Reachability based Online Safety Verification for High-Density Urban Air Mobility Trajectory Planning

Abenezer G. Taye*
George Washington University, Washington, DC, 20052, USA

Joshua R. Bertram†
Iowa State University, Ames, IA, 50021, USA

Chuchu Fan‡
Massachusetts Institute of Technology, Cambridge, MA, 02139, USA

Peng Wei§
George Washington University, Washington, DC, 20052, USA

This paper presents a safe and scalable real-time trajectory planning framework for high-density Urban Air Mobility (UAM). The framework extends our previously developed highly efficient Markov Decision Process (MDP) based trajectory planner by adopting a correct-by-construction approach to verify the safety of planning actions. The trajectory planner works in a decentralized manner that allows each aircraft to generate its safe trajectory by accounting for the reachable sets of nearby aircraft. The proposed safety verification module employs a highly scalable data-driven reachability analysis tool to ensure collision-free trajectory planning. Furthermore, the utilized tool over-approximates the reachable set of each aircraft using a discrepancy function, which it learns online from simulation traces. We finally demonstrate the efficacy of the proposed trajectory planner with simulation experiments of up to 120 aircraft in a UAM setting.

I. Introduction

Urban Air Mobility (UAM) is a novel concept in which partially or fully autonomous air vehicles transport passengers and cargo in dense urban environments by providing a safe, efficient, and accessible on-demand air transportation system [1]. For early adopters, this technology offers the promise of bypassing ground-based freeways and hours-long commutes. As the technology matures, it will connect urban centers with outlying towns extending the reach of metropolitan areas.

UAM operation is a multi-agent safety-critical application, where safety and scalability are the two primary design considerations. Therefore, a UAM trajectory planning framework needs to generate trajectories efficiently while guaranteeing that the generated trajectories satisfy the system's safety requirements. These two problems — developing a scalable trajectory planner and safety verification of autonomous systems — are fundamentally challenging problems in and of themselves, and are often addressed independently in the literature. However, in the UAM setting, we need to consider both requirements simultaneously.

The task of guaranteeing the safe operation of autonomous systems is often referred to as verification and validation. Several approaches to verification and validation have been proposed in the literature. These approaches can be broadly classified as formal methods and sampling-based approaches. Sampling-based approaches generate a finite number of scenarios to assess the performance of a system. Hence, they have the advantage of being easier to implement and evaluate the performance of an autonomous system. However, they can not account for all possible behaviors of the system, which is an essential element in verification and validation. Therefore, in recent years, formal methods are receiving considerable research attention due to their ability to capture all the possible behaviors of the system.

*Graduate Student, Department of Mechanical & Aerospace Engineering, abenezertaye@gwu.edu, AIAA Student Member.
†Graduate Student, Department of Electrical & Computer Engineering, bertram1@iastate.edu, AIAA Student Member.
‡Assistant Professor, Department of Aeronautics and Astronautics, chuchu@mit.edu
§Assistant Professor, Department of Mechanical & Aerospace Engineering, pwei@gwu.edu, AIAA Senior Member.

This study aims to address the problem of developing a UAM trajectory planning framework that is computationally efficient and guarantees safe navigation of UAM aircraft. The two main components of the proposed framework are the Markov Decision Process (MDP) based trajectory planner and a reachability analysis module, which the trajectory planner utilizes to gather information about the future states of the aircraft. The approaches we used to formulate the trajectory planning problem and compute the reachable sets of the aircraft are proven to be highly scalable [2] [3]. Adopting such formulations enables the developed UAM trajectory planning framework to be computationally efficient. Furthermore, the algorithm allows each aircraft to make its own decisions in a distributed manner using inputs from sensors such as radar, LIDAR, or systems such as ADS-B.

II. Related Work

NASA, Uber, and Airbus have been exploring the use of vertical takeoff and landing (VTOL) aircraft for Urban Air Mobility (UAM) [4–8]. In general, the UAM concept calls for UAM aircraft taking off from small-scale airports known as vertiports, where VTOL aircraft depart and arrive.

An unstructured airspace approach is known as *"free flight"* has been proposed to cope with the ongoing congestion of the current ATC system. It was shown in [9, 10] that free flight with airborne separation can handle a higher traffic density, and [11] found free flight can also bring fuel and time efficiency. In a free flight framework, each aircraft is responsible for separation assurance and conflict resolution. [12] shows that free flight is potentially feasible due to enabling technologies such as Global Positioning Systems (GPS), data link communications like Automatic Dependence Surveillance-Broadcast (ADS-B) [13], Traffic Alert and Collision Avoidance Systems (TCAS) [14], but would require robust onboard computation.

In terms of algorithms used for trajectory planning, there is extensive literature on the topic. These algorithms can be broadly classified as centralized and decentralized methods. In centralized methods, the state of each aircraft, obstacles, trajectory constraints, and the terminal area's state are observable to the controller via sensors, radar, etc., and a central supervising controller resolves conflicts between aircraft. The central controller precomputes trajectories for all aircraft before flight, typically by formulating the problem in an optimal control framework and solving the problem with various methods; examples are: 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]. One common thread among centralized approaches is that to pursue a global optimum, they must consider each aircraft and obstacle in space, which leads to scalability issues with large numbers of aircraft and obstacles. In addition, as new aircraft enter the scene, centralized algorithms typically need to recompute part or all of the problem to arrive at a new global optimum.

On the other hand, decentralized methods scale better with the number of aircraft and objects in the system but typically cannot obtain globally optimal solutions. Furthermore, decentralized methods may be deemed more robust than centralized approaches [29] because they are not generally prone to a single point of failure. In decentralized systems, each aircraft resolves conflicts locally, and the underlying method can be considered either cooperative or non-cooperative. Computational scalability and solution quality or optimality are significant design trade-offs between centralized and decentralized trajectory planning strategies. In [3], we proposed an MDP-based decentralized UAM trajectory planning algorithm that is highly scalable. The algorithm operates in a free flight manner, and in this study, it is extended by incorporating an online safety verification module that enables the trajectory planner to generate safe trajectories.

