THE PUBLISHING HOUSE
OF THE ROMANIAN ACADEMY

PROCEEDINGS OF THE ROMANIAN ACADEMY, Series A,
Volume 25, Number 1/2024, pp. 45–53
DOI: 10.59277/PRA-SER.A.25.1.07

A COMPARATIVE ANALYSIS OF PATH PLANNING ALGORITHMS FOR UNMANNED AERIAL VEHICLES

Mirela-Mădălina BIVOLARU[1,2]

[1] "Politehnica" University of Bucharest, Faculty of Aerospace Engineering
[2] Romanian Space Agency
E-mail: mirela.bivolaru@rosa.ro

Abstract. In terms of trajectory planning, the main objective is to reach a certain location in the shortest time possible, avoiding obstacles along the way. This paper proposes two path planning methods, in an environment with presence of obstacles, Probabilistic RoadMap and a fusion between Voronoi diagram and Dijkstra's Algorithm. Path planning for both algorithms is developed in the XY coordinate plane, therefore only horizontal movement is considered. This means that the UAV must fly around obstacles and cannot fly over or under them. The use of two-dimensional space in favour of three-dimensional space reduces the implementation time of the algorithm for solving the problem. Finally, an analysis of the length and the computational time of the trajectories generated by the proposed algorithms is made.

Keywords: trajectory planning, UAV, Probabilistic RoadMap, Voronoi, Dijkstra's Algorithm.

1. INTRODUCTION

Unmanned aerial vehicles (UAVs) are aircrafts that do not have a human pilot or passengers on board and are able to operate a wide variety of tasks, especially tasks that can be dangerous or difficult for humans to perform. Furthermore, UAVs can be operated for missions where their use provides a cost reduction.

Recently, UAVs have gained a lot of attention, their applications knowing a significant diversification. Their great potential is notable in several fields, such as traffic monitoring [1], cargo delivery [2], fire management [3], agricultural monitoring [4], border surveillance [5], reconnaissance [6], environmental and meteorological monitoring [7], search and rescue missions [8] etc.

Although the number of UAVs significantly increased, trajectory generation that guarantees a collision free path in complex environments is still challenging. Besides obstacle avoidance, real-time planning capability and optimal fuel consumptions are frequently encountered requirements to design a UAV.

Many different methods have been used for robot trajectory planning. Reference [9] uses Mixed Integer Linear Programming (MILP) for trajectory planning of UAVs and vehicles with turn and minimum velocity constraints. Bortoff adjusted Voronoi's trajectory planning algorithm to avoid radio detection of UAVs [10]. This method is useful for two-dimensional space, where obstacles are represented by points in space, being difficult to implement in three-dimensional space. Autonomous path planning algorithm based on a tangent intersection and target guidance strategy (APPATT) proposed in [11] is an efficient technique for path planning in presence of obstacles. In [12] a trajectory planning method based on the artificial potential field (APF) in windy environments is proposed. Genetic Algorithm (GA) is an efficient method to develop optimal trajectories, [13–15] used this technique to design an autonomous path planning. Rapidly-exploring Random Trees (RRTs), another effective method for motion planning in presence of obstacles, was introduced for the first time in [16]. Results of the efficiency of this method were presented in [17–19]. A* algorithm is another method that is frequently encountered in literature for path planning [15, 19]. A complex method that gives near-optimal solutions is presented in [20]. This method aims to collect information from the environment while avoiding banned areas, over a planned horizon.

2. METHODOLOGY

In this section, two algorithms for trajectory planning in the two-dimensional space, probabilistic roadmap, respectively Dijkstra's algorithm, are presented. These are two simple, efficient and fast methods of creating an optimal trajectory through known obstacles.

2.1. Probabilistic roadmap (PRM)

Probabilistic roadmap is a trajectory planning method in static environments, created by Lydia Kavraki, Petr Svestka, Jean-Claude Latombe and Mark H. Overmars [16]. This consists in creating a roadmap that contains randomly generated nodes in spaces where there are no obstacles, and the actual trajectory is formed by drawing lines through the nodes, so that the aerial platform reaches from the starting point to the end point.

This method has two stages: one for learning and the other one for query.

