Received: 29 June 2024 | *Revised:* 28 July 2024 | *Accepted:* 28 August 2024
DOI: 10.37917/ijeee.22.1.34 *Vol. 22 | Issue 1 | June 2026*

Iraqi Journal for Electrical and Electronic Engineering
Original Article

Open Access
IJEEE University of Basrah College of Engineering

A Hybrid Approach Using Vector Field Histogram and Deep Reinforcement Learning for Dynamic Path Planning

Maysam Hameed Qasim*, Salah Al-Darraji
Department of Computer Science, College of Computer Science & Information Technology, University of Basrah, Basrah, Iraq

Correspondance
*Maysam Hameed Qasim
Department of Computer Science, College of Computer Science & Information Technology,
University of Basrah, Basrah, Iraq
Email: pgs.maysam.qasm@uobasrah.edu.iq

Abstract
Autonomous mobile robots (AMRs) are becoming increasingly important in different domains such as healthcare, warehouse automation and household duties, but still encounter problems when it comes to moving around unfamiliar and dynamic environments. This study proposes an advanced robotic navigation system which combines the Soft Actor-Critic (SAC) approach and Vector Field Histogram (VFH) for path planning and avoidance obstacles in completely unknown environments. This system leverages the strengths of deep reinforcement learning and real-time obstacle detection to achieve robust and efficient navigation in certain scenarios. The SAC strategy optimizes robot navigation using policy networks and Q-networks, while the VFH method addresses obstacle avoidance by sensor data processing and dynamically adjusting the robot's angular velocity to avoid collision. For testing and implementing this system, Gazebo simulation and Robot Operating System (ROS) are used. Experimental results demonstrated that the proposed method outperformed the standard technique and achieved a high success rate in path planning and obstacles avoidance.

Keywords
Mobile Robot, Navigation, Obstacle Avoidance Techniques, Soft Actor Critic.

I. Introduction

Scientific and technical advances have led to the use of mobile robots in a range of everyday life disciplines including distribution centers, warehouses, hospitals and automated cleaning. Path planning has been given more consideration and development because it is essential to allow these robots to navigate autonomously through a specific environment from the starting point to goal point without collision with any obstacle, while taking into consideration variables like time, distance, and smoothness. The primary objective in path planning in complex situations is to find a safe, obstacle-free route that minimizes the distance covered [1].

Path planning comes in two forms: global and local. In order to do global planning, the autonomous robot must possess complete environmental knowledge, including start and finish locations as well as the placements of all obstacles in fully known surroundings. On the other hand, local planning is appropriate in situations where environmental information is not entirely known and is appropriate for settings that are either partially known or unknown [2]. While moving obstacles require real-time planning, static obstacles require knowing the start and destination positions [3].

Each of the different techniques used to set up path planning techniques has its own way of working and is best suited to specific tasks. While some path planning techniques are optimized for static environments, others are made to function well in dynamic ones. Path-planning techniques can be generally categorized into two sets: heuristic approaches and classical approaches [4]. The artificial potential field [5],vector field histogram [6], mathematical programming [7],and

the roadmap approach [8] are examples of often mentioned classical methodologies. Heuristic methods include methods like particle swarm optimization [9], artificial intelligence [10], bacterial foraging optimization [11], genetic algorithm [12], and more. These techniques are widely used in path-planning tasks and can be adjusted to various surrounding conditions [13].

Some of the path planning approaches mentioned above suffer from inefficiency when dealing with unknown and difficult environments due to heavy dependence on environmental map information. Increasing attention is being paid to creating methods that can learn on their own and does not require maps of the environment [14].

Artificial intelligence (AI) has advanced quickly in the last few years, especially in the areas of deep learning (DL) and reinforcement learning (RL). One notable feature of reinforcement learning is that it does not rely on maps, and it learns path planning tactics by trial and error and using sensors that communicate with the environment directly [15].

Deep reinforcement learning (DRL) is the combination of two different paradigms in AI: DL and RL. This unification enables AI systems to handle complicated tasks better since they can take advantage of their individual strong points more effectively [16]. DRL is a heuristic method which is part of machine learning. Its applications can be found in many sectors, including image recognition, gaming, and task decision-making. The navigation systems in robots including tasks like planning, perception, and localization, rely heavily on DRL. Unlike traditional path planning technique that depend on models or preset rules, a DRL agent learns by interacting with its environment to find the efficiency path [17].

Soft actor-critic (SAC) is a model-free variant of the DRL technique, known for its robustness and rapid convergence compared to other approaches. This makes SAC ideal for training agents in new and challenging scenarios [18].

Vector field histogram (VFH), created by Borenstein and Koren, is a classical technique that uses range sensor inputs to compute obstacle-free steering directions for a robot. This allows the robot to successfully navigate its surroundings and identify obstacles [19].

In this work, a heuristic method was used based on SAC technique and integrated with the classical technique based on VFH technique for path planning in completely unknown environments. The key contributions of the current study can be summarized below:

1. This study presents an innovative method that merges the soft actor-critic (SAC) technique with vector field histogram (VFH) method. The objective of this hybrid approach is to allow the robot to navigate around both static and moving obstacles while effectively planning paths in complex environments. By incorporating SAC's proficiency in continuous action spaces with VFH's real-time obstacle detection and avoidance capabilities, this solution significantly enhances the robot's adaptability and responsiveness.

2. SAC technique is good at teaching robot to interact with it environments, using soft updates, past experiences, and real-time obstacle information for safe and goal-oriented actions.

3. VFH technique employs sensing data to identify safe routes and obstructions in order to enhance the SAC algorithm's capability to safely and efficiently pass dynamic environments.

4. Furthermore, the combination of behavioral choice and motion planning layers in the architecture of the proposed method enhances both path accuracy and efficiency. It enables real-time adaptation to complex scenarios by generating continuous control commands based on inputs from environmental perception.

5. The performance of the suggested method was evaluated in two different scenarios and compared with the standard SAC technique under identical conditions. The simulation outcomes clearly show that the proposed approach is effective in navigating around both static and moving obstacles while avoiding obstacles.

The rest of the paper is structured as: The relevant work, containing a variety of path planning and collision avoidance methods, is covered in Section II. The proposed approach's path planning pipeline is presented in Section III, which also covers the VFH technique for obstacle avoidance and the SAC strategy for path planning. The methodology is presented in Section IV, along with the soft actor-critic network structure design, reward function, and SAC technique improvements. The experimental simulation setup used to evaluate the suggested approach is described in Section V. The experiment outcomes are shown in Section VI, along with a thorough efficiency and comparison analysis. Finally, Section VII offers an extensive summary of the research 's contents.

II. Related Work

The two main sub-tasks of mobile robot's path planning in interior surroundings are detecting and avoiding obstacles and path planning. The difficulty of local path planning involves avoiding both static and moving obstacles, such as walls and furniture, and moving objects like persons and equipment. Approaches intended here are to apply strategies to navigate safely in crowded indoor environments. They use sensors to detect obstacles and plan a safe path free of collisions [20].

