A Dijkstra Algorithm for Fixed-Wing UAV Motion Planning Based on Terrain Elevation

Felipe Leonardo Lôbo Medeiros and José Demisio Simões da Silva

Instituto de Estudos Avançados, Rod. dos Tamoios, km 5,5, Putim, CEP 12.228-001, São José dos Campos, São Paulo, Brasil
felipe@ieav.cta.br
Instituto Nacional de Pesquisas Espaciais, Av. dos Astronautas 1.758, Jardim da Granja, CEP 12227-010, São José dos Campos, São Paulo, Brasil
demisio@lac.inpe.br

Abstract. Abstract. The automatic motion or trajectory planning is essential for several tasks that lead to the autonomy increase of Unmanned Aerial Vehicles (UAVs). This work proposes a Dijkstra algorithm for fixed-wing UAVs trajectory planning. The navigation environments are represented by sets of visibility graphs constructed through the terrain elevations of these environments. Digital elevation models are used to represent the terrain elevations. A heuristics to verify if a trajectory is collision-free is also proposed in this work. This heuristics is a method of grid-based local search which presents linear computational time $O(n_p)$, where n_p is the number of verification steps. This heuristics is compared with another method for collision verification. Results are presented in this work.

Keywords: fixed-wing UAV; motion planning; Dijkstra algorithm; digital elevation model; grid-based local search.

1 Introduction

The automatic motion or trajectory planning is essential for several tasks that lead to the autonomy increase of Unmanned Aerial Vehicles (UAVs). This work approaches the problem of trajectory planning for fixed-wing UAVs navigating to constant altitudes. This problem is similar to the trajectory planning for nonholonomic wheeled mobile robots [1]. A trajectory is a sequence of waypoints connected by straight line segments and arcs that allow to the fixed-wing UAV navigates between an initial and a final waypoint of the navigation environment. A trajectory must be collision-free and dynamically feasible [1], i. e., a trajectory must allow the UAV navigate from a waypoint to the next waypoint without violating the UAV dynamics and kinematic constraints. A formal definition of dynamically feasible trajectory is presented in [1]. Fig. 1 presents an example of a dynamically feasible collision-free trajectory planned for a fixed-wing UAV.

A recent revision of UAV motion planning methods is presented in [2]. As mentioned in this revision, Dijkstra algorithms assure optimality when used to plan navigation paths, which are sequences of waypoints connected by straight line segments. However, the paths planned by these algorithms must be transformed in trajectories through the application of smoothing methods. A Dijkstra algorithm is proposed in [3] to solve the single-destination UAV shortest trajectory problem based on visibility graphs, without the necessity of application of smoothing methods. This algorithm is a modification of the original Dijkstra algorithm [4] used for the single-source shortest path problem. The Modified Dijkstra Algorithm (MDA) uses rectangular representations of the non-navigable regions of the navigation environment to create the collision-free nodes and edges of the visibility graphs. The rectangular representation is also used by a method to check a trajectory is collision-free. In the worst case, the computational time of this method is $O(n_o)$, where n_o is the number of obstacles or non-navigation regions of the navigation environment. Due to this computational time, the MDA is not efficient for navigation environments with a large number of obstacles as, for example, the navigation environments with surfaces represented by digital elevation models [5]. A digital elevation model is a computational representation of the terrain elevation of a navigation environment.

Therefore, this work proposes an Elevation-based Dijkstra Algorithm (EDA) that is an adaptation of the MDA [3] aiming at the planning of dynamically feasible collision-free trajectories through visibility graphs and digital elevation models.

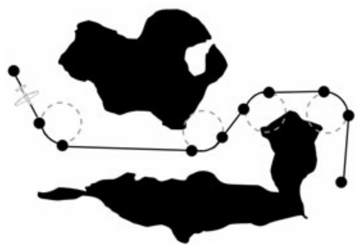


Fig. 1. Example of a planned trajectory. The planned trajectory is represented by *circles*, *straight line segments* and *arcs*. The *circles* indicate the waypoints of the trajectory. The *black polygons* represent the non-navigable regions of the navigation environment. The *stippled gray circles* indicate the circles defined by the UAV turning radius.

