A Q-Learning Approach to Flocking With UAVs in a Stochastic Environment

Shao-Ming Hung, *Member, IEEE*, and Sidney N. Givigi, *Senior Member, IEEE*

***Abstract*—In the past two decades, unmanned aerial vehicles (UAVs) have demonstrated their efficacy in supporting both military and civilian applications, where tasks can be dull, dirty, dangerous, or simply too costly with conventional methods. Many of the applications contain tasks that can be executed in parallel, hence the natural progression is to deploy multiple UAVs working together as a force multiplier. However, to do so requires autonomous coordination among the UAVs, similar to swarming behaviors seen in animals and insects. This paper looks at flocking with small fixed-wing UAVs in the context of a model-free reinforcement learning problem. In particular, Peng's $Q(\lambda)$ with a variable learning rate is employed by the followers to learn a control policy that facilitates flocking in a leader–follower topology. The problem is structured as a Markov decision process, where the agents are modeled as small fixed-wing UAVs that experience stochasticity due to disturbances such as winds and control noises, as well as weight and balance issues. Learned policies are compared to ones solved using stochastic optimal control (i.e., dynamic programming) by evaluating the average cost incurred during flight according to a cost function. Simulation results demonstrate the feasibility of the proposed learning approach at enabling agents to learn how to flock in a leader–follower topology, while operating in a nonstationary stochastic environment.**

***Index Terms*—Flocking, Q-learning, reinforcement learning (RL), unmanned aerial vehicles (UAVs).**

I. Introduction

IN THE past two decades, we have witnessed unprecedented growth of unmanned aerial vehicles (UAVs) in both military and civilian applications [1], [2]. Examples include the use of UAVs for intelligence, surveillance, and reconnaissance [3], search and rescue, mining operations [4], and agriculture [5]. Many of these applications contain tasks that are parallel in nature, thus can benefit from cooperation in terms of effectiveness [6]. Cooperative multirobot systems offer synergy [7]–[9], shorter task completion time (through parallelism) [6], greater spatial coverage, reduction in cost (through smaller, simpler, and cheaper robots), and increased robustness (through redundancy) [10]. However, one of the fundamental challenges of multirobot systems is to coordinate teams of robots to achieve a group objective or behavior [11], more importantly, to do so autonomously [12]. One solution to this problem is to emulate swarming behaviors, since animals and insects naturally exhibit effective group coordination [13]. In particular, the use of flocking [14] to coordinate multiagent systems is now a common practice, and can be found in a wide range of applications including game design, computer generated animations [15], mobile sensor networks [16], [17], and robotics [10], [13], [18].

The behavior of flocking emerges from a group when agents within move according to their own observations of their neighbors, while adhering to rules for separation, alignment, and cohesion. In control theory, distributed control laws, supported with rigorous proofs and simulated results, demonstrate the feasibility of simulated flocking [18]–[20]. Moreover, extensive research has gone into developing different methods of coordinating multiple nonlinear dynamical agents based on solving consensus problems [21], [22]. Specific to ground robots, [23]–[26] have all experimented with wheeled robots, performing various forms of flocking using different hardware solutions for communication and networking in controlled indoor environments. In the domain of aerial vehicles, Welsby and Melhuish [27] flew three motorized blimp-like objects in an indoor environment to demonstrate flocking in three dimensions. Hauert *et al.* [28] deployed ten small fixed-wing UAVs outdoors to study the tradeoff between communication range and flight dynamics. More recently, Quintero *et al.* [29] experimented with flocking in a leader–follower topology, where the problem of determining the follower's policy was setup as a stochastic optimal control problem solved using dynamic programming (DP). In addition to the novel algorithm, the policy generated was successfully tested on small fixed-wing UAVs flocking together to perform the task of target tracking. Similarly, Vásárhelyi *et al.* [31] implemented their flocking algorithm from [30] and created the first decentralized multicopter flock that performed stable autonomous outdoor flight with up to ten agents.

One of the underlying assumptions with traditional control systems is the availability of an accurate model of the plant and disturbances [32]. Often, these models are either unknown, nonlinear, complex, or changing during operation, thus hindering traditional analytical approaches [33]. As an alternative, machine learning techniques have been explored to provide robotic systems with tools to learn through interactions with the environment [34]. In particular, reinforcement learning (RL) allows the design of intelligent robots that can learn

Manuscript received April 23, 2015; revised October 27, 2015; accepted December 14, 2015. Date of publication January 5, 2016; date of current version December 14, 2016. This paper was recommended by Associate Editor T. Vasilakos.

The authors are with the Department of Electrical and Computer Engineering, Royal Military College of Canada, Kingston, ON K7K 7B4, Canada (e-mail: david.hung@rmc.ca; sidney.givigi@rmc.ca).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCYB.2015.2509646

without models, making robots more versatile and adaptable to dynamic environments.

RL is a general class of machine learning algorithms that aims at enabling an agent to learn how to behave through interactions with the environment in the form of actions and rewards. Interactions involve the learning agent performing specific actions, and observing the resulting state and reward. By estimating the values of each state, the agent constructs a mapping of states to action, known as a policy, that maximizes the expected long-term rewards. In essence, rather than being instructed on what to do and told whether the act was right or wrong, the agent learns through experiences [35]. RL has been applied to a large number of problems such as communications [36]–[39], vehicular networks [40], and computing [41].

The application of RL to flocking has been successfully demonstrated in simulation, where particle-based agents use Q-learning [42] to learn how to flock [43] and avoid predators [44]. More recently, La and Sheng [16] and La *et al.* [45] developed a hybrid system that combines a low level flocking controller with a high level RL module. The learning module determines safe spaces to navigate to, so that network topology and connectivity can be maintained while avoiding predators. Even though these works have incorporated RL methods in their approaches, there are two issues that have not been addressed. The first issue is learning in nonstationary stochastic environments, where the mean and variance of state transitions change over time. In these environments, the learned policies will change with the environment and do not necessarily converge to a single solution. The second issue is that the approaches thus far employ particle-based agents, which are only applicable for omni-directional robots or rotary UAVs. In comparison to rotary aircrafts, fixed-wings have superiority over range, endurance, and payload capacity, but requires a different approach to control in order to account for the nonholonomic constraints due to the dynamics of the aircraft. So far, there has been no published results on the application of RL to flocking with fixed-wing UAVs in nonstationary stochastic environments.

In light of this, we adopt the flocking framework proposed by Quintero *et al.* [29] and reformulate it in the context of a model-free RL task. In contrast to [29], which computes an optimal control policy offline and assumes knowledge of the system in the form of a model, our approach provides online learning of a control policy and relaxes the assumption of having a model (i.e., model-free) of the environment as prior knowledge. The advantage of a model-free approach is that it can be applied to different platforms without the plant and disturbance models, which implies greater adaptability to changing environments and unforeseen situations. Unlike other works in multiagent coordination, which focus on developing graph-based, low level control laws for coordinating nonlinear dynamical agents [20]–[22], we assume that coordination occurs through the leader–follower relationship along with the sharing of knowledge or experiences among the followers. The former assumption establishes the flocking behavior between the followers and a single leader, while the latter provides a common policy among the followers to abide by, which requires the followers to operate at different altitudes in order to avoid collisions.

The main contribution of this paper is the application of Q-learning, specifically Peng's $Q(\lambda)$ [46], to flocking with small fixed-wing UAVs in simulation, and testing the algorithm in a nonstationary stochastic environment. Using the same stochastic environmental models, we compare our approach to Quintero's, herein known as the Quintero's dynamic programming (QDP) approach, by measuring the performances of the respective policies at maintaining the desired flocking behavior according to a cost function. The policies generated using QDP provide a good benchmark because the approach determines the optimal control policy given that a stochastic model is available. If the same model is used to simulate online learning, then we would expect the resulting policy to be comparable to the policy generated using QDP.

By applying RL to flocking, we investigate the feasibility of designing intelligent fixed-wing agents that can learn to flock in nonstationary stochastic environments. Such method could find use in coordinating teams of small fixed-wing UAVs, where the plant and disturbance models may be unknown or incomplete.

This paper is organized as follows. In Section II, we begin by defining the flocking problem in the context of RL, followed by the notation to be used throughout this paper. Then in Section III, we introduce the proposed RL algorithm for solving the flocking problem. Subsequently, in Section IV we discuss the convergence of the proposed algorithm, and in Section V, we present the simulation and evaluation process. In Section VI, the results demonstrating the feasibility of the algorithm are provided. Finally, the conclusions and future work are presented in Section VII.

II. Problem Statement

In this section, we will formulate flocking as a model-free RL problem, in the form of an Markov decision process (MDP) that is nearly identical to the one proposed in [29]. For consistency and ease of comparison, the notation used in this paper follows [29] and [47]. We will begin with a general notation for an MDP, followed by a detailed definition of the individual elements with respect to the flocking problem.

Classical RL problems assume an MDP defined by a tuple $(S, A, P^a_{ss'}, R^a_{ss'})$, where:

1) S is a finite set of states;
2) A is a finite set of actions;
3) $P^a_{ss'}$ is the state transition probability function or model that defines the probability of observing the next state $s' \in S$ after executing action $a \in A$ in state $s \in S$;
4) $R^a_{ss'}$ is the reward function that specifies rewards after executing action $a \in A$ in state $s \in S$ and resulting in the next state $s' \in S$.

