Pathfinding in a 3D Grid for UAV Navigation

Vivian Chiciudean, Florin Oniga
Computer Science Department
Technical University of Cluj-Napoca
Cluj-Napoca, Romania
Chiciudean.Au.Vivian@student.utcluj.ro, Florin.Oniga@cs.utcluj.ro

***Abstract*—In this paper, we propose an approach for computing the 3D trajectory of UAVs between two locations when obstacles are present. The result is an obstruction free path, close to optimal. First, the precomputed 3D environment map is converted into a discrete voxel space. Next, the A* algorithm is applied on the discretized space. The A* method provides the optimal path with respect to the graph equivalent representation of the voxel space, however in the continuous 3D space the resulting path will cause unnecessary steering maneuvers for the UAV. The solution is to smooth the resulting path using an iterative linear approximation approach. A new representation of the 3D path is obtained, consisting of line segments and control points. Therefore, we manage to transform the A* path from the graph equivalent representation of the voxel space to the continuous representation of the 3D environment. The resulting control points can be used as intermediate destinations during autonomous UAV navigation. Experiments are performed on multiple scenarios, demonstrating that the proposed method shortens the standard A* path for UAV navigation in the 3D environment.**

Keywords—Pathfinding, 3D, UAV, A*, Navigation, Grid, Bresenham, voxelization, path smoothing

I. INTRODUCTION

The concept of Unmanned Aerial Vehicles dates since the beginning of the 20th century, and has significantly evolved over time, due to the technological progress. Drones are now used in a wide range of applications, including cinematography, surveillance and monitoring of hard-to-reach areas of interest, agriculture, transportation and aerial mapping. The size can vary from centimeters to meters and they can carry a wide range of loads in these conditions. These devices allow, by either mounting a camera sensor or other specialized sensors, to perceive the environment in which they are roaming.

Nowadays, one of the major features needed for UAVs is autonomous navigation. However, this involves a lot of challenges. The first major problem is that the weather can be unfavorable, destabilizing the drone and its sensors. Simultaneously, the geolocation position may be inaccurate, which can lead to the loss or collision of the drone. Thus, in order to localize, avoid obstacles and plan paths, complex algorithms involving a variety of specialized sensors are required. Furthermore, the execution time for these operations should be within seconds or less. If the UAV is placed in an environment that is not fully observable, the detection of obstacles can be very difficult and misleading, requiring very high costs. In this regard, control points that depict intermediate locations for the UAV trajectory can be used to represent an obstacle free path during navigation.

For real world scenarios where autonomous UAV navigation is needed, the continuous space must be represented appropriately to allow finite searches, a process known as discretization. Discretization brings certain advantages, but with them come various limitations that will impact navigation.

The goal of the work described in this paper is to get an obstacle free path as close as possible to the real optimal one, in a timely manner, relying on the 3D map of the environment for navigation. Thus, we will focus on the techniques required to create a navigable path, the necessary refinements and the application of these principles to the UAV domain: mesh-based 3D map of the environment, voxel representation, an iterative custom approach to refine the path provided by the A* algorithm and to select the best control (intermediate) trajectory points.

In the following sections, we will present recent advancements and other research in this topic of interest, we'll outline the main necessary steps proposed, followed by their rigorous detailing, and lastly, we shall evaluate the proposed approach and present relevant results.

II. RELATED WORK

There are various approaches to digitally represent a real environment, but one of the most prominent ways is the concept of polygonal mesh. A mesh consists of a set of points, representing polygon vertices, grouped and interconnected to approximate various surfaces. We will use the triangle as the representative polygon because it is implicitly a convex shape, which will speed up the computations. With the help of a specialized camera sensors mounted on the drone and specific post-processing of the images obtained from the real environment, impressive results can be achieved in terms of the digital reconstruction of the scene. An in-depth review of the latest reconstruction methods is presented in [1], while [2] shows an effective method for reconstructing the three-dimensional digital world using images captured by drones, along with a comparison of tools that can assist and speed up the process. The Structure from Motion approach is often used to create a polygonal mesh for virtual modeling of the real world.

In order to obtain a discrete representation of the environment, [3] presents a method for voxelization of polygonal meshes that accurately eliminates common voxelization artifacts at edges and vertices. Reference [4] presents the advantages and disadvantages of the various

voxel shapes used for voxelization. In this paper, we propose to use cuboidal voxels with a standard 26-neighborhood, hence simplifying the subsequent processes of voxelization and pathfinding.

