drones

Article

Autonomous Surveying of Plantation Forests Using Multi-Rotor UAVs

Tzu-Jui Lin * and Karl A. Stol

Department of Mechanical and Mechatronics Engineering, The University of Auckland, Auckland 1010, New Zealand
* Correspondence: tlin442@aucklanduni.ac.nz

Abstract: Modern plantation forest procedures still rely heavily on manual data acquisition in the inventory process, limiting the quantity and quality of the collected data. This limitation in collection performance is often due to the difficulty of traversing the plantation forest environment on foot. This work presents an autonomous system for exploring plantation forest environments using multi-rotor UAVs. The proposed method consists of three parts: waypoint selection, trajectory generation, and trajectory following. Waypoint selection is accomplished by estimating the rows' locations within the environment and selecting points between adjacent rows. Trajectory generation is completed using a non-linear optimization-based constant speed planner and the following is accomplished using a model predictive control approach. The proposed method is tested extensively in simulation against various procedurally generated forest environments, with results suggesting that it is robust against variations within the scene. Finally, flight testing is performed in a local plantation forest, demonstrating the successful application of our proposed method within a complex, uncontrolled environment.

Keywords: unmanned aerial vehicle; exploration planning; trajectory planning; plantation forests

check for updates

Citation: Lin, T.-J.; Stol, K.A. Autonomous Surveying of Plantation Forests Using Multi-Rotor UAVs. *Drones* **2022**, *6*, 256. https://doi.org/10.3390/drones6090256

Academic Editor: Eben Broadbent

Received: 31 July 2022
Accepted: 8 September 2022
Published: 16 September 2022

1. Introduction

Modern plantation forestry is an industry with the explicit goal of maximizing the yield of high-quality wood within a managed area. One part of plantation forest management is the forest inventory process used to estimate the volume of merchandisable timber within a plot of trees. Successful application of this will provide the forester with data to identify any issues with the volumetric growth rate and to make an informed decision regarding the forest harvest date. In New Zealand, forest inventories are conducted using the regression-based analysis technique MARVL [1], where systematic sampling is used to select small groups of trees for extrapolating data regarding the entire forest. This sampling process is known as cruising and is generally completed on foot using manual means [2], which is subject to cost and labor availability limitations. In pursuit of higher accuracies and more frequent surveys, modern methods have moved towards better parameter estimation methods with plot imputation [3], which calibrates large-scale aerial surveys using ground data. Some existing forms of data collection used are Airborne Laser Scanning (ALS) [4,5], Terrestial or Mobile Laser Scanning (TLS/MLS) [6,7] to collect the aerial and ground calibration data.

Unfortunately, the under-canopy environment is a hostile environment for humans and ground-based robots due to uneven terrain, branching, and undergrowth, all of which are seen in the plantation forests shown in Figure 1. The presence of these obstacles means that ground-based traversal within plantation forests is heavily impeded, resulting in slow and expensive under-canopy data acquisition. Fortunately, recent advancements in UAV technologies have produced low-cost, high-agility multi-rotor UAVs suitable for operation within complex environments such as plantation forests. With new sensors and processing methods capable of accurately capturing surrounding spatial information in real-time, multi-rotor UAVs carrying these sensors are rapidly gaining popularity in recent literature as remote sensors.

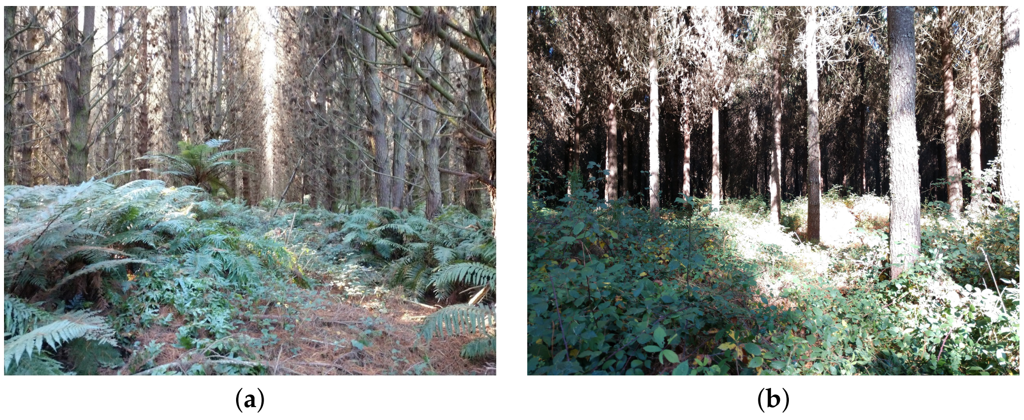


Figure 1. Examples of plantation forests which are (**a**) traversable, and (**b**) untraversable on foot in Kaingaroa Forest, Rotorua, New Zealand. Note the presence of an available flight corridor in both environments.

Existing literature regarding the operation of UAVs within under-canopy environments can be approximately split into two categories; the first category generally focuses on the mapping or data extraction aspects, and the second consists of works related to enabling autonomous flight or navigation within the forest. In general, works related to data capture in this area use manual flight with either LiDAR [8–10], or Stereo Vision/Structure from Motion [11–13]. These works usually aim to produce a survey-grade map so that parameters such as tree count, diameter at breast height, and tree taper can be extracted. The second set of works consists of those attempting to fly in forests autonomously; production of survey-grade maps and extraction of parameters are secondary in these works, with the main focus on safely traversing through the forest. Literature regarding the autonomous navigation of multi-rotor UAVs within under-canopy environments is relatively scarce, likely due to the niche use case and lack of advanced see and avoid capabilities of commercial multi-rotor systems. Nevertheless, projects such as [14] used a Hokuyo UTM-30LX 2D Planar LiDAR and an Artificial Potential Field (APF) controller for obstacle avoidance with a target velocity vector. A similar approach is followed in [15] using the same 2D LiDAR setup with a vector field-based controller for navigation. Both of these systems achieved autonomous flight between two rows of trees without collision. Similarly, [16,17] both also use a rigidly mounted 2D LiDAR for environmental sensing, coupled with Graph-SLAM [18] for motion estimation in the environment. However, all of the previously noted projects only attempt to localize and implement see-and-avoid systems as opposed to fully navigating the environment for a complete survey.

Outside of the forestry space, general-purpose trajectory planning and exploration is an active area of research in robotics. In particular, optimization-based methods are rapidly gaining popularity as they can rapidly produce safe, dynamically feasible trajectories. The general methodology followed is to formulate a cost function related to the trajectory, its derivatives, and some form of collision penalty. Various approaches are used throughout literature, for example, discrete waypoints [19], high-order polynomials with quadratic programming [20], b-splines with a local [21] or global [22] map, fixed-duration piecewise polynomials [23], and safe-corridors [24–26]. In particular, [20,22,24] demonstrate the ability to traverse unknown forest environments without collision autonomously but still require manual waypoint selection. General exploration of unknown environments is also an active area of research with many existing works; for example, [27,28] use a two-tiered approach, where frontiers are first extracted, and the order they are visited is decided by solving a traveling salesman problem. In [29], a library of motion primitives is used to search for positions with maximum information gain, while in [30–32] the search is instead conducted using rapidly-exploring random-trees (RRT) to sample the already explored region. More recently, probabilistic roadmaps (PRM) have also been explored in [33]. Unfortunately, generalized exploration methods often do not offer the best coverage performance, as shown in our previous paper [34].

This paper details the development of a UAV platform capable of autonomously exploring under-canopy environments within plantation forests to produce faster surveys. To achieve this goal, the navigation strategy outlined in our previous work [34] is extended to function in the absence of any structural assumptions about the forest. Specifically, the contributions of this work are as follows:

- A method for online waypoint placement for autonomous coverage planning in plantation forests
- A nonlinear optimization-based trajectory generation method to rapidly plan constant-speed, dynamically feasible, and safe trajectories within complex environments
- Experimental flight testing results in both simulation and a local plantation forest to verify the proposed method

Towards these contributions, Section 2 details the flown UAV hardware and methodologies used for autonomous navigation within the plantation. Section 3 details simulation environments and experiments used to determine the theoretical performance of the proposed system. Finally Section 4 details flight tests performed in an outdoor plantation environment to validate the proposed methods.

2. Materials and Methods

Because autonomous flight within plantation forests is a novel task, no sufficiently compact off-the-shelf hardware capable of carrying an appropriate LiDAR with sufficient onboard computer and endurance exists. As a result, a prototype multi-rotor UAV has been developed for flight testing of the proposed methodology, mainly out of necessity as it would allow a significantly larger margin of error before any collisions occurred. Unfortunately, the smaller UAV and comparatively heavy payload result in somewhat poor endurance, which can only be improved with lighter sensors or batteries with higher energy density. Figure 2 shows this survey UAV, consisting of a quadrotor with slightly tilted propellers for increased yaw authority. A Livox MID-70 LiDAR, along with an Intel Realsense T265 completes the sensor setup, and an Intel NUC handles onboard processing for autonomous flight. ROS Noetic [35] is used as the middleware for the software implementation, and R3Live [36] is used for pose estimation.

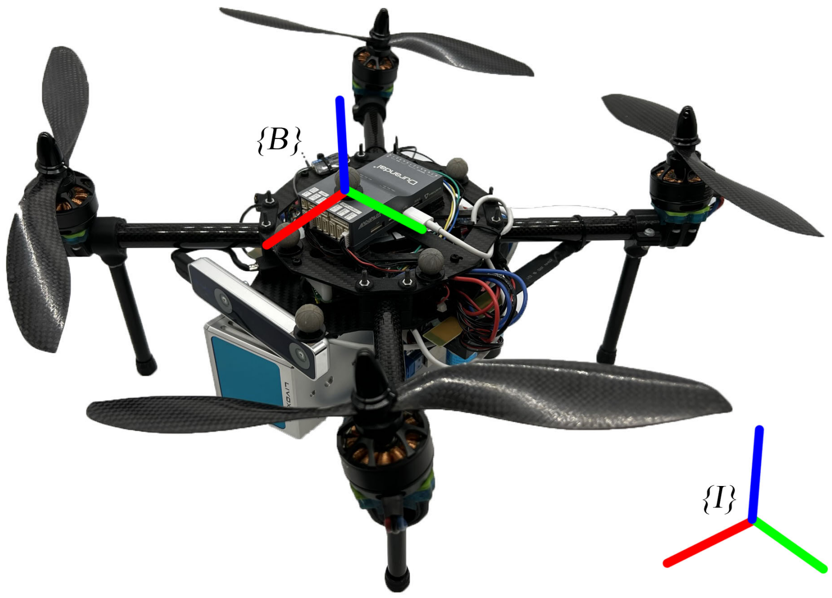


Figure 2. UAV used for flight testing. The carried payload consists of a Livox MID-70 LiDAR, Intel Realsense T265, and an Intel NUC. The body and inertial frames are denoted as $\{B\}$ and $\{I\}$ respectively.

For clarity, the notation $\{F_X\}$, $\{F_Y\}$, $\{F_Z\}$ represents the right-hand X, Y, and Z axes in the ENU frame F. Position, orientation, and transformations to frame $\{Y\}$ from with respect to frame $\{X\}$ is denoted as ${}^{Y}P_X$, ${}^{Y}R_X$, and ${}^{Y}T_X$ respectively. The inertial frame $\{I\}$ is used as the fixed world frame, while $\{B\}$ defines the location of the flight controller. While additional fixed LiDAR and camera frames are used during the computation of pose estimates and mapping information, they are excluded for clarity.

For autonomous navigation, three tasks must be completed: waypoint selection, trajectory generation, and trajectory following. The first task generates the sequence of waypoints to be visited to explore the environment fully, the second task produces a collision-free and dynamically feasible flight plan to reach the goal, and the last task executes the generated trajectory. An overview of the proposed system is shown in Figure 3.

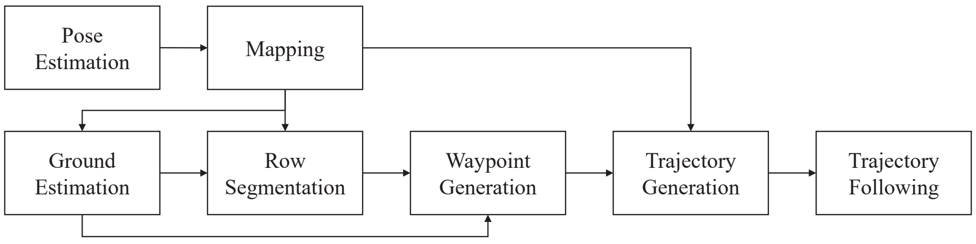


Figure 3. Overall task flow during the survey process. Pose estimation and mapping are not within the scope of this work.

To ensure robust operations within any plantation environment, the proposed method does not use any priori information regarding the structure of the plantation environment. All mapping data used in a single flight is generated onboard and online during that flight using ranging data from the onboard LiDAR, and pose estimates are generated using R3Live. This work does not use Global Navigation Satellite Systems (GNSS) due to the lack of sufficient accuracy without a clear line of sight to the sky.

2.1. Waypoint Generation

Typical New Zealand plantation forests consist of Radiata pine ideally planted on a regularly spaced square grid to allocate an approximately equal land area to each tree. However, planting in a grid pattern is difficult, so real plantations are typically established on a row-by-row basis where trees are sequentially planted in parallel rows. This planting process produces distinct rows of trees with relatively consistent spacing between them but no consistent structure in the orthogonal direction, the effects of which can be seen in Figure 4.

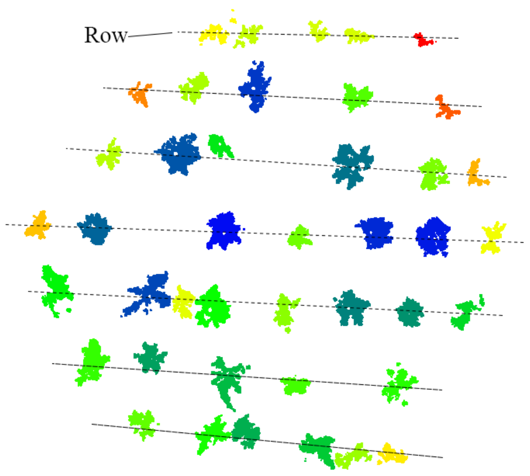


Figure 4. Horizontal slice of a point cloud in a typical New Zealand plantation forest with rows labeled. Note the lack of consistent structure in the orthogonal direction.

Our previous work [34] introduces the concept of using a lawnmowing pattern aligned with the row direction to produce faster surveys, as shown in Figure 5. This work extends this concept to enable outdoor operation in realistic forests with irregular spaced rows. This task aims to produce a set of waypoints along the corridor between two adjacent rows of trees and to generate a new set of waypoints when the trajectory follower finishes executing the desired trajectory. The process of determining goal waypoints broadly consists of ground plane estimation and removal, tree clustering, row extraction, and finally, waypoint placement. It should be noted that while multi-UAV systems have the potential to greatly increase the coverage area through partitioning of the exploration space, they are outside of the scope of this work.

