Optimizing Mobile Robot Navigation Through Neuro-Symbolic Fusion of Deep Deterministic Policy Gradient (DDPG) and Fuzzy Logic

Muhammad Faqiihuddin Nasary [0009-0009-3373-1356], Azhar Mohd Ibrahim [0000-0003-3294-2249],Suaib Al Mahmud [0000-0002-0091-7949], Amir Akramin Shafie [0000-0003-2440-7537] and Muhammad Imran Mardzuki[0009-0009-5817-8374]

Advanced Multi-Agent System Lab
Department of Mechatronics Engineering
International Islamic University Malaysia

Abstract. Mobile robot navigation has been a sector of great importance in the autonomous systems research arena for a while. For ensuring successful navigation in complex environments several rule-based traditional approaches have been employed previously which possess several drawbacks in terms of ensuring navigation and obstacle avoidance efficiency. Compared to them, reinforcement learning is a novel technique being assessed for this purpose lately. However, the constant reward values in reinforcement learning algorithms limits their performance capabilities. This study enhances the Deep Deterministic Policy Gradient (DDPG) algorithm by integrating fuzzy logic, creating a neuro-symbolic approach that imparts advanced reasoning capabilities to the mobile agents. The outcomes observed in the environment resembling real-world scenarios, highlighted remarkable performance improvements of the neuro-symbolic approach, displaying a success rate of 0.71% compared to 0.39%, an average path length of 35 meters compared to 25 meters, and an average execution time of 120 seconds compared to 97 seconds. The results suggest that the employed approach enhances the navigation performance in terms of obstacle avoidance success rate and path length, hence could be reliable for navigation purpose of mobile agents.

Keywords: DDPG, Fuzzy Logic, Mobile Robot Navigation, Obstacle Avoidance, Simulation.

1 Introduction

Mobile robotics has advanced significantly, with a growing focus on incorporating AI and machine learning techniques to improve navigation capabilities [1]–[6]. Deep reinforcement learning, a subset of reinforcement learning that employs deep neural networks, has been highly effective in various tasks, including mobile robot navigation. In this regard, the DDPG algorithm, a variant of deep reinforcement learning [7] is used in this study to enhance the navigation capabilities of a mobile robot. The DDPG algorithm is implemented using MATLAB and Simulink software, which are widely used in the field of control systems and robotics. DDPG algorithm is a classic deep

reinforcement learning technique [8]–[10], which has a significant advantage in continuous control issues. The DDPG algorithm is built around the actor critical method, the DQN algorithm, and the deterministic strategy gradient (DPG). DQN algorithm uses deep neural network's powerful function fitting ability to map environmental state into action strategy and state action pair to value function, avoiding the problem of Q table's large storage space. The integration of the Deep Q- Network (DQN) algorithm within the Deep Deterministic Policy Gradient (DDPG) algorithm is significant as it leverages the powerful function fitting ability of deep neural networks to map environmental states to action strategies and state-action pairs to value functions [3], [11], [12], effectively addressing the issue of large storage space required by Q tables. This integration enhances the overall performance and scalability of the DDPG algorithm, enabling it to tackle complex decision-making problems in reinforcement learning with greater efficiency and effectiveness.

The current limitations of mobile robot navigation in dynamic and unstructured environments pose significant challenges in the field of robotics. So many traditional methods have been employed before for navigation, obstacle avoidances and path planning purposes [13]–[22]. Traditional navigation methods, such as pre- programmed paths or sensor-based localization, may prove inadequate in such scenarios [23]. Therefore, more sophisticated navigation approaches are required to enable mobile robots to learn and adapt to dynamic environments [11]. Self-learning approaches have been employed in a few studies before to tackle this issue [23], [24], [33]–[36], [25]–[32]. Still the problems persist to several degrees because of the existence of non-reasoning capabilities in self learning algorithms.

This research aims to examine the efficacy of the DDPG algorithm, particularly in conjunction with a neuro-symbolic approach, in enhancing the navigation capabilities of a mobile robot within dynamic and unstructured environments and to assess the navigational accuracy, adaptability, and robustness exhibited by the DDPG algorithm across a diverse array of simulated environments, thus quantitatively evaluating its performance in a comprehensive manner. Specifically, the ability of the algorithm to guide a mobile robot through complex and dynamic environments while avoiding obstacles and reaching a desired target will be assessed [37]. The outcomes of this study will not only demonstrate the potential of the DDPG algorithm but also highlight areas for improvement, paving the way for future research in this exciting field.

1.1 Deep Deterministic Policy Gradient Algorithm

This study investigates the use of the DDPG algorithm for reinforcement learning in a scenario where a mobile robot equipped with range sensors navigates an environment to avoid obstacles. The DDPG algorithm is well-suited for continuous action spaces and utilizes an actor-critic architecture where the actor network represents the policy, and the critic network represents the value function. To simulate range sensor readings,

an occupancy map of a known environment was employed, and collision avoidance was achieved by determining optimal controls for the robot in terms of linear and angular velocity. The DDPG agent utilized sensor readings as input and generated velocity controls as output. By iteratively exploring the environment and receiving rewards as feedback, the agent learned to make better decisions and effectively avoid obstacles.

1.2 Neuro-symbolic approach through Fuzzy Logic in reward function

This research work introduces a novel neuro-symbolic approach that enhances AI with reasoning capabilities for mobile robot navigation.

By integrating evaluative Fuzzy logic functions, the proposed system harmonizes deep reinforcement learning (DRL) with symbolic reasoning, enabling the robot to reason and optimize DRL computations during navigation. The key emphasis lies in the construction of the reward function, where fuzzy logic plays a pivotal role. This integration empowers the system with the ability to learn, adapt, and operate effectively in unknown environments, facilitating seamless real-world deployment and interaction. The incorporation of fuzzy logic in the reward function enables the system to achieve robust and interpretable decision-making capabilities, thus significantly enhancing the overall efficiency and reliability of the navigation process. This research highlights the crucial significance of incorporating fuzzy logic to enhance reasoning and optimize DRL computations in mobile robot navigation systems.

