Efficient Unmanned Aerial Systems Navigation with Collision Avoidance in Dense Urban Environments

Josh Bertram, *Graduate Student Member, IEEE,* Joseph Zambreno, *Senior Member, IEEE* and Peng Wei, *Member, IEEE*

***Abstract*—Unmanned Aerial Systems (UAS) are an emerging type of airborne traffic under active research expected to carry cargo and passengers in the future over dense population centers. One challenge is identifying algorithms which can compactly represent and navigate the available airspace while avoiding conflict with buildings and other UAS. In this paper, we explore a decentralized method coupling a medial axis graph for global navigation through the city with local collision avoidance of buildings and other UAS to obtain collision-free efficient navigation through an urban environment. We study trade-offs of using Optimal Reciprocal Collision Avoidance (ORCA), Rapidly-exploring Random Trees (RRT and RRT*), and Fast Markov Decision Process (FastMDP) as the collision avoidance algorithms. We examine low-altitude UAS navigating through a portion of New York City dense with skyscrapers to study the effectiveness of the algorithms in a challenging environment. We show that ORCA, RRT, RRT*, and FastMDP all are fairly efficient for 2D problems, but as the problem becomes more realistic (3D, constrained motion, aware of other UAS), FastMDP provides the best overall performance among the four algorithms studied.**

***Index Terms*—Collision avoidance, Markov Decision Process.**

I. Introduction

Unmanned Aerial Systems (UAS) including package delivery drones, air-taxis, and similar aircraft are envisioned for Urban Air Mobility (UAM) [1]–[5] and Advanced Air Mobility (AAM) applications [6] where aircraft fly between specially designed vertiports which will allow vertical take-off and landing (VTOL) aircraft to safely ascend and land in urban environments. Many papers focus on aircraft navigating between these vertiports over large areas on the order of the size of a metropolis area or between cities. [7]–[14] In this paper, we examine navigation through a city where UAS may need to fly between and around building in a dense urban environment such as New York City. While the regulatory environment may not be ready to approve operators openly flying small UAS or larger air-taxis near or between buildings anytime soon, we imagine that trusted emergency services operators (e.g., police, fire, disaster response) in urban areas may be granted special privileges to perform operations in dense urban environments like New York City. Current events also suggest that in war-torn areas such as Ukraine, humanitarian aid workers might benefit from UAS which could navigate war-torn cities in an effort to identify survivors. For these reasons we still believe that this concept is worth exploring as we hope it may inform future related research.

In this paper, we break the problem into a "global" path planning problem through the city, and a "local" collision avoidance problem with buildings and aircraft. The contributions of this paper are:

1) development of a method to traverse a dense urban environment using the medial axis of a polygon-with-holes representation of the buildings,
2) efficient waypoint sequencing along the shortest paths constructed from the medial axis graph
3) extension of the FastMDP algorithm to polygon-shaped obstacles including formal proofs, and
4) comparison of performance of ORCA [15], RRT [16], RRT* [17], and FastMDP [18] for the local collision avoidance problem.

We direct the reader to [19] which is an excellent survey paper covering the problem background and approaches that have been used for path planning, collision avoidance, and trajectory optimization. Within the framework described by the survey paper, we note generally that the FastMDP algorithm can be understood as having a receding horizon similar to Model Predictive Control [20], [21] where the objective function being maximized is the value function for an underlying Markov Decision Process (MDP) [22], [23], and the resulting trajectory follows the optimal policy of the MDP. Note that MDP-based approaches do not explicitly separate the problem into path planning, collision avoidance, and trajectory optimization, but instead solves all three simultaneously. In this paper, FastMDP is used to solve collision avoidance and trajectory optimization as a combined problem leaving path planning through the city to a higher level algorithm.

This work differs from other related papers as follows. In [24], navigation through an indoor environment with static and moving obstacles is performed using an optimal control approach. To simplify the problem, obstacles were identified, clustered, and modelled as rectangular prisms which is a compatible assumption for an indoor building environment where many walls and obstacles are rectangular in nature. The algorithm optimally avoids obstacles, minimizing deviations from a human generated flight path provided as a set of waypoints with computations taking on the order of 80 to 120 ms. In this paper, we show a method to handle more complex obstacles modelled as convex polygons and generate the waypoints through a graph representing the city.

In [25], flight planning for UAS performing building inspections is examined. Flight paths are pre-computed from building

J. Bertram and J. Zambreno are with the Department of Electrical and Computer Engineering, Iowa State University, Ames, IA, 50011 USA e-mail: bertram1@iastate.edu, zambreno@iastate.edu.

P. Wei is with George Washington University. email: pwei@gwu.edu.

information data files which describe construction plans for a building and allow a set of waypoints to be constructed from points of interest input by the operator. A voxel based occupancy grid is constructed to identify a collision free space to operate within. The grid is converted to a graph, and A* search is used to compute a flight path where the result favors slow, careful movements around a building for safe and steady operation of the on-board inspection cameras.

In [26], a similar approach is used to this paper where a polygon-straight skeleton is found for an environment with rectangular obstacles, and the skeleton is then used as a seed for improving the performance of RRT. In this work, rather than using the skeleton to improve RRT performance in finding a solution for the global problem, we instead use the skeleton to provide waypoints for a simpler problem to the local path planner. Additionally, we exploit the radii of the skeleton for improved waypoint sequencing during flight, which may allow for improved path generation.

In [27], a UAV is provided a voxel map as an occupancy grid, and uses A* search at each time step to compute a reference path that will achieve high-level goals (e.g., reaching a set of goal locations) and the reference path is provided to an underlying Model Predictive Control module to compute a trajectory. Trajectory optimization is performed and regulates the UAVs speed, for example, by slowing down as buildings are approached. Several tuning parameters are provided that allow the designer to tune the risk taking behavior of the UAV as it prosecutes its mission, with the desired behavior of the UAVs being a "tactical" behavior where the UAV will sneak slowly along building walls and wish dash across open areas in order to remain better concealed. In this paper, we assume that the building layout is known a priori, and we then pre-compute a data structure from the medial axis graph which allows for efficient lookup of shortest path while in flight. We note that generally speaking MDP-based methods simultaneously solve the path planning, collision avoidance, and trajectory optimization problems and in this paper we use FastMDP, ORCA, RRT, and RRT* to perform collision avoidance and trajectory generation. Note that nothing precludes the use of a trajectory optimization algorithm underneath FastMDP, but doing so is out of scope for this paper.

The paper is organized as follows: Section II discusses the algorithms evaluated in this paper, Section III develops the approach used in solving the problem including extending the FastMDP algorithm to polygonal obstacles, Section IV describes the simulation environment based on a portion of New York City, Section V describes the test results, and Section VI provides concluding remarks.

II. Background

Optimal Reciprocal Collision Avoidance (ORCA) [15] is a well known algorithm that performs collision avoidance by computing regions which could potentially be occupied by other agents (velocity obstacles) and then selects a velocity for the agent which is outside the occupied area of all other agents. ORCA assumes agents are holonomic, can instantaneously change velocity in any direction, and is optimal under the assumption that all other agents are also using ORCA. ORCA is used in this paper as a representation of what optimal or nearly-optimal collision avoidance with respect to other aircraft might look like as a useful comparison data point for the other algorithms.

Rapidly-exploring Random Trees (RRT) [16] is a well-known algorithm from robotics than uses a random sampling approach to expand a tree of future states out from the initial state until a goal state is reached. An extension to RRT known as RRT* (pronounced "RRT-star") [17] improves upon the paths generated by RRT by periodically performing a limited rebalancing of the tree known as "rewiring" which leads to smoother, simpler paths through the space. RRT and its variants are normally used for path-finding problems through complex environments with many difficult obstacles to avoid. RRT and RRT* are good algorithms to compare with as they should do an excellent job of avoiding buildings and aircraft, but with increasing computational cost as the tree becomes more complex. RRT and RRT* are both capable of solving our "global" path planning problem, but would require a very large amount of time to do so (minutes in our early experiments.)

