Safe and Scalable Real-Time Trajectory Planning Framework for Urban Air Mobility[*]

Abenezer G. Taye[†]
George Washington University, Washington, DC, 20052, USA

Roberto Valenti[‡]
MathWorks, Natick, MA 01760, USA

Akshay Rajhans[§]
MathWorks, Natick, MA 01760, USA

Anastasia Mavrommati[¶]
MathWorks, Natick, MA 01760, USA

Pieter J. Mosterman[‖]
MathWorks, Natick, MA 01760, USA

Peng Wei[**]
George Washington University, Washington, DC, 20052, USA

This paper presents a real-time trajectory planning framework for Urban Air Mobility (UAM) that is both safe and scalable. The proposed framework employs a decentralized, free-flight concept of operation in which each aircraft independently performs separation assurance and conflict resolution, generating safe trajectories by accounting for the future states of nearby aircraft. The framework consists of two main components: a data-driven reachability analysis tool and an efficient Markov Decision Process (MDP) based decision maker. The reachability analysis over-approximates the reachable set of each aircraft through a discrepancy function learned online from simulated trajectories. The decision maker, on the other hand, uses a 6-degrees-of-freedom guidance model of fixed-wing aircraft to ensure collision-free trajectory planning. Additionally, the proposed framework incorporates reward shaping and action shielding techniques to enhance safety performance. The proposed framework was evaluated through simulation experiments involving up to 32 aircraft in a generic city-scale area with a 15 km radius, with performance measured by the number of Near Mid Air Collisions (NMAC) and computational time. The results demonstrate the planner's ability to generate safe trajectories

[*]This work was presented in part at the AIAA AVIATION 2022 Forum, Taye, A. G., Bertram, J., Fan, C., Wei, P. (2022). "Reachability based online safety verification for high-density urban air mobility trajectory planning," AIAA Paper 2022-3542, Chicago, IL, June 2022.

[†]Graduate Student, Department of Mechanical & Aerospace Engineering, abenezertaye@gwu.edu, AIAA Student Member.

[‡]Research Scientist, MathWorks Advanced Research and Technology Office, rvalenti@mathworks.com

[§]Research Scientist, MathWorks Advanced Research and Technology Office, arajhans@mathworks.com

[¶]Research Scientist, MathWorks Advanced Research and Technology Office, amavromm@mathworks.com

[‖]Research Scientist, MathWorks Advanced Research and Technology Office

[**]Associate Professor, Department of Mechanical & Aerospace Engineering, pwei@gwu.edu, AIAA Senior Member.

for the aircraft in polynomial time, showing its scalability. Moreover, the action shielding and reward shaping strategies show up to a 78.71% and 85.14% reduction in NMAC compared to the baseline planner, respectively.

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. This technology aims to provide a safe, efficient, and accessible on-demand air transportation system [1], offering an alternative to traditional ground-based transportation methods. Furthermore, as the technology advances, it will connect urban centers to outlying areas, expanding the reach of metropolitan regions.

UAM operation is a multi-agent system, with several aircraft simultaneously operating in a city-scale domain. This necessitates a UAM trajectory planning framework to generate and manage trajectories for each aircraft in a computationally efficient manner. Moreover, since UAM is a safety-critical system, safety must be guaranteed at all times of operation. These two problems — developing a scalable trajectory planner and safety verification of autonomous systems — are fundamentally challenging in and of themselves and are often addressed independently in the literature. However, because UAM is a safety-critical multi-agent system, a UAM trajectory planning framework needs to address safety and scalability in tandem.

The literature on multi-agent trajectory planning algorithms is extensive and can broadly be 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 [2], nonlinear programming [3, 4], mixed-integer linear programming [5–8], mixed-integer quadratic programming [9], sequential convex programming [10, 11], second-order cone programming [12], evolutionary techniques [13–15], and reinforcement learning [16]. One common thread among centralized approaches is that to pursue a global optimum, they must consider each aircraft and obstacle in space, leading to scalability issues with a large number 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 more robust than centralized approaches [17] 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 [18], we proposed a Markov Decision Process (MDP) based decentralized UAM trajectory planning algorithm that is highly scalable. The algorithm operates in a free-flight manner. This study is extended by incorporating an online safety verification module that enables the trajectory planner to generate safe trajectories.

The task of guaranteeing the safe operation of autonomous systems is often called 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 involve generating 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. As a result, formal methods, which can capture all possible behaviors of the system, have gained significant research attention in recent years.

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 [19]. However, this approach is computationally inefficient and often fails to give the necessary guarantees [20]. 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 [21, 22]. 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 synthesize each aircraft's trajectory online formally. An efficient reachability analysis module that explores 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 [23], CORA [24], SpaceEx [25], and Flow* [26]. Although these approaches provide formal soundness guarantees, they are computationally expensive. Hence, they can not be used online in the presence of many aircraft. In this study, to over-approximate the reachable set of an aircraft, we implemented a sensitivity analysis-based approach from DryVR [27]. DryVR has been demonstrated to be highly scalable and recently implemented in [28] to generate a safe operation volume for unmanned aircraft systems (UAS) traffic management. The reachability analysis module, then, is integrated with our previously developed MDP-based trajectory planner [18] to guide the motion of multiple UAM vehicles between vertiports.

We presented a preliminary version of this paper at the AIAA Aviation 2022 conference [29]. The main differences between the AIAA conference paper and this paper are summarized as follows: 1) We have enhanced the performance of the baseline trajectory planner proposed in the conference paper by replacing some of the important components. These include adopting an accurate aircraft guidance model, developing a new reward function, and constructing an action space that results in better performance. 2) We have reformulated the trajectory planning framework to incorporate an action shielding strategy that prevents the aircraft from choosing control actions that could compromise safety, thereby enhancing the safety properties of the planner presented in the conference paper. 3) To further improve the safety features of the trajectory planner proposed in the conference paper, we have developed a reward-shaping scheme and integrated it into the reward function of the MDP formulation. 4) We have developed a new UAM scenario that enables us to test the collision avoidance capabilities of the system under adverse conditions, wherein each agent in the environment is required to avoid all other agents.

This paper is organized as follows: In Section I, we review previous works related to the problem at hand. Section II outlines the problem, and section III presents the mathematical formulation of the two main components, the MDP and reachability analysis. We also provide an overview of the proposed trajectory planning framework, including the role of each component in the trajectory planning procedure. In Section IV, we discuss the implemented UAM scenario and present the results for the nominal trajectory planner (without any safety reinforcement) and the two other approaches proposed to improve the safety of the trajectory planner, namely, action shielding and reward shaping. Finally, in Section V, we provide the conclusion of this work.

II. Problem Formulation

A. Problem Description

This study aims to address the problem of developing a UAM trajectory planning framework that is computationally efficient and guarantees the safe navigation of UAM aircraft. As shown in Figure 1, the two main components of the proposed framework are the 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 [18][27]. Adopting such formulations makes the developed UAM trajectory planning framework computationally efficient.

In this paper, we assumed a noise-free sensing scheme in the environment where each aircraft in the system can access the real-time position information of nearby intruder aircraft. The actual implementation of such a sensing scheme for UAM operations can be possible using Automatic Dependent Surveillance-Broadcast (ADS-B) as recommended by the FAA [30]. Regarding the communication scheme, the framework is assumed to be implemented in a distributed manner with onboard computation and planning. Hence, there is no need for any communication between aircraft since every aircraft will sense and make its own decision.