2 Methods

2.1 Overview of the Hybrid System Architecture of DDPG and Fuzzy Logic

In the hybrid system architecture as shown in Figure 1, fuzzy logic is integrated into the DDPG algorithm by incorporating it into the reward function. The reward function serves as a critical component in reinforcement learning algorithms, providing feedback to the agent based on its actions in the environment. In this case, the fuzzy logic system is used to define the reward values based on the inputs obtained from the environment. These inputs can include various factors such as proximity to obstacles, speed, and other relevant information. The fuzzy logic system processes these inputs and generates a reward value that captures the desirability or undesirability of the agent's actions.

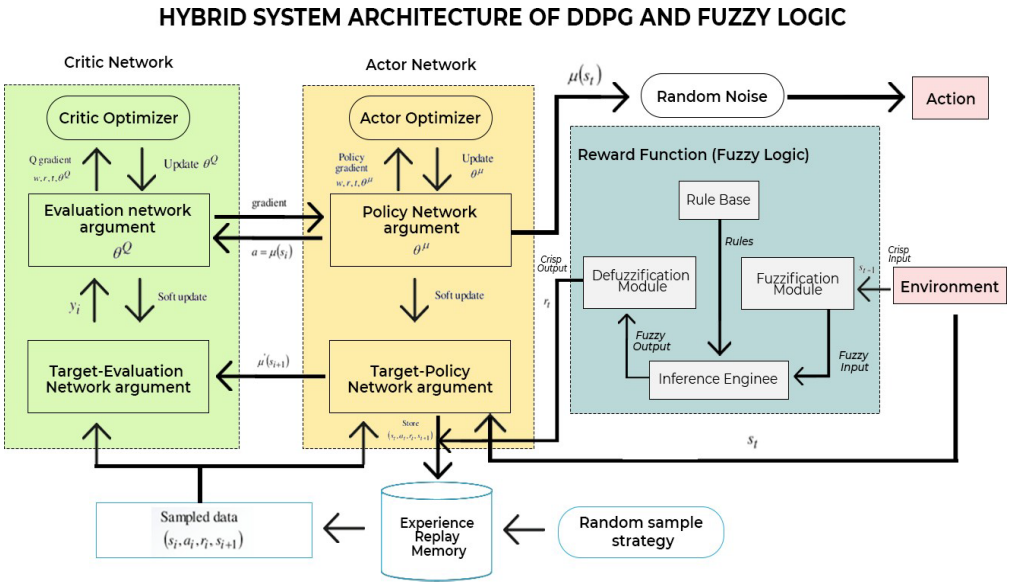


Fig. 1. Environment Interface which responsible to take the action, give the observation and reward signals.

2.2 Constructing DDPG Agent

The DDPG algorithm, like its counterpart, the Actor-Critic algorithm, consists of two primary networks known as the Actor and Critic networks. In the Actor-network, the output is a determined action, defined by $a = \mu\,(s, a|\theta^{\mu})$. The estimation network, referred to as $\mu_{\theta}(s)$, is responsible for producing real-time actions. The parameters for this network are indicated by θ^{μ}. Through updates to the parameter denoted by θ^{μ}, it generates action A based on the current state described by s_t. This action then interacts with the environment, resulting in the generation of the next state s_{t+1} and its corresponding reward r_{t+1}. The Actor target network updates parameters within the Critic network and determines the optimal action denoted by a_{t+1} based on the next state s_{t+1} obtained from the experience replay.

The primary objective of the critic network in the DDPG algorithm is to approximate the value function $Q(s, a|\theta^{Q})$. To achieve this, the Critic estimation network updates its parameters θ^{Q} and calculates two important values: the current Q value $Q(s_t, a_t, \theta^{Q})$

and the target Q value $yi = r + \gamma^{Q'}(st+1, a^{t+1}, \theta^{Q'})$. The target Q value involves the Q′ component, which is computed by the Critic target network. In this context, the discount factor γ plays a crucial role, as it determines the weightage given to future rewards. Its value ranges between 0 and 1.
The critic network undergoes a training process where the goal is to minimize the loss function L, as depicted in Equation (1).

$$L = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i \,|\theta^Q))^2 \tag{1}$$

Where *yi* is given by Equation (2).

$$y_i = r\,(s_i a_i) + \gamma Q'\left(s_{i+1}, \mu'\left(s_{i+1}\middle|\theta^{\mu'}\right)\middle|\ \theta^{Q'}\right) \tag{2}$$

The Actor network is updated through a policy gradient show in Equation (3).

$$\nabla_{\theta\mu} J \ \approx\ \frac{1}{N}\sum_i \nabla_a Q\ (s, a \mid \theta^Q)\,|_{s=s_i, a=\mu(s_i)} \nabla_{\theta\mu}\mu\ (s\,|\theta^\mu)|_{s_i} \tag{3}$$

Whereas, the Target networks are updated through Equation (4) and Equation (5), with $\tau \ll 1$.

$$\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'} \tag{4}$$
$$\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'} \tag{5}$$

2.3 Constructing Reward Function using Fuzzy Logic

The reward function employed in this study was primarily constructed using Fuzzy Logic, which involved three crucial input variables: Obstacle Distance (with 5 Membership Functions), Linear Speed (with 5 Membership Functions), and Angular Speed (with 3 Membership Functions). Figure 2 shows the Membership Functions for Linear Speed, Angular Speed, Obstacle Avoidance and Total Reward.

The Fuzzy Logic framework allowed for the formulation of a total of 75 rules (resulting from the combination of the membership functions), as shown in Table 1. This comprehensive reward system was designed to guide the agent's behavior effectively and ensure appropriate decision- making in the given environment.
Control surfaces play a crucial role in fuzzy logic systems as they serve as the bridge

between linguistic input variables and linguistic output variables. These surfaces, as shown in Figure 3, are responsible for mapping the fuzzy sets of input variables to fuzzy sets of output variables, based on a set of predefined rules and membership functions.

By utilizing control surfaces, fuzzy logic systems can effectively process and interpret linguistic input information and generate appropriate linguistic output responses. This enables the system to handle imprecise, uncertain, or qualitative data and make intelligent decisions based on human-like reasoning. Thus, control surfaces in fuzzy logic form a fundamental component that facilitates the transformation of linguistic information into actionable control actions, making them essential for a wide range of applications in decision-making, control systems, and artificial intelligence domains.