EDA presents three main distinctions regarding the MDA. The first distinction is the elaboration of the proposed algorithm to solve the single-pair shortest trajectory problem. As second distinction, EDA plans trajectories of the class k-trajectories proposed in [1]. A k-trajectory is a Dubins curve whose length and form can be adjusted by the parameter k. The last distinction is that this work also proposes a grid-based local search heuristics to verify if a trajectory is collision-free. This heuristics is invariant to the number of obstacles and presents a computational time of $O(n_p)$, in the worst case, where n_p is the number of verifications. Considering navigation environments with a large number of obstacles, in comparison with the method used in the modified Dijkstra algorithm, this heuristics allows a significant decrease of the computational time, becoming the main contribution of this work.

The remainder of this paper is organized as follows. Section 2 presents the construction of visibility graphs which represent the navigation environment used in this work. Section 3 presents the Dijkstra algorithm proposed in this work and presents the results of the application of this algorithm. Final conclusions based on the obtained results are presented in Section 4.

2 Visibility Graphs

A visibility graph is a set of nodes defined by the convex vertexes of polygons and connected by edges that do not intersect such polygons [6]. In path planning problems, visibility graphs are used to represent the navigable regions of navigation environments and allow the planning of the shortest paths between two waypoints. Thus, the nodes of a visibility graph are waypoints and the polygons are the obstacles or non-navigable regions of the navigation environment. Fig 2. presents an example of visibility graph.

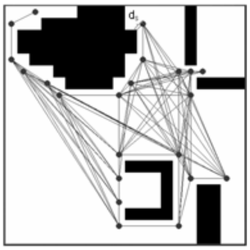


Fig. 2. Example of a visibility graph. The *black polygons* represent the non-navigable regions. The *gray circles* and the *gray straight line segments* represent respectively the nodes and edges of the visibility graph.

In this work, a navigation environment can be understood as a surface with a constant altitude a_n and divided in a regular grid B. Each region $b_{lc} \in B$ corresponds to a cell e_{lc} of the elevation grid E of the respective digital elevation model. The non-navigable regions b_{lc} are defined by the cells e_{lc} with elevation superior or equal to a_n – a_s., where: a_n is the navigation altitude; and a_s is a height of safety.

A single navigation environment is represented by a set of visibility graphs. The nodes and edges of the visibility graphs are determined by the application of two algorithms proposed in [5] which construct a visibility graph directly from a digital elevation model, considering: a constant altitude of navigation a_n; a constant height of safety a_s; and a constant distance of safety d_s.

Each node of a visibility graph constructed by the first of the algorithms proposed in [5] is a geographic coordinate which belongs to the axis that intersects the extremity and the central coordinate of a non-navigable region b_{lc}. The distance between a node and the extremity that defines this node is the distance of safety d_s. The second algorithm connects the nodes through edges that do not intersect the non-navigable regions and present a distance equal to d_s of these regions.

These algorithms were applied to the construction of visibility graphs based on the digital elevation model presented in Fig. 3a. The datum and projection of this digital elevation model are, respectively, the WGS-84 and the geographic projection. The resolution of this model is r_s = 30 meters (m), i. e., each cell e_{lc} represents a region with height and width equal to 30 m. The elevation grid E is a square matrix with order 1205. Two results of the application of these algorithms are presented in Fig. 3b

