APPLICATION OF ADVANCED REAL-TIME RRT* AND INCREMENTAL BACKSTEPPING CONTROL FOR ROTARY-WING UNMANNED AIRCRAFT SYSTEMS

Jung Woo An, sunday3244@konkuk.ac.kr, Department of Aerospace and Information Engineering, Konkuk University (Korea)

Min Gyu Lim, mglim1995@konkuk.ac.kr, Department of Aerospace and Information Engineering, Konkuk University (Korea)

Yoo-Seung Choi, chldbtmd1234@konkuk.ac.kr, Department of Aerospace and Information Engineering, Konkuk University (Korea)

Ji Won Woo, jiwon.woo@lignex1.com, Aerospace/Drone R&D Lab, LIGnex1 (Korea)

Chang-Joo Kim, cjkim@konkuk.ac.kr, Department of Aerospace and Information Engineering, Konkuk University (Korea)

ABSTRACT

RRT is a well-known algorithm used for path and motion planning and exhibits good performance in generating a path quickly and efficiently. Many attempts were made to use these advantages for generating paths in real time; however, critical problems were observed when the algorithm was applied to aircraft. Unlike other mobile vehicles, aircraft require careful consideration of their limits, such as velocity, turn radius, and load factors. Only a few studies consider this; hence, in this study, we propose a method to consider important dynamic characteristics when planning paths in real time. The Advanced Real-Time RRT* algorithm suggested in this paper considers the limits by adopting the concept of “trajectory” as a new approach. A trajectory contains information on the location, velocity, and acceleration of an aircraft at each waypoint. If a trajectory is determined to be infeasible for flight, a different path is created and the path is converted to a new trajectory. In addition, the proposed algorithm was combined with a nonlinear controller called the incremental backstepping control, which is a robust controller having an excellent trajectory tracking performance. For validating the system, a series of simulations were conducted using a high-fidelity rotorcraft mathematical model as a system plant. Simulation results show that the proposed algorithm is sufficient for generating paths in a dynamic environment.

1. INTRODUCTION

Unmanned aerial vehicles (UAVs) have attracted considerable attention because of their growth and utility. In accordance with this trend, many studies on UAVs have been actively conducted. A record of the first UAVs can be found in the biography of Archibald Montgomery Low[1], which describes the attacks on German Zeppelins using UAVs equipped with early radio technology[2]. UAVs have been used for military purposes; however, in recent years, many attempts to use them for civilian purposes have been made. In particular, rotary-wing UAVs are used in urban areas because they can take off and land vertically and fly at lower altitudes than that of fixed-wing aircraft. They are used in various fields such as construction sites, data collection, relay communication, and surface weather observation[3–5]. As the range of application of unmanned rotorcrafts widens, demands have been increasing for the development of a rotary-wing unmanned aircraft system (RUAS). The main function of the system is to achieve a degree of autonomy high enough to replace a pilot. The development of a RUAS that can perform this role has been ongoing for a long time, and in February 2022, the UH-60 (Blackhawk) finally succeeded in take-off, mission performance, path planning and replanning in real-time, and landing without a pilot[6]. The Defense Advanced Research Projects Agency (DARPA), which manages the unmanned UH-60 flight project, said that the autonomous software in UH-60 would make flying both smarter and safer with reduced pilot workloads. As stated by the DARPA, the development of a highly autonomous RUAS is required to reduce a pilot's workload. According to the classification proposed by Kendoul, it is important to improve the capabilities of the guidance, navigation, and control systems that comprise a RUAS[7]. Level 4 is the highest level of autonomy that an unmanned helicopter can achieve, allowing it to generate a path in real time and accurately follow the path using a robust control system.

Many studies have focused on generating paths in real time to achieve the highest level of autonomy.

However, they have only suggested technical methods and have not considered the physical properties of an aircraft. For example, the algorithm RT-RRT* replans and optimizes the path every time a mobile vehicle reaches a waypoint[8]. Because RRT* samples the vertices randomly, the length of the waypoint intervals is not constant. Therefore, if an aircraft is flying at a constant speed, the cycle of updating the path using RT-RRT* is not constant. In this study, we proposed a new algorithm to solve this problem: the advanced real-time RRT*, which uses the concept of trajectory. In this paper, we describe the way our proposed method updates the path during the same period and considers the physical characteristics of the aircraft. In addition, the performance of this algorithm was validated by integrating it with a flight control system consisting of incremental backstepping control (IBSC), and the high fidelity Bo-105 model was used to simulate the integrated system. Simulation results show that the proposed algorithm is appropriate for generating paths in real time and is applicable to dynamic environments.

2. PATH PLANNING ALGORITHMS

In this section, we explain digital terrains and path-planning algorithms using RRT to illustrate advanced real-time RRT*. First, we describe the composition of a digital environment in which algorithms can be implemented. In this study, we constructed a digital terrain using real terrain information[9]. Another way to implement a virtual digital environment is through a numerical method, such as radial basis functions. The environment that we created was static because mobile objects were absent; therefore, mobile obstacles were added to convert it into a dynamic environment. Next, we discuss the RRT* and line-of-sight path optimization (LOSPO) algorithms. Only a brief description on the RRT* algorithm is provided; a full description may be obtained from other studies(Please cite these studies[10,11]). We explain the LOSPO algorithm used to optimize the path in detail because it is the core of this study.

2.1. Digital Terrain

In this study, the environment used to simulate the path-planning algorithms was created by depicting real terrain data. The more complex the terrain, the easier it is to compare the performances of the algorithms. Therefore, the region near 37°25'20.2″ " N and 127°01'21.9"″ E, which contains many high and low mountains, was chosen. Figure 1 represents the terrain, where the path-planning algorithms covered in this study were simulated.

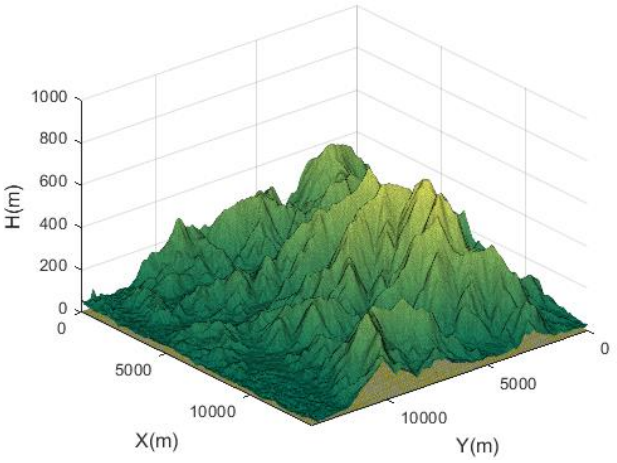

Figure 1. Digital terrain

2.2. RRT* Algorithm

Certain concepts must be introduced to define the path-planning problem in this study. We set the starting point as $\boldsymbol{x_{init}}$ and the RRT* tree stretches from the starting point to $\boldsymbol{x_{goal}}$, the final point. Mobile obstacles may appear within the configuration space or searching environment, and any node marked $\boldsymbol{x_{rand}}$ is randomly sampled to avoid these obstacles. If the $\boldsymbol{x_{rand}}$ node is sufficiently close to a node in the RRT* tree and can be connected without colliding with obstacles, then the $\boldsymbol{x_{rand}}$ node becomes the $\boldsymbol{x_{new}}$ node and is added to the tree. According to the RRT*rules, adding final nodes to a tree while expanding the tree can create a continuum from the initial node to the final node. Cost $\boldsymbol{C}$ of the path can be defined in various ways; in this study, the length of the path was set as the cost. Therefore, the optimal path is the shortest path that can be planned in the environment and the cost of this path is called $\boldsymbol{C_{opt}}$.

The RRT* algorithm is a modification of RRT, with the addition of “re-wiring” and “best parent selecting” processes. It is helpful to refer to LaValle’s paper to understand the RRT algorithm[12]. RRT* is an incremental sampling-based algorithm that finds the initial path rapidly and optimizes the path as the process occurs iteratively[10,11]. The process flow of RRT* is presented in Algorithm 1 and the techniques are explained in reference[13].