There are a lot of possible solutions to deal with the problem of navigating in a virtual space, presented in [5], [6], [7] and [8]. However, we want to use a virtual space discretized by voxels, equivalent to a three-dimensional grid. Regarding this, [9] and [10] give a viewpoint on two-dimensional grid search, comparing different search algorithms and representations. A* is commonly used in a variety of pathfinding problems, including games [11]. In our approach, the algorithm A* will be used as baseline with a proposed post processing step that shortens the path length. A* is known for its speed, using Euclidean distance as heuristic in order to reduce the search space. An in-depth examination of the characteristics of possible heuristics and their optimizations is presented in [12] and [13].

An extremely difficult problem that has been identified is the fact that the minimum distance obtained by the A* algorithm on the 3D grid is not the same as the minimum distance from the continuous space. This limitation is caused by two key factors. The first issue is that the turning angle along the trajectory is limited to a multiple of 45 degrees due to the topology of the voxel space (the 26 possible neighbors that allow straight or diagonal motion). The second issue is due to the unfeasibility to design a heuristic that provides a robust estimation of the cost for the path in an environment with obstacles. A potential solution for this is to use another algorithm that is closer to the minimal path, such as Lazy-Theta* [14], or a real optimal algorithm, ANYA [15]. Both solutions can achieve any angle path planning on grids, but both have a greater search space, and, implicitly, a greater computational complexity. Another idea, for avoiding sudden and frequent direction changes along the A* path is presented in [16], the path being smoothed with various curve models (polynomial, spline etc.), but without aiming explicitly to shorten the path. In order to avoid an increased response time and also to smooth and shorten the A* path, our proposed approach is a smoothing technique that also provide a series of control points as a new representation of the path. The shortest path in Euclidean space is a straight line, so we can smooth, where appropriate, the trajectory using linear approximation. In this way, we will obtain a closer to ideal route, maintain the algorithm's speed and search space and also have the control points that can guide the autonomous UAV.

III. Overview of the Proposed Approach

The input of the proposed approach is a 3D virtual scene mesh-based representation built from multiple images of a real-world environment, using the approach of [17]. The main steps for obtaining the smoothed A* trajectory with the relevant intermediate control points are as follows:

- Mesh discretization by voxelizing the space of interest and defining the search volume
- Searching for a path between two desired points, using the A* algorithm
- Apply the proposed iterative scheme to smooth the path and extract relevant control points.

IV. Detailed Methodology

A. Environment input representation

The input map, generated using the reconstruction approach described in [17], is a triangular mesh consisting of a series of vertices and triangle facets that represent surfaces and scene objects (Fig. 1 and Fig. 2). In particular, the test scenario used has 700 thousand triangle vertices and 1.5 million faces, covering an area of approximatively 200 x 200 meters with an elevation variance of 40 meters. The scene can be identified at the following coordinates (46.697391, 23.547135) in Google Maps.

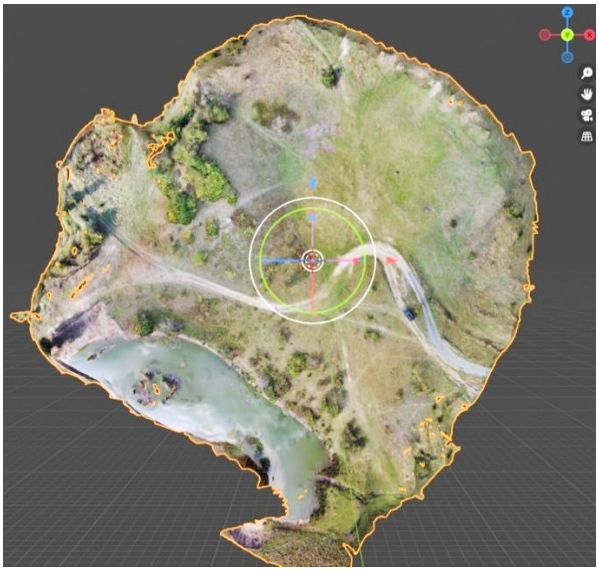

Fig. 1: Input map, top view

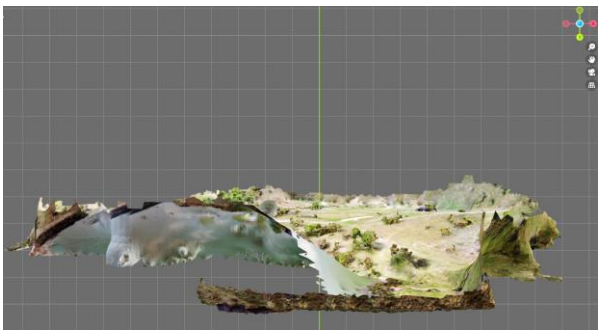

Fig. 2: Input map, perspective view

B. Voxel space representation

This pre-processing step will provide an appropriate discrete representation of the scene, that allows searching for an obstacle-free path in a timely manner.