FastMDP [18], [28] is a recently proposed algorithm which solves a limited subclass of Markov Decision Processes (MDPs) [22], [23] and has been applied to numerous aerospace related problems [29]–[34], notably including collision avoidance. FastMDP exploits special structure present in the solution of the MDP (the value function) and uses this structure to compute the solution of the MDP very quickly (usually dozens or hundreds of milliseconds.) One of FastMDP's limitations is that it requires an efficient distance metric (one which can be computed quickly with low computational cost) through the MDP's transition function, which for simple environments may be an L1 (Manhattan) or L2 (Euclidean) norm. Thus, it is difficult to apply FastMDP to the urban environment problem examined in this paper, as there is no easy-to-compute norm through the environment.

The difficult nature of this problem requires some additional framework for these algorithms to be used effectively.

III. Approach

In this paper, we break the problem into a "global" approach which finds a good path through the city from a starting location to a goal location represented as series of waypoints, and a "local" approach which then avoids collisions while navigating to the next waypoint. At each time step of the simulation, the "local" problem is re-solved using the current state of the environment, resulting in a receding horizon-like approach which can adapt to the unknown future positions of other aircraft.

Buildings are represented as convex polygons derived from floor plans of 3D models of New York City buildings available from [35]. A polygon-with-holes is defined by a bounding polygon large enough to contain all of the building polygons; the building polygons are then treated as holes within the bounding polygon. Given a convex polygon with convex holes, the polygon's medial axis can be found, which is the set of points in the polygon's interior which are equidistant from at

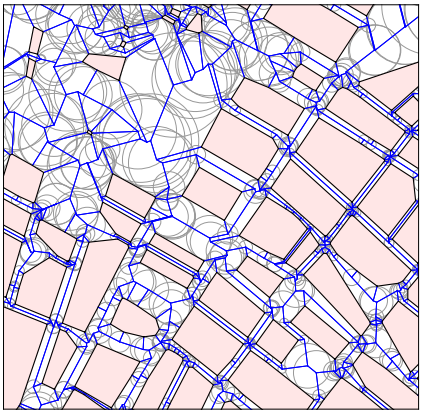

Fig. 1. Medial Axis With Radii Shown - Basis for Medial Axis Graph. Radii shown as shaded gray circles which occur only within the free space (shaded white.)

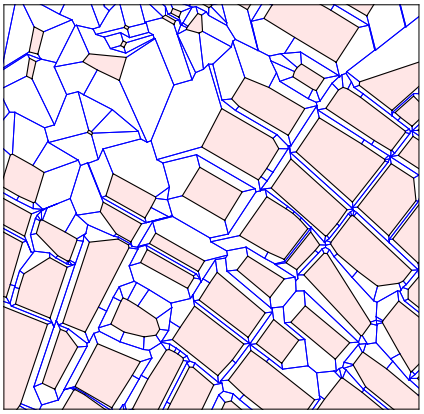

Fig. 2. Medial Axis of Area Between Buildings in an Urban Environment. Buildings are shaded red, the free space is shaded white, and the medial axis is shown as blue line segments.

least two of the polygon's sides. The medial axis (or "straight skeleton") is a well known concept from computational geometry [36] and has been used for indoor navigation [37], computer aided drafting/modelling [38], and generating virtual endoscopy of colonoscopy scans for radiologists [39]. There are many ways to compute and interpret the medial axis; see [38], [40] for an overview and details. The medial axis can be defined using line segments, but an alternative way of representing it is as a graph, which we will refer to as the *medial axis graph* (MAG), where the nodes of the graph are endpoints of the medial axis line segments, and edges of the graph describe connected endpoints and correspond to the line segments. See Figure 2 for an example medial axis derived from our polygonal building representations. Each node of the MAG also is assigned a radius, which is the distance to the nearest point on the boundary (interior or exterior) of the polygon-with-holes. The nodes with their radius can be viewed as "supports" for the skeleton and contain all of the information needed to reconstruct both the skeleton and the polygon itself [38]. See Figure 1 for an illustration of the medial axis with radii depicted as gray circles centered around their corresponding node.

The medial axis can be computed by computational geometry packages such as CGAL [41], though for large number of polygons the time to find the medial axis can be large – too long for generation at run-time. (In our case, it took approximately 5 hours to precompute the MAG. We believe that MAG computation should also be able to be accelerated on GPU to reduce the MAG computation time significantly, but did not explore that idea under this paper's scope.) Thus, for our problem we pre-compute the MAG and then also pre-compute the pair-wise distances through the MAG for every pair of nodes. Thus, given any node of the MAG, we can use a lookup table to find the distance to any other node and can perform simple operations such as finding the next closest node along the shortest path from a start to and end node. This forms the basis of our approach's ability to navigate our city. While we only examine a small area of New York City in this paper, without loss of generality, this approach could easily be extended to a tile-based approach where the MAG for the current area of the city being traversed is loaded. Note also that while we start with 3D building models, we found it computationally expensive to compute the MAG in 3D and have simplified path planning problem between the buildings to a 2D problem. The MAG is used to efficiently identify the regions of free space which can be safely navigated by the UAV, but the algorithms are free to then navigate that obstacle-free space in 3D (at least, the FastMDP, and 3D versions of RRT and RRT*.) While not addressed in this paper, it would be a straightforward extension to compute the MAG at different altitudes as "MAG slices" and then link the MAG slices together in order to allow the UAVs to navigate over the top of shorter buildings.

Given a start location we find the closest node of the MAG (the "start node"), and then given the goal location, we find the closest node of the MAG (the "exit node") and can then easily compute the distance from the start to the goal through the MAG. Starting with the start node, we use the next node in the shortest path through the graph as our waypoint for the "local" problem. As we approach the waypoint, we reach a threshold where we can sequence the waypoint to the next waypoint along the shortest path. Recall that each node of the MAG has a radius; this radius is used as the threshold for sequencing the next waypoint. Moreover, if we are ever within the radius of any MAG node, it is safe to sequence to the next waypoint. This leads to the waypoint sequence logic of Algorithm 1. Thus, the local collision avoidance algorithm is provided with an intermediate goal which is "ahead" of the agent along the shortest path to the goal. ORCA and RRT/RRT* can consume this intermediate goal directly, but FastMDP requires extension in order to deal with polygonal negative rewards, which we develop next.

A. FastMDP and Negative Reward Regions

FastMDP has previously been shown to perform collision avoidance of aircraft, but in prior work FastMDP could only handle "point source" rewards where positive and negative rewards occur at specific points in the state space. A CPU and GPU version of FastMDP have been explored in prior papers. The CPU version of FastMDP used in this paper was shown to have computational complexity of $O(|R| \times |A| \times |D|)$ where $|R|$ is the number of discrete rewards in the reward function, $|A|$ is the number of actions (or control inputs) that the agent can take at each time step, and $|D|$ is the number of discrete time steps in the receding horizon problem. In this work, we extend FastMDP to handle polygon-shaped negative rewards and use these to represent collisions with buildings. This is a useful extension to FastMDP which has broader impacts to other types of obstacles in the environment which may also be represented with polygons, such as weather, but we leave those other applications to future work. We now extend the FastMDP formalism in [28] from point source rewards to polygonal rewards.

Formally, let us define a distance function $\delta(s, s_i)$ as the minimum number of actions $a \in A$ needed to reach state $s_i \in S$ starting from state $s \in S$:

$$\delta(s, s_i) = \min_t \{t \mid T(s, a_1, a_2, \cdots, a_t = s_i\}, \tag{1}$$

where $a_1, a_2, \cdots, a_t$ represent a sequence of actions taken at each time step, and $T(s, a_1, \cdots, a_t)$ represents the transition function applied at each time step t using the sequence of actions starting at state s.

We define a *region of negative reward* as a set $\mathcal{D}_i$ where an indicator function $\mathcal{I}$ is defined such that $\mathcal{I}(s_i) = 1$ for all $s_i \in S$ which lie within the boundary of $\mathcal{D}_i$ and otherwise zero. We likewise define the MDP reward function such that $R(s_i) = r_d, \forall s_i \in \mathcal{D}_i$, otherwise 0, where $r_d < 0$ is a negative, scalar value. Let $S_i = \{s_i \in S \mid \mathcal{I}(s_i) = 1\}$ be the set of states within the boundary of the region.