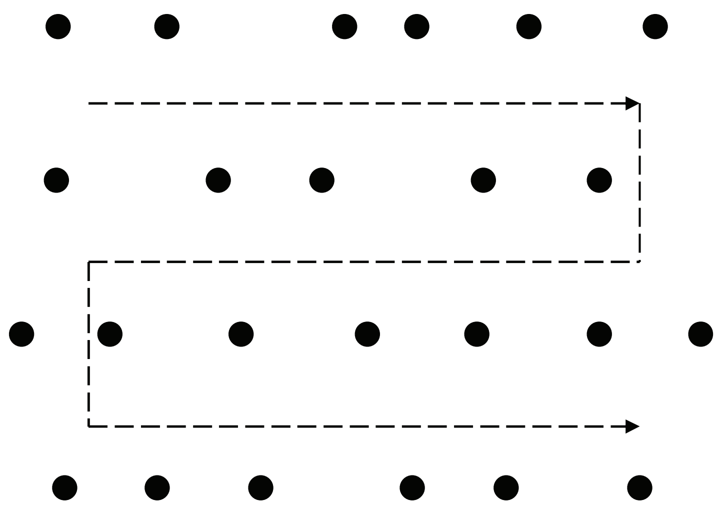


Figure 5. Approximate survey plan consisting of a lawnmowing pattern aligned with plantation rows.

The first task of this process is ground plane estimation to remove the influence of ground elevation in the clustering process. The ground mesh is first estimated using Cloth Simulation Filter (CSF) [37] with the global occupancy grid serving as the input point cloud. Ground elevation can then be removed from the global point cloud by subtracting the corresponding estimated elevation, resulting in all estimated ground points occupying an elevation of 0m. Next, individual parallel rows of trees are identified through clustering to produce an estimate of the flight corridor between adjacent rows. A slice can then be taken from the elevation corrected point cloud and flattened to form a planar view of the forest at the current UAV altitude. DBSCAN [38] is then used to cluster these points to produce a list of estimated tree locations $\boldsymbol{C_s}$. A search-based approach is used to determine the most likely direction of rows; consider the scenario presented in Figure 6, the stem locations can be collapsed along $\{I_X\}$ to form distinct clusters, as shown.

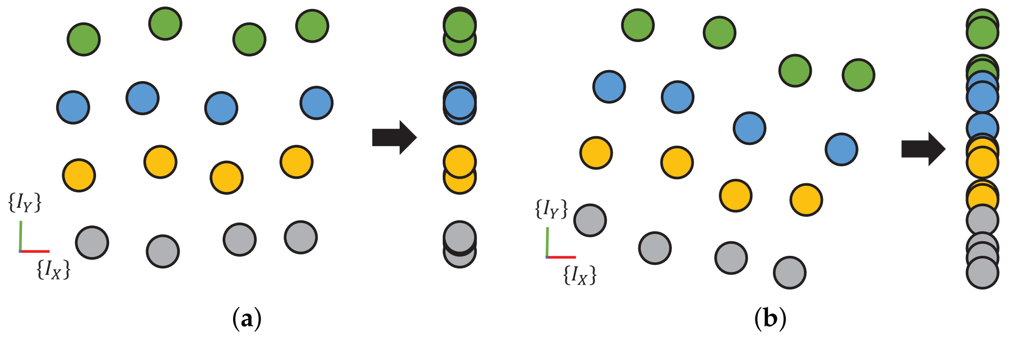


Figure 6. Top-down views of (**a**) good, and (**b**) poor alignment between $\{I_X\}$ and the principle row directions of the plantation forest.

Given the good alignment of the row direction with $\{I_X\}$, distinct clusters will be observed in the collapsed state as shown in Figure 6a. Alternatively, if the row alignment is poor, the case in Figure 6b will be present, with fewer large clusters present in the scene. Thus, the most probable row direction consists of the direction that minimizes the spread of each present cluster. Using the sum of the spread of each cluster as a cost function, a search can be performed to determine the best alignment between $\{I_X\}$ and the estimated row directions. The functions for this are shown in Equations (1) and (2), optimizing for minimum cost.

$$argminJ(\theta) = \frac{n_c}{n_v}\sum_{j=0}^{n_c}((\max(\boldsymbol{L}_j) - \min(\boldsymbol{L}_j))/n_j)^2 \tag{1}$$

where

$$\boldsymbol{L} = DBSCAN(\boldsymbol{C_s} \times \begin{bmatrix} sin(\theta) \\ cos(\theta) \end{bmatrix});\ \boldsymbol{C_s} = \begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \\ \vdots & \\ x_{n_c} & y_{n_c} \end{bmatrix} [x_i, y_i] = {}^{C_i}P_I \tag{2}$$

where $\boldsymbol{L_j}$ denotes the list of transformed tree locations corresponding to cluster $\boldsymbol{j}$, $\boldsymbol{n_j}$ denotes the number of trees in cluster $\boldsymbol{j}$, $\boldsymbol{n_v}$ denotes the number of clusters with more than five trees, and $\boldsymbol{n_c}$ denotes the number of clusters. Since row estimates are generated based on identified cluster information, the accuracy of the $\boldsymbol{\theta}$ estimate does not need to be high. Because this work does not consider initial traversal to the survey starting location, an assumption is made that the inertial frame $\boldsymbol{\{I_x\}}$ is approximately aligned with the row direction. Therefore, a coarse search is performed for $\boldsymbol{\theta} \in [\boldsymbol{-30^\circ, 30^\circ}]$ in 1° increments. The cluster labels from the minimum cost $\boldsymbol{\theta}$ value are used to label the unmodified tree locations $\boldsymbol{C}$, resulting in clusters of tree locations corresponding to independent rows. Finally, a least-squares fit is used to determine the estimated rows present within the scene and converted into the $\boldsymbol{r, \theta}$ representation, as shown in Equations (3) and (4).

Let $\boldsymbol{x} = \begin{bmatrix} \boldsymbol{m} \\ \boldsymbol{c} \end{bmatrix}$ be the least squared solution to $\boldsymbol{Ax = B}$

$$\boldsymbol{A} = \begin{bmatrix} x_1 & 1 \\ x_2 & 1 \\ \vdots & \\ x_n & 1 \end{bmatrix};\ \boldsymbol{B} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \text{For } \boldsymbol{L} = \{\boldsymbol{L_1, L_2, \cdots, L_n}\},\quad \boldsymbol{x_i, y_i \in L_j}$$

$$\boldsymbol{\theta = atan2(1, -m)} \tag{3}$$

$$\boldsymbol{r = c \times sin(\theta)} \tag{4}$$

As all segmented rows are approximately parallel, their relative position in the environment can be determined by sorting the list of rows by their associated r. Consider the environment shown in Figure 7, consisting of two approximately parallel rows ordered by r. Defining

$$r_{ij} = 0.5(r_i + r_j) \tag{5}$$

$$\theta_{ij} = 0.5(\theta_i + \theta_j) \tag{6}$$

where r_{ij}, θ_{ij} defines the estimated parameters of the traversal corridor between the tree rows defined by r_i, r_j. Since each corridor needs to be traversed at least once to fully explore the environment, the traversal order for N corridors then becomes $\{r_{12}, r_{23}, \cdots, r_{N-1,N}\}$. Lastly, a set of waypoints can be established for traversal down a single corridor using Equations (7)–(9).

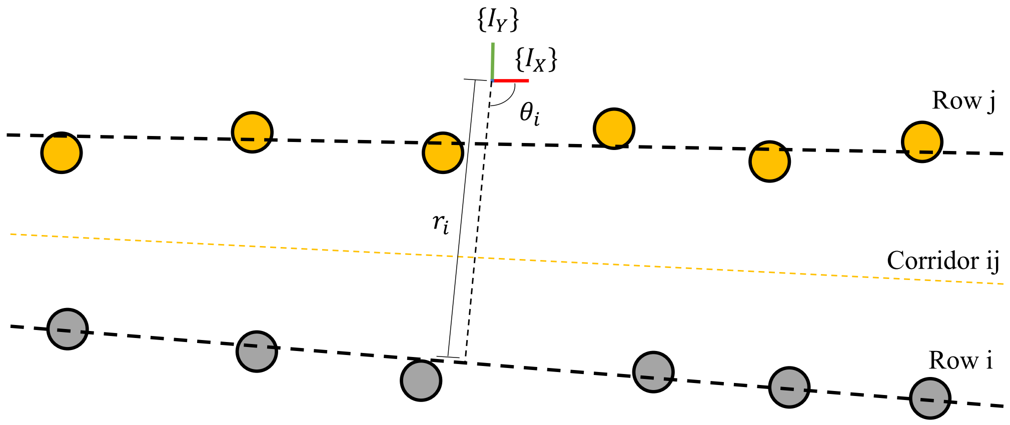


Figure 7. Sample environment consisting of two rows with r, θ labeled. The dotted yellow line denotes the traversable corridor.

$$\text{Let } w_s = \{0, Y(0), Z(0, Y(0)) + Z_r\}^T, w_e = \{0, Y(m), Z(m, Y(m)) + Z_r\}^T$$

$$Y(x) = x \times \tan(90^\circ - \theta_{ij}) + \frac{r_{ij}}{sin(\theta_{ij})} \tag{7}$$

$$w_i = |w_d| \times m_{inc} + w_s \quad \text{For} \quad i > 0, i \leq \frac{m}{m_{inc}} \tag{8}$$

$$w_d = \begin{cases} w_e - w_s & mod(i,2) = 1 \\ w_s - w_e & \text{otherwise} \end{cases} \tag{9}$$

where $Z(x,y)$ is the estimated ground elevation at $\{I_x, I_y\} = \{x, y\}$, Z_r is the reference altitude above ground, m_{inc} is the spacing between successive waypoints, w_i denotes the ith waypoint down a single corridor, and w_s, w_e denotes the start and end coordinates for a single corridor. Each successive corridor is traversed in the reverse direction to remove the need to return before each traversed row. The generated list of waypoints is passed to the trajectory planner, where any occupied waypoints are skipped; the traversal of a single corridor is considered complete when the entire generated trajectory has been fully executed, or no additional waypoints can be visited. Exploration is terminated when the required number of rows is surveyed, and a waypoint at $\{0, 0, Z(0,0)\}$ is sent to command a return to the origin.

2.2. Trajectory Generation

The trajectory planner performs three tasks; maintaining an up-to-date global map of the environment, generating the actual flight trajectory, and finally replanning if the trajectory becomes unsafe. The pipeline overview is presented in Figure 8 and uses an optimization-based approach to generate high-quality trajectories rapidly. In line with many other optimization-based trajectory planners, a Euclidean Signed Distance Field (ESDF) is used for mapping information. The ESDF is built online with point cloud data from the MID-70 LiDAR using FIESTA [39].

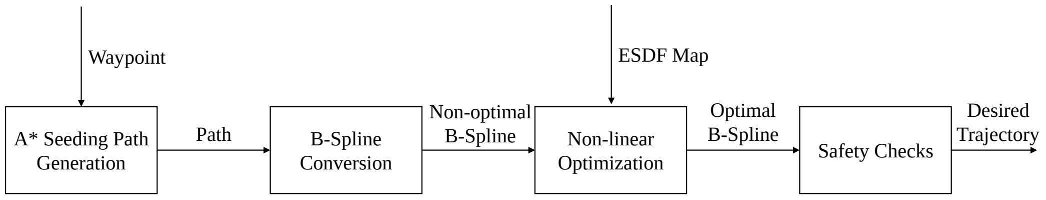


Figure 8. Overall trajectory planner pipeline.

Trajectories are represented as uniform clamped B-Splines, which are often used to define smooth curves using a small number of discrete points. A B-Spline is defined by the evenly spaced knot vector $\boldsymbol{t}$ with interval $\boldsymbol{t_{int}}$ and control points $\boldsymbol{P_0, P_1, \cdots P_n}$ where $\boldsymbol{P_i}$ represents a position in 3D space such that $\boldsymbol{P_i} \in \mathbb{R}^3$, where:

$$\boldsymbol{t} = \{\boldsymbol{t_0, t_1, \cdots t_m}\}, \boldsymbol{t_{i+1} - t_i = t_{int}}\ ; \boldsymbol{i} \in [\boldsymbol{0, m-1}] \tag{10}$$

$$\boldsymbol{P} = \{\boldsymbol{P_0, P_1, \cdots P_n}\} \tag{11}$$

Departing from how B-Splines are typically defined, the knot vector is not constrained to $[\boldsymbol{0,1})$ and is instead defined such that $\boldsymbol{t_i} \in [\boldsymbol{0, t_m})$. This definition of the knot vector is convenient because it allows for direct time allocation of the trajectory during optimization. B-Splines offer some beneficial properties in the context of trajectory planning, particularly $C^{N\text{-}1}$ continuity, and the strong convex hull property. A degree 3 B-Spline guarantees C^2 continuity at all points, which means that the trajectory is guaranteed to be continuous up to the 2nd derivative. The convex hull property also guarantees that any points of the spline will fall within the convex hull formed by the neighboring control points. Specifically, if $\boldsymbol{t} \in [\boldsymbol{t_i, t_i + 1})$, all points in that piecewise portion of the B-Spline are contained entirely in the convex hull formed by control points $\boldsymbol{P_{i-2}, P_{i-1}, P_i}$. These properties greatly simplify the enforcement of collision constraints and dynamic limits.

The first step in the trajectory generation process is producing a seeding path, with the only requirement being that this path is collision-free and approximately optimal, since the shape and dynamic properties of the trajectory will be heavily modified during optimization. It should be noted, however, that this solution should ideally be close to the optimal path to increase the probability of the optimized trajectory converging on the global minima instead of less optimal local minima. Because these requirements are very loose, an A* search [40] in 3D using 26-way connectivity is performed with the Manhattan distance heuristic to rapidly generate a traversal path to the goal. For speed reasons, the A* search grid has the same voxel resolution as the ESDF (0.2m), allowing direct ESDF lookups to be used for occupancy checking. The output of this search is a discrete set of poses $\boldsymbol{O}$ which encodes $\boldsymbol{n_o}$ discrete positions in the inertial frame $\{\boldsymbol{I}\}$ containing a collision-free path to the goal, namely

$$\boldsymbol{O} = \{\boldsymbol{o_1, o_2, \cdots, o_{n_o}}\}^T \text{ where } \boldsymbol{o_i} = \{{}^{o_i}\boldsymbol{P_I}\} \tag{12}$$

The generated path is used to seed the initial trajectory by creating a B-Spline with control points $\boldsymbol{P} = \boldsymbol{O}$, spaced equally at time interval $\boldsymbol{t_{int}}$, initially set to 0.1 s.