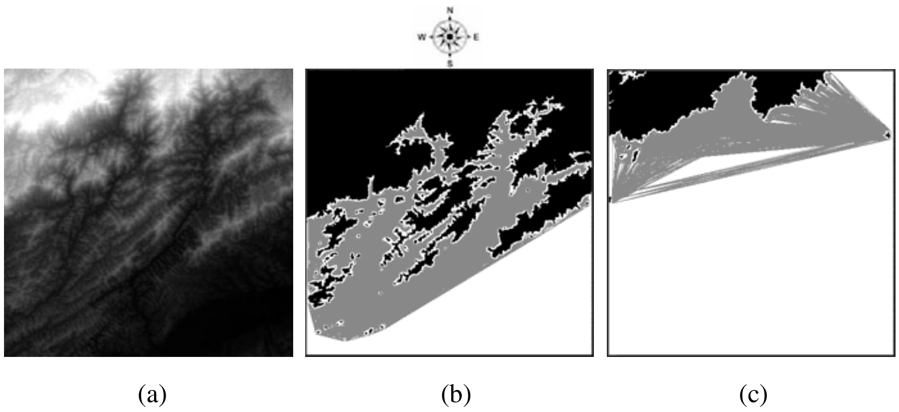


Fig. 3. (**a**) Digital elevation model used in this work. The *clear areas* represent the areas with larger elevation. Sets of visibility graphs constructed by the mentioned algorithms: (**b**) a_n = 1100 m; and (**c**) a_n = 1700 m. The *black polygons* indicate the non-navigable regions. The *gray circles* and the *gray straight line segments* represent, respectively, the nodes and edges of the sets of visibility graphs.

Table 1. Features of the obtained sets of visibility graphs

Sets of Visibility graphs	n_n	n_e	n_o
Fig. 3b	1489 nodes	65454 edges	724548 non-navigable cells
Fig. 3c	243 nodes	9540 edges	159478 non-navigable cells

and Fig. 3c, considering the height of safety a_s = 300 m and the distance of safety d_s = 100 m. Features of the obtained sets of visibility graphs are presented in Table 1. It is used the notation n_o for the total number of non-navigable cells b_{lc}.

3 Dijkstra Algorithm Based on Terrain Elevation

As mentioned previously, this work proposes an EDA to solve the single-pair shortest trajectory problem for fixed-wing UAVs. The algorithm is based on visibility graphs and digital elevation models of the navigation environments.

In [1] is proposed the class of *k*-trajectories based on the kinematic constraints of an UAV. An example of *k*-trajectory is presented in Fig. 4a. Given a path composed by the waypoints w_{i-1}, w_i and w_{i+1}, a *k*-trajectory is the unique, dynamically feasible, minimum-time trajectory which allows the navigation from the waypoint w_i to the waypoint w_{i+1}, passing directly through *p(k)* [1].

As proven in [1], for k = 1, the trajectory is the shortest *k*-trajectory. Due to this fact, the proposed EDA plans *k*-trajectories, for k = 1 (1-trajectory), as presented in Fig. 4b. However, the EDA can be adapted simply for others *k* values. The proposed EDA is presented in Table 2.

This way, analyzing the Fig. 4b, a 1-trajectory planned by the EDA through a visibility graph is composed by: a straight line segment $\overline{w_1 w_{2e}}$; an arc from w_{2e} to w_{2s}; a sequence of a straight line segments $\overline{w_{(i-1)s} w_{ie}}$ connected by arcs from w_{ie} to w_{is}; and a straight line segment $\overline{w_{(n_t-1)s} w_{n_t}}$, for i varying from 3 to n_t - 1, where n_t is the number of elements of the trajectory. In Fig. 4, $\vec{q}$ is an unit vector defined by the unit vectors $\vec{q}_{i-1}$ and $\vec{q}_{i+1}$.

The waypoints w_1 and w_{n_t} are the initial waypoint w_s and the final waypoint w_d, respectively. These waypoints can be any coordinate of the navigable regions and they are inserted in the set of visibility graphs which represents the navigation environment. The waypoints w_l are nodes of the visibility graph, for l varying from 1 to n_n. The waypoints w_{ie} and w_{is} are, respectively, the input waypoint and the output waypoint of the curve defined by the UAV turning radius.