In the voxel representation built from the input mesh, each voxel that contains part of the mesh is considered *occupied*.

Because the input map to be voxelized has a large size (200 x 200 x 40 meters), the voxel size needs to be tailored in order to have an acceptable resolution but, also, to have a manageable memory footprint. Assuming a size of 0.3 meters for each axis of the volumetric elements (cubes), the discretized space becomes very large, resulting in approximately 59 million voxels. Since the map does not show height variations over its entire extent, we can take advantage of the sparse scene. Because we want to retain information about obstacles in an efficient way in terms of memory, we

propose to retain information through two important data structures. First, we'll need a three-dimensional vector of integer elements that represents every voxel in the volume of interest. The values in this multi-dimensional vector encode the following information. If the value is -1 (*empty*) it means that we do not have information of interest in that discretized area. Otherwise, it means that we have an *occupied* voxel and the voxel value is an index in a one-dimensional vector that stores all the occupied voxels. The one-dimensional vector retains, for each occupied voxel, relevant information such as 3D position and other features that might be used for future development. Basically, we leverage the discretization and memory storage of a sparse map.

Fig. 3 shows an example of scene discretization using a size of 0.3 meters on the X, Y and Z axes for a volumetric element and a particular volume of interest (2.1 meters in height, 6.3 meters in length and 3.3 meters in width). To highlight the discretization, the drawing of each voxel was made with a size of 0.2 meters instead of 0.3 meters, hence the free spaces between the volumetric elements. In this example, the property of *empty* / *occupied* for the generated voxels is not taken into account.

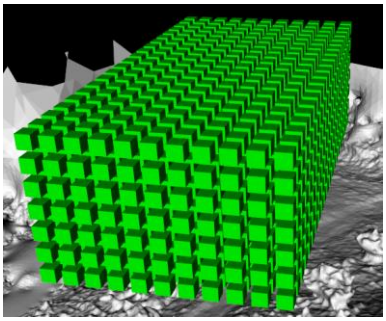

Fig. 3: Voxel volume (all voxels) example

Fig. 5 shows all occupied voxels in the red polygon of Fig. 4 using 0.3 meters on each axis for the display of volumetric elements.

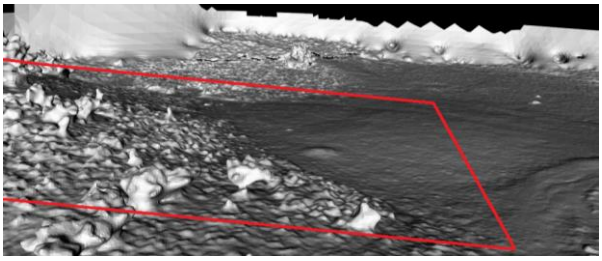

Fig. 4: Map volume of interest (textureless mesh, for a better visualization)

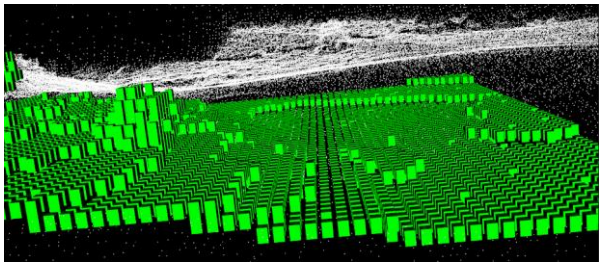

Fig. 5: Voxelized volume of interest (different perspective, for the area depicted in Fig. 4). Only the mesh vertices are displayed (white points)

To have a reliable representation of the environment, the voxel size must be chosen small enough to represent all the relevant objects and surfaces from the scene. However, we must also consider the size of the UAV. For instance, in this research work, the DJI Matrice 210 V2 RTK drone was used. This drone size is (at least on two axes) almost 3 times the size of the selected dimension of the voxel and will be considered during the A* pathfinding.

