Dynamic Unmanned Aircraft System Traffic Volume Reservation Based on Multi-Scale A* Algorithm

Jun Xiang*, Victor Amaya† Jun Chen‡
San Diego State University, San Diego, CA 92182

With the recent advancements in unmanned aircraft system (UAS) technology, unmanned aerial vehicles (UAVs) are widely used or proposed to carry out various daily tasks in the low altitude airspace, such as delivery and urban air mobility. In order to safely integrate the UAS traffic into the congested airspace in the urban area, the current UAS traffic management system proposed by NASA will reserve a static traffic volume for the whole planned trajectory, which is safe but not efficient. In this paper, we propose a dynamic traffic volume reservation method for the UAS traffic management system based on a multi-scale A* algorithm. The planning airspace is represented as a multi-resolution grid, where the resolution will get coarser as the distance to the planning drone gets larger. Therefore, each drone will only need to reserve a temporary traffic volume along the finest flight path in its local area, which helps release the airspace back to others on the far side. Moreover, the multi-scale A* can run nearly in real-time due to a much smaller search space, which enables dynamically rolling planning to consider updated information. The presented numerical results support the advantages of the proposed approach.

I. Introduction

Companies such as Airbus and Amazon are pioneers to a future, where Unmanned Aircraft Systems (UAS) will be the predominant technology in the skies that will assist humans in their everyday lives. Airbus' Urban Air Mobility (UAM) program wants to take their four-seat multi-copter vehicle, named CityAirbus, to the sky and have it function as a flying cab. While Amazon wants to improve its delivery times by having unmanned drones deliver packages as opposed to a person. This arises multiple safety concerns, as air traffic increases there is a higher chance of collisions to occur when sharing airspace with other aircraft. It is a complex topic to discuss how to incorporate manned and unmanned aircraft in the shared airspace. Currently, UAS, such as Unmanned Air Vehicle (UAV), man-controlled drones, and other smaller aircraft must operate under certain airspace restrictions such as a 400 ft. altitude limit, and be kept exclusively under a class G airspace. There have been various ideas on how to incorporate UAS into other airspaces that contain manned aircraft [1, 2]. The FAA, Federal Aviation Administration, developed a concept of operations for Unmanned Aircraft System Traffic Management (UTM), where a UAS will be notified, before its launch, of any conflicting scheduling with other aircraft. UTM applies strategic de-confliction by either spatial separation, temporal separation, or a combination of both to ensure no collision occurs [3].

As it is shown in the Figure 1, the current UTM system proposed by NASA and FAA will reserve a static traffic volume for the whole planned trajectory, which is safe but not efficient. The fact is that there is no need to reserve the traffic volume in the far away area relative to the UAV's current location. To overcome this limitation, the procedure of dynamically updating the reserved traffic volume could be a good option. While there have been multiple methods to create a dynamically updating flight path, they often tend to be very time consuming. To improve the algorithm's efficiency, this paper proposes using a Multi-Scale A* (MSA*) algorithm that can dynamically reserve the volume of local airspace for each UAV in this system, which efficiently provides a method to integrate UAVs into the airspace with urban settings.

The benefits of using this MSA* algorithm have two-folds. Firstly, the planning airspace are represented as a multi-resolution grid, where the resolution will get coarser as the distance to the planning UAV gets larger. Therefore, each UAV will only need to reserve a temporary traffic volume along a finest flight path in its local area, which helps free the airspace in the far side. Secondly, the multi-scale A* can run nearly in real-time due to a much smaller search space, which enables dynamically rolling planning to consider updated information.

The paper's structure is as follows. Section II discusses related work; Section III formulates the problem; Section IV explains the idea of a MSA* algorithm and how to use the MSA* for dynamic UAS traffic volume reservation; Section V shows the simulation results by comparing MSA* and original A*; Section VI concludes this paper.

*Ph.D. Student, Department of Aerospace Engineering, AIAA Student Member, jxiang9143@sdsu.edu
†Graduate Student, Department of Aerospace Engineering, AIAA Student Member, VAmaya@sdsu.edu
‡Assistant Professor, Department of Aerospace Engineering, AIAA Member, Jun.Chen@sdsu.edu

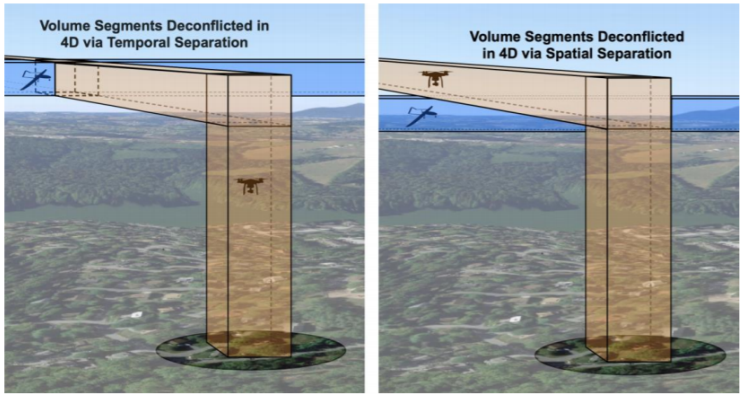


Figure 1: Pre-departure Static Airspace Volume Reservation [3]

II. Literature Review

Urban Air Mobility (UAM) and Unmanned Aircraft Systems (UAS) are the next step in increasing business productivity and improving peoples' time efficiency. Many concepts have been brought up concerning the integration of urban air mobility airspace sectors. Thipphavong, D. [1] describes the emergence of UAM concepts of operation such as take off and landing areas, the occupied airspace, the communication and surveillance needed to traverse the occupied airspace in an urban setting and the potential hazards that aircrafts may encounter. With an increase in air space traffic density, there needs to be a method to keep all these aircraft from colliding, and one such method is through reserving airspace volume [1, 3].

