2021 International Conference on Unmanned Aircraft Systems (ICUAS)
Athens, Greece. June 15-18, 2021

A UAV Global Planner to Improve Path Planning in Unstructured Environments

Lidia Rocha[1], Marcela Aniceto[1], Igor Araújo[1] and Kelen Vivaldini[1]

Abstract— **A good performance for path planning is essential to carry out real-world missions. In this paper, the path planning consists of a global planner, that finds the optimal path, and in a local planner, recalculates the path to avoid obstacles. The main focus is to improve the performance of local planning techniques decreasing the complexity. There are two main ways to do it: bidirectional algorithms, improving time, and global planners, improving time and completeness. Thus, we propose a global planner algorithm that generates auxiliary nodes, backtracking by the goal node. We perform a comparison among A*, Bi A*, Artificial Potential Field (APF), Bi APF, Rapid Exploring Random Tree (RRT), and Bi RRT with and without the global planner through statistical metrics of time, path length, CPU, and memory. The results show the advantages of using bidirectional algorithms and the proposed global planner. The bidirectional algorithms decrease the time to return to the trajectory and sometimes assist in the algorithm's completeness. The proposed global planner reduced the planning time by 91.6% and improved the completeness of all algorithms in an unstructured indoor environment.**

I. Introduction

In recent years, Unmanned Aerial Vehicle (UAV) have been adopted to carry out important tasks in several interesting applications in unstructured environments [1], enabling various research segments, such as the monitoring of eucalyptus [2], underground mine environments [3], surveillance [4], urban environment [5], search and rescue [6], [7], and others. To carry out these tasks efficiently a path planning that aims to solve unstructured environments is essential.

Path planning for UAVs aims to find an optimal path, avoiding obstacles [8]. The planner needs to make optimal decisions to solve UAV's mission-critical operations. These decisions require a map or graph of the mission scenario so that UAVs are aware of their locations and obstacles [9], [10].

The main techniques that uses maps are classical and widely employed to solve path planning problems [11], [12] given the fact that they are easy techniques implementation and have good results. Among the classic techniques, were chosen from the 3 main subgroups: The RRT of the sampling-based algorithm group, the A* (A-Star) of the graph search group, and the Artificial Potential Field (APF) of the potential field group.

According to [13], classical techniques have disadvantages because they have high computational costs and problems when working with uncertainties in the scenarios. These algorithms tend to have significant complications to return a trajectory in large and unstructured environments due to the computational cost, increasing uncertainties [14].

Thus, to reduce these problems, the following bidirectional techniques of these algorithms were developed: Bidirectional RRT [15], Bidirectional A* [16], and Bidirectional APF [17]. The goal of those techniques is to divide the algorithm's complexity to improve response time and completeness.

In unstructured outdoor environments, the UAVs need to consider many obstacles and different shapes, like forests. Furthermore, in unstructured indoor environments, the UAVs tends to get lost during navigation due to many narrow path options [6], like caves. In that scenario, the local planner algorithm has difficulty turning back at the initial nodes, aiming to choose a better path when it is significantly ahead.

The classic algorithms generate a trajectory in an indoor environment with a high computational cost because it has more obstacles and has fewer trajectory options, consequently, taking a longer time. In outdoor environments, there is the possibility that not all algorithms return a trajectory due to the high complexity of the environment, causing the algorithms to fall into a minimum location [13]. On the other hand, with bidirectional algorithms, the complexity of both environments will be decreased. In this way, the response time will be shorter and may improve the completeness, potentially finding a viable path [18].

However, there are still scenarios in which bidirectional algorithms may not find a viable path. Even if the complexity is reduced by half, there is still a possibility that the algorithm will fall to a minimum local in a large environment. However, to carry out missions in the real-world, the path planning algorithms are necessary once they are optimized to work within milliseconds to support real-time without collisions [14].

One way to optimize the algorithm's completeness and even better, the response time, is with the adoption a global planner, which is responsible for showing the shortest path for the UAV carry out, consequently bringing lower computational and time costs for the local planner. In addition that, to preventing the algorithm from falling into local minima, improving its completeness [19].

This article proposes a global planner that uses the auxiliary nodes in areas where the local planning algorithm's complexity is high. Lessening the algorithm's chances of falling to a local minimum. In the same way, the algorithm's time to return a trajectory is also reduced since the complexity decreases.

[1]Lidia, Marcela, Igor and Kelen authors are with Computing Department, Federal University of São Carlos, São Carlos lidia@estudante.ufscar.br, marcelaaniceto@gmail.com, higuinho@gmail.com, vivaldini@ufscar.br

According to [20], [21], [22], as the algorithm's implementation is geometrical, 3D validation would bring the same results as a 2D validation. Additionally is possible to fly in certain unstructured environments considering only two dimensions, as forests [23], [24], which also need, to decrease the complexity of the algorithms to return a trajectory on time.

In this context, the proposed global planner was validated in 2D to obtain more accurate data about time planning and the distance between the start and goal nodes. Along with that, this paper describes the complexity analysis of each algorithm and proves them by analyzing the results. Therefore, it is possible to know the time to return the trajectory and the use of memory of the 3D path planners. The main difference for 3D planning is that the auxiliary points can be placed in 3 dimensions.

This paper is organized as followed: Section II presents a brief description of the local planning approach; Section III describes the proposed algorithm for the global planner; Section IV explain the environment of simulation and how each data was collected; the tests in simulated environments and discussion are given in Section V; finally, Section VI presents the conclusion and future works.

II. Local Planning Techniques

This section presents a brief description of each techquines implemented, such as A-Star (A*), Bidirectional A-Star (Bi A*), Rapidly-Exploring Random Tree (RRT), Bidirectional Rapidly-Exploring Random Exploring Tree (Bi RRT), Artificial Potential Field (APF), and Bidirectional Artificial Potential Field (Bi APF). Besides, it shows how the paths are smoothed and the time and space complexity of the algorithms.