It should be noted that A* is used in this work only for ease of implementation, therefore many other polynomial-time path-finding algorithms such as D* Lite, $\boldsymbol{\theta}$*, JPS, Kinodynamic A*, etc. can be used in place of A* to generate the initial seeding path. However, probabilistic methods such as RRT(*) are less suitable as any initial solutions tend to be poor quality, and compute times are less predictable depending on the source and quality of available randomness, both of which affect the generation time and quality of the resulting trajectory.

Once the seeding B-Spline is created, the next step is to refine the trajectory to satisfy safety and dynamic constraints, as the only guarantee at this part of the process is collision safety. To refine the quality of this seeding trajectory, a non-linear optimization problem can be formulated around the B-Spline's control points and time interval. Leveraging Google Ceres' [41] Levenberg-Marquardt solver as a least-squares solver, the cost function $F(x)$ is defined as:

$$F(x) = F_{collision}(x) + F_{velocity}(x) + F_{acceleration}(x) + F_{time}(x) + F_{continuity}(x) \tag{13}$$

where the parameter vector x is defined as:

$$x = \{P_0, P_1, P_2, \cdots, t_{int}\} \tag{14}$$

Because the B-Splines are uniform with a fixed interval of t_{int} between each successive control point, the trajectory can be re-timed during the optimization process by including the time interval as an optimization parameter. This decouples the time of the trajectory with the number of waypoints, allowing for more flexibility in the number of control points used. The inclusion of $F_{continuity}(x)$ is used to enforce end-to-end continuity when multiple waypoints are provided to the planner; this is set to 0 when only a single waypoint is supplied. When a single trajectory consists of multiple sequentially visited waypoints, the optimization problem becomes one with multiple end-to-end joined B-Splines, which appends additional control points and time intervals to the parameter vector, namely

$$x = \{P_{0,0}, P_{0,1}, \cdots, P_{1,0}, P_{1,1}, \cdots, t_{int_0}, t_{int_1}, \cdots\} \tag{15}$$

where $P_{i,j}$ denotes the ith control point, and t_{int_j} denotes the time interval of the jth B-Spline.

Collision constraints in the planner are formulated by exploiting the strong convex hull property to guarantee a safe trajectory without needing to conduct expensive collision checks across the entire trajectory. Consider the scenario presented in Figure 9, where a degree 3 B-Spline trajectory tightly wraps around a corner. The initial portion of the B-Spline must be contained within the triangle formed by P_0 to P_2. Suppose the distance between any two control points P_i and P_j is denoted $d_{i,j}$, the convex hull of the initial portion is guaranteed to collision-free if the ESDF distance at P_i is greater than $\sum_{j=i}^{i+1} d_{j,j+1}$. Unfortunately, calculating $d_{i,j}$ becomes expensive quickly as it needs to be recomputed at every step, so a relaxed variant of this constraint is used; $d_{i,j}$ is assumed to be the same as the voxel grid resolution. Given voxel resolution d_{vox}, the seeding control points O are separated by d_{vox} due to the A* search resolution; as the seeding path is non-optimal, the actual distance between control points post-optimization will likely be less than d_{vox}. Therefore, it is likely that the trajectory will be collision-free as long as the ESDF distance $d_{ESDF}^{P_i}$ at every P_i is greater than at least two times d_{vox}. Therefore, collision threshold d_{col} can be used as a tuning factor trading off speed for safety; smaller values allow the UAV to take faster routes at the expense of a higher probability of unsafe trajectories. To prevent the execution of unsafe trajectories, a collision check is performed at discrete intervals, and the trajectory is rejected if any point fails this check. The collision cost is, therefore, defined as:

$$F_{collision}(x) = \sum_{i=0}^{n} (G(P_i)) \tag{16}$$

$$G(P_i) = \begin{cases} W_{collision}((d_{col}) - d_{ESDF}^{P_i})^2 & \text{if } d_{ESDF}^{P_i} < d_{col} \\ 0 & \text{otherwise} \end{cases} \tag{17}$$

$$d_{col} \geq 2 \times d_{vox}$$

where $W_{collision}$ is the collision weight.

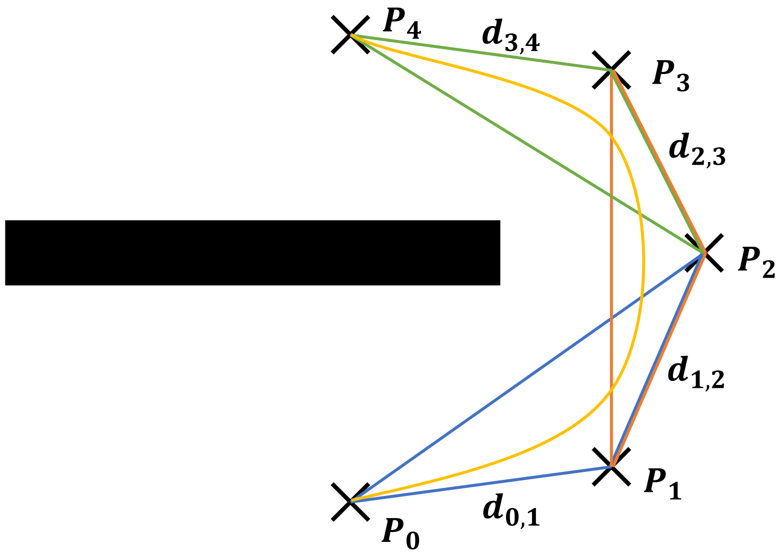


Figure 9. Worst case B-Spline collision scenario with degree 3. The trajectory is guaranteed to be collision-free if the three convex hulls shown in blue, orange, and green are obstacle free.

Velocity constraints are handled as a combination of initial/final state constraints, maximum velocity constraints, and speed constraints, namely:

$$F_{velocity}(x) = F_{initialVel}(x) + F_{finalVel}(x) + F_{speed}(x) + F_{maxVel}(x) \tag{18}$$

Because the B-Spline is C^2 continuous, the velocity at any point can be determined by inspecting the derivative of the B-Spline. The control point defining the velocity profile at index i can be computed based on the position control points P_i and P_{i+1} using the derivative of the position B-Spline. Because the convex hull property similarly constrains the velocity B-Spline, the profile's maximum velocity is also constrained by the magnitude of the velocity control points. The velocity control points are defined by:

$$P_i^{vel} = \frac{3}{int}(P_{i+1} - P_i) \tag{19}$$

where

$$int = \begin{cases} t_{int_i} \times (i+1) & \text{if } i < 3 \\ t_{int_i} \times (n-i-1) & \text{if } i > (n-4) \\ t_{int_i} \times 3 & \text{otherwise} \end{cases}$$

Therefore, the velocity state at control point P_i, and the initial and final velocity can be constrained by Equation (20). The velocity weight $W_{velState}$ is set to a relatively large value to encourage constant speed traversal in the environment.

$$F_{velState}(P_i^{vel}, V_{ref}) = W_{velState}||P_i^{vel} - V_{ref}||^2 \tag{20}$$

$$F_{initialVel}(x) = F_{velState}(P_0^{vel}, V_{initial}) \tag{21}$$

$$F_{finalVel}(x) = F_{velState}(P_{n-1}^{vel}, V_{final}) \tag{22}$$

Because forest surveying is the intended use case of this trajectory planner, there is an additional speed constraint to encourage a constant trajectory speed for more even coverage of the environment. The speed constraint is implemented by F_{speed}, which penalizes the planner when the speed at any particular control point differs from the desired survey speed.

$$F_{speed}(x) = \sum_{i=0}^{n-1} W_{speed}(||P_i^{vel}|| - S_{ref}) \tag{23}$$

where S_{ref} is the scalar reference speed for the survey. Finally, the maximum velocity can be constrained by defining:

$$F_{velMax}(x) = \sum_{i=0}^{n-1} swW_{velMax}(|P_i^{vel}| - V_{max})^2 \tag{24}$$

$$sw = \begin{cases} 1 & \text{if } ||P_i^{vel}|| > V_{max} \\ 0 & \text{otherwise} \end{cases}$$

where W_{velMax} is the velocity feasibility penalty weight; deliberately set large to discourage trajectories that violate dynamic limits.

Similarly, the acceleration B-Spline is simply the derivative of the velocity B-Spline, and the convex hull property still applies. Therefore, the acceleration control points and, by extension, maximum accelerations are defined as:

$$P_i^{acc} = \frac{2}{int_h}(P_{i+1}^{vel}) - \frac{2}{int_l}(P_i^{vel}) \tag{25}$$

$$int_l = \begin{cases} t_{int_i} \times i & \text{if } i < 3 \\ t_{int_i} \times (n-i) & \text{if } i > (n-3) \\ t_{int_i} \times 2 & \text{otherwise} \end{cases}$$

$$int_h = \begin{cases} t_{int_i} \times (i+1) & \text{if } (i-1) < 3 \\ t_{int_i} \times (n-i-1) & \text{if } i > (n-4) \\ t_{int_i} \times 2 & \text{otherwise} \end{cases}$$

The maximum acceleration is constrained to dynamic limits by:

$$F_{accMax}(x) = \sum_{i=0}^{n-2} swW_{accMax}(|P_i^{acc}| - A_{max})^2 \tag{26}$$

$$sw = \begin{cases} 1 & \text{if } ||P_i^{acc}|| > A_{max} \\ 0 & \text{otherwise} \end{cases}$$

where W_{accMax} is the acceleration feasibility penalty weight; deliberately set large to discourage the optimizer from exceeding dynamic limits.

Lastly, continuity constraints must also be added to ensure smoothness when multiple B-Splines define the trajectory. For this, it is important to guarantee the B-Splines are continuous up to the second derivative at the joints. Thus, the overall continuity cost function can be defined as:

$$F_{continuity}(x) = F_{velCont}(x) + F_{accCont}(x) \tag{27}$$

Note that it is unnecessary to add a position constraint in the continuity cost function; let P_i^j denote control point i of B-Spline j. Because the B-Spline is clamped, it is sufficient to set $P_0^{i+1} = P_{n_i}^i$ for $i \in [0, n_s - 1]$ to guarantee end-to-end position constraints. The optimization can then proceed by using $P_{n_i}^i$ in place of P_0^{i+1}. Velocity and acceleration constraints are handled similarly to initial and final velocity constraints, namely:

$$F_{velCont}(x) = \sum_{i=0}^{n_s-1} W_{velCont}||P_{n_i-1}^{vel_i} - P_0^{vel_i+1}|| \tag{28}$$

$$F_{accCont}(x) = \sum_{i=0}^{n_s-1} W_{accCont}||P_{n_i-1}^{acc_i} - P_0^{acc_i+1}|| \tag{29}$$

Because the planning process uses an incrementally built ESDF for collision safety, trajectories may become unsafe when new obstacles are observed and added to the map. Collision checks are therefore continuously performed at 20Hz on the currently executing trajectory to ensure the safety of the UAV. Since the ESDF encodes the distance to the nearest obstacles, the trajectory is collision-free as long as $d_{ESDF}^{C(t)} > d_{uav} + d_{tolerance}$ for $t \in [0, t_m]$ is guaranteed, where $d_{ESDF}^{C(t)}$ is the ESDF distance at point $C(t)$, d_{uav} is the radius of the UAV, and $d_{tolerance}$ is the collision tolerance. In general, the thresholds are set such that $d_{col} > d_{uav} + d_{tolerance}$ to reduce the probability of a generated trajectory failing the collision check.

If a new collision is found, the trajectory can be split into two portions; the committed and uncommitted portion, as shown in Figure 10. The committed portion consists of two seconds of the trajectory immediately following the current time, and the remainder of the trajectory is the uncommitted portion. Modifying the committed portion of the trajectory generally requires aggressive control inputs from the trajectory follower and, therefore, should be avoided unless there is an immediate collision threat. The uncommitted portion, however, can be modified at any time without impacting control.

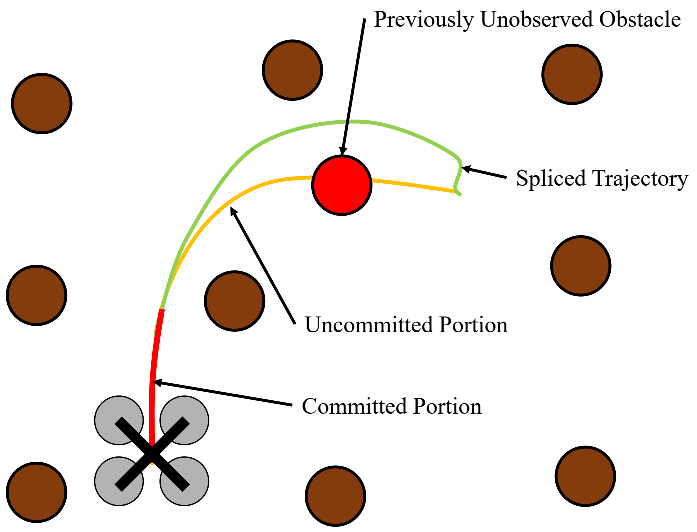


Figure 10. Top down view of a typical replanning event, showing the elements of a newly observed obstacle and a splice event.

If the collision lies within the committed portion, the current trajectory is immediately aborted by splicing a soft-stop trajectory one second into the future. This soft-stop splice consists of a fixed number of control points with the start and end clamped to the splice location. Following this, a new trajectory is then generated from the current location to reach the goal waypoint. A less disruptive trajectory splice can be performed if the collision lies in the uncommitted portion. In this mode, a new trajectory is generated at the transition point between the committed and uncommitted portions such that the initial state of the newly optimized trajectory matches the currently executing trajectory at the splice point. This freshly generated trajectory can then be seamlessly spliced into the uncommitted portion of the current trajectory and does not produce a performance penalty, as the splice occurs outside of the control horizon.

2.3. Trajectory Following

Once a trajectory has been generated, the UAV needs to be able to execute the generated trajectory accurately. Because of the groundwork laid in Section 2.2, it can be assumed that a collision-free, dynamically feasible trajectory is available ahead of time. A Model Predictive Control (MPC) trajectory controller using velocity setpoints for multi-rotor UAVs is used. The trajectory follower used is outlined in our previous paper [42]; therefore, this section will only briefly summarize the method. ACADO toolkit's [43] code generation tool is used to generate an efficient solver that is run every timestep for the MPC implementation. In contrast to classical control methods for trajectory tracking, which are reactive in nature and can only attempt to minimize the current error, MPC is predictive, and attempts to minimize both the current and future error. This allows the controller to more accurately track complex trajectories that are known ahead of time.

The aim of this controller is to drive the body frame $\{\boldsymbol{B}\}$ along a trajectory in SE3 space described by one or more B-Splines $\boldsymbol{C}(\boldsymbol{t})$ in the inertial frame $\{\boldsymbol{I}\}$ such that $\boldsymbol{C}(\boldsymbol{t}) = {}^{B}\boldsymbol{P}_I^{sp}$ at time $\boldsymbol{t}$. Consider a simplified version of the cascaded controller shown below in Figure 11. The controller can effectively be split into two portions, the unified portion acting in inertial frame $\{\boldsymbol{I}\}$, and the airframe-specific portions acting in body frame $\{\boldsymbol{B}\}$. In many existing works, setpoints are injected at the angular rate level as each additional control loop produces additional latency. This work injects the setpoint at the velocity loop, producing distinct advantages. First, the setpoint is injected into the unified portion of the controller, so any airframes capable of tracking arbitrary velocity setpoints can be controlled without any modifications. Furthermore, attitude states no longer need to be tracked, as all control now occurs in the inertial frame.