From a safety verification standpoint, trajectory planning of autonomous systems has recently been studied in two main directions: *design-then-verify* and *verify-while-design*. *Design-then-verify* is a commonly used approach where the task of trajectory planning is performed first; then, the system is evaluated using different verification tools to determine whether it satisfies the safety requirements or not. However, this approach is computationally inefficient and often fails to give the necessary guarantees [30]. On the other hand, the *verify-while-design* approach, also known as correct-by-construction, integrates the verification process into the control design in a closed-loop manner. Thus the approach becomes computationally efficient and enables the system to satisfy the safety requirements by its very nature.

In this study, we adopted the *verify-while-design* approach to formally synthesize each aircraft's trajectory online. An efficient reachability analysis module that conducts an exhaustive exploration of all possible behaviors of an aircraft has been used to satisfy the reach-avoid property of the system. Several reachability analysis formulations of a dynamical system have been proposed in the literature. These methods include Hamilton-Jacobi-based reachability analysis formulations [31], CORA [32], SpaceEx [33], and Flow* [34]. Although these approaches provide formal soundness guarantees, they are computationally expensive. Hence, they can not be used online in the presence of a large number of aircraft. In this study, to over-approximate the reachable set of an aircraft, we adopted a sensitivity analysis-based tool known as DryVR [2]. DryVR has been demonstrated to be highly scalable and recently implemented in [35] to generate safe operation volume for unmanned aircraft systems (UAS) traffic management. The reachability analysis module, then, is integrated with our previously developed Markov Decision Process (MDP) based trajectory planner [3] to guide the motion of multiple UAM vehicles between vertiports.

III. Modeling and Solution Methods

A. Vehicle Dynamics

The aircraft model used in this paper is based on a pseudo-6DOF formulation proposed in [36]. The performance limits provided in Table 1 are enforced to make the aircraft perform similar to Dubin's aircraft [37] with gentle climbs and bank angles suitable for a UAM passenger aircraft.

The kinematic equations are:

$$\dot{x} = V \cos\gamma \cos\psi \tag{1}$$

$$\dot{y} = V \cos\gamma \sin\psi \tag{2}$$

$$\dot{z} = V \sin\gamma \tag{3}$$

The equations of motion for the aircraft are:

$$\dot{V} = g\left[n_x \cos\alpha - \sin\gamma\right], \tag{4}$$

$$\dot{\gamma} = \frac{g}{V}\left[n_f \cos\phi - \cos\gamma\right], \tag{5}$$

$$\dot{\psi} = g\left[\frac{n_f \sin\phi}{V\cos\gamma}\right], \tag{6}$$

where the acceleration exerted out the top of the aircraft n_f in g is defined as:

$$n_f = n_x \sin\alpha + L. \tag{7}$$

In the above equations, x, y, z represent the position in NED coordinates in meters, where altitude $h = -z$, V is airspeed in meters/second, ϕ, ψ, γ, α denote roll angle, horizontal azimuth angle, flight path angle, and angle of attack with respect to the flight path vector, respectively in radians. n_x denotes throttle acceleration directed out the aircraft's nose in g's, and L is lift acceleration.

V_{min} (Kts)	V_{max} (Kts)	$\dot{\psi}_{min}$ (deg/sec)	$\dot{\psi}_{max}$ (deg/sec)	α_{min} (deg)	α_{max} (deg)	ϕ_{min} (deg)	ϕ_{max} (deg)	γ_{min} (deg)	γ_{max} (deg)
47	133	-30	30	-5	20	-20	20	-20	20

Table 1 **Limits on aircraft performance to approximate an air taxi**

B. Markov Decision Process (MDP) Formulation

In this paper, we formulate aircraft trajectory planning as Markov decision process (MDP), where the state transitions will be governed by the vehicle dynamics described in Section III.A. MDPs are formulated as the tuple (s_t, a_t, r_t, t) where $s_t \in S$ is the state at a given time t, $a_t \in A$ is the action taken by the agent at time t as a result of the decision process. r_t is the reward received by the agent as a result of taking the action a_t from s_t and arriving at s_{t+1}, and $T(s_t, a, s_{t+1})$ is a transition function that describes the dynamics of the environment and capture the probability $p(s_{t+1}|s_t, a_t)$ of transitioning to a state s_{t+1} given the action a_t taken from state s_t.

A policy π can be defined that maps each state $s \in S$ to action $a \in A$. From a given policy $\pi \in \Pi$ a value function $V^{\pi}(S)$ can be computed that computes the expected return that will be obtained within the environment by following the policy π.

The solution of an MDP is termed the optimal policy π^*, which defines the optimal action $a^* \in A$ that can be taken from each state $s \in S$ to maximize the expected return. From this optimal policy π^*, the optimal value function $V^*(s)$ can be computed, which describes the maximum expected value that can be obtained from each state $s \in S$. Furthermore, from the optimal value function $V^*(s)$, the optimal policy π^* can also easily be recovered.