B. Aircraft Dynamics

The aircraft model used in this paper is based on a 6-DOF kinematic guidance model formulation provided in [31]. The original guidance model assumes the presence of wind; hence, it contains wind-related parameters. However, since

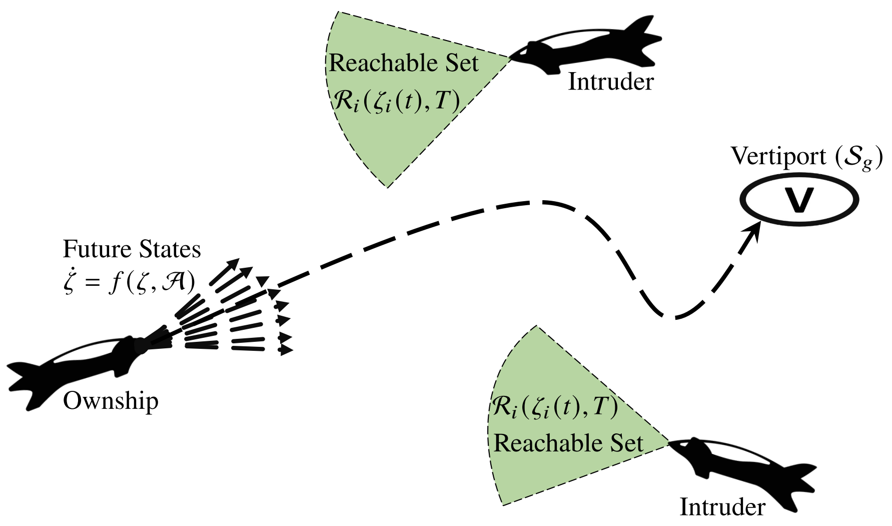

Reachable Set
$\mathcal{R}_i(\zeta_i(t), T)$
Intruder
Vertiport ($\mathcal{S}_g$)
V
Future States
$\dot{\zeta} = f(\zeta, \mathcal{A})$
Ownship
$\mathcal{R}_i(\zeta_i(t), T)$
Reachable Set
Intruder

Fig. 1 Working principle of the proposed trajectory planner.

we are not considering the presence of wind in this study, we used a simplified model given in Equation 1, where $\dot{x}, \dot{y}, \dot{z}$ are north, east, and down velocities of the aircraft with respect to the inertial reference frame. γ is the flight-path angle, and V is the speed of the aircraft. ϕ, χ, and ψ represent the roll, course, and heading angles, respectively. b_γ, b_V, and b_ϕ are positive constants that depend on the implementation of the autopilot and the state estimation schemes. The superscript $*^c$ as in γ^c, V^c, and ϕ^c denotes the commanded values given to the autopilot.

$$\begin{cases} \dot{x} = & V \cos\psi \cos\gamma \\ \dot{y} = & V \sin\psi \cos\gamma \\ \dot{z} = & V \sin\gamma \\ \dot{\chi} = & \frac{g}{V} \tan\phi \cos(\chi - \psi) \\ \dot{\gamma} = & b_\gamma(\gamma^c - \gamma) \\ \dot{V} = & b_V(V^c - V) \\ \dot{\phi} = & b_\phi(\phi^c - \phi) \end{cases} \tag{1}$$

III. Methodology

A. Markov Decision Process Formulation

In this paper, we formulate the aircraft trajectory planning problem as a Markov decision process (MDP), where the state transitions will be governed by the vehicle dynamics described in Section II.B. MDPs are formulated as the tuple (s_t, a_t, r_t, t) where $s_t \in \mathbf{S}$ is the state at a given time t within the state space $\mathbf{S}$. $a_t \in \mathcal{A}$ denotes the action taken by the agent at time t from the action set $\mathcal{A}$. r_t is the reward received by the agent as a result of taking action a_t from s_t and arriving at s_{t+1}, and $\mathbf{T}(s_t, a, s_{t+1})$ 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 map each state $s \in \mathbf{S}$ to action $a \in \mathcal{A}$. From a given policy $\pi \in \mathbf{\Pi}$, a value function $\mathbf{V}^{\pi}(S)$ can be computed that represents the expected return that will be obtained within the environment by following the policy π. The solution of an MDP is the optimal policy π^*, which defines the optimal action $a^* \in \mathcal{A}$ that can be taken from each state $s \in \mathbf{S}$ to maximize the expected return. From this optimal policy π^*, the optimal value function $\mathbf{V}^*(s)$ can be computed, which describes the maximum expected value obtained from each state $s \in \mathbf{S}$. Furthermore, from the optimal value function $\mathbf{V}^*(s)$, the optimal policy π^* can also easily be recovered.

1. State Space

The environment is a continuous state space placed on a spherical volume of $15km$ radius, based on [1]. Given the dynamics of an aircraft:

$$\dot{\boldsymbol{\zeta}}(t) = f(\boldsymbol{\zeta}(t), \mathbf{u}(t)), \tag{2}$$

where, $f : \mathbb{R}^n \times \mathbb{R} \rightarrow \mathbb{R}^n$ is a continuous function. $\boldsymbol{\zeta}$ denotes the aircraft states, which includes the x, y, z positions, heading angle ψ, the flight path angle γ, the course χ, the roll angle ϕ, and the speed V. The trajectory of an aircraft $\boldsymbol{\xi} : \mathbb{R}^n \times \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}^n$ is the solution to the differential equation (2). It represents how the state variables of the aircraft evolve through time. For a given initial set $\boldsymbol{\zeta}_0 \in \mathbb{R}^n$, the state of the system at time t is $\boldsymbol{\xi}(\boldsymbol{\zeta}_0, t) = \boldsymbol{\zeta}(t)$. The control input $\mathbf{u}(t)$ 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 ($\mathbf{s}_o$) contains all the states of an aircraft ($\boldsymbol{\zeta}$) and the states of every other aircraft denoted as f_j, $\forall j \in \mathbf{J}$, where $\mathbf{J}$ represents a set containing all aircraft in the system except the ownship. Thus, we can define $\mathbf{s}_o$ as $\mathbf{s}_o = [\boldsymbol{\zeta}, f_1, \ldots, f_j]$.

2. Action Space

The action space of the MDP is composed of the individual action spaces of the three inputs: the commanded flight-path angle ($\boldsymbol{\gamma}^c$), the commanded roll angle $\boldsymbol{\phi}^c$, and the commanded airspeed ($\mathbf{V}^c$). The action space of V^c is composed of 10 linearly spaced discrete values between $25m/s$ and $70m/s$. The minimum speed of $25m/s$ is chosen based on the stall speed performance of the aircraft [32]. On the other hand, the action spaces of $\boldsymbol{\gamma}^c$ and $\boldsymbol{\phi}^c$ are discrete sets of actions 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 gets further away from its 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 $\boldsymbol{\gamma}^c$ and $\boldsymbol{\phi}^c$ are logarithmically spaced within a range of 15 input values.

