Small UAV Flight Planning in Urban Environments

Min Xue*
NASA Ames Research Center, Moffett Field, CA 94035

Melissa Wei†
Computer Science, Cornell University, Ithaca, NY 14850

This work proposes a fast algorithm for generating obstacle-free and wind-efficient flight paths at a constant above-ground-level altitude in urban environments, because a fast flight path planning algorithm is an essential function or service needed for enabling small Unmanned Aerial Vehicle (sUAV) to operate in urban environments within Class G airspace. The proposed method first converts the 3D path planning problem to a 2D problem by constructing an obstacle map at a given above-ground-level altitude. A quad-tree decomposition is then used to build a search space in terms of obstacle occupancy and wind difference. The wind cost of traveling through each cell is defined based on energy consumption under various wind conditions. A repulsive potential is also adopted to make sure the flight plans stay away from obstacles. The Theta* search algorithm, a variant of A* algorithm, is applied to mitigate the path angle change constraints introduced by grid-based graphs. With the Theta* and post smoothing techniques, an obstacle-free, wind efficient, and constant above-ground-level flight plan can be quickly generated for small UAV operations in urban environments while meeting the lateral path angle constraints. The results showed that the path planning algorithm is efficient and can be finished within several seconds. With a proper choice of wind coefficient, the proposed path planning algorithm outperforms the multiple-shooting trajectory optimization method even in an obstacle-free environment. With the flexibility of incorporating other geo-related costs and the efficiency in computation, the proposed algorithm shows potential for real-time flight path planning in complex urban environments.

I. Introduction

In order to accommodate soaring commercial interest in small Unmanned Aerial Vehicle (sUAV) operations, NASA has been leading the development of the Unmanned aerial system Traffic Management (UTM) system together with FAA and industry. Within the UTM architecture, besides basic functions or services like authentication, authorization, and strategic deconfliction, many other functions or services are also needed to enable large scale operations of small UAVs. UAV flight planning is one of them. The UTM concept requires that small UAVs operate in Class G airspace, which is typically below 400 feet Above Ground Level (AGL). This restriction, together with urban obstacles and winds, brings challenges to the UAV flight path planning task.

Flight path planning algorithms in aviation are typically used to find paths or trajectories that minimize fuel, flight time, and/or propagated noise [1]. Three groups of methods are usually applied to solve this type of problem. The first group typically includes trajectory optimization algorithms including *indirect methods* and *direct methods* [2]. The *indirect methods* use the calculus of variations or the Maximum Principle of Pontryagin. The *direct methods* transform the original optimal control problem into a nonlinear parameter optimization problem by using multiple shooting methods and direct collocation methods. This group of methods is normally used to find two dimensional trajectories that can minimize fuel and flight time [3], such as wind-optimal trajectories [4]. The second group of methods treats the trajectory optimization as a highly-constrained linear or nonlinear problem and solves them using generalized optimization techniques, such as *genetic algorithms*, *simulated annealing* [5], and *mixed-integer programming* [6]. The third group consists of path planning methods. They are typically used in robot motion planning [7] to avoid obstacles including the *roadmap method*, *potential field* method, *cell decomposition* method, *probabilistic roadmap* method, and the *rapidly-exploring random tree* [8, 9] method. In an obstacle-rich environment where the problem is highly constrained, the path planning methods are widely used as they can find a near optimal shortest path with much less computational time. There has been a lot of research in UAV planning using path planning methods lately [10]. Ramana et. al. [11] proposed an obstacle-avoidance path planning algorithm that also takes into account the turn radius and climb rate constraints of a UAV; Primatesta et. al. [12] developed a risk-aware path planning algorithm to minimize the

*Aerospace Research Engineer, Aviation Systems Division. Mail Stop 210-15. AIAA senior member.
†Student, Computer Science, Cornell University

risk to the population on the ground based on a risk-map constructed using probabilities of crashing, impact, and fatality. Chakrabarty et. al. [13] developed a real-time kinematic tree path planner to avoid obstacles when the small UAVs are aloft.

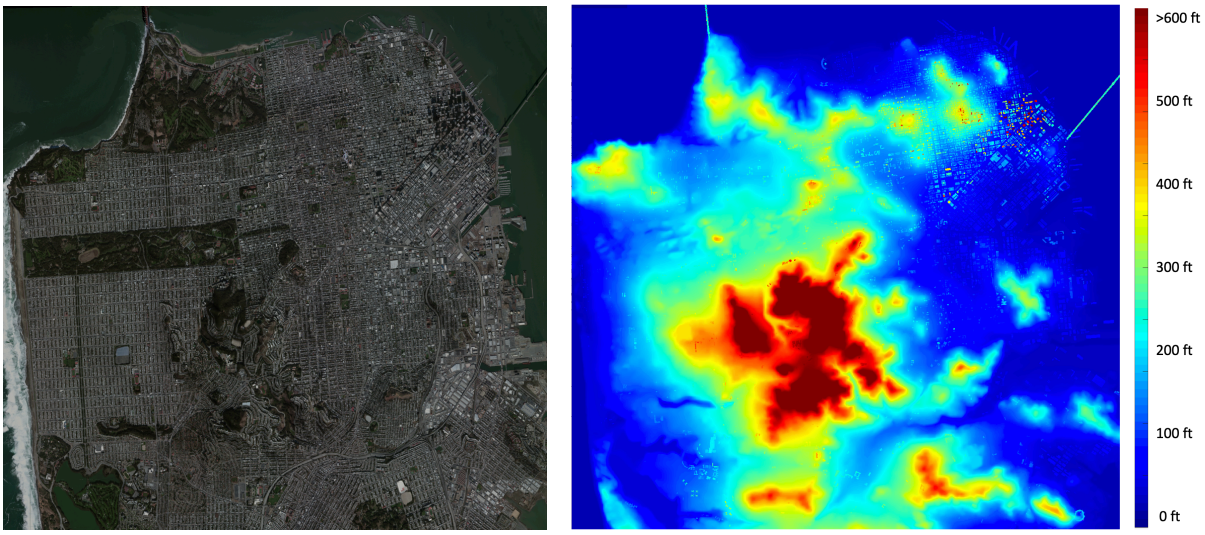
Since small UAVs are required to operate below AGL 400 feet, flight planning with a constant Median Sea Level (MSL) as in traditional aviation is not well suited for sUAV operations. An AGL-based flight plan becomes necessary for small UAV operations. This work proposes a path planning algorithm for generating obstacle-free and wind-efficient sUAV paths at a constant AGL in urban environments. By constructing obstacle maps at given AGL levels, this method converts a three dimensional AGL-based flight planning to a two dimensional problem. A quad-tree decomposition is then used to build the search space in terms of obstacle occupancy and wind difference. The wind cost of traveling through each cell is defined based on steady thrusts calculated for a given UAV model under various wind conditions, and a repulsive potential was adopted to make sure the flight plans are not too close to obstacles. With the Theta* search algorithm and post smoothing techniques, an obstacle-free, wind-efficient, and constant AGL flight plan can be quickly generated for small UAV operations in urban environments.

