuous data investigation. Then we used innovative real-time graph-based SLAM and Deep-CNN for mapping indoor surroundings. Enforcing Deep-CNN increased RTAB-Map SLAM performance. Table 1 shows the existing comparison table.

Tabel 1: Existing Comparison

Authors	Techniques	Advantages	Limitations
Surmann et al. [21]	GA3C	Actor-critical methods learn efficiently with less data.	Computational cost
Lee et al. [22]	DNN	It can handle complex data.	DNNs are vulnerable to overfitting.
Li et al. [23]	DNN and Hough transform	It can detect various shapes.	Very expensive
Liu et al. [24]	LLfN	Robots learn autonomously, eliminating reliance on labeled data.	Training and testing will take time.
Sadeghi Esfahlani et al. [25]	Deep-CNN and SLAM	Real-time graph-based SLAM	Computational cost

2.1 Problem Statement

While A3C significantly increases learning efficiency by training several agents at the same time, it frequently needs large amounts of CPU resources and can be inconsistent during training, particularly in highly dynamic situations. DQN has achieved outstanding success in a variety of applications; however, it faces limitations like overestimation bias and weak

convergence. The Hough Transform, which is widely employed for feature detection, is computationally demanding and frequently fails to cope with noise and complexity in real-world settings. Graph-based SLAM delivers accurate and consistent mapping. However, it is computationally costly and scales poorly in large regions. The Lifelong Learning for Navigation System intends to continuously modify and enhance the robot's navigation abilities over time. Because of such limitations in existing studies, this proposed study creates a novel prioritized double Q-network and hybrid optimization for mobile root navigation systems.

3. Software System Design

The mobile robot's software system was designed using Ubuntu 18.04. The architecture included three layers: application, control, and driver. The ROS-based control layer was the most important component. The system gathers, fuses, and processes data gathered by robot sensors before constructing diagrams, planning paths, and navigating autonomously based on control commands. The ROS framework's distributed architecture allows for individual module design, compilation, and loose coupling at runtime. Figure 1 shows the architecture of a mobile robot navigation system. The application layer handles high-level operations like scheduling and navigation. It provides the general goal for the robot's navigation. The control layer connects the application and driver layers. It uses environmental information, like a map it develops and the goal path, to plan where the robot should navigate. The driver layer is responsible for directly controlling the robot's movements. It turns the control layer's navigation commands into robot-specific motor controls, like linear as well as angular velocities.

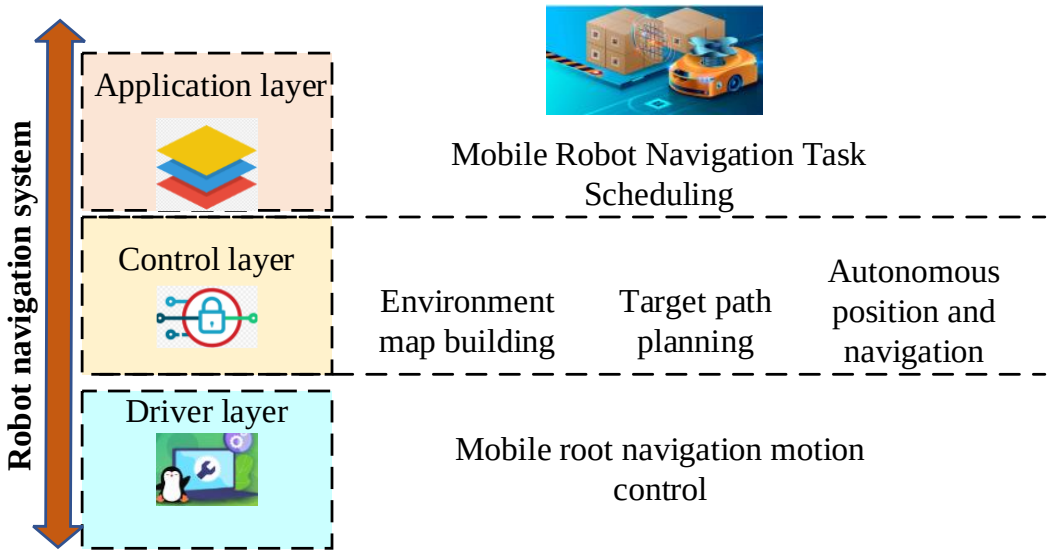

Figure 1: Architecture of mobile robot navigation system

4. Proposed methodology

Mobile robots are increasingly adopted for tasks in various environments. This research proposes an AI-powered mobile robot navigation system specifically designed for dynamic decluttering applications. The system will leverage advanced deep learning for environment perception, sophisticated multi-objective optimization for task planning, and DRL for advanced path planning in active environments. This project aims to develop a robust, efficient, and adaptable solution for automated decluttering tasks. Figure 2 shows the architecture of the proposed method. Here, the Multi-Objective Pumafish Optimization Algorithm is used for task scheduling and will explore and exploit search spaces to find a set of solutions representing trade-offs between conflicting objectives. Path planning is taking place through a prioritized double Q-network, which is used for optimizing multiple objectives. A novel technique like MADDPG is used for multi-robot navigation. A vision transformer is used for detecting obstacles in the path. After that, the robot will recharge under the guidance of the system.

