AIAA AVIATION Forum
June 15-19, 2020, VIRTUAL EVENT
AIAA AVIATION 2020 FORUM

10.2514/6.2020-2909

UAS Conflict Resolution in Continuous Action Space Using Deep Reinforcement Learning

Jueming Hu*[1], Xuxi Yang†[2], Weichang Wang‡[1], Peng Wei§[2], Lei Ying¶[3], and Yongming Liu‖[1]
[1]*Arizona State University, Tempe, AZ, USA 85281*
[2]*Iowa State University, Ames, IA, USA 50011*
[3]*University of Michigan, Ann Arbor, Ann Arbor, MI, USA 48109*

Ensuring safety and providing obstacle conflict alerts to small unmanned aircraft is vital to their integration into civil airspace. There are many techniques for real-time robust drone guidance, but many of them need expensive computation time or large memory requirements, which is not applicable to deploy onboard of an aircraft with limited computation resources. To provide a safe and efficient computational guidance of operations for unmanned aircraft, we provide a framework using deep reinforcement learning algorithm to guide autonomous UAS to their destinations while avoiding the static and moving obstacles through continuous control. After offline training, the model only requires less than 100KB of memory, and the online computation for the conflict resolution advisory only takes 2ms. For the algorithm verification and validation, an airspace simulator is built in Python and numerical experiments show that the trained model can provide accurate and robust guidance with the environment uncertainty.

Nomenclature

s_t	=	aircraft state at time step t
a_t	=	action at time step t
$V(s)$	=	value function for state s
$R(s_t, a_t)$	=	reward function at time step t
d_g	=	the distance from the UAS to its goal
v_x	=	UAS velocity in x-axis
v_y	=	UAS velocity in y-axis
P_y	=	obstacle's position in y-axis

I. Introduction

From delivery drones to autonomous electrical vertical take-off and landing (eVTOL) passenger aircraft, modern unmanned aircraft systems (UAS) can perform many different tasks efficiently, including goods delivery, surveillance, public safety, weather monitoring, disaster relief, search and rescue, traffic monitoring, videography, and air transportation [1, 2]. With the fast improvement of modern technology and the dramatic decrease in the prime cost of UAVs, the amount of UAVs will be growing rapidly within the next 20 years [3, 4]. With the fast development of UAVs, more drone operations are likely to occur in metropolitan areas close to buildings or airports, thus advanced conflict resolution tools and decision-making approaches will be needed to manage this influx of UAS under UAS Traffic Management (UTM) [2].

Jueming Hu and Xuxi Yang contributed equally to this paper.

*PhD student in Mechanical Engineering, School for Engineering of Matter, Transport & Energy, Arizona State University, 501 E Tyler Mall, ENGRC 476, Tempe, AZ 85287, Jueming.Hu@asu.edu; Student Member.

†Graduate Research Assistant, Department of Aerospace Engineering, xuxiyang@iastate.edu. Student Member AIAA.

‡PhD student in School of Electrical, Computer & Energy Engineering, Arizona State University, 650 E Tyler Mall, GWC 431, Tempe, AZ 85281

§Assistant Professor, Department of Aerospace Engineering, pwei@iastate.edu. Senior Member AIAA.

¶Professor, Electrical Engineering and Computer Science Department of the University of Michigan, Ann Arbor, 1301 Beal Avenue, 4423 EECS, Ann Arbor, MI 48109-2122, leiying@umich.edu,

‖Corresponding Author, Professor, School for Engineering of Matter, Transport & Energy, Arizona State University, 501 E Tyler Mall, ENGRC 419, Tempe, AZ 85287, 480-965-6883, Yongming.liu@asu.edu; Associate fellow AIAA.

In future UTM, the technical challenge is to provide concepts, technologies, and procedures that enable safe and efficient flight operations for orders-of-magnitude UAS, to avoid conflicts with other UAS, buildings, and static obstacles. For UAS conflict resolution systems, research efforts can be divided into centralized algorithms and decentralized algorithms.

In centralized methods, the conflicts between aircraft are resolved by a central supervising controller (centralized Air Traffic Controller). Under such scenario, the state of each aircraft, the obstacle information, the trajectory constraint as well as the terminal condition are known to the centralized controller, which in return designs the individual whole trajectory for all aircraft before the flight, typically by formulating it as an optimal control problem. These methods can be based on semidefinite programming [5], nonlinear programming [6, 7], mixed integer linear programming [8–11], mixed integer quadratic programming [12], sequential convex programming [13, 14], second-order cone programming [15], evolutionary techniques [16, 17], and particle swarm optimization [18]. These centralized methods often pursue the global optimum for all the aircraft. However, as the number of aircraft grows, the computation time of these methods typically scales exponentially. Moreover, these centralized planning approaches typically need to be re-run, as new information in the environment is updated (e.g. a new aircraft enters the airspace).

On the other hand, decentralized methods scale better with respect to the number of agents and are more robust since they do not possess a single point of failure [19], which relies on aircraft being sufficiently equipped and automated that they can operate relatively independently from the existing ATC system and are therefore not subject to its capacity limits [20–22]. Model Predictive Control [23, 24] can be used to solve the collision avoidance problem but the computation load is relatively high. Potential field method [25] is computationally fast. However, a navigation function is required to make it a complete path planner [26, 27], which involves discretizing the configuration space. Geometry based algorithms [28–31] can be applied for conflict resolution and the computation time only grows linearly with the increasing number of aircraft. The drawback of these geometric approaches is that it cannot look ahead for more than one step and the outcome can be local optimal in the view of the global trajectory. Monte Carlo Tree Search algorithm [32, 33] can look ahead for several time steps, but the aircraft can only adopt several discretized actions and the online computation load is relatively high. With offline training, the reinforcement learning [34, 35] for conflict resolution can be executed online efficiently with a promising performance.

However, to the best of our knowledge, there are not many papers discussing using deep RL algorithms for UAS in high-level contexts, such as navigation, monitoring, or other complex task-based applications in continuous action space. This paper presents a method for using deep reinforcement learning to allow the UAS to navigate successfully in continuous action space. The benefit of calculating in continuous space is that there is no need to discretize the state space or smooth results after the algorithm running. Additionally, accuracy is improved comparing with discretized state/action space. The model can also take uncertainty into consideration. The development of such a model is very valuable for UAS traffic management.