The logarithmically spaced input set in degree is computed as follows:

$$\boldsymbol{\gamma}^c = [-19.99, -16.24, -12.66, -9.26, -6.02, -2.94, -0.01, 0, 0.01, 2.94, 6.02, 9.26, 12.66, 16.24, 19.99] \quad (3)$$

$$\boldsymbol{\phi}^c = [-19.99, -16.24, -12.66, -9.26, -6.02, -2.94, -0.01, 0, 0.01, 2.94, 6.02, 9.26, 12.66, 16.24, 19.99] \quad (4)$$

Finally, the joint action space becomes:

$$\mathcal{A} = \{\boldsymbol{\gamma}^c, \boldsymbol{\phi}^c, \mathbf{V}^c\}. \quad (5)$$

3. Reward Function

The reward function is the primary mechanism we use to control the behavior of an MDP agent's behavior. 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 utilized both positive and negative rewards, as depicted in Table 1. Positive rewards are collected by each aircraft as they make progress toward their destination. Conversely, negative rewards are used to penalize the aircraft when they enter the reachable set of intruder aircraft. The use of positive and negative rewards enables the aircraft to fly to their assigned vertiport while avoiding possible collisions with other nearby aircraft.

Table 1 Reward function for each aircraft

Reward source	Reward magnitude	Location	Decay factor	Description
Intruder aircraft	$-(100t + 500)$	Inside reachable-set of intruder	0.97	Collision avoidance
Destination	200	Manually placed	0.999	Vertiport attraction

4. Value Function

Once the MDP is formulated as a tuple of (s_t, a_t, r_t), we need to solve the formulated MDP to arrive at the optimal solution. The specific value function structure is adopted from [33], where the authors present a highly scalable solution approach to deterministic terminating MDPs. The methods and proofs for computing the value function are detailed in the full paper. However, the key insight proposed in the approach is that it is possible to rewrite the value function of deterministic terminating MDPs solely as a function of the discount factor, reward magnitude, and distance metric between the current state of the agent and the reward source.

$$\mathbf{V}(s) = \kappa^{+\delta(s,s_i^+)} \cdot r_i^+ + \kappa^{-\delta(s,s_i^-)} \cdot r_i^-, \quad (6)$$

where κ, r_i, and $\delta(s, s_i)$ represent the discounting factor, reward magnitude, and distance to the reward source s_i from the current state s, respectively. Notations κ^+/κ^-, r_i^+/r_i^-, and s_i^+/s_i^- differentiate between positive and negative rewards.

B. Reachability Analysis

To avoid collision between aircraft, the MDP formulation of the trajectory planner presented in the previous subsections requires identifying the possible future states of the intruder aircraft. In this paper, we adopted a reachability analysis-based method to compute all the possible future states of each nearby aircraft. In this study, the concept of discrepancy function is adopted from [27] 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 primarily used to measure the convergence or divergence nature of trajectories formally [34]. Hence, it generates the over-approximation of the reachable set by providing the upper and lower bounds of the trajectories. In [34], 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,

$$\|\boldsymbol{\xi}(\zeta_0, t) - \boldsymbol{\xi}(\zeta_0', t)\| \leq \beta(\zeta_0, \zeta_0', t), \tag{7}$$

where, $\boldsymbol{\xi}(\zeta_0, t)$ and $\boldsymbol{\xi}(\zeta_0', t)$ 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.

$$\text{for any } t, \text{ as } \zeta_0 \rightarrow \zeta_0',\ \beta(\cdot, \cdot, t) \rightarrow 0. \tag{8}$$

The first requirement expresses β as a function of the initial conditions of any two trajectories and the elapsed time. 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 [35]. However, in this study, we use a tool known as DryVR [27] 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 a 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:

$$\beta(u, v, t) = \|u - v\| K e^{\hat{\gamma} t}, \tag{9}$$

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

Considering Equation (9) and the first requirement of a discrepancy function in Equation (7):

$$\|\boldsymbol{\xi}(\boldsymbol{\zeta}_0, t) - \boldsymbol{\xi}(\boldsymbol{\zeta}'_0, t)\| \leq \|\boldsymbol{\zeta}_0 - \boldsymbol{\zeta}'_0\| K e^{\hat{\gamma} t}, \quad \forall t \in [0, T]. \tag{10}$$

Equation (10) can be rearranged by taking logarithms of both sides as:

$$\ln\left(\frac{\left\|\boldsymbol{\xi}(\boldsymbol{\zeta}_0, t) - \boldsymbol{\xi}(\boldsymbol{\zeta}'_0, t)\right\|}{\left\|\boldsymbol{\zeta}_0 - \boldsymbol{\zeta}'_0\right\|}\right) \leq \ln K + \hat{\gamma} t, \quad \forall t \in [0, T]. \tag{11}$$

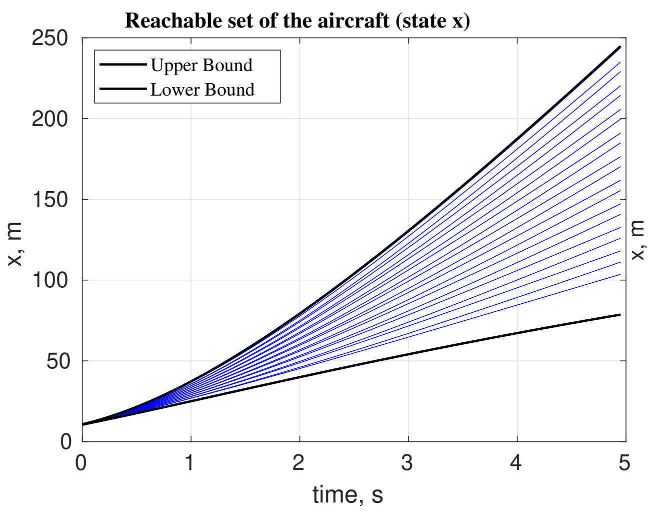
The above inequality has a general structure of:

$$\mu \leq a\nu + b, \quad \forall (\mu, \nu) \in \boldsymbol{\Gamma}. \tag{12}$$

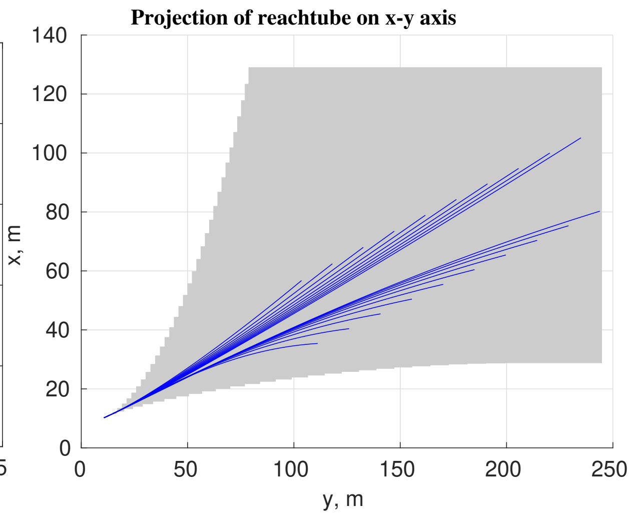
where for $\boldsymbol{\Gamma} \subseteq \mathbb{R} \times \mathbb{R}$, a pair (a, b) is a linear separator and (μ, ν) represents $\left(\ln \frac{\|\boldsymbol{\xi}(\boldsymbol{\zeta}_0,t)-\boldsymbol{\xi}(\boldsymbol{\zeta}'_0,t)\|}{\|\boldsymbol{\zeta}_0-\boldsymbol{\zeta}'_0\|}, t\right)$ in (11). Therefore, the learning task is identifying the (a, b) values from sampling points that make the inequality in (12) a linear separator for the large portion of points in $\boldsymbol{\Gamma}$. 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 [27] 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 [27]. To minimize the conservative nature of the discrepancy function, we adopt a piece-wise exponential discrepancy function of the form $\beta(\boldsymbol{\zeta}_0, \boldsymbol{\zeta}'_0, t) = \|\boldsymbol{\zeta}_0 - \boldsymbol{\zeta}'_0\| K e^{\sum_{j=1}^{i-1} \gamma_j (t_j - t_{j-1}) + \gamma_i (t - t_{i-1})}$ from [27]. This enables us to divide the time window for the reachable set into several smaller segments and find discrepancy parameters for each segment, resulting in less conservative reachable bounds.

