

Reinforcement Learning for Autonomous UAV Navigation Using Function Approximation

Huy Xuan Pham, Hung Manh La, *Senior Member, IEEE*, David Feil-Seifer, and Luan Van Nguyen

Abstract—Unmanned aerial vehicles (UAV) are commonly used for search and rescue missions in unknown environments, where an exact mathematical model of the environment may not be available. This paper proposes a framework for the UAV to locate a missing human after a natural disaster in such environment, using a reinforcement learning (RL) algorithm. A function approximation based RL algorithm is proposed to deal with a large number of states representation and to obtain a faster convergence time. We conducted both simulated and real implementations to show how the UAVs can successfully learn to carry out the task without colliding with obstacles. Technical aspects for applying RL algorithm to a UAV system and UAV flight control were also addressed.

I. INTRODUCTION

Using unmanned aerial vehicles (UAV), or drones, to navigating through unknown environment is becoming more widespread, as they can carry a wide range of sensors with relative low operation costs and high flexibility, be able to operate in the environments are normally inaccessible, or pose hazardous risks for human rescuers [1]. Their capabilities have been demonstrated in Search and rescue (SAR) missions, such as urban SAR [2], in marine SAR [3], or in other disasters control such as wildfire monitoring [4], [5]. One issue is that most current research relies on the accuracy of the model, for example, exact location or trajectory of the target and obstacles, 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. Reinforcement learning (RL) could help overcome this issue by allowing a UAV or a team of UAVs to learn and navigate through the changing environment without the need for modeling [8].

RL algorithms have already been extensively researched in UAV applications, as in many other fields of robotics [9], [10]. Several 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 a 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 adverse 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 with improving RL performance in UAV applications. 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.

To the best of our knowledge, using a RL algorithm for a UAV's navigation and path planning in SAR context is still an open area of research. Many works 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 a simple SAR task, that is to find a missing immobile human, in an unknown environment. Using an effective RL algorithm with function approximation, the drone can learn to find the human's location from an arbitrary starting position in shortest possible way, without colliding with any obstacle. The proposed algorithm also allows scalability in implementation in realistic context by using function approximation to reduce the convergence time and the size of the state space problem. The main contribution of the paper is to provide a framework for applying a RL algorithm to enable UAV to operate in unknown environment.

The remainder of the paper is organized as follows. Section II provides more detail on problem formulation, and the approach we use to solve the problem. Basic knowledge in RL and Q-learning is provided in section III. Section VI discusses the issue of scalability in RL and the use of function approximation technique. The detailed learning algorithm for quadrotors is proposed in section VII. We conduct a simulation of our problem on section VI, and provide a comprehensive implementation of the algorithm in section VII. Finally, we conclude our paper and provide future work in section VIII.

This material is based upon work supported by the National Aeronautics and Space Administration (NASA) Grant No. NNX15AI02H issued through the Nevada NASA Research Infrastructure Development Seed Grant, and the National Science Foundation (NSF) #IIS-1528137. The views, opinions, findings and conclusions reflected in this publication are solely those of the authors and do not represent the official policy or position of the NASA and NSF.

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 (e-mail: hla@unr.edu).

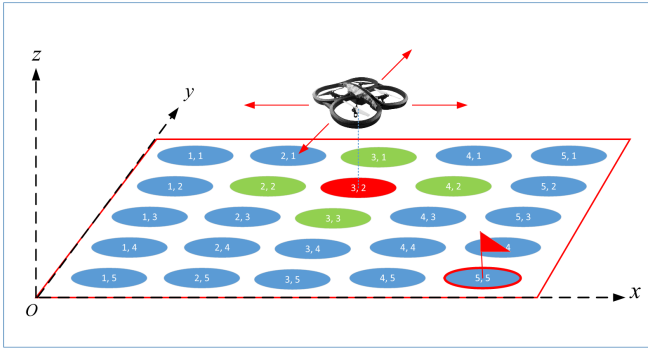


Fig. 1. A UAV navigating in bounded environment with discretized state space, represented by discrete circles. The red circle is the UAV's current state, the green circles are the options that the UAV can choose in the next iteration. The unknown goal position is marked by a red flag.