Let $S^Z = \{s \in S \mid R_{\mathcal{D}_i}(s) = 0\}$ be the set of all states where zero reward is obtained. Now, let $S^- = \{s \in S \mid R_{\mathcal{D}_i}(s) < 0\}$ be the set of all states where negative reward is received (that is, the set of states that lie within region $\mathcal{D}_i$.) Let us denote the set of all states k steps from S^Z as $S^k = \{s \in S \mid \min_{s_z \in S^Z} \delta(s, s_z) = k\}$ where $k \geq 0$ is an integer, with $S^0 = \{S^Z\}$, S^1 with all states one step from S^Z, etc. For any $k > 0$, $S^k \subset S^-$ and $S^- = \cup_{k=1}^K S^k$. For any bounded, non-infinite region $\mathcal{D}_i$, note that k is bounded by some integer $K = \max_{s \in S^-, s_z \in S^Z} \delta(s, s_z)$ which represents the maximum number of steps required to reach S^0 from any $s \in S^-$. By definition, the reward received at any state $s_0 \in S^Z$ is 0 and the reward received for any state $s_k \in S^k$ with $k > 1$ is the negative reward $r_d < 0$.

Let us assume a set of states $\mathcal{T} = \{s_1, s_2, s_3, \cdots, s_N\}$ which form a trajectory through the state space S at time steps $t = \{1, \cdots, N\}$. From standard MDP literature, let us define the future discounted return over a trajectory T as $G = \sum_{i=1}^N \gamma^i R_i$, where $0 < \gamma < 1$ is known as the discount factor of the infinite horizon MDP and R_i is the reward obtained at each time step.

Lemma 1. *For an MDP containing only a region of negative reward $\mathcal{D}_i$, the optimal action from S^0 is to remain in S^0 resulting in a return of: $G_0 = \sum_{i=1}^N \gamma^i \cdot R_i = 0$.*

Proof. Assume that we begin at a state $s_0 \in S^0$. If we take an action $a \in A$ and transition to a next state $s' \in S$, there are two possible outcomes:

$$s' \in \begin{cases} S^0 & \text{if } T(s_0, a) \in S^0 \\ S^k, k > 0 & \text{otherwise.} \end{cases} \tag{2}$$

Recall that $S^0 = S^Z$ and that the reward received at state $s_z \in S^Z$ is 0, and the reward received in $S^k, k > 1$ is a negative reward $r_d < 0$. If we consider an infinite trajectory of states $\mathcal{T}_0 = \{s_1, s_2, \cdots, s_N \mid s_i \in S^Z, \forall\, i \in \{1, \cdots, N\}\}$ that all lie within S^0, then the reward received at each step is $R_i = 0$, and the future discounted return of the trajectory is:

$$G_{\mathcal{T}_0} = \sum_{i=1}^N \gamma^i \cdot R_i = \sum_{i=1}^N \gamma^i \cdot 0 = 0. \tag{3}$$

If we consider an infinite trajectory of states $\mathcal{T}_1 = \{s_1, s_2, \cdots, s_N \mid \exists\, s_i \in S^-, \forall\, i \in \{1, \cdots, N\}\}$ where one or more states lie within S^-, then the reward received at each state within S^- is the negative reward $r_d < 0$:

$$\begin{aligned} G_{\mathcal{T}_1} &= \sum_{i=1}^N \gamma^i \cdot R_i \\ &= \sum_{i=1}^j \gamma^i \cdot 0 + \sum_{i=j}^k \gamma^i \cdot r_d + \sum_{i=k}^m \gamma^i \cdot 0, \end{aligned} \tag{4}$$

and the future discounted return of the trajectory is guaranteed to be less than $G_{\mathcal{T}_0}$:

$$\begin{aligned} G_{\mathcal{T}_0} &> G_{\mathcal{T}_1} \\ \sum_{i=1}^N \gamma^i \cdot R_i &> \sum_{i=1}^N \gamma^i \cdot R_i \\ \sum_{i=1}^N \gamma^i \cdot 0 &> \sum_{i=1}^j \gamma^i \cdot 0 + \sum_{i=j}^k \gamma^i \cdot r_d + \sum_{i=k}^m \gamma^i \cdot 0 \\ 0 &> \sum_{i=j}^k \gamma^i \cdot r_d \end{aligned} \tag{5}$$

Therefore, if we start at some state $s_0 \in S^0$, then the optimal action $a^* \in A$ is one which leads to maximum future expected return, namely trajectory $\mathcal{T}_0$. Thus we can say that S^0 is absorbing, as once entering S^0 the optimal action is to remain in S^0. An infinite trajectory starting within S^0 then always stays within S^0 leading to a return

Algorithm 1 Waypoint Sequencing through the MAG

```
procedure SEQUENCEWAYPOINT(current_position)
    d_waypoint ← ‖current_position − waypoint.position‖_2
    while d_waypoint < waypoint.radius do
        waypoint ← next closest waypoint in MAG
        d_waypoint ← ‖current_position − waypoint.position‖_2
    end while
end procedure
```

of $G_0 = \sum_{i=0}^{\infty} \gamma^i 0 = 0$. Moreover, any trajectory that traverses only states in S^0 has a return which we will denote $G_0 = G_{\mathcal{T}_0} = \sum_{i=1}^{N} \gamma^i \cdot R_i = 0$. □

Lemma 2. *For an MDP containing only one region of negative reward $\mathcal{D}_i$, the optimal trajectory from a state $s_k \in S^k, k > 0$ is to move to a state $s_0 \in S^0$ in a minimal number of actions:*

$$\mathcal{T}_k^* = \min_t \{t \mid T(s, a_1, a_2, \cdots, a_t) = s_0 \in S^0\}, \tag{6}$$

Proof. Assume that the initial state $s_k \in S^k$ with $k = 1$ and taking an action $a \in A$ which leads to a next state $s' \in S$. The following transitions are possible:

$$s' \in \begin{cases} S^0 & \text{if } T(s_k, a) \in S^0 \text{ (Case A)} \\ S^1 & \text{if } T(s_k, a) \in S^1 \text{ (Case B)} \\ S^k, k > 1 & \text{otherwise (Case C).} \end{cases} \tag{7}$$

Lemma 1 shows that S^0 is absorbing and has a return $G_0 = 0$, thus for Case A the return would be $G_A = r_d + \gamma \cdot G_0 = r_d$, with $r_d < 0$. For Case B, if we temporarily denote the return starting from $s_k, k = 1$ as G_1, then the return of Case B is defined recursively as $G_B = r_d + \gamma G_1$ as we return back to S^1. While the return of G_1 is currently unspecified, it is clear that $G_B < G_A$ and therefore Case B is not optimal as at best we added one additional step before reaching S^0 which only serves to reduce our return. For Case C, we see a similar situation as any $S^k, k > 1$ implies that the distance to S^0 has increased, meaning that the return has at least k r_d terms in it: $G_C = r_d + \cdots \gamma^i \cdot r_d + G_0$. Therefore, Case C is also not optimal. Thus, for $s_k \in S^k, k = 1$, the optimal action is to transition to S^0 for a return of $G_1 = r_d + \gamma \cdot G_0$.

Now consider starting from some $s_k \in S^k, k > 1$ and taking an action $a \in A$ leading to a next state $s' \in S$. Possible outcome are:

$$s' \in \begin{cases} S^j, j < k & \text{if } T(s_k, a) \in S^j, j < k \text{ (Case A)} \\ S^k & \text{if } T(s_k, a) \in S^k \text{ (Case B)} \\ S^m, m > k & \text{otherwise (Case C).} \end{cases} \tag{8}$$

Following similar logic, for Case A, this results in a return of $G_A = r_d + \gamma \cdot G_j$. For Case B, if we temporarily denote the return starting from $s_k \in S^k$ as G_k, then the return of Case B is defined recursively as $G_B = r_d + \gamma \cdot G_k$ as we return to S^k. While the return of G_k is currently unspecified, it is clear that $G_B < G_A$ and therefore Case B is not optimal. For Case C, $S^m, m > k$ implies that the distance to S^0 has increased, meaning that the return has at least k r_d terms in it: $G_C = r_d + \cdots + \gamma^i \cdot r_d + G_k$. Therefore, Case C is also not optimal. Thus, at each level S^k we find that the optimal action is to take the action that reduces k. Given that S^k is defined in terms of the minimum number of actions it takes to reach S^0, it is therefore only possible to reduce k by 1 at each step, and the optimal trajectory then steps down from higher levels of S^k to S^0 and is then:

$$\begin{aligned} \mathcal{T}_k^* &= \{s_k \in S^k, s_{k-1} \in S^{k-1}, \cdots, s_0 \in S^0, s_0 \in S^0, \cdots\} \\ &= \min_t \{t \mid T(s, a_1, a_2, \cdots, a_t) = s_0 \in S^0\}, \end{aligned} \tag{9}$$

noting again that once S^0 is reached, it is an absorbing state with return $G_0 = 0$. □

