future internet

MDPI

Article

EAOA: Energy-Aware Grid-Based 3D-Obstacle Avoidance in Coverage Path Planning for UAVs

Alia Ghaddar * and Ahmad Merei

Department of Computer Science, International University of Beirut, Beirut P.O. Box 146404, Lebanon; 41430485@students.liu.edu.lb

* Correspondence: alia.ghaddar@liu.edu.lb

Received: 31 December 2019; Accepted: 31 January 2020; Published: 8 February 2020

Abstract: The presence of obstacles like a tree, buildings, or birds along the path of a drone has the ability to endanger and harm the UAV's flight mission. Avoiding obstacles is one of the critical challenging keys to successfully achieve a UAV's mission. The path planning needs to be adapted to make intelligent and accurate avoidance online and in time. In this paper, we propose an energy-aware grid based solution for obstacle avoidance (EAOA). Our work is based on two phases: in the first one, a trajectory path is generated offline using the area top-view. The second phase depends on the path obtained in the first phase. A camera captures a frontal view of the scene that contains the obstacle, then the algorithm determines the new position where the drone has to move to, in order to bypass the obstacle. In this paper, the obstacles are static. The results show a gain in energy and completion time using 3D scene information compared to 2D scene information.

Keywords: coverage path planning; obstacle avoidance; unmanned aerial vehicle; energy consumption; completion time

1. Introduction

The usage of an Unmanned Aerial Vehicle (UAV) (also known as a drone) has become very fast-growing recently [1]. It pervades many application domains to achieve different missions such as terrain monitoring and smart agriculture [2,3], photogrammetry [4], harsh and disaster environments management [5], smart policing, wildfire tracking [6], and many more. One of the main active research areas for UAV is the Coverage Path Planning (CPP). It aims to build a path for the drone to explore every location in an Area of Interest (AoI). Often, the main concerns in CPP are the efficiency and completeness of the coverage. The effectiveness of a CPP work is mainly related to the following performance indices such as the path length [7,8], the mission completion time [9,10], and the number of turns. These metrics are related with energy consumption [11,12]. Most of the approaches try to reduce them in order to save energy.

One of the key challenges in CCP is also to build paths in scenes with obstacles or pop-up threats. Areas could have obstacles and non-flying zones. Obstacles could be static or mobile (such as trees and birds). Pop-up threats are obstacles which are not known a priori by the drone. Different path planning methods rely on two-dimensional or three-dimensional information of the scene to build the trajectory path [13].

Recent approaches in 3D path planning concentrate more on detecting and avoiding obstacle in a 3D view [13–19]. They have not given priority to compromise between the energy efficiency and the completeness of the area coverage [20] in the presence of obstacles. In our work, we propose an energy aware 3D obstacle avoidance (EAOA) which plans a full coverage path for an area of interest with the ability to avoid obstacles in a 3D view using a grid-based technique.

3D trajectory planning shows great benefits, unlike 2D path planning, with graphical limitations. Solutions are not only constrained by UAV size, weight, and flying mechanisms. There should be a compromise between the trajectory length, the total coverage rate for the area, and the energy consumption of the drone. The number of turns a drone makes during the coverage mission is also another issue and needs to be minimized. More turning points means more energy consumption of the drone [21].

In this paper, we present a grid-based path planning and a 3D obstacle avoidance technique. The planned path relies on a previous paper [14]. It combines the top-view and the frontal-view of the area that includes the obstacle. The algorithm generates the first path offline using the top-view then re-plans it using the front-view of the scene. We deal with static obstacles whose locations are known a priori to the UAV. We aim to avoid the obstacle with the shortest path and least number of turns possible taking into consideration two main factors, energy consumption and completion time.

The remainder of this paper is organized as follows: Section 2 discusses related works; Section 3 provides details about our work; Section 4 presents the evaluation metrics and results; and Section 5 concludes the paper.

2. Related Works

During the path planning mission, avoiding obstacles or non-flying zones is one of the critical challenges for UAVs. The trajectory needs not only to bypass obstacles but also to be efficient in terms of energy consumption and completion time. Obstacles could be mobile such as pigeon, waving flag, or could be fixed such as street lighting column, a building, etc.

Different research works [13,15,16] rely on either two-dimensional or three-dimensional methods for obstacle detection and avoidance. In [17], a 3D-SWAP technique for 3D collision avoidance with multiple UAVs is proposed. UAVs rely on local data from sensors and positions of other neighboring UAVs. Their approach is decentralized and requires low computational load. Another approach is presented in [18]. Authors propose a Model Predictive Control (MPC), and it aims to find an optimal path for an (UAV) in a changing environment. The work aims to avoid collision in the presence of non fly zones and pop-up threats as well as fixed obstacles.

Authors in [19] propose a Fast Geometric Avoidance algorithm (FGA) for 3D collision avoidance. The algorithm takes into account the avoidance start time based on kinematic considerations, collision probability, and navigational constraints. The efficiency of the work was evaluated with Monte Carlo simulations and scenarios in an aircraft simulator. To adapt a real-time path planning to UAV performances and capability (variations and fluctuations), authors in [22] propose an adaptive path planning for UAVs.

In [23], a method to detect changes in image patches is proposed in order to understand variations in the environment and obstacle sizes. In [24], the authors propose an approach for obstacle detection and avoidance. It uses a monocular camera on UAV and motion control measurements to imitate the human behavior in detecting objects in the environment.

Authors in [25] developed an algorithm to build a map of the environment and to generate a collision free path for autonomous UAVs. They used an RGB-D camera with sensors and tested their work in an indoor environment in the presence of obstacles. A fixed wing UAV is used in [26] in two phases. They propose a solution for obstacle avoidance by first generating an offline path, then by relying on it to generate an intelligent real-time pop-up threats avoidance path.

The work in [27] was inspired by the phenomenon of fluid disturbance to re-plan and tune the trajectory for a 3D collision-free path. The authors use the interfered fluid dynamical system (IFDS) algorithm for UAV. In [28], the noisy environment is not taken into consideration. The later approach relies on sensor readings, to localize the obstacle and understand its motion, in order to determine its velocity and autonomously plan a collision-free path.

Another algorithm for autonomous obstacle avoidance is proposed in [29]. It relies on radars and image processing of video frames to detect and avoid obstacles. It aims to reduce the computational load as UAVs are energy- and memory-constrained. In [30], authors use board passive sensors like vision systems on fixed wing drones. UAV flies at lower attitude for aerial surveillance. Determining the position of the obstacle relies on local map of the environment with measurements from sensors aboard the drone. Authors in [31] provides a 2D solution to avoid obstacles using a grid based technique. They propose an algorithm that plans a coverage path for UAVs over irregular shaped areas using a grid-based method with minimum energy consumption. Authors enhanced the algorithm proposed by Valente et al. [25], by substituting the Cost Function with a novel Energy-Cost Function. The novel algorithm comes up with 17% in energy gain compared to the algorithm provided in [25]. They also included two techniques that significantly decrease the computation Time. Authors choose different starting mission points in order to decide about the best path in an offline mode. For each path planned, they rely on the cost function to move from one cell to another unvisited neighboring cell. This produces changes in direction and consequently increases the number of turns. In our work, the UAV movement is parallel to the longest side of the grid (from the top view), which reduces the number of turns and the path length. As in [14], we take into consideration the percentage of the area in the cell to be covered, in order to increase the coverage rate. The work in [24] drops the whole cells that contain obstacles. Hence, if the obstacle occupies a small portion of a cell, the whole cell won't be covered, thus the area coverage rate decreases. Additionally, avoiding the whole cell increases the path length and the turning angles. Consequently, it increases the completion time and energy consumption.

Authors in [14] propose an energy aware coverage path planning method. They use 2D grid partitioning to scan an area of interest. Scenarios with one drone and different regular and irregular area shapes were considered. The algorithm determines the turning positions as well as the start point and the scan direction of the area. The path is planned in an offline mode and showed efficient results in terms of energy saving, area coverage, and completion time. However, the areas of interest were without obstacles and nonflying zones. In this work, we develop an extension for this algorithm. The objective is to cover an area of interest in the presence of obstacles. Our proposed algorithm re-plans the original path by scanning the facing area which contains obstacles. We try to avoid obstacles with the shorter path possible while keeping the completion time and energy consumption to a minimum.

3. Energy-Aware Grid-Based 3D-Obstacle Avoidance (EAOA)

3.1. Basic Approach

Different grid-based related works avoid the whole cells that contain obstacles by moving immediately to another grid-cell [10,31]. In Figure 1, the obstacle $Ob1$ in the second column is avoided by moving at the edge of the cell. However, in the third column, another way to avoid the obstacle $Ob2$ is by making a detour (the basic approach). We noticed that, by making a detour, the path will have a lower number of turns and shorter path length. In Section 4.2, we present the results of adopting this approach (Scenario 1). As relying on the 2D top view doesn't always give a shorter path or ensures an optimal coverage, we extend the basic approach by including 3D information. We rely on the area's frontal view to decide about the obstacles shape and location and the detour to make. In the next section, we present preliminary concepts and definitions. Results of this approach are presented in Section 4.2–Scenario 2.