*A. A**

The A* algorithm, developed by [25], aims to improve the computational cost and processing time presented by the graphs in search problems. The algorithm is based on a heuristic, defined to guide the search process [26]. This heuristic estimates the cost of traveling from the origin to the destination, usually determined by the distance between the nodes, which tends to accelerate the exploratory process. The algorithm also tracks the cost needed to reach each position (f_n), which 3can be seen in Equation 1, where the cost is g_n, the heuristic is h_n, and the n means that refers to the current node..

$$f_n = g_n + h_n \tag{1}$$

*B. Bi A**

Bidirectional A* starts two searches parallel from start to finish and one from finish to the goal node until they find each other. A big tree is worse than two small trees, so, computationally, it is better to have two small search trees.

The redirect approach abandons simultaneous searches in the forward and backward directions. Instead, it performs a forward search for a short time, chooses the best forward candidate, and then performs a backward search, not to the starting point, but to that candidate. It chooses the best backward candidate and performs a forward search from the best forward candidate to the best backward candidate. This process continues until the two candidates are on the same point [16].

C. RRT

The RRT generates nodes randomly and is connected to the closest available node [27]. The algorithm expands at free space towards randomly positioned nodes. In this way, a tree is built, checking if there is a collision in each interaction. If the distance to the random node and the closest vertex is greater than the growth factor, it is limited to that factor's value.

D. Bi RRT

RRT variation was implemented to improve its performance [28], the bidirectional RRT, which uses two RRT [29] trees. One of the trees is born from the source node, and the other from the destination node, for this reason, being called RRT-bidirectional [30], [31]. As the RRT tree grows, the algorithm tries to connect both trees while they are expanded. If it succeeds, it will be considered the last expansion of the tree. Otherwise, the two trees change roles, the tree that was expanding now will try to connect, and the other will expand [32]. If the path is still not found, the first tree expands to the closest possible point [33].

The bidirectional RRT divides at the computational cost between the exploration of the environment and the growth of trees, one towards the other, until they are connected, and the solution found [15].

E. APF

A Potential Field is any physical field that fulfills Laplace's equation. Potential Fields in the world are calculated for each search between two nodes. The UAV always goes to the next position with the lowest potential, analyzing every 10 centimeters. The electrical-charge goes to a node where the electrical potential is lower until reaching the goal. Some common examples of Potential Fields include electrical, magnetic, and gravitational fields.

The Attractive-Repulsive Potential method is based on an attractive Potential Field to the target and a repulsive Potential Field to the world's obstacles. The sum of these two potential gives us the robot's current potential (Eq.2).

$$U(n) = U_{att}(n) + U_{rep}(n) \tag{2}$$

The attractive potential depends on the distance from the target. Where x and y are coordinates of the current node, x_{goal} and y_{goal} are the goal node. C is a constant. KP is a control variable for the gradient magnitude (Eq.3).

$$U_{att}(n) = C * KP * \sqrt{(x - x_{goal})^2 + (y - y_{goal})^2} \tag{3}$$

The repulsive potential depends on distances from obstacles, being calculated according to each obstacle, and the sum of them is the total repulsive function. This force is useful in keeping the robot away from the boundaries. Where rr is the robot radius, C is a constant, ETA is a repulsive potential gain and x_o and y_o are coordinates of the obstacles (Eq.4).

$$U_{rep}(n) = C*ETA*(\frac{1}{\sqrt{(x-x_o)^2+(y-y_o)^2}} - \frac{1}{rr})^2 \quad (4)$$

F. Bi APF

The artificial potential field can also occur bidirectional, as implemented by [17]. The authors vary the checkpoint for choosing the next node between the starting and target node at each iteration. However, we implemented the exchange between the source and objective node to occur only when one of the searches is at a local minimum. The objective node makes the first calculation of the potential field. As proposed by [17], there would still be cases where local minimums would occur, especially in unstructured and unknown environments, similarly, the path would be very winding due to so many changes to the field.

G. UAV Constraints

According to [34], the trajectories to be followed by UAVs need to minimize the jerk while maintaining constant torque. We implemented an improvement that reduces the number of curves in the trajectory, checking unnecessary nodes in the curve and delete them [35].

The path needs to be smoothed, aiming at continuous movement [36] [37]. For this reason, b-spline curves were applied to the trajectory.

Cubic Spline interpolation is an approximation technique that divides the interval of interest into several subintervals and interpolates, as smoothly as possible, based on cubic polynomials [38]. Thus, the trajectory generated is smoother, making it easier for the UAV to carry out.

H. Complexity Analysis

The time complexity of A* in the worst case is $bigO(MovesObsN)$. N is the path length; $Moves$ is the number of possible directions that the UAV can make (in a discrete environment are 4: up, left, right, down); Obs is the number of obstacles in the environment. The complexity is directly proportional to the path length and the number of obstacles in the environment. By reducing one of these, the algorithm tends to have a faster response.

The space complexity A* is $bigO(2N + 2E)$. N is each of the arrays where the (x, y) coordinates of the path will be stored, and E is each of the arrays that will keep the environment's coordinates. Using arrays instead of a matrix to save the path makes it possible to access more quickly (in Python) and make the code more flexible, as well as saving the environment, due to the fact that only spaces with obstacles are known, so more memory would be being used than necessary in contrast to using an array.

The time complexity of Bi A* is $bigO(bigO(A*^2))$ because despite the instructions being executed in the same loop, the functions to find the best path are repeated, influencing the growth rate in all aspects. Bi A*'s space complexity is the same as A*, although each array of the trajectory is only created up to half of the path, four arrays are used, being equivalent to A*.

The RRT time complexity in the worst case is $bigO(ItObs)$. It is the number of iterations, and Obs the number of obstacles. The best case is when it gets the least number of iterations. Depending on randomness, that is, what will be investigated in the next node.