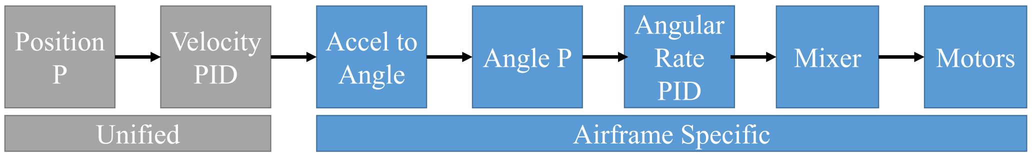


Figure 11. Simplified version of the forward paths of the cascaded multi-rotor controller with unified and airframe-specific portions highlighted.

Assuming the velocity loop is tuned appropriately, the entire airframe-specific portion can be collapsed into a first-order approximation, namely:

$$\frac{{}^{B}\dot{\boldsymbol{P}}_I}{{}^{B}\dot{\boldsymbol{P}}_I^{sp}} = \frac{\boldsymbol{1}}{\boldsymbol{\tau s + 1}} \tag{30}$$
$$\boldsymbol{\tau} = \{\boldsymbol{\tau_X}, \boldsymbol{\tau_Y}, \boldsymbol{\tau_Z}\}^T$$

where ${}^{B}\dot{\boldsymbol{P}}_I$ is the velocity of body frame $\{\boldsymbol{B}\}$ in inertial frame $\{\boldsymbol{I}\}$, ${}^{B}\dot{\boldsymbol{P}}_I^{sp}$ is the velocity setpoint, and $\boldsymbol{\tau} \in \mathbb{R}^3$ is a vector of the velocity time constants in the body frame $\{\boldsymbol{B}\}$. The overall architecture of the controller is presented in Figure 12, with position and velocity control provided by PX4's internal controllers and feedback via PX4's ECL.

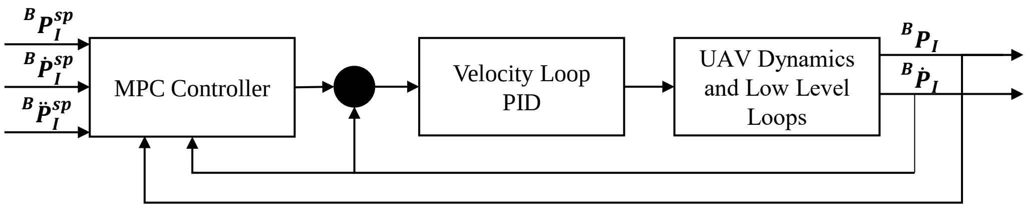

${}^{B}\boldsymbol{P}_I^{sp}$
${}^{B}\dot{\boldsymbol{P}}_I^{sp}$
${}^{B}\ddot{\boldsymbol{P}}_I^{sp}$
MPC Controller
Velocity Loop PID
UAV Dynamics and Low Level Loops
${}^{B}\boldsymbol{P}_I$
${}^{B}\dot{\boldsymbol{P}}_I$

Figure 12. Proposed controller structure.

the state vector $\boldsymbol{x}$, and the system outputs $\boldsymbol{y}$ is defined as:

$$\boldsymbol{x} = \boldsymbol{y} = \{{}^B\boldsymbol{P}_I, {}^B\dot{\boldsymbol{P}}_I, {}^B\dot{\boldsymbol{P}}_I^{sp}\}^T \tag{31}$$

where ${}^B\dot{\boldsymbol{P}}_I \in \mathbb{R}^3$ is the derivative of the position vector ${}^B\boldsymbol{P}_I$, and ${}^B\dot{\boldsymbol{P}_I}^{sp}$ is the reference velocity from the trajectory planner. The control input to the system is defined as:

$$\boldsymbol{u} = \{{}^B\ddot{\boldsymbol{P}}_I^{sp}\} \tag{32}$$

where ${}^B\ddot{\boldsymbol{P}_I}^{sp}$ is the derivative of the velocity setpoint. Furthermore the reference trajectory can be packaged to define the reference system state $\boldsymbol{x}^{ref}$ and $\boldsymbol{u}^{ref}$:

$$\boldsymbol{x}^{ref}(t) = \{{}^B\boldsymbol{P}_I^{ref}, {}^B\dot{\boldsymbol{P}}_I^{ref}, {}^B\dot{\boldsymbol{P}}_I^{ref}\}^T = \{\boldsymbol{C}(t), \dot{\boldsymbol{C}}(t), \dot{\boldsymbol{C}}(t)\} \tag{33}$$

$$\boldsymbol{u}^{ref}(t) = \{{}^B\ddot{\boldsymbol{P}_I}^{ref}\} = \{\ddot{\boldsymbol{C}}(t)\} \tag{34}$$

where ${}^B\ddot{\boldsymbol{P}}_I^{ref}$, ${}^B\dot{\boldsymbol{P}}_I^{ref}$, and ${}^B\boldsymbol{P}_I^{ref}$ are the reference acceleration, velocity, and position of the reference trajectory $\boldsymbol{C}(t)$ at time $\boldsymbol{t}$.

The UAV model is formulated using the transfer function outlined in Equation (30), namely Let $\boldsymbol{A}_m, \boldsymbol{B}_m, \boldsymbol{C}_m, \boldsymbol{D}_m$ describe the linear state-space system

$$\dot{x} = \boldsymbol{A}_m \boldsymbol{x} + \boldsymbol{B}_m \boldsymbol{u} \tag{35}$$

$$\boldsymbol{y} = \boldsymbol{C}_m \boldsymbol{x} + \boldsymbol{D}_m \boldsymbol{u} \tag{36}$$

$$\boldsymbol{A}_m = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{0} & -\boldsymbol{\tau}_{inv}\boldsymbol{I} & \boldsymbol{\tau}_{inv}\boldsymbol{I} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} \quad \boldsymbol{B}_m = \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{0} \\ \boldsymbol{I} \end{bmatrix} \quad \boldsymbol{C}_m = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{I} \end{bmatrix} \quad \boldsymbol{D}_m = \boldsymbol{0}$$

where $\boldsymbol{I} \in \mathbb{R}^{3\times3}$ is the 3 by 3 identity matrix, $\boldsymbol{\tau}_{inv} \in \mathbb{R}^3$ is the element-wise inverse of vector $\boldsymbol{\tau}$, and each 0 represents a 3-by-3 matrix of zeros. Let the sample time of the MPC controller be $\boldsymbol{t}_s$ and the length of the optimization horizon be $\boldsymbol{t}_s \times \boldsymbol{N}$ where $\boldsymbol{N}$ is the number of samples; the optimization problem is formulated as:

$$\begin{gathered}
\text{let } \boldsymbol{x}_k^{ref} = \boldsymbol{x}^{ref}(\boldsymbol{k} \times \boldsymbol{t}_s),\ \boldsymbol{u}_k^{ref} = \boldsymbol{u}^{ref}(\boldsymbol{k} \times \boldsymbol{t}_s) \\
\boldsymbol{arg}\,\mathbf{min}\,\boldsymbol{J}(\boldsymbol{x}) = \sum_{k=0}^{N-1}\left(\left(\boldsymbol{x}_k - \boldsymbol{x}_k^{ref}\right)^T \boldsymbol{Q}\left(\boldsymbol{x}_k - \boldsymbol{x}_k^{ref}\right) + \left(\boldsymbol{u}_k - \boldsymbol{u}_k^{ref}\right)^T \boldsymbol{R}\left(\boldsymbol{u}_k - \boldsymbol{u}_k^{ref}\right)\right) \\
+\left(\boldsymbol{x}_N - \boldsymbol{x}_N^{ref}\right)^T \boldsymbol{P}\left(\boldsymbol{x}_N - \boldsymbol{x}_N^{ref}\right) \\
\text{subject to} \\
\boldsymbol{x}_{k+1} = \boldsymbol{A}_d \boldsymbol{x}_k + \boldsymbol{B}_d \boldsymbol{u}_k \\
-\boldsymbol{v}_{max} \le {}^B\dot{\boldsymbol{P}}_I^{ref} \le \boldsymbol{v}_{max} \\
-\boldsymbol{a}_{max} \le {}^B\ddot{\boldsymbol{P}}_I^{ref} \le \boldsymbol{a}_{max}
\end{gathered} \tag{37}$$

where $\boldsymbol{A}_d$ and $\boldsymbol{B}_d$ are the discrete-time variants of the state and input matrices $\boldsymbol{A}_m$ and $\boldsymbol{B}_m$, $\boldsymbol{x}_k$ and $\boldsymbol{u}_k$ is the predicted system state vector and control input vector at sample $\boldsymbol{k}$. $\boldsymbol{Q} \in \mathbb{R}^{9x9}$ and $\boldsymbol{R} \in \mathbb{R}^{3x3}$ are the system state and control input weighting matrices, respectively, and are generally diagonal. $\boldsymbol{P} \in \mathbb{R}^{9x9}$ is the solution to the algebraic Riccati equation. Vectors $\boldsymbol{v}_{max} \in \mathbb{R}^3$ and $\boldsymbol{a}_{max} \in \mathbb{R}^3$ are used to enforce hard limits on the velocity setpoint ${}^B\dot{\boldsymbol{P}}_I^{ref}$ and rate of change of velocity setpoint ${}^B\ddot{\boldsymbol{P}}_I^{ref}$.

To strike a balance between trajectory tracking performance and required onboard compute, the MPC formulation in this work uses sample time $\boldsymbol{t}_s = \mathbf{0.1}$ *s* and control and prediction horizon of 2 s, producing a total of $\boldsymbol{N} = \mathbf{20}$ steps in each optimization.

To compensate for the relatively short optimization horizon, the controller is run at 50Hz using positions linearly interpolated along the reference trajectory.

The formulation outlined so far does not control the UAV's yaw angle, as the UAV's position and yaw angle are assumed to be decoupled. In reality, the yaw angle of the UAV must be continuously controlled to point in the direction of travel to guarantee safety. A simple P controller is used to control the yaw angle of the UAV and determined via Equation (38).

$$\dot{\psi^{sp}} = K_{\psi}(\psi^{sp} - \psi) \tag{38}$$

where ψ is the current yaw angle of the UAV, ψ^{sp} is the yaw setpoint, K_{ψ} is the yaw P gain, and $\dot{\psi^{sp}}$ is the yaw rate setpoint.

The yaw setpoint ψ^{sp} is set such that the sensor is pointed towards the Euclidean center of the trajectory section between 1 to 2 s in the future, as shown in Equation (39).

$$\begin{gathered} \text{let } V(t) = \sum_{i=t+1}^{t+2} (C(t)) \\ \psi^{sp} = atan2(V(t)[1], V(t)[0]) \end{gathered} \tag{39}$$

where $V(t)[1]$ is the y-axis value, and $V(t)[0]$ is the x-axis value of $V(t)$.

Because it is impossible to guarantee that the sensor is pointed in the correct direction without using an infinitely large yaw gain, a stall event is inserted when the setpoint leaves the horizontal FOV of the sensor. When this happens, the trajectory follower momentarily commands a zero velocity setpoint at the current position, and the yaw controller is given time to reorient the sensor. Trajectory tracking is resumed as soon as the sensor is oriented in the correct direction.

2.4. Runtime

It should be noted that the operating speed of the proposed method is generally limited by the availability of onboard computers, as safety of the UAV is dependent on the method continuously performing collision checks and generating new trajectories. In general, the most time-consuming task is the maintenance of the ESDF map, as a large number of voxels have to be sequentially updated. During simulation, the average optimization time for a single trajectory is approx. 30 ms, and varies with the length of the trajectory, while trajectory tracking takes approx. 250 us per step. In the event that faster cycle times are required, the resolution of the ESDF map or maximum sensor range, and the number of control points in the B-Spline could be reduced at the expense of requiring larger collision tolerances.

3. Simulation Tests

Simulation testing is conducted using Gazebo with PX4's Software-In-The-Loop (SITL) environment via a simulated MID-70 sensor attached to the 3DR Iris model. The environments used for simulation are procedurally generated using the parameters acquired in [34], outlined in Table 1. A simplified view of these parameters in a forest environment are shown in Figure 13. A simulated ground plane is produced using Perlin Noise [44] to create cases with various terrain conditions. Three metrics are used to quantify the degree of each potential obstacle present in the environment, namely the amount of branching, the slope of the ground, and the roughness of the ground. Each of these metrics represents a relative comparison in difficulty, with higher values indicating a more challenging environment for traversal.

Table 1. Summary of parameters extracted from plantation forest point clouds.

Parameter	Distribution Type	Mean(m)/a	SD(m)/b
Row spacing (m)	Gaussian	4.42	0.37
Row deviation (m)	Gaussian	0.01	0.78
Tree spacing (m)	Gamma	2.61	2.24
Branch length (Low-branching) (m)	Gamma	2.94	0.37
Branch length (High-branching) (m)	Gamma	7.31	0.37
Branch height (m)	Gaussian	4.76	1.01
Branch Elevation Angle (rad)	Gaussian	0.23	0.62
Tree diameter (m)	Gaussian	0.52	0.14

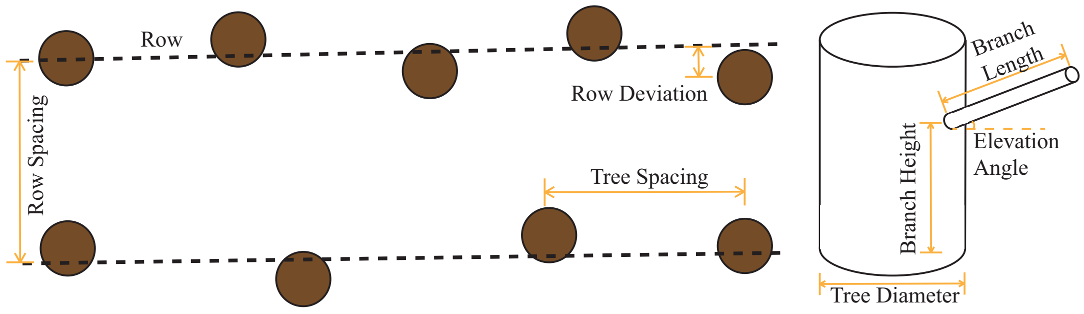


Figure 13. Illustration of parameters modeled in Table 1.

3.1. Branching

Aside from the stems, the most significant considered source of obstacles are low-hanging branches in the environment; branches that directly intrude on the available space between two rows of stems produce significantly more impact than those parallel to the rows. In particular, diagonal branches have a more significant effect than vertical or horizontal branches by creating larger zones where the UAV cannot traverse. Therefore, the branching metric is defined as the normalized sum of rectangular areas occupied in the perpendicular direction of the row as outlined in Equation (40).