3.2. Preliminaries

In our previous work [14], the area is divided into grid-cells. Each grid-cell has a rectangular shape and covers a portion of the area. The UAV footprint has a circular shape and is projected to the area portion below the drone. Hence, each grid-cell represents the inscribed rectangle in the UAV

projected footprint. The original planned path starts with the longest grid-side and crosses the center of the cells (see Figure 2a). It is plotted using the 2D top-view area scanning and is generated offline.

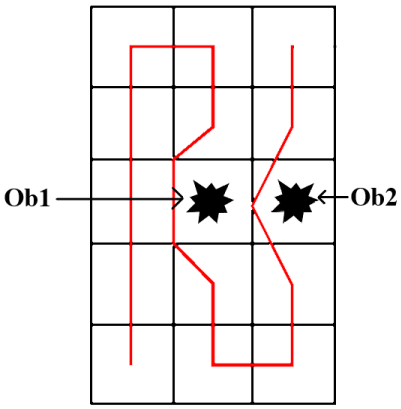
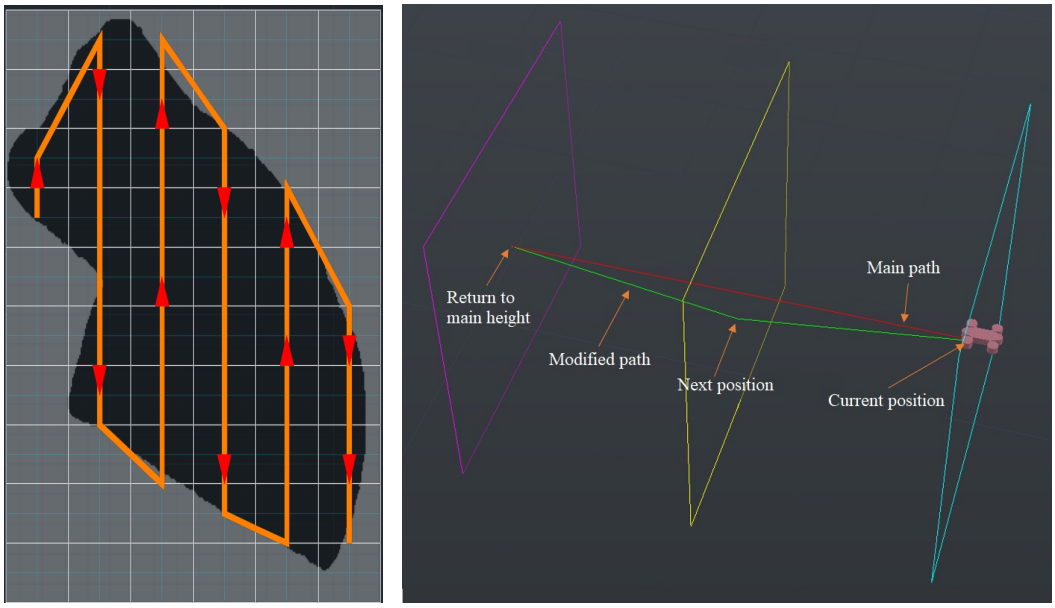


Figure 1. Different ways in avoiding obstacles.

We denote C_χ^t a grid cell in this figure and P_χ the perpendicular plane to C_χ^t which passes by the center of the cell. Hence, two consecutive planes P_χ and $P_{\chi+1}$ on the same grid column are h unit apart, where h is the length of C_χ^t. In this work, we assume the obstacles positions are fixed and known and we want to avoid them using 3D information. We lean on the original path in Figure 2a. The drone will follow it, and, right before the cell that contains the obstacle, an image for the front area is captured in order to decide later about the next drone's position. The captured image frame is to be defined in the coming section.

To illustrate, in Figure 2b, we assume having an obstacle in plane P2. The red line represents the original path in Figure 2a. The drone re-plans the path and changes it dynamically according to the locations of the obstacles. The green line represents the new trajectory. The goal of this work is to determine the new position of the drone while keeping the total trajectory path and energy consumption to a minimum.



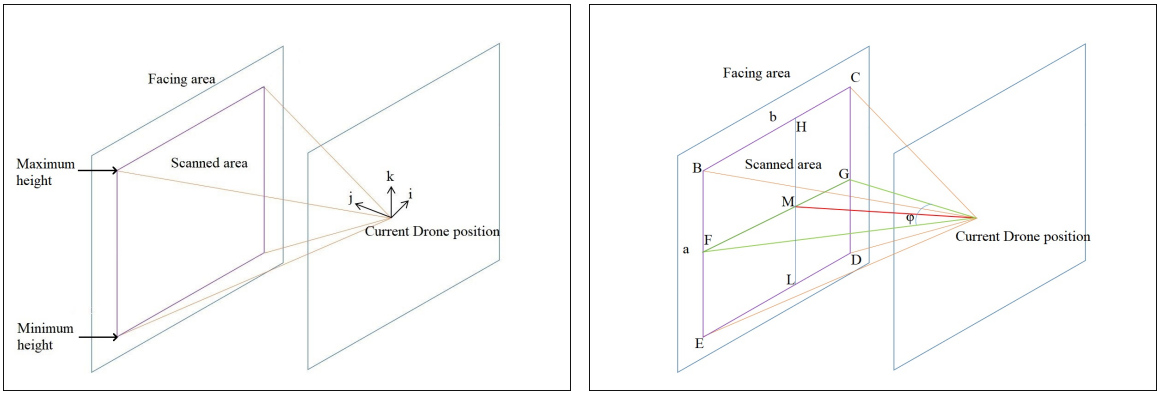
(**a**) (**b**)

Figure 2. Illustration of the top and frontal views. (**a**) top view: a binary image representing an area of interest [14]: black cells are obstacles, grey cells are free areas; (**b**) example of a frontal view with obstacle. The red line is the original path and the green one is the new path.

The main phases in our work are: (1) Determining the frame dimensions in the front view, (2) locating the next drone's position.

3.3. Frame Dimensions

Figure 3a represents the 3D view. The (i, j) plane represents the area top view, (i, k) plane is parallel to the facing view.



(**a**) Drone position at A and facing area (**b**) Scanned area dimensions

Figure 3. 3D view.

After locating the drone in the (i, j) plane, the planner will decide to range the minimum and maximum allowed heights for the drone (denoted h_{min} and h_{max}, respectively). This range will specify the frame dimensions of the facing area. We denote F as the scanned front frame. We assume that the drone is located at point A, with height h_{ijk}.

In Figure 3b, the frame is represented by the rectangle BCDE of area a $\times$ b, where a and b represent the length and width, respectively. We aim to determine the frame dimensions. The length $a = (h_{max} - h_{min})$. We assume the facing view is captured with a camera angle ϕ = 62°, similarly to the work of Al-Kaff et al. [24]. Authors found that any object detected out of this area angle will not cause any danger to the UAV.. Point M is the projection of A. Both points represent actually the centers of two consecutive grid-cells in the plane (i, j). To calculate the length (BE) and width (BC) of the scanned frame, we have to find the dimensions of the rectangular pyramid (ABCDE). Let us take the isosceles triangle (AFG) that has angle ϕ, and height h = AM. The side AF (denoted R) can be computed from Equation (1), since AM is known and equal to the length of the top-view grid cell:

$$R = \frac{h}{cos(\frac{\phi}{2})}. \tag{1}$$

Hence, the length b of the frame F can be computed as follows:

$$b = 2 \times R \times sin(\frac{\phi}{2}). \tag{2}$$

Now that the frame dimensions are known, the next step is to transform F to a grid-based form denoted Γ. The dimensions of each cell in Γ fit to the height and width of the drone. Hence, we denote v_{ij} a cell located in row i and column j in the grid Γ and has dimensions L$\times$ W (L for length and W for width).