1. State Space

The environment is considered as a continuous state space placed on a volume of $25km \times 25km \times 25km$. Given the dynamics of an aircraft:

$$\dot{\zeta}(t) = f(\zeta(t), u(t)), \tag{8}$$

where, $f : \mathbb{R}^n \times \mathbb{R} \longrightarrow \mathbb{R}^n$ is a continuous function. ζ denotes the aircraft states, which includes the x, y, z position, heading angle ψ, the flight path angle γ, and the speed V. The trajectory of an aircraft $\xi : \mathbb{R}^n \times \mathbb{R}_{\geq 0} \longrightarrow \mathbb{R}^n$ is the solution to the differential equation (8) and represents how the state variables of the aircraft evolve through time. For a given initial set $x_0 \in \mathbb{R}^n$, the state of the system at time t is $\xi(\zeta_0, t) = \zeta(t)$. The control input $u(t)$ is comprised of the thrust n_x, the rate of change of angle of attack $\dot{\alpha}$, and the rate of change of the roll angle $\dot{\phi}$. In addition, a single state in the state space (s_o) contains all the states of an aircraft (ζ) and the position and velocity of every other aircraft. For each other aircraft $f_j, \in J$: the position $f_{j,x}, f_{j,y}, f_{j,z}$ and velocity $f_{j,v_x}, f_{j,v_y}, f_{j,v_z}$. Thus, we can define s_o as $s_o = [\zeta, f_1, ..., f_j]$, where j represents the number of other aircraft.

2. Action Space

The action space of the MDP is composed of the individual action spaces of the three inputs: thrust n_x, rate of change of attack $(\dot{\alpha})$, and rate of change of the roll angle $(\dot{\phi})$. The action space of n_x is composed of discrete values sampled from a linear function, $n_x = [-2, -1, 0, 1, 2, 3, 4]$. On the other hand, the action space of $\dot{\alpha}$ and $\dot{\phi}$ is a discrete set of actions that are sampled from a logarithm function through the range of each input. Such an action space enables one to take more control actions when the inputs are near zero and coarse control actions as the aircraft get further away from their trajectory. As a result, fine control actions can be taken when a small correcting action to adjust small deviations from the trajectory is desired, and large control actions can be taken when a significant change in the course of the aircraft trajectory is desired. Consequently, the inputs of $\dot{\alpha}$ and $\dot{\phi}$ are logarithmically spaced within a range of 15 input values.

The logarithmically spaced input set in degree/second are computed as follows:

$$\dot{\alpha} = [-19.99, -16.24, -12.66, -9.26, -6.02, -2.94, -0.01, 0, 0.01, 2.94, 6.02, 9.26, 12.66, 16.24, 19.99] \tag{9}$$

$$\dot{\phi} = [-19.99, -16.24, -12.66, -9.26, -6.02, -2.94, -0.01, 0, 0.01, 2.94, 6.02, 9.26, 12.66, 16.24, 19.99] \tag{10}$$

Finally, the joint action space becomes:

$$\mathcal{A} = \{\dot{\alpha}, \dot{\phi}, n_x\}. \tag{11}$$

3. Reward Function

The primary mechanism we use to control the behavior of an MDP agent is through the reward function. A reward function $R(s_t, a_t, s_{t+1})$ represents the reward that an agent, currently at s_t, collects after taking a control action a_t and arriving at s_{t+1}. In this work, we have utilized both positive and negative rewards, as depicted in Table 2, to guide the aircraft to their destination while avoiding possible collision with other nearby aircraft and the terrain.

Reward source	Reward magnitude	Location	Decay factor	Comment
Nearby intruder aircraft	-1000	Inside reachable-set of intruder	0.97	Collision avoidance
Terrain	-1000	Manually placed	0.99	Terrain avoidance
Destination	200	Manually placed	0.999	Vertiport attraction

Table 2 **Reward function for each aircraft**

C. Reachability Analysis

One of the critical components of the present trajectory planning scheme is computing a reachable set for each nearby intruder aircraft. In this study, the concept of discrepancy function is adopted from [2] to formulate the reachability analysis problem. This section summarizes discrepancy functions and how they can be used to compute the reachable set of a dynamical system.

A *discrepancy function* is a continuous function that is primarily used to formally measure the convergence or divergence nature of trajectories [38]. Hence, it generates the over-approximation of the reachable set by providing the upper and lower bounds of the trajectories. In [38], it has been demonstrated that discrepancy functions are generalizations of other well-known proof certificates such as Contraction metrics and Incremental Lyapunov functions. A discrepancy function $\beta : \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}_{\geq 0}$ has two requirements:

(1) β upper bounds the distance between the trajectories,

$$\|\xi(\zeta_0, t) - \xi(\zeta_0', t)\| \leq \beta(\zeta_0, \zeta_0', t), \tag{12}$$

where, $\xi(\zeta_0, t)$ and $\xi(\zeta_0', t)$ represent any pair of trajectories with initial conditions ζ_0 and ζ_0', respectively.

(2) β converges to zero as the initial states of the trajectories converge.

$$\text{for any } t, \text{ as } \zeta_0 \rightarrow \zeta_0',\ \beta(.,.,t) \rightarrow 0. \tag{13}$$

The first requirement expresses β as a function of initial conditions of any two trajectories and the elapsed time, and it upper bounds the distance between the trajectories at any time so that every possible state of the system is represented in the reachable set. On the other hand, the second requirement is used to keep the over-approximation error low.

There are methods developed in the literature to compute β from differential equations [39]. However, in this study, we used a tool known as DryVR [2] that formulates the problem of finding the discrepancy function as a problem of learning linear separator to achieve high computational efficiency. The learning linear separator approach does not depend on the system's dynamics and uses few simulations to arrive at a discrepancy function with probabilistic correctness guarantees.

The discrepancy function adopted in DryVR is an exponential function that grows and shrinks with time, and has a general form:

$$\beta(u, v, t) = \|u - v\| K e^{\hat{\gamma} t}, \tag{14}$$

where, K and $\hat{\gamma}$ (we write $\hat{\gamma}$ to distinguish from γ which is the flight path angle) are constants that govern the behaviour of the exponential function and we learn them using the learning linear separator approach.