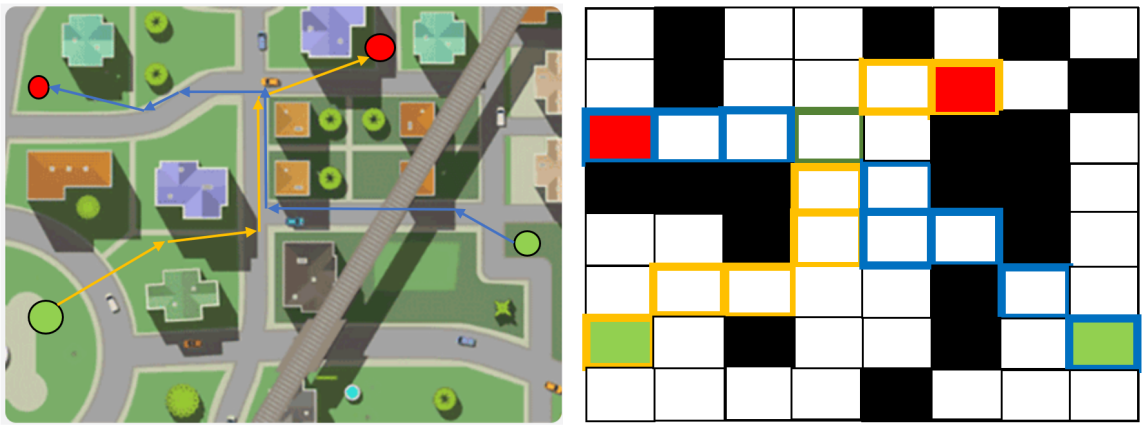
The Federal Aviation Administration (FAA) has been working on a similar concept of operations for managing air traffic of Unmanned Aircraft Systems [3]. As shown in Fig. 1, this concept uses an airspace reservation method in the pre-departure stage of an aircraft. In this deconflicting method for aircraft, they suggest temporal separation, which delays the time of the aircraft's departure to avoid colliding with another aircraft in the shared airspace, spatial separation, which changes the airspace reservation of the departing aircraft to avoid colliding with an aircraft in its previous reserved airspace, or a combination of both. Most airspace reservation methods are designed around a static environment where in the reserved airspace a flight path will generated from the initial state to the goal state [4, 5]. Bertram, J, and Zambreno, J [6] employed the use of an algorithm known as FastMDP to assist in the de-conflicting of aircraft in a dense airspace environment and reserving the airspace for an aircraft prior to its departure. Although this concept functions properly in de-conflicting aircraft by reserving the airspace, since it is done during the pre-departure stage only, then the rest of the airspace stays reserved for that aircraft solely.

To further improve UAS path planning to avoid any temporal separation, various path planning algorithms have been employed with a collision avoidance emphasize to allow multiple UAVs with distinct functions to coexist and operate in the same airspace. These algorithms allow for UAS to create a path in highly dense airspace following an optimal path based on risk factors that can be referred to as other UAS and buildings. Biology inspired algorithms have proven successful in creating multiple optimal paths in a designated space such as Ant Colony Optimization [7], Artificial Bee Colony [8], and Cockroach Swarm Optimization [9]. However, various issues have been discovered with these types of path planning methods that make them difficult to apply in UAM scenarios. One issue is that along with the increase in iterations, the convergence speed of the algorithm correspondingly changes rapidly, which makes them impractical in realistic scenarios [10]. A more suitable method for path planning with unmanned airspace management can be found in graph-based algorithms such as Dijkstra's algorithm [11], Bellman-Ford algorithm [12], A-Star algorithm (A*) [13], and rapid exploring random tree (RRT) [14]. The first three share a similar attribute while the RRT, in comparison to A-star, creates longer paths making it less efficient when calculating for the shortest path from the initial to a goal state.

As stated previously, a commonly used graph-based algorithm to search for the optimal path is the A* algorithm. Primatesta, S,Guglieri, G, and Alessandro, R. [15] implemented the A* algorithm over urban areas. This was further enhanced through the implementation of a Conflict-Free A* (CFA) algorithm in a 3D environment created through an AirMatrix that decomposes an urban airspace into blocks, along with a point mass model that is set on the planned path to avoid static and dynamic obstacles in the created environment [16]. While applying the A* algorithm allows to calculate for the optimal path, or path of least resistance, it comes with the downside of long processing times. A solution to help reduce the processing time was sought out by Zhang, N., Zhang, M. and Low, K. [17]. They created

a real time urban environment that is divided into a 3D voxel world to run an efficient path finding algorithm , Jump Point Search (JPS) [18]. JPS can greatly reduces the computing time for the path-finding algorithm by reducing the search points calculated in the 3D environment to form the optimal path while the A* algorithm computes more points that are ultimately ignored by the algorithm. A similar solution to reduce computing time while using the A* algorithm is introduced in Kambhampati and Davis [19]. A hierarchical refinement scheme of an environments search space is abstracted in a quad-tree, where the risk region is given the coarsest resolution and excluded from the refinement to reduce the search space. This is done with the purpose of obtaining a multiscale version of the A* algorithm which allows for preprocessing of the environment in a top to bottom dyadic decomposition which divides the problem so that it can be solved at a much smaller and quicker rate and then be merged, therefore creating a much faster and efficient A* algorithm. The similar idea was enhanced by Lim,J. and Tsiotras, P. [20] by creating a multi-scale grid to process an optimal path across a risk map efficiently.

III. Problem Statement



a) Urban Airspace Map **b) Urban Airspace Grid**

Figure 2: UAS traffic volume reservation based grid map concept

The key to the success of the interaction of UAS in urban environments relies on developing an real-time air traffic management system that can handle dynamical scenarios with dense air traffic. To assist in this development, this work will consider a 2D urban airspace as shown in Fig. 2a. By applying the grid world idea, a two dimensional array of blocks is created in this urban airspace, as shown in Fig. 2b. The block sets is $B = \{b_1, b_2, ..., b_I\}$, I is the total number of blocks in the set B, which depends the size of the considered airspace and the size of each unit block.