The paper is organized as follows. In Section II, the backgrounds of Markov Decision Process and Deep Reinforcement Learning are introduced. Section III presents the model formulation using Markov Decision Process for UAS conflict resolution in continuous action space. In Section IV, the numerical experiments are presented to show the capability of the proposed approach to make the UAS learn to avoid conflict. Comparisons between the different obstacle scenarios are provided. Section V concludes this paper.

II. Background

In this section, we briefly review the backgrounds of Markov Decision Process (MDP) and Reinforcement Learning (DRL).

A. Markov Decision Process (MDP)

Since the 1950s, MDPs [36] have been well studied and applied to a wide area of disciplines [37–39], including robotics [40, 41], automatic control [42], economics, and manufacturing. In an MDP, the agent may choose any action a that is available based on current state s at each time step. The process responds at the next time step by moving into a new state s' with certain transition probability and gives the agent a corresponding reward r.

More precisely, the MDP includes the following components:

1) The state space $\mathcal{S}$ which consists of all the possible states.
2) The action space $\mathcal{A}$ which consists of all the actions that the agent can take.
3) Transition function $\mathcal{T}(s_{t+1}|s_t, a_t)$ which describes the probability of arriving at state s_{t+1}, given the current state s_t and action a_t.

4) The reward function $\mathcal{R}(s_t, a_t, s_{t+1})$ which decides the immediate reward (or expected immediate reward) received after transitioning from state s to state s', due to action a. In general, the reward will depend on the current state, current action, and the next state. However, the reward function may only depend on the current state s_t, which will be the case in this paper.
5) A discount factor $\gamma \in [0, 1]$ which decides the preference for immediate reward versus future rewards. Setting the discount factor less than 1 is also beneficial for the convergence of cumulative reward.

In an MDP problem, a policy π is a mapping from the state to a distribution over actions (known as stochastic policy)

$$\pi : \mathcal{S} \rightarrow \text{Prob}(\mathcal{A})$$

or to one specific action (known as deterministic policy)

$$\pi : \mathcal{S} \rightarrow \mathcal{A} \tag{1}$$

The goal of MDP is to find an optimal policy π^* that, if followed from any initial state, maximizes the expected cumulative immediate rewards:

$$\pi^* = \underset{\pi}{\text{argmax}} \mathbb{E}[\sum_{t=0}^{T} R(s_t, a_t)|\pi] \tag{2}$$

Q-function and value function are two important concepts in MDP. The optimal Q-function $Q^*(s, a)$ represents the expected cumulative reward received by an agent starting from state s and picks action a, and chooses action optimally afterward. Therefore, $Q^*(s, a)$ is an indication of how good it is for an agent to pick action a while being at state s. The optimal value function $V*(s)$ denotes the maximum expected cumulative reward when starting from state s, which can be expressed as the maximum of $Q^*(s, a)$ over all possible actions:

$$V^*(s) = \max_a Q^*(s, a), \quad \forall s \in \mathcal{S} \tag{3}$$

B. Deep Reinforcement Learning

Reinforcement learning [43] is an efficient algorithm to solve the MDP problem. With the advent of deep learning, Deep Reinforcement Learning (DRL) achieved much success recently, including game of GO [44], Atari games [45, 46], Warcraft [47]. In general, deep reinforcement learning can be divided into value-based learning [45, 48] and policy-based algorithm [49–51]. In this paper, we consider a policy-based DRL algorithm to generate policies for the agent. Comparing with value-based DRL algorithms, the policy-based algorithm can learn stochastic policies, which is beneficial when there is uncertainty in the environment.

Typically, the policy-based algorithm uses function approximator such as neural network to approximate the policy $\pi(s)$, where the input is the current state and output is the probability of each action (for discrete action space) or an action distribution (for continuous action space). After each trajectory τ, the algorithm updates the parameter of the function approximator to maximize the cumulative reward using gradient ascent:

$$\nabla_\theta J(\pi_\theta) = \underset{\tau \sim \pi_\theta}{\text{E}} \left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) R(\tau) \right] \tag{4}$$

where $J(\pi_\theta)$ is the expected cumulative reward of policy π parameterized by θ, $\pi_\theta(a_t|s_t)$ is the probability of action a_t for state s_t, and $R(\tau)$ is the cumulative reward gathered by the agent for state trajectory τ in one episode. The general idea of Equation 4 is to reduce the probability of sampling an action that leads to a lower return and increase the probability of action leads to higher reward. But one issue is the cumulative reward usually has very high variance, which makes the convergence speed to be slow. To address this issue, researchers proposes actor-critic algorithm [43] where a critic function is introduced to approximate the state value function $V(s_t)$. By subtracting the value function $V(s_t)$, the expectation of the gradient keeps unchanged and the variance is reduced dramatically:

$$\nabla_\theta J(\pi_\theta) = \underset{\tau \sim \pi_\theta}{\text{E}} \left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) (R(\tau) - V(s_t)) \right] \tag{5}$$

where the function approximator $V(s_t)$ is updated to approximate the value function for s_t.

III. Markov Decision Process Formulation

The objective of the proposed conflict resolution algorithm is to find the shortest path for a UAS to its goal while avoiding conflict with other UAS and static obstacles. Guiding the UAS to its destination is a discrete-time stochastic control process that can be formulated as a Markov Decision Process (MDP). In the following subsections, we introduce the MDP formulation by describing its action space, state space, terminal state, and reward function. For this work, a method of deep reinforcement learning, proximal policy optimization algorithm developed in [51], is adopted. The reason and details are also introduced in this Section.

A. Action Space

The action in this study represents the change in the heading angle for the controlled UAS at each time step. Since UAS is more flexible than manned aircraft and there is no available regularization on UAS heading change, the action space is set to $\mathcal{A} = [-\pi, \pi]$. More specifically, at each time step, the agent will select an action $a \in \mathcal{A}$, and change its heading angle ψ:

$$\psi_{t+1} = \psi_t + a \tag{6}$$

In real-world applications, however, making a sharp turn is usually not desirable for the controlling of a UAS. Thus a penalty of large heading change due to the power consumption may be considered in future work.

B. State Space

The state of the formulated MDP includes all the necessary information for an aircraft to make optimal actions. In this paper we let s_t denote the agent's state at time t. The state can be divided into two parts, that is $s_t = [s_t^0, s_t^1]$, where s_t^0 denotes the part that is related to the agent itself and the goal, and s_t^1 denotes the part related to the environment such as obstacles. To speed up the training of the DRL algorithm, we transform the state by following the robot-centric parameterization in [35], where the agent is located at the origin and the x-axis is pointing toward its goal.

