Proceedings of 2024 WRC Symposium
on Advanced Robotics and Automation (WRC SARA)
Beijing, China, August 23, 2024

2024 WRC Symposium on Advanced Robotics and Automation (WRC SARA) | 979-8-3315-0610-0/24/$31.00 ©2024 IEEE | DOI: 10.1109/WRCSARA64167.2024.10685828

Continuous Motion Planning for Mobile Robots Using Fuzzy Deep Reinforcement Learning

Fenghua Wu[1], Wenbing Tang[1*], Yuan Zhou[2], Shang-Wei Lin[1], Yang Liu[1], and Zuohua Ding[2]

***Abstract*— Autonomous navigation for mobile robots has found a promising solution with deep reinforcement learning (DRL). This method stands out for its efficiency, particularly because it can operate without requiring extensive labeled datasets during the training process. Current DRL-based motion planning approaches can be categorized into two groups: DRL with discrete action spaces and DRL with continuous action spaces. The former aims to select the best action from a set of predefined actions but suffers from high computational costs. The latter computes actions directly from the continuous action space but may encounter local optima, leading to motion failures. To overcome these challenges, we introduce a novel approach, integrating Deep Deterministic Policy Gradient (DDPG) with fuzzy logic, for the motion of mobile robots. The Actor and Critic networks in our DDPG framework feature an architecture incorporating a Long Short-Term Memory (LSTM) network to handle varying numbers of obstacles in the environment. To address local optima in continuous DRL, we utilize fuzzification to discretize continuous actions into a small number of fuzzy sets and defuzzification to generate continuous actions. Consequently, the Actor generates fuzzy membership degrees, and the defuzzification process maps these degrees into continuous actions. In this way, our method leverages low-dimensional discrete DRL to ensure task completion. We conduct comprehensive experiments to evaluate the performance of our method. Compared with other DRL-based methods, the results show that our method can generate collision-free and smooth motion trajectories in real time and improve motion planning performance significantly.**

I. Introduction

As computing power and artificial intelligence techniques progress swiftly, neural network-based approaches, particularly deep reinforcement learning (DRL), have found extensive application in robot path planning, aiming to improve computational efficiency [1], [2], [3], [4], [5]. However, current DRL methods mainly focus on the discrete action space, which will limit the motion capability of mobile robots, resulting in long motion distance or motion time (experimental results will be given in the experiments). There is little work on DRL methods in continuous action spaces.

To bridge this gap, this paper aims to design a DRL-based motion planning method for mobile robots in a continuous action space. However, applying the current continuous-action DRL methods, such as DDPG (Deep Deterministic Policy Gradient), A3C (Asynchronous Advantage Actor-Critic), and SAC (Soft Actor Critic), to the motion planning of mobile robots suffers from the local optima challenge. DRL is suitable for simulating the behavior of a complex system, such as multi-DOF (degree of freedom) manipulators, in relatively simple environments because the simple environment reduces the complexity of the learning task. However, the complexity of motion planning for mobile robots lies in the dynamic and unpredictable environments with varying numbers of obstacles, which makes the task significantly harder. When continuous actions are considered, it is difficult to design a reward function that provides meaningful positive feedback for reaching the destination. Safety- and task-related rewards may not sufficiently guide the robot due to the infinite actions that could lead to positive feedback without necessarily leading to the destination. Consequently, the problem of local optima arises because many actions might seem beneficial (positive feedback) but do not lead to the ultimate goal, causing the robot to get stuck in suboptimal states and fail to reach the destination.

To mitigate this challenge, we introduce a fuzzy DRL approach, `Fuzzy-DDPG`, to the motion planning of mobile robots. `Fuzzy-DDPG` leverages the task-completion capacity of low-dimensional discrete DRL and the continuous-discrete transformation mechanism of fuzzy logic. Therefore, `Fuzzy-DDPG` can ensure that the robot arrives at the destination successfully and generates smoother motion. First, the idea of fuzzification and defuzzification processes in fuzzy inference systems is applied to the motion command, i.e., the velocity of the robot. On one hand, the velocity is fuzzified into velocity fuzzy sets. This means the continuous velocity is discretized into a set of fuzzy sets, quantified by their membership degrees. The DRL model can then focus on the computation of membership degrees. On the other hand, based on these membership degrees, we can compute the continuous velocity via defuzzification. Second, we apply the DDPG framework to train a DRL model to compute the membership degrees at each time instant. The DRL model in `Fuzzy-DDPG` consists of two DNNs: an Actor model, a policy network that computes the membership degrees of the velocity fuzzy sets based on the current system state,

* Corresponding author.

This work was supported in part by the National Research Foundation, Singapore, and DSO National Laboratories under the AI Singapore Programme (AISG Award No: AISG2-GC-2023-008), the NRF Investigatorship NRF-NRFI06-2020-0001, Academic Research Fund Tier 2 by Ministry of Education in Singapore under Grant No. MOE-T2EP20120-0004, the Natural Science Foundation of China under Grant Nos. 61972150 and 62132014, Zhejiang Provincial Key Research and Development Program of China under Grant 2022C01045, and Science Foundation of Zhejiang Sci-Tech University (ZSTU) Under Grant No. 24232204-Y.

[1] F. Wu, W. Tang, S.-W. Lin, and Y. Liu are with the College of Computing and Data Science, Nanyang Technological University, Singapore 639798 {fenghua.wu, wenbing.tang, shang-wei.lin, yangliu}@ntu.edu.sg

[2] Y. Zhou and Z. Ding are with the School of Information Science and Technology, Zhejiang Sci-Tech University, Hangzhou, Zhejiang 310018, P. R. China yuanzhou@zstu.edu.cn, zouhuading@hotmail.com

i.e., the concatenation of the robot's and obstacles' states, and an Critic model, a Q-value network that computes the Q value of a state-action pair, utilizing the current system state and membership degrees from the Actor. To deal with a varying number of obstacles, both Actor and Critic consist of an LSTM (Long Short Term Memory) model and an MLP (Multi-Layer Perceptron) model. To train the DRL model, we first apply ORCA (Optimal Reciprocal Collision Avoidance) [6] to generate a set of trajectories. For each state-action pair in a trajectory, we future generate the corresponding one-step reward and the membership degrees with respect to the velocity fuzzy sets for the training. When the DRL model is deployed for real-time motion planning, the Actor outputs the membership degrees based on the current robot state and the environment state, and the defuzzification process finally computes the continuous velocity.