In the *learning stage*, a roadmap is built by repeated and random generation of some nodes in the obstacle-free areas. Nodes are connected to each other, forming segments. The roadmap is a *graph* $G=(N, E)$, where N represents the *nodes*, and E represents the *edges* that connect two nodes, forming local roads. These local routes are processed very quickly by a motion planner, called *local motion planner*. Local roads are not stored in the map because their reconstruction is simple, in a short time, thus saving considerable memory space.

The *query stage* consists of searching for the optimal trajectory in the roadmap. Denoting S the start point and F the end point, the algorithm searches for two nodes S' and F' to connect in N with S and F respectively. If this operation is successful, E is searched for a sequence of edges connecting nodes S' and F'. This process is repeated until a feasible trajectory results.

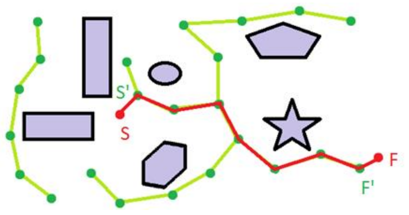


Fig. 1 – UAV Roadmap.

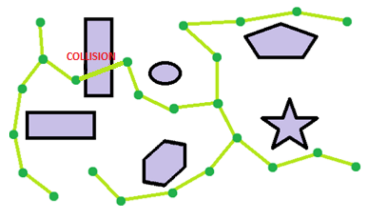


Fig. 2 – Learning stage-construction.

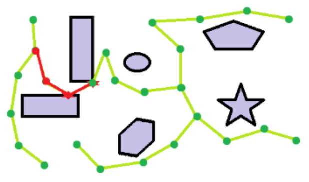


Fig. 3 – Learning stage-expansion.

2.1.1. Learning stage

The learning stage consists of two successive steps: construction and expand. The objective is to obtain a graph with connections between points, uniformly distributed; even in hard-to-reach places in free space, C_f, there must be at least a few nodes. In the second step, the connections between the points are improved. The nodes in G which, according to the heuristic evaluations, are in the difficult areas of the space C, are selected and nodes in their neighbourhood are generated.

2.1.1.1. Construction

Initially, the graph $G=(N,E)$ is empty. Nodes are randomly generated and added to N. For each new node c, N is searched for nodes to connect to, using a local planner. A local planner, which can be used by all holonomic robots, connects any two configurations with a straight line and checks this segment for collision. A collision check method is proposed in [21]. If a feasible route is constructed between c and a node n, the segment (c,n) is added to E. It is not stored because storing it takes a lot of time and memory space.

The selection of the nodes with which to connect is made as follows: first, a domain N_c – “candidate neighbours” is selected, consisting of nodes that are at a certain distance from c. Then the nodes in N_c are sorted in ascending order of the distance from c. Node c is connected with each node in N_c, in the order previously obtained. Only the pairs of nodes whose relative distance is less than a *maxdist* constant are kept. It is defined:

$$N_c = \{c' \in N \mid D(c,n) \leq \text{maxdist}\}. \tag{1}$$

When the local planner successfully connects two nodes, the connected components of G are updated. Therefore, there is no need to search the graph to see if a node chosen from N_c is connected to c or not.

The distance function D is used to construct and arrange the domain N_c; is defined so that, for any pair (c,n), D(c,n) calculates the chances that the local planner fails to generate a feasible path between these points. The function D(c,n) can be defined as:

$$D(c,n) = \max_{x \in \mathrm{UAV}} \|x(n) - x(c)\| \tag{2}$$

where x represents a point on the UAV, $x(c)$ is the position of x when the UAV is at c, and $||x(n)–x(c)||$ represents the Euclidean distance between the points $x(n)$ and $x(c)$.

2.1.1.2. Expansion

If the number of nodes generated in the construction stage is large enough, the N domain creates a relatively even coverage of free space. This step aims to improve the connections created in the construction stage.

The main idea is to select several nodes from N whose probability of being in hard-to-reach regions is high and expand them. By expanding a node c is meant to select a new free node in the neighbourhood of c, add it to the domain G and try to connect it with other nodes in N in the same way used in the construction step. Therefore, the expansion step increases the density of nodes on the map in the difficult regions of C_f. Since the gaps between the components of the graph G are usually located in these areas, the connectedness possibilities of G increase.

At the end of the construction step, for each node c, the failure rate is defined:

$$r_f(c) = \frac{f(c)}{n(c) + 1} \tag{3}$$

where $n(c)$ represents the number of times c has connected with another node, and $f(c)$ represents the number of failures.