More specifically, $s_t^0 = [d_g, v_x, v_y]$, where d_g is the agent's distance to goal, v_x, v_y denote the agent's velocity. The position of goal is added only in the simulations of moving obstacle avoidance. We use w_t to denote the information of environment (which represents moving/static obstacles in this paper) and set $s_t^1 = [w_t^1, w_t^2, ..., w_t^n]$. w_t^i indicates the information of obstacle i.

In the simulations of static obstacle avoidance, $w_t^i = [P_y^i, d_i]$, where P_y^i is the position in y-axis of obstacle i and d_i is the agent's distance to the center of obstacle i. P_y^i is introduced to help the agent learn the global optimal solution, for example, when approaching the obstacle, turn a small degree clockwise if P_y^i is positive, which means the agent is on the right side of the line passing the obstacle center and the goal.

In the simulations of moving obstacle avoidance, $w_t^i = [P_x^i, P_y^i, V_x^i, V_y^i, d_i]$, where V^i is the velocity of intruder i and d_i is the ratio of the distance between the agent and the intruder to the separation requirement. It should be noted that the agent and intruders are considered as a point mass in this study.

All the parameters are normalized when querying the neural network.

C. Terminal state

In the current study, the conflict is defined to be when the distance from the agent to the obstacle is less than a minimum separation distance. A risked-based anisotropic safety bound proposed in [52] can be considered in future work. To encourage exploration, the terminal state of this MDP includes two different types of states:

- Conflict state: $\begin{cases} \text{The distance from the agent to the static obstacle center is less than the summation of the obstacle size and minimum separation distance,} \\ \text{The distance from the agent to the intruder is less than the minimum separation distance.} \end{cases}$
- Goal state: the agent reaches the goal position.

D. Reward Function

To guide the agent to reach its goal and avoid conflict, the reward function is developed to award accomplishments while penalizing conflicts or not moving towards the goal.

In the simulations of static obstacle avoidance, the reward function, $R(s_t, a_t)$, is expressed as the following form, where we set a reward for the goal state and a penalty for the conflict state. For other states, the reward reflecting the

distance to the goal is used.

$$R(s,a) = \begin{cases} 10, & \text{if } s \text{ is goal state,} \\ -0.001 \cdot d_g - 0.05 - 16, & \text{if } s \text{ is conflict state,} \\ -0.001 \cdot d_g - 0.05, & \text{otherwise.} \end{cases} \tag{7}$$

In the simulations of intruder avoidance, the reward function, $R(s_t, a_t)$, is expressed in the following form, similar to Equation (7).

$$R(s,a) = \begin{cases} 65, & \text{if } s \text{ is goal state,} \\ -c_g \cdot d_g - c_0 - \sum_i c_{1i} \cdot e^{(-d_i/c_{2i}+c_{3i})^3+c_{4i}} - 20, & \text{if } s \text{ is conflict state,} \\ -c_g \cdot d_g - c_0 - \sum_i c_{1i} \cdot e^{(-d_i/c_{2i}+c_{3i})^3+c_{4i}}, & \text{otherwise.} \end{cases} \tag{8}$$

In this reward function, $c_g, c_0, c_{1i}, c_{2i}, c_{3i}, c_{4i}$ are coefficients of different cost, and should be balanced to help the agent learn conflict resolution and achieve the goal simultaneously.

The exponential formula with third order is to make the agent get large penalty when it is close to the minimum separation with the intruder. With the coefficients set in the three-intruder case in Section IV.B.2, the exponential function part, $-10 \cdot e^{(-d_i/20-0.7)^3+1}$, is shown as follows. When the distance from agent to obstacle/intruder decreases from 400m to 250 m, the reward function value will gradually decrease from 0 to -25. This reward setting is able to help the agent avoid conflicts with other intruders at a relatively early stage.

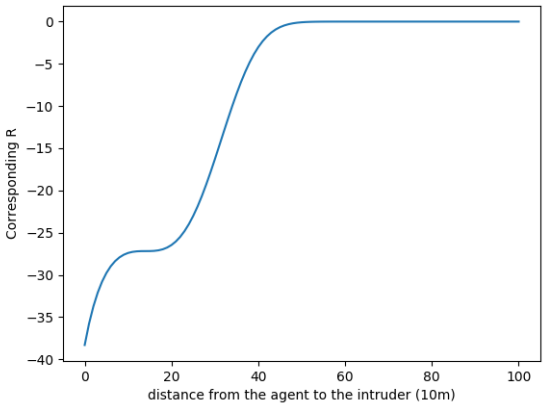


Fig. 1 Reward related to the distance from the agent to the intruder.

E. Proximal Policy Optimization Algorithm

One drawback of $\nabla_\theta J(\pi_\theta)$ proposed in [43], also shown in Eq. (5), is that one bad update can lead to large destructive effects and hinder the final performance of the model. Proximal Policy Optimization (PPO) algorithm [51] was proposed to solve this problem by introducing a policy changing ratio describing the change from previous policy to the new policy:

$$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)} \tag{9}$$

where θ_{old} and θ denote the network weights before and after update.

By restricting policy changing ratio in the range $[1-\epsilon, 1+\epsilon]$ with ϵ set to 0.2 in this paper, the PPO loss function for the actor and critic network is formulated as follows:

$$L_\pi(\theta) = -\mathop{\mathrm{E}}_{\tau\sim\pi_\theta}[\min(r_t(\theta)\cdot A_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\cdot A_t] - \beta \cdot H(\pi(\cdot|s_t)) \tag{10}$$

$$L_v = \|R_\tau - V(s_t)\| \tag{11}$$

where ϵ is a hyperparameter that bounds the policy changing ratio $r_t(\theta)$. In Equations (10) and (11), the advantage function $A_t := R_\tau - V(s_t)$ measures whether or not the action is better or worse than the policy's default behavior. Also, the policy entropy $\beta \cdot H(\pi(\cdot|s_t))$ is also added to the actor loss function which used to encourage exploration by discouraging premature convergence to sub-optimal deterministic polices.

In the implementation, we use two layers Multilayer perceptron (MLP) with 64 hidden units each for both actor and critic networks. Tanh function is chosen as the activation function for the hidden layers. A linear activation function is used for the output layer of the actor and critic network.

