Safe and Scalable Urban Air Mobility Trajectory Planner: User Manual

Abenezer Taye
Advanced Research
Technology Office (MARTO), MathWorks, 1 Lakeside Campus Drive, Natick, MA 01760

August 7, 2022

1 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. 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. The current implementation provides a trajectory planner that adopts a verify-while-design approach to formally synthesize each aircraft's trajectory online.

The purpose of this user manual is to explain the concept behind the framework (Section 2), explain what the implementation provides (Section 3), and demonstrate the implementation through detailed examples (Section 4).

2 Background

The trajectory planning framework works in a decentralized manner, where each aircraft will be responsible for choosing a control action that satisfies the reach-avoid property. To achieve this, the framework utilizes two main modules: The Reachability Analysis module and the Trajectory Planner.

Reachability Analysis: While building the negative rewards, the framework considers the reachable sets of nearby intruder aircraft. In this implementation, a data-driven reachability analysis tool known as DryVR has been used to generate the reachable sets of aircraft. DryVR uses the concept of discrepancy function to formulate the reachability analysis problem. A detailed explanation of the tool can be found in [1]. However, the general steps followed to implement the reachability analysis module are provided in Algorithm 1.

Algorithm 1 Reachability Analysis

Input: Action set $\mathcal{A}$, aircraft dynamics $\dot{\zeta}(t)$, initial state ζ_0, time horizon T
Output: Reachable set $\beta(\zeta_0, \zeta_0', t)$
$\Gamma(t) \leftarrow f(\zeta(t), \mathcal{A})$
$\|\zeta_0 - \zeta_0'\| \leftarrow \mathcal{D}_C(\Gamma(t_0))$
$\|\xi(\zeta_0, t) - \xi(\zeta_0', t)\| \leftarrow \mathcal{D}_C(\Gamma(t))$
$\mu(t) \leftarrow \ln \frac{\|\xi(\zeta_0,t) - \xi(\zeta_0',t)\|}{\|\zeta_0 - \zeta_0'\|}$
$\nu(t) \leftarrow t$
$\texttt{covhull}(\mu(t)) \leftarrow \sum_i^n \mu_i = \nu_i a_i + b_i$
$a_i \leftarrow \frac{\Delta\mu_i}{\Delta t}$, $b_i \leftarrow \mu_i - \nu_i a_i$
$\beta(\zeta_0, \zeta_0', t) \leftarrow \|\zeta_0 - \zeta_0'\| K e^{\sum_{j=1}^{i-1} \gamma_j (t_j - t_{j-1}) + \gamma_i (t - t_{i-1})}$
return $\beta(\zeta_0, \zeta_0', t)$

Trajectory Planner: is based on Markov Decision Process (MDP) and 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.

The working principle of the framework is summarized in Fig. 1.

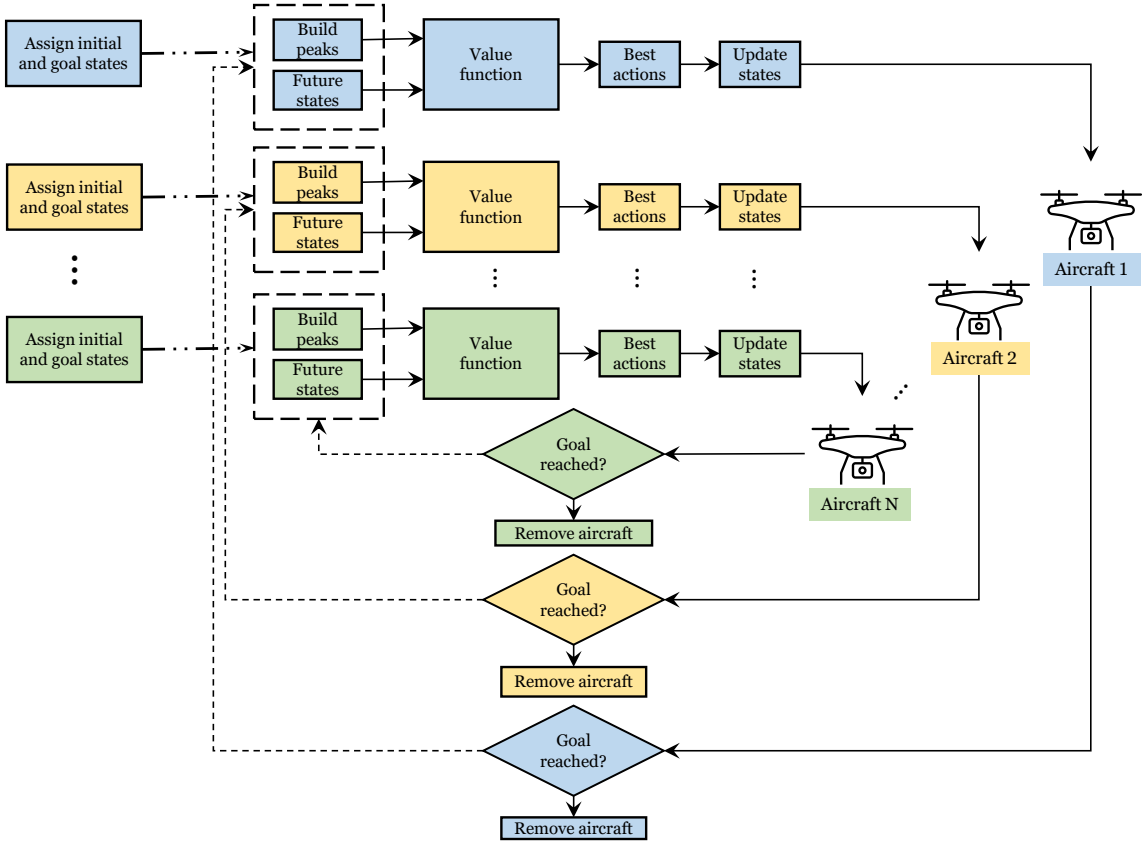

Figure 1: Schematic diagram representation of the trajectory planner

The aircraft model used in this implementation is based on a pseudo-6DOF formulation of fixedwing 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.

3 Implementation Contents

The implementation provides two main functionalities: an MDP based decision maker, and Reachability Analysis. We highlight the key functions that make up these two functionalities.

- `DryVR:` a class implementing the reachable set of each aircraft given the current states of the aircraft and the time horizon.
- `Ownship:` a class implementing each aircraft in the system, given the initial states, final states, and the control actions.
- `scenarioGenerator:` a function that generates the scenario in which the system operates, given the number of aircraft in the system. The current implementation generates circular scenario only, but it can easily be extended to generate other scenarios as well.
- `neighboringStates:` a vectorized function that takes the current state of the aircraft and action set, and returns all the future states of the aircraft.
- `buildPeaks:` a function that identifies both positive and negative reward sources for each aircraft, given the state of the aircraft. Note: the Reachability Analysis class will be utilized inside this function.
- `valueFunction:` a function that takes the current state of the aircraft and the reward sources, and returns the values of each future states of the aircraft.
- `terminalDetection:` a function that is used to monitor the progress of the execution.

3.1 Example

A detailed example of the implementation and how each module works has been given in the provided MATLAB Live Script. You can look at the "UAMTrajectoryPlannerExample.mlx" file in the main directory.

References