II. PROBLEM FORMULATION

This paper considers a SAR operation that is to find an immobile human in a bounded environment after a natural disaster, which is full of debris and unknown obstacles. By assuming the missing human is immobile, we suppose that the human location is static over time. A quadcopter-type UAV, equipped with Ultra Wide Band (UWB) radar that can detect humans underneath rubble, tasked to safely navigate in the environment to find this person (Figure 1). The radar can measure the distance between the human being and the robot [17]. The UAV also has on-board laser scanners to detect obstacles (debris, wall, border, ...). We assume the UAV can localize itself in the environment and the system is fully observable. If we have full information about the environment, for instance, the exact shapes and locations of the obstacles, a robot motion planning can be constructed based on the model of the environment, and the problem becomes very common. Traditional control methods, such as potential field [18], [19], are available to solve such problem, but they normally requires prior knowledge of obstacles' shapes and locations. In many realistic cases, however, building such model is not possible because the data of the environment are not available or difficult to obtain. Another problem with using the potential field method is that the robot/drone can get trapped in local maxima/minima, for example with concave obstacles (i.e, U shape obstacle). Work such as [20] uses vector fields to overcome this issues, but again, requires the knowledge of obstacles' locations. Since RL algorithms can rely only on the data obtained directly from the system, it is naturally a solution option for our problem. In the learning process, the agent can map the situations it faces to appropriate actions to achieve its goal in an optimal fashion.

III. REINFORCEMENT LEARNING AND Q LEARNING

RL has become popular recently, thanks to its capability in solving learning problems without relying on a model of the environment. An agent builds up its knowledge of the surrounding environment by accumulating its experience through interacting with the environment. The agent seeks to

maximize a numerical signal, called reward, that measures the performance of the agent. Assuming that the environment has Markov property, where the next state and reward of an agent only depend on the current state [8]. Because the environment is fully observable, the system is a Markov Decision Process generalized as a tuple $\langle S, A, T, R \rangle$, where S is a finite set of states, and $s_k \in S$ is the state of the agent at time step k . A is a finite set of actions, and $a_k \in A$ is the action the agent takes at time 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 [21], in which the agent computes optimal value function and records them into a tabular database, called Q-table. This knowledge of choosing an action in a given state, can be used for the agent to 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 ϵ greedy, with $0 < \epsilon < 1$, as follows:

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

To guarantee the convergence of the Q-learning algorithm, in practice, normally these state and action set, S and A , will be represented approximately [22] since the continuous space is too large. This paper assumed the environment as a discrete grid (Figure 1). To keep the problem simple, in this paper we will consider the UAV operated at a constant altitude. 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. The absolute position of the drone can be used to define its state in the system, however, it would make the solution highly depends on the environment, and risk over-fitting

in the learning process. Therefore, we defined the state in a more general way as: $s_k \triangleq \{d, o^N, o^S, o^E, o^W\} \in S$, where d is the relative distance from the UAV and the target, measured by the on-board radar. This distance is discretized to have a finite number of rounded values. o^N, o^S, o^E and o^W are the distances to the nearest obstacle in North, West, South or East direction, respectively, that are detected by the on-board laser scanners. Because the laser scanners have limited range, these distances can be discretized to have one of four possible value: $\{0\}$ if the UAV is on the surface of an obstacle; $\{1\}$ if it is close to the obstacle, but within a predefined safe distance for the UAV to pass through; $\{2\}$ if there is a far obstacle detected; and $\{3\}$ if the obstacle is not detectable or out of range.

If the UAV reached the search object (i.e., a person), identified by pre-described information, at unknown location G , it will get a big reward. Reaching other places that is not the desired goal will result in a small penalty (negative reward). If the UAV collides with debris or obstacles X , it will get a big penalty. In summary, the reward is defined as follows:

$$r_{k+1} = \begin{cases} 100, & \text{if } s_{k+1} \equiv G \\ -10, & \text{if } s_{k+1} \equiv X \\ -1, & \text{otherwise.} \end{cases} \quad (3)$$

IV. Q-LEARNING WITH FUNCTION APPROXIMATION

For Q-learning algorithm to guarantee correct convergence, one must make sure that all state - action pairs continue to be updated [8]. It would be a serious problem if the dimension of the Q table grew, since it would exponentially increase the time needed for the RL algorithm to converge. Additionally, the space needed to store each state-action pair's values also puts pressure on the physical hardware constraints of the agent, like memory and hard drive space. Therefore, to allow scalable implementation in realistic environment, the size of the Q-table need to be reduced while still representing the distinct value for each state - action pair. This could be done by using function approximation techniques to approximate the state - action value function (Q-function). In this work, we employ a technique called Fixed Sparse Representation (FSR) approximation to map the original Q table to a parameter vector θ [23]:

$$\begin{aligned} \hat{Q}_k(s_k, a_k) &= \sum_{l=1}^n \phi_l(s_k, a_k) \theta_l \\ &= \phi^T(s_k, a_k) \theta, \end{aligned} \quad (4)$$

where $\phi : S \times A \rightarrow \mathbb{R}$ is state and action - dependent basis function vector with n elements. Each element is defined by:

$$\phi_l(s_k, a_k) = \begin{cases} 1, & \text{if } s_k = S_i, a_k = A_j; S_i \in S; A_j \in A \\ 0, & \text{otherwise.} \end{cases} \quad (5)$$