In this paper, Section II introduces the proposed flight planning algorithm including search graph construction, cost definition, search algorithm, and post smoothing. Section III presents results and performance of the flight path planning algorithm. Section IV concludes this work.

II. Flight Path Planning Algorithm

In order to generate an obstacle-free wind-efficient flight path with reasonable computational effort, an algorithm based on path planning techniques was developed. Using the terrain and wind data for San Francisco city as an example, this section presents the graph construction, cost definition, and search method used in this algorithm.

A. Search Graph Construction



(a) 3D City Model for San Francisco

(b) San Francisco Height Map

Fig. 1 San Francisco 3D city model and height map

A high-resolution 3D city model for the San Francisco city [14] is shown in Fig. 1(a). To capture details for visualization, triangles were used as basic units in the data set and each terrain object is composed of thousands of triangles. There are two types of objects for the terrain: buildings and land structures. Using this 3D model, a height map with a resolution of three feet by three feet was generated by extracting terrain height information from the provided object file. Figure 1(b) shows the resulting height map, where warm color and cold color represent high and low altitudes, respectively. The total dimensions for the city height map spanned 54,843 feet by 50,872 feet (or 9.02 nmi by 9.02 nmi) after the conversion. The altitude ranges from 0 ft to 934 ft MSL for land, and the highest elevation for

man-made structure or buildings is 1,657 ft MSL.

1. Obstacle Map at Given Above-ground Levels

The UTM concept limits small UAV operations to Above Ground Level (AGL) 400 feet and below, which in fact allows sUAVs to fly at a high MSL in certain areas. For instance, in areas where the ground elevation is 1,000 ft, if there is no other restriction, a sUAV can fly up to 1,400 ft MSL. Meanwhile, with this restriction, a flight plan with a constant MSL, as in traditional aviation, may not be acceptable for sUAV operations, especially in cities like San Francisco, where ground elevation has up to 934 feet difference from the highest to the lowest location. Therefore, an AGL-based flight plan inevitably becomes the only option for small UAV operations. To generate AGL-based flight plans, a height map needs to be constructed in terms of AGL levels. Figure 2 shows a notional picture of AGL-based flight operations.

