Aerial Robotics in Urban Environments: Optimized Path Planning and SITL Assessments

1st Seyyed-Ahmad Abtahi
The Faculty of New Sciences and Technologies
University of Tehran
Tehran, Iran
ahmadabtahi70@aut.ac.ir

2nd M.A. Amiri Atashgah
The Faculty of New Sciences and Technologies
University of Tehran
Tehran, Iran
atashgah@ut.ac.ir

3rd Bahram Tarvirdizadeh
The Faculty of New Sciences and Technologies
University of Tehran
Tehran, Iran
bahram@ut.ac.ir

4th Mohammad Shahbazi
School of Mechanical Engineering
Iran University of Science and Technology
Tehran, Iran
shahbazi@iust.ac.ir

2023 11th RSI International Conference on Robotics and Mechatronics (ICRoM) | 979-8-3503-0810-5/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICRoM60803.2023.10412604

Abstract—This paper focuses on guiding aerial robots through city environments. The goal is to create an optimal and safe flight path, avoiding obstacles in urban landscapes with tall buildings. We combine the A* algorithm and the Medial Axis algorithm to develop a well-balanced trajectory that ensures both safety and efficiency. To better understand narrow passages in cities, we analyze and preprocess two-dimensional maps. The A* algorithm forms the basis, while the Medial Axis algorithm identifies secure routes through tight spaces. Additionally, we explore new methods to seamlessly integrate multiple path planning algorithms, offering a comprehensive solution for navigating urban airspace. Our proposed method is rigorously evaluated using simulation tools, mimicking real-world scenarios. The results consistently demonstrate superior safety and efficiency compared to alternative algorithms. Even when subjected to atmospheric disturbance tests, our approach maintains adherence to critical operational constraints, ensuring enhanced safety and shorter flight paths. This study significantly contributes to advancing aerial robotics in urban areas, providing a promising way to enhance safety and efficiency in complex environments.

Keywords—Optimal Path Planning, Urban Environments, Hybrid Algorithms, A*, Medial Axis, SITL

I. INTRODUCTION

A path-planning algorithm determines a path from an object to a target point or destination; It's a strategic issue that specifies a robot's consecutive steps over time to reach the target. The challenge here is to avoid collisions with other objects in the path. A robot's environment can be presented in various ways, from a continuous geometrical description to a decomposed holonomic map and even a topological map [1]. Converting a continuous environment model into a discrete map, suitable for a path-planning algorithm, is the first step in every path-planning algorithm. Path-planning methods are categorized in various ways; e.g., classic and reactive [2]. Classic methods are based on math and statistics, while reactive methods that have become popular in the last decade imitate living organisms and nature. Roadmap [3], potential field [4], and cell decomposition path-planning algorithms such as A* [5] and Voronoi are examples of classic methods. Genetic [6], neural networks [7], and learning methods [8] are some of the best-known reactive methods.

While each has unique advantages, no presented algorithm has addressed all the existing issues, such as travel time, path length, and unintended events. Hence combining the mentioned methods might lead to a more comprehensive and successful path-planning method for a robot or device. Studies that propose a new algorithm, including ours, usually use static, non-holonomic, and deterministic environments. Path planning is not a trivial problem; Each of the various existing algorithms has its advantages and disadvantages. None of them meet all the needs on their own. A* is a classic and widespread algorithm; Its multiple updates since its introduction in 1968 have eliminated many disadvantages. Inefficiency in dynamic environments, being time-suboptimal in crowded environments, lack of flexibility in unexpected paths, and lack of an effective optimal solution to the problem of passing through obstacles are fundamental shortcomings of classic algorithms in general and A* in particular [9]. A* algorithms offer different solutions to all these issues, which will be discussed in detail in the next section. Our research studies one of the main but less explored problems of classic algorithms, namely path-planning near obstacles; it presents an intelligent solution to prevent the elimination of narrow passages in the map. Most methods avoid narrow passages to keep the moving agent safe.

The primary algorithm we use is A*, while the medial axis algorithm is employed to create safe narrow passages. This innovation makes the path safe and shortens some narrow passages. We call our mixed method A* based on the medial axis algorithm (MAA*). Techniques such as creating a safety margin around the obstacles, adding costs near the obstacles, or using a complementary graph to make the path safe have been tried. The result was blocking the narrow passages, the inappropriate performance of the method, or too long paths [10][11][12]. MAA* fixes this issue and can be easily upgraded to another version of A*, such as Anytime A* for making the system real-time [13] and D* [14] for dynamic environments. Many modifications have been applied to A*, and each has performed better in specific environmental conditions [15]. Finally, we simulate our proposed method to explore its performance and capabilities. The rest of the paper is organized as follows.

We discuss similar research in the second section and carefully examine the advantages and disadvantages of those path-planning algorithms. Section three presents the preprocessing phase of our method, i.e., analyzing two-dimensional maps to distinguish the narrow passages. We offer two methods to distinguish between the close obstacles and examine the pros and cons of each. In Section four, we present three different methods to combine A* and medial axis algorithms. Section five analyzes the methods of section three and gives the best solution. It also presents simulation test results and compares them with those of the original algorithms. The conclusion section summarizes the paper and provides suggestions to extend it further.