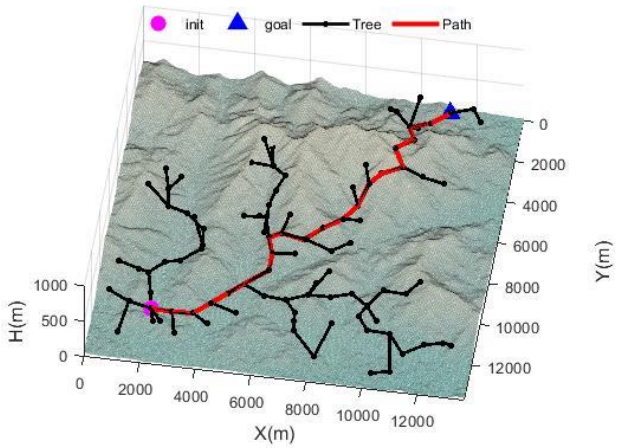

Figure 2. The result of path planning after applying RRT* (nodes: 100)

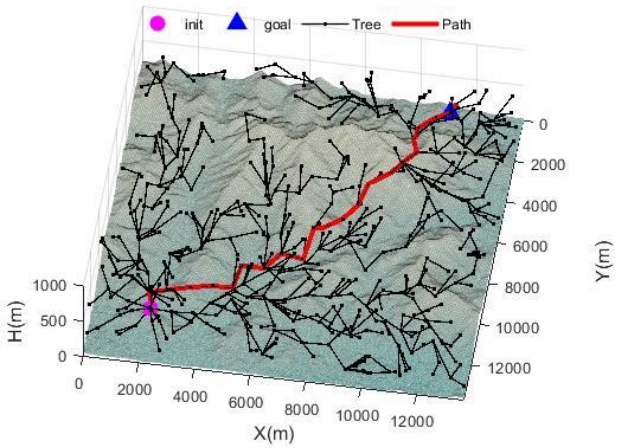

Figure 3. The result of path planning after applying RRT* (nodes: 500)

Algorithm 1	RRT*

$T \leftarrow \{x_{init}\}$

$for\ i = 1\ to\ n\ do$

  $x_{rand} \leftarrow RandomlySampling(i)$

  $x_{Near} \leftarrow Nearest(T, x_{rand})$

  $if CollisionFree(x_{rand}, x_{Near}) then$

    $if dist(x_{rand}, x_{Near}) > \rho_{Near} then$

      $x_{new} \leftarrow Attract(x_{rand}, x_{Near}, \rho_{Near})$

    $else$

      $x_{new} \leftarrow x_{rand}$

    $end\ if$

    $T \leftarrow ChooseBestParent(T, x_{new})$

    $T \leftarrow Rewire(T, x_{new})$

  $end\ if$

$end\ for$

$return\ T$

The path generated using RRT* did not collide with the terrain. One of the characteristics of the RRT* algorithm is that it can optimize the path when more nodes are added. The result of path planning when 100 nodes are added is shown in Figure 2; the cost of the path is 1.4035e+07. Figure 3 shows that when the number of sampled nodes is 500, the cost is 1.3593e+07, which is lower than the previously mentioned cost.

2.3. Line-of-Sight Path Optimization

Although RRT* can generate an optimal path through repetition, it requires the sampling of many nodes and is time consuming. To improve the efficiency of path optimization, optimization methods such as RRT*-smart have been developed[11,14]. However, because such methods also require sampled nodes to optimize the path, we developed a new method called LOSPO, which does not require random sampling of nodes and uses the initial path generated by the RRT* algorithm[15]. First, the path generated by RRT* is divided at regular intervals, as shown in Figure 4. The initial node then searches for a node to which it can connect in a straight line. The search starts at the final node and traverses the path in reverse order. If a node that can be connected to the initial node is identified, a new path is created by connecting the two nodes. The newly created path is divided again at regular intervals, and a node that can be connected to the second node is identified. While repeating this process, if any node is connected to the final node in a straight line, the iteration is stopped and an optimal trajectory is generated.

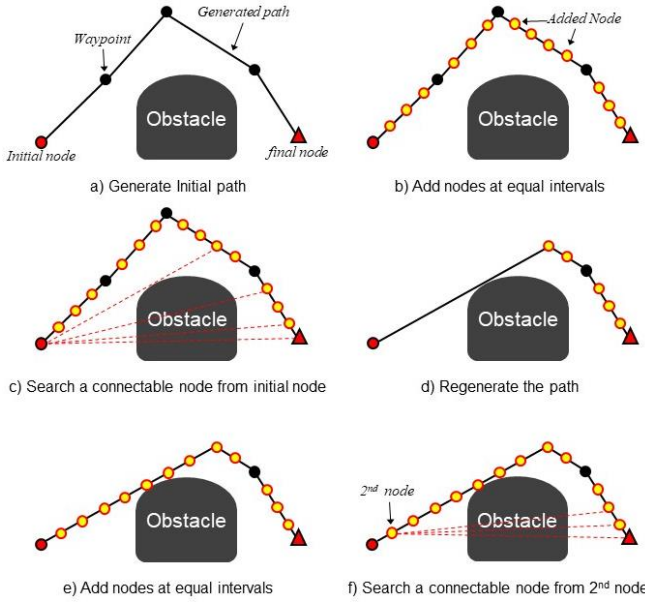

Figure 4. Process of LOSPO

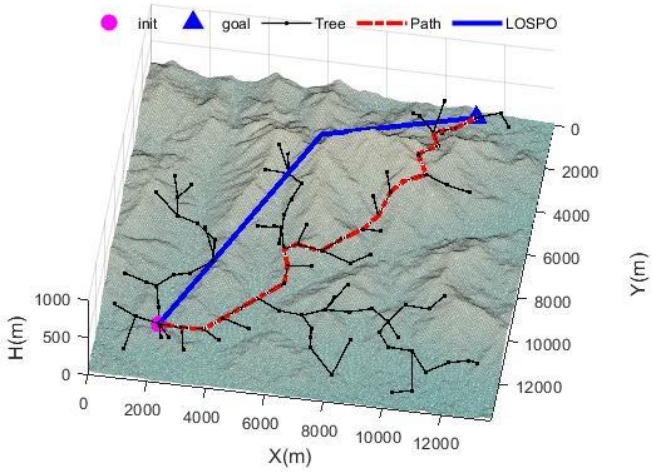

Figure 5. Result of path optimization applying LOSPO

In the LOSPO algorithm, the period taken for generating the optimal path can be shortened and the size of the stored data can be reduced because the location of the added node can be defined by the initial path and the number of nodes can be adjusted using the length of the interval. When the performances of RRT*, RRT*-smart, and LOSPO (combined with RRT*) are compared, LOSPO exhibits the best performance under the same conditions[15,16]. Therefore, the LOPSO method is suitable for solving the path-planning problem in real-time because it can generate the optimal path in a short time.

Figure 5 illustrates the results of applying LOSPO after the initial path was generated using RRT*. In this case, the cost of the path generated by LOSPO was 1.1740e+08, which is lesser than the cost of the initial path by 16.35%.

3. ADVANCED REAL-TIME RRT* ALGORITHM

Real-time path planning is a challenging problem faced by many robots and vehicles in dynamic environments [17]. During path planning, a general path is difficult to find in an unstructured environment; however, it can be easily obtained using a sampling-based path-planning algorithm[8]. Algorithms using A* and RRT* are effective for solving real-time path-planning problems. An RRT-based algorithm has the following advantages over the A* algorithm in solving this problem[18].

1) The RRT-based algorithm can be applied to general dynamic models as a sampling basis.

2) RRT-based algorithms can be easily applied to real-time online implementations while ensuring route optimization.

3) Constraints need not be set on the search environment, and each trajectory can be checked even with complex constraints.

Many real-time algorithms use RRT, such as RT-RRT, RRTX, and RRT* GL[8,19,20]. However, such algorithms have a critical limitation: they cannot react immediately to mobile obstacles or changes in the dynamic environment because they update the path generated previously only when the robot or vehicle arrives at the next waypoint. In other words, the move and update phases do not occur simultaneously but sequentially. The algorithms also assume that the robot and obstacle move to the next waypoint simultaneously, regardless of the distance between the waypoints. If this assumption is applied to a real-world scenario, moving objects must change their speed each time they pass waypoints. In terms of aircraft control, this requirement places a heavy burden on the pilot or flight control system and causes errors between the simulation and real conditions. To solve such problems, we present a method that can generate and replan paths in real time and can be applied in real environments.