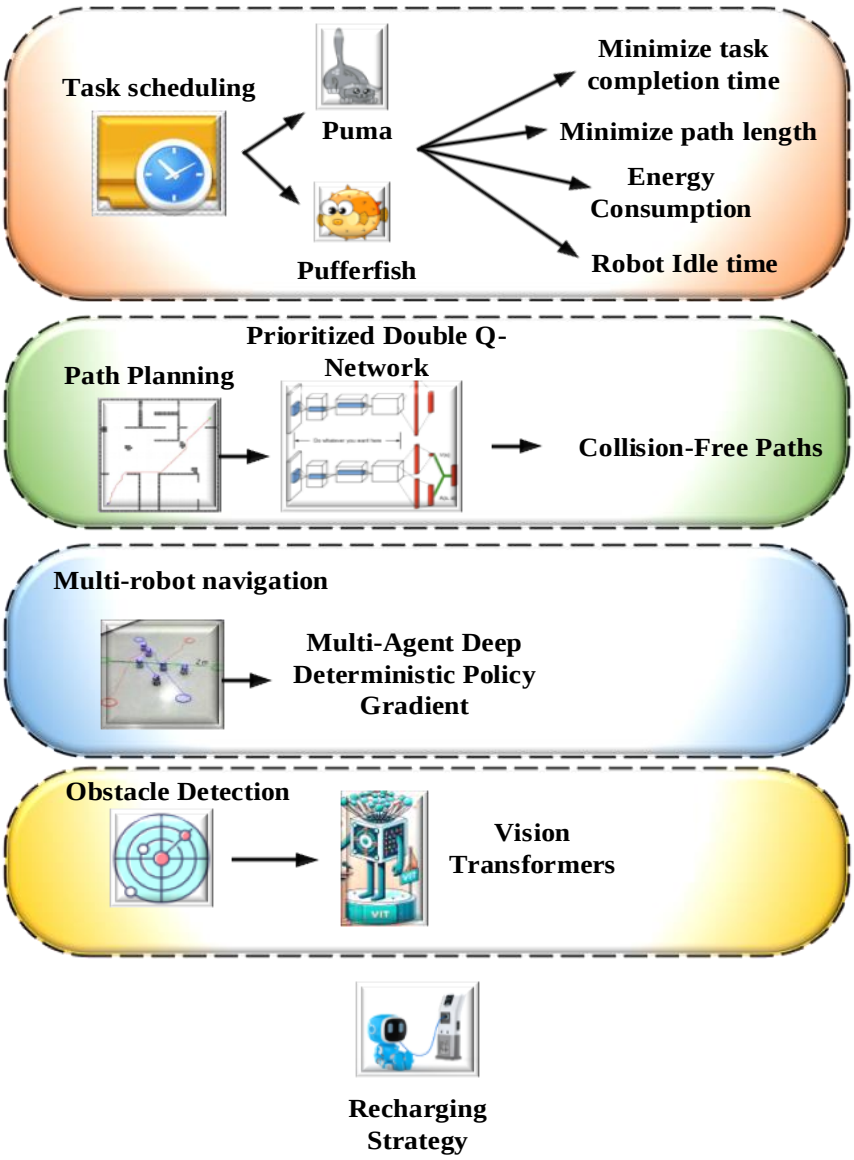

Figure 2: Architecture of Proposed methods

4.1 Task Scheduler using Multi-Objective Pumafish Optimization Algorithm

The Multi-Objective Pumafish Optimization Algorithm (proposed) is used for task scheduling, which combines the best features of both Puma and Pufferfish optimization. This hybrid method is designed to handle many objectives, such as maximizing resource use, lowering job completion time, and balancing stress.

4.1.1 Initialization stage

The Puma Optimization Algorithm is motivated by cougar hunting habits and excels at exploring and discovering multiple options in the early stages of the search. PO members provide values to the problem's decision variables based on their search space location. Each

PO member provides an alternate approach to the problem and can be described mathematically as a vector with each element representing a decision variable. A matrix can be used to model the community of vectors and initializes the primary location of each PO member is shown in Equation 1 and 2.

$$Y = \begin{bmatrix} Y_1 \\ \vdots \\ Y_j \\ \vdots \\ Y_M \end{bmatrix}_{M\times n} = \begin{bmatrix} y_{1,1} & \cdots & y_{1.d} & \cdots & y_{1,n} \\ \vdots & \ddots & \vdots & \iddots & \vdots \\ y_{j,1} & \cdots & y_{j,d} & \cdots & y_{j,n} \\ \vdots & \iddots & \vdots & \ddots & \vdots \\ y_{M,1} & \cdots & y_{M,d} & \cdots & y_{M,n} \end{bmatrix}_{M\times n} \tag{1}$$

$$y_{j,d} = ka_d + t.\left(va_d - ka_d\right) \tag{2}$$

Where, Y indicates the PO population matrix, Y_j stands for the j^{th} candidate solution, $y_{j,d}$ indicates the d^{th} dimension within the search area, M stands for the amount of population followers, n denotes the amount of decision parameters, t represents a random quantity from [0, 1], and ka_d as well as va_d represent the upper as well as lower bounds of the d^{th} decision parameter, respectively. The evaluated results for the difficult objective function could be written as a vector using the Equation 3.

$$E = \begin{bmatrix} E_1 \\ \vdots \\ E_j \\ \vdots \\ E_M \end{bmatrix}_{M\times 1} = \begin{bmatrix} E\left(Y_1\right) \\ \vdots \\ E\left(Y_j\right) \\ \vdots \\ E\left(Y_M\right) \end{bmatrix}_{M\times 1} \tag{3}$$

The vector E indicates the evaluated objective purpose.

4.1.2 Exploration phase using Puma Optimization

The exploration phase, which employs Puma Optimization, focuses on varying the search process in order to identify a wide range of possible answers. In this phase, many techniques are used to make sure that the method effectively explores different areas of the solution space. When designing a search agent for a feature, the position of other search agents with higher objective function values is used to determine the candidate search agent for attack. To identify the set of search agent for every population member, use the Equation 4.

$$ZQ_j = \left\{ Y_l : E_l < E_j \; and \; l \neq j \right\}, where \; j = 1, 2, .., M \; and \; l \in \left\{ 1, 2, .., M \right\} \qquad (4)$$

Where, ZQ_j indicates the established potential pufferfish locations in the j^{th} predator, Y_l indicates the population member through a higher detached function values of j^{th} predator. If the goal function value improves at a new position, it will replace the prior location of the equivalent member, as per Equation 5 and 6.

$$y_{j,i}^{q1} = y_{j,i} + t_{j,i} \cdot \left(PQ_{j,i} - J_{j,i} \cdot y_{j,i} \right) \qquad (5)$$

$$Y_j = \begin{cases} Y_j^{q1}, \; E_j^{q1} \leq E_j \\ Y_j, \quad else \end{cases} \qquad (6)$$

Where, PQ_j represents the randomly chosen search agent for the j^{th} predator from the ZQ_j set. Where, $PQ_{j,i}$ is the i^{th} dimension, Y_j^{q1} indicates the novel position estimated through j^{th} predator depending upon the first stage of proposed PO.

4.1.3 Exploitation phase using Pufferfish Optimization

In this phase of POA, population members' locations are updated utilizing a imitation of a pufferfish's defense device opposing feature attacks. When threatened by a predator, a search agent forms a scope of pointed spine by satisfying its flexible stomach by water. When calculating a new position for a POA member, objective function values are compared to determine if the new position provides an improved solution for the issues. If the outcome is yes, the novel position is accepted through the relevant POA search agent. If not, the innovative position is improper and attains a weaker result, therefore the member remains in their previous position. The updating process of every POA member depends on increasing the cost of the objective function is given in Equation 7 and 8.

$$y_{j,i}^{q2} = y_{j,i} + \left(1 - 2t_{j,i} \right) \cdot \frac{va_i - ka_j}{r} \qquad (7)$$

$$Y_j = \begin{cases} Y_j^{q2}, \; E_j^{q2} \leq E_j \\ Y_j, \quad else \end{cases} \qquad (8)$$

Where, Y_j^{q2} represents the novel position estimated through j^{th} predator depending upon second stage of the suggested POA, $y_{j,i}^{q2}$ indicates the i^{th} dimension, E_j^{q2} indicates the OF

value, $t_{j,i}$ indicates random integers within the interval [0, 1], as well as r represents the iteration timer. Figure 3 shows the flow work of hybrid optimization techniques.