Based on the cities landscape such as buildings, as static obstacles, and other aircraft, as dynamic obstacles, a unit block $b_i \in B$ has a time-varying binary risk value $r_{i,t}$ that is defined as follows:

$$r_{i,t} = R(b_i, t) = \begin{cases} 1 & \text{if block } i \text{ is occupied at time } t \\ 0 & \text{if block } i \text{ is available at time } t \end{cases} \tag{1}$$

where $R(b_i, t) : \mathbf{B} \longrightarrow r \in [0, 1]$ is the risk value function. Therefore, for static obstacles such as buildings, the risk value for the occupied block is always 1. Occupied blocks will be denoted by darker shaded nodes as the nodes depicted in Fig. 2b.

The UAS, in this scenario as a drone, has the purpose to travel from its initial location $(x, y)_{Initial}$ in the graphed map to a set destination $(x, y)_{Goal}$. The drone will be set to follow a conflict free optimized path $P = \{b_1, b_2, ..., b_k\}$, which is created to avoid both static and dynamic obstacles, while search the path with the lowest risk values in the given airspace. $b_{Initial} = b_1$ and $b_{Goal} = b_k$ refer to the blocks where the path will begin and end respectively, they are denoted by:

$$b_{Initial} = \arg\min_b \|(x, y)_b, (x, y)_{Initial}\| \tag{2}$$

$$b_{Goal} = \arg\min_b \|(x, y)_b, (x, y)_{Goal}\| \tag{3}$$

where x and y are the coordinates placed at the center point of the block b in the 2D airspace.

In the grid map, conflict free flight is guaranteed by avoiding unitizing the same block at the same time. Each drone will follow the reserved airspace designated by the path algorithm, as shown in Fig. 2b. Both the yellow path for drone 1 and blue path for drone 2 are generated away from higher risk areas, where $r_{i,t} > 0$. Time displacement is often taken into consideration when generating the path, so when the drones' path align they will avoid intercepting each other. The time displacement with static path works well when the air traffic is sparse. To improve the airspace utility, a dynamic method based on a multi-scale graph will be discussed in the following sections.

IV. Multi-Scale A* Algorithm

This Multi-Scale A* (MSA*) method is inspired by the work in [20], where a multi-resolution framework is constructed to assist path planning with multiple agents, where each agent's abstract path is improved through the information provided by the other agents. To further increase the efficiency of the algorithm a fine resolution graph representation is constructed around the agents to focus the processing power only on the optimal path it will take. While a coarse resolution representation is kept farther away from the agents to reduce the dimensionality of the search space, which in relation reduces the processing power and time to run the algorithm.

The generated environment will use a multi-scale grid, which is constructed based on the agent's current location. Therefore, the multi-scale grid map is varying for different agents or the same agent at different locations. The purpose for using a multi-scale grid is that a lot more simulations can be run at a quicker succession due to the fact that the MSA* can nearly run in real-time due to its smaller search-space. This allows the collection of new environmental data more efficiently. In this work, the type of environmental data includes how the flight path of the UAS is affected as it changes its reserved airspace volume when encountering any dynamic or static obstacles, and the likelihood of collisions occurring as air traffic density increases.

A. Multi-resolution Graph

In this paper, we consider the environment Ω as a block based full-resolution graph $G_f = (\mathbf{B}, \mathbf{E})$, where $\mathbf{B}$ is the block set, $\mathbf{E}$ is the edge set. $b \in \mathbf{B}$ is a unit block, where $r(b)$ is the risk value given by the equation (1), and $o(b) = (x, y)_b$ is the center position of the block. This full-resolution graph $G_f = (\mathbf{B}, \mathbf{E})$ can be abstracted as a full quadtree $\mathbf{T} = (\mathbf{N})$, where $\mathbf{N}$ is the node set. The root node of the quadtree, which is the parent for all other nodes , represents all the blocks in the block graph.

The quadtree node $n_{l,o} \in \mathbf{N}$ represents a block set $B_n \subset B$, where $l \in \mathbf{Z}^+$ is the node level in quadtree, $o(n) = (x, y)_o$ is the center position of the node. The total size of the represented block set is $2^l * 2^l$. Each leaf node ($l = 0$) represents all the information in a 1x1 block(unit block) such as center, position, and risk. Each non-leaf node($l > 0$) represents its 4 direct children at level $l - 1$, and its risk and center position are the average risks and center position of its 4 direct children. Each child represents 1/4 of the blocks its direct parent represents. For example, in Fig. 3(c), the red node represents four unit blocks and the green node only represents one unit block.

In practice, some nodes in the full quadtree are not necessary. An agent a_i can cut off all the undesired nodes (select desired nodes) and generate a partial tree. The partial tree builds a non-empty multi resolution graph $G(\mathbf{N}_i, \mathbf{E}_i)$ where $\mathbf{N}_i$ is the set of the selected nodes and $\mathbf{E}_i$ is the corresponding edge set. The current agent selects nodes following the hierarchical system and goes down the set nodes in each depth layer. When a node with children is selected by the agent, then its children node and parent node will be excluded from $\mathbf{N}_i$. This ensures that the selected node in $\mathbf{N}_i$ completely covers the environment Ω without overlap. $\mathbf{N}_i$ is a subset of the full node set $\mathbf{N}$. The node $n_{l,o}$ will be selected if the straight line distance between the center point o of the node, and the current location p of the agent, is greater than the 2^l multiplied by α, which is a parameter set by the user. Otherwise, its children will be selected instead. The larger the α is, the more nodes will be selected.

