

# Autonomous UAV Navigation Using Reinforcement Learning

Huy X. Pham, Hung. M. La, David Feil-Seifer, Luan V. Nguyen

**Abstract**— Unmanned aerial vehicles (UAV) are commonly used for missions in unknown environments, where an exact mathematical model of the environment may not be available. This paper provides a framework for using reinforcement learning to allow the UAV to navigate successfully in such environments. We conducted our simulation and real implementation to show how the UAVs can successfully learn to navigate through an unknown environment. Technical aspects regarding to applying reinforcement learning algorithm to a UAV system and UAV flight control were also addressed. This will enable continuing research using a UAV with learning capabilities in more important applications, such as wildfire monitoring, or search and rescue missions.

## I. INTRODUCTION

Using unmanned aerial vehicles (UAV), or drones, in missions involving navigating through unknown environment, such as wildfire monitoring [1], target tracking [2]–[4], or search and rescue [5], is becoming more widespread, as they can host a wide range of sensors to measure the environment with relative low operation costs and high flexibility. One issue is that most current research relies on the accuracy of the model describing the target, or prior knowledge of the environment [6], [7]. It is, however, very difficult to attain this in most realistic implementations, since the knowledge and data regarding the environment are normally limited or unavailable. Using reinforcement learning (RL) is a good approach to overcome this issue because it allows a UAV or a UAV team to learn and navigate through the changing environment without a model of the environment [8].

RL algorithms have already been extensively researched in UAV applications, as in many other fields of robotics [9], [10]. Many papers focus on applying RL algorithm into UAV control to achieve desired trajectory tracking/following. In [11], Faust et al. proposed a framework using RL in motion planning for UAV with suspended load to generate trajectories with minimal residual oscillations. Bou-Ammar et al. [12] used RL algorithm with fitted value iteration to attain stable trajectories for UAV maneuvers comparable to model-based feedback linearization controller. A RL-based learning automata designed by Santos et al. [13] allowed parameters tuning for a PID controller for UAV in a tracking problem, even under adversary weather conditions. Waslander et al. [14] proposed a test-bed applying RL for accommodating the nonlinear disturbances caused by complex airflow in UAV control. Other papers discussed problems in improving

Huy Pham and Luan Nguyen are PhD students, and Dr. Hung La is the director of the Advanced Robotics and Automation (ARA) Laboratory. Dr. David Feil-Seifer is an Assistant Professor of Department of Computer Science and Engineering, University of Nevada, Reno, NV 89557, USA. Corresponding author: Hung La, email: hla@unr.edu

RL performance in UAV application. Imanberdiyev et al. [15] used a platform named TEXPLORE which processed the action selection, model learning, and planning phase in parallel to reduce the computational time. Zhang et al. [16] proposed a geometry-based Q-learning to extend the RL-based controller to incorporate the distance information in the learning, thus lessen the time needed for an UAV to reach a target.

However, to the best of our knowledge, there are not many papers discussing about using RL algorithm for UAVs in high-level context, such as navigation, monitoring or other complex task-based applications. Many papers often did not provide details on the practical aspects of implementation of the learning algorithm on physical UAV systems. In this paper, we provide a detailed implementation of a UAV that can learn to accomplish tasks in an unknown environment. Using a simple RL algorithm, the drone can navigate successfully from an arbitrary starting position to a goal position in shortest possible way. The main contribution of the paper is to provide a framework for applying a RL algorithm to enable UAV to operate in such environment. The remaining of the paper is organized as follows. Section II provides more detail on problem formulation, and the approach we use to solve the problem. Basics in RL and how we design the learning algorithm are discussed in section III. We conduct a simulation of our problem on section IV, and provide details on UAV control in section V. Subsequently, a comprehensive implementation of the algorithm will be discussed in section VI. Finally, we conclude our paper and provide future work in section VII.

## II. PROBLEM FORMULATION

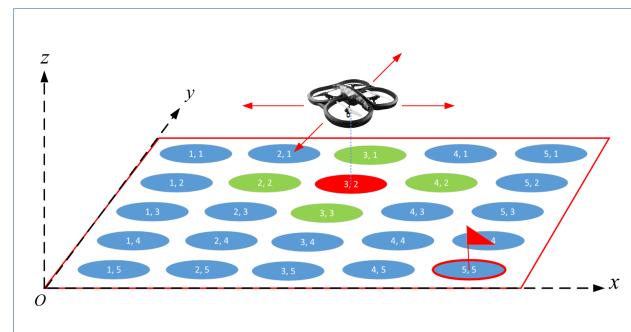


Fig. 1. A UAV navigating in closed environment with discretized state space, represented by discrete circles. The red circle is the UAV's current state, the green circles are the options that the UAV can choose in the next iteration. The goal is marked by a red flag.