Let θ_j, ϕ_j, l_j be the elevation and azimuth angle, and length of branch j

$$D_B(T) = \frac{1}{numTree} \sum_{i=0}^{numTree} \left(\sum_{j=0}^{numBranch} (\sin(\phi_j)(l_j)^2 \sin(\theta_j) \cos(\theta)) \right) \tag{40}$$

where $numTree$ is the number of trees in the environment, and $numBranch$ is the number of branches corresponding to tree i.

3.2. Slope and Roughness

Since elevation changes heavily influence row segmentation and waypoint selection, any slope in the environment will increase the traversal difficulty. To define metrics describing the slope and roughness of each simulation testing environment. Consider a general plane representation

$$Z(x,y) = ax + by + c \tag{41}$$

where a, b, c are dimensionless constants representing a least-squares fit of a plane in 3D space. Consider the ground plane as a list of i tuples $G = (x_0, y_0, z_0), (x_1, y_1, z_1), \cdots (x_i, y_i, z_i)$. The slope difficulty metric is described in Equation (42).

Let $\boldsymbol{Ax} = \boldsymbol{b}$ be an over-constrained system

$$\boldsymbol{A} \in \mathbb{R}^{i \times 3}, \boldsymbol{x} \in \mathbb{R}^{3}, \boldsymbol{b} \in \mathbb{R}^{i}$$

where

$$\boldsymbol{A} = \begin{bmatrix} x_0 & y_0 & 1 \\ x_1 & y_1 & 1 \\ & \vdots & \\ x_i & y_i & 1 \end{bmatrix} \boldsymbol{x} = \begin{bmatrix} a \\ b \\ c \end{bmatrix} \boldsymbol{b} = \begin{bmatrix} z_0 \\ z_1 \\ \vdots \\ z_i \end{bmatrix}$$

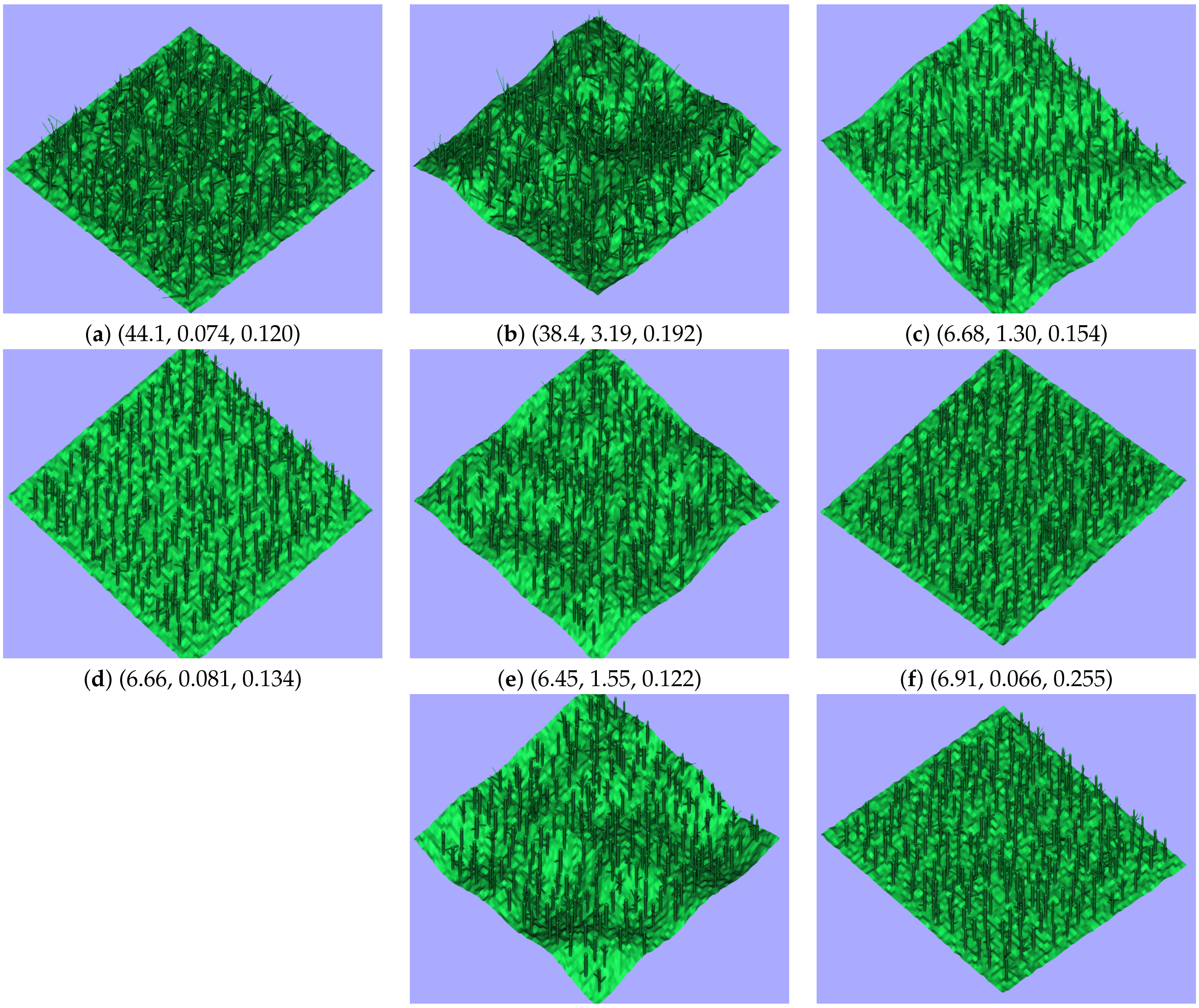
$$D_s(G) = atan(\sqrt{a^2 + b^2}) \tag{42}$$

Roughness is defined as the RMS of the difference between the plane's estimated elevation at position (x, y) compared to the actual ground elevation. The roughness of the ground plane is therefore defined as

$$D_r(G) = \sqrt{\sum_{k=0}^{i} ((F_z(x_k, y_k) - (z_k))^2)} \tag{43}$$

3.3. Simulation Environments

A set of eight simulation environments are selected from a large pool of generated environments, each representing a specific combination of difficulty metrics. These environments are shown in Figure 14, and the metrics describing each one is outlined in Table 2. In particular, test case (h) is the baseline test case with low branching, slope, and roughness. Test cases (a), (d)–(g) are picked to examine the effects of branching, slope, and roughness alone, as each case is picked to minimize the other two difficulty metrics. Lastly, test cases (b) and (c) test the overall performance when subject to mixed test cases.



(**a**) (44.1, 0.074, 0.120) (**b**) (38.4, 3.19, 0.192) (**c**) (6.68, 1.30, 0.154)

(**d**) (6.66, 0.081, 0.134) (**e**) (6.45, 1.55, 0.122) (**f**) (6.91, 0.066, 0.255)

(**g**) (6.57, 3.57, 0.103) (**h**) (6.48, 0.079, 0.076)

Figure 14. Simulation test environments. The three numbers in the subfigure captions correspond to the branching, roughness, and slope metrics.

Table 2. Summary of parameters of the generated forest environments shown in Figure 14.

Label	Branching	Roughness	Slope	Type
(a)	44.1	0.074	0.120	High Branching
(b)	38.4	3.19	0.192	Mixed Difficult
(c)	6.68	1.30	0.154	Mixed Medium
(d)	6.66	0.081	0.134	Medium Slope
(e)	6.45	1.55	0.122	Medium Roughness
(f)	6.91	0.065	0.255	High Slope
(g)	6.57	3.57	0.103	High Roughness
(h)	6.48	0.079	0.076	Baseline

3.4. Survey Time

The first set of simulated tests is conducted to examine the effects on survey time of the proposed method against changes in the environment. In this series of experiments, the simulated UAV is initially placed between two rows and tasked to explore five corridors, with a traversal distance of 20 m down each corridor for an overall survey area of approx. 22 m by 22 m with a target survey speed of 1m/s. Once the traversal of the required number of corridors is complete, the UAV is tasked with returning to the starting location. To separate overhead effects from the UAV returning to the origin and traversal down the five corridors, the recorded time is split into the survey and return phases, which are timed separately. The time taken for the survey phase of each test environment is shown in Figure 15, the sensitivity of survey times to each difficulty metric is shown in Figure 16, and the return times are in Figure 17. To prevent samples that do not fully explore the environment from skewing the results with faster surveys, surveys that do not observe every tree are marked as failed and not included. The final survey results are tabulated in Table 3.

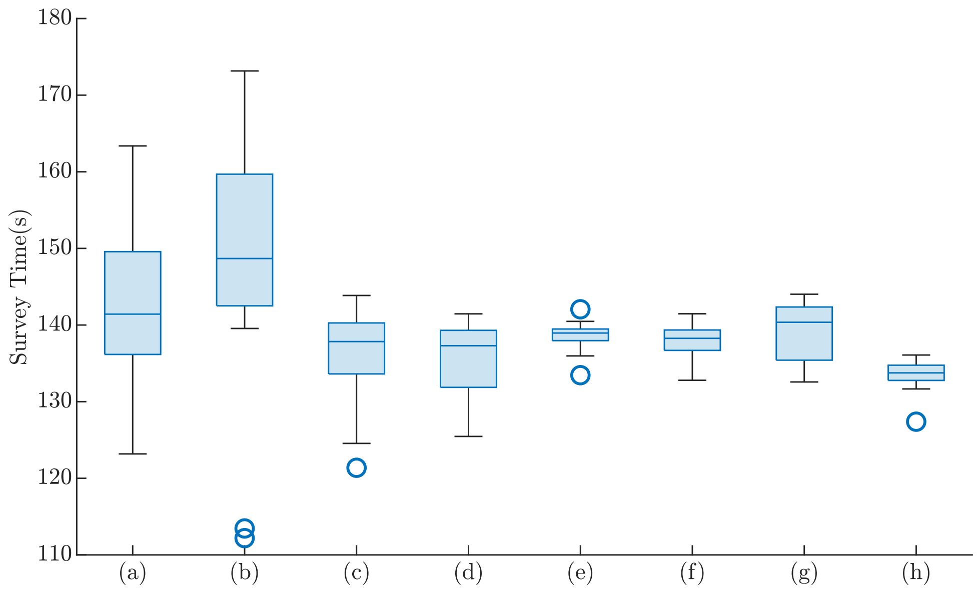


Figure 15. Summary of survey time for each test environment. Trials that did not produce full coverage are not included. Circles denote outliers in the survey time.

Table 3. Summary of survey time for each test environment.

Label	Trails	Incomplete	Mean (s)	SD (s)	Min (s)	Max (s)
(a)	26	4	143	10.0	123	163
(b)	26	11	147	16.9	112	173
(c)	28	1	137	5.2	121	144
(d)	30	0	136	4.5	125	141
(e)	30	0	139	1.5	133	142
(f)	28	0	138	2.3	133	141
(g)	29	0	139	3.7	133	144
(h)	26	0	134	1.7	127	136
Theoretical Minimum *	-	0	120	0	120	-

* Assuming a fixed survey speed of 1m/s and a traversal of 20 m down five rows.

Inspection of Figure 15 shows that aside from environments (a) and (b), the proposed method produces similar survey times, both in mean and spread for all other test environments. In particular, the worst performing environment (c) produces a 3.7% increase in mean survey time and an approx. 200% increase in standard deviation compared to the baseline environment (h). The mixed difficulty samples (a) and (b), however, produce noticeably higher mean and standard deviation increases of 6.7%/9.7% and 488%/894%

respectively, with a corresponding increase in the number of incomplete surveys. Since environments (a) and (b) are the only ones with increased branching, this suggests that branching is the only parameter out of the three tests that affect traversal performance. Figure 16 supports this finding as there appears to be no visible correlation between survey time, slope, or roughness, and only a weak correlation with branching is seen. Little variation is seen in the return times outlined in Figure 17, with differences across operating environments likely arising from differences in the test scenes. This suggests that the proposed method is robust against various environmental conditions.

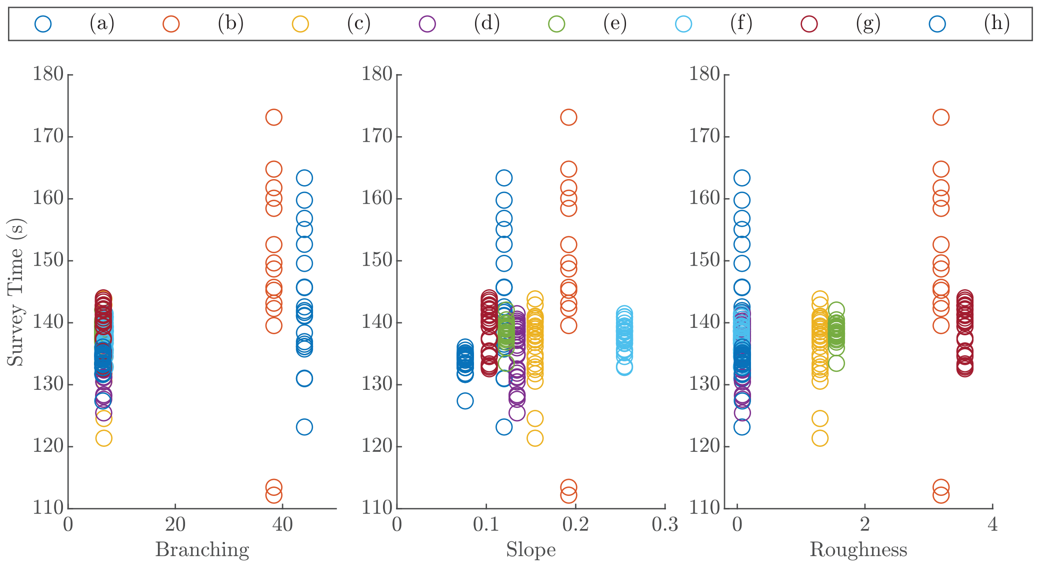


Figure 16. Effects on survey time with varying test environments.

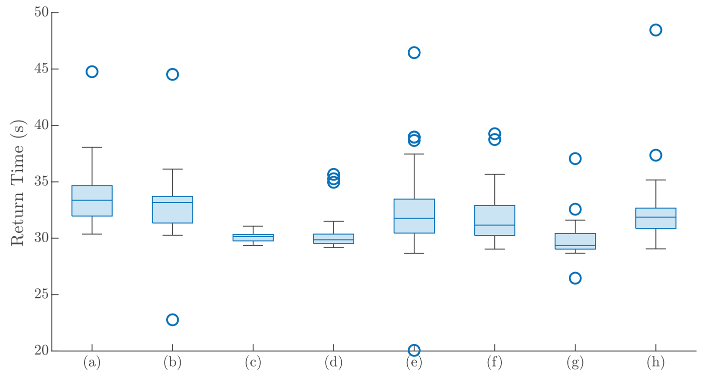


Figure 17. Summary of return times for each test environment with incomplete trials removed.

