International Journal of
Geo-Information

Article

Real-Time Efficient Exploration in Unknown Dynamic Environments Using MAVs

Haytham Mohamed [1,*], Adel Moussa [1,2], Mohamed Elhabiby [3] and Naser El-Sheimy [1]

1 Department of Geomatics Engineering, University of Calgary, Calgary, AB T2N 1N4, Canada; amelsaye@ucalgary.ca (A.M.); elsheimy@ucalgary.ca (N.E.-S.)
2 Department of Electrical and Computer Engineering, Port-Said University, Port-Said 42523, Egypt
3 Public Works Department, Ain Shams University, Cairo 11566, Egypt; mmelhabi@ucalgary.ca
* Correspondence: haytham.abdalla@ucalgary.ca; Tel.: +2-01-101-199-333

Received: 21 September 2018; Accepted: 14 November 2018; Published: 18 November 2018

Abstract: Micro aerial vehicles (MAVs) have been acknowledged as an influential technology for indoor search and rescue operations. The time constraint is a crucial factor in most search and rescue operations. The employed MAVs in indoor environments are characterized by short endurance flight time and limited payload weights. Hence, adding more batteries to extend the flight time is practically not feasible. Typically, most of the indoor missions' environments might not be accessed and remain unknown. Working in such environments requires effective exploration and information gathering to save time and maximize the coverage area. Furthermore, due to the dynamism of such environments, choosing the least risky trajectory is an important task. This paper proposes a real-time active exploration technique which is capable of efficiently generating paths that minimize the vehicle's risk and maximize the coverage area. Furthermore, it accomplishes real-time monitoring of sudden changes in the estimated map, due to the dynamic objects, by reevaluating at real-time the destination and trajectory to minimize the risk on the chosen path and simultaneously preserving the maximization of the coverage area. Ultimately, recording the implemented trajectory of the vehicle also assists in time-saving as the vehicle depends on this trajectory during the exit process. The performance of the technique is studied under static and dynamic environments and is also compared with different algorithms.

Keywords: scan matching; SLAM; laser rangefinder; point registration; least squares; line tracking; PCA; ICP; MAV; reference key frame

1. Introduction

Exploration of unknown environments using robots, such as unmanned ground vehicles (UGVs) and unmanned aerial vehicles (UAVs), is a problem of acquiring local knowledge to proceed along investigating and discovering novel places in the exploring environment [1,2]. Exploration is divided into two types, namely directed and undirected exploration [3]. Directed exploration considers the gained knowledge or previous history to facilitate future exploration. On the other hand, the undirected exploration algorithms do not consider the gained knowledge along the trajectory. Completeness and effectiveness are two key tasks for typical exploration [1]. Thus, the robot requires discovering as much of the environment as possible to perform completeness, while simultaneously achieves this completeness with minimum efforts, for instance time and power consumption. Various applications such as search and rescue operations [4] and reconnaissance [5] require exploration to accomplish their missions.

Micro aerial vehicles (MAVs) are categorized as miniature UAVs. Their small size makes them superior to their peers in discovering narrow places. On the other hand, they are also characterized by short flight time due to the payload restriction. MAVs have been acknowledged as an influential technology for indoor search and rescue operations, providing real-time mapping of the environment, locating victims, and determining the hard-to-access areas after a natural disaster. The time constraint is also a crucial factor in most search and rescue operations. Using larger batteries to extend the flight time is constrained by the payload limitation. Such limitation might affect the exploration of the indoor missions' environments and limit its accessibility. Consequently, working in such environments requires effective exploration by generating efficient paths to maximize the coverage area in short time [1,6]. The problem of the path planning (i.e., trajectory generation) is a problem of finding a proper trajectory to a definite destination, according to the limitations of the environment and the vehicle as well [7,8]. Thus, this problem is considered to be an optimization problem. The vehicle has to generate a real-time path from the current position to the goal position with particular optimization criteria such as shortest path [9]. Furthermore, the generated path must be free of the collision [9].

Many algorithms and methods, such as Dijkstra [10,11], A-star (A*) [12,13], artificial potential field (APF) [14], rapidly-exploring random tree (RRT) [15,16], and bidirectional RRT [17,18], have been developed for path planning problem which focuses on finding the optimum route to the destination. Such techniques, however, do not consider efficiency regarding the exploration of the area, such as maximizing the visited area. They have different optimization criteria for the trajectory generation, such as lowest executed time, shortest path, least power consumption and lowest time complexity.

Dijkstra's algorithm is a shortest path algorithm based on a weighted graph and it aims to find the least cost path between two nodes in a graph, namely source and goal nodes. The A* algorithm is an extension of the Dijkstra's algorithm. However, the A* algorithm is based on a heuristic approach. Thus, the A* algorithm differs from the Dijkstra's algorithm in the description of the node cost indices. The APF method depends on creating a potential field that consists of attraction and repulsion forces to/from the goal and obstacles, respectively. Hence, the vehicle moves under the influence of both forces from the source to the position of the goal. RRT addresses the path planning problem based on a sampling-based algorithm. The RRT algorithm also finds a path between the start and the destination nodes, but it may not be the optimal path due to the random behavior. The bidirectional RRT is proposed to enhance the time consumption of the RRT algorithm by generating two trees at both start and destination nodes instead of one tree at the start node only.

Due to dynamism of such environments, several algorithms were developed to regenerate at real-time a new path to avoid the moving objects, such as dynamic artificial potential field method (DAPF) [19], extended RRT algorithm (ERRT) [20], Dynamic RRT algorithm (DRRT) [21], reconfigurable random forest algorithm (RRF) [22], and multipartite RRT (MP-RRT) [23]. The DAPF method is divided into predictor and planner parts. The moving obstacles are indexed, and its motion is predicted using the predictor part. The attractive and repulsive forces are defined by the planner according to the vehicle's position, goal, and finally the position of the static and dynamic obstacles. The ERRT algorithm maintains all the intermediate nodes after successful path generation in a waypoint cache. After that, the algorithm uses these nodes to bias the random selection of the nodes during a new planning iteration. The DRRT algorithm is similar to ERRT algorithm in which the algorithm preserves the previous successful path. However, the DRRT algorithm uses the previous branches that are connected to the source node and not the previous intermediate nodes. The RRF algorithm constructs a forest of disconnected RRTs in advance. During the implementation, the algorithm removes invalid nodes due to a sudden change in the environment. Then, the algorithm connects to one of the previously constructed RRT in the forest until reaches the goal. The MP-RRT algorithm combines the strengths of ERRT and DRRT algorithms. The MP-RRT algorithm biases the sampling distribution across previously successful nodes and selects previously used branches.

Working in unknown dynamic environments requires effective exploration and information gathering. Therefore, developing an active exploration technique which is capable of accomplishing real-time monitoring of sudden changes in the estimated map by reevaluating the destination and trajectory is of immense importance. Besides, the exploration technique should be efficient by maximizing the visited area to increase the effectiveness of the search and rescue operations. Ultimately, recording the implemented trajectory of the vehicle also assists in time-saving as the vehicle depends on this trajectory during the exit process.

This paper is organized as follows: Section 2 illustrates an overview structure of the proposed method. The representation of the static and dynamic map objects is explained in Section 3. Section 4 demonstrates optimized real-time trajectory generation. Dynamic objects monitoring is illustrated in Section 5. The experimental results are presented in Section 6. Finally, the conclusion is provided in Section 7.

2. Overview Structure of the Proposed Method

Figure 1 shows the overview structure of the proposed method. The reference key frame (RKF) algorithm is used to construct the map of the unknown environment and localize the vehicle inside this map [24]. A filtering approach is proposed by Mohamed et al. [25] that can distinguish between static and dynamic objects in the exploring environment. The filtering approach represents the static and dynamic objects of the environment using two separate grids (i.e., map); the first grid represents the static objects, while the second grid represents the dynamic objects. However, the static and dynamic information should be amended to facilitate the computing of the cost function map to generate a proper trajectory according to the current situation and the exploring environment. Thus, the distance transform (DT) method is utilized to represent the static objects [26], while the Gaussian kernel is used to represent the dynamic objects. The outcomes of the DT and Gaussian kernel are fed to the adjusted A* algorithm as a heuristic value matrix, as described below. The adjusted A* algorithm is responsible for generating the required path. However, this generated path can be suspended either because the dynamic objects monitoring process observes intruders within a predetermined area or the required destination is reached. Thus, the algorithm generates another path according to the new situation.

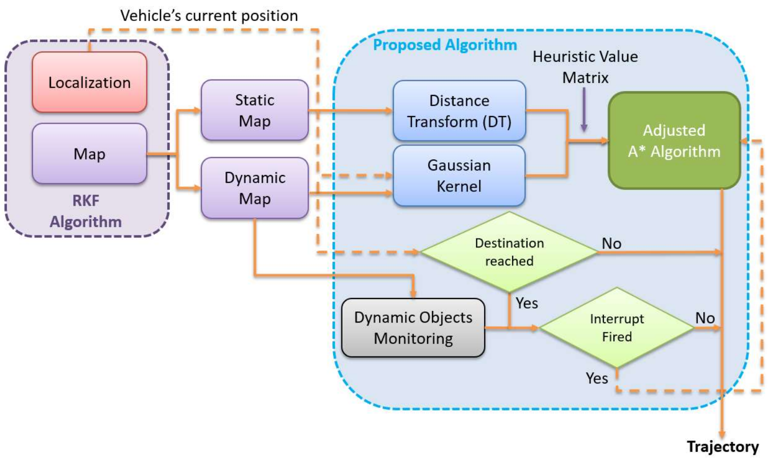


Figure 1. Overview of the system structure.

3. Representation of Static and Dynamic Map Objects

3.1. Static Representation Using Distance Transform (DT) Method

The DT is a method applied to transform the binary image to a distance image. It is a function $(D_P(.))$ whereby each image pixel (p) is assigned a minimum non-negative value $(D_P(p))$ corresponding to distance from (p) to the nearest obstacle (q) in the binary image (G).

$$D_P(p) = \min_{p \in G}(d(p,q)), \tag{1}$$

Figure 2 illustrates the designation of each pixel of the image with a distance to the nearest obstacle pixel using distance transform method. DT method has a variety of computation methods; the implemented method, in this research work, is the Euclidean distance which represents the straight-line distance between two pixels.