The real-time path generation technique presented in this study uses the difference between the path and the trajectory. According to literature, a trajectory contains information about time, through which the exact position and velocity of an object can be known[21]. The manner in which a trajectory stores information about time is different from that about path; it is advantageous in missions where an aircraft timing is critical, such as rescue or delivery[15].

The $N+1$ waypoints created through the path-planning method are expressed in Equation (1).

(1) $$\boldsymbol{P}_j = \left(x_j, y_j, h_j\right)^T \Big|_{j=0}^{j=N}$$

Point $\boldsymbol{P}_j$ indicates the location of the j[th] point and has no information about time. First the aircraft is assumed to be flying at a constant speed of V_{ref}, and temporal information is temporarily assigned to each point. The time required for the aircraft to fly can be calculated using Equations (2) and (3) for each segment.

(2) $$t_0 = 0$$

(3) $$t_j = \frac{\sqrt{\left(x_j - x_{j-1}\right)^2 + \left(y_j - y_{j-1}\right)^2 + \left(h_j - h_{j-1}\right)^2}}{V_{ref}}, j = 1, \cdots N$$

After calculating the time required, the waypoints are updated with time information, as shown in Equation (4).

(4) $$\overline{\boldsymbol{P}}_j = \left(t_j, x_j, y_j, h_j\right)^T \Big|_{j=0}^{j=N}$$

The transformed vertex in Equation. (4) can be expressed as a function of time using spline interpolation. When expressed as a function of time, it must be expressed in the form of a continuous function to obtain accurate information on the position, speed, and acceleration of the aircraft. If the distance between segments is excessively long, vertices are added between waypoints, as shown in Figure 6. Otherwise, the transformed trajectory cannot accurately describe the existing path.

The length of the segment can be set as a constant by setting the interval between the vertices added, as shown in Figure 6, as the product of the flight speed of the aircraft V_{ref} and the constant timeΔt_{ref}. The connection between the existing vertices and added vertices can be converted into a flyable trajectory using 7[th] order spline interpolation proposed in [22]. The arrival time and heading angle of the aircraft at each waypoint are generated using spline interpolation. Each trajectory containing the position and orientation parameters can be created independently. For simplicity, only x-trajectory generation is considered here. Using a dimensionless variable τ, the spline polynomial for the j[th] segment can be expressed as Equation (5) (let M be the new number of waypoints after adding new waypoints, as shown in Figure 6).

(5) $$x(\tau) = \sum_{k=0}^{k=7} a_{j,k}\tau^k, \quad j = 1,2, \cdots M$$

(6) $$\tau = \frac{t - t_{j-1}}{t_j - t_{j-1}} \in [0,1]$$

The M segments are expressed as a continuous function using the polynomial shown in Equation 5, for which the eight M-coefficients $\left\{a_{j,k}\right\}_{j=1}^{j=M}\Big|_{k=0}^{k=7}$ must be determined for each polynomial. The coefficients can be calculated by applying the generated waypoint data and continuity and smoothness conditions at each waypoint. If this method is also applied to determine the positions of y and h, waypoint data according to time can be obtained from Equation (7).

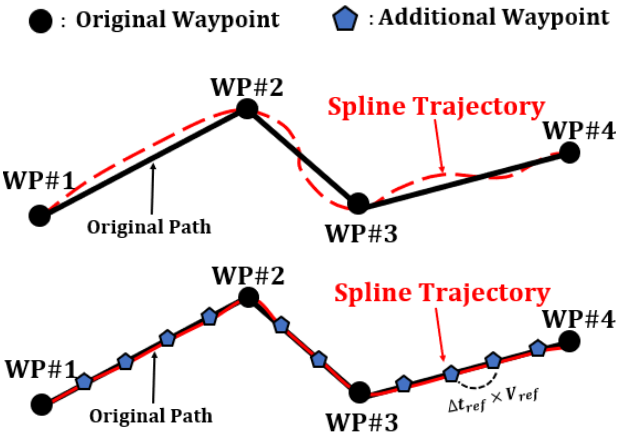

Figure 6. Example of spline interpolation with added path points

(7) $$\bar{\boldsymbol{r}}_j = \left(t_j, x_j, y_j, h_j\right)^T \Big|_{j=0}^{j=M}$$

Information on the heading angle of each segment is required for accurate trajectory-tracking control; the angle can be calculated using Equation (8).

(8) $$\psi_j = tan^{-1}\left(\frac{x_{j+1} - x_j}{y_{j+1} - x_j}\right), j = 0,1, \cdots M$$

By combining Equations (7) and (8), the trajectory and heading angle that the aircraft must follow for all waypoints can be expressed using Equation (9).

(9) $$\boldsymbol{r}_j = \left(t_j, x_j, y_j, h_j, \psi_j\right)^T \Big|_{j=0}^{j=M}$$

When an aircraft tries to follow a given path by converting the path into a trajectory, it changes the speed or converts the path to a flyable trajectory if the trajectory is beyond the dynamic limit of the aircraft. The aircraft parameters used in this verification process include the limits of the aircraft's normal load factor, ascent rate ($\boldsymbol{v_c}$), flight path angle ($\boldsymbol{\gamma_c}$), speed ($\boldsymbol{V}$), and acceleration ($\boldsymbol{a}$). First, the trajectory length corresponding to each segment is obtained using Equation (10).

(10) $$s(t) = \int_{t_0}^{t} V(t)dt$$

The curvature (κ) and unit vectors corresponding to each trajectory can be calculated using Equations (11)–(12)[23].Here, $\boldsymbol{u}_t, \boldsymbol{u}_n, \boldsymbol{u}_b$ denote tangent, normal, and binormal vectors, respectively.

(11) $\boldsymbol{\kappa} = \frac{\ddot{\boldsymbol{r}}(t)}{V^2} - \frac{(\dot{\boldsymbol{r}} \cdot \ddot{\boldsymbol{r}})}{V^3}\boldsymbol{u}_t$

(12) $\boldsymbol{u}_t = \frac{d\boldsymbol{r}}{ds}, \boldsymbol{u}_n = \frac{\boldsymbol{\kappa}}{\|\boldsymbol{\kappa}\|}, \boldsymbol{u}_b = \boldsymbol{u}_t \times \boldsymbol{u}_n$

The velocity and acceleration vectors are then expressed using Equations (13) and (14) from the Frenet-Serret formulas employing differential geometric theory.

(13) $\dot{\boldsymbol{r}}(t) = V\boldsymbol{u}_t$

(14) $\ddot{\boldsymbol{r}}(t) = \dot{V}\boldsymbol{u}_t + \|\boldsymbol{\kappa}\|V^2\boldsymbol{u}_n$

Finally, the corresponding normal load factor ($\boldsymbol{n}$) acting on the aircraft is represented in Equation (15) using gravitational acceleration (g) and unit vector ($\boldsymbol{u}_g$).

(15) $\boldsymbol{n} = \frac{\|\boldsymbol{\kappa}\|V^2}{g}\boldsymbol{u}_n + \boldsymbol{u}_g$

Using Equations (10)–(15), the trajectory can be examined for flight feasibility. If any trajectory parameters exceed the aircraft performance limits, local replanning can be applied to correct the trajectory effectively.

Another advantage of converting a path into a trajectory is that a new path can be generated in consideration to the direction and speed of other aircraft or moving objects. Air traffic control optimizes the movement of multiple aircraft and ensures flight safety by maintaining a separation between the aircraft[24]. Transforming the path into a trajectory gives the location that the target aircraft should reach during flight. If the performance of the trajectory-tracking control system is excellent, it is possible to determine the position of the aircraft accurately and to replan the maneuver or other paths to avoid surrounding obstacles. As shown in Figure 6, if the position of the aircraft is updated at every t_{ref} and the path is replanned considering the changed position of obstacles, the aircraft can fly safely without colliding with the obstacles.