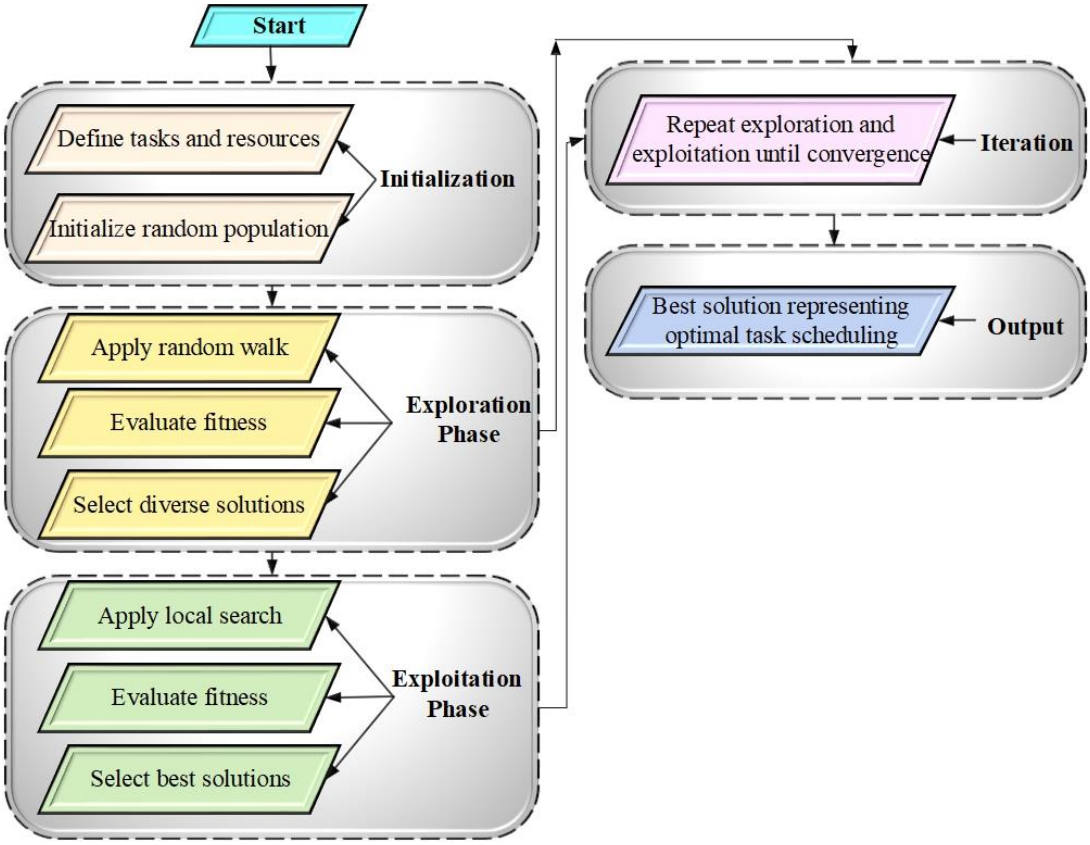

Figure 3: Flow chart of Hybrid Optimization

4.2 Path Planning Using Prioritized Double Q-Network

Train a Deep Reinforcement Learning agent like Prioritized Double Q-Network (PDQN), which is the combination of Double DQN with Prioritized Experience Replay in a simulated environment. The agent will plan collision-free paths considering dynamic obstacles and optimizing multiple objectives, including navigation speed, collision risk, and energy usage. In this research, apply DDQN to a prioritized replay of experiences method. This method uses two types of neural networks that have various parameters, temporary freezing the correlation technology, and prioritized experience replay (PER) to address overestimation in natural DQN and decrease the quantity of knowledge needed for learning. Choosing the minibatch for reducing iterations and training time for the model. Figure 4 shows the architecture of PDQN.

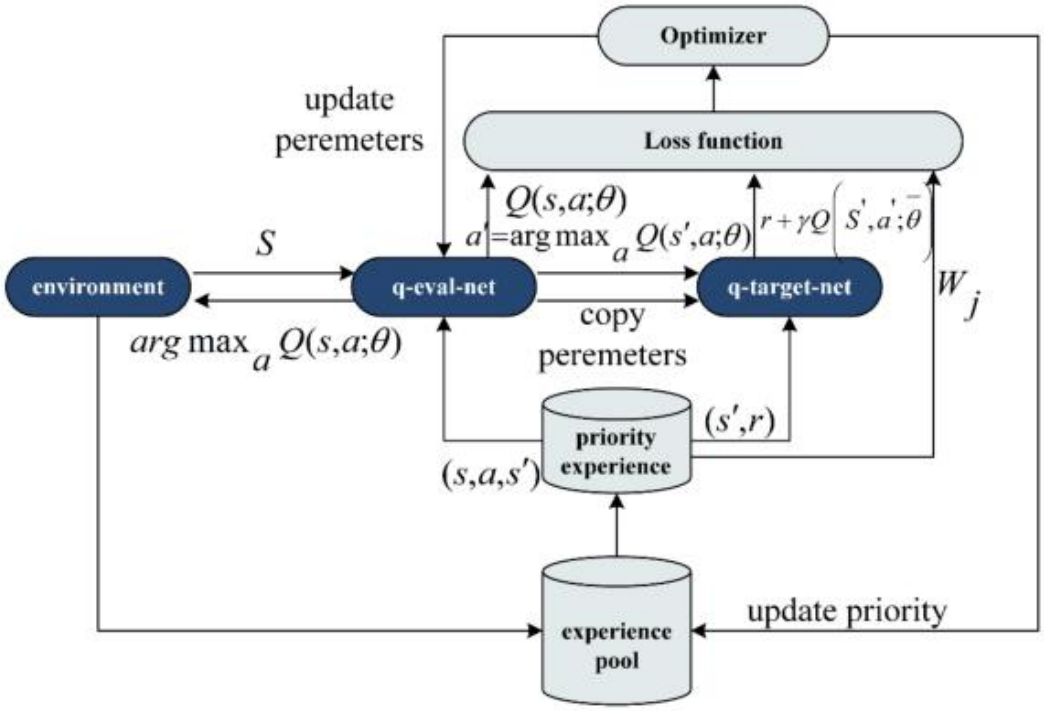

Figure 4: Architecture of PDQN

The agent analyzes the atmosphere's state, next chooses an action using q-eval-net that proceeds into the environment. Then it provides the reward as well as the following state for the agent. Then this agent modifies the state also saves (p,b,t,q') to the experience lake. During the learning method, the agent selects experiences through the pool and selects most beneficial action for the next step using q-eval-net. After that, the activity and state were fed through q-target-net for generating q-target. This state is fed through q-eval-net, which calculates the action's q-eval as well as the error among desired q-eval. Then gradient descent algorithm changes the variables. Lastly, the importance of understanding is changed within the pool. Later an established chapter interval, q-eval-net variables are transferred to q-target-net. Experience's priority and possibility of selection were given in Equation 9 and 10:

$$q_j = \left|\delta_j\right| + \varepsilon \tag{9}$$

$$q_j = \frac{q_j^{\alpha}}{\sum_{j=1}^{l} q_j^{l}} \tag{10}$$

Where, q_j has the importance of experience; δ_j represents the TD fault of experience, ε indicates the hyperparameter that prevents experience with TD equivalent to 0 being chosen, $q(j)$ becomes the possibility of sampling that have being designated; α represents the

hyperparameter that controls the sampling preference in uniform along with greedy sampling; and $\alpha \in (0,1)$. When α = 0, sampling is uniform, when α = 1, sampling is greedy. Prioritized experience replay changes sample distribution, possibly leading to varying model values. Then employ importance sampling to ensure that every sample has a unique chance of selection and has the similar impact on GD. The sample weight v_j, which was included in the loss function and this has been given in Equation 11 and 12.

$$v_j = \left(\frac{1}{M} * \frac{1}{q(j)} \right)^{\beta} \tag{11}$$

$$v_j = \left(\frac{q(j) * M}{\max y_j (v_j)} \right)^{-\beta} \tag{12}$$

Where, v_j represents the weightiness of the experience within the retention pool, v_j becomes the weightiness on the knowledge within the minibatch, M represents the size of the memory, and β represents a hyperparameter utilized for offsetting the influence of ranked experience playback on conjunction outcomes $\beta \in (0,1)$. The priority samples of loss function are calculated by Equation 13 to 15,

$$q_{eval} = S(p_r, b_r; \theta_r) \tag{13}$$

$$q_{target} = t + \gamma Q\left(p_{r+1}, \arg\max_b Q(p_{r+1}, b; \theta_r); \theta_r^- \right) \tag{14}$$

$$loss_{function} = \frac{1}{l} \sum_{i=1}^{l} v_i \left(q_{eval} - q_{target} \right)^2 \tag{15}$$

So, this technique can minimize the obstacles in the path.

4.2.1 Multi-Agent Deep Deterministic Policy Gradient (MADDPG)

MADDPG provides an effective technique for cooperative navigation within multi-robot schemes. It is effectively a RL method designed for scenarios with numerous agents working together. Figure 5 shows the architecture of MADDPG model.

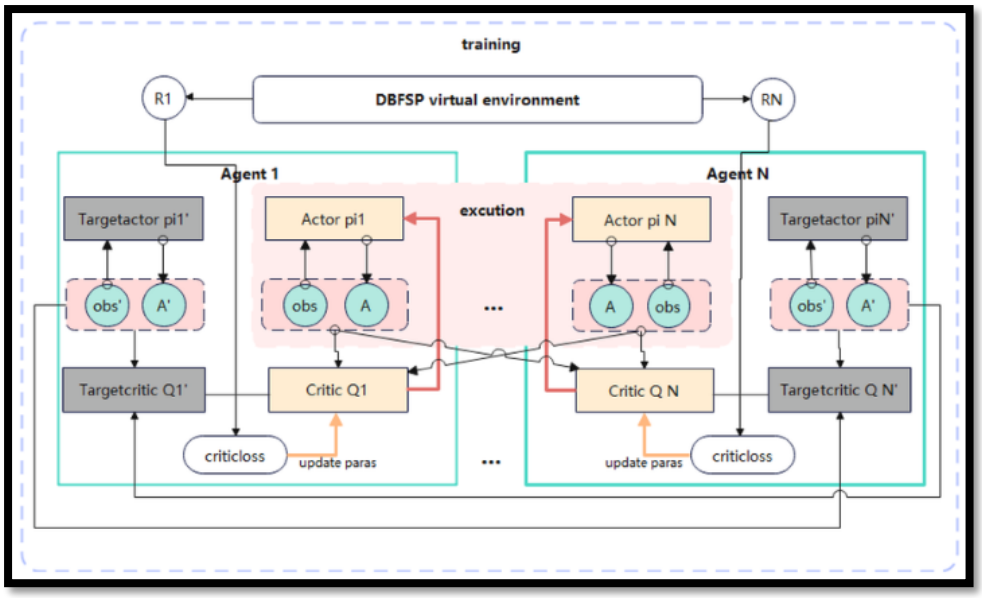

Figure 5: Architecture of MADDPG model

In RL studies, DDPG developed as PG → DPG → DDPG. The PG solution technique uses a probability distribution function $\pi_\theta(b/p)$ to describe the optimal strategy for each step, which are followed by action sampling based on the probability distribution at each step to produce the current optimal action value $S^\pi(p,b)$. The procedure of generating actions is random, and the learned policy is equally stochastic. A gradient of the approach can be expressed as:

$$\nabla_\theta I(\theta) = F_{p\square q^\pi, b\square \pi_\theta}\left[\nabla_\theta \log \pi_\theta(b/p) Q^\pi(p,b)\right] \tag{16}$$

After getting the stochastic strategy, researchers need to sample the optimal probability distribution to determine the exact action. Frequent sample of a high-dimensional vector can be highly computational. The function μ, represents the ideal behavior policy which eliminates the necessity for sampling. The policy gradient can be expressed as Equation 17:

$$\nabla_\theta I(\theta) = F_{p\square D}\left[\nabla_\theta \mu_\theta(b/p) \nabla_b Q^\mu(p,b)\Big|_{b=\mu_\theta(p)}\right] \tag{17}$$

The centralised states' action value function is updated utilizing temporal difference error, identical to the DQN, as seen in Equation 18.

$$K\left(\theta_j\right)=F_{y,b,t,y'}\left[\left(Q_j^{\mu}\left(y,b_1,..,b_m\right)-x\right)^2\right],$$
$$x=t_j+\gamma Q_j^{\mu'}\left(y',b'_1,..,b]_m\right)\Big|_{b_i'=\mu_i'(O_i)} \tag{18}$$

Where, μ' represent the set of target policies having delayed parameters θ_j'. Training data is randomly obtained from the experience replayed buffer. Each agent has four different networks to learn: actor, critic, and two target networks.

4.3 Obstacle Detection using Vision Transformers (ViT)

The Vision Transformer (ViT) is a DL model which has gained importance for its effectiveness in computer vision applications once dominated by CNN. The Vision Transformer can also use for detecting obstacles. The Vision Transformer accepts an input image that frequently divides into patches. Every patch represents a smaller rectangular or square portion of the original image. Patch Embedding is the linear embedding of an input image into a lower-dimensional vector. This embedding technique helps in the conversion of the image's spatial information into a suitable processing of Transformer design. Figure 6 shows the architecture if vision transformer.