Suppose that we have a closed environment in which the prior information about it is limited. We would like a

flying robot, for example a quadcopter-type UAV, start at an arbitrary position to reach a goal that is pre-described to the robot (Figure 1). We assume that at any position, the UAV can observe its state, i.e. its position. If we have full information about the environment, for instance, the exact distance to the target or the locations of the obstacles, a robot motion planning can be constructed based on the model of the environment, and the problem becomes common. Traditional control methods, such as potential field [17], [18], are available to solve such problem. In many realistic cases, however, building models is not possible because the environment is insufficiently known, or the data of the environment is not available or difficult to obtain. Since RL algorithms can rely only on the data obtained directly from the system, it is a natural option to consider for our problem. In the learning process, the agent needs to map the situations it faces to appropriate actions so as to maximize a numerical signal, called reward, that measures the performance of the agent.

To carry out the given task, the UAV must have a learning component to enable it to find the way to the goal in an optimal fashion. Based on its current state  $s_k$  (e.g., UAV's position) and its learning model, the UAV decides the action to the next state  $s_{k+1}$  it wants to be. A desired position then will be taken as input to the position controller, that calculates the control input  $u(t)$  to a lower-level propellers controller. This low-level controller will control the motors of the UAV to generate thrust force  $\tau$  to drive it to the desired position. Note that the position controller must be able to overcome the complex nonlinear dynamics of UAV system, to achieve stable trajectories for the UAV when flying, as well as hovering in the new state. Figure 2 shows the block diagram of our controller.

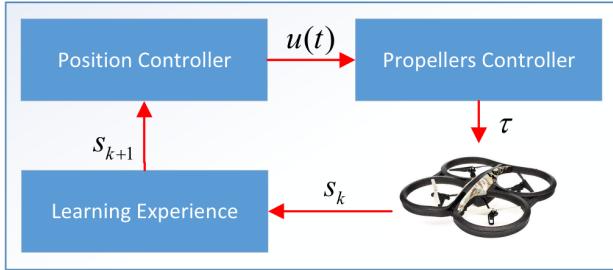


Fig. 2. Reinforcement Learning model.

### III. REINFORCEMENT LEARNING AND Q LEARNING

RL becomes popular recently thanks to its capabilities in solving learning problem without relying on a model of the environment. The learning model can be described as an agent–environment interaction in Figure 3. An agent builds up its knowledge of the surrounding environment by accumulating its experience through interacting with the environment. Assuming that the environment has Markovian property, where the next state and reward of an agent only depends on the current state [8]. The learning

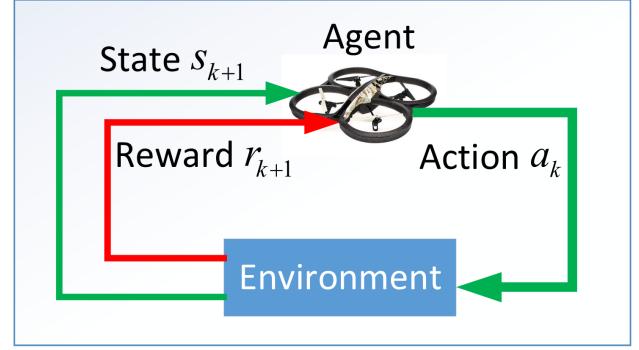


Fig. 3. Reinforcement Learning model.

model can be generalized as a tuple  $< S, A, T, R >$ , where:

- $S$  is a finite state list, and  $s_k \in S$  is the state of the agent at step  $k$ ;
- $A$  is a finite set of actions, and  $a_k \in A$  is the action the agent takes at step  $k$ ;
- $T$  is the transition probability function,  $T : S \times A \times S \rightarrow [0, 1]$ , is the probability of agent that takes action  $a_k$  to move from state  $s_k$  to state  $s_{k+1}$ . In this paper, we consider our problem as a deterministic problem, so as  $T(s_k, a_k, s_{k+1}) = 1$ .
- $R$  is the reward function:  $R : S \times A \rightarrow \mathbb{R}$  that specifies the immediate reward of the agent for getting to state  $s_{k+1}$  from  $s_k$  after taking action  $a_k$ . We have:  $R(s_k, a_k) = r_{k+1}$ .