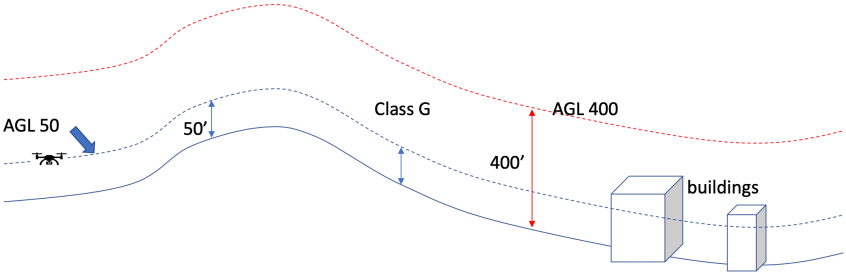


Fig. 2 Notional graph for flight operations at a constant AGL level

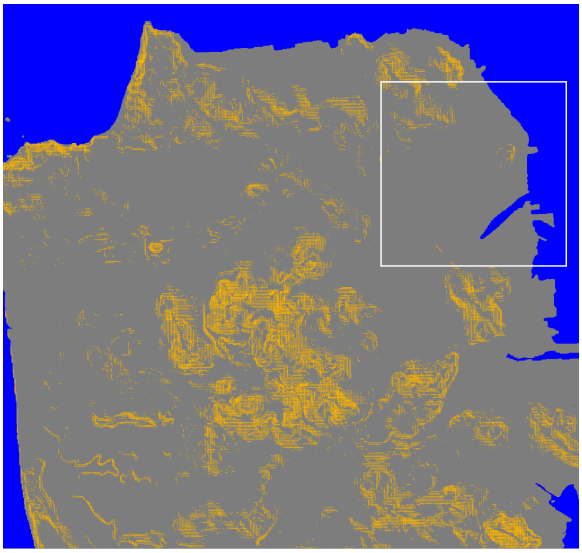


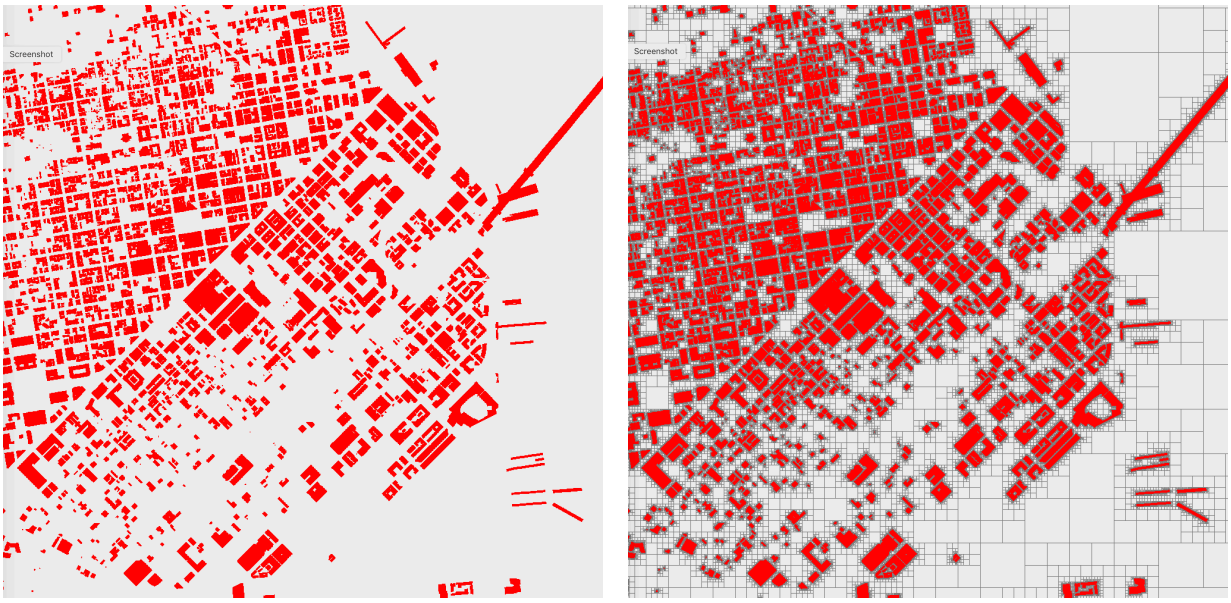
Fig. 3 The map of steep slope areas (shown in yellow) in San Francisco

However, the slope of the ground terrain may exceed the vehicle's maximum climb and descent rates. To investigate this issue, a gradient map of the San Francisco area was generated using algorithms from Canny edge detection [15]. Assuming a UAV can climb at a rate of 16.5 feet per second at a cruise speed of 40 knots, the maximum slope for a UAV to operate at its cruise speed would be around 0.3. The yellow spots shown in Fig. 3 represent areas that exceed the range of [-0.3, 0.3] and therefore may not be suitable for constant AGL flights. Some of these areas may even be impossible for UAVs to operate under the current UTM concept, simply because UAVs cannot stay in class G in those areas with their cruise speeds due to their climb and descent rate limits. This concern must be addressed in the future UTM work. In this work, studies are limited to the SF downtown area in the white box in Fig. 3, where the yellow area is not of great concern. To produce flight plans at a constant AGL altitude, obstacle maps at the same AGL altitude were constructed based on the height difference between overall terrain and the ground. Figure 4(a) presents an obstacle map

of SF downtown area at 30 ft AGL. With this approach the AGL-based flight path planning can then be simplified to a 2D problem.

2. Graph for Obstacles

A quad-tree decomposition was used to build a search space for obstacle-free flight planning. In this decomposition method, the obstacle map (as shown in red color in Fig. 4(a)) is treated as a root cell. If a cell is completely occupied by an obstacle, it is marked as "FULL". On the other hand, if a cell doesn't contain any obstacle, it is defined as "EMPTY". If a cell has some free space while partially occupied by obstacles, it is defined as "MIXED". Starting from the root cell, a "MIXED" cell will be recursively decomposed into four quadrants until it becomes either a "FULL" cell or a "EMPTY" cell or the decomposition reaches a pre-defined depth level or cell size. Fig. 4(b) shows the resulting quad-tree decomposition for obstacles with a minimum cell size of 6 feet.



(a) An obstacle map of downtown San Francisco at AGL30

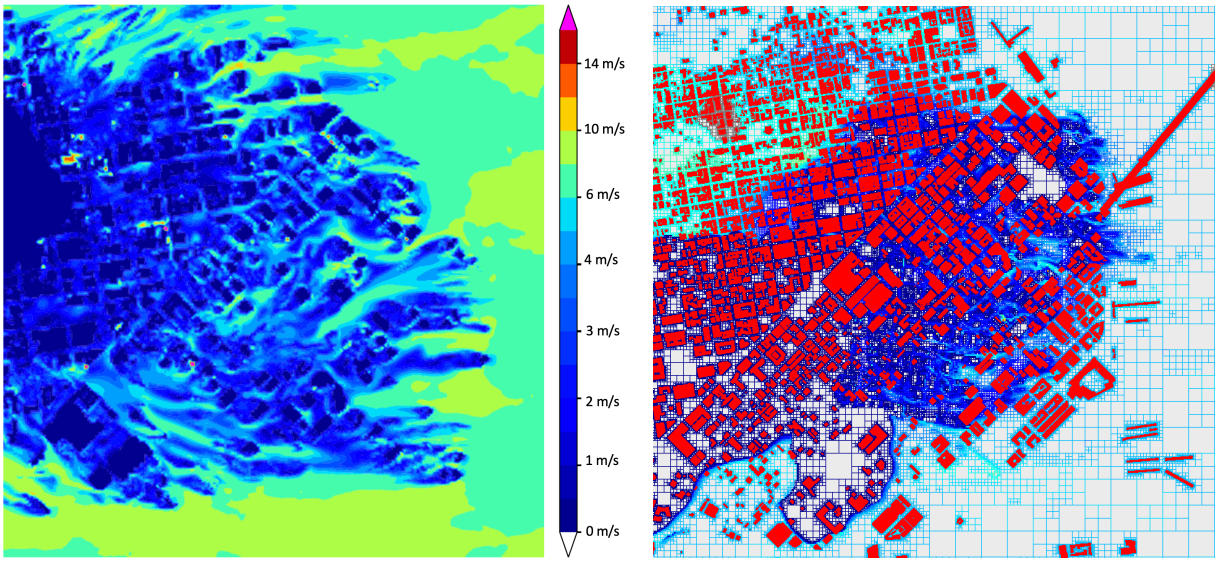
(b) Decomposition based on the obstacle map at AGL30

Fig. 4 Obstacle map and its quad-tree decomposition for downtown San Francisco at AGL30

