Autonomous Free Flight Operations in Urban Air Mobility with Computational Guidance and Collision Avoidance

Xuxi Yang, Peng Wei, *Member, IEEE*

***Abstract*—The use of electrical vertical takeoff and landing (eVTOL) aircraft to provide efficient, high-speed, on-demand air transportation within a metropolitan area is a topic of increasing interest, which is expected to bring fundamental changes to the city infrastructures and daily commutes. NASA, Uber, and Airbus have been exploring this exciting concept of Urban Air Mobility (UAM), which has the potential to provide meaningful door-to-door trip time savings compared with automobiles. However, the ability to manage many of these eVTOL aircraft safely in a congested urban area presents a challenge unprecedented in air traffic management. In order to enable safe and efficient autonomous on-demand free flight operations in UAM, a computational guidance algorithm with collision avoidance capability is designed and analyzed. The approach proposed in this paper is to formulate this problem as a Markov Decision Process (MDP) and solve it using an online algorithm Monte Carlo Tree Search (MCTS). For the sake of illustration, a high-density free flight airspace simulator is created to test the performance of this algorithm. Numerical experiment results show that this proposed algorithm has fewer conflicts and near mid-air collisions when compared with Optimal Reciprocal Collision Avoidance (ORCA), a state-of-the-art collision avoidance strategy.**

I. Introduction

A. Motivation

Over the past several years, there has been an increasing interest in Urban Air Mobility (UAM) operations, including NASA, Uber, and Airbus [1]–[3]. Companies such as Airbus, Bell, Embraer, Joby, Zee Aero, Pipistrel, Volocopter, and Aurora Flight Sciences have been working to build and test electric vertical takeoff and landing (eVTOL) aircraft. The UAM operations are expected to fundamentally change cities and people's lives to reduce commute time and stress. In UAM, orders of magnitude more aircraft than those operating today would be required for this transportation mode to serve a significant proportion of the public [4]. In this paper, we focus on overcoming one specific technical barrier: creating a safe onboard guidance and collision avoidance system for eVTOLs to operate in very-high-density air traffic conditions, through combining the power of onboard aircraft intelligence (vehicle technology) and the advantage of the free flight idea (airspace operation concept).

The concept of "Free Flight" was proposed previously for future air transportation applications because it has the potential to cope with the ongoing congestion of the current ATC system. It was shown in previous work [5], [6] that free flight with airborne separation is able to handle a higher traffic density by utilizing more airspace volume. Besides, free flight can also bring fuel and time efficiency [7]. In a free flight framework, it is implied that aircraft will be responsible for their own separation assurance and conflict resolution [7]. The loss of an airway structure may make the process of detecting and avoiding conflicts between aircraft more complex. Previous studies [8] show that detecting and avoiding conflicts in free flight is potentially feasible because of enabling technologies such as Global Positioning Systems (GPS), data link communications like Automatic Dependence Surveillance-Broadcast (ADSB) [9], Traffic Alert and Collision Avoidance Systems (TCAS) [10], and powerful onboard computation capability. In addition, automated conflict detection and resolution tools [11] will be required to aid pilots and ground controllers in ensuring traffic separation and conflict resolution.

In this paper, under the free flight framework, a computational guidance algorithm with collision avoidance capability is proposed to guide the aircraft to its destination, where the aircraft dynamics is modeled based on the tandem tilt-wing eVTOL (Airbus Vahana) from Airbus A^3 [12], as shown in Fig. 1. The proposed algorithm in this paper uses Markov Decision Process (MDP) and Monte Carlo Tree Search (MCTS), where the input of this algorithm is the current position and velocity of other surrounding aircraft, and the position of the trip destination for the aircraft we are controlling. Through obtained sensing information of other aircraft, the controlled aircraft will perform online sequential decision making to select actions in real-time with onboard avionics computation. The series of actions will guide the aircraft to reach its goal and avoid potential conflicts quickly. The proposed algorithm provides a potential solution framework to enable autonomous on-demand free flight operations in urban air mobility.

B. Related Work

Decades of research have explored a variety of approaches for designing collision avoidance systems for aircraft, which can be categorized based on the following criteria:

1) Centralized/Decentralized [13]: whether the problem is solved by a central supervising controller (centralized) or by each aircraft individually (decentralized).
2) Planning/Reacting [14]: the planning approach generates feasible or even optimal paths ahead of time; whereas

X. Yang was with the Department of Aerospace Engineering, Iowa State University, Ames, IA, 50011 USA, e-mail: xuxiyang@iastate.edu.

P. Wei was with the Department of Aerospace Engineering, Iowa State University, Ames, IA, 50011 USA, e-mail: pwei@iastate.edu.

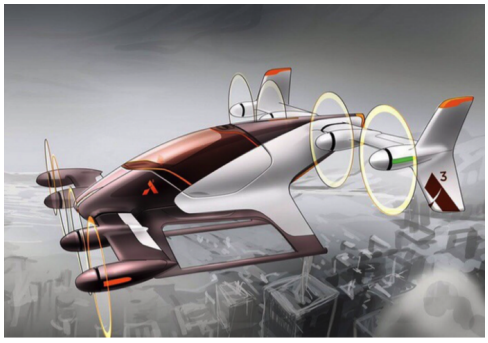


Fig. 1: Airbus Vahana with tandem tilt-wing configuration during the cruise phase [12].

the reacting approach typically uses an online collision avoidance system to respond to dangerous situations.

3) Cooperative/Non-cooperative [13]: whether there exists online communication between aircraft or between aircraft and the central controller.

In the following, we will briefly discuss the related work categorized based on the first criterion: centralized method and decentralized method. The second and third criteria will be also discussed in each category.

In centralized methods, the conflicts between aircraft are resolved by a central supervising 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 central controller (thus centralized methods are always cooperative), and the central controller in return designs the whole individual trajectory for all aircraft before the flight, typically by formulating it to an optimal control problem. These methods can be based on semidefinite programming [15], nonlinear programming [16], [17], mixed-integer linear programming [18]–[21], mixed-integer quadratic programming [22], sequential convex programming [23], [24], second-order cone programming [25], evolutionary techniques [26], [27], and particle swarm optimization [28]. Besides formulating this problem using optimal control framework, roadmap methods such as visibility graph [29] and Voronoi diagrams [30] can also handle the path planning problem for aircraft. However, calculating the exact solutions will become impractical when the state space becomes large or high-dimensional. To address this issue, sample-based planning algorithms are proposed, such as probabilistic roadmaps [31], RRT [32], and RRT* [33]. These centralized methods often pursue the global optimality of the solution. 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 [34]. In decentralized methods, all the conflicts are resolved by each aircraft individually. Decentralized methods can be cooperative and non-cooperative. Researchers have proposed several algorithms under the case where the communication between aircraft can be successfully established (cooperative) [35]. Algorithms in [36], [37] are based on message-passing schemes, which resolve local (e.g., pairwise) conflicts without needing to form a joint optimization problem between all members of the team. In [13], every agent is allotted a time slot to compute a dynamically feasible and collision-free path using mixed-integer linear programming. In [38], the author recast the global optimization problem as several local problems, which are then iteratively solved by the agents in a decentralized way. In the Decentralized Model Predictive Control approach [39], the aircraft solve their own sub-problem one after the other and send the action to other subsystems through communication.

The work in this paper focuses on scenarios where communication cannot be reliably established (non-cooperative) and the aircraft will take action at each time step based on the sensor information. Many works fall in this category: Model Predictive Control [40], [41] can be used to solve the collision avoidance problem, but the computation load is relatively high. Potential field method [42] is computationally fast. However, a navigation function is required to make it a complete path planner [43], [44], which involves discretizing the configuration space. With the help of machine learning and reinforcement learning [45]–[49], collision avoidance algorithm can have a promising performance, but usually needs much time to train. Using the Monte Carlo Tree Search algorithm to solve this problem [50] does not need model training, and the algorithm can finish in any predefined computation time. However, the aircraft can only adopt several discretized actions at each time step. Geometry based algorithms [51]–[54] can also be applied for collision avoidance problem 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 (it only pays attention to the current action and does not consider the effect of subsequent actions). The outcome can be local optimal in the view of the global trajectory.

In this paper, we formulate this guidance and collision avoidance system as a MDP and solve this MDP using the online algorithm MCTS. There are similar works using MDP formulation which solve this problem offline [55]–[57]. Offline solvers require large computation time upfront to compute the optimal policy for the full state space and discrete MDP formulations. Offline methods are typically not adaptive to changes in the environment because the policy is determined ahead of time. Also, the state space of many problems is too large to adequately represent as a finite set of enumerable states. Comparing with offline methods, online methods address the shortcomings of offline methods by only planning for the current state and a small number of possible plans with longer online computation time. Since online algorithms do not plan for the whole state space, discrete MDP formulations are not required. Online algorithms are also able to account for changes in the environment because they are executed once at each decision point, allowing for updates between these points. There is also Partially Observable MDP (POMDP) formulation for this problem, which aims to account for the state uncertainty due to the sensor noise [58], where

the authors use an online search algorithm to solve this POMDP. The major difference with this paper is that they use depth-first, branch-and-bound search and do not consider the state transition uncertainty, while the MCTS algorithm is robust to the dynamical model uncertainty through balancing exploration and exploitation.