The objective of the agent is to find a course of actions based on its states, called a policy, that ultimately maximizes its total amount of reward it receives over time. In each state, a state - action value function  $Q(s_k, a_k)$ , that quantifies how good it is to choose an action in a given state, can be used for the agent to determine which action to take. The agent can iteratively compute the optimal value of this function, and from which derives an optimal policy. In this paper, we apply a popular RL algorithm known as Q-learning [19], in which the agent computes optimal value function and records them into a tabular database, called Q-table. This knowledge can be recalled to decide which action it would take to optimize its rewards over the learning episodes. For each iteration, the estimation of the optimal state - action value function is updated following the Bellman equation [8]:

$$Q_{k+1}(s_k, a_k) \leftarrow (1 - \alpha)Q_k(s_k, a_k) + \alpha[r_{k+1} + \gamma \max_{a'} Q_k(s_{k+1}, a')], \quad (1)$$

where  $0 \leq \alpha \leq 1$  and  $0 \leq \gamma \leq 1$  are learning rate and discount factor of the learning algorithm, respectively. To keep balance between exploration and exploitation actions, the paper uses a simple policy called  $\epsilon$  greedy, with  $0 < \epsilon < 1$ , as follows:

$$\pi(s) = \begin{cases} \text{a random action } a, & \text{with probability } \epsilon; \\ a \in \arg \max_{a'} Q_k(s_k, a'), & \text{otherwise.} \end{cases} \quad (2)$$

In order to use Q-learning algorithm, one must define the set of states  $S$ , actions  $A$  and rewards  $R$  for an agent in the system. Since the continuous space is too large to guarantee the convergence of the algorithm, in practice, normally these set will be represented as discrete finite sets approximately [20]. In this paper, we consider the environment as a finite set of spheres with equal radius  $d$ , and their centers form a grid. The center of the sphere now represents a discrete location of the environment, while the radius  $d$  is the error deviation from the center. It is assumed that the UAV can generate these spheres for any unknown environment. The state of an UAV is then defined as their approximate position in the environment,  $s_k \triangleq c = [x_c, y_c, z_c] \in S$ , where  $x_c, y_c, z_c$  are the coordinates of the center of a spheres  $c$  at time step  $k$ . For simplicity, in this paper we will keep the altitude of the UAV as constant to reduce the number of states. The environment becomes a 2-D environment and the spheres now become circles. The state of the drone at time step  $k$  is the lateral position of center  $c$  of a circle  $s_k = [x_c, y_c]$ . Figure 1 shows the discrete state space of the UAV used in this paper.

In each state, the UAV can take an action  $a_k$  from a set of four possible actions  $A$ : heading North, West, South or East in lateral direction, while maintaining the same altitude. After an action is decided, the UAV will choose an adjacent circle where position is corresponding to the selected action. Note that the its new state  $s_{k+1}$  is now associated with the center of the new circle. Figure 1 shows number of options the UAV can take (in green color) in a particular state. Note that if the UAV stays in a state near the border of the environment, and selects an action that takes it out of the space, it should stay still in the current state. The rewards that an UAV can get depend whether it has reached the pre-described goal  $G$ , recognized by the UAV using a specific landmark, where it will get a big reward. Reaching other places that is not the desired goal will result in a small penalty (negative reward):

$$r_{k+1} = \begin{cases} 100, & \text{if } s_{k+1} \equiv G \\ -1, & \text{otherwise.} \end{cases} \quad (3)$$

#### IV. CONTROLLER DESIGN AND ALGORITHM

In this section, we provide a simple position controller design to help a quadrotor-type UAV to perform the action  $a_k$  to translate from current location  $s_k$  to new location  $s_{k+1}$  and stay hovering over the new state within a small error radius  $d$ . Define  $p_t$  is the real-time position of the UAV at time  $t$ , we start with a simple proportional gain controller:

$$u(t) = K_p(p(t) - s_{k+1}) = K_p e(t), \quad (4)$$

where  $u(t)$  is the control input,  $K_p$  is the proportional control gain, and  $e(t)$  is the tracking error between real-time position  $p(t)$  and desired location  $s_{k+1}$ . Because of the nonlinear dynamics of the quadrotor [18], we experienced excessive overshoots of the UAV when it

navigates from one state to another (Figure 5), making the UAV unstable after reaching a state. To overcome this, we used a standard PID controller [21] (Figure 4). Although the controller cannot effectively regulate the nonlinearity of the system, work such as [22], [23] indicated that using PID controller could still yield relatively good stabilization during hovering.

$$u(t) = K_p e(t) + K_i \int e(t) dt + K_d \frac{de}{dt}. \quad (5)$$

---

#### Algorithm 1: PID + Q-LEARNING.

---

```

Input: Learning parameters: Discount factor  $\gamma$ , learning rate  $\alpha$ , number of episode  $N$ 
Input: Control parameters: Control gains  $K_p, K_i, K_d$ , error radius  $d$ 
1 Initialize  $Q_0(s, a) \leftarrow 0, \forall s_0 \in S, \forall a_0 \in A$ ;
2 for  $episode = 1 : N$  do
3   Measure initial state  $s_0$ 
4   for  $k = 0, 1, 2, \dots$  do
5     Choose  $a_k$  from  $A$  using policy (2)
6     Take action  $a_k$  that leads to new state  $s_{k+1}$ :
7     for  $t = 0, 1, 2, \dots$  do
8       
$$u(t) = K_p e(t) + K_i \int e(t) dt + K_d \frac{de}{dt}$$

9       until  $\|p(t) - s_{k+1}\| \leq d$ 
10      Observe immediate reward  $r_{k+1}$ 
11      Update:
12        
$$Q_{k+1}(s_k, a_k) \leftarrow (1 - \alpha)Q_k(s_k, a_k) + \alpha[r_{k+1} + \gamma \max_{a'} Q_k(s_{k+1}, a')]$$

13    until  $s_{k+1} \equiv G$ 