3. Graph for Wind and Obstacles

To generate a flight plan that can also minimize wind impact, wind profiles need to be considered as well when constructing the search graph. In this work, a set of high-resolution (1m X 1m) wind profiles were generated for downtown San Francisco area using computational fluid dynamics (CFD) based simulations with a given dominant wind direction. This set of wind profiles range from MSL 0 ft to MSL 360 ft. Fig. 5(a) shows a sample wind profile at MSL 120 ft with magnitude varying from 0 to 14 meters per second, where the dominant wind was assumed to be coming from the East. Similar to constructing the obstacle map, these MSL-based wind profiles are first converted to AGL based. Then, to incorporate wind profile into the search graph, an extra decomposition criteria in the quad-tree decomposition is added: a cell will also be decomposed into four quadrants if the difference of wind vectors inside this cell exceeds a predefined threshold, even if the cell is not "MIXED". The wind-induced decomposition will start after the obstacle-induced decomposition. If a final or leaf cell from the obstacle-induced decomposition meets the criteria, the decomposition will continue until any of the three thresholds is met: (1) the difference of wind vectors, including magnitudes and directions, in a cell is less than a predefined value; (2) the size of a cell size reaches the lowest bound; or (3) the depth of the quad-tree exceeds a predefined threshold. Fig. 5(b) presents the final decomposition after introducing the wind difference, where the decomposition thresholds for wind difference are set to 0.5 mps and one degree. Different grid line colors represent different wind magnitudes, where colder colors denote lower wind

magnitudes.



(a) A sample map of CFD-simulated wind at 120 ft MSL

(b) Decomposition based on the obstacle and wind map at 30 ft AGL

Fig. 5 Incorporating wind into quad-tree decomposition for downtown San Francisco at AGL30

B. Cost definition

Once the graph is built, coefficients or weights are assigned to each leaf node or cell to help achieve obstacle-free and wind-efficient trajectories. The cost of traveling through the current cell is then calculated by:

$$c_{step} = U_{rep} \cdot w \cdot d_i + p_{i,j} \tag{1}$$

where d_i denotes the distance needed to travel through the current cell. U_{rep} and w represent the repulsive potential and wind coefficients, respectively, and $p_{i,j}$ is the penalty for unacceptable path angle changes from node j to node i. The definitions of these parameters will be found in the following subsection. Assuming the cost associated with the parent node is g_{i-1}, the cost for the current node g_i can then be written as:

$$g_i = g_{i-1} + c_{step} \tag{2}$$

1. Cost to avoid obstacles and penalize unacceptable path angle changes

A large coefficient will be assigned to the 'FULL' and 'MIXED' cells to prevent small UAVs from flying through obstacles. For 'EMPTY' cells, a repulsive potential (shown in Eqn. 3) is applied to avoid the flight path getting too close to the obstacles, where the Q^* is a constant and can be seen as the buffer size around an obstacle and D is the distance from the sUAV to the obstacle. Once the distance is greater than the threshold Q^*, there is no penalty and the coefficient is one. Similarly, the penalty of unacceptable heading angle change between lateral path angle γ_j(at node j) and γ_i (at node i) is defined as in Eqn. 4, which is activated once the path angle change exceeds a predefined Γ. The Γ is set to 45° in this work.

$$U_{rep} = \begin{cases} Q^* - D + 1, & \text{if } D \le Q^*. \\ 1, & \text{Others.} \end{cases} \tag{3}$$

$$p_{i,j} = \begin{cases} \infty, & \text{if } \mid \gamma_i - \gamma_j \mid \ge \Gamma. \\ 0, & \text{Others.} \end{cases} \tag{4}$$

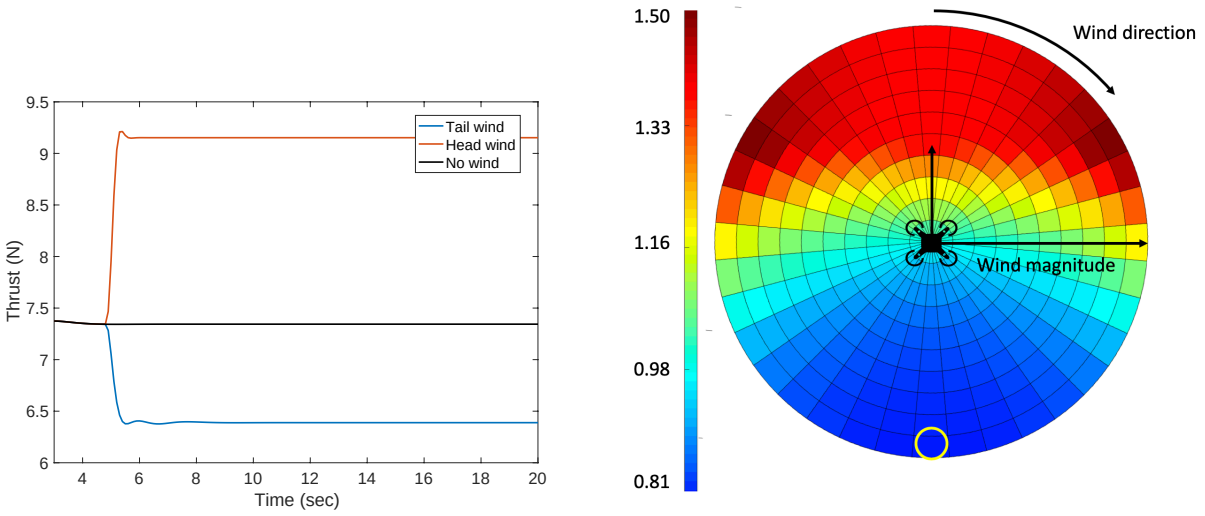
2. Wind Coefficients

In this subsection, two types of wind coefficients are derived for minimum energy and minimum time, respectively. To find energy-efficient paths in a wind field, the steady-state thrust T that is applied to maintain a desired ground speed under various wind conditions is calculated using sUAV dynamics [16]. The thrust is approximated [16–18] as a linear function of the square of motor rotational speed ω as in Eqn. 5, where k_f is a constant coefficient for a given motor. The power consumption is then approximated [16, 19] as a linear function of the cube of motor rotational speed (shown in Eqn. 6), where k_m is a constant coefficient between motor rotational speed and motor-generated torque. Using the steady-state thrust, when there is no wind, as a reference T_{ref}, the ratio between thrusts is computed and used to define a wind coefficient w to approximate the energy cost under various wind conditions (shown in Eqn. 7).