The RRT space complexity is $bigO(2N)$ as Bi RRT. Both complexities are the same since the algorithm returns a node at each iteration, and it is unnecessary to create an array for each side of the Bi RRT. The nodes can be implemented directly in just one array. The time complexity of Bi RRT is $bigO(2bigO(RRT))$ because the iterations run in sequence until they reach the objective node.

The APF time complexity in the worst case is $bigO(N + Obs)$. The best case is directly proportional to the path length. Obs is added because the array with obstacles is generated at the algorithm's initialization and reused. The Bi APF time complexity is $bigO(2bigO(APF))$ for the same reason as the time complexity between RRT and Bi RRT. The space complexity of APF and Bi APF is $bigO(2N)$, for the same reason as RRT and Bi RRT.

In all the algorithms presented, the worst case will be the same as the expected case. The $bigO(Obs)$ complexity cannot be improved in any of the algorithms as it will always be necessary to check all obstacles for a collision. Therefore, by decreasing the number of obstacles in the environment, the algorithm tends to be faster. The complexity of APF and Bi APF are less affected by the number of obstacles in the environment.

III. Proposed Global Planner

This Section explains the proposed global planner and how it works.

A. Motivation

The proposed global planner's main goal is to decrease the algorithms' time complexity, making them return the path faster and carry out missions that they previously did not manage. This algorithm's general idea is to break the trajectory's complexity into several auxiliary points, making it easier for the local planner to find the best path. In this way, several smaller trajectories are created, decreasing the algorithms' complexity and increasing the completeness.

As seen in Section II-H, A*, Bi A*, RRT, and Bi RRT are algorithms that have their complexity proportional to the number of obstacles in the scenario. The complexity of the APF and Bi APF depends on the obstacles but on a smaller scale. The proposed global planner will check collision with all obstacles in the scenario. Consequently, these constants in time complexity are unchanged.

The APF and Bi APF are the most improved algorithms with our algorithm because their complexity depends only on the number of obstacles, on a small scale, and the path length, decreasing considerably. The other algorithms are more influenced by the number of obstacles and variables in which the global planner does not directly optimize, such as the number of iterations. For the other algorithms, the proposed global planner will shorten the time to return the path and improve the completeness, but on a smaller scale than APF and Bi APF.

A* and Bi A* also depend on the path length and the number of obstacles. Even though the trajectory size is reduced, the complexity will not be as reduced as APF and Bi APF because the time complexity is proportional to the number of obstacles. The proposed algorithm also optimizes RRT and Bi RRT because the auxiliary node are closer than the goal node. Hence, less iterations are necessary. Finally, if the algorithm follows numerous auxiliary nodes to the goal node, there is less chance of RRT and Bi RRT being lost in the middle of the path, increasing its completeness.

B. Proposed Algorithm

The proposed algorithm creates auxiliary nodes from the goal until the start node, while also creating two maps, the first starting from the goal node (m1) and the second starting from the start node (m2). Each map considers the distance between the current and goal or start node and the number of obstacles between them. As shown in Equation 5, where d is the distance between the nodes, K_n is the number of obstacles in a line of sight until the current node, and L is the penalty when the trajectory between the current and goal node crosses a obstacle.

$$loss = \log_d + L * K_n \tag{5}$$

This equation is based on the attenuation formula, shown in [39]. The distance value is given in logarithm to prioritize values that are most likely to have no collision. Those are, the values closest to the beginning.

The first map helps determine which are the best auxiliary nodes to be followed from the goal node. These are the nodes that had no collision and have a lower value of $loss$, calculated with the Equation 5. The second map aims to guide what are the best direction to be followed. For example, the first map can generate many better points to follow. However, if we consider the second map, the best path to follow will go towards the start node. Therefore, the next step is to create a heat map by adding both maps.

With the heat map generated, considering the relation between the start and goal node, final adjustments are made to UAV that does not loop in the scenario and does not collide with any obstacle. Firstly, depending on the UAV, the algorithm checks whether the radius and current position do not collide with any wall. If true, this node is disregarded, and the algorithm searches for a new best auxiliary node. Otherwise, the algorithm checks if the node is in the same direction from where it was explored. With it is possible to avoid repeating the nodes, which do not go towards the goal node. If it is in the same direction, a new node is requested. If not, that node is the auxiliary node.

The algorithm saves the angle between the current and next node in each movement. With this information, it is possible to compare the angles formed between the previous and the current node. If the angles are on opposite sides in the cartesian plane, the UAV is going in the same direction as the last node. For example: In the first movement, the UAV makes an angle of 45°, and in the second, it would make an angle of 225°. The second node would be creating a new node in the same direction as the first, probably creating a loop.

Auxiliary nodes are chosen based on the values of this new map. This choice can be seen in Figure 1, which uses the indoor scenario, presented in Section IV-C, to exemplify algorithm operation.

The red nodes have highest values while the blues, the lowest value. It is possible to see the sum of the first two maps, mentioned previously, in Figure 1 (a). The blue nodes directly connect with the start node, and the nodes with an even darker blue, almost black, left the goal node without collision. When adding both maps, we have that the best node to be chosen as an auxiliary node is the one that has the highest value among the nodes that had no collision. That is the node with no collision and has more influence with the path from the start node.

Figure 1 (a) shows the goal node and the heat map used to define the first auxiliary node. Figure 1 (b) shows the chosen auxiliary node and the new heat map used to discover the next auxiliary node. Finally, Figure 1 (c) shows the chosen auxiliary node and the new heat map to be considered. In this case, the next node will already be the start node. The local planning algorithms generate a trajectory until each of the auxiliary nodes until reaching the goal node. This algorithm's workflow can be seen in Figure 2.

IV. Methods

In this section, we present the preliminary information for the paper and explain the scenarios used in the simulations. The simulations were carry out on a notebook with an Intel Core i7 7700HQ processor with 16 GB of RAM and an Intel® HD Graphics 620.

A. Simulated Environment

In this research, we use MRS UAV system in the Gazebo simulator [40], Robot Operating System (ROS) [41] for communication, and the programming language Python 2.7.