Corollary 1. *For an MDP containing only a region of negative reward $\mathcal{D}_i$, starting from a state $s_k \in S^k, k > 0$ the future discounted return is: $G_k = \sum_{i=0}^{k} \gamma^i \cdot r_d$.*

Proof. The proof is a straightforward application of Lemma 2:

$$\begin{aligned} G_k &= R(s_k) + \cdots + \gamma R(s_1) + \gamma R(s_0) + \cdots \gamma R(s_0) \\ &= r_d + \gamma r_d + \cdots + \gamma^k r_d + \gamma^{k+1} 0 + \cdots \gamma^{k+n} 0 \\ &= r_d + \sum_{i=1}^{k} \gamma^i \cdot r_d + 0 \\ &= \sum_{i=0}^{k} \gamma^i \cdot r_d \end{aligned} \tag{10}$$

□

Definition 1. *Let us now define the* k-distance *function $K_i(s)$, which returns for a given state $s \in S$ the value of k for which the state belongs in S^k with respect to region $\mathcal{D}_i$, then we may define the value function as follows.*

Corollary 2. *For an MDP containing only a region of negative reward $\mathcal{D}_i$, the value function $V^*(s)$ for any state $s \in S$ is: $V^*(s) = G_{K_i(s)}$*

Proof. The value function is defined as the future expected reward starting at each state $s \in S$. Lemma 1 defined the return G_0 for states $s \in S^0$ and Lemma 2 define the return G_k for states $s \in S^k, k > 0$. As $S = \cup_{i=0}^{N} S^k$ and $k = K(s)$ provides which S^k the state s belongs to, then $V^*(s) = G_{K(s)}$. □

Consider an MDP with two regions of negative reward, $\mathcal{D}_i$ and $\mathcal{D}_j$. To distinguish between the the regions, let us denote S_k with respect to region $\mathcal{D}_i$ as S_i^k and let us denote S_k with respect to region $\mathcal{D}_j$ as S_j^k. S_i^k and S_j^k can be understood as level sets within each region $\mathcal{D}_i$ and $\mathcal{D}_j$.

Theorem 1. *The value function resulting from two disjoint regions $\mathcal{D}_i$ and $\mathcal{D}_j$ separated by at least $\delta(s_i, s_j) > 1$, $\forall s_i \in \mathcal{D}_i$, $\forall s_j \in \mathcal{D}_j$ is:*

$$V^*(s) = \min_{s \in S} V_p(s), \ \forall p \in \{i, j\}, \tag{11}$$

where $V_p(s)$ indicates the value function resulting from only region $\mathcal{D}_p$ being present in the MDP.

Proof. Recall the k-distance metric from Definition 1, $K_i(s)$ which for a state $s_i \in \mathcal{D}_i$ with $s_i \in S_i^k$ for some k, returns the minimum number of steps k which are required to reach S^Z from s_i.

As each region is separated by S^Z level sets S^k are established for $k = \{0, 1, \cdots, K\}$ for each region $\mathcal{D}_i$ and $\mathcal{D}_j$. As the regions are separated by S^Z, for a given state $s_i \in \mathcal{D}_i$, the distance to S^Z is unaffected by the presence of region $\mathcal{D}_j$. That is, $\forall s_i \in \mathcal{D}_i, K_i(s_i|\mathcal{D}_i) = K_i(s_i|\mathcal{D}_i, \mathcal{D}_j)$ where the notation $K_i(s_i|\mathcal{D}_i)$ indicates an MDP where only $\mathcal{D}_i$ is present and $K_i(s_i|\mathcal{D}_i, \mathcal{D}_j)$ indicates an MDP where both $\mathcal{D}_i$ and $\mathcal{D}_j$ are present. Likewise, $\forall s_j \in \mathcal{D}_j, K_j(s_j|\mathcal{D}_j) = K_j(s_j|\mathcal{D}_j, \mathcal{D}_i)$. Therefore, the distances $K_i(s_i)$ and $K_j(s_j)$ are independent of

each other, and are equivalent to the case where each region $\mathcal{D}_i$ and $\mathcal{D}_j$ are present in the MDP in isolation. It then follows that S_i^k and S_j^k are also independent from each other and are equivalent to the case where each region $\mathcal{D}_i$ and $\mathcal{D}_j$ are present in the MDP in isolation. That is, $S_i^k = S_{i,j}^k$ where $S_{i,j}^k = \{s \in S_i^k | \mathcal{D}_i, \mathcal{D}_j\}$ indicates the S_i^k when region $\mathcal{D}_j$ is also present in the MDP.

With independence of the regions established, using Corollary 1 and Corollary 2 the value function of $\mathcal{D}_i$ is $V_i^*(s) = G_{K_i(s)} = \sum_{t=0}^{k} \gamma^t \cdot r_i$, where r_i is the constant negative reward associated with region $\mathcal{D}_i$, and the value function of $\mathcal{D}_j$ is $V_j^*(s) = G_{K_j(s)} = \sum_{t=0}^{k} \gamma^t \cdot r_j$, where r_j is the constant negative reward associated with region $\mathcal{D}_j$. We note that $V_i^*(s) \leq 0$ and $V_j^*(s) \leq 0$ and given the independence of $\mathcal{D}_i$ and $\mathcal{D}_j$, the resulting value function is therefore $V^*(s) = \min_{s \in S} V_p(s),\ \forall p \in \{i, j\}$. □

Theorem 1 can be readily extended to N disjoint regions $\{\mathcal{D}_1, \mathcal{D}_2, \cdots, \mathcal{D}_N\}$ where all pairs of regions $\{\mathcal{D}_i, \mathcal{D}_j\}$, $i, j \in \{1, \cdots, N\}, i \neq j$ are separated by at least $\delta(s_i, s_j) > 1,\ \forall s_i \in \mathcal{D}_i,\ \forall s_j \in \mathcal{D}_j$.

For our problem, each of these disjoint regions $\mathcal{D}_i$ is a convex polygon. Well known algorithms exist to determine distance from a convex polygon which are reasonably efficient (linear in the number of polygon vertices), which in essence computes distance to the closest line segment. However, given the large number of polygons in our problem, a GPU based implementation was used which performs minimum distance checks to all polygon edges in parallel with a convention adopted that points within the polygon have negative distance and points on the exterior of the polygon have positive distance (i.e, a *signed distance function*.) This signed distance function is used as the distance metric $\delta(s, s_i)$ and the k-distance function $K_i(s)$, and our value function for each polygon is computed per Corollary 2. To provide some additional safety margin around buildings, we inflate the building polygon by 5 meters by subtracting this value from the signed distance function. We refer to the GPU function which computes the minimum signed-distance over all polygons as $SDF(s_i)$, where $s_i \in S$. We omit the implementation description due to space considerations and the well-known nature of the computation.

In addition to the polygon based negative rewards, intruder aircraft receive negative rewards around their future expected position derived from the current position and velocity. We additionally apply negative rewards if the agent exceeds a minimum or maximum altitude. Algorithm 2 summarizes the FastMDP algorithm used for this paper. On line 2 we initialize inputs into the algorithm including possible actions the aircraft can take and any constraints on the aircraft motion. On line 9 we begin building "peaks", which compute the value function contribution from each reward. We project forward dynamics of the aircraft on line 13 to establish the future state of the aircraft for each possible action. The value function contribution for reaching the goal is computed on lines 18-23, where we determine our distance d_p to the goal and compute the discounted future reward on line 21 using the reward magnitude r_p and the MDP discount factor γ. On lines 25-30, we compute the negative reward value function contribution for each intruder aircraft using the distance d_n, the reward r_n and the discount factor γ. Note that we also defined a boolean value ρ_n which is used to only update the value function if the distance d_n to the negative reward is within the predefined risk well radius. Thus if we are outside the radius, then the resulting value contribution is zero. On lines 32-37 we compute the value function contribution for buildings with distance d_b computed from the minimum signed distance over all building polygons, the penalty r_b for colliding with a building, the discount factor γ, and a boolean value ρ_b which tests whether the state is inside the building polygon. Thus if a state is inside the building polygon, a penalty is applied, otherwise no penalty is applied. On line 38 we compute the intermediate negative value contribution from all buildings and intruder aircraft, and on lines 40-42 we compute altitude penalties if our current altitude is above or below a specified altitude threshold. On line 43 we compute the value function from all sources. On line 51 we identify the most valuable action and on 47 each action is taken simultaneously in simulation. Note that all aircraft make their action selections independent of all intruders and without knowledge of any other intruders' actions, making this is a decentralized implementation.