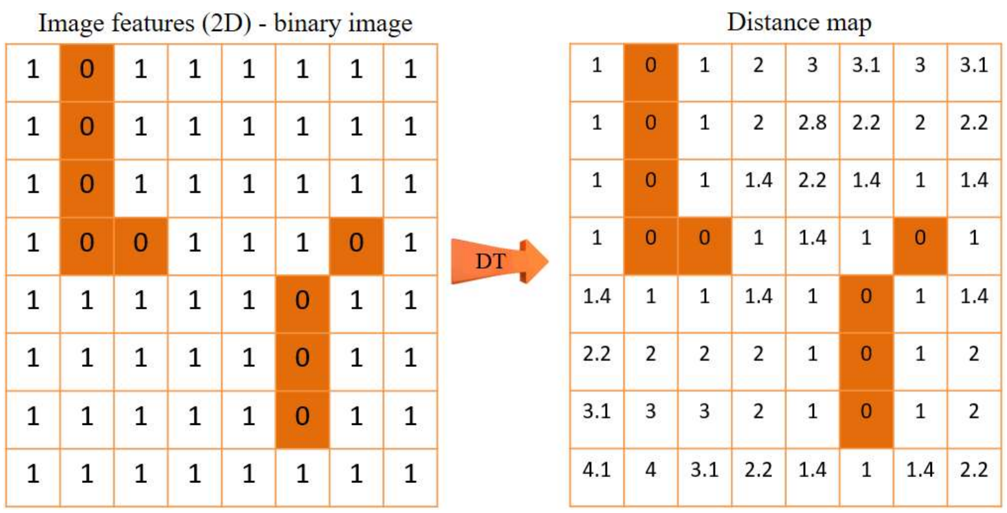


Image features (2D) - binary image
DT
Distance map

Image features (2D) - binary image

1	0	1	1	1	1	1	1
1	0	1	1	1	1	1	1
1	0	1	1	1	1	1	1
1	0	0	1	1	1	0	1
1	1	1	1	1	0	1	1
1	1	1	1	1	0	1	1
1	1	1	1	1	0	1	1
1	1	1	1	1	1	1	1

Distance map

1	0	1	2	3	3.1	3	3.1
1	0	1	2	2.8	2.2	2	2.2
1	0	1	1.4	2.2	1.4	1	1.4
1	0	0	1	1.4	1	0	1
1.4	1	1	1.4	1	0	1	1.4
2.2	2	2	2	1	0	1	2
3.1	3	3	2	1	0	1	2
4.1	4	3.1	2.2	1.4	1	1.4	2.2

Figure 2. Distance transform (DT) method using Euclidean distance.

The result of this implementation (i.e., distance map) is inverted to acquire two objectives: The first objective is to use the decreasing intensity levels from the static obstacles and its boundaries to create a cost representation. The prohibitive costs are allocated at the static obstacles and its boundaries, while the low costs are allocated at the intermediate area between the obstacles. Due to the prohibitive costs of the obstacles and its neighbors, the thickness of the static obstacles seems to be extended in such a way to prepare a safe area as far as possible from the obstacles, as shown in the examples below. Hence, this safe area creates a candidate path that intermediates the obstacles to avoid collisions, as shown hereafter. The second objective is to prevent the vehicle to pass through narrow gaps. Figures 3 and 4 show an example of a binary image of an indoor environment with two gaps, sufficient and insufficient gaps, and how the DT can be used to detect the sufficient space for a vehicle to pass. The green ellipse shows the sufficient gap, while the red circle shows the insufficient gap. The position of the vehicle is represented by the blue triangle.

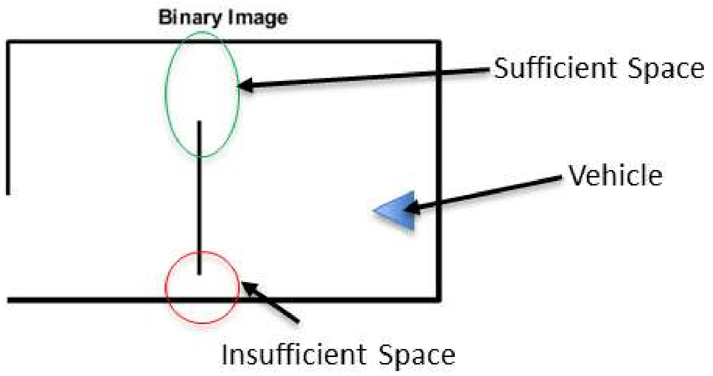


Figure 3. Binary image for indoor environment.

Figure 4 represents the corresponding distance transform of the image in Figure 3 where the darker areas represent the close areas to the obstacles and the bright areas represent the far areas from the obstacles. This cost representation helps in preparing a safe path for the vehicle by accepting the areas above a specific safety tolerance to be among the candidate areas for the planned trajectory.

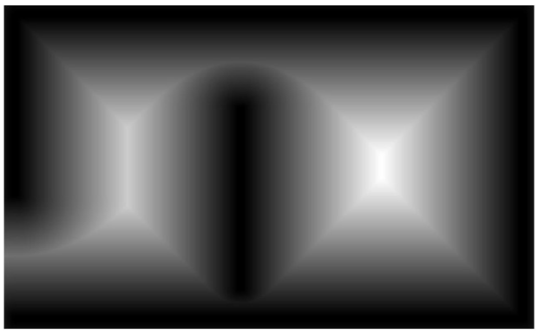


Figure 4. The result of the DT method using Euclidean distance.

Figures 5 and 6 illustrate the binary images of parts of the gathered datasets and the corresponding DT results. It is obvious from the presented figures that the DT method achieves the objectives above. Part of the constructed map (Map I), using reference key frame algorithm [24], is presented as a binary image, as shown in Figure 5a with its DT depicted in Figure 5b. The walls are successfully extended, and a candidate safe area is prepared which is represented by brightest color, between the walls to be as far as possible from all current static obstacles, as shown in Figure 5b.

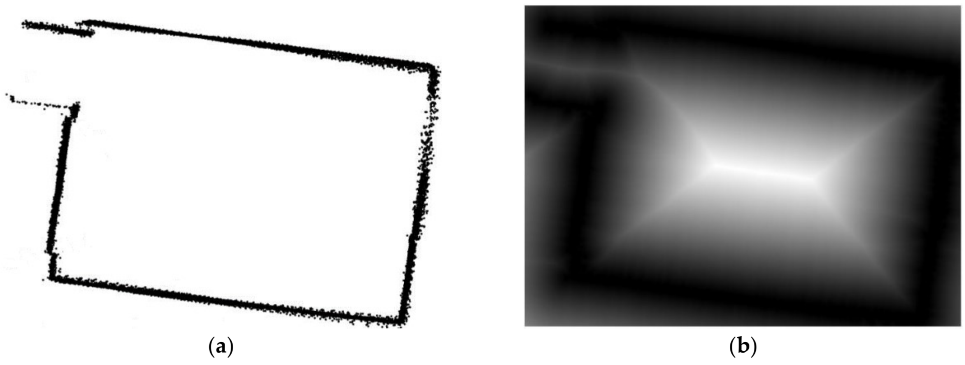


Figure 5. Part of the constructed map I: (**a**) binary representation; and (**b**) the corresponding result using the DT method.

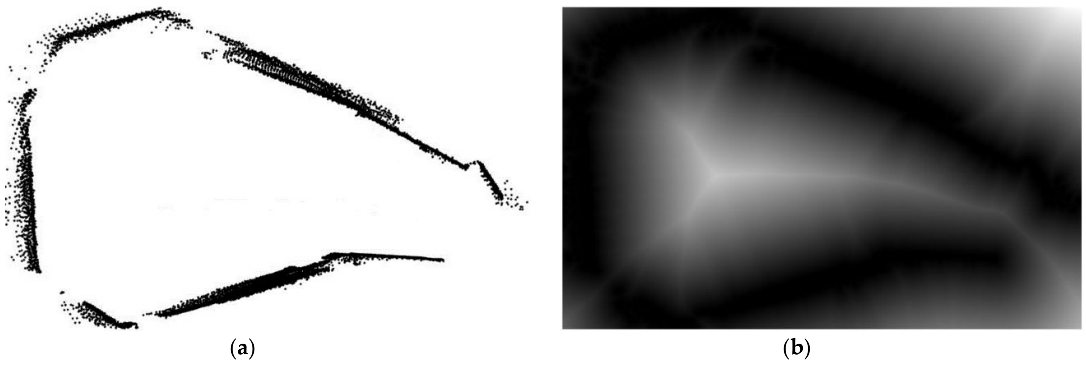


Figure 6. Part of the constructed map II: (**a**) binary representation; and (**b**) the corresponding result using the DT method.

Part of another constructed map (Map II) is presented as a binary image, as shown in Figure 6a. All the narrow gaps between the walls can be easily excluded (threshold) leaving only one feasible gap for the vehicle to pass through, as shown in Figure 6b. The safe path is also presented by the white color in the middle of all static objects.

3.2. Dynamic Representation Using Gaussian Kernel

The dynamic grid includes information about the moving objects inside the exploring environment. The dynamic object and its moving history are expressed as occupied cells with different probability values. The highest probability is assigned to the cell of the most recent occupation, while the probability values of the rest of the cells are decreasing as time proceeds. Keeping the historical information of the dynamic objects over time is conducted to give a more robust representation of the movement of the dynamic objects, especially when dealing with noisy measurements (i.e., low-cost laser scanner rangefinder). Each occupied cell is represented using the Gaussian kernel, where the mean value of the Gaussian is the center of the cell, while the standard deviation is selected to cover two or more of the neighboring cells to build safety margin around the dynamic objects. The selection of the standard deviation depends on the density of the dynamic objects inside the explored environment. The standard deviation increases with the highly populated environments and vice versa to reflect the extent of the occupation uncertainty under the different situations.

Figure 7a demonstrates the representation of straight-line movement of a dynamic object using successive Gaussian kernels. The highest peak presents the cell of current occupation of the dynamic object, while the descending behavior peaks show the history of the object movement. Maintaining the history information in the dynamic grid is chosen according to the application by adjusting the probability computation of the cells. Figure 7b demonstrates the same straight-line movement of the dynamic object as in Figure 7a. However, it shows a short period of the history information in the dynamic grid as the Gaussian kernels occupy only three cells.

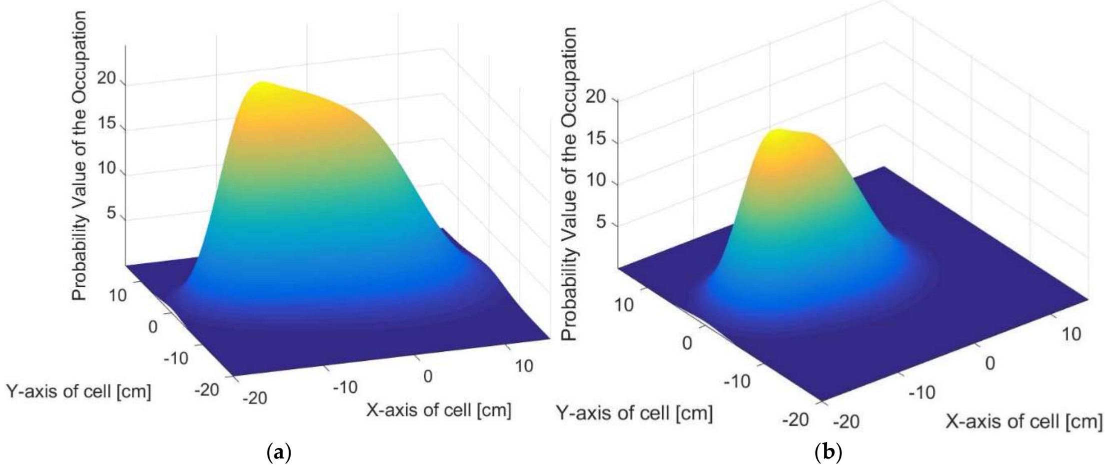


Figure 7. Representation of the dynamic movement using successive Gaussian kernels: (**a**) a long period of the history information; and (**b**) a short period of the history information.