Considering equation (14) and the first requirement of a discrepancy function in equation (12):

$$\forall t \in [0, T].\ \|\xi(\zeta_0, t) - \xi(\zeta_0', t)\| \leq \|\zeta_0 - \zeta_0'\| K e^{\hat{\gamma} t} \tag{15}$$

Equation (15) can be rearranged by taking logs on both sides as:

$$\forall t \in [0, T].\ \ \ln \frac{\|\xi(\zeta_0, t) - \xi(\zeta_0', t)\|}{\|\zeta_0 - \zeta_0'\|} \leq \ln K + \hat{\gamma} t \tag{16}$$

The above inequality has a general structure of:

$$\forall (\mu, \nu) \in \Gamma.\ \ \mu \leq a\nu + b, \tag{17}$$

where, for $\Gamma \subseteq \mathbb{R} \times \mathbb{R}$, a pair (a, b) is a linear separator and (μ, ν) represents $\left(\ln \frac{\|\xi(\zeta_0,t) - \xi(\zeta_0',t)\|}{\|\zeta_0 - \zeta_0'\|}, t\right)$ in (16). Therefore, the learning task is identifying the (a, b) values from sampling points that makes the inequality in (17) a linear separator for the large portion of points in Γ. The sampling points are assumed to be drawn based on unknown distribution $\mathcal{D}$. The probabilistic algorithm provided in Algorithm 1 has been proposed in [2] to identify the appropriate values of (a, b). The separator discovered by the above algorithm has a correctness guarantee with high probability. The proof can be obtained in [2].

The procedure to over-approximate the reachable set of an aircraft is presented in Algorithm 2 and its subroutine in Algorithm 1. First, the aircraft dynamics, upper and lower bounds of the initial set, the number of traces (aircraft

Algorithm 1 Learning linear separator

```
1: Input: set of traces $\Gamma$
2: Output: linear separator $(a, b)$
3: Draw $k$ pairs $(\mu_1, \nu_1), ..., (\mu_k, \nu_k)$ from $\Gamma$ according to distribution $\mathcal{D}$;
4: Find $(a, b)$ by solving the linear program $\mu_i \leq a\nu_i + b$ for all $i = \{1, ..., k\}$.
5: return $(a, b)$
```

trajectory simulations), and time horizon will be given to the algorithm. The algorithm then generates trajectories by randomly choosing initial states within the given bound and over-approximates the reachable set for the given time horizon. Figures 1 and 2 show how a reachable set of an aircraft can be over-approximated by simulating several trajectories from the current state. Figure 1 is a projection of the reachable set of an aircraft on XY-plane, and figure 2 is the projection of the reachable set of an aircraft on a Z-plane.

Algorithm 2 Reachable set over-approximation

```
1: Input: upper and lower bound initial sets $(\zeta_o, \zeta'_o)$ , aircraft dynamics $\dot{\zeta}(t)$, number of traces $N$, time horizon $T$
2: Output: reachable set $\beta(\zeta_0, \zeta'_0, t)$
3: for $i \leftarrow 1$ to $N$ do
4:     $\zeta_0 \leftarrow$ randomly choose initial states from $(\zeta_o, \zeta'_o)$
5:     trace$[i] \leftarrow \dot{\zeta}(\zeta_0, T)$
6: end for
7: $\Gamma \leftarrow$ trace
8: $(a, b) \leftarrow$ LearningLinearSeparator($\Gamma$)
9: $(K, \hat{\gamma}) \leftarrow (a, b)$
10: $\beta(\zeta_0, \zeta'_0, t) \leftarrow \|\zeta_0 - \zeta'_0\| K e^{\hat{\gamma} t}$
11: return $\beta(\zeta_0, \zeta'_0, t)$
```

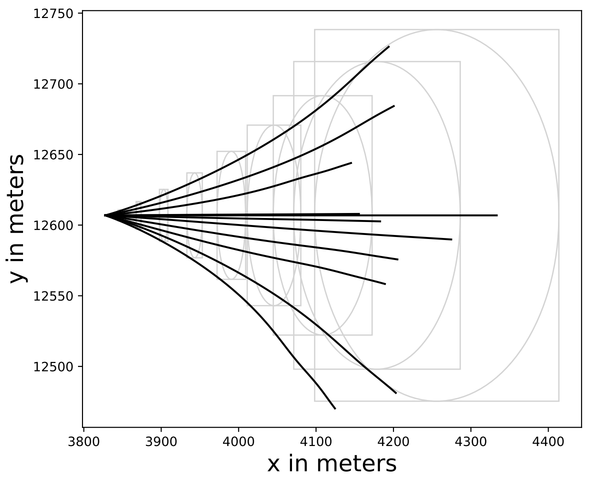

Fig. 1 **XY-plane projection of reach-tube**

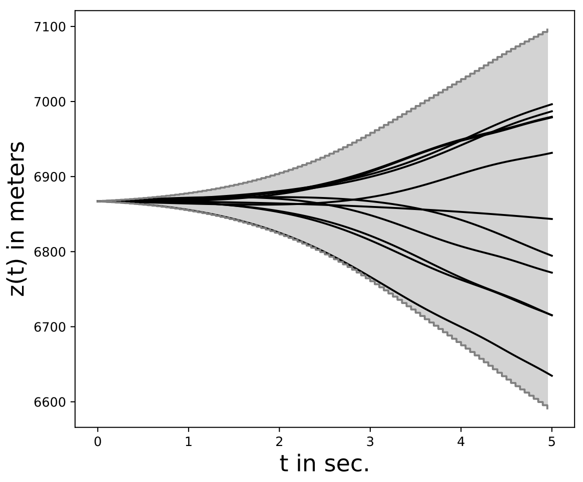

Fig. 2 **Z-plane projection of reach-tube**

D. The Proposed Trajectory Planning Framework

As shown in figure 3, the trajectory planning structure comprises two main modules: The Reachability Analysis module and the Trajectory Planner.

