Autonomous UAV Navigation: A DDPG-based Deep Reinforcement Learning Approach

Omar Bouhamed[1], Hakim Ghazzai[1], Hichem Besbes[2] and Yehia Massoud[1]
[1]School of Systems & Enterprises, Stevens Institute of Technology, Hoboken, NJ, USA
[2]University of Carthage, Higher School of Communications of Tunis, Tunisia

***Abstract*—In this paper, we propose an autonomous UAV path planning framework using deep reinforcement learning approach. The objective is to employ a self-trained UAV as a flying mobile unit to reach spatially distributed moving or static targets in a given three dimensional urban area. In this approach, a Deep Deterministic Policy Gradient (DDPG) with continuous action space is designed to train the UAV to navigate through or over the obstacles to reach its assigned target. A customized reward function is developed to minimize the distance separating the UAV and its destination while penalizing collisions. Numerical simulations investigate the behavior of the UAV in learning the environment and autonomously determining trajectories for different selected scenarios.**

***Index Terms*—Autonomous navigation, deep reinforcement learning, obstacle avoidance, unmanned aerial vehicle.**

I. Introduction

Smart cities are witnessing a rapid development to provide satisfactory quality of life to its citizens [1]. The establishment of such cities requires the integration and use of novel and emerging technologies. In this context, unmanned areal vehicles (UAV), *aka* drones, are continuously proving their efficiency in leveraging multiple services in several fields, such as good delivery and traffic monitoring (e.g. Amazon is starting to use UAVs to deliver packages to customers). UAVs are easy to deploy with a three dimensional (3D) mobility as well as a flexibility in performing difficult and remotely located tasks while providing bird-eye view [2], [3]. Path planning remains one of key challenges that need to be solved to improve UAV navigation especially in urban areas. The core idea is to devise optimal or near-optimal collision-free path planning solutions to guide UAVs to reach a given target, while taking into consideration the environment and obstacle constraints in the area of interest.

In recent studies, such as [4], the authors adopted the ant colony optimization algorithm to determine routes for UAVs while considering obstacle avoidance for modern air defence syste. In [5], a combination of grey wolf optimization and fruit fly optimization algorithms is proposed for the path planning of UAV in oilfield environment. In [6]–[8], the UAV path planning problems were modeled as mixed integer linear programs (MILP) problem. Also, in [9], a 3D path planning method for multi-UAVs system or single UAV is proposed to find a safe and collision-free trajectory in an environment containing obstacles. However, most of the solutions are based on MILP which are computationally complex or evolutionary algorithms, which do not necessarily reach near-optimal solutions. Moreover, the existing approaches remain centralized where a central node, e.g. a control center runs the algorithm and provides to the UAV its path plan. Centralized approaches restrain the system and limit its capabilities to deal with real-time problems. They impose a certain level of dependency and cost additional communication overhead between the central node and the flying unit. Hence, artificial intelligence (AI), precisely, reinforcement learning (RL) come out as a new research tendency that can grant the flying units sufficient intelligence to make local decisions to accomplish necessary tasks. In [10] and [11], the authors presented a Q-learning algorithm to solve the autonomous navigation problem of UAVs. Q-learning was also employed to establish paths while avoiding obstacles in [12]. However, the authors used discrete actions (i.e. the environment is modeled as a grid world with limited UAV action space, degree of freedom). which may reduce the UAV efficiency while dealing with real-world environment, where the flying units operate according to a continuous action space.

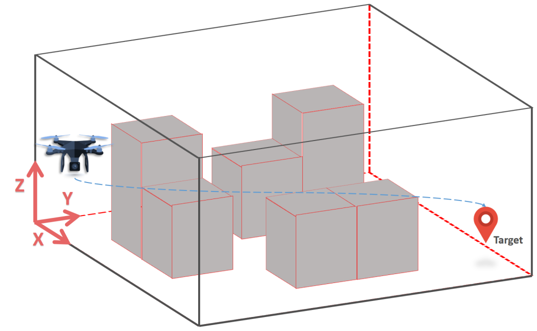

Fig. 1: Illustration of the autonomous obstacle-aware UAV navigation in an urban environment.