In this system, we use UAV F-450. It is a quadcopter kit has a low-cost and is easy to acquire and assemble [42]. This model has connections to install microcontrollers and micro computers, such as Jetson for the use of Cuda, RPLidar, laser scanner, blue fox camera, among others [43].

The UAV uses the structure showed in Figure 3. The MRS system provides an estimation system for supporting flights through realistic simulations. The technique used for localization was the visual-inertial state estimation (VIO) [44]. The control is also provided by the system, the control adopted was Model Predictive Control (MPC), which stands out when applied to problems with many manipulated control variables and their restrictions, changes in control objectives, equipment failure tolerance, and time delays [45]. The local

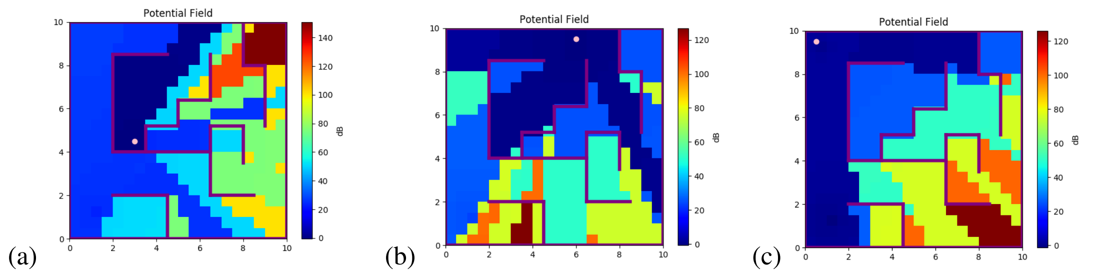
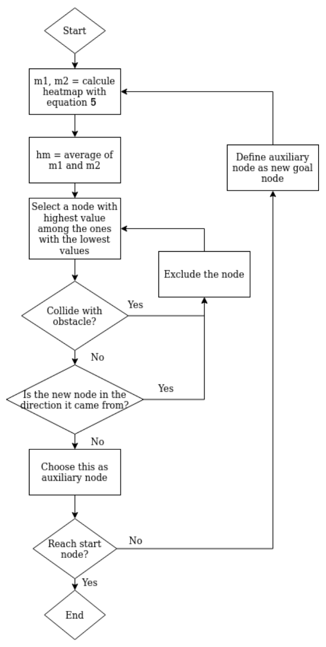


Fig. 1. Heat Map for each Auxiliaries Node. In (a) First,(b) Second and (c) Third iteration of global planner in indoor environment



Start
m1, m2 = calcule heatmap with equation 5
hm = average of m1 and m2
Select a node with highest value among the ones with the lowest values
Collide with obstacle?
Yes
No
Is the new node in the direction it came from?
Yes
No
Choose this as auxiliary node
Reach start node?
No
Yes
End
Exclude the node
Define auxiliary node as new goal node

Fig. 2. Algorithm Workflow

and global planner modules were explained in the Sections II and III, respectively.

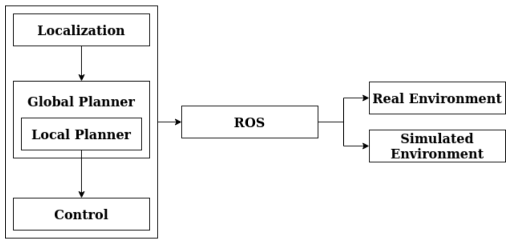


Fig. 3. UAV Structure

To flight, the VIO algorithm returns to the current position (x, y). With these nodes, the global planner calculates the best auxiliary nodes and sends them to the local planner to generate the best path to complete the mission. The control receives these nodes and moves the UAV for each one, which can be easily replicated to simulated environments and real-world experiments [40].

B. Parameters

All algorithms consider the radius 40 cm, for the F-450 in view of the real-world model has 22 cm and 18 cm, so the simulation has approximately 20 cm for the security area. The path resolution is 0.1 meters. In other words, for the scenario with 100 nodes, the real size is 10 meters. The goal threshold for RRT, Bi RRT, APF, A* and Bi A* is 20 cm. The parameters of each algorithm are shown in Table I.

TABLE I
PARAMETERS FOR EACH ALGORITHM

Algorithm	Parameters	Value
A* and Bi A*	Robot radius	40 cm
	Path resolution	0.1%
RRT and Bi RRT	Robot radius	40 cm
	Path resolution	0.1%
	Sample rate	5%
	Expansion distance	5 meters
	Max iterations	5000
APF and Bi APF	Robot radius	40 cm
	Path resolution	0.1%
	Repulsive potential gain	0.8
	Attractive potential gain	1
	Max iterations	1000

C. Scenarios

In this paper two scenarios were used, as showed in Figure 4. Each scenario has 10 meters. The first scenario is an unstructured outdoor environment, and the second scenario is an unstructured indoor environment.

Figure 5 shows 3D view of these scenarios. The yellow circle represents the start node (1, 1) in both scenarios, and the red circles are the goal node. The second scenario has two goal nodes because the UAV needs to reach them in order. The goal node for the first scenario is (9.5, 9.5), and for the second scenario is (3, 4.5) and (5, 4.7), respectively.

The indoor environments are more complicated for classic algorithms because they have narrow passages, challenging the local planners [46] [47]. A* and Bi A* are based on graphs, so it needs to build a graph throughout the environment. The graph will be created mainly on the main diagonal between the initial and objective in the outdoor scenario. However, in an indoor setting, the graph will need to cover almost the entire environment around the walls, having a higher computational cost and taking longer.