Trajectory Planner: The proposed framework works in a decentralized manner, where each aircraft will be responsible for choosing a control action that satisfies the reach-avoid property defined below. To achieve this, it first forward projects the future states of an aircraft using the dynamics of the aircraft and the control actions provided in the action space. Then, it computes the positive and negative rewards for the projected states and picks the control action that maximizes the total reward.

Reachability Analysis: While building the negative rewards, the framework considers the reachable sets of nearby intruder aircraft and the terrain around the aircraft. The algorithms discussed in section III.C will be utilized to compute the reachable sets,

Reach-avoid property: For an aircraft starting from an initial state $\zeta(0)$, we say the reach-avoid property is satisfied if and only if its trajectory $\zeta(t)$, (1) never enters into an unsafe set $\mathcal{S}_u$, and (2) reaches a goal set $\mathcal{S}_g$ within a finite time horizon T. These two conditions can be expressed mathematically as:

$$(\forall t \in 0 \leq t \leq T,\ \xi(\zeta(0), t) \cap \mathcal{S}_u = \emptyset) \bigwedge (\exists\, t\ 0 \leq t \leq T,\ \xi(\zeta(0), t) \cap \mathcal{S}_g \neq \emptyset) \tag{18}$$

In the above equation, the unsafe set $\mathcal{S}_u$ is composed of the reachable sets of nearby intruders and the terrain.

Theorem 1: Consider aircraft i has access to other nearby intruder aircraft's dynamics and current states. In addition, consider aircraft i has the information about the terrain of the environment. Then, aircraft i can choose a control action from the action space $\mathcal{A}$ for its next state that is guaranteed to satisfy the reach-avoid property given in equation 18.

Proof: Consider the reach-avoid property is not satisfied for aircraft i. Such assumption entails that either the aircraft has entered an unsafe state $\mathcal{S}_u$, or it is not progressing to its goal state $\mathcal{S}_g$. However, because the reachable sets of nearby aircraft and the terrain information are accessible to aircraft, it can choose a control action that enables the aircraft to avoid entering the reachable sets of nearby aircraft. In addition, since the MDP-based trajectory planner generates a reward that motivates the aircraft to move to its destination, aircraft i will always progress towards its destination. Hence, Theorem 1 is true by contradiction. ■

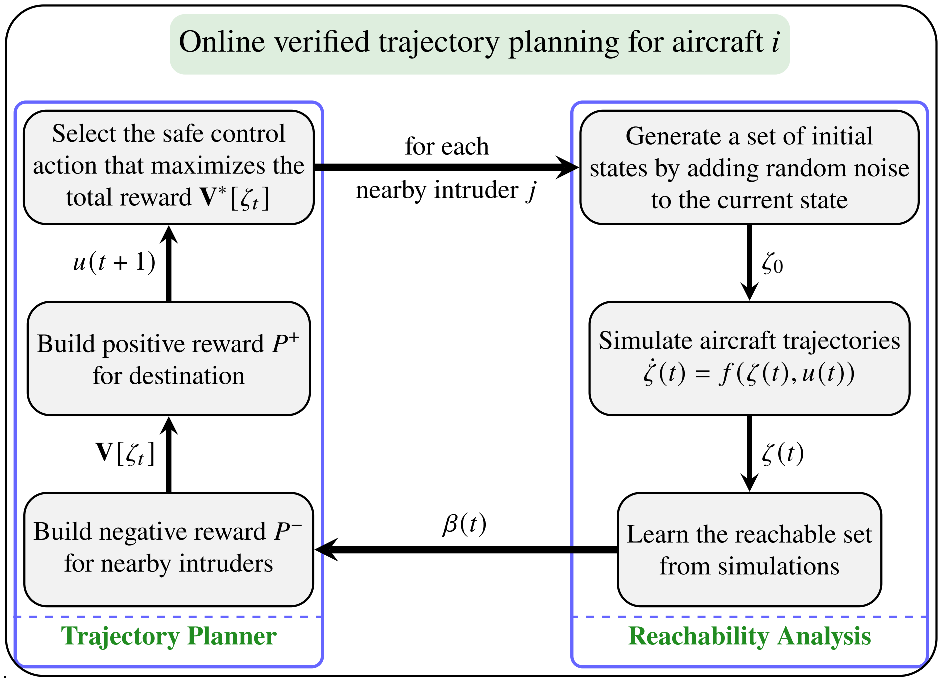
Online verified trajectory planning for aircraft i
Select the safe control action that maximizes the total reward $\mathbf{V}^*[\zeta_t]$
for each nearby intruder j
Generate a set of initial states by adding random noise to the current state
$u(t+1)$
ζ_0
Build positive reward P^+ for destination
Simulate aircraft trajectories $\dot{\zeta}(t) = f(\zeta(t), u(t))$
$\mathbf{V}[\zeta_t]$
$\zeta(t)$
Build negative reward P^- for nearby intruders
$\beta(t)$
Learn the reachable set from simulations
Trajectory Planner
Reachability Analysis