The RL interaction is depicted in Fig. 1, where at each time step (assuming discrete time), the agent in state $s \in S$ chooses an action $a \in A$. As a consequence of that action, the agent moves to a new state $s' \in S$ with probability $P(s'|s, a)$, and receives a reward $R(s', s, a)$. Ultimately, the goal of the agent

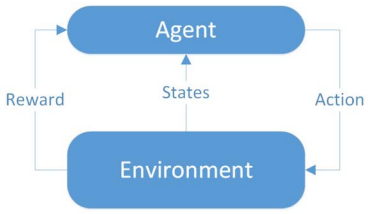


Fig. 1. Agent-environment interaction.

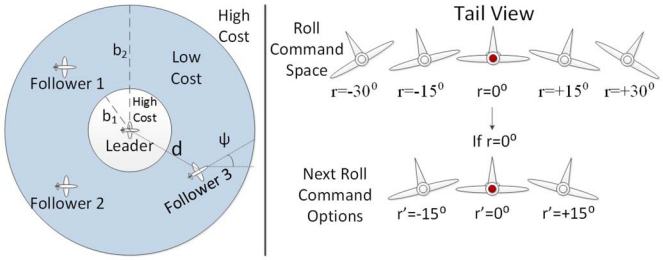


Fig. 2. Left: the top view of relationship between leader and followers. Right: the action space.

is to learn a policy F by mapping states to actions, such that the expected discounted rewards are maximized.

In our flocking scenario, the leader and the followers, also known as agents, are modeled as small fixed-wing UAVs flying at a constant average speed and fixed altitude. The leader has its own set of mission dependent control policy (e.g., target tracking, mapping, etc.), while the followers are to flock with the leader and minimize the overall cost, which is a function of distance and heading relative to the leader. To illustrate this idea, the left diagram in Fig. 2 shows the leader centered on an annulus, and the followers positioned within the shaded region where the cost is low. If the followers move too far or too close to the leader, their respective costs will increase.

The agents maneuver by selecting a roll angle setpoint, which is regulated by an autopilot and is updated once every second [i.e., 1 s zero order hold (ZOH)]. This implies that the simulation is run in discrete time-steps denoted by k. Collision avoidance between the agents is de-conflicted by operating at different altitudes, thus the same control policy can be used for each of the followers. Consequently, the objective is for the followers to learn a policy that flocks with the leader, while minimizing costs, and without knowledge of the state transition model. In solving this RL problem, the followers will learn a strategy that describes the best roll angle setpoint for a given state. When each follower adheres to the same strategy, the aggregate behavior emulates flocking in a leader–follower topology.

A. State Representation

The individual agents are represented using a four state UAV model defined as $\xi := (x, y, \psi, \phi)$, where $(x, y) \in \mathbb{R}^2$ is the planar position, $\psi \in \mathbb{S}^1$ is the heading, and $\phi \in \mathbb{S}^1$ is the roll angle [29]; $\mathbb{S}^1$ (i.e., 1-sphere) represents the $(n+1)$-dimensional Euclidean space, in this case of a circle. The kinematic model used to simulate the transitions of the UAVs will be presented in Section II-C. For the purpose of flocking in a leader–follower topology, we are only concerned with the relative dynamics between the leader and the follower. Consequently, their respective UAV states ξ_l and ξ_f are combined to construct the system state space as $z := [z_1, z_2, z_3, z_4, z_5, z_6]^\top \in \mathcal{Z}$, where $\mathcal{Z} := \mathbb{R}^2 \times [-\pi, \pi) \times \mathbb{S}^1 \times \mathbb{S}^1 \times C$ [29]; C is defined as

$$C := \{0°, \pm 15°, \pm 30°\}. \tag{1}$$

Similar to [29], the planar position of the follower relative to the leader represented by the pair (z_1, z_2) is defined as

$$\begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} \cos\psi_l & \sin\psi_l \\ -\sin\psi_l & \cos\psi_l \end{bmatrix} \begin{bmatrix} x_f - x_l \\ y_f - y_l \end{bmatrix}. \tag{2}$$

In addition, the remaining substates (z_3, z_4, z_5, z_6) shown in (3) represent the difference in the heading between the leader and follower, the follower's roll state, the leader's roll state, and the leader's roll command, respectively

$$(z_3, z_4, z_5, z_6) := (\psi_f - \psi_l, \phi_f, \phi_l, r_l). \tag{3}$$

In the simulation, we assume that the leader's roll command is determined randomly, thereby introducing additional stochasticity to the problem. Furthermore, we assume that the leader's state and roll command are broadcast through a wireless communication channel to the followers. In practice, the leader's roll command would be mission dependent (e.g., target tracking and mapping).

Lastly, the system state space is discretized as $S = X^2 \times \Psi \times C^3$, where $X = \{-150, -145, \ldots, 150\}$ and $\Psi = \{0°, 15°, \ldots, 345°\}$. For clarity, X divides the planar surface into a 61×61 grid space, Ψ divides a circle that is centered on the leader into pie shaped sections each with a subtended angle of 15°, and C defines the discretized roll angle states.

B. Action Space

The fixed-wing agents maneuver by selecting their respective roll commands $r \in C$, and holding them for 1 s or until the next command is selected. In order to mitigate any adverse effects on the mission caused by sharp changes in the roll, the next roll command is defined as $r' \in U(r)$ [29] where

$$U(r) := \{r, r \pm 15°\} \cap C. \tag{4}$$

The right illustration in Fig. 2 depicts the five roll-angle states that the agents can be in and the options for the next roll command; the agent can maintain current roll-angle or change by as much as ±15°. From a systems point of view, limiting the action space in our scenario helps alleviate the curse of dimensionality [48] at the cost of maneuverability.

C. State Transition Model

UAV kinematics are most accurately captured by a six degree of freedom aircraft model. However, by assuming the UAVs fly at a constant altitude and velocity, as well as with inertially coordinated turns, we can simplify the model down to four degrees of freedom. A coordinated turn is where the bank angle is set so that the centrifugal force acting on the aircraft is equal and opposite to the horizontal component of the lift acting in the radial direction [49]. This flight condition

is commonly used in manned flights for passenger comfort. To make up for the loss of unmodeled dynamics and account for environmental disturbances, stochasticity is introduced in the roll, airspeed, and each of the substates of the model.

We adopt the continuous-time UAV kinematic model from [29] and include additional terms (η_x, η_y, η_ψ) in the first three states to represent disturbances that cause the xy-planar position and heading of the UAVs to change. The additional terms enables simulating nonstationarity in the state transitions. By applying an 1 s ZOH to the roll command r, we create the discrete-time model, where the time steps are indexed by $k \in \mathbb{Z}_{\geq 0}$. This stochastic model generates state transitions for the individual agents, which are then combined using (2) and (3) to construct the system state transition model $P(z'|z, r_f)$. The continuous-time model is defined as

$$\dot{\xi} = \frac{d}{dt}\begin{pmatrix} x \\ y \\ \psi \\ \phi \end{pmatrix} = \begin{pmatrix} s\cos\psi + \eta_x \\ s\sin\psi + \eta_y \\ -(\alpha_g/s)\tan\phi + \eta_\psi \\ f(\phi, r) \end{pmatrix} \tag{5}$$

where α_g is the acceleration due to gravity, and s is the nominal airspeed of the UAVs. The airspeed is drawn from a normal distribution $\mathbb{N}(\bar{s}, \sigma^2)$, and held constant for the duration of the ZOH. In addition, the disturbance terms are drawn from normal distributions $\mathcal{N}(\overline{\eta}_x, \sigma_x^2)$, $\mathcal{N}(\overline{\eta}_y, \sigma_y^2)$, and $\mathcal{N}(\overline{\eta}_\psi, \sigma_\psi^2)$, respectively. Lastly, the function $f(\phi, r)$ defines the roll dynamics, which is sampled from a collection of stochastic second-order roll trajectories.

To generate the trajectories, we simulate the initial condition response of the roll dynamics using a second-order system, where the undamped natural frequency ω_n and damping ratio ζ are selected based on the autopilot parameters of a small UAV [50]. For each roll command (increase by 15°, decrease by 15°, and maintain current roll), we collect the corresponding reference tracking error trajectories denoted as $\{\hat{e}_i(\tau)\}$, $\{\check{e}_i(\tau)\}$, and $\{\overline{e}_i(\tau)\}$, where $\tau \in [0, 1]$, $i \in \{1, \ldots, 1000\}$, and the accents indicate the respective roll commands that the error trajectories correspond to [29]. For example, a roll command to increase the roll angle by 15° would draw a sample from $\{\hat{e}_i(\tau)\}$. The left plot in Fig. 3 shows the collection of error trajectories for an increase of 15° in the roll angle setpoint. With the collection of error trajectories, the sample roll trajectories $\phi_i(\tau, r)$ are generated according to

$$\phi_i(\tau, r) = e_i(\tau) + r. \tag{6}$$

The right plot in Fig. 3 depicts a sampled roll trajectory transitioning from 15° to 30°.

