Y6 Tricopter Autonomous Evacuation in an Indoor Environment Using Q-Learning Algorithm

Andriy Sarabakha*†, *Student Member*, *IEEE*, and Erdal Kayacan*, *Senior Member*, *IEEE*
*School of Mechanical & Aerospace Engineering, Nanyang Technological University, Singapore
†ST Engineering-NTU Corporate Laboratory, Nanyang Technological University, Singapore
E-mail: andriy001@e.ntu.edu.sg, erdal@ntu.edu.sg

***Abstract*— Emergency situations in large public and residential buildings, earthquake, fire, flood, terrorist attacks, cause extreme physical and emotional behaviours, inter alia, anxiety, hyperactivity, anger, etc.; in these situations, people are often unable to take the right action or even unable to make a decision. This paper addresses the problem of generating a building evacuation plan with the help of a Y6 coaxial tricopter UAV in an emergency situation where GPS signal is not available. The proposed algorithm, stochastic Q-Learning, learns the shortest path to leave the building. The traditional 2D space navigation is extended to the challengeable 3D space, which makes our approach more applicable in the real world. The emergency evacuation system proposed in herein can navigate people to evacuate a building safely in the wake of an emergency situation.**

I. Introduction

In the last decades, technological developments in sensor, actuator and microcontroller industries have paved the way from autopilots to unmanned aerial vehicles (UAVs) where the pilot is not needed anymore. Especially, in risky environments, it is appealing to be able to execute a mission in a fully autonomous way. Consequently, UAVs are gaining increasing interest because of their limitless military and commercial applications, e.g., disaster rescue [1], infrastructure inspection [2] and map reconstruction [3]. Regardless of the assigned task, the main aim of a UAV is to perform that task safely and efficiently. A critical task is considered in this paper, the autonomous evacuation, which refers to the capability of selecting and following a path from a current position to a safe final position with the help of a UAV. However, the target's location is often either invisible or unidentified. The problem of controlling UAVs in an unknown and uncertain environment is of essential importance in many applications [4].

Even if, as human-beings, our inventiveness can reach extreme levels and we have highly capable computers, nature still has the best source of engineering designs. For example, it is no coincidence that the nose of an aeroplane is very similar to that of a dolphin. Not only the physical appearance but also the behaviour of animals is opening the doors for new theories. Just like the learning process in nature, reinforcement learning (RL) is based on the past experiences. Therefore, RL is a learning technique based on the common sense concept: if an action is performed an adequate number of times, then the tendency to remember the consequence of that action is reinforced. In RL, the learner is a decision-making agent who makes observations, takes actions and receives rewards or penalties. Besides, RL approach relies on the theory of Markov decision processes (MDP) [5].

In this paper, we focus our attention on a special RL algorithm called Q-Learning that can learn the optimal policy from delayed rewards. Q-Learning algorithm has numerous successful applications, e.g., game learning [6], optimisation problems [7] and financial applications [8]. Moreover, unlike common model-based approaches, Q-Learning does not require any information about the model or the environment [9]. This is the reason why Q-Learning is suitable for the cases which are difficult to determine a priori, complex to program or impossible to use in time-varying environments. In light of its aforementioned capabilities, Q-Learning is an appropriate choice for the high-level UAV controller.

In this investigation, the agent for Q-Learning is a Y6 coaxial tricopter vertical take-off and landing (VTOL) UAV. Y6 configuration has some advantages: for instance, if one motor or electronic speed controller fails, the system can still land safely. On the other hand, tricopters have a significant disadvantage such as having highly nonlinear dynamics. In spite of this disadvantage, Y6 coaxial tricopter is chosen for this implementation since the proposed Q-Learning algorithm compensates the aforementioned disadvantages while benefiting from the safe structure of the system.

To the best of our knowledge, this is the first time in the literature that the Q-Learning is used for the UAV's evacuation navigation. Our proposed method provides the optimal path selection for a tricopter-like aircraft in an indoor unknown grid environment. The emergency evacuation system proposed in herein can navigate people to evacuate a building safely in the wake of an emergency situation. The algorithm takes state transition and reward functions as an input and then learns the shortest path, made up of a set of actions, to leave the building. In addition, the UAV learns the map of the environment.

This paper is structured as follows. Section II briefly reviews MDP and Q-Learning algorithm. In Section III we present the Y6 tricopter's dynamic model and the control scheme for the velocity tracking. Section IV describes the evacuation problem and provides dynamical simulations for evaluating the efficiency of our solution. Finally, Section V closes this paper with conclusions and future works.

