The 11th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI 2014)
Nov. 12-15, 2014 at Double Tree Hotel by Hilton, Kuala Lumpur, Malaysia

Spline-RRT* Based Optimal Path Planning of Terrain Following Flights for Fixed-Wing UAVs

Dasol Lee[1] and David Hyunchul Shim[2]
[1]Department of Aerospace Engineering, Korea Advanced Institute of Science and Technology, Daejeon, 305-701, Korea
(Tel: +82-42-350-3764; E-mail: dasol@kaist.ac.kr)
[2]Department of Aerospace Engineering, Korea Advanced Institute of Science and Technology, Daejeon, 305-701, Korea
(Tel: +82-42-350-3724; E-mail: hcshim@kaist.ac.kr)

Abstract - This paper describes a path planning algorithm for fixed-wing UAV flights that follow the local terrain. The proposed algorithm utilizes a spline-RRT* planner in which the tree structure is extended using a spline method to generate smooth paths without any post-processing. In addition, a cost function ensures paths are sufficiently far from several hazardous positions and close to the surface of the local terrain. Therefore, the resulting paths are geometrically and dynamically feasible for terrain following flights, and are asymptotically optimal. A series of simulations demonstrates the successful utilization of the algorithm in a path planning application for terrain following flights for fixed-wing UAVs.

Keywords - Optimal Path Planning, Terrain Following Flight, Fixed-Wing UAV, RRT*, Spline-RRT*

1. Introduction

Unmanned Aerial Vehicles (UAVs) are currently employed in a range of missions, with terrain following flights being one of their most important applications. Therefore, many path or trajectory planning algorithms have been proposed. For example, an optimal three dimensional terrain following and collision avoidance algorithm for an aircraft has been reported [1]. However, this study did not consider hazardous obstacles on the given terrain. The integrated terrain following algorithm combines path and trajectory planning, guidance and control, and navigation [2], but is limited by the fact that the lateral and longitudinal paths are calculated separately in the path planning phase.

To overcome this weakness, we have considered sampling-based path planning algorithms, which are widely used in robotics research. These algorithms, such as the rapidly exploring random tree (RRT) [3] and its RRT* variant [4], can generate paths efficiently in highly constrained planning problems. In this study, we utilize the spline-RRT* algorithm [5] to generate terrain following flight paths. This approach is validated by several simulations, and produces asymptotically optimal results.

The rest of this paper is organized as follows. Section 2 describes the spline-RRT* algorithm, and a cost function is derived in Section 3. Simulation results are presented in Section 4, and we give our conclusions in Section 5.

2. The Spline-RRT* Algorithm

This section describes the spline-RRT* algorithm, which is the basis of path generation for terrain following UAV flights.

2.1 Overview of the Spline-RRT* Algorithm

The spline-RRT* algorithm is an asymptotically optimal path planner, and is based on the RRT* algorithm. A spline technique is employed in the proposed algorithm to generate smooth paths, which are especially important for fixed-wing UAVs. A dynamic feasibility check and a geometric collision check are performed during the tree extension, ensuring that the resultant paths are both geometrically and dynamically feasible, as well as being asymptotically optimal.

2.2 Algorithm 1: Extend Spline-RRT*

To extend the tree structure over the workspace, we utilize an **Extend Spline-RRT*** function. The pseudo code of this function is shown in **Algorithm 1**, and its important sub-functions are defined as follows.

GetAngle($\mathbf{v_1}$,$\mathbf{v_2}$): This function returns the angle between vectors $\mathbf{v_1}$ and $\mathbf{v_2}$.