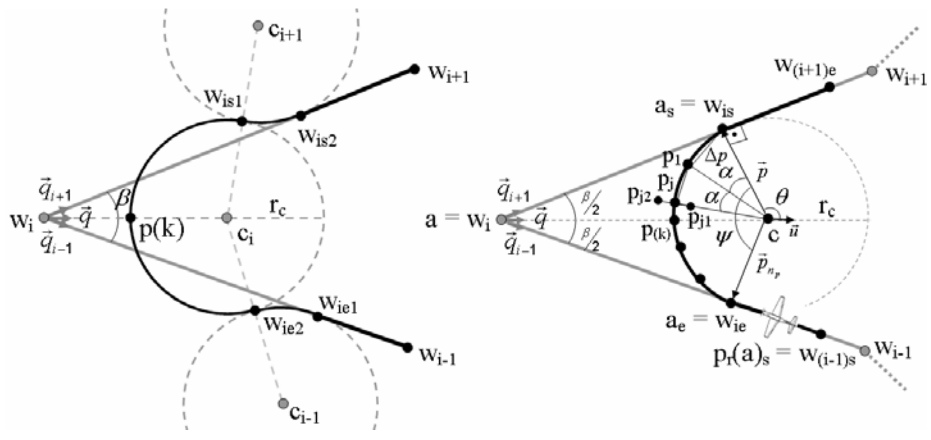


Fig. 4. (**a**) Example of a k-trajectory from w_{i-1} to w_{i+1}. (**b**) Discretization of the arc from w_{ie} to w_{is} of a 1-trajectory planned by EDA. The trajectories are represented by *black straight line segments*, *black arcs* and *black circles*.

Due to the this UAV turning radius, the waypoints w_{ie} and w_{is} must comply with the constraints given by Eq. 1 and Eq. 2. The planned 1-trajectory is dynamically feasible, if and only if the Eq. 1 and the Eq. 2 be not violated for i varying from 1 to n_t. These constraints are assured by the line 10 of the EDA, as presented in Table 2.

$$d(w_i, w_{ie}) \le d(w_i, w_{(i-1)s}), \forall i. \quad (1)$$

$$d(w_i, w_{is}) \le d(w_i, w_{(i+1)e}), \forall i. \quad (2)$$

Where: d is the distance between two waypoints.

Table 2. Elevation-based Dijkstra Algorithm. Where: *d'* is the distance between the initial node w_s and the analyzed node; p_r is the previous node of the analyzed node; r_c is the UAV turning radius; and h_{cs} is the heuristics to verify if an arc from a_e to a_s is collision-free.

```
Index  Elevation-based Dijkstra Algorithm (EDA)
1      For $i$ varying from 1 to $n_n$ do
2          store $w_i$ in $Q$
3          $d'(w_i) \leftarrow \begin{cases} 0, para\ w_i = w_s \\ \infty, para\ w_i \neq w_s \end{cases}$
4      $a \leftarrow w_s$
5      while $a \neq w_d$ do
6          remove $a$ from $Q$
7          for each neighbor $v$ of $a$ do
8              $f \leftarrow d'(a) + d(a,v)$
9              determine $a_e$, $a_s$, $p_r(a)_s$ through $p_r(p_r(a))$, $p_r(a)$, $a$ and $v$
10             if $d(a,a_s) > d(a,v_e)$ or $d(a,a_e) > d(a,p_r(a)_s)$ do
11                 $f \leftarrow \infty$
12             else
13                 if $h_{cs}(a_e, a_s, E) = 1$ do
14                     $f \leftarrow \infty$
15             if $f < d(v)$ do
16                 $d'(v) \leftarrow f$
17                 $p_r(v) \leftarrow a$
18         $a \leftarrow min\ (d(w_i))$, for $w_i \in Q$
19     while $a \neq w_s$ do
20         store $a$ in the stack $T$
21         $a \leftarrow p_r(a)$
```

Each $\overline{w_{(i-1)s} w_{ie}}$ is a straight line segment of the edge that connect the nodes w_{i-1} and w_i of the visibility graph. This fact assures that the straight line segments of a trajectory are always collision-free, as explained in the previous section.

However, the visibility graph does not assure that the arcs are collision-free. Therefore, it is necessary the use of a method to verify if the arcs of a trajectory are collision-free. As mentioned previously, this work proposes a heuristics h_{cs} to verify collision situations in the arcs. The line 13 of the EDA is the application of this heuristics.

Section 3.1 presents the heuristics proposed in this work to verify collision situations. Section 3.2 presents results of the application of the proposed EDA.

3.1 Grid-Based Local Search Heuristics

The heuristics proposed in this work verifies if a fixed-wing UAV navigating through an arc intercepts some non-navigable cell $e_{lc} \geq (a_n - a_s)$ through a local search in *E*. As presented in Fig. 4b, the heuristics discretizes the arc from a_e to a_s in a sequence of points p_j, for *j* varying from 1 to $n_p = \psi / \alpha$. Analyzing the Fig. 5a, at each iteration *j*, the heuristic verifies if the polygon that contains the possible trajectory of the UAV between p_{j-1} and p_j intercepts some non-navigable cell b_{lc}. The heuristics is presented in Table 3.