Unlike existing RL-based solutions which are usually operating on a discretized environment, the proposed framework aims to provide UAV autonomous navigation with continuous action space to reach fixed or moving targets dispersed within a 3D space area while considering the UAV safety. A deep deterministic gradient decent (DDPG)-based approach is modeled with the objective to allow an UAV determine the best course to accomplish its missions safely, i.e. obstacle avoidance. A reward function is designed to guide the UAV toward its destination while penalizing any crash. During the training phase, we adopt a transfer learning approach to train the UAV how to reach its destination in a free-space environment (i.e., source task). Then, the learned model is fed to other models (i.e., new task) dedicated to different environments with specific obstacles' locations so that the UAV can learn how to avoid obstacles to navigate to the destination. During the prediction phase, it determines the path within the training environment by figuring out which route to take to reach any randomly generated static or dynamic destination from any

978-1-7281-3320-1/20/$31.00 ©2020 IEEE

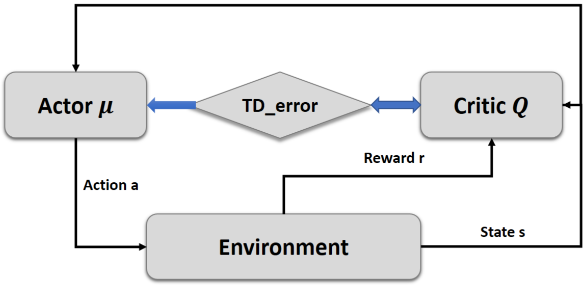

Fig. 2: Architecture of the actor-critic learning model.

arbitrary starting position. In the simulations, we investigate the behavior of the autonomous UAVs for different scenarios including obstacle-free and urban environments. It is shown that the UAV smartly selects paths to reach its target while avoiding obstacles either by crossing over or deviating them.

II. Problem Description

In this section, we present the system model and describe the actions that can be taken by the UAV to enable its autonomous navigation.

A. System Model

Autonomous navigation for UAVs in real environment is complex. Hence, Without loss of generality, we create a virtual 3D environment with high matching degree to the real-world urban areas. Unlike most of the existing virtual environments, which are studied in literature and usually modeled as a grid world, in this paper, we focus on a free space environment containing 3D obstacles that may have diverse shapes as illustrated in Fig. 1. Consequently, the UAV has the freedom to take any direction and speed to reach its target unlike grid world, which restricts the freedom of UAV into a finite set of actions. The goal is to train the UAV to fly safely from any arbitrary starting position to reach any destination in the considered area with continuous action space. The UAV, defined as u, is characterized by its 3D Cartesian geographical location $loc_u = [x, y, z]$ and initially situated at $loc_u(0) = [x_0, y_0, z_0]$. The destination d is defined by its 3D location $loc_d = [x_d, y_d, z_d]$. In this paper, the investigated system assumes the following assumptions:

- The environment obstacles have different heights. Each one of them is represented by a 3D polygon characterized by its the starting point $[x_{obs}, y_{obs}]$, the set containing the edges of the base edg_{obs}, and its height h_{obs}. Hence, if having an altitude higher than the obstacle's height, the UAV can cross over the obstacles. Otherwise, the UAV can avoid it by flying around.
- The destination location is known to the UAV and it can be either static or dynamic (i.e., the target location can evolve over time). If the destination location is dynamic then it follows a random pre-defined trajectory, that is unknown by the UAV.