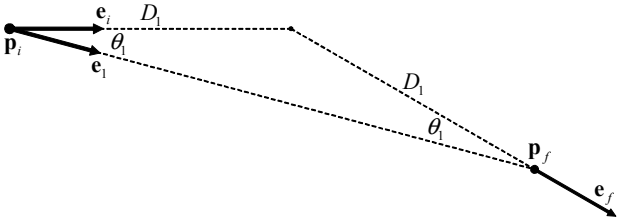

Fig. 1. Schematic for sub-functions **GetDirection** and **GeometricallyFeasible**

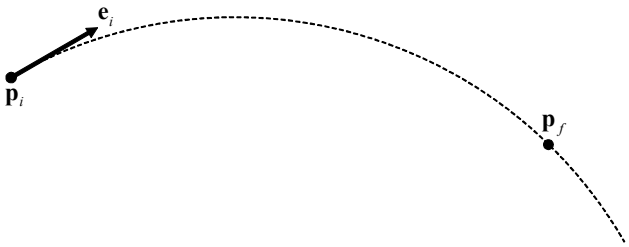

Fig. 2. Schematic for a sub-function, **DynamicallyFeasible** The dotted line is part of a circle which is centered at $\mathbf{p}_c$ with a radius R_{turn}.

GetDirection($\mathbf{x_1}$,$\mathbf{v_2}$): This function returns the unit direction vector $\mathbf{e}_f$ (see Fig. 1). $\mathbf{e}_f$ is obtained by rotating the vector $\mathbf{e}_1$ by an angle of θ_1 around the axis $\mathbf{e}_{axis}$. The input arguments $\mathbf{x_1}$ and $\mathbf{v_2}$ are $\{(\mathbf{p}_i, \mathbf{e}_i)\}$ and $\mathbf{p}_f$, respectively. The vectors $\mathbf{e}_1$ and $\mathbf{e}_{axis}$ are defined as:

$$\mathbf{e}_1 = \frac{\mathbf{p}_f - \mathbf{p}_i}{\left|\mathbf{p}_f - \mathbf{p}_i\right|}, \qquad \mathbf{e}_{axis} = \frac{\mathbf{e}_i \times (\mathbf{p}_f - \mathbf{p}_i)}{\left|\mathbf{e}_i \times (\mathbf{p}_f - \mathbf{p}_i)\right|} \tag{1}$$

Algorithm 1: Extend Spline-RRT* ($\mathbf{G}$,$\mathbf{p}_{rand}$)

```
01: V' ← V;  E' ← E;  c_new,optimal ← ∞;
02: x_nearest ← Nearest (G, p_rand);
03: p_new ← Steer (x_nearest, p_rand);
04: X_near ← Near (G, p_new, |V|);
05: for all x_near ∈ X_near do
06:   if DynamicallyFeasible (x_near, p_new) then
07:     e_new ← GetDirection (x_near, p_new);
08:     x_new ← {(p_new, e_new)};
09:     if GeometricallyFeasible (x_near, x_new) then
10:       c_new,cur ← Cost (x_init, x_new); // Parent node of x_new is x_near
11:       if c_new,cur < c_new,optimal then
12:         x_min ← x_near;  x_new,optimal ← x_new;
13:         c_new,optimal ← c_new,cur;
14:       end if
15:     end if
16:   end if
17: end for
18: V' ← V' ∪ {x_new,optimal};  E' ← E' ∪ {(x_min, x_new,optimal)};
19: for all x_near ∈ X_near \ {x_min} do
20:   if DynamicallyFeasible (x_new,optimal, p_near) then
21:     e_near,ref ← GetDirection (x_new,optimal, p_near);
22:     if GetAngle (e_near,ref, e_near) < θ_allow then
23:       if GeometricallyFeasible (x_new,optimal, x_near) then
24:         c_near,cur ← Cost (x_init, x_near);
25:         c_near,new ← Cost (x_init, x_near); // x_new,optimal's child is x_near
26:         if c_near,new < c_near,cur then
27:           x_parent ← Parent (x_near);
28:           E' ← E' \ {(x_parent, x_near)};
29:           E' ← E' ∪ {(x_new,optimal, x_near)};
30:         end if
31:       end if
32:     end if
33:   end if
34: end for
35: return G' = (V', E');
```