Table 1. Rules table for reward function using three main input variables.

OD	AS	LS				
		VL	L	M	H	VH
VNR	L	P	P	P	G	G
	M	P	P	P	P	G
	H	VP	VP	VP	VP	G
NR	L	G	G	G	G	G
	M	P	P	P	P	E
	H	VG	VG	VG	VG	E
M	L	VG	VG	VG	VG	E
	M	G	G	G	VG	E
	H	P	G	G	G	VG
F	L	E	VE	VE	VE	VE
	M	VG	E	E	E	E
	H	VG	E	E	E	VE
VF	L	A	A	A	A	A
	M	VE	VE	A	A	A
	H	VE	VE	VE	VE	VE

OD: Obstacle Avoidance
MF: Very Near (Very Near), Near (NR), M (Medium), Far (F), Very Far)

AS: Angular Speed
MF: Low (L), Medium (M), High (H)

LS: Linear Speed
MF: Very Low (VL), Low (L), Medium (M), High (H), Very High (VH)

Total Rewards (Grey Box)
MF: Very Poor (VP), Poor (P), Good (G), Very Good (VG), Excellent (E), Very Excellent (VE), Amazing (A)

INPUT 1: LINEAR SPEED		
Range [0 0.3] m/s		
Name	**Type**	**Parameters (m/s)**
Very Low	Triangular	[0 0 0.075]
Low	Triangular	[0 0.075 0.15]
Medium	Triangular	[0.075 0.15 0.225]
High	Triangular	[0.15 0.225 0.3]
Very High	Triangular	[0.225 0.3 0.3]

(a)

INPUT 2: ANGULAR SPEED		
Range [-0.3 0.3] rad/s		
Name	**Type**	**Parameters (rad/s)**
Low	Triangular	[-0.3 -0.3 0]
Medium	Triangular	[-0.3 0 0.3]
High	Triangular	[0 0.3 0.3]

(b)

INPUT 3: OBSTACLE AVOIDANCE		
Range [0 15] m		
Name	**Type**	**Parameters (m)**
Very Near	Triangular	[0 0 4]
Near	Triangular	[0 4 8]
Moderate	Triangular	[4 8 12]
Far	Triangular	[8 12 15]
Very Far	Triangular	[12 15 15]

(c)

OUTPUT: TOTAL REWARD		
Range [0 0.6]		
Name	**Type**	**Parameters**
Very Poor	Triangular	[0 0 0.1]
Poor	Triangular	[0 0.1 0.2]
Good	Triangular	[0.1 0.2 0.3]
Very Good	Triangular	[0.2 0.3 0.4]
Excellent	Triangular	[0.3 0.4 0.5]
Very Excellent	Triangular	[0.4 0.5 0.6]
Amazing	Triangular	[0.5 0.6 0.6]

(d)

Fig. 2. The membership Functions for (a) Linear Speed. (b) Angular Speed. (c) Obstacle Avoidance. (d) Total Reward.

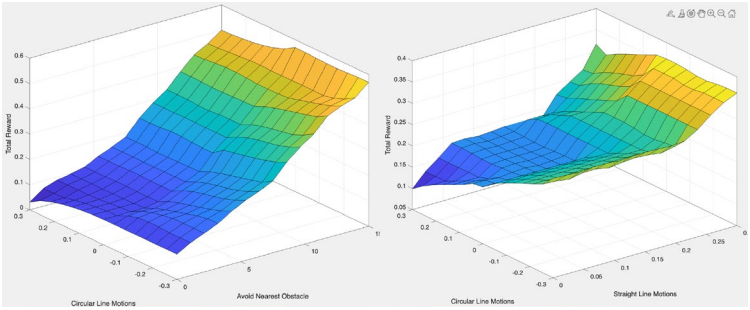


Fig. 3. Control Surface between two input variables. Obstacle Distance & Angular Speed (Left), Obstacle Distance & Linear Speed

2.4 Agents' Training

To effectively train agents, it is crucial to determine the appropriate training options. In this study, the agent undergoes training for a maximum of 6000 episodes, with each

episode limited to a certain number of time steps. The Episode Manager dialog box in breakdown of the agent's performance at each step. Adjustable hyperparameters such as the maximum number of episodes, maxSteps, and stopping criteria are dependent on the specific task and available resources. The Reinforcement Learning Episode Manager dialog box facilitates the visual assessment of the agent's performance, while the command line display provides a comprehensive analysis that aids in understanding the agent's behavior and interpreting the results. The environment of the whole training can be seen in Figure 4. The significance of training and testing an agent in different environments lies in evaluating the robustness and adaptability of the trained model.

The initial simulation employed the DDPG algorithm, focusing on training and testing an agent in a simple environment named "simplemap." This environment was basic, with few obstacles. In contrast, the current simulation pushed the agent into a more intricate setting, such as “complexMap” and “officeMap” replicating a real office environment via lidar scans. The goal was to evaluate the agent's adaptability to novel environments.

Understanding DDPG's traits in this experiment is crucial. This algorithm aims to optimize return by approximating the Q-function and policy using neural networks: critic and actor networks. The capacity of these networks significantly impacts the agent's adaptability. Notably, DDPG excels in scenarios with continuous action spaces.

The latest simulation showcased the agent's proficiency in navigating a new environment using the same trained model, emphasizing DDPG's adaptability, as evident in Figure 4. This hints that DDPG has learned a policy effective not only in the simplemap but also in more complex settings. It's noteworthy that the DDPG algorithm, designed for continuous action spaces, hinges on the capabilities of its neural networks: the critic and actor networks, pivotal for its generalization ability.