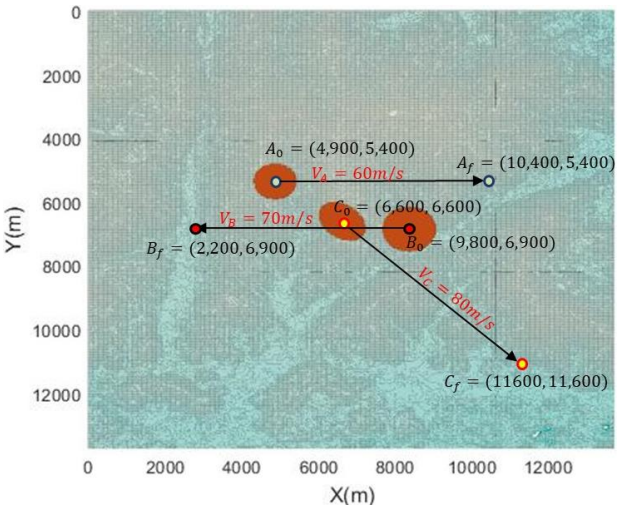

Figure 7. Moving Obstacles for simulation

To consider the movement of moving obstacles, obstacles with detailed information were set up, as shown in Figure 7 and Table 1.

Table 1. Specifications of Moving obstacles of Figure 7

	Obstacles		
	A	B	C
Major axis (m)	1,200	1,500	1,500
Minor axis (m)	1,200	1,500	1,050
Initial point (m)	x=4,900 y=5,400	x=9,800 y=6,900	x=6,000 y=6,600
Final point (m)	x=10,400 y=5,400	x=2,200 y=6,900	x=11,600 y=11,600
Velocity	60 m/s	70 m/s	80 m/s

Figure 8 is a visual representation of the results of applying the advanced real-time RRT* to an aircraft flying at a constant speed of $V_{ref} = 50\,m/s$ in a dynamic environment. The generated path was converted into a trajectory using 7th order spline interpolation, and the positions of the obstacles and aircraft were updated every 0.1 seconds.

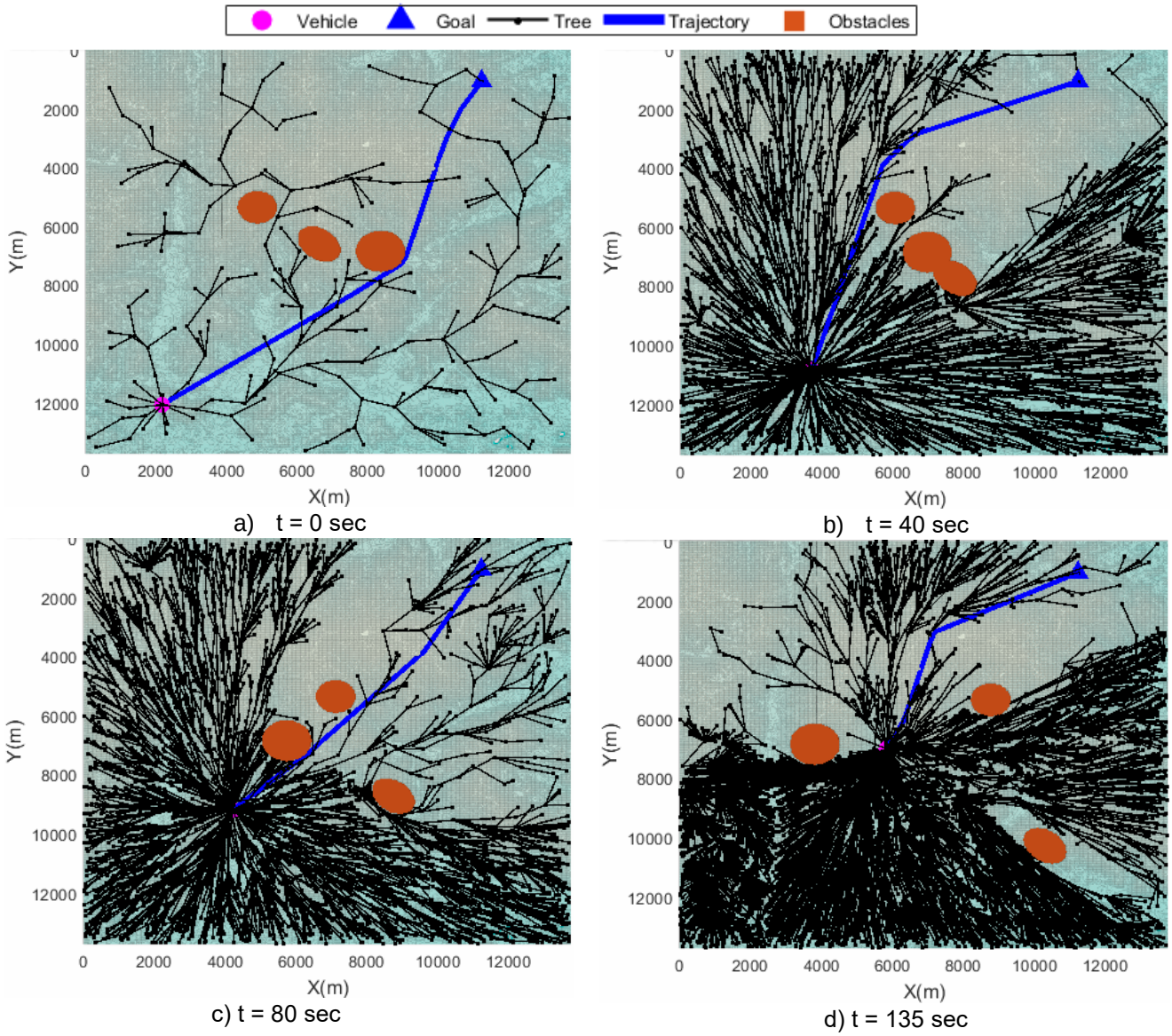

Figure 8. Results of Advanced Real-Time RRT*

Figure 8 shows the results of the advanced real-time RRT* algorithm. First, the initial path was generated while avoiding obstacles, as shown in Figure 8 a). This is not different from the results of applying the LOSPO algorithm in the same environment. However, in this case, as the obstacles moved, as described in Table 1, the path was replanned during the simulation. Therefore, the path for reaching the final node was replanned, as shown in Figure 8 b)–d). The position of the aerial vehicle was updated every 0.1 seconds, and the optimal path was generated. Figure 9 shows the detailed data on the trajectory described in Figure 8. The heading angle at every waypoint is also represented. The heading angles were calculated using Equation (8).

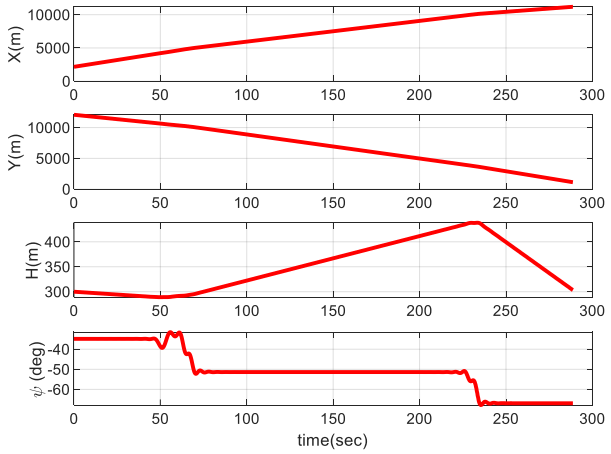

Figure 9. Trajectory of the result shown in Figure 8.

Figures 8 and 9 show that the advanced real-time RRT* algorithm proposed in this study can successfully generate paths in real time and in a dynamic environment. Next, we address a flight control system that generates instructions to follow paths accurately.

4. FLIGHT CONTROL SYSTEM DESIGN

Many parameters need to be considered to design a flight control system for trajectory tracking, such as the performance of the controller, robustness to uncertainty, and the aircraft operating range. IBSC is a nonlinear control technique well known for its excellent tracking performance[25]. It is a combination of backstepping control (BSC) and incremental dynamics. Based on the Lyapunov theory[26], BSC guarantees globally asymptotical stability. Although BSC exhibits sufficiently good performance, incremental dynamics were incorporated to improve the robustness to external disturbances and model uncertainty. The robustness of the model was improved by using the current state of the actuator of the aircraft and an estimate of the acceleration[27,28]. The structure of the IBSC used in this study is shown in Figure 10.