Moreover, the Gaussian kernels are also used to represent the previously executed trajectory of the vehicle. Thus, the previous trajectory is treated as dynamic objects. Contrary to dynamic object representation, each vehicle's position is represented independently from the previous positions by a separate Gaussian kernel. There is no decreasing of the probability values as time proceeds because the entire trajectory is utilized as an indication of the visited locations, as shown in Figure 8. Therefore, the algorithm will not revisit the same place twice because this visit will raise the cost which is minimized by the algorithm.

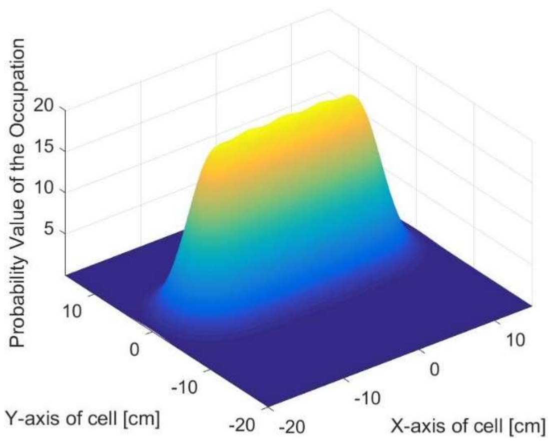


Figure 8. A part of the vehicle's trajectory representation using Gaussian kernel.

Although the Gaussian kernels are utilized to represent the previous trajectory for maximizing the visited area during the exploration, these probability values of the occupations are reversed to be very low value, negative values, to attract the vehicle to use it during the exit process for time saving.

4. Optimized Real-Time Trajectory Generation

The MAV must be able to choose a proper destination according to the surrounding environment and the constraints of the vehicle such as its size. Using the partially constructed map, the vehicle detects empty exits in the current environment, such as open doors, windows, or even free spaces between obstacles inside a room, thereby enabling the vehicle to estimate the position of the exits and choose them as its next candidate destinations. Furthermore, both static and dynamic representations are used as risk cost functions. The vehicle will then decide the most efficient trajectory from the candidate destinations, while maximizing the visited area of the surrounding environment with the least cost.

Adjusted A Algorithm*

A* algorithm is considered to be the most efficient heuristic path planning approach [12,27]. The goal of this algorithm is to find the shortest path from a known node's position to a known node's destination passing through a group of nodes. The search space is divided into two lists in this algorithm, namely open and closed lists. The open list is the list of nodes that needs to be investigated, while the closed list is the list of nodes that have already been investigated. The A* algorithm uses some parameters, called node data, to achieve the shortest path, such as H, G, and F values, and parent nodes. The H value, which is called Heuristic value, represents a cost distance from each node in the search space to the destination node and this value is registered in each node. The G value represents the cost of the movement from one node to another and it is also registered in each node. The F value is the summation of both H and G values. The parent node represents the node that reaches the current node.

$$F_i(t) = G_i(t) + H_i(t), \tag{2}$$

where

G: movement's cost value matrix from one node to another
H: heuristic value matrix
F: summation of both G and H values
i: the *ith* node in the matrix
t: epoch

For occupancy grid map representation, each cell of the grid is represented as a node. The A* algorithm begins by computing the *H* value for every node in the search space. From the start node, the algorithm computes the *G* value for all nodes of the first neighbor level. After that, the *F* value is calculated for each node, as shown in Figure 9. The *H* values are represented by the red numbers that shows the number of steps required to reach the destination node. The *G* values are represented by the black numbers, which are in integer form after using Euclidean distance computation. The *F* values are represented by the green numbers. The blue numbers are the number of the nodes. The black squares are obstacles on the map. The open list contains the nodes under investigation such as nodes 8, 9, 10, 15, 17, 22, 23, and 24. The closed list contains node 16. Node 16 is the parent for the next selected node.

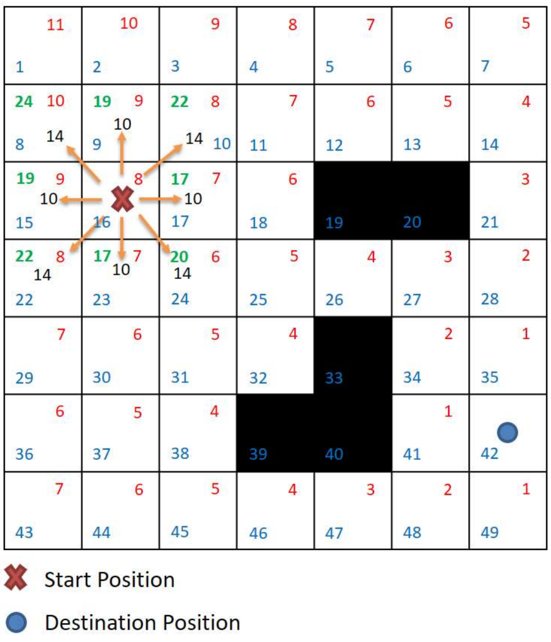


Figure 9. A* algorithm.

In the next step, the A* algorithm chooses the current candidate node with the lowest F value. Hence, nodes 17 and 23 are selected with no priority for the node to start with. The same procedure is then repeated for both nodes (i.e., nodes 17 and 23). After several steps, the closed list is extended by the investigated nodes until reaching the destination node. In addition, the parent nodes are stacked with the chosen nodes that express the shortest path.

However, A* algorithm uses the distance to the destination as a heuristic to focus on finding the shortest path. For fulfilling the desired requirements such as avoiding obstacles, maximizing the visited areas, and minimizing the risk on the generated path, an adjusted version of A* algorithm is introduced.

In this version, as mentioned above, the DT method is used to represent the static obstacles, while the Gaussian kernel is used to represent the dynamic objects and the trajectory of the vehicle. These representations are summed together and fed into the A* algorithm as cost functions with the shortest path. Consequently, the *H* value (Heuristic value) is designated to the merged cost functions instead of the Euclidean distance value from each node to the destination node (goal). Hence, the *F* values of the cells near the static obstacles, the cells of the dynamic objects, and the implemented trajectory are magnified due to the excessive cost of the *H* values. Therefore, the generated path avoids all hazardous areas, such as static obstacles and dynamic objects, to minimize the risk on the generated path and implicitly avoid collisions. Furthermore, the performed trajectory is bypassed to maximize the visited areas. Ultimately, dealing with the cells of the occupancy grid generates a trajectory characterized by jaggedness and is often inconvenient. Thus, the generated path is smoothed, to be proper for the UAV, using the moving average filter [28].

$$S_{op}(i) = \frac{1}{n} \sum_{j=\frac{-(n-1)}{2}}^{\frac{(n-1)}{2}} S_{ip}(i+j), \tag{3}$$

where

S_{op}: the output signal
S_{ip}: the input signal
n: number of points in the average
i: index of the signal

Once the A* algorithm generates a path, the algorithm is turned to be idle until interruption takes place. This interruption occurs either when the vehicle reaches the current destination, or the generated path is abruptly changed, for instance dynamic object blocks the generated path. Whenever the MAV penetrates the circle of acceptance surrounding the destination point, the algorithm starts to generate a new path to a new destination.

After finishing the required mission, the Gaussian kernels of the implemented trajectory, which are used to maximize the visited area during the exploration, are utilized to create a return path by inverting their cost values to attract the vehicle to use it during the exit process.

5. Dynamic Objects Monitoring

Due to the limitations of the indoor environment, the moving objects cannot typically move with high speed, but they can perform the steep maneuver. Hence, the motion prediction of the moving objects is not convenient for every moving object, for instance people. The prediction of their movements is an arduous process due to their high maneuverability. Moreover, extra computations have to be executed in real-time for all detected moving objects, which increases the computational burden of the algorithm. Instead of motion prediction, the proposed method creates a safety margin, for instance a circle with a pre-assigned radius, according to the surrounding environment, around the vehicle's body. Once the moving object penetrates the safety margin, the algorithm stops executing the generated path and begins to regenerate a new path according to the current situation. Figure 10 shows a sanctuary that surrounds the vehicle from all directions to perceive the intruders and after that launches the interrupt. The blue dotted line represents the border of the safety margin, while the brownish circle is the safety margin. The radius of the circle is denoted as (r).

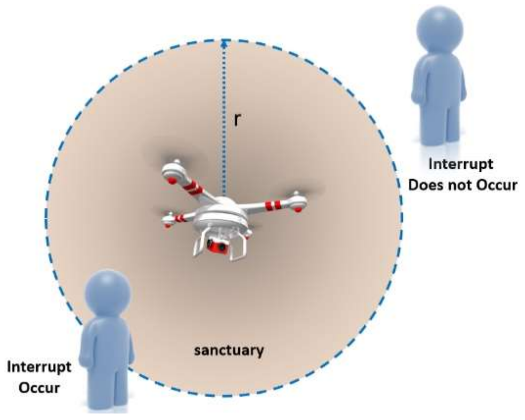


Figure 10. Monitoring of the dynamic objects.

6. Experimental Results

To evaluate the efficiency of the proposed path generation method, the proposed method is compared with A*, APF, RRT, and bidirectional RRT using simulated and real datasets in static and dynamic environments. Table 1 shows the environment status and type for each experimental dataset. All the comparisons have been performed using the same computing platform (MATLAB) on a Dell Inspiron 7000 series, Intel core i7-7500U 2.7 GHz, 16 GB RAM, 64-bit operating system.

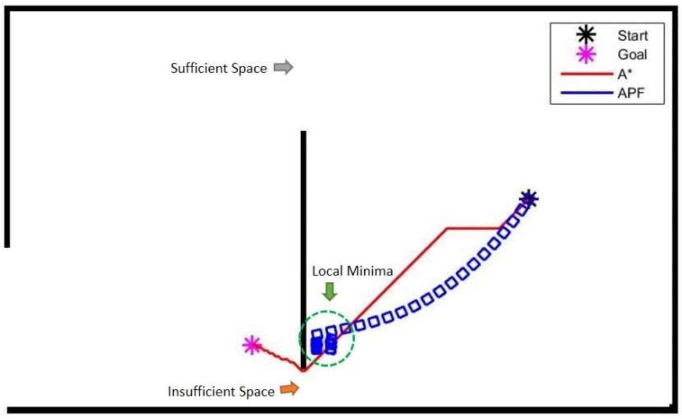
Table 1. The environment status and type for each experimental dataset.