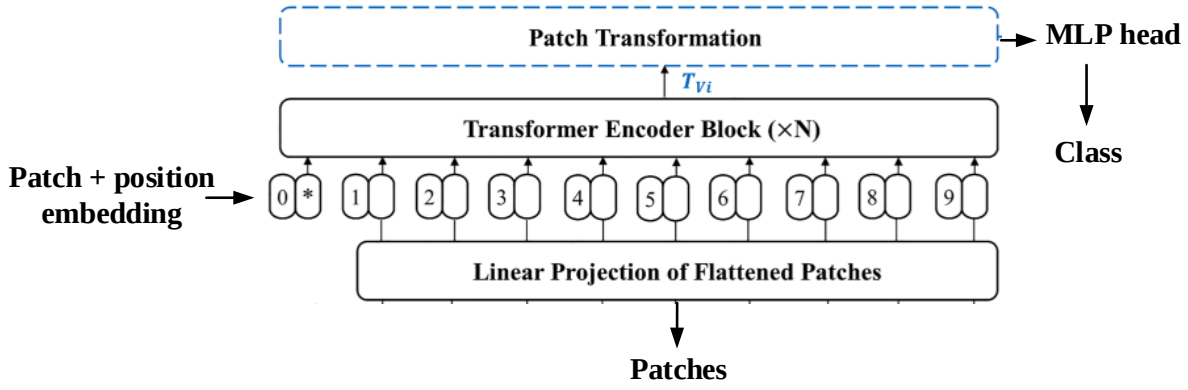

Figure 6: Architecture of ViT transformer

The Transformer Encoder is important to the Vision. Transformer is made up of many Transformer encoder blocks. Each block usually contains some mechanisms. The model's Multi-head Self-Attention technique allows it to focus on different portions of the image while analyzing each patch, collecting global dependencies. Feedforward NN analyze the attention mechanism's output to generate each patch's final features. Layer Normalization as well as Residual Connections are utilized to stabilize and enhance information flow across the network. After the patches have been handled by the Transformer blocks, the model gets fitted with a

classification head. For obstacle detection, this head often includes a few more layers that convert the final features to predictions. The model's final result could be a probability distribution across different types of obstacles or a segmentation map highlighting obstacle regions. The avoidance algorithm will use a hybrid of Transformer-based detection and Deep Reinforcement Learning to dynamically adjust the robot's path.

4.3.1 Avoidance Algorithm

An avoidance algorithm supports a robot in navigating its surroundings by detecting and avoiding obstacles. It utilizes sensor data to identify obstacles and subsequently calculates a path to get to its target. The avoidance system will utilize a combination of Transformer-based detection as well as Deep Reinforcement Learning to continuously alter the robots naturally. Figure 8 shows the flow chart of avoidance algorithm.

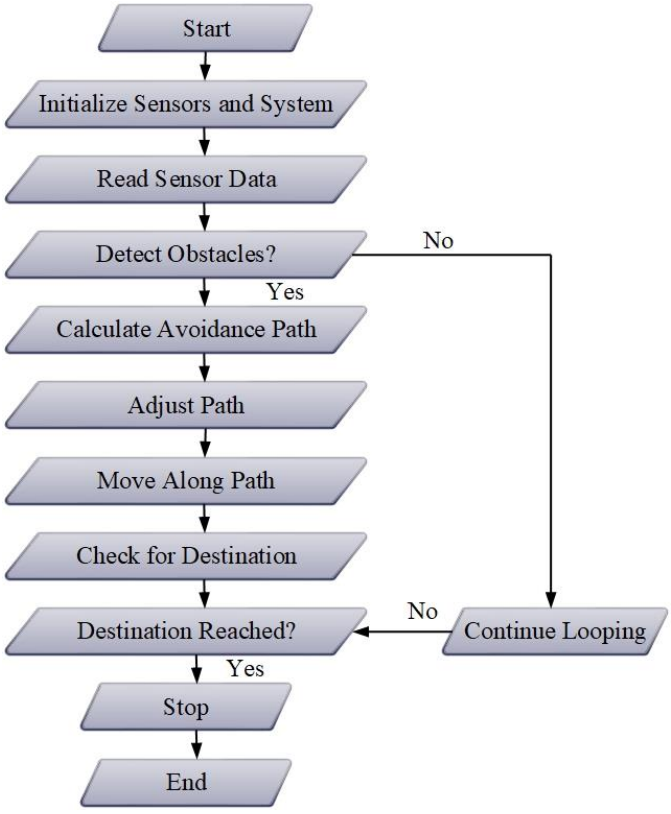

Figure 8: Flowchart of avoidance algorithm

The robot starts its navigation duty. Sensors such as LiDAR and cameras collect information about the surroundings. The program analyzes the sensor data to identify whether there is an obstacle on the robot's route. If yes, a challenge is spotted, the algorithm will prepare an avoidance move. If no obstacles are discovered, the robot can proceed directly to its destination. The algorithm generates a new path which avoids the obstacle based on its position

and the capabilities of the robot. This could include halting, turning, or taking a detour. The robot carries out the anticipated avoidance move. This could include controlling motors or steering devices. If no obstacles are identified after an avoidance maneuver or initially, the robot travels directly to its destination.

4.4 Recharging Module

This method creates a symbiotic connection among a robot's task schedule with an intelligent recharging approach, which helps to optimise the robot's functioning. The task scheduler serves as the robot's brain, providing its goals including the order in which they should be completed. It generates a thorough roadmap for the robot's workday based on criteria such as task location, length, and expected energy consumption. However, the intelligent recharging approach serves as a smart assistant, continually checking the robot's battery capacity and power consumption to make sure it has enough energy to complete its job. These two technologies can ensure that the robot runs at optimal efficiency, executing tasks on schedule and without run out of power.

According to the task schedule and expected energy usage, the algorithm estimates when the robot will require a recharge to finish its tasks. If the next task is close to a charging station, the robot may recharge before beginning the task to avoid having to return later with a low battery. If the system connects to the electric grid, it may prioritize charge during off-peak hours while electricity is less expensive. The technology can modify charging patterns to reduce battery stress and extend its life.

5. Result and Discussion

The proposed system will be evaluated in a simulated environment with varying degrees of clutter and dynamic obstacles. The modelling has been done in Python. Metrics for evaluation will include energy consumption, success rate etc. Table 2 shows evaluation metrics of proposed study.