Other approximation technique based on Radial Basis Function (RBF) can be used also. A comparison between FSR and RBF can be founded in our previous work [24]. To illustrate the space saved using this technique, let D and

O be the discretized sets of distance to the target and to the obstacles, respectively, and let $|\cdot|$ denote the number of element of a set. The original Q-value table requires the size of $|S| \cdot |A| = (|D| \cdot |O|^4) |A|$. If we approximated the Q-table as in equation (4) and (5), both $\phi(s, a)$ and θ are column vectors of the size $(|D| + 4|O|) |A|$, which is much less than the space required in the original one. The saving ratio, that is $\frac{|D| + 4|O|}{|D| \cdot |O|^4}$, will grow as the state space grows.

After approximation, the update rule in (1) for Q-function becomes the update rule for the parameter [22]:

$$\begin{aligned} \theta_{k+1} &\leftarrow \theta_k + \alpha [r_{k+1} + \gamma \max_{a' \in A} \phi^T(s_{k+1}, a') \theta_k \\ &\quad - \phi^T(s_k, a_k) \theta_k] \phi(s_k, a_k). \end{aligned} \quad (6)$$

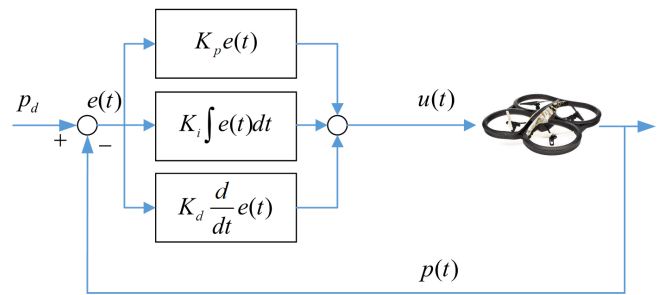


Fig. 2. The PID control diagram with 3 components: proportional, integral and derivative terms.

V. 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} within allowed error e . Let $p(t)$ denote the real-time position of the UAV at real time t , that can be estimated by the heading action and the difference between distance d at current state s_k and desired state s_{k+1} . We start with a simple proportional gain controller:

$$u(t) = K_p(p(t) - s_{k+1}) = K_p e(t), \quad (7)$$

where $u(t)$ is the control input, K_p is the proportional control gain, and $e(t)$ is the tracking error between the real-time position $p(t)$ and desired location s_{k+1} . Because of the nonlinear dynamics of the quadrotor [19], we experienced excessive overshoots of the UAV when it navigates from one state to another (Figure 3-left), making the UAV unstable after reaching a state. To overcome this, we used a standard PID controller [25] (Figure 2). Although the controller cannot effectively regulate the nonlinearity of the system, work such as [26], [27] indicated that using PID controller could still yield relatively good stabilization during hovering.

$$u(t) = K_p e(t) + K_i \int e(t) dt + K_d \frac{de}{dt}. \quad (8)$$

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 eliminating the Integral component of the PID control to achieve stable trajectory. Note that $u(t)$ is calculated in the Inertial frame, and should be transformed to the UAV's Body frame before feeding to the propellers controller as linear speed [19]. Figure 3-right shows the result after tuning. Algorithm 1 shows the PID + Approximated Q learning algorithm used in this paper.

Algorithm 1: PID + APPROXIMATED Q-LEARNING.