II. Literature Survey

As stated, the primary algorithm used is A*; a commonly-used algorithm. Peter Hart, Nils Nilsson, and Bertram Rafael from Stanford Research Institute published the main idea of the algorithm in 1968 [5]; It can be considered an extension of the Dijkstra algorithm [16]. While old, the algorithm's various modifications and updates have made it much more efficient over time. Erke et al. [12] presented an A*-based method for intelligent vehicles' path planning. They addressed reaching obstacles' corners and path-planning with and without safety margin in A*. The authors used a separate graph and changed the A* cost function using that graph to fix the issue and keep away from the corners. The required human intervention to draw the graph and increased computational cost due to using an algorithm to design the graph substantially decreases the path optimality in this method. Benzaid et al. [17] introduced a three-dimensional path-planning algorithm by combining the medial surface and A* algorithms and smoothing the resulting path using the Bézier curves method. This makes the path safe but increases its length.

Authors in [9] give a general review and comparison of different path-planning algorithms and mention the better search speed performance of A* and Dijkstra compared to meta-heuristic algorithms. [18] presents a new method that reduces the computational time of A* in a stationary environment using phasing and dividing the path. In[19], SAS1 path-planning is used; Changing the coefficients of the path cost function and making them continuous have improved SAS's multi-UAV path-planning performance. Zhang [20] blends a variant of A*, Anytime Repairing A*, with Kalman filtering to predict target movements. This enhances the A* ability to track moving targets in a dynamic environment. In [21], the Authors mix the Voronoi diagram, visibility graph, and artificial potential field algorithms to strike a balance between the safety and length of the path. The combined algorithm's path is shorter than the synthetic potential field and Voronoi diagram but is suboptimal; The algorithm is also very complex. Costa et al. [22] discuss several path-planning algorithms, including genetic, RRT, A*, and TEA*. TEA* is A* plus time which can be used in dynamic environments and multi-robot systems. Weak performance in in long paths and complex environments is a significant disadvantage of the A* algorithm; Most A* updates were applied to the primary traditional algorithm.

Authors of [23] use medial axis and visibility graph algorithms for path-planning in an internal two-dimensional environment. They divide the interior environment of the building into subdivisions to ensure the ultimate safety of the designed path. In [24], using a variant of A* proves the real-time efficiency of the algorithm in crowded environments with many obstacles, both in simulated and natural environments. Chen et al. [25] changed the cost function equation of the A* algorithm to improve some of its features. They reduce the computation time significantly by making A* directional. In [26], the cost function and some of its weight coefficients are modified to fix some of the issues of the A*; Simulated results indicate a decrease in computation time. Qu et al. [27] mix Genetic, Voronoi, and A* algorithms for combat zone path-planning. They use the A* and Voronoi algorithms to plan paths that are safest and farthest from the enemy radars. Tang et al. [28] use a mixture of A* and interpolation algorithms' advantages and introduce the geometric A* algorithm using geometric rules. Their algorithm can design a smoothed optimal path while reducing computation time. Authors of [29] offer a variant of A* for target tracking and study parameters such as minimum rotation angle and weight coefficients. They also present a method to assign different targets to different fliers. Thoa et al. [30] discuss the advantages and disadvantages of various classic and reactive algorithms. They believe that most recent research has been done on static environments. Static environment algorithms lay the foundation to develop dynamic environment algorithms. Chen et al. [11] study heuristic and cost function details to simplify A* computationally. They increase the cost near the obstacles in the cost function to make the path safe in A*. However, this limits the path in narrow passages. They also compare genetic and ant-colony algorithms with A* and point to slow convergence and the possibility of stopping in local extrema as the disadvantages of the mentioned algorithms. He et al. [31] compare the real-time aspects of ACO, Floyd, and Dijkstra algorithms. Dijkstra, equal to A*, gives them the best results in finding the optimal path.

Benders et al. [32] use A* to plan an obstacle-free path in windy environments. The authors of [33] use A* to design a map to enhance the performance of unmanned flying robots. They present a method to model the flier environment. Li et al. [34] study the path coordination between unmanned airborne and terrain robots. They use A* to find the optimal path. Gupta et al. [35] use A* to fly multiple unmanned robots. They showed that cooperation between multiple fliers could eliminate the surveillance and reconnaissance issues a single flier faces on its own. Their simulation result showed a computation time decrease from 13.59 to 3.89 seconds as the number of fliers went from one to four. Kwak et al. [36] also employed A* for surveillance operations in unmanned flying robots. Sun et al. [37] used the algorithm for real-time prediction of the plan. They used a three-stage prediction algorithm to assign tasks, improve the path, and smooth it. Li et al. [38] utilized the algorithm to plan an optimal path between a base and backup station and a seller and a customer to deliver goods. Many improved A* algorithms have been published, such as Block A*, D*, Field D*, Fringe Saving A*

[1] Sparse A* algorithm

(FSA*), and Generalized Adaptive A* (GAA*). These are regularly used in path planning for unmanned fliers. Chengjun et al. [39] used an improved A* to plan the path. They studied various criteria, such as fuel consumption, environment, and obstacles. Song et al. [40] presented a modified A* algorithm. They created a path based on a weighted graph and applied the modified A* to that path to optimize the path. [18] is another example of using a modified A* to create an optimal path. Ma et al. [41] provided a coordinated method based on the Lyapunov function and A*; Their simulations indicate the shortest path and time needed to design a path. Penin et al. [42] presented a bidirectional A* with intermittent measurement while taking the shortest time and acceptable measurements into account. Liang et al. [43] applied the algorithm to Digital Elevation Maps (DEM) used in military and civil applications. They employed the Bresenham method and Bézier curves for the secondary processing of the path. Mengying et al. [44] compared the advantages, disadvantages, potential capabilities, and theory of improved A* with those of APF and PSO algorithms. They concluded that it has been widely used to avoid obstacles based on their data analysis. Now that we've covered the literature on A* and its updates, let's consider innovation in this research.