To assess the effectiveness of the proposed method, we obtained two models trained in two scenarios: one with a fixed number of obstacles and another with a varying obstacle count. We then tested the two well-trained models in three different environments containing different numbers of obstacles. To gauge efficiency, we benchmarked our approach against three state-of-the-art DRL methods. Our findings demonstrate that our method outperforms existing solutions, achieving (1) the highest success rate, (2) the lowest computation and navigation time, and (3) the shortest path lengths. In summary, the key contributions are as follows:

- Fuzzy logic is applied to handle continuous action spaces in the DRL-based motion planning of mobile robots, ensuring the achievement of motion tasks and real-time computation efficiency.
- A new fuzzy continuous DRL method, `Fuzzy-DDPG`, is proposed for motion planning of mobile robots operating in environments with varying numbers of obstacles.
- Extensive testing is carried out to thoroughly verify both the effectivenss and performance of `Fuzzy-DDPG`.

II. Related Work

DRL has emerged as a popular end-to-end approach with the capability to map sensor data to actions and has shown promising results in robot motion planning. According to their inputs, current DRL motion planning methods can be categorized into two main groups: sensor-based DRL methods [7], [8] and agent-based DRL methods [1], [2], [3]. The former approaches use raw sensor data as direct input and compute the motion commands. The latter ones rely on preprocessed agent-level information as input, such as the agent's position and velocity. However, these methods are proposed for discrete action spaces, limited to discrete actions. Hence, it may exist the action dependency problem [4] and trajectory oscillation problem [9], which is not appropriate for the complex applications of today's robots.

To address continuous control tasks, some DRL methods have been proposed in recent years, such as DDPG [10], and Proximal Policy Optimization (PPO) [11]. Some researchers have attempted to apply these methods to robotics. For example, a DDPG method is applied to control a 7-DoF manipulator [12]. Liu and Huang [13] developed a DDPG-based approach to regulate aerial robotic arms. Ye *et al.* [14] proposed a PPO-based method to learn an automated lane change strategy. They usually focus on simple motion tasks. However, utilizing continuous DRL methods to generate motion commands for mobile robots directly is still a significant challenge in complex motion planning tasks. Therefore, we propose to use fuzzy DDPG to generate membership degrees, rather than concrete motion actions.

Since it does not require the precise mathematical model of the controlled process, fuzzy logic has been widely applied in many robot control tasks, such as collision avoidance [15], object tracking [16], and motion planning [17]. In these applications, there are usually three main steps: fuzzification, inference, and defuzzification. A key challenge is to design an effective inference system to generate proper membership degrees and feed them into the defuzzification module. To overcome this challenge, we proposed to learn a mapping from states to membership degrees using a policy network in an end-to-end manner.

III. Problem Statement

This paper focuses on navigating a holonomic mobile robot through an unknown environment. The robot must navigate to a designated endpoint, $\mathbf{p}_g = [x_g, y_g]$, from its starting point, denoted as $\mathbf{p}_i = [x_i, y_i]$. The challenge lies in maneuvering around a varying number of obstacles, including both pedestrians and other robots. Let ρ be the robot's safety radius. The motion is discretized over time with a discrete step size of Δt. For each discrete moment t, where t is a non-negative integer, the robot determines a collision-free velocity, which is then implemented over the subsequent time, beginning at $t\Delta t$ and ending just before $(t+1)\Delta t$. At any given time t, we can represent the robot's state and action as follows: $\mathbf{se}_t = [\mathbf{p}_t, \mathbf{v}_t^-, \mathbf{p}_g, v_f, \rho] \in \mathbb{R}^8$ and $\mathbf{v}_t$, where $\mathbf{p}_t = [x_t, y_t]$ represents the robot's current position; $\mathbf{v}_t^-$ is the robot's velocity during the preceding time interval $[(t-1)\Delta t, t\Delta t)$; $\mathbf{p}_g$ is the target position; v_f denotes the robot's preferred speed; ρ is the safety radius (as previously mentioned); $\mathbf{v}_t$ is the velocity that the motion planner needs to determine at time t for the upcoming interval. It means that the action space can be defined as $\mathbb{A} = [v^x_{\min}, v^x_{\max}] \times [v^y_{\min}, v^y_{\max}]$, where $v^x_{\min}$, $v^y_{\min}$, $v^x_{\max}$, and $v^y_{\max}$ are determined when the robot was produced. At time t, consider that the robot detects a set of obstacles, denoted as $o^1_t, \ldots, o^n_t$. The states of these obstacles are collectively represented as $\mathbf{so}_t = \mathbf{so}^1_t, \cdots, \mathbf{so}^n_t$. For each individual obstacle o^j_t, its state is defined as $\mathbf{so}^j_t = [\mathbf{p}^j_t, \mathbf{v}^j_t, \rho^j] \in \mathbb{R}^5$, where $\mathbf{p}^j_t = [x^j_t, y^j_t]$, $\mathbf{v}^j_t = [vx^j_t, vy^j_t]$, and ρ^j are the position, velocity, and radius of o^j_t, respectively. Hence, the system state space can be represented as $\mathbb{S} \triangleq \{\mathbf{s}_t\} = \{[\mathbf{se}_t, \mathbf{so}_t]\}$. The motion planning task is to determine a policy $\pi : \mathbb{S} \mapsto \mathbb{A}, \forall \mathbf{s}_t \in \mathbb{S}, \pi(\mathbf{s}_t) = \mathbf{v}_t \in \mathbb{A}$, which will assign each state a collision-free action, enabling the robot to reach its destination via the most time-efficient trajectory while completely avoiding obstacles. Formally, it can be described as:

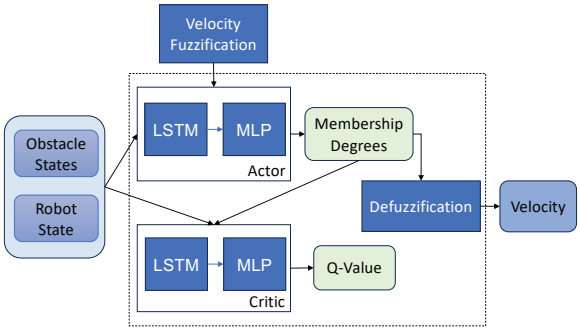


Fig. 1. The general architecture of Fuzzy-DDPG.

$$\arg\min_{\pi} \ T \tag{1}$$

$$s.t. \quad \mathbf{p}_0 = \mathbf{p}_i, \ \mathbf{p}_T = \mathbf{p}_g, \tag{2}$$

$$\|\mathbf{p}_t - \mathbf{p}_t^j\| \geq \rho + \rho^j, j \in \{0, 1, \cdots\}, \forall t, \tag{3}$$

$$\mathbf{p}_{t+1} = \mathbf{p}_t + \mathbf{v}_t \Delta t, \forall t \in \{0, 1, \cdots, T-1\}, \tag{4}$$

where (1) serves as the optimization objective, aiming to minimize the motion time T; (2) represents the task constraint; (3) defines safety constraints to avoid collisions; (4) describes the robot's kinematics. Hence, Fuzzy-DDPG aims to determine the continuous values for $\mathbf{v}_t$ at each time instant.

IV. Fuzzy Motion Planning Method

Fig. 1 shows the architecture overview of Fuzzy-DDPG. Fuzzy-DDPG starts with the fuzzification of the velocity of the robot. Then, it trains a control policy by iteratively training two deep neural networks: an Actor and a Critic. Finally, taking the robot's state and those of the nearby obstacles as inputs, Fuzzy-DDPG computes the membership degrees via the well-trained Actor and generates the continuous velocity via defuzzification.

A. Architecture of Fuzzy-DDPG

1) Velocity Fuzzification: Fuzzy-DDPG embeds fuzzy logic into conventional DDPG framework to mitigate the convergence issue in continuous DRL methods and the limited actions in discrete DRL methods.

As described in our problem statement, the control command is the robot's velocity $\mathbf{v}_t = (v_t^x, v_t^y)$ at each time instant. First, we design fuzzy sets to model the velocity components v^x and v^y, respectively. Since v^x and v^y can be positive or negative, where the positive and negative signs indicate the direction, either velocity component is described by five fuzzy sets: NL (negative large), NS (negative small), M (medium), PS (positive small), and PL (positive large). NL and NS indicate that the robot takes a large and a small negative value on the corresponding axis (x-axis for v^x and y-axis for v^y), respectively. PL and PS denote that the robot takes a large and a small positive value on the corresponding axis (x-axis for v^x and y-axis for v^y), respectively. M represents the robot moving slightly. Note that we can design more fuzzy sets to model velocity components, however, it may cause a curse of dimensionality, resulting in a sparsity of useful experiences and making it challenging to generalize learning across states.

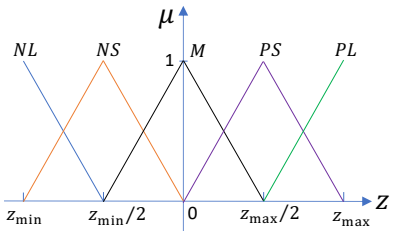


Fig. 2. The membership functions of the five fuzzy sets for the velocity. Note that $z_{\min} = v_{\min}^x$ (resp., $v_{\min}^y$) and $z_{\max} = v_{\max}^x$ (resp., $v_{\max}^y$) for v^x (resp., v^y).

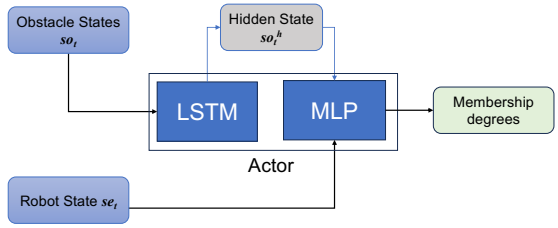
In fuzzy logic, a fuzzy set can be defined by a unique membership function. Given an universe of discourse $Z = [z_{\min}, z_{\max}]$, a particular value z in Z and a fuzzy set Φ, the corresponding membership function, denoted as $\mu_\Phi : Z \to [0, 1]$, maps each value $z \in Z$ to the interval $[0, 1]$, where $\mu_\Phi(z) \in [0, 1]$ quantifies the grade of membership of z to the fuzzy set Φ. Usually, users can select different membership functions, such as Triangular, Trapezoidal, and Gaussian functions, based on their experience and the application domains. In this paper, we employ the triangular membership functions for the fuzzy sets, as their performance has been demonstrated in numerous robotic applications [17], [18]. Recall that $v_{\min}^x < 0$ and $v_{\min}^y < 0$. Therefore, the membership functions of the five fuzzy sets are given in Fig. 2.

Based on the membership functions, given an action $a_t = (v_t^x, v_t^y)$, we can compute the membership degree related to each fuzzy set, i.e., $\mu(v_t^x) = [\mu_{NL}(v_t^x), \mu_{NS}(v_t^x), \mu_M(v_t^x), \mu_{PS}(v_t^x), \mu_{PL}(v_t^x)]$, $\mu(v_t^y) = [\mu_{NL}(v_t^y), \mu_{NS}(v_t^y), \mu_M(v_t^y), \mu_{PS}(v_t^y), \mu_{PL}(v_t^y)]$, and $\mu(a_t) = [\mu(v_t^x), \mu(v_t^y)]$.