Algorithm 1: Reachability Analysis

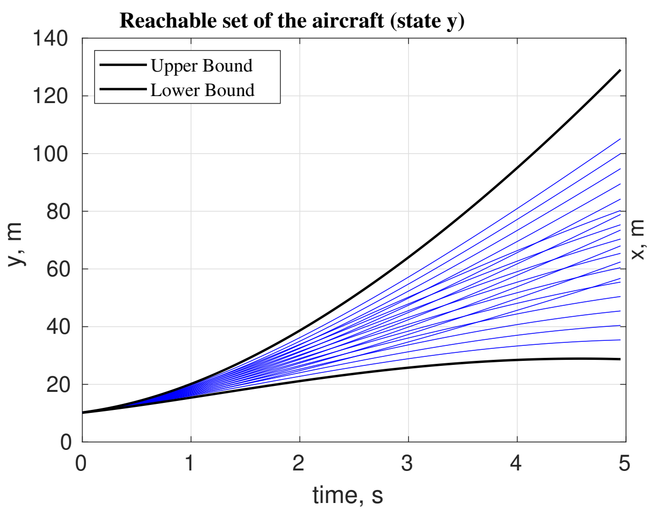
```
Procedure ReachabilityAnalysis():
    Input : Action set $\mathcal{A}$, aircraft dynamics $\dot{\boldsymbol{\zeta}}(t)$, initial state $\boldsymbol{\zeta}_0$, time horizon $T$
    Output : Reachable set $\mathcal{R}_i(\boldsymbol{\zeta}_i(t), T)$
1   $\boldsymbol{\Gamma}(t) \leftarrow f(\boldsymbol{\zeta}(t), \mathcal{A})$; /* randomly sample from $\mathcal{A}$ and generate a set of trajectories */
2   $\|\boldsymbol{\zeta}_0 - \boldsymbol{\zeta}'_0\| \leftarrow \mathcal{D}_{\mathbb{C}}(\boldsymbol{\Gamma}(t_0))$ ;   /* compute distance between initial states */
3   $\|\boldsymbol{\xi}(\boldsymbol{\zeta}_0, t) - \boldsymbol{\xi}(\boldsymbol{\zeta}'_0, t)\| \leftarrow \mathcal{D}_{\mathbb{C}}(\boldsymbol{\Gamma}(t))$;   /* compute distance between trajectories */
4   $\mu(t) \leftarrow \ln \frac{\|\boldsymbol{\xi}(\boldsymbol{\zeta}_0,t)-\boldsymbol{\xi}(\boldsymbol{\zeta}'_0,t)\|}{\|\boldsymbol{\zeta}_0-\boldsymbol{\zeta}'_0\|}$;   /* compute sensitivity parameters */
5   $\nu(t) \leftarrow t$
6   $\sum_i^n \mu_i = \nu_i a_i + b_i \leftarrow$ covhull$(\mu(t), \nu(t))$;   /* compute discrepancy parameters */
7   $a_i \leftarrow \frac{\Delta\mu_i}{\Delta t}$, $b_i \leftarrow \mu_i - \nu_i a_i$
8   $\beta(\boldsymbol{\zeta}_0, \boldsymbol{\zeta}'_0, t) \leftarrow \|\boldsymbol{\zeta}_0 - \boldsymbol{\zeta}'_0\| K e^{\sum_{j=1}^{i-1} \gamma_j (t_j - t_{j-1}) + \gamma_i (t - t_{i-1})}$;   /* compute the piece-wise exponential
      discrepancy function */
9   $\mathcal{R}_i(\boldsymbol{\zeta}_i(t), T) \leftarrow \beta(\boldsymbol{\zeta}_0, \boldsymbol{\zeta}'_0, t)$
```



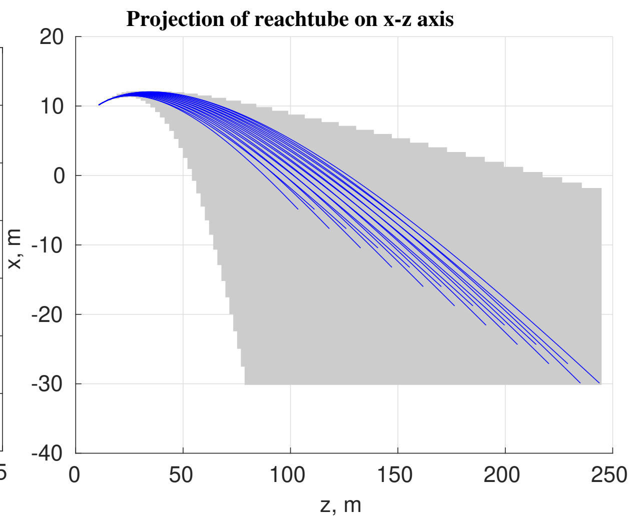
(a) **Reachable set of the aircraft (state x)**



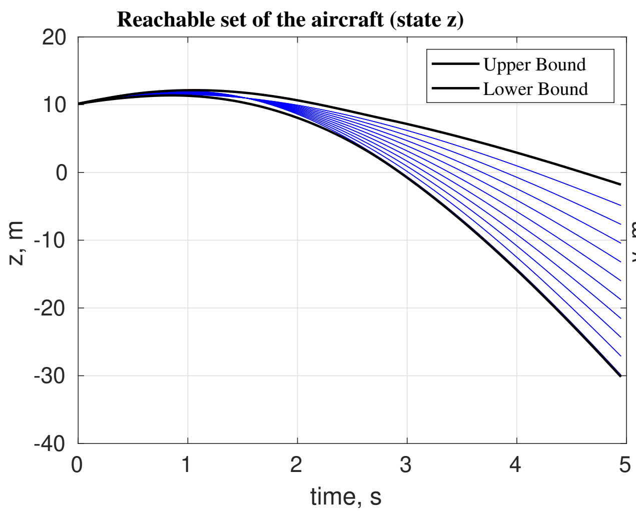
(b) **Projection of reach-tube on x-y axis**