DynamicallyFeasible($\mathbf{x_1}$,$\mathbf{v_2}$): This function returns a Boolean. In the example shown in Fig. 2, **TRUE** is returned if R_{turn} is greater than the minimum turning radius $R_{\min}$ of the target UAV. Otherwise, **FALSE** is returned. $\{(\mathbf{p}_i, \mathbf{e}_i)\}$ and $\mathbf{p}_f$ are the inputs $\mathbf{x_1}$ and $\mathbf{v_2}$, respectively. R_{turn} is calculated as:

$$R_{turn} = \left|\mathbf{p}_i - \mathbf{p}_c\right| \tag{2}$$

$$\begin{bmatrix} \mathbf{e}_i \\ 2(\mathbf{p}_i - \mathbf{p}_f) \\ \mathbf{e}_i \times (\mathbf{p}_f - \mathbf{p}_i) \end{bmatrix} \mathbf{p}_c^{\,T} = \begin{bmatrix} \mathbf{e}_i \bullet \mathbf{p}_i \\ |\mathbf{p}_i|^2 - |\mathbf{p}_f|^2 \\ (\mathbf{e}_i \times (\mathbf{p}_f - \mathbf{p}_i)) \bullet \mathbf{p}_i \end{bmatrix} \tag{3}$$

GeometricallyFeasible($\mathbf{x_1}$,$\mathbf{x_2}$): This is another Boolean function. If the path segment $\mathbf{B}(s)$ is within a geometrically collision-free space, the function returns a **TRUE** value. Otherwise, it returns **FALSE**. $\{(\mathbf{p}_i, \mathbf{e}_i)\}$ and $\{(\mathbf{p}_f, \mathbf{e}_f)\}$ generate the input arguments, $\mathbf{x_1}$ and $\mathbf{x_2}$, respectively, as represented in Fig. 1. Equations (4)-(6) calculate path segment $\mathbf{B}(s)$, where s is a real number in the interval [0, 1].

$$\mathbf{B}(s) = (1-s)^3 \mathbf{p}_i + 3s(1-s)^2 \mathbf{p}_1 + 3s^2(1-s)\mathbf{p}_2 + s^3 \mathbf{p}_f \tag{4}$$

$$\mathbf{p}_1 = \mathbf{p}_i + \mathbf{e}_i \left|\mathbf{p}_f - \mathbf{p}_i\right| / 3 \tag{5}$$

$$\mathbf{p}_2 = \mathbf{p}_f - \mathbf{e}_f \left|\mathbf{p}_f - \mathbf{p}_i\right| / 3 \tag{6}$$

Cost($\mathbf{x_a}$,$\mathbf{x_b}$): This function evaluates the cost of the transition from vertex $\mathbf{x_a}$ to vertex $\mathbf{x_b}$. The function is designed to generate paths that are suitable for terrain following flights. Further details on this function are given in Section 3.

Algorithm 1 can be largely divided into two parts. Lines 1-18 relate to the selection of a parent vertex that minimizes the cost of $\mathbf{x}_{new}$, and lines 19-34 describe a re-wiring process. This process aims to lower the cost of $\mathbf{x}_{near}$ by replacing its parent vertex with $\mathbf{x}_{new,optimal}$, which is the newly extended vertex in line 18 of **Algorithm 1**.

3. Cost Function for Terrain Following Flights

The spline-RRT* algorithm returns the minimum cost path in the existing tree structure. This cost decreases as the number of vertices is increased. Therefore, designing a suitable cost function is a very important process in the generation of paths intended for terrain following flights. An efficient path for a terrain following flight stays close to the terrain surface, but away from hazardous points such as Surface-to-Air Missile (SAM) sites.

The cost function given in Eq. (7) considers these two factors; see Fig. 5 for a diagrammatic explanation. Figure 6 shows a cost graph in which only hazardous points are considered to generate this graph.

$$\mathbf{Cost}(\mathbf{x}_a,\mathbf{x}_b)=\left(\sum_{j=a}^{b-1} f_{c,e}(\mathbf{x}_j,\mathbf{x}_{j+1})\right)/\ \mathbf{PathLength}(\mathbf{x}_a,\mathbf{x}_b)$$
$$f_{c,e}(\mathbf{x}_j,\mathbf{x}_{j+1})=\sum_{k=1}^{nE}\left[\left(\sum_{l=1}^{nS}\frac{w_1}{w_2+\left|\mathbf{p}_{e,k}-\mathbf{p}_{S,l}\right|}\right)+\mathbf{RelAlt}(\mathbf{p}_{e,k})\right] \quad (7)$$