The state transitions are generated by randomly selecting a roll error trajectory from the collection, and applying it to the kinematic model. For an agent that is positioned at the origin ($x = 0, y = 0$) and facing the $+x$ direction, Fig. 4 shows two sets of three possible collections of resulting states on the xy plane, one for each of the three different roll commands. The two sets labeled as $M1$ and $M2$ illustrate the effects of changing the mean and standard deviations of the disturbance terms, making it possible to create a nonstationary environment.

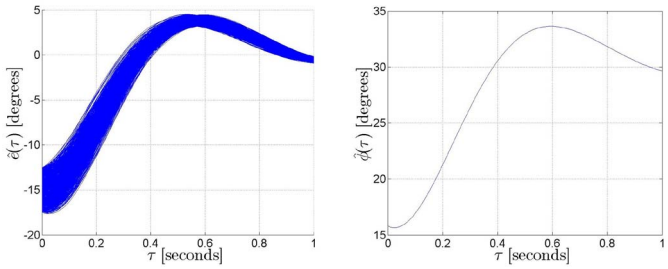


Fig. 3. Left: simulated roll trajectories of +15° change in roll setpoint. Right: sample roll trajectory changing roll-angle setpoint from 15° to 30°.

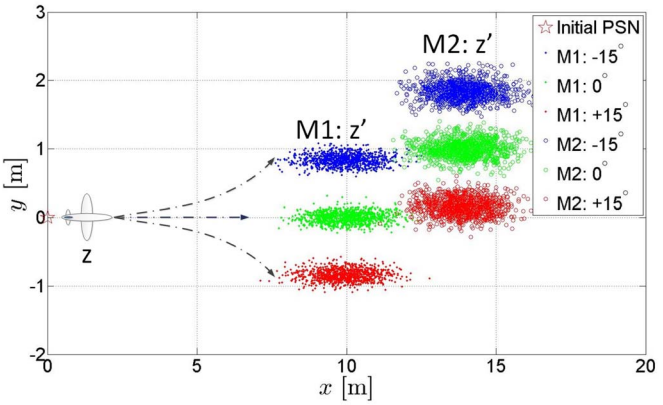


Fig. 4. M1: collection of possible resulting UAV states due to stochasticity in roll and airspeed. M2: collection of possible resulting UAV states due to stochasticity in x, y, heading, roll, and airspeed.

D. Reward Scheme

To facilitate flocking, the reward function is represented as a cost function from [29], defined as

$$g(z) = \max\left\{d, \frac{b_1|z_3|}{\pi(1 + \beta d)}\right\} \tag{7}$$

where $d = \max\{b_1 - \rho, 0, \rho - b_2\}$ and β is a tuning parameter that modifies the impact of d. The parameters b_1 and b_2 define the inner and outer radius of an annulus Λ centered on the leader, shown in the left illustration of Fig. 2, and given by

$$\Lambda = \left\{(z_1, z_2) \in \mathbb{R}^2 : b_1 \leq \rho \leq b_2\right\} \tag{8}$$

where $\rho := \sqrt{z_1^2 + z_2^2}$. The first argument in the maximization function of (7) governs the separation and cohesion rules of flocking, and the second argument assigns a cost to the absolute difference of the heading angles, hence reinforces the alignment rule. Fig. 5 shows the cost plot relative to the leader who is centered at (0, 0), As shown, the cost increases when the follower is near or far away from the leader. This is consistent with Fig. 2, where the ring shaped region around the leader incurs the lowest cost.

E. RL Objective

With the elements of the MDP defined, it becomes clear that the RL problem is to learn the optimal control policy

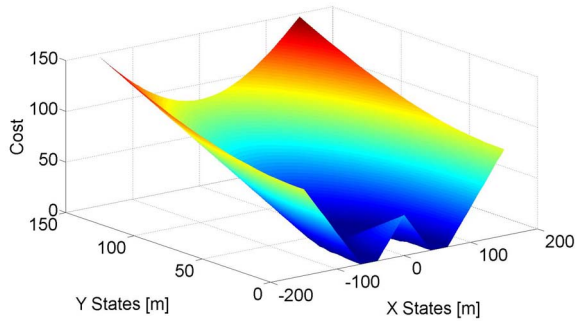


Fig. 5. Sliced illustration of the cost for a subset of state-actions.

$F_k^* : \mathcal{Z} \to C, k \in \{0, \ldots, \infty\}$ that minimizes

$$J(z[0]) = E\left[\sum_{k=0}^{\infty} g(z[k])|z[0]\right], \ \forall\, z[0] \in \mathcal{Z} \tag{9}$$

where $E[\cdot]$ denotes expectation, and $z[k]$ refers to $z(t)$ at discrete time instances [29]. This minimization can be solved using DP to perform value iteration [29], provided that the model is *a priori* knowledge. In the context of RL, we assume that the learning agents do not have the model, and must employ a model-free approach, in this case Q-learning. The idea behind Q-learning is to incrementally estimate the value $Q^*(s, a)$ of a state-action pair (s, a), thereby assigning a value to the expected long-term reward that can be obtained by taking action a in state s. Taking into account of the reward being a cost, and replacing the standard notations for ours, $Q^*(z, r)$ can be determined by solving the Bellman [48] equation

$$Q^*(z, r) = \sum_{z'} P^r_{zz'}\left(g(z') + \gamma \min_{r'} Q^*(z', r')\right)$$

where $P^r_{zz'}$ represents the state transition probability (i.e., from z to z' due to action r), and $0 \leq \gamma \leq 1$ is the discount factor. Subsequently, the optimal control policy can be determined by

$$F^*(z) = \arg\min_r Q^*(z, r).$$

In model-free RL problems, $P^a_{ss'}$ (general form) is unknown, and the only form of feedback is through rewards or punishments (e.g., cost). Hence, as the agent interacts with the environment $Q^*(z, r)$ is incrementally estimated using

$$Q'(z, r) \leftarrow Q(z, r) + \alpha\left[g(z) + \gamma \min_{r'} Q(z', r') - Q(z, r)\right]$$

where $0 < \alpha \leq 1$ is the learning parameter or rate [47]. By taking the minimum Q-value of the next state, the algorithm becomes exploration insensitive under the assumption that each state-action pair is visited an infinite number of times, and α is decreased appropriately over time [51].

III. Reinforcement Learning Algorithm

The Q-learning algorithm used in our experiments are based on Peng's $Q(\lambda)$ [46], which consists of two separate updates of the Q-table. The one-step update is defined as

$$Q_{k+1}(z_k, r_k) = Q_t(z_k, r_k) + \alpha\delta \tag{10}$$

where

$$\delta = g(z_{k+1}) + \gamma \min_{r_{k+1}} Q(z_{k+1}, r_{k+1}) - Q(z_k, r_k). \tag{11}$$

Subsequently, the second update based on the eligibility traces [47] is defined for all z, r as

$$Q_{k+1}(z_k, r_k) = Q_k(z_k, r_k) + \alpha\delta' \mathrm{Tr}(z, r) \tag{12}$$

where

$$\delta' = g(z_{k+1}) + \gamma \min_{r_{k+1}} Q(z_{k+1}, r_{k+1}) - \min_r Q(z_k, r_k) \tag{13}$$

and

$$\mathrm{Tr}(z, r) = \begin{cases} \gamma\lambda \mathrm{Tr}(z, r) + 1 & \text{if } z = z_k, \ \& \ r = r_k \\ \gamma\lambda \mathrm{Tr}(z, r) & \text{otherwise.} \end{cases} \tag{14}$$

The number of state-action pairs that need to be updated at each time step grows linearly with time, and worst case it could encompass the entire state-action space [46]. In consideration of keeping the updating process at a manageable level, a five-step backup is used. This means the previous five state-action pairs are remembered and assigned with the current temporal-difference (TD) error as shown in (13).

According to [47], to ensure convergence with the probability of one, the learning parameter must satisfy the following conditions:

$$\sum_{k=1}^{\infty} \alpha_k(r) = \infty \quad \sum_{k=1}^{\infty} \alpha_k^2(r) < \infty. \tag{15}$$

However, since it is desirable to have the estimated Q-values continue to vary in response to the most recent feedback, the second condition does not hold true in nonstationary environments [47]. When operating in nonstationary environments, it is common to use a fixed learning parameter, or adaptive strategies that determine the optimal size based on some meta-stepsize parameter [52]. The difficulty in tuning the learning parameter is that a high rate is needed for faster convergence, but it also increases the chances of divergence [52]. Without diving into the intricacies of determining an optimal learning parameter, we consider a rather simple method defined as

$$\alpha(\delta) = \max\left(\min\left(1, \left(\frac{|\delta|}{\varrho}\right)^p\right), \varsigma\right) \tag{16}$$

where $\varrho \in \mathbb{R}_{>0}$ and $p \in \mathbb{R}_{\geq 1}$ are tuning parameters and ς sets a lower bound. In essence, if there is a large difference between the feedback and what the learner already knows, then the difference (i.e., TD-error) is weighted more by using a larger learning parameter. Similarly, if the difference is small, then most likely the estimated Q-values are converging, or significant changes have yet been propagated (i.e., exploration issue), therefore the learning parameter should be relatively small. For practical purposes, the lower bound ς ensures that α does not become too small (i.e., learning becomes slow).