At the beginning of the expansion stage, for each node c in N, the weight $w(c)$ is generated, proportional to the failure rate, but scaled so that all the added weights give the value one. A more extensive discussion of expansion techniques can be found in [22].

2.1.2. Query stage

During this stage, routes are searched between the start and end points using the roadmap constructed in the learning stage. The free space C_f is assumed to be connected and the map consists of a single G component.

To connect S to G, consider nodes in G at increasing distance from S and try to connect it to each of the nodes via a local planner until one of the connections is made. Nodes that are at a greater distance than *maxdist* are ignored because the local scheduler has a low chance of connecting them with the start point. If the trial fails, one or more stochastic "walks" are performed, like the one in the learning step. The same procedure applies to connect F with G.

2.2. Voronoi diagrams and Dijkstra's algorithm

This section will present a method to obtain a roadmap using Voronoi diagrams and Dijkstra's algorithm. Voronoi diagrams will be used to connect the area where the UAV operates, and Dijkstra's algorithm creates the shortest path between any two points on the road map.

2.2.1. Voronoi diagrams

Given several points in space, the Voronoi diagram divides the space into several regions, using the nearest neighbour rule: each point is associated with the region closest to it.

Let S be a set of n points in the plane. For two distinct points, $p,q \in S$, the dominance of p over q is defined as the subset of planes that are at least as close to p as they are to q:

$$\mathrm{dom}(p,q) = \{x \in R^2 | \delta(x,p) \le \delta(x,q)\} \tag{4}$$

where δ represents the Euclidean distance function.

dom(*p*,*q*) is a closed half-plane determined by the perpendicular bisector of the segment (*p*,*q*). This bisector separates all points in the plane close to *p* from those in the plane close to *q*, thus determining the separation of *p* from *q*. The region of a set $p \in S$, reg(*p*), represents the portion of the plane dominated by *p*:

$$\operatorname{reg}(p) = \bigcap_{q \in S-\{p\}} \operatorname{dom}(p, q) \tag{5}$$

Because the regions are formed by intersecting $n-1$ half-planes, they have convex polygonal shapes. The boundary of a region can have at most $n-1$ edges and vertices. Every point on the edge is equidistant from two sets, and every vertex is equidistant from at least three sets. As a consequence, the regions are connected to each other, having common edges and vertices. This division of space is called a Voronoi diagram, *V*(*S*). *V*(*S*) contains exactly *n* regions. There are no vertices if all points in *S* are collinear; this implies the existence of regions consisting of only one edge. A more detailed presentation of how to generate Voronoi diagrams can be found in [23].

2.2.2. Dijkstra's algorithm

Dijkstra's algorithm solves the problem of finding the shortest path in a graph $G=(V,M)$, where *V* represents vertices and *E* represents edges. Edges must have positive values.

Given a point (node) in the graph, this algorithm finds the shortest (minimum cost) path between it and any other point. Dijkstra's algorithm can also be used to find the cost of the shortest path between two points.

Let the starting point be called *the initial node* and the distance from the initial node to Y *the distance of node Y*. Dijkstra's algorithm assigns initial values to the distances which it improves step by step. For a better understanding of this technique, the following algorithm is proposed:

1. Each node is assigned a preliminary distance value: zero for the initial node, and infinity for the other nodes (Fig. 4a).

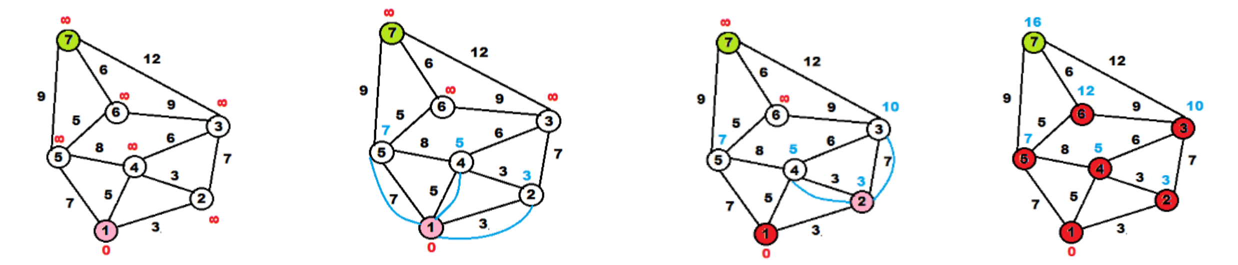
2. Mark all unvisited nodes. Sets the initial node as *in use*. A set of unvisited nodes called *the unvisited set* is created, which contains all the nodes.