3.4. Determining the Next Position of the Drone

In this section, we aim to determine the closest location where the drone has to move to, in order to avoid an obstacle on its way. The main steps are described below:

- Step 1: A heatmap of the frame F is generated. To exemplify, Figure 4 shows a 24×40 heatmap of a tree in the frame:

$$\Gamma = \{v_{ij} \mid v_{ij} \in [0,1], i \in [1,b], j \in [1,a]\} \tag{3}$$

where v_{ij} represents a grid cell element in Γ. We denote Γ^w the set of zero-cells in the grid:

$$\Gamma^w = \{v_{ij} \mid v_{ij} = 0\} \tag{4}$$

We denote Γ^c the set of non zero-cells in the grid ($\Gamma^c = \Gamma \backslash \Gamma^w$):

$$\Gamma^c = \{v_{ij} \mid v_{ij} \in]0,1]\} \tag{5}$$

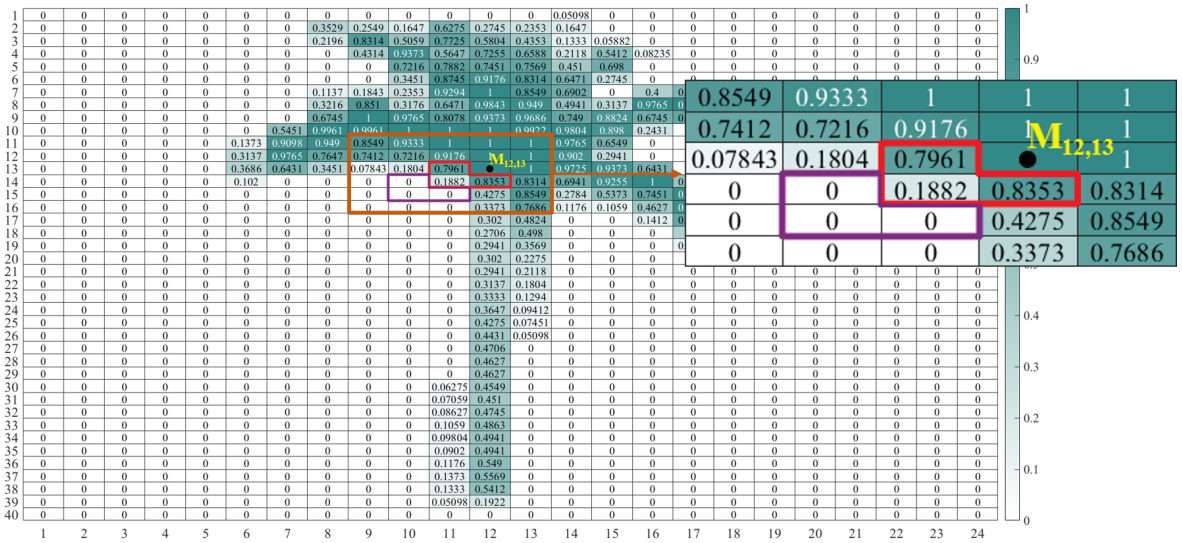


Figure 4. Example of a tree Heatmap in the frame. The drone position is (12, 13). The red box represents boundary cells, and the purple box represents solution cells.

- Step 2: To recall, M is the projection of A in the frame. We check its position in the grid. If M is located in a cell in Γ^w, then no obstacle is on the drone's way. The current path will not be modified. However, if M is located at a non-zero cell (i.e., a cell in Γ^c), then the obstacle exists on the drone's way. Hence, the path should be modified. For this purpose, we will investigate the list of cells surrounding M to find a loophole for the drone and make a detour.

We denote $\mathcal{N}_{ij}$ the neighbors of a grid-cell v_{ij}. It is the list of cells that directly surround v_{ij}:

$$\mathcal{N}_{ij} = \{v_{pq} \mid p \in [i-1, i+1], q \in [j-1, j+1]\} - \{v_{ij}\} \tag{6}$$

We denote B_m the nearest non-zero cells to M that are in Γ^c (see Equation (7)). We call them boundary cells. We denote d(v1,v2) the Euclidean distance between two points v1 and v2:

$$B_m = \{v_{ij} \in \Gamma^c \mid \forall v_{pq} \in \Gamma^c, d(v_{ij}, M) = min\{d(v_{pq}, M)\} \wedge \exists v \in \mathcal{N}_{ij} \mid v \in \Gamma^w\} \tag{7}$$

We aim to find the nearest v_{ij} to M and B_m such that v_{ij} belongs to Γ^w. We denote S_n the set of solutions, where n represents the number of white cells that directly surround B_m:

$$S_n = \{v_{ij} \in \mathcal{N}_{B_m} \cap \Gamma^w \mid \forall v_{pq} \in \mathcal{N}_{B_m} \cap \Gamma^w, d(v_{ij}, M) = min\{d(v_{pq}, M)\}\} \tag{8}$$

To find the list of solutions S_n, we loop in a reverse spiral mode from M (see Figure 5) until we reach a shell that contains cells with $v_{ij} = 0$.

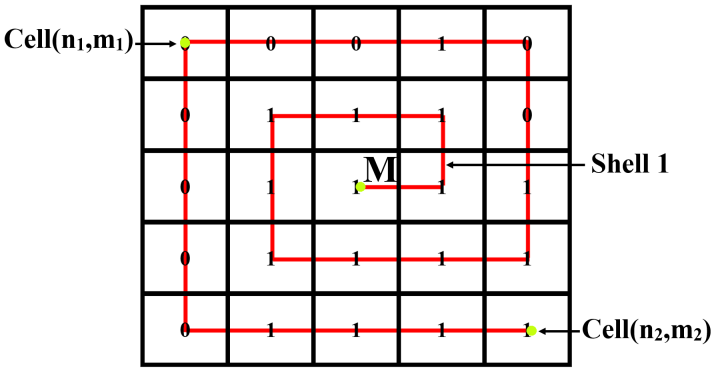


Figure 5. Determine set of Solutions S_n.

To illustrate, in Figure 4, the projection of the drone is located at cell (12, 13) in the grid. The set of boundary cells B_m is represented by the red boxes ($|B_m| = 3$). The set of possible solutions S_n is represented by the purple boxes ($|S_n| = 3$). It should be noted that, if $S_n = \emptyset$, the planner has to increase the frame dimensions.

- Step 3: To choose from S_n the most convenient solution. We denote $G = B_m \cup S_n$; see Figure 6. Then, divide each cell into four sub-cells (see Figure 7a). The reason behind this is to have better understanding of the obstacles' location in order to determine the nearest next position of the drone. After applying the sub division, a new solution set $S' \subset S_n$ is generated (see Figure 7b). The dashed red box represents the cell where the drone has to move to in order to avoid the obstacle.

0.1804	0.7961	1
0	0.1882	0.8353
0	0	0.4275

Figure 6. Group of obtained cells G.

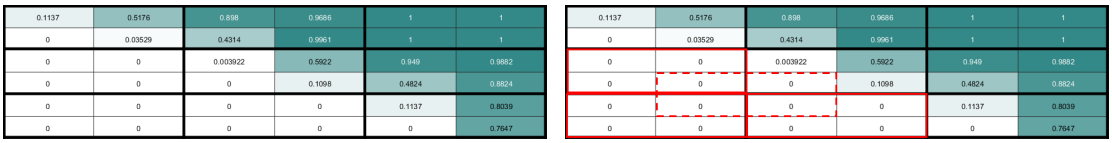


Figure 7. Finding the new position of the drone. (**a**) the heatmap after subdivision—set of new solutions after subdivision; (**b**) the dashed box is the best solution.

The flowchart in Figure 8 summarizes the main steps in our work. Algorithm 1 checks the current drone's cell-position (named M) in the facing area. If M is located in a zero-value cell, it becomes the new drone's position. Otherwise, the algorithm checks the neighboring cells to find the nearest non-zero value cell.

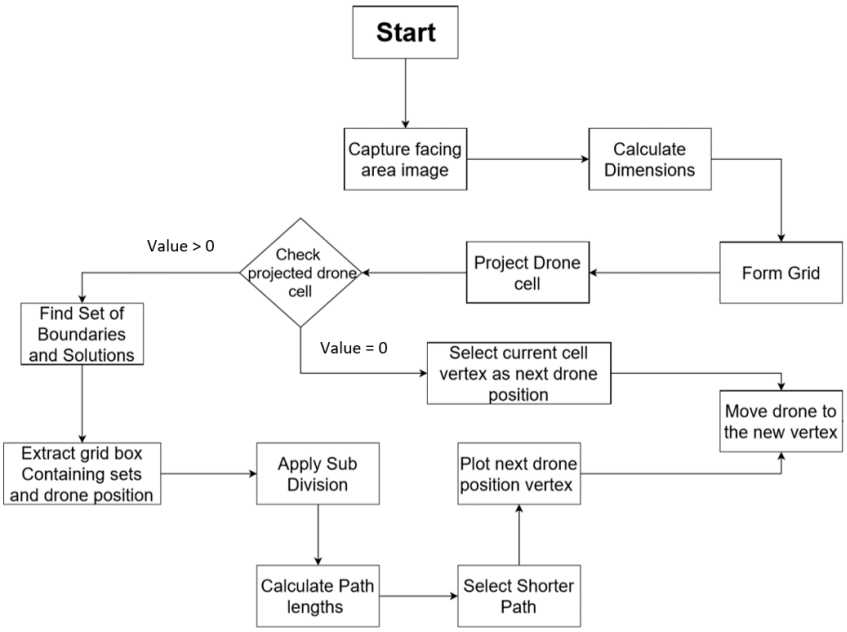


Figure 8. Flow chart of the obstacle detection and avoidance.