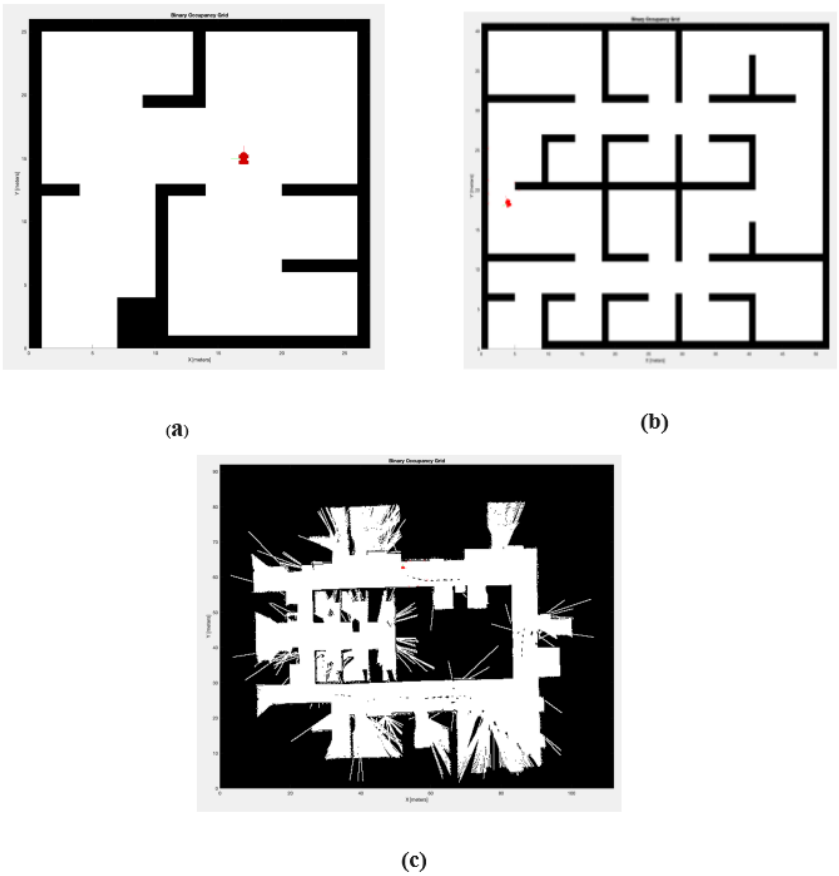


Fig. 2. Snapshots of collision-free trajectories of mobile robots based on the trained agents using (a) simple map. (b) complex map. (c) office map.

3 Experiments & Results

3.1 DDPG Agent Training Results

As shown in Figure 5, the observed rise in average reward per episode demonstrates the successful learning of the agent using the DDPG algorithm. This improvement in reward, both on a per-episode basis and when considering the average across all episodes, indicates the agent's enhanced decision- making capabilities as it accumulates more

experience. This progress aligns with the fundamental objective of reinforcement learning, emphasizing the effectiveness of the DDPG algorithm in maximizing cumulative rewards over time by enabling the agent to refine its decision-making through experiential learning.

3.2 Mobile Robot Simulation Results

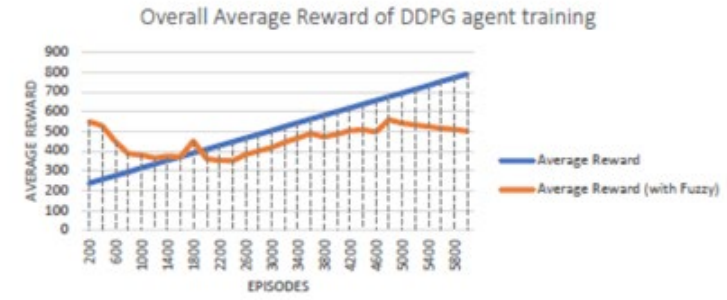
In the obstacle avoidance simulation for a mobile robot, the rewards and penalties were carefully designed to guide the robot's navigation. Rewards were assigned for achieving objectives, such as avoiding the nearest obstacle or moving in a straight line, while penalties were imposed for undesirable actions like circular movements. The DDPG algorithm aims to maximize the expected return, which is the cumulative sum of rewards while minimizing penalties. The agent's behavior is encouraged to avoid obstacles, maintain linear velocity, and avoid circular motions.

It is important to consider that the reward and penalty scheme was tailored to the specific task and environment of the simulation and may require adjustments for different scenarios. Nonetheless, the simulation demonstrate the effectiveness of the DDPG algorithm in learning an effective obstacle avoidance policy within the given environment for both maps.

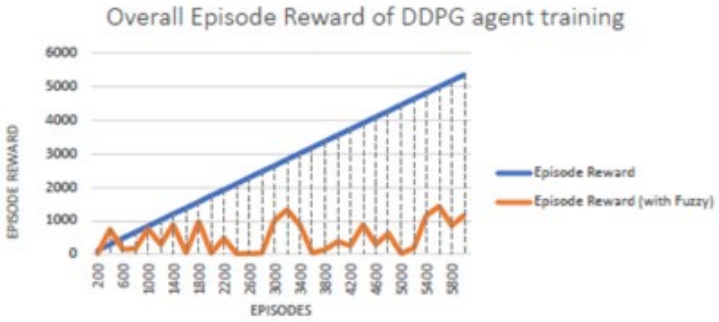
4 Discussions

4.1 Overall Reward for DDPG Training

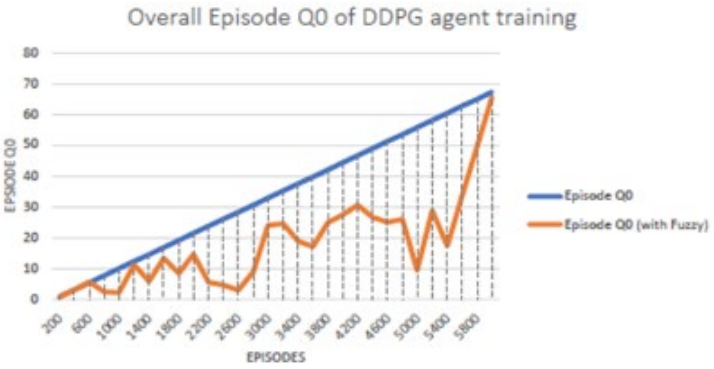
The increase in average reward per episode as shown in Figure 5 can be attributed to two key factors in the DDPG algorithm. Firstly, the actor network learns an improved policy by adjusting its parameters to select actions that lead to higher rewards. This process, known as policy gradient, enables the agent to make better decisions over time. Secondly, the critic network learns a more accurate value function by adjusting its parameters to estimate the expected cumulative reward of state-action pairs. This process, known as Q-learning, enhances the agent's ability to assess the value of actions. The successful learning of both the actor and critic networks contributes to the observed rise in average reward per episode, showcasing the effectiveness of the DDPG algorithm in optimizing decision-making and overall performance.