B. UAV Actions

For each taken action, we assume that the UAV chooses a distance to cross according to a certain direction in the 3D space during Δt units of time. The action is modeled using the spherical coordinates (ρ, ϕ, ψ) as follows:

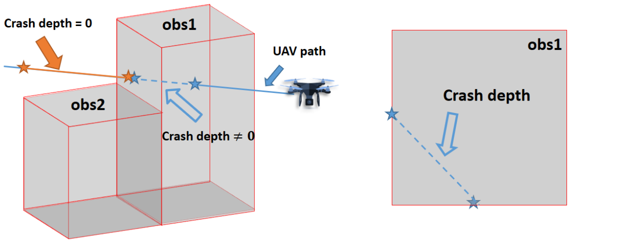

Fig. 3: Example representing a collision scenario. The UAV's altitude is less than the obstacle's height (obs1). The action is chosen such that the UAV crosses the obstacle. This results in an obstacle penalty reflecting the underwent depth.

$$x_u = x_u + \rho \sin\phi \cos\psi, \quad y_u = y_u + \rho \sin\phi, \\ \text{and } z_u = z_u + \rho \cos\phi, \qquad (1)$$

where ρ is the traveled radial distance by the UAV in each step ($\rho \in [\rho_{min}, \rho_{max}]$), where ρ_{max} is the maximum distance that the UAV can cross during the step length Δt. Its value depends on the maximum speed of the UAV denoted by v_{max}. The parameter ψ denotes the inclination angle ($\psi \in [0, 2\pi]$), and ϕ represents the elevation angle ($\phi \in [0, \pi]$). For instance:

- if $\rho = \rho_{max}$, $\phi = \pi$, and any value of ψ, the UAV moves by ρ_{max} along the Z axis.
- if $\rho = \rho_{max}$, $\phi = \pi/2$, and $\psi = 0$, the UAV moves along the x axis.

The distance between the UAV and its target is defined as $D(u, d)$.

III. DDPG Learning for Autonomous Scheduling

DDPG was developed as an extension of deep Q-network (DQN) algorithms introduced by Mnih et al. [13], which was the first approach combining deep and reinforcement learning but only by handling low-dimensional action spaces. DDPG is also a deep RL algorithm, that has the capability to deal with large-dimensional/infinite action spaces. It tries to find an efficient behavior strategy for the agent to obtain maximal rewards in order to accomplish its assigned tasks [14]. This DPG algorithm has the capability to operate over continuous action spaces which is a major hurdle for classic RL methods like Q-learning.

DDPG is based on the actor-critic algorithm. It is essentially a hybrid method that combines the policy gradient and the value function together. The policy function μ is known as the actor, while the value function Q is referred to as the critic. Essentially, the actor output is an action chosen from a continuous action space, given the current state of the environment $a = \mu(s|\theta^\mu)$, which, in our case, has the form of a tuple $a = [\rho, \phi, \psi]$. As for the critic, its output $Q(s, a|\theta^\mu)$ is a signal having form of a Temporal Difference (TD) error to criticize the actions made by the actor knowing the current state of the environment. A diagram summarizing the actor-critic architecture is given in Fig. 2.

Note that the training phase of the DDPG model is executed for M episodes where each one of them accounts for T steps. We use the index t to denote an iteration within a single episode where $t = 1, \ldots, T$. The actor and critic are designed with neural networks. The value network is updated based

Algorithm 1 DDPG

```
1:  Randomly initialize critic $Q(s,a|\theta^{\mu})$ and actor $\mu(s|\theta^{\mu})$ neural networks with weights $\theta^{Q}$ and $\theta^{\mu}$.
2:  Initialize target networks $Q'$ and $\mu'$ with weights $\theta^{Q'} \leftarrow \theta^{Q}$, $\theta^{\mu'} \leftarrow \theta^{\mu}$.
3:  Initialize replay buffer $b$.
4:  for episode = 1,...,$M$ do
5:      Receive first observation $s_1$.
6:      for t = 1,...,$T$ do
7:          Select $a_t$ based on $\epsilon$-greedy algorithm: select random action $a_t$ with $\epsilon$ probability, otherwise $a_t = \mu(s_t|\theta^{\mu})$ according to the current policy.
8:          Execute action $a_t$ and observe reward $r_t$ and new state $s_{t+1}$.
9:          Store transition $[s_t, a_t, r_t, s_{t+1}]$ in $b$.
10:         Sample a random batch of $N$ transitions $[s_j, a_j, r_j, s_{j+1}]$.
11:         Set $y_j = r_j + \gamma Q'(s_{j+1}, \mu'(s_{j+1}|\theta^{\mu'})|\theta^{Q'})$.
12:         Update critic by minimizing the loss:
                $L = \frac{1}{N}\sum_j (y_j - Q(s_j, a_j|\theta^{Q}))^2$
13:         Update the actor policy using policy gradient:
            $\nabla_{\theta^{\mu}}\mu|_{s_j} \approx \frac{1}{N}\sum_j \nabla_a Q(s,a|\theta^{Q})|_{s=s_j, a=\mu(s_j)} \nabla_{\theta^{\mu}}\mu(s|\theta^{\mu})|_{s_j}$
14:         Update the target networks:
                $\theta^{Q'} \leftarrow \nu\theta^{Q} + (1-\nu)\theta^{Q'}$
                $\theta^{\mu'} \leftarrow \nu\theta^{\mu} + (1-\nu)\theta^{\mu'}$
15:     end for
16: end for
```