3. For the current node (*in use*), a preliminary distance to all unvisited nodes in its vicinity is calculated (Fig. 4b). For example, if node A is assigned distance 6, and the edge connecting A to a neighbouring node B has length 2, then the distance to B, passing through A, is 6+2 = 8. If the distance is less than B's previous preliminary distance, then the smaller distance will be kept. Although a neighbouring node has been examined, it is not marked as visited and remains in the unvisited crowd.

4. After calculating preliminary distances to all neighbouring nodes, mark the current node as *visited* and remove it from the unvisited set (Fig. 4c). A visited node will never be examined again.

5. If the destination node has been marked as visited (when planning a route between two specific nodes) or if the smallest preliminary distance between nodes in the unvisited set is infinite (when there is no connection between the initial node and the remaining unvisited nodes), then the algorithm stops.

6. Unvisited nodes which are marked with the smallest preliminary distance are selected and set as the current node, then steps 3–6 are repeated until the shortest path to the destination is found (Fig. 4d).



a) assigning a preliminary distance value b) calculating a preliminary distance c) marking the current node as visited d) finding the shortest path

Fig. 4 – Dijkstra's algorithm.

3. SIMULATIONS

In this section are illustrated the results of the simulations obtained for trajectory planning, using the two algorithms previously presented. The implementation of the algorithms is made using the same obstacle map, followed by a comparative analysis of them in section 3.3.

3.1. Probabilistic roadmap

To create the map, 3 two-dimensional obstacles were generated (Fig. 5), positioned so that they almost uniformly occupy the area in which the UAV operates. Figure 6 illustrates the direct path between the start point and the destination point, and it is noted that obstacles are present along its length, so a linear path to the destination point is not an option in this case.

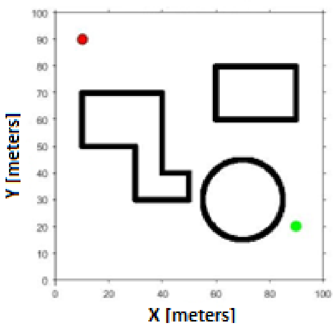


Fig. 5 – Obstacles map.

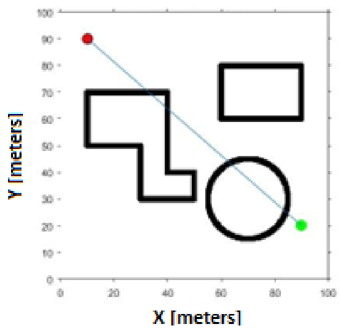


Fig. 6 – Linear trajectory.

To avoid collision with the peripheral parts of the obstacles (corners, edges, etc.), the contour of the surface of the obstacles was increased by the size of the robot's wingspan.

Randomly, 1500 nodes were generated, these being connected to each other by edges whose length is less than 1 meter. The number of nodes was chosen by making several successive iterations, thus 1500 nodes proved to be optimal in relation to their uniform distribution on the map. The length of the edges was chosen in such a way that the trajectory does not present areas with visibly too large deviations. Figure 7 shows that the distribution of nodes and edges is relatively uniform over the entire surface of the map.

The resulting trajectory using the algorithm presented is illustrated in Fig. 8. It can be observed that the path from the start point to the destination point does not intersect with any obstacle, and the PRM algorithm creates a good length. Along the route, unnecessary changes in the head angle are observed, therefore smoothing the trajectory is necessary.

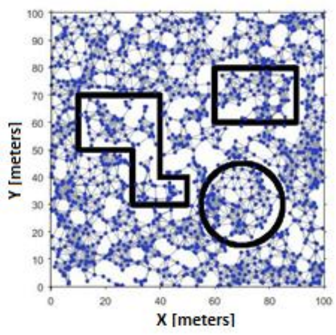


Fig. 7 – PRM.