Overall Average Reward of DDPG agent training
AVERAGE REWARD
EPISODES
Average Reward
Average Reward (with Fuzzy)

(a)


Overall Episode Reward of DDPG agent training
EPISODE REWARD
EPISODES
Episode Reward
Episode Reward (with Fuzzy)

(b)


Overall Episode Q0 of DDPG agent training
EPISODE Q0
EPISODES
Episode Q0
Episode Q0 (with Fuzzy)

(c)

Fig. 3. 6000 navigation tasks are generated in each testing environment **(a)** Average reward represents the average reward over multiple episodes. **(b)** Episode Reward reflects the reward obtained during a single episode **(c)** Episode Q0 represents the estimate of the discounted long-term reward at the beginning of each episode, based on the initial observation of the environment.

4.2 Mobile Robot Simulation Performance

In the initial mobile robot simulation, the DDPG algorithm was employed to train and test an agent in a simplified environment called Simple Map and Complex Map. This environment had minimal obstacles and flat terrain. In contrast, the current simulation introduced the agent to a more realistic and complex environment called Office Map (Figure 6), generated from lidar scans of a real-world office setting. This extension aimed to assess the algorithm's ability to generalize to unfamiliar environments. The results, depicted in Figure 6 using the same trained model, demonstrate the DDPG algorithm's capacity for generalization as the agent successfully navigated the new environment. This suggests that the algorithm has learned a policy that is effective in both simplified and realistic environments.

The use of fuzzy logic within the neuro-symbolic approach has notably surpassed the initial method (non-fuzzy) reliant on fixed values. Quantitative analysis showcases a substantial enhancement in obstacle avoidance efficiency. The incorporation of multiple input variables—such as obstacle distance, linear, and angular speeds—within the fuzzy logic system enables a more intricate decision-making process. This approach capitalizes on fuzzy logic's capacity to manage imprecise data, effectively capturing complex interrelationships between variables to generate refined and adaptable actions.

When navigating narrow passages, the neuro-symbolic approach employing fuzzy logic outperforms the fixed values approach. The system's sensitivity to obstacle proximity allows the mobile robot to detect and respond more effectively to tight spaces. Leveraging membership functions and fuzzy inference, this approach finely adjusts the robot's actions, adeptly maneuvering obstacles in confined areas while maintaining safety. Such precision results in smoother and more dependable navigation within constrained environments.

Additionally, the neuro-symbolic approach utilizing fuzzy logic showcases heightened caution in handling obstacles. Employing membership functions and rule-based reasoning enables the system to assess potential risks associated with varying levels of obstacle proximity. With a broader range of input values and membership functions, the system offers a detailed evaluation of the environment, ensuring the robot maintains a safe distance, reducing collision risks, and enhancing overall safety during navigation.

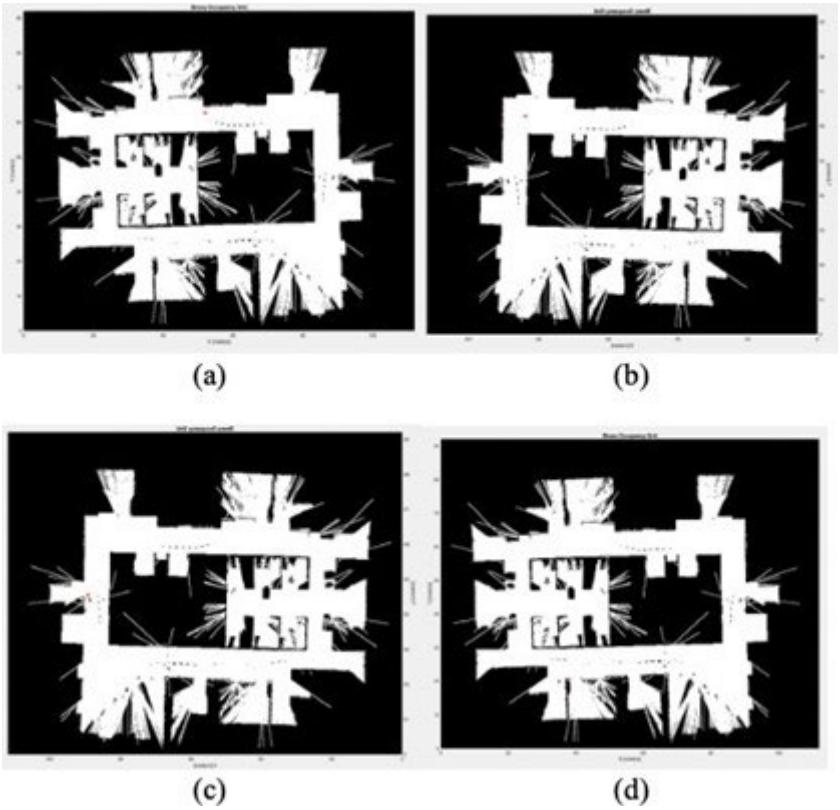


Fig. 4. Snapshots of collision-free trajectories of mobile robots based on the trained agents in office map at time (a) 15 seconds. (b) 30 seconds. (c) 45 seconds. (d) 60 seconds.