Dataset Name	Environment Type	Environment Status
Dataset I	environment I/simulation	static
Dataset II	environment II/real	static
Dataset III		dynamic
Dataset IV	environment III/real	static
Dataset V		dynamic

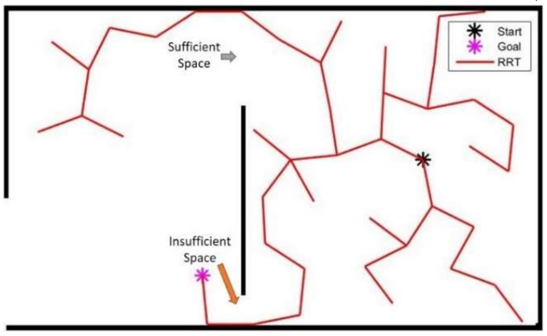
The adjusted A* algorithm generates a path within a square window, the dimension of the window is equal to the maximum detection range of the utilized laser scanner rangefinder, 6 m. Since the size of individual grid cell is 5 cm, and each grid cell represents a node. Therefore, the search domain is created from 240 $\times$ 240 nodes (i.e., grid cells).

For Dataset I, Figure 11 illustrates the simulation results of generating paths using A* algorithm, APF method, RRT algorithm and bidirectional RRT algorithm, respectively. The black and magenta asterisks represent the positions of the source and goal points, respectively. In Figure 11a, the generated path using A* algorithm is represented by the red line, while the generated path using APF method is represented by the blue squares. The generated path of the A* algorithm passes through the insufficient space, as shown by the orange arrow, due to shortest path behavior of the algorithm. Furthermore, the generated path is chosen close to the static object, which exposes risk to the vehicle. The APF method fails to provide a solution because the method is trapped in local minima, as shown by the green circle. The processing time of the A* algorithm is 466 ms. The processing time of the APF method is not available because the method is trapped. Figure 11b,c shows the generated paths using different iterations of RRT algorithm. The red and blue lines represent the generated paths in different iterations. It is clear that each iteration of the algorithm generates a new path due to the random nature of the algorithm. The generated path goes across the insufficient space, as shown in Figure 11b, while the generated path goes across the sufficient space, as shown in Figure 11c. Thus, the algorithm does not guarantee that the generated path passes through the sufficient space to reach the goal point. The processing times of the iterations are 3573 and 4205 ms. Figure 11d,e demonstrates the generated paths using different iterations of bidirectional RRT algorithm. The red and blue lines represent the two generated trees from each seed (i.e., source and goal points). The green line represents the connection between the two trees. The algorithm also does not guarantee that the generated path passes through the sufficient space. The processing times of the iterations are 3312 and 2179 ms.

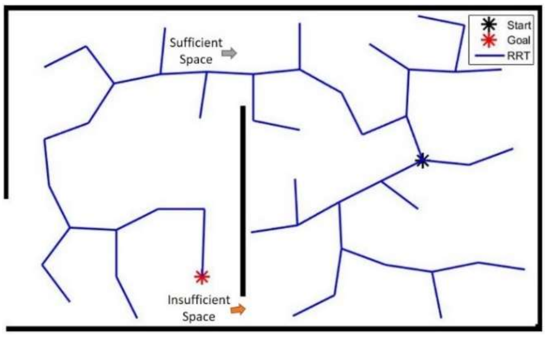
Figure 12 demonstrates the cost map of the static representation using the proposed method which is computed from the static objects only using the DT method. The static obstacles are represented by the brightest color which represents the highest cost, whereas the cost gradually decreases for the extension of the obstacles. On the other hand, the lowest cost is represented by the darkest color which represents the least available risky areas. This cost map is fed into the adjusted A* algorithm as a heuristic value. Figure 13 shows the generated trajectory within the extension of the static objects. The source point is represented by the blue asterisk, while the goal point is represented by the red asterisk. The generated path is represented by the green line. Obviously, the insufficient space is completely blocked by performing the obstacle extension process. Thus, the insufficient space will not be a candidate path for the vehicle. Moreover, the generated path adheres to the darkest color/least cost areas in the map to efficiently decrease the flight risk of the vehicle.



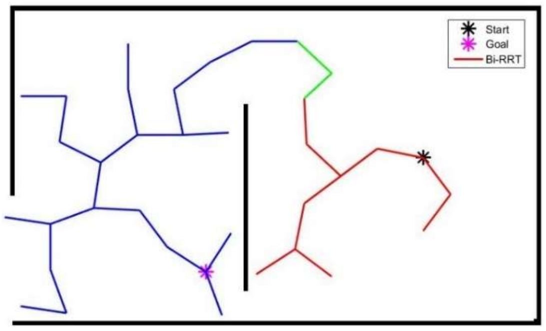
(**a**)



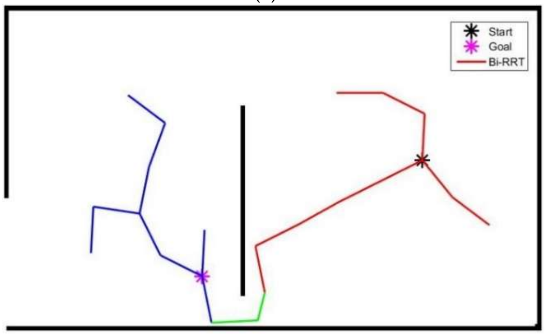
(**b**)



(**c**)



(**d**)



(**e**)

Figure 11. Generated path, for Dataset I, using: (**a**) A* algorithm and APF method; (**b**) first iteration of the RRT algorithm; (**c**) second iteration of the RRT algorithm; (**d**) first iteration of the bidirectional RRT algorithm; and (**e**) second iteration of the bidirectional RRT algorithm.

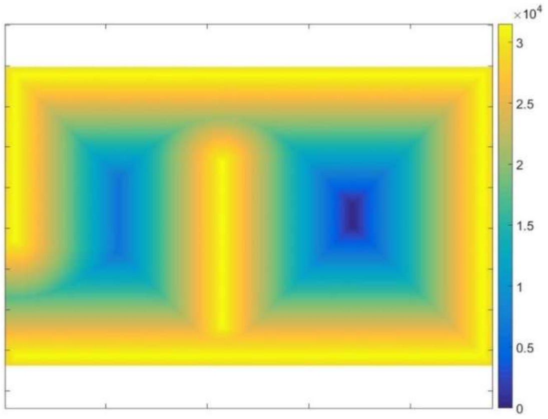


Figure 12. The cost map of the static representation for Dataset I.

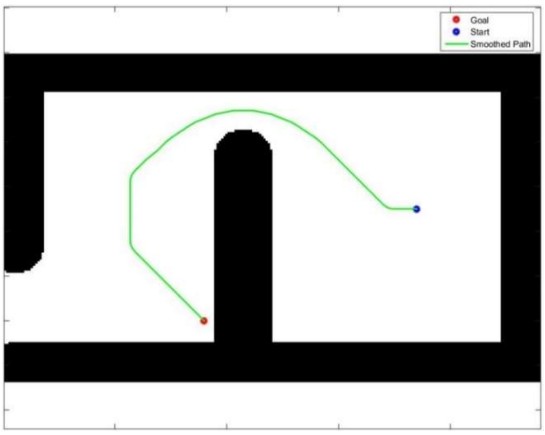


Figure 13. The generated path, for Dataset I, within the extension of the static obstacles.

Figure 14 demonstrates the 2D representation of the generated path. The blue asterisk represents the position of the starting point, while the red asterisk represents the position of the goal point. The smoothed path is represented by the green line. Since the only factor affecting the path generation is the static environment, the generated path is attempted to be in the middle of the obstacles to minimize the risk of the generated path. The processing time of the proposed algorithm is 550 ms. The processing time has also been tested on a light weight board (UDOO X86) and the computed time is 610 ms. This embedded system consists of a single board computer which is based on Quad Core 64-bit new-generation X86 processors made by Intel$^{®®}$. This board meets the deployment requirements on small size UAVs [29].

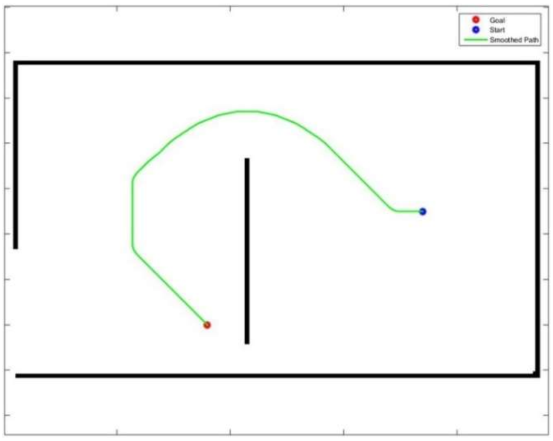


Figure 14. 2D representation of the generated path, for Dataset I, using the adjusted A* algorithm in a static environment.

Figure 15 shows the generated path before and after smoothing process and it also represents a zoomed area of part of Figure 14. The magenta line represents the generated path before the smoothing process. The green line represents the smoothing path. The generated path before the smoothing process is crimped. This behavior is because the constructed map is built using an occupancy grid. Subsequently, the generated path transfers from one cell to another without prejudice to the optimization goal (i.e., minimize the risk). The crimp behavior affects the smoothness of the flight which includes, but is not limited to, endurance time and battery consumption.

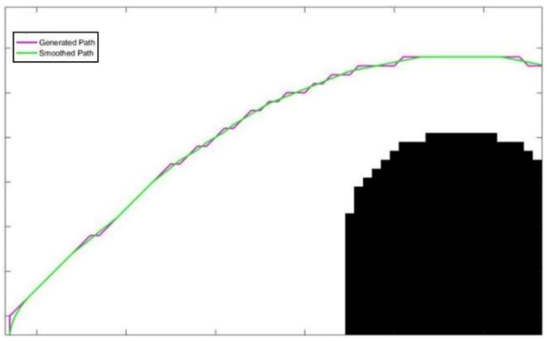


Figure 15. Representation of the generated and smoothed paths.

Figure 16 depicts the 3D representation of the generated path. The blue asterisk represents the starting position, while the red asterisk represents the goal position. The smoothed path is represented by the green line. As described above, the static obstacles are represented using the DT method. Thus, the maximum height is the maximum cost, and it represents the obstacle grid cells, which decreases in magnitude to represent the extension of the obstacles until reaches approximately the middle distance in-between the obstacles. It is clear that the generated trajectory is created inside the valley in-between the extended obstacles to prepare a safe path for the MAV.