where subscripts a and b represent the tree depth of the corresponding vertex, w_1 and w_2 are weighting factors, sub-function **RelAlt** returns the relative altitude between the position of input argument $\mathbf{p}_{e,k}$ and the terrain surface, and sub-function **PathLength** calculates the path length from $\mathbf{x_a}$ to $\mathbf{x_b}$. Total number of SAM sites on the given terrain is denoted by nS, and nE is the number of times the cost calculation is executed, as shown in Fig. 5. $\mathbf{p}_{e,k}$ denotes the position of the k-th cost evaluation on a given single edge, and $\mathbf{p}_{S,l}$ is the position of the l-th SAM site, as represented in Fig. 5.

The proposed cost function ensures that the resulting paths become closer to the terrain surface while remaining an appropriate distance from the hazardous points.

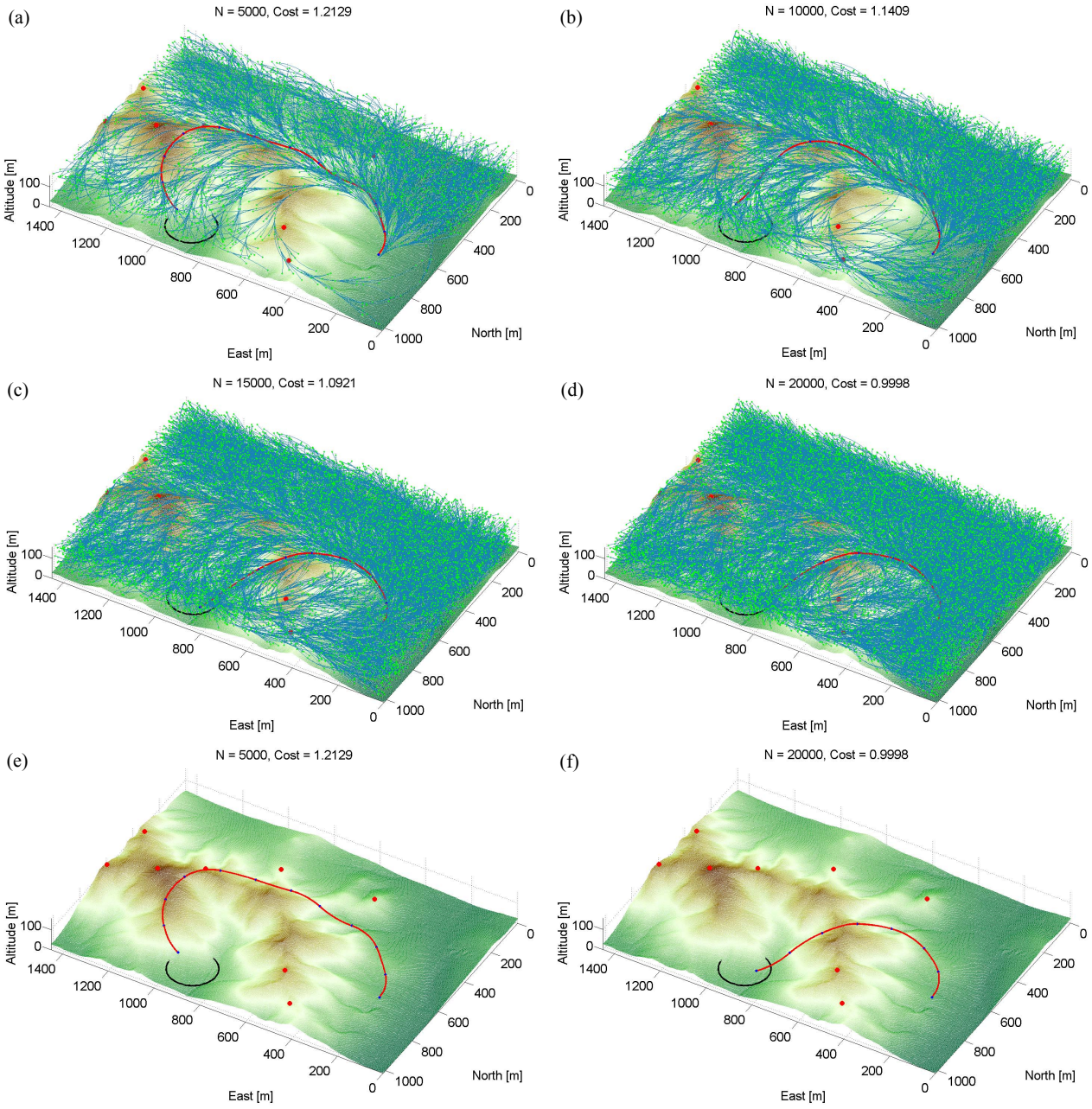

Fig. 3. Generated paths for terrain following flight

Red lines show the resulting path, the blue points are knots, the black circle centered at the goal position with a radius of R_{goal} denotes the goal area, and the red points indicate hazardous points such as SAM sites. Subfigures (a)–(d) show the vertices and edges of the tree structure with the resulting path, and subfigures (e)–(f) show only the resulting path for clarity. The number of graph vertices and the cost of the resulting path are (a) 5000 and 1.2129, (b) 10000 and 1.1409, (c) 15000 and 1.0921, and (d) 20000 and 0.9998.

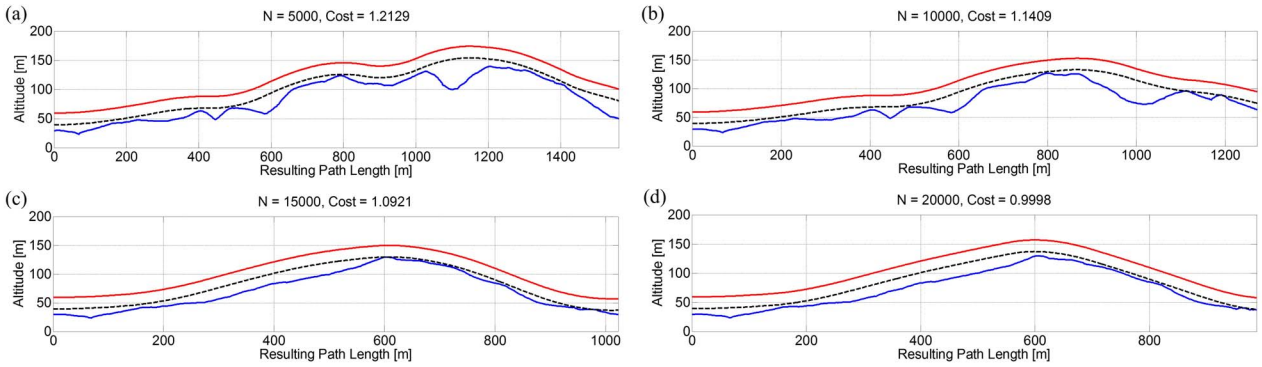

Fig. 4. Altitude graph of the generated paths