$$T = k_f \cdot \omega^2 \tag{5}$$

$$P = k_m \cdot \omega^3 \tag{6}$$

$$w = \left(\frac{T}{T_{ref}}\right)^{1.5} \tag{7}$$



(a) Thrust history at various wind conditions

(b) Coeffcient matrix for various wind vectors

Fig. 6 Generation of wind coefficients

Figure 6(a) shows time histories of thrust for a multi-copter flying with a ground speed of 15 mps (29 knots) in three different wind conditions: 10 mps tail wind, no wind, and 10 mps head wind. The figure shows that the small UAV entered the wind field at the 5th second and then experienced a transition state and finally settled down at a steady state. The steady-state thrusts are 6.39, 7.34, and 9.15 Newton for tail wind, no wind, and head wind, respectively. The energy coefficients for these three wind conditions were then calculated as 0.81, 1.0, and 1.40, respectively. A set of such wind coefficients was then generated for wind vectors with various direction and magnitude. Figure 6(b) presents such a wind coefficient matrix at a desired ground speed of 15 mps, where the coefficient varies from 0.81 to 1.50. In the polar coordinate system shown in this figure, the radial coordinate represents the wind magnitude from 1 to 10 mps and the angular coordinate denotes the clock-wise wind direction from 0° to 360°. The grid circled by a yellow line corresponds the coefficient under a tail wind (180°) of 15 mps, which is 0.81. As expected, tail winds are favored from an energy-saving perspective, and head winds increase the energy consumption.

Calculation of the wind coefficient for finding time-optimal paths in a wind field is straightforward. Assuming that the vehicle airspeed is higher than the wind speed, the coefficient w is simply defined in terms of the vehicle and wind

speeds as shown in Eqn. 8, where $\vec{V}_{ac}$ and $\vec{V}_w$ are the velocity vectors for aircraft and wind, respectively. This definition will incentivize flying with a tailwind and penalize traveling with a headwind.

$$w = \frac{|\vec{V}_{ac}|}{|\vec{V}_{ac}| + \frac{\vec{V}_w \cdot \vec{V}_{ac}}{|\vec{V}_{ac}|}} \tag{8}$$

C. Search algorithm

Once the graph is built, the Theta* [20] search algorithm, a variant of A* algorithm, is applied to find the optimal path. The key difference between Theta* and A* is that Theta* allows the parent of a vertex to be any previous predecessor, whereas in A* the parent must be the adjacent predecessor. This change allows any-angle paths and mitigates the rigid path angle constraint introduced by the grid-based search graph. The pseudo code of the Theta* search algorithm is shown in Algorithm 1. The step cost $c(S, S')$ represents the c_{step} as shown in Eqn. 1. The g-cost is the cost from the start node to current node as in Eqn. 2, where both obstacles and wind condition are taken into account. And the h-value is proportional to the straight-line distance from current node to the goal, while the proportion is set to underestimate the cost from current node to the goal, so the admissibility of the algorithm is preserved. The part between Line 2.19 and 2.24 in Algorithm 1 is the same as in A*, while the part between Line 2.11 and 2.16 is introduced by the Theta* method to allow search more predecessors and to essentially introduce more path angle options.

Algorithm 1: Theta* Search

```
2.1  g(s_start) ← 0; parent(s_start) ← s_start; open ← s_start     /* Initialization */
2.2  while open ≠ ∅ do
2.3      s ← open.pop()
2.4      if s = s_goal then
2.5          return s
2.6      else
2.7          closed ← closed ∪ S
2.8          for s' ∈ neighbor(s) do
2.9              /* Theta* path */
2.10             if LineOfSight(parent(s), s') then
2.11                 if g(parent(s)) + c(parent(s), s') < g(s') then
2.12                     g(s') ← g(parent(s)) + c(parent(s), s')
2.13                     parent(s') ← parent(s)
2.14                     if s' ∈ open then
2.15                         open.remove(s')
2.16                     open.add(s', g(s') + h(s'))
2.17             else
2.18                 /* A* path */
2.19                 if g(s) + c(s, s') < g(s') then
2.20                     g(s') ← g(s) + c(s, s')
2.21                     parent(s') ← s
2.22                     if s' ∈ open then
2.23                         open.remove(s')
2.24                     open.add(s', g(s') + h(s'))
```

D. Post smoothing

Although the Theta* method takes heading angle change constraints into account during the search process, a simple and effective post smoothing technique [21] was applied to further smooth the path angle transition while not losing much optimality [20]. The pseudo procedure of the post smoothing method is shown below. The main idea is starting from the end node of the path. For each node, if the upstream node of its parent node can be reached in a straight line

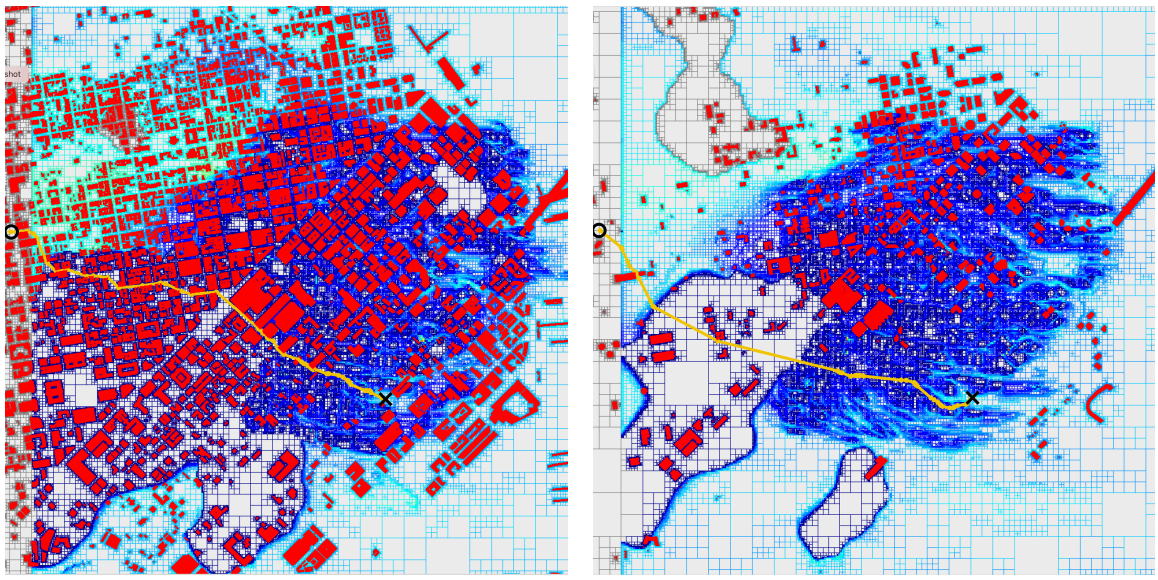
and can help reduce the overall cost, then this upstream node will be defined as the new parent for the current node.