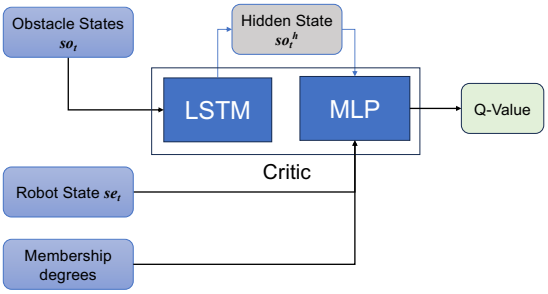
2) Actor and Critic: Following the standard DDPG framework, there are two networks in Fuzzy-DDPG: the Actor A_θ and the Critic Q_ϕ. As shown in Fig. 3, either model consists of an LSTM network and an MLP network. The Actor A_θ takes the system state $\mathbf{s}_t = [\mathbf{se}_t, \mathbf{so}_t]$ as input, and computes the membership degrees. Based on the fuzzification, the Actor should generate 10 kinds of membership degrees, 5 for v^x and 5 for v^y. Therefore, the output layer of the MLP in A_θ is a Softmax layer with 10 neurons. Given the state-action pair $(\mathbf{s}_t, a_t)$, the Critic Q_ϕ takes the system state $\mathbf{s}_t$ and the corresponding membership degrees as input and computes the Q-value for $(\mathbf{s}_t, a_t)$. Note that the membership degrees are generated from the fuzzification process when the velocity a_t is given, otherwise from the Actor $A_\theta(\mathbf{s}_t)$.

3) Velocity Generation via Defuzzification: Like other fuzzy inference systems, defuzzification is required to generate crisp velocities from the membership degrees computed by the Actor A_θ. Note that the output neurons of the Actor are divided into two groups, each of which contains five neurons, representing the membership degrees of the five fuzzy sets for v^x and v^y, respectively. Therefore, the defuzzification process includes the generation for v^x from the first five neurons and v^y from the second five neurons.

There are various defuzzification strategies, such as the Center of gravity (COG) Method, Center of Sums Method (COS), Bisector of Area Method (BOA), Weighted Average Method, Mean of Maximum (MOM), and Largest of Maximum (LOM). In this paper, the COG method is selected



(a) Actor.



(b) Critic.

Fig. 3. Architectures of Actor and Critic in `Fuzzy-DDPG`. Either model contains an LSTM to handle the varying dimensions of obstacle states.

because it is successfully used in robot navigation [17], [19], [20], [21], [22]. The COG method computes the center of gravity of the area under the given membership degree. It can be calculated by $z^* = \int \mu(z) z dz / \int \mu(z) dz$, where z denotes any velocity component (i.e., v_t^x or v_t^y), and $\mu(z)$ is the membership degrees generated by the Actor.

B. Training of `Fuzzy-DDPG`

At each time instant t, the robot interacts with the environment and stores state transition $(\mathbf{s}_t, \mathbf{v}_t, r_t, \mathbf{s}_{t+1})$ into the memory buffer, where the reward $r : \mathbb{S} \times \mathbb{A} \to \mathbb{R}$ is defined to guide the training process. Following [2], [23], the reward function is defined as follows:

$$r(\mathbf{s}_t, \mathbf{v}_t) = \begin{cases} 1 & \text{if } \mathbf{p}_{t+1} = \mathbf{p}_g, \\ -0.25 & \text{if } d_{min} \le 0, \\ -0.1 + 0.5 d_{min} & \text{if } 0 < d_{min} \le 0.2, \\ 0 & \text{otherwise.} \end{cases}$$

where d_{min} is the minimal distance between the robot and all obstacles during the next time duration $[t\Delta t, (t+1)\Delta t)$.

`Fuzzy-DDPG` trains the Actor and the Critic iteratively by randomly sampling a minibatch of m transitions from the replay memory buffer. Specifically, `Fuzzy-DDPG` optimizes the Critic by minimizing the following loss:

$$\mathcal{L}(w_\phi) = \frac{1}{m} \sum_t [q_t - Q_\phi(\mathbf{s}_t, \mu(\mathbf{v}_t); w_\phi)]^2, \quad (5)$$

$$q_t = r_t + \gamma Q_\phi^*(\mathbf{s}_{t+1}, A_\theta^*(\mathbf{s}_{t+1}; w_\theta^*); w_\phi^*), \quad (6)$$

where w_ϕ are the weights to be trained in Q_ϕ, $\{(\mathbf{s}_t, \mathbf{v}_t, r_t, \mathbf{s}_{t+1})\}$ is the training data set, m is the batch size, $\mu(\mathbf{v}_t)$ is the fuzzification of $\mathbf{v}_t$ based on the fuzzy sets, $\gamma \in (0, 1]$ is the discount factor, and $A_\theta^*(\cdot; w_\theta^*)$ and $Q_\phi^*(\cdot; w_\phi^*)$ are the target Actor and Critic updated periodically.

The Actor is updated to maximize the expected Q-values by using sampled policy gradient [24]. Hence, the loss function of the Actor to be minimized is:

$$\mathcal{L}(w_\theta) = -\frac{1}{m} \sum_t Q_\phi(\mathbf{s}_t, A_\theta(\mathbf{s}_t; w_\theta); w_\phi), \quad (7)$$

where w_θ are the weights to be trained, $Q_\phi(\cdot; w_\phi)$ is trained early in the same episode.

The target networks will be updated every ω episodes based on the following equations:

$$w_\theta^* \leftarrow \tau w_\theta + (1 - \tau) w_\theta^*, \quad (8)$$

$$w_\phi^* \leftarrow \tau w_\phi + (1 - \tau) w_\phi^*, \quad (9)$$

where $\tau \in (0, 1]$ is the parameter that controls the rate of the soft update, and w_θ and w_ϕ are the weights of the current Actor and Critic networks.