Input: Learning parameters: Discount factor γ , learning rate α , number of episode N

Input: Control parameters: Control gains K_p, K_i, K_d

- 1 Initialize $\theta_0(s_0, a_0) \leftarrow 0, \forall s_0 \in S, \forall a_0 \in A$;
- 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** $\|p(t) - s_{k+1}\| \leq e$
- 10 Observe immediate reward r_{k+1}
- 11 Estimate $\phi(s_k, a_k)$ based on equation (5)
- 12 Update:

$$\theta_{k+1} \leftarrow \theta_k + \alpha [r_{k+1} + \gamma \max_{a' \in A} (\phi^T(s_{k+1}, a') \theta_k) - \phi^T(s_k, a_k) \theta_k] \phi(s_k, a_k).$$
- 13 **until** $s_{k+1} \equiv G$

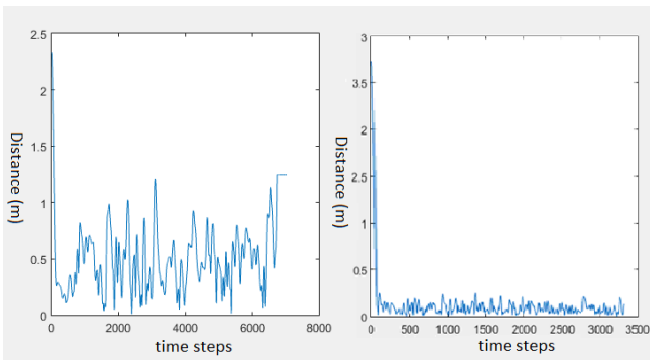


Fig. 3. Distance error between the UAV and the target without using a PID controller (left) and with using a PID controller (right).

VI. SIMULATION

In this section, we conducted a simulation in a MATLAB environment to prove the navigation concept using RL, and to compare the convergence speeds between original Q-learning and approximated Q-learning algorithm in section . We defined our environment as a discrete 10 by 10 board, that

had debris and obstacles at unknown locations to the UAV, marked by rectangles with a X mark (Figure 4). We presume that the altitude of the UAV was constant. 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 missing human. If it collides with an obstacle or the environment boundary, it will get back to previous step, and receive big penalty of -10, otherwise it will take a mild negative reward (penalty) of -1. We chose a learning rate $\alpha = 0.1$, and discount rate $\gamma = 0.9$.

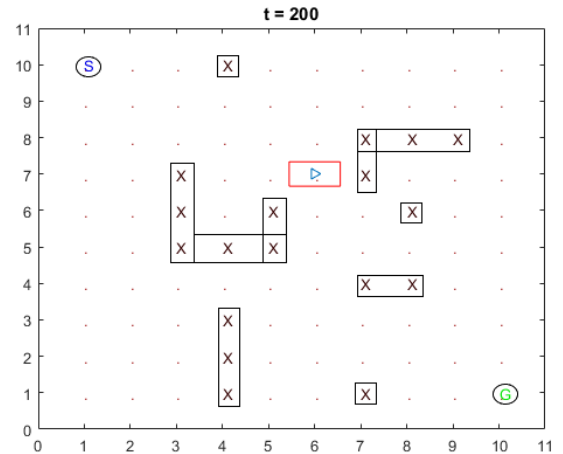
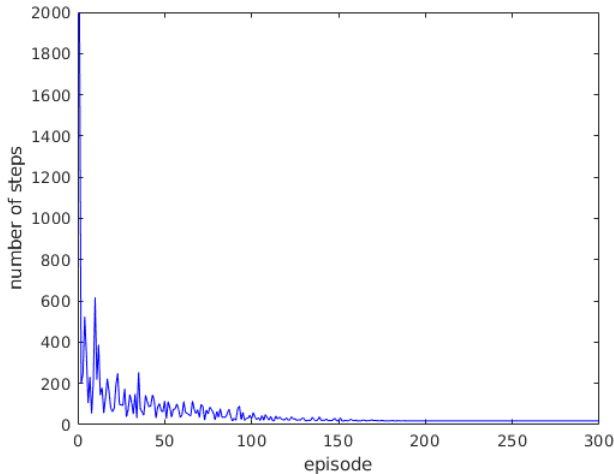


Fig. 4. The simulated environment at time step $t = 200$. Label S shows the original starting point of the UAV, and label G shows the unknown position of the missing human. The UAV is denoted by a triangle with its field of view denoted by a red rectangle. The obstacles in the environment are denoted by rectangles with a X mark.