Algorithm 1 getDronePosition

Input: M : Cell, F: Frame
Output: new Drone's position denoted $S_{position}$

$S_n : Cell[]$
if $M.getvalue() == 0$ **then**
  $S_{position} \leftarrow M$
else
  $shellNumber \leftarrow 1$
  $CheckNeighborCells(M, shellNumber, S_n, F)$
  $PossibleDronePos \leftarrow FilterSolution(M, S_n)$
  $B_m \leftarrow FindBoundaryCells(PossibleDronePos, F)$
  $G \leftarrow B_m \Cup S_n$
  {Perform subdivision on G }
  {Group every 4 subcells after subdivision in one vertex (denote it $v^{'}$)}
  {We denote $(S_n)^{'}$ the list of vertices $v^{'}$}
  $S_{position} = FilterSolution(M, (S_n)^{'})$
end if
return $S_{position}$

Algorithm 2 (*CheckNeighborCells*) looks for the list of possible solutions S_n around M in the surrounding shells in a spiral mode starting from M. The list of solutions is filtered by Algorithm 3 (*FilterSolution*) to choose the nearest possible solution cell S' to M which could be the next drone's position.

Algorithm 2 CheckNeighborCells

Input: M: Cell, shellNumber : int, S_n : Cell[], F :Frame
Output: List of possible Solutions S_n

$n1 \leftarrow (M.getI() - shellNumber)$
$n2 \leftarrow (M.getI() + shellNumber)$
$m1 \leftarrow (M.getK() - shellNumber)$
$m2 \leftarrow (M.getK() + shellNumber)$
$Celltemp = null$
$a = F.getLength()$
$b = F.getWidth()$
if $(n2 \leq b \wedge m2 \leq a)$ **then**
  {look into the current shell for a solution (check each cell v_{ik})}
  for $i \leftarrow n1$ to $n2$ **do**
    for $k \leftarrow m1$ to $m2$ **do**
      if $(i! = M.getI() \wedge k! = M.getK())$ **then**
        if $(v_{ik} == 0)$ **then**
          $S_n.add(v_{ik})$
        end if
      end if
    end for
    if $(S_n.isEmpty())$ **then**
      $CheckNeighborCells(M, ++shellNumber, S_n, F)$
    end if
  end for
end if

Algorithm 3 FilterSolution

Input: M : Cell, S_n : Cell []
Output: The nearest solution to cell M denoted S'

$x2 \leftarrow M.getI()$
$y2 \leftarrow M.getK()$
for $index \leftarrow 0$ to $S_n.size()$ **do**
  $x1 \leftarrow S_n.get(index).getI()$
  $y1 \leftarrow S_n.get(index).getK()$
  {Find distance to M}
  $distance \leftarrow \sqrt{(x2 - x1)^2 * W + (y2 - y1)^2 * L}$
  if $(index == 0)$ **then**
    $old = distance$
  end if
  if $(old > distance)$ **then**
    $old \leftarrow distance$
    $minIndex \leftarrow index$
  end if
end for
$S' \leftarrow S_n.get(minIndex)$
return S'

In order to locate the boundaries of the obstacle around S', Algorithm 4 (*FindBoundaryCells*) finds the list of boundary cells (denoted B_m). We combine S_n and B_m in a group box G, and to decide whether the drone will pass safely through the solution cell, we perform a subdivision on the cells. This helps with finding a better solution nearest to M than the one found before the subdivision. We

group every four zero-value subcells in one vertex (denoted $v^{'}$). The list of these vertices is denoted $(S_n)^{'}$ and is filtered later using Algorithm 3 to find the exact drone position (denoted $S_{position}$). In the next section, we evaluate the performance of our approach.

Algorithm 4 FindBoundaryCells

Input: S': Cell, F :Frame
Output: List of boundary cells B_m to S'

```
1: B_m : cell[]
2: n1 ← (S'.getI() − 1)
3: n2 ← (S'.getI() + 1)
4: m1 ← (S'.getK() − 1)
5: m2 ← (S'.getK() + 1)
6: for i ← n1 to n2 do
7:    for k ← m1 to m2 do
8:       if (i! = S'.getI() ∧ k! = S'.getK()) then
9:          if (v_ik > 0 ∧ v_ik <= 1) then
10:            B_m.add(v_ik)
11:         end if
12:      end if
13:   end for
14: end for
15: return B_m
```

4. Evaluation Metrics and Results

In this section, we present different scenarios with different obstacle shapes along the drone's path. To evaluate the performance of the work, we rely on the following metrics: the energy consumption, completion time, path length, and turning angles.

4.1. Evaluation Metrics

The cells in the grid can be seen as vertices in a graph-like structure $G(V, E)$, where V is the list of vertices and E is list of edges. We denote v the center of a grid-cell before subdivision and consider it a vertex in the graph (refer to Figure 9). Each vertex has coordinates in a Cartesian plane.

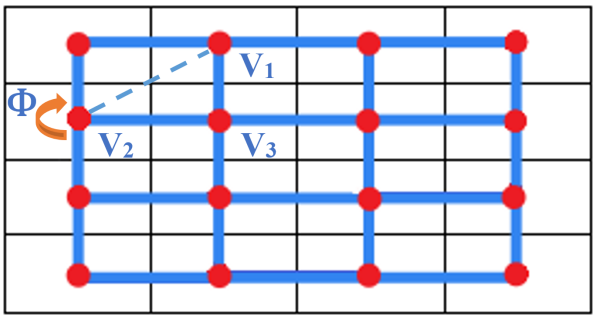


Figure 9. Graph representation: red points are vertices and blue lines represent edges.

4.1.1. Turning Angle

The power consumption is affected by the turn rate [32]. How many turns are needed during the whole flight mission is a main concern. Assume that the drone is coming from vertex v_1 to v_2 and is going to make a turn at v_2 to head later to v_3 (see Figure 9). The turning angle is computed as follows [14]:

$$\Phi_{123} = \pi - \cos^{-1}(\angle v_1 v_2 v_3) \quad (9)$$

where $\cos^{-1}(\angle v_1v_2v_3)$ is computed from the law of cosines in $\triangle v_1v_2v_3$. Hence, the angle can be found from the Euclidean distance between vertices as follows:

$$\Phi_{123} = \pi - \cos^{-1}\left[\frac{({e_{12}}^2 + {e_{23}}^2 - {e_{31}}^2)}{2e_{12}e_{23}}\right] \tag{10}$$

where $e_{ij} = d(v_i, v_j)$ is the Euclidean distance between two vertices v_i and v_j.

4.1.2. Completion Time

Optimizing the trajectory length and the mission completion time is essential in energy-efficient path planning techniuqes for UAVs [7,10,33,34]. The mission completion time includes flying time and hovering time. We lean on the following formula from [4] to find the completion time denoted τ:

$$\tau = \frac{\top}{v} + \sum_{\vartheta=1}^{\mho} \frac{\Phi_\vartheta}{\varpi} \tag{11}$$

where $\top$ is the path length, v is UAV speed, while $\mho$ is the number of turns, Φ_ϑ is the angle of ϑth turn, and ϖ is the UAV rotation rate.

4.1.3. Energy Consumption

The total energy cost of a UAV (denoted E_{total}) is the energy consumed to travel the total distance $\top_{total}$ (denoted $E(\top_{total})$) and the energy consumed to perform the turns (denoted $E(\Phi)$) [14]:

$$E_{total} = E(\top_{total}) + E(\Phi) = \sum_{i\in \mathrm{V}}\sum_{j\in \mathrm{V}} \lambda e_{ij} + \sum_{i\in \mathrm{V}}\sum_{j\in \mathrm{V}}\sum_{k\in \mathrm{V}} \gamma \frac{180}{\pi}\Phi_{ijk} \tag{12}$$

where λ is the energy consumed per unit distance, and γ is the energy consumed per angle degree. In this work, we assume that λ = 0.1164 KJ/m and γ = 0.0173 KJ/degree.

4.2. Results

- Scenario 1: The area of interest is 100 m$\times$ 115 m. The top view grid is 10 rows$\times$ 6 columns (see Figure 10). The marked red cells contain obstacles. The total trajectory path will be compared to the one obtained in the basic scenario. For the dimensions of the frame, we assume F is 16.66 m$\times$ 15 m. Each grid-cell in F is 0.69$\times$ 0.5, Γ is 40 rows$\times$24 columns. We assume that the drone is located at height 10.5 m and faces cell (12,13) in Γ. Below, we present the heatmaps and the drone's position at each obstacle.