IV. Numerical Experiments

Numerical experiments are presented in this section to evaluate the proposed conflict resolution model in continuous action space. There are two simulation cases: static obstacle avoidance and moving obstacle avoidance. As for static obstacle avoidance, we investigate the performance on different obstacle shapes and sizes, and uncertainty in UAS operation. We also study the two-intruder and three-intruder cases for moving obstacle avoidance. In all the simulations, one pixel in the figure represents 10 m in the real world. The implementation of PPO algorithm is conducted by OpenAI Baselines [53].

A. Static Obstacle Avoidance

The simulation environment is the free flight airspace with 4km length and 4km width. The UAS speed is set to 20 m/s. During the training process, the starting position of the aircraft is randomly sampled from four edges of the airspace boundary and is an array of the integer type for simplification. The goal is located at (2500, 2500) m. In this experiment, we study two types of static obstacles: circular obstacle and rectangular obstacle, as shown in Fig. 2. The plus sign represents the goal position and the blue region represents the no-passing area, including the obstacle itself and a buffer for safety consideration. For each case, the deep reinforcement learning model is trained for 15 million time steps. The learning curves are shown in Fig. 3, which shows the cumulative reward is growing and the policy is converging to the optimal solution. To visualize the performance of the proposed conflict resolution model, we generate a testing set of 160 trajectories starting from different origins. The origin for testing is chosen every 100 m on each edge of the airspace boundary. Heading angles in 160 trajectories are collected and the heading angle is plotted every 15 timesteps.

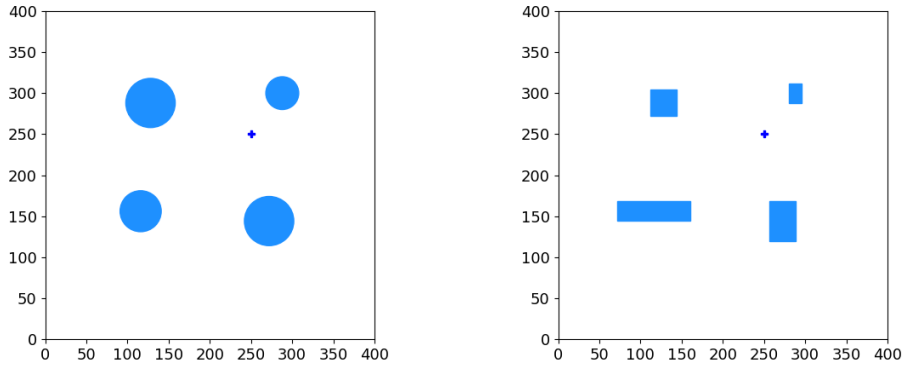
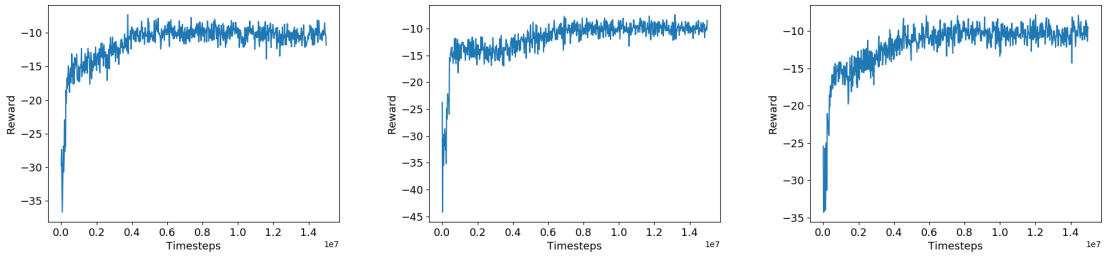


Fig. 2 (a) Circular obstacle environment. (b) Rectangular obstacle environment.

1. Circular Obstacle Avoidance

The static obstacle setup for this case study is shown in Fig. 2(a) and the testing result of 160 trajectories starting from different locations on the airspace boundary is shown in Fig. 4. The black arrow represents the agent's selected heading direction at each position.

From Fig. 4, it can be seen that the agent is selecting the heading angle pointing to the goal and tending to avoid



(a) (b) (c)

Fig. 3 The learning curve for (a) circular obstacle avoidance, (b) rectangular obstacle environment, and (c) circular obstacle avoidance with uncertainty.

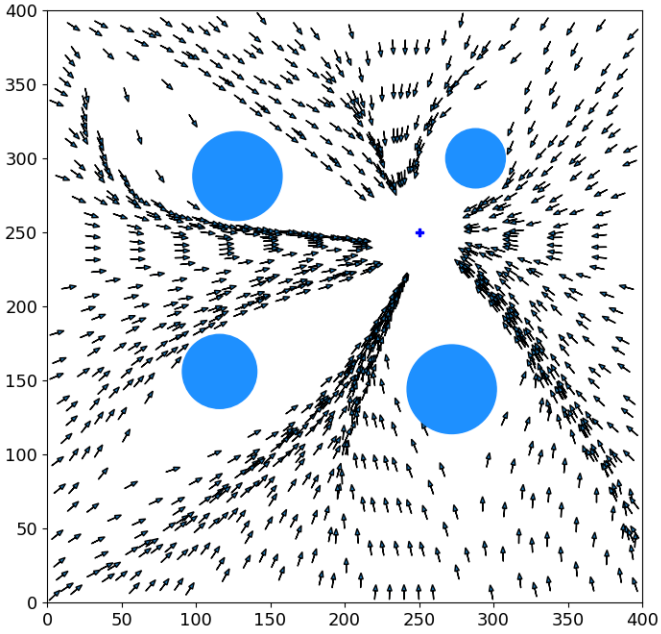


Fig. 4 Results of circular obstacles avoidance.

the no-passing region. Also, the agent chooses the optimal behavior according to the relative position of the agent, obstacle, and goal. When the agent is on the left side of the line passing the obstacle center and the goal, it takes a counterclockwise turn to avoid the obstacle. For the 160 generated trajectories in Fig. 4, there are six failures where the agent fails to maintain the separation with the static obstacle. For example, the most obvious failure is near the upper-left obstacle, where one heading angle points towards the obstacle. The possible reason is the policy network gets stuck at the local optimum since the two trajectories next to it behave well. It can be difficult for the agent to learn if the origin is behind the obstacle and close to it, the same phenomena also exist for the other three obstacles.

2. Rectangular Obstacle Avoidance

The difference between this rectangular obstacle case and the previous circular obstacle case in simulation is the condition when checking whether the agent is at conflict state. The environment for this case is shown in Fig. 2(b) and

the testing result of 160 trajectories is shown in Fig. 5.