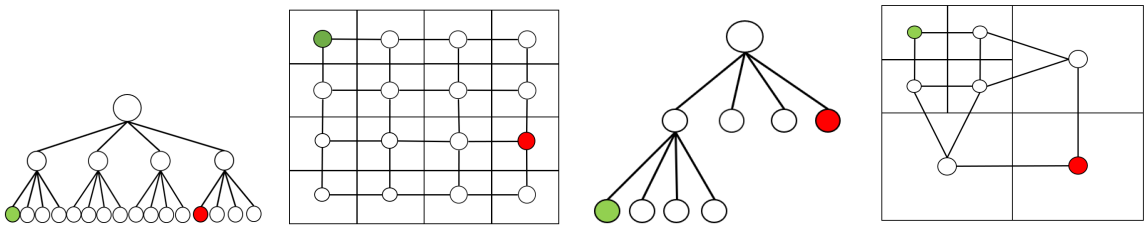
$$\|o - p\| > \alpha 2^l \tag{4}$$

In this paper, all edge cost between two nodes will be same as the straight line distance between their center points.

$$E(n, n') = \|o(n) - o(n')\| \tag{5}$$

B. Graph Construction

In Fig. 3, the formulation for multi-scale graph $G(\mathbf{N}_i, \mathbf{E}_i)$ begins with: a) The block-based world encoded in a full quadtree, $\mathbf{T}$, which shows the parent nodes and children nodes with the initial state, shown as the green node, and the goal state, shown as the red node. b) From the full tree, $\mathbf{T}$, the graph in full resolution is constructed using all the leaf nodes from the created tree. c) When the agent is activated, the multi-scale graph $\mathbf{G}$ selects the nodes $\mathbf{N}_i$ from the

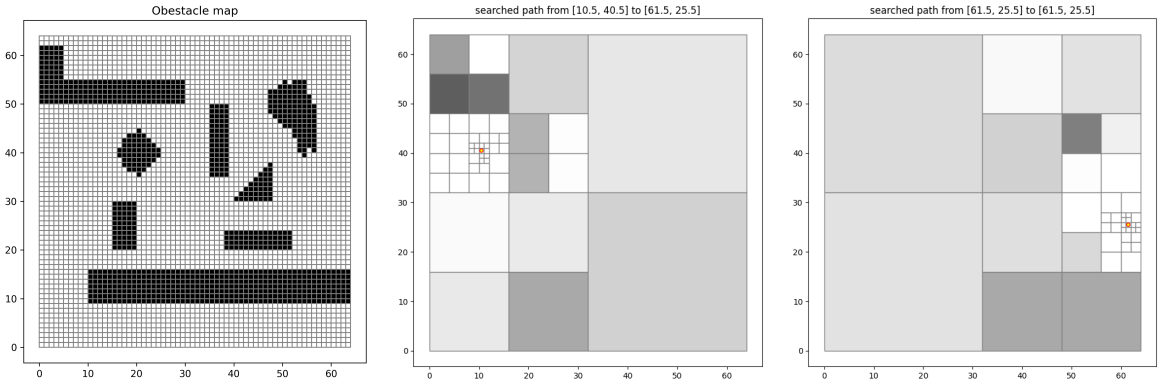


a) full quadtree b) full-resolution map c) partial tree d) multi-resolution map

Figure 3: Illustration of the multi-scale graph construction

full tree **T** based on the rule introduced in section IV A. d) The new graph is now created where the initial state still locate the block represented by the leaf node(shown as the green node), while some other blocks are represented by the higher-level nodes, such as the block the goal state(shown as the red node) located at.

Based on this idea, a multi-scale grid example with obstacles is shown in Fig. 4. The original obstacle map with full scale grid (64×64) is shown in Fig. 4a. When the agent is near the origin as shown in Fig 4b, the far away area has a coarser resolution and the local area still maintains the finest resolution. Thus the search space is greatly reduced to 49 nodes. Similarly, the multi-scale grid map will update when the agent moves to a different area as shown in Fig. 4c, the search space is reduced to only 40 nodes. Moreover, the gray-scale for each node represents how many percent of the area is occupied by obstacles. In other words, the risk is higher when it is darker.



a) Full Scale Grid with Obstacles b) Multi-Scale Grid: near origin c) Multi-Scale Grid: near destination

Figure 4: Multi-scale grid example with obstacles

C. Path-Planning Problem

Based on the multi-scale graph **G**, a path $\mathbf{P} = \{n_1, n_2, ..., n_k\}$ on **G** is an set of nodes n_i, where any two consecutive nodes are connected. A variance of A* algorithm is used to search a path on this multi-scale graph, with a given start node n_1 and end node n_k. For the A* algorithm, the cost value g, which is used as the cost of the path moving from the current node n to the immediate next node point n', is equal to the sum of the edge cost and the risk of the next node:

$$g(n, n') = E(n, n') + r(n') \tag{6}$$

The heuristic value h for each node is set to the straight line distance between the current node to the destination n_k.

D. Dynamic UAS Traffic Volume Reservation based on MSA*

In A* based path $P = \{n_1, n_2, ..., n_k\}$ on a multi-scale graph **G**, only the first n_d connected nodes represent the finest 1×1 unit block. The other nodes represent multiple blocks on the far side. We will only reserve those n_d leaf nodes representing the finest unit blocks as the traffic volume reservation for agent a_i, and release the non-leaf nodes

representing the large area for others. Thus, the traffic volume reservation $\mathcal{D}_i$ is updated as $\mathcal{D}_i = \{n_1, n_2, ..., n_{n_d}\}$. The main procedure of the dynamic UAS traffic volume reservation based on MSA* is shown in Algorithm 1.