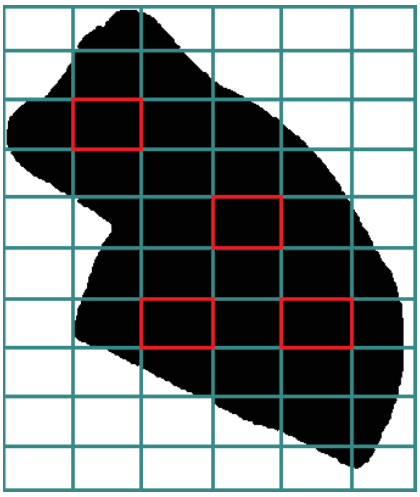


Figure 10. Area of Interest in [4].

- First, Obstacle: We assume having a palm tree of height 15 m. The heatmap of the front area is shown in Figure 4. The drone's new position is the center of the dashed group in Figure 7b. The drone thus chooses to go down through the cell solution by performing a horizontal left turn by 3 and a vertically down turn by 3°, then cuts a distance of 11.5744 m. This distance is between initial drone position and the new drone's position ($S_{position}$). To get to initial the height behind the tree, it does 6° vertically back up, crossing the same distance.

- Second Obstacle We assume having a signboard of height 13 m. The heatmap of the front area is shown in Figure 11. The projection of the drone is M(12,13). Figure 12 shows the heatmap after the subdivision. The nearest border cell is (12,5). The drone passes through the cell by performing a vertical upward turn by 16°, then cuts a distance of 11.985 m followed by a vertical turn down of 33°. Then, it returns to the initial height behind the obstacle. The red cell represents the solution cell (i.e., the final drone position $S_{position}$).

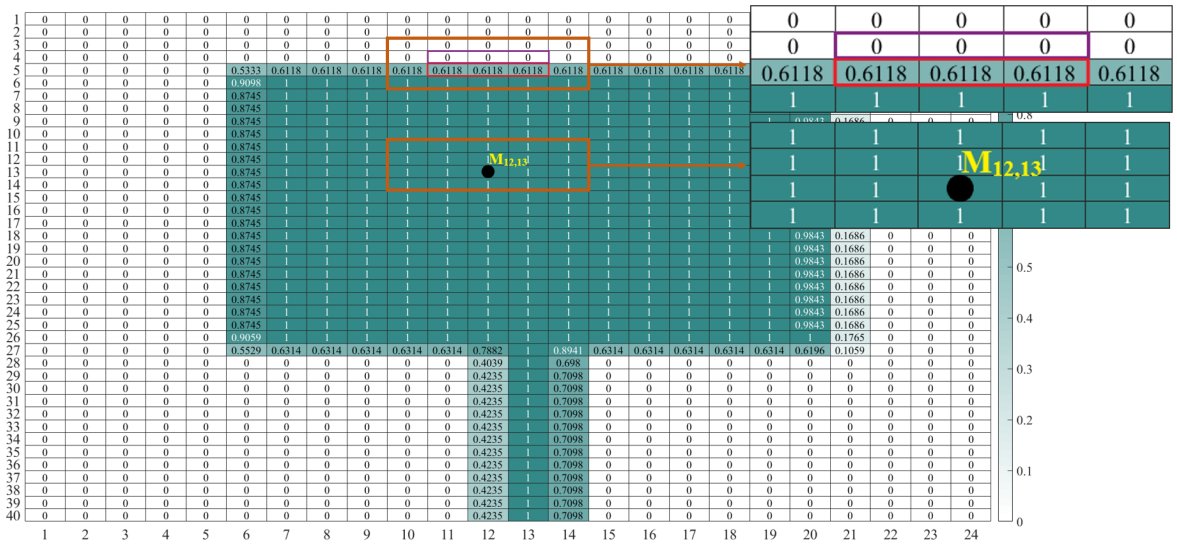


Figure 11. Heatmap with signboard as obstacle (second obstacle).

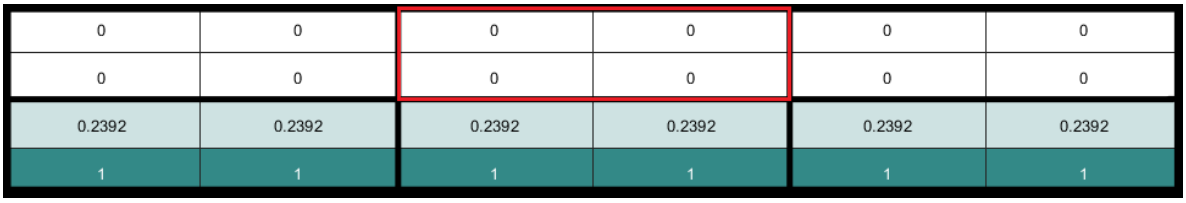


Figure 12. Subdivision for the grid Box *G* for the second obstacle.

- Third Obstacle: Assume having a tree of height 11.6 m. The heatmap is shown in Figure 13. The projection of the drone position is the red cell (12,13). The nearest border and solution cells are shown in Figure 14. The dashed red box represents the solution cell (i.e., the final drone position $S_{position}$) .

The drone will pass through the solution cell. It performs a horizontal turn by 2°, then turns upward vertically by 2° cutting a distance of 11.506 m. Then, it turns vertically down 3° to return to the initial height behind the obstacle and another turn to get back to the initial direction.

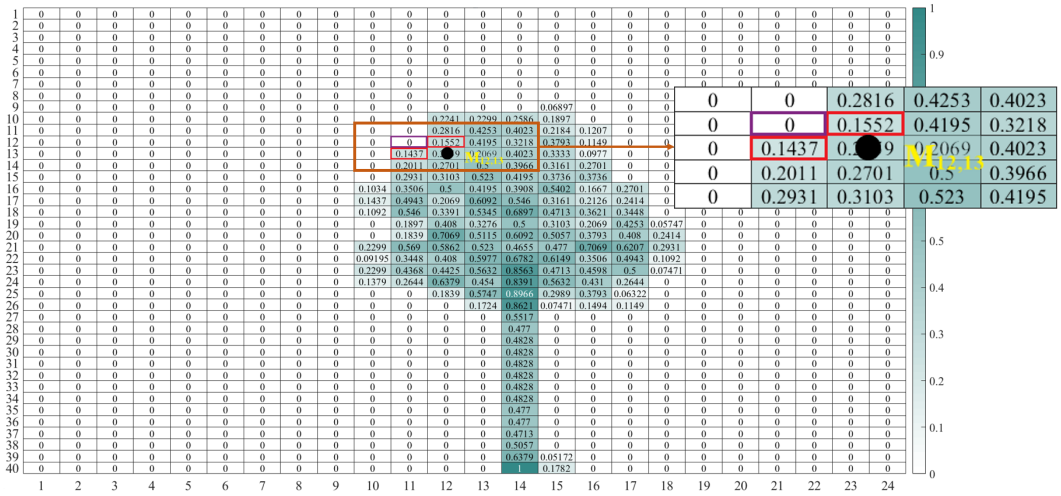


Figure 13. Tree heatmap (third obstacle).

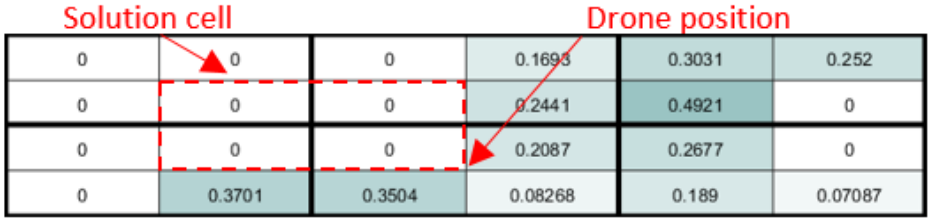


Figure 14. Subdivision for the grid Box *G* for the third obstacle.

- Fourth obstacle: Assuming the obstacle is part of a building, the heatmap is shown in Figure 15. The projection of the drone position is cell (12,13). After the subdivision, the cell (3,13) is selected solution ($S_{position}$). The red cells represent the boundaries set B_m, and the blue cells represent the solution cells S_n. The final drone position is represented in Figure 16 (the red box). The drone will pass through the cell. It performs a horizontal turn by 28°cutting a distance of 13.07 m. Then, it performs a horizontal turn by 57°. After that, it returns to the initial path direction.