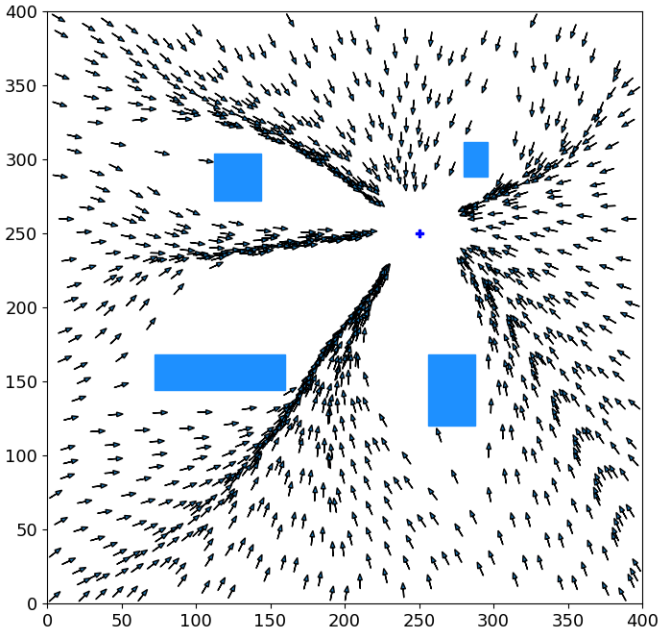


Fig. 5 Results of rectangular obstacles avoidance.

The performance in Fig. 5 is similar to the result in Fig. 4. Three failures occur near the left two and the lower-right obstacles. Compared to Fig. 4, the density of the arrows in this fixed airspace is different. In Fig. 5, the agent is likely to pass the four lines connecting each vertex to the goal. Another high-density line is between the left two obstacles. In Fig. 4, since the obstacle located at the top-left corner is large enough to block the line connecting the top-left vertex to the goal, the agent chooses to go down to bypass it. As for the upper right obstacle, the agent doesn't have a preference to avoid it shown in Fig. 4 because it is almost on the line between the goal and the top-right vertex. Results in Fig. 4 and Fig. 5 show that the proposed model is able to make the UAS learn to find the shortest path and also avoid obstacles for different obstacle sizes or shapes.

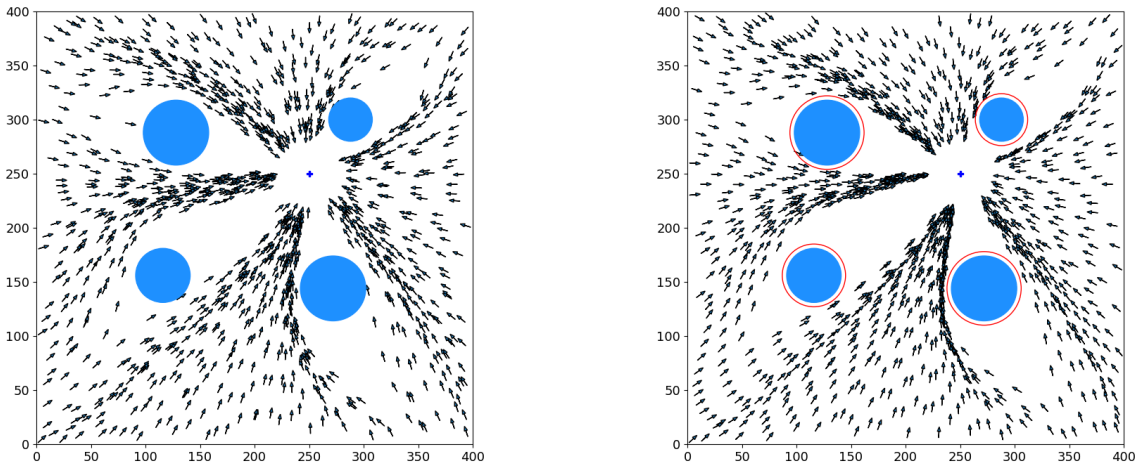
3. Circular Obstacle Avoidance with Uncertainty

This case is studied to see the performance of handling uncertainty by the proposed conflict resolution model. UAS operation is stochastic and randomness exists in almost every aspect of UTM. Inclusion of uncertainty quantification of aircraft operation is critical for future safety analysis (e.g., deviation from a trajectory plan due to wind, true speed, positioning error) [54–56]. Thus, to model the uncertainties in the real world, the true position of UAS is assumed to be within a circle of radius of 40 m and the predicted position as the center. Such uncertainty is considered when calculating the agent's position at the next time step after taking action a.

The testing results are shown in Fig. 6. In Fig. 6(a), the agent's position with uncertainty is plotted. While uncertainty is not with the agent's position in Fig. 6(b), and the uncertainty of 40 m is added to the obstacle, which is indicated by the red circle. As expected, the UAS tries to keep 40 m away from the obstacles. There are four failures in Fig. 6(a) and five failures in Fig. 6(b). So either method can work when doing the simulations of collision avoidance.

B. Moving obstacle avoidance

For the moving intruder aircraft avoidance case, the simulation environment is the airspace with 2km length and 2km width. The speeds of both the agent and intruder are set to 20 m/s. The goal is located at the middle point of the



(a) (b)

Fig. 6 (a) Results with probabilistic agent's position. (b) Results with deterministic agent's position.

top edge, (1000, 2000) m. When the agent is within 200 m away from the goal, it is treated as mission completion. There are two scenarios for moving obstacle avoidance: two-intruder and three-intruder cases. The minimum separation requirement is 250 m for each case. The reward coefficients are listed in Table 1. For each case, the deep reinforcement learning model is trained for 15 million timesteps. The learning curves are shown in Fig. 7. The blue lines are the plots of reward at each timestep and the orange lines indicate the smoothed average result at each time step. To visualize the performance of the proposed conflict resolution model, we generate a testing set of 500 episodes following the setting during the training process. The minimum distance of the agent to the intruders within each episode is collected.