Blue lines represents the altitude of the terrain, the black dotted lines are the initial path height, and the red lines show the final altitude profile of the generated path after applying an altitude offset. Subfigures (a)–(d) show the altitude graphs when the number of vertices is (a) 5000, (b) 10000, (c) 15000, and (d) 20000.

4. Simulation

To evaluate the performance of the proposed algorithm, we conducted a series of simulations using spline-RRT* as the planning algorithm with the proposed cost function. The workspace for path planning was a scaled altitude map of real terrain.

The simulation parameters and values are summarized in Table 1, and the results are shown in Fig. 3 (generated paths) and Fig. 4 (altitude graph of the terrain and the generated paths). As shown in Fig. 3(a)–(d), the tree structure is three dimensionally extended over the workspace as the number of vertices increases. The path generated with 5000 vertices (see Fig. 3(e)) is not appropriate for terrain following flights, as it passes close to hazardous positions. When the number of vertices is increased to 20000 (see Fig. 3(f)), the generated path is sufficiently separated from the hazardous positions. This behavior is caused by the asymptotic optimality of the spline-RRT* and the proposed cost function. A cost graph related to SAM sites only is shown in Fig. 6. The altitude graph shown in Fig. 4 exhibits similar behavior. As the number of vertices increases, the altitude profile of the generated path becomes closer to the terrain surface. This can be seen by comparing Fig. 4(a) and (d). The red lines in Fig. 4 indicate the final altitude profile after applying an altitude offset.