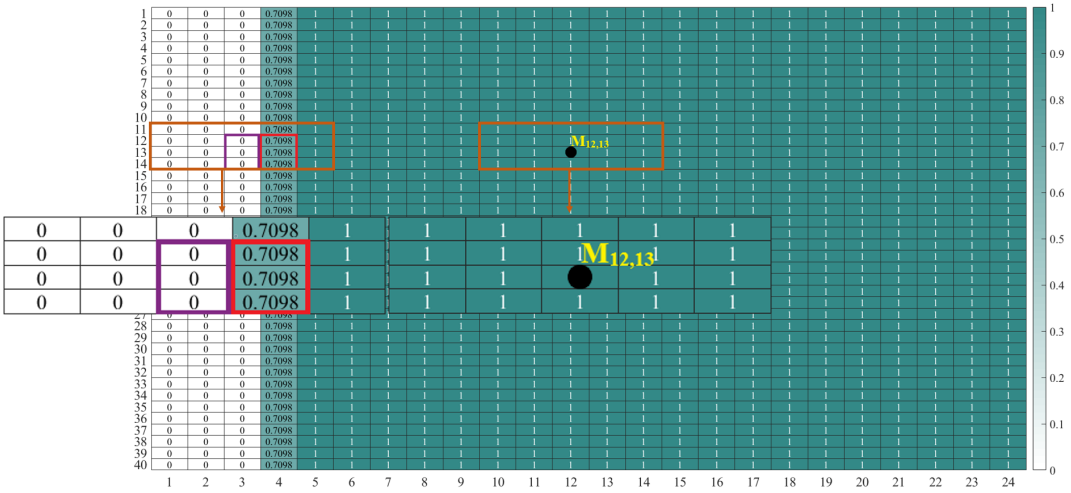
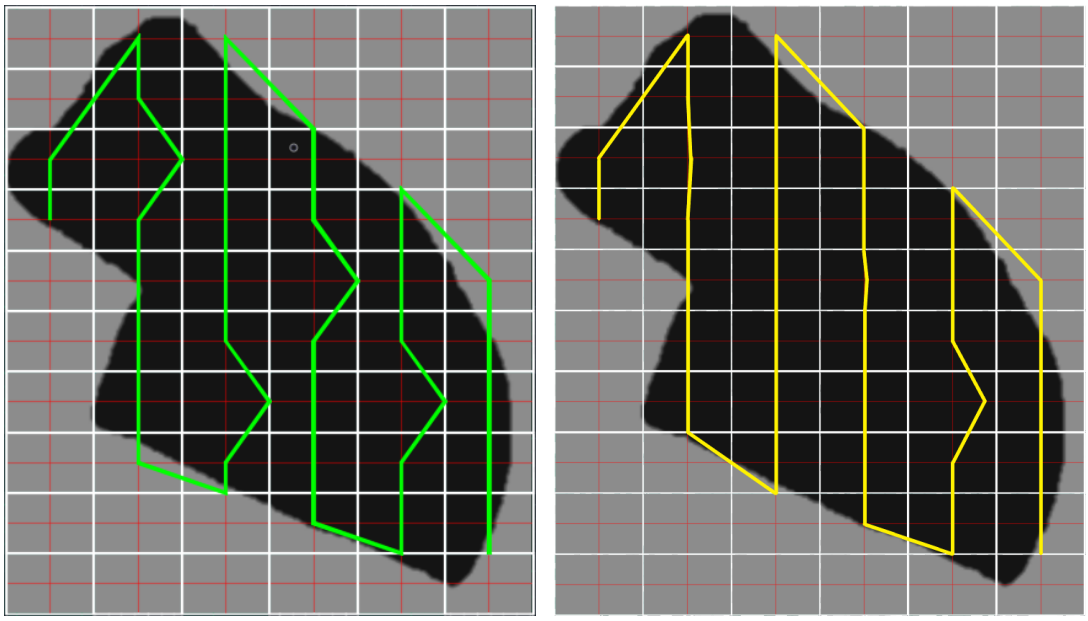


Figure 15. Heatmap of the frame containing part of a building.

0	0	0.4314	1	1	1
0	0	0.4314	1	1	1
0	0	0.4314	1	1	1
0	0	0.4314	1	1	1

Figure 16. Subdivision for the grid Box *G* for the fourth obstacle.

At the end of the mission, the resulted path by the basic approach is presented in Figure 17a. The total path obtained by our work is shown in Figure 17b.



(**a**) (**b**)

Figure 17. (**a**) total trajectory paths. Total path generated by our 2D basic obstacle avoidance algorithm; (**b**) total path generated in our 3D obstacle avoidance algorithm.

The paths are seen from the top view. Figure 18 shows the total trajectory path in our work seen from a different view. We compared our trajectory plan to the basic approach. The UAV speed is considered 8 m/s, and the UAV rotation rate is 30 degree/s. Table 1 shows 13.20% gain in completion time and 10.37% gain in energy by EAOA over the basic approach. It should be noted that the 3D path is 4.04% shorter than the 2D path in the basic approach with reduction of 25% in the total turn angles.

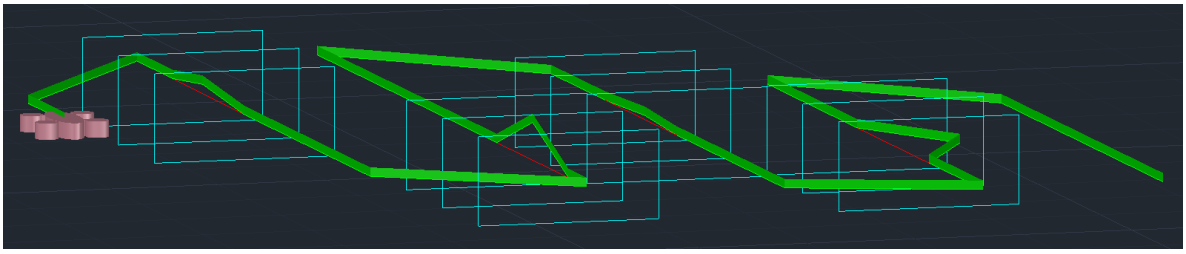
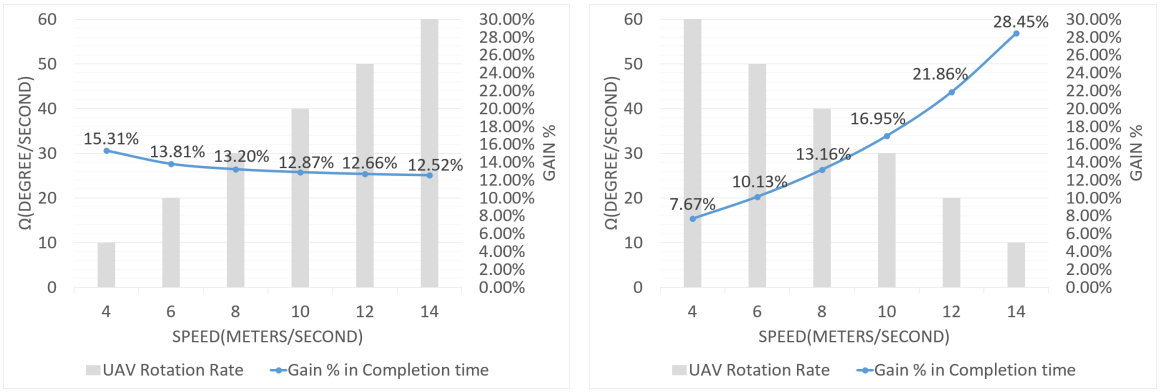


Figure 18. Total trajectory path at the end of the mission.

Table 1. Our basic approach vs. EAOA.

	(a) Basic Approach	(b) Our Work (EAOA)
Route length $\top$ in meters	507.003	486.514
UAV speed υ in m/s	8	8
Turns in degree	1476	1107
Rotation rate ϖ in degree/s	30	30
Completion-time τ in sec	112.575	97.714
Energy consumption E_{total} in KJ	84.551	75.782

In order to understand the behavior of the work, we varied the speed and the rotation rate. The UAV speed varies between 4 and 14 m/s, and the UAV rotation rate varies between 10 and 60 degree/s. Figure 19a shows the gain in completion time while increasing both the speed and rotation rate. Figure 19b shows the gain in completion time in our 3D work vs. a 2D path while increasing the speed and decreasing the rotation rate.



(**a**) (**b**)

Figure 19. Gain in completion time in EAOA compared to the basic approach. (**a**) gain in completion time while increasing both the speed and rotation rate; (**b**) gain in completion time while increasing the speed and decreasing the rotation rate.

Results show that the mission completion time is reduced by an average of 14.46% in 3D path, with energy saving of 10.37% compared to the results obtained in the basic approach. It should be noted that the variation between the speed and the rotation rate influences on the completion time but not on the energy consumption. The gain in energy remains 10.37% due to the fact that λ and γ are the average total energy consumption with standard deviation for the distance, and turn, respectively [35].

- Scenario 2: In this scenario, we adopt the area used in [31] (see Figure 20). The area is 202 m $\times$ 172.2 m. It is divided into a grid of 8 rows $\times$ 10 columns (top view grid). It shows the area of interest with the presence of obstacles.

  Figure 21a shows a Google Maps 3D view of the cell containing the obstacles. It has two trees of height max 4 m. Figure 21b shows the Google Maps street view image for the facing area. We compared our work to [31], in which the drone flies at height of 10 m. We assume that the UAV speed varies between 4 and 14 m/s and UAV rotation rate varies between 10 and 60 degree/s.