Algorithm 2: Post Smoothing

3.1 $s \leftarrow s_{goal}$
3.2 **while** $s \neq s_{start}$ **do**
3.3 **if** $parent(parent(s))$ *is visible from* s **and** $g(parent(parent(s))) + c(parent(parent(s)), s) < g(s)$ **then**
3.4 $g(s) = g(parent(parent(s))) + c(parent(parent(s)), s)$
3.5 $parent(s) \leftarrow parent(parent(s))$
3.6 $S \leftarrow Parent(S)$

III. Results

Using the method described in the previous section, given an origin and destination pair, an obstacle-free and wind-efficient path is quickly generated. Figure 7(a) and 7(b) show two final paths (shown as orange curves) that are obstacle-free and wind-efficient for operations in downtown San Francisco at AGL 30 ft and 90 ft, respectively. The black circles denote origin, and black crosses denote destinations. The operational environments are quite different at different AGLs: the obstacle field is more complicated at lower AGL and wind is stronger at higher AGL.



(a) A path at AGL 30 ft

(b) A path at AGL 90 ft

Fig. 7 Obstacle-free and wind-efficient sUAV paths with constant AGL over downtown San Francisco

A. Computational performance

To examine the efficiency of the proposed method, experiments were conducted on a MacBook Pro with 2.5 GHz Intel Core i7 and 18 GB memory. The computational times for these cases are presented in Table 1. Case I, II, and III have the same search area, origin, and destination except for different flight altitudes. Without loss of generality, the comparison among Cases I, II, and III shows that when the flight altitude increases, the time spent for building and searching decreases, mainly because the number of obstacles was reduced. On the other hand, Cases IV and V have longer paths than previous cases and the search areas increased to 8 km×8 km and 10 km×10 km, respectively. The times consumed at all phases were slightly increased and the overall times are slightly larger (over 9 seconds), which is pretty fast considering the size and complexity of the search space. The breakdown also showed the computational times were almost evenly split among the reading, building, and searching phases. The computational cost of the post smoothing phase was negligible.

Table 1 Computational time breakdown

	Case I 4km×4km AGL 30ft	Case II 4km×4km AGL 60ft	Case III 4km×4km AGL 150ft	Case IV 8km×8km AGL 30ft	Case V 10km×10km AGL 30ft
Reading inputs (s)	3.08	3.14	3.01	3.49	3.78
Building (s)	1.63	1.26	0.73	1.80	2.04
Searching (s)	2.55	2.60	1.88	2.05	3.91
Post-smoothing (s)	0.01	0.01	0.01	0.01	0.03
Total (s)	7.27	7.01	5.63	7.31	9.76

B. Vertical profile and lateral path angle change

Figure 8 shows the vertical profile of the final path in Case I. The solid line represents the flight path, and the dashed line denotes the ground terrain. The slopes of the flight path and terrain were well below 10%, which is acceptable for most small UAVs. However, as discussed in the previous section, the terrain slopes in areas highlighted in Fig. 3 would exceed the maximum climb and descent rates if small UAVs were requested to fly there at constant AGL.

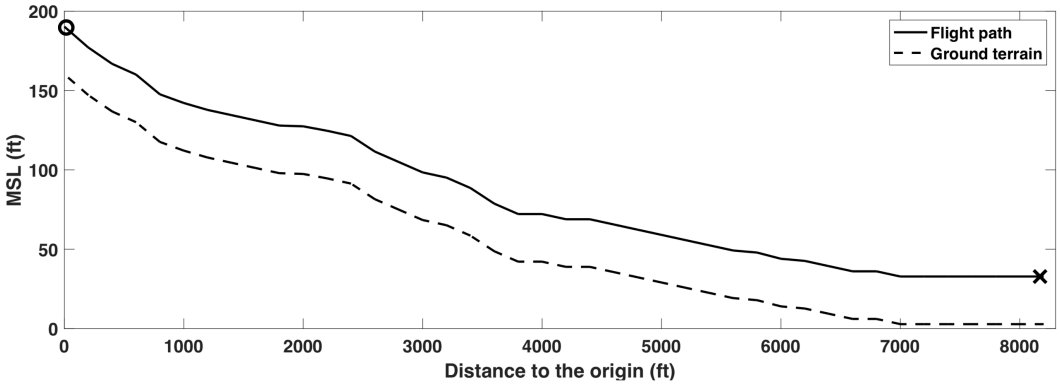


Fig. 8 Vertical profile for the final path in Case I

Figure 9 presents the absolute values of lateral path angle changes between two consecutive segments of the path in Case I. As expected, all path angle changes are less than 45°, which was defined as the limit in the cost function. The Theta* method worked well in meeting the constraints for path angle changes.