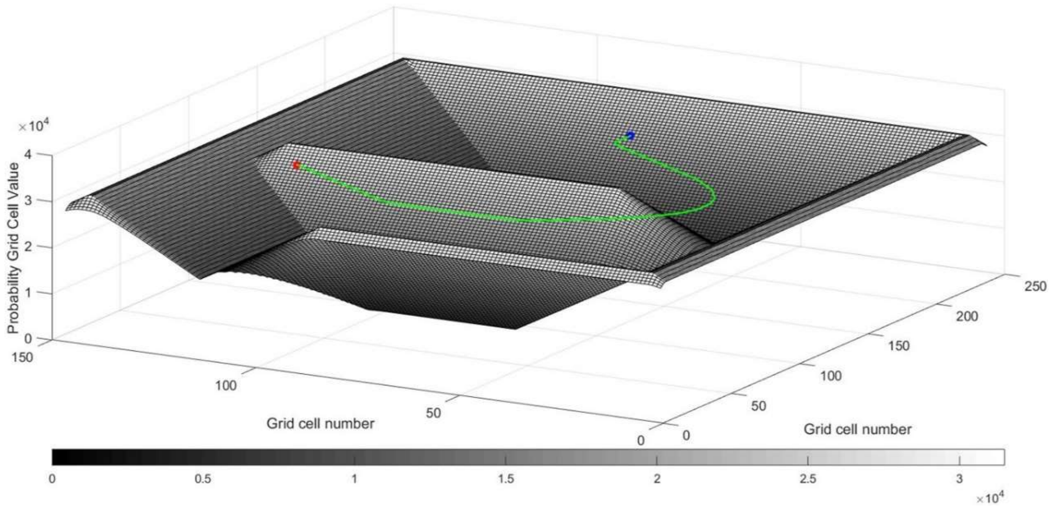


Figure 16. 3D representation of the generated path, for Dataset I, using the adjusted A* algorithm in a static environment.

For Dataset II, Figure 17 illustrates the generated path using A* algorithm, APF method, RRT algorithm, and bidirectional RRT, respectively. The black and magenta asterisks represent the positions of the source and goal points, respectively. In Figure 17a, the generated path is represented by the red line. The destination point is located between two close static objects as represented by the orange arrows. Since the current situation does not cause hurdles to the A* algorithm, an appropriate path is generated between the two close static obstacles. The processing time of the algorithm is 690 ms. Since the potential field is created depending on the attractive and the repulsive forces, the APF method is sensitive to their magnitudes. The current situation affects the result of the generated path according to the magnitudes of the attractive and repulsive forces. When the attractive force is much higher than the repulsive force, an appropriate path is generated, as shown in Figure 17b. The processing time of the algorithm is 1826 ms. However, when the repulsive force is approximately equal to the attractive force, the APF method fails to generate a path, a back and forth behavior, as represented by the orange arrows, is observed. This is because, whenever the APF method tries to reach the trapped destination, between the two close obstacles, the repulsive forces of the two obstacles repel the generated path, as shown in Figure 17c, which indicates the sensitivity of the APF algorithm to the successful tuning of the attraction and repulsion forces parameters. Figure 17d,e shows the generated path using the RRT and bidirectional RRT algorithms, respectively. The generated path is represented by the red line as shown in Figure 17d, while the red and blue lines are used to represent the generated path, as shown in Figure 17e, and the green line is used as a conjunction between the bidirectional trees. Typically, the random behavior of both algorithms affects the result of the generated path, as it is changed every time the algorithm is conducted. The processing time of RRT and bidirectional RRT are 1229 and 854 ms, respectively.

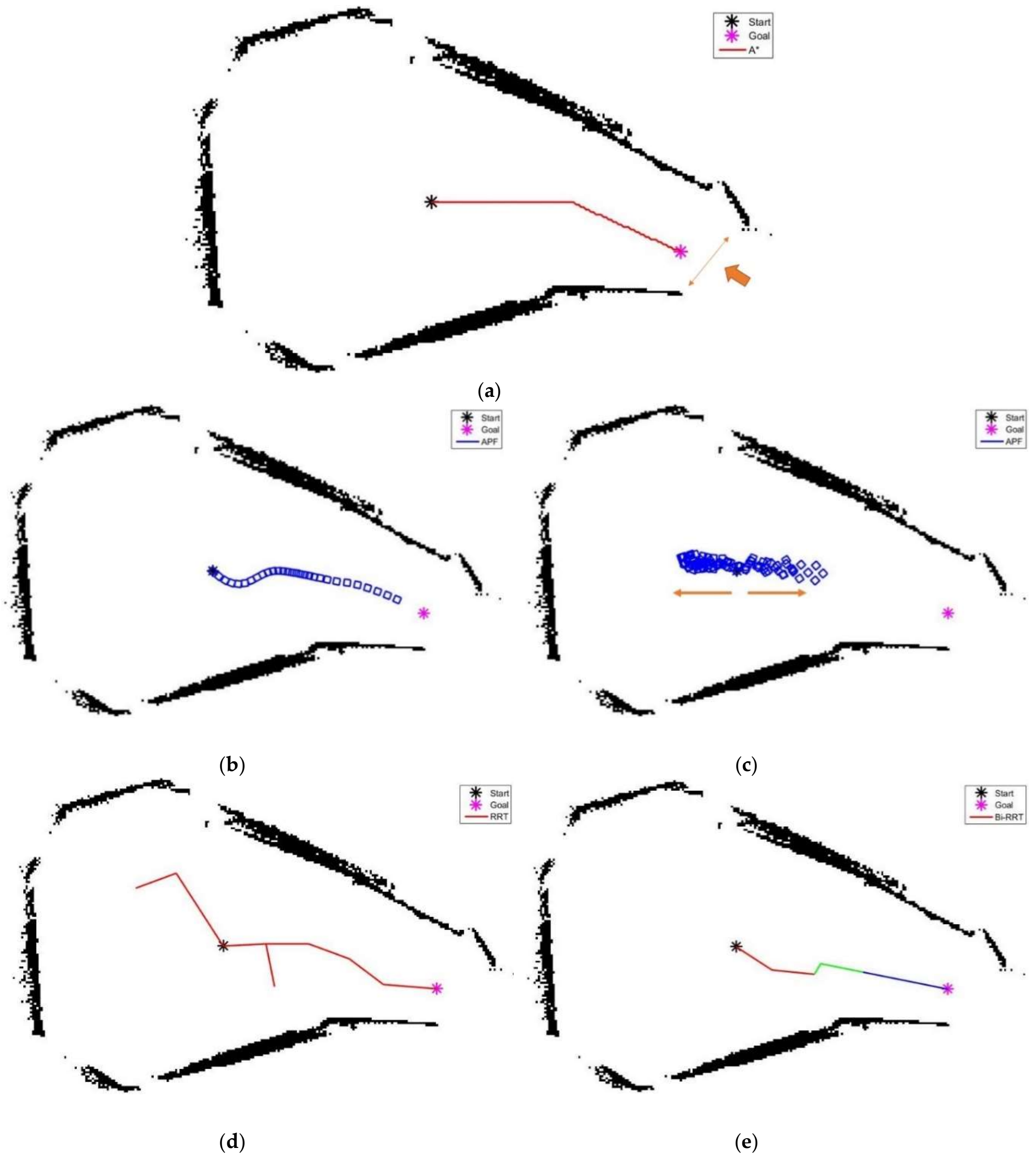


Figure 17. Generated path, for Dataset II, using: (**a**) A* algorithm; (**b**) first iteration of APF; (**c**) second iteration of APF; (**d**) RRT; and (**e**) bidirectional RRT.

Figure 18 demonstrates the generated path using the proposed method in a static environment. Figure 18a shows the cost map of the static representation. The brightest color represents the static obstacles (i.e., highest cost), whereas the cost gradually decreases for the extension of the obstacles. On the other hand, the darkest color represents the lowest cost which represents the least available risky areas. The previously executed trajectory is represented by the two bright squares as shown by the black arrows. The implemented trajectory is computed using the Gaussian kernel as mentioned earlier. Figure 18b demonstrates the 2D representation of the generated path. The blue asterisk represents the starting position, while the red asterisk represents the goal position. The smoothed path is represented by the green line. Since the only factor affecting the path generation is the static environment, the generated path is attempted to be in the middle of the obstacles to minimize the risk of the generated path. The processing time of the algorithm is 399 ms. Figure 18c depicts the 3D representation of the generated path. The blue asterisk represents the starting position, while the red asterisk represents the goal position. The smoothed path is represented by the green line. As described above, the static obstacles are represented using the DT method. Thus, the maximum height is the maximum cost, and it represents the obstacle grid cells, which decreases in magnitude to represent the extension of the obstacles until reaches approximately the area in-between the obstacles. It is clear that the generated trajectory is created inside the valley in-between the extended obstacles to prepare a safe path for the MAV. The protuberance, as indicated by the yellow arrows, represents the previously executed trajectory. These protuberances prevent the vehicle to revisit the same place twice to maximize the visited area.

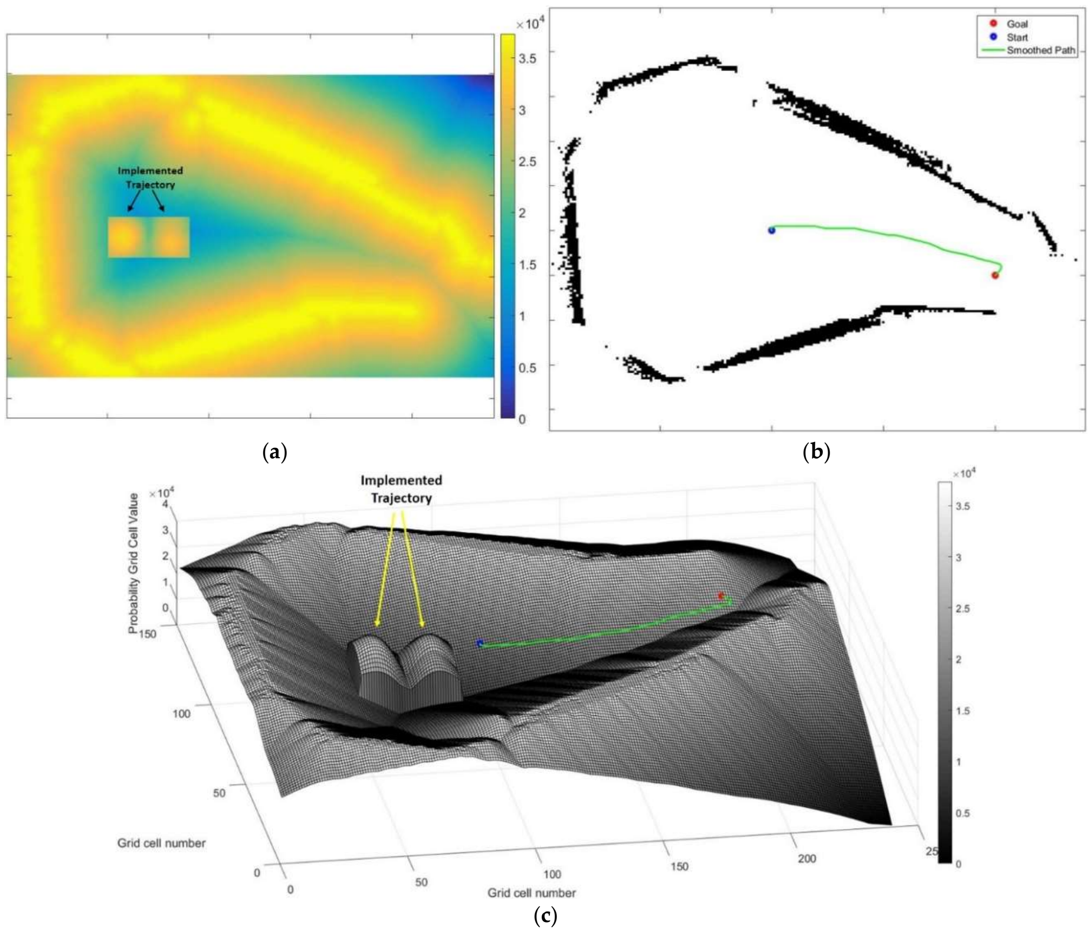
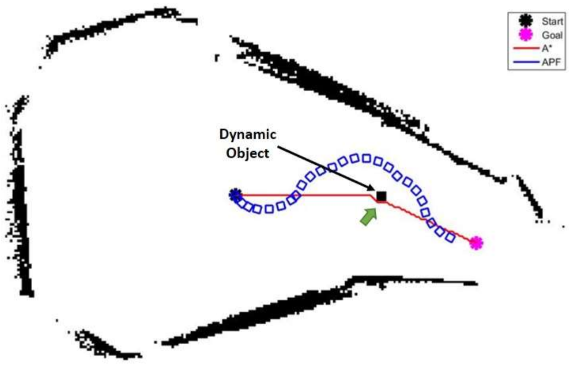
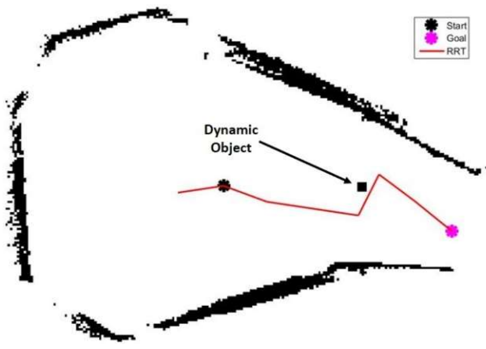