4.3 Implementation of Fuzzy Logic in Reward Function

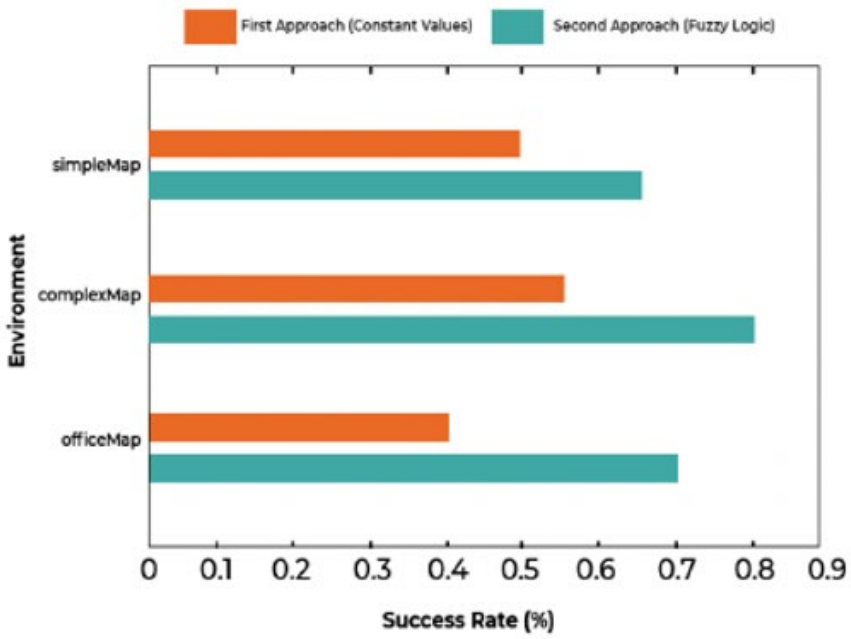
The integration of fuzzy logic into the reward and penalty system of the Deep Deterministic Policy Gradient (DDPG) algorithm as shown in Figure 1 bestows an excessive of advantageous traits to mobile robot navigation. This ingenious process endows the system with remarkable adaptability, allowing it to adeptly handle imprecise or uncertain environments. By seamlessly incorporating fuzzy logic into the reward and penalty system, the system becomes resilient, capable of acclimating to diverse environmental conditions, and deftly responding to situations that lack precise numerical values. Consequently, this heightened adaptability amplifies the learning and training process by

enabling the robot to glean insights from a wider array of scenarios, making astute decisions based on fuzzy inputs.

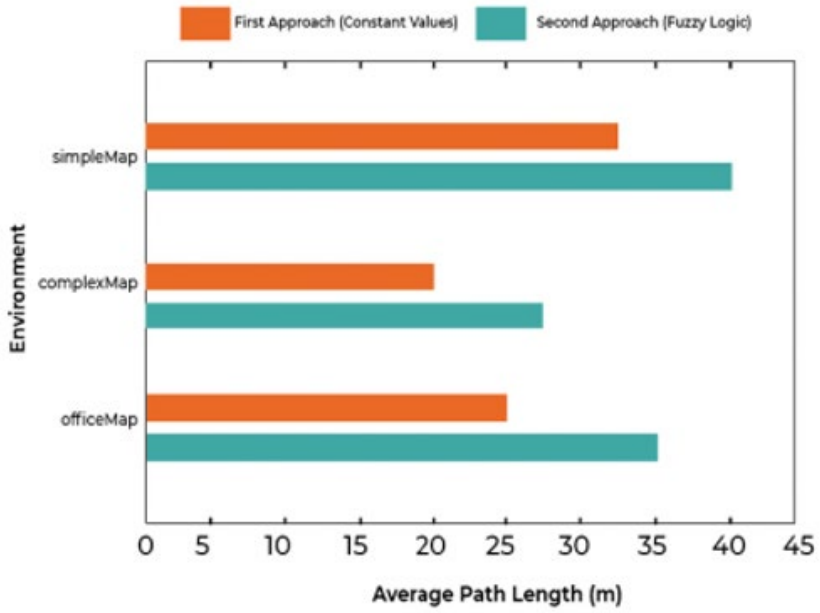
Another notable advantage of integrating fuzzy logic compared to using constant values approach as shown in Figure 7, lies in its ability to bestow robustness upon the system, mitigating the deleterious effects of noise, uncertainties, and variations commonly encountered in real- world settings. The inclusion of fuzzy inputs in the reward and penalty system imparts a formidable shield against external disturbances, bolstering the system's fortitude in less controlled and unpredictable environments. Such fortitude enhances the system's ability to learn and navigate with utmost efficacy, regardless of the surrounding uncertainties.

Based on Figure 7, it shows that by employing Fuzzy Logic in the reward function, yields superior navigation performance, even if it comes at the cost of increased execution time. The incremental time required for processing the Fuzzy Logic-based reward function as our second approach is justified by the substantial improvements achieved in terms of success rates, average path lengths, and the robot's overall ability to navigate through dynamic and unstructured environments.

This approach outperforms the first approach that relies on constant values in terms of quantitative performance metrics and qualitative aspects of adaptability and robustness. The flexibility of the neuro-symbolic approach allows for iterative refinement and optimization of the reward function, resulting in improved navigation capabilities and learning outcomes. The findings emphasize the trade-off between execution time and performance and highlight the value of integrating neuro- symbolic approaches, such as Fuzzy Logic, to enhance the navigation capabilities of mobile robots.


First Approach (Constant Values)
Second Approach (Fuzzy Logic)
simpleMap
complexMap
officeMap
Environment
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
Success Rate (%)

(a)


First Approach (Constant Values)
Second Approach (Fuzzy Logic)
simpleMap
complexMap
officeMap
Environment
0
5
10
15
20
25
30
35
40
45
Average Path Length (m)

(b)