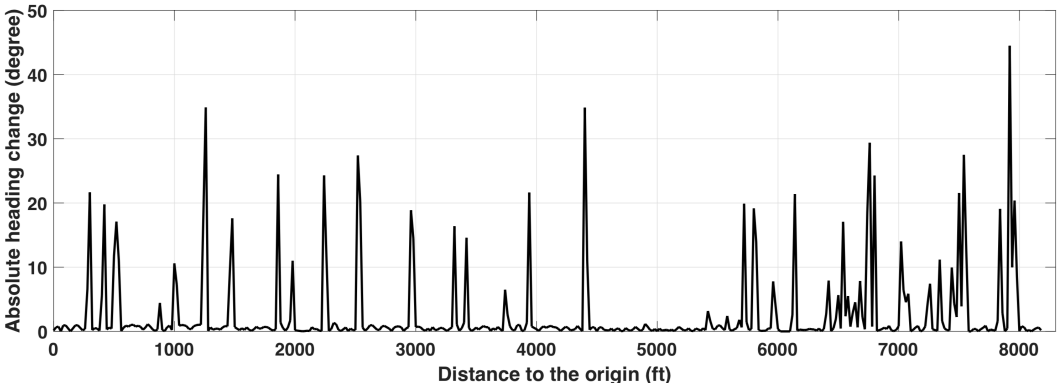
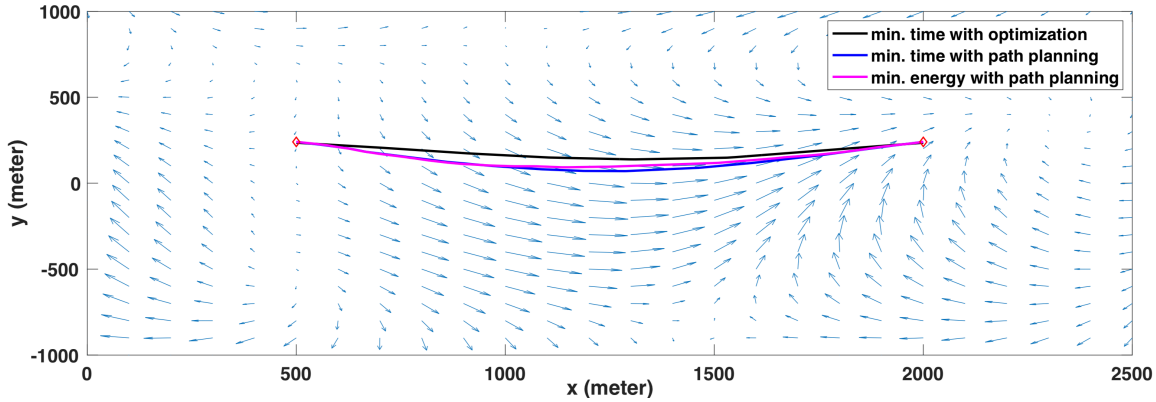


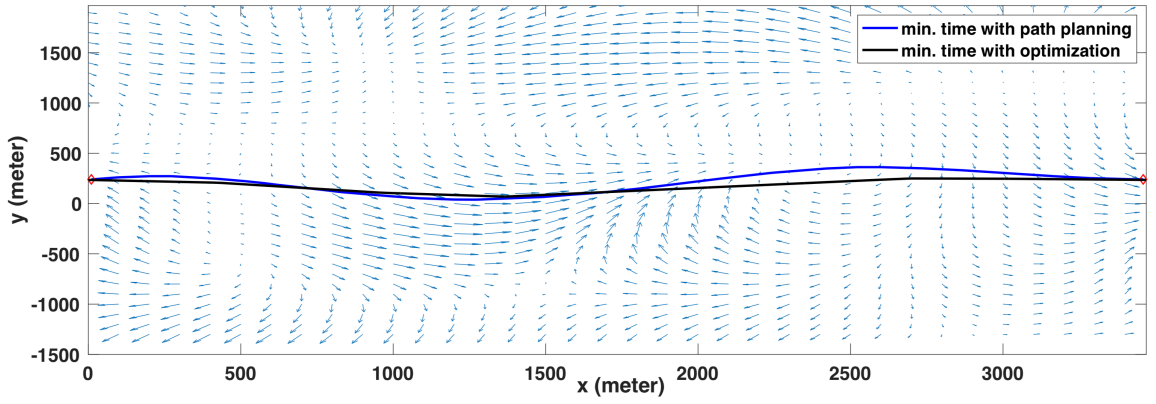
Fig. 9 Heading angle change (absolute) for the final path in Case I

C. Comparison with the conventional trajectory optimization method

It is not suprising that the proposed path-planning algorithm can work well in obstacle-rich environments. To examine the effectiveness of the path-planning method in an obstacle-free environment, a multiple-shooting based trajectory optimization method was compared for finding wind-optimal trajectories. An optimization demo tool [22] that uses the multiple-shooting optimization method was applied to produce sample wind fields and minimum time trajectories under wind conditions. The Theta* based path planning method proposed in this work was then applied in the same wind field to find wind-optimal trajectories for comparison. Figure 10(a) shows the trajectories generated by these two different methods. The black curve represents the minimum-time wind-optimal trajectory generated by the multiple-shooting optimization method. The blue curve represents the path generated by the path planning algorithm with the wind coefficient minimizing the travel time (in Eqn. 8). The magenta curve reprents the path generated by the path planning algorithm using the wind coefficient minimizing the energy (as in Eqn. 7). The final flight times for these three trajectories (black, blue, and magenta) are 85.7s, 85.4s, and 85.7s respectively. The path planning method with time-minimum wind coefficient actually outperforms the multiple-shooting trajectory optimization method. Similar results hold for longer paths as shown in Fig. 10(b), where the wind field is a bit more complicated: the travel times for the multiple-shooting method (black curve) and path-planning method (blue curve) are 213.4s and 212.1s, respectively. With a proper choice of wind coefficient, the Theta* based path planning method performs surprisingly well.



(a) Comparison of short paths



(b) Comparison of long paths

Fig. 10 Comparison of wind optimal trajectories generated with different methods

D. Discussion

Experiments in this study showed the grid-based path planning algorithm can solve the flight planning problem efficiently while not sacrificing much optimality. Although only obstacles, wind, and path angle change were incorporated

in the search cost in this work, other location-based factors such as dynamic and static restricted airspace, weather, ground risk, noise exposure, population density, and navigation and communication signal strength could also be incorporated to the cost function without much increase in computational time. Like any other multiple-objective optimization problem, how to construct/weigh these multiple costs still needs to be addressed carefully. Once the multiple-objective cost function is constructed, the grid-based Theta* path planning algorithm should be able to handle the search more efficiently than conventional trajectory optimization approaches.

IV. Conclusions

A grid-based Theta* path planning algorithm was introduced to generate obstacle-free and wind-efficient paths for sUAVs at a constant AGL in urban environments. This method first converts the 3D path planning problem to a 2D problem by constructing an obstacle map at a given AGL. A quad-tree decomposition is then used to build the search space in terms of obstacle occupancy and wind difference. The wind cost of traveling through each cell is defined based on UAV power consumption under various wind conditions. A repulsive potential is also adopted to make sure the flight plan stays away from obstacles. The Theta* search algorithm was applied to find the flight path having the lowest cost with the capability of mitigating the path angle change constraints. With the proposed Theta* algorithm and post smoothing techniques, an obstacle-free, wind-efficient, and constant above-ground-level flight plan was efficiently generated for small UAV operations in urban environments while meeting lateral path angle constraints.