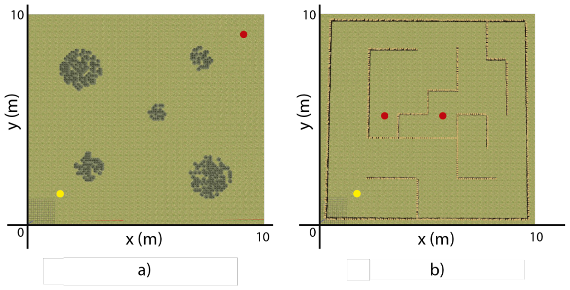


Fig. 4. Top View of Scenarios. In (a) Outdoor environment and (b) Indoor environment

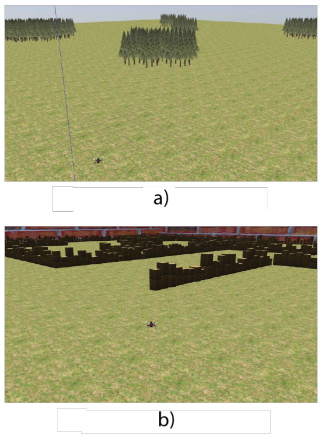


Fig. 5. 3D View of Scenarios. In (a) Outdoor environment and (b) Indoor environment

RRT and BI RRT* have the euclidean distance between the current and the goal node as a heuristic and are algorithms based on sampling (random values). Therefore, the algorithms may go to a node with a short euclidean distance but with an obstacle between the nodes. If this happens, the algorithm will reach another node that has a short distance to the target. However, this action will be computationally costly and will likely exceed the maximum number of iterations.

APF and Bi APF have a higher probability to stuck in a minimum location in an indoor than in an outdoor environment because they have poor performance in narrow passages and difficulties to solve problems in environments with symmetrical obstacles [48].

V. Simulations and Discussion

This section presents the results and discusses them. The simulations were carried out in an indoor and outdoor environment, with and without the proposed global planner. Each algorithm was executed ten times to generate statistical data and reliable results.

The Tables presented in this section follow the heading:

- *PP T* is the time of the local planner in *seconds*;
- *Distance* is the path length in *meters*;
- *CPU* is the use of processing in *MHz*;
- *Memory* is the use of memory in *MB*;
- *GP T* is the time of the global planner in *seconds*.

The data in the tables is the average of the executions and the standard deviation.

Table II presents the statistical results of simulations in an outdoor environment without the proposed global planner. In this table, we can see that all bidirectional algorithms are faster than their original versions. The memory usage for the algorithms is approximately the same, as shown by the complexity analysis in the Section II-H.

The CPU usage of all algorithms was also similar, except for Bi A*, which is almost double. The CPU usage did not affect the planning time because, for the computer on which the simulations were made, the algorithm used just 30% of processing. However, it is essential to analyze whether the UAV processor could support the processing without braking for onboard flights.

According to Table II, the path length returned by the algorithms maintains almost the same average. Thus the fact that the algorithm is bidirectional does not affect the path length of the generated trajectory. The standard deviation for the algorithms A*, Bi A*, APF, and Bi APF is 0 because they are exact techniques, meaning that, all simulations will always return the same trajectory. RRT and Bi RRT are sampling-based techniques, so in each simulation, the trajectory returned is different. However, they returned a low standard deviation, showing that the average distance is reliable.

TABLE II
Statistical Analysis in Outdoor Environment without Global Planner

Alg	Average			
	PP T	Distance	CPU	Memory
A-Star	0.49 ±0.05	23.47 ±0	459.48 ±72.52	1.33 ±0.37
Bi A-Star	0.21 ±0.007	23.77 ±0	812.84 ±211.96	1.3 ±0.38
RRT	0.44 ±0.1	24.68 ±0.97	588.84 ±191.8	1.29 ±0.36
Bi RRT	0.36 ±0.09	24.74 ±0.67	535.08 ±84	1.3 ±0.37
APF	2.5 ±0.2	26.73 ±0	558.04 ±149.24	2.08 ±0.4
Bi APF	0.43 ±0.03	21.07 ±0	509.04 ±73.08	1.35 ±0.36

Table III presents the outdoor environment results with the proposed global planner. In this scenario, all the algorithms managed to return an adequate UAV trajectory with and without the proposed global planner. In other words, all algorithms had a 100% completeness.

Analyzing only the difference among the original algorithms for the bidirectional ones, some aspects. Stood out the bidirectional algorithms returns the trajectory in a shorter time. The bidirectional algorithms' trajectory size is also smaller, except for Bi RRT, which returned a larger trajectory. RRT and Bi RRT's standard deviation was 0 because the algorithms are following auxiliary points, so their trajectory already tends to follow only one direction. Further, the algorithm is also used to decrease curves and then smooth them, leaving all trajectories equal.

The use of memory was similar for all algorithms, even with the proposed global planner. The use of processing has increased in all algorithms, except Bi A*. The increase was small for all algorithms, except for APF and Bi APF. With that, we can see that the global planner uses much more processing than these two algorithms, but that for others, the difference is not much different.

TABLE III
STATISTICAL ANALYSIS IN OUTDOOR ENVIRONMENT WITH GLOBAL PLANNER

Alg	Average				
	GP T	PP T	Distance	CPU	Memory
A-Star	0.044 ±0.004	0.42 ±0.21	33.65 ±0	792.96 ±157.08	1.28 ±0.34
Bi A-Star	0.04 ±0.008	0.16 ±0.08	25.24 ±0	574.28 ±245.84	1.26 ±0.33
RRT	0.039 ±0.005	0.36 ±0.31	33.54 ±0	490 ±50.68	1.25 ±0.31
Bi RRT	0.041 ±0.006	0.29 ±0.25	42.04 ±0	614.32 ±189.56	1.46 ±0.38
APF	0.043 ±0.003	0.21 ±0.08	33.44 ±0	999.88 ±231.56	1.33 ±0.35
Bi APF	0.043 ±0.003	0.06 ±0.006	25.16 ±0	733.6 ±290.64	1.32 ±0.33

The time of the local planner was faster in all cases with the proposed global planner. In some cases, the difference was small. As for Bi A*, the time was less than just 0.5 seconds. However, the global planner's time is 0.04 seconds, so even when using both algorithms, the final time considering the proposed global planner is less than local planner time without the proposed global planner. Moreover, there are also cases in which the time was 91.6% shorter, as in the APF.