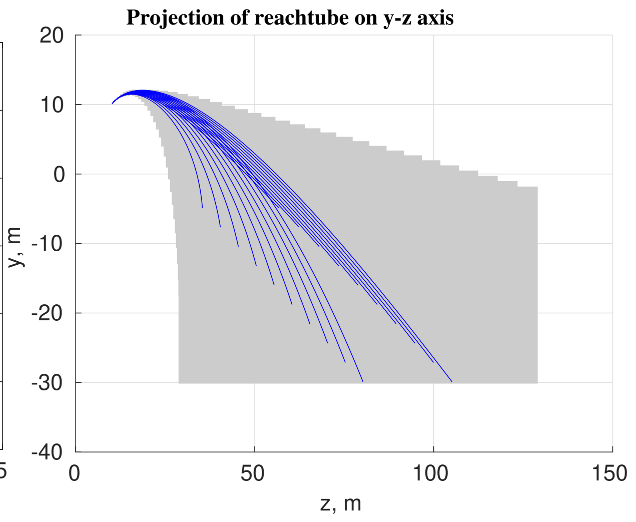
(c) **Reachable set of the aircraft (state y)**



(d) **Projection of reach-tube on x-z axis**



(e) **Reachable set of the aircraft (state z)**



(f) **Projection of reach-tube on y-z axis**

Fig. 2 Generated reachable set using Algorithm 1.

The procedure to over-approximate the reachable set of an aircraft is outlined in Algorithm 1. The inputs to the algorithm include the aircraft dynamics, the action set $\mathcal{A}$, the initial states of the aircraft ζ_0, and the time horizon T. The algorithm then generates trajectories by randomly choosing from the set of control actions. It then computes the maximum pair-wise distance between the initial states and each trajectory using Chebyshev distance and gets the sensitivity parameters for each time step ($\mu(t)$ and $\nu(t)$). The convex hull of these parameters is then determined, and the values a and b are obtained, which represent the discrepancy function K and $\hat{\gamma}$.

Figures 2a to 2f show how a reachable set of aircraft can be over-approximated by simulating several trajectories from the current state. Figures 2a, 2c, and 2e depict the reachable sets of x, y, and z states of the aircraft, respectively. Figure 2b, 2d, and 2f show the projections of the reach-tube of an aircraft on different planes.

C. The Proposed Trajectory Planning Framework

The detailed working procedure of the trajectory planning is provided in Algorithm 2. Here, we highlight the two main modules: Reachability Analysis and 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.B will be utilized to compute the reachable sets.

The overall operational procedure of the proposed trajectory planner is such that the framework first assigns initial and goal states for each aircraft in the system. Subsequently, for each aircraft, it identifies the positive and negative reward sources as discussed in III.A.3. After the reward sources are identified, it forward projects the future states of the aircraft using the action sets and computes the values of each future state using the value function as given in Equation 6. The best action yielding the maximum total reward is then selected, and the states of the aircraft are updated using the chosen control action. This process is repeated iteratively for each aircraft until each aircraft reaches its designated destination vertiport.

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 follows:

$$(\forall t \in 0 \leq t \leq T,\ \xi(\zeta(0), t) \cap \mathcal{S}_u = \emptyset) \bigwedge (\exists\, t\ 0 \leq t \leq T,\ \xi(\zeta(0), t) \cap \mathcal{S}_g \neq \emptyset) \tag{13}$$

Algorithm 2: Online Verified Trajectory Planning Framework

```
Procedure TrajectoryPlanner(world state):
1    $S_0 \leftarrow$ randomly initialize aircraft states
2    repeat
3        for each aircraft $i$ do
4            $\zeta_t \leftarrow$ current state of the ownship
5            $\Gamma(t) \leftarrow f(\zeta(t), \mathcal{A})$ ;   /* project future states of the ownship using the action set */
6            $P^+ \leftarrow$ vertiport location ;   /* build positive reward for destination */
7            $\zeta_j \leftarrow$ identify nearby aircraft
8            $\mathcal{R}_i(\zeta_i(t), T) \leftarrow$ Reachability Analysis($\zeta_j$) ;   /* compute the reach set using Algorithm 1 */
9            $P^- \leftarrow \mathcal{R}_i(\zeta_i(t), T)$ ;   /* build negative reward */
10           for $\zeta \in \Gamma$ do
11               $d_p \leftarrow \|\zeta_j - \texttt{location}(P^+)\|_2$
12               $r_p \leftarrow \texttt{reward}(P^+)$
13               $\gamma_p \leftarrow \texttt{discount}(P^+)$
14               $V_p^+ \leftarrow |r_p| \cdot \gamma_p^{d_p}$ ;   /* compute positive values for each future state */
15               $V_{\texttt{max}}^+ \leftarrow \max_p V_p^+$
16           for $n_i \in P^-$ do
17               $d_n \leftarrow \|\zeta_j - \texttt{location}(n_i)\|_2$
18               $\rho_n \leftarrow d_n < \texttt{radius}(n_i)$
19               $r_n \leftarrow \texttt{reward}(n_i)$
20               $\gamma_n \leftarrow \texttt{discount}(n_i)$
21               $V_{n_i}^- \leftarrow \texttt{int}(\rho_n) \cdot |r_n| \cdot \gamma_n^{d_n}$ ;   /* compute negative values for each future state */
22           if altitude($\zeta_t$) < penalty altitude then
23               $V_{\texttt{terrain}} \leftarrow 1000 - \texttt{altitude}(\zeta_t)$
24           else
                 $V_{\texttt{terrain}} \leftarrow 0$
25           $V^*[\zeta_i] \leftarrow V_{\texttt{max}}^+ - V_{\texttt{max}}^- - V_{\texttt{terrain}}$ ; /* compute total values for each future state */
26           $i_{\texttt{max}} \leftarrow \operatorname{argmax}_{\zeta}(V^*)$
27           $\zeta_{t+1} \leftarrow Z_1[i_{\texttt{max}}]$
28           $S_{t+1}[i] \leftarrow \zeta_{t+1}$
     until each aircraft reaches its final destination;
```

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 information about the environment's terrain. 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 13.

Assumption 1: We assumed a noise-free sensing scheme that enables our ownship aircraft to access the current states of nearby intruder aircraft's current position.

Assumption 2: We assumed a homogeneous fleet of aircraft where each aircraft in the system has the same dynamics. Therefore, the ownship has access to the dynamics and the action set of each nearby aircraft.

Proof: Consider the reach-avoid property is not satisfied for aircraft i. Such an 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, 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. ■

IV. Results and Discussion

In this section, the performance of the proposed method is discussed. 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 the number of Near Mid Air Collisions (NMAC). Mean computation time, which is the time taken in each step by the algorithm to compute the safe trajectory for a single aircraft, demonstrates the computational efficiency of the method. On the other hand, NMAC, defined as a loss of 152 meters of horizontal and 30 meters of vertical separation [36], is used to evaluate the ability of the algorithm to guide the aircraft and avoid collisions.

A. Scenario Description

A snapshot of the simulation environment we used to evaluate the performance of the trajectory planning framework is shown in Fig 3. The simulation defined a geographical bounding box that encompasses a volume of $15km$ radius. The aircraft are assigned to take off from their origin vertiports and fly to destination vertiports located on the opposite side of their origin. The environment is configurable to accommodate a variable number of vertiports and aircraft, which utilize the proposed trajectory planning framework in a distributed manner.