```

---

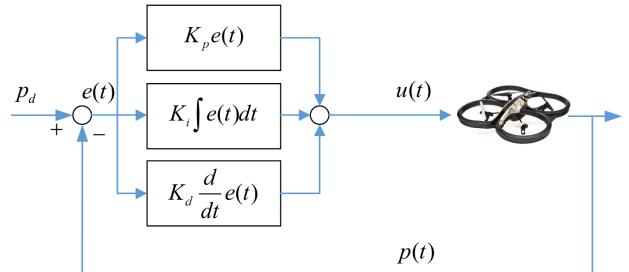


Fig. 4. The PID control diagram with 3 components: proportional, integral and derivative terms.

Generally, the derivative component can help decrease the overshoot and the settling time, while the integral component can help decrease the steady-state error, but can cause increasing overshoot. During the tuning process, we increased the Derivative gain while eliminated the Integral component of the PID control to achieve stable trajectory. Note that  $u(t)$  is calculated in the Inertial

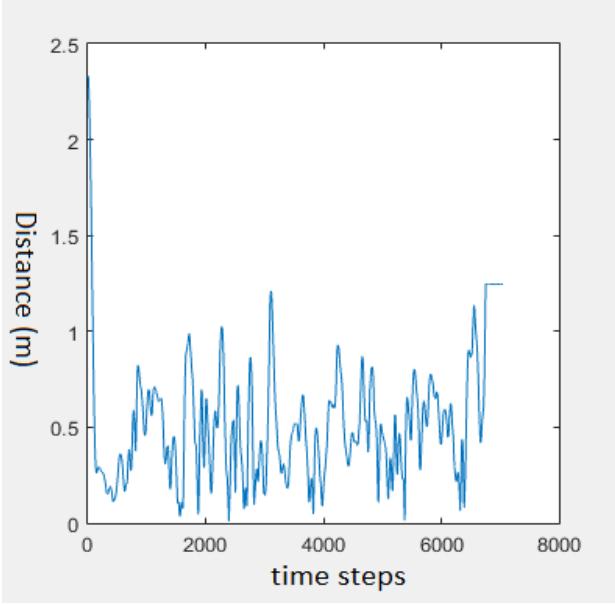


Fig. 5. Distance error between the UAV and the target before tuning.

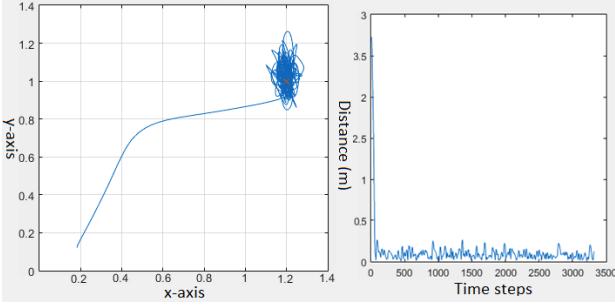


Fig. 6. Distance error between the UAV and the target after tuning.

frame, and should be transformed to the UAV's Body frame before feeding to the propellers controller as linear speed [18]. Figure 6 shows the result after tuning. The UAV is now able to remain inside a radius of  $d = 0.3\text{m}$  from the desired state. The exact values of these parameters will be provided in section VI. Algorithm 1 shows the PID + Q learning algorithm used in this paper.

## V. SIMULATION

In this section, we conducted a simulation on MATLAB environment to prove the navigation concept using RL. We defined our environment as a 5 by 5 board (Figure 7). Suppose that the altitude of the UAV was constant, it actually had 25 states, from  $(1, 1)$  to  $(5, 5)$ . The UAV was expected to navigate from starting position at  $(1, 1)$  to goal position at  $(5, 5)$  in shortest possible way. Each UAV can take four possible actions to navigate: forward, backward, go left, go right. The UAV will have a big positive reward of  $+100$  if it reaches the goal position, otherwise it will take a negative reward (penalty) of  $-1$ . We chose a learning rate  $\alpha = 0.1$ , and discount rate  $\gamma = 0.9$ .