on Bellman equation [15] by minimizing the mean-squared loss between the updated Q value and the origin value, which can be formulated as shown in Algorithm 1 (line 11). As for the policy network's update (line 13), it is based on the deterministic policy gradient theorem [16].

There are also some practical tricks that are used to enhance the performance of the framework. A trade off between exploration and exploitation is made by the use of ϵ-greedy algorithm, where a random action a_t is selecting with ϵ probability, otherwise a precise action $a_t = \mu(s_t|\theta^{\mu})$ is selected according to the current policy with a $1-\epsilon$ probability. Furthermore, an experience replay buffer b, with size B, is used during the training phase to break the temporal correlations. Each interaction with the environment is stored as tuples in the form of $[s_t, a, r, s_{t+1}]$, which are the current state, the action to take, the reward of performing action a at state s_t, and the next state, respectively (Algorithm 1 (line 9)) and, during the learning phase, a randomly extracted set of data from the buffer is used (Algorithm 1 (line 10)). Also, target networks are exploited to avoid the divergence of the learning algorithm caused by the direct updates of the networks weights with the gradients obtained from the TD error signal.

A. Reward Function

In an obstacle-constrained environment, the UAV must avoid obstacles and autonomously navigate to reach its destination in real-time. Therefore, the reward function, denoted by f_r, is modeled such that it encourages the UAV to reach its destination and, at the same time, penalizes it when crashing. Thus, the reward function is composed of two terms: target guidance reward and obstacle penalty. The target guidance reward, denoted by f_{gui}, is used to motivate the flying unit to reach its target as fast as possible, while the obstacle penalty, denoted by f_{obp} is responsible for alerting the UAV to keep a certain safety distance off the obstacles. The reward function is formulated as follows:

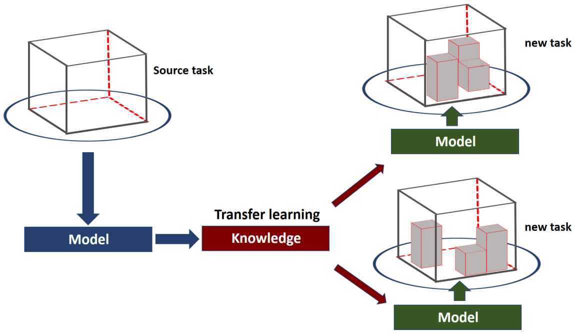

Fig. 4: Illustration of the transfer-learning technique.