C. Contributions

In this paper, we propose an algorithm which is an extension of previous work [50], where the authors formulate the computational guidance problem with collision avoidance function as a Markov Decision Process (MDP) and solve this MDP using the Monte Carlo Tree Search (MCTS) Algorithm. The contributions of this paper are:

1) This paper proposed a computational guidance algorithm that performs consistently better than the baseline algorithm ORCA [54] under high-density air traffic scenarios with uncertainty, achieving the new state-of-the-art.
2) The algorithm used in this paper is a variation of the MCTS-UCT algorithm. We truncate the tree search process and use a heuristic value function to denote the value of the intermediate node during the search process, which improves the computation efficiency of the online algorithm.
3) Comparing with previous works that focus on one vs. one collision avoidance (one ownship vs. one intruder aircraft), in this paper, we show the proposed algorithm provides a framework to handle an arbitrary number of intruders, and present the numerical simulation results up to 80 intruders. This provides the necessary step to make the computational guidance algorithm more practical.
4) Besides, the proposed algorithm is the foundation and building block for the more general case of multiple cooperative aircraft.

The structure of the paper is as follows: in Section II, the background of MDP and MCTS will be introduced. In Section III, the description of the problem and its mathematical formulation of MDP are presented. Section IV presents the designed MCTS algorithm to solve this problem and a brief description of applying the ORCA method to solve this collision avoidance problem. The numerical experiment and results are shown in Section V. Section VI is the conclusion.

II. Background

In this section, we briefly review the background of the Markov Decision Process and Monte Carlo Tree Search.

A. Markov Decision Process (MDP)

Since the 1950s, MDPs [59] have been well studied and applied to a wide area of disciplines [60]–[62], including robotics [63], [64], automatic control [65], economics, and manufacturing. In a 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 Markov Decision Process (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.

In a MDP problem, a policy π is a mapping from the state to one specific action (known as deterministic policy)

$$\pi : \mathcal{S} \to \mathcal{A}$$

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

$$\pi^* = \underset{\pi}{\operatorname{argmax}} \mathbb{E}[\sum_{t=0}^{T} R(s_t, a_t)|\pi] \tag{1}$$

B. Monte Carlo Tree Search (MCTS)

Monte Carlo Tree Search (MCTS) is a method for finding optimal decisions in a given domain by taking random samples in the decision space and building a search tree according to the results [66], [67]. It has already had a profound impact on Artificial Intelligence (AI) approaches for domains that can be represented as trees of sequential decisions, particularly games and planning problems [68]–[70], including the current state-of-art computer program AlphaZero in the Game of Go [71].

The basic MCTS process is building a search tree incrementally and asymmetrically. For each iteration of the algorithm, a tree policy is used to find the most urgent node of the current tree. The tree policy attempts to balance considerations of exploration (look in areas that have not been well sampled yet) and exploitation (look in areas which appear to be promising). A simulation is then rolled out from the selected node and the search tree is updated according to the result. This involves the addition of a child node corresponding to the action taken from the selected node and an update of the statistics of its ancestors. Moves are made during this simulation according to some default policy, which in the simplest case is to make uniformly random moves. A great benefit of MCTS is that the values of intermediate states do not have to be evaluated, as for depth-limited minimax search, which significantly reduces the amount of domain knowledge required. Only the value of the terminal state at the end of each simulation is required.

III. Problem Formulation

A. Problem Statement

This paper aims to control an aircraft through a series of actions so that the aircraft can arrive at its destination while avoiding potential conflicts with other intruder aircraft during the flight. This is a sequential decision-making problem that can be formulated as a MDP problem. In this MDP problem, the action is decided directly from the state, which incorporates all the information (the position and velocity of intruder aircraft, the position of the destination) for the agent to decide which action is optimal for the corresponding state.

In this paper, a high-density free flight airspace scenario is considered: one aircraft (the ownship) is equipped with the MCTS algorithm and will try to avoid the conflicts with other intruder aircraft. Here the aircraft performance data is based on Airbus Vahana aircraft [72].

When controlling the aircraft, only horizontal actions are considered in this paper, which means all the aircraft will be flying at the same altitude and this problem can be solved in two dimensions. This assumption makes it possible to incorporate multiple flight levels to deal with the high-density air traffic in UAM.

Besides, we also assume the aircraft can get the intruder aircraft information (position and velocity) through the sensor perfectly. Future work would include test the performance of this algorithm under different levels of measurement uncertainties [73], [74].

The objectives for this specific MDP problem are two-fold: the first is to guide the aircraft to the goal state in a short time, and the second is to avoid any conflicts between the controlled aircraft and other intruder aircraft. Therefore, the reward function should be able to capture both two objectives.

Based on the above description, this problem will be mathematically formulated as a MDP problem in the next subsection.

B. MDP Formulation

1) State Space:: A state includes all the information the ownship needs for its decision making: the position and velocity of all the aircraft including ownship and intruders, together with the goal position. For the intruder aircraft k, we use $(i_x^{(k)}, i_y^{(k)})$, $(i_{vx}^{(k)}, i_{vy}^{(k)})$ to denote its position and velocity. For the ownship, its position (o_x, o_y), velocity (o_{vx}, o_{vy}), speed o_v, heading angle o_ψ, and the bank angle o_ϕ are included in the state. To sum up, if there are n intruders, 1 ownship, and 1 goal, the current state will be a vector of length $4 \times n + 7 \times 1 + 2$. Note that the state variables defined here are continuous. In general, for a MDP with continuous state variables, it is not clear how to best represent the policy, since it is impossible to enumerate all possible state-action mappings. For previous MDP-based algorithms to solve conflict avoidance problems, some possible approaches to represent the policy include using a grid-based discretization of the state space $\mathcal{S}$ and the action space $\mathcal{A}$ [56], [75] or using some policy compression techniques [76]. Comparing with previous methods, the advantage of the MCTS algorithm is that we do not need to discretize the state space. For each state, the MCTS algorithm will generate an action for the aircraft to follow in real-time.

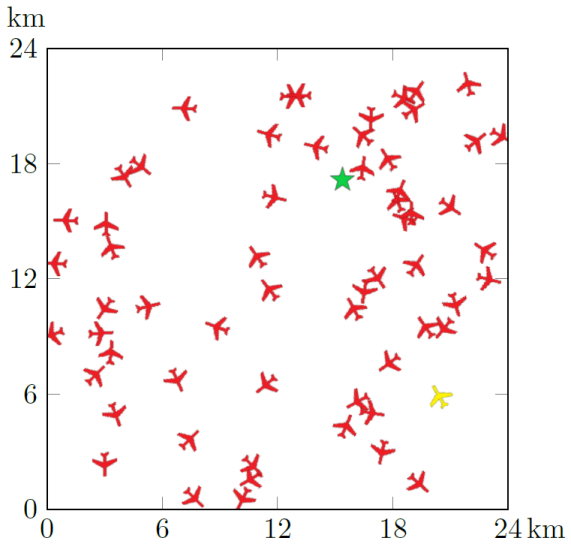


Fig. 2: An example state of the MDP formulation.

Fig. 2 shows an example state of this MDP. This state includes all the aircraft information in a $24km \times 24km$ map. It should be noted that the displayed aircraft size in this figure is not proportional from its size in real world, which is approximately 6m by 6m [72]. In this figure, the yellow aircraft is the ownship, the red aircraft are the intruders, and the green star is the goal position for the ownship.

2) Action Space: At the beginning of each time step (5 seconds), the ownship can choose to change both its bank angle and acceleration at certain rates.

For the bank angle, the ownship can choose to turn right, turn left, or go straight. More precisely, the advisory for the change of bank angle constitutes the action set $\mathcal{A}_\phi = \{-5°/s, 0°/s, +5°/s\}$ where negative corresponds to right turn and positive to the left turn. For the passenger comfort, we restrict the bank angle to lie between $-25°$ and $25°$. When the heading angle action leads the bank angle to go beyond $\pm 25°$, the bank angle will be clipped to $\pm 25°$.

For the acceleration, the ownship can choose one acceleration from the action set $\mathcal{A}_a = \{-5m/s^2, 0m/s^2, 5m/s^2\}$. Similar to the bank angle, we restrict the speed to be in between $50m/s$ and $80m/s$ [77] following the aircraft performance data of Airbus Vahana [12], [72], [78].

At each time step, the ownship will choose one action $(a_\phi, a_a) \in \mathcal{A}_\phi \times \mathcal{A}_a$ and the ownship will maintain the action during this time step.

It is natural to consider extending the set of actions (conflict resolution advisories) to include more options than 9. However, using the MCTS algorithm to calculate the optimal action will be more time consuming with the extended action space since the tree size will grow exponentially with the number of actions. Because computation time is an important factor for the online algorithm, some techniques can be used for extending action space in future steps, such as truncated Monte Carlo search algorithms [79] or using a policy network to

narrow down the search to high-value actions [68]. In this paper, we use 3 by 3 action space to keep our scope more focused.

3) Dynamical Model: Based on the current state and current action, the following kinematic model will be used to compute state transition for ownship:

$$\begin{aligned} \dot{o}_v &= a_a + \epsilon_{\dot{v}} \\ \dot{o}_\phi &= a_\phi + \epsilon_{\dot{\phi}} \\ \dot{o}_\psi &= \frac{g \tan o_\phi}{o_v} \\ \dot{o}_x &= o_v \cos o_\psi \\ \dot{o}_y &= o_v \sin o_\psi \end{aligned} \tag{2}$$

where a_a is the acceleration and a_ϕ is the changing rate of bank angle.