III. CONTRIBUTIONS

To combine A* and medial axis algorithms, we generate a grid map with the appropriate cell size; Fig. 1-b is a binary gridded version of the San Francisco (Fig. 1-a). Fig. 1-b demonstrate a 'two and a half' dimensional map; This means considering the main map at the flying robot's altitude (20 meters above the ground) and then drawing the other two dimensions while paying no attention to the altitude. The algorithm applied to each cell/block/square pattern differs according to the environmental conditions inside that cell. Let's elaborate on what we mean by environmental conditions. Getting close to the obstacles in the path is a significant issue in A*. It is often fixed by creating a safe margin around the obstacles. This makes the obstacles in a cell the main factor in choosing that cell's algorithm. Cells with very close obstacles or narrow passages between obstacles can't use traditional A* efficiently and are labeled as a 'medial axis' block. Blocks with obstacles safely distanced from each other with no narrow passages are labeled 'A*' cells.

How can we identify close obstacles in a cell? What algorithm should we use? Here are our two novel methods.

A. Machine Learning

This method uses binary classification algorithms to select between A* and medial axis algorithms. Common algorithms for binary classification include logistic regression, k-nearest neighbors, decision tree support, vector machine, and naïve Bayes. Considering the challenge in this research, the selection of the classification algorithm doesn't affect the result much, as we'll discuss later. We use a decision tree algorithm; The algorithm is trained using the percentage and size of the obstacles and their number in each block.

For training data, the map in Fig. 1-b is divided into 361 cells: 50*50 pixels in each. The 'algorithm label' for each cell is assigned manually. Those samples were then used to train the decision tree. The post-training accuracy of that algorithm for labeling cells was 80 to 90 percent. Considering the nature of the research that aims to improve path safety through accurate cell identification, any shred of doubt cannot be tolerated. According to our studies, no existing artificial intelligence algorithm, including machine learning and deep learning, will not achieve the 100% accuracy needed to make a significant difference in this case. This method is quite adequate, for example, for satellite images with no precise boundary. However, we need another method to provide doubtless accuracy. The following section discusses such a method.

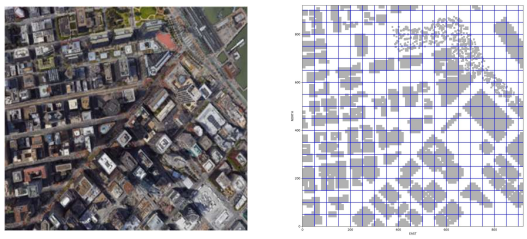
B. Map Processing

This method is based on machine vision concepts. It's purposes are distinguishing between different obstacles within a cell or in an image, determining the distance between all distinguished obstacles, and identifying obstacles too close to each other; A cell will be labeled A* if there are close obstacles and 'medial axis' otherwise. This labeling identifies narrow passages inside a cell. This way, the flying robot will go through a trouble-free path with the maximum possible distance from the obstacles thanks to the medial axis algorithm.

We borrowed connected component labeling and analysis, and boundary detection concepts from map/image processing to identify narrow passages; These concepts let us identify the narrow passages. The black rectangle in Fig. 2 shows a minimum distance of one cell between two obstacles. This means the medial axis algorithm must be used here.

The general workflow is as follows. For each cell:

1. Assign a number to each obstacle to distinguish them
2. Identify obstacle edges' not on the boundary of the map or cell
3. Determine the least distance between the obstacles
4. Select the appropriate algorithm based on the allowable distance between obstacles



a) Satellite map of San Francisco [45]

b) Binary map with cells

Fig. 1. The generated grid map

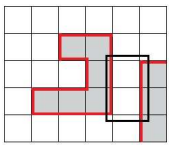


Fig. 2. Identifying a narrow passage and labeling it

IV. THE PROPOSED ALGORITHMS

After the text edit has been completed, the paper is ready for the template. Duplicate the template file by using the Save As command, and use the naming convention prescribed by your conference for the name of your paper. In this newly created file, highlight all of the contents and import your prepared text file. You are now ready to style your paper; use the scroll down window on the left of the MS Word Formatting toolbar.

A. Independent Combining and Planning

The proposed method follows Algo. 1 step by step. The flying robot uses a combination of such paths according to the new approach. This most basic blending of the algorithm has many disadvantages. This method guarantees a complete and safe path using the combined algorithm. However, path planning needs to be done twice with both methods from source to destination. This increases the computation time and wastes lots of time. Increasing the path length due to the algorithm change between cells is another disadvantage that makes the path suboptimal. These issues make the method impractical. Let's look at other proposed methods that eliminate such disadvantages.