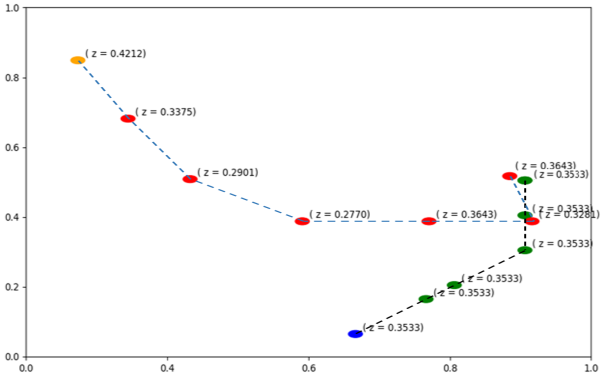

Fig. 5: Path followed by UAV based on the dynamic target location. Yellow dot: initial UAV position, blue dashed line: trajectory of the UAV, blue dot: target initial location, and dashed black line: the target trajectory.

$$f_r(D(u,d),\sigma) = (1-\beta)f_{gui}(D(u,d)) + \beta f_{obp}(\sigma), \quad (2a)$$

$$f_{gui}(D(u,d)) = \exp(-5D(u,d)^2), \quad (2b)$$

$$f_{obp}(\sigma) = \exp(-100\sigma) - 1, \quad (2c)$$

where σ is the crash depth explained in Fig. 3 and β is a variable that regulates the balance between f_{obp} and f_{gui}. The obstacle penalty is modeled as a function of the crash depth σ to conserve the continuous nature of the reward function instead of using discrete penalty, which proved to be more efficient to help the model to converge. In fact, when the crash depth is high, the UAV receives a higher penalty, whereas a small crash depth results in a lower penalty. The use of this approach helps the UAV learn efficiently over the training episodes how to adjust its trajectory to avoid obstacles.

B. Training Phase and Transfer Learning

Transfer learning is a machine learning technique used to transfer the knowledge to speed up training and improve the performance of deep learning models. The proposed approach to train the UAV consists in two steps. Initially, we train the model in an obstacle-free environment. Training in such environment, grants the UAV the capability to reach any target in the covered 3D area with continuous space action. Then, the trained model on the obstacle-free environment will serve as a base for future models trained on other environments with obstacles. Afterwards, we transfer the acquired knowledge (i.e. source task) and use it to improve the UAV learning of new tasks where it updates its path based on the obstacle locations while flying toward its target. The adopted transfer learning technique applied to DDPG for autonomous UAV

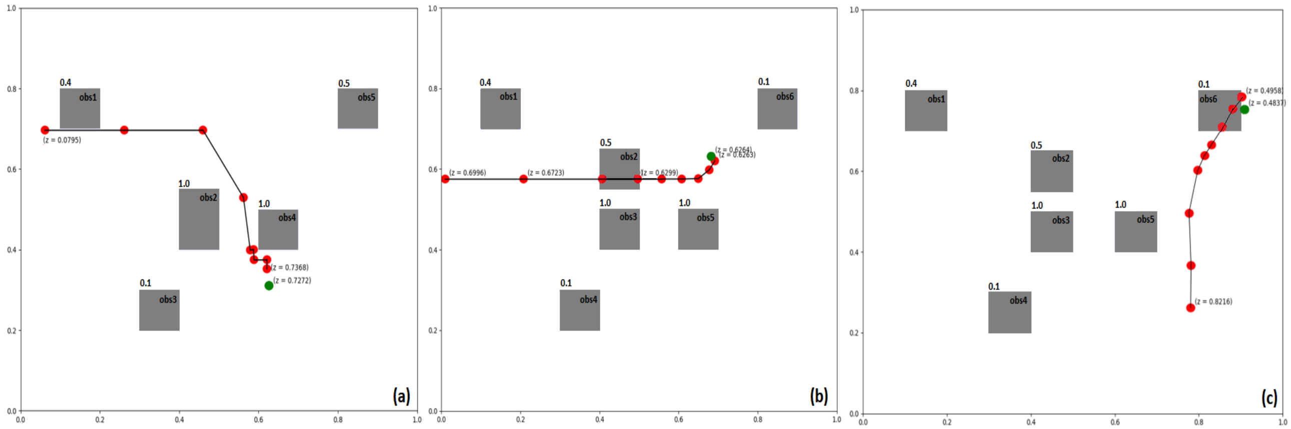

Fig. 6: Path followed by UAV based on the static target location. Red dots refers to the UAV, green dot refers to the target, and gray boxes to the obstacles. (a) environment 1 (env1) and (b, c) two cases in environment 2 (env2).