C. Searching for an initial path

As mentioned, to find a free path between two points we will initially use the standard A* algorithm. This involves using a priority queue to select the most appropriate node to explore next. To determine the priority, we will consider two factors: first, the actual distance between the starting node and the node we wish to explore, and second, the approximate distance between the node to be investigated and the destination node, using a heuristic based on Euclidean distance. The Euclidean distance between the voxels (x1, y1, z1) and (x2, y2, z2) is defined by (1).

$$d^2 = (x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2 \quad (1)$$

The search will take place in a neighborhood of 26 voxels, as illustrated in Fig. 6, with each neighbor being examined to see if it is an obstacle or not. Depending on the drone size, this test must be done on the proximity of each checked neighbor to ensure that the drone can safely navigate there. If the voxel is completely navigable, it will be kept as a potential exploration node.

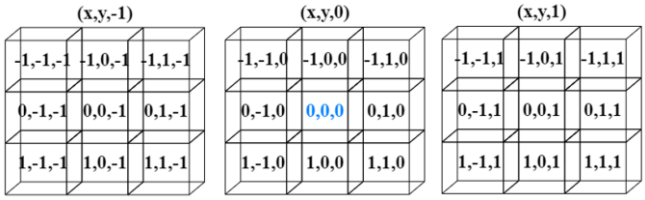

Fig. 6: 26-voxel-neighborhood with relative displacements

As previously stated, voxelization of the space imposes a limitation on changing the direction of travel. The turning angle along the trajectory is limited to a multiple of 45 degrees due to the topology of the voxel space. Thus, there will be small differences between the real minimal path and the obtained path. This leads to errors similar to those in Fig. 7, where the trajectory formed by the red cubes represents the resulting A* path and the green line indicates the desired path.

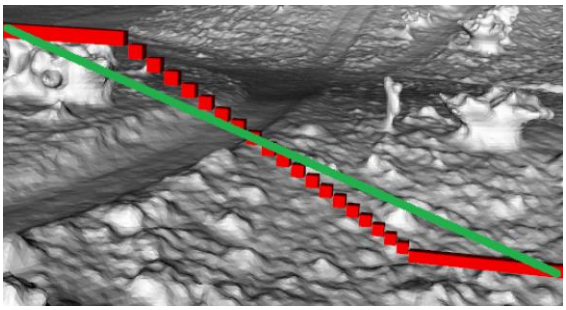

Fig. 7: Difference between A* result and the continuous optimal path

Another issue is that the utilized heuristic does not enable us to determine when to change direction. If we combine the current heuristic with another heuristic that is based on the distance from a straight line, as in [18], we would obtain real minimal paths but only if no obstacles are present. When obstacles are encountered on the path, the trajectory will present unwanted direction changes. This situation is shown

in Fig. 8, the red path is the result of A* using Euclidean distance and the distance from the straight-line as a combined heuristic, and the green path is the desired and optimal one.

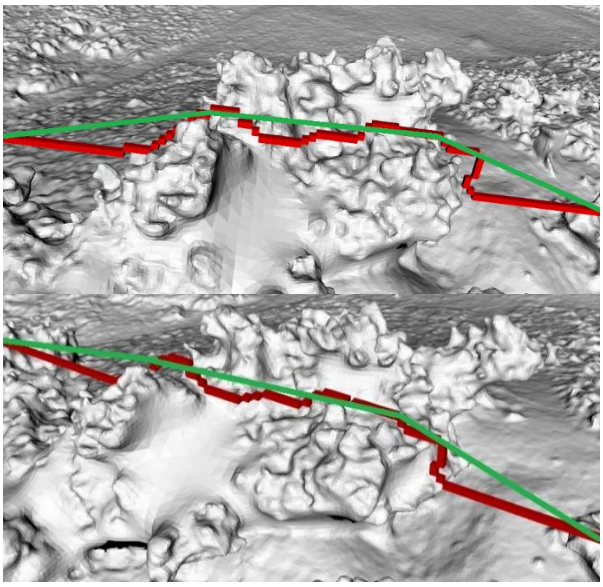

Fig. 8: Combined heuristic (red), two different perspectives of the same path

Therefore, the A* path in the voxel space will not be minimal in the corresponding continuous space due to these two limitations, requiring further refinement.

D. *Path smoothing*

A* implies visibility in the immediate neighborhood and not the full interconnection of the voxels, as a complete graph. Fig. 9 shows the connectivity assumed by the A* algorithm in the voxel space representation of the continuous space. Each voxel has such a vicinity during the exploration phase of A*, seen as a vertex with connecting edges. The spheres in this image represent graph vertices, the green one being the reference, and the lines depict the graph edges in the A* search.