After the aircraft execute an advisory, a normally distributed noise $\epsilon_{\dot{\phi}}$ with a standard deviation of 4° will be added to the bank angle, and a normally distributed noise $\epsilon_{\dot{v}}$ with a standard deviation of $2m/s$ will be added to the speed. Similarly for the intruder aircraft, a normally distributed noise with a standard deviation of $10m$ will be added to its position at each time step. The noises here aim to account for the uncertainties in the environment and aircraft dynamics.

4) Terminal State: For the consideration of safety, the conflict is defined to be when the distance of two aircraft is less than a minimum separation distance $r^{min} = 0.3$ nautical miles [80]. This separation standard was chosen using the definition of well clear for Unmanned Aircraft Systems (UAS) according to Cook and Brooks [81]. For large UAS in high-altitude airspace, the Horizontal Miss Distance (HMD) is defined to be 0.66nmi. For small UAS (55lbs vehicle or less) in low-altitude controlled airspace around airports, the horizontal separation is set to be a HMD of 0.36nmi. Using those values as a reference, the nominal spatial separation standards picked for this UAM application are set to 0.3nmi horizontally. These are tighter than UAS standards because it is assumed that enhanced equipage capabilities will be installed onboard UAM aircraft [80].

Based on the above separation requirements, the terminal state of this MDP includes the following three different types of states, which can be determined directly through the state information.

1) The distance from ownship to any intruder is less than r^{min} (referred to as a conflict state in the following);
2) The ownship flies out of the map (referred to as a boundary state in the following);
3) The ownship reaches the goal position (referred to as a goal state in the following).

5) Reward Function: The goal in this paper is to make an aircraft quickly reach its destination and avoid potential conflict. These two objectives can be captured in the reward function defined as follows:

$$R(s) = \begin{cases} 1, & \text{if } s \text{ is goal state,} \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

With this reward setting, reaching a conflict state or a boundary state before the goal state will terminate the whole process with a reward of 0. Reaching a goal state will terminate this process with a reward of 1. So when maximizing the reward, the agent will try to reach the goal state and avoid conflict states and boundary states. Therefore we do not need to introduce a penalty for boundary states or conflict states.

IV. Solution Method

In this section, we will introduce our proposed solution approach and the baseline method. We will describe how to apply these methods to solve the formulated problem.

A. MCTS Algorithm

For the MDP formulated above, the most popular algorithm in the MCTS family, the Upper Confidence Bound for Trees (UCT) [82], is used to solve this problem. UCT has some promising properties: it is very efficient and guaranteed to be within a constant factor of the best possible bound on the growth of regret (the regret is the expected loss due to not selecting the best action), and it can balance exploration and exploitation very well [82].

In the MCTS algorithm, the nodes in the search tree denote the states in the state space of the MDP problem formulated in Section III. In the remaining part of this paper, the state and the node will be used interchangeably. The child nodes of a node are all the possible next states (nodes) resulting from different actions from the current state (node). Since there are nine actions in the action space at each time step, each node will have at most nine child nodes by executing these nine different actions.

MCTS algorithm selects actions by lookahead search. Each edge (s, a) of the search tree stores an action value $Q(s, a)$ and its visit count $N(s, a)$. The tree is traversed by simulation, starting from the root state, which is the current state we are considering.

In **selection** step, the ownship will select a child node with maximum value in Equation (4), so as to maximize the mean action value $\overline{X}_j$ plus a uncertainty bonus:

$$UCT = \overline{X}_j + 2C\sqrt{\frac{2\ln n}{n_j}} \tag{4}$$

Here the first term $\overline{X}_j$ is referred as exploitation term, which is directly from the formula

$$\overline{X}_j = Q_j/n_j \tag{5}$$

where the number n_j is the times the child node j has been visited before and the value Q_j is the total reward of all playouts that passed through this child node (so that Q_j/n_j is an approximation of the child node's state-action value). The second term $2C\sqrt{2\ln n/n_j}$ is referred as an exploration term where n is the number of times the current (parent) node has been visited, and C is a constant to balance the exploration and exploitation. A higher C value will emphasize exploration and a lower C value will encourage exploitation. It should be noted that the value of C depends on the value scale of $\overline{X}_j$. Since the value of $C = 1/\sqrt{2}$ was shown by Kocsis and Szepesvari to satisfy the Hoeffding inequality with rewards in the range

$[0, 1]$ [83], it is reasonable to set $C = 1/\sqrt{2}$ in this paper. With a reward range different than $[0, 1]$, a different value of C may be needed.

If more than one child node has the same maximal value, the tie is broken randomly [83]. It is generally understood that $n_j = 0$ yields a UCT value of ∞, so that if a node is never visited previously, it will be assigned to the largest possible value, to ensure that every child will be considered at least once before any expansion [67]. This is the strategy used in this paper.

The second step for the UCT algorithm is **expansion**, which happens when the ownship is at a new node which it has never visited before. This step is adding this new node to the current tree under its parent node (the previous state), and setting its visiting number to 1 and cumulative reward to 0.

The third step is **roll out**, which aims to estimate the value for the newly added state in the expansion state. After a new node is added to the tree, its value will be determined by running a simulation to a terminal state following a random policy, until reaching a terminal state with a final reward. It should be noted that simulating to a terminal state usually requires many steps, which is time-consuming, and we will address this limitation of the MCTS algorithm in the next subsection "Estimated Value Function".

After an action is selected, the next state for the ownship and intruder aircraft will be updated based on the dynamical model in Equation (2).

The final step of MCTS is **backpropagation**. After simulating the whole process to a terminal state, the final reward and visit count of all traversed edges are updated. Each traversed edge accumulates the reward and increases the visit count by 1, and we can get the mean state-action value from the total reward and visit count.

One iteration of the above four steps is called one simulation. If the computation budget allows (e.g., the decision needs to be made in 100ms), sufficient simulations will be repeated, which can provide a good approximation for the values of different nodes. When the simulation stops and a decision needs to be made, the most promising node will be selected by performing exploitation (set $C = 0$ in Equation (4)).

Estimated Value Function: One concern for the proposed algorithm is that the ownship may not have sufficient time to run this algorithm, since the algorithm presented in this paper is an online algorithm, which means it is vital for the ownship to compute quickly to make decisions. Since simulating this process to a terminal state usually needs many steps, simulating this process to a fixed search depth d is beneficial to reduce computation time in the roll out step. More specifically, if the algorithm simulates to a fixed search depth d and reaches a non-terminal state, the agent will use the estimated value function as the terminal reward and backpropagate this reward information. An example of the building of the state-action decision tree is given in Fig. 3, where the search depth is fixed at 2. For the estimated value function, intuitively, if at a state where the ownship is close to the goal state, this state should be a better state without any other information, so the following estimated value function

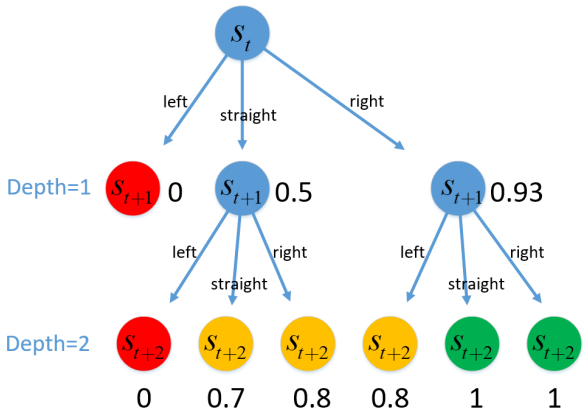


Fig. 3: Illustration of the state-action tree built in the MCTS algorithm with search depth 2. For illustration purposes, we only consider three actions in this case: turn left, turn right, and go straight. Here red node denotes the conflict state; the green node denotes the goal state. Yellow nodes mean that the agent simulates to depth two and uses the estimated value function as the final reward for the non-terminal state, depending on the distance between ownship and goal position. The state-action value of each node at time step $t + 1$ is the average of all its child node values. Based on this illustration, the agent will select to turn right at the current state s_t.

is used for the non-terminal states, so that the ownship can judge the goodness of any non-terminal state:

$$\tilde{V}(s) = 1 - \frac{d(o, g)}{\max d(o, g)}, \text{ if } s \text{ is non-terminal state} \quad (6)$$

where $d(o, g)$ denotes the distance from ownship to goal position. $\max d(o, g)$ is the maximum distance from ownship to the goal state, which is the diagonal distance of the map (if the map has an irregular convex shape, we can use the diameter of this convex shape). In this way, if there is no conflict with intruder aircraft or the border (which has reward 0), the ownship will get a positive reward between 0 and 1, depending on how far the ownship is from the goal state.

The above procedure is summarized in Algorithm 1. In this pseudo code, we use v to denote the node and s to denote the state information of node v. $s(v)$ means the state of a node and $v(s)$ means the node created from state s. $Q(v)$ is the total reward of all playouts that passed through the node v and $N(v)$ is the times the node v has been visited before. $d(v)$ represents the search depth of the node v.

B. Optimal Reciprocal Collision Avoidance (ORCA) Method

A popular approach to this computational guidance problem is the Optimal Reciprocal Collision Avoidance (ORCA) Method [54]. The ORCA method is an efficient computational algorithm and its safety has been demonstrated in realistic applications [84], [85]. Similar to our algorithm, the ORCA method is also a reaction-based method that specifies one-step action for the current geometric configuration. In this part, we present how to use the ORCA method to solve this problem compare its performance with the MCTS algorithm.