II. Markov Decision Process and Q-Learning

A. Markov Decision Process

Given a set of states S, a set of actions A and a set of rewards R, a controlled process with dynamics of Markov is defined as follows:

$$\begin{cases} s_{t+1} & = \delta\left(s_t, a_t\right) \\ r_t & = \mathrm{r}\left(s_t, a_t\right) \end{cases}, t = 0, 1, \ldots, \tag{1}$$

where $s_t \in S$ is the state of the agent at time t, $a_t \in A$ is the action taken by the agent in the state s_t and $r_t \in R$ is the reward observed after performing the action a_t in the state s_t. The goal is to maximise the expected total discounted reward starting from the initial state s_0:

$$\max_{s \in S} \mathrm{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t s_0 = s\right], \tag{2}$$

where $\gamma \in [0, 1]$ is the discount factor. The process $(s_t, a_t, r_t; t \geq 1)$ is called Markov decision process (MDP).

Let $\mathrm{V}^{\pi}(s)$ denotes the value of state s under a policy π, and it is the expected return given that the policy is started in state s. Consequently, the optimal value of a state is the value of the best possible expected return that can be obtained from that state and it is denoted by $\mathrm{V}^*(s) = \sup_{\pi} \mathrm{V}^{\pi}(s)$.

B. Q-Learning Algorithm

A particular model-free RL technique, Q-Learning, can be used to compute an optimal policy for an MDP (1). It learns a state-action function $\mathrm{Q}(s, a)$ which gives the expected desirability of performing an action a in a state s. Once Q function is learned, the optimal policy $\pi^*(s)$ can be computed by simply choosing the action a with the highest value. Besides, Q-Learning always finds an optimal policy for MDP (1) [10]. In our work we use Q-Learning technique for non-deterministic cases to solve the evacuation problem.

Each time the agent performs an action $a \in A$ in its environment, it moves from state $s \in S$ to state $s' \in S$ and receives a reward $r \in R$ to indicate the desirability of the resulting state s'. The agent's goal is to maximise the total received reward. It achieves this by learning which action is optimal in every state. Therefore, the algorithm needs Q function which contains the desirability of a state-action combination:

$$\mathrm{Q} : S \times A \to R. \tag{3}$$

The basic Q-Learning scenario is shown in Fig. 1.

Initially, Q is set to 0 for all state-action pairs. Then, each time the agent performs an action, the Q function is updated. It takes the old value and makes an update based on the new information:

$$\mathrm{Q}_{t+1}\left(s_t, a_t\right) = \mathrm{Q}_t\left(s_t, a_t\right) + \alpha_t \left[r_t + \gamma_t \max_{a'} \mathrm{Q}_t\left(s_{t+1}, a'\right) - \mathrm{Q}_t\left(s_t, a_t\right)\right], \tag{4}$$

where $\alpha_t \in [0, 1]$ is the learning rate. An episode ends when state s_t is a final state. The pseudo-code of the Q-Learning is given in Algorithm 1.

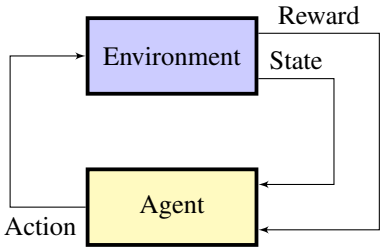


Fig. 1. The basic reinforcement learning scenario.

III. Y6 Tricopter Dynamics

A. Y6 Tricopter Model

Let the world fixed inertial reference frame be $\mathcal{F}_W = \{\vec{\mathbf{x}}_W, \vec{\mathbf{y}}_W, \vec{\mathbf{z}}_W\}$ and the body frame be $\mathcal{F}_B = \{\vec{\mathbf{x}}_B, \vec{\mathbf{y}}_B, \vec{\mathbf{z}}_B\}$. The origin of the body frame is located at the centre of mass (COM) of the tricopter. The tricopter configuration with reference frames is illustrated in Fig. 2. The six propellers generate six forces (f_1, f_2, f_3, f_4, f_5 and f_6), directed along $\vec{\mathbf{z}}_B$, and six torques (τ_1, τ_2, τ_3, τ_4, τ_5 and τ_6) with the module proportional to the speed of rotation.

The vector of control inputs is considered directly as [12]:

$$\mathbf{u} = \begin{bmatrix} T & \tau_\phi & \tau_\theta & \tau_\psi \end{bmatrix}^T, \tag{5}$$

where T is the total thrust directed along $\vec{\mathbf{z}}_B$ and applied to its center of mass, whereas τ_ϕ, τ_θ and τ_ψ are the three rotational torques acting around $\vec{\mathbf{x}}_B$, $\vec{\mathbf{y}}_B$ and $\vec{\mathbf{z}}_B$ axes, respectively. Under these considerations, the relation between $\mathbf{u}$ and ω_i, $i = 1, \ldots, 6$, becomes:

$$\begin{cases} T = f_1 + f_2 + f_3 + f_4 + f_5 + f_6 \\ \tau_\phi = l\left(-\sin\frac{\pi}{3}(f_3 + f_4) + \sin\frac{\pi}{3}(f_5 + f_6)\right) \\ \tau_\theta = l\left(f_1 + f_2 - \cos\frac{\pi}{3}(f_3 + f_4 + f_5 + f_6)\right) \\ \tau_\psi = \tau_1 - \tau_2 + \tau_3 - \tau_4 + \tau_5 - \tau_6, \end{cases} \tag{6}$$

where l is the arm length.

