# Development of an intelligent UAV path planning approach to minimize the costs in flight distance, time, altitude, and obstacle collision

Shiwei Lin
*Faculty of Engineering and Information Technology*
*University of Technology Sydney*
Sydney, Australia

Xiaoying Kong
*Faculty of Engineering and Information Technology*
*University of Technology Sydney*
Sydney, Australia

Li Liu
*Faculty of Engineering and Information Technologies*
*The University of Sydney*
Sydney, Australia

***Abstract*— This paper presents a UAV path planning approach to consider flight cost functions. The UAV flight paths are generated using quintic Hermite interpolation. These paths are constant in speed, and the algorithm generates the paths by iteration to ensure the path segments are smooth. We designed a Waypoint-Matrix to store the points of the path. The paths are aimed to reach the defined destination by passing the waypoints and avoiding obstacles. In this approach, we developed flight cost functions to evaluate the paths, and path length, flight time, altitude, and collision. This approach is validated by simulation results.**

***Keywords—Path planning, obstacle avoidance, UAV, cost function***

## I. Introduction

Path planning and obstacle avoidance are major processes in Unmanned Aerial Vehicles (UAV) navigation. Path planning is to find a path between one position and one destination position. UAV navigation can be operated in several environments, such as indoor area, air, and underwater. UAVs can carry cameras, sensors, urgent suppliers, or communications equipment, and they can be controlled remotely or operate autonomously [1]. UAVs can perform civilian applications, containing aerial photography, video shoots, aerial mapping, inspection, reconnaissance and surveillance, wildfire suppression, 3D modeling, agricultural services, traffic monitoring, intelligence, environmental monitoring, and search-and-rescue [1-6].

To achieve the missions of UAVs, terrain following flight is involved in protecting UAVs from collisions. Terrain following is to find a path that is close to the ground [1]. Goal assignment, team composition, path optimization, and resource allocation are contained to support multiple UAVs coordination [7]. Mostly, solving the best path planning is related to deterministic search algorithms. But there was a trend of using non-deterministic algorithms, such as the particle swarm optimization algorithm and the genetic algorithm [8].

Also, Dijkstra's algorithm, Bellman-Ford algorithm, Floyd-Warshall's algorithm, the vortex search algorithm, and A* can be used for finding the shortest path [9-10]. A discretization of the airspace is meaningful for applying the path planning algorithms [1,3,5,8]. From previous research, quintic Hermite interpolation can also be used to calculate path segments [1,3]. Cost functions are utilized to evaluate the optimal path to ensure the path has minimal cost when compared with other paths. Costs can be calculated based on the actual costs or estimated costs. Cost functions of UAVs usually contain the consideration of path length, flight altitude, danger zones, energy consumption, threats, flight time, and path segments [8,9,11-14].

We present the paper to solve the problem of finding the optimal path to the destination for UAVs. The paper is organized as follows. Section II reviews the related work for UAV path planning. Section III presents the path planning algorithm considering cost functions. Section IV presents the simulation results to validate our approach. We conclude the paper in Section V.

## II. Related work

Path planning must consider avoiding obstacle to achieve UAV navigation successfully and safely. For harbor environments, the autonomous surface craft uses 2D laser sensor and the reactive algorithm for obstacle avoidance with a linear prediction [15]. Probabilistic roadmap, artificial potential fields, grid-based methods, genetic algorithm, rapidly-exploring random tree, B-spline curves, and A* search are offered for 2D or 3D path planning [16]. P-norms, sampling-based planning, polygonal shapes and Dubins curves are to define obstacles [4]. Splines, Bezier curves, polynomials are functions that are employed for quadrotor UAV trajectory planning, and the semi-analytical path planning method is proposed [4].

Additionally, to perform autonomous UAV missions, humans' cognitive abilities can be combined with fast automation by a flight control system that is based on software framework and the instantaneous Task Specification using Constraints (iTaSC) [2]. Flight path planning can be found on discretizing the airspace by a special 3D network, and the path considers the performance of the UAV and no-fly areas [1,3]. More importantly, the planning must respond quickly to changing environmental conditions, tactical considerations and operational requirements, and the algorithm must consider differential constraints [3]. Artificial potential field method has been improved to plan a 3D path in a dynamic environment [16].