Algorithm 1: DYNAMIC TRAFFIC VOLUME RESERVATION

Initial traffic volume reservation $\mathcal{D}_i = \emptyset$;
Initial location $p = (x, y)_{initial}$ and goal location $g = (x, y)_{Goal}$;
while $\|p - g\| > 0$ **do**
    Construct multi-scale graph **G** based on the current location p and new messages on obstacles;
    Use Eqns (2 and 3) to find $b_{Initial}$ and b_{Goal} on the graph **G** and their corresponding nodes n_1 and n_k;
    Run MSA* to find the optimal path: $P = \{n_1, n_2, ..., n_k\}$;
    Get the finest flight path $\{n_1, n_2, ..., n_{n_d}\} \subset P$ in local area of p;
    Update the traffic volume reservation $\mathcal{D}_i = \{n_1, n_2, ..., n_{n_d}\}$;
    Send out the reserved traffic volume $\mathcal{D}_i$ as messages;
    Update the location $p = o(n_2) = (x, y)_o$;

V. Results

In this section, a series of numerical study are conducted to demonstrate the feasibility and efficiency of our proposed MSA* algorithm.

A. Experiment Setting

To test the solution quality of the proposed algorithm, a simulated drone flight environmental grid map for urban areas is built, shown as Fig.5. The unit block size is 1 x 1. In this map, the white block represents the available area that the UAV can pass, and the colored block represents the block is occupied and the UAV should avoid it. In the dynamic map, there are obstacles that occupy random available blocks for a certain time. The conflict occurs when a UAV enters an occupied block in this paper. The UAV can only move to the adjacent blocks from a block in one step. All the h value for search is set to the straight line distance between the block and destination. All desired areas used in the experiment are generated by cropping this map. In this section, all the tests were implemented in Python 3.8 and were run on an Intel CPU i7-8700k @3.70GHz with 16GB RAM.

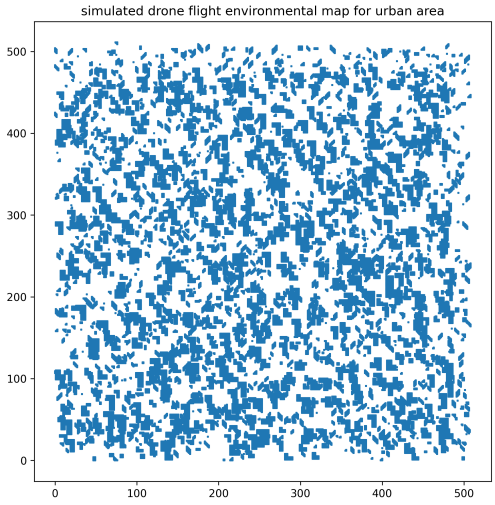


Figure 5: Urban area map

B. A Simple Case with Static Obstacles

The Fig. 6 shows a simple example of A* path planning on a 64 X 64 map. The white block is the available block and the black block is the obstacle. The UAV is currently located at block (2.5, 2.5) and its target is block (60.5, 60.5). The block with the red frame is the path generated by the original A* algorithm with A* considering every single block in

this map. It cost 26.54 seconds, A* algorithm successfully found an available path to the block and the path is 73-step long. The Fig. 7 shows a simple example of our MSA* method on the same map. The intensity of gray represents the number of obstacles. For the blocks near the UAV, the MSA* considers all of them. For the blocks that are far, the MSA* considers them as one big block by representing them with one node. The further the blocks are, the less and larger block is considered. At step 1, MSA* considers 28 blocks located at the bottom left side of the map and only 3 blocks from the other sides of the map. the MSA* found a 7-step path to the big block the target is located at. The MSA* places the agent on the first block of the path, then runs another path planning. The MSA* runs place-planning until the agent is placed on the target block. Although MSA* runs planning many times, each planning costs much less time, around 0.006 seconds. It cost only 0.45 seconds for MSA* algorithm to find the path to the exact target block. The length of the path is 75 steps, as shown in Table 1.

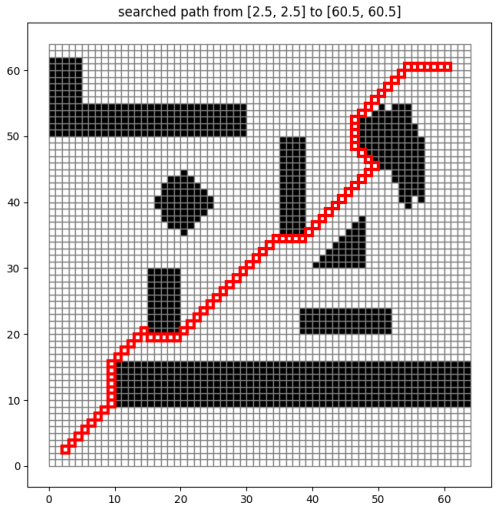


Figure 6: Path planning results with original A*

Table 1: Comparison of MSA* and A*

Method	Total Runtime (s)	Travel Distance (steps)	Number of nodes
A*	26.54	73	64× 64
MSA*	0.45	75	less than 50