The absolute position of a tricopter is described by three Cartesian coordinates $\mathbf{p} = \begin{bmatrix} x & y & z \end{bmatrix}^T$ of its COM in $\mathcal{F}_W$

Algorithm 1 Q-Learning algorithm.

```
Initialize all Q(s, a) to 0
for all episodes do
    Choose a state s ∈ S
    while s ≠ terminal state do
        Choose an action a ∈ A
        s' ← δ(s, a)
        r ← r(s, a)
        Q(s, a) ← Q(s, a) − α[r + γ max_a' Q(s', a') − Q(s, a)]
        s ← s'
    end while
end for
for all states s do
    π(s) ← {a| max_a Q(s, a)}
end for
```

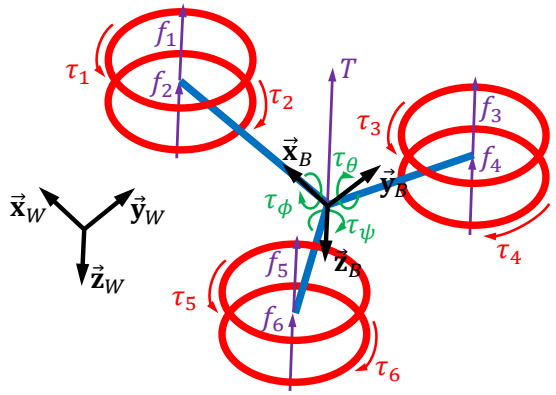


Fig. 2. Y6 tricopter model with reference frames.

and its attitude by the three Euler's angles $\mathbf{o} = \begin{bmatrix}\phi & \theta & \psi\end{bmatrix}^T$. These three angles are respectively called roll, pitch and yaw.

The time derivative of the position gives the absolute velocity $\mathbf{v} = \begin{bmatrix}\dot{x} & \dot{y} & \dot{z}\end{bmatrix}^T = \begin{bmatrix}v_x & v_y & v_z\end{bmatrix}^T$ of the tricopter's COM in $\mathcal{F}_W$. Let $\mathbf{v}_B \in \mathbb{R}^3$ be the absolute velocity of the tricopter in $\mathcal{F}_B$. Therefore, $\mathbf{v}$ and $\mathbf{v}_B$ are related by

$$\mathbf{v} = \mathbf{R}\mathbf{v}_B, \tag{7}$$

where $\mathbf{R} \in \mathsf{SO}(3)$ is the rotation matrix from $\mathcal{F}_B$ to $\mathcal{F}_W$:

$$\mathbf{R} = \begin{bmatrix} c_\psi c_\theta & c_\psi s_\phi s_\theta - c_\phi s_\psi & s_\phi s_\psi + c_\phi c_\psi s_\theta \\ c_\theta s_\psi & c_\phi c_\psi + s_\phi s_\psi s_\theta & c_\phi s_\psi s_\theta - c_\psi s_\phi \\ -s_\theta & c_\theta s_\phi & c_\phi c_\theta \end{bmatrix}, \tag{8}$$

where c_θ and s_θ denote respectively $\cos\theta$ and $\sin\theta$. The time derivative of the attitude gives the angular velocity $\boldsymbol{\omega} = \begin{bmatrix}\dot{\phi} & \dot{\theta} & \dot{\psi}\end{bmatrix}^T$ in $\mathcal{F}_W$ and the angular velocity $\boldsymbol{\omega}_B = \begin{bmatrix}\omega_\phi & \omega_\theta & \omega_\psi\end{bmatrix}^T$ in $\mathcal{F}_B$. The relation between $\boldsymbol{\omega}$ and $\boldsymbol{\omega}_B$ is

$$\boldsymbol{\omega} = \mathbf{T}\boldsymbol{\omega}_B, \tag{9}$$

in which $\mathbf{T}$ is the transformation matrix given by [11]

$$\mathbf{T} = \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi\sec\theta & \cos\phi\sec\theta \end{bmatrix}. \tag{10}$$

Using the Newton-Euler equations about the COM, the dynamic equations for the tricopter are the following [13]:

$$\begin{cases} m\dot{\mathbf{v}} = \mathbf{F}_e \\ \mathbf{I}\dot{\boldsymbol{\omega}}_B = -\boldsymbol{\omega}_B \times \mathbf{I}\boldsymbol{\omega}_B + \boldsymbol{\tau}_e, \end{cases} \tag{11}$$

where m is the mass, $\mathbf{I} = \mathrm{diag}(I_x, I_y, I_z)$ is the inertia matrix, $\boldsymbol{\tau}_e = \begin{bmatrix}\tau_\phi & \tau_\theta & \tau_\psi\end{bmatrix}^T$ is the vector of external torques and $\mathbf{F}_e$ is the vector of external forces [14]:

$$\mathbf{F}_e = \begin{bmatrix} -\left(\cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi\right)T \\ -\left(\cos\phi\sin\theta\sin\psi - \sin\phi\cos\psi\right)T \\ -\cos\phi\cos\theta T + mg \end{bmatrix}, \tag{12}$$

in which g is the gravity acceleration ($g = 9.81\mathrm{m/s}^2$). Finally, using dynamic and kinematic differential equations (7), (9) and (11), the tricopter model is obtained [15]

$$\begin{cases} \dot{x} = v_x & \dot{v}_x = -\frac{c_\phi c_\psi s_\theta + s_\phi s_\psi}{m}T \\ \dot{y} = v_y & \dot{v}_y = -\frac{c_\phi s_\psi s_\theta - c_\psi s_\phi}{m}T \\ \dot{z} = v_z & \dot{v}_z = -\frac{c_\phi c_\theta}{m}T + g \\ \dot{\phi} = \omega_\phi + s_\phi t_\theta \omega_\theta + c_\phi t_\theta \omega_\psi & \dot{\omega}_\phi = \frac{I_y - I_z}{I_x}\omega_\theta\omega_\psi + \frac{1}{I_x}\tau_\phi \\ \dot{\theta} = c_\phi\omega_\theta - s_\phi\omega_\psi & \dot{\omega}_\theta = \frac{I_z - I_x}{I_y}\omega_\phi\omega_\psi + \frac{1}{I_y}\tau_\theta \\ \dot{\psi} = \frac{s_\phi}{c_\theta}\omega_\theta + \frac{c_\phi}{c_\theta}\omega_\psi & \dot{\omega}_\psi = \frac{I_x - I_y}{I_z}\omega_\phi\omega_\theta + \frac{1}{I_z}\tau_\psi, \end{cases} \tag{13}$$

where t_θ denotes $\tan\theta$.

B. Control Scheme

For the velocity tracking, we use the nonlinear geometric controller on the special Euclidean group $\mathsf{SE}(3)$ [16]. If $\mathbf{v}^* = \begin{bmatrix}v_x^* & v_y^* & v_z^*\end{bmatrix}^T$ is the desired velocity, the error for the velocity is given by $\mathbf{e}_\mathbf{v} = \mathbf{v} - \mathbf{v}^*$.

We want the tricopter to point towards the direction of the movement. Therefore, the desired direction of the first body-fixed axis is

$$\vec{\mathbf{x}}_B^* = \begin{bmatrix}v_x^* & v_y^* & 0\end{bmatrix}^T. \tag{14}$$

Now, the desired direction of the second and third body-fixed axes can be computed

$$\begin{cases} \vec{\mathbf{z}}_B^* = \frac{-k_\mathbf{v}\mathbf{e}_\mathbf{v} - mg\mathbf{e}_3}{\|-k_\mathbf{v}\mathbf{e}_\mathbf{v} - mg\mathbf{e}_3\|} \\ \vec{\mathbf{y}}_B^* = \frac{\vec{\mathbf{z}}_B^* \times \vec{\mathbf{x}}_B^*}{\|\vec{\mathbf{z}}_B^* \times \vec{\mathbf{x}}_B^*\|}, \end{cases} \tag{15}$$

where $k_\mathbf{v}$ is some positive constant and $\mathbf{e}_3 = \begin{bmatrix}0 & 0 & 1\end{bmatrix}^T$. We also assume that $\vec{\mathbf{x}}_B^*$ is not parallel to $\vec{\mathbf{z}}_B^*$.

The rotation matrix for the desired attitude is

$$\mathbf{R}^* = \begin{bmatrix}\vec{\mathbf{y}}_B^* \times \vec{\mathbf{z}}_B^* & \vec{\mathbf{y}}_B^* & \vec{\mathbf{z}}_B^*\end{bmatrix} \in \mathsf{SO}(3). \tag{16}$$

The attitude error is chosen to be

$$\mathbf{e}_\mathbf{R} = \frac{1}{2}\left[\mathbf{R}^{*T}\mathbf{R} - \mathbf{R}^T\mathbf{R}^*\right]^\vee, \tag{17}$$

where $^\vee$ is the vee map: $\mathsf{SO}(3) \to \mathbb{R}^3$. The tracking error for the angular velocity is

$$\mathbf{e}_{\boldsymbol{\omega}} = \boldsymbol{\omega}_B - \mathbf{R}^T\mathbf{R}^*\left[\mathbf{R}^{*T}\dot{\mathbf{R}}^*\right]^\vee. \tag{18}$$

Finally, the control inputs (5) are chosen as follows:

$$\begin{cases} T = \left(k_\mathbf{v}\mathbf{e}_\mathbf{v} + mg\mathbf{e}_3\right)^T\mathbf{R}\mathbf{e}_3 \\ \boldsymbol{\tau}_e = -k_\mathbf{R}\mathbf{e}_\mathbf{R} - k_{\boldsymbol{\omega}}\mathbf{e}_{\boldsymbol{\omega}} + \boldsymbol{\omega}_B \times \mathbf{I}\boldsymbol{\omega}_B, \end{cases} \tag{19}$$

where $k_\mathbf{v}$, $k_\mathbf{R}$ and $k_{\boldsymbol{\omega}}$ are some positive constant gains. The overall structure of the controller is illustrated in Fig. 3.