Algorithm 1 MCTS-UCT algorithm

function UCTSEARCH(s_0)
  create root node v_0 with state s_0
  while within computational budget **do**
    $v_l \leftarrow$ TREEPOLICY(v_0)
    $reward \leftarrow$ DEFAULTPOLICY($s(v_l)$)
    BACKUP($v_l, reward$)
  return a(BESTCHILD($v_0, 0$))

function TREEPOLICY(v)
  while v is nonterminal **and** $d(v) \leq d$ **do**
    if v not fully expanded **then**
      return EXPAND(v)
    else
      $v \leftarrow$ BESTCHILD(v, C)
  return v

function EXPAND(v)
  choose $a \in$ untried actions from $A(s(v))$
  $s(v') =$ PROCEED($s(v), a$)
  add the new child v' to v
  return v'

function BESTCHILD(v, c)
  return $\underset{v' \in \text{children of } v}{\operatorname{argmax}} \frac{Q(v')}{N(v')} + c\sqrt{\frac{2\ln N(v)}{N(v')}}$

function DEFAULTPOLICY(s)
  while s is nonterminal **and** $d(v(s)) \leq d$ **do**
    choose $a \in A(s)$ uniformly at random
    $s \leftarrow$ PROCEED(s, a)
  return reward for state s

function BACKUP($v, reward$)
  while v is not null **do**
    $N(v) \leftarrow N(v) + 1$
    $Q(v) \leftarrow Q(v) + reward$
    $v \leftarrow$ parent of v

function PROCEED(s, a)
  $s' \leftarrow$ next state from current s, a
  $d(v(s')) \leftarrow d(v(s)) + 1$
  return s'

The basic idea of ORCA is that, at each time step, the ownship first decides the conflict-free velocity set with all the other aircraft for at least a preset amount of time τ, which we denote as $ORCA^{\tau}$. Next, the ownship will select its new velocity as close as possible to its preferred velocity (which is the velocity point directly to its destination) in the set $ORCA^{\tau}$.

Since in our problem, we are only controlling one ownship aircraft. So in contrast to the original ORCA paper [54] where each agent will take half of the responsibility to remain on a collision-free trajectory, in this paper we let the ownship take the full responsibility when selecting the conflict-free velocity.

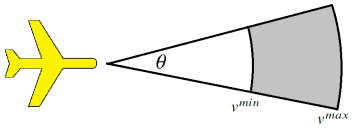


Fig. 4: Sample area of the aircraft for ORCA algorithm, which we denote as set S.

Besides, since we are controlling the eVTOL aircraft, the selected velocities also need to satisfy the dynamic constraint. Specifically, we restrict the ownship to select the velocity with speed in the range $(v - 5m/s, v + 5m/s)$ where v is the current speed of the ownship. One drawback of the ORCA algorithm is that it always selects the velocity directly and cannot incorporate the bank angle model in a straightforward way. So here we restrict the ownship to select velocity in the 10-degree field of view, which means the ownship can change its heading angle less than $10°/s$. This dynamics constraint is shown in Fig. 4, where the ownship will select a velocity in the shaded area, which we denote as set S. In this figure, θ is 20°, v^{min} and v^{max} are chosen to be $v - 5m/s, v + 5m/s$ where v is the current speed of the ownship. The the ORCA algorithm becomes the following optimization problem

$$v^{new} = \underset{v \in ORCA^{\tau} \bigcap S}{\operatorname{argmax}} ||v - v^{pref}|| \tag{7}$$

Since we are considering the dynamics constraint which makes the feasible action space non-convex, the solution approach using linear programming in the original paper [54] is not applicable. We solve the above optimization problem by the following approach. After calculating the set $ORCA^{\tau}$, we sample N points uniformly in the set S. Among all the sampled points, we choose one point in set $ORCA^{\tau}$ that is closest to the preferred velocity, as shown in Equation (7). However, when the density of air traffic is very high, the intersection of $ORCA^{\tau}$ and S may be empty. In this case, we will select the "safest possible" velocity for the ownship among the N sampled points, i.e., the velocity that minimally "penetrates" the constraints in set $ORCA^{\tau}$.

Here we note the differences between the MCTS algorithm and the ORCA algorithm. First, the ORCA algorithm can select any velocity in the shaded area while the MCTS algorithm can only choose from 9 discrete actions at each time step. We will show the reduced action space compared to ORCA does not hurt the performance of the MCTS algorithm. Second, since the ORCA algorithm does not have a penalty for the ownship flying out of the map, we allow the ownship to fly out of the map and then fly back while implementing the ORCA algorithm. Third, since the original ORCA algorithm is proposed in the deterministic case and we are introducing uncertainties in this paper, we increase the conflict radius of each aircraft to give the ORCA an extra buffer to avoid conflicts between aircraft.

V. Numerical Experiments

A. Simulator

To test the performance of the proposed algorithm and the baseline algorithm, an airspace simulator was built in Python where the aircraft can fly freely in the two-dimensional en route airspace. The airspace has 24km length and 24km width, which is designed for future Urban Air Mobility free flight operations. The assessment of the algorithm involves running 1000 episodes in this simulator and then take the average of the statistics as the algorithm performance. Here one episode means the ownship flying from its initial position to a terminal state.

At the beginning of each episode, the initial position of ownship is at the bottom right corner of the map, and the initial speed of the ownship is set to $60m/s$, the initial flight direction is set to point directly to the center of the map. Then a fixed number of intruder aircraft are generated with speed uniformly distributed between $50m/s$ and $80m/s$ and heading angle uniformly distributed between 0° and 360°. The goal position of the ownship is also uniformly generated on the map.

With the above initialization, after each time step, the state update for the ownship and the intruder aircraft will be based on the dynamical model in Equation (2). When an intruder flies out of the map, a new intruder will be randomly generated in the airspace so that the number of intruder aircraft is fixed. When an intruder aircraft is generated, we prevent it being too close to the ownship, in which case the ownship might not be able to avoid this intruder no matter what action it takes.

During the simulation, the total number of conflicts between the ownship and intruder aircraft will be recorded. When the ownship reaches the goal state or flies out of the map or has a near mid-air collision with any intruder aircraft, this episode will end and a new episode will start. The near mid-air collision (NMAC) standard is defined to be 500 feet by the Aeronautical Information Manual [86]. Note in this simulator, when a conflict happens, we do not terminate this episode but just record this conflict. This is because in a conflict state, the MCTS algorithm can still work to guide ownship escaping this conflict and avoid potential NMAC.

We also note that there is an airspace simulator from NASA named Fe3 (Flexible engine for Fast-time evaluation of Flight environments). It was used to test the performance of different low-altitude high-density air traffic operations [87]–[89]. In their simulator, the goal position and origin are on the boundary of the map. This simulator is more suitable for the case where big airspace was divided into several small sectors and we only need to control aircraft in one small sector to guarantee there is no conflict between aircraft. When the aircraft exit the current sector and move into the next sector, another controller will take over this aircraft. For the simulator used in this paper, it can be used in a scenario where bounded airspace contains both static obstacles (walls or geofences) and dynamic obstacles (intruders or birds), such as airspace above a small city. In this case, the goal position and origin will be in the same airspace region.

B. Results with fixed velocity intruders

In the following experiments, we run 1000 episodes in the simulator for each algorithm with different parameter settings, where we keep the velocity of the intruder aircraft fixed. We record and compare the percentage of these 1000 episodes where the ownship reaches the goal state successfully, the percentage of episode termination due to a NMAC, average conflicts in each episode, and average running time for each decision making step.

1) Performance of MCTS algorithm with different parameters: In the MCTS algorithm, there are two parameters that are important to the performance of this algorithm: the number of simulations n that each time a decision needs to be made and the search depth d, which is discussed in Section IV. In this experiment, the number of intruders is varied from 10 to 80 with step 10, the number of simulations is varied from 100 to 900 with step 200, and the search depth is chosen from 2, 3, and 4. Based on the results and performance comparison, the best parameters n and d are chosen to conduct the second experiment.

Since the performance trend of the MCTS algorithm is consistent for different numbers of intruder aircraft, in Fig. 5 we only show the result where the number of intruder aircraft is 80. The figure also includes error bars indicating 95% confidence interval of the Monte Carlo simulation results.

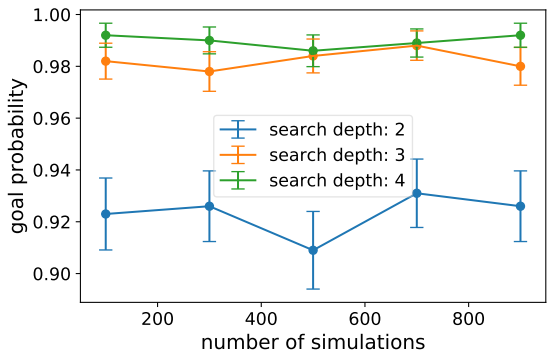
Fig. 5a, 5c, and 5c show that the search depth $d = 2$ performs worse than deeper search depths (the ownship reaches fewer goal states and has more NMACs and conflicts). This is because when the search depth is deeper, the ownship can look further into the future and thus can take action to avoid the conflicts foreseen in the further future. The search depth $d = 3$ and $d = 4$ do not show much difference from the results. They perform equally well. From Fig. 5a, 5b, and 5c, we also observe that the number of simulations does not affect the performance much, which implies that in this problem, the MCTS algorithm does not need too many simulations to have an accurate approximation of the action value.