Moreover, high-fidelity geographical information systems, high-performance-embedded computing hardware, and automated digital autopilots systems are contributed to achieving autonomous operations of UAVs [6]. The propagation problems have been solved by boundary-shift method and piecewise linear shift map method [17]. The true terrain data can support to analyze the availability of the

Ground Based Augmentation System [18]. Three-dimensional terrain following needs to evaluate dynamic and kinematic equations of the aircraft and utilizes the direct transcription method for the optimal control problem [19]. The spline-RRT* algorithm is optimal path planner of terrain following [20].

Furthermore, cost functions are developed to estimate the optimal path for autonomous navigation. Algorithm A* can be evaluated by the sum of the actual costs of optimal paths from the start point to the destination, or by the amount of the actual costs and the heuristic estimate of the cost from the start point to the target [9,21]. A* algorithm can also be examined by the actual cost and the estimated cost from the start point to the destination [14]. Longer paths, the higher average altitude, paths passing danger zones, requiring more power and fuel of the UAV, collision with the ground, and paths which are not smoothed by circular arcs are leading to higher cost [8,13].

Besides, cruise steady power consumption, relative position change, mission energy, charging path, and airborne equipment are estimated as energy consumption [11]. Costs can be determined with the finishing times discovered by approximating straight-line path, forbid zones, terrain threats, path regulation and turn maneuvers affect the costs [7,12-13]. The threat costs of each vehicle and fuel consumption are required to be minimised, and path length and speed affect fuel consumption [22]. The costs of fuel also link to whether the UAV is climbing or descending [23]. When executing the spline-RRT* algorithm, the number of vertices have an impact on costs [20]. The height deviations and the deviations of the corners are summed up to cost functions [10]. Table I lists the comparisons of path planning methods and cost functions in literature.

TABLE I. PATH PLANNING METHODS

| No. | Papers | Methodologies | Equations |
|---|---|---|---|
| 1 | [1,3,5] | The network-based method with the discretization of the airspace | |
| 2 | [2] | Combining iTaSC with software framework for the flight control system | |
| 3 | [4] | Analytically define trajectories with the discretization of the path | |
| 4 | [6,9,12-14,21] | Algorithm A* | $f(n) = g(n) + h(n)$ where g(n) is the actual cost of the best path from the start point to a node n, and h(n) is the heuristic estimate of the cost of the best path from n to a preferred point of n. <br><br> $\overline{f(x)} = w_1 * \overline{f_{DıS}(x)} + w_2 * \overline{f_{Threat}(x)} + w_3 * \overline{f_{Turn}(x)} + w_4 * \overline{f_{Forbıd}(x)}$ s.t. $\sum_{i=1}^{4} W_i = 1$ where $\overline{f_ı(x)}$ is the normalized $f_i(x)$, $w_i$ are the weights of distance, threat zones, turn maneuvers and forbid zones. |
| 5 | [8] | The genetic algorithm | $F_{cost} = C_{length} + C_{altitude} + C_{danger\ zones} + C_{power} + C_{collsion} + C_{fuel} + C_{smoothing}$ where $F_{cost}$ is larger with longer paths, the higher average altitude, paths passing danger zones, requiring more power, collision with the ground, requiring more fuel, and paths which are not smoothed by circular arcs. |
| 6 | [9] | Bellman-Ford algorithm, Dijkstra's algorithm, Floyd-Warshall's algorithm | |
| 7 | [10] | The vortex search algorithm | $\min f = \min(f_{path} + f_{height} + f_{cornor})$ where $f_{path}$ is the length of the path, $f_{height}$ is summing up the height deviations, and $f_{cornor}$ is summing up the deviation of the corners. |
| 8 | [13] | The continuous optimization method | $\tilde{p}(C(x)) = W_1P_T(C(X)) + W_2P_{unflyable}(C(X)) + W_3P_{alt}(C(X)) + W_4P_r(C(X))$ where $W_1$, $W_2$, $W_3$, and $W_4$ are constants, $P_T$ is terrain threats, $P_{unflyable}$ is un-flyable zones, $P_{alt}$ is altitude constraint, and $P_r$ is path regulation. |
| 9 | [15] | A reactive algorithm by a linear prediction | |
| 10 | [16] | Artificial potential field method | |
| 11 | [17] | The split-step Fourier transform algorithm for the irregular terrain | |
| 12 | [19] | Direct transcription method | |
| 13 | [20] | The spline-RRT* algorithm | $Cost(x_a, x_b) = (\sum_{j=a}^{b-1} f_{c,e}(X_j, X_{j+1})) / \text{PathLength}(X_a, X_b)$ where *a* and *b* are the tree depth of the related vertex, and sub-function PathLength is the path length. |
| 14 | [11,22-23] | The particle swarm optimization algorithm | $J = \sum_{i=1}^{N} J_{t,i}(\varepsilon_j, v_i)$ where $J_{t,i}$ is the threat cost for each vehicle; N is the number of vehicles; i = 1,2,3…,N; $\varepsilon_j$ is the path description; $v_i$ is velocity. <br><br> $Ci = \sum_{k=0}^{Q-1}(\lVert w_i^k - w_i^{k+1} \rVert + k_C v_C - k_d v_d)$ where $\lVert w_i^k - w_i^{k+1} \rVert$ is the length of the path between the point $w_i^k$ and the point $w_i^{k+1}$; $k_C$ and $k_d$ are the gains of climbing and descending; $v_C$ and $v_d$ are climb and descent of the path. <br><br> $E(t) = E_0 - \Delta E_{cruise}(t) - \Delta E_{switch}(t) - \Delta E_{mission}(t) - \Delta E_{charge}(t) - \Delta E_{device}(t)$ |