Figure 18. Adjusted A* algorithm in a static environment: (a) cost map; (b) 2D representation; and (c) 3D representation.

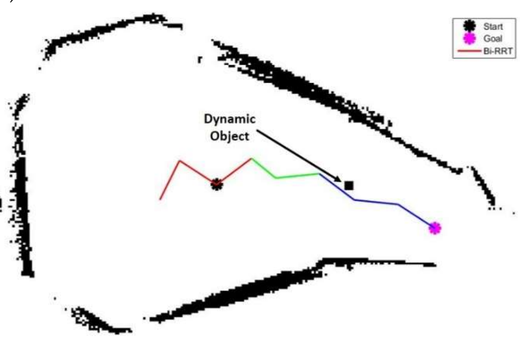
Dataset III is characterized by the presence of dynamic objects that add more complexity to the environment. Figure 19 depicts the generated path using A* algorithm, APF method, RRT algorithm, and bidirectional RRT algorithm, respectively. The blue and magenta asterisks represent the positions of the source and goal points, respectively. The dynamic object is represented by the black square. In Figure 19a, the generated path using A* algorithm is represented by the red line, while the generated path using APF method is represented by the blue squares. The A* algorithm deals with the dynamic object as a static object in the moment of generating the path. However, the algorithm cannot detect the movement of the dynamic object in the successive epochs unless the algorithm is re-run every epoch. Thus, extra computations have to be executed to achieve all the iterations. Even though the dynamic object stops at the time of generating the path, the shortest path behavior forces the generated path to be close to the dynamic object as shown by the green arrow. This obviously creates a risky situation for the vehicle. On the other hand, the APF method is influenced by the potential field that is produced from the attractive and repulsive forces from the goal and obstacles, respectively. Consequently, the potential field is frequently changing due to the effect of the moving object in the environment. Moreover, the APF method lacks the monitoring of the dynamic objects. Therefore, the traditional APF method fails to provide a solution in the dynamic environments [19]. However, the trajectory is plotted as if the dynamic object stops at the time of generating the path. The processing time of the A* algorithm and the APF method are 684 and 2108 ms, respectively. Figure 19b,c shows the generated path using the RRT and bidirectional RRT algorithms, respectively. The generated path is represented by the red line as shown in Figure 19b, while the red and blue lines are used to represent the generated path, as shown in Figure 19c, and the green line is used as conjunction between the bidirectional trees. The generated random path in the sample space at time (t) may be either occupied with a dynamic object soon (t + n), where n is a positive integer, or blocked by a dynamic object in the near future as well, as the dynamic object is changing its allocation every time. Consequently, a collision will occur between the vehicle and the dynamic object. The processing time of the RRT and bidirectional algorithms are 1085 and 935 ms, respectively.



(**a**)



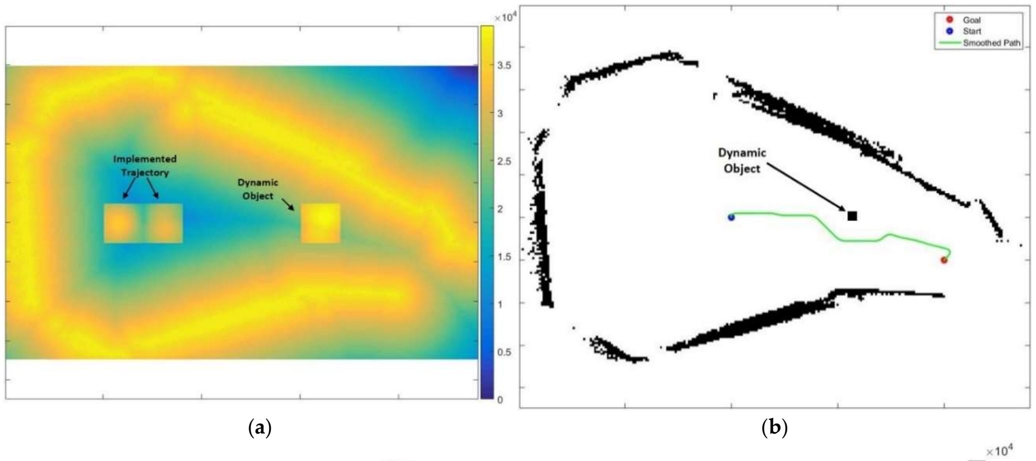
(**b**)

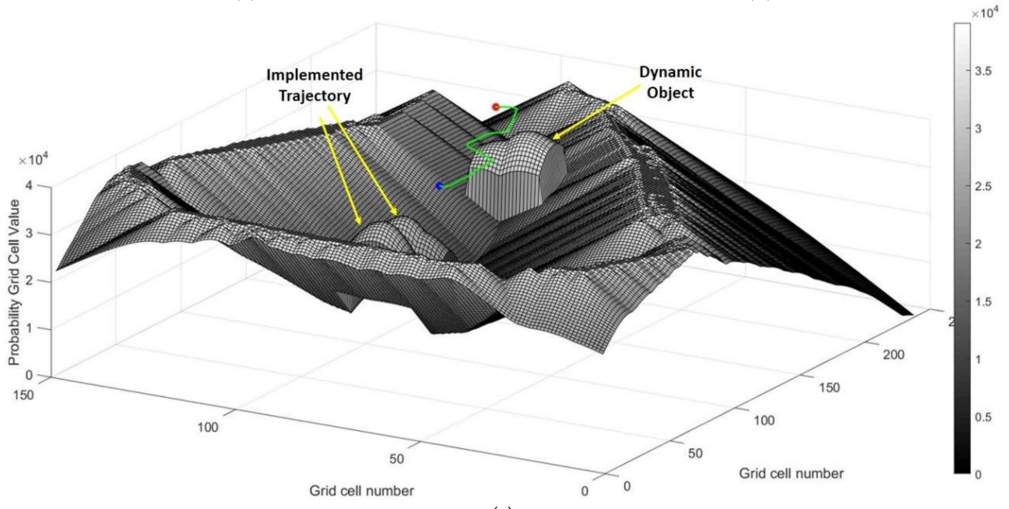


(**c**)

Figure 19. Generated path, for Dataset III, using: (**a**) A* algorithm and APF method; (**b**) RRT algorithm; and (**c**) bidirectional RRT algorithm.

Figure 20 demonstrates the generated path using the proposed method in a dynamic environment. Figure 20a shows the cost map of the static and dynamic representations, which is computed from the static and dynamic objects using the distance transform method and Gaussian kernel, respectively. The brightest color represents the highest cost which is the static, implemented trajectory and dynamic object. The previously executed trajectory is represented by the two successive bright squares. The previously executed trajectory is represented using the Gaussian kernel. The dynamic object is represented by the bright square which is computed using the Gaussian kernel as well. On the other hand, the darkest color represents the lowest cost which represents the least available risky areas. This cost map is fed into the adjusted A* algorithm as a heuristic matrix. Figure 20 demonstrates the 2D representation of the generated path. The blue asterisk represents the starting position, while the red asterisk represents the goal position. The smoothed path is represented by the green line. The black square represents a dynamic object. By comparing the static and dynamic results of the proposed method, it is clear that the generated path deviates to avoid the dynamic object and minimize the risk of the generated path. The amount of the deviation varies according to the static and dynamic costs. For crowded environments, the cost of the dynamic objects is approximately high with respect to the static cost, but for uncongested environments, the cost of the dynamic objects is approximately low with respect to the static cost. The processing time of the algorithm is 374 ms. Figure 20c demonstrates the 3D representation of the generated path. The blue asterisk represents the starting position, while the red asterisk represents the goal position. The smoothed path is represented by the green line. The consecutive protuberances represent the previously executed trajectory, while the single protuberance represents the dynamic object. The performed deviation depends on the total computed costs from the static and dynamic objects in the current scene. It is clear that the generated path endeavors to avoid the dynamic object and simultaneously avoids proximity to the static object according to their costs.





(c)

Figure 20. Adjusted A* algorithm in a dynamic environment: (**a**) cost map; (**b**) 2D representation; and (**c**) 3D representation.

For Dataset IV, Figure 21 illustrates the generated paths using A* algorithm, APF method, RRT algorithm, and bidirectional RRT algorithm. The black and magenta asterisks represent the positions of the source and goal points, respectively. In Figure 21a, the generated path using A* algorithm is represented by the red line, while the generated path using APF method is represented by the blue squares. The generated path of the A* algorithm is not far from the static obstacle, as shown by the green arrow, due to shortest path behavior of the algorithm. This closeness increases the risk of the vehicle. Due to the attraction and repulsion forces of the goal and obstacles, respectively, as well as the dynamic of the vehicle, the generated path of the APF method fails to reach the goal as the vehicle collides with the static object, as shown by the green circle. The processing time of the A* algorithm is 131 ms, while the processing time of the APF method is not available due to the collision. Figure 21b shows the generated path of different iteration using APF method. The source and goal points are represented by the black and red asterisks, respectively. The generated path is plotted by the blue squares. Since the APF method is sensitive to the attraction and repulsion forces, as well as the corridor width, which is approximately 120 cm. The attractive force is tuned to be much greater than the repulsion force. As a result, a generated path is created from the source point to the goal point. The processing time is 2172 ms. Figure 21c,d shows the generated path using the RRT and bidirectional RRT algorithms, respectively. The generated path is represented by the red line, as shown in Figure 21c, while the red and blue lines are used to represent the generated path, as shown in Figure 21d, and the green line is used as a conjunction between the bidirectional trees. In addition to the diversity of the results in each iteration due to the random behavior of both algorithms, both algorithms can build new intermediate nodes far away from the goal point that consumes time. The RRT algorithm generates the path in the opposite direction of the movement, as shown by the green arrow. Thus, the RRT algorithm can revisit the same place twice. The bidirectional RRT creates intermediate nodes through small gaps due to the miss occupation in the static obstacles, as presented by the green arrow. The processing times of the RRT and bidirectional RRT are 2839 and 1320 ms, respectively.