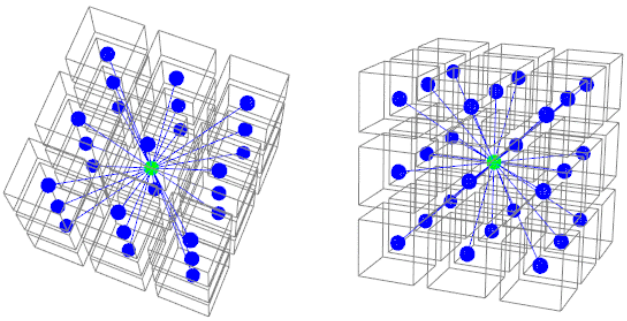

Fig. 9: The direct neighborhood of one voxel

The smoothing technique proposed is based on line segments and control points to approximate a shorter path and to reduce the number of direction changes. Thus, if the refined trajectory is used for autonomous navigation, the drone travels with fewer risks while approaching objects. The algorithm takes as input the A* path, represented by the array of voxels $P_1..P_n$. The smoothing begins by marking the starting point as a control point, followed by iterating the path and tracing lines in the voxel space between the last marked control point and the current point. If the line drawn to the current point intersects with *occupied* voxels, the previous point becomes the current control point. A pseudocode description is highlighted in **Algorithm 1**.

Algorithm 1 Smoothing algorithm

Require: A valid input path

$P \leftarrow path$
$Start \leftarrow P_1$
$ControlPoints \leftarrow [Start]$
repeat
  $Stop \leftarrow P_i$
  $Line \leftarrow Bresenham(Start, Stop)$
  if *Line* intersects obstacle **then**
    $Start \leftarrow P_{i-1}$
    $ControlPoints.insert(Start)$
  end if
until entire *path* is processed
Return $ControlPoints$

In order to render the three-dimensional lines and to check for occupied locations in the voxel space representation, the Bresenham algorithm is used to generate the discrete line points [19]. The example of a line approximation for the two-dimensional case, using this algorithm, is presented in Fig. 10.

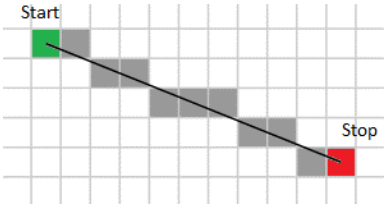

Fig. 10: Bresenham line approximation

The smoothing of the path also brings a possible relaxation of the heuristic, favoring a greedier strategy and therefore decreasing the execution time and search space. Even if we obtain paths that are not necessarily perfect, once we smooth the path, it will get closer to the optimal one.

V. Experimental Results and Discussion

The proposed solution was implemented in C++ on a machine with an i7 CPU running at 4.7 GHz, 8 cores, 16 GB RAM and a dedicated NVIDIA GeForce RTX 2060 6 GB graphics card for visualization.

A comparison of the running time using the A* algorithm and the proposed method (A* + smoothing) for different path lengths is presented in TABLE I.

TABLE I. Execution Time Comparison

Path length	Method	Time (ms)
Short (10 m)	A*	13.4
	Proposed method	14.9
Medium (50 m)	A*	253.5
	Proposed method	258.7
Long (90 m)	A*	2715.7
	Proposed method	2728.3

Fig. 11 shows the first path determined after running the A* algorithm across a distance of around 85 meters in a straight line, on an uneven terrain with vegetation.

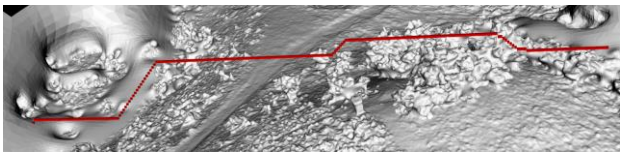

Fig. 11: The path resulting from the A* algorithm, perspective view

Fig. 12 highlights the control points obtained after smoothing the initial path.

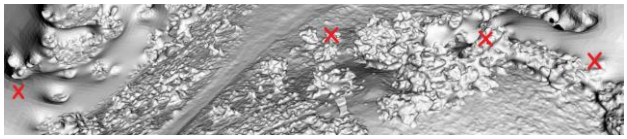

Fig. 12: Control points

Finally, Fig. 13 shows the final path obtained by the proposed method.