| No. | Papers | Methodologies | Equations |
|---|---|---|---|
| | | | where $E_0$ is the initial energy, $\Delta E_{cruise}(t)$ is cruise steady power consumption, $\Delta E_{switch}(t)$ is the energy consumption of relative position change, $\Delta E_{mission}(t)$ is the energy consumption of mission energy, $\Delta E_{charge}(t)$ is the energy consumption of charging path, and $\Delta E_{device}(t)$ is the energy consumption of airborne equipment. |

UAVs have several applications, and autonomous navigation plays a crucial role in these missions. Path planning is the basis of autonomous navigation, which ensures UAVs can reach the destination point successfully from a start point without collision. Robot motion planning is similar to the principle of UAV motion planning, but UAVs cannot support sharp turns in a path [5]. The dynamic and the kinematic behavior of the UAV should be taken into account, so the path must be smooth [3]. From the literature, most papers focus on considering path length for costs function or energy consumptions, ignoring other factors that may have impacts on costs. Some cost functions involve constants, which may raise the difficulties of implementing them.

## III. Path planning algorithm

### A. *Description of the algorithm*

This paper presents a new UAV path planning algorithm to find an optimal path from the defined start point to the destination and minimize the costs. A sequence of landmarks can achieve successful navigation, and each landmark can be related to one or multiple preferred directions [5]. For applying the planning algorithm, the airspace will be discretised to meaningful representations for the algorithm.

We design a Waypoint-Matrix approach. Waypoints are defined in the specified order. Each waypoint is represented as *(x, y, z)* for 3D navigation. The matrix with *n* waypoints is described as:

$$\text{Waypoints} = \begin{bmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \\ \vdots & \vdots & \vdots \\ x_n & y_n & z_n \end{bmatrix} \tag{1}$$

Based on the research in literature, determining the flight path segment can be achieved by quintic Hermite interpolation [1, 3]. Using *(x(t), y(t), z(t))* to locate the position of the UAV at the specific time *t*. We define $t_{max}$ as the longest flight time from Point 1 to Point 2. For the path segment between Point 1 and Point 2, polynomials *x(t)*, *y(t)*, *z(t)* are:

$$\begin{gathered} x(0) = x_1, \quad x(t_{max}) = x_2 \\ x'(0) = v_{1x}, \quad x'(t_{max}) = v_{2x} \\ x''(0) = 0, \; x''(t_{max}) = 0 \end{gathered} \tag{2}$$

$$\begin{gathered} y(0) = y_1, \quad y(t_{max}) = y_2 \\ y'(0) = v_{1y}, \quad y'(t_{max}) = v_{2y} \\ y''(0) = 0, \; y''(t_{max}) = 0 \end{gathered} \tag{3}$$

$$\begin{gathered} z(0) = z_1, \quad z(t_{max}) = z_2 \\ z'(0) = v_{1z}, \quad z'(t_{max}) = v_{2z} \\ z''(0) = 0, \; z''(t_{max}) = 0 \end{gathered} \tag{4}$$

Based on quintic Hermite interpolation, the equation of *x(t)* is:

$$x(t) = a_5 t^5 + a_4 t^4 + a_3 t^3 + a_2 t^2 + a_1 t + a_0 \tag{5}$$

*y(t)* and *z(t)* is calculated by the same formula as (5). The coefficients can be calculated by calculating the six conditions for polynomial *x(t)* with the equation of *x(t)*, and the linear system (7) is the equations for solving the coefficients for *x(t)*, *y(t)*, *z(t)*.