The average running time in Fig. 5d is growing linearly with the number of simulations and the search depth is also the main factor slowing down the algorithm because the search tree is deeper.

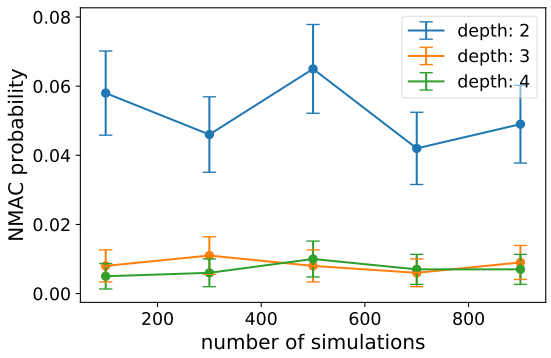
Based on the results in Fig. 5, since increasing the search depth d to 4 and increasing the number of simulations will increase the computation time while cannot bring much performance improvement, the number of simulations $n = 100$ and search depth $d = 3$ were chosen for the next experiment to compare with ORCA algorithm.

2) Performance of ORCA algorithm with different parameters: For the ORCA algorithm, the preset time τ decides how long the agent can "see" in the future, and the number of sampling points N decides the optimality of the selected velocity. In this experiment, we did a stress test (maintain the number of intruder aircraft as 80) for these two parameters with different values, by setting τ to 30, 60, 90 and N to 10, 50, 100, 200. The result is shown in Fig. 6.

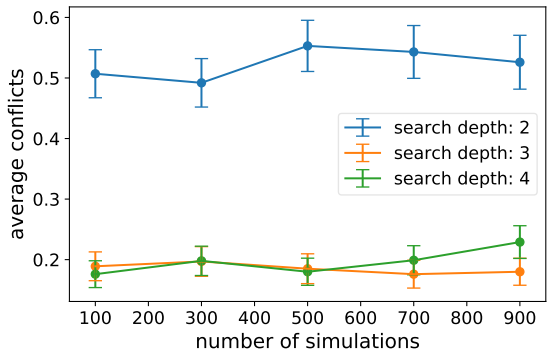
From Fig. 6 we can see that with a larger number of sampling points, the performance is better with more reached goals and fewer conflicts. However, the ORCA algorithm does



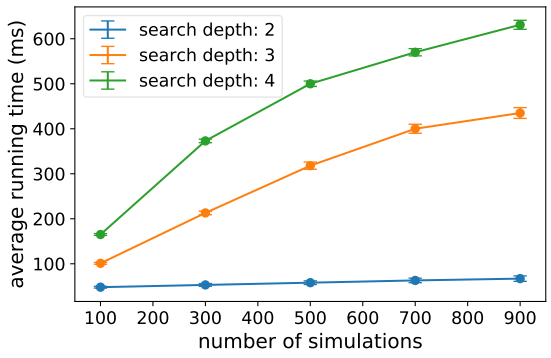
(a) Probability of reaching goal state



(b) Probability of having a NMAC

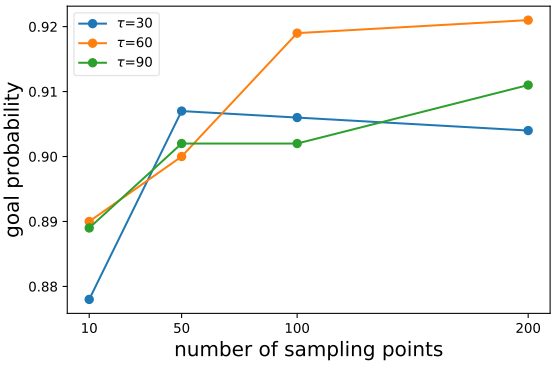


(c) Average conflicts in each episode

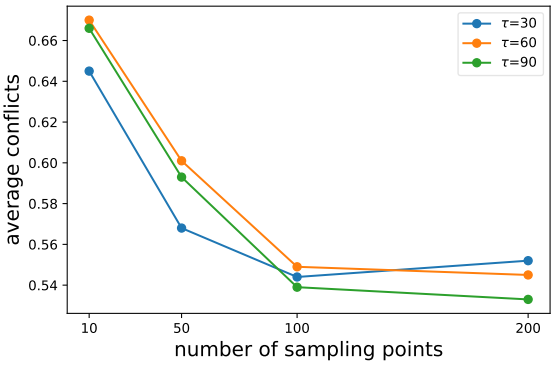


(d) Average running time at each decision step

Fig. 5: Performance of MCTS algorithm with different parameters when there is 80 intruder aircraft.



(a) Probability of reaching goal state



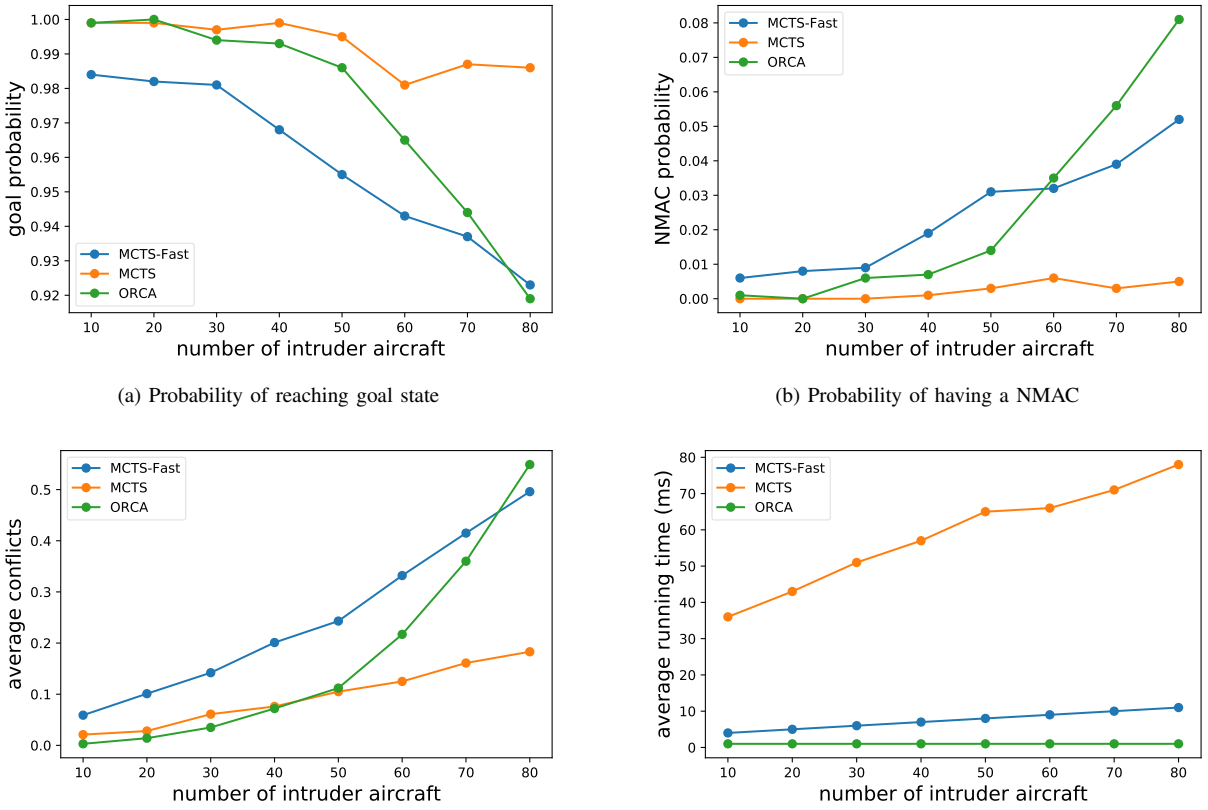
(b) Average conflicts in each episode

Fig. 6: Performance of ORCA algorithm with different parameters when there is 80 intruder aircraft.

not show much improvement by increasing N from 100 to 200. Since the computation time is increasing linearly with N, we choose N to be 100 for the next experiment. Also, we choose the parameter τ to be 60 since from the result it performs slightly better than the other two in terms of the reached goals.

3) Comparison between MCTS, MCTS-Fast, ORCA algorithm: In this experiment, we change the number of intruder aircraft from 10 to 80 with step 10 and compare the performance of the MCTS algorithm and ORCA algorithm, with the parameter values selected from the above experiments. Besides, we also examine the performance of the MCTS algorithm in an extreme case where the decision time is very short. In this extreme case, we only allow the MCTS to run 10 simulations to build the search tree and then select an action. We call this variant of the MCTS algorithm "MCTS-Fast". We generate 1000 same episodes for the three algorithms and Fig. 7 shows the performance results of algorithm MCTS, MCTS-Fast, and ORCA.

The three algorithms in Fig. 7 show similar patterns: with



(a) Probability of reaching goal state

(b) Probability of having a NMAC

(c) Average conflicts in each episode

(d) Average running time at each decision step

Fig. 7: Performance of MCTS, ORCA, MCTS-Fast with different number of intruder aircraft.

the increasing number of intruders, the number of conflicts and probability of having a NMAC are both increasing, and the probability of reaching the goal state is decreasing. For all three algorithms, the ownship can reach the goal state with more than 90% in the 1000 episodes, and the average conflicts in each episode are under 0.6, which means all three algorithms are very effective at avoiding potential collisions. In addition, when the intruder number is below a certain threshold (approximately under 40), the ORCA and MCTS algorithms can help the ownship reach goal state without encountering any NMAC for more than 99% of the time. Also, with a limited computation time, the MCTS-Fast algorithm shows a little performance loss, demonstrating the robustness of the MCTS algorithm.