Table IV presents the statistical results of simulations in an indoor environment without the proposed global planner. In the table, we can see that RRT, APF, and Bi APF were unable to return a trajectory without collision to the scenario. In this case, we observed the bidirectional algorithms, decreasing the time, and managing to improve the algorithm's completeness, such as Bi RRT. The original RRT was unable to find a trajectory using the number of pre-defined iterations, whereas the Bi RRT succeeded, despite having a much longer planning time. In the case of A* and Bi A*, the bidirectional algorithm time was 59% shorter.

The memory usage for the algorithms is approximately the same, as shown by the complexity analysis in the II-H section. The CPU usage of all the algorithms was also similar.

According to Table IV, the path length returned by the algorithms maintains almost the same average. Bi RRT has returned to a trajectory with greater path length and in a longer time. We can understand that the algorithm had difficulties returning the path and probably must have found a path in the last iterations.

TABLE IV
STATISTICAL ANALYSIS IN INDOOR ENVIRONMENT WITHOUT GLOBAL PLANNER

Alg	Average			
	PP T	Distance	CPU	Memory
A-Star	0.56 ±0.06	41.87 ±0	518.84 ±127.4	1.37 ±0.36
Bi A-Star	0.23 ±0.02	41.75 ±0	455.28 ±30.68	1.4 ±0.37
RRT	∞	∞	∞	∞
Bi RRT	2 ±0.86	52.53 ±0	532.56 ±48.16	1.39 ±0.36
APF	∞	∞	∞	∞
Bi APF	∞	∞	∞	∞

Table V presents the indoor environment results with the proposed global planner. In this scenario, all the algorithms managed to return an adequate UAV trajectory with the proposed global planner. Our algorithm helps the algorithms return to the trajectory in less time and improves the completeness of the algorithms.

Some aspects were noted by analyzing only the difference between the original algorithms and their bidirectional versions. The bidirectional algorithms return a smaller trajectory except for Bi RRT, which returned slightly larger than RRT.

Memory usage was similar for all algorithms, even with the use of the global planner. The use of processing has increased in all algorithms. Considering the difference in CPU usage between the four tables presented in this section, we can conclude that the proposed global planner uses approximately 200 MHz of processing.

TABLE V
STATISTICAL ANALYSIS IN INDOOR ENVIRONMENT WITH GLOBAL PLANNER

Alg	Average				
	GP T	PP T	Distance	CPU	Memory
A-Star	0.053 ±0.008	0.42 ±0.33	44.17 ±0	654.64 ±90.72	1.408 ±0.35
Bi A-Star	0.053 ±0.01	0.2 ±0.17	43.66 ±0	539 ±154.84	1.37 ±0.35
RRT	0.052 ±0.007	0.46 ±0.36	43.77 ±0	523.6 ±47.65	1.6 ±0.4
Bi RRT	0.052 ±0.008	0.42 ±0.33	46.88 ±0	676.76 ±199.92	1.69 ±0.4
APF	0.063 ±0.012	0.27 ±0.14	52.84 ±0	582.68 ±53.48	1.42 ±0.38
Bi APF	0.062 ±0.008	0.11 ±0.06	43.6 ±0	557.48 ±62.16	1.44 ±0.36

Performing the time analysis in Table III and V, we can recognize the global planner's time in all simulations is approximately 0.04 seconds, , regardless of the scenario, with a standard deviation of at most 0.0008. That is, the proposed global planner can return the auxiliary points practically in real-time.

The time of the local planner was faster in all cases when with the proposed global planner. Mainly, for Bi RRT, where

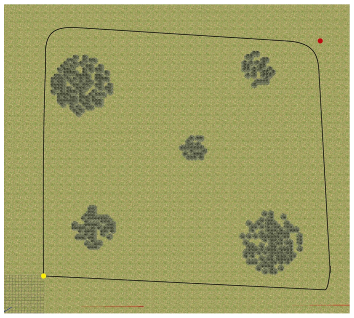


Fig. 6. Bi APF in Outdoor Environment

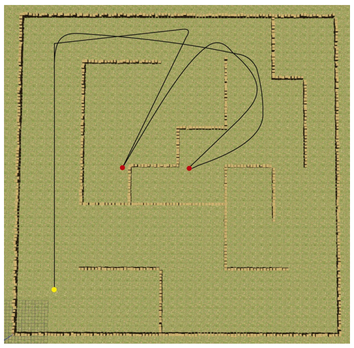


Fig. 7. Bi APF in Indoor Environment

the time decreases by 79%. Even, there were cases in which the difference was small—as for Bi A*, the time decreased by only 14%.

As in the outdoor scenario, in the indoor scenario, when adding the time of the global planner and local planner algorithms with the proposed algorithm, it is less than the time of the local planner when not with our global planner.

As shown in the II-H section, APF and Bi APF would be the most affected algorithms with our global planner as their time complexity depends mainly on the path length. As the proposed algorithm creates several small trajectories, APF and Bi APF were more optimized than the other algorithms.

The use of the proposed global planner increases the CPU usage by approximately 200 MHz. If used to perform simulations on the computer or perform flights where processing takes place on the computer, there will be no problems as it is small processing for a computer. To perform onboard flights is essential to analyze whether the microcontroller will handle an extra 200 MHz.

In indoor and outdoor scenarios, with the proposed global planner, the Bi APF algorithm returned to its trajectory in less time, with the shortest distance. Figures 6 and 7 show the trajectory returned by Bi APF with the global planner in outdoor and indoor scenario, respectively. The trajectories returned in both cases preserve the drone's torque to the maximum, increasing its Jerk. When curves are necessary, they are smoothed—being ideal trajectories for a UAV to follow.