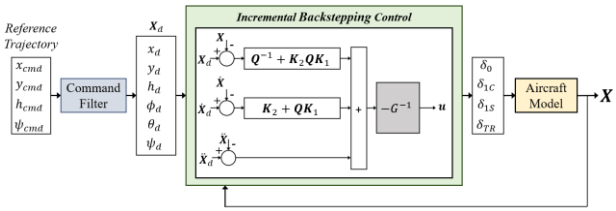

Figure 10. Structure of incremental backstepping controller

The motion equations of the rotary-wing aircraft were adopted from reference[29], and the simplified form is shown in Equation (16). The aircraft variables at time $t = t_0$ are defined in Equation (19). The detailed derivation of the equation of aircraft motion can be obtained from reference[22]. The control inputs $\boldsymbol{\delta_0}, \boldsymbol{\delta_{1C}}, \boldsymbol{\delta_{1S}}$, and $\boldsymbol{\delta_{TR}}$ represent the collective, lateral cyclic, longitudinal cyclic, and pedal, respectively.

(16) $\ddot{\boldsymbol{x}} = \boldsymbol{f}_{na}(\boldsymbol{x}, \dot{\boldsymbol{x}}, \boldsymbol{u}_p)$

(17) $\boldsymbol{x} = (x, y, z, \phi, \theta, \psi)^T$

(18) $\boldsymbol{u}_p = (\delta_0, \delta_{1C}, \delta_{1S}, \delta_{TR})^T$

(19) $\boldsymbol{x}_0 = \boldsymbol{x}(t_0), \boldsymbol{u}_{p,0} = \boldsymbol{u}_p(t_0)$

By applying incremental mechanics to Equation (16), the equations of motion related to the control effective matrix $\boldsymbol{G_0}$ and the control inputs are derived using Equation (21).

(20) $\ddot{\boldsymbol{x}} \approx \boldsymbol{f}_{na0} + \boldsymbol{F}_0 \begin{pmatrix} \Delta\boldsymbol{x} \\ \Delta\dot{\boldsymbol{x}} \end{pmatrix} + \boldsymbol{G}_0 \Delta\boldsymbol{u}_p$

(21) $\ddot{\boldsymbol{x}} \approx \ddot{\boldsymbol{x}} + \boldsymbol{G}_0 \Delta\boldsymbol{u}_p$

The motion equation represented in Equation (21) is an underactuated system with only four control inputs for six state variables. To reduce the burden of control system design, it is useful to apply the slack variable approach, which fully actuates the system [30]. First, as shown in Equations (22)–(25), a slack variable vector and a control effective matrix are set.

(22) $\Delta\boldsymbol{u}_s = (u_{s1} \quad u_{s2})^T$

(23) $\Delta\boldsymbol{u} \triangleq (\Delta\boldsymbol{u}_p \quad \Delta\boldsymbol{u}_s)^T$

(24) $\boldsymbol{G}_s = \frac{\partial \boldsymbol{f}_{na}}{\partial \Delta\boldsymbol{u}_s}$

(25) $\boldsymbol{G} \triangleq (\boldsymbol{G}_0 \quad \boldsymbol{G}_s)$

Using the above equations, the motion equation is defined as follows.

(26) $\ddot{\boldsymbol{x}} = \boldsymbol{x}_0 + \boldsymbol{G}\Delta\boldsymbol{u} + \Delta\boldsymbol{\zeta}$

(27) $\Delta\boldsymbol{\zeta} \triangleq -\boldsymbol{G}_s \Delta\boldsymbol{u}_s$

Next, the desired state variables $\boldsymbol{x}_d$ and virtual control $\boldsymbol{\alpha}$ are defined to set the error dynamics for the BSC. Equations (28) and (29) represent the error dynamics, where $\boldsymbol{z}_1$ is the error of the state variables and $\boldsymbol{z}_2$ is the differential error.

(28) $\boldsymbol{z}_1 = \boldsymbol{x} - \boldsymbol{x}_d$

(29) $\boldsymbol{z}_2 = \dot{\boldsymbol{x}} - \dot{\boldsymbol{x}}_d$

The control Lyapunov function (CLF) is defined in Equation (30), where $\boldsymbol{Q}$ and $\boldsymbol{\Lambda_\xi}$ are positive square weight matrices. The derivative of the CLF must satisfy the stability condition ($\dot{V} < 0$).

(30) $V = \frac{1}{2}\boldsymbol{z}_1^T\boldsymbol{Q}\boldsymbol{z}_1 + \frac{1}{2}\boldsymbol{z}_2^T\boldsymbol{z}_2 + \frac{1}{2}\Delta\boldsymbol{\xi}^T\boldsymbol{\Lambda}_\xi^{-1}\Delta\boldsymbol{\xi}$

(31) $\dot{V} = \boldsymbol{z}_1^T\boldsymbol{Q}\dot{\boldsymbol{z}}_1 + \boldsymbol{z}_2^T\dot{\boldsymbol{z}}_2 + \Delta\boldsymbol{\xi}^T\boldsymbol{\Lambda}_\xi^{-1}\Delta\dot{\boldsymbol{\xi}} = \boldsymbol{z}_1^T\boldsymbol{Q}(\boldsymbol{\alpha} - \dot{\boldsymbol{x}}_d) + \boldsymbol{z}_2^T(\boldsymbol{Q}^T\boldsymbol{z}_1 + \ddot{\boldsymbol{x}}_0 + \boldsymbol{G}\Delta\boldsymbol{u} - \ddot{\boldsymbol{\alpha}}) + \Delta\boldsymbol{\xi}^T(\boldsymbol{\Lambda}_\xi^{-1}\Delta\dot{\boldsymbol{\xi}} + \boldsymbol{z}_2)$

Using the Lyapunov theory, the virtual control and control input variables and control parameters are calculated as follows:

(32) $\boldsymbol{\alpha} = -\boldsymbol{Q}^{-1}\boldsymbol{K}_1\boldsymbol{z}_1 + \dot{\boldsymbol{x}}_d$

(33) $\Delta\dot{\boldsymbol{\xi}} = -\boldsymbol{\Lambda}_\xi\boldsymbol{z}_2$

(34) $\Delta\boldsymbol{u} = -\boldsymbol{G}^{-1}\{(\boldsymbol{K}_2 + \boldsymbol{Q}^{-1}\boldsymbol{K}_1)\dot{\boldsymbol{z}}_1 + (\boldsymbol{Q}^T + \boldsymbol{K}_2\boldsymbol{Q}^{-1}\boldsymbol{K}_1)\boldsymbol{z}_1 + \ddot{\boldsymbol{x}}_0 - \ddot{\boldsymbol{x}}_d\}$

Two mission-task elements, pirouette and slalom,

were constructed to verify the flight control system with reference to the design standard of a rotorcraft called ADS-33E-PRF[31]. The pirouette examines the ability to accomplish precision control simultaneously in the pitch, roll, yaw, and heave axes; the trajectory is shown in Figure (11). The slalom tests the ability and turning coordination to maneuver aggressively in a forward flight; its trajectory is shown in Figure (12). The velocity of the rotorcraft was set as $50m/s$ (27 knots). The control gains selected for each axis is given in Equation (35), and the control parameters were set as shown in Equation (36).

$$\begin{aligned} \boldsymbol{K}_1 &= diag(1.3125, 1.3125, 1.3125, 1.3125 \\ &\quad ,11.8125,11.8125) \\ \boldsymbol{K}_2 &= diag(1.5, 1.5, 1.5, 1.5, 4.5\ 4.5) \end{aligned} \tag{35}$$

$$\begin{aligned} \boldsymbol{Q} &= diag(1.1429, 1.1429, 1.1429, 1.1429 \\ &\quad ,0.3810, 0.3810) \\ \boldsymbol{\Lambda_\xi} &= diag(0, 0, 0, 0, 1, 1) \end{aligned} \tag{36}$$

Figures (13) and (15) show the results, and Figures (14) and (16) show the errors in each trajectory. Based on these results, it can be concluded that the flight control system designed in this study provided appropriate control inputs for tracking the reference trajectory. Specifically, the position errors were less than 1 m and the heading angle errors were less than 0.2 °.