Table 1 Reward Coefficient

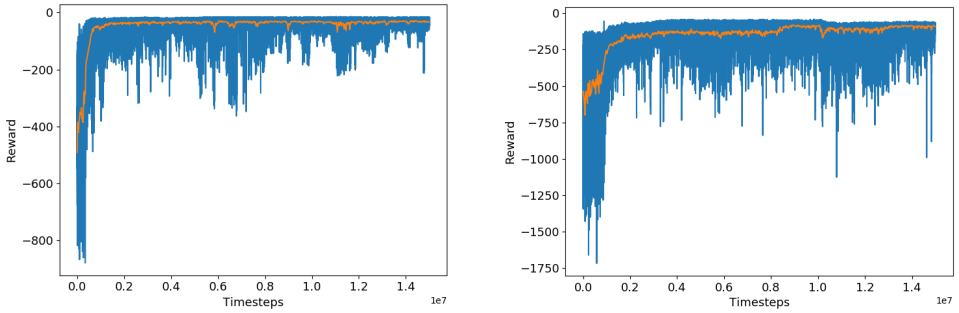
Coefficient	c_g	c_0	c_{11}	c_{12}	c_{13}	c_{2i}	c_{3i}	c_{4i}
Two-Intruder Avoidance	0.005	0.1	4.5	4.5	NA	20	-0.7	1
Three-Intruder Avoidance	0.007	0.15	10	6	8	20	-0.7	1

1. Two-Intruder Avoidance

The study in moving obstacle avoidance starts from the case with two intruders. The starting position of intruder 1 is on the left edge, in the range of (0, 500) m and (0, 1000) m. Intruder 2 starts from the left edge, in the range of (2000, 1500) m and (2000, 2000) m. The heading angles of intruder 1 and intruder2 are fixed to $\frac{\pi}{8}$ and $\frac{-7\pi}{8}$, respectively. The origin of the agent is either (650, 0) m or (1350, 0) m. The result of minimum distance is shown as points in Fig. 8. It can be seen that the points are all above the line which represents the minimum separation requirement. There is no failure case in the testing set.

2. Three-Intruder Avoidance

In the three-intruder case, intruder 1 starts from the top edge, in the range of (50, 2000) m and (350, 2000) m. Its heading angle is $\frac{-2\pi}{9}$. The origins and heading angles of intruder 2 and intruder 3 are set to be the same as that of intruder 1 and intruder 2, respectively in the two-intruder case. The origin of agent is either (800, 0) m or (1500, 0) m. The result of minimum distance is shown in Fig. 9. There is no failure case, indicating the model succeeds to avoid the



(a) (b)

Fig. 7 (a)The learning curve of the two-intruder case. (b) The learning curve of The three-intruder case.

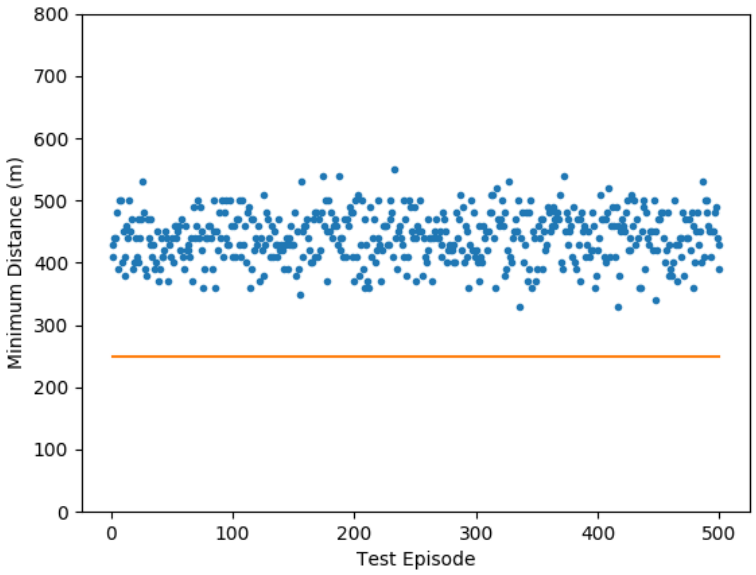


Fig. 8 Results of two-intruder avoidance.

three intruders.

V. Conclusion

In this work, we present a method for using deep reinforcement learning to allow the UAS to navigate successfully in urban airspace with continuous action space. Both static and moving obstacles are simulated and the trained UAS has the capability to achieve the goal and do conflict resolution simultaneously. We also investigate the performance on different static obstacle shapes and sizes, and under uncertainty in UAS operation. Intruders with a fixed heading angle are considered in the training process of the moving obstacle experiments. To make the proposed algorithm more practical in the real-world, future work would incorporate intruder aircraft with randomly sampled heading angles to the simulation environment.

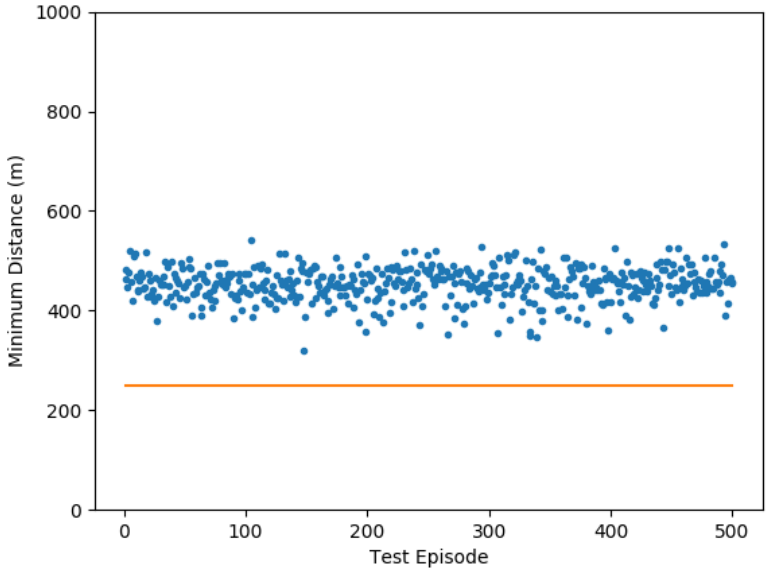


Fig. 9 Results of three-intruder avoidance.

Acknowledgments

The research reported in this paper was supported by funds from NASA University Leadership Initiative program (Contract No. NNX17AJ86A, PI: Yongming Liu, Technical Officer: Kai Goebel and Anupa Bajwa). The support is gratefully acknowledged.

References

[1] Balakrishnan, K., Polastre, J., Mooberry, J., Golding, R., and Sachs, P., "Blueprint for the sky," *The roadmap for the safe integration of autonomous aircraft. Airbus A*, Vol. 3, 2018.

[2] Kopardekar, P., Rios, J., Prevot, T., Johnson, M., Jung, J., and Robinson, J. E., "Unmanned aircraft system traffic management (UTM) concept of operations," 2016.