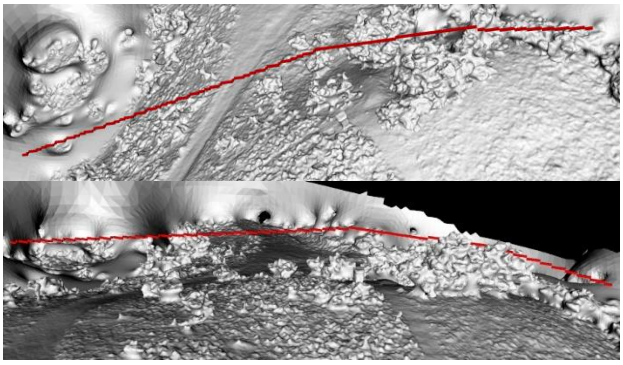

Fig. 13: Smoothed path, two different perspective views

The final path is smoother and shorter than the initial one. Additionally, compared to the results obtained with A* using combined heuristics, the smoothed path does not get so close to obstacles, thus increasing the safety of the navigation. In order to obtain a quantitative evaluation of the presented method, we proposed a statistical approach by which we will compare, using different test scenarios, the length of the obtained paths. We will consider both a high elevation region of the voxel volume, so that there are no obstacles, and a low elevation region where each path encounters at least one obstacle.

The first evaluation is performed on obstacle-free paths generated in the high elevation region. We analyze the difference between the real Euclidean path, the path generated by the proposed method and the path generated by A*. In this assessment, 180 paths obtained using 180 randomly generated pairs of start and stop voxels are used, without any obstacles in between. Fig. 14 and Fig. 15 illustrate the generated straight paths, with a mean length of 50 meters and a standard deviation of 21 meters.

TABLE II. STATISTICAL COMPARISON OF PATH LENGTH IN AN OBSTRUCTION-FREE ENVIRONMENT, 180 PATHS, MEAN VALUES (%)

Method	Distance (%)
Straight line (Euclidean)	100
A*	105.2
Proposed method	100

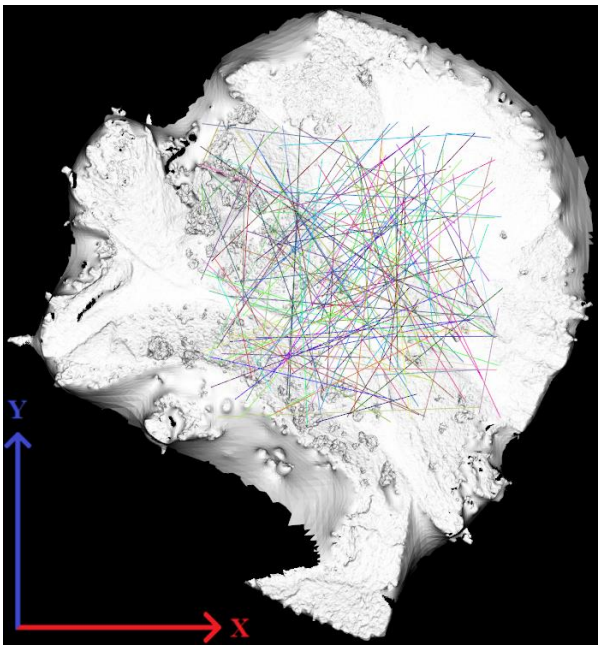

Fig. 14: Top view of the 180 paths

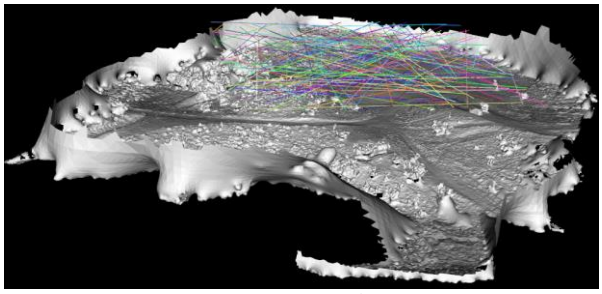

Fig. 15: Perspective view of the 180 paths

The obstacle-free environment allows us to analyze the variation of the difference between the Euclidean distance, equal to the distance of the proposed method (between the start and end control voxels of the straight path), and the distance generated with the help of A*. TABLE II presents a statistical view in terms of path shortening for the obstacle-free environment using the two possible methods compared with the ground truth (Euclidian). The mean value of the A* paths is larger than the smoothed paths by 5.2%, with the standard deviation of 2.6%.

Also, since there are no obstacles, we can make an analysis of the variation of distance according to the angle formed with the abscissa of the coordinate system. This procedure is illustrated with the help of Fig. 16, where the X axis represents the angle of the path with respect to the abscissa of the voxel coordinate system, and the Y axis represents the percentage difference between the path length generated by A* and the proposed method. The largest differences are obtained for angles whose value is halfway between two perfect paths in the discrete space, i.e. 22.5 degrees, 67.5 degrees, 112.5 degrees and 157.5 degrees. The paths aligned with the main axes and bisectors are not improved by the smoothing technique because, in these situations, the Euclidian heuristic in the discrete A* search provides a cost similar with the distance in the continuous space.