C. Planning with `Fuzzy-DDPG`

Once the training process is completed, a robot will use the trained `Fuzzy-DDPG` for motion generation. Fig. 4 shows the planning process of `Fuzzy-DDPG`. First, a robot identifies its own state $\mathbf{se}_t$, and then detects its surrounding obstacles' states $\mathbf{so}_t$. $\mathbf{so}_t$ are first sorted according to their collision risks in an ascending order and then fed to the LSTM component in the Actor to compute the hidden state $\mathbf{so}_t^h$. Second, the MLP component of the Actor takes $\mathbf{so}_t^h$ and $\mathbf{se}_t$ as the input and computes the membership degrees. Third, the membership degrees are fed to the defuzzification module, generating an action $\mathbf{v}_t$. Finally, the robot executes the new action within the next time duration. Repeating the above steps iteratively in a closed-loop mode in interaction with the environment, a robot navigates to its designated destination, successfully avoiding collisions with obstacles encountered along the way.

V. Experimental Evaluation

A. Evaluation Setup

In this section, we perform a series of simulation experiments to assess the capabilities of `Fuzzy-DDPG`. Our experiments are carried out within the simulation environment developed by [3]. All models are implemented using PyTorch, and both training and testing are conducted on an Nvidia RTX A4000 GPU. For the training of `Fuzzy-DDPG`, we set the training episodes η to 30,000, and in each episode, the maximum number of time steps T_m is 100 with $\Delta t = 0.25$. The network architectures of the Actor (target Actor) and Critic (target Critic) are (56, 150, 100, 10) and (56, 150, 100, 1), respectively. They are trained in an end-to-end way with a learning rate of 0.001. The value of the minibatch is set to 100 and the size of the replay memory is 100,000. Additionally, the target networks are updated every 50 steps, $\gamma = 0.9$, and $\tau = 0.0001$. Each robot has a preferred speed $v_f = 1$ and a safe radius $\rho = 0.3$. In each replay, the initial positions of the obstacles are randomly generated, and they need to move to the position opposite their starting points relative to the origin. For the defuzzification module, the ranges of v^x and v^y are $[-1, 1]$. The hidden state dimension of each LSTM is set to 50.

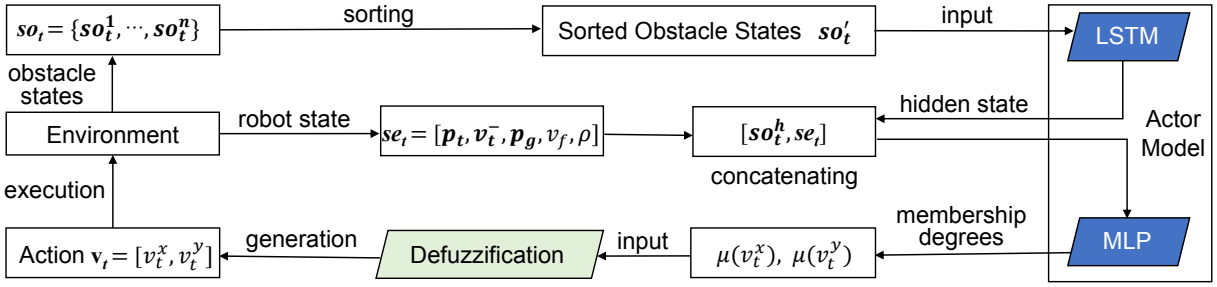


Fig. 4. Planning process with Fuzzy-DDPG.

TABLE I
MOTION PLANNING PERFORMACE OF Fuzzy-DDPG UNDER STATIC AND DYNAMIC TRAINING SCENARIOS.

Training	Testing Obstacles	Success Rate	Collision Rate	Timeout Rate	Decision Time	Navigation Time	Length
Static	1-4	1.000	0.000	0.000	0.001	9.423	7.889
	5	1.000	0.000	0.000	0.001	9.250	7.706
	6-9	0.944	0.058	0.000	0.001	9.363	7.755
	Average	0.981	0.019	0.000	0.001	9.345	7.783
Dynamic	1-4	1.000	0.000	0.000	0.001	9.404	7.768
	5	1.000	0.000	0.000	0.001	9.250	7.708
	6-9	0.971	0.029	0.000	0.001	9.342	7.746
	Average	0.990	0.010	0.000	0.001	9.332	7.741

B. Performance of Fuzzy-DDPG

To evaluate its effectiveness, Fuzzy-DDPG is trained under two environments: *static*, where the model is trained in the environment with five obstacles, and *dynamic*, where the model is trained in the environment with a varying number of obstacles, from 1 to 10. During the testing phase, we randomly generated 10 test sets, each containing three scenario groups based on the number of obstacles in the test cases: 1-4 obstacles, 5 obstacles, and 6-9 obstacles. Each group in each set contains 200 test cases, resulting in 600 test cases in each test set. The metrics used to evaluate the experiments include *success* rate, *collision* rate, *timeout* rate, and the successful executions' average decision time (*decision time*), average navigation time (*navigation time*), and average trajectory length (*length*).

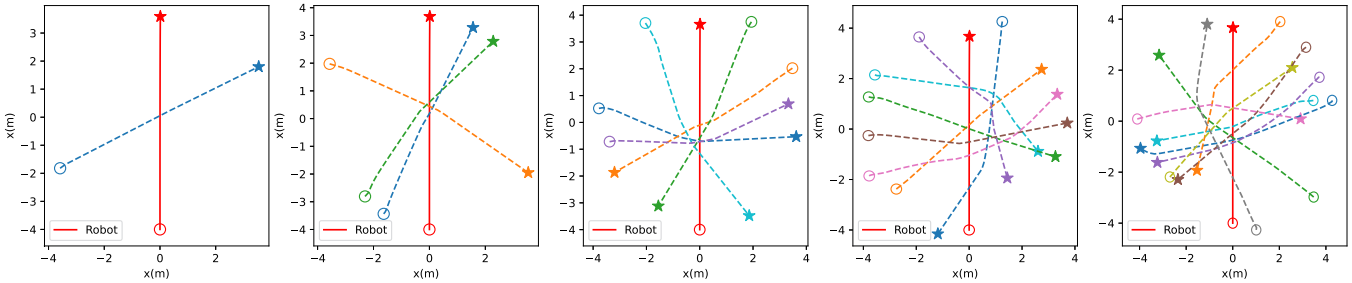
Table I shows the motion performance of Fuzzy-DDPG in different situations. From the tables, we can find that Fuzzy-DDPG performs well in all scenarios. Under static training, Fuzzy-DDPG achieves a high *success* rate and a low *collision* rate (98.1% and 1.9% on average). Moreover, Fuzzy-DDPG shows a short decision time (0.001s on average), indicating that the robot can generate actions in real time. Furthermore, Fuzzy-DDPG can generate good-quality trajectories with short path lengths and reduced navigation time (7.783 and 9.345s on average, respectively). For dynamic training, Fuzzy-DDPG shows a higher *success* rate (99% vs. 98.1%) and a lower *collision* rate (1% vs. 1.9%). The reason is that the model is trained with a varying number of obstacles and thus shows better performance in crowded dynamic environments, e.g., 6-9 obstacles. For other metrics, we can find that Fuzzy-DDPG shows similar results under two training scenarios. Therefore, Fuzzy-DDPG is suitable for different environments.