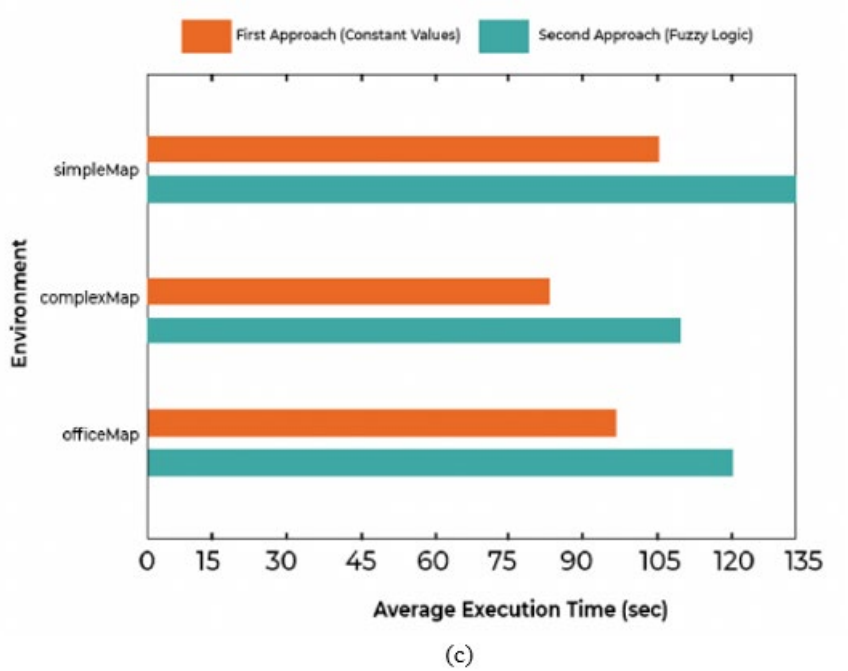


Fig. 5. 6000 navigation tasks are generated in each testing environments. (a) Success rate demonstrates the successful probability to avoid obstacle. (b) Average path demonstrates the mean path lengths for all successful navigation. (c) Average execution time demonstrates the average value of execution time for all successful navigation

5 Conclusion

The successful implementation of the DDPG algorithm in MATLAB and Simulink enabled the simulation of mobile robot navigation in dynamic and unstructured environments. The results demonstrate that the algorithm significantly enhanced the robot's navigation capabilities by maximizing the reward obtained in the simulation. Moreover, the integration of fuzzy logic into the reward function further improved the system's performance and adaptability. The application of fuzzy logic successfully introduced enhanced adaptability, robustness to noise and variability, human-like decision making, smooth transitions, and interpretability to the learning process. This research highlights the potential of DRL methods, specifically DDPG, for mobile robot navigation, and underscores the importance of incorporating fuzzy logic to optimize and validate the performance of DRL-based navigation algorithms in simulation-based studies.

Disclosure Statement
The authors declare that they have no competing interests.

References

[1] X. Lu, H. Woo, A. Faragasso, A. Yamashita, and H. Asama, "Robot navigation in crowds via deep reinforcement learning with modeling of obstacle uni-action," *Adv. Robot.*, vol. 37, no. 4, pp. 257–269, Feb. 2023, doi: 10.1080/01691864.2022.2142068.

[2] C. Shyalika, T. Silva, and A. Karunananda, "Reinforcement Learning in Dynamic Task Scheduling: A Review," *SN Comput. Sci.*, vol. 1, no. 6, 2020, doi: 10.1007/s42979-020-00326-5.

[3] X. Gao, L. Yan, Z. Li, G. Wang, and I.-M. Chen, "Improved Deep Deterministic Policy Gradient for Dynamic Obstacle Avoidance of Mobile Robot," *IEEE Trans. Syst. Man, Cybern. Syst.*, vol. 53, no. 6, pp. 3675–3682, 2023, doi: 10.1109/TSMC.2022.3230666.

[4] A. Bahamid, A. M. Ibrahim, A. Ibrahim, I. Z. Zahurin, and A. N. Wahid, "Intelligent robot-assisted evacuation: A review," *J. Phys. Conf. Ser.*, vol. 1706, no. 1, 2020, doi: 10.1088/1742-6596/1706/1/012159.

[5] Y. Chen, C. Cheng, Y. Zhang, X. Li, and L. Sun, "A Neural Network-Based Navigation Approach for Autonomous Mobile Robot Systems," *Appl. Sci.*, vol. 12, no. 15, 2022, doi: 10.3390/app12157796.

[6] M. R. M. Romlay, M. I. Azhar, S. F. Toha, and M. M. Rashid, "Two-wheel balancing robot; review on control methods and experiments," *Int. J. Recent Technol. Eng.*, vol. 7, no. 6, pp. 106–112, 2019.

[7] T. P. Lillicrap *et al.*, "Continuous control with deep reinforcement learning," *4th Int. Conf. Learn. Represent. ICLR 2016 - Conf. Track Proc.*, 2016.

[8] M. M. A. Lashin and W. S. E. S. Saleh, "Road Safety Policies for Saudi Females: A Fuzzy Logic Analysis," *Sustain.*, vol. 14, no. 7, 2022, doi: 10.3390/su14074154.

[9] M. E. ÇİMEN, Z. GARİP, M. EMEKLİ, and A. F. BOZ, "Fuzzy Logic PID Design using Genetic Algorithm under Overshoot Constrained Conditions for Heat Exchanger Control," *Iğdır Üniversitesi Fen Bilim. Enstitüsü Derg.*, vol. 12, no. 1, pp. 164–181, 2022, doi: 10.21597/jist.980726.

[10] L. A. Nguyen, "Fuzzy simulations and bisimulations between fuzzy automata," *Int. J. Approx. Reason.*, vol. 155, pp. 113–131, 2023, doi: https://doi.org/10.1016/j.ijar.2023.02.002.

[11] Y. Wang, Y. Fang, P. Lou, J. Yan, and N. Liu, "Deep Reinforcement Learning based Path Planning for Mobile Robot in Unknown Environment," *J. Phys. Conf. Ser.*, vol. 1576, no. 1, 2020, doi: 10.1088/1742-6596/1576/1/012009.

[12] Y. Li *et al.*, "Prediction Model for Geologically Complicated Fault Structure Based on Artificial Neural Network and Fuzzy Logic," *Sci. Program.*, vol. 2022, 2022, doi: 10.1155/2022/2630953.

[13] J. Ni, L. Wu, X. Fan, and S. X. Yang, "Bioinspired intelligent algorithm and its applications for mobile robot control: A survey," *Comput. Intell. Neurosci.*, vol. 2016, 2016, doi: 10.1155/2016/3810903.

[14] O. Takahashi and R. J. Schilling, "Motion Planning in a Plane Using Generalized Voronoi Diagrams," *IEEE Trans. Robot. Autom.*, vol. 5, no. 2, pp. 143–150, 1989, doi: 10.1109/70.88035.

[15] P. Bhattacharya and M. L. Gavrilova, "Roadmap-Based Path Planning - Using the Voronoi Diagram for a Clearance-Based Shortest Path," *IEEE Robot. Autom. Mag.*, vol. 15, no. 2, pp. 58–66, 2008, doi: 10.1109/MRA.2008.921540.