Table 3. Heuristics to verify if an arc from a_e to a_s is collision-free

Index	**Grid-based Local Search Heuristics (h_{cs})**
1	calculate n_p
2	$collision \leftarrow 0$
3	$j \leftarrow 1$
4	while $j < n_p$ and $collision = 0$ do
5	calculate p_j, p_{j1}, p_{j2} and p'_j
6	through the proposed theorem, if the polygon defined by $\overline{p_{(j-1)1}p_{(j-1)2}}$, $\overline{p_{(j-1)1}p_{j1}}$, $\overline{p_{(j-1)2}p'_j}$, $\overline{p_{j1}p_{j2}}$ and $\overline{p_{j2}p'_j}$ intercepts some non-navigable cell b_{lc} do
7	$collision \leftarrow 1$
8	else
9	$j \leftarrow j + 1$
10	return $collision$

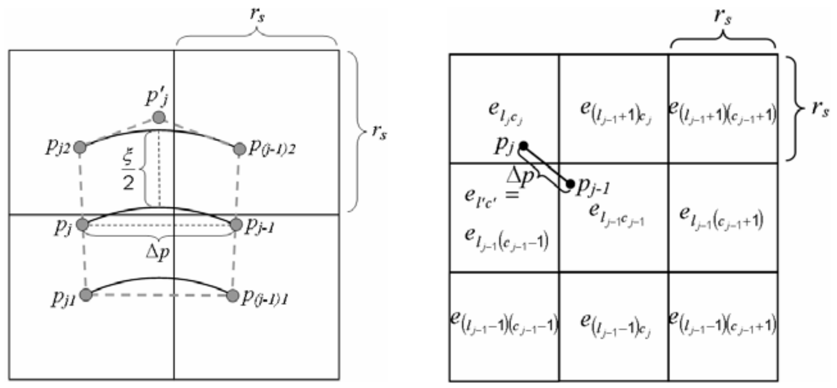
Each point p_j is defined by Eq. 3.

$$p_j = \begin{cases} c + \begin{bmatrix} r_c\, cos(\theta + j\alpha) \\ r_c\, sin(\theta + j\alpha) \end{bmatrix}, \textit{if c, p and } w_i \textit{ is counterclockwise} \\ c + \begin{bmatrix} r_c\, cos(\theta - j\alpha) \\ r_c\, sin(\theta - j\alpha) \end{bmatrix}, \textit{if c, p e } w_i \textit{ is clockwise} \end{cases} . \tag{3}$$

The angle α is associated with the distance Δp between p_j and p_{j+1}. The distance Δp is based on the distance between two robot states or positions presented in [7] to verify collision situations. The problem of this approach is to assure that any collision is detected by the discretization. This work proposes a theorem that assures that the discretization of an arc in n_p points p_j spaced by Δp always allows the collision verification. The theorem is also valid for the straight line segments that define the pentagon of collision verification.

Theorem: given: a navigation surface B and the respective elevation grid E; the wingspan $\xi \leq \Delta p$ and the length $\zeta \leq \Delta p$ of the UAV; and a discretization of an arc in n_p points p_j, a straight line segment $\overline{p_{j-1}p_j}$ with length $\Delta p < r_s$ is collision-free if: the cells $b_{l_{j-1}c_{j-1}}$ and $b_{l_j c_j}$ that contain respectively p_{j-1} and p_j are navigable; and the cell $b_{l'c'}$ intercepted by $\overline{p_{j-1}p_j}$ is navigable, when $l_j \neq l_{j-1}$ and $c_j \neq c_{j-1}$, as presented in Fig. 5b.

Proof: if $\Delta p < r_s$ then the point p_j belongs to a cell $b_{l_j c_j}$ of the neighboring of the cell $b_{l_{j-1}c_{j-1}}$, where $|l_j - l_{j-1}| \leq 1$ and $|c_j - c_{j-1}| \leq 1$. Therefore, the segment $\overline{p_{j-1}p_j}$ only intercepts a cell $b_{l'c'}$ when $l_j \neq l_{j-1}$ and $c_j \neq c_{j-1}$. Due to this fact, $\overline{p_{j-1}p_j}$ does not intercept any other cell b_{lc}.