In reference to the designed scenario, it is important to note that, as depicted in Figure 3, all aircraft are scheduled to travel through a central location in the environment. The environment simulates the operation of 32 aircraft, each rendered as a red circle. The black boxes represent the vertiports where the aircraft take off and navigate towards. The black lines represent the aircraft trajectories for the next ten steps. The blue lines indicate the paths traveled by aircraft. This scenario, although unlikely to occur in a typical UAM setting, serves as a means to evaluate the detect-and-avoid (DAA) capabilities of the system under adverse conditions where strategic deconfliction fails.

We present experimental results on a different number of aircraft assigned to fly to their designated goal states. The algorithm utilized in these experiments has been implemented using MATLAB. Additionally, a video demonstration showcasing the results of the algorithm for 8[*], 16[†], and 32[‡] aircraft can be viewed on YouTube.

All experiments were conducted on a 3.20 GHZ Intel Xeon (R) CPU with 125.4 GB RAM. Each experiment was

[*]https://youtu.be/Ynfl1Js3RCU
[†]https://youtu.be/bVnMpdz8ANU
[‡]https://youtu.be/49wUrg0ZIko

repeated 25 times for each aircraft number, with randomly generated initial locations for the aircraft. The computational time and NMACs for each aircraft number are reported.

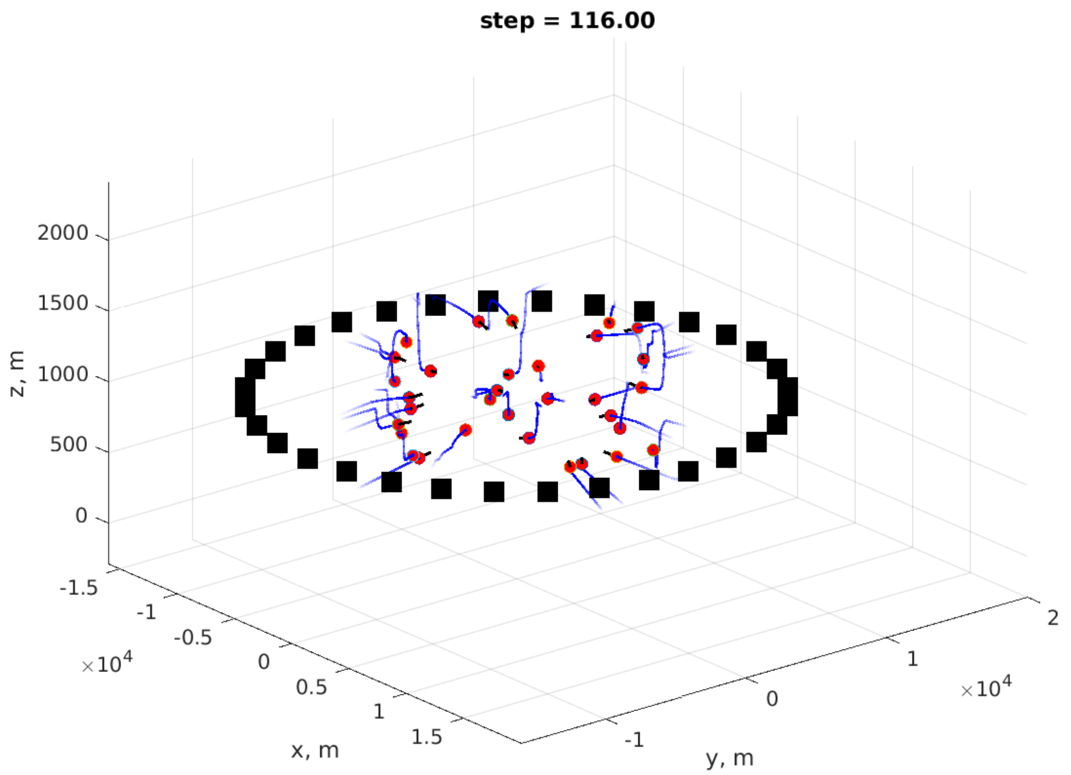


Fig. 3 Snapshot of the simulation environment.

Table 2 NMAC performance

Aircraft	mean	std
2	0	0
4	0	0
8	0	0
16	1.36	5.12
32	9.96	10.91

Table 3 Computation time performance

Aircraft	mean (sec)	std (sec)	throughput (sec)
2	0.03	0.03	31.05
4	0.05	0.06	99.46
8	0.11	0.14	489.16
16	0.17	0.21	1815.25
32	0.27	0.29	6448.43

The experimental results demonstrate the effectiveness of the proposed trajectory planner in guiding the motion of each aircraft from its initial position to its assigned vertiport. Tables 2 and 3 present the trajectory planner's NMAC and computational time performances of the trajectory planner. As shown in Table 3, the mean computational time of the framework increases as the number of aircraft in the system increases, but it grows in a polynomial order with the increased number of aircraft, indicating the scalability of the approach. Table 3 also presents the throughput performance of the algorithm, defined as the total time taken to guide all aircraft involved in the system to their assigned vertiport successfully.

On the other hand, despite utilizing a formal verification scheme based on reachability analysis, as indicated in Table 2, there were instances of NMACs observed in the environment as the number of aircraft increased. This is primarily due to the fact that the MDP formulation converts hard constraints, such as collisions, into benign conditions represented by negative rewards. As a result, in congested environments, there may be instances of momentary violations of safety constraints. In the subsequent subsections, we will discuss the methods employed to address this issue.

B. Action Shielding

One potential solution to the challenge of enforcing hard constraints on an MDP agent is through the implementation of action shielding [37]. This approach filters the agent's actions through a mechanism that prevents actions leading to unsafe states, as illustrated in Figure 4. The value of states is used to filter out actions that result in unsafe states. Specifically, if the value of a state, resulting from a certain action, is negative — which in our context indicates that the aircraft is within the reachable set of the intruder — the shield will remove the action from the set of valid actions. However, in instances where all control actions lead to unsafe states, this technique results in a deadlock since all control actions in the action set are filtered out by the shield. To circumvent this scenario, we propose an alternative control action for a short time horizon. In these rare cases, the action sets result in aggressive maneuvers but ensure safe operation by compromising comfort. Moreover, we have included the number of times (average) these action sets have been activated in the action-shielding strategy in Table 4 to provide additional insight into the effect of these control actions on overall aircraft operation. As such, the new control action set (in degree) to be implemented during a deadlock will be:

$$\boldsymbol{\gamma}^c = [-70, -60, -50, -40, -30, 30, 40, 50, 60, 70] \tag{14}$$

$$\boldsymbol{\phi}^c = [-70, -60, -50, -40, -30, 30, 40, 50, 60, 70] \tag{15}$$

Tables 4 and 5 present the performance of the trajectory planner with the added enhancement of action shielding with regard to the number of NMACs and computational time, respectively. As shown in Table 4, it is evident that the addition of action shielding has resulted in a significant improvement in the safety performance of the trajectory planner. However, as demonstrated in Table 5, the change in the computational time is minimal.