As shown in Fig. 7b and Fig. 7c, in terms of the percentage of NMAC episodes and the number of average conflicts, the MCTS algorithm performs better than the ORCA algorithm, especially when the intruder aircraft number is large (air traffic density is high).

Figure 7d shows that the computation time of all three algorithms is growing linearly with the increasing number of intruder aircraft. On average, the MCTS algorithm needs around 50ms to 60ms to generate the output action (the running time under 100ms is acceptable for a real-time decision-making system, given that the decision is made every 5 seconds), while the ORCA algorithm only needs 2ms to finish the computation. This shows that the improved performance of the MCTS algorithm comes at the price of longer computation time.

TABLE I: Performance of the baseline

NMAC probability	$51.26\% \pm 0.98\%$
Goal probability	$48.74\% \pm 0.98\%$
Average conflicts	1.92 ± 0.05

In Table 1 we show the result of the baseline where no actions are taken for all of the aircraft (e.g., the aircraft is flying straight towards its goal), and plot the average result over 1000 independent experiments. From this table we can see when no actions are taken, the ownship will have a NMAC event over 50% episodes. In each episode, the ownship has 2 conflicts with other aircraft on average. This baseline results show the promising performance of the proposed computational guidance algorithm under a high-density air traffic scenario.

C. Results with varying velocity intruders

To understand the performance of the MCTS algorithm in more realistic scenarios, in this subsection, we also conduct a numerical experiment with the intruder aircraft that vary velocities stochastically. This more complicated scenario aims to show that the MCTS algorithm is able to adapt to different

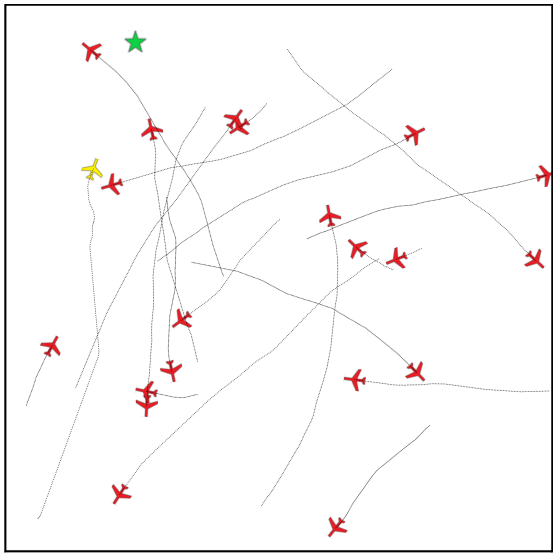


Fig. 8: A scenario with intruder aircraft that change heading stochastically.

behaviors of the intruder aircraft, which is useful when the flight intention information is not available to the ownship.

In this experiment, the intruder aircraft is following the dynamical model described in Equation (2). At each time step, there is 10% probability that the intruder aircraft will change its bank angle at a rate uniformly distributed in the range $[-20°/s, 20°/s]$. Fig. 8 plots the resulting trajectory for this case study, where the trail behind each aircraft represents its historical trajectory.

To incorporate the trajectories uncertainties of the intruder aircraft, in the MCTS algorithm we implemented forward projection to predict the near future position of the intruder aircraft based on its current position and velocity. We also highlighted other alternative trajectory prediction algorithms [90]–[94], and we expect the performance of the MCTS could be improved with a more accurate trajectory prediction algorithm.

The numerical experiment results over 1,000 random episodes for this scenario are shown in Fig. 9, which record the average number of conflicts/NMACs in each episode. The figure also includes error bars indicating 95% confidence interval of the simulation results. We can see the due to the trajectory uncertainty of the intruder aircraft, the average number of conflicts/NMACs is larger compared to scenarios with fixed velocity intruders. But the statistics are still at a low level, which shows the MCTS algorithm is robust to different behavior patterns of the intruder aircraft. Note the experiment result variance in Fig. 9b is larger than the variance in Fig. 9a, since the NMAC event is rarer in the numerical experiment.

D. Discussions

Through the numerical simulation results presented above, we can see that with a longer online computation time, the MCTS algorithm performs better than the ORCA algorithm in terms of the number of conflicts/NMACs. The reason is that at each time step, ORCA selects an action that is only based on the current state without considering the action for the next state. While the MCTS algorithm can look ahead for 3 steps (search depth is 3). When there are 9 actions to choose at each time step, the ownship can estimate the outcome up to $9^3 = 729$ different action combinations (similar to the illustration in Fig. 3, where the ownship can explore 7 different action combinations with search depth 2). With a deeper search depth than the ORCA algorithm, the solution of the MCTS algorithm is closer to the global optimum.

One limitation in this work is that we assume full observability, where the ownship can sense the intruder aircraft information (position and velocity) perfectly. However, in practice when the state information becomes imperfect, some techniques such as Kalman Filter or Partially Observable MDP will be necessary to filter out the sensor noise.

We also observe another limitation of the MCTS algorithm through the numerical experiment. In Fig. 10, we plot the resulted trajectories by following the actions from the MCTS algorithm and ORCA algorithm when there is no intruder aircraft blocking the way to the goal position. In this case, the ORCA algorithm can select a velocity pointing directly to the goal position since the action space for the ORCA algorithm is continuous (the shaded area in Fig. 4). MCTS algorithm only has 3 fixed turning rate of bank angle, which makes it difficult to point directly to the goal position. This will make the ownship keep changing the heading angle when approaching the goal with extra fuel cost.

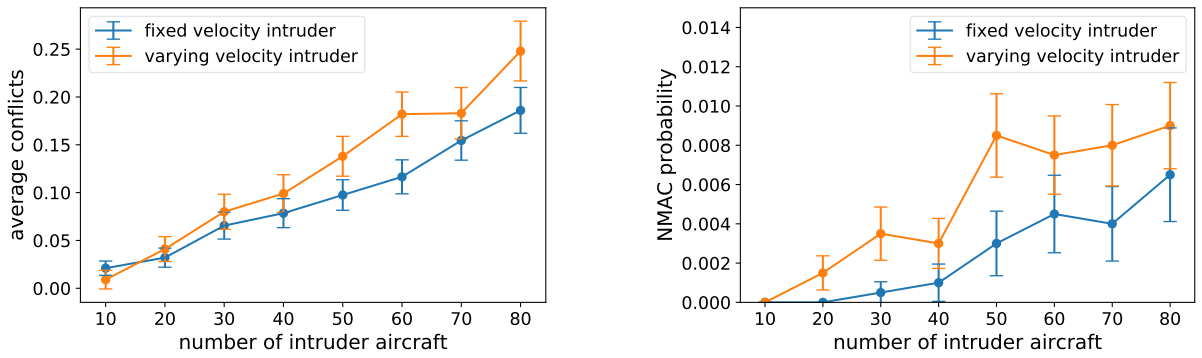
VI. Conclusion

A shift from human-centric air traffic control systems towards higher levels of autonomy is required to enable safe and efficient Urban Air Mobility operations. In this paper, we proposed a computational guidance algorithm with collision avoidance capability for autonomous free flight operations, which can guide the eVTOL aircraft to its destination through controlling the bank angle and acceleration. The problem is formulated as a Markov Decision Process (MDP) with uncertainty in aircraft dynamics and the environment. This MDP is solved by Monte Carlo Tree Search (MCTS) algorithm with an estimated value function to reduce the computation time. Numerical experiments in the airspace simulator with different intruder aircraft behaviors show that our algorithm has promising performance and outperforms the baseline algorithm Optimal Reciprocal Collision Avoidance (ORCA) in dense air traffic scenarios. This proposed algorithm provides a potential solution framework to enable autonomous on-demand free flight operations in urban air mobility.

The contribution of this research is to provide a state-of-the-art computational guidance algorithm with collision avoidance capability. Furthermore, this research integrates the power of onboard aircraft autonomy and the advantage of the free flight concept for airspace operations to enable safe and efficient flight operations in on-demand urban air transportation.

Acknowledgment

This material is based upon work supported by the National Science Foundation under award #1565979. This work



(a) Average conflicts in each episode

(b) Probability of having a NMAC

Fig. 9: Performance of MCTS algorithm with fixed velocity intruders and varying velocity intruder.

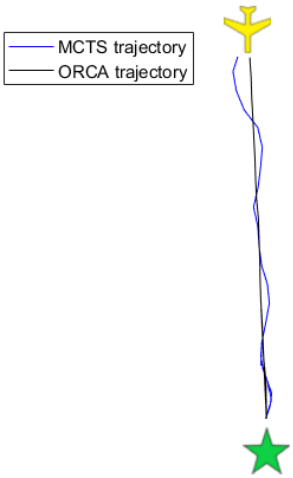


Fig. 10: The trajectories generated by MCTS algorithm and ORCA algorithm when there is no intruder.

has benefited from discussions with Mykel Kochenderfer at Stanford, Karthik Balakrishnan and Richard Golding at A^3 by Airbus, and Josh Bertram at Iowa State University. The authors thank Mykel Kochenderfer for inspiring the research idea, the A^3 Altiscope team for their guidance and support throughout this work, and Josh Bertram for the discussion on the numerical experiments.