Over the years, researchers in the subject of navigation for mobile robots have investigated a variety of strategies, spanning from classical artificial intelligence techniques to modern bio-inspired techniques and DRL methods. The current section will focus on pertinent research that specifically addresses dynamic path planning approach obstacle identification, and avoidance using DRL techniques. Research has shown the capability of DRL techniques in effectively determining optimal paths for robots in dynamic environments. However, DRL has limitations, particularly because it lacks all-encompassing global data. If robots on wheels do not have a full understanding of the environment around them, they may face difficulties in navigating complex situations. To get beyond the constraints of DRL methods, researchers have started integrating these approaches with other strategies to prevent collisions. This combination technique leads to improvement the navigational capabilities of robots in complex surroundings.

Runqi et al. [21] suggested a two-layer, hierarchical deep learning-based control framework for mobile robots to plan and guide their exploration. They used a recurrent deep neural network (RDNN) technique for motion planning, a collision-free control technique based on deep reinforcement learning, and a noisy prioritized experience replay technique for increased exploration rate. Experimental results showed improved performance and shorter training time.

Runnan et al. [22] proposed a method using deep Q-networks and the concept of reinforcement learning. They utilized two reward levels to handle collisions and weights to find a compromise between goal approach and obstacle avoidance. The findings showed that robots can maneuver through obstacles with ease.

Kumaar and Sreeja [23] suggested a topological environment representation and deep Q-Learning method for learning initial paths. A new $\beta\beta$-decay transfer learning algorithm balances experience, exploration, and exploitation, achieving over 98% accuracy in various service scenarios using reinforcement learning.

In their study, Zheng et al. [24] presented a dynamic action selection technique for the purpose of enhancing manipulator trajectory planning in dynamic situations. They used a changeable guide item, a time-energy function, and an artificial potential field approach. This integration enhances trajectory planning efficiency and stability, increasing convergence rates by 3-5 times on DRL approaches such as DDPG, SAC, and TD3.

Yang et al. [25] suggested an enhanced technique for mobile robots' navigation planning, based on the SAC method, to improve robot performance in complex environments with static and dynamic obstacles. The algorithm uses state dynamic normalization and priority replay buffer techniques to enable quick obstacle avoidance and target reach. Experimental results show improved performance.

Hui et al. [26] presented a Long Short-Term Memory (LSTM) into the Deep Deterministic Policy Gradient (DDPG) network that integrates past and present robot states to decide what to do. In addition, a reward function has been modified for faster movement and a Batch Norm layer is introduced. The application of normalization techniques increases learning effectiveness. To prepare for the following robot action, varied noise is added. Tests reveal that the algorithm increases efficiency and success rate, improves generalization, and speeds up convergence.

Kobayashi et al. [27] proposed a dynamic weight coefficients system for the Dynamic Window Approach (DWA) that uses Q-learning to optimize path and weight coefficients to overcome the drawbacks of fixed weight coefficients.

Tan et al. [28] proposed PL-TD3, a novel approach for mobile navigation robots that enhances dynamic obstacle detection and convergence speed. The usage of the LSTM neural network with the PER enhances the efficacy of the technique. The approach is tested in both static and dynamic situations; the findings indicate that, in all cases, PL-TD3 performs better than TD3 in terms of execution time and path length. The problems of sluggish convergence and misperceived obstacles are addressed by this technique.

In summary, merging DRL with other technology shows high potential for improving the navigation capabilities of robots for Path planning and collision avoidance.

III. Path Planning and Collision Avoidance Approaches

A. Soft Actor-Critic Method for Path Planning

A stochastic actor-critic method, also known as soft actor-critic (SAC), is used to learn continuous action space policies using approximation functions. It alternates among policy improvement and evaluation that allows finding the value function of the policy at hand based on maximum entropy. The SAC technique approximates a state-value network, a Q-value critic network, and an actor policy network with three different networks. These networks predict actions and create a temporal-difference error for each time step. The purpose of SAC is to maximize both rewards and policy entropy, which refers to the degree of uncertainty associated with a variable. A replay memory should be used to train and evaluate a policy function that saves experiences from agent during training sessions. This helps in avoiding accumulation of difference when approximating more competent Q-function about the problem that is being solved. For learning, randomly sample experiences from previous episodes.

SAC is combined with maximum entropy DRL. The opti-

mization objective can be represented by the following equation [29]:

$$\pi^*_{\max} = \arg\max_{\pi} \sum_{t=0}^{T} \mathbb{E}_{(s_t,a_t)\sim p_\pi} \left[r(s_t,a_t) + \alpha \left(-\log \pi(\cdot|s_t) \right) \right] \tag{1}$$

where $H(\pi(\cdot \mid s_t)) = -\log \pi(\cdot \mid s_t)$ is the policy entropy, or the degree of unpredictability of the policy.α is the coefficient of temperature, which stands for whether maximizing rewards or maximizing policy entropy is the more likely optimization objective for SAC [30].
$\sum_{t=0}^{T} \mathbb{E}_{(s_t,a_t)\sim P_\pi} [r(s_t,a_t)]$ represents the primary reward objective of the goal function. The goal is to maximize this reward function.

B. *Vector Field Histogram Approach for Obstacle Avoidance*

The vector field histogram (VFH) technique is a computationally efficient technique constructed for dynamic obstacle avoidance in mobile robots [31].This technology processes sensor data received from the environment, transforming the surrounding Cartesian coordinate map into a polar density map. The method divides the robot's range of vision into angular bins, each of which stands for a different area of the environment.

In this system, the VFH technique is initialized with a predefined number of bins and angular range. By dividing the entire range by the number of bins, one can obtain the angular resolution and a regular distribution of sectors. The obstacle density for each bin is then ascertained by processing data scan ranges of the sensor. The method calculates the obstacle density by summing all the sensor data within each bin and normalizing these values based on the maximum range observed. The density is set to zero in a bin if no obstacles are found there. It ensures that the approach may dynamically adapt based on the presence or absence of obstacles in the surrounding area. After computing the obstacle densities, the technique points to the sector with the low density, indicating the direction with the least obstacles. The corresponding angle of this sector is then chosen as the robot's direction, guiding it through the path of least obstacles. This implementation of the VFH method thus enhances real-time obstacle avoidance by dynamically adaptation to the scenarios and efficiently computing the optimal heading steering based on obstacle density.

IV. METHODOLOGY

The study aims to improve path planning for mobile robot in static and dynamic environments by combining the SAC with VFH methods. The proposed method addresses challenges faced by DRL technique in avoiding obstacles.