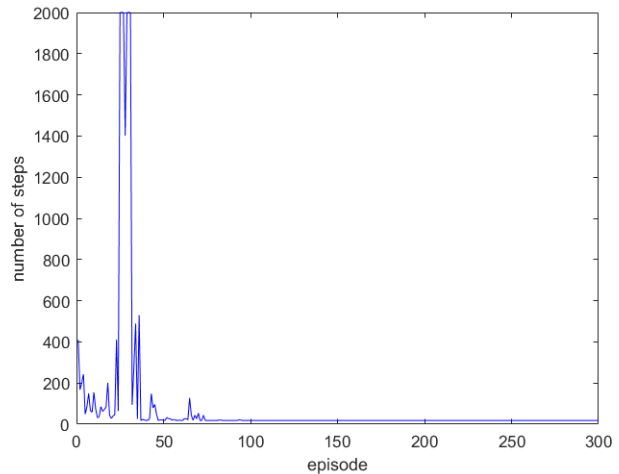
VII. IMPLEMENTATION

Figure 5 shows the result of our simulation on MATLAB for (a) the original Q-learning algorithm and (b) the approximated Q-learning algorithm. It took about 160 episodes for the normal Q-learning to train the UAV to find out the optimal course of actions that it should take to reach the missing human without colliding with any obstacle, while it only took 75 episodes using the approximated Q-learning algorithm. It is obvious that the reduction in space, therefore, in number of the state-action pairs in approximated Q-learning algorithm leads to the reduction of convergence speed. In both algorithms, the optimal number of steps that the UAV should take was 18 steps, resulting in reaching the target in shortest possible way. The difference between the first episode and the last ones was obvious: it took up to 2000 steps for the UAV to reach the target in early episodes, while it took only 18 steps in the later ones.

For the real-time implementation, we used a quadrotor Parrot AR Drone 2.0, and the Motion Capture System from Motion Analysis [28] installed in our Advanced Robotics and Automation (ARA) lab. 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



(a) Normal Q-learning



(b) Approximated Q-learning

Fig. 5. The time steps taken in each episode of the simulation with (a) normal Q-learning algorithm, and (b) Approximated Q-learning algorithm. The convergence speed in (b) is greatly reduced, thanks to the approximation technique.

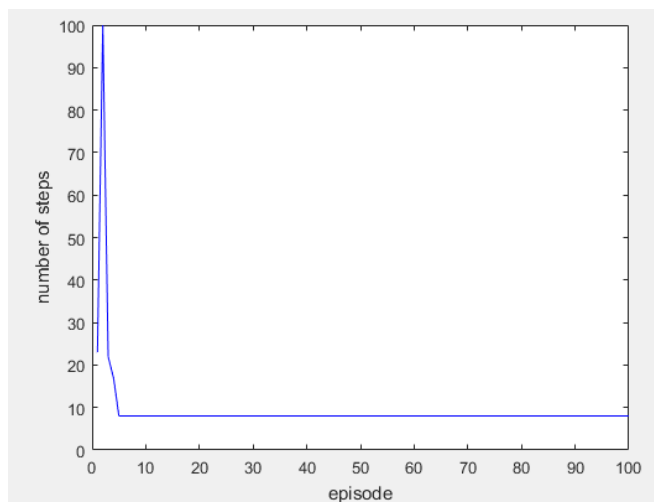


Fig. 6. The time steps taken by the UAV in each episode of the implementation. The learning converges after 5 episodes.

action. We implemented the PID controller in section to help the UAV carry out its action.

We carried out the experiment using Algorithm 1 with 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. 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 human position, or receive a penalty if it hits the boundaries or obstacles, 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 5 episodes to

find out the optimal course of actions (8 steps) to reach to the goal from a certain starting position (Figure 6). Figure 7 shows the optimal trajectory of the UAV during the last episode.

VIII. CONCLUSION

This paper presented a technique to train a quadrotor to learn to navigate to the target point using a PID+ Approximated Q-learning algorithm in an unknown environment. The real-world implementation and simulation outputs similar result, and showed that the UAVs can successfully learn to navigate through the environment without the need for a mathematical model. It also noted that the application of function approximation technique greatly reduced the speed of convergence of the algorithm, and the size required to implement the original problem, thus it could be scaled to implement in more realistic environment. 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 [29], [30]. In the future, we will also continue to work on using UAV with learning capabilities in more important application in SAR and natural disaster relief. The research can be extended into multi-agent systems [31], [32], where the learning capabilities can help the UAVs to have better coordination and effectiveness in solving real-world problem.

REFERENCES

- [1] Y. Liu and G. Nejat, "Robotic urban search and rescue: A survey from the control perspective," *Journal of Intelligent & Robotic Systems*, vol. 72, no. 2, pp. 147–165, 2013.
- [2] 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.