The learning algorithm we propose, herein known as Q-flocking, is shown in episodic procedural in Algorithm 1. For clarity, the following steps outlines what occurs within the follower (i.e., learning agent).

1) Receive the leader's state and combine it with own state to create a system state.

Algorithm 1 Q-Flocking

initialize $Q(z, r)\ \forall\ z \in Z$, $r \in C$, γ, ϵ, T_{sim}, terminal conditions
  repeat(for each episode)
    Start On-line Learning
    initialize $z_0 \leftarrow (\xi_f, \xi_l)$ randomly
    while ($bound_{min} \leq \rho \leq bound_{max}$ || $k \leq T_{sim}$) **do**
      choose $r_{k,l} \in U(r_{k-1,l})$ randomly
      $r_{k,f} = \arg\min_r Q(z_k, r)$ or ϵ-greedy
      $(\xi_{k+1,f}, \xi_{k+1,l}) \leftarrow$ Sim Real UAV($\xi_{k,f}$,$r_{k,f}$,$\xi_{k,l}$,$r_{k,l}$)
      $z_{k+1} \leftarrow$ Create System State ($\xi_{k+1,f}, \xi_{k+1,l}, r_{k,l}$)
      $\delta_k = g(z_{k+1}) + \gamma \min_{r_f} Q_k(z_{k+1}, r_f) - Q(z_k, r_{k,f})$
      $\delta'_k = g(z_{k+1}) + \gamma \min_{r_f} Q_k(z_{k+1}, r_f) - \min_{r_f} Q_k(z_k, r_f)$
      $\alpha \leftarrow$ Equation (16)
      for each state-action pair **do**
        $Tr(z, r) = \gamma\lambda Tr(z, r)$
        $Q_{k+1}(z, r) \leftarrow Q_k(z, r) + \alpha Tr(z, r)\delta'$
      end for
      $Q_{k+1}(z_k, r_{k,f}) \leftarrow Q_k(z_k, r_{k,f}) + \alpha\delta$
      $Tr(z_k, r_{k,f}) = Tr(z_k, r_{k,f}) + 1$
      $z_k \leftarrow z_{k+1}$; $k \leftarrow k + 1$; $\rho = \sqrt{z_1^2 + z_2^2}$
    end while
    End On-line Learning
  until desired number of episodes

2) Select an action based on the ϵ-greedy method, which is to explore or lookup the Q-table using the system state to find the greedy action.
3) Execute the action selected and observe the change in own state.
4) Receive the leader's state and combine it with own state to create a new system state.
5) Calculate the cost using the new system state.
6) Update the Q-table table with the one-step update and the eligibility trace update.
7) Update the eligibility traces.
8) Go to step 2 and repeat until the end of the episode.

IV. Convergence of the Algorithm

In this section, we will discuss the convergence conditions of Algorithm 1 on two fronts. The first are the conditions set out in (15), which pertain to the learning parameter. The second are the conditions that must be satisfied for Peng's $Q(\lambda)$ to converge to Q^*. The idea is to show that in principle the algorithm converges to Q^* even though it consists of parameters that may violate theoretical criteria for convergence.

In a stationary environment, as each state-action pairs are visited an infinite amount of times, the TD-error δ becomes very small and eventually reaches zero in the limit. Furthermore, by removing the lower bound ς from (16), it becomes

$$\lim_{\delta \to 0} \alpha(\delta) = 0. \tag{17}$$

This means that by removing the lower bound from (16), the proposed variable learning method would satisfied both conditions in (15), and ensure convergence with the probability of one. In practice, though, UAVs operate in a nonstationary environment where the TD-error could be small, but most likely never zero. For this reason, it is common to use a fixed learning parameter or adaptive learning parameter to ensure that the agent never stops learning, and that learning does not become too slow (i.e., tiny learning rate) [47]. For the same practical reasons, we incorporated the lower bound ς into (16).

According to [46] and [47], Peng's $Q(\lambda)$ has not been theoretically proven to converge to Q^*. However, by gradually reducing λ or making the policy more greedy, Peng's $Q(\lambda)$ may still converge to Q^*. As λ reduces to zero over time $Q(\lambda)$ becomes the one-step Q-learning algorithm, which has been shown to converge to Q^* with the probability of one [53]. Similarly, as the policy becomes more greedy, then $Q(\lambda)$ eventually learns the greedy policy, which in a stationary environment should converge to Q^*. In a nonstationary environment though, the agents must continue to explore the state space in order to learn about any changes in the environment, hence it would be impractical to reduce λ or to make the policy more greedy over time. In pursuance of designing functional algorithms that account for the stochasticity in the physical world, we have opted for convergence to a good policy rather than the optimal policy. Moreover, even though Peng's $Q(\lambda)$ has not been theoretically proven to converge to Q^*, several studies have shown empirically that it outperforms the one-step Q-learning algorithm, Watkins's $Q(\lambda)$ and Sarsa(λ) [47].

Regarding to the convergence, a nonstationary environment can be represented as a finite number of environment modes [54] such that Peng's $Q(\lambda)$ can be applied to each one of the problems individually. By doing so, Q-learning would work as a function approximation device [55] and the stochastic approximation algorithm is guaranteed to converge [56] for each mode. This is a good assumption for the scenario proposed in this paper, since one may assume that the environment can be represented by a finite number of possible wind conditions. Therefore, the learning algorithm would always converge to a solution for a given environment mode; this may be inferred from the results presented in Section VI.

V. Simulation and Evaluation Process

A. Simulation Process

The simulation process occurs in a series of episodes, where 50 000 followers are simultaneously learning how to flock with a single leader. In order to reduce the number of tables required and expedite the learning process, a single table is shared among the 50 000 learning agents [57], [58]. The duration of each episode is limited to 30 time steps (each step represents 1 s wall-clock time), or when the followers travel outside of X^2. More specifically, the individual followers cease to update the Q-table when they travel outside of X^2.

Each episode begins with selecting random UAV states for all of the agents, and selecting a random action for the leader. This constitutes the system state that is used by the followers to lookup the Q-table for the best action. Exploration is achieved by taking random actions based on the ϵ-greedy method. Once the actions have been determined, the stochastic kinematic model generates the next UAV states, which lead to a new set of system states. Subsequently, the respective

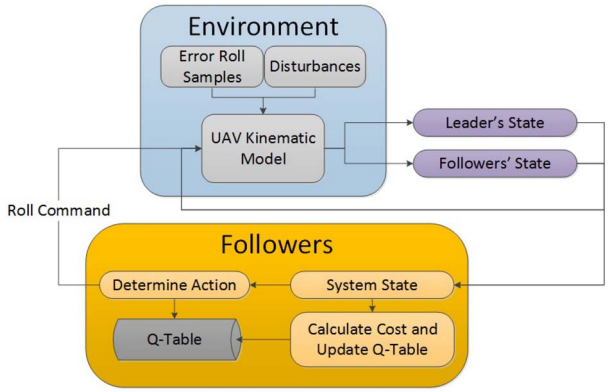


Fig. 6. Followers-environment interaction.

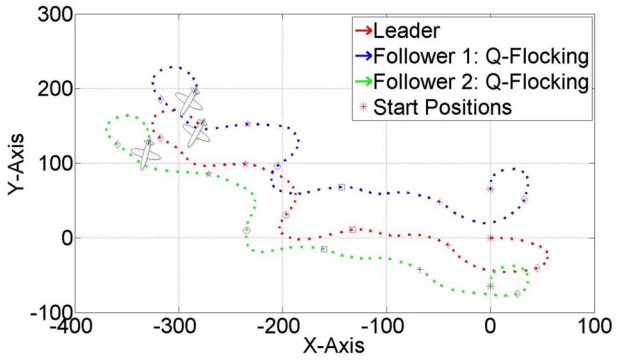


Fig. 7. Simulated trajectories of two followers using a learned policy to flock with the leader.

costs are calculated, the Q-table is updated, and the whole process repeats itself until the values in the Q-table converge. A block diagram depicting the simulated followers-environment interaction is shown in Fig. 6. The blocks highlighted in blue represent the environment or model that generates the state transitions, which is assumed to be unknown to the learning agents. The blocks highlighted in orange represent what is internal to the learning agent, where Q-learning takes place. Lastly, the Q-table (highlighted in gray) represents a tabular repository of state-action values that gets updated with the latest estimated values and is looked up for the best action to take at each time step. Fig. 7 provides a visualization of how the followers would physically flock with the leader.