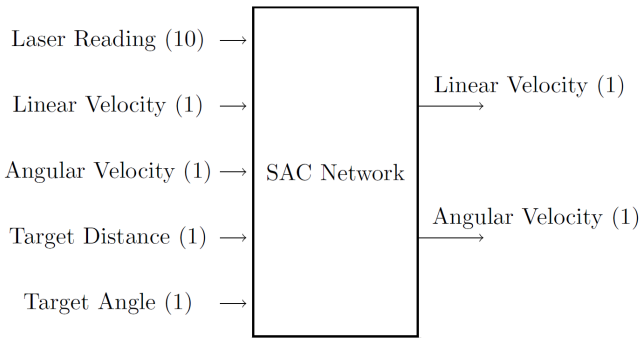

Fig. 1. Inputs and Outputs Network

A. *Soft Actor Critic Network Design*

The inputs to the neural network are (14), (10) which represent the laser detector readings, while the other (2) are the past angular velocities and linear acquired by the agent, as shown in Fig. 1. The robot's relative position, including the angle and distance to the target, is indicated via the two remaining inputs. These inputs represent the state space that the robot receives from its environment. The state in DRL is an illustration of the present environment in which the robot is located. It can be detected by the robot, and it has all the appropriate data about the environment that the robot needs to identify and make a decision about.The angles, at which the laser readings, are obtained with respect to the mobile robot range ($-\frac{\pi}{2}$ to $\frac{\pi}{2}$) degrees.

The network outputs sent to the agent are linear velocity and angular velocity, which is the action space that represents the robot's possible movements. Both the linear velocities and angular velocities that are forwarding to the robot are produced by the network.

Fig. 2 shows the network configuration of the SAC. The mobile robot's current location within the environment is determined by the actor-network inputs. Before the input reaches the output layer, it navigates three fully linked neural network layers, each with (512) units. The network's structure, as detailed in [32] served as the basis for determining the number of layers and units. The velocity (linear and angular) that is forwarding to the agent is generated by the output layer. However, "tanh" is used as the activation function, and the actions are limited to the interval $(-1, 1)$.

The critic-network gives the Q-value for the agent's current state and action. The present state value is predicted in the value network. For processing the state inputs, the only two networks used three fully-connected neural network layers. Using a linear activation function, the Q-value and the value

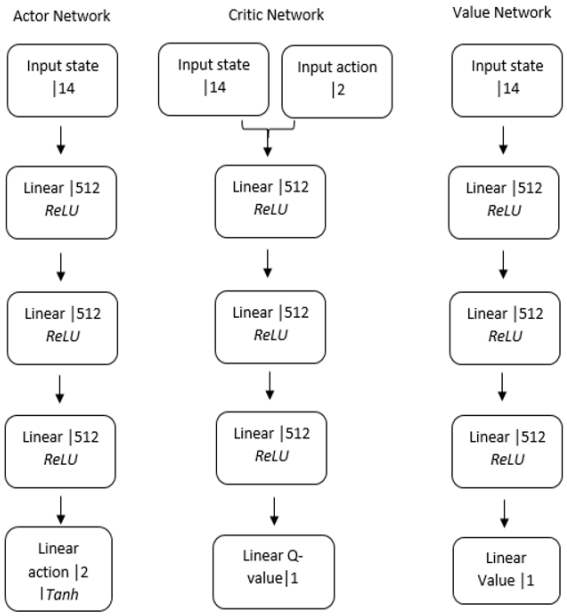

Fig. 2. Structure of SAC Network

of the present situation are determined [33]:

$$y = ka + c \tag{2}$$

where y is the expected value given the current state, k represents the weights for trained, a is the final layer's input, and c is the last layer's bias.

The neural network can learn the best weights and biases using this process to predict accurately Q-values or state values for any given input.

B. Reward Function

Once the surroundings are established, it is possible to simulate a mobile robot to carry out a path planning task. It is main to define the system of rewards and penalties for the DRL network. The penalties and rewards assigned to the (robot) agent are numerical values derived from a function model. This model is constructed using empirical knowledge obtained during the problem-solving process. Therefore, the network will do a backpropagation and feedforward process to acquire knowledge of the hyperparameters.

In their work [34] the authors presented three criteria for the reward function. One of the conditions in this work was removed. This resulted in an extended duration of training. However, the new reward function is superior to the prior one. The reward function employed in this study was as follows:

$$R(state, action) = \begin{cases} R_{goal} \text{ if } d_t < c_d \\ R_{collision} \text{ if } minimum_x < c_o \end{cases} \tag{3}$$

From threshold 10čm, it is verified that the robot has reached its target, and it receives a positive reward (R_{goal}). However, the robot receives a negative reward $(R_{collision})$ if it collides with an obstacle. The variable "c_o" represents the distance is 15čm between the agent and an obstacle. Both circumstances are adequate to terminate the training iteration. We have chosen to exclusively provide rewards when the robot encounters collisions and reaches the goals. This decision was made based on our observation that alternative forms of rewards created a deceptive interpretation of the map for the agent, making it difficult for us to deliver rewards that align with our objectives. By maximizing these benefits, the agent would also accomplish our aims.

C. Enhancement of Soft Actor Critic Technique

In this paper, the focus was on improving mobile robot navigation capabilities in environments with dynamic obstacles by integrating the SAC technique with the VFH. The primary motivation for this approach is that while DRL methods like SAC have shown significant potential in various fields, they often face challenges when it comes to effectively avoiding dynamic obstacles in complex environments.

The SAC technique, known for its efficiency in solving tasks with continuous action spaces, is used to improve a policy that dictates the robot's movement approaches based on a given state input. The SAC method focuses on two main tasks: developing a policy function to choose actions and a value function to predict the outcomes of those actions. The policy is modified to maximize both entropy and expected rewards, which encourages exploration.

Our goal is to create a more reliable and adaptable navigation system by integrating VFH, a well-established method for real-time obstacle detection and avoidance. By combining SAC's robust decision-making capabilities with VFH's precise perception, autonomous navigation in dynamic and unpredictable environments can be significantly enhanced.

The VFH approach is incorporated, which processes data from LIDAR or other sensors to create a histogram of obstacle densities around the robot. VFH divides the sensor data into bins, calculates the obstacle density for each bin, and identifies the bin with the lowest obstacle density to determine a collision-free direction. This direction, representing the nearest safe path, is essential for navigating through crowded areas. The integration procedure is as follows: the state vector is sent into the SAC via the VFH approach, which preprocesses sensory inputs in order to determine a safe direction. This state vector enhances the data that SAC uses to make decisions by containing both the raw sensor data and the heading obtained from the VFH. After SAC selects an action based on this enhanced state, VFH may modify the action to guarantee immediate safety depending on the proximity of the obsta-

cle at that moment. This successfully combines reactive and strategic decision-making. Furthermore, the results of these actions (both as initially decided by SAC and as modified by VFH) along with the rewards generated from the environment, are fed back into SAC's learning process. This feedback loop allows SAC to adapt its policy not only based on long-term navigation goals but also incorporating the efficacy of immediate obstacle avoidance maneuvers suggested by VFH. This integrated approach is designed to improve the robot's ability to navigate efficiently and safely, leveraging the strengths of both SAC's high-level planning and VFH's local obstacle avoidance capabilities.

Through this methodology, this study aims to demonstrate that the synergy between SAC and VFH can significantly enhance the robot's navigation and decision-making capabilities, particularly in environments where dynamic obstacle avoidance is critical.