[3] Jenkins, D., Vasigh, B., Oster, C., and Larsen, T., *Forecast of the commercial UAS package delivery market*, Embry-Riddle Aeronautical University, 2017.

[4] Holden, J., and Goel, N., "Fast-Forwarding to a Future of On-Demand Urban Air Transportation," *San Francisco, CA*, 2016.

[5] Frazzoli, E., Mao, Z.-H., Oh, J.-H., and Feron, E., "Resolution of conflicts involving many aircraft via semidefinite programming," *Journal of Guidance, Control, and Dynamics*, Vol. 24, No. 1, 2001, pp. 79–86. https://doi.org/10.2514/2.4678.

[6] Raghunathan, A. U., Gopal, V., Subramanian, D., Biegler, L. T., and Samad, T., "Dynamic optimization strategies for three-dimensional conflict resolution of multiple aircraft," *Journal of guidance, control, and dynamics*, Vol. 27, No. 4, 2004, pp. 586–594. https://doi.org/10.2514/1.11168.

[7] Enright, P. J., and Conway, B. A., "Discrete approximations to optimal trajectories using direct transcription and nmiscar programming," *Journal of Guidance, Control, and Dynamics*, Vol. 15, No. 4, 1992, pp. 994–1002. https://doi.org/10.2514/3.20934.

[8] Schouwenaars, T., De Moor, B., Feron, E., and How, J., "Mixed integer programming for multi-vehicle path planning," *Control Conference (ECC), 2001 European*, IEEE, 2001, pp. 2603–2608. https://doi.org/10.23919/ECC.2001.7076321.

[9] Richards, A., and How, J. P., "Aircraft trajectory planning with collision avoidance using mixed integer linear programming," *American Control Conference, 2002. Proceedings of the 2002*, Vol. 3, IEEE, 2002, pp. 1936–1941. https://doi.org/10.1109/ACC.2002.1023918.

[10] Pallottino, L., Feron, E. M., and Bicchi, A., "Conflict resolution problems for air traffic management systems solved with mixed integer programming," *IEEE transactions on intelligent transportation systems*, Vol. 3, No. 1, 2002, pp. 3–11. https://doi.org/10.1109/6979.994791.

[11] Vela, A., Solak, S., Singhose, W., and Clarke, J.-P., "A mixed integer program for flight-level assignment and speed control for conflict resolution," *Decision and Control, 2009 held jointly with the 2009 28th Chinese Control Conference. CDC/CCC 2009. Proceedings of the 48th IEEE Conference on*, IEEE, 2009, pp. 5219–5226. https://doi.org/10.1109/CDC.2009.5400520.

[12] Mellinger, D., Kushleyev, A., and Kumar, V., "Mixed-integer quadratic program trajectory generation for heterogeneous quadrotor teams," *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, IEEE, 2012, pp. 477–483. https://doi.org/10.1109/ICRA.2012.6225009.

[13] Augugliaro, F., Schoellig, A. P., and D'Andrea, R., "Generation of collision-free trajectories for a quadrocopter fleet: A sequential convex programming approach," *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, IEEE, 2012, pp. 1917–1922. https://doi.org/10.1109/iros.2012.6385823.

[14] Morgan, D., Chung, S.-J., and Hadaegh, F. Y., "Model predictive control of swarms of spacecraft using sequential convex programming," *Journal of Guidance, Control, and Dynamics*, Vol. 37, No. 6, 2014, pp. 1725–1740. https://doi.org/10.2514/1.G000218.

[15] Acikmese, B., and Ploen, S. R., "Convex programming approach to powered descent guidance for mars landing," *Journal of Guidance, Control, and Dynamics*, Vol. 30, No. 5, 2007, pp. 1353–1366. https://doi.org/10.2514/1.27553.

[16] Delahaye, D., Peyronne, C., Mongeau, M., and Puechmorel, S., "Aircraft conflict resolution by genetic algorithm and B-spline approximation," *EIWAC 2010, 2nd ENRI International Workshop on ATM/CNS*, 2010, pp. 71–78.

[17] Cobano, J. A., Conde, R., Alejo, D., and Ollero, A., "Path planning based on genetic algorithms and the monte-carlo method to avoid aerial vehicle collisions under uncertainties," *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, IEEE, 2011, pp. 4429–4434. https://doi.org/10.1109/ICRA.2011.5980246.

[18] Pontani, M., and Conway, B. A., "Particle swarm optimization applied to space trajectories," *Journal of Guidance, Control, and Dynamics*, Vol. 33, No. 5, 2010, pp. 1429–1441. https://doi.org/10.2514/1.48475.

[19] Pallottino, L., Scordio, V. G., Frazzoli, E., and Bicchi, A., "Probabilistic verification of a decentralized policy for conflict resolution in multi-agent systems," *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, IEEE, 2006, pp. 2448–2453. https://doi.org/10.1109/ROBOT.2006.1642069.

[20] Moore, M. D., and Goodrich, K. H., "High speed mobility through on-demand aviation," *2013 Aviation Technology, Integration, and Operations Conference*, 2013, p. 4373. https://doi.org/10.2514/6.2013-4373.

[21] Gawdiak, Y., Holmes, B., Sawhill, B., Herriot, J., Ballard, D., Creedon, J., Eckhause, J., Long, D., Hemm, R., Murphy, C., et al., "Air transportation strategic trade space modeling and assessment through analysis of on-demand air mobility with electric aircraft," *12th AIAA Aviation Technology, Integration, and Operations (ATIO) Conference and 14th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, 2012, p. 5594.

[22] Mueller, E. R., Kopardekar, P. H., and Goodrich, K. H., "Enabling Airspace Integration for High-Density On-Demand Mobility Operations," *17th AIAA Aviation Technology, Integration, and Operations Conference*, 2017, p. 3086. https://doi.org/10.2514/6.2017-3086.

[23] Shim, D. H., and Sastry, S., "An evasive maneuvering algorithm for UAVs in see-and-avoid situations," *American Control Conference, 2007. ACC'07*, IEEE, 2007, pp. 3886–3891. https://doi.org/10.1109/ACC.2007.4283147.

[24] Shim, D. H., Kim, H. J., and Sastry, S., "Decentralized nmiscar model predictive control of multiple flying robots," *Decision and control, 2003. Proceedings. 42nd IEEE conference on*, Vol. 4, IEEE, 2003, pp. 3621–3626. https://doi.org/10.1109/CDC.2003.1271710.