IV. Simulation

To create a challenging test bed, a 3D model of a section of New York City was downloaded from [35]. Detailed floor plan outlines were extracted from the models and convex hulls were created from the floor plans. As the models were very detailed, some buildings floorplans overlapped (balconies, overhangs, skywalks, etc), some buildings were highly clustered (e.g., small buildings with adjoining walls) such that the resulting convex polygons numbered in the thousands. To reduce the polygon count, adjacent and overlapping buildings were clustered and merged into larger convex polygons which subsumed the smaller buildings resulting in a final polygon count of 330 buildings. Furthermore, in places where curves were very detailed, the number of vertices was reduced by expanding the polygon subtly in these areas to create a nearly equivalent convex polygon with fewer vertices than the original. The maximum vertex count of any polygon was reduced from approximately 130 vertices down to a maximum vertex count of 11. Start and goal locations were chosen randomly in free space and the "global" algorithm was used to identify the first waypoint. The "local" algorithm is called at each time step to control the motion of the UAS. As the UAS fly towards their waypoints, waypoints are sequenced per Algorithm 1. Through the simulation, the closest point of approach between UAVs is tracked along with the closest approach to any building.

RRT, RRT*, and FastMDP are implemented in 2D and 3D, with serious consideration given only to the 3D cases. RRT, RRT*, and FastMDP are all limited in their turn rate and pitch rate. All four algorithms are constrained to a maximum airspeed of 20 m/s in this simulation, which given the narrow corridors between buildings is a reasonable limit (perhaps bordering on aggressive.)

Algorithm 2 FastMDP algorithm.

procedure FASTMDP
  $\mathbf{S} \leftarrow$ *initial aircraft states*
  $\mathbf{A} \leftarrow$ *aircraft actions (a priori)*
  $\mathbf{L} \leftarrow$ *aircraft limits (a priori)*
  while aircraft remain **do**
    for each aircraft **do**
      $s_t \leftarrow \mathbf{S}[aircraft]$
      // Build peaks
      $\mathbf{P}^+ \leftarrow$*pos reward for intermediate goal*
      $\mathbf{P}^{\mathbf{B}} \leftarrow$*neg rewards from buildings*
      $\mathbf{P}^{\mathbf{A}} \leftarrow$*neg rewards from other aircraft*
      // Perform forward projection of aircraft dynamics
      $\mathbf{\Delta} \leftarrow fwdProject(s_t, \mathbf{A}, \mathbf{L})$
      // Compute the value at each reachable state
      $\mathbf{V}^* \leftarrow$ *allocate space for each reachable state*
      for $s_j \in \mathbf{\Delta}$ **do**
        // First for positive peaks
        for $p_i \in \mathbf{P}^+$ **do**
          $d_p \leftarrow \|s_j - \mathbf{location}(p_i)\|_2$ ▷ distance
          $r_p \leftarrow \mathbf{reward}(p_i)$
          $\mathbf{V}^+(p_i) \leftarrow |r_p| \cdot \gamma^{d_p}$
        end for
        $V^+_{max} \leftarrow \max_{p_i} \mathbf{V}^+$
        // Next for negative peaks for intruders
        for $n_k \in \mathbf{P}^{\mathbf{A}}$ **do**
          $d_n \leftarrow \|s_j - \mathbf{location}(n_k)\|_2$ ▷ distance
          $\rho_n \leftarrow d_n < \mathbf{radius}(n_k)$ ▷ within radius
          $r_n \leftarrow \mathbf{reward}(n_k)$
          $\mathbf{V}^-(n_k) \leftarrow \rho_n \cdot |r_n| \cdot \gamma^{d_n}$
        end for
        // Next for negative peaks for buildings
        for $n_b \in \mathbf{P}^{\mathbf{B}}$ **do**
          $d_b \leftarrow SDF(n_b)$ ▷ distance
          $\rho_b \leftarrow d_b < 0$
          $r_b \leftarrow \mathbf{reward}(d_b)$
          $\mathbf{V}^-(n_b) \leftarrow \rho_b \cdot |r_b| \cdot \gamma^{d_b}$
        end for
        $V^-_{max} \leftarrow \max_{n_k, n_b} \mathbf{V}^-$
        // Altitude penalties
        $alt \leftarrow altitude(s_t)$
        $V_{alt} \leftarrow apply\_altitude\_penalty(alt)$
        $V^-_{max} \leftarrow \max(V^-_{max}, V_{alt})$
        $\mathbf{V}^*[s_j] \leftarrow V^+_{max} - V^-_{max}$
      end for
      // Identify the most valuable action
      $a_{max} \leftarrow \arg\max_a(\mathbf{V}^*)$
      // Record each aircraft's action
      $\mathbf{A}^*{}_{t+1}[aircraft] \leftarrow a_{max}$
    end for
    // Now that all aircraft have selected an action, apply it
    $\mathbf{S} = SimulationUpdate(\mathbf{A}^*{}_{t+1})$
  end while
end procedure

Sensors are ignored within the scope of this paper and perfect knowledge is assumed. In reality, sensor uncertainty, communications delays, multi-path effects, and available localization and navigation fix methods would all be challenging for this problem. Existing sensors such as GPS and ADS-B would not be able to provide sufficient location resolution or update rates, and employing the proposed approach in an urban environment may, for example, require special infrastructure for localization (e.g., markers or beacons at intersections or along buildings) and communication (e.g., special radio links used to broadcast position updates throughout the city.) LIDARs and other sensors may be able to provide local area estimates, but reflection and multipath returns may cause additional uncertainties. While these effects are outside the scope of this paper, we have previously empirically studied the performance of FastMDP under uncertainty in [34] and other papers and have found that it seems to perform well in the presence of uncertainty in action (or control input) and uncertainty in intruder position and intruder intent, though more work is required on the theory.

RRT, RRT*, and FastMDP all require knowledge of whether a point is within an obstacle. Efficient in-polygon checks are available in libraries, but given the large number of polygons in this environment, this takes too long to perform. We opted to provide a GPU implementation to all three algorithms. In-polygon checks of convex polygons can be implemented very efficiently on GPUs, and our long term intended target is an embedded GPU board such as an NVIDIA Jetson NX, which is a low Size Weight and Power (SWAP) computer intended for robotics, automotive, and unmanned aerial vehicles. While FastMDP has been implemented fully in the GPU in [33], in order to provide an apples-to-apples comparison, we have only implemented a small portion of FastMDP in the GPU for this paper – namely the portion that computes distance from the polygon obstacles, which is very similar to the GPU-based in-polygon check. With the exception of this distance computation, we have the rest of the FastMDP algorithm on the CPU so that FastMDP does not have an unfair boost in performance due only to the GPU. We felt this was a reasonable compromise that allows the GPU to overcome the large number of polygons which drags down all of the algorithms' performance while still providing a level playing field to usefully compare the algorithms. We also note that from [15], ORCA should be able to support convex obstacles; however, we were unable to replicate this functionality using the RVO-2 version of the algorithm published by the authors.

For the tests, the same set of randomly generated start and goal locations for 100 UAS was provided, with each UAS starting in the simulation at its randomly generated start location one second after the previous UAS. This results in a gradual ramp-up of the number of UAS over time with a sustained density of UAS during the middle of the simulation which tapers off as UAS reach their destinations. As each algorithm is tested with the same start and goal locations (and start times), this allows for a fair comparison between algorithms.