Table I: Task-completion rate

Scenarios	obstacle free	obstacles env1	obstacles env2
Completion rate	100%	84%	82%

navigation is illustrated in Fig. 4. Once the training phase is completed offline, the UAV is capable to make instant decisions, while interacting with the environment, to manage real-time missions.

IV. Simulation Results

In this section, we study the behavior of the system for selected scenarios. We also visualize the efficiency of the framework in terms of crash rate and tasks accomplishment. To do so, we assume that the UAV starting location loc_u, its target location loc_d, and the obstacles' parameters are randomly generated within a cube-shaped area with 100 m edge length. We make sure that the locations of both the targets and the UAV are outside the obstacles. The rest of the simulation parameters are set as follows: $\beta = 4$, $\nu = 0.99$, $\rho_{\max=0.2}$ m, $T = 100$, $M = 40000$, $(\epsilon_{end}, \epsilon_{start}) = (0.1, 0.9)$, $B = 10000$, and $N = 256$. The simulations are executed using Python.

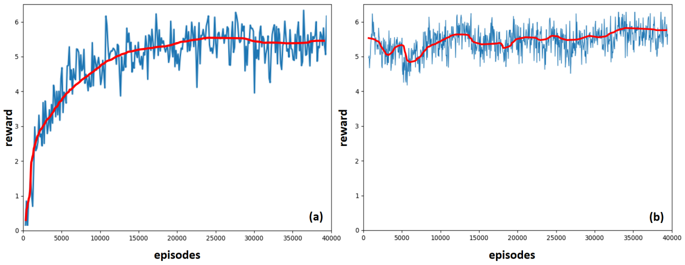

Fig. 7: The reward received by the UAV during the training phase: (a) the source task (obstacle-free) and (b) the environment with obstacles

For the sake of clarity, the figures concerning the UAV path planning are presented in only 2D dimension area (i.e., plan(x,y)) and we provide beside each dot, the altitude of either the target or the UAV. In the first scenario, we consider an obstacle-free environment. The destination location is assumed to be dynamic, that it keeps moving in a randomly generated way. As shown in Fig. 5, the UAV is successfully adapting its trajectory based on the location of its target until it reaches it. This shows that the UAV succeeded in learning how to update each direction in order to "catch" its assigned destination.

In the next scenarios, the obstacles are added in a random disposition with different heights as shown in Fig. 6. In these cases, we assume that the target destinations are static. In Fig. 6(a), the UAV successfully reached its destination location while avoiding the obstacles. In Fig. 6(b), on its way to the destination, the UAV crossed over $obs2$ ($z_u = 0.63 > h_{obs2} = 0.5$) in order to reach faster its target location unlike the case in Fig. 6(a), where the UAV could not cross over $obs2$ to reach its destination as soon as possible because of the obstacle height (maximum height). In Fig. 6(c), having a higher altitude than $obs6$, the UAV crossed over $obs6$ to reach its target. In all cases, scenarios show some lacking in precision to reach the target location due to the fact of using infinite action space which makes it hard to get pinpoint accuracy. These scenarios showed that the UAV successfully learned how to avoid obstacles to reach its destination.

In Fig. 7, we present the reward received by the UAV during its training phase, in Fig. 7(a) shows that the UAV learns to obtain the maximum reward value in an obstacle-free environment. We successfully obtained a trained model capable of reaching targets in 3D environment with continuous action space. Then, using the knowledge gathered by the first training, we trained the model to be able to avoid obstacles. Fig. 7(b) shows that the UAV model has converged and reached the maximum possible reward value.

During the testing phase and as shown in Table I, for the obstacle-free environment, the UAV successfully reached its target for the tested cases, 100% success rate for 1000 test case. As for the environment with obstacles, in the case of env1, the UAV successfully reached its target safely for 84% of the 1000 tested scenarios and in the case of env2, the reached its target safely for 82% of the 1000 tested scenarios.