B. Evaluation of Policy

Evaluation of the policies occur both during the learning process as the policies evolve, and at the very end when the policies converge. In both cases, the evaluation procedure is the same. To evaluate a policy, we simulate a follower using that policy to flock with a single leader over 1000 random trajectories, and measure the costs incurred. The duration of each trajectory is 120 time-steps, and a single trajectory is denoted as Υ_n, with the subscript n indicating the trajectory number. The cost of each trajectory is averaged over the entire run (i.e., 120 time-steps), and at each time-step the cost is calculated using (7). Fig. 8 depicts a single run of a random trajectory for a leader and follower pair, where the leader's

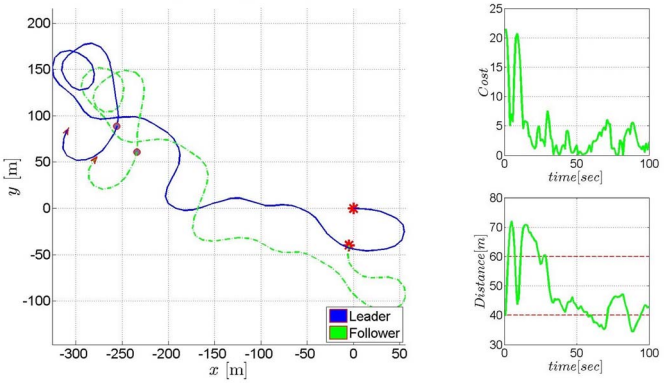


Fig. 8. Plot of a single set of leader and follower trajectory for evaluation.

actions are random, and the follower looks up the best action to take according to a learned policy. The cost is plotted in the top right corner and the distance between the pair is plotted in the bottom right corner. The same set of random trajectories are used to assess all of the policies, and the collection of Υ_1 to Υ_{1000} for each policy is denoted as Γ, where $\Gamma(F)$ denotes the collection or set of average costs incurred by using policy F. The average of Γ denoted as Γ_{Ave} is given as

$$\Gamma_{\text{Ave}} = \frac{1}{1000} \sum_{n=1}^{1000} \Upsilon_n. \tag{18}$$

With Q-flocking, the policies are evaluated once every episode to capture the evolution of the policies, which is reflected in the convergence of Γ_{Ave} over time. On the other hand, the policies generated using the QDP approach are evaluated on every backup of value-iteration. The term backup refers to a single update of all the state-action pairs done using DP. The learned policies using Q-flocking are denoted as $F_Q(\alpha)$, where α indicates the learning parameter used. Policies generate using the QDP approach are denoted as $F_{\text{DP}}(\varpi)$, where $\varpi \in \mathbb{R}_{>0}$ indicates the number of backups.

To compare the policies, the mean and the standard deviation of Γ for each policy are computed after 1000 episodes, or in the case of QDP, the policy with the lowest Γ_{Ave} is used. In addition, t-tests are performed on Γ to assess the statistical significance of the average cost data. The t-test results indicate which policy performs better with a level of statistical confidence.

VI. Simulation Results

We conducted two sets of experiments, where the stochasticity of UAV kinematic models in each were specified differently as shown in Table I. In Experiment I (Exp I), the kinematic model denoted as M_1, accounted for stochasticity in the roll angle dynamics and the airspeed. In Experiment II (Exp II), additional disturbances (nonsymmetrical) were introduced to the model, denoted as M_2, by setting η_x, η_y, η_ψ, and their respective standard deviations to nonzero values (see Table I). Furthermore, the followers in Exp II were bootstrapped with the learned policies from Exp I to simulate the perception of a nonstationary stochastic environment. In both experiments,

TABLE I
SIMULATION PARAMETERS

General Parameters								
	γ	λ	b_1	b_2	β	α_g	$\overline{s}$	σ_s
Values	0.8	0.9	40	65	0.05	9.8	10	0.8

	Disturbance Parameters			Learning Parameters	
	$\overline{\eta}_x, \overline{\eta}_y$	$\overline{\eta}_\phi$	$\sigma_x, \sigma_y, \sigma_\phi$	α: Static	α: (ϱ, p, ς)
Exp I	0	0	0	0.1, 0.8	$(120, 3, 0.005)$
Exp II	5	0.1π	0.01	0.1, 0.8	$(120, 3, 0.005)$

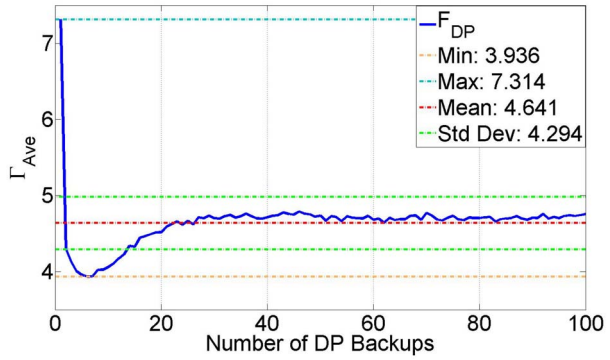


Fig. 9. Γ_{Ave} per the number of DP backups for the QDP approach in Exp I.

we simulated and compared the resulting policies produced from Q-flocking and QDP. However, since the QDP approach calculates a policy offline [29] based on a known model, its policy F_{DP} does not change in Exp II.

A. Experiment I

The aim of Exp I was to demonstrate the feasibility of using Q-flocking to learn how to flock in a stochastic environment defined by the model M_1. The experiment was divided into two parts. In part one, F_{DP} policies were generated according to the QDP approach and using the model M_1. This involved performing value-iteration to propagate values through the state-action space. A total of 100 policies were generated over the course of backing up 100 times. Γ_{Ave} for each of the F_{DP} policies are shown in Fig. 9. According to this graph, the policy generated after the sixth backup $F_{\text{DP}}(6)$ incurred the lowest cost, and as the number of backups increased the average cost converged to approximately 4.7 (see Fig. 9). From the perspective of a model-based approach, the trend in Fig. 9 suggests that the ideal planning horizon with the model M_1 is six backups, and that it becomes impractical to consider what the impact of the current action has beyond six time-steps into the future. This is due to the fact that the stochasticity increases with every time-step; the transitions for the leader and followers are determined based on some randomness in the airspeed and roll. With each increment into the horizon to consider, the randomness creates additional uncertainty as to where the leader and followers will end up. Fig. 10 illustrates the exponential growth of decisions to consider as the horizon increases, which also corresponds to an increase in stochasticity. Since $F_{\text{DP}}(6)$ incurred the lowest Γ_{Ave}, it was used as the benchmark in Exp I.

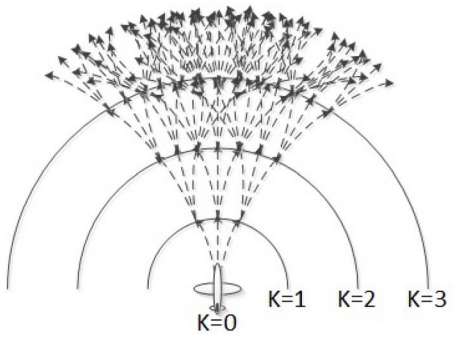


Fig. 10. Exponential growth of decisions and transitions to consider as the horizon increases.

TABLE II
Γ_{Ave} OF POLICIES IN EXP I AND EXP II

	Exp I		Exp II	
Method	μ	σ	μ	σ
$F_Q(0.1)$	3.768	2.274	43.730	35.541
$F_Q(0.8)$	6.436	2.957	72.500	66.724
$F_Q(var)$	3.809	2.224	43.730	35.541
$F_{DP}(6)$	3.936	2.106	82.063	50.315

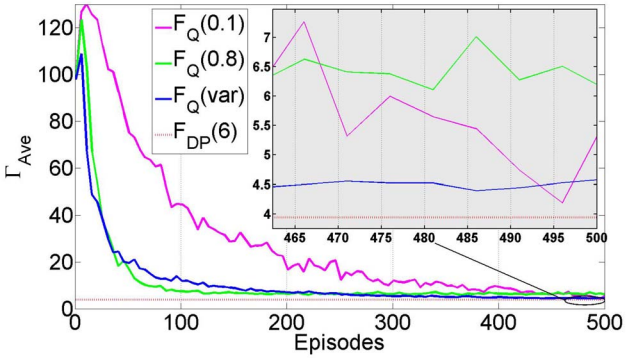


Fig. 11. Learning curve for Q-flocking in Exp I.

In part two of Exp I, we simulated Q-flocking using the parameters specified in Table I and tracked Γ as the policies evolved with each episode. Based on Fig. 11, $\Gamma_{\text{Ave}}(F_Q)$ decreases and converges to their respective values as the agents acquired more experience, along with better estimations of the state-action values. Using $F_Q(0.1)$ as the benchmark for comparing the learned policies, the proposed method for determining the learning parameter provides faster convergence when compared to the static rate $\alpha = 0.1$, especially during the initial transitional phase. Moreover, the variable learning rate converges to a lower Γ_{Ave} when compared to $F_Q(0.8)$. In essence, using variable learning rates offer faster convergence due to a higher learning rate during transitional phases, and convergences to a policy that incurs on average a lower cost compared to policies that utilize higher learning rates. The approximate convergence values for the Q-flocking policies after 1000 episodes are summarized in Table II.