The results showed that the computational time of this algorithm is reasonable for real-time applications. The vertical profile and path angle change showed the feasibility of resulting paths, although there is a concern for the small UAVs flying at a constant AGL in an area where the slope exceeds the maximum climb/descent rates of the vehicle. Experiments also showed that, with a proper choice of the wind coefficient, the Theta*-based path planning algorithm outperformed the multiple-shooting trajectory optimization method in an obstacle-free environment. Overall, through experiments the proposed algorithm showed efficiency in flight path planning in complex urban environments and potential to incorporate geo-related costs.

V. Acknowledgement

The authors gratefully acknowledge the contribution of Dr. John Melton (NASA) and his team in generating the CFD-based wind profiles and the contribution of Sam Allison (intern student from Oklahoma State Univeristy) in calculating small UAV steady-state thrusts under various wind condition. The authors would also like to thank Russell Paielli (NASA) for the discussion regarding operations under the current UTM concept and Dr. Hok Ng (NASA) for the discussion of aircraft trajectory optimization in winds.

References

[1] Xue, M., and Atkins, E. M., "Noise-minimum Runway-Independent Aircraft Approach Design for Baltimore-Washinton International Airport," *Journal of Aircraft*, Vol. 43, No. 1, 2005, pp. 39–51.

[2] Betts, J., "Survey of Numberical Methods for Trajectory Optimization," *Journal of Guidance, Control, and Dynamics*, Vol. 21, 1998, pp. 193–207.

[3] Zhao, Y., Bryson, A. E., and Slattery, R., "Generalized Gradient Algorithm For Trajectory Optimization," *Journal of Guidance, Control, and Dynamics*, Vol. 13, No. 6, 1990, pp. 1166–1169.

[4] Ng, H. K., Sridhar, B., Chen, N. Y., and Li, J., "Three-Dimensional Trajectory Design For Reducing Climate Impact of Trans-Atlantic Flights," *AIAA Aviation Forum*, Atlanta, GA, 2014.

[5] Xue, M., and Atkins, E. M., "Terminal Area Trajectory Optimization Using Simulated Annealing," *44th AIAA Aerospace Science Meeting and Exhibit*, Reno, Nevada, 2006.

[6] Richards, A., Schouwenaars, T., How, J. P., and Feron, E., "Spacecraft Trajectory Planning with Avoidance Constraints Using Mixed-Integer Linear Programming," *Journal of Guidance, Control, and Dynamics*, Vol. 25, No. 4, 2002, pp. 755–764.

[7] Latombe, J. C., *Robot Motion Planning*, Kluwer Academic Press, Norwell, MA, 1991.

[8] Cheng, P., Shen, Z., and LaValle, S. M., "RRT-based Trajectory Design for Autonomous Automobiles and Spacecraft," *Archives of Control Sciences*, Vol. 11, No. 3-4, 2001, pp. 167–194.

[9] LaValle, S. M., *Planning Algorithms*, Cambridge University Press, Cambridge, 2006.

[10] Goerzen, C., Kong, Z., and Mettler, B., "A Survey of Motion Planning Algorithms from the Perspective of Autonomous UAV Guidance," *Journal of Intelligent and Robotic Systems*, Vol. 57, 2010, pp. 65–100.

[11] Ramana, M., Varma, S. A., and Kothari, M., "Motion Planning for a Fixed-Wing UAV in Urban Environments," *4th IFAC Conference on Advances in Control and Optimization of Dynamical Systems (ACODS)*, Tiruchirappalli, India, 2016.

[12] 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.

[13] Chakrabarty, A., Stepanyan, V., and Krishnakumar, K., "Real-Time Path Planning for Multi-copters Flying in UTM-TCL4," *AIAA SciTech Forum*, San Diego, California, 2019.

[14] 3D CAD Browser, "San Francisco 3D City Model," `https://www.3dcadbrowser.com/3d-model/san-francisco-city`, 2019. Online; accessed May, 2019.

[15] Canny, J., "A Computational Approach To Edge Detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 8, No. 6, 1986, pp. 679–698.

[16] Xue, M., Rios, J., Silva, J., Ishihara, A., and Zhu, Z., "Fe^3: An Evaluation Tool for Low-Altitude Air Traffic Operations," *AIAA Aviation Forum*, Atlanta, GA., 2018.

[17] Russell, C. R., Jung, J., Willink, G., and Glasner, B., "Wind Tunnel and Hover Performance Test Results for Multicopter UAS Vehicles," *American Helocipter Society 72nd Annual Forum*, West Palm Beach, FL, 2016.

[18] Foster, J. V., and Hartman, D. C., "High-Fidelity Multirotor Unmanned Aircraft System Simulation Development for Trajectory Prediction Under Off-Nominal Flight Dynamics," *17th AIAA Aviation Technology, Integration, and Operations Conference*, Denver, Colorado, 2017.

[19] Nicoud, J. D., and Zufferry, J. C., "Toward Indoor Flying Robots," *IEEE International Conference on Intelligent Robots and Systems*, Lausanne, Switzerland, 2002.

[20] Nash, A., Daniel, K., Koenig, S., and Felner, A., "Theta*: Any-Angle Path Planning on Grids," *AAAI'07 Proceedings of the 22nd national conference on Artificial intelligence*, Vancouver, British Columbia, Canada, 2007.

[21] Botea, A., Muller, M., and Schaeffer, J., "Near Optimal Hierarchical Path-finding," *Journal of Game Development*, Vol. 1, 2004, pp. 7–28.

[22] Fenelon, M., "Demo: Finding an Optimal Path Using MATLAB and Optimization Toolbox," `https://www.mathworks.com/matlabcentral/fileexchange/36321-demo-finding-an-optimal-path-using-matlab-and-optimization-toolbox`, 2020. MATLAB Central File Exchange. Retrieved March 30, 2020.