By comparing the results from the two scenarios, we can find that a Fuzzy-DDPG trained with five obstacles can generalize well beyond the training environment, e.g., 1-4 obstacles and 6-9 obstacles. Consequently, the proposed fuzzy method can achieve good motion planning performance and guarantee generalization ability, benefiting from the capability of fuzzy logic in dealing with uncertain and imprecise knowledge [25], [26]. In addition, with the increase of obstacles, Fuzzy-DDPG maintain a stable performance in all metrics (e.g., a *decision* time of 0.001s for all testing scenarios), without significant decreases. This result means that Fuzzy-DDPG is universal and can be applied to sparse and crowded environments. The reason is that by leveraging the criticality-guided LSTM model, we can retrieve and filter more dangerous obstacles while avoiding consuming time and computation costs on non-urgent ones.

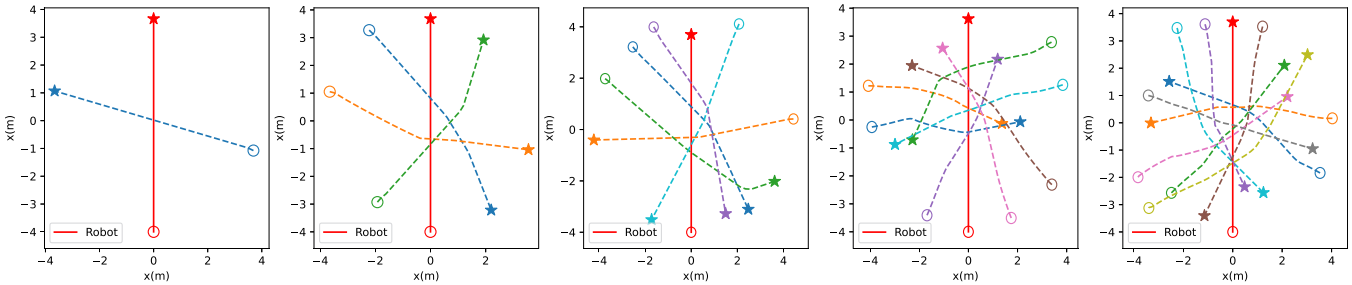
In the sequel, to further demonstrate the effectiveness of Fuzzy-DDPG, we show some example trajectories generated by our method in different situations. Fig. 5 shows the travelled trajectories of the robot and obstacles. From Fig. 5(a), we can find that Fuzzy-DDPG performs well in all test cases. In addition, we can find that Fuzzy-DDPG can generate smoother trajectories for the robot, compared with the trajectories of obstacles controlled by ORCA. The reason is that the proposed method can determine an action from continuous action spaces as long as it is feasible by the physical constraints of the robot. Hence, Fuzzy-DDPG can find a smoother trajectory to move to the target position. In Fig. 5(b), we can observe that Fuzzy-DDPG can also control the robot reaching its target position. In addition, with the increase of obstacles in the environment, ORCA shows relatively tortuous trajectories, while our method can still generate smooth trajectories. Note that in these examples, the robot moves along similar paths, however, their trajectories are different, i.e., the moving velocities along these paths vary with surrounding obstacles.

C. Comparisons with Other DRL methods

In this section, we further compare the performance of Fuzzy-DDPG with other state-of-the-art DRL methods. We choose three three prominent and well-regarded DRL methods: LSTM-RL [3], Crit-LSTM-DRL [2], and DDPG [27]. In detail, by sorting the obstacles' states according to their distances from the robot, LSTM-RL employs an LSTM module to convert the varying-dimensional obstacle states into a fixed-size hidden state. Crit-LSTM-DRL, on the other hand, introduces a novel sorting strategy by focusing on the criticality of obstacles to the robot. LSTM-RL and Crit-



(a) Trajectories generated by Fuzzy-DDPG with static training. Left to Right: Test cases containing 1, 3, 5, 7, and 9 obstacles, respectively.



(b) Trajectories generated by Fuzzy-DDPG with dynamic training. Left to Right: Test cases containing 1, 3, 5, 7, and 9 obstacles, respectively.

Fig. 5. Trajectories generated by the models trained under static and dynamic training environments. The robot begins at $[-4, 0]$ and aims to reach $[4, 0]$. The robot's trajectories are shown by solid lines. Dashed lines represent the obstacles' trajectories. All obstacles are controlled by ORCA. Circles mark the starting positions for both the robot and the obstacles. Stars indicate the destinations for the robot and the obstacles.

TABLE II

PERFORMANCE COMPARISON BETWEEN FUZZY-DDPG AND OTHER DRL METHODS OVER TEN TEST SETS.