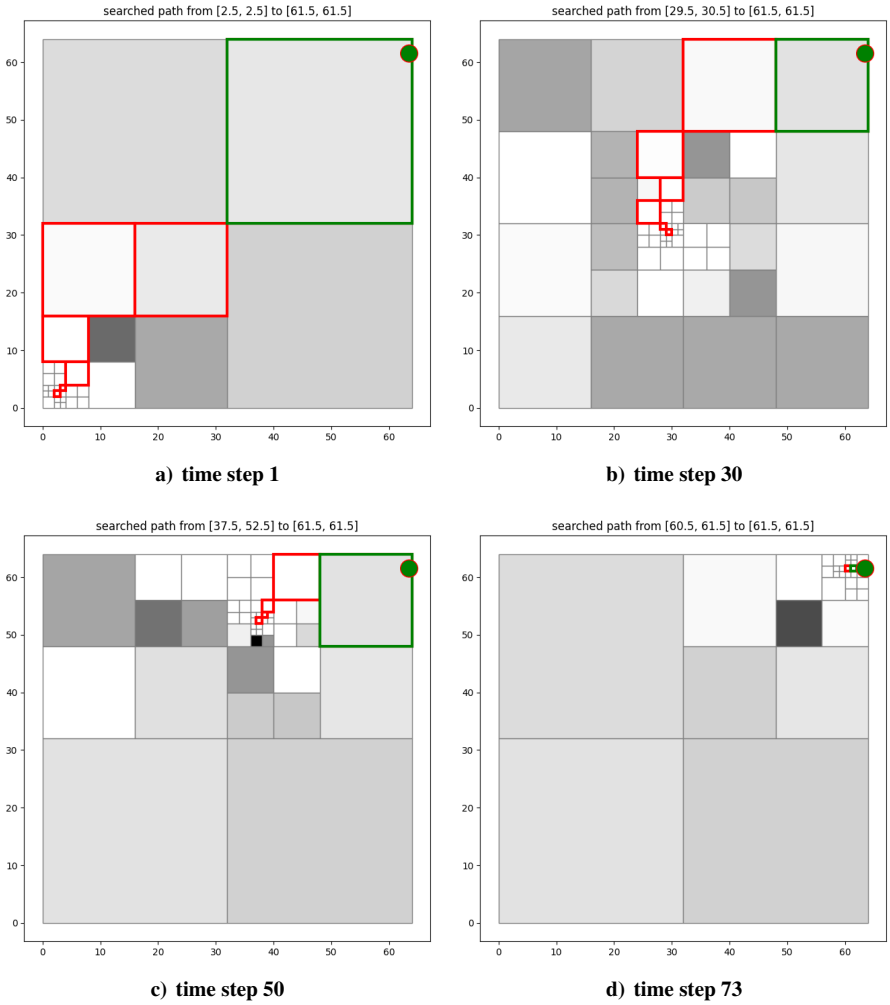
C. Parameter Sensitivity Analysis of MSA*

1. *Varying Map Size*

As shown in Fig. 8, the running time of MSA* increases moderately as increasing of map size, while the running time of A* increases dramatically. The mean running time of these two methods is the same when the map size is 16x16. However, when the map size is 128x128, the running time of A* is 211 times longer than MSA*. When map size is 256x256, the running time of A* is 400 times longer than MSA*. The results show that the MSA* has a large time efficiency advantage. As shown in Fig. 9, MSA* performs slightly worse than A* in finding the shortest path on every map.

2. *Varying Explore Distance*

As shown in Fig. 10, longer explore distance(bigger alpha) can decrease total travel distance significantly but increase running time a bit. when alpha is equal to 1, the mean travel distance is 801.2 and the total run time is 8.2. In contrast, when alpha is equal to 3, the mean travel distance is 268.2, which is already close to the original A*, and the average total run time is 97.2 seconds, which is only 2.6% of original A*. An appropriate explore distance can be chosen to meet the demand.



a) time step 1 **b) time step 30**

c) time step 50 **d) time step 73**

Figure 7: Case 1: MSA* from origin [2.5, 2.5] to goal [61.5, 61.5]

D. Dynamic Obstacles

MSA* is tested on the dynamic map. The dynamic map looks like the Fig. 11. The black block represents the static obstacle and the grey block represents the moving obstacle. The red block represents the reserved airspace at each step. The MSA* is able to find the path to the destination without any collision. As shown in Fig. 12, there is no sign that the path length will significantly increase when dynamic obstacles are involved. The results show that MSA* can easily deal with dynamic obstacles.