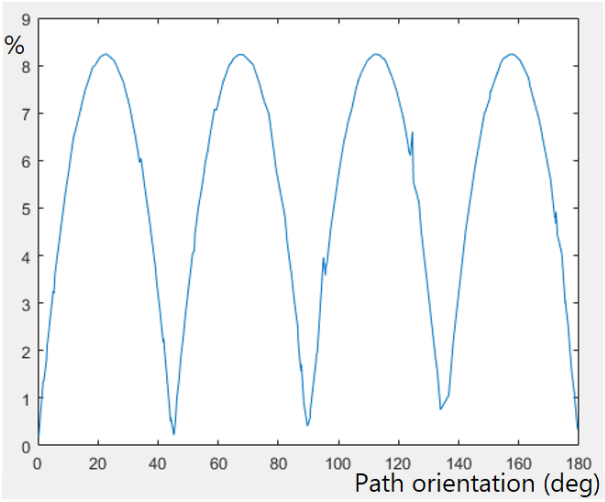

Fig. 16: The difference (%) between the path length generated by A* and the proposed method, for all the possible path orientations

The second evaluation is performed on paths generated in the low elevation region, where obstacles are present. We analyze the difference between the path generated by the proposed method and the path generated by A*. In this assessment, 50 paths obtained using 50 randomly generated pairs of start and stop voxels are used (Fig. 17 and Fig. 18). The paths have a mean length of 30 meters and a standard deviation of 12 meters. The chosen region has a size of about 30 x 30 meters and a high density of obstacles (mainly vegetation).

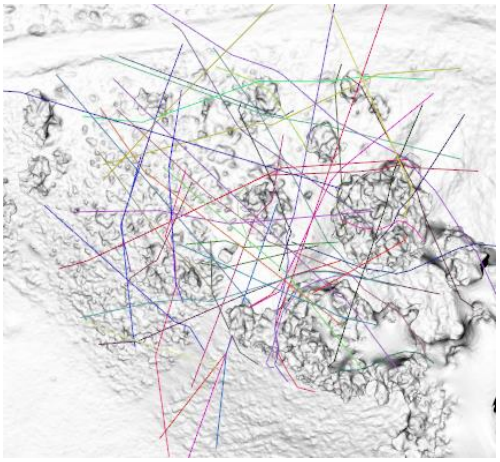

Fig. 17: Top view of the 50 paths

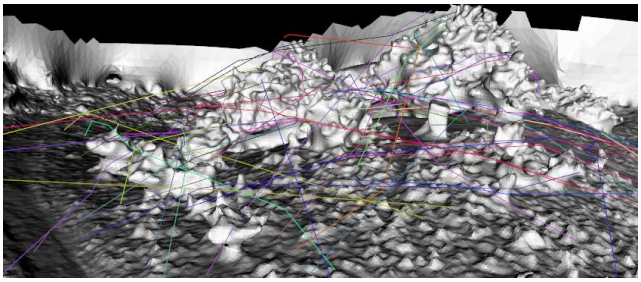

Fig. 18: Perspective view of the 50 paths

The 50 paths are generated so that there is at least one obstacle between the start and stop points. Therefore, we will not be able to compare the distance with a straight line, so we will take the A* path length as a reference. TABLE III shows the path length improvement for the chosen region using the 50 paths and the two possible methods. The smoothed paths are smaller than the A* paths by 6.5% (mean value), with the standard deviation of 2%.

TABLE III. STATISTICAL COMPARISON OF PATH LENGTH IN AN ENVIRONMENT WITH OBSTACLES, 50 PATHS, MEAN VALUES (%)

Method	Distance(%)
A*	100
Proposed method	93.5

VI. CONCLUSIONS

An approach for computing the 3D path was presented in this paper, with application for UAV autonomous navigation. The solution is an iterative method based on extracting a set of control points from the initial path and interconnecting them using line segments. In this sense, the proposed method improves the paths generated by A*, shortening the path length by up to 8% and minimizing the unnecessary steering maneuvers without significantly affecting the running time. The software of an autonomous drone can take advantage of the control points and use them to navigate in a real environment.