VI. Conclusions

This paper proposed a UAV global planner to avoid minimum local and improve autonomous flights for UAVs, focusing on constructing path planning for unstructured environments. Our global planner creates several auxiliary nodes to decrease the complexity of the local planner algorithms. The simulations were performed considering the original classic algorithms and bidirectional ones.

The bidirectional algorithms were faster than the classic algorithms and improved Bi RRT's completeness in an indoor environment. With the proposed global planner, all algorithms' completeness have been improved, making all algorithms find a trajectory in the indoor environment. Moreover, it optimized the local planner algorithms' time from 14% to 91.6%.

The use of memory for original algorithms, bidirectional algorithms, utilizing the proposed global planner, or not, is the same. And, the CPU usage increased by approximately 200 MHz. The time to generate the auxiliary nodes is approximately 0.04 seconds, so even with the local and global planner together, the final path planner time will be less than without the proposed global planner.

This method can be easily implemented in any ROS-based system, as was done with the control system, to do a complete mission. For example, when adding the computer vision and exploration path planning modules to carry out monitoring missions. We intend to develop a 3D global planner for future works.

Acknowledgment

The authors acknowledge Brazilian research agencies CNPq and CAPES for the financial support of this research.

References

[1] M. Blösch, S. Weiss, D. Scaramuzza, and R. Siegwart, "Vision based mav navigation in unknown and unstructured environments," in *2010 IEEE International Conference on Robotics and Automation*. IEEE, 2010, pp. 21–28.
[2] K. C. Vivaldini, T. H. Martinelli, V. C. Guizilini, J. R. Souza, M. D. Oliveira, F. T. Ramos, and D. F. Wolf, "Uav route planning for active disease classification," *Autonomous Robots*, vol. 43, no. 5, 2018.
[3] H. Li, A. V. Savkin, and B. Vucetic, "Autonomous area exploration and mapping in underground mine environments by unmanned aerial vehicles," *Robotica*, vol. 38, no. 3, pp. 442–456, 2020.
[4] S. Schopferer and S. Benders, "Minimum-risk path planning for long-range and low-altitude flights of autonomous unmanned aircraft," in *AIAA Scitech 2020 Forum*, 2020.
[5] X. He, J. R. Bourne, J. A. Steiner, C. Mortensen, K. C. Hoffman, C. J. Dudley, B. Rogers, D. M. Cropek, and K. K. Leang, "Autonomous chemical-sensing aerial robot for urban/suburban environmental monitoring," *IEEE Systems Journal*, vol. 13, no. 3, pp. 3524–3535, 2019.
[6] P. Corke, *Robotics, vision and control: fundamental algorithms in MATLAB® second, completely revised*. Springer, 2017, vol. 118.
[7] V. San Juan, M. Santos, and J. M. Andújar, "Intelligent uav map generation and discrete path planning for search and rescue operations," *Complexity*, vol. 2018, 2018.
[8] S. Aggarwal and N. Kumar, "Path planning techniques for unmanned aerial vehicles: A review, solutions, and challenges," *Computer Communications*, 2019.
[9] G. Francis, L. Ott, R. Marchant, and F. Ramos, "Occupancy map building through bayesian exploration," *The International Journal of Robotics Research*, vol. 38, no. 7, pp. 769–792, 2019.
[10] A. Majeed and S. Lee, "A fast global flight path planning algorithm based on space circumscription and sparse visibility graph for unmanned aerial vehicle," *Electronics*, vol. 7, p. 375, 12 2018.

[11] P. O. N. Saian *et al.*, "Optimized a-star algorithm in hexagon-based environment using parallel bidirectional search," in *2016 8th International Conference on Information Technology and Electrical Engineering (ICITEE)*. IEEE, 2016, pp. 1–5.

[12] R. Kumar, A. Mandalika, S. Choudhury, and S. S. Srinivasa, "Lego: Leveraging experience in roadmap generation for sampling-based planning," *arXiv preprint arXiv:1907.09574*, 2019.

[13] B. Patle, A. Pandey, D. Parhi, A. Jagadeesh, *et al.*, "A review: On path planning strategies for navigation of mobile robot," *Defence Technology*, 2019.

[14] P. C. Chen, "Learning to improve path planning performance," in *Artificial Intelligence And Automation*. World Scientific, 1998, pp. 466–502.

[15] P. Raja and S. Pugazhenthi, "Optimal path planning of mobile robots: A review," *International journal of physical sciences*, vol. 7, no. 9, pp. 1314–1320, 2012.

[16] G. Nannicini, D. Delling, L. Liberti, and D. Schultes, "Bidirectional a search for time-dependent fast paths," vol. 5038, 05 2008, pp. 334–346.

[17] D. McIntyre, W. Naeem, and X. Xu, "Cooperative obstacle avoidance using bidirectional artificial potential fields," in *2016 UKACC 11th International Conference on Control (CONTROL)*. IEEE, 2016, pp. 1–6.

[18] Y. Dabachine, B. Bouikhalene, and A. Balouki, "Bidirectional search algorithm for airport ground movement," in *2018 International Arab Conference on Information Technology (ACIT)*. IEEE, 2018, pp. 1–9.

[19] P. Gao, Z. Liu, Z. Wu, and D. Wang, "A global path planning algorithm for robots using reinforcement learning," in *2019 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. IEEE, 2019, pp. 1693–1698.

[20] E. J. Gómez, F. M. Martínez Santa, and F. H. M. Sarmiento, "A comparative study of geometric path planning methods for a mobile robot: Potential field and voronoi diagrams," in *2013 II International Congress of Engineering Mechatronics and Automation (CIIMA)*, 2013, pp. 1–6.

[21] J. Carsten, D. Ferguson, and A. Stentz, "3d field d: Improved path planning and replanning in three dimensions," in *2006 IEEE/RSJ international conference on intelligent robots and systems*. IEEE, 2006, pp. 3381–3386.