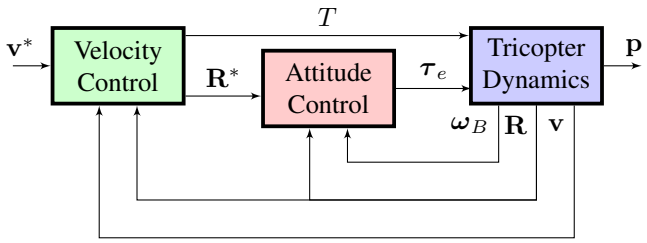


Fig. 3. Control system for the Y6 tricopter UAV.

IV. Simulation Results

A. Learning Approach

In this paper, the learning task consists in generating an evacuation plan from a building to help people in an emergency situation. An example of the top view of a building is depicted in Fig. 4a. Every door and window can be either open or closed, and their state might change over the time.

By applying the Algorithm 1, the agent, a Y6 tricopter UAV in this case, will learn through experience. According to our scenario, the agent can pass from one cell to the adjacent cell, but has no knowledge of the environment.

To simplify the learning task, the environment map is considered as a grid map (the grid map for the building in Fig. 4a is shown in Fig. 4b). For each cell of the grid world corresponds a state in the set of states S. So, if the dimension of the map is $d_x \times d_y \times d_z$, we have $d_x \cdot d_y \cdot d_z$ states. Each state is characterized by three parameters ($s_x \in [1, \dots, d_x]$, $s_y \in [1, \dots, d_y]$ and $s_z \in [1, \dots, d_z]$) corresponding to the coordinate of the cell. In our case, the floor is divided into five levels. Therefore, the map is composed by $20 \times 20 \times 5 = 2000$ cells. One cell on the map is a cube of about $1\mathrm{m}^3$.

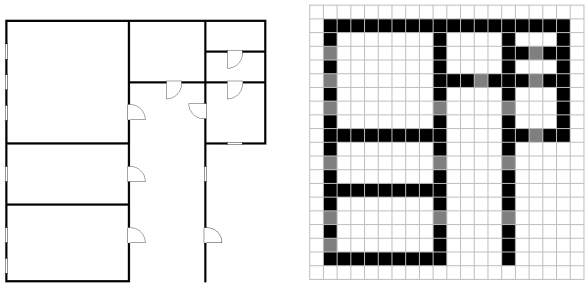
In each state, the agent can perform six actions: move forward, move backwards, move right, move left, move up and move down. If the agent has a wall in front, it can even move forward, but it will crash obtaining a negative reward. The transition function is non-deterministic: when the agent wishes to go forward, there exists a non-zero probability of ending up in an adjacent cell or even remain in the starting cell.

The goal cells have a positive reward $+100$ whereas the walls have negative rewards -1. The door or the window can either have null rewards, if it is open during the passage, or, like in the case of walls, negative rewards -1, if it is closed. The empty cells give no reward.

B. Training

In each training episode, the UAV starts from a known initial position $\mathbf{p}_0$ which is chosen randomly among all free cells. We use the variable learning rate α_t in (4):

$$\alpha_t = \frac{1}{1 + \mathrm{visit}_t(s, a)}, \tag{20}$$



(a) A sample of a building plan. (b) The grid map of the building.

Fig. 4. An example of a building plan in Fig. 4a and the correspondent grid map for the second level in Fig. 4b.

where $\mathrm{visit}_t(s, a)$ is the number of times state-action pair (s, a) was visited up to time t. Moreover, the discount factor γ is constant and equal to 0.9. The next action performed by the agent is chosen by combining exploitation and exploration:

$$\mathrm{P}(a_i|s) = \frac{k^{\mathrm{Q}(s,a_i)}}{\sum_{j=1}^{6} k^{\mathrm{Q}(s,a_j)}}, i = 1, \dots, 6, \tag{21}$$

where $k > 0$ determines how strongly the selection favours exploration over exploitation and it increases monotonously over time. During the exploitation stage, an action a that maximises $\mathrm{Q}(s, a)$ is selected, while during the exploration stage, an action a that minimises $\mathrm{Q}(s, a)$ is selected.

In order to evaluate quantitatively the efficiency of the obtained solution, we compute the distance to the closest emergency exit for each step. As can be seen from Fig. 5, the distance to an exit increases during the exploration (blue curve), while the distance decreases with steps during the exploitation (green, pink and yellow curves).

In Fig. 6 the learning process is shown. The brightness of each cell indicates the number that the UAV visited that cell. Before the learning starts, the map is completely monotonous, like in Fig. 6a. During the exploration stage, shown in Fig. 6b, the UAV tries to discover entire map. After the exploitation stage starts, the UAV follows more and more often the optimal trajectory. In Fig. 6c we can see that the cells closer to an exit are, the higher number of visits they have.