$$\begin{gathered} x(0) = a_0 = x_1, \\ x'(0) = a_1 = v_{1x}, \\ x''(0) = a_2 = 0 \end{gathered} \tag{6}$$

$$\begin{bmatrix} t_{max}^3 & t_{max}^4 & t_{max}^5 \\ 3t_{max}^2 & 4t_{max}^3 & 5t_{max}^4 \\ 6t_{max} & 12t_{max}^2 & 20t_{max}^3 \end{bmatrix} * \begin{bmatrix} a_3 \\ a_4 \\ a_5 \end{bmatrix} = \begin{bmatrix} x_2 - x_1 - v_{1x}t_{max} \\ v_{2x} - v_{1x} \\ 0 \end{bmatrix} \tag{7}$$

The $t_{max}$ of the Point 1 and Point 2 is calculated by (8):

$$\begin{gathered} Distance = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2} \\ t_{max} = \frac{Distance}{v} \end{gathered} \tag{8}$$

where $(x_1, y_1, z_1)$ is the coordinate of the start point P1, $(x_2, y_2, z_2)$ is coordinate of the destination point P2. *v* is the constant speed.

With autonomous navigation and the defined waypoints, UAVs can perform aerial photography and environmental monitoring of the specified locations with equipping the camera. Additionally, costs of an optimal path can be affected by flight time, flight height, path length, energy consumption, the risk of detection, and danger zones [5, 11]. UAVs can maintain at a certain altitude to reduce the possibility of collisions with obstacles [6]. The flight path with minimal risk is usually related to a longer flight path length [5].

### B. *Implementation of the Algorithm*

We implement the algorithm using the following process. The waypoints are recorded in the Waypoint-Matrix. The algorithm uses iterations to calculate the path to get a smooth path curve. To calculate the flight time, the Euclidean distance of the waypoints is calculated, then using distance to get $t_{max}$. For getting more precise trajectory, *t* is set to from 0 to $t_{max}$ with line space of 0.01s. *t* is being input to finding the value of $\begin{bmatrix} a_3 \\ a_4 \\ a_5 \end{bmatrix}$, and with the value of the coefficients, the path segment can be created. The points of the path are represented in the same way as the Waypoint-Matrix:

$$\begin{bmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \\ \vdots & \vdots & \vdots \\ x_n & y_n & z_n \end{bmatrix}$$

The algorithm is described in Figure 1, Figure 2, and Figure 3.

Figure 1 shows the algorithm of generating the path, and the matrix imports the waypoints. M is getting the size of the matrix, then set the values of X1, Y1, Z1, X2, Y2, Z2 by locating the value within the matrix. Call the *GetTime* function to calculate $t_{max}$ of each path segment. The *GetTime* function calls *GetPathSegments* function to get the positions. Using an empty matrix to store the positions of each path segment. The algorithm involves iterations to get path segments of every two waypoints.

**Algorithm 1** GenerationOfPath

```
procedure PATH($matrix$)
    $M \leftarrow size(matrix, 1)$
    $KeyPoints \leftarrow []$
    if $M > 1$ then
        for $i \leftarrow 1 : M - 1$ do
            $X1 \leftarrow matrix(i, 1)$
            $Y1 \leftarrow matrix(i, 2)$
            $Z1 \leftarrow matrix(i, 3)$
            $X2 \leftarrow matrix(i + 1, 1)$
            $Y2 \leftarrow matrix(i + 1, 2)$
            $Z2 \leftarrow matrix(i + 1, 3)$
            $Points \leftarrow GetTime(X1, Y1, Z1, X2, Y2, Z2, V1X, V2X, V1Y, V2Y, V1Z, V2Z)$
            $SizeOfPoints \leftarrow size(Points, 1)$
            if $i < 2$ then
                $KeyPoints \leftarrow [Keypoints; Points]$
            else
                if $SizeOfPoints = 2$ then
                    $Points \leftarrow Points(2, :)$
                    $KeyPoints \leftarrow [Keypoints; Points]$
                else
                    $Points \leftarrow Points(2 : SizeOfPoints, :)$
                    $KeyPoints \leftarrow [Keypoints; Points]$
```

Fig. 1. Path generation algorithm

Figure 2 presents the algorithm of getting time. The values of *X1*, *Y1*, *Z1*, *X2*, *Y2*, *Z2* are passed by the parameters, and the constant speed *V* is input as 18. Calculate the Euclidean distance between two waypoints, then calculate $t_{max}$ by using the distance to divide the constant speed. Then get the max value of *t*, and set the line space as 0.01. Call *GetPathSegments* function to get the positions of the path segment between every two waypoints.