Algorithm 1: independent Combining and path planning

Input: *2D real map*

Output: *path planning with method 1 of MAA* algorithm*

```
B = create a binary map from real map
S, G = Initialize start and goal points on the binary map B in line 1
M = extract medial axis from B
Divide B and M into 50*50 blocks
Label the blocks in line 4 with A* or Medial Axis algorithm
A = Path planning with A* algorithm (from S to G)
R = Path planning with A* algorithm on Medial axis roadmap (from S to G)
foreach block in path A do
    if block is Medial axis labeled then
        Substitute path A with the closest path in B in that block
Bound the separated path from S to G
End
```

B. External Boundary and Cell Targetting Planning

This proposed method tries to make the most of the advantages of both algorithms. The sub-optimality of the previous method makes us think of a proper, comprehensive method. The novel technique in this section effectively fixes most of the current issues. Here each cell is considered a target. The final target is decomposed into several intermediary targets. Successively reaching each middle target takes the flier robot to the final target. A* algorithm uses two cost functions for path planning: a performance cost function corresponding to the agent's 4- or 8-directional performance in each moment and a heuristic one that depends on the distance from the agent to the end target at that moment. As the flier gets closer to the target, the cost decreases. Based on the definitions of the cost functions, the path planning follows Algo. 2.

A* central equation is:

$$F(n) = g(n) + h(n) \tag{1}$$

In which the end target is the nth node. Assuming 'm' means the middle target and m_n is the final target, then path L should go through the following intermediate targets.

$$L = m1\ ,\ m2,\ ....,\ mn \tag{2}$$

Algorithm 2: External Boundary and Cell Targeting Planning

Input: *2D real map*

Output: *path planning with method 2 of MAA* algorithm*

```
B = create a binary map from real map
S, G = Initialize start and goal points on the binary map B in line 1
M = extract medial axis from B
Divide B and M into 50*50 blocks
Label the blocks in line 4 with A* or Medial Axis algorithm
P = Extract outer boundary points in every block in line 4
Delete a point in P that equals one (obstacle points)
While (W != G) do
    for i = 1 to i = 3 step 1 do
        W = Find the closest point to the goal in P by Euclidean distance to find out which block is next to expand in the current block
        If the label of the next block is different from the current block then
            Set W as local goal
            Break
        Set W as local goal
    A = Path planning with A* algorithm (from S to W) based on the label of block
    TOTAL PATH = TOTAL PATH + A
End
```

According to Algo. 2, if the number of the cells that the path passes through is p, the minimum and maximum number of middle targets will be as follows:

$$\frac{p}{3} \le len(L) = n \le p \tag{3}$$

Finally, the cost function of (1) can be stated as follows:

$$F(n) = \left(g(m_1) + h(m_1)\right) + \ldots + \left(g(m_n) + h(m_n)\right) \quad (4)$$

As before m_1 to m_{n-1} are middle targets and m_n is the original end target. Using 'b' symbol to show the corresponding cell (or cell-set) borders of each intermediate target in (2), we'll have

$$B = b1, b2, \ldots, bn \quad (5)$$

Defining the distance function d(i,j) between points i and j as

$$d(i, j) = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2} \quad (6)$$

Gives the following equation for the path to middle target m_k in cell b_k of size r*r

$$m_k = \min_{g=1, \ldots, 4*r} (d(b_k[g], m_n)) \quad (7)$$

In this equation g = 4r is the maximum external boundary points in an r*r cell.

C. Map Modification and Planning Method

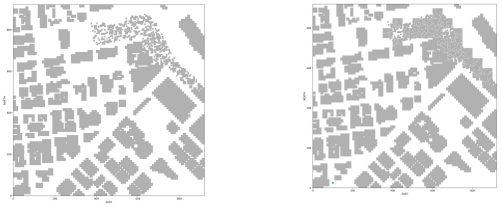
Unlike the previously proposed methods, whose main idea was to change the algorithm in each cell, we don't want to modify the A* algorithm here. The general idea of this method is to modify the environment map instead. The modification of the map eliminates the possibility of the main path getting close to risky passages that are close to obstacles without eliminating narrow passages. In cells with narrow passages (MA), all passages but MA are deleted. Algo. 3 shows the pseudo-code for this method. We don't need to change the algorithms in run time in this method. As the risky passages are closed, the method searches faster. As we don't change the algorithm here, easily updating to another version of the A* algorithm can be considered another advantage of this algorithm. Fig. 3-a and Fig. 3-b show the environment map before and after applying the modifications. This will be our method of choice for combining A* and medial axis algorithms. We call it the medial axis based on A* (MAA*). It ensures reaching a safe, optimal path without any constraint on the A* algorithm.

Algorithm 3: Map Modification and Planning Method

Input: *2D real map*

Output: *path planning with method 3 of MAA* algorithm*

1 ***B*** *= create a binary map from real map*
2 ***S, G*** *= Initialize start and goal points on the binary map* ***B*** *in line 1*
3 ***M*** *= extract medial axis from* ***B***
4 *Divide* ***B*** *and* ***M*** *into 50*50 blocks*
5 *Label the blocks in line 4 with A* or Medial Axis algorithm*
6 *Set all points in* ***B*** *with MA labeled block to one (make them obstacles)*
7 *Set all points in M with MA labeled block to zero (make them free space)*
8 *Applying the traditional A* algorithm to design the route from* ***S*** *to* ***G***
9 *End*



a) Before modifying the map b) After modifying the map

Fig. 3. The third proposed algorithm

V. SITL Test Results and Analysis

This section presents the proposed methods' results, SITL evaluations, and analysis of them. The presented measurements and simulation results will be quantitative and precise to make a merit-based selection of the best possible method. We will discuss the advantages and disadvantages of each combination algorithm according to its results. We use the same path and map for all of them for better comparison and consistency (Fig. 4). The path starts at (100, 25) and ends at (900, 900). Table I. compares the combination methods under the same conditions. In this table, the safety margin is 2 (m) for A* with safety margin and 1 (m) for the other proposed methods. The table shows that the first method gives a pretty enhanced output; however, includes more computation time. The path is also longer than the third method caused by the 'breaks' in the middle due to the change of the algorithm. In the second proposed method, the path is suboptimal, but the computation time is less than in the first method. This can be a method of choice when time is of prime importance. The third method is the most optimal and has a much better time performance than the first one. It makes it our best method (Fig. 4 shows the paths designed by 4 different methods according to Table I.).