[16] E. J. Gómez, F. M. M. Santa, and F. H. M. Sarmiento, "A comparative study of geometric path planning methods for a mobile robot: Potential field and voronoi diagrams," in *2013 II International Congress of Engineering Mechatronics and Automation (CIIMA)*, 2013, pp. 1–6. doi: 10.1109/CIIMA.2013.6682776.

[17] R. Abiyev, D. Ibrahim, and B. Erin, "Navigation of mobile robots in the presence of obstacles," *Adv. Eng. Softw.*, vol. 41, no. 10, pp. 1179–1186, 2010, doi: https://doi.org/10.1016/j.advengsoft.2010.08.001.

[18] A. R. Soltani, H. Tawfik, J. Y. Goulermas, and T. Fernando, "Path planning in construction sites: performance evaluation of the Dijkstra, A∗, and GA search algorithms," *Adv. Eng. Informatics*, vol. 16, no. 4, pp. 291–303, 2002, doi: https://doi.org/10.1016/S1474-0346(03)00018-1.

[19] E. Masehian and M. R. Amin-Naseri, "A voronoi diagram-visibility graph-potencial field compound algorith for robot path planning," *J. Robot. Syst.*, vol. 21, no. 6, pp. 275–300, 2004, doi: 10.1002/rob.20014.

[20] M. Weigl, B. Siemiàatkowska, K. A. Sikorski, and A. Borkowski, "Grid-based mapping for autonomous mobile robot," *Rob. Auton. Syst.*, vol. 11, no. 1, pp. 13–21, 1993, doi: https://doi.org/10.1016/0921-8890(93)90004-V.

[21] A. Ghorbani, S. Shiry, and A. Nodehi, "Using Genetic Algorithm for a Mobile Robot Path Planning," in *2009 International Conference on Future Computer and Communication*, 2009, pp. 164–166. doi: 10.1109/ICFCC.2009.28.

[22] S. Ahmadzadeh and M. Ghanavati, "Navigation of Mobile Robot Using the PSO Particle Swarm Optimization," *J. Acad. Appl. Stud.*, vol. 2, 2012, [Online]. Available: https://api.semanticscholar.org/CorpusID:18077178

[23] K. Zhu and T. Zhang, "Deep reinforcement learning based mobile robot navigation: A review," *Tsinghua Sci. Technol.*, vol. 26, no. 5, pp. 674–691, 2021, doi: 10.26599/TST.2021.9010012.

[24] H. Hu, K. Zhang, A. H. Tan, M. Ruan, C. Agia, and G. Nejat, "A Sim-to-Real Pipeline for Deep Reinforcement Learning for Autonomous Robot Navigation in Cluttered Rough Terrain," *IEEE Robot. Autom. Lett.*, vol. 6, no. 4, pp. 6569–

6576, 2021, doi: 10.1109/LRA.2021.3093551.

[25] B. Liu, L. Wang, and M. Liu, “Lifelong Federated Reinforcement Learning: A Learning Architecture for Navigation in Cloud Robotic Systems,” *IEEE Robot. Autom. Lett.*, vol. 4, no. 4, pp. 4555–4562, 2019, doi: 10.1109/LRA.2019.2931179.

[26] M. Algabri, “Self-learning Mobile Robot Navigation in Unknown Environment Using Evolutionary Learning,” vol. 20, no. 10, pp. 1459–1468, 2014.

[27] S. H. Yusuf, “Mobile Robot Navigation Using Deep Reinforcement Learning,” 2022.

[28] F. Fathinezhad, V. Derhami, and M. Rezaeian, “Supervised fuzzy reinforcement learning for robot navigation,” *Appl. Soft Comput.*, vol. 40, pp. 33–41, 2016, doi: https://doi.org/10.1016/j.asoc.2015.11.030.

[29] P.-H. Ciou, Y.-T. Hsiao, Z.-Z. Wu, S.-H. Tseng, and L.-C. Fu, “Composite Reinforcement Learning for Social Robot Navigation,” in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 2553–2558. doi: 10.1109/IROS.2018.8593410.

[30] A. V Bernstein, E. V Burnaev, and O. N. Kachan, “Reinforcement Learning for Computer Vision and Robot Navigation BT - Machine Learning and Data Mining in Pattern Recognition,” P. Perner, Ed., Cham: Springer International Publishing, 2018, pp. 258–272.

[31] H. Hase *et al.*, “Ultrasound-Guided Robotic Navigation with Deep Reinforcement Learning,” in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 5534–5541. doi: 10.1109/IROS45743.2020.9340913.

[32] L. Liu, D. Dugas, G. Cesari, R. Siegwart, and R. Dubé, “Robot Navigation in Crowded Environments Using Deep Reinforcement Learning,” in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 5671–5677. doi: 10.1109/IROS45743.2020.9341540.

[33] G. Chen *et al.*, “Robot Navigation with Map-Based Deep Reinforcement Learning,” Feb. 2020, [Online]. Available: http://arxiv.org/abs/2002.04349

[34] S. Sun, X. Zhao, Q. Li, and M. Tan, “Inverse reinforcement learning-based time-dependent A* planner for human-aware robot navigation with local vision,” *Adv. Robot.*, vol. 34, no. 13, pp. 888–901, Jul. 2020, doi: 10.1080/01691864.2020.1753569.

[35] J. Bruce, N. Sünderhauf, and P. Mirowski, “One-Shot Reinforcement Learning for Robot Navigation with Interactive Replay,” no. Nips, 2017.

[36] M.-C. Su, D.-Y. Huang, C.-H. Chou, and C.-C. Hsieh, “A reinforcement-learning approach to robot navigation,” in *IEEE International Conference on Networking, Sensing and Control, 2004*, 2004, pp. 665-669 Vol.1. doi: 10.1109/ICNSC.2004.1297519.

[37] Y. Dong and X. Zou, “Mobile Robot Path Planning Based on Improved DDPG

Reinforcement Learning Algorithm," in *2020 IEEE 11th International Conference on Software Engineering and Service Science (ICSESS)*, 2020, pp. 52–56. doi: 10.1109/ICSESS49938.2020.9237641.