(a) (b)

Fig. 5. (**a**) Verification of collision through a polygon that contains the trajectory of the UAV between the waypoints p_{j-1} and p_j. The *black arcs* represent the trajectory of the UAV. The polygon is represented by the *gray stippled pentagon*. The *black cells* represent a set of cells b_{lc}. (**b**) Local search in the neighboring of a cell $e_{l_{j-1}c_{j-1}}$ of the grid E.

This way, the angle α is defined by Eq. 4 to assure the distance Δp.

$$\alpha = 2 arcsin\left(\frac{\Delta p}{2r_c}\right). \tag{4}$$

When an UAV navigates through an arc, the projection of the wingspan ξ of the UAV is equal to $\xi sin(90^o - \omega)$, where ω is the lift angle of the UAV. However, as presented in Fig. 5a, it was considered that this projection, distance between p_{j1} and p_{j2}, is equal to ξ, as a way to increase the safety of the planned trajectories.

3.2 Application of the Proposed EDA

EDA was applied to the trajectory planning considering the sets of visibility graphs obtained in Section 2.

Initially, a set of specifications was considered to apply the EDA. Analyzing the Eq. 4, it can be observed that n_p decreases when the value of Δp increases. Due to this fact, the value of Δp is specified to be equal to $0.99r_s$ in the application of the algorithm. Analyzing the Fig. 4b, it can be observed that $0^o < \psi < 180^o$. Then, it was considered that the worst case for n_p occurs when: ψ is equal to 179.99^o; and to each iteration j of the heuristics, three cells are evaluated for each straight line segment of the pentagon. Therefore, in the worst case, the number of cells verified by the heuristic is $n_c = 12n_p = 12(179.99/2arcsin(0.99r_s/2r_c))$, considering that the straight line segment $\overline{p_{(j-1)1}p_{(j-1)2}}$ is evaluated in the previous iteration.

Table 4. Comparison between the methods to verify collision situations in an arc

Sets of Visibility graphs	proposed heuristics $O(n_p)$	method used in MDA $O(n_o)$
Fig. 3b	n_p = 32 iterations, n_c = 384 cells	n_o = 724548 cells
Fig. 3c	n_p = 32 iterations, n_c = 384 cells	n_o = 159478 cells

Table 4 presents a comparison between the proposed heuristics and the method used in MDA to verify collisions in an arc, analyzing the time complexity in the worst case. It was considered r_c = 300 m, $\Delta p = 0.99 r_s$ = 29.7 m and ψ = 179.99°.

Analyzing the obtained results, the proposed heuristics is more efficient than the method used in MDA, when the navigation environment is represented by a regular grid and $n_o > 12 n_p$ cells, in the worst case.