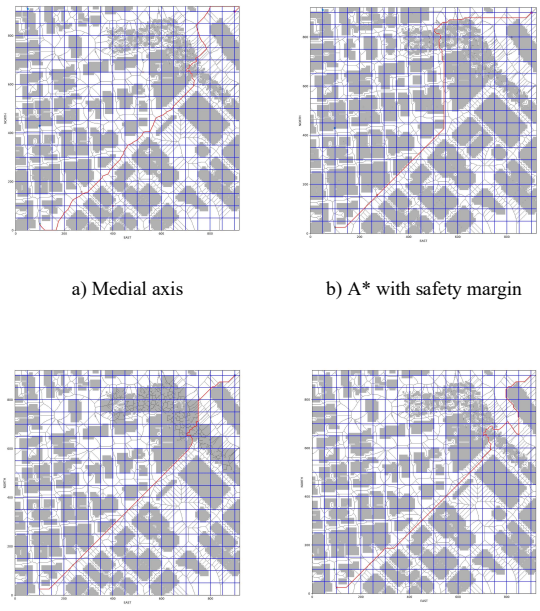
MAA* performs better than A* in avoiding obstacles. It doesn't mean MAA* is generally more optimal than A*. MAA* ability to pass through narrow passages without colliding with obstacles gives it shorter paths. Narrow passages are not blocked as in A* with a safety margin. Let's assume the flying robot has a positioning error of 1.5 meters. The safety margin in A* should be 1 (m) to block the narrow passages, as in Fig. 5-a. In the binary map in Fig. 5, black and white points represent obstacles and open spaces, and the distance between each pair of points is 1 (m) as well. Setting a 1-meter safety margin blocks the open path in both Fig. 5-a and Fig. 5-b. However, as shown by the blue arrow in Fig. 5-a, MAA* algorithm follows a zigzag path. If we use an appropriate control algorithm, as in the simulations, the moving agent will follow the yellow path in Fig. 5-a, where

the distance to the obstacles will be 1.5 meters. However, in A*, the safety margin should be 1 meter or the distance between each pair of points in the environment should be half a meter. This doubles the points and computations. Due to the 1.5-meter positioning error of the flying robot, the path in Fig. 5-b cannot be taken. Increasing the safety margin to block that path will also block the path in Fig. 5-a. However, using the red box pattern in Fig. 5-b, i.e., an open point with two obstacle points at the sides, closes the Fig. 5-b path and leaves Fig. 5-a path open, which is suitable for MAA* combined algorithm.

TABLE I. COMPARISON OF THE THREE COMBINATION METHODS

Type Of Algorithm	*Computation Time(s)*	*Path Lenth(m)*
A* without safety margin	27. 806	972
A* with safety margin	21.545	1254
Medial Axis	1.452	1286
First combination algorithm	26,884	1021
Second combination algorithm	17.514	1199
Third combination algorithm	24.107	997

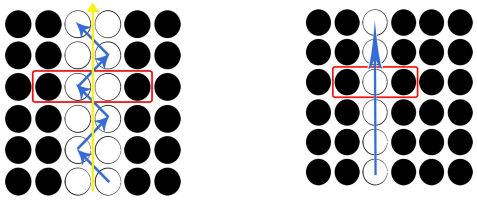
All path planning computations in tables I. and II. are done using colab.research.google.com. This site gives Intel® Xeon® two-core CPU at 2.3 GHz, a two-core GPU at 1.59 GHz, and 12 GB of RAM as its equivalent computational infrastructure.



a) Medial axis b) A* with safety margin

c) Third combination algorithm d) Second combination algorithm

Fig. 4. Narrow passage test



A) Binary map with zigzag path B) Binary map with a straight path

Fig. 5. Binary map

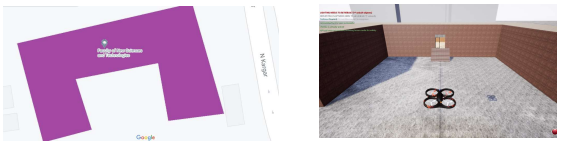
A. SITL Evaluations

In this section, we give simulation results to verify and complement the optimizations of MAA* algorithm. Considering the better graphic capabilities of Unreal Engine, the simulations used version 4.25 of this game engine. An environment resembling The Faculty of New Sciences and Technologies, University of Tehran, was designed for the required tests. Fig. 6-a shows a satellite image of the faculty, and Fig. 6-b shows the simulation environment inside the faculty. To use an appropriate quadcopter, we used Airsim, which uses a carrot-chasing algorithm by default. This is often considered a 'high-level' algorithm since it works by just specifying the waypoints.