[25] Khatib, O., and Mampey, L., "Fonction decision-commande d'un robot manipulateur," *Rep*, Vol. 2, No. 7, 1978, p. 156.

[26] Rimon, E., and Koditschek, D. E., "Exact robot navigation using cost functions: the case of distinct spherical boundaries in E/sup n," *Robotics and Automation, 1988. Proceedings., 1988 IEEE International Conference on*, IEEE, 1988, pp. 1791–1796.

[27] Connolly, C. I., Burns, J. B., and Weiss, R., "Path planning using Laplace's equation," *Robotics and Automation, 1990. Proceedings., 1990 IEEE International Conference on*, IEEE, 1990, pp. 2102–2106. https://doi.org/10.1109/ROBOT.1990.126315.

[28] Han, S.-C., Bang, H., and Yoo, C.-S., "Proportional navigation-based collision avoidance for UAVs," *International Journal of Control, Automation and Systems*, Vol. 7, No. 4, 2009, pp. 553–565. https://doi.org/10.1007/s12555-009-0407-1.

[29] Park, J.-W., Oh, H.-D., and Tahk, M.-J., "UAV collision avoidance based on geometric approach," *SICE Annual Conference, 2008*, IEEE, 2008, pp. 2122–2126. https://doi.org/10.1109/SICE.2008.4655013.

[30] Krozel, J., Peters, M., and Bilimoria, K., "A decentralized control strategy for distributed air/ground traffic separation," *AIAA Guidance, Navigation, and Control Conference and Exhibit*, 2000, p. 4062. https://doi.org/10.2514/6.2000-4062.

[31] Van Den Berg, J., Guy, S. J., Lin, M., and Manocha, D., "Reciprocal n-body collision avoidance," *Robotics research*, Springer, 2011, pp. 3–19. https://doi.org/10.1007/978-3-642-19457-3_1.

[32] Yang, X., and Wei, P., "Autonomous On-Demand Free Flight Operations in Urban Air Mobility using Monte Carlo Tree Search," *8th International Conference on Research in Air Transportation (ICRAT)*, ICRAT, 2018.

[33] Yang, X., and Wei, P., "Scalable Multi-Agent Computational Guidance with Separation Assurance for Autonomous Urban Air Mobility," *Journal of Guidance, Control, and Dynamics*, 2020. https://doi.org/10.2514/1.G005000.

[34] Ong, H. Y., and Kochenderfer, M. J., "Markov Decision Process-Based Distributed Conflict Resolution for Drone Air Traffic Management," *Journal of Guidance, Control, and Dynamics*, 2016, pp. 69–80. https://doi.org/10.2514/1.G001822.

[35] Chen, Y. F., Liu, M., Everett, M., and How, J. P., "Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning," *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, IEEE, 2017, pp. 285–292. https://doi.org/10.1109/ICRA.2017.7989037.

[36] Bellman, R., "A Markovian Decision Process," *Indiana Univ. Math. J.*, Vol. 6, 1957, pp. 679–684.

[37] Howard, R. A., "Dynamic programming and Markov processes," 1964. https://doi.org/10.1126/science.132.3428.667.

[38] White, D. J., "A survey of applications of Markov decision processes," *Journal of the operational research society*, Vol. 44, No. 11, 1993, pp. 1073–1096. https://doi.org/10.1057/jors.1993.181.

[39] Feinberg, E. A., and Shwartz, A., *Handbook of Markov decision processes: methods and applications*, Vol. 40, Springer Science & Business Media, 2012. https://doi.org/10.1007/978-1-4615-0805-2.

[40] Koenig, S., and Simmons, R., "Xavier: A robot navigation architecture based on partially observable markov decision process models," *Artificial Intelligence Based Mobile Robotics: Case Studies of Successful Robot Systems*, 1998, pp. 91–122.

[41] Thrun, S., "Probabilistic robotics," *Communications of the ACM*, Vol. 45, No. 3, 2002, pp. 52–57. https://doi.org/10.1017/S0269888906210993.

[42] Mariton, M., *Jump linear systems in automatic control*, M. Dekker New York, 1990.

[43] Sutton, R. S., and Barto, A. G., *Reinforcement learning: An introduction*, MIT press, 2018.

[44] Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al., "Mastering the game of go without human knowledge," *Nature*, Vol. 550, No. 7676, 2017, p. 354.

[45] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al., "Human-level control through deep reinforcement learning," *Nature*, Vol. 518, No. 7540, 2015, p. 529.

[46] Hessel, M., Modayil, J., Van Hasselt, H., Schaul, T., Ostrovski, G., Dabney, W., Horgan, D., Piot, B., Azar, M., and Silver, D., "Rainbow: Combining improvements in deep reinforcement learning," *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[47] Vinyals, O., Ewalds, T., Bartunov, S., Georgiev, P., Vezhnevets, A. S., Yeo, M., Makhzani, A., Küttler, H., Agapiou, J., Schrittwieser, J., et al., "Starcraft ii: A new challenge for reinforcement learning," *arXiv preprint arXiv:1708.04782*, 2017.

[48] Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D., "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[49] Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., and Kavukcuoglu, K., "Asynchronous methods for deep reinforcement learning," *International conference on machine learning*, 2016, pp. 1928–1937.

[50] Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P., "Trust region policy optimization," *International conference on machine learning*, 2015, pp. 1889–1897.

[51] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O., "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[52] Hu, J., Erzberger, H., Goebel, K., and Liu, Y., "Risk-Based Dynamic Anisotropic Operational Safety Bound for Rotary UAV Traffic Control," *Proceedings of the Annual Conference of the PHM Society*, Vol. 11, 2019.

[53] Dhariwal, P., Hesse, C., Klimov, O., Nichol, A., Plappert, M., Radford, A., Schulman, J., Sidor, S., Wu, Y., and Zhokhov, P., "OpenAI Baselines," https://github.com/openai/baselines, 2017.

[54] Liu, Y., and Goebel, K., "Information Fusion for National Airspace System Prognostics," *PHM Society Conference*, Vol. 10, 2018.

[55] Pang, Y., Yao, H., Hu, J., and Liu, Y., "A Recurrent Neural Network Approach for Aircraft Trajectory Prediction with Weather Features From Sherlock," *AIAA Aviation 2019 Forum*, 2019, p. 3413.

[56] Pang, Y., Xu, N., and Liu, Y., "Aircraft Trajectory Prediction using LSTM Neural Network with Embedded Convolutional Layer," *Proceedings of the Annual Conference of the PHM Society*, Vol. 11, 2019.