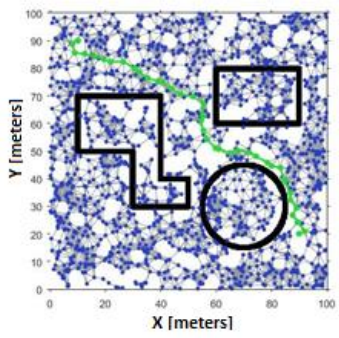


Fig. 8 – PRM-generated trajectory.

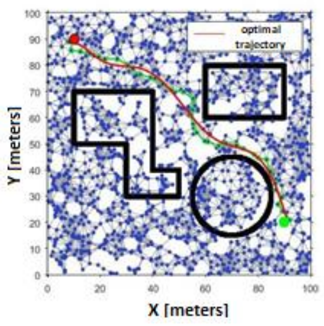


Fig. 9 – PRM-final trajectory

The final trajectory is shown in Fig. 9. This was obtained by modelling a six-point line on the trajectory generated by the PRM method.

3.2. Path planning using Voronoi diagrams and Dijkstra's algorithm

To create the map, the same obstacle shapes and positions as those used in the PRM algorithm were used. Also, the start and the finish points are the same as previously used. The length, width, and diameter of the obstacles were increased by the size of the UAV's wingspan, in order to avoid collision with them, in case of generating some points of the trajectory on the outer surfaces of the obstacles.

In the first phase, 1000 coordinate points (x,y) are randomly generated. Around these points, the Voronoi diagram is formed, as in Fig. 10. The surface is divided into small polygons, therefore 1000 points being sufficient to create the road map.

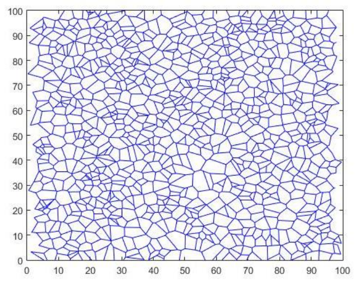
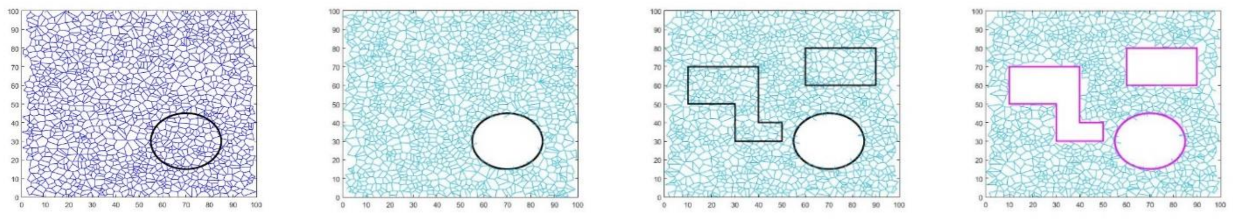


Fig. 10 – Voronoi diagram.

Next, the first circular obstacle was generated, as in Fig. 11a. Voronoi diagram cells are present inside the circle, and if a trajectory is generated between the start point and the destination point, it will not consider obstacle avoidance, the path passing through it. Therefore, it is necessary to remove the cells inside the obstacles. After removing the Voronoi cells inside the circle, the roadmap looks like Fig. 11b.

In the following, the polygonal obstacles are also introduced and, analogously, the content inside them is removed. The results are shown in Figs. 11c and 11d.



a) creating the circular obstacle b) elimination of cells inside the circular obstacle c) creation of polygonal obstacles d) final obstacle map

Fig. 11 – Generation of obstacles in the Voronoi diagram.

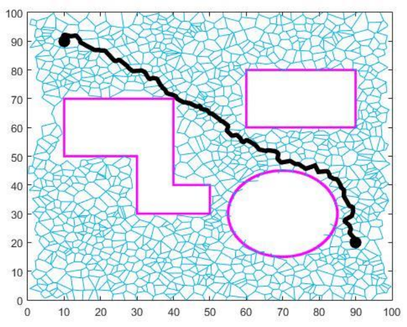


Fig. 12 – Dijkstra's algorithm-generated trajectory.

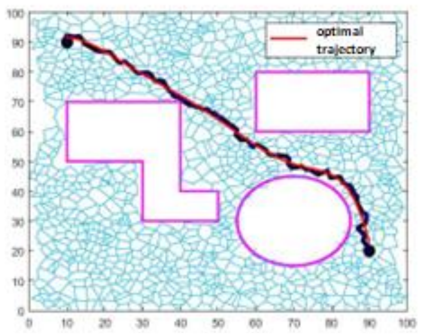


Fig. 13 – Dijkstra's algorithm-final trajectory.