The three proposed path planning methods, namely the third combination algorithm, A* with safety margin and A* without safety margin, are simulated to show the advantages and disadvantages of each. Fig. 7-b indicates the ability of the combined algorithm to pass through narrow passages without colliding with the obstacles. Obstacles data was produced to create two-dimensional readable maps for the algorithms in the simulator. Fig. 7-a shows the path planned by MAA*. Fig. 8 illustrates the flying robot's path from takeoff to landing and becoming dormant in the three-dimensional reference system. Table II. demonstrates that considering the time and path length, MAA* performs much better than A* with safety margin in the simulation.

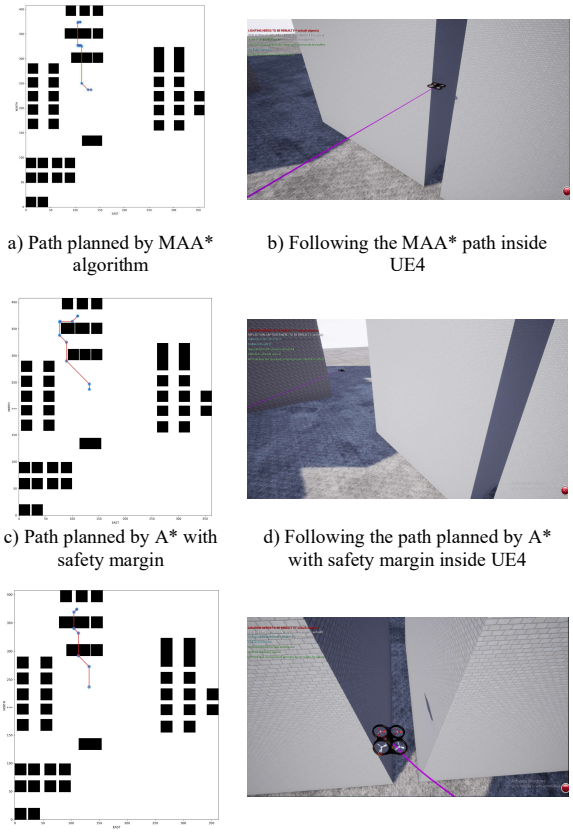
TABLE II. COMPARISON OF THE THREE COMBINATION METHODS

Algorithm	*Average computation time (s)*	*Path length (m)*
MAA* algorithm, the third method	5.412	154
A* with safety margin	7.284	161
A* without safety margin	5.268	138



a) Google-Map image of the Faculty of New Sciences and Technologies b) Simulation environment in UE4

Fig. 6. Graphical 6 DOF simulation environment



a) Path planned by MAA* algorithm

b) Following the MAA* path inside UE4

c) Path planned by A* with safety margin

d) Following the path planned by A* with safety margin inside UE4

e) Path planned by A* without safety margin

f) Following the path planned by A* without safety margin inside UE4

Fig. 7. Different SITLS Simulations

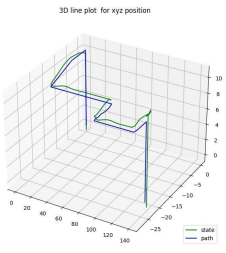
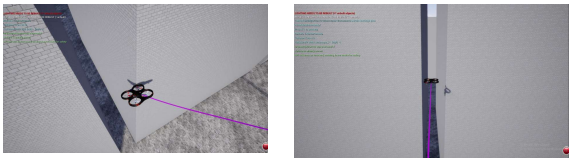


Fig. 8. Three-dimensional diagram of quadcopter's subsequent path

Fig. 7-c demonstrates that A* with a safety margin cannot pass narrow passages and if forced to go around the building in the simulation, as is evident in Fig 7-d; No collision happened. Fig. 7-e shows that A* without safety margin can pass through narrow passages but with two collisions. It nevertheless follows that path till reaching the end target; Fig. 7-f shows the collision moment.

Adding an atmospheric disturbance of a constant forward-moving (x-axis direction) wind with a 14 m/s velocity has an evident effect on the flying robot. Fig. 9 shows the moment of the flier's collision in the A* algorithm without safety margin. Contrary to the collision in Fig. 7-f, this collision stops the flier and disrupts its path. This clearly shows the negative effect of the wind disturbance on the A* Algorithm without a safety margin.



a) Following the A* without safety margin algorithm path inside UE4 with added wind

b) Following the MAA* path inside UE4 with added wind

Fig. 9. SITL Simulation with added atmospheric disturbances

As shown in Fig. 9-b, the same noise doesn't affect MAA* algorithm performance much and the flier continues its path.

VI. Conclusions

This research aimed to improve path planning for aerial robots in urban environments by combining various algorithms and creating a Software-in-the-Loop (SITL) evaluation framework. Our work resulted in an optimized and secure method tailored to this challenging task. Our simulations highlighted challenges in environments with narrow passages, affecting path length and computation time in traditional algorithms like A* , especially when safety margins are considered. A faces collision issues without safety margins, which our proposed Medial Axis A* (MAA*) algorithm effectively overcomes. MAA* offers three key advantages over A*: adept navigation through narrow passages crucial for urban aerial robots as visually depicted in Figure 7, enhanced safety in negotiating obstacles corners with precision as evidenced in Figure 9, and increased efficiency by reducing search efforts near obstacles, resulting in shorter planning times compared to A*. MAA* is versatile, adapting to dynamic scenarios with mobile obstacles. Using satellite imagery instead of pre-existing maps for identifying narrow passages is a potential improvement. Opportunities also exist to combine the Medial Axis with other approaches like Artificial Potential Fields and Probabilistic Roadmaps, paving the way for innovative path planning techniques. In conclusion, this research advances path planning for aerial robots in complex urban settings. Combining the core algorithm with path-smoothing methods for both dynamic and static constraints holds promise for future research. Our contributions aim to enhance aerial robotics, ensuring safe and efficient navigation in urban environments, and inspiring further exploration in this dynamic field.