V. Simulation Experiments

A. Experimental Setup

Experiments were conducted on Intel® Core™ i7-1165G7 @ 2.80GHz × 8, 8GB RAM, running Ubuntu 20.04 LTS. The simulations were done with ROS running as the backbone and Gazebo. Python was preferred for programming while the methods employed PyTorch library to build neural networks. The robotic platform that was chosen is Turtlebot3 model Burger.

Table I summarizes the parameters used in this work. These parameters are chosen according to trial and error.

TABLE I.
Parameters Setting of SAC and VFH

Methods	Parameters	Setting
SAC	Learning Rate (lr)	0.0003
	Batch Size	256
	Discount Factor (α)	0.99
	Replay Buffer Size	150,000
	Target Update Interval	1
	Hidden Layer Size	512
	Exploration Strategy	Gaussian Noise ($\sigma = 0.1$)
	Target Entropy	-2
	Action Range (Linear Velocity)	[0.0, 0.22] m/s
	Action Range (Angular Velocity)	[-2, 2] rad/s
VFH	Number of Bins	36
	Min Angle	$-\pi$
	Max Angle	π
	Obstacle Density Threshold	0.1

B. Simulation Environments

In this section, the performance of the proposed approach was examined in two different environments: a simple environment and a complex environment. The scenario that contains static obstacles represents the simple environment, while the scenario that includes static and moving obstacles represents the complex environment.

For comparison, tests were conducted using the standard SAC algorithm and the improved SAC approach in both environments. The performance was evaluated based on several metrics, including the success rate of obstacle avoidance and the efficiency of the learning methods.

1) Static Environment

The first environment is depicted in Fig. 3, with $5\,m \times 5\,m$ and several static obstacles placed within a rectangular boundary. These obstacles are positioned to create a series of pathways and potential dead ends, challenging the robot to navigate effectively. The obstacle placement creates multiple potential routes, some of which may lead to dead ends, requiring the robot to backtrack and find alternative paths. The primary objective for the robot is to navigate through the maze while avoiding collisions with the static obstacles. This environment tests the robot's ability to make real-time decisions, utilize sensory data effectively, and adapt its path to reach the desired destination, emphasizing strategic decision-making and efficient path finding.

There is a list of 14 goals that are dispersed around the environment and designed to allow for a continuous path these targets do not all appear at the same time in the simulation environment, but rather one active target is randomly selected from the list. Fig. 4 shows positions of the targets in an environment. Each time, one target is randomly selected from the list, and the robot groups these targets sequentially. When a mobile robot collision into the wall or some other object, it is penalized for doing so in the form of negative reward and then that episode ends.

2) Dynamic Environment

The second environment, shown in Fig. 5, has an area of $7\,m \times 7\,m$, and is quite different from the static environment, as it provides complex and challenging training scenarios, and allows the moving obstacles to move in different patterns. It also helps the robot deal with real-world scenarios and enhances the efficiency of the collision avoidance algorithm. This environment was designed to test the navigation and obstacle avoidance capabilities of an agent in a dynamic setting. The maze-like structure consists of several pathways and obstacles, including multiple "U" shaped obstacles that pose

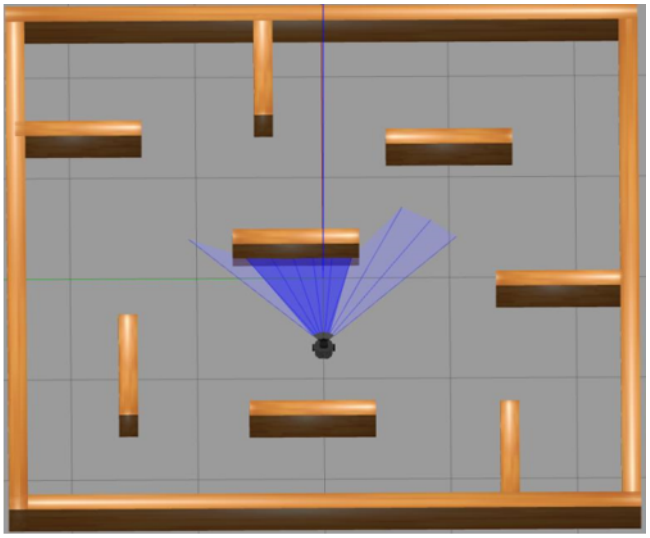

Fig. 3. Static environment.

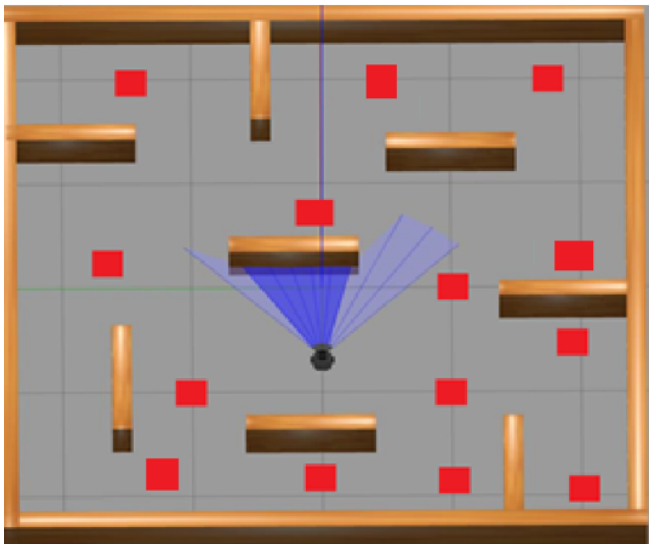

Fig. 4. Distribution of goals.

significant challenges due to their symmetrical nature. These ”U” shaped structures create potential dead ends, requiring the robot to make strategic decisions to navigate effectively. This indicates that the environment is more complicated than before, requiring the intelligent system to develop a more effective collision avoidance plan.

In this environment, there are white spheres that represent moving obstacles, which add an additional layer of complexity. The robot must continuously adjust its path to avoid the dynamic waypoints, which are simulated by these moving obstacles. Fig. 6 shows the trajectory of moving obstacles. The movement of these obstacles requires the robot to not only navigate the static maze but also to adapt to the changing positions of the obstacles in real-time. The dynamic nature of the moving obstacle introduces variability in the environment, forcing the robot to continuously reassess its surroundings and update its trajectory.

There is also a list of 14 goals dispersed around the environment. Each time, one target is randomly selected from the list, and the robot groups these targets sequentially. Fig. 7 shows Positions of targets distributed around environment. Should the mobile robot run into a wall or any moving object, the current episode ends, and a negative reward is given for this action.

VI. Results

A. Training Phase

The information gathered from the awards was used in the training evaluation stage. In this study, we were able to measure the agent's level of environmental learning since the reward is closely related to the agent's performance. The training environments were created in a way that motivates the agent to come up with fresh ideas for achieving his suggested objective in these kinds of situations. Turtlebot3, a mobile robot, was trained in the situations shown in Fig. 3. Then 3,000 episodes were conducted in the first environment using two methodologies: standard SAC and improved SAC. And 5,000 episodes were conducted in the second environment in order to test the learning ability, as shown in Fig. 5. A step of neural network evaluation was formed during the training phase since the system had signals for exploration.