The final trajectory was obtained using Dijkstra's algorithm, detailed in the previous chapter. Figure 12 illustrates that road obtained has, as in the case of the PRM algorithm, unnecessary changes in the head angle. Therefore, this trajectory must also be optimized. As with the first algorithm used, the curvature of a line was used, but in seven points, this time. The final path using Dijkstra's algorithm, followed by its optimization, is shown in Fig. 13.

3.3. Comparative analysis of obstacle avoidance algorithms

In this section, two trajectory planning methods used by UAVs, Probabilistic RoadMap and an algorithm using Voronoi diagrams and Dijkstra's algorithm, were presented. Both methods are based on stochastic map exploration. PRM creates a map containing randomly generated nodes in unobstructed areas and edges connecting the points to find a free path. Dijkstra's algorithm explores the edge cost of the Voronoi diagram, thus finding the shortest path between two points.

The two trajectory planning procedures were compared using the same map, and the positions of the start [x_0 y_0] and destination points [x_f y_f] were changed five times. Each algorithm produced ten trajectories.

Table 1

Comparative analysis

Simulation number	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
[x_0 y_0] [m]	[10 90]	[10 90]	[10 10]	[10 10]	[80 5]	[80 5]	[70 10]	[70 10]	[5 60]	[5 60]
[x_f y_f] [m]	[90 20]	[90 20]	[50 60]	[50 60]	[60 90]	[60 90]	[10 80]	[10 80]	[95 70]	[95 70]

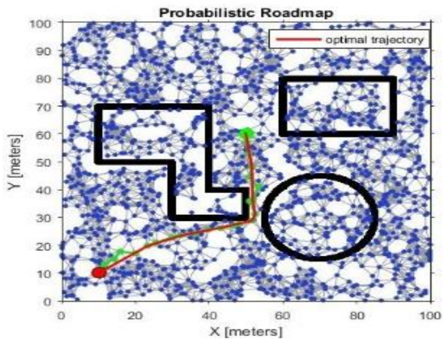


Fig. 14 – PRM-trajectory for Simulation number 3.

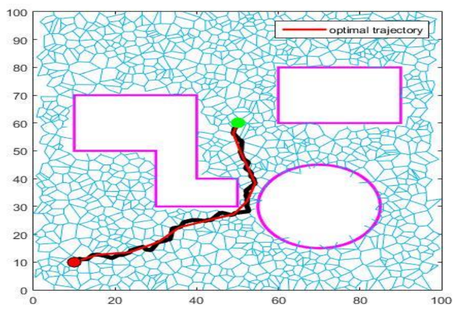


Fig. 15 – Djekstra's algorithm-trajectory for Simulation number 3.

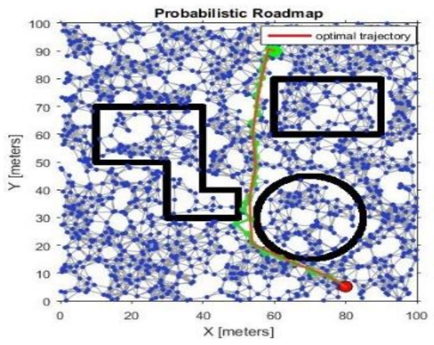


Fig. 16 – PRM-trajectory for Simulation number 5.

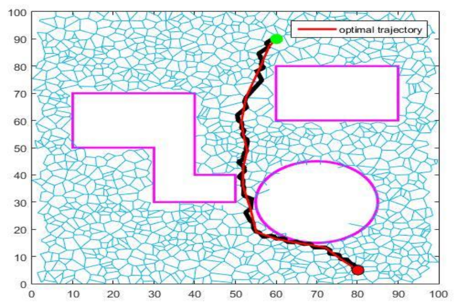


Fig. 17 – Djekstra's algorithm-trajectory for Simulation number 5.