Figure 8 shows the result of our simulation on MATLAB. It took 39 episodes to train the UAV to find out the

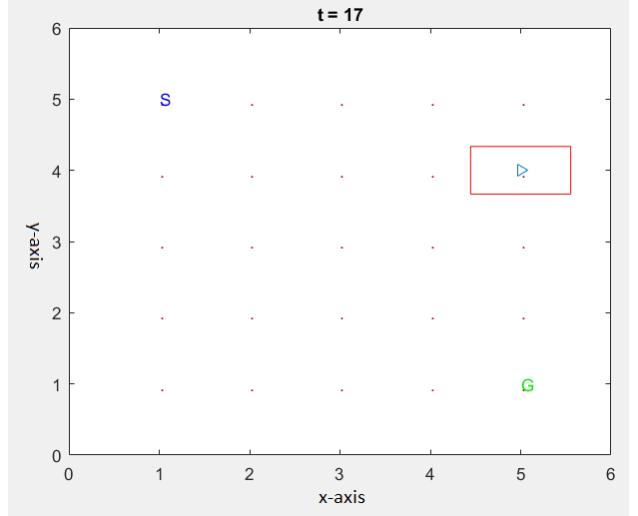


Fig. 7. The simulated environment at time step  $t = 17$ . Label S shows the original starting point, and Label G shows the goal.

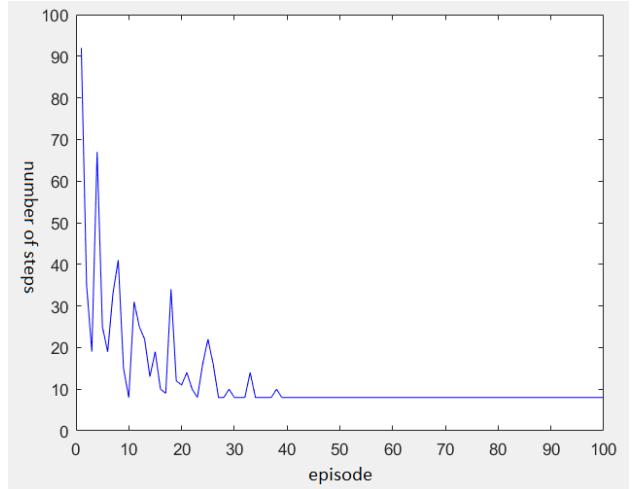


Fig. 8. The time steps taken in each episode of the simulation.

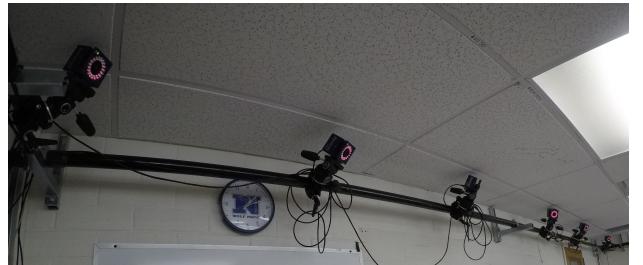


Fig. 9. Motion capture system from Motion Analysis.

optimal course of actions it should take to reach the target from a certain starting position. The optimal number of steps the UAV should take was 8 steps, resulting in reaching the target in shortest possible way.

## VI. IMPLEMENTATION

For our real-time implementation, we used a quadrotor Parrot AR Drone 2.0, and the Motion Capture System from Motion Analysis [24] installed in our Advanced