**Algorithm 2** GetTime

```
procedure GETTIME($X1, Y1, Z1, X2, Y2, Z2, V1X, V2X, V1Y, V2Y, V1Z, V2Z$)
    $Equ \leftarrow (X2 - X1)^2 + (Y2 - Y1)^2 + (Z2 - Z1)^2$
    $Distance \leftarrow sqrt(Equ)$
    $tmax \leftarrow Dis/V$
    $t \leftarrow 0 : 0.01 : ceil(tmax)$
    $PathSegments \leftarrow GetPathSegments(t, X1, Y1, Z1, X2, Y2, Z2,$
                                $V1X, V2X, V1Y, V2Y, V1Z, V2Z)$
```

Fig. 2. Time generation algorithm

Figure 3 presents the algorithm of getting path segments. The values of *X1*, *Y1*, *Z1*, *X2*, *Y2*, *Z2* are passed by the parameters, and *V1X*, *V2X*, *V1Y*, *V2Y*, *V1Z*, *V2Z* are input as 18 m/s. Calculate the coefficients between two waypoints, then calculate points of the path segment by the coefficients. Then storing the positions of the path segment in a matrix.

**Algorithm 3** GetPathSegments

```
procedure GETPATHSEGMENTS($t, X1, Y1, Z1, X2, Y2, Z2, V1X, V2X, V1Y, V2Y, V1Z, V2Z$
    $A \leftarrow [t(end)^3 \quad t(end)^4 \quad t(end)^5;$
              $3 * t(end)^2 \quad 4 * t(end)^3 \quad 5 * t(end)^4;$
              $6 * t(end) \quad 12 * t(end)^2 \quad 20 * t(end)^3];$
    $XB \leftarrow [X2 - X1 - (V1X. * t(end)); V2X - V1X; 0];$
    $x \leftarrow A \backslash XB$
    $Xpoints \leftarrow x(3, 1) * t^5 + x(2, 1) * t^4 + x(1, 1) * t^3 + V1 * t + X1$
    $YB \leftarrow [y2 - y1 - (V1Y. * t(end)); V2Y - V1Y; 0];$
    $y \leftarrow A \backslash YB$
    $Ypoints \leftarrow y(3, 1) * t^5 + y(2, 1) * t^4 + y(1, 1) * t^3 + V1 * t + Y1$
    $ZB \leftarrow [z2 - z1 - (V1Z. * t(end)); V2Z - V1Z; 0];$
    $z \leftarrow A \backslash ZB$
    $Zpoints \leftarrow z(3, 1) * t^5 + z(2, 1) * t^4 + z(1, 1) * t^3 + V1 * t + Z1$
    $Points \leftarrow []$
    for $i \leftarrow 0 : size(t, 2) - 1$ do
        $Points \leftarrow [points; Xpoints(i + 1) Ypoints(i + 1) Zpoints(i + 1)]$
```

Fig. 3. Path segments generation algorithm

## C. *Cost functions*

A generated path is evaluated by the cost function, and it is optimal if it is with minimal costs. Based on the research [8], each value of cost function can be defined within [0, 1] by a fraction. The total costs add all values to compare the paths.

The costs function $f_{cost}$ is defined as:

$$f_{cost} = f_{length} + f_{time} + f_{altitude} + f_{collision} \tag{9}$$

where $f_{length}$ is the cost function of the path length, $f_{time}$ is the cost function of flight time, $f_{altitude}$ is the cost function of mean altitude, $f_{collision}$ is the cost function of collision.

### 1) *Path length*

The cost function $f_{length}$ for path length is developed as follows:

$$f_{length} = 1 - \frac{\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}}{\sum_{i=1}^{n} \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2 + (z_{i+1} - z_i)^2}} \tag{10}$$

where $(x_1, y_1, z_1)$ is the start point, $(x_2, y_2, z_2)$ is the destination point, and $n$ is the number of the points of the path segment. $\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}$ is the length of the straight line between Point 1 and Point 2. $\sum_{i=1}^{n} \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2 + (z_{i+1} - z_i)^2}$ is the actual cost of path length.

### 2) *Flight time*

The cost function $f_{time}$ for flight time is developed as follows:

$$ActualT = \sum_{i=1}^{n} \frac{\sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2 + (z_{i+1} - z_i)^2}}{v_{avg}} \tag{11}$$

$$Time = \frac{\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}}{v_0} \tag{12}$$

$$f_{time} = 1 - \frac{Time}{ActualT} \tag{13}$$

where $ActualT$ is the actual cost of time when going through the path, and $v_{avg}$ is the average velocity during the flight. $Time$ is the estimated cost of going through the straight line between Point 1 and Point 2, and $v_0$ is the initial velocity.