In a simple environment with static obstacles, the robot

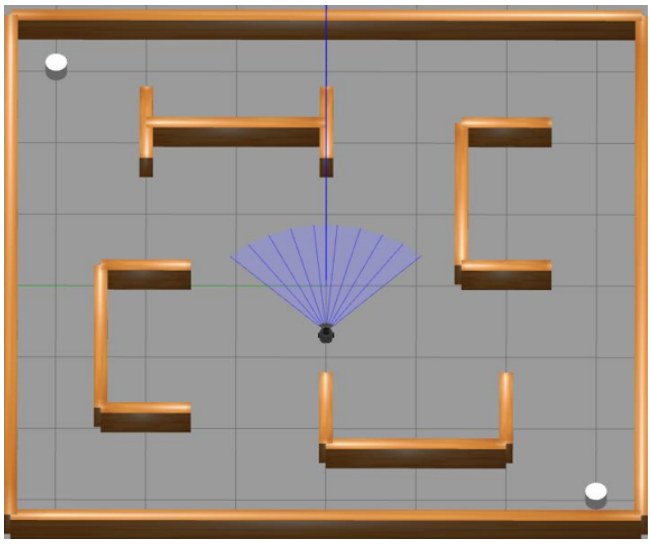

Fig. 5. Dynamic environment.

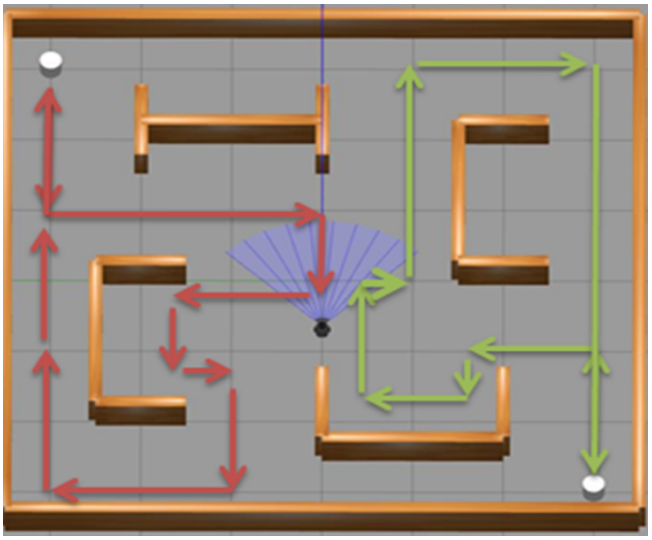

Fig. 6. Dynamic obstacle movements.

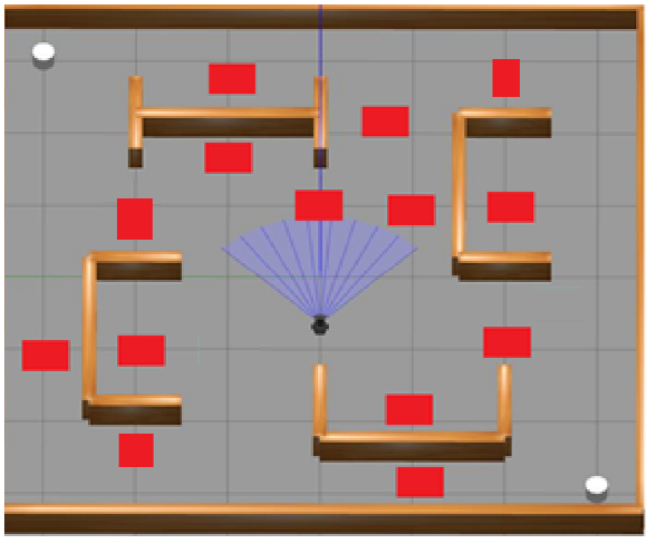

Fig. 7. Distribution of goals.

collected rewards over the course of training episodes when using the standard SAC method. This indicates that it, to some extent, was able to learn and improve its policy. However, the standard SAC algorithm struggled with collisions and had difficulty in effectively planning paths around the static obstacles, leading to frequent collisions and sub-optimal navigation performance. The success rate varies between 60 % and 40 % for every 100 episodes, highlighting the significant challenges the robot faced in avoiding obstacles. Fig. 8 shows the robot's navigation through an environment where it experiences a collision.

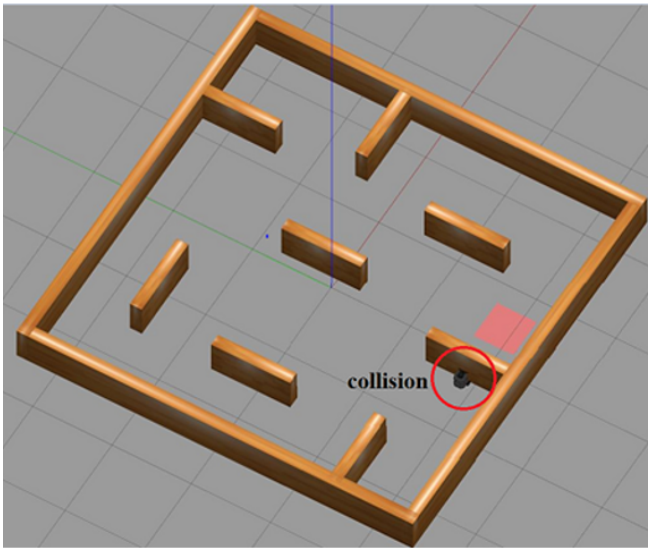

Fig. 8. Collision in first environment.

When integrated with VFH, the SAC method showed significant improvement. The navigation paths became more efficient, and the success rate of obstacle avoidance increased. Fig. 9 illustrates the sequence of images showing the robot's path through the environment to reach its goal. The VFH technique provided precise obstacle detection, enabling the robot to plan more effective routes around the static obstacles, resulting in fewer collisions and improved navigation time. As a result, the success rate reached 100 % by effectively avoiding obstacles. Fig. 10 shows a comparison between the

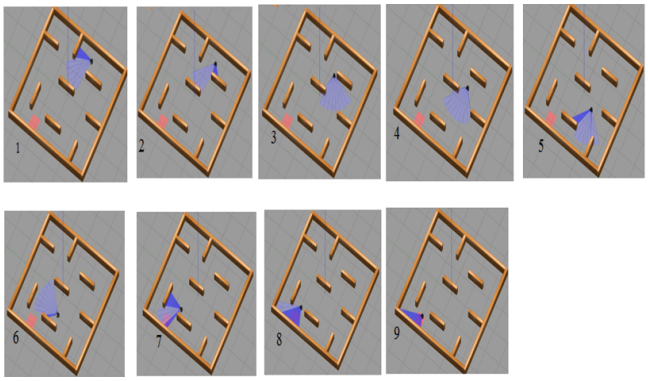

Fig. 9. Robot's trajectory towards the goal.

improved SAC technique and the standard SAC technique, the data collected for the reward function during training. The results indicate that the Improved SAC algorithm outperforms the standard SAC method in terms of rewards accumulated.