After 2×10^6 iterations, the optimal policy is learned. Q-Learning algorithm provides the optimal path based on the highest value of the function $\mathrm{Q}(s, a)$ (4) among the six actions in the grid world. The graphical representation of the optimal actions is shown in Fig. 7 for each floor level (Fig. 7a for level 1, Fig. 7b for level 2 and Fig. 7c for level 3). The UAV has six possible movements in each cell. On the map, each movement is represented by a different symbol: move forward ($\uparrow$), move backwards ($\downarrow$), move right ($\rightarrow$), move left ($\leftarrow$), move up ($\cdot$) and move down ($\circ$).

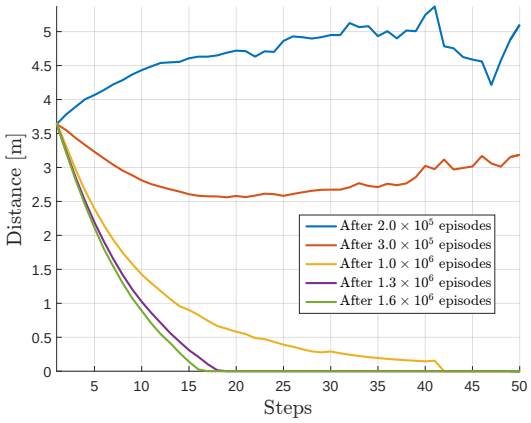
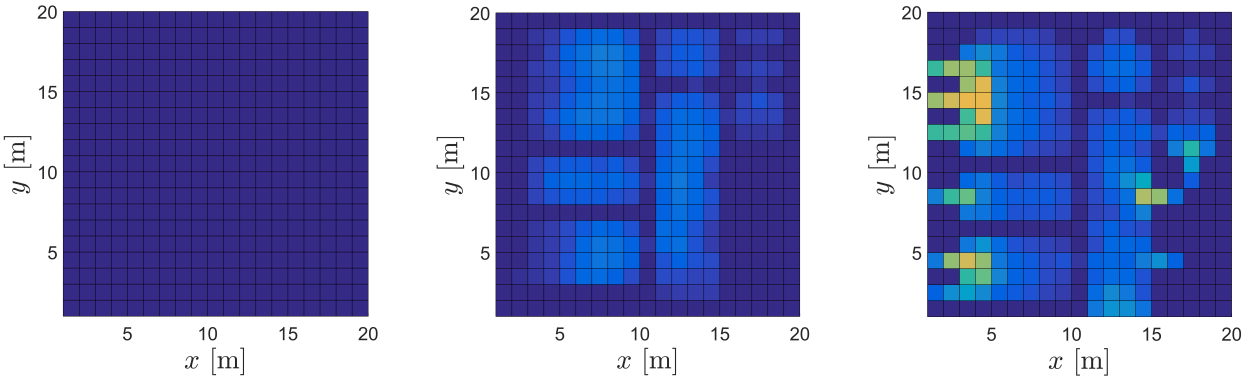
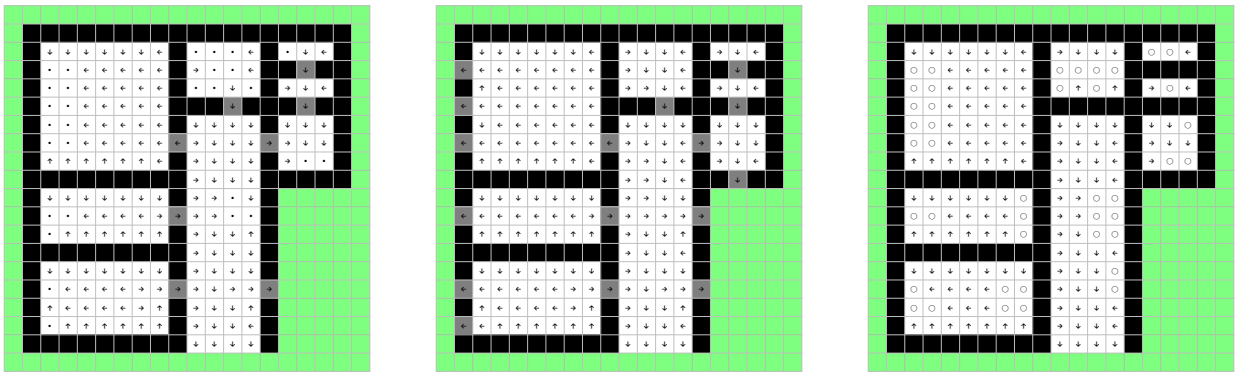


Fig. 5. The distance to the closest exit.



(a) After 0 episodes (b) After 2×10^5 episodes (c) After 2×10^6 episodes

Fig. 6. The histogram map of the number of visits for the second level of the building in Fig. 4b. On the left, the initial map, in the middle, the exploration stage, and, on the right, the exploitation stage. Brighter is the cell's colour, more times the cell was visited.