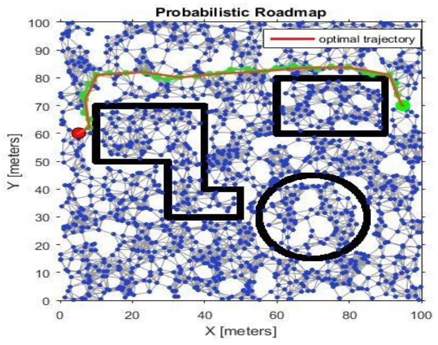


Fig. 18 – PRM-trajectory for Simulation number 7.

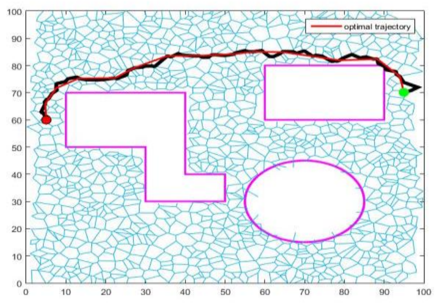


Fig. 19 – Djekstra's algorithm-trajectory for Simulation number 7.

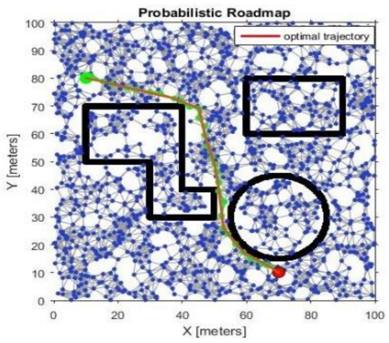


Fig. 20 – PRM-trajectory for Simulation number 9.

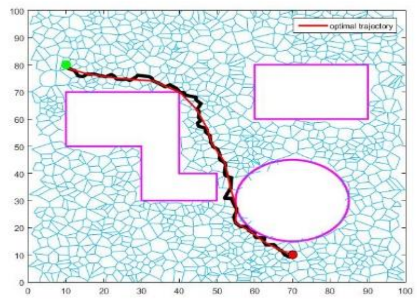


Fig. 21 – Djekstra’s algorithm-trajectory for Simulation number 9.

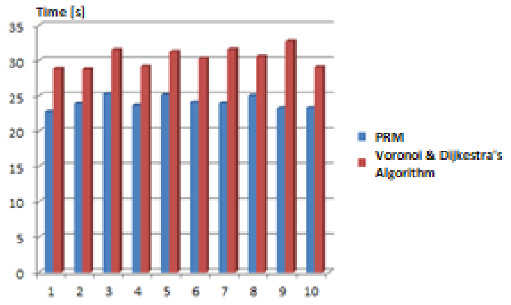


Fig. 22 – Computational time necessary to implement the algorithms.

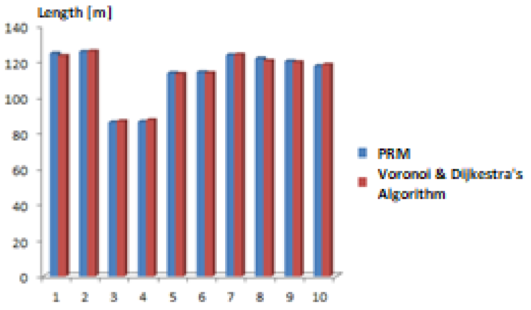


Figure 23 – Path length.

Figure 22 shows that the average roadmap generation time of the first algorithm is 24 seconds, about 6 seconds less than the average time of the second algorithm. Although the number of points generated for creating the road map is 50% higher for the PRM algorithm, it is observed to be faster. In terms of trajectory length (Fig. 23), there were no large differences, resulting in the cost-effectiveness of the two algorithms.

4. CONCLUSIONS

In this paper, the simulations of the trajectory planning algorithms, Probabilistic Roadmap and Dijkstra’s algorithm, were presented, followed by a comparative analysis of them. An obstacle-free path from the start point to the destination point, in the same environment, is generated with both methods. The two algorithms are equally efficient in the two-dimensional environment, the computational time being quite small, and the differences between the lengths of the trajectories are insignificant. A study on the effectiveness of these methods in the 3D environment can be an interesting subject for forthcoming research.