Table 2: Evaluation Metrics

Metrics	**Formula**
Energy Consumption	$E = \sum_{j=1}^{m} Q_j \cdot r_j$

Success rate	$success_{rate} = \frac{(No.of\ successful\ trials)}{Tot.No\ of\ Trials} \times 100$
Path length	$L = \sum_{j=1}^{n} g_j$

5.1 Performance Analysis

The performance of the proposed study is compared with other existing techniques to prove the efficiency of the proposed work. Energy consumption, success rate, as well as path length, are the major metrics examined, and they provide data on the system's efficiency, dependability, and efficacy. Figure 9 shows the 2-robot path planning.

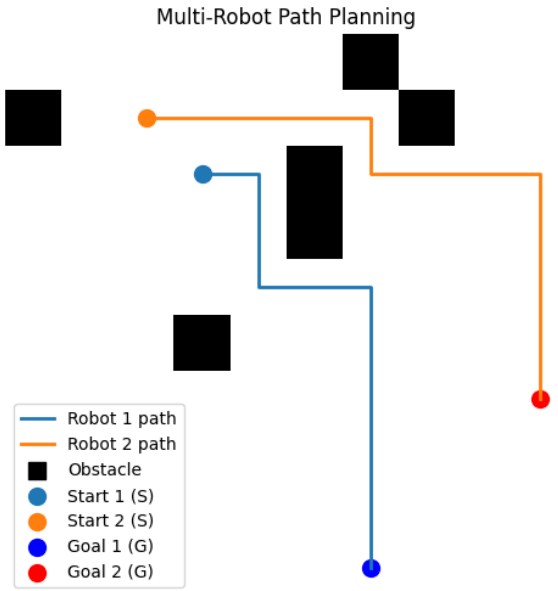

Figure 9: Path Planning

Robot 1 is denoted by a blue line that begins at "Start 1 (S)" and ends at "Goal 1 (G)". Robot 2 is represented by an orange line that begins with an orange dot marked "Start 2 (S)" and ends at a red dot labeled "Goal 2 (G)". The black squares indicate impediments within the environment that robots must negotiate. The paths of both robots are designed to prevent collisions involving the black square obstacles while achieving their respective goals. The pathways are designed to avoid collisions among the robots, exhibiting good multi-robot path planning. Both robots take extremely direct pathways to their destinations while avoiding obstacles suggesting good path planning. Figure 10 shows the 3-robot path planning.

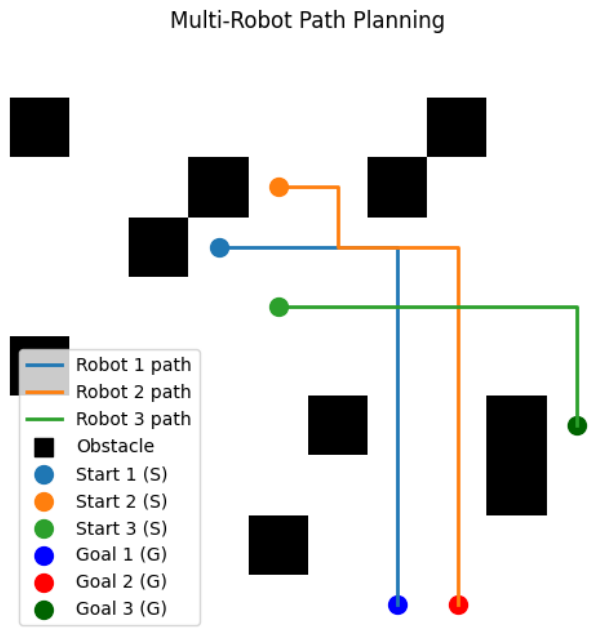

Figure 10: Multi-robot path planning

The robot's starting position is the point where the robot starts its excitement. The route that every robot will travel from its starting point (S) to its destination (G). The program determines all possible courses for each robot and chooses one that prevents collisions between other robots as well as obstacles within the environment. An obstacle in the surroundings that robots have to avoid while getting to their destination. Obstacles can be either immovable items like walls or furniture, or mobile objects such as people or other robots. Figure 11 shows the energy consumption of proposed and existing approaches.

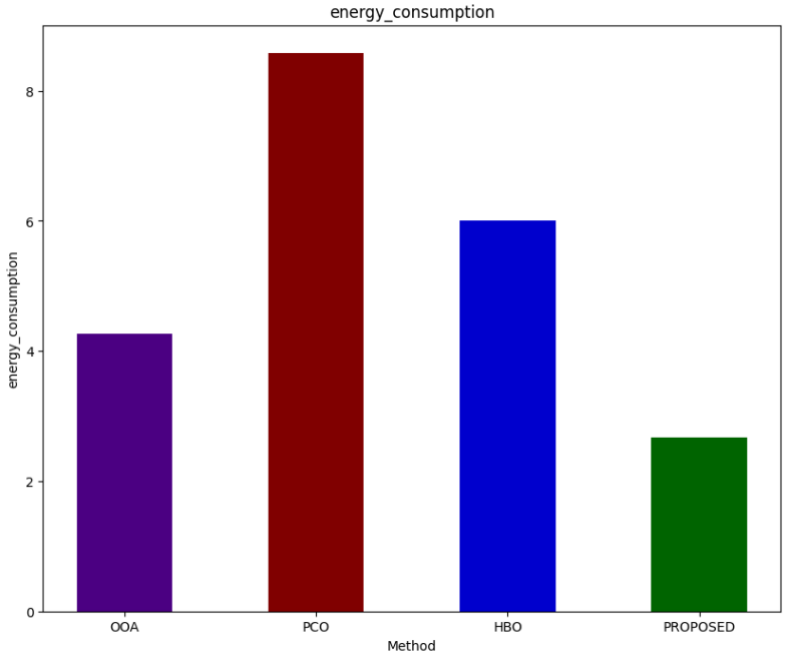

Figure 11: Energy Consumption

Energy consumption provides a significant factor in mobile robot growth and operation. It directly impacts a robot's mission duration, operational range, as well as overall efficacy. Robots require battery life to function, thus improving their efficiency is critical to ensuring they can accomplish their work without regular recharging. This is especially important in robots deployed in distant or hazardous environments wherever access to power sources may be restricted. Here, the energy consumption of proposed has a low value compared with other existing studies. Figure 12 shows the success rate of navigation.