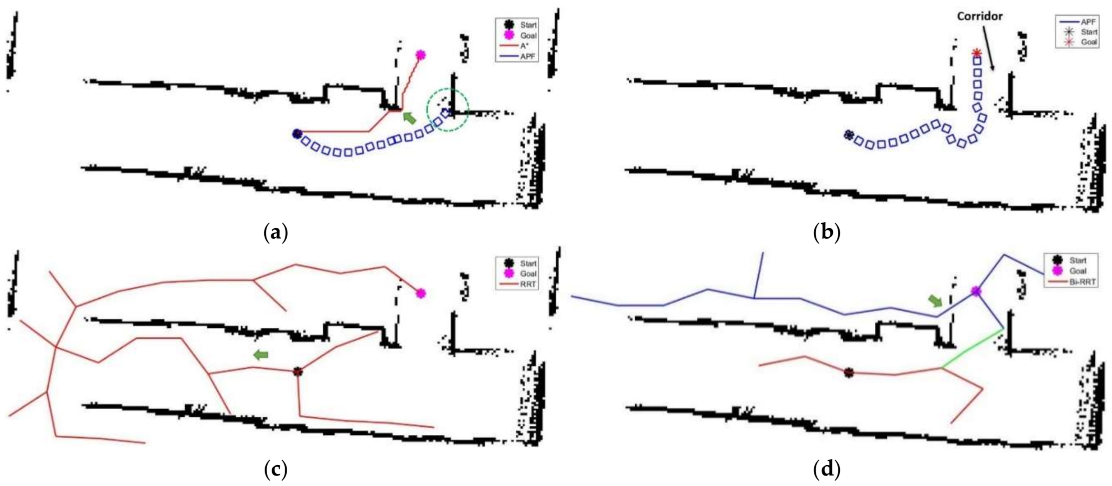


Figure 21. Generated path, for Dataset IV, using: (**a**) A* algorithm and APF method; (**b**) second iteration of APF method; (**c**) RRT algorithm; and (**d**) bidirectional RRT algorithm.

Figure 22 demonstrates the generated path using the proposed method in a static environment. Figure 22a shows the cost map of the static representation. The brightest color represents the static obstacles (i.e., highest cost), whereas the cost gradually decreases for the extension of the obstacles. On the other hand, the darkest color represents the lowest cost which represents the least available risky areas. The previously executed trajectory is represented by the two bright squares, as shown by the black arrows. Due to the presence of the high cost of the previously executed trajectory, the MAV cannot go back to reach the goal point, as shown by the dashed arrow, to maximize the visited area. Figure 22b demonstrates the 2D representation of the generated path. The blue asterisk represents the starting position, while the red asterisk represents the goal position. The smoothed path is represented by the green line. Obviously, the generated path is attempted to be in the middle of the obstacles as usual under different scenarios to minimize the risk of the generated path. The processing time of the algorithm is 286 ms. Figure 22c depicts the 3D representation of the generated path. The blue circle represents the starting position, while the red circle represents the goal position. The smoothed path is represented by the green line. As described above, the implemented trajectory is represented using the Gaussian kernels. Thus, the protuberance represents the occurrence of the implemented trajectory as shown by the yellow arrows. Thus, the vehicle is driven to explore new places to maximize the visited area.

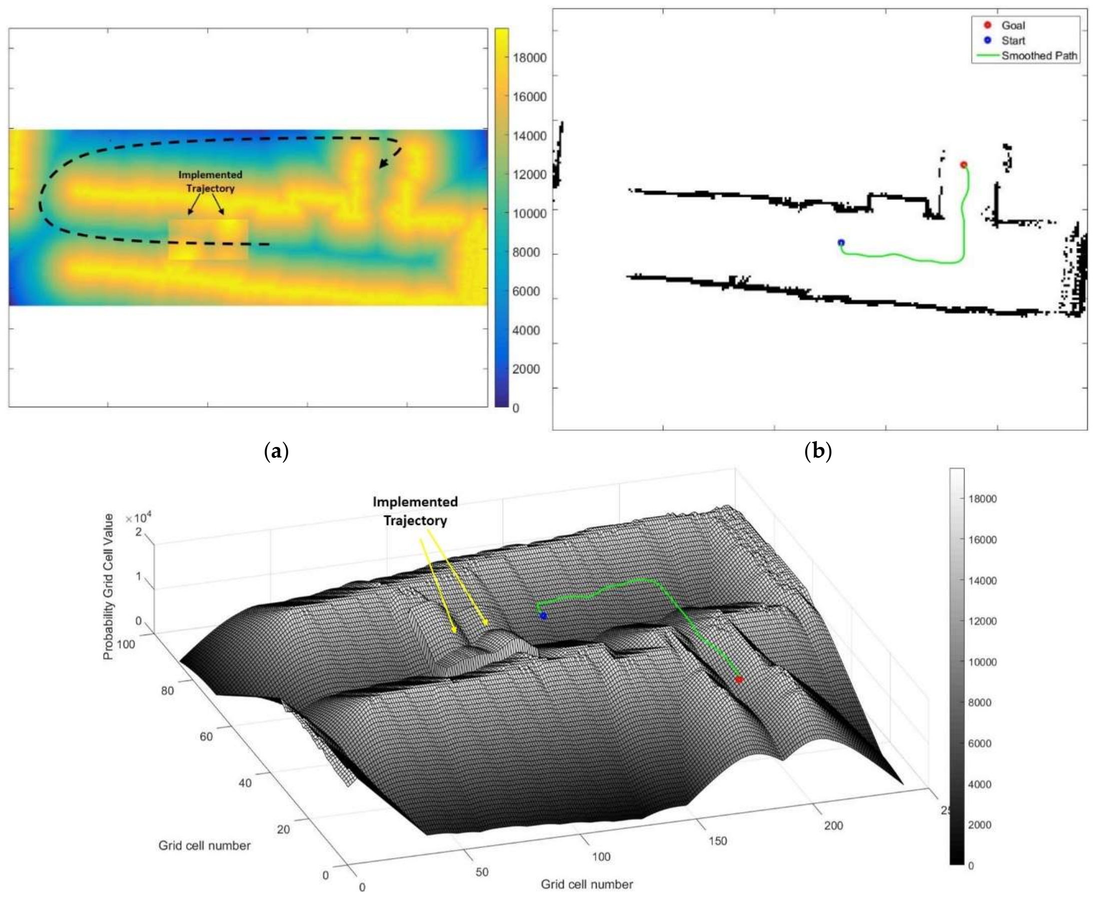


Figure 22. Adjusted A* algorithm in a static environment: (**a**) cost map; (**b**) 2D representation; and (**c**) 3D representation.

For Dataset V, a dynamic scenario is created in which a dynamic object moves inside a narrow corridor where the MAV cannot safely pass. Figure 23 depicts the generated path using A* algorithm, APF method, RRT algorithm, and bidirectional RRT algorithm. The black and magenta asterisks represent the positions of the source and goal points, respectively. In Figure 23a, the generated path using A* algorithm is represented by the red line, while the generated path using APF method is represented by the blue squares. Despite the existence of a dynamic object in the entrance of the corridor, the A* algorithm generates a path through the corridor to reach the goal point. Furthermore, the generated path is near to the static object, as shown by the green arrow. This obviously creates a risky situation for the vehicle. On the other hand, the APF method fails to reach the goal point, and a collision occurs due to the repulsive forces that are generated from the static and dynamic objects. The processing time of the A* algorithm is 136 ms, while the processing time of the APF method is not available due to the collision. Figure 23b shows the generated path using the RRT algorithm. The generated path is represented by the red line. The generated path is also passed across the corridor beside the dynamic object, as shown by the green arrow, to reach the goal point. This also creates a risky situation for the MAV. The processing time of the RRT algorithm is 2323 ms. Figure 23c shows the generated path using the bidirectional RRT algorithm. The red and blue lines are used to represent the generated path and the green line is used as a conjunction between the bidirectional trees. The algorithm generates the path through the corridor and closes to the dynamic object. The vehicle is exposed to the same risky situation. The processing time of the bidirectional RRT algorithm is 436 ms.

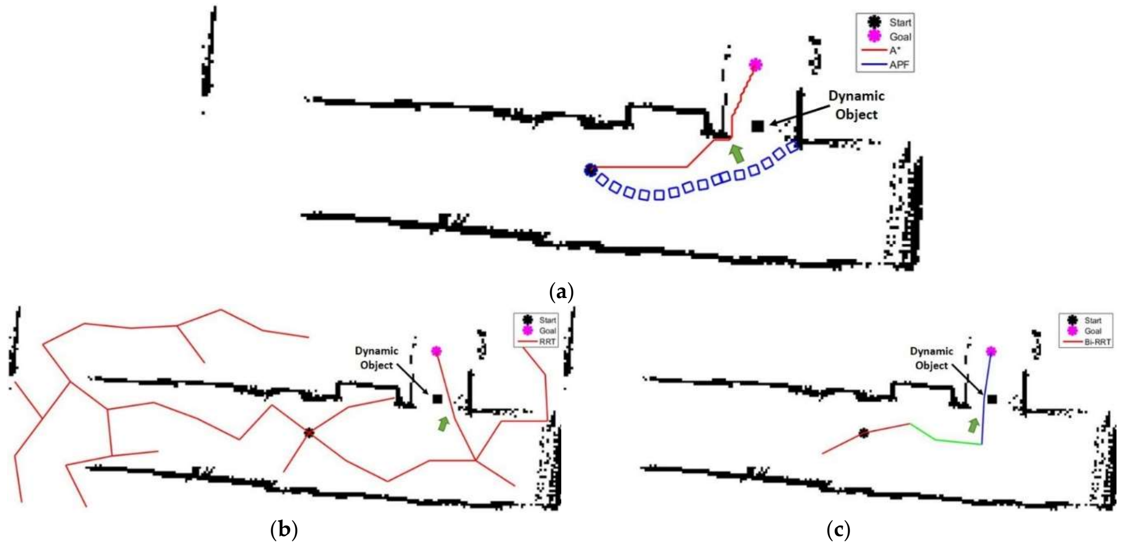


Figure 23. Generated path, for Dataset V, using: (**a**) A* algorithm and APF method; (**b**) RRT algorithm; and (**c**) bidirectional RRT algorithm.