### 3) *Altitude*

The cost function $f_{altitude}$ is proposed as:

$$f_{altitude} = 1 - \frac{\sqrt{(z_2 - z_1)^2}}{\sum_{i=1}^{n} \sqrt{(z_{i+1} - z_i)^2}} \tag{14}$$

where $z_i$ is the altitude of the point located in the path. $\sqrt{(z_2 - z_1)^2}$ is the altitude difference between Point 1 and 2, and $\sum_{i=1}^{n} \sqrt{(z_{i+1} - z_i)^2}$ is the sum of the altitude difference between pairs of points in the path.

### 4) *Collison*

The cost function $f_{collision}$ for collision is as Equation (15).

$$f_{collision} = \frac{\sum_{c=1}^{cn} \sqrt{(x_{c+1} - x_c)^2 + (y_{c+1} - y_c)^2 + (z_{c+1} - z_c)^2}}{\sum_{i=1}^{n} \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2 + (z_{i+1} - z_i)^2}} \tag{15}$$

where $cn$ is the number of points that have collisions with the terrain or objects, and $\sum_{c=1}^{cn}\sqrt{(x_{c+1}-x_c)^2+(y_{c+1}-y_c)^2+(z_{c+1}-z_c)^2}$ is the sum of the path segments which hit the terrain.

## IV. Simulation Results

To validate our approach, we use simulation to compute UAV paths and calculate cost functions for different path designs. Simulation of the algorithm was implemented with Matlab. The constant flight speed is set to 18 m/s. Because the safety margin for UAVs is 50 m above the ground [1], so the path should be close to the altitude.

Figure 4 shows the different generated paths with the same start point and the destination. The waypoints are marked by "*".

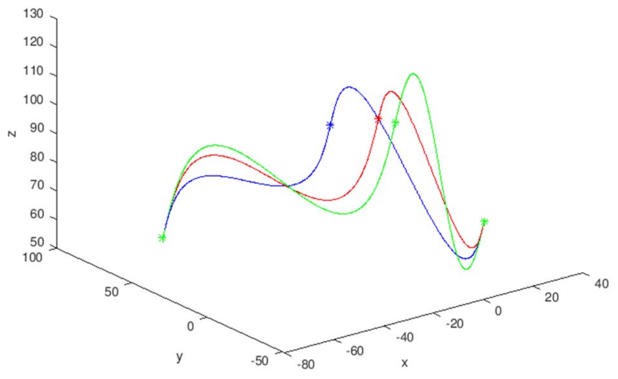

Fig. 4. The generated paths

Figure 5 integrated the paths with the terrain, and it uses different colors to indicate the different path. It presents the paths from a three-dimensional view.

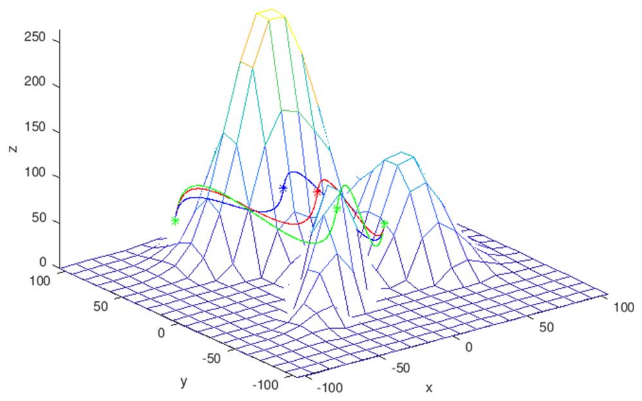

Fig. 5. The generated paths from the side viewpoint

Figure 6 shows another view of the paths. Three paths passed different waypoints. Different colors mark the different altitudes of the mountain.

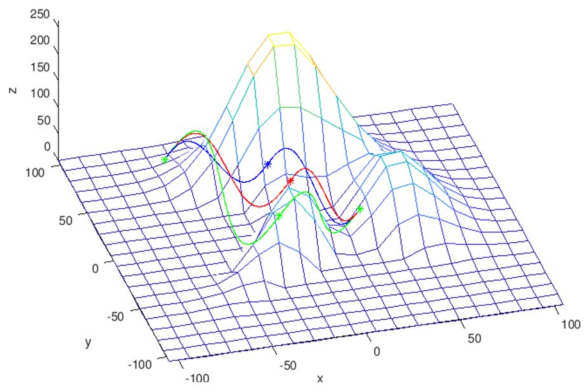

Fig. 6. The generated paths from the top viewpoint

Figure 7 shows a two-dimensional view of the generated paths.