3.5. Coverage

The second important metric to evaluate for performance is the coverage of the environment, particularly the number of observed points corresponding to each tree. Thus, this series of experiments explores the sensitivity of coverage performance to varying environments for the proposed methods. Recorded data from the flights outlined in Section 3.4 are used to determine the number of points in a 1m by 1m square centered at the reference location of each tree. To eliminate run-to-run variation, the densities around each tree in the environment is averaged across all runs, providing the results shown in Figure 18 and Table 4. Figure 19 shows the distribution of coverage within each test environment. Points from the return portion of the flight are not included to avoid inflating coverage of trees observable from the return path.

Table 4. Summary of coverage for each test environment.

Label	Mean (s)	SD (s)	Min (s)	Max (s)
(a)	2055	687	1099	3453
(b)	757	492	139	2218
(c)	1477	600	532	2629
(d)	1188	748	137	3551
(e)	1830	797	707	3894
(f)	1540	617	631	3044
(g)	1094	289	435	1500
(h)	1625	682	699	3400

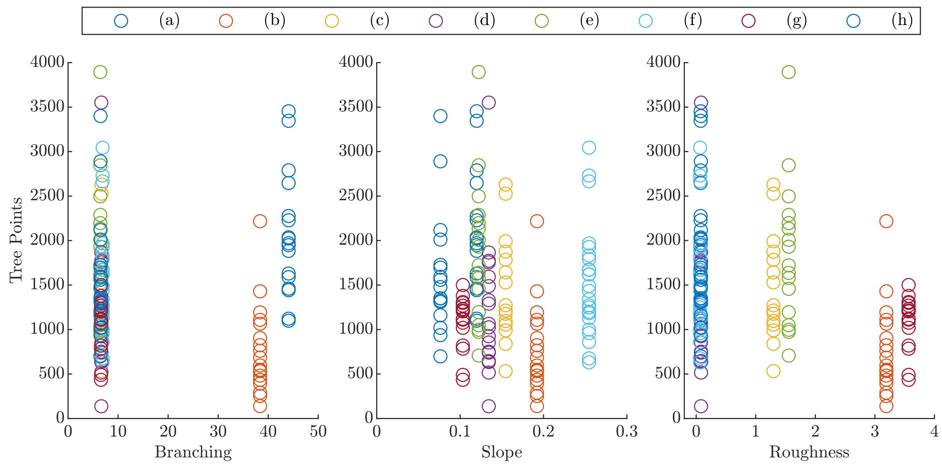


Figure 18. Effects on coverage when the three test parameters are varied.

Inspecting Table 4 shows two groups of mean coverages. The first of these groups consists of those with high mean points (>1400/stem) consisting of test environments (a), (c), (e), (f), and (h), and the second group consists of low mean points (<1200/stem) consisting of environments (b), (d), (g). Except for sample (d), both samples (b) and (g) consist of high-roughness environments. This suggests that this difference can be partially attributed to the limited LiDAR FOV, as rough terrain would likely result in more LiDAR points striking the ground within the FOV of any forward-facing inclines. This is supported by Figure 18, which shows a decrease in coverage with increasing roughness, but no clear correlation to the other two parameters.

All test environments appear to offer relatively even coverage when looking at Figure 19. Interestingly, however, the spread of coverage does not appear to be governed by roughness, as the high-roughness test case (g) offers the lowest standard deviation and visually the most consistent point density around each tree. Similarly, samples (b) and (d) appear to have a low point count but offer relatively similar coverage across all observed stems. Nevertheless, all trees within the environment have been observed at least once, suggesting that the proposed method generally produces good coverage across various test conditions.

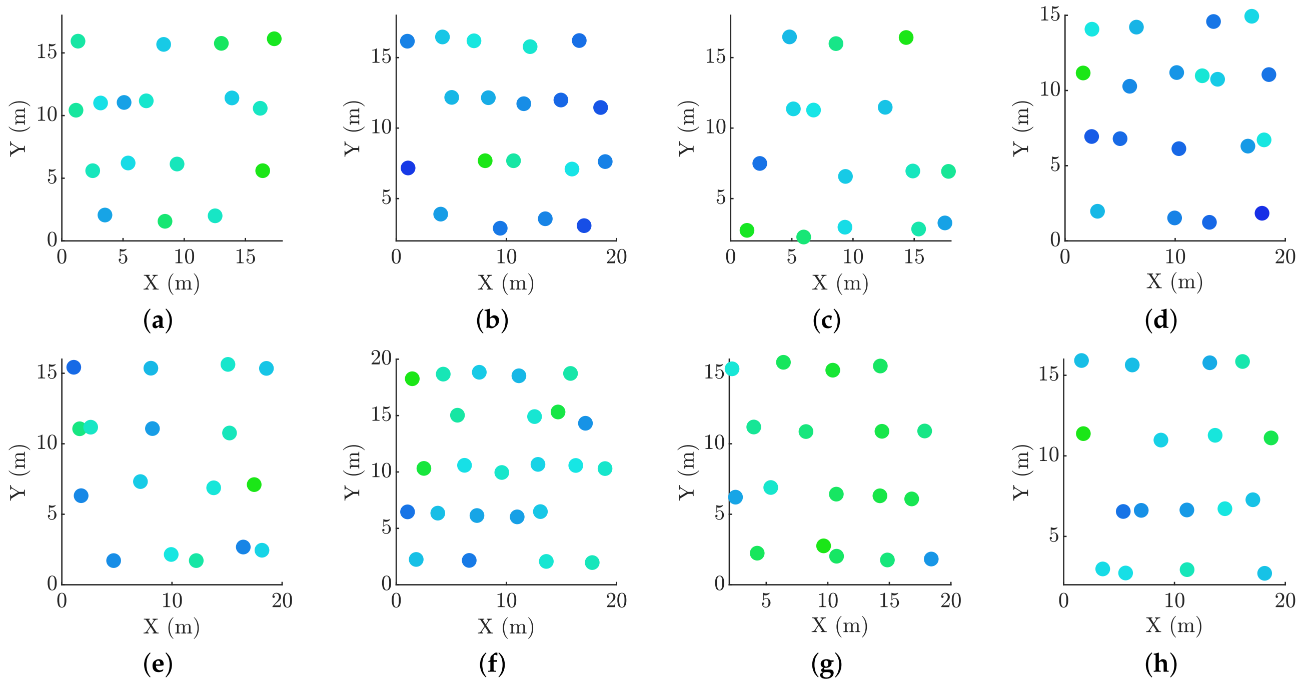


Figure 19. Averaged coverage maps for trials in environments (**a**–**h**), normalized to the maximum of each specific sample. Blue hues indicate trees with fewer observed points, and green hues indicate more observed points. Non-uniformity is shown as the difference in hues.

3.6. Effects of Survey Speed

As one of the planner's objectives is to produce fixed-speed trajectories, an additional tuning factor is the target survey speed. In this series of tests, the proposed method is used on the baseline test environment (g) and tasked to explore the environment at target speeds between 1 m/s to 4 m/s in 1 m/s increments. Figure 20 shows the resulting survey time and tree coverages. Expectedly, survey time and coverage decrease logarithmically as the survey speed increases, with a decrease of 44% between target speeds of 1 m/s and 2 m/s, decreasing to 23% between 2 ms/ and 3 m/s, and no decrease between 3 m/s and 4 m/s. Environment coverage shows the same trend, with a reduction of approx. 39% between survey speeds of 1 m/s and 2 m/s, but no further noticeable decreases from speeds of 2 m/s and beyond.

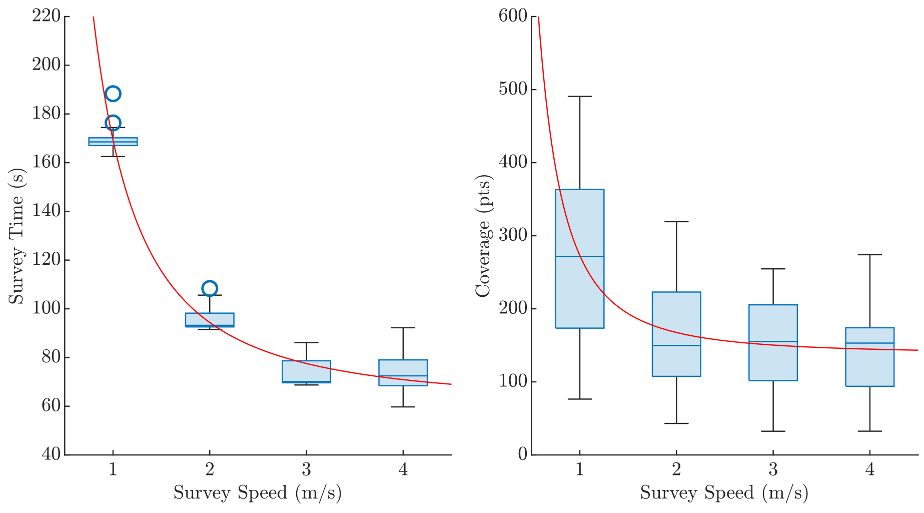


Figure 20. Effects on survey time (**left**) and coverage (**right**) with varying speed. Exponential fit through all results is shown as a red line.

Interestingly, the spread of survey times at a survey speed of 4 m/s is significantly higher when compared to at 3 m/s, with an increased spread of 88%. However, effects on the coverage appear to be relatively small at survey speeds above 2 m/s with little change in both coverage, mean, and spread. The increase in the spread and mean survey time between target speeds of 3 m/s and 4 m/s likely arises from the planner producing more unsafe trajectories from unobserved obstacles. This effect can be seen in Figure 21; the average speed of the survey does not increase at a target speed of 4 m/s compared to 3 m/s, but the spread of mean speeds is larger. In all cases, the actual mean speed is lower than the target speed, with the deficit increasing with increasing target speed.

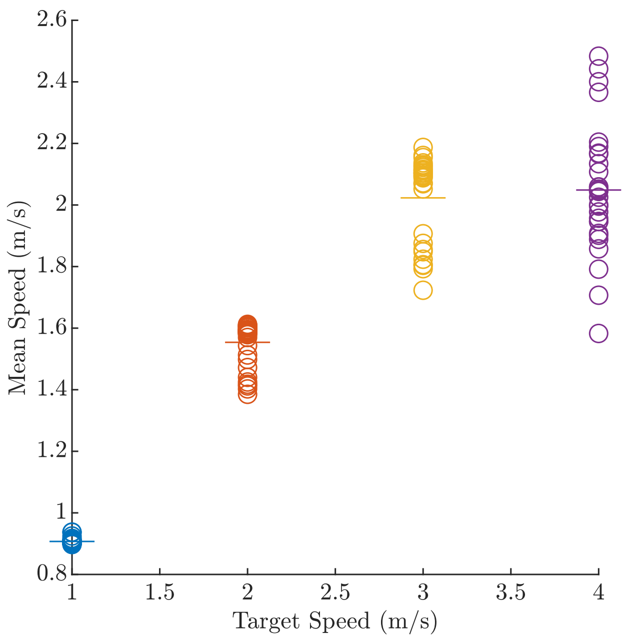


Figure 21. Plot of mean speed vs. target survey speed. Note the plateau in mean speed at 3 m/s and 4 m/s. The averaged mean speed across all samples for each target speed is indicated by a horizontal line.

3.7. Comparison to Existing Methods

Finally, to examine the relative performance of the proposed method against other state-of-the-art exploration planners, an experiment is conducted using FUEL [28] to fully explore the same 22 m by 22 m area in environment (a) as the earlier trials. To ensure consistency between the two tests, the sensor model used in FUEL is updated to match that of the Livox MID-70, and the maximum velocity is limited to 1 m/s. Figure 22 shows the path taken within the environment for the two methods, with the proposed method completing the survey in 169 s, while FUEL took 306 s for completion. This disparity appears to be a result of FUEL needing to explicitly observe the entire environment, which results in a large amount of yaw action and traversal near the boundaries of the exploration volume as a result of the low FOV sensor. This manifests as a less consistent survey path shown in red in Figure 22. In contrast, the proposed method allows for longer straight sections, which allows the UAV to traverse at the target speed of 1 m/s for longer. This experiment demonstrates the performance advantage the proposed traversal method achieves against a high-performance generalized exploration method.

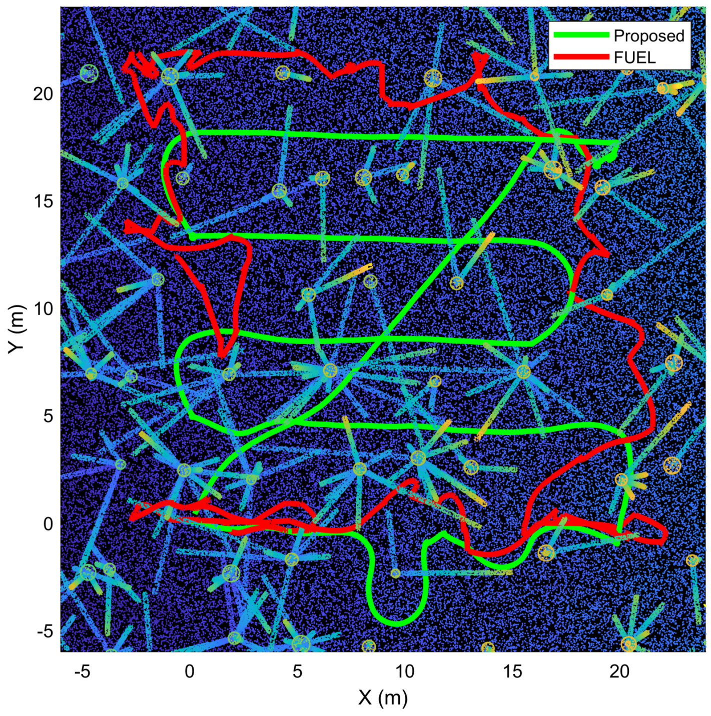


Figure 22. Path taken for complete coverage for the proposed method and FUEL.

4. Flight Tests

Flight testing is performed at a local Pinus Radiata plantation to validate the proposed surveying method. The test site shown in Figure 23 is located in Riverhead Forest, Auckland, New Zealand (−36.725286, 174.547957) and consists of seven parallel rows spaced approximately 4.5 m apart with flat terrain. While there is little undergrowth in the environment, the environment is unpruned and contains many branches down to ground level. Despite branching, sufficient space exists between adjacent rows of trees to allow traversal down and between rows. The presence of trees in the middle of traversed rows produces some corridors which are not continuous.