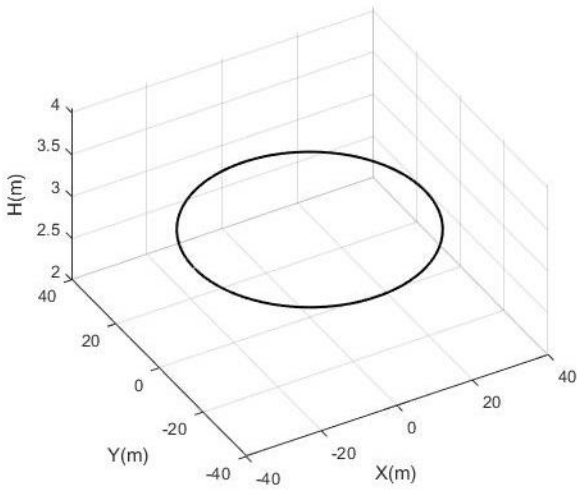

Figure 11. Trajectory of pirouette

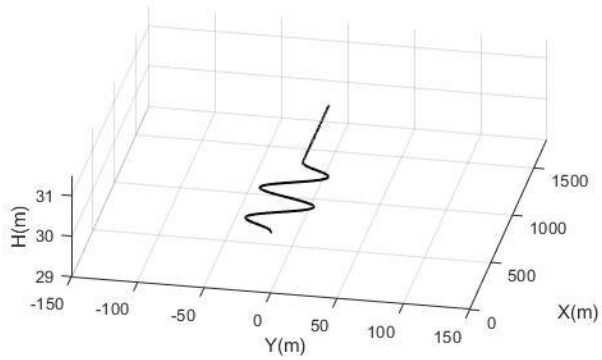

Figure 12. Trajectory of slalom

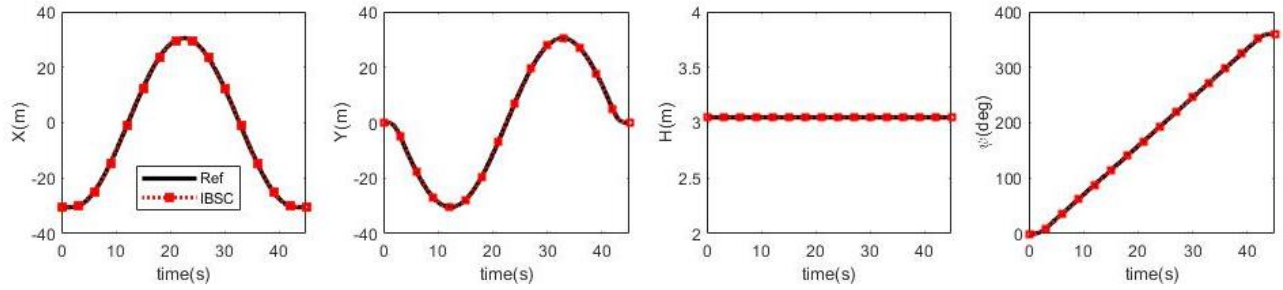

Figure 13. Trajectory tracking results (Pirouette maneuver)

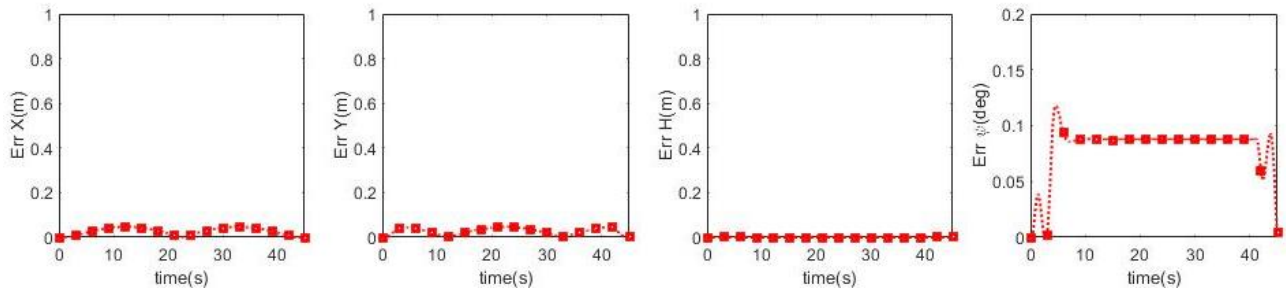

Figure 14. Trajectory tracking errors (Pirouette maneuver)

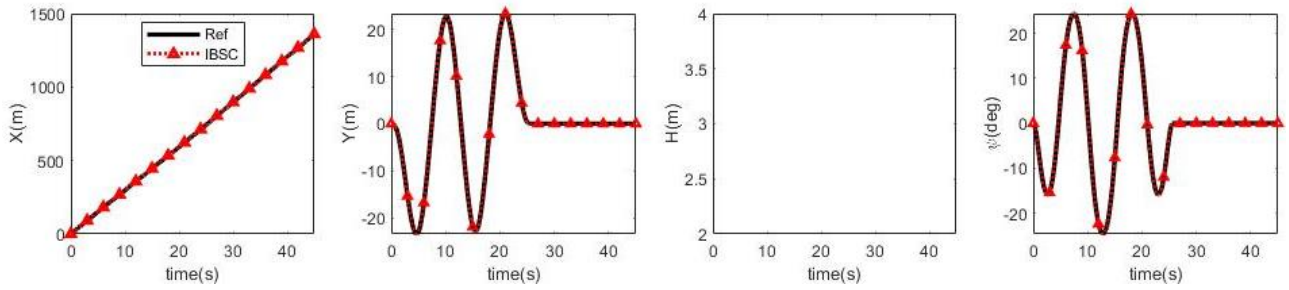

Figure 15. Trajectory tracking results (Slalom maneuver)

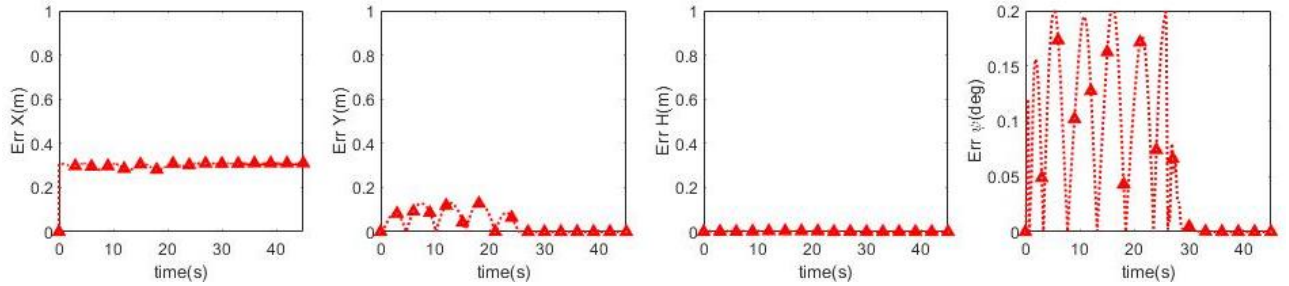

Figure 16. Trajectory tracking errors (Slalom maneuver)

5. APPLICATION

The flight control system was integrated with the guidance system consisting of advanced real-time RRT* (covered in Section 3). The guidance system creates the trajectory that the rotorcraft must follow and delivers the trajectory data to the flight control system. The guidance system updates and optimizes the trajectory while considering the position and characteristics of the rotorcraft in real-time. The simulation was performed in a dynamic environment, as shown in Figure 8.