References

[1] L. Gipson, "Nasa embraces urban air mobility, calls for market study," https://www.nasa.gov/aero/nasa-embraces-urban-air-mobility, 2017, accessed: 2020-05-19.

[2] J. Holden and N. Goel, "Fast-forwarding to a future of on-demand urban air transportation," *San Francisco, CA*, 2016.

[3] "Future of urban mobility," http://www.airbus.com/newsroom/news/en/2016/12/My-Kind-Of-Flyover.html, 2017, accessed: 2020-05-19.

[4] E. R. Mueller, P. H. Kopardekar, and K. H. Goodrich, "Enabling airspace integration for high-density on-demand mobility operations," in *17th AIAA Aviation Technology, Integration, and Operations Conference*, 2017, p. 3086.

[5] J. M. Hoekstra, R. N. van Gent, and R. C. Ruigrok, "Designing for safety: the 'free flight' air traffic management concept," *Reliability Engineering & System Safety*, vol. 75, no. 2, pp. 215–232, 2002.

[6] K. D. Bilimoria, S. R. Grabbe, K. S. Sheth, and H. Q. Lee, "Performance evaluation of airborne separation assurance for free flight," *Air Traffic Control Quarterly*, vol. 11, no. 2, pp. 85–102, 2003.

[7] M. S. V. Clari, R. C. Ruigrok, J. M. Hoekstra, and H. G. Visser, "Cost-benefit study of free flight with airborne separation assurance," *Air Traffic Control Quarterly*, vol. 9, no. 4, pp. 287–309, 2001.

[8] C. Tomlin, G. J. Pappas, and S. Sastry, "Conflict resolution for air traffic management: A study in multiagent hybrid systems," *IEEE Transactions on automatic control*, vol. 43, no. 4, pp. 509–521, 1998.

[9] S. Kahne and I. Frolow, "Air traffic management: Evolution with technology," *IEEE Control Systems*, vol. 16, no. 4, pp. 12–21, 1996.

[10] W. H. Harman, "Tcas- a system for preventing midair collisions," *The Lincoln Laboratory Journal*, vol. 2, no. 3, pp. 437–457, 1989.

[11] J. Krozel and M. Peters, "Conflict detection and resolution for free flight," *Air Traffic Control Quarterly*, vol. 5, no. 3, pp. 181–212, 1997.

[12] Z. Lovering, "Vahana configuration trade study - part i – vahana," https://acubed.airbus.com/blog/vahana/vahana-configuration-trade-study-part-i/, 2016, accessed: 2020-05-19.

[13] T. Schouwenaars, J. How, and E. Feron, "Decentralized cooperative trajectory planning of multiple aircraft with hard safety guarantees," in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, 2004, p. 5141.

[14] R. Siegwart, I. R. Nourbakhsh, and D. Scaramuzza, *Introduction to autonomous mobile robots*. MIT press, 2011.

[15] E. Frazzoli, Z.-H. Mao, J.-H. Oh, and E. Feron, "Resolution of conflicts involving many aircraft via semidefinite programming," *Journal of Guidance, Control, and Dynamics*, vol. 24, no. 1, pp. 79–86, 2001.

[16] A. U. Raghunathan, V. Gopal, D. Subramanian, L. T. Biegler, and T. Samad, "Dynamic optimization strategies for three-dimensional conflict resolution of multiple aircraft," *Journal of guidance, control, and dynamics*, vol. 27, no. 4, pp. 586–594, 2004.

[17] P. J. Enright and B. A. Conway, "Discrete approximations to optimal trajectories using direct transcription and nonlinear programming," *Journal of Guidance, Control, and Dynamics*, vol. 15, no. 4, pp. 994–1002, 1992.

[18] T. Schouwenaars, B. De Moor, E. Feron, and J. How, "Mixed integer programming for multi-vehicle path planning," in *Control Conference (ECC), 2001 European*. IEEE, 2001, pp. 2603–2608.

[19] A. Richards and J. P. How, "Aircraft trajectory planning with collision avoidance using mixed integer linear programming," in *American Control Conference, 2002. Proceedings of the 2002*, vol. 3. IEEE, 2002, pp. 1936–1941.

[20] L. Pallottino, E. M. Feron, and A. Bicchi, "Conflict resolution problems for air traffic management systems solved with mixed integer programming," *IEEE transactions on intelligent transportation systems*, vol. 3, no. 1, pp. 3–11, 2002.

[21] A. Vela, S. Solak, W. Singhose, and J.-P. Clarke, "A mixed integer program for flight-level assignment and speed control for conflict resolution," in *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.

[22] D. Mellinger, A. Kushleyev, and V. Kumar, "Mixed-integer quadratic program trajectory generation for heterogeneous quadrotor teams," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE, 2012, pp. 477–483.

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

[24] D. Morgan, S.-J. Chung, and F. Y. Hadaegh, "Model predictive control of swarms of spacecraft using sequential convex programming," *Journal of Guidance, Control, and Dynamics*, vol. 37, no. 6, pp. 1725–1740, 2014.

[25] B. Acikmese and S. R. Ploen, "Convex programming approach to powered descent guidance for mars landing," *Journal of Guidance, Control, and Dynamics*, vol. 30, no. 5, pp. 1353–1366, 2007.

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

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

[28] M. Pontani and B. A. Conway, "Particle swarm optimization applied to space trajectories," *Journal of Guidance, Control, and Dynamics*, vol. 33, no. 5, pp. 1429–1441, 2010.

[29] G. Hoffmann, D. G. Rajnarayan, S. L. Waslander, D. Dostal, J. S. Jang, and C. J. Tomlin, "The stanford testbed of autonomous rotorcraft for multi agent control (starmac)," in *The 23rd Digital Avionics Systems Conference (IEEE Cat. No. 04CH37576)*, vol. 2. IEEE, 2004, pp. 12–E.

[30] J. K. Howlet, G. Schulein, and M. H. Mansur, "A practical approach to obstacle field route planning for unmanned rotorcraft," 2004.

[31] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[32] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," 1998.

[33] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The international journal of robotics research*, vol. 30, no. 7, pp. 846–894, 2011.

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

[35] S. Wollkind, J. Valasek, and T. Ioerger, "Automated conflict resolution for air traffic management using cooperative multiagent negotiation," in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, 2004, p. 4992.

[36] O. Purwin, R. D'Andrea, and J.-W. Lee, "Theory and implementation of path planning by negotiation for decentralized agents," *Robotics and Autonomous Systems*, vol. 56, no. 5, pp. 422–436, 2008.

[37] V. R. Desaraju and J. P. How, "Decentralized path planning for multi-agent teams in complex environments using rapidly-exploring random trees," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 2011, pp. 4956–4961.

[38] G. Inalhan, D. M. Stipanovic, and C. J. Tomlin, "Decentralized optimization, with application to multiple aircraft coordination," in *Decision and Control, 2002, Proceedings of the 41st IEEE Conference on*, vol. 1. IEEE, 2002, pp. 1147–1155.

[39] A. Richards and J. How, "Decentralized model predictive control of cooperating uavs," in *43rd IEEE Conference on Decision and Control*, vol. 4. Citeseer, 2004, pp. 4286–4291.

[40] D. H. Shim and S. Sastry, "An evasive maneuvering algorithm for uavs in see-and-avoid situations," in *American Control Conference, 2007. ACC'07*. IEEE, 2007, pp. 3886–3891.

[41] D. H. Shim, H. J. Kim, and S. Sastry, "Decentralized nonlinear model predictive control of multiple flying robots," in *Decision and control, 2003. Proceedings. 42nd IEEE conference on*, vol. 4. IEEE, 2003, pp. 3621–3626.

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

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

[44] C. I. Connolly, J. B. Burns, and R. Weiss, "Path planning using laplace's equation," in *Robotics and Automation, 1990. Proceedings., 1990 IEEE International Conference on*. IEEE, 1990, pp. 2102–2106.

[45] T. Zhang, G. Kahn, S. Levine, and P. Abbeel, "Learning deep control policies for autonomous aerial vehicles with mpc-guided policy search," in *Robotics and Automation (ICRA), 2016 IEEE International Conference on*. IEEE, 2016, pp. 528–535.

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

[47] D.-T. Pham, N. P. Tran, S. Alam, V. Duong, and D. Delahaye, "A machine learning approach for conflict resolution in dense traffic scenarios with uncertainties," 2019.

[48] S. Li, M. Egorov, and M. Kochenderfer, "Optimizing collision avoidance in dense airspace using deep reinforcement learning," *arXiv preprint arXiv:1912.10146*, 2019.

[49] J. Hu, X. Yang, W. Wang, P. Wei, L. Ying, and Y. Liu, "Uas conflict resolution in continuous action space using deep reinforcement learning," in *AIAA AVIATION 2020 FORUM*, 2020, p. 2909.

[50] X. Yang and P. Wei, "Autonomous on-demand free flight operations in urban air mobility using monte carlo tree search," in *8th International Conference on Research in Air Transportation (ICRAT)*. ICRAT, 2018.

[51] S.-C. Han, H. Bang, and C.-S. Yoo, "Proportional navigation-based collision avoidance for uavs," *International Journal of Control, Automation and Systems*, vol. 7, no. 4, pp. 553–565, 2009.

[52] J.-W. Park, H.-D. Oh, and M.-J. Tahk, "Uav collision avoidance based on geometric approach," in *SICE Annual Conference, 2008*. IEEE, 2008, pp. 2122–2126.