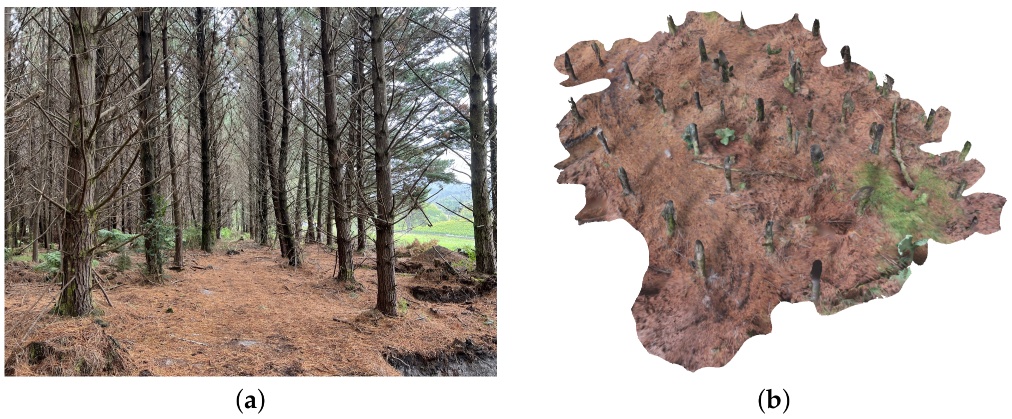


Figure 23. (**a**) Single row view of the plantation site used for flight testing, note the presence of some low-hanging branches, and (**b**) 3D scan of the proposed test site showing the survey region; the scan area is approx. 25 m by 30 m.

Ten flights are conducted to evaluate the method, four of which are large-scale flights intending to cover the entire survey environment. The other six consists of small-scale flights covering smaller areas to test multiple different initial conditions. In addition to the survey and return times outlined previously, these trials also evaluate the number of attempted and completed corridors. These are defined as:

- Attempted corridors—the number of identified corridors and an attempt to traverse these corridors have been made
- Completed corridors—the number of correctly traversed corridors

4.1. Large Flights

Simplified views of the four flights are presented in Figure 24, and top-down views of the flight path overlaid on the reconstructed pointcloud for each flight is shown in Figure 25. Metrics summarizing the performance of each test flight is shown in Table 5. Of the four flights performed, tests (b) and (c) successfully returned to the origin, while (a) and (d) performed a partial and no return from the loss of the pose estimate. The UAV successfully selects waypoints and navigates between three to five corridors within the environment in all four cases. In particular, cases (a) and (b) show more complex routes around multiple trees rather than the mainly straight routes seen while traversing down a single corridor. This verifies the capability of the planner to produce safe trajectories with correctly placed waypoints when operating in a complex plantation environment.

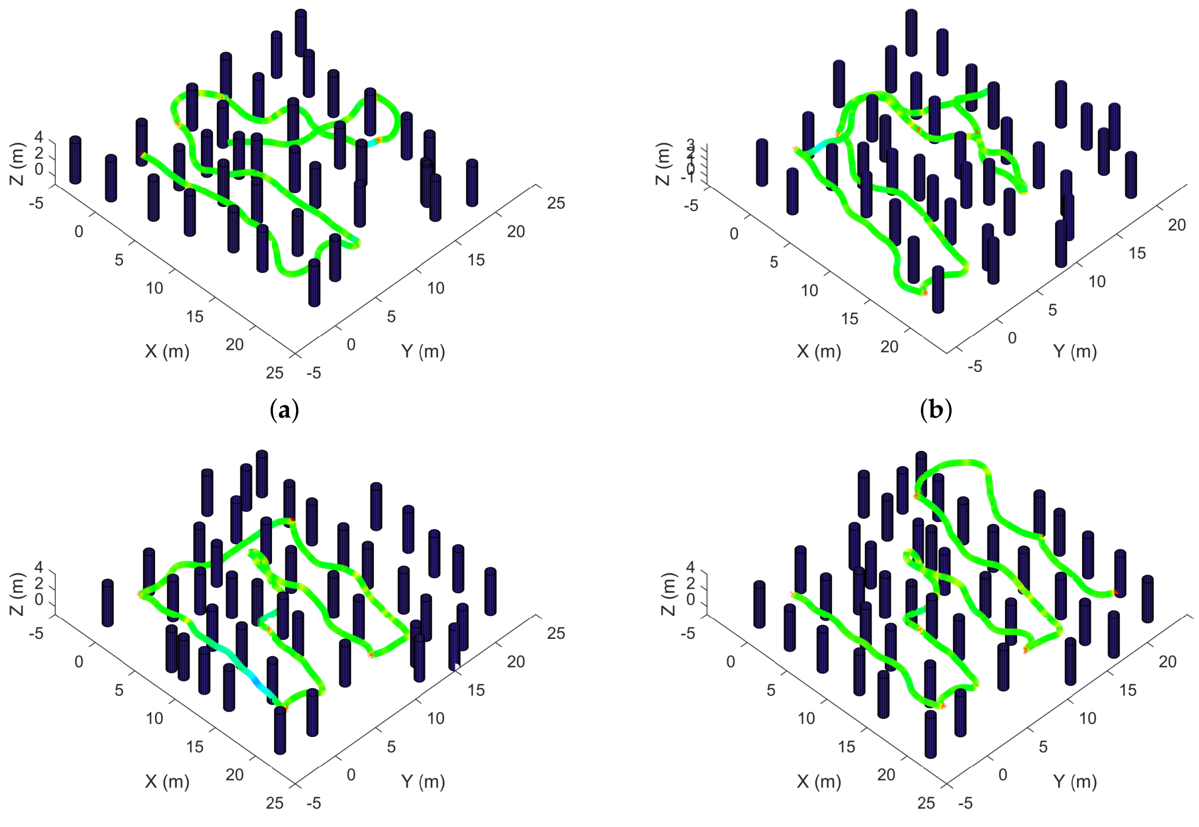


Figure 24. Simplified views of the large-scale test flights (**a**–**d**), the path followed is shown as a solid line overlaid with stem locations within the environment. Greener hues indicate a flight speed closer to the 1 m/s target speed, while red hues indicate slower speeds taken.

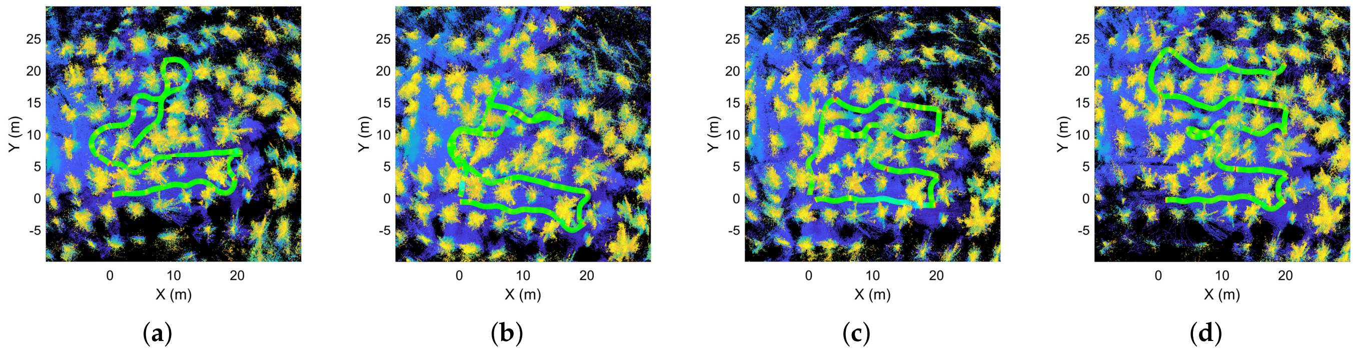


Figure 25. Top-down views of large-scale test flights (**a**–**d**), with the path taken overlaid on the reconstructed point cloud.

Table 5. Summary of large scale flight tests.

Label	Attempted	Completed	Survey Time (s)	Return Time (s)
(a)	3	2	114.5	-
(b)	4	3	135.5	25.5
(c)	4	4	106.4	18.7
(d)	5	5	157.8	-

Looking instead at the speed distribution along the survey path, the UAV maintains the target survey speed of 1 m/s throughout most traversal tests. A significant decrease in survey speed occurs at the last waypoint of each corridor, with smaller drops in regions with less free space and spaces with a large number of obstacles. The speed profile for the survey (d) shown in Figure 26 illustrates these events, with significant drops in survey speed occurring at approximately 30 s, 45 s, 80 s, and 120 s from the start of the survey and more minor fluctuations between these points. The large drops to near zero speed correspond with the end of each corridor and encompass the time taken to produce new waypoints in the neighboring corridor and generate a new trajectory. The smaller decreases in speed are likely caused by one of two things: FOV constraints reducing flight speed until the LiDAR is oriented in the correct direction, or trajectory splice events causing small local discontinuities in the trajectory. Overall, these experiments suggest that the proposed method can often correctly estimate the position of the rows of trees, and the trajectory planner and follower can successfully generate and execute safe trajectories for the survey. However, there is room for improvement of efficiency through optimizing regions with slow survey speed.

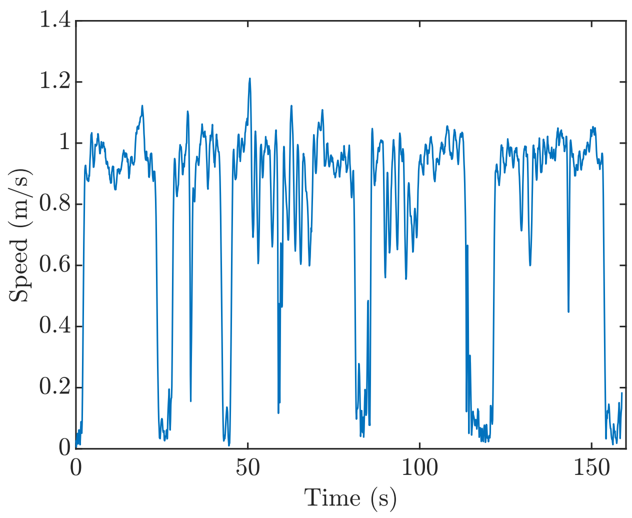


Figure 26. Plot of estimated speed during trial (d).

4.2. Small Flights

Since the size of the test plot is limited, a set of six small flights through the environment is conducted to examine the robustness of the proposed method against different initial conditions. A random starting point with sufficient adjacent rows to complete the survey is selected for each trial. In each task, the UAV uses the proposed method to traverse approx. 12 m down each corridor, with the complete survey consisting of three corridors. The total survey area for each of the trails is approx. 14 m by 14 m. Top-down views of each flight within the reconstructed point cloud are shown in Figure 27, and a performance summary is outlined in Table 6.

Table 6. Summary of small scale flight tests.

Label	Attempted	Completed	Survey Time (s)	Return Time (s)
(a)	3	3	49.8	15.3
(b)	3	3	66.7	20.4
(c)	3	3	61.8	20.2
(d)	3	3	62.9	23.1
(e)	3	3	52.6	20.4
(f)	3	3	69.7	-

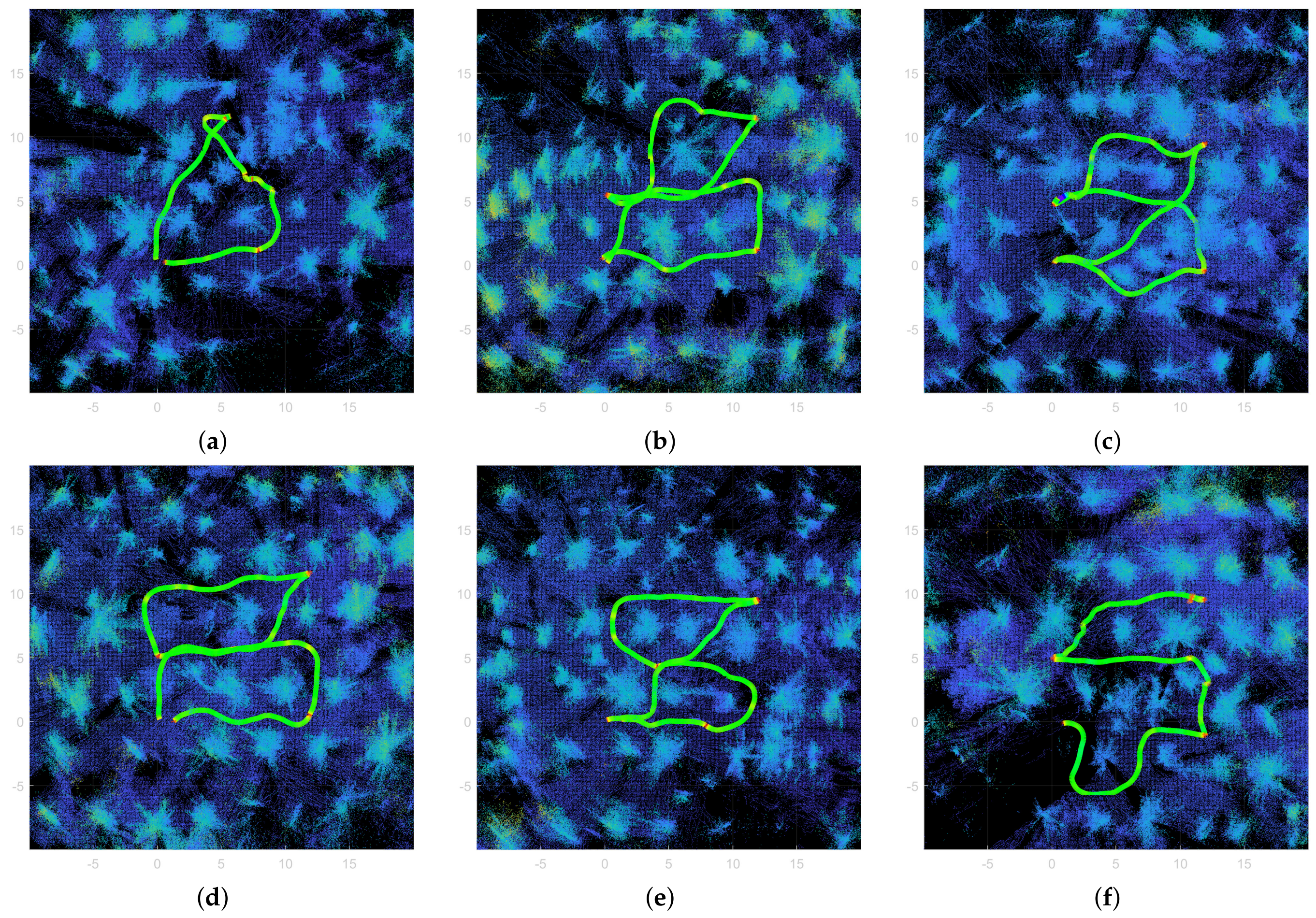


Figure 27. Top-down views of path taken during small-scale test flights (**a**–**f**) overlaid on the reconstructed point clouds. The path taken during the survey is shown as the line.