REFERENCES

[1] Chang G, Demin L, Guanglin Z, Menglin Z. Real-time path planning in urban area via VANET-assisted traffic information sharing. IEEE Transactions on Vehicular Technology 2018;67(7):5635–5649.
[2] Suttinee S, Dusit N, Puay-Siew T, Ping W. Joint ground and aerial package delivery services: A stochastic optimization approach. IEEE Transactions on Intelligent Transportation Systems 2019;20(6):2241–2254.
[3] Restas A. Forest fire management supporting by UAV based air reconnaissance results of Szendro Fire Department, Hungary. In: 2006 First International Symposium on Environment Identities and Mediterranean Area. 2006, pp. 73–77.
[4] Chuan L, Guangjie H, Tiantian X, Lei S. An adaptive path planning scheme towards chargeable UAV-IWSNs to perform sustainable smart agricultural monitoring. In: 2020 IEEE 18th International Conference on Industrial Informatics (INDIN). 2020, pp. 535–540.
[5] Sanket D, Archana K, Hirkani T, Omkar A, Amar B. Border surveillance monitoring application. In: 2019 5th International Conference on Computing, Communication, Control and Automation (ICCUBEA). 2019, pp. 1–6.
[6] Rui L, Hongzhong M. Research on UAV swarm cooperative reconnaissance and combat technology. In: 2020 3rd International Conference on Unmanned Systems (ICUS). 2020, pp. 996–999.

[7] Prisacariu V, Cheval S. Using UAV-LTA for environmental monitoring. In: “Air and Water – Components of the Environment” Conference Proceedings. Cluj-Napoca, Romania; 2019, pp. 39–52.
[8] Silvagni M, Tonoli A, Zenerino E, Chiaberge M. Multipurpose UAV for search and rescue operations in mountain avalanche events. Geomatics, Natural Hazards and Risk 2017;8(1):18–33.
[9] Schouwenaars T, How J, Feron E. Receding horizon path planning with implicit safety guarantees. In: Proceedings of the 2004 American Control Conference. 2004, pp. 5576–5581.
[10] Bortoff S. Path planning for UAVs. In: Proceedings of the 2000 American Control Conference ACC. 2000, pp. 364–368.
[11] Liu H, Li X, Fan M, Wu G, Pedrycz W, Suganthan PN. An autonomous path planning method for unmanned aerial vehicle based on a tangent intersection and target guidance strategy. IEEE Transactions on Intelligent Transportation Systems 2020;23(4):3061–3073.
[12] Jayaweera H, Hanoun S. Path planning of Unmanned Aerial Vehicles (UAVs) in windy envirnments. Drones 2022;6(5):101.
[13] Xin J, Xin J, Jiabao Z, Cui Y, Sheng J. An improved genetic algorithm for path-planning of unmanned surface vehicle. Sensors 2019;19(11):2640.
[14] Tuncer A, Yildirim A. Dynamic path planning of mobile robots with improved genetic algorithm. Computers & Electrical Engineering 2012;38:1564–1572.
[15] Jeauneau V, Jeauneau L, Kotenkoff A. Path planner methods for UAVs in real environment. IFAC-PapersOnLine 2018;51:292–297.
[16] Kavraki L, Svestka P, Latombe JC, Overmars MH. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. IEEE Transactions on Robotics and Automation, 1996;12(4):566–580.
[17] Meng J, Pawar V, Kay S, Li A. UAV path planning system based on 3D informed RRT* for dynamic obstacle avoidance. In: IEEE International Conference on Robotics and Biomimetics. 2018, pp. 1653–1658.
[18] Ramana MV, Varma SA, Kothari M. Motion planning for a fixed-wing UAV in urban environments. IFAC-PapersOnLine 2016;49(1):419–424.
[19] Peng T, Chen Z, Zhou Y. A RRT path planning algorithm based on A* for UAV. In: 4th International Conference on Informatics Engineering & Information Science (ICIEIS2021); Proceedings of the SPIE, vol. 12161. 2022, p. 1216106.
[20] Ergezer H, Leblebicioğlu K. Online path planning for unmanned aerial vehicles to maximize instantaneous information. International Journal of Advanced Robotics 2021;18(3):17298814211010379.
[21] Barraquand J, Latombe, JC. Robot motion planning: A distributed representation approach. The International Journal of Robotics Research 1991;10(6):628–649.
[22] Kavraki L. Random networks in configuration space for fast path planning [Ph.D. thesis], Stanford University; 1994.
[23] Aurenhammer F. Voronoi diagrams – A survey of a fundamental geometric data. ACM Computing Surveys 1991;23: 345–405.

Received October 26, 2022