In the complex environment with dynamic obstacles, the limitations of the SAC algorithm became more apparent. The presence of moving obstacles led to frequent collisions and

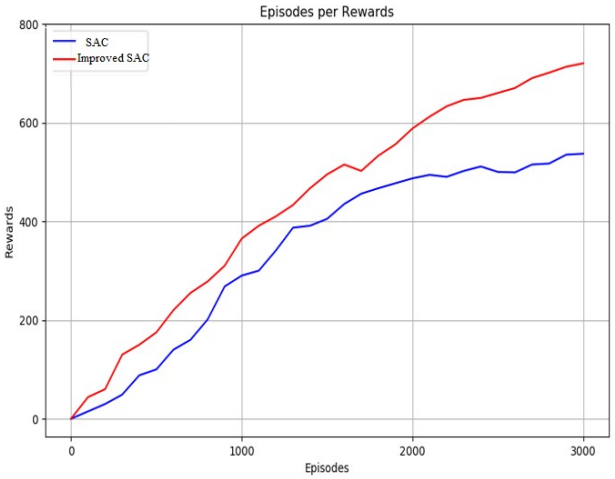

Fig. 10. Average cumulative reward in first environment.

failed attempts, significantly reducing the success rate and efficiency of the navigation. The success rate varies between 20 % and 10 % for every 100 episodes, highlighting the significant challenges the robot faced in avoiding obstacles. Fig. 11 shows the robot's navigation through an environment where it experiences a collision The improved SAC approach excelled in the complex environment. The VFH technique ability to detect and avoid dynamic obstacles in real-time complemented SAC's decision-making process, resulting in a higher success rate and more efficient navigation paths. The robot was able to adapt to the dynamic changes in the environment more effectively, reducing collisions and overall navigation time. As a result, the success rate reached 100 % by effectively avoiding obstacles. Fig. 12 shows a sequence of images illustrating the robot's ability to avoid moving obstacles. Fig. 13 illustrates

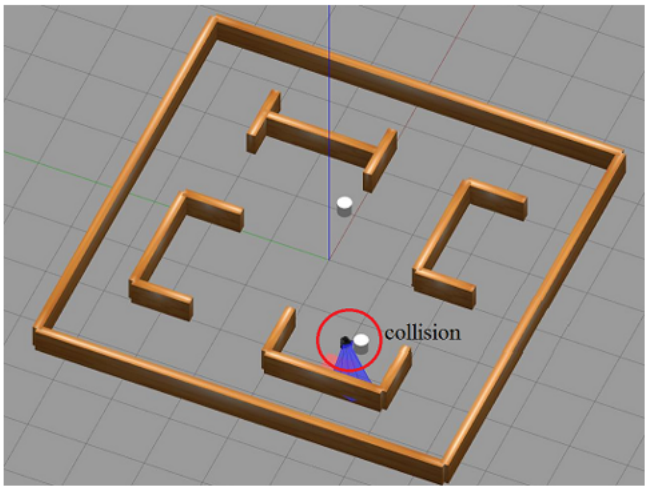

Fig. 11. Collision in second environment.

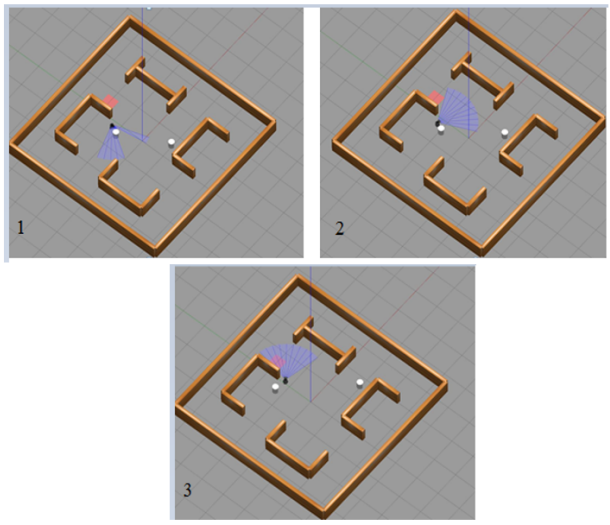

Fig. 12. Dynamic obstacle avoidance.

the sequence of images showing the robot's path through the environment to reach its goal. Fig. 14 illustrates the information collected for the reward function during the training stage, comparing the performance of the standard SAC method with improved SAC technique. The results indicate that the Improved SAC algorithm outperforms the standard SAC method in terms of rewards accumulated.

B. Tasting Phase

In the testing phase, five iterations of tests were performed on the robot, each consisting of 1000 steps per episode and one pre-determined goal. The outcomes of these interactions are shown in Table II which indicates the collisions that occurred, the time for the robot to reach its goal, and in what step receive the reward. In order to compare the performance of standard SAC and improved SAC techniques in two different environments, a trajectory of the path that the robot frequented was constructed for each method, as illustrated in Fig. 15. The goal is moving from the starting point to the goal represented by the red square without colliding with any obstacles. Red circle represented the places where the robot collided into some obstacle. The white dots represent the path of moving obstacles.

The robot's trajectory with the improved SAC approach can be seen in a static environment, indicating that it selected the best path without colliding into any obstacles. Conversely, with the standard SAC method, it was noticed that the robot collides with a stationary obstacle. As for the dynamic environment, the improved SAC approach can show that the robot has chosen the best trajectory without hitting any obsta-

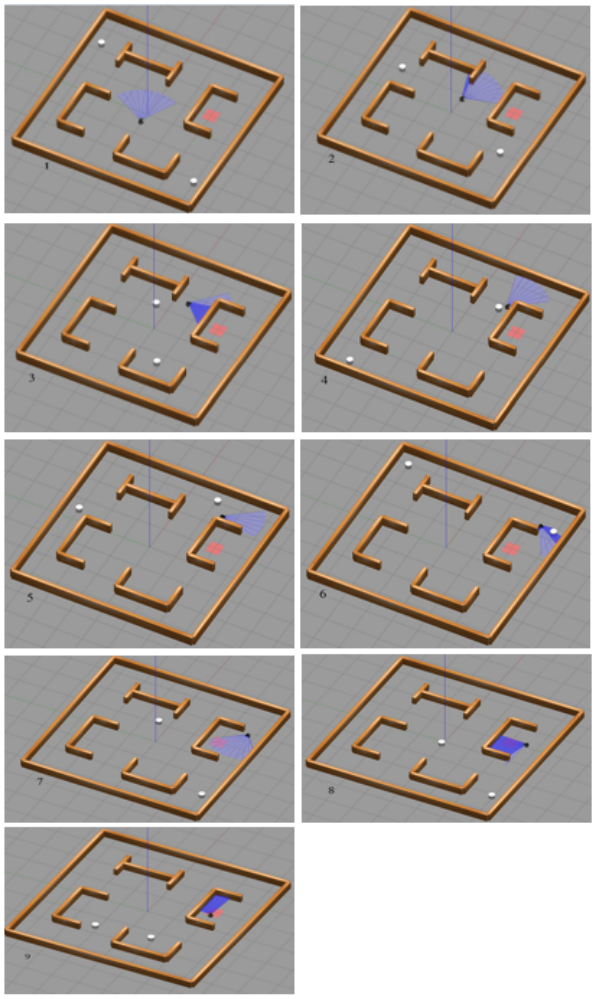

Fig. 13. Robot's trajectory towards the goal in dynamic environment.