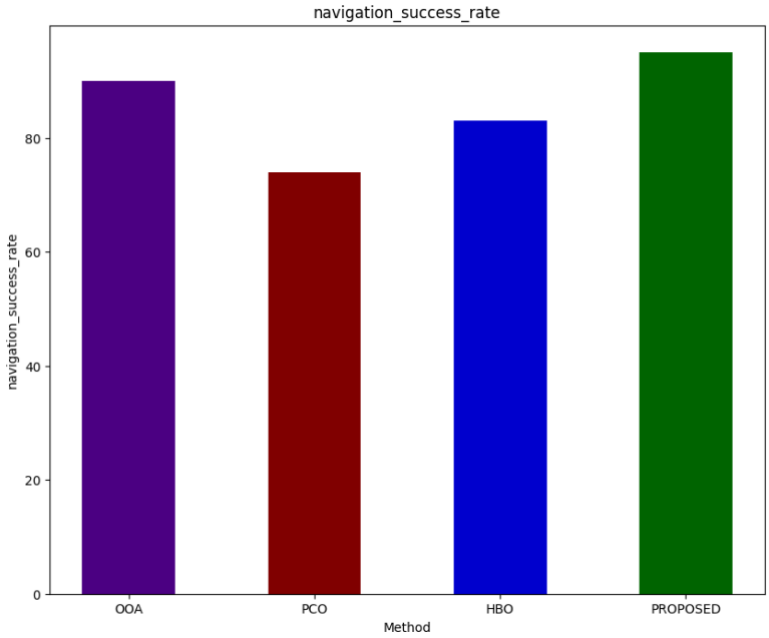

Figure 12: The success rate navigation of proposed and existing techniques

Navigation systems depend on complex relationships among real-world objects (maps, GPS, sensors) as well as software components. OOA could result in an extremely detailed model that is difficult to preserve and adjust. Navigation involves real-time processing and decision-making. OOA does not specifically address these issues, which may result in inefficiencies or delays. Because of such limitations, the proposed achieve a success rate in navigation. Figure 13 indicates the path length of the proposed and existing approaches.

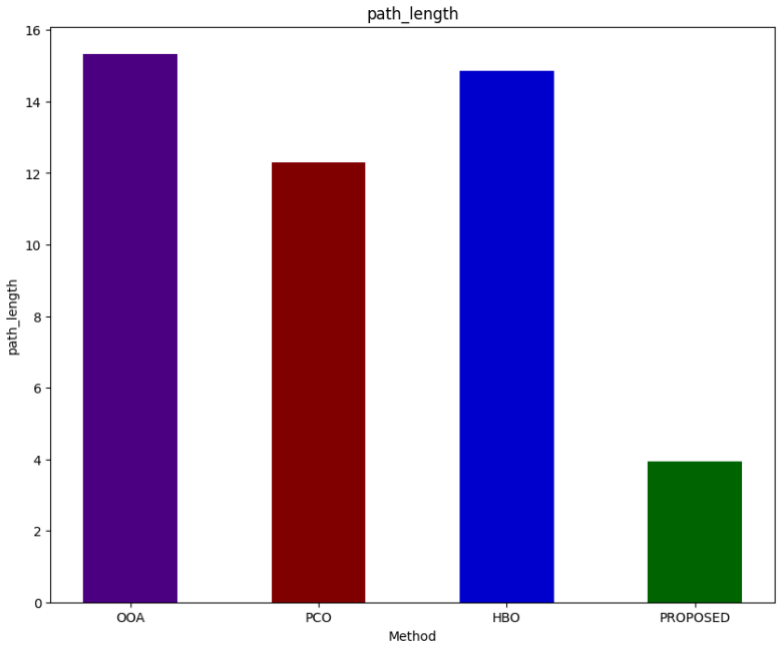

Figure 13: Path length of existing and proposed study

The average path length is an essential metric in network analysis that calculates the average amount of steps taken along the shortest distances for all feasible network node pairs. This measure shows how well data or assets may be carried over a network. The average path length changes by method, with the proposed having the lowest distance than HBO, OOA, and PCO. The proposed strategy looks to be the most effective, with the robot traveling only around 6 feet on average. HBO ranks second at approximately 8 feet, while OOA has a path length of 10 feet. PCO appears to be the least effective approach, averaging approximately 14 feet. Figure 14 shows the overall score of the existing and proposed technique.

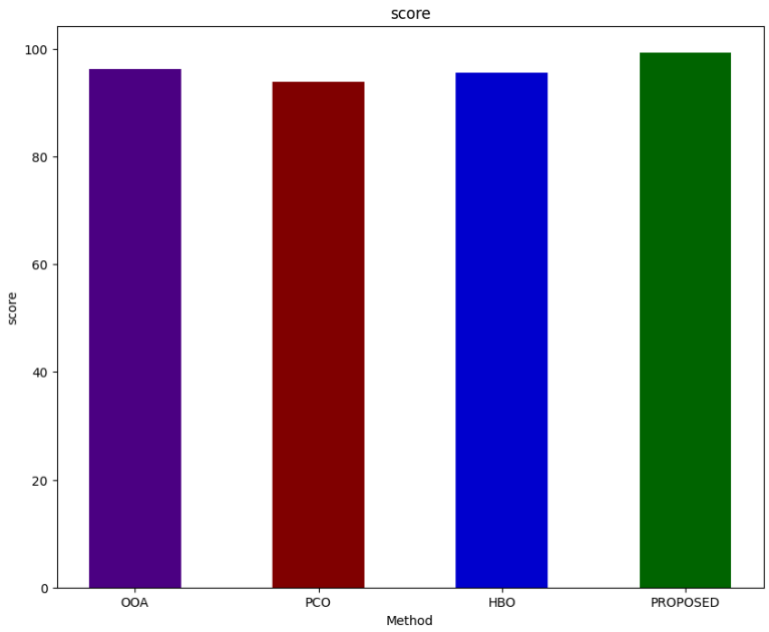

Figure 14: Overall score

OOA excels in replicating object behavior and interaction. However, path planning considers the complete environment and optimizing the total path, which OOA may not fully capture. PCO could be a method that promotes reducing individual path segments over total path length. Some advanced path-planning techniques are computationally expensive, especially in complicated situations. Here, the proposed method attains better results as compared with other existing approaches. Table 3 shows the comparison analysis of the suggested approach.

Table 3: Overall Comparison Analysis

Method	**Path Length**	**Energy Consumption**	**Navigation Success Rate**	**Score**
OOA	15.327	4.26	90	96.29
PCO	12.289	8.58	74	93.83
HBO	14.86	6	83	95.61
Proposed	3.942	2.67	95	99.26

6. Conclusion and Future Scope

Mobile robots are widely used and require good navigation, particularly for path exploration. Aiming at navigational problems, this research aims for developing a new approach for the navigation of mobile robots which incorporates state-of-the-art optimization and machine learning methodologies. The Multi-Objective Pumafish Optimization Algorithm (MOPFOA) is used to schedule tasks and prioritized double Q-network (PDQN) is used for collision free path planning for single robot system and Multi-Agent Deep Deterministic Policy Gradient (MADDPG) for multiple robot system. ViTs improve the perception of obstacles, and a dual method using transformer-based detection and deep reinforcement learning improves path corrections. Lastly, an independent battery charging system keeps the operation going on and performance achieved for this proposed work in path length is 3.94, energy consumption is 2.67, navigation success rate is 95 and overall score is 99.26. In the future, researchers aim to investigate the efficacy of novel RL algorithms in increasing complex situations.