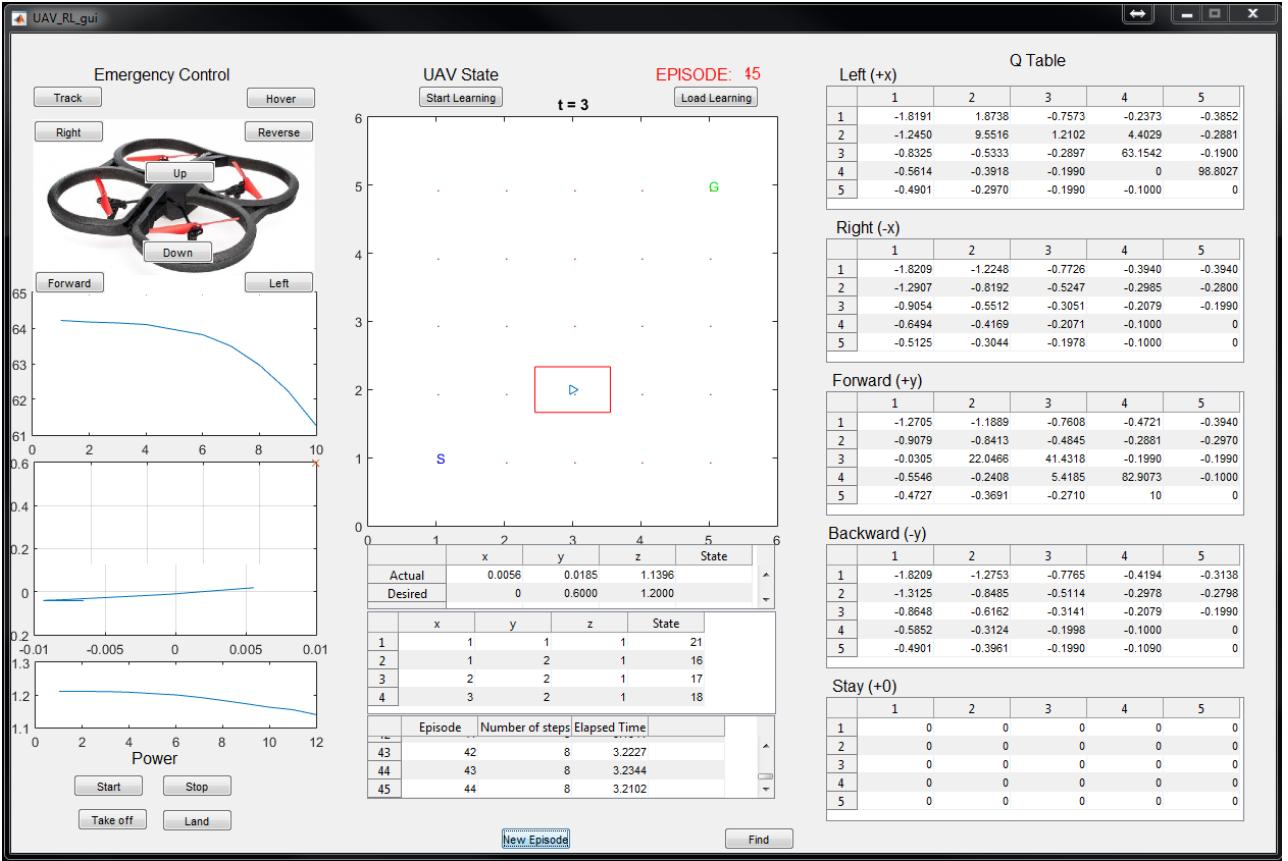


Fig. 10. MATLAB GUI for control the learning process of the UAV.

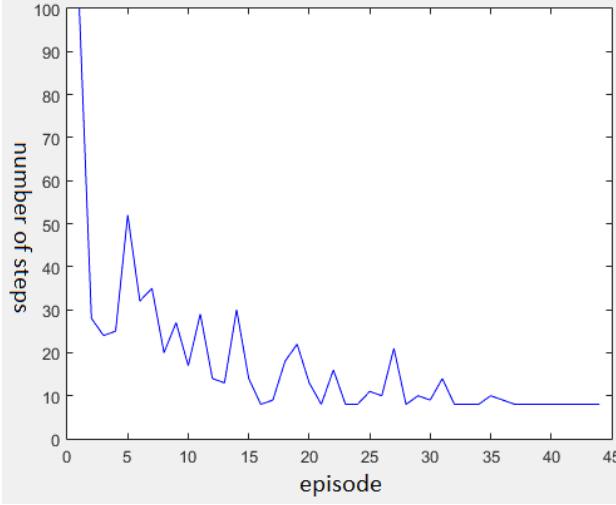


Fig. 11. Number of steps in each episode in real implementation.

Robotics and Automation (ARA) lab (Figure 9). The UAV could be controlled by altering the linear/angular speed, and the motion capture system provides the UAV's relative position inside the room. To carry out the algorithm, the UAV should be able to transit from one state to another, and stay there before taking new action. We implemented the PID controller in section IV to help the UAV carry out its action. Obviously, the learning process

was a lengthy one. Therefore, to overcome the physical constraint on UAV's battery life cycle, we also designed a GUI on MATLAB to help discretize the learning process into episodes (Figure 10). For better control of the learning progress, the GUI shows information of the current position of the UAV within the environment, the steps the UAV has taken, the current values of Q table, and the result of this episode comparing to previous episodes. It also helped to save the data in case a UAV failure happened, allowing us to continue the learning progress after the disruption.