C. Reward Shaping

Many existing techniques in the literature address the issue of undesirable behavior exhibited by MDP agents through the use of reward engineering or reward shaping. Reward shaping refers to the process of modifying the reward received by the agent to elicit desired behavior, as outlined in [38]. In other words, instead of using the traditional MDP

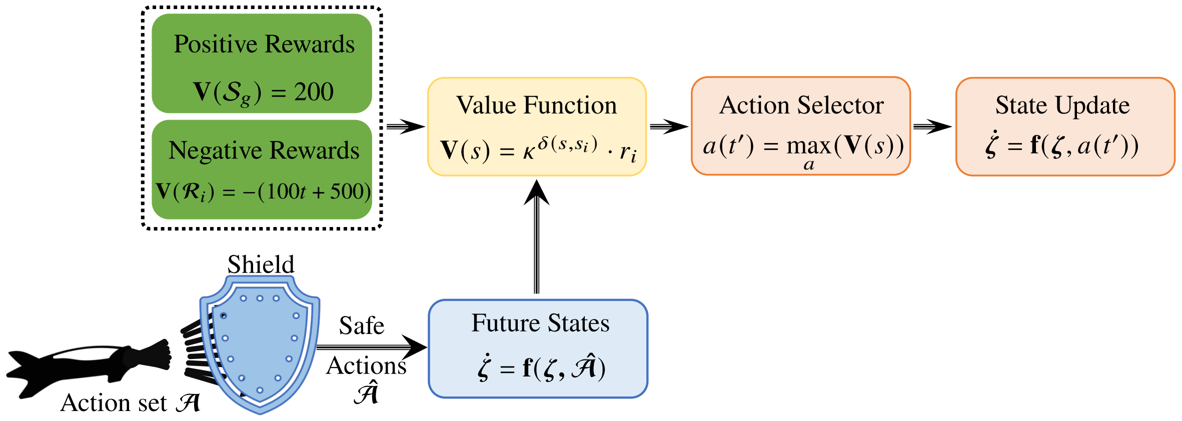

Positive Rewards
$\mathbf{V}(\mathcal{S}_g) = 200$
Negative Rewards
$\mathbf{V}(\mathcal{R}_i) = -(100t + 500)$
Value Function
$\mathbf{V}(s) = \kappa^{\delta(s,s_i)} \cdot r_i$
Action Selector
$a(t') = \max_a(\mathbf{V}(s))$
State Update
$\dot{\zeta} = \mathbf{f}(\zeta, a(t'))$
Shield
Safe Actions $\hat{\mathcal{A}}$
Future States
$\dot{\zeta} = \mathbf{f}(\zeta, \hat{\mathcal{A}})$
Action set $\mathcal{A}$

Fig. 4 The implemented action shielding procedure.

Table 4 NMAC performance

Aircraft	mean	std	# of deadlock
2	0	0	0
4	0	0	0
8	0	0	0
16	1.40	2.63	0.01
32	2.61	4.76	0.13

Table 5 Computation time performance

Aircraft	mean (sec)	std (sec)	throughput (sec)
2	0.03	0.02	30.00
4	0.05	0.06	92.55
8	0.10	0.12	395.10
16	0.17	0.19	1594.73
32	0.23	0.27	5623.16

$\mathbf{M} = (\mathbf{S}, \mathbf{A}, \mathbf{T}, \kappa, R)$, we use a transformed MDP $\mathbf{M}' = (\mathbf{S}, \mathbf{A}, \mathbf{T}, \kappa, R')$, where $R' = R + F$ is the reward function in the transformed MDP, and $F : \mathbf{S} \times \mathbf{A} \times \mathbf{S} \rightarrow R$ is a bounded real-valued function known as the reward-shaping function. The specific reward shaping function employed in this study is a difference of potentials $F(s, a, s') = \Phi(s') - \Phi(s)$, where Φ is the value function over states [39].

$$F(s, a, s') = \kappa(\mathbf{V}^*(s')) - \mathbf{V}^*(s), \tag{16}$$

where, κ is the discount factor and $\mathbf{V}^*(s')$ and $\mathbf{V}^*(s)$ are the values of the current and future states.

Tables 6 and 7 present the performance of the trajectory planner with the enhancement of reward shaping in terms of the number of NMAC and computational time, respectively. From Table 7, we can see that the reward shaping technique has led to a superior improvement in safety performance when compared to the action shielding technique. However, the impact on computational time is negligible.

A performance comparison of the three proposed methods is shown in Figure 5. The results, as depicted in Figure 5a, indicate that the baseline trajectory planner, which does not utilize any reinforcement techniques, exhibits poor safety performance. In contrast, the trajectory planner utilizing reward shaping demonstrates the best performance. While the implementation of action shielding improves the performance of the baseline trajectory planner, it still falls

Table 6 NMAC performance

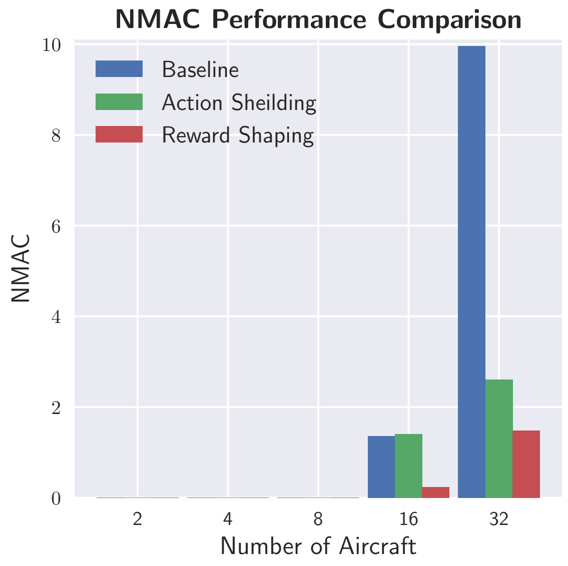
Aircraft	mean	std
2	0	0
4	0	0
8	0	0
16	0.24	1.20
32	1.48	3.63

Table 7 Computation time performance

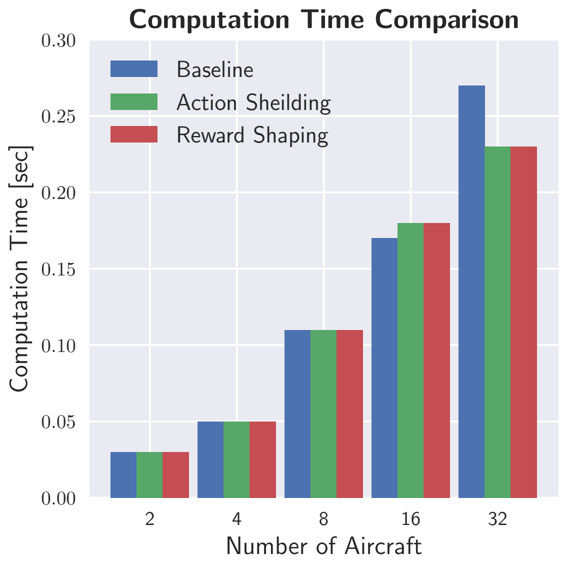
Aircraft	mean (sec)	std (sec)	throughput (sec)
2	0.03	0.02	29.68
4	0.05	0.06	101.37
8	0.11	0.13	463.09
16	0.18	0.23	1968.56
32	0.23	0.27	6311.54

short in comparison to the trajectory planner utilizing reward shaping. Figure 5b illustrates the computational time performance comparison of the proposed methods, where it is evident that the differences in performance are minimal.