References

1. Ajeil, Fatin Hassan, Ibraheem Kasim Ibraheem, Ahmad Taher Azar, and Amjad J. Humaidi. "Grid-based mobile robot path planning using aging-based ant colony optimization algorithm in static and dynamic environments." *Sensors* 20, no. 7 (2020): 1880.

2. Xu, Yuan, Tongqian Liu, Bin Sun, Yong Zhang, Siamak Khatibi, and Mingxu Sun. "Indoor Vision/INS Integrated Mobile Robot Navigation Using Multimodel-Based Multifrequency Kalman Filter." *Mathematical Problems in Engineering* 2021, no. 1 (2021): 6694084.

3. Gharajeh, Mohammad Samadi, and Hossein B. Jond. "Hybrid global positioning system-adaptive neuro-fuzzy inference system based autonomous mobile robot navigation." *Robotics and Autonomous Systems* 134 (2020): 103669.

4. Teso-Fz-Betoño, Daniel, Ekaitz Zulueta, Ander Sánchez-Chica, Unai Fernandez-Gamiz, and Aitor Saenz-Aguirre. "Semantic segmentation to develop an indoor navigation system for an autonomous mobile robot." *Mathematics* 8, no. 5 (2020): 855.

5. Wu, Haibing, Bo Tao, Zeyu Gong, Zhouping Yin, and Han Ding. "A standalone RFID-based mobile robot navigation method using single passive tag." *IEEE Transactions on Automation Science and Engineering* 18, no. 4 (2020): 1529-1537.

6. Hewawasam, H. S., M. Yousef Ibrahim, and Gayan Kahandawa Appuhamillage. "Past, present and future of path-planning algorithms for mobile robot navigation in dynamic environments." *IEEE Open Journal of the Industrial Electronics Society* 3 (2022): 353-365.

7. Tripathy, Hrudaya Kumar, Sushruta Mishra, Hiren Kumar Thakkar, and Deepak Rai. "CARE: A collision-aware mobile robot navigation in grid environment using improved breadth first search." *Computers & Electrical Engineering* 94 (2021): 107327.

8. Ran, T., L. Yuan, and J. B. Zhang. "Scene perception based visual navigation of mobile robot in indoor environment." *ISA transactions* 109 (2021): 389-400.

9. Quan, Hao, Yansheng Li, and Yi Zhang. "A novel mobile robot navigation method based on deep reinforcement learning." *International Journal of Advanced Robotic Systems* 17, no. 3 (2020): 1729881420921672.

10. Surmann, Hartmut, Christian Jestel, Robin Marchel, Franziska Musberg, Houssem Elhadj, and Mahbube Ardani. "Deep reinforcement learning for real autonomous mobile robot navigation in indoor environments." *arXiv preprint arXiv:2005.13857* (2020).

11. Sotnik, Svitlana, Syed Khalid Mustafa, M. Ayaz Ahmad, Vyacheslav Lyashenko, and Oleksandr Zeleniy. "Some features of route planning as the basis in a mobile robot." (2020).

12. Ab Wahab, Mohd Nadhir, Ching May Lee, Muhammad Firdaus Akbar, and Fadratul Hafinaz Hassan. "Path planning for mobile robot navigation in unknown indoor environments using hybrid PSOFS algorithm." *IEEE Access* 8 (2020): 161805-161815.

13. Wang, Xixun, Yoshiki Mizukami, Makoto Tada, and Fumitoshi Matsuno. "Navigation of a mobile robot in a dynamic environment using a point cloud map." *Artificial Life and Robotics* 26 (2021): 10-20.

14. Nguyen, Toan Van, Minh Hoang Do, and Jaewon Jo. "MoDeT: a low-cost obstacle tracker for self-driving mobile robot navigation using 2D-laser scan." *Industrial Robot: the international journal of robotics research and application* 49, no. 6 (2022): 1032-1041.

15. Gia Luan, Phan, and Nguyen Truong Thinh. "Real-time hybrid navigation system-based path planning and obstacle avoidance for mobile robots." *Applied sciences* 10, no. 10 (2020): 3355.

16. Iakovlev, Roman, and Anton Saveliev. "Approach to implementation of local navigation of mobile robotic systems in agriculture with the aid of radio modules." *Telfor Journal* 12, no. 2 (2020): 92-97.

17. Farag, Karoline Kamil A., Hussein Hamdy Shehata, and Hesham M. El-Batsh. "Mobile robot obstacle avoidance based on neural network with a standardization technique." *Journal of Robotics* 2021, no. 1 (2021): 1129872.

18. de Oliveira Júnior, Alexandre, Luis Piardi, and Paulo Leitao. "Implementation of a navigation system for a mobile robot in a dynamic environment using AR tags to increase localization accuracy." In *1st Symposium of Applied Science for*, p. 39. 2021.

19. Ajeil, Fatin Hassan, Ibraheem Kasim Ibraheem, Ahmad Taher Azar, and Amjad J. Humaidi. "Autonomous navigation and obstacle avoidance of an omnidirectional mobile robot using swarm optimization and sensors deployment." *International Journal of Advanced Robotic Systems* 17, no. 3 (2020): 1729881420929498.

20. Wang, Dongshu, Yuhang Hu, and Tianlei Ma. "Mobile robot navigation with the combination of supervised learning in cerebellum and reward-based learning in basal ganglia." *Cognitive Systems Research* 59 (2020): 1-14.

21. Surmann, Hartmut, Christian Jestel, Robin Marchel, Franziska Musberg, Houssem Elhadj, and Mahbube Ardani. "Deep reinforcement learning for real autonomous mobile robot navigation in indoor environments." *arXiv preprint arXiv:2005.13857* (2020).

22. Lee, Min-Fan Ricky, and Sharfiden Hassen Yusuf. "Mobile robot navigation using deep reinforcement learning." *Processes* 10, no. 12 (2022): 2748.

23. Li, Jing, Jialin Yin, and Lin Deng. "A robot vision navigation method using deep learning in edge computing environment." *EURASIP Journal on Advances in Signal Processing* 2021, no. 1 (2021): 22.

24. Liu, Bo, Xuesu Xiao, and Peter Stone. "A lifelong learning approach to mobile robot navigation." *IEEE Robotics and Automation Letters* 6, no. 2 (2021): 1090-1096.

25. Sadeghi Esfahlani, Shabnam, Alireza Sanaei, Mohammad Ghorabian, and Hassan Shirvani. "The deep convolutional neural network role in the autonomous navigation of mobile robots (SROBO)." *Remote Sensing* 14, no. 14 (2022): 3324.