We carried out the experiment using identical parameters to the simulation. The UAV operated in a closed room, which is discretized as a 5 by 5 board. It did not have any knowledge of the environment, except that it knew when the goal is reached. Given that the altitude of the UAV was kept constant, the environment actually has 25 states. The objective for the UAV was to start from a starting position at (1, 1) and navigate successfully to the goal state (5, 5) in shortest way. Similar to the simulation, the UAV will have a big positive reward of +100 if it reaches the goal position, otherwise it will take a negative reward (penalty) of -1. For the learning part, we selected a learning rate  $\alpha = 0.1$ , and discount rate  $\gamma = 0.9$ . For the UAV's PID controller, the proportional gain  $K_p = 0.8$ , derivative gain  $K_d = 0.9$ , and integral gain  $K_i = 0$ . Similar to our simulation, it took the UAV 38 episodes to find out the optimal course of actions (8 steps) to reach

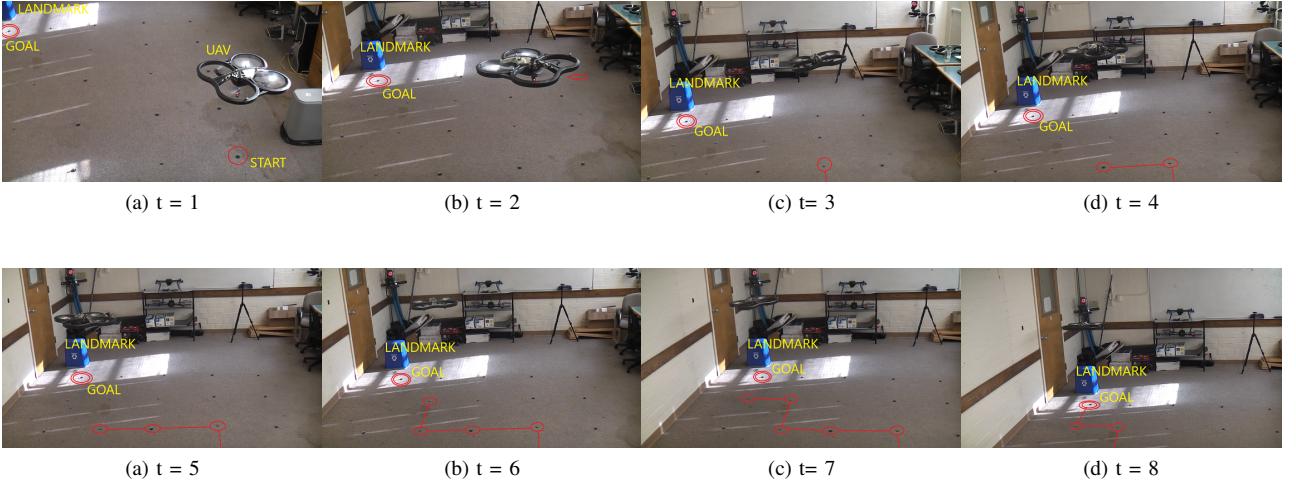


Fig. 12. Trajectory of the UAV during the last episode. It shows that the UAV reaches the target in the shortest possible way.

to the goal from a certain starting position (Figure 11). The difference between the first episode and the last ones was obvious: it took 100 steps for the UAV to reach the target in the first one, while it took only 8 steps in the last ones. Figure 12 shows the optimal trajectory of the UAV during the last episode.

## VII. CONCLUSION

This paper presented a technique to train a quadrotor to learn to navigate to the target point using a PID+Q-learning algorithm in an unknown environment. The implementation and simulation outputs similar result, showed that the UAVs can successfully learn to navigate through the environment without the need of a mathematical model. This paper can serve as a simple framework for using RL to enable UAVs to work in an environment where its model is unavailable. For real-world deployment, we should consider stochastic learning model where uncertainties, such as wind and other dynamics of the environment, present in the system [4], [25]. In the future, we will also continue to work on using UAV with learning capabilities in more important application, such as wildfire monitoring, or search and rescue missions. The research can be extended into multi-agent systems [26], [27], where the learning capabilities can help the UAVs to have better coordination and effectiveness in solving real-world problem.