Training	Method	Success Rate		Collision Rate		Timeout Rate		Decision Time		Navigation Time		Length	
		Mean	Variance	Mean	Variance	Mean	Variance	Mean	Variance	Mean	Variance	Mean	Variance
Static	Fuzzy-DDPG	**0.981**	1.20E-04	0.019	1.23E-04	**0.000**	0.00E+00	**0.001**	2.59E-10	**9.423**	5.29E-04	**7.783**	1.37E-02
	LSTM-RL	0.865	3.02E-04	**0.002**	1.85E-06	0.133	3.11E-04	0.053	6.75E-08	10.814	2.99E-02	10.442	9.02E-03
	Crit-LSTM-DRL	0.979	3.04E-04	0.008	1.79E-05	0.013	2.82E-04	0.064	1.41E-04	12.952	1.53E-01	11.408	8.24E-02
	DDPG	0.562	7.24E-04	0.089	1.60E-02	0.349	1.72E-02	**0.001**	3.97E-07	13.551	5.27E-01	9.104	1.09E-01
Dynamic	Fuzzy-DDPG	0.990	1.17E-04	0.010	1.17E-04	**0.000**	0.00E+00	**0.001**	3.96E-09	**9.332**	1.03E-03	**7.741**	1.62E-04
	LSTM-RL	0.976	1.13E-04	0.023	9.53E-05	0.001	1.78E-06	0.056	5.87E-05	12.186	1.89E-03	11.226	6.57E-02
	Crit-LSTM-DRL	**0.994**	2.57E-05	**0.003**	3.44E-06	0.003	2.33E-05	0.068	2.24E-05	10.838	1.20E-02	9.285	4.53E-02
	DDPG	0.510	4.20E-04	0.095	1.41E-02	0.395	1.86E-02	**0.001**	1.54E-07	10.383	1.53E-01	7.811	4.95E-05

LSTM-DRL select an action from a discrete action space via learning a value network. DDPG generates continuous actions directly from a well-trained Actor; for fair comparisons, we apply the sorting strategy and the LSTM module from Crit-LSTM-DRL to handle changes in the number of obstacles; then, the hidden state of LSTM is fed to the Actor. All methods are trained for 30,000 episodes under consistent learning settings. Table II shows the mean and variance of all metrics for different methods, calculated across 10 test datasets. From Table II, we have the following findings.

- **Fuzzy-DDPG can improve motion planning performance significantly.** Compared with the two discrete methods LSTM-RL and Crit-LSTM-DRL, Fuzzy-DDPG achieves the highest success rate in the static training environment (98.1% vs. 86.5% and 97.9%). In the dynamic training environment, Fuzzy-DDPG achieves a similar success rate with Crit-LSTM-DRL (99.0% vs. 99.4%), but high-quality trajectories (lower navigation time (9.332 vs. 10.838) and shorter path length (7.741 vs. 9.285)). Compared with DDPG, Fuzzy-DDPG also shows a significant margin of improvement in success rate (98.1% vs. 56.2% for the static training situation and 99.0% vs. 51.0% for the dynamic training situation). The findings indicate that it will typically result in the robot either colliding with obstacles or failing to reach its designated target position when controlled by DDPG. They highlight the significant potential of fuzzy logic in motion planning, which has been given insufficient attention in many prior DRL methods.
- **Fuzzy-DDPG can reduce decision-making time consumption.** Among all methods, we can observe that Fuzzy-DDPG and DDPG have the minimal *decision* time. The decision time of LSTM-DRL and Crit-LSTM-RL is 0.053s and 0.064s, respectively, while Fuzzy-DDPG generate an action only takes about 0.001s. The reason is that Fuzzy-DDPG and DDPG determine an action using the well-trained Actor, rather than selecting the action with maximum value from a discrete action space. However, LSTM-RL and Crit-LSTM-RL need to traverse the entire action space to find the action with the maximum value, which is time-consuming. In addition, we can find that Fuzzy-DDPG and DDPG have similar *decision* time (both around 0.001s). The result indicates that the defuzzification process has minimal impact on the system's ability to perform real-time planning efficiently.

- **`Fuzzy-DDPG` can control the robot to reach its target position faster.** For the navigation time, we find that `Fuzzy-DDPG` has the lowest motion time (about 9.423s under the static training situation and 9.332s under the dynamic situation), which is lower than the time consumed by the other three methods. The reason is that by expanding the candidate actions from a finite set to a continuous space, the proposed method can compute more suitable actions to control the robot's movement toward its target position faster.
- **`Fuzzy-DDPG` can generate shorter trajectories.** From the table, we can find that `Fuzzy-DDPG` generates the shortest trajectories. Under the static (resp., dynamic) training situation, the average trajectory length generated by `Fuzzy-DDPG` is about 7.783 (resp., 7.741), while the shortest of the other three methods is about 9.104 (resp., 7.811), both generated by DDPG. Consequently, we can conclude that `Fuzzy-DDPG` achieves a better motion performance than other DRL-based methods.

Finally, for variances in Table II, we can find that `Fuzzy-DDPG` has the minimum variances on average. The comparison results indicate that `Fuzzy-DDPG` can maintain a more stable performance across different environments. In conclusion, the proposed method can generate collision-free and smooth motion trajectories in real time to navigate a robot moving to its target position successfully.

VI. Conclusion

In this paper, we proposed a new continuous DRL motion planning method `Fuzzy-DDPG`, integrating fuzzy logic into the DDPG framework, to generate real-time and continuous actions for a robot moving in dynamic environments. First, the velocity is fuzzified based on pre-defined fuzzy sets. A novel architecture, consisting of an LSTM and an MLP, is proposed for Actor and Critic, which can deal with a varying number of obstacles in the environment. When `Fuzzy-DDPG` is well-trained, we apply its Actor to generate fuzzy membership degrees. Finally, a defuzzification module is proposed to determine continuous actions according to the membership degrees. Our assessment of `Fuzzy-DDPG` encompassed diverse scenarios featuring varying obstacle quantities. The findings demonstrate `Fuzzy-DDPG`'s capability to generate safe and efficient motion plans in real-time across these different conditions.

References

[1] Y. F. Chen, M. Liu, M. Everett, and J. P. How, "Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2017, pp. 285–292.

[2] L. Xu, F. Wu, Y. Zhou, H. Hu, Z. Ding, and Y. Liu, "Criticality-guided deep reinforcement learning for motion planning," in *2021 China Automation Congress (CAC)*, 2021, pp. 3378–3383.