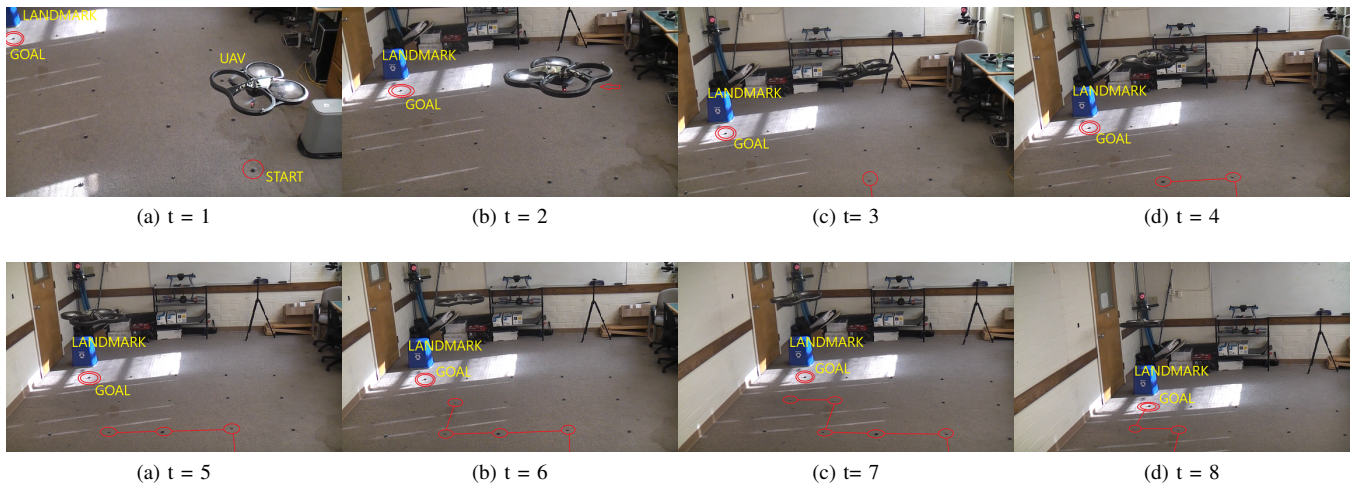


Fig. 7. Trajectory of the UAV during the last episode. It shows that the UAV reaches the missing human in the shortest possible way.

- [3] S. Yeong, L. King, and S. Dol, "A review on marine search and rescue operations using unmanned aerial vehicles," *Int. J. Mech. Aerosp. Ind. Mech. Manuf. Eng.*, vol. 9, no. 2, pp. 396–399, 2015.
- [4] H. X. Pham, H. M. La, D. Feil-Seifer, and M. C. Deans, "A distributed control framework for multiple unmanned aerial vehicles for dynamic wildfire tracking," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, pp. 1–12, 2018.
- [5] 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," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sept 2017, pp. 6648–6653.
- [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] J. Rovňáková and D. Kocur, "Toa association for handheld uwb radar," in *Radar Symposium (IRS), 2010 11th International*. IEEE, 2010, pp. 1–4.
- [18] 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.
- [19] 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.
- [20] A. D. Dang, H. M. La, and J. Horn, "Distributed formation control for autonomous robots following desired shapes in noisy environment," in *Multisensor Fusion and Integration for Intelligent Systems (MFI), 2016 IEEE International Conference on*. IEEE, 2016, pp. 285–290.
- [21] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, no. 3-4, pp. 279–292, 1992.
- [22] L. Busoniu, R. Babuska, B. De Schutter, and D. Ernst, *Reinforcement learning and dynamic programming using function approximators*. CRC press, 2010, vol. 39.
- [23] A. Geramifard, T. J. Walsh, S. Tellex, G. Chowdhary, N. Roy, J. P. How, et al., "A tutorial on linear function approximators for dynamic programming and reinforcement learning," *Foundations and Trends® in Machine Learning*, vol. 6, no. 4, pp. 375–451, 2013.
- [24] H. X. Pham, H. M. La, D. Feil-Seifer, and L. V. Nguyen, "Performance comparison of function approximation-based q learning algorithms for autonomous uav navigation," in *The 15th International Conference on Ubiquitous Robots (UR)*, June 2018.
- [25] R. C. Dorf and R. H. Bishop, *Modern control systems*. Pearson, 2011.
- [26] 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.
- [27] 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.
- [28] "Motion analysis corporation." [Online]. Available: <https://www.motionanalysis.com/>
- [29] 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.
- [30] 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.
- [31] H. M. La and W. Sheng, "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>
- [32] —, "Distributed sensor fusion for scalar field mapping using mobile sensor networks," *IEEE Transactions on Cybernetics*, vol. 43, no. 2, pp. 766–778, April 2013.