In terms of future work, a comparison can be made between the proposed method and other state of the art pathfinding algorithms. Another possible development is the identification and integration of more complex smoothing methods and trajectory shapes that approximate the real movement pattern of the drone, regardless of its speed. In the current phase, the proposed method is only valid for low speeds, without considering the dynamic maneuverability of the drone.

ACKNOWLEDGMENT

This work was partially supported by the “SEPCA-Integrated Semantic Visual Perception and Control for Autonomous Systems” grant funded by Romanian Ministry of Education and Research, code PN-III-P4-ID-PCCF-2016-0180.

References

[1] J. L. Schönberger and J.-M. Frahm, "Structure-from-Motion Revisited," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, 2016.

[2] S. Jiang, C. Jiang and W. Jiang, "Efficient structure from motion for large-scale UAV images: A review and a comparison of SfM tools," *ISPRS Journal of Photogrammetry and Remote Sensing* , pp. 230-251, 2020.

[3] J. Huang, R. Yagel, V. Filippov and Y. Kurzion, "An accurate method for voxelizing polygon meshes," in *IEEE Symposium on Volume Visualization*, North Carolina, 1998.

[4] D. Cavagnino and M. Gribaudo, "Discretization of 3D models using voxel elements of different," in *Computational Aesthetics in Graphics, Visualization, and Imaging*, Styria, 2010.

[5] R. Graham, H. McCabe and S. Sheridan, "Pathfinding in Computer Games," *The ITB Journal*, vol. IV, no. 2, pp. 57-81, 2003.

[6] Z. A. Algfoor, M. S. Sunar and H. Kolivand, "A Comprehensive Study on Pathfinding Techniques for Robotics and Video Games," *International Journal of Computer Games Technology* , pp. 1-11, 2015.

[7] Z. He, M. Shi and C. Li, "Research and application of path-finding algorithm based on unity 3D," in *2016 IEEE/ACIS 15th International Conference on Computer and Information Science (ICIS)*, Okayama, 2016.

[8] A. Botea, B. Bouzy, M. Buro, C. Bauckhage and D. Nau, "Pathfinding in Games," *Artificial and Computational Intelligence in Games*, pp. 21-31, 2013.

[9] P. Yap, "Grid-Based Path-Finding," in *Proceedings of the 15th Conference of the Canadian Society for Computational Studies of Intelligence on Advances in Artificial Intelligence*, Heidelberg, 2002.

[10] P. Mehta, H. Shah, S. Shukla and S. Verma, "A Review on Algorithms for Pathfinding in Computer Games," in *International Conference on Innovations in Information Embedded and Communication Systems*, Coimbatore, 2015.

[11] X. Cui and H. Shi, "A*-based Pathfinding in Modern Computer Games," *International Journal of Computer Science and Network Security*, pp. 125-130, 2011.

[12] A. Y. Kapi, "A Review on Informed Search Algorithms for Video Games Pathfinding," *International Journal of Advanced Trends in Computer Science and Engineering*, pp. 2756-2764, 2020.

[13] S. K. Sharma and S. Kumar, "Comparative analysis of manhattan and euclidean distance metrics using A* algorithm," *Journal of Research in Engineering and Applied Sciences*, pp. 196-198, 2016.

[14] A. Nash, S. Koenig and C. A. Tovey, "Lazy Theta*: Any-Angle Path Planning and Path Length Analysis in 3D," in *Proceedings of the Third Annual Symposium on Combinatorial Search*, Atlanta, 2010.

[15] D. Harabor, A. Grastien, D. Öz and V. Aksakalli, "Optimal Any-Angle Pathfinding In Practice," *Journal of Artificial Intelligence Research*, pp. 89-118, 2016.

[16] A. Ravankar, A. Ravankar, Y. Kobayashi and Y. Hoshino, "Path smoothing techniques in robot navigation: State-of-the-art," *Sensors*, vol. XVIII, no. 9, p. 3170, 2018.

[17] H. Florea, V.-C. Miclea and S. Nedevschi, "WildUAV: Monocular UAV Dataset for Depth Estimation Tasks," in *2021 IEEE 17th International Conference on Intelligent Computer Communication and Processing (ICCP)*, Cluj-Napoca, 2021.

[18] S. Aine, S. Swaminathan, V. Narayanan, V. Hwang and M. Likhachev, "Multi-heuristic A*," *The International Journal of Robotics Research*, vol. 35, no. 1-3, pp. 224-243, 2016.

[19] J. E. Bresenham, "Algorithm for computer control of a digital plotter," *IBM Systems Journal*, vol. 4, no. 1, pp. 25-30, 1965.