All flights successfully traversed the target area, providing complete coverage of the trees within the approx. 14 m by 14 m bounding square. In particular, test cases (a), (c), and (f) consist of initial conditions where the UAV is unable to traverse down every corridor in a straight path continuously, and safe trajectories are generated and executed in each case. Aside from (f), all tests produced and executed a safe return trajectory. Furthermore, experiments (a), (c), and (f) demonstrate initial conditions where straight line traversal is not possible due to obstacles present. In all cases, the planner successfully determined a safe trajectory to navigate around these obstacles and complete the survey. These flight experiments demonstrate the robustness of the proposed method against varying conditions and untraversable regions within the scene.

5. Conclusions

This paper presents a system for autonomous exploration of plantation forests using multi-rotor UAVs. First, a method for navigation waypoint generation using estimated tree row positions from occupancy data is presented. Next, a non-linear optimization-based approach is proposed to generate smooth, constant velocity survey trajectories using B-Splines and an ESDF map to create dynamically feasible and collision-free trajectories in complex environments. Lastly, an MPC trajectory tracker is implemented to follow the generated trajectories. Simulation testing is performed using eight procedurally generated forests replicating spacing information from real plantation forests, completing 22 m by 22 m surveys in approx. 140 s. These results demonstrate the proposed method's ability to operate in various environments and initial conditions. Lastly, four large-scale and six small-scale flight tests are successfully performed within a local plantation forest, demonstrating our proposed method's correct row identification, waypoint placement, and navigation capabilities.

The results of this research pave the way for fully autonomous navigation in plantation forests, with potential extension into other row-based environments such as orchards or vineyards. While this study explores the autonomous navigation problem during the survey aspect, the current iteration of the system still requires manual traversal to reach the starting point of the survey. Therefore, future work could consist of investigations into long-range traversal methods allowing for arbitrary starting positions, i.e., from outside of the forest. Furthermore, alternative methods for waypoint generation and/or row segmentation could be explored for additional survey speeds or accuracy. Finally, other navigation methods such as machine-learning-based end-to-end approaches could be explored for potential performance gains in this application.

Author Contributions: Conceptualization, T.-J.L. and K.A.S.; methodology, T-J.L. and K.A.S.; software, T.-J.L.; validation, T.-J.L. and K.A.S.; formal analysis, T.-J.L.; investigation, T.-J.L.; resources, T.-J.L.; data curation, T.-J.L.; writing—original draft preparation, T.-J.L., writing—review and editing, T.-J.L. and K.A.S.; visualization, T.-J.L.; supervision, T.-J.L. and K.A.S.; project administration, T.-J.L. and K.A.S. All authors have read and agreed to the published version of the manuscript.

Funding: The lead author is supported by a University of Auckland Doctoral Scholarship.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data supporting the findings in this paper is available from the corresponding author upon reasonable request.

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

References

1. Deadman, M.W.; Goulding, C.J. A method for assessment of recoverable volume by log types. *N. Z. J. For. Sci.* **1979**, *9*, 225–239.
2. Interpine Innovation. *PlotSafe Overlapping Feature Crusing Forest Inventory Procedures*; Interpine Innovation: Rotorua, New Zealand, 2007; p. 49.
3. Hudak, A.T.; Crookston, N.L.; Evans, J.S.; Hall, D.E.; Falkowski, M.J. Nearest neighbor imputation of species-level, plot-scale forest structure attributes from LiDAR data. *Remote Sens. Environ.* **2008**, *112*, 2232–2245. [CrossRef]
4. Puliti, S.; Dash, J.P.; Watt, M.S.; Breidenbach, J.; Pearse, G.D. A comparison of UAV laser scanning, photogrammetry and airborne laser scanning for precision inventory of small-forest properties. *For. Int. J. For. Res.* **2020**, *93*, 150–162. [CrossRef]

5. Mielcarek, M.; Kamińska, A.; Stereńczak, K. Digital Aerial Photogrammetry (DAP) and Airborne Laser Scanning (ALS) as Sources of Information about Tree Height: Comparisons of the Accuracy of Remote Sensing Methods for Tree Height Estimation. *Remote Sens.* **2020**, *12*, 1808. [CrossRef]
6. Bauwens, S.; Bartholomeus, H.; Calders, K.; Lejeune, P. Forest inventory with terrestrial LiDAR: A comparison of static and hand-held mobile laser scanning. *Forests* **2016**, *7*, 127 . [CrossRef]
7. Kukko, A.; Kaijaluoto, R.; Kaartinen, H.; Lehtola, V.V.; Jaakkola, A.; Hyyppä, J. Graph SLAM correction for single scanner MLS forest data under boreal forest canopy. *ISPRS J. Photogramm. Remote Sens.* **2017**, *132*, 199–209. [CrossRef]
8. Wang, Y.; Kukko, A.; Hyyppä, E.; Hakala, T.; Pyörälä, J.; Lehtomäki, M.; El Issaoui, A.; Yu, X.; Kaartinen, H.; Liang, X.; et al. Seamless integration of above- and under-canopy unmanned aerial vehicle laser scanning for forest investigation. *For. Ecosyst.* **2021**, *8*, 10. [CrossRef]
9. Hyyppä, E.; Hyyppä, J.; Hakala, T.; Kukko, A.; Wulder, M.A.; White, J.C.; Pyörälä, J.; Yu, X.; Wang, Y.; Virtanen, J.P.; et al. Under-canopy UAV laser scanning for accurate forest field measurements. *ISPRS J. Photogramm. Remote Sens.* **2020**, *164*, 41–60. [CrossRef]
10. Hyyppä, J.; Yu, X.; Hakala, T.; Kaartinen, H.; Kukko, A.; Hyyti, H.; Muhojoki, J.; Hyyppä, E. Under-Canopy UAV Laser Scanning Providing Canopy Height and Stem Volume Accurately. *Forests* **2021**, *12*, 856. [CrossRef]
11. Del Perugia, B.; Krisanski, S.; Taskhiri, M.S.; Turner, P. Below-canopy UAS photogrammetry for stem measurement in radiata pine plantation. *Proc. Remote Sens. Agric. Ecosyst. Hydrol.* **2018**, 11, 1078309. [CrossRef]
12. Krisanski, S.; Taskhiri, M.S.; Turner, P. Enhancing Methods for Under-Canopy Unmanned Aircraft System Based Photogrammetry in Complex Forests for Tree Diameter Measurement. *Remote Sens.* **2020**, *12*, 1652. [CrossRef]
13. Kuželka, K.; Surový, P. Mapping Forest Structure Using UAS inside Flight Capabilities. *Sensors* **2018**, *18*, 2245. [CrossRef] [PubMed]
14. Jiang, S. Towards Autonomous Flights of an Unmanned Aerial Vehicle (UAV) in Plantation Forests. Master's Thesis, The University of Auckland, Auckland, New Zealand, 2016.
15. Chiella, A.C.B.; Machado, H.N.; Teixeira, B.O.S.; Pereira, G.A.S. GNSS/LiDAR-Based Navigation of an Aerial Robot in Sparse Forests. *Sensors* **2019**, *19*, 4061. [CrossRef]
16. Chisholm, R.A.; Cui, J.; Lum, S.K.Y.; Chen, B.M. UAV LiDAR for below-canopy forest surveys. *J. Unmanned Veh. Syst.* **2013**, *1*, 61–68. [CrossRef]
17. Cui, J.Q.; Lai, S.; Dong, X.; Liu, P.; Chen, B.M.; Lee, T.H. Autonomous navigation of UAV in forest. In Proceedings of the 2014 International Conference on Unmanned Aircraft Systems, ICUAS 2014—Conference Proceedings, Orlando, FL, USA, 27–30 May 2014. [CrossRef]
18. Thrun, S.; Montemerlo, M. The graph SLAM algorithm with applications to large-scale mapping of urban structures. *Int. J. Robot. Res.* **2006**, *25*, 403–429. [CrossRef]
19. Zucker, M.; Ratliff, N.; Dragan, A.D.; Pivtoraiko, M.; Klingensmith, M.; Dellin, C.M.; Bagnell, J.A.; Srinivasa, S.S. CHOMP: Covariant Hamiltonian optimization for motion planning. *Int. J. Robot. Res.* **2013**, *32*, 1164–1193. [CrossRef]
20. Oleynikova, H.; Burri, M.; Taylor, Z.; Nieto, J.; Siegwart, R.; Galceran, E. Continuous-time trajectory optimization for online UAV replanning. In Proceedings of the IEEE International Conference on Intelligent Robots and Systems 2016, Daejeon, Korea, 9–14 October 2016; pp. 5332–5339. [CrossRef]
21. Usenko, V.; Von Stumberg, L.; Pangercic, A.; Cremers, D. Real-time trajectory replanning for MAVs using uniform B-splines and a 3D circular buffer. In Proceedings of the IEEE International Conference on Intelligent Robots and Systems, Vancouver, BC, Canada, 24–28 September 2017 , pp. 215–222. [CrossRef]
22. Zhou, B.; Pan, J.; Gao, F.; Shen, S. RAPTOR: Robust and Perception-Aware Trajectory Replanning for Quadrotor Fast Flight. *IEEE Trans. Robot.* **2021**, vol. *37*, pp 1992–2009 . [CrossRef]
23. Mellinger, D.; Kumar, V. Minimum snap trajectory generation and control for quadrotors. In Proceedings of the IEEE International Conference on Robotics and Automation, Shanghai, China, 9–13 May 2011; pp. 2520–2525. [CrossRef]
24. Tordesillas, J.; Lopez, B.T.; How, J.P. FASTER: Fast and Safe Trajectory Planner for Flights in Unknown Environments. In Proceedings of the 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Macau, China, 3–8 November 2019, pp. 1934–1940. [CrossRef]
25. Deits, R.; Tedrake, R. Efficient mixed-integer planning for UAVs in cluttered environments. In Proceedings of the IEEE International Conference on Robotics and Automation, Seattle, WA, USA, 26–30 May 2015; pp. 42–49. ICRA.2015.7138978. [CrossRef]
26. Gao, F.; Wang, L.; Zhou, B.; Zhou, X.; Pan, J.; Shen, S. Teach-Repeat-Replan: A Complete and Robust System for Aggressive Flight in Complex Environments. *IEEE Trans. Robot.* **2020**, *36*, 1526–1545. [CrossRef]
27. Meng, Z.; Qin, H.; Chen, Z.; Chen, X.; Sun, H.; Lin, F.; Ang, M.H. A Two-Stage Optimized Next-View Planning Framework for 3-D Unknown Environment Exploration, and Structural Reconstruction. *IEEE Robot. Autom. Lett.* **2017**, *2*, 1680–1687. [CrossRef]
28. Zhou, B.; Zhang, Y.; Chen, X.; Shen, S. FUEL: Fast UAV Exploration using Incremental Frontier Structure and Hierarchical Planning. *IEEE Robot. Autom. Lett.* **2021**, *6*, 779–786. [CrossRef]
29. Dharmadhikari, M.; Dang, T.; Solanka, L.; Loje, J.; Nguyen, H.; Khedekar, N.; Alexis, K. Motion Primitives-based Path Planning for Fast and Agile Exploration using Aerial Robots. In Proceedings of the 2020 IEEE International Conference on Robotics and Automation (ICRA), Paris, France, 31 May–31 August 2020; pp. 179–185. [CrossRef]

30. Bircher, A.; Kamel, M.; Alexis, K.; Oleynikova, H.; Siegwart, R. Receding Horizon "Next-Best-View" Planner for 3D Exploration. In Proceedings of the 2016 IEEE International Conference on Robotics and Automation (ICRA), Stockholm, Sweden, 16–21 May 2016; pp. 1462–1468. [CrossRef]
31. Schmid, L.; Pantic, M.; Khanna, R.; Ott, L.; Siegwart, R.; Nieto, J. An Efficient Sampling-Based Method for Online Informative Path Planning in Unknown Environments. *IEEE Robot. Autom. Lett.* **2020**, *5*, 1500–1507. [CrossRef]
32. Papachristos, C.; Khattak, S.; Alexis, K. Uncertainty-aware receding horizon exploration and mapping using aerial robots. In Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA), Singapore, 29 May–3 June 2017; pp. 4568–4575. [CrossRef]
33. Xu, Z.; Deng, D.; Shimada, K. Autonomous UAV Exploration of Dynamic Environments Via Incremental Sampling and Probabilistic Roadmap. *IEEE Robot. Autom. Lett.* **2021**, *6*, 2729–2736. [CrossRef]
34. Lin, T.J.; Stol, K.A. Faster Navigation of Semi-Structured Forest Environments using Multi-Rotor UAVs. *Robotica* **2022**, submitted.
35. Stanford Artificial Intelligence Laboratory. Robotic Operating System. Available online: https://www.ros.org (accessed on 15 June 2022).
36. Lin, J.; Zhang, F. R3LIVE: A Robust, Real-Time, RGB-Colored, LiDAR-Inertial-Visual Tightly-Coupled State Estimation and Mapping Package. In Proceedings of the 2022 International Conference on Robotics and Automation (ICRA), Philadelphia, PA, USA, 23–27 May 2022; pp. 10672–10678. [CrossRef]
37. Zhang, W.; Qi, J.; Peng, W.; Wang, H.; Xie, D.; Wang, X.; Yan, G. An Easy-to-Use Airborne LiDAR Data Filtering Method Based on Cloth Simulation. *Remote Sens.* **2016**, *8*, 501. [CrossRef]
38. Ester, M.; Kriegel, H.P.; Sander, J.; Xu, X. *A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise*; AAAI Press: Palo Alto, CA, USA, 1996; pp. 226–231.
39. Han, L.; Gao, F.; Zhou, B.; Shen, S. FIESTA: Fast Incremental Euclidean Distance Fields for Online Motion Planning of Aerial Robots. In Proceedings of the IEEE International Conference on Intelligent Robots and Systems, Macau, China, 3–8 November 2019; pp. 4423–4430. [CrossRef]
40. Hart, P.E.; Nilsson, N.J.; Raphael, B. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Trans. Syst. Sci. Cybern.* **1968**, *4*, 100–107. [CrossRef]
41. Agarwal, S.; Mierle, K.; Team, T.C.S. Ceres Solver. Available online: https://github.com/ceres-solver/ceres-solver (accessed on 15 June 2022.).
42. Lin, T.J.; Stol, K.A. Fast Trajectory Tracking of Multi-Rotor UAVs using First-Order Model Predictive Control. In Proceedings of the 2021 Australian Conference on Robotics and Automation (ACRA), Melbourne, Australia, 6–8 December 2021.
43. Houska, B.; Ferreau, H.J.; Diehl, M. ACADO toolkit—An open-source framework for automatic control and dynamic optimization. *Optim. Control Appl. Methods* **2011**, *32*, 298–312. [CrossRef]
44. Perlin, K. An Image Synthesizer. In Proceedings of the SIGGRAPH '85: 12th Annual Conference on Computer Graphics and Interactive Techniques, San Francisco, CA, USA, 22–26 July 1985; Association for Computing Machinery: New York, NY, USA, 1985; pp. 287–296. [CrossRef]