Figure 17 shows the integration result of the guidance system using the advanced real-time RRT* and the flight control system applying the IBSC. The path generated by the integrated system is not significantly different from that of the guidance system alone, which indicates that the rotorcraft (BO-105) can follow the path without exceeding its limits. To illustrate the tracking performance in detail, the errors in the reference trajectory are shown in Figure 18. All the position errors were less than 1 m, and the heading-angle error was less than 0.2°. However, when the flight time was approximately 70 s and 230 s, the error in the heading angle increased while vibrating. From Figure 8, it can be inferred that the heading angle must be changed to avoid moving obstacles. At approximately 230 s, the heading angle decreased rapidly to allow the aircraft bypass a high mountain peak. On the other hand, the position error did not increase significantly at 230 s. It indicates that the integrated system was sensitive to changes in the heading angle.

Figures 19 and 20 show the state and control input variables of the rotorcraft during the simulation. Although the variables also changed rapidly at 70 and 230 s, the new variables did not exceed the BO-105 limits, indicating that the rotorcraft flight was stable.

6. CONCLUSION

In this paper, we described a new path planning algorithm called the Advanced Real-Time RRT* and the process of integrating the guidance and flight control systems. The real-time route planning algorithm was designed by adopting the RRT*, LOSPO, and curve-fitting techniques. The algorithm generates paths to avoid mobile obstacles and considers the limits of the rotorcraft. We also employed nonlinear control (backstepping control) using incremental dynamics to provide robustness to uncertainty. The system integrated with guidance and flight control was simulated in a dynamic environment, and its ability to plan appropriate paths in real time and generate control inputs that follow the path accurately was verified. The system proposed in this study achieved the highest level of autonomy proposed by Kendoul; therefore, the study can be very useful in future RUAS research.

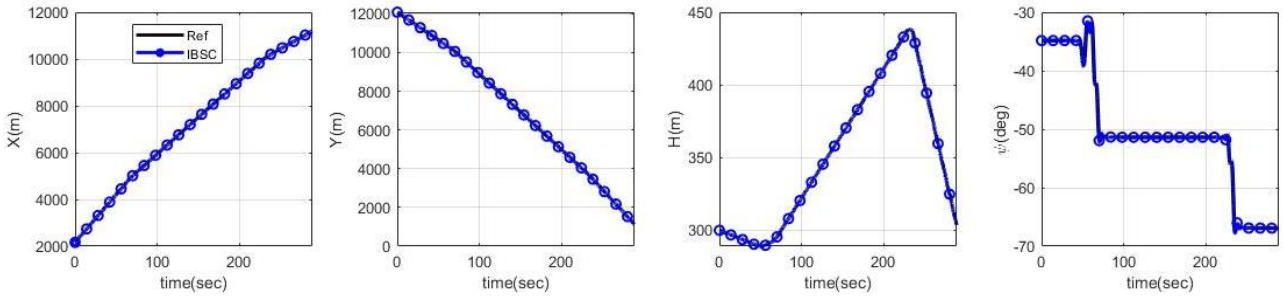

Figure 17. Trajectory tracking results of the integrated system

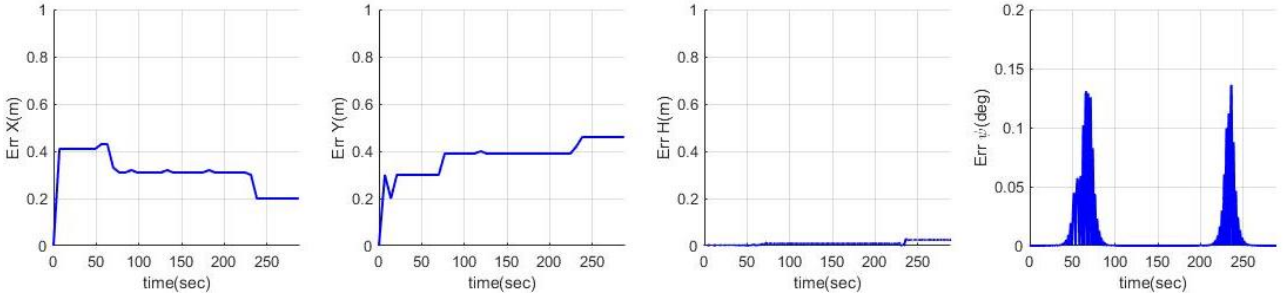

Figure 18. Trajectory tracking errors of the integrated system

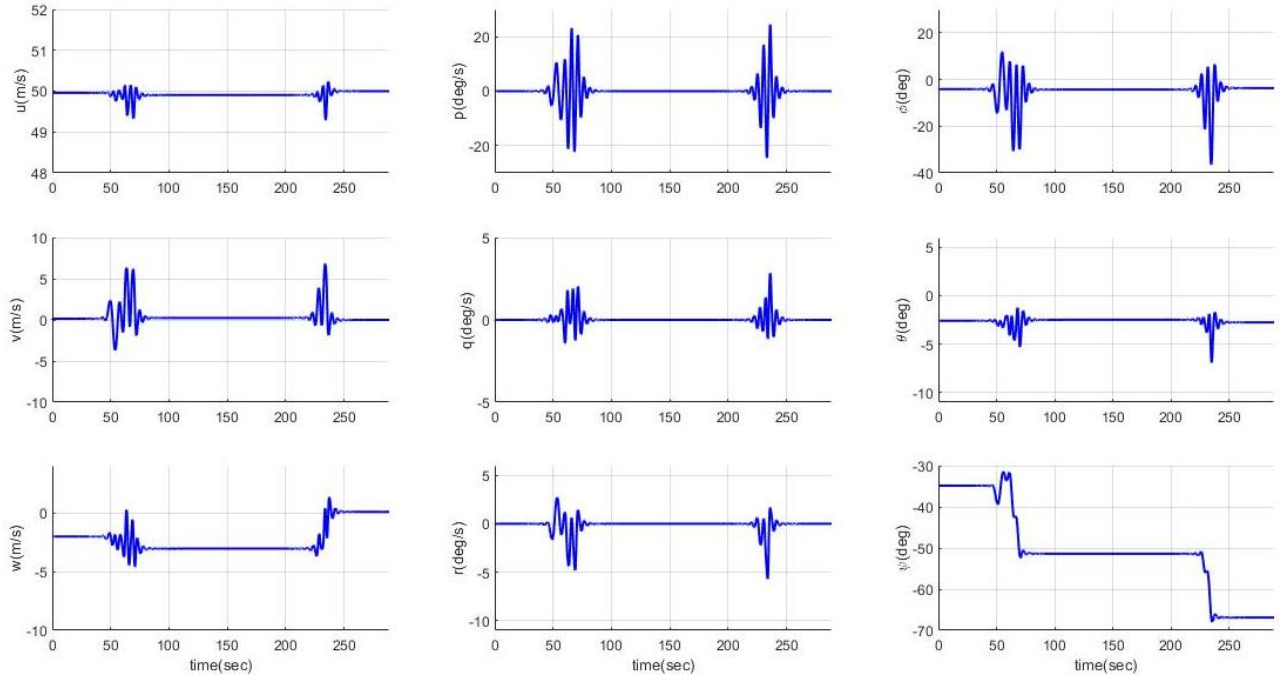

Figure 19. State variables of the BO-105

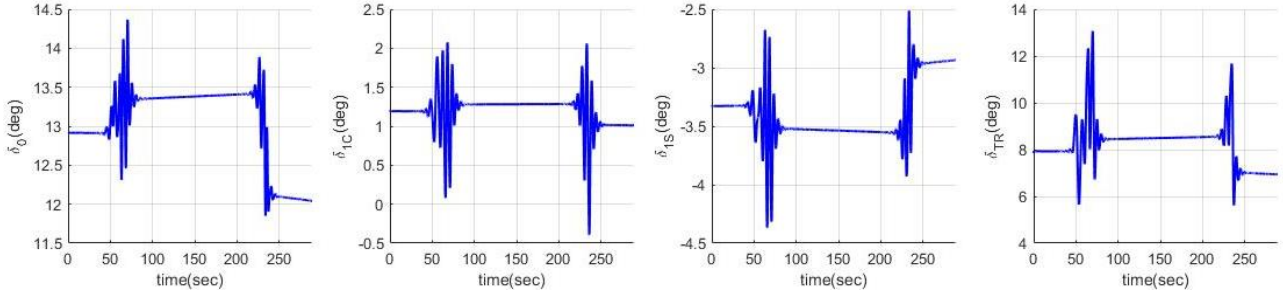

Figure 20. Control inputs variables generated by the flight control system

Acknowledgement

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (No. 2020R1A6A1A03046811). And this work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korean government (MSIT) (NRF-2020R1A2C2011955).