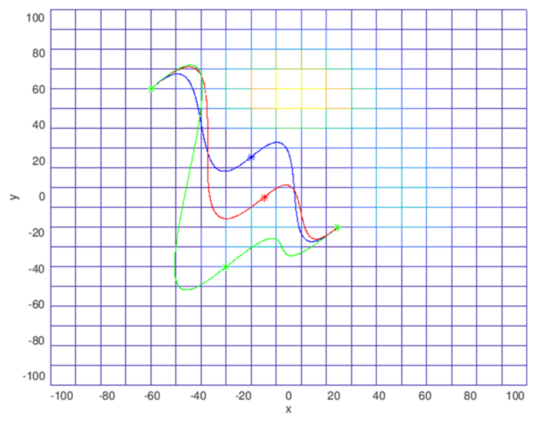

Fig. 7. The 2D view of the generated path

Figure 8 shows the path which is generated by the RRT* algorithm with obstacle avoidance. It is used to make a comparison with the quintic Hermite interpolation approach. The path is marked by red. It is more time-consuming than the quintic Hermite interpolation apporach when generating the path.

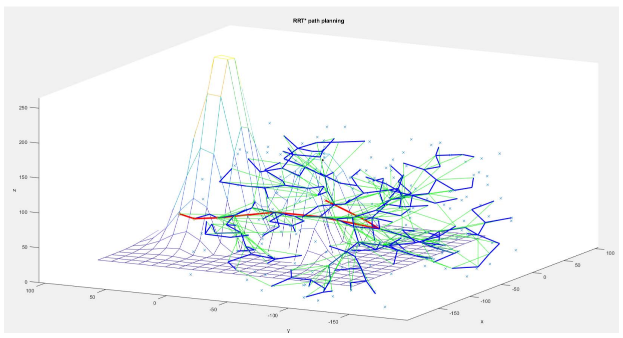

Fig. 8. The RRT* generated path from the side viewpoint

From the results of Figure 5, Figure 6, Figure 7, and Figure 8, there is not any collision.

The paths are evaluated by the costs function, as shown in Table II.

TABLE II. COSTS FUNCTION

| Costs | Blue Path | Red Path | Green Path | RRT* Path |
|---|---|---|---|---|
| $f_{length}$ | 0.4726 | 0.5218 | 0.6452 | 0.6272 |
| $f_{time}$ | 0.9986 | 0.9989 | 0.9994 | 0.9184 |
| $f_{altitude}$ | 0.9526 | 0.9549 | 0.9667 | 0.9075 |
| $f_{collision}$ | 0 | 0 | 0 | 0 |
| $f_{cost}$ | 2.4239 | 2.4755 | 2.6113 | 2.4531 |

From the comparisons of cost values among four paths, the path with minimal cost is the Blue Path. The minimal cost is 2.4239.

## V. CONCLUSION

The 3D path planning algorithm considers the terrain and protects the paths from collisions. The algorithm implements quintic Hermite interpolation for planning the path. To generate the path segments, iterations and matrixes are involved. The waypoints are used to improve the possibility of successful navigation, and they are defined in the specific order. The paths are required to pass these waypoints to reach the destination. The cost function is proposed by this paper, and each cost is limited to the range of [0, 1], then sum the costs up to get the total cost. The cost function considers path length, time, altitude and collision. This paper can be used for UAV 3D path planning. The future work should more recognize the dynamic and the kinematic behaviors of UAVs, and automatically generated the waypoints.

## REFERENCES

[1] L. Babel, "Three-dimensional route planning for unmanned aerial vehicles in a risk environment," J Intell Robot Syst, vol. 71, pp. 255-269, 2013.

[2] J. Verbeke et al., "A constraint-based flight control system architecture for UAVs using the iTaSC framework," 2016 International Conference in Unmanned Aircraft Systems (ICUAS), pp. 310-319, Jun. 2016.

[3] L. Babel, "Flight path planning for unmanned aerial vehicles with landmark-based visual navigation," Robotics and Autonomous Systems, vol. 62, pp. 142-150, 2014.