In [27], it was noted that using hard "buffers" around obstacles (e.g., by inflating the boundary of an obstacle by a safety margin), useful corridors for navigation are lost that otherwise could have been navigated. We note here that for each algorithm, we tuned the safety margin to minimize this effect. In the case of RRT and RRT*, the safety margins are a hard constraint that cannot be violated per the normal RRT formulation and manifests as RRT and RRT* refusing to form a new edge in the tree that would enter the obstacle region (which includes the buffer.) In the case of FastMDP (like all standard MDPs), the obstacles are soft constraints which are balanced against possible rewards. The magnitude of the rewards and the penalties are chosen such that the

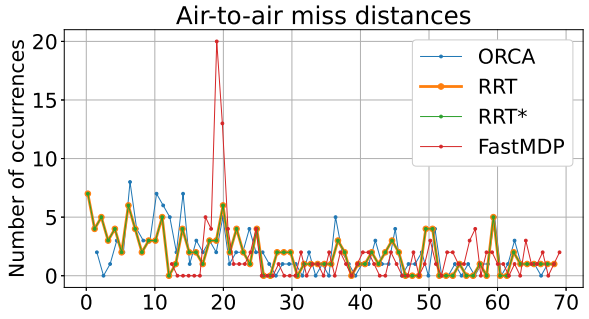

Fig. 3. Air-to-Air Miss Distances

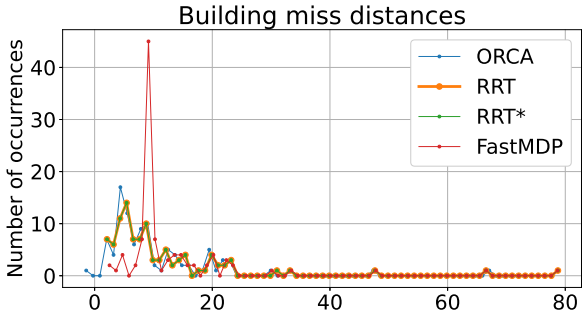

Fig. 4. Building Miss Distances

penalty of a collision outweighs any possible reward, thus collision avoidance is achieved. Due to the way negative reward propagates in an MDP, the safety margin is selected such that it provides an advance warning to the MDP agent of an impending collision and the magnitude of the penalty is a gradient decreasing with increased distance from the obstacle. The safety margin must also be large enough to account for the dynamics of the aircraft (e.g., bank angle, turn rate, g-forces, climb/descent rate limits, etc.) FastMDP will then balance the desire for obtaining the reward (e.g. reaching the next waypoint) with any penalties (e.g., collision with building or collision with another aircraft.) In practice, this allows FastMDP when traversing a corridor to hug the wall closer to a building (a moderate penalty) to avoid collision with another aircraft (a more severe penalty) while still making progress towards the waypoint (the reward.)

V. Results and Discussion

Figures 3 and 4 compare collision avoidance performance of each of the algorithms. In the plots, the miss distance in meters is plotted along the x-axis, and the y-axis shows the number of occurrences. The plots shown are histograms showing the miss distances that occurred over the aircraft flights, where miss distances were binned into histogram bin values indicated by the points on the plots and the line segments of the plots are meant to help identify and compare the histograms. The x-axis was truncated above a certain value for each plot (approx 70 - 80 meters). Any value on the x-axis ≤ 0 indicates that a collision occurred. The plots also reveal distributions of the miss distances; the location and relative magnitude of peaks in this distribution provide insight into how well the algorithms perform collision avoidance. Note that in the plots, RRT and RRT* performed very similarly, and their plots are largely overlaid atop one another. This is indicated in the plot with RRT* having a solid green line and RRT having a thicker plot with larger points, though this may be difficult to discern so we highlight this for the reader.

Figure 3 shows air-to-air miss distances between UAS. The plots indicate that RRT and RRT* both had numerous collisions (miss distance of 0 meters) while ORCA had miss distances of approximately 1-2 meters. FastMDP in these tests maintained a minimum miss distance of 13 meters. If we consider the area under each of the plots, we see that there are numerous instances of near-collisions with other aircraft routinely through the flights for ORCA, RRT, and RRT*. As ORCA is limited to 2D evasion in these experiments, we could expect more conflicts for ORCA, but considering the area under the curves, it appears ORCA has done moderately better than RRT and RRT*, which both can evade in 3D and which should naturally lead to better evasion. Examining the paths generated by RRT and RRT*, we find that due to the tree expansion performed by RRT and RRT*, both algorithms seem sensitive to poor selection of early nodes (or "trunks") in the tree. During tree expansion, RRT and RRT* both rapidly subdivide the space. During subsequent expansion steps, the closest node to the newly generated random sample is found and expansion is performed at the selected node. This leads to situations where early branches are less-than-optimal choices, but it is difficult for RRT and RRT* to overcome these early suboptimal branches, meaning that if an early suboptimal branch leads to a higher probability of collision, RRT and RRT* have a limited ability to recover. Though RRT* has the rewiring mechanism and did qualitatively generate better trees in our inspections, we did not find an appreciable improvement in collision avoidance behavior. When examining other conflicts, we also note that RRT and RRT* can exhibit a failure mode where no tree expansion is possible because due to motion constraints, nearly all nodes that are created are infeasible because they lie within the hard constraints of collision avoidance. We view this as the unintended consequence of a hard constraint approach. It is easy to view a hard constraint which cannot be violated as always desirable over a soft constraint approach and in some situations where adequate control is possible, this is valid position to take. Decentralized collision avoidance however is not a fully controllable system as the other agents may take actions which lead to further conflict (both agents may for example choose to avoid in the same direction leading to a head-on collision.) In these cases a soft constraint may be a better approach than a hard constraint for cases where hard constraints lead to an infeasible optimization problem.

In Figure 3, we see that FastMDP has a large spike around 20 meters. In the FastMDP formulation, the agent experiences a gradually increasing penalty starting at 20 meters, so it makes sense to see the spike start at 20 meters and rapidly fall off. We see a couple of instances around 12-13 meters where FastMDP did not maintain a 20 meter separation. This occurs because FastMDP (like any MDP) treats penalties as

TABLE I
ALGORITHM PERFORMANCE COMPARISONS

Algorithm	Physics	Flight Constraints	Building Aware	Aircraft Aware	Time per Frame
ORCA	2D	No	No	Yes	1.3 ms
RRT-2D	2D	No	Yes	No	20 - 30 ms
RRT-2D-A	2D	No	Yes	Yes	70 - 200 ms
RRT-3D-A	3D	Yes	Yes	Yes	170 - 2000 ms
RRT*-2D	2D	No	Yes	No	30 - 80 ms
RRT*-2D-A	2D	No	Yes	Yes	120 - 385 ms
RRT*-3D-A	3D	Yes	Yes	Yes	900 - 7000 ms
FastMDP	3D	Yes	Yes	Yes	32 - 35 ms

soft constraints and FastMDP begins responding to penalties once they begin to incur. Thus when tuning FastMDP, we chose a value for collision avoidance (20 meters) that was larger than the value we were concerned about (0 meters) so that the agent would have time to respond.

Moving to Figure 4 we now examine building miss distances. Here we see that ORCA had collisions with buildings (miss distances ≤ 0.) This should not suprise us at all, as ORCA is only aware of UASs in this experiment and is not aware of buildings. Most likely, while performing collision avoidance with other UAS, ORCA unwittingly crossed into the boundary with a building. Above 2 meters, we see that ORCA, RRT, and RRT* all had approximately equivalent building avoidance behavior, which confirms that the medial axis graph provides a baseline level of building collision avoidance by providing conflict free paths for the local planning problem. We see again that FastMDP has a spike in the plot, this time around 10 meters which is the value we begin applying penalty for buildings. We see that FastMDP responds by avoiding buildings and maintains a better collision avoidance profile under 10 meters than ORCA, RRT, and RRT* suggesting that FastMDP's ability to respond to building collisions is due to a factor beyond the baseline collision avoidance provided by the medial axis graph.

Taken together, in these tests we see that FastMDP is doing a better job of balancing both collisions with other aircraft and collisions with buildings leading to overall superior collision avoidance behavior. Figure 5 shows a sample run with multiple aircraft simultaneously navigating the city environment, with the "ownship" indicated with a blue "X", the intermediate goal indicated with a green dashed line, intruder aircraft indicated with a red "X", and the MAG in dashed blue lines.