## REFERENCES

- [1] H. X. Pham, H. M. La, D. Feil-Seifer, and M. Deans, “A distributed control framework for a team of unmanned aerial vehicles for dynamic wildfire tracking,” *arXiv preprint arXiv:1704.02630*, 2017.
- [2] A. C. Woods, H. M. La, and Q. P. Ha, “A novel extended potential field controller for use on aerial robots,” in *Automation Science and Engineering (CASE), 2016 IEEE International Conference on*. IEEE, 2016, pp. 286–291.
- [3] A. C. Woods and H. M. La, “Dynamic target tracking and obstacle avoidance using a drone,” in *International Symposium on Visual Computing*. Springer, 2015, pp. 857–866.
- [4] F. Muñoz, E. Quesada, E. Steed, H. M. La, S. Salazar, S. Commuri, and L. R. Garcia Carrillo, “Adaptive consensus algorithms for real-time operation of multi-agent systems affected by switching network events,” *International Journal of Robust and Nonlinear Control*, vol. 27, no. 9, pp. 1566–1588, 2017.
- [5] T. Tomic, K. Schmid, P. Lutz, A. Domel, M. Kassecker, E. Mair, I. L. Grix, F. Ruess, M. Suppa, and D. Burschka, “Toward a fully autonomous uav: Research platform for indoor and outdoor urban search and rescue,” *IEEE robotics & automation magazine*, vol. 19, no. 3, pp. 46–56, 2012.
- [6] H. M. La, “Multi-robot swarm for cooperative scalar field mapping,” *Handbook of Research on Design, Control, and Modeling of Swarm Robotics*, p. 383, 2015.
- [7] H. M. La, W. Sheng, and J. Chen, “Cooperative and active sensing in mobile sensor networks for scalar field mapping,” *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 45, no. 1, pp. 1–12, 2015.
- [8] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press Cambridge, 1998, vol. 1, no. 1.
- [9] H. M. La, R. Lim, and W. Sheng, “Multirobot cooperative learning for predator avoidance,” *IEEE Transactions on Control Systems Technology*, vol. 23, no. 1, pp. 52–63, 2015.
- [10] H. M. La, R. S. Lim, W. Sheng, and J. Chen, “Cooperative flocking and learning in multi-robot systems for predator avoidance,” in *Cyber Technology in Automation, Control and Intelligent Systems (CYBER), 2013 IEEE 3rd Annual International Conference on*. IEEE, 2013, pp. 337–342.
- [11] A. Faust, I. Palunko, P. Cruz, R. Fierro, and L. Tapia, “Learning swing-free trajectories for uavs with a suspended load,” in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE, 2013, pp. 4902–4909.
- [12] H. Bou-Ammar, H. Voos, and W. Ertel, “Controller design for quadrotor uavs using reinforcement learning,” in *Control Applications (CCA), 2010 IEEE International Conference on*. IEEE, 2010, pp. 2130–2135.
- [13] S. R. B. dos Santos, C. L. Nascimento, and S. N. Givigi, “Design of attitude and path tracking controllers for quad-rotor robots using reinforcement learning,” in *Aerospace Conference, 2012 IEEE*. IEEE, 2012, pp. 1–16.
- [14] S. L. Waslander, G. M. Hoffmann, J. S. Jang, and C. J. Tomlin, “Multi-agent quadrotor testbed control design: Integral sliding mode vs. reinforcement learning,” in *Intelligent Robots and Systems, 2005.(IROS 2005). 2005 IEEE/RSJ International Conference on*. IEEE, 2005, pp. 3712–3717.
- [15] N. Imanberdiyev, C. Fu, E. Kayacan, and I.-M. Chen, “Autonomous navigation of uav by using real-time model-based reinforcement learning,” in *Control, Automation, Robotics and Vision (ICARCV), 2016 14th International Conference on*. IEEE, 2016, pp. 1–6.
- [16] B. Zhang, Z. Mao, W. Liu, and J. Liu, “Geometric reinforcement learning for path planning of uavs,” *Journal of Intelligent & Robotic Systems*, vol. 77, no. 2, pp. 391–409, 2015.

- [17] S. S. Ge and Y. J. Cui, "Dynamic motion planning for mobile robots using potential field method," *Autonomous robots*, vol. 13, no. 3, pp. 207–222, 2002.
- [18] A. C. Woods and H. M. La, "A novel potential field controller for use on aerial robots," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2017.
- [19] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, no. 3-4, pp. 279–292, 1992.
- [20] L. Busoniu, R. Babuska, B. De Schutter, and D. Ernst, *Reinforcement learning and dynamic programming using function approximators*. CRC press, 2010, vol. 39.
- [21] R. C. Dorf and R. H. Bishop, *Modern control systems*. Pearson, 2011.
- [22] J. Li and Y. Li, "Dynamic analysis and pid control for a quadrotor," in *Mechatronics and Automation (ICMA), 2011 International Conference on*. IEEE, 2011, pp. 573–578.
- [23] K. U. Lee, H. S. Kim, J. B. Park, and Y. H. Choi, "Hovering control of a quadrotor," in *Control, Automation and Systems (ICCAS), 2012 12th International Conference on*. IEEE, 2012, pp. 162–167.
- [24] "Motion analysis corporation." [Online]. Available: <https://www.motionanalysis.com/>
- [25] H. M. La and W. Sheng, "Flocking control of multiple agents in noisy environments," in *2010 IEEE International Conference on Robotics and Automation*, May 2010, pp. 4964–4969.
- [26] ——, "Dynamic target tracking and observing in a mobile sensor network," *Robotics and Autonomous Systems*, vol. 60, no. 7, pp. 996 – 1009, 2012. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0921889012000565>
- [27] ——, "Distributed sensor fusion for scalar field mapping using mobile sensor networks," *IEEE Transactions on Cybernetics*, vol. 43, no. 2, pp. 766–778, April 2013.