Autonomous Aerial Sensor Network Placement in Rainforests: Exploration and Detection of Deployment Locations

MSc Thesis Report

Rita Santos Raminhos

TU Delft

[This page is intentionally left blank]

Autonomous Aerial Sensor Network Placement in Rainforests: Exploration and Detection of Deployment Locations

MSc Thesis Report

by

Rita Santos Raminhos

to obtain the degree of Master of Science
at the Delft University of Technology
to be defended publicly on November 26, 2024 at 15:00

Thesis committee:

Chair:	Prof Dr. G.C.H.E De Croon
Supervisors:	Dr. S. Hamaza
External examiner:	Dr. R.T. Rajan
Place:	Faculty of Aerospace Engineering, Delft
Project Duration:	September, 2023 - October, 2024
Student number:	5607817

Cover: Photogragh by Tim Laman, from the National Geographic image collection

An electronic version of this thesis is available at `http://repository.tudelft.nl/`.

Faculty of Aerospace Engineering · Delft University of Technology

Preface

Although nature commences with reason and ends in experience it is necessary for us to do the opposite, that is to commence with experience and from this to proceed to investigate the reason.

Leonardo da Vinci

This thesis contains the work of the past year of my life. Over a year ago, I was tasked with proposing an autonomous navigation solution to deploy a sensor network, with multiple sensor nodes, in optimized locations in a rainforest environment. This was simultaneously exciting and challenging, since I started out with very limited practical experience, but determined to learn and work with a drone in the MavLab. For the opportunity to work on such an interesting project, with real-life applications and real hardware, I want to thank my supervisor, Salua Hamaza. And also for all the support, patience and guidance throughout the project, as well as giving me the freedom to make my own decisions and trail my own path.

The path was not an easy one and I wish to acknowledge everyone who helped me through it. I am thankful to everyone in the MavLab, for making it a very open and helpful environment, where I could learn so much. To Georg, Giorgia, Mahima and Nils, who I met along the way, I could not be more grateful for all your support and friendship. A special thank you goes to Mahima and Nils, for helping me in the hardest period of this thesis and being there on the day of Dora's first flight.

I came to Delft three years ago. It was the biggest adventure of my life, and the most rewarding one. I made incredible friends, for whom I am truly thankful, and who made Delft feel like home. To Lea, I am so happy that we met on that first introduction day and stayed really good friends. To Chaimae, thank you for all the moments we spent together and the fun trips and for having the patience to practice dutch with me. To Elena, Magda, Sal and Ale, you were like my family in the Netherlands, always checking on me in the last year. To António, Johan, Gigi, Leo, Micha, Sara and Sofia, thank you for keeping me sane in all the study breaks in the library, for supporting me and also all the fun moments. And to Teresa, Diogo and Mariana, your friendship is very important to me and I am so happy we manage to stay close, even apart.

To Michele, my best friend, the last three years were so special because you were there by my side. You were there every day of this thesis. You were by my side to glue and fold and unfold Ikea curtains, to help me 3d print and debug and brainstorm, to feed me pasta and to hug me. You were so patient and taught me so much, about everything. Grazie mille, per tutto, eu gosto muito de ti.

I dedicate this thesis to the most important people in my life, my parents and Mariana. You are my unconditional support, I am so grateful, I really could not have done this without you. Thank you for always believing in me, even when I did not, and pushing me to work hard and pursue my dreams. Obrigada, eu adoro-vos.

Contents

1 Introduction 1

I Scientific Article 3

II Literature Review 21

References 59

1 Introduction

Recent studies predict that global warming will have severe consequences for animal species that inhabit tropical rainforests, including resident bird populations [1, 2]. Therefore, scientists are increasingly concerned with researching new solutions and technologies to perform biodiversity surveys in these environments. The aim of these technologies is to sense and collect data *in situ*, which can then be used for more accurate and extensive monitoring and conservation studies.

One of the main approaches taken by researchers to conduct remote sensing in environments such as rainforests is the deployment of wireless sensor networks. Large wireless sensor networks, with multiple sensor nodes that are placed directly in the environment, can collect larger amounts of data, over longer periods of time, compared to other approaches, such as mobile sensing with a moving platform [3]. With regards to monitoring bird populations, acoustic sensor networks are among the most widely researched options [4, 5]. The main reasons are the lower dimensions of the data compared to visual sensors and novel post-processing methods using machine learning to distinguish between species [6]. Thus, the data obtained can provide deeper insights on the effects of climate change, for different bird populations.

Placement of acoustic sensor networks in the rainforest presents with unique challenges related to 1) the characteristics of the environment, such as remoteness and very dense vegetation, 2) the network architecture, for example, the power, sensing and communication capabilities of the sensor nodes and 3) the deployment process itself. The first problem can be addressed if the deployment is performed autonomously, using a quadrotor platform, and the sensor nodes are placed on top of the rainforest canopy, where bird sounds are actually detected with less interference. Besides allowing access to hard-to-reach areas, autonomous navigation in the emergent layer of the rainforest is less challenging as the environment is significantly less cluttered. It is also less computationally demanding, as GPS-guided navigation is possible. The second and third problems can be tackled if a detection method is employed, to identify and map suitable locations to place individual sensor nodes on top of the canopy. Furthermore, the sensor network architecture can also be considered in the selection of suitable deployment positions.