[53] J. Krozel, M. Peters, and K. Bilimoria, "A decentralized control strategy for distributed air/ground traffic separation," in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, 2000, p. 4062.

[54] J. Van Den Berg, S. J. Guy, M. Lin, and D. Manocha, "Reciprocal n-body collision avoidance," in *Robotics research*. Springer, 2011, pp. 3–19.

[55] S. Temizer, M. Kochenderfer, L. Kaelbling, T. Lozano-Pérez, and J. Kuchar, "Collision avoidance for unmanned aircraft using markov decision processes," in *AIAA guidance, navigation, and control conference*, 2010, p. 8040.

[56] M. J. Kochenderfer and J. Chryssanthacopoulos, "Robust airborne collision avoidance through dynamic programming," *Massachusetts Institute of Technology, Lincoln Laboratory, Project Report ATC-371*, 2011.

[57] M. J. Kochenderfer, J. E. Holland, and J. P. Chryssanthacopoulos, "Next-generation airborne collision avoidance system," Massachusetts Institute of Technology-Lincoln Laboratory Lexington United States, Tech. Rep., 2012.

[58] T. B. Wolf and M. J. Kochenderfer, "Aircraft collision avoidance using monte carlo real-time belief space search," *Journal of Intelligent & Robotic Systems*, vol. 64, no. 2, pp. 277–298, 2011.

[59] R. Bellman, "A markovian decision process," *Indiana Univ. Math. J.*, vol. 6, pp. 679–684, 1957.

[60] R. A. Howard, "Dynamic programming and markov processes," 1964.

[61] D. J. White, "A survey of applications of markov decision processes," *Journal of the operational research society*, vol. 44, no. 11, pp. 1073–1096, 1993.

[62] E. A. Feinberg and A. Shwartz, *Handbook of Markov decision processes: methods and applications*. Springer Science & Business Media, 2012, vol. 40.

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

[64] S. Thrun, "Probabilistic robotics," *Communications of the ACM*, vol. 45, no. 3, pp. 52–57, 2002.

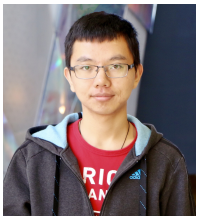
[65] M. Mariton, *Jump linear systems in automatic control*. M. Dekker New York, 1990.

[66] R. Coulom, "Efficient selectivity and backup operators in monte-carlo tree search," in *International conference on computers and games*. Springer, 2006, pp. 72–83.

[67] C. B. Browne, E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton, "A survey of monte carlo tree search methods," *IEEE Transactions on Computational Intelligence and AI in games*, vol. 4, no. 1, pp. 1–43, 2012.

[68] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam,

M. Lanctot *et al.*, "Mastering the game of go with deep neural networks and tree search," *nature*, vol. 529, no. 7587, p. 484, 2016.
[69] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel *et al.*, "Mastering chess and shogi by self-play with a general reinforcement learning algorithm," *arXiv preprint arXiv:1712.01815*, 2017.
[70] A. J. Champandard, "Monte-carlo tree search in total war: Rome ii's campaign ai," *AIGameDev.com: http://aigamedev.com/open/coverage/mcts-rome-ii*, 2014.
[71] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton *et al.*, "Mastering the game of go without human knowledge," *Nature*, vol. 550, no. 7676, p. 354, 2017.
[72] "Acubed vahana," http://evtol.news/aircraft/a3-by-airbus/, Nov 2018, accessed: 2020-05-19.
[73] C. Allignol, N. Barnier, N. Durand, G. Manfredi, and É. Blond, "Assessing the robustness of a uas detect & avoid algorithm," in *12th USA/Europe Air Traffic Management Research and Development Seminar*, 2017.
[74] H. A. Blom and G. Bakker, "Safety evaluation of advanced self-separation under very high en route traffic demand," *Journal of Aerospace Information Systems*, vol. 12, no. 6, pp. 413–427, 2015.
[75] H. Y. Ong and M. J. Kochenderfer, "Markov decision process-based distributed conflict resolution for drone air traffic management," *Journal of Guidance, Control, and Dynamics*, pp. 69–80, 2016.
[76] K. D. Julian, J. Lopez, J. S. Brush, M. P. Owen, and M. J. Kochenderfer, "Policy compression for aircraft collision avoidance systems," in *Digital Avionics Systems Conference (DASC), 2016 IEEE/AIAA 35th*. IEEE, 2016, pp. 1–10.
[77] P. Pradeep and P. Wei, "Energy optimal speed profile for arrival of tandem tilt-wing evtol aircraft with rta constraint," *IEEE CSAA Guidance, Navigation and Control Conference*, 2018.
[78] Airbus. (2016) Acubed by airbus group. https://www.airbus-sv.com/projects/1. Accessed: 2020-05-19.
[79] G. Tesauro and G. R. Galperin, "On-line policy improvement using monte-carlo search," in *Advances in Neural Information Processing Systems*, 1997, pp. 1068–1074.
[80] C. Bosson and T. A. Lauderdale, "Simulation evaluations of an autonomous urban air mobility network management and separation service," in *2018 Aviation Technology, Integration, and Operations Conference*, 2018, p. 3365.
[81] S. P. Cook and D. Brooks, "A quantitative metric to enable unmanned aircraft systems to remain well clear," *Air Traffic Control Quarterly*, vol. 23, no. 2-3, pp. 137–156, 2015.
[82] L. Kocsis and C. Szepesvári, "Bandit based monte-carlo planning," in *European conference on machine learning*. Springer, 2006, pp. 282–293.
[83] L. Kocsis, C. Szepesvári, and J. Willemson, "Improved monte-carlo search," *Univ. Tartu, Estonia, Tech. Rep*, vol. 1, 2006.
[84] D. Alejo, J. Cobano, G. Heredia, and A. Ollero, "Optimal reciprocal collision avoidance with mobile and static obstacles for multi-uav systems," in *Unmanned Aircraft Systems (ICUAS), 2014 International Conference on*. IEEE, 2014, pp. 1259–1266.
[85] P. Conroy, D. Bareiss, M. Beall, and J. v. d. Berg, "3-d reciprocal collision avoidance on physical quadrotor helicopters with on-board sensing for relative positioning," *arXiv preprint arXiv:1411.3794*, 2014.
[86] Federal Aviation Administration. (2017) Near midair collision reporting. http://www.faraim.org/aim/aim-4-03-14-530.html. Accessed: 2020-05-19.
[87] M. Xue and J. Rios, "Initial study of an effective fast-time simulation platform for unmanned aircraft system traffic management," in *17th AIAA Aviation Technology, Integration, and Operations Conference*, 2017, p. 3073.
[88] V. Bulusu, R. Sengupta, E. R. Mueller, and M. Xue, "A throughput based capacity metric for low-altitude airspace," in *2018 Aviation Technology, Integration, and Operations Conference*, 2018, p. 3032.
[89] M. Xue, J. Rios, J. Silva, Z. Zhu, and A. K. Ishihara, "Fe3: An evaluation tool for low-altitude air traffic operations," in *2018 Aviation Technology, Integration, and Operations Conference*, 2018, p. 3848.
[90] D. H. Williams and S. M. Green, "Flight evaluation of center-tracon automation system trajectory prediction process," 1998.
[91] M. Prandini, J. Hu, J. Lygeros, and S. Sastry, "A probabilistic approach to aircraft conflict detection," *IEEE Transactions on intelligent transportation systems*, vol. 1, no. 4, pp. 199–220, 2000.
[92] W. Liu and I. Hwang, "Probabilistic trajectory prediction and conflict detection for air traffic control," *Journal of Guidance, Control, and Dynamics*, vol. 34, no. 6, pp. 1779–1789, 2011.
[93] S. T. Barratt, M. J. Kochenderfer, and S. P. Boyd, "Learning probabilistic trajectory models of aircraft in terminal airspace from position data," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 9, pp. 3536–3545, 2018.
[94] Y. Pang, H. Yao, J. Hu, and Y. Liu, "A recurrent neural network approach for aircraft trajectory prediction with weather features from sherlock," in *AIAA Aviation 2019 Forum*, 2019, p. 3413.



Xuxi Yang is a fourth-year Ph.D. candidate in Department of Aerospace Engineering at Iowa State University, working as a research assistant at the Intelligent Aerospace Systems Laboratory (IASL) under the supervision of Professor Peng Wei. He received his bachelor's degree at Harbin Institute of Technology with major in Applied Mathematics. His research interests include Deep Reinforcement Learning, Machine Learning, Decision Theory, with applications in Air Traffic Control/Management (ATC/M), UAS Traffic Management (UTM).

Peng Wei is an assistant professor in Iowa State University Aerospace Engineering Department, with courtesy appointments in Electrical and Computer Engineering Department and Computer Science Department. By contributing to the intersection of control, optimization, machine learning, and artificial intelligence, he develops autonomy and human-in-the-loop decision support tools for air transportation and aviation systems. His current focus is on safety, efficiency, and scalability for decision-making systems in uncertain and dynamic environments. Recent applications include: Air Traffic Control/Management (ATC/M), Airline Operations, UAS Traffic Management (UTM), eVTOL Urban Air Mobility (UAM) and Autonomous Drone Racing (ADR). Prof. Wei received his Ph.D. degree in Aerospace Engineering from Purdue University in 2013 and a bachelor degree in Automation from Tsinghua University in 2007. He is an associate editor for the AIAA Journal of Aerospace Information Systems.