On the other hand, the proposed method does not generate a path to the destination, due to the movement of the dynamic object in the entrance of the corridor, to minimize the risk on the vehicle. Hence, the proposed method either keeps the vehicle for a certain period until the dynamic object moves or generates a new path to a new destination if feasible. Ultimately, Table 2 illustrates a comparison between the proposed method among other methods/algorithms. It is clear that the proposed method generates a path and safely reaches the goal in all the experiments except the last experiment as aforementioned. Furthermore, the proposed method is characterized by the largest average distance to the nearest obstacle with the least average processing time.

Table 2. A comparison between the proposed method among other methods/algorithms.

		A*	**APF**	**RRT**	**Bidirectional RRT**	**Proposed Method**
Number of Reaching Goal	Safe	1	2	1	1	4
	Risk	4	0	3	3	0
	Failed	0	3	1	1	0
	Does not generate Path	0	0	0	0	1
Average Processing Time	[ms]	421.4	2035.3	2542.3	1506.0	402.2
Average Distance to Nearest Obstacle	[cm]	12.0	53.0	44.3	45.4	64.0

7. Conclusions

In robotics, path planning is a problem of finding a proper trajectory from the current position to a definite destination, according to the limitations of the environment and the vehicle. The environment could be unknown and/or dynamic, while the limitations of the vehicle can be for instance the vehicle's size and maximum velocity of the vehicle. The objective of path planning problem varies based on the application requirements such as finding the shortest path or the fastest path. The time constraint is a critical factor in most path planning problems, especially search and rescue operations. Hence, maximizing the visited area during such operations is a fundamental task. Furthermore, minimizing the risk on the generated path is also an important task. Therefore, an efficient exploration method is proposed to maximize the visited area of unknown static/dynamic environment and simultaneously minimize the risk of the generated path. The proposed method builds the cost function of the static objects using the distance transform method, while uses the Gaussian kernel for the representation of the dynamic objects. Furthermore, the previously executed trajectory is represented by Gaussian kernels as a cost function. Thus, the proposed method generates the path that achieves the minimum summation of the cost functions. Additionally, the proposed method can reevaluate the destination and the trajectory for sudden situations because an interruption is fired once an interception of a dynamic object occurs inside the sanctuary area around the vehicle. Ultimately, the previously executed trajectory is recorded for the entire mission, and the proposed method uses this trajectory during the exit process for time saving. For validation and evaluation, the efficiency of the proposed method is compared with various algorithms and methods, such as A* algorithm, APF method, RRT algorithm, and bidirectional RRT, under different scenarios. The proposed method exhibits efficiency in avoiding both static and dynamic obstacles. The average distance to the nearest obstacle of the A* algorithm, APF, RRT, bidirectional RRT, and the proposed method is 12, 53, 44.3, 45.4, and 64 cm, respectively. The proposed method maximizes the visited area as it never goes to same place twice. Furthermore, the average processing time of the A* algorithm, APF, RRT, bidirectional RRT, and the proposed method is 421.4, 2035.3, 2542.3, 1506.0, and 402.2 ms, respectively. The proposed results indicate the eligibility of the proposed method for real-time applications.

Author Contributions: This research work was accomplished under the supervision of N.E.-S. H.M. and A.M. designed and implemented the proposed algorithm and performed the experiments. H.M. wrote the paper. N.E.-S. contributed the UAV and the sensors used in the experiments. A.M., M.E., and N.E.-S. reviewed and provided feedback on the paper.

Acknowledgments: This work was supported by Naser El-Sheimy research funds from NSERC and Canada Research Chairs programs. Thanks also go to the funding of the first author by the Egyptian government.

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

References

1. Quintero M., C.G.; Oñate López, J.; Bertel R., F.A. Intelligent exploration and surveillance algorithms for multi-agents robotics systems. In Proceedings of the 38th Latin America Conference on Informatics, Medellin, Colombia, 1–5 October 2012; pp. 1–10.
2. Yamauchi, B.; Schultz, A.; Adams, W. Integrating Exploration and Localization for Mobile Robots. *Adapt. Behav.* **1999**, *7*, 217–229. [CrossRef]
3. Tijsma, A.D.; Drugan, M.M.; Wiering, M.A. Comparing exploration strategies for Q-learning in random stochastic mazes. In Proceedings of the 2016 IEEE Symposium Series on Computational Intelligence (SSCI), Athens, Greece, 6–9 December 2016; pp. 1–8.
4. Sugiyama, H.; Tsujioka, T.; Murata, M. Collaborative Movement of Rescue Robots for Reliable and Effective Networking in Disaster Area. In Proceedings of the 2005 International Conference on Collaborative Computing: Networking, Applications and Worksharing, San Jose, CA, USA, 19–21 December 2005; pp. 1–7.
5. Hougen, D.F.; Benjaafar, S.; Bonney, J.C.; Budenske, J.R.; Dvorak, M.; Gini, M.; French, H.; Krantz, D.G.; Li, P.Y.; Malver, F.; et al. A miniature robotic system for reconnaissance and surveillance. In Proceedings of the 2000 IEEE International Conference on Robotics and Automation, San Francisco, CA, USA, 24–28 April 2000; Volume 1, pp. 501–507.
6. Hoffmann, R.; Weikersdorfer, D.; Conradt, J. Autonomous indoor exploration with an event-based visual SLAM system. In Proceedings of the 2013 European Conference on Mobile Robots, Barcelona, Spain, 25–27 September 2013; pp. 38–43.

7. Buniyamin, N.; Ngah, W.W.; Sariff, N.; Mohamad, Z. A simple local path planning algorithm for autonomous mobile robots. *Int. J. Syst. Appl. Eng. Dev.* **2011**, *5*, 151–159.
8. Surgade, A.; Bhavsar, N.; Thale, K.; Deshpande, A.; Mohite, D. Real-time navigation for indoor environment. In Proceedings of the 2017 International Conference on Advances in Computing, Communication and Control (ICAC3), Mumbai, India, 1–2 December 2017; pp. 1–6.
9. Richter, C.; Bry, A.; Roy, N. Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments. In *Springer Tracts in Advanced Robotics*; Springer: Cham, Switzerland, 2016; Volume 114, pp. 649–666.
10. Barbehenn, M. A note on the complexity of Dijkstra's algorithm for graphs with weighted vertices. *IEEE Trans. Comput.* **1998**, *47*, 263. [CrossRef]
11. Kang, H.I.; Lee, B.; Kim, K. Path Planning Algorithm Using the Particle Swarm Optimization and the Improved Dijkstra Algorithm. In Proceedings of the 2008 IEEE Pacific-Asia Workshop on Computational Intelligence and Industrial Application, Wuhan, China, 19–20 December 2008; pp. 1002–1004.
12. Malaek, S.; Kosari, A. Novel minimum time trajectory planning in terrain following flights. *IEEE Trans. Aerosp. Electron. Syst.* **2007**, *43*, 2–12. [CrossRef]
13. Tsai, Y.-J.; Lee, C.-S.; Lin, C.-L.; Huang, C.-H. Development of Flight Path Planning for Multirotor Aerial Vehicles. *Aerospace* **2015**, *2*. [CrossRef]
14. Malaek, S.M.; Kosari, A.R. Dynamic Based Cost Functions for TF/TA Flights. *IEEE Trans. Aerosp. Electron. Syst.* **2012**, *48*, 44–63. [CrossRef]
15. Ferguson, D.; Stentz, A. Anytime RRTs. In Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems, Beijing, China, 9–15 October 2006; pp. 5369–5375.
16. Lavalle, S.M.; Lavalle, S.M.; Kuffner, J.J., Jr. Rapidly-Exploring Random Trees: Progress and Prospects. In Proceedings of the Fourth International Workshop on Algorithmic Foundations of Robotics, Hanover, NH, USA, 16–18 March 2000; pp. 293–308.
17. Qureshi, A.H.; Ayaz, Y. Intelligent bidirectional rapidly-exploring random trees for optimal motion planning in complex cluttered environments. *Robot. Auton. Syst.* **2015**, *68*, 1–11. [CrossRef]
18. Xinggang, W.; Cong, G.; Yibo, L. Variable probability based bidirectional RRT algorithm for UAV path planning. In Proceedings of the 26th Chinese Control and Decision Conference (2014 CCDC), Changsha, China, 31 May–2 June 2014; pp. 2217–2222.
19. Acuña, R.; Terrones, A.; Certad-H, N.; Fermín-León, L.; Fernández-López, G. Dynamic Potential Field Generation Using Movement Prediction. In Proceedings of the Fifteenth International Conference on Climbing and Walking Robots and the Support Technologies for Mobile Machines, Baltimore, MD, USA, 23–26 July 2012.
20. Bruce, J.; Veloso, M. Real-time randomized path planning for robot navigation. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and System, Lausanne, Switzerland, 30 September–4 October 2002; Volume 3, pp. 2383–2388.
21. Ferguson, D.; Kalra, N.; Stentz, A. Replanning with RRTs. In Proceedings of the IEEE International Conference on Robotics and Automation, Orlando, FL, USA, 15–19 May 2006; Volume 2006, pp. 1243–1248.
22. Li, T.Y.; Shie, Y.C. An incremental learning approach to motion planning with roadmap management. In Proceedings of the 2002 IEEE International Conference on Robotics and Automation, Washington, DC, USA, 11–15 May 2002; Volume 4, pp. 3411–3416.
23. Zucker, M.; Kuffner, J.; Branicky, M. Multipartite RRTs for Rapid Replanning in Dynamic Environments. In Proceedings of the 2007 IEEE International Conference on Robotics and Automation, Roma, Italy, 10–14 April 2007; pp. 1603–1609.
24. Mohamed, H.; Moussa, A.; Elhabiby, M.; El-Sheimy, N.; Sesay, A. A Novel Real-Time Reference Key Frame Scan Matching Method. *Sensors* **2017**, *17*, 1060. [CrossRef] [PubMed]
25. Mohamed, H.; Moussa, A.; Elhabiby, M.; El-Sheimy, N.; Sesay, A.B. Corner Features Aided Indoor Slam for Unmanned Vehicles. In Proceedings of the 10th International Conference on Mobile Mapping Technology (MMT), Cairo, Egypt, 6–8 May 2017.
26. Fabbri, R.; Costa, L.D.F.; Torelli, J.C.; Bruno, O.M. 2D Euclidean distance transform algorithms. *ACM Comput. Surv.* **2008**, *40*, 1–44. [CrossRef]
27. Wu, Q. Incremental Routing Algorithms for Dynamic Transportation Networks. Master's Thesis, University of Calgary, Calgary, AB, Canada, 2006.

28. Smith, S.W. *The Scientist and Engineer's Guide to Digital Signal Processing*, 2nd ed.; California Technical Pub.: San Diego, CA, 1999.
29. Mostafa, M.; Zahran, S.; Moussa, A.; El-Sheimy, N.; Sesay, A.; Mostafa, M.; Zahran, S.; Moussa, A.; El-Sheimy, N.; Sesay, A. Radar and Visual Odometry Integrated System Aided Navigation for UAVS in GNSS Denied Environment. *Sensors* **2018**, *18*, 2776. [CrossRef] [PubMed]