The box plot in Fig. 12 illustrates the distributions of the average costs incurred by each of the policies. Since there were substantial overlapping between the distributions, two-sample t-tests were performed on the data sets in order to determine

TABLE III
T-TEST RESULTS FOR Q-FLOCKING IN EXP I

$H1_0$

Policy	T-values	DoF	Significance	Paired Differences Mean	Std. Dev.	99.9% Confidence Interval Lower	Upper
$F_Q(0.1)$	-1.712	1998	0.087	-0.168	2.192	-0.491	0.155
$F_Q(0.8)$	21.780	1998	1.529E-94	2.501	2.567	2.122	2.879
$F_Q(var)$	-1.309	1998	0.191	-0.127	2.165	-0.446	0.192

$H2_0$

Policy	T-values	DoF	Significance	Paired Differences Mean	Std. Dev.	99.9% Confidence Interval Lower	Upper
$F_Q(0.1)$	-1.712	1998	0.044	-0.168	2.192	-Inf	-0.006
$F_Q(0.8)$	21.780	1998	1.000	2.501	2.567	-Inf	2.690
$F_Q(var)$	-1.309	1998	0.095	-0.127	2.165	-Inf	0.033

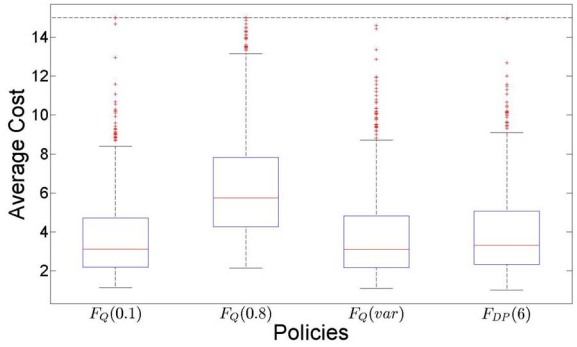


Fig. 12. Box plot of Γ for Q-flocking policies and $F_{\text{DP}}(6)$ in Exp I.

whether the learned policies were statistically different from $F_{\text{DP}}(6)$. The following two sets of hypotheses denoted as $H1$ and $H2$, were considered for each of the learned policies.

1) *$H1_0$:* $\Gamma(F_Q)$ and $\Gamma(F_{\text{DP}}(6))$ are the same.
2) *$H1_A$:* $\Gamma(F_Q)$ and $\Gamma(F_{\text{DP}})$ are not the same.
3) *$H2_0$:* $\Gamma(F_Q)$ is larger than $\Gamma(F_{\text{DP}}(6))$.
4) *$H2_A$:* $\Gamma(F_Q)$ is smaller than $\Gamma(F_{\text{DP}}(6))$.

The subscript 0 represents the null hypothesis, while the subscript A represents the alternative.

The t-test results for Exp I are summarized in Table III. The first column lists the learned policies using their respective learning parameters. In the second column, the t-values pertaining to each policy are shown. The t-values indicate the difference between the means of the two samples such that the larger the t-value, the larger the difference between the means. Next, the third column shows the degrees of freedom, which relates to the number of values in the data sets; more data means higher degrees of freedom, as well as smaller sampling error. Both t-value and degrees of freedom are used to determine the significance of the test, commonly known as the p-value. In practice, p-values smaller than 0.001 (i.e., the significance level) suggests very strong evidence against the null hypothesis H_0. For completeness, the paired difference results including the mean, the unpooled estimated standard deviation, and the confidence intervals [i.e., $100 \times (1 - 0.001)\%$] are also included in Table III.

According to the results of the $H1$ test shown in Table III, the p-values for $F_Q(0.1)$ and $F_Q(\text{var})$ are both higher than 0.001, while the p-value for $F_Q(0.8)$ is less than 0.001. This means that there are no significant differences between the distribution of $\Gamma(F_{\text{DP}}(6))$ and the distribution of both $\Gamma(F_Q(0.1))$ and $\Gamma(F_Q(\text{var}))$. In other words, $F_Q(0.1)$ and $F_Q(\text{var})$ are both comparable to $F_{\text{DP}}(6)$ in performance. Based on the results of the $H2$ test, we can confirm that on average $F_Q(0.8)$ incurred higher costs in comparison to $F_{\text{DP}}(6)$.

B. Experiment II

The aim of Exp II was to demonstrate that with Q-flocking the followers would be able to adapt their policies from Exp I to the new environment, defined by the model M_2. In accordance with the QDP approach, which computes policies offline, the F_{DP} policies remained unchanged in Exp II even though the environment has. This implies that followers relying on F_{DP} may incur a higher cost in an environment that F_{DP} was not optimize for.

Similar to Exp I, we simulated Q-flocking and tracked Γ_{Ave} for each of the policies as they evolved with each episode in the new environment. Likewise, we evaluated the F_{DP} policies with M_2, where in addition to stochasticity in the roll angle dynamics and the airspeed, we introduced nonsymmetrical noises into the model. The additional noises were used to simulate disturbances such as constant winds, or mechanical damage that occurs mid-flight, which may lead to weight and balance issues. Such disturbances can alter the flight characteristics of the UAVs. It should be noted that the additional disturbances were only applied to the followers, since it made more sense to show that the leader and follower would, in practice, experience different disturbances.

As shown in Fig. 13, the F_{DP} policies generated between 40 and 60 backups seem to incur the lowest cost in the new environment. However, to remain in line with the QDP approach, $F_{\text{DP}}(6)$ from Exp I was used as the benchmark in Exp II.

According to the learning curves shown in Fig. 14 and the convergence values compiled in Table II, $\Gamma_{\text{Ave}}(F_Q(0.1))$, $\Gamma_{\text{Ave}}(F_Q(0.8))$, and $\Gamma_{\text{Ave}}(F_Q(\text{var}))$ all converged to a lower

TABLE IV
T-TEST RESULTS FOR Q-FLOCKING IN EXP II

$H1_0$

				Paired Differences			
						99.9% Confidence Interval	
Policy	T-values	DoF	Significance	Mean	Std. Dev.	Lower	Upper
$F_Q(0.1)$	-19.678	1998	6.165E-79	-38.333	43.559	-44.752	-31.913
$F_Q(0.8)$	-3.619	1998	3.037E-04	-9.563	59.092	-18.271	-0.854
$F_Q(var)$	-20.037	1998	1.577E-81	-37.840	42.228	-44.063	-31.616

$H2_0$

				Paired Differences			
						99.9% Confidence Interval	
Policy	T-values	DoF	Significance	Mean	Std. Dev.	Lower	Upper
$F_Q(0.1)$	-19.678	1998	3.082E-79	-38.333	43.559	-Inf	-35.127
$F_Q(0.8)$	-3.619	1998	1.518E-04	-9.563	59.092	-Inf	-5.214
$F_Q(var)$	-20.037	1998	7.887E-82	-37.840	42.228	-Inf	-34.732

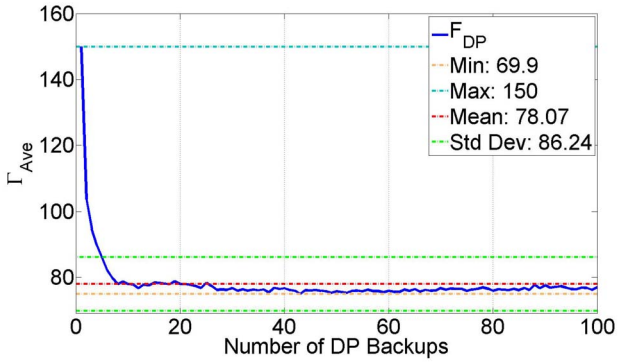


Fig. 13. Γ_{Ave} per number of DP backups for the QDP approach in Exp II.

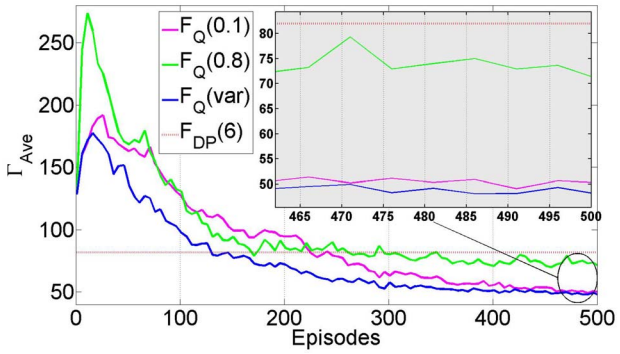


Fig. 14. Learning curve for Q-flocking in Exp II.

value than $\Gamma_{\text{Ave}}(F_{\text{DP}}(6))$. This means that by using the Q-flocking approach, the followers were able to adapt their policies to M_2, and as a result, their policies incurred a lower average cost than $F_{\text{DP}}(6)$. Furthermore, the learning curves show $F_Q(\text{var})$ converging faster during the initial readjustment phase. The readjustment phase (e.g., episodes 1–200) is when the agents have to relearn how to behave in a new environment, thereby causing their performance to suffer temporarily [59].

T-tests using the same hypotheses as proposed in Exp I were performed to compare the policies learned in Exp II to $F_{\text{DP}}(6)$. According to the box plots shown in Fig. 15 and the

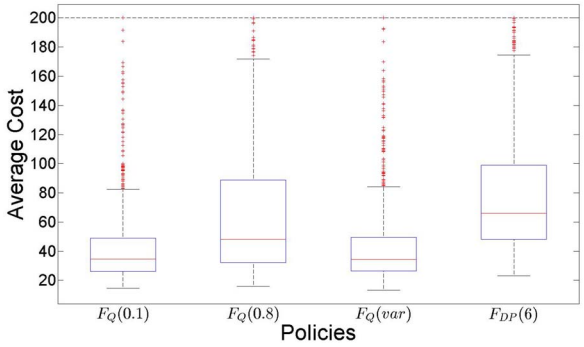


Fig. 15. Box plot of Γ for Q-flocking policy and $F_{\text{DP}}(6)$ in Exp II.

t-test results compiled in Table IV, we can confirm with statistically strong evidence (i.e., $p < 0.001$) that the distributions of $\Gamma(F_Q(0.1))$, $\Gamma(F_Q(0.8))$, and $\Gamma(F_Q(\text{var}))$ are significantly different from $\Gamma_{\text{Ave}}(F_{\text{DP}}(6))$, thus the learned policies must be different from $F_{\text{DP}}(6)$. In addition, there is significant evidence ($p < 0.001$) to confirm that all of the learned policies have incurred average costs that are lower in comparison to $F_{\text{DP}}(6)$.