Fig. 3 **Schematic diagram representation of the proposed method**

Algorithm 3 Online Verified Trajectory Planning Framework

```
1:  Procedure OnlineVerifiedTrajectoryPlanner(AircraftState, WorldState)
2:  $\mathbf{S}_0 \leftarrow$ randomly initialize aircraft states
3:  repeat
4:    for each aircraft $i$ do
5:      $\zeta_t \leftarrow$ current state of the ownship
6:      $Z_1 \leftarrow$ forward project all the possible states of the ownship for the next 1 second
7:      $Z_{10} \leftarrow$ forward project all the possible states of the ownship for the next 10 seconds
8:      $P^+ \leftarrow$ build positive reward for destination
9:      $P^* \leftarrow$ build negative reward for terrain
10:     $\zeta_j \leftarrow$ identify nearby aircraft
11:     $\beta(\zeta_0, \zeta_0', t) \leftarrow \texttt{ReachableSetOver-approximation}(\zeta_j)$
12:     $P^- \leftarrow$ build negative reward using $\beta(\zeta_0, \zeta_0', t)$
13:
14:     for $\zeta_j \in Z_{10}$ do
15:       for $p_i \in P^+$ do
16:         $d_p \leftarrow \|\zeta_j - \texttt{location}(p_i)\|_2$
17:         $r_p \leftarrow \texttt{reward}(p_i)$
18:         $\gamma_p \leftarrow \texttt{discount}(p_i)$
19:         $V^+_{p_i} \leftarrow |r_p| \cdot \gamma_p^{d_p}$
20:       end for
21:       $\mathbf{V}^+_{\texttt{max}} \leftarrow \max_{p_i} \mathbf{V}^+_{p_i}$
22:
23:       for $n_i \in \{P^-, P^*\}$ do
24:         $d_n \leftarrow \|\zeta_j - \texttt{location}(n_i)\|_2$
25:         $\rho_n \leftarrow d_n < \texttt{radius}(n_i)$
26:         $r_n \leftarrow \texttt{reward}(n_i)$
27:         $\gamma_n \leftarrow \texttt{discount}(n_i)$
28:         $V^-_{n_i} \leftarrow \texttt{int}(\rho_n) \cdot |r_n| \cdot \gamma_n^{d_n}$
29:       end for
30:       $\mathbf{V}^-_{\texttt{max}} \leftarrow \max_{n_i} \mathbf{V}^-_{n_i}$
31:
32:       if $\texttt{altitude}(\zeta_t) < \texttt{penalty\_altitude}$ then
33:         $\mathrm{V}_{\texttt{terrain}} \leftarrow 1000 - \texttt{altitude}(\zeta_t)$
34:       else
35:         $\mathrm{V}_{\texttt{terrain}} \leftarrow 0$
36:       end if
37:     end for
38:
39:     $\mathbf{V}^*[\zeta_i] \leftarrow V^+_{\texttt{max}} - V^-_{\texttt{max}} - V_{\texttt{terrain}}$
40:     // Identify the most valuable action and the corresponding next state
41:     $i_{\texttt{max}} \leftarrow \underset{\zeta}{\operatorname{argmax}}(\mathbf{V}^*)$
42:     $\zeta_{t+1} \leftarrow Z_1[i_{\texttt{max}}]$
43:     $\mathrm{S}_{t+1}[\mathrm{i}] \leftarrow \zeta_{t+1}$
44:   end for
45: until each aircraft reaches its final destination
46: end procedure
```

IV. Simulation and Results

In this section, the performance of the proposed method is discussed. We used our previously developed MDP-based trajectory planning framework [3] as a baseline to compare the performance of the current approach. The previous approach utilizes a simple trajectory projection scheme to predict the future states of the nearby intruder aircraft and manually place negative rewards at those predicted future states. In contrast, we used a reachability analysis module to compute the possible future states of intruder aircraft and place the negative rewards everywhere inside the reachable set. By doing so, we significantly decrease the number of near-mid air collisions (NMACs).

A snapshot of the simulation environment we used to compare the performance of the two trajectory planning frameworks is shown in Fig 4. We defined a geographical bounding box that covers a volume of 25km by 25km by 25km with a terrain taken from NASA shuttle radar topography mission (SRTM) radar data [40] in the Lake Tahoe, California area. The aircraft will be randomly assigned to take off from their origin vertiports to their destination vertiports. In the experiment, the aircraft are colored according to the goal state (vertiport) they are flying to, and the color-coded trajectories represent the flight history of each aircraft. Thus, the environment can accommodate a configurable number of vertiports and aircraft that use the proposed trajectory planning framework in a distributed manner.

Since the objective of this paper is to develop a safe and scalable UAM trajectory planning framework, the two criteria we used to evaluate the performance of the proposed algorithm are mean computational time and near mid-air collision (NMAC). Mean computation time, which is the average time in each step that takes for the algorithm to compute the safe trajectory, demonstrates the computational efficiency of the method. On the other hand, NMAC, which is defined as the number of aircraft coming within 500 meters of another aircraft during flight, gives the ability of the algorithm to guide the aircraft while avoiding collisions.