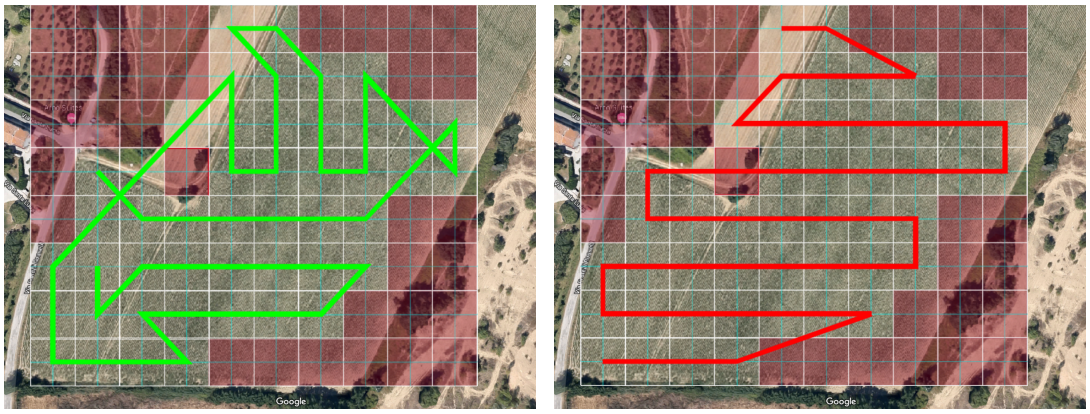
Figure 20. Scenario 2: Area of Interest [31].

(**a**) (**b**)

Figure 21. The Facing area in cell (4,4). (**a**) Google Maps 3D image for the cell containing obstacle; (**b**) Google Maps street view image for the facing area.

The green line in Figure 22a represents the path generated by [31]. The red path in Figure 22b is generated in our work.

Table 2 shows that our path is 3.89% shorter than the path in [31]. It also shows a reduction of 37.77% in the total turn angles. This means better energy saving. Results show that the mission completion time is reduced by an average of 15.22% in our work, with energy saving of 11.30% compared to the results obtained in [31].



(**a**) (**b**)

Figure 22. Trajectory Planning in our work versus the work in [31]. (**a**) trajectory planning by Di Franco et al. [31]; (**b**) trajectory plan by our work.

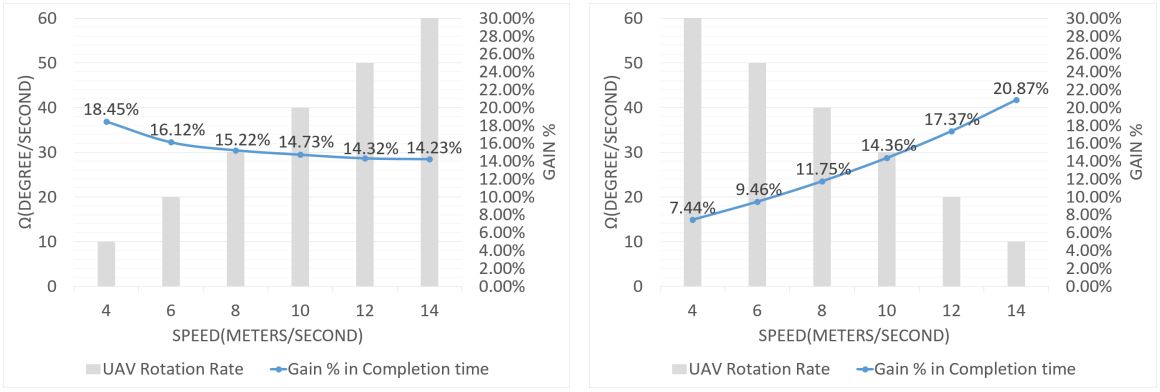
It should be noted that the area is not totally covered by the work in [31]. The cell that contains obstacle (example cell (4,4)) is discarded. However, if the obstacle belongs to a small part of the cell, it is better not to discard the whole cell if there is a possibility to avoid it while maintaining a higher coverage. In our work, by tackling the frontal view, we can have a better understanding of the obstacle location which ensures a better coverage of the area.

Table 2. Results obtained by CPP in Ref. [31] (Figure 22a) vs. EAOA (Figure 22b).

	(a) Ref. [31]	**(b) Our work (EAOA)**
Route length $\top$ in meters	1075.5555	1033.702
UAV speed v in m/s	8	8
Turns in degree	2025	1260
Rotation rate ϖ in degree/s	30	30
Completion-time τ in sec	201.944	171.212
Energy consumption E_{total} in KJ	160.229	142.122

Authors in [31] declare that the cost function presented in the original approach in [10] has not been able to achieve an optimum energy solution. Therefore, they revised the cost-effectiveness and updated the cost-function. They proposed an approach based on an accurate energy model that correctly estimates the energy needed to complete the coverage mission. They showed that their work achieves minimum energy consumption. Our work achieves 11.3% energy gain over the work in [31] which in turn achieves a gain up to 17% over the approach in [10]. Moreover, our work achieves complete area coverage with shorter path over the work in [31], as they avoid the whole obstacle cell which results in more number of turns and less coverage rate.

Figure 23a shows the gain in completion time in our 3D work vs. 2D work while increasing both the speed and rotation rate. Figure 23b shows the gain in completion time while increasing the speed and decreasing the rotation rate. Results show that the mission completion time is reduced by an average of 15.95% in EAOA compared to the results obtained in [31].



(**a**) (**b**)

Figure 23. Gain in completion time. (**a**) gain in completion time in our work compared to [31] while increasing both speed and rotation rate; (**b**) gain in completion time in our work compared to [31] while increasing speed and decreasing rotation rate.

5. Conclusions

We proposed an Energy-aware Grid-based 3D-Obstacle Avoidance during path planning missions. The algorithm is named EAOA and has two phases. It first generates an original path offline using the area top-view grid, then scans the frontal view of the area to detect an obstacle along the drone's path. The algorithm re-plans the path online to find a detour and bypass the obstacle. The results show a gain in completion time and energy saving. We aim in the future works to test the algorithm in an actual flight and study the behavior of the algorithm in a noisy environment and in the presence of dynamic obstacles or pop-up threats. We also intend to boost up the performance of the algorithm by applying more optimization using dynamic programming.

Author Contributions: Conceptualization, A.M.; Investigation, A.G. and A.M.; Supervision, A.G.; Validation, A.M.; Writing—original draft, A.G. and A.M.; Writing—review and editing, A.G. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Acknowledgments: The authors would like to thank the editors and the anonymous reviewers, for their valuable comments which improved the clarity and quality of the paper.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Fotouhi, A.; Qiang, H.; Ding, M.; Hassan, M.; Giordano, L.G.; García-Rodríguez, A.; Yuan, J. Survey on UAV Cellular Communications: Practical Aspects, Standardization Advancements, Regulation, and Security Challenges. *IEEE Commun. Surv. Tutor.* **2019**, *21*, 3417–3442. . [CrossRef]
2. Basilico, N.; Carpin, S. Deploying teams of heterogeneous UAVs in cooperative two-level surveillance missions. In Proceedings of the 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Hamburg, Germany, 28 September–2 October 2015; pp. 610–615. doi:10.1109/IROS.2015.7353435. [CrossRef]
3. Lottes, P.; Khanna, R.; Pfeifer, J.; Siegwart, R.; Stachniss, C. UAV-Based Crop and Weed Classification for Smart Farming. In Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA), Singapore, 29 May–3 June 2017. doi:10.1109/ICRA.2017.7989347. [CrossRef]
4. Nam, L.; Huang, L.; Li, X.; Xu, J. An approach for coverage path planning for UAVs. In Proceedings of the 2016 IEEE 14th International Workshop on Advanced Motion Control (AMC), Auckland, New Zealand, 22–24 April 2016; pp. 411–416. doi:10.1109/AMC.2016.7496385. [CrossRef]