[22] Y. Liu, Q. Wang, H. Hu, and Y. He, "A novel real-time moving target tracking and path planning system for a quadrotor uav in unknown unstructured outdoor scenes," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 49, no. 11, pp. 2362–2372, 2018.

[23] X. Hu, M. Wang, C. Qian, C. Huang, Y. Xia, and M. Song, "Lidar-based slam and autonomous navigation for forestry quadrotors," in *2018 IEEE CSAA Guidance, Navigation and Control Conference (CGNCC)*. IEEE, 2018, pp. 1–6.

[24] R. Aggarwal, A. Soderlund, M. Kumar, and D. Grymin, "Risk aware suas path planning in an unstructured wildfire environment," in *2020 American Control Conference (ACC)*. IEEE, 2020, pp. 1767–1772.

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

[26] F. DuchoĖ, A. Babineca, M. Kajana, P. BeĖoa, M. Floreka, T. Ficoa, and L. Jurišicaa, "Path planning with modified a star algorithm for a mobile robot," *Procedia Engineering*, vol. 96, pp. 59–69, 2014.

[27] M. Lavalle and S. J. J. Kuffner", ""rapidly-exploring random trees: Progress and prospects"," in *"Proc. Workshop on the Algorithmic Foundations of Robotics"*, "San Francisco", "2000".

[28] I. Noreen, A. Khan, and Z. Habib, "A comparison of rrt, rrt* and rrt*-smart path planning algorithms," *International Journal of Computer Science and Network Security (IJCSNS)*, vol. 16, no. 10, p. 20, 2016.

[29] S. Klemm, J. Oberländer, A. Hermann, A. Roennau, T. Schamm, J. M. Zollner, and R. Dillmann, "Rrt-connect: Faster, asymptotically optimal motion planning," in *2015 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. IEEE, 2015, pp. 1670–1677.

[30] S. LaValle, "Planning algorithms. cambridge, uk: Cambridge university press," 2006.

[31] J. J. Kuffner and S. M. LaValle, "Rrt-connect: An efficient approach to single-query path planning," in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, vol. 2. IEEE, 2000, pp. 995–1001.

[32] Y. Xue, X. Zhang, S. Jia, Y. Sun, and C. Diao, "Hybrid bidirectional

rapidly-exploring random trees algorithm with heuristic target graviton," in *2017 Chinese Automation Congress (CAC)*. IEEE, 2017, pp. 4357–4361.

[33] E. Koslosky *et al.*, "Geração de trajetória em espiral e navegação com desvio de obstáculos para veículos aéreos não-tripulados," Master's thesis, Universidade Tecnológica Federal do Paraná, 2018.

[34] U. Goel, S. Varshney, A. Jain, S. Maheshwari, and A. Shukla, "Three dimensional path planning for uavs in dynamic environment using glow-worm swarm optimization," *Procedia computer science*, vol. 133, pp. 230–239, 2018.

[35] I. Noreen, A. Khan, K. Asghar, and Z. Habib, "A path-planning performance comparison of rrt*-ab with mea* in a 2-dimensional environment," *Symmetry*, vol. 11, no. 7, p. 945, 2019.

[36] P. Pandey, A. Shukla, and R. Tiwari, "Three-dimensional path planning for unmanned aerial vehicles using glowworm swarm optimization algorithm," *International Journal of System Assurance Engineering and Management*, vol. 9, no. 4, pp. 836–852, 2018.

[37] F. Stoican, I. Prodan, D. Popescu, and L. Ichim, "Constrained trajectory generation for uav systems using a b-spline parametrization," in *2017 25th Mediterranean Conference on Control and Automation (MED)*. IEEE, 2017, pp. 613–618.

[38] S. McKinley and M. Levine, "Cubic spline interpolation," *College of the Redwoods*, vol. 45, no. 1, pp. 1049–1060, 1998.

[39] L. Rocha, S. C. Ferreira, A. Carvalho, and J. Araújo, "Performance evaluation of the coverage and capacity of an ieee 802.11ac wireless network," *Brazilian Journal of Development*, vol. 6, pp. 31 160–31 172, 05 2020.

[40] T. Baca, M. Petrlik, M. Vrba, V. Spurny, R. Penicka, D. Hert, and M. Saska, "The mrs uav system: Pushing the frontiers of reproducible research, real-world deployment, and education with autonomous unmanned aerial vehicles," 2021.

[41] A. Koubâa *et al.*, *Robot Operating System (ROS)*. Springer, 2017, vol. 1.

[42] DJI, *FlameWheel 450*, DJI, 05 2015, user Manual. [Online]. Available: http://dl.djicdn.com/downloads/flamewheel/en/F450_User_Manual_v2.2_en.pdf

[43] D. Team, *Pixhawk Flight Controller*, Dronecode, 03 2020, user Manual. [Online]. Available: https://docs.px4.io/v1.9.0/en/flight_controller/pixhawk.html

[44] Y. He, J. Zhao, Y. Guo, W. He, and K. Yuan, "Pl-vio: Tightly-coupled monocular visual–inertial odometry using point and line features," *Sensors*, vol. 18, no. 4, p. 1159, 2018.

[45] B. Mehta and Y. Reddy, "Advanced process control systems," pp. 547–557, 2015.

[46] D. Hsu, T. Jiang, J. Reif, and Z. Sun, "The bridge test for sampling narrow passages with probabilistic roadmap planners," in *2003 IEEE international conference on robotics and automation (cat. no. 03CH37422)*, vol. 3. IEEE, 2003, pp. 4420–4426.

[47] W. Wang, X. Xu, Y. Li, J. Song, and H. He, "Triple rrts: an effective method for path planning in narrow passages," *Advanced Robotics*, vol. 24, no. 7, pp. 943–962, 2010.

[48] Y. Koren and J. Borenstein, "Potential field methods and their inherent limitations for mobile robot navigation," in *Proceedings. 1991 IEEE International Conference on Robotics and Automation*, 1991, pp. 1398–1404 vol.2.