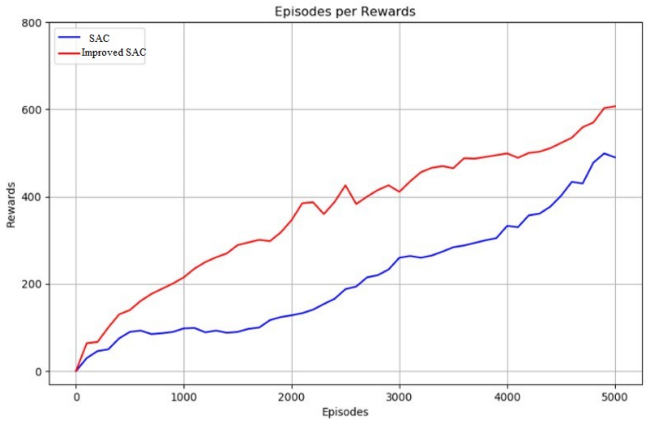

Fig. 14. Average cumulative reward in second environment.

cles and with only a few needless detours. On the contrary, when using standard SAC method, it was observed from the robot's trajectory that there are multiple collisions with a lot of extraneous paths.

It was noticed that in the static scenario, the standard SAC method completed five iterations with an average time of 38.5 seconds (excluding collision iterations) and an average of 159.5 steps per episode (considering only episodes without collisions), as shown in the Table II. It experienced one collision and achieved an accuracy of 80 %. In contrast, the improved SAC technique demonstrated improved performance with no collisions across all five iterations, achieving 100 % accuracy. However, it had an average time of 43.6 seconds and an average of 177.8 steps per episode. The study also revealed that the robot performed well in the static environment when using improved SAC technique and avoided colliding into any obstacles. Furthermore, there was only one collision in the standard SAC method.

In the dynamic scenario, the performance of the standard SAC approach was noticeably less consistent. From the Table II, one can observe that it had an average time of 121.3 seconds (excluding collision episodes) and an average of 574.3 steps per episode (considering only episodes without collisions). It experienced two out of five episodes resulting in collisions, yielding an accuracy of 60 % . On the other hand, the improved SAC method showed significant improvements, completing all episodes without any collisions and achieving 100 % accuracy. The improved SAC approach also had an average of 264 steps per episode, average time was 61.2 seconds. Comparably, in the dynamic environment, the robot had two points of collision while using the standard SAC technique, whereas improved SAC approach had no collisions.

VII. Conclusions

This work suggests a novel method for path planning mobile robot in two different scenarios, considering both moving and static obstacles while prioritizing efficiency and performance. The method improves the SAC technique by incorporating it with the VFH approach for obstacle avoidance. The pipeline of the proposed method includes using the SAC technique for navigating and the VFH method for real-time obstacle avoidance. Generally, the proposed method consistently outperformed the standard SAC approach in both static and dynamic scenarios, particularly in terms of collision avoidance and accuracy, making it a more robust choice for dynamic path planning and obstacle avoidance tasks.

The performance of the method was verified by using Gazebo simulations. The outcomes demonstrate that the pro-

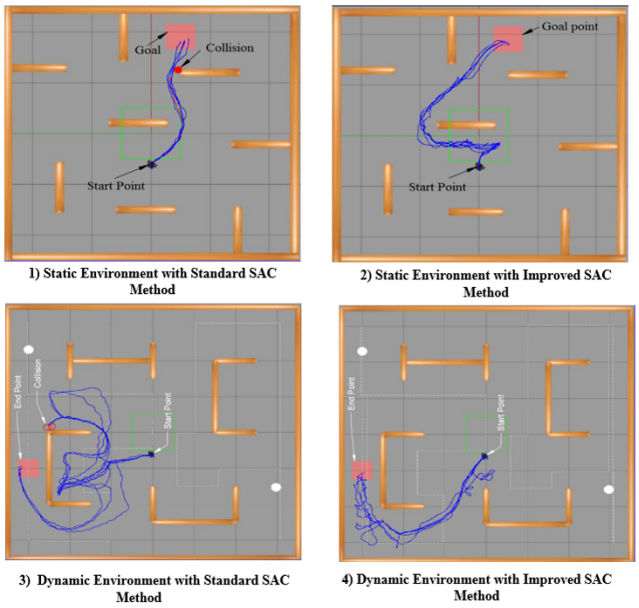

Fig. 15. Trajectory of mobile robot.

posed method effectively processes path planning and dynamic collision avoidance for mobile robots. Additionally, the experimental results indicate that the technique offers significant benefits in path planning and safety when avoiding multiple moving obstacles, thereby enhancing the robots' efficiency in such scenarios.

This incorporated approach shows potential in improving the navigation capabilities of mobile robots, making them more proficient in navigating challenging and unpredictable scenarios.

CONFLICT OF INTEREST

The authors have no conflict of relevant interest to this article.

TABLE II.
TEST OUTCOMES

Environments	Techniques	Episodes	Times (sec)	Steps	Collision	Accuracy
Static	SAC	1	38	154	No	80%
		2	42	169	No	
		3	-	163 collision step	Yes	
		4	41	165	No	
		5	33	150	No	
	Improved SAC	1	41	167	No	100%
		2	60	243	No	
		3	43	164	No	
		4	39	163	No	
		5	35	152	No	
Daynamic	SAC	1	-	283 collision step	Yes	60%
		2	124	590	No	
		3	153	723	No	
		4	-	286 collision step	Yes	
		5	87	410	No	
	Improved SAC	1	72	306	No	100%
		2	45	217	No	
		3	70	281	No	
		4	48	216	No	
		5	71	300	No	

REFERENCES

[1] S. Al-Ansarry and S. Al-Darraji, "Hybrid rrt-a*: An improved path planning method for an autonomous mobile robots.," *Iraqi Journal for Electrical & Electronic Engineering*, vol. 17, no. 1, 2021.

[2] Z. Haruna, M. B. Mu'azu, A. Umar, and G. O. Ufuoma, "Path planning algorithms for mobile robots: A survey," in *Motion Planning for Dynamic Agents*, IntechOpen, 2023.

[3] C. Liu, S. Xie, X. Sui, Y. Huang, X. Ma, N. Guo, and F. Yang, "Prm-d* method for mobile robot path planning," *Sensors*, vol. 23, no. 7, p. 3512, 2023.

[4] J. A. Abdulsaheb and D. J. Kadhim, "Classical and heuristic approaches for mobile robot path planning: A survey," *Robotics*, vol. 12, no. 4, p. 93, 2023.