References

[1] B. K. Patle, G. Babu L, A. Pandey, D. R. K. Parhi, and A. Jagadeesh, "A review: On path planning strategies for navigation of mobile robot," Def. Technol., vol. 15, no. 4, pp. 582–606, 2019.

[2] L. Yang, J. Qi, D. Song, J. Xiao, J. Han, and Y. Xia, "Survey of Robot 3D Path Planning Algorithms," J. Control Sci. Eng., vol. 2016, pp. 1–22, Jan. 2016.

[3] L. E. Kavraki, P. Svestka, J.-. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," IEEE Trans. Robot. Autom., vol. 12, no. 4, pp. 566–580, 1996.

[4] O. Khatib, “Real-time obstacle avoidance for manipulators and mobile robots,” in Proceedings. 1985 IEEE International Conference on Robotics and Automation, 1985, vol. 2, pp. 500–505.

[5] P. E. Hart, N. J. Nilsson, and B. Raphael, “A Formal Basis for the Heuristic Determination of Minimum Cost Paths,” IEEE Trans. Syst. Sci. Cybern., vol. 4, no. 2, pp. 100–107, 1968.

[6] C. Fonlupt, D. Robilliard, and P. Preux, “Preventing Premature Convergence via Cooperating Genetic Algorithms,” Proc. Mand., vol. 60, pp. 1–6, 1997.

[7] and S. C. A. M. G. Roy Glasius, Andrzej Komoda, “Neural Networks Dynamics for path planning, and obstacles avoid.” 1994.

[8] R. S. Sutton and A. G. Barto, “Reinforcement learning: An introduction, 2nd ed.,” Reinforcement learning: An introduction, 2nd ed. The MIT Press, Cambridge, MA, US, pp. xxii, 526–xxii, 526, 2018.

[9] H. Shin and J. Chae, “A Performance Review of Collision-Free Path Planning Algorithms,” Electronics, vol. 9, p. 316, Feb. 2020.

[10] H. Yu, M. Xingang, S. Wang, and Y. Hu, “Nursing Robot Safety Path Planning Based on Improved A star Algorithm,” 2019.

[11] T. Chen, G. Zhang, X. Hu, and J. Xiao, “Unmanned aerial vehicle route planning method based on a star algorithm,” in 2018 13th IEEE Conference on Industrial Electronics and Applications (ICIEA), 2018, pp. 1510–1514.

[12] S. Erke, D. Bin, N. Yiming, Z. Qi, X. Liang, and Z. Dawei, “An improved A-Star based path planning algorithm for autonomous land vehicles,” Int. J. Adv. Robot. Syst., vol. 17, no. 5, pp. 1–13, 2020.

[13] E. Hansen, S. Zilberstein, and V. Danilchenko, “Anytime Heuristic Search: First Results,” Oct. 1998.

[14] A. Stentz, “Optimal and Efficient Path Planning for Partially-Known Environments,” 1994 Int. Conf. Robot. Autom., vol. 4, Feb. 2000.

[15] F. Duchon et al., “Path planning with modified A star algorithm for a mobile robot,” Procedia Eng., vol. 96, pp. 59–69, 2014.

[16] E. W. Dijkstra, “A note on two problems in connexion with graphs,” Numer. Math., vol. 1, no. 1, pp. 269–271, 1959.

[17] K. Benzaid, R. Marie, N. Mansouri, and O. Labbani-igbida, “Filtered Medial Surface Based Approach for 3D Collision-Free Path Planning Problem,” vol. 2018, 2018.

[18] A. K. Guruji, H. Agarwal, and D. K. Parsediya, “Time-efficient A* Algorithm for Robot Path Planning,” Procedia Technol., vol. 23, no. April, pp. 144–149, 2016.

[19] N. Bo, X. Li, J. Dai, and J. Tang, “A Hierarchical Optimization Strategy of Trajectory Planning for Multi-UAVs,” in 2017 9th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC), 2017, vol. 1, pp. 294–298.

[20] Z. Zhang, T. Long, Z. Wang, G. Xu, and Y. Cao, “UAV Dynamic Path Planning using Anytime Repairing Sparse A * Algorithm and Targets Motion Estimation (IEEE / CSAA GNCC)*,” 2018 IEEE CSAA Guid. Navig. Control Conf., pp. 1–6.

[21] E. Masehian, “A Voronoi Diagram – Visibility Graph – Potential Field Compound Algorithm for Robot Path Planning,” no. May 2018, 2004.

[22] M. Costa and M. Silva, A Survey on Path Planning Algorithms for Mobile Robots. 2019.

[23] M. Xu et al., “AN INDOOR NAVIGATION APPROACH CONSIDERING OBSTACLES AND SPACE SUBDIVISION OF 2D PLAN,” vol. XLI, no. July, pp. 12–19, 2016.