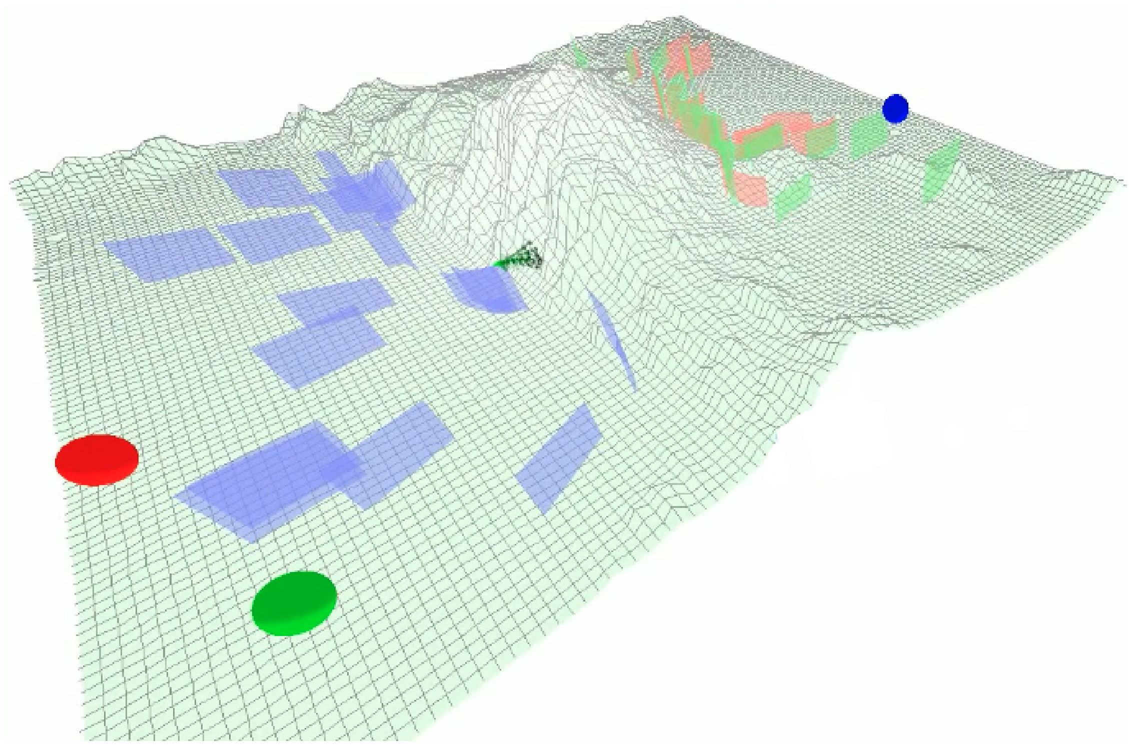

Fig. 4 **Snapshot of the simulation environment. The environment simulates three color-coded teams of aircraft (red, green, blue). Each team is colored based on the vertiport the aircraft in the team are navigating towards. The aircraft trajectories for the past few steps are rendered as a box-like shape. The aircraft are at the front tip of the trajectories and move to their assigned vertiports, rendered as spheres.**

We present experimental results on a different number of aircraft assigned to fly to one of the three goal states. All experiments were carried out on a 3.20 GHZ Intel Xeon (R) CPU with 125.4 GB RAM. We repeated each experiment 10 times for each aircraft number using randomly generated initial locations for the agents*. We report the average

*Here is a sample video of the simulation for 60 aircraft: https://youtu.be/FAf5OPqlGqQ runtime and near mid-air collision for each aircraft number.

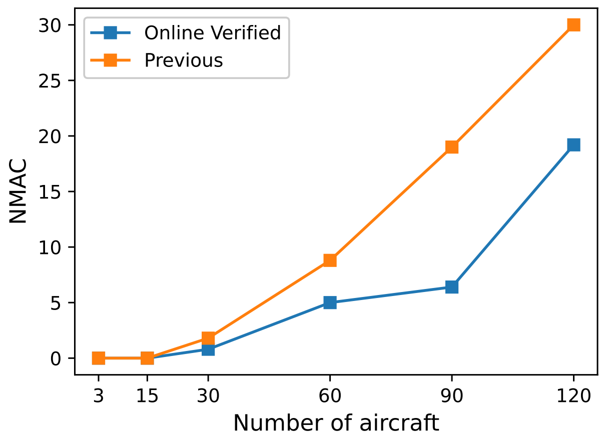

Fig. 5 **NMAC performance comparison**

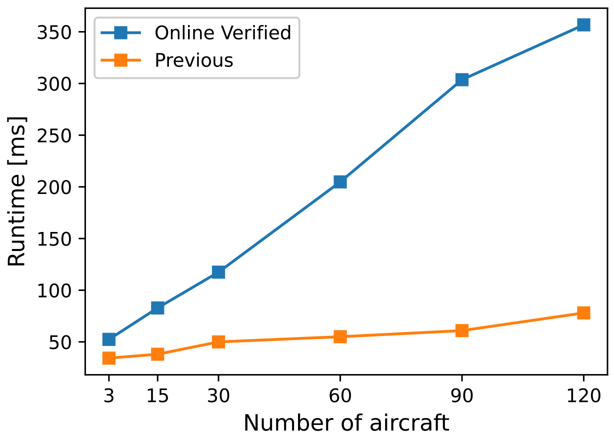

Fig. 6 **Runtime performance comparison**

The experiments show that the current online verified approach outperforms the previous technique in terms of NMAC, as shown in Fig 5, which implies the current approach improves the safe operation of the aircraft. Furthermore, although we used a reachability analysis-based formal verification scheme, we observed few NMACs in the environment as the number of aircraft increased. This happens mainly due to the fact that an MDP formulation inherently converts the hard constraints (collisions) into benign conditions (negative rewards). Hence, when the environment gets congested, some aircraft may choose to violate the safety constraint momentarily. In addition, Fig 6 presents the mean computational time to execute the algorithm per aircraft. From the figure, one can observe that even though the computational time for the current approach has increased significantly, it is still growing in a linear order ($O(n)$ performance) with the increased number of aircraft.

V. Conclusion

In this work, we address the problem of developing a safe and scalable UAM trajectory planning framework. The framework operates in a decentralized manner, where each aircraft exploits the information about its surroundings to plan its trajectory. An MDP-based trajectory planner and a data-driven reachability analysis module have been used to synthesize each aircraft's trajectory online formally. The framework's efficacy has been tested in several scenarios, and the results reveal the trajectory planner's computational efficiency and safe operation. Future works focus on improving the MDP-based trajectory planner to handle hard constraints such as collisions well. In addition, we plan to improve the quality of the generated trajectories, such as minimizing flight trip time.

Acknowledgments

This project is supported by the NASA Grant 80NSSC21M0087 under the NASA System-Wide Safety (SWS) program.

References

[1] Patterson, M. D., Antcliff, K. R., and Kohlman, L. W., "A proposed approach to studying urban air mobility missions including an initial exploration of mission requirements," , 2018. URL `https://ntrs.nasa.gov/citations/20190000991`, accessed on 03.29.2022.