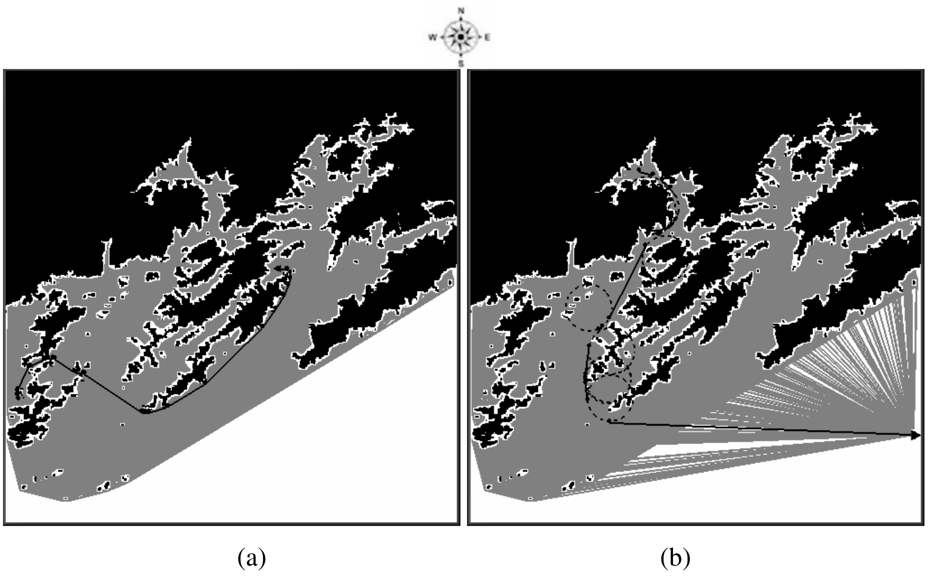


Fig. 6. 1-trajectories planned by EDA, considering: (**a**) r_c = 300 m, w_s = (latitude, longitude) = (-23.075, -46.072), and w_d = (-22.98, -45.87); and (**b**) r_c = 2000 m, w_s = (-22.91, -45.95), and w_d = (-23.0, -45.735). The planned 1-trajectories are represented by the *black straight line segments*, *black circles* and *black stippled arcs*. The arcs of the 1-trajectories are sections of the *black stippled circles* defined by r_c. The *black arrow* indicates the waypoint w_d. The *gray circles* and the *gray straight line segments* represent, respectively, the nodes and edges of the sets of visibility graphs.

Examples of 1-trajectories planned by EDA are presented in Fig. 6, considering d_s = 100 m. As demonstrated in the previous section, these shortest 1-trajectories are collision-free and dynamically feasible for fixed-wing UAVs with the following constraints: $\xi < r_s$ = 30 m; ζ < 30 m; and $r_{cmin} \le r_c \le r_{cmax}$, where r_{cmin} and r_{cmax} are the minimum and maximum turning radius of the vehicle, respectively. The edges that connect the waypoint w_d to the set of visibility graphs can be visualized in Fig. 6b.

In all the tests, the time to plan was inferior to 2.95 seconds (s). The tests were run in a computer with one processor of 1.7 GHz and with 1 GB of RAM.

4 Conclusions

The obtained results prove the efficiency of the Elevation-based Dijkstra algorithm (EDA) proposed in this work to plan trajectories for fixed-wing UAVs. These trajectories are dynamically feasible and collision-free. The algorithm presents a local search heuristics to verify collision situation that is based on the elevation grid of the navigation environment. This proposed heuristics allows the EDA an improvement in comparison with the MDA, decreasing, significantly, the computational time.

As future work, the proposed heuristics will be used in Rapidly–exploring Random Trees (RRTs) aiming for the fixed-wing UAV trajectory planning. The results will be compared with the results obtained by the EDA applied to visibility graphs.

References

1. Anderson, E.P., Beard, R.W., McLain, T.W.: Real-time dynamic trajectory smoothing for unmanned air vehicles. IEEE Transactions on Control Systems Technology 13(3), 471–477 (2005)
2. Goerzen, C., Kong, Z., Mettler, B.: A Survey of Motion Planning Algorithms from the Perspective of Autonomous UAV Guidance. Journal of Intelligent and Robotic Systems, 65–100 (2010)
3. Kuwata, Y., How, J.P.: Stable trajectory design for highly constrained environments using receding horizon control. In: Proceedings of the 2004 American Control Conference, Boston, Massachusetts, pp. 902–907 (June 2004)
4. Dijkstra, E.W.: A Note on Two Problems in Connection with Graphs. Numerische Mathematik 1, 269–271 (1959)
5. Medeiros, F.L.L., Silva, J.D.S.: Grafos de Visibilidade Aplicados à Representação Computacional de Ambientes de Navegação Aérea. In: X– Simpósio de Aplicações Operacionais em Áreas de Defesa (SIGE), São José dos Campos – SP (2008)
6. Nilsson, N.J.: A Mobile Automaton: An Application of Artificial Intelligence Techniques. In: Proceedings of the International Joint Conference on Artificial Intelligence, pp. 509–520. ACM, New York (1969)
7. Sanchez, G., Latombe, J.C.: A Single-Query Bi-Directional Probabilistic Roadmap Planner with Lazy Collision Checking. In: Jarvis, R.A., Zelinsky, A. (eds.) Published in Robotics Research: The Tenth Int. Symp. Springer Tracts in Advanced Robotics, pp. 403–417. Springer, Heidelberg (2003)