[4] J. Jamieson, and J. Biggs, "Path planning using concatenated analytically-defined trajectories for quadrotor UAVs," Aerospace, vol. 2, pp. 155-170, 2015.

[5] L. Babel, "Trajectory planning for unmanned aerial vehicles: a network optimization approach," Math Meth Oper Res, vol. 74, pp. 343-360, 2011.

[6] M. Pelosi, C. Kopp, and M. Brown, "Range-limited UAV trajectory using terrain masking under radar detection risk," Applied Artificial Intelligence, vol. 26, pp. 743-759, 2012.

[7] J. S. Bellingham, M. Tillerson, M. Alighanbari, and J. P. How, "Cooperative path planning for multiple UAVs in dynamic and uncertain environments," Proceedings of the 41st IEEE Conference on Decision and Control, pp. 2816-2822, 2002.

[8] V. Roberge, M. Tarbouchi, and G. Labonte, "Comparison of parallel genetic algorithm and particle swarm optimization for real-time UAV path planning," IEEE transactions on industrial informatics, vol. 9, no. 1, pp. 132-141, Feb. 2013.

[9] B. M. Sathyaraj, L. C. Jain, A. Finn and S. Drake, "Multiple UAVs path planning algorithms: a comparative study," Fuzzy Optim Decis Making, vol. 7, pp. 257-267, 2008.

[10] C. Wang, P. Liu, T. Zhang, and J. Sun, "The adaptive vortex search algorithm of optimal path planning for forest fire rescue UAV," 2018 IEEE 3rd Advanced Information Technology, Electronic and Automation Control Conference, pp. 400-403, 2018.

[11] L. Li, J. Wu, Y. Xu, J. Che, and J. Liang, "Energy-controlled optimization algorithm for rechargeable unmanned aerial vehicle network," 2017 12th IEEE Conference on Industrial Electronics and Applications (ICIEA), pp. 1337-1342, Jun. 2017.

[12] X. Yang, M. Ding and C. Zhou, "Fast marine route planning for UAV using improved sparse A* algorithm," 2010 Fourth International Conference on Genetic and Evolutionary Computing, pp. 190-193, 2010.

[13] Y. Wang, and X. Zeng, "Three dimensional unmanned aerial vehicle path planning by a continuous optimization method," Proceedings of the 34th Chinese Control Conference, pp. 2379-2383, Jul. 2015.

[14] B. Meng, and X. Gao, "UAV path planning based on bidirectional sparse A* search algorithm," 2010 International Conference on Intelligent Computation Technology and Automation, pp. 1106-1109, 2010.

[15] T. Bandyophadyay, L. Sarcione, and F.S. Hover, "A simple reactive obstacle avoidance algorithm and its application in Singapore harbor," Field and Service Robotics. Ed. Andrew Howard, vol. 62, pp. 455–465, 2010.

[16] L. Liu, R. Shi, S. Li, and J. Wu, "Path planning for UAVS based on improved Artificial Potential Field Method through changing the repulsive potential function," Proceedings of 2016 IEEE Chinese Guidance, Navigation and Control Conference, pp. 2011-2015, Aug. 2016.

[17] Y. Wang, L. Guo, Q. Li, and X. Guan, "Comparison of three different methods for terrain modeling in parabolic equation", 2017 International Applied Computational Electromagnetics Society Symposium (ACES), pp. 1-2, 2017.

[18] I. Sayim, T. Kavzoglu, and E. K. Sahin, "GBAS availability analysis for Trabzon airport using true terrain masking data," 7th International Conference on Recent Advances in Space Technologies, pp. 1-5, 2015.

[19] I. Khademi, B. Maleki, and A. N. Mood, "Optimal three dimensional terrain following/terrain avoidance for aircraft using direct transcription method," 19th Mediterranean Conference on Control and Automation, pp. 254-258, Jun. 2011.

[20] D. Lee and D. H. Shim, "Spline-RRT* based optimal path planning of terrain following flights for fixed-wing UAVs," The 11th International Conference on Ubiquitous Robots and Ambient Intelligence, pp. 257-261, Nov. 2014.

[21] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," IEEE transactions of systems science and cybernetics, vol. ssc-4, no. 2, pp. 100-107, Jul. 1968.

[22] T. W. McLain, P. R. Chandler, S. Rasmussen, and M. Pachter, "Cooperative control of UAV rendezvous," Proceedings of the American Control Conference, pp. 2309-2314, 2001.

[23] P. B. Sujit, and R. Beard, "Multiple UAV path planning using anytime algorithms," 2009 American Control Conference, pp. 2978-2983, 2009.