[3] C. Chen, Y. Liu, S. Kreiss, and A. Alahi, "Crowd-robot interaction: Crowd-aware robot navigation with attention-based deep reinforcement learning," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2019, pp. 6015–6022.

[4] L. Chen, X. Hu, B. Tang, and Y. Cheng, "Conditional dqn-based motion planning with fuzzy logic for autonomous driving," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 4, pp. 2966–2977, 2020.

[5] M. Everett, Y. F. Chen, and J. P. How, "Motion planning among dynamic, decision-making agents with deep reinforcement learning," in *IEEE/RSJ Int.l Conf. Intell. Robot. Syst.*, 2018, pp. 3052–3059.

[6] J. Van Den Berg, S. J. Guy, M. Lin, and D. Manocha, "Reciprocal n-body collision avoidance," in *Robot. Res.* Springer, 2011, pp. 3–19.

[7] P. Long, T. Fan, X. Liao, W. Liu, H. Zhang, and J. Pan, "Towards optimally decentralized multi-robot collision avoidance via deep reinforcement learning," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2018, pp. 6252–6259.

[8] K. Wu, H. Wang, M. A. Esfahani, and S. Yuan, "Learn to navigate autonomously through deep reinforcement learning," *IEEE Transactions on Industrial Electronics*, vol. 69, no. 5, pp. 5342–5352, 2021.

[9] J. Yu, H. Piao, Y. Hou, L. Mo, X. Yang, and D. Zhou, "Doma: Deep smooth trajectory generation learning for real-time uav motion planning," in *Proceedings of the International Conference on Automated Planning and Scheduling*, vol. 32, 2022, pp. 662–666.

[10] T. Lilicrap, J. Hunt, A. Pritzel, N. Hess, T. Erez, D. Silver, Y. Tassa, and D. Wiestra, "Continuous control with deep reinforcement learning," in *Int. Conf. Representation Learn.*, 2016.

[11] Y. Gu, Y. Cheng, C. P. Chen, and X. Wang, "Proximal policy optimization with policy feedback," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 52, no. 7, pp. 4600–4610, 2021.

[12] S. R. Afzali, M. Shoaran, and G. Karimian, "A modified convergence ddpg algorithm for robotic manipulation," *Neural Processing Letters*, vol. 55, no. 8, pp. 11 637–11 652, 2023.

[13] Y.-C. Liu and C.-Y. Huang, "Ddpg-based adaptive robust tracking control for aerial manipulators with decoupling approach," *IEEE Transactions on Cybernetics*, vol. 52, no. 8, pp. 8258–8271, 2021.

[14] F. Ye, X. Cheng, P. Wang, C.-Y. Chan, and J. Zhang, "Automated lane change strategy using proximal policy optimization-based deep reinforcement learning," in *IEEE Intell. Veh. Symposium*, 2020, pp. 1746–1752.

[15] M. Faisal, R. Hedjar, M. Al Sulaiman, and K. Al-Mutib, "Fuzzy logic navigation and obstacle avoidance by a mobile robot in an unknown dynamic environment," *International Journal of Advanced Robotic Systems*, vol. 10, no. 1, p. 37, 2013.

[16] S. Liu, S. Huang, X. Xu, J. Lloret, and K. Muhammad, "Efficient visual tracking based on fuzzy inference for intelligent transportation systems," *IEEE Trans. Intell. Transp. Syst.*, 2023.

[17] W. Tang, Y. Zhou, T. Zhang, Y. Liu, J. Liu, and Z. Ding, "Cooperative collision avoidance in multirobot systems using fuzzy rules and velocity obstacles," *Robotica*, vol. 41, no. 2, pp. 668–689, 2023.

[18] V. Kreinovich, O. Kosheleva, and S. N. Shahbazova, "Why triangular and trapezoid membership functions: A simple explanation," *Recent Developments in Fuzzy Logic and Fuzzy Sets: Dedicated to Lotfi A. Zadeh*, pp. 25–31, 2020.

[19] A. Zhu and S. X. Yang, "Neurofuzzy-based approach to mobile robot navigation in unknown environments," *IEEE Trans. Syst., Man, Cybern. C*, vol. 37, no. 4, pp. 610–621, 2007.

[20] S. Harisha, P. Kumar, M. Krishna, and S. Sharma, "Fuzzy logic reasoning to control mobile robot on pre-defined strip path," *World Academy of Sci. Eng. Technol.*, vol. 42, pp. 642–646, 2008.

[21] Y. Liang, L. Xu, R. Wei, and H. Hu, "Adaptive fuzzy control for trajectory tracking of mobile robot," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots. Syst.*, 2010, pp. 4755–4760.

[22] C. Dong, Z. Yu, X. Chen, H. Chen, Y. Huang, and Q. Huang, "Adaptability control towards complex ground based on fuzzy logic for humanoid robots," *IEEE Trans. Fuzzy Syst.*, vol. 30, no. 6, pp. 1574–1584, 2022.

[23] W. Tang, Y. Zhou, Y. Liu, Z. Ding, and J. Liu, "Robust motion planning for multi-robot systems against position deception attacks," *IEEE Transactions on Information Forensics and Security*, 2023.

[24] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *Int. Conf. Mach. Learn.*, 2014, pp. 387–395.

[25] Y. Ren, Y. Sun, and L. Liu, "Fuzzy disturbance observers-based adaptive fault-tolerant control for an uncertain constrained automatic flexible robotic manipulator," *IEEE Trans. Fuzzy Syst.*, 2023.

[26] M. Van, Y. Sun, S. Mcllvanna, M.-N. Nguyen, M. O. Khyam, and D. Ceglarek, "Adaptive fuzzy fault tolerant control for robot manipulators with fixed-time convergence," *IEEE Trans. Fuzzy Syst.*, 2023.

[27] Y. Dong and X. Zou, "Mobile robot path planning based on improved ddpg reinforcement learning algorithm," in *IEEE Int. Conf. Softw. Eng. Serv. Sci.*, 2020, pp. 52–56.