[24] B. MacAllister, J. Butzke, A. Kushleyev, H. Pandey, and M. Likhachev, “Path planning for non-circular micro aerial vehicles in constrained environments,” in 2013 IEEE International Conference on Robotics and Automation, 2013, pp. 3933–3940.

[25] J. Chen, M. Li, and Z. Yuan, “An Improved A * Algorithm for UAV Path Planning Problems,” no. Itnec, pp. 958–962, 2020.

[26] M. Lin, K. Yuan, C. J. Shi, and Y. Wang, “Path Planning of Mobile Robot Based on Improved A * Algorithm,” pp. 3570–3576, 2017.

[27] Y. Qu, Q. Pan, and J. Yan, “Flight Path Planning of UAV Based on Heuristically Search and Genetic Algorithms,” pp. 45–49, 2005.

[28] G. Tang, C. Tang, C. Claramunt, X. Hu, and P. Zhou, “Geometric A-Star Algorithm: An Improved A-Star Algorithm for AGV Path Planning in a Port Environment,” IEEE Access, vol. 9, pp. 59196–59210, 2021.

[29] C. Zhang and H. Liu, “Analysis for UAV Heuristic Tracking Path Planning Based on Target Matching,” 2018 9th Int. Conf. Mech. Aerosp. Eng., pp. 34–39, 2018.

[30] T. Thoa, C. Copot, D. Trung, and R. De Keyser, “Heuristic Approaches in Robot Path Planning : A Survey,” Rob. Auton. Syst., 2016.

[31] Z. He and L. Zhao, “The Comparison of Four UAV Path Planning Algorithms Based on Geometry Search Algorithm,” in 2017 9th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC), 2017, vol. 2, pp. 33–36.

[32] S. Benders and S. Schopferer, “A line-graph path planner for performance constrained fixed-wing UAVs in wind fields,” in 2017 International Conference on Unmanned Aircraft Systems, ICUAS 2017, 2017, pp. 79–86.

[33] Z. Lv, L. Yang, Y. He, Z. Liu, and Z. Han, “3D environment modeling with height dimension reduction and path planning for UAV,” in 2017 9th International Conference on Modelling, Identification and Control (ICMIC), 2017, pp. 734–739.

[34] J. Li, G. Deng, C. Luo, Q. Lin, Q. Yan, and Z. Ming, “A Hybrid Path Planning Method in Unmanned Air/Ground Vehicle (UAV/UGV) Cooperative Systems,” IEEE Trans. Veh. Technol., vol. 65, no. 12, pp. 9585–9596, 2016.

[35] S. K. Gupta, P. Dutta, N. Rastogi, and S. Chaturvedi, “A control algorithm for co-operatively aerial survey by using multiple UAVs,” in 2017 Recent Developments in Control, Automation & Power Engineering (RDCAPE), 2017, pp. 280–285.

[36] J. Kwak and Y. Sung, “Autonomous UAV Flight Control for GPS-Based Navigation,” IEEE Access, vol. 6, pp. 37947–37955, 2018.

[37] X. Sun, W. Yao, Y. Liu, and N. Qi, “Triple-stage path prediction algorithm for real-time mission planning of multi-UAV,” Electron. Lett., vol. 51, pp. 1490–1492, 2015.

[38] B. Y. Li, H. Lin, H. Samani, L. Sadler, T. Gregory, and B. Jalaian, “On 3D autonomous delivery systems: Design and development,” in 2017 International Conference on Advanced Robotics and Intelligent Systems (ARIS), 2017, pp. 1–6.

[39] Z. Chengjun and M. Xiuyun, “Spare A∗ search approach for UAV route planning,” in 2017 IEEE International Conference on Unmanned Systems (ICUS), 2017, pp. 413–417.

[40] X. Song and S. Hu, “2D path planning with dubins-path-based A∗ algorithm for a fixed-wing UAV,” in 2017 3rd IEEE International Conference on Control Science and Systems Engineering (ICCSSE), 2017, pp. 69–73.

[41] X. Ma, Z. Jiao, Z. Wang, and D. Panagou, “3-D Decentralized Prioritized Motion Planning and Coordination for High-Density Operations of Micro Aerial Vehicles,” IEEE Trans. Control Syst. Technol., vol. 26, no. 3, pp. 939–953, 2018.

[42] B. Penin, P. R. Giordano, and F. Chaumette, “Minimum-Time Trajectory Planning Under Intermittent Measurements,” IEEE Robot. Autom. Lett., vol. 4, no. 1, pp. 153–160, 2019.

[43] H. Liang, H. Bai, R. Sun, R. Sun, and C. Li, “Three-dimensional path planning based on DEM,” in 2017 36th Chinese Control Conference (CCC), 2017, pp. 5980–5987.

[44] Z. Mengying, W. Hua, and C. Feng, “Online path planning algorithms for unmanned air vehicle,” in 2017 IEEE International Conference on Unmanned Systems (ICUS), 2017, pp. 116–119.

[45] “1050 Market St - Google Maps.” [Online]. Available: https://www.google.com/maps/place/1050+Market+St,+San+Francisco,+CA+94111,+USA/@37.7922344,-122.3985024,164m/data=!3m1!1e3!4m5!3m4!1s0x80858063e43d2d0f:0x8d1e43e1feee3fcf!8m2!3d37.7929764!4d-122.3967721. [Accessed: 25-Jan-2022].