Reference

[1] Ursula Bloom, He Lit The Lamp, Burke, 1958.

[2] S. Mills, The Dawn of the Drone, casemate, 2019.

[3] I. Jeelani, M. Gheisari, Safety challenges of UAV integration in construction: Conceptual analysis and future research roadmap, Saf. Sci. 144 (2021) 105473. https://doi.org/10.1016/j.ssci.2021.105473.

[4] C. Yan, L. Fu, J. Zhang, J. Wang, A Comprehensive Survey on UAV Communication Channel Modeling, IEEE Access. 7 (2019) 107769–107792. https://doi.org/10.1109/ACCESS.2019.2933173.

[5] A. Fotouhi, H. Qiang, M. Ding, M. Hassan, L.G. Giordano, A. Garcia-Rodriguez, J. Yuan, Survey on UAV Cellular Communications: Practical Aspects, Standardization Advancements, Regulation, and Security Challenges, IEEE Commun. Surv. Tutorials. 21 (2019) 3417–3442. https://doi.org/10.1109/COMST.2019.2906228.

[6] ALIAS equipped Black Hawk helicopter completes first uninhabited flight, 2022. https://www.darpa.mil/news-events/2022-02-08 (accessed February 8, 2022).

[7] F. Kendoul, Survey of advances in guidance, navigation, and control of unmanned rotorcraft systems, J. F. Robot. 29 (2012) 315–378. https://doi.org/10.1002/rob.20414.

[8] K. Naderi, J. Rajamaki, P. Hamalainen, RT-RRT∗: A real-time path planning algorithm based on RRT∗, in: Proc. 8th ACM SIGGRAPH Conf. Motion Games, MIG 2015, 2015. https://doi.org/10.1145/2822013.2822036.

[9] B.S. Morse, T.S. Yoo, P. Rheingans, D.T. Chen, K.R. Subramanian, Interpolating implicit surfaces from scattered surface data using compactly supported radial basis functions, in: Proc. Int. Conf. Shape Model. Appl., IEEE Computer Society, n.d.: pp. 89–98. https://doi.org/10.1109/SMA.2001.923379.

[10] S. Karaman, E. Frazzoli, Sampling-based algorithms for optimal motion planning, Int. J. Rob. Res. 30 (2011) 846–894. https://doi.org/10.1177/0278364911406761.

[11] F. Islam, J. Nasir, U. Malik, Y. Ayaz, O. Hasan, RRT*-Smart: Rapid convergence implementation of RRT* towards optimal solution, 2012 IEEE Int. Conf. Mechatronics Autom. ICMA 2012. (2012) 1651–1656. https://doi.org/10.1109/ICMA.2012.6284384.

[12] Steven M. LaValle, Rapidly-exploring random trees: A new tool for path planning, 1998.

[13] Y. Li, W. Wei, Y. Gao, D. Wang, Z. Fan, PQ-RRT*: An improved path planning algorithm for mobile robots, Expert Syst. Appl. 152 (2020) 113425. https://doi.org/10.1016/j.eswa.2020.113425.

[14] I. Noreen, A. Khan, Z. Habib, A Comparison of RRT, RRT* and RRT*-Smart Path Planning Algorithms, IJCSNS Int. J. Comput. Sci. Netw. Secur. 16 (2016) 20–27. http://cloud.politala.ac.id/politala/1. Jurusan/Teknik Informatika/19. e-journal/Jurnal Internasional TI/IJCSNS/2016 Vol. 16 No. 10/20161004_A Compar ison of RRT, RRT and RRT - Smart Path Planning Algorithms.pdf.

[15] J.W. Woo, J.-Y. An, M.G. Cho, C.-J. Kim, Integration of path planning, trajectory generation and trajectory tracking control for aircraft mission autonomy, Aerosp. Sci. Technol. 118 (2021) 107014. https://doi.org/10.1016/j.ast.2021.107014.

[16] K.H. Lee, J.W. An, J.S. Park, S.B. Lee, C.J. Kim, Study on Path Optimization using LOSPO and Performance Analysis, (2022) 253–254.

[17] M.N. Rastgoo, B. Nakisa, M.F. Nasrudin, M.Z.A. Nazri, A critical evaluation of literature on robot path planning in Dynamic environment, J. Theor. Appl. Inf. Technol. 70 (2014) 177–185.

[18] J.S. Gutmann, M. Fukuchi, M. Fujita, Real-time path planning for humanoid robot navigation, IJCAI Int. Jt. Conf. Artif. Intell. (2005) 1232–1237.

[19] M. Otte, E. Frazzoli, RRT X : Real-Time Motion Planning / Replanning for Environments with Unpredictable Obstacles, (n.d.). https://doi.org/10.1007/978-3-319-16595-0.

[20] Y. Kuwata, J. Teo, G. Fiore, S. Karaman, E. Frazzoli, J.P. How, Real-time motion planning with applications to autonomous urban driving, IEEE Trans. Control Syst. Technol. 17 (2009) 1105–1118. https://doi.org/10.1109/TCST.2008.2012116.

[21] K. Kant, S.W. Zucker, Toward Efficient Trajectory Planning: The Path-Velocity Decomposition, Int. J. Rob. Res. 5 (1986) 72–89. https://doi.org/10.1177/027836498600500304.

[22] C.J. Kim, D.H. Lee, S.W. Hur, Efficient and Robust Inverse Simulation Techniques Using Pseudo-Spectral Integrator with Applications to Rotorcraft Aggressive Maneuver Analyses, Int. J. Aeronaut. Sp. Sci. (2019). https://doi.org/10.1007/s42405-019-00160-x.

[23] R.S. Edition, Elementary differential geometry, 2009. https://doi.org/10.1007/978-3-7643-9971-9_7.

[24] D. Delahaye, S. Puechmorel, P. Tsiotras, E. Feron, Mathematical Models for Aircraft Trajectory Design: A Survey, in: 2014: pp. 205–247. https://doi.org/10.1007/978-4-431-54475-3_12.

[25] P.V. Kokotovic, The joy of feedback: nonlinear and adaptive, IEEE Control Syst. 12 (1992) 7–17. https://doi.org/10.1109/37.165507.

[26] Y. Zou, Nonlinear robust adaptive hierarchical sliding mode control approach for quadrotors, Int. J. Robust Nonlinear Control. 27 (2017) 925–941. https://doi.org/10.1002/rnc.3607.

[27] P. Lu, E.-J. Van Kampen, Q.P. Chu, Robustness and Tuning of Incremental Backstepping Approach, in: AIAA Guid. Navig. Control Conf., American Institute of Aeronautics and Astronautics, Reston, Virginia, 2015. https://doi.org/10.2514/6.2015-1762.

[28] B.-J. Jeon, M.-G. Seo, H.-S. Shin, A. Tsourdos, Understandings of incremental backstepping controller considering measurement delay with model uncertainty, Aerosp. Sci. Technol. 109 (2021) 106408. https://doi.org/10.1016/j.ast.2020.106408.

[29] C.-J. Kim, S.-H. Lee, S.-W. Hur, Kinematically Exact Inverse-Simulation Techniques with Applications to Rotorcraft Aggressive-Maneuver Analyses, Int. J. Aeronaut. Sp. Sci. 21 (2020) 790–805. https://doi.org/10.1007/s42405-020-00249-8.

[30] M. Kim, Y. Hong, S. Lee, Y. Kim, Slack Variables Generation via QR Decomposition for Adaptive Nonlinear Control of Affine Underactuated Systems, IFAC-PapersOnLine. 49 (2016) 188–193. https://doi.org/10.1016/j.ifacol.2016.09.033.

[31] C.L. Blanken, R.H. Hoh, D.G. Mitchell, D.L. Key, Test Guide for ADS-33E-PRF, HA1 Tech. Rep. No. 1130-1. (2008) 226. http://www.dtic.mil/docs/citations/ADA510050.