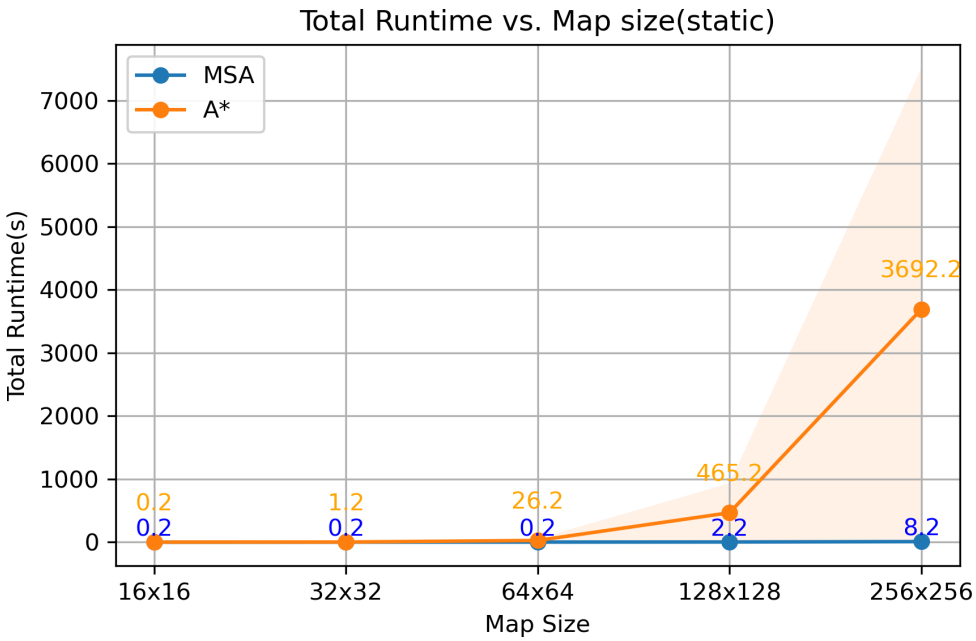


Figure 8: Time Cost Comparison between Original A* and MSA

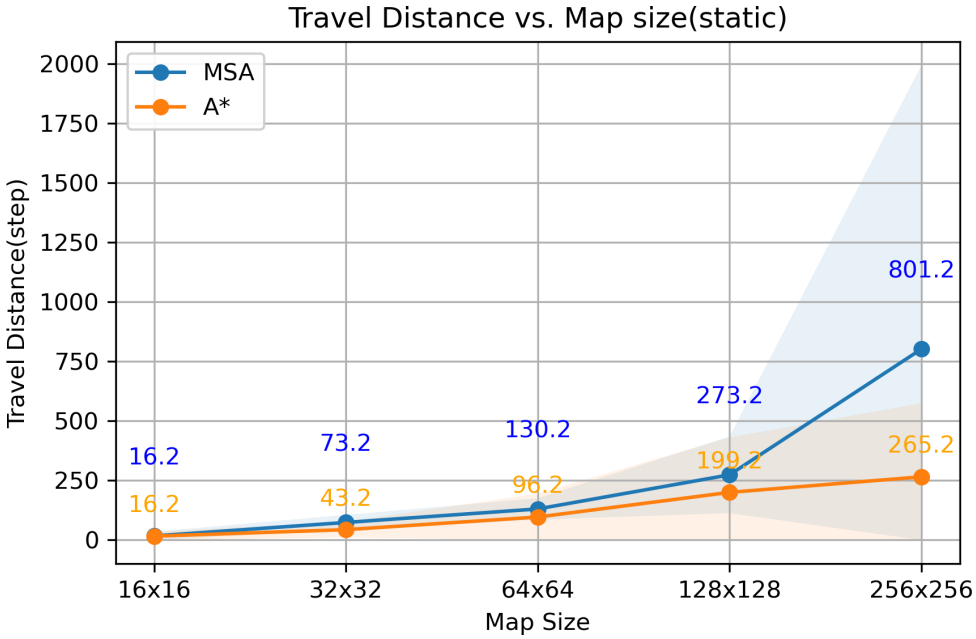


Figure 9: Path Length Comparison between Original A* and MSA

VI. Conclusion

In this paper, we propose a dynamic traffic volume reservation method for the UAS traffic management system based on a multi-scale A* algorithm. In the multi-scale map, a node can represent many unit blocks if those blocks are less important and a node may only represent one unit block if that block is close to the agent. The multi-scale map can effectively decrease the size of the node list for A* algorithm while keeping essential map information. According to our experiences, it is proved that the MSA* is able to run nearly in real-time without failing to reach its destination. MSA* can adapt to various tasks from fast reaction to cost-efficiency search. Our experiences also

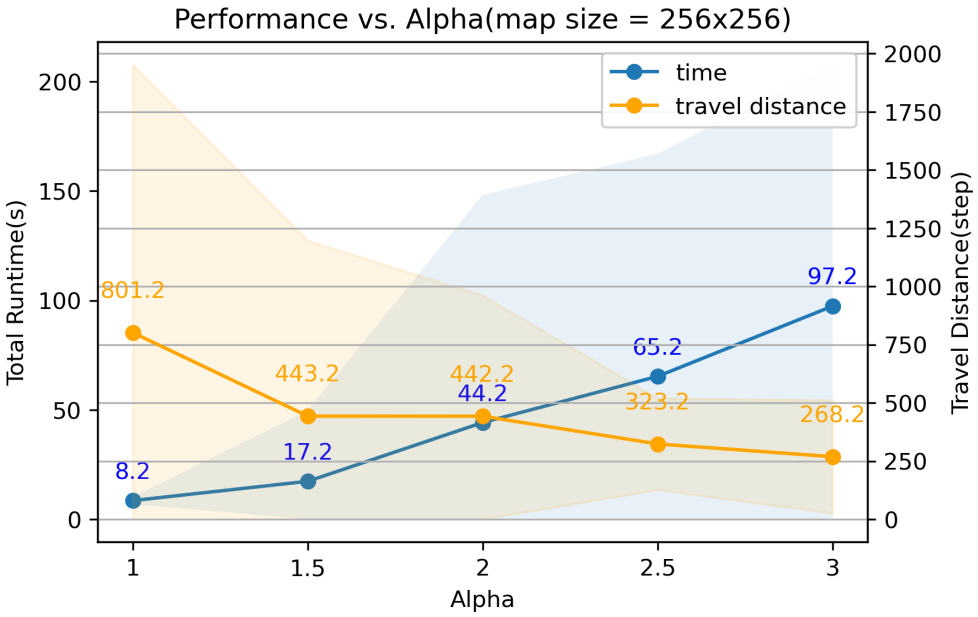


Figure 10: Performance comparison between different alpha

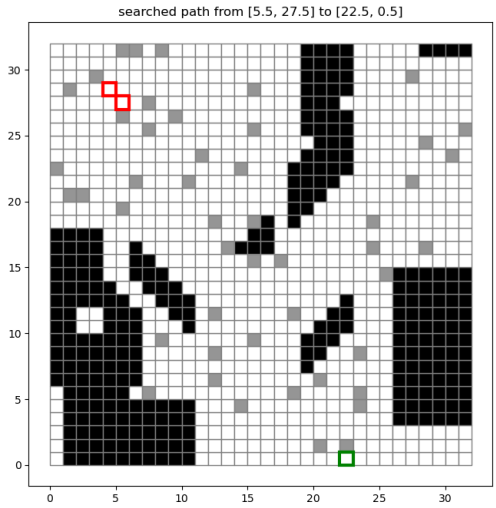


Figure 11: MSA* searching in a dynamic environment

prove that the MSA* can deal with the dynamic environment. In the future, the Multi-scale method can be applied to the BFS method other than A* due to small search depth. Furthermore, its dynamic UAS traffic volume reservation should be helpful in multi-agent path planning. We would also like to apply a Multi-scale search in a 3D or unknown environment.

Acknowledgment

We would like to thank the National Science Foundation (NSF) under Grants CMMI-2138612 for the support of this work.