Therefore, the proposed algorithm and cost function generate paths that are suitable for terrain following flight, are geometrically and dynamically feasible, and are asymptotically optimal.

Table 1. Simulation parameter values

Initial heading angle	180 [deg]
Initial flight path angle	0 [deg]
Initial position	(700, 200, -40) [m], NED
Goal position	(900, 900, -50) [m], NED
Altitude offset	20 [m]
nS and nE	8 and 5
w_1 and w_2	1500 and 75
R_{goal} and $R_{\min}$	100 and 185 [m]

5. Conclusion

In this paper, we have presented a planning algorithm for generating paths which are intended to terrain following flights of fixed-wing UAVs. The proposed algorithm utilizes the spline-RRT* and a cost function. The planner generates smooth paths that are geometrically and dynamically feasible, and are asymptotically optimal. Additionally, the proposed cost function ensures that the generated paths are sufficiently separated from hazardous positions in the given terrain. Simulations confirmed the efficacy of our approach, showing that this planning algorithm can be utilized to plan the flight paths of terrain following fixed-wing UAVs.

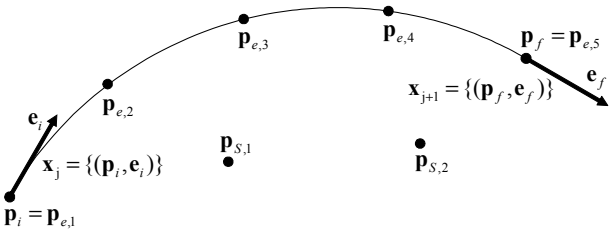

Fig. 5. Schematic of sub-function **Cost**

Only a single edge is illustrated in this figure. Here, nS and nE are 2 and 5, respectively.

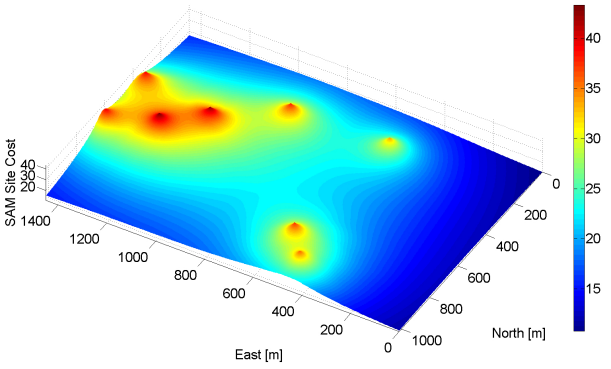

Fig. 6. SAM site related cost graph

In this graph, the cost values are calculated on the surface of the terrain, and relative altitude related cost is not considered.

Acknowledgement

The authors are grateful for the financial support provided by the Agency for Defense Development, funded by the Korean government (No. UE124026JD).

References

[1] T. Sharma, P. Williams, C. Bil, and A. Eberhard, "Optimal Three Dimensional Terrain Following and Collision Avoidance," Australia and New Zealand Industrial and Applied Mathematics Journal, Vol. 47, pp. 695-711, 2007.

[2] G. Oh, H. Kim, J. Seo, and Y. Kim, "Integrated Terrain Following Algorithm for UCAV," *International Conference on Control, Automation, and Systems*, Jeju Island, Korea, Oct. 17-21, 2012.

[3] S. M. LaValle and J. J. Kuffner, "Rapidly-exploring random trees: Progress and prospects," Algorithmic and Computational Robotics: New Directions, pp. 293-308, 2000.

[4] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," The International Journal of Robotics Research, Vol. 30, No. 7, pp. 846-894, 2011.

[5] D. Lee, H. Song, and D. H. Shim, "Optimal Path Planning Based on Spline-RRT* for Fixed-Wing UAVs Operating in Three-Dimensional Environments," *International Conference on Control, Automation, and Systems*, Gyeonggi-do, Korea, Oct. 22-25, 2014.