Table I compares the computational performance of the algorithms. Originally, ORCA and only simple versions of RRT and RRT* were compared to FastMDP to establish a baseline. The rows RRT-2D and RRT*-2D capture these early baselines which were 2D implementations of RRT without knowledge of flight constraints and without awareness of aircraft. We see that in this case, the computational performance of these variants to FastMDP were comparable, though FastMDP implemented 3D collision avoidance and was aware of the other aircraft. To ensure fair comparisons between the algorithms, the complexity of RRT was gradually increased. The RRT-2D-A and RRT*-2D-A variants added awareness of other aircraft, which were modelled as disc-shaped obstacles that RRT needed to route around. These variants started to see an increase in computation time, which started to make the RRT variants look less favorable compared to FastMDP. When switching to 3D with flight constraints, this dramatically increase the computation needed for RRT and RRT* (shown in the RRT-3D-A and RRT*-3D-A rows in the table) as the restricted turn radius resulted in having to build trees with a larger number of nodes and the third dimension (altitude) also required more nodes to explore. Additionally, the RRT-3D-A and RRT*-3D-A rows also experienced many cases where a solution could not be found. In these cases, rather than a path to the goal, we instead settled for a path to the node of the tree which was nearest to the intermediate goal. We see also that the RRT*-3D-A row also sees a marked increase in computation time which is due to the need to perform more building polygon hit testing due to RRT*'s rewiring operation, which attempts to find better connections between existing nodes in the tree and serves to rebalance the tree in a limited way. For this problem due to the large number of polygons, this hit testing is an expensive operation, even when performed with GPU acceleration as was done for all of these results. Thus we see that RRT and RRT* start to become untenable as the problem becomes more realistic and complex. Recall that ORCA is excellent at avoiding collisions with other aircraft, but has no knowledge of buildings or ability to avoid them in this simulation. Note that the collision avoidance behaviors shown in Figures 3 and 4 used the best-possible collision avoidance variants of RRT-3D-A and RRT*-3D-A and did not show performance for the RRT-2D, RRT-2D-A, RRT*-2D, or RRT*-2D-A variants. The RRT-3D-A and RRT*-3D-A are the fairest comparison with FastMDP as none of the algorithms should have any unfair advantages over the others in terms of physical space they can occupy to avoid collision with other aircraft.

For future work, the authors would like to extend these algorithms to a full 3D representation where we also consider flight over buildings. Early exploration showed there are significant computation barriers that will need to be overcome, likely requiring a full GPU-based implementations of all algorithms for fair comparisons to be made which is out of scope of this paper. Additionally, an alternative to the MAG may be required to represent the airspace structure, perhaps in the form of a multi-altitude MAG where multiple 2D MAGs are stacked on

top of each other. Another important extension for future work would be to incrementally update the MAG allowing edges of the graph to become blocked and allowing an efficient rerouting to be computed from the pre-computed all-pairs-shortest-distance lookup table to allow for better dynamic rerouting.

VI. Conclusion

This paper identified an efficient method for UAS to navigate an urban environment and compared the collision avoidance and computational performance of four algorithms, ORCA, RRT, RRT*, and FastMDP where ORCA was treated as the baseline for the other algorithms. The results show that in a simple enough environment RRT and RRT* have a reasonable computational performance comparable to FastMDP, but as the problem becomes more realistic RRT and RRT* struggle to remain efficient primarily due to having to create more nodes in a 3D environment with constrained motion. FastMDP was shown to perform much better than RRT and RRT* for the 3D environment with constrained motion and knowledge of other UAS. In terms of collision avoidance behavior, FastMDP was shown to perform better than ORCA, RRT, and RRT* for both buildings and aircraft in these simulations.

VII. Acknowledgement

The authors wish to thank the reviewers for their thoughtful comments during the review process. Their constructive, professional comments have helped to improve the overall quality and clarity of the manuscript.

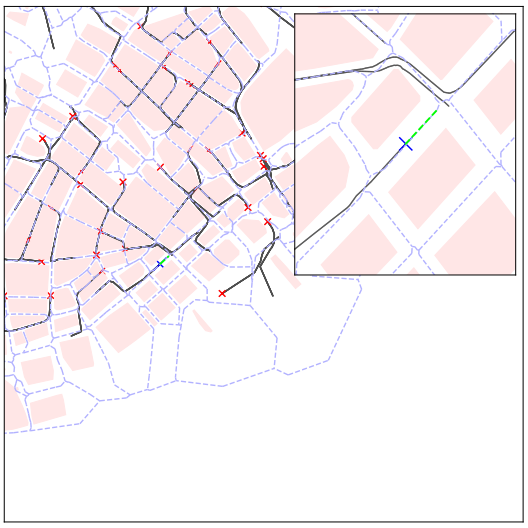

Fig. 5. Example Navigation of Multiple Agents using FastMDP

References

[1] M. Mulvaney and M. Kratsios, "M-18-22: Memorandum for the Heads of Executive Departments and Agencies - FY 2020 Administration Research and Development Budget Priorities," Executive Office of the President, the White House, Tech. Rep., 2018.

[2] R. Vought and K. Droegemeier, "M-19-25: Memorandum for the Heads of Executive Departments and Agencies - FY 2021 Administration Research and Development Budget Priorities," Executive Office of the President, the White House, Tech. Rep., 2019.

[3] David P. Thipphavong and Rafael Apaza and Bryan Barmore and Vernol Battiste and Barbara Burian and Quang Dao and Michael Feary and Susie Go and Kenneth H. Goodrich and Jeffrey Homola and Husni R. Idris and Parimal H. Kopardekar and Joel B. Lachter and Natasha A. Neogi and Hok Kwan Ng and Rosa M. Oseguera-Lohr and Michael D. Patterson and Savita A. Verma, "Urban air mobility airspace integration concepts and considerations," in *In Proceedings of Aviation Technology, Integration, and Operations Conference*, Atlanta, GA, 2018. [Online]. Available: doi:10.2514/6.2018-3676

[4] The Federal Aviation Administration, "Urban Air Mobility (UAM) Conceopt of Operations v1.0," Tech. Rep., 2020. [Online]. Available: https://nari.arc.nasa.gov/sites/default/files/attachments/UAM_ConOps_v1.0.pdf

[5] National Academies of Sciences, Engineering, and Medicine, "Advancing Aerial Mobility: A National Blueprint," The National Academies Press, Washington, DC, Tech. Rep., 2020. [Online]. Available: https://doi.org/10.17226/25646

[6] National Aeronautics and Space Administration, "Advanced Air Mobility," Tech. Rep., 2020. [Online]. Available: https://www.nasa.gov/aam

[7] R. F. Vitalle, Y. Zhang, B. Normann, and N. Shen, *A Model for the Integration of UAM operations in and near Terminal Areas*. [Online]. Available: https://arc.aiaa.org/doi/abs/10.2514/6.2020-2864

[8] B. German, M. Daskilewicz, T. K. Hamilton, and M. M. Warren, *Cargo Delivery in by Passenger eVTOL Aircraft: A Case Study in the San Francisco Bay Area*. [Online]. Available: https://arc.aiaa.org/doi/abs/10.2514/6.2018-2006

[9] X. Yang and P. Wei, "Scalable multi-agent computational guidance with separation assurance for autonomous urban air mobility," *Journal of Guidance, Control, and Dynamics*, vol. 43, no. 8, pp. 1473–1486, 2020. [Online]. Available: https://doi.org/10.2514/1.G005000

[10] X. Yang and P. Wei, "Autonomous free flight operations in urban air mobility with computational guidance and collision avoidance," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–14, 2021.

[11] C. Bosson and T. A. Lauderdale, *Simulation Evaluations of an Autonomous Urban Air Mobility Network Management and Separation Service*. [Online]. Available: https://arc.aiaa.org/doi/abs/10.2514/6.2018-3365

[12] S. Verma, J. Keeler, T. E. Edwards, and V. Dulchinos, *Exploration of Near term Potential Routes and Procedures for Urban Air Mobility*. [Online]. Available: https://arc.aiaa.org/doi/abs/10.2514/6.2019-3624

[13] S. Tarafdar, M. Rimjha, N. Hinze, S. Hotle, and A. A. Trani, "Urban air mobility regional landing site feasibility and fare model analysis in the greater northern california region," in *2019 Integrated Communications, Navigation and Surveillance Conference (ICNS)*, 2019, pp. 1–11.

[14] W. B. Cotton and D. J. Wing, *Airborne Trajectory Management for Urban Air Mobility*. [Online]. Available: https://arc.aiaa.org/doi/abs/10.2514/6.2018-3674

[15] J. van den Berg, S. J. Guy, M. Lin, and D. Manocha, "Reciprocal n-body collision avoidance," in *Robotics Research*, C. Pradalier, R. Siegwart, and G. Hirzinger, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 3–19.