(a) Level 1 (b) Level 2 (c) Level 3

Fig. 7. The graphical representation of the optimal policy. Each movement is represented by a different symbol: move forward (↑), move backwards (↓), move right (→), move left (←), move up (·) and move down (○). Green cells are safe cells.

C. Dynamical Simulations

For the dynamical simulations, the tricopter model is implemented in ROS and Gazebo simulator. The tricopter intrinsic parameters are reported in Table I. These parameters are chosen close to the ones of a real tricopter. The UAV's position is measured with a simulated sensor. The attitude and angular velocities are provided by the simulated IMU.

In Fig. 8 the 3D model of the building is shown which plan was depicted in Fig. 4a. Figure 8 shows also the optimal trajectory, computed with Q-Learning, from different cells to the best emergency exit. The red dots indicate the initial position inside the building, the green dots indicate the final safe position outside the building.

TABLE I

UAV'S INTRINSIC PARAMETERS.

Parameter	Value	Unit
I_x	3×10^{-2}	$[\text{kg} \cdot \text{m}^2]$
I_y	6×10^{-2}	$[\text{kg} \cdot \text{m}^2]$
I_z	3×10^{-2}	$[\text{kg} \cdot \text{m}^2]$
l	2×10^{-1}	[m]
m	1.8	[kg]

V. Conclusions and Future Works

In this paper, we present a solution for ensuring a safe and autonomous evacuation in an unknown structured grid environment by using a tricopter UAV. Hence, the main task is the generation of the optimal building evacuation route in an emergency situation. Thus, we rely on the stochastic Q-Learning technique which uses the rewards from the environment to learn the optimal policy for the UAV's evacuation. In other words, our algorithm takes a state transition function and a reward function as the input, and learns how to leave a building where the agent resides. Finally, the UAV learns the optimal policy for each grid cell of the building.

In the future, the function $\mathrm{Q}(s, a)$ can be represented with an artificial neural network (ANN) instead of a look-up table. The inputs to the ANN will be the four positive integers: three for x, y and z coordinates of the cell correspondent to the state s and one for the action a. The output will have two values: one representing $\mathrm{Q}(s, a)$ and the other $\mathrm{visit}(s, a)$.

Acknowledgement

The research was partially supported by the ST Engineering - NTU Corporate Lab through the NRF corporate lab@university scheme.