C. Discussion

The results from Exp I demonstrate that by using Q-flocking, the followers were able to learn policies that facilitate flocking with a single leader, while operating in a simulated stochastic environment as defined by M_1. In addition, the *t*-test results confirmed that there are no significant differences between the costs incurred by $F_Q(0.1)$ and $F_Q(\text{var})$ in comparison to $F_{\text{DP}}(6)$. This means that $F_Q(0.1)$ and $F_Q(\text{var})$ have both converged to a near optimal policy, since according to the QDP approach, $F_{\text{DP}}(6)$ is the optimal control policy (lowest cost) for an environment defined by M_1. Consequently, we have shown empirically that Q-flocking can converge to a near optimal policy provided that enough time is allotted for the policies to converge. The learning curves from Exp I showed that in comparison to static learning parameters, the proposed variable learning parameter offers faster convergence during transitional phases without converging to policies that incur a higher cost (e.g., policies that used $\alpha = 0.8$).

The results from Exp II demonstrate that by using Q-flocking, the followers were able to adapt their policies to the new environment (i.e., M_2), and incur on average a lower cost in comparison to $F_{\mathrm{DP}}(6)$. Consequently, this validates the feasibility of using Q-flocking to enable the followers to learn and adapt their flocking policies in a nonstationary stochastic environment.

VII. Conclusion

In this paper, we applied Q-learning to flocking so that agents modeled as small fixed-wing UAV can learn how to flock in a simulated nonstationary stochastic environment. We formulated a model-free RL flocking framework for fixed-wing UAVs based on [29], and proposed an algorithm called Q-flocking to solve the RL problem. The algorithm is based on Peng's $Q(\lambda)$ augmented with a variable learning parameter.

According to the simulation results, the agents successfully applied Q-flocking to learn control policies that facilitate flocking in a leader–follower topology, while operating in a nonstationary stochastic environment. Furthermore, the results show that under the same simulation conditions, the policies learned with either a low α or the proposed variable α exhibit statistically comparable performances to the optimal control policy generated using the QDP approach.

As UAVs become more and more prevalent in military and industrial applications, there is a drive toward cooperative multi-UAV systems in order to benefit from economies of scale. For such systems to operate effectively, individual agents will require more than preprogrammed rules and good controllers; they will need tools for learning how to execute complex tasks and learn to adapt to unfamiliar situations. Consequently, the integration of RL and flocking with fixed-wing UAVs in a simulated nonstationary stochastic environment is a step toward developing intelligent agents that can learn to flock in the physical world.

Future work in this area includes applying better exploration algorithms and other forms of adaptive learning rates in order to speed up the learning process [60]. In addition, for practicality, the state-space will need to be reduced by restructuring the problem using function approximation methods, while boundary conditions have to also be considered to ensure safe exploration, and lastly, additional tools such as building a model from samples and planning ahead with the learned model will provide realistic means of operating in the physical world.

Acknowledgment

The authors would like to thank Dr. S. Quintero for his help with the experiments.

References

[1] B. Lerner, *UAVs and Force: Current Debates and Future Trends in Technology, Policy and the Law*, Center for Security Policy, Washington, DC, USA, 2013.

[2] K. Anderson, "Rise of the drones: Unmanned systems and the future of war," *Hearings Before the Subcommittee on National Security and Foreign Affairs, 111th Cong., 2nd Sess.*, 2010. [Online]. Available: https://fas.org/irp/congress/2010_hr/drones1.pdf

[3] J. P. How *et al.*, "Increasing autonomy of UAVs," *IEEE Trans. Robot. Autom.*, vol. 16, no. 2, pp. 43–51, Jun. 2009.

[4] X. Liu *et al.*, "UAV-based low-altitude aerial photogrammetric application in mine areas measurement," in *Proc. 2nd Int. Workshop Earth Observ. Remote Sens. Appl. (EORSA)*, Shanghai, China, Jun. 2012, pp. 240–242.

[5] P. Tokekar, J. V. Hook, D. Mulla, and V. Isler, "Sensor planning for a symbiotic UAV and UGV system for precision agriculture," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Tokyo, Japan, Nov. 2013, pp. 5321–5326.

[6] E. Adamey and U. Ozguner, "A decentralized approach for multi-UAV multitarget tracking and surveillance," in *Proc. SPIE Ground/Air Multisensor Interoperability Integr. Netw. Persistent ISR III*, vol. 8389. 2012, p. 15.

[7] T. Shima and S. J. Rasmussen, *UAV Cooperative Decision and Control: Challenges and Practical Approaches*. Philadelphia, PA, USA: Soc. Ind. Appl. Math., 2009.

[8] Z. Wang and M. Perc, "Aspiring to the fittest and promotion of cooperation in the prisoner's dilemma game," *Phys. Rev. E*, vol. 82, Aug. 2010, Art. ID 021115.

[9] Z. Wang, Z. Wang, X. Zhu, and J. J. Arenzon, "Cooperation and age structure in spatial games," *Phys. Rev. E*, vol. 85, Jan. 2012, Art. ID 011149.

[10] Y. U. Cao, A. S. Fukunaga, and A. Kahng, "Cooperative mobile robotics: Antecedents and directions," *Auton. Robots*, vol. 4, no. 1, pp. 7–27, 1997.

[11] L. E. Parker, "Multiple mobile robot teams, path planning and motion coordination," in *Encyclopedia of Complexity and Systems Science*, R. A. Meyers, Ed. New York, NY, USA: Springer, 2009, pp. 5783–5800.

[12] G. Acampora, M. Gaeta, V. Loia, and A. V. Vasilakos, "Interoperable and adaptive fuzzy services for ambient intelligence applications," *ACM Trans. Auton. Adapt. Syst.*, vol. 5, no. 2, May 2010, Art. ID 8.

[13] Y. Tan and Z.-Y. Zheng, "Research advance in swarm robotics," *Defen. Technol.*, vol. 9, no. 1, pp. 18–39, 2013.

[14] C. W. Reynolds, "Flocks, herds and schools: A distributed behavioral model," in *Proc. 14th Annu. Conf. Comput. Graph. Interact. Tech. SIGGRAPH*, New York, NY, USA, 1987, pp. 25–34.

[15] G. Colqui, M. Tomita, T. Hattori, and Y. Chigusa, "New video synthesis based on flocking behavior simulation," in *Proc. 3rd Int. Symp. Commun. Control Signal Process. (ISCCSP)*, St. Julian's, Malta, 2008, pp. 936–941.

[16] H. M. La and W. Sheng, "Distributed sensor fusion for scalar field mapping using mobile sensor networks," *IEEE Trans. Cybern.*, vol. 43, no. 2, pp. 766–778, Apr. 2013.

[17] S. H. Semnani and O. A. Basir, "Semi-flocking algorithm for motion control of mobile sensors in large-scale surveillance systems," *IEEE Trans. Cybern.*, vol. 45, no. 1, pp. 129–137, Jan. 2015.

[18] R. Olfati-Saber, "Flocking for multi-agent dynamic systems: Algorithms and theory," *IEEE Trans. Autom. Control*, vol. 51, no. 3, pp. 401–420, Mar. 2006.

[19] N. Moshtagh and A. Jadbabaie, "Distributed geodesic control laws for flocking of nonholonomic agents," *IEEE Trans. Autom. Control*, vol. 52, no. 4, pp. 681–686, Apr. 2007.

[20] W. Dong, "Flocking of multiple mobile robots based on backstepping," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 41, no. 2, pp. 414–424, Apr. 2011.

[21] X. Wang, J. Qin, and C. Yu, "ISS method for coordination control of nonlinear dynamical agents under directed topology," *IEEE Trans. Cybern.*, vol. 44, no. 10, pp. 1832–1845, Oct. 2014.

[22] J. Qin, W. X. Zheng, and H. Gao, "Coordination of multiple agents with double-integrator dynamics under generalized interaction topologies," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 42, no. 1, pp. 44–57, Feb. 2012.

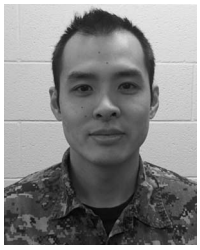
[23] A. E. Turgut, H. Çelikkanat, F. Gökçe, and E. Şahin, "Self-organized flocking with a mobile robot swarm," in *Proc. 7th Int. Joint Conf. Auton. Agents Multiagent Syst. (AAMAS)*, vol. 1. 2008, pp. 39–46.