[2] Fan, C., Qi, B., Mitra, S., and Viswanathan, M., "DryVR: Data-Driven Verification and Compositional Reasoning for Automotive Systems," *Computer Aided Verification*, edited by R. Majumdar and V. Kunčak, Springer International Publishing, Cham, 2017, pp. 441–461.

[3] Bertram, J., and Wei, P., "Distributed computational guidance for high-density urban air mobility with cooperative and non-cooperative collision avoidance," *AIAA Scitech 2020 Forum*, 2020, p. 1371.

[4] Gipson, L., "NASA Embraces Urban Air Mobility, Calls for Market Study," , Nov 2017. URL `https://www.nasa.gov/aero/nasa-embraces-urban-air-mobility/`, accessed on 06.08.2021.

[5] "Uber Elevate | The Future of Urban Air Transport," , 2017. URL `https://www.uber.com/us/en/elevate/`, accessed on 06.08.2021.

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

[7] "Urban Air Mobility," , 2018. URL `http://publicaffairs.airbus.com/default/public-affairs/int/en/our-topics/Urban-Air-Mobility.html`, accessed on 08.13.2018.

[8] Airbus, "Future of urban mobility," , Apr 2019. URL `https://www.airbus.com/newsroom/news/en/2016/12/My-Kind-Of-Flyover.html`, accessed on 06.08.2021.

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

[10] Bilimoria, K., Sheth, K., Lee, H., and Grabbe, S., "Performance evaluation of airborne separation assurance for free flight," *18th Applied Aerodynamics Conference*, 2000, p. 4269.

[11] Valenti Clari, M., Ruigrok, R., and Hoekstra, J., "Cost-benefit study of free flight with airborne separation assurance," *AIAA Guidance, Navigation, and Control Conference and Exhibit*, 2001, p. 4361.

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

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

[14] Harman, W. H., "TCAS- A system for preventing midair collisions," *The Lincoln Laboratory Journal*, Vol. 2, No. 3, 1989, pp. 437–457.

[15] 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.

[16] 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.

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

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

[19] Richards, A., and How, J. P., "Aircraft trajectory planning with collision avoidance using mixed integer linear programming," *Proceedings of the 2002 American Control Conference (IEEE Cat. No. CH37301)*, Vol. 3, IEEE, 2002, pp. 1936–1941.

[20] 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.

[21] Vela, A., Solak, S., Singhose, W., and Clarke, J.-P., "A mixed integer program for flight-level assignment and speed control for conflict resolution," *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*, IEEE, 2009, pp. 5219–5226.

[22] Mellinger, D., Kushleyev, A., and Kumar, V., "Mixed-integer quadratic program trajectory generation for heterogeneous quadrotor teams," *2012 IEEE international conference on robotics and automation*, IEEE, 2012, pp. 477–483.

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

[24] 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.

[25] 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.

[26] Delahaye, D., Peyronne, C., Mongeau, M., and Puechmorel, S., "Aircraft conflict resolution by genetic algorithm and B-spline approximation," *Proceedings of ENRI Int. Workshop on ATM/CNS. Tokyo, Japan*, Vol. 4, 2010, pp. 71–77.

[27] 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," *2011 IEEE International Conference on Robotics and Automation*, IEEE, 2011, pp. 4429–4434.

[28] 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.

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

[30] Wang, Y., Huang, C., Wang, Z., Wang, Z., and Zhu, Q., "Verification in the Loop: Correct-by-Construction Control Learning with Reach-avoid Guarantees," *arXiv preprint arXiv:2106.03245*, 2021.

[31] Bansal, S., Chen, M., Herbert, S., and Tomlin, C. J., "Hamilton-Jacobi reachability: A brief overview and recent advances," *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, IEEE, 2017, pp. 2242–2253.

[32] Althoff, M., "An introduction to CORA 2015," *Proc. of the Workshop on Applied Verification for Continuous and Hybrid Systems*, 2015.

[33] Frehse, G., Le Guernic, C., Donzé, A., Cotton, S., Ray, R., Lebeltel, O., Ripado, R., Girard, A., Dang, T., and Maler, O., "SpaceEx: Scalable verification of hybrid systems," *International Conference on Computer Aided Verification*, Springer, 2011, pp. 379–395.

[34] Chen, X., Ábrahám, E., and Sankaranarayanan, S., "Flow*: An analyzer for non-linear hybrid systems," *International Conference on Computer Aided Verification*, Springer, 2013, pp. 258–263.

[35] Hsieh, C., Sibai, H., Taylor, H., Ni, Y., and Mitra, S., "SkyTrakx: A Toolkit for Simulation and Verification of Unmanned Air-Traffic Management Systems," *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, IEEE, 2021, pp. 372–379.

[36] Park, H., Lee, B.-Y., Tahk, M.-J., and Yoo, D.-W., "Differential game based air combat maneuver generation using scoring function matrix," *International Journal of Aeronautical and Space Sciences*, Vol. 17, No. 2, 2016, pp. 204–213.

[37] McLain, T., Beard, R. W., and Owen, M., "Implementing dubins airplane paths on fixed-wing uavs," 2014.

[38] Duggirala, P. S., Mitra, S., and Viswanathan, M., "Verification of annotated models from executions," *2013 Proceedings of the International Conference on Embedded Software (EMSOFT)*, IEEE, 2013, pp. 1–10.

[39] Fan, C., and Mitra, S., "Bounded verification with on-the-fly discrepancy computation," *International Symposium on Automated Technology for Verification and Analysis*, Springer, 2015, pp. 446–463.

[40] Rodriguez, E., Morris, C., Belz, J., Chapin, E., Martin, J., Daffer, W., and Hensley, S., "An assessment of the SRTM topographic products," , 2005. URL `https://www2.jpl.nasa.gov/srtm/SRTM_D31639.pdf`, accessed on 03.29.2022.