[16] S. M. LaValle, "Rapidly-exploring random trees : A new tool for path planning," *The annual research report*, 1998.

[17] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011. [Online]. Available: https://doi.org/10.1177/0278364911406761

[18] J. R. Bertram, "Applying fastmdp to complex aerospace-related problems," Ph.D. dissertation, Iowa State University, 2022.

[19] J. A. Marshall, W. Sun, and A. L'Afflitto, "A survey of guidance, navigation, and control systems for autonomous multi-rotor small unmanned aerial systems," *Annual Reviews in Control*, vol. 52, pp. 390–427, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1367578821000882

[20] S. Koo, S. Kim, and J. Suk, "Model predictive control for uav automatic landing on moving carrier deck with heave motion," *IFAC-PapersOnLine*, vol. 48, no. 5, pp. 59–64, 2015, 3rd IFAC Workshop on Multivehicle Systems. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S240589631500703X
[21] I. Prodan, S. Olaru, R. Bencatel, J. Borges de Sousa, C. Stoica, and S.-I. Niculescu, "Receding horizon flight control for trajectory tracking of autonomous aerial vehicles," *Control Engineering Practice*, vol. 21, no. 10, pp. 1334–1349, 2013. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0967066113001020
[22] R. E. Bellman, *Dynamic Programming*. Princeton University Press, 1957.
[23] R. Bellman, "A markovian decision process," *Journal of Mathematics and Mechanics*, vol. 6, no. 5, pp. 679–684, 1957. [Online]. Available: http://www.jstor.org/stable/24900506
[24] E. Aldao, L. M. González-deSantos, H. Michinel, and H. González-Jorge, "Uav obstacle avoidance algorithm to navigate in dynamic building environments," *Drones*, vol. 6, no. 1, 2022. [Online]. Available: https://www.mdpi.com/2504-446X/6/1/16
[25] H. Freimuth and M. König, "Planning and executing construction inspections with unmanned aerial vehicles," *Automation in Construction*, vol. 96, pp. 540–553, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S092658051730290X
[26] Y. Dong, E. Camci, and E. Kayacan, "Faster rrt-based nonholonomic path planning in 2d building environments using skeleton-constrained path biasing," *Journal of Intelligent & Robotic Systems*, vol. 89, no. 3, pp. 387–401, 2018.
[27] J. A. Marshall, R. B. Anderson, W.-Y. Chien, E. N. Johnson, and A. L'Afflitto, "A guidance system for tactical autonomous unmanned aerial vehicles," *Journal of Intelligent & Robotic Systems*, vol. 103, no. 4, p. 71, Nov 2021. [Online]. Available: https://doi.org/10.1007/s10846-021-01526-8
[28] J. R. Bertram, "A new solution for markov decision processes and its aerospace applications," *Graduate Theses and Dissertations. 17832.*, 2020. [Online]. Available: https://lib.dr.iastate.edu/etd/17832
[29] J. Bertram, X. Yang, M. W. Brittain, and P. Wei, *Online Flight Planner with Dynamic Obstacles for Urban Air Mobility*. AIAA Aviation 2019 Forum, 2019. [Online]. Available: https://arc.aiaa.org/doi/abs/10.2514/6.2019-3625
[30] J. Bertram and P. Wei, *Distributed Computational Guidance for High-Density Urban Air Mobility with Cooperative and Non-Cooperative Collision Avoidance*. AIAA Scitech 2020 Forum, 2020. [Online]. Available: https://arc.aiaa.org/doi/abs/10.2514/6.2020-1371
[31] ——, *An Efficient Algorithm for Self-Organized Terminal Arrival in Urban Air Mobility*. AIAA Scitech 2020 Forum, 2020. [Online]. Available: https://arc.aiaa.org/doi/abs/10.2514/6.2020-0660
[32] ——, *An Efficient Algorithm for Multiple-Pursuer-Multiple-Evader Pursuit/Evasion Game*. AIAA Scitech 2021 Forum, 2021. [Online]. Available: https://arc.aiaa.org/doi/abs/10.2514/6.2021-1862
[33] J. Bertram, P. Wei, and J. Zambreno, *Scalable FastMDP for Pre-departure Airspace Reservation and Strategic De-conflict*. AIAA Scitech 2021 Forum, 2021. [Online]. Available: https://arc.aiaa.org/doi/abs/10.2514/6.2021-0779
[34] ——, "A fast markov decision process-based algorithm for collision avoidance in urban air mobility," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–14, 2022.
[35] City of New York. (2021) NYC 3D Model By Community District, MN CD 01 district. [Online]. Available: https://www1.nyc.gov/site/planning/data-maps/open-data/dwn-nyc-3d-model-download.page
[36] O. Aichholzer and F. Aurenhammer, "Straight skeletons for general polygonal figures in the plane," in *International computing and combinatorics conference*. Springer, 1996, pp. 117–126.
[37] M. Rezanejad, B. Samari, I. Rekleitis, K. Siddiqi, and G. Dudek, "Robust environment mapping using flux skeletons," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2015, pp. 5700–5705.
[38] P. J. Vermeer, "Medial axis transform to boundary representation conversion," Ph.D. dissertation, Purdue University, 1994.
[39] M. Wan, F. Dachille, and A. Kaufman, "Distance-field based skeletons for virtual navigation," in *Proceedings Visualization, 2001. VIS'01.* IEEE, 2001, pp. 239–560.
[40] N. D. Cornea, D. Silver, and P. Min, "Curve-skeleton properties, applications, and algorithms," *IEEE Transactions on visualization and computer graphics*, vol. 13, no. 3, p. 530, 2007.
[41] The CGAL Project, *CGAL User and Reference Manual*, 5.4 ed. CGAL Editorial Board, 2022. [Online]. Available: https://doc.cgal.org/5.4/Manual/packages.html

Josh Bertram Josh Bertram is a PhD candidate at Iowa State University and a Machine Learning / Artificial Intelligence researcher at the Johns Hopkins University Applied Physics Lab. Previously, he completed his Masters at Iowa State University and focused on Markov Decision Processes and their applications to aerospace problems. Josh worked at Collins Aerospace for 16 years in real-time embedded systems and as a Machine Learning / Artificial Intelligence engineer. Josh's interests are in the intersection of embedded computing, avionics, and artificial intelligence where size, weight, and power constraints restrict the types of algorithms that can be employed.

Joseph Zambreno Joseph A. Zambreno has been with the Department of Electrical and Computer Engineering at Iowa State University since 2006, where he is currently a Professor, Associate Chair, and director of the Reconfigurable Computing Lab (RCL). Prior to joining ISU he was at Northwestern University in Evanston, IL, where he graduated with his Ph.D. degree in Electrical and Computer Engineering in 2006, his M.S. degree in Electrical and Computer Engineering in 2002, and his B.S. degree summa cum laude in Computer Engineering in 2001. While at Northwestern University, Dr. Zambreno was a recipient of a National Science Foundation (NSF) Graduate Research Fellowship, a Northwestern University Graduate School Fellowship, a Walter P. Murphy Fellowship, and the EECS department Best Dissertation Award for his Ph.D. dissertation titled "Compiler and Architectural Approaches to Software Protection and Security." He is a recent recipient of the NSF CAREER award (2012), as well as the ISU award for Early Achievement in Teaching (2012), the College of Engineering Outstanding Achievement in Teaching Award (2019), and the ECpE department's Warren B. Boast undergraduate teaching award (2009, 2011, 2016).

Peng Wei Peng Wei is an assistant professor in the Department of Mechanical and Aerospace Engineering at the George Washington University, with courtesy appointments at Electrical and Computer Engineering Department and Computer Science Department. By contributing to the intersection of control, optimization, machine learning, and artificial intelligence, he develops autonomy and decision support tools for aeronautics, aviation and aerial robotics. His current focus is on safety, efficiency, and scalability of decision making systems in complex, uncertain and dynamic environments. Recent applications include: Air Traffic Control/Management (ATC/M), Airline Operations, UAS Traffic Management (UTM), eVTOL Urban Air Mobility (UAM) and Autonomous Drone Racing (ADR). He is leading the Intelligent Aerospace Systems Lab (IASL). He is an associate editor for AIAA Journal of Aerospace Information Systems. He received his Ph.D. degree in Aerospace Engineering from Purdue University in 2013.