[24] M. J. Mataric, "Interaction and intelligent behavior," Ph.D. dissertation, Dept. Electr. Eng. Comput. Sci., Massachusetts Inst. Technol., Cambridge, MA, USA, 1994.

[25] A. T. Hayes and P. Dormiani-Tabatabaei, "Self-organized flocking with agent failure: Off-line optimization and demonstration with real robots," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, vol. 4. Washington, DC, USA, 2002, pp. 3900–3905.

[26] A. Campo, S. Nouyan, M. Birattari, R. Groß, and M. Dorigo, "Negotiation of goal direction for cooperative transport," in *Ant Colony Optimization and Swarm Intelligence* (LNCS 4150), M. Dorigo *et al.*, Eds. Heidelberg, Germany: Springer, 2006, pp. 191–202.

[27] J. Welsby and C. Melhuish, "Autonomous minimalist following in three dimensions: A study with small-scale dirigibles," in *Proc. Towards Intell. Mobile Robots*, Manchester, U.K., 2001. [Online]. Available: http://apt.cs.manchester.ac.uk/ftp/pub/TR/UMCS-01-4-1.html
[28] S. Hauert *et al.*, "Reynolds flocking in reality with fixed-wing robots: Communication range vs. maximum turning rate," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, San Francisco, CA, USA, 2011, pp. 5015–5020.
[29] S. A. P. Quintero, G. E. Collins, and J. P. Hespanha, "Flocking with fixed-wing UAVs for distributed sensing: A stochastic optimal control approach," in *Proc. Amer. Control Conf. (ACC)*, Washington, DC, USA, 2013, pp. 2025–2031.
[30] C. Virágh *et al.*, "Flocking algorithm for autonomous flying robots," *Bioinspir. Biomim.*, vol. 9, no. 2, 2014, Art. ID 025012.
[31] G. Vasarhelyi *et al.*, "Outdoor flocking and formation flight with autonomous aerial robots," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2014, pp. 3866–3873.
[32] E. L. Hayes, "Machine learning for intelligent control: Application of reinforcement learning techniques to the development of flight control systems for miniature UAV rotorcraft," M.S. thesis, Dept. Mech. Eng., Univ. Canterbury, Christchurch, New Zealand, 2013.
[33] M. Graña, B. Fernandez-Gauna, and J. M. Lopez-Guede, "Cooperative multi-agent reinforcement learning for multi-component robotic systems: Guidelines for future research," *Paladyn*, vol. 2, no. 2, pp. 71–81, 2011.
[34] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *Int. J. Robot. Res.*, vol. 32, no. 11, pp. 1238–1274, 2013.
[35] A. V. Vasilakos and G. I. Papadimitriou, "A new approach to the design of reinforcement schemes for learning automata: Stochastic estimator learning algorithm," *Neurocomputing*, vol. 7, no. 3, pp. 275–297, 1995.
[36] M. A. Khan, H. Tembine, and A. V. Vasilakos, "Game dynamics and cost of learning in heterogeneous 4G networks," *IEEE J. Sel. Areas Commun.*, vol. 30, no. 1, pp. 198–213, Jan. 2012.
[37] P. P. Demestichas, V. A. G. Stavroulaki, L. M. Papadopoulou, A. V. Vasilakos, and M. E. Theologou, "Service configuration and traffic distribution in composite radio environments," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 34, no. 1, pp. 69–81, Feb. 2004.
[38] P. B. F. Duarte, Z. M. Fadlullah, A. V. Vasilakos, and N. Kato, "On the partially overlapped channel assignment on wireless mesh network backbone: A game theoretic approach," *IEEE J. Sel. Areas Commun.*, vol. 30, no. 1, pp. 119–127, Jan. 2012.
[39] L. Zhou, N. Xiong, L. Shu, A. Vasilakos, and S.-S. Yeo, "Context-aware middleware for multimedia services in heterogeneous networks," *IEEE Intell. Syst.*, vol. 25, no. 2, pp. 40–47, Mar./Apr. 2010.
[40] Y. Zeng, K. Xiang, D. Li, and A. V. Vasilakos, "Directional routing and scheduling for green vehicular delay tolerant networks," *Wireless Netw.*, vol. 19, no. 2, pp. 161–173, 2013.
[41] G. Wei, A. V. Vasilakos, Y. Zheng, and N. Xiong, "A game-theoretic method of fair resource allocation for cloud computing services," *J. Supercomput.*, vol. 54, no. 2, pp. 252–269, 2010.
[42] C. Watkins, "Learning from delayed rewards," Ph.D. dissertation, Dept. Comput. Sci., Univ. Cambridge, Cambridge, U.K., 1989.
[43] K. Morihiro *et al.*, "Reinforcement learning scheme for flocking behavior emergence," *J. Adv. Comput. Intell. Intell. Informat.*, vol. 11, no. 2, pp. 155–161, 2007.
[44] K. Morihiro, T. Isokawa, H. Nishimura, and N. Matsui, "Characteristics of flocking behavior model by reinforcement learning scheme," in *Proc. Int. Joint Conf. SICE-ICASE*, Busan, Korea, Oct. 2006, pp. 4551–4556.
[45] H. M. La, R. Lim, and W. Sheng, "Multirobot cooperative learning for predator avoidance," *IEEE Trans. Control Syst. Technol.*, vol. 23, no. 1, pp. 52–63, Jan. 2015.
[46] J. Peng and R. J. Williams, "Incremental multi-step Q-learning," *Mach. Learn.*, vol. 22, no. 1, pp. 283–290, 1996.
[47] R. S. Sutton and A. G. Barto, *Introduction to Reinforcement Learning*, 1st ed. Cambridge, MA, USA: MIT Press, 1998.
[48] R. E. Bellman, *Dynamic Programming*. Princeton, NJ, USA: Princeton Univ. Press, 1957.
[49] R. W. Beard and T. W. McLain, *Small Unmanned Aircraft: Theory and Practice*. Princeton, NJ, USA: Princeton Univ. Press, 2012.
[50] T. M. Foster and W. J. Bowman, "Dynamic stability and handling qualities of small unmanned-aerial-vehicles," Ph.D. dissertation, Dept. Mech. Eng., Brigham Young Univ., Provo, UT, USA, 2005.
[51] M. van Otterlo and M. Wiering, "Reinforcement learning and Markov decision processes," in *Reinforcement Learning: State of the Art* (Adaptation, Learning, and Optimization), vol. 12. Heidelberg, Germany: Springer, 2012, pp. 3–42.
[52] W. Dabney and A. G. Barto, "Adaptive step-size for online temporal difference learning," in *Proc. Conf. Artif. Intell. (AAAI)*, Toronto, ON, Canada, 2012, pp. 872–878.
[53] C. J. C. H. Watkins and P. Dayan, "Technical note: Q-learning," *Mach. Learn.*, vol. 8, nos. 3–4, pp. 279–292, 1992.
[54] S. P. M. Choi, D.-Y. Yeung, and N. L. Zhang, "Hidden-mode Markov decision processes for nonstationary sequential decision making," in *Sequence Learning* (LNCS 1828), R. Sun and C. L. Giles, Eds. Heidelberg, Germany: Springer, 2001, pp. 264–287.
[55] T. Lane, M. Ridens, and S. Stevens, "Reinforcement learning in non-stationary environment navigation tasks," in *Advances in Artificial Intelligence* (LNCS 4509), Z. Kobti and D. Wu, Eds. Heidelberg, Germany: Springer, 2007, pp. 429–440.
[56] H. Kushner and G. G. Yin, *Stochastic Approximation and Recursive Algorithms and Applications* (Stochastic Modelling and Applied Probability). New York, NY, USA: Springer, 2003.
[57] J. Huang, B. Yang, and D.-Y. Liu, "A distributed Q-learning algorithm for multi-agent team coordination," in *Proc. 4th Int. Conf. Mach. Learn. Cybern.*, vol. 1. Guangzhou, China, Aug. 2005, pp. 108–113.
[58] M. Tan, "Multi-agent reinforcement learning: Independent vs. cooperative agents," in *Proc. 10th Int. Conf. Mach. Learn.*, Amherst, MA, USA, 1993, pp. 330–337.
[59] B. C. Da Silva, E. W. Basso, A. L. C. Bazzan, and P. M. Engel, "Dealing with non-stationary environments using context detection," in *Proc. 23rd Int. Conf. Mach. Learn.*, Pittsburgh, PA, USA, 2006, pp. 217–224.
[60] I. Noda, "Recursive adaptation of stepsize parameter for non-stationary environments," in *Adaptive and Learning Agents* (LNCS 5924), M. E. Taylor and K. Tuyls, Eds. Heidelberg, Germany: Springer, 2010, pp. 74–90.



Shao-Ming Hung (M'15) received the M.A.Sc. degree in electrical and computer engineering from the Royal Military College of Canada, Kingston, ON, Canada.

He is currently an Aerospace Engineering Officer with the Royal Canadian Air Force. His current research interests include autonomous robots and machine learning.

Sidney N. Givigi (SM'14) received the Ph.D. degree in electrical and computer engineering from Carleton University, Ottawa, ON, Canada.

He is currently an Associate Professor with the Department of Electrical and Computer Engineering, Royal Military College of Canada, Kingston, ON, Canada. His current research interests include autonomous systems and robotics.