[5] L. Bing, "Path planning of mobile robot based on improved artificial potential field method," *International Journal of Engineering Continuity*, vol. 2, no. 2, pp. 55–61, 2023.

[6] N. Abdalmanan, K. Kamarudin, M. A. A. Bakar, M. H. F. Rahiman, A. Zakaria, S. M. Mamduh, and L. M. Kamarudin, "2d lidar based reinforcement learning for multi-target path planning in unknown environment," *IEEE Access*, vol. 11, pp. 35541–35555, 2023.

[7] A. Marashian and A. Razminia, "Mobile robot's path-planning and path-tracking in static and dynamic environments: Dynamic programming approach," *Robotics and Autonomous Systems*, vol. 172, p. 104592, 2024.

[8] Y. Sung, J. Das, and P. Tokekar, "Decision-theoretic approaches for robotic environmental monitoring–a survey," *arXiv preprint arXiv:2308.02698*, 2023.

[9] R. Raj and A. Kos, "An optimized energy and time constraints-based path planning for the navigation of mobile robots using an intelligent particle swarm optimization technique," *Applied Sciences*, vol. 13, no. 17, p. 9667, 2023.

[10] L. Yang, P. Li, S. Qian, H. Quan, J. Miao, M. Liu, Y. Hu, and E. Memetimin, "Path planning technique for mobile robots: A review," *Machines*, vol. 11, no. 10, p. 980, 2023.

[11] Y. Long, S. Liu, D. Qiu, C. Li, X. Guo, B. Shi, and M. S. AbouOmar, "Local path planning with multiple constraints for usv based on improved bacterial foraging optimization algorithm," *Journal of Marine Science and Engineering*, vol. 11, no. 3, p. 489, 2023.

[12] P. G. Luan and N. T. Thinh, "Hybrid genetic algorithm based smooth global-path planning for a mobile robot," *Mechanics Based Design of Structures and Machines*, vol. 51, no. 3, pp. 1758–1774, 2023.

[13] L. Liu, X. Wang, X. Yang, H. Liu, J. Li, and P. Wang, "Path planning techniques for mobile robots: Review and prospect," *Expert Systems with Applications*, vol. 227, p. 120254, 2023.

[14] F. Li, Y.-C. Kim, and B. Xu, "Non-standard map robot path planning approach based on ant colony algorithms," *Sensors*, vol. 23, no. 17, p. 7502, 2023.

[15] A. Jabini and E. A. Johnson, "A deep reinforcement learning approach to sensor placement under uncertainty," *IFAC-PapersOnLine*, vol. 55, no. 27, pp. 178–183, 2022.

[16] L. Chen, Z. Jiang, L. Cheng, A. C. Knoll, and M. Zhou, "Deep reinforcement learning based trajectory planning under uncertain constraints," *Frontiers in Neurorobotics*, vol. 16, p. 883562, 2022.

[17] H. Liu, Y. Shen, S. Yu, Z. Gao, and T. Wu, "Deep reinforcement learning for mobile robot path planning," *arXiv preprint arXiv:2404.06974*, 2024.

[18] K. Kasaura, S. Miura, T. Kozuno, R. Yonetani, K. Hoshino, and Y. Hosoe, "Benchmarking actor-critic deep reinforcement learning algorithms for robotics control with action constraints," *IEEE Robotics and Automation Letters*, vol. 8, no. 8, pp. 4449–4456, 2023.

[19] J. Borenstein, Y. Koren, *et al.*, "The vector field histogram-fast obstacle avoidance for mobile robots," *IEEE transactions on robotics and automation*, vol. 7, no. 3, pp. 278–288, 1991.

[20] C. Chronis, G. Anagnostopoulos, E. Politi, G. Dimitrakopoulos, and I. Varlamis, "Dynamic navigation in unconstrained environments using reinforcement learning algorithms," *IEEE Access*, 2023.

[21] R. Chai, H. Niu, J. Carrasco, F. Arvin, H. Yin, and B. Lennox, "Design and experimental validation of deep reinforcement learning-based fast trajectory planning and control for mobile robot in unknown environment," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 35, no. 4, pp. 5778–5792, 2022.

[22] R. Huang, C. Qin, J. L. Li, and X. Lan, "Path planning of mobile robot in unknown dynamic continuous environment using reward-modified deep q-network," *Optimal Control Applications and Methods*, vol. 44, no. 3, pp. 1570–1587, 2023.

[23] A. N. Kumaar and S. Kochuvila, "Mobile service robot path planning using deep reinforcement learning," *IEEE Access*, 2023.

[24] L. Zheng, Y. Wang, R. Yang, S. Wu, R. Guo, and E. Dong, "An efficiently convergent deep reinforcement learning-based trajectory planning method for manipulators in dynamic environments," *Journal of Intelligent & Robotic Systems*, vol. 107, no. 4, p. 50, 2023.

[25] L. Yang, J. Bi, and H. Yuan, "Dynamic path planning for mobile robots with deep reinforcement learning," *IFAC-PapersOnLine*, vol. 55, no. 11, pp. 19–24, 2022.

[26] H. Gong, P. Wang, C. Ni, and N. Cheng, "Efficient path planning for mobile robot based on deep deterministic policy gradient," *Sensors*, vol. 22, no. 9, p. 3579, 2022.

[27] M. Kobayashi, H. Zushii, T. Nakamura, and N. Motoi, "Local path planning: Dynamic window approach with q-learning considering congestion environments for mobile robot," *IEEE Access*, 2023.

[28] Y. Tan, Y. Lin, T. Liu, and H. Min, "Pl-td3: A dynamic path planning algorithm of mobile robot," in *2022 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pp. 3040–3045, IEEE, 2022.

[29] Y. Cao, K. Ni, T. Kawaguchi, and S. Hashimoto, "Path following for autonomous mobile robots with deep reinforcement learning," *Sensors*, vol. 24, no. 2, 2024.

[30] Z. He, L. Dong, C. Song, and C. Sun, "Multiagent soft actor-critic based hybrid motion planner for mobile robots," *IEEE transactions on neural networks and learning systems*, vol. 34, no. 12, pp. 10980–10992, 2022.

[31] K. Katona, H. A. Neamah, and P. Korondi, "Obstacle avoidance and path planning methods for autonomous navigation of mobile robot," *Sensors*, vol. 24, no. 11, p. 3573, 2024.

[32] L. Tai, G. Paolo, and M. Liu, "Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation," in *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pp. 31–36, IEEE, 2017.

[33] J. C. de Jesus, V. A. Kich, A. H. Kolling, R. B. Grando, M. A. d. S. L. Cuadros, and D. F. T. Gamarra, "Soft actor-critic for navigation of mobile robots," *Journal of Intelligent & Robotic Systems*, vol. 102, no. 2, p. 31, 2021.

[34] J. C. Jesus, J. A. Bottega, M. A. Cuadros, and D. F. Gamarra, "Deep deterministic policy gradient for navigation of mobile robots in simulated environments," in *2019 19th International Conference on Advanced Robotics (ICAR)*, pp. 362–367, IEEE, 2019.