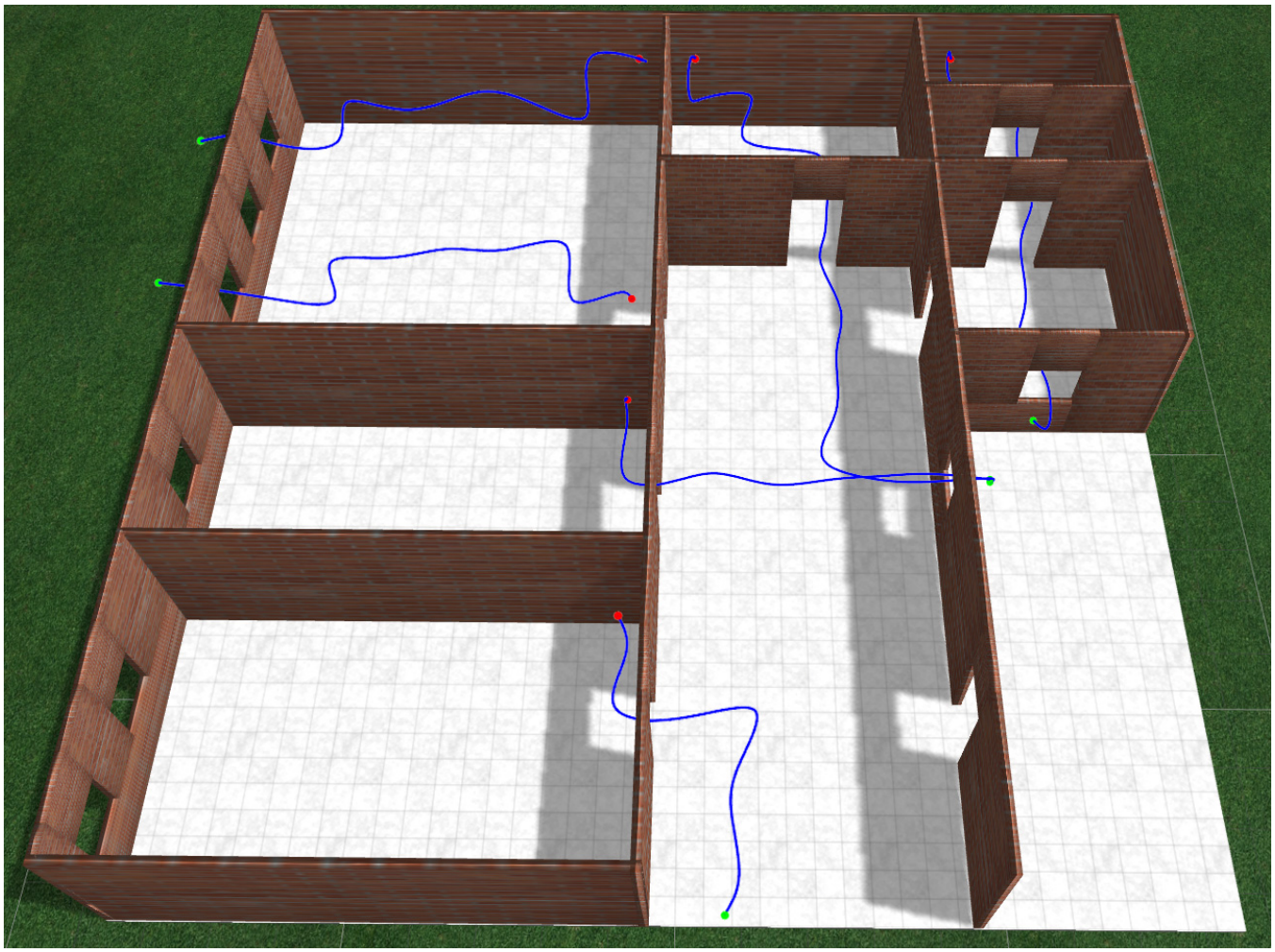


Fig. 8. UAV's optimal trajectories to the closest exit from different initial locations. The red dots indicate the initial position inside the building, the green dots indicate the final safe position outside the building.

REFERENCES

[1] J. Wu and G. Zhou, "Real-Time UAV Video Processing for Quick-Response to Natural Disaster," in *2006 IEEE International Symposium on Geoscience and Remote Sensing*, July 2006, pp. 976–979.

[2] I. Sa and P. Corke, *Vertical Infrastructure Inspection Using a Quadcopter and Shared Autonomy Control*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 219–232.

[3] C. Fu, A. Carrio, and P. Campoy, "Efficient Visual Odometry and Mapping for Unmanned Aerial Vehicle Using ARM-Based Stereo Vision Pre-Processing System," in *Unmanned Aircraft Systems (ICUAS), 2015 International Conference on*, 2015, pp. 957–962.

[4] D. Tamilselvi, S. M. Shalinie, and G. Nirmala, "Q-Learning for Mobile Robot Navigation in Indoor Environment," in *Recent Trends in Information Technology (ICRTIT), 2011 International Conference on*, June 2011, pp. 324–329.

[5] C. Chen, H. X. Li, and D. Dong, "Hybrid Control for Robot Navigation - A Hierarchical Q-Learning Algorithm," *IEEE Robotics Automation Magazine*, vol. 15, no. 2, pp. 37–47, June 2008.

[6] J. Li and W. Liu, "A Novel Heuristic Q-Learning Algorithm for Solving Stochastic Games," in *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, June 2008, pp. 1135–1144.

[7] L. M. Bello, P. Mitchell, and D. Grace, "Application of Q-Learning for RACH Access to Support M2M Traffic over a Cellular Network," in *European Wireless 2014; 20th European Wireless Conference; Proceedings of*, May 2014, pp. 1–6.

[8] D. P. Bertsekas and H. Yu, "Q-Learning and Enhanced Policy Iteration in Discounted Dynamic Programming," in *49th IEEE Conference on Decision and Control (CDC)*, Dec 2010, pp. 1409–1416.

[9] J. W. Lee, J. Park, J. O, J. Lee, and E. Hong, "A Multiagent Approach to Q-Learning for Daily Stock Trading," *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, vol. 37, no. 6, pp. 864–877, Nov 2007.

[10] E. Alpaydin, *Introduction to Machine Learning*, 2nd ed. The MIT Press, 2010.

[11] S. Bouabdallah, "Design and Control of Quadrotors with Application to Autonomous Flying," Ph.D. dissertation, EPFL, Dec 2007.

[12] R. Mahony, V. Kumar, and P. Corke, "Multirotor Aerial Vehicles: Modeling, Estimation and Control of Quadrotor," *Robotics Automation Magazine, IEEE*, vol. 19, no. 3, pp. 20–32, 2012.

[13] V. Mistler, A. Benallegue, and N. M'Sirdi, "Exact Linearization and Noninteracting Control of a 4 Rotors Helicopter via Dynamic Feedback," in *Robot and Human Interactive Communication, 2001. Proceedings. 10th IEEE International Workshop on*, 2001, pp. 586–593.

[14] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, May 2011, pp. 2520–2525.

[15] A. Sarabakha, "Reactive Obstacle Avoidance for Quadrotor UAVs Based on Dynamic Feedback Linearization," Master's thesis, "Sapienza" - Università di Roma, July 2015.

[16] T. Lee, M. Leoky, and N. H. McClamroch, "Geometric Tracking Control of a Quadrotor UAV on SE(3)," in *Decision and Control (CDC), 49th IEEE Conference on*, 2010, pp. 5420–5425.