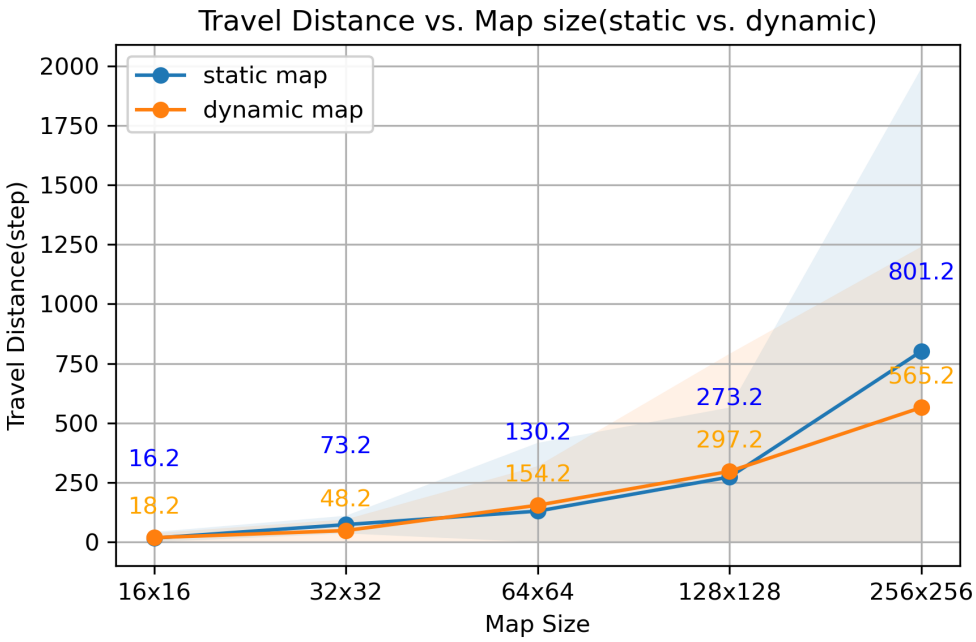


Figure 12: MSA* Time Cost Comparison on static and dynamic obstacle

References

[1] Thipphavong, D. P., Apaza, R., Barmore, B., Battiste, V., Burian, B., Dao, Q., Feary, M., Go, S., Goodrich, K. H., Homola, J., et al., "Urban air mobility airspace integration concepts and considerations," *2018 Aviation Technology, Integration, and Operations Conference*, 2018, p. 3676.

[2] Wu, P., Li, L., Xie, J., and Chen, J., "Probabilistically guaranteed path planning for safe urban air mobility using chance constrained rrt," *AIAA AVIATION 2020 FORUM*, 2020, p. 2914.

[3] FAA, "Unmanned Aircraft System Traffic Management," 2020.

[4] Wu, P., Xie, J., and Chen, J., "Safe path planning for unmanned aerial vehicle under location uncertainty," *2020 IEEE 16th International Conference on Control & Automation (ICCA)*, IEEE, 2020, pp. 342–347.

[5] Du, B., Chen, J., Sun, D., Manyam, S. G., and Casbeer, D. W., "UAV Trajectory Planning With Probabilistic Geo-Fence via Iterative Chance-Constrained Optimization," *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[6] Bertram, J., Wei, P., and Zambreno, J., "Scalable FastMDP for Pre-departure Airspace Reservation and Strategic De-conflict," *AIAA Scitech 2021 Forum*, 2021, p. 0779.

[7] Duan, H., Yu, Y., Zhang, X., and Shao, S., "Three-dimension path planning for UCAV using hybrid meta-heuristic ACO-DE algorithm," *Simulation Modelling Practice and Theory*, Vol. 18, No. 8, 2010, pp. 1104–1115.

[8] Lei, L. and Shiru, Q., "Path planning for unmanned air vehicles using an improved artificial bee colony algorithm," *Proceedings of the 31st Chinese control conference*, IEEE, 2012, pp. 2486–2491.

[9] Kwiecień, J. and Pasieka, M., "Cockroach swarm optimization algorithm for travel planning," *Entropy*, Vol. 19, No. 5, 2017, pp. 213.

[10] Mulani, M. and Desai, V. L., "Design and Implementation Issues in Ant Colony Optimization," *International Journal of Applied Engineering Research*, Vol. 13, No. 16, 2018, pp. 12877–12882.

[11] Dijkstra, E. W. et al., "A note on two problems in connexion with graphs," *Numerische mathematik*, Vol. 1, No. 1, 1959, pp. 269–271.

[12] Chong, Z. and Fan, Z., "Study on UAV path planning based on Bellman–Ford algorithm," *Journal of Projectiles, Rockets, Missiles and Guidance*, Vol. 5, 2007, pp. 077.

[13] Hart, P. E., Nilsson, N. J., and Raphael, B., "A formal basis for the heuristic determination of minimum cost paths," *IEEE transactions on Systems Science and Cybernetics*, Vol. 4, No. 2, 1968, pp. 100–107.

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

[15] Primatesta, S., Guglieri, G., and Rizzo, A., "A Risk-Aware Path Planning Strategy for UAVs in Urban Environments," *Journal of Intelligent & Robotic Systems*, Vol. 95, No. 2, 2019, pp. 629–643.

[16] Dai, W., Pang, B., and Low, K. H., "Conflict-free four-dimensional path planning for urban air mobility considering airspace occupancy," *Aerospace Science and Technology*, Vol. 119, 2021, pp. 107154.

[17] Zhang, N., Zhang, M., and Low, K. H., "3D path planning and real-time collision resolution of multirotor drone operations in complex urban low-altitude airspace," *Transportation Research Part C: Emerging Technologies*, Vol. 129, 2021, pp. 103123.

[18] Tanner, B., "Jump Point Search Analysis," *Florida State University. fsu. edu*, 2014.

[19] Kambhampati, S. and Davis, L., "Multiresolution path planning for mobile robots," *IEEE Journal on Robotics and Automation*, Vol. 2, No. 3, 1986, pp. 135–145.

[20] Lim, J. and Tsiotras, P., "MAMS-A*: Multi-Agent Multi-Scale A," *2020 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2020, pp. 5583–5589.