V. Conclusion

In this paper, we have developed an efficient framework for autonomous obstacle-aware UAV navigation in urban areas. Using a DDPG-based deep reinforcement learning approach, the UAV determines its trajectory to reach its assigned static or dynamic destination within a continuous action space. A transfer learning approach is devised in order to maximize a reward function balancing between target guidance and obstacle penalty. The simulation results exhibit the capability of UAVs in learning from the surrounding environment to determine their trajectories in real-time.

References

[1] S. P. Mohanty, U. Choppali, and E. Kougianos, “Everything you wanted to know about smart cities: The internet of things is the backbone,” *IEEE Consumer Electronics Magazine*, vol. 5, no. 3, pp. 60–70, July. 2016.

[2] M. B. Ghorbel, D. Rodríguez-Duarte, H. Ghazzai, M. J. Hossain, and H. Menouar, “Joint position and travel path optimization for energy efficient wireless data gathering using unmanned aerial vehicles,” *IEEE Transactions on Vehicular Technology*, vol. 68, no. 3, pp. 2165–2175, Mar. 2019.

[3] H. Ghazzai, H. Menouar, A. Kadri, and Y. Massoud, “Future uav-based its: A comprehensive scheduling framework,” *IEEE Access*, vol. 7, pp. 75 678–75 695, June 2019.

[4] J. Chen, F. Ye, and T. Jiang, “Path planning under obstacle-avoidance constraints based on ant colony optimization algorithm,” in *IEEE 17th International Conference on Communication Technology (ICCT'17)*, China, Oct. 2017.

[5] F. Ge, K. Li, W. Xu, and Y. Wang, “Path planning of uav for oilfield inspection based on improved grey wolf optimization algorithm,” in *Chinese Control And Decision Conference (CCDC'19)*, China, June. 2019.

[6] Z. Zhang, J. Wang, J. Li, and X. Wang, “Uav path planning based on receding horizon control with adaptive strategy,” in *29th Chinese Control And Decision Conference (CCDC'17), Chongqing, China*, May 2017.

[7] A. Bahabry, X. Wan, H. Ghazzai, H. Menouar, G. Vesonder, and Y. Massoud, “Low-altitude navigation for multi-rotor drones in urban areas,” *IEEE Access*, vol. 7, pp. 87 716–87 731, June 2019.

[8] A. Bahabry, X. Wan, H. Ghazzai, G. Vesonder, and Y. Massoud, “Collision-free navigation and efficient scheduling for fleet of multi-rotor drones in smart city,” *IEEE International Midwest Symposium on Circuits and Systems (MWSCAS'19), Dallas, TX, USA*, Aug. 2019.

[9] L. Lifen, S., L. Shuandao, and W. Jiang, “Path planning for uavs based on improved artificial potential field method through changing the repulsive potential function,” in *IEEE Chinese Guidance, Navigation and Control Conference (CGNCC'16)*, China, Aug. 2016.

[10] C. Yan and X. Xiang, “A path planning algorithm for uav based on improved q-learning,” in *2nd International Conference on Robotics and Automation Sciences (ICRAS'18), China*, June 2018.

[11] O. Bouhamed, H. Ghazzai, H. Besbes, and Y. Massoud, “Q-learning based routing scheduling for a multi-task autonomous agent,” *IEEE International Midwest Symposium on Circuits and Systems (MWSCAS'19), Dallas, TX, USA*, Aug. 2019.

[12] V. N. Sichkar, “Reinforcement learning algorithms in global path planning for mobile robot,” in *International Conference on Industrial Engineering, Applications and Manufacturing (ICIEAM'19), Sochi, Russia.*, Mar 2019, pp. 1–5.

[13] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, “Human-level control through deep reinforcement learning,” *Nature*, vol. 518, Feb. 2015.

[14] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, “Continuous control with deep reinforcement learning.” in *ICLR*, 2016.

[15] R. Bellman, *Dynamic Programming*, ser. Dover Books on Computer Science. Dover Publications, 2013.

[16] T. Lillicrap, J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, “Continuous control with deep reinforcement learning,” *CoRR*, 09 2015.