Based on the observed results, the advantages of incorporating action-shielding and reward-shaping strategies into the baseline trajectory planner include improved safety and computational time. However, a disadvantage of the reward-shaping strategy is the challenge associated with designing an appropriate potential function to shape the original MDP formulation to achieve the desired behavior. Additionally, the disadvantages of the action-shielding strategy include the deadlock situation it causes and the aggressive maneuvers required to extricate the aircraft from this deadlock situation.



(a) NMAC performance comparison



(b) Computation time comparison

Fig. 5 Performance comparison of the three proposed methods.

V. Conclusion

This study proposes a safe and scalable trajectory planning framework for urban air mobility (UAM) systems. The proposed framework operates in a decentralized manner, allowing each aircraft to independently plan its trajectory based on information about its surrounding environment. The framework employs a Markov Decision Process (MDP)-based trajectory planner and a data-driven reachability analysis module to synthesize each aircraft's trajectory in real-time. To enhance safety performance, techniques such as reward shaping and action shielding have been explored to be included in the overall framework. The effectiveness of the framework has been evaluated through simulations involving up to 32 aircraft in UAM scenarios, and the results demonstrate the computational efficiency and safe operation of the trajectory planner.

Acknowledgments

This project is partially supported by 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," *Annual Forum and Technology Display*, NASA NF1676L-28586, May 2018.

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

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

[4] 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. https://doi.org/10.2514/3.20934.

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

[6] 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. https://doi.org/10.1109/ACC.2002.1023918.

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

[8] 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. https://doi.org/10.1109/CDC.2009.5400520.

[9] 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. https://doi.org/10.1109/ICRA.2012.6225009.

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

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

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

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

[14] 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," *IEEE International Conference on Robotics and Automation*, IEEE, 2011, pp. 4429–4434. https://doi.org/10.1109/ICRA.2011.5980246.

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

[16] Razzaghi, P., Tabrizian, A., Guo, W., Chen, S., Taye, A., Thompson, E., and Wei, P., "A Survey on Reinforcement Learning in Aviation Applications. arXiv 2022," *arXiv preprint arXiv:2211.02147*, June 2023.

[17] 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. https://doi.org/10.1109/ROBOT.2006.1642069.

[18] 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. https://doi.org/10.2514/6.2020-1371.

[19] Duggirala, P. S., Mitra, S., Viswanathan, M., and Potok, M., "C2E2: A verification tool for stateflow models," *Tools and Algorithms for the Construction and Analysis of Systems: 21st International Conference, TACAS 2015, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2015, London, UK, April 11-18, 2015, Proceedings 21*, Springer, 2015, pp. 68–82. https://doi.org/10.1007/978-3-662-46681-0_5.

[20] Wang, Y., Huang, C., Wang, Z., Wang, Z., and Zhu, Q., "Design-while-verify: correct-by-construction control learning with verification in the loop," *Proceedings of the 59th ACM/IEEE Design Automation Conference*, 2022, pp. 925–930. https://doi.org/10.1145/3489517.3530556.

[21] Fan, C., Qin, Z., Mathur, U., Ning, Q., Mitra, S., and Viswanathan, M., "Controller synthesis for linear system with reach-avoid specifications," *IEEE Transactions on Automatic Control*, Vol. 67, No. 4, 2021, pp. 1713–1727. https://doi.org/10.1109/TAC.2021.3069723.

[22] Fisac, J. F., Chen, M., Tomlin, C. J., and Sastry, S. S., "Reach-avoid problems with time-varying dynamics, targets and constraints," *Proceedings of the 18th International Conference on Hybrid Systems: Computation and Control*, 2015, pp. 11–20. https://doi.org/10.1145/2728606.2728612.

[23] 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. https://doi.org/10.1109/CDC.2017.8263977.

[24] Althoff, M., "An Introduction to CORA 2015," *ARCH14-15. 1st and 2nd International Workshop on Applied veRification for Continuous and Hybrid Systems*, EPiC Series in Computing, Vol. 34, EasyChair, 2015, pp. 120–151. https://doi.org/10.29007/zbkv.

[25] 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. https://doi.org/10.1007/978-3-642-22110-1.

[26] 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. https://doi.org/10.1007/978-3-642-39799-8.

[27] 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. https://doi.org/10.1007/978-3-319-63387-9_22.

[28] 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. https://doi.org/10.1109/ITSC48978.2021.9564492.

[29] Taye, A. G., Bertram, J., Fan, C., and Wei, P., "Reachability based Online Safety Verification for High-Density Urban Air Mobility Trajectory Planning," *AIAA AVIATION 2022 Forum*, 2022, p. 3542. https://doi.org/10.2514/6.2022-3542.

[30] Federal Aviation Administration, "ADS-B (Automatic Dependent Surveillance-Broadcast)," , 2024. URL https://www.faa.gov/about/office_org/headquarters_offices/avs/offices/afx/afs/afs400/afs410/ads-b, accessed: 2024-02-23.

[31] Beard, R. W., and McLain, T. W., *Small unmanned aircraft: Theory and practice*, Princeton University Press, 2012. https://doi.org/10.1515/9781400840601.

[32] Kimberlin, R. D., *Flight Testing of Fixed Wing Aircraft*, American Institute of Aeronautics Astronautics, 2003. https://doi.org/10.2514/4.861840.

[33] Bertram, J., Wei, P., and Zambreno, J., "A Fast Markov Decision Process-Based Algorithm for Collision Avoidance in Urban Air Mobility," *IEEE Transactions on Intelligent Transportation Systems*, Vol. 23, No. 9, 2022, pp. 15420–15433. https://doi.org/10.1109/TITS.2022.3140724.

[34] 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. https://doi.org/10.1109/EMSOFT.2013.6658604.

[35] 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. https://doi.org/10.1007/978-3-319-24953-7_32.

[36] Weinert, A., Alvarez, L., Owen, M., and Zintak, B., "A Quantitatively Derived NMAC Analog for Smaller Unmanned Aircraft Systems Based on Unmitigated Collision Risk," 2020. https://doi.org/10.20944/preprints202011.0503.v1.

[37] Könighofer, B., Lorber, F., Jansen, N., and Bloem, R., "Shield synthesis for reinforcement learning," *Leveraging Applications of Formal Methods, Verification and Validation: Verification Principles: 9th International Symposium on Leveraging Applications of Formal Methods, ISoLA 2020, Rhodes, Greece, October 20–30, 2020, Proceedings, Part I 9*, Springer, 2020, pp. 290–306. https://doi.org/10.1007/978-3-030-61362-4_16.

[38] Memarian, F., Goo, W., Lioutikov, R., Niekum, S., and Topcu, U., "Self-supervised online reward shaping in sparse-reward environments," *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2021, pp. 2369–2375. https://doi.org/10.1109/iros51168.2021.9636020.

[39] Ng, A. Y., Harada, D., and Russell, S., "Policy invariance under reward transformations: Theory and application to reward shaping," *International Conference on Machine Learning (ICML)*, Vol. 99, 1999, pp. 278–287.