5. Luo, C.; Miao, W.; Ullah, H.; McClean, S.; Parr, G.; Min, G. Unmanned Aerial Vehicles for Disaster Management. In *Geological Disaster Monitoring Based on Sensor Networks*; Durrani, T., Wang, W., Forbes, S., Eds.; Springer: Singapore, 2019; pp. 83–107. doi:10.1007/978-981-13-0992-2_7. [CrossRef]
6. Pham, H.; La, H.; Feil-Seifer, D.; Deans, M. A Distributed Control Framework for a Team of Unmanned Aerial Vehicles for Dynamic Wildfire Tracking. In Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vancouver, BC, Canada, 24–28 September 2017. doi:10.1109/IROS.2017.8206579. [CrossRef]
7. Artemenko, O.; Dominic, O.J.; Andryeyev, O.; Mitschele-Thiel, A. Energy-Aware Trajectory Planning for the Localization of Mobile Devices Using an Unmanned Aerial Vehicle. In Proceedings of the 2016 25th International Conference on Computer Communication and Networks (ICCCN), Waikoloa, HI, USA, 1–4 August 2016. doi:10.1109/ICCCN.2016.7568517. [CrossRef]
8. Xu, A.; Viriyasuthee, C.; Rekleitis, I. Complete Optimal Terrain Coverage using an Unmanned Aerial Vehicle. In Proceedings of the 2011 IEEE International Conference on Robotics and Automation, Shanghai, China, 9–13 May 2011; pp. 2513–2519. doi:10.1109/ICRA.2011.5979707. [CrossRef]
9. Ost, G. Search Path Generation with UAV Applications Using Approximate Convex Decomposition. Master's Thesis, Automatic Control, Linköping University, Linköping, Sweden, 2012.
10. Valente, J.; Sanz, D.; Cerro, J.; Barrientos, A.; de Frutos, M. Near-optimal coverage trajectories for image mosaicing using a mini quad-rotor over irregular-shaped fields. *Precis. Agric.* **2013**, *14*, 115–132. doi:10.1007/s11119-012-9287-0. [CrossRef]
11. Maza, I.; Ollero, A. Multiple UAV cooperative searching operation using polygon area decomposition and efficient coverage algorithms. In *Distributed Autonomous Robotic Systems 6*; Alami, R., Chatila, R., Asama, H., Eds.; Springer: Tokyo, Japan, 2007; Volume 6, pp. 221–230. doi:10.1007/978-4-431-35873-2_22. [CrossRef]
12. Torres Anaya, M.; Pelta, D.; Verdegay, J.; Torres, J. Coverage path planning with unmanned aerial vehicles for 3D terrain reconstruction. *Expert Syst. Appl.* **2016**, *55*, 441–451. doi:10.1016/j.eswa.2016.02.007. [CrossRef]
13. Radmanesh, M.; Kumar, M.; Guentert, P.H.; Sarim, M. Overview of Path-Planning and Obstacle Avoidance Algorithms for UAVs: A Comparative Study. *Unmanned Syst.* **2018**, *6*, 95–118. [CrossRef]
14. Ghaddar, A.; Merei, A. Energy-Aware Grid Based Coverage Path Planning for UAVs. In Proceedings of the SENSORCOMM 2019: The Thirteenth International Conference on Sensor Technologies and Applications, Nice, France, 27–31 October 2019, pp. 34–45. Available online: https://www.researchgate.net/profile/Ahmad_Merei/publication/335570774_Energy-Aware_Grid_Based_Coverage_Path_Planning_for_UAVs/links/5e00e2e14585159aa4959742/Energy-Aware-Grid-Based-Coverage-Path-Planning-for-UAVs.pdf (accessed on 3 February 2020).
15. Stefansson, T. 3D Obstacle Avoidance for Drones Using a Realistic Sensor Setup. Ph.D. Thesis, KTH Royal Institute of Technology, Stockholm, Sweden, 2018.
16. Zhao, L. 3D Obstacle Avoidance for Unmanned Autonomous System (UAS). Ph.D. Thesis, University of Nevada, Las Vegas, NV, USA, 2015.
17. Ferrera, E.; Alcántara, A.; Capitán, J.; Castaño, Á.R.; Marrón, P.J.; Ollero, A. Decentralized 3D Collision Avoidance for Multiple UAVs in Outdoor Environments. *Sensors* **2018**, *18*, 4101. [CrossRef] [PubMed]
18. Lee, J.W.; Walker, B.; Cohen, K. Path Planning of Unmanned Aerial Vehicles in a Dynamic Environment. Infotech@Aerospace, 2011. Available online: https://arc.aiaa.org/doi/abs/10.2514/6.2011-1654 (accessed on 3 Febbruary 2020).
19. Lin, Z.; Castano, L.; Mortimer, E.; Xu, H. Fast 3D Collision Avoidance Algorithm for Fixed Wing UAS. *J. Intell. Robot. Syst.* **2019**. Available online: https://doi.org/10.1007/s10846-019-01037-7 (accessed on 3 Febbruary 2020).
20. Cabreira, M.; Brisolara, L.; Ferreira, P., Jr. Survey on Coverage Path Planning with Unmanned Aerial Vehicles. *Drones* **2019**, *3*, 4. doi:10.3390/drones3010004. [CrossRef]
21. Xiong, R.; Shan, F. DroneTank: Planning UAVs' Flights and Sensors' Data Transmission under Energy Constraints. *Sensors* **2018**, *18*, 2913. doi:10.3390/s18092913. [CrossRef] [PubMed]
22. Liu, W.; Zheng, Z.; Cai, K. Adaptive path planning for unmanned aerial vehicles based on bi-level programming and variable planning time interval. *Chin. J. Aeronaut.* **2013**, *26*, 646–660. doi:10.1016/j.cja.2013.04.041. [CrossRef]

23. Mori, T.; Scherer, S. First results in detecting and avoiding frontal obstacles from a monocular camera for micro unmanned aerial vehicles. In Proceedings of the 2013 IEEE International Conference on Robotics and Automation, Karlsruhe, Germany, 6–10 May 2013; pp. 1750–1757. doi:10.1109/ICRA.2013.6630807. [CrossRef]
24. Al-Kaff, A.; Garcia, F.; Martin, D.; de la Escalera, A.; Armingol, J.M. Obstacle Detection and Avoidance System Based on Monocular Camera and Size Expansion Algorithm for UAVs. *Sensors* **2017**, *17*, 1061. doi:10.3390/s17051061. [CrossRef] [PubMed]
25. Iacono, M.; Sgorbissa, A. Path following and obstacle avoidance for an autonomous UAV using a depth camera. *Robot. Auton. Syst.* **2018**, *106*, 38–46. doi:10.1016/j.robot.2018.04.005. [CrossRef]
26. Esposito, J.F. Real-Time Obstacle and Collision Avoidance System for Fixed Wing Unmanned Aerial Systems. Ph.D. Thesis, University of Kansas, Lawrence, KS, USA, 2013.
27. Liu, C.; Wang, H.; Yao, P. UAV Autonomous Collision Avoidance Method Based on Three-dimensional Dynamic Collision Region Model and Interfered Fluid Dynamical System. *DEStech Trans. Eng. Technol. Res.* **2017**. doi:10.12783/dtetr/icca2016/6007. [CrossRef]
28. Choi, M.; Rubenecia, A.; Shon, T.; Choi, H.H. Velocity Obstacle Based 3D Collision Avoidance Scheme for Low-Cost Micro UAVs. *Sustainability* **2017**, *9*, 1174. doi:10.3390/su9071174. [CrossRef]
29. Parappat, P.; Kumar, A.; Mittal, R.; Khan, S. Obstacle Avoidance by Unmanned Aerial Vehicles Using Image Recognition Techniques. In Proceedings of the International Conference on Circuits, Systems, Communications and Computers, 2014; Volume 1, pp. 378–381. Available online: https://www.researchgate.net/profile/Suhel_Khan2/publication/264238395_Obstacle_avoidance_by_unmanned_aerial_vehicles_using_image_recognition_techniques/links/545bedd70cf2f1dbcbcb085b/Obstacle-avoidance-by-unmanned-aerial-vehicles-using-image-recognition-techniques.pdf (accessed on 3 February 2020).
30. Cruz, G.; Encarnacao, P. Obstacle Avoidance for Unmanned Aerial Vehicles. *J. Intell. Robot. Syst.* **2012**, *65*, 203–217. doi:10.1007/s10846-011-9587-z. [CrossRef]
31. Cabreira, T.M.; Ferreira, P.R.; Di Franco, C.; Buttazzo, G.C. Grid-Based Coverage Path Planning with Minimum Energy over Irregular-Shaped Areas with UAVS. In Proceedings of the 2019 International Conference on Unmanned Aircraft Systems (ICUAS), Atlanta, GA, USA, 11–14 June 2019. doi:10.1109/ICUAS.2019.8797937. [CrossRef]
32. Gramajo, G.; Shankar, P. An Efficient Energy Constraint Based UAV Path Planning for Search and Coverage. *Int. J. Aerosp. Eng.* **2017**, *2017*, 8085623. doi:https://doi.org/8085623. [CrossRef]
33. Li, D.; Wang, X.; Sun, T. Energy-optimal coverage path planning on topographic map for environment survey with unmanned aerial vehicles. *Electron. Lett.* **2016**, *52*, 699–701. doi:10.1049/el.2015.4551. [CrossRef]
34. Recchiuto, C.; Nattero, C.; Sgorbissa, A.; Zaccaria, R. Coverage Algorithms for Search and Rescue with UAV Drones. In Proceedings of the Workshop of the XIII AIIA Symposium on Artificial Intelligence, Pisa, Italy, 10–12 December 2014.
35. Modares, J.; Ghanei, F.; Mastronarde, N.; Dantu, K. UB-ANC planner: Energy efficient coverage path planning with multiple drones. In Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA), Singapore, 29 May–3 June 2017; pp. 6182–6189. doi:10.1109/ICRA.2017.7989732. [CrossRef]