The aim of this research is to propose a framework for an aerial sensor network placement mission. While previous studies often suggest random or uniform placement of sensor nodes within the target environment [7, 8], this research specifically focuses on placing acoustic sensor nodes atop the rainforest canopy. The emphasis is on autonomous deployment in predefined locations, minimizing travel distance to enhance efficiency.Thus, the main goal is to propose an initial proof-of-concept for a path planner for safe and autonomous exploration and detection of suitable deployment locations. In doing so, the following research question shall be answered:

How to design an efficient path planner for an aerial sensor network, to explore and detect precise deployment locations in an obstacle dense environment such as the rainforest using a quadrotor?

The structure of this report is divided in two parts. First, Part I presents the scientific article, where all the work developed to answer the research question is detailed, along with the results obtained in several flight experiments. Then, Part II contains the literature study conducted prior to the thesis work, where different path planning algorithms were introduced and analyzed. This section can be reviewed to gain further insight into the choices made throughout the thesis.

[This page is intentionally left blank]

Part I

Scientific Article

[This page is intentionally left blank]

Autonomous Aerial Sensor Network Placement in Rainforests: Exploration and Detection of Deployment Locations

Rita Santos Raminhos
Department of Control & Operations,
Section of Control & Simulation,
BioMorphic Intelligence Lab
Delft University of Technology, The Netherlands

***Abstract*—The effects of climate change put increasing strain on rainforests and their inhabitants, highlighting the demand for technological developments to aid in biodiversity monitoring and conservation efforts. This research work proposes a novel framework for the autonomous placement of acoustic sensor networks on top of the rainforest canopy, using a quadrotor platform. In order to tackle the challenge posed by multirotors' limited autonomy, an initial canopy exploration mission is considered and an exploration planner is developed to detect suitable locations on top of the rainforest canopy for sensor node deployment. Specifically, green detection and pointcloud projection is performed online and combined with our proposed sampling method to accomplish targeted exploration, towards the estimated location of detected green components. Flight experiments are conducted across multiple scenarios that mimic distinct rainforest features. The results validate the system architecture and demonstrate the effectiveness of our smart sampling method, laying the foundation for future autonomous sensor network exploration missions at larger-scale.**

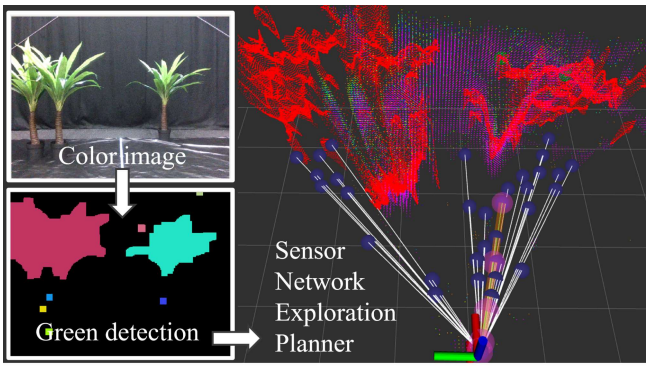


Fig. 1: Our proposed ROS2 exploration planner sampling paths towards detected green components, captured in RViz.

I. Introduction

Biodiversity loss as a consequence of climate change has been measured by scientists in recent decades [1]. There is ample evidence of the impact on different plant and animal species in distinct environments around the world, raising the need for more and more extensive biodiversity conservation campaigns. Rainforest environments, while representing only six percent of Earth's land surface, are inhabited by over fifty percent of known animal and plant species [2]. Therefore, studies to perform biodiversity monitoring in the rainforest are essential but pose specific difficulties due to the characteristics of these environments, namely their remoteness, dense vegetation and high humidity levels. New technologies are being developed by researchers to aid in biodiversity survey and monitoring in such challenging environments [3, 4, 5, 6]. Multirotors are used in these works as they are lightweight and agile platforms and thus easier to operate in complex environments, compared to other Unmanned Aerial Vehicles (UAVs).

Two trends are observed currently in biodiversity monitoring applications: missions where data collection is conducted online, using UAVs with longer endurance capabilites [7], and missions where data collection is achieved over longer periods of time, via the deployment of wireless sensor networks (WSNs) in the environment, as they enable the collection of larger amounts of environment data [8]. WSNs combine multiple sensor nodes with communication, localization and processing capabilities that depend on the application [9]. Sensor nodes that capture acoustic information are suited for performing biodiversity surveys in the rainforest environment, due to the lower dimensions of the data and increasing success of post-processing solutions [10]. Most works currently assume that the placement of the sensor nodes in a sensor network is random [11, 12] or uniform [13]. When placed on the top of the tree canopy, acoustic sensors capture the sounds of birds, bats and gliders that inhabit the emergent layer, where navigation for multirotors also becomes less complex than in the lower levels of the rainforest. We propose a framework for the deployment of an acoustic sensor network on the rainforest canopy using a lightweight quadrotor. The focus of our framework is on autonomous navigation and deployment, while giving consideration to the limitations of quadrotors, namely their limited autonomy. Therefore, we divide the aerial placement mission into three phases: exploration mission, offline planning and deployment mission.

The methodology developed in this paper focuses on the

first stage of deployment, the initial exploration of the environment to identify suitable locations in the canopy for sensor deployment. An exploration path planner is proposed to plan local paths around obstacles towards possible regions of interest. Our planner uses *nvblox* for generation of an occupancy map [14]. The generation of sampling-based paths is based on the Motion Primitives-based Planner proposed by M. Dharmadhikari et al. [15]. The main contributions of the proposed approach are listed below:

- A ROS2-based local exploration planner, combined with a complete volumetric mapping and detection pipeline;
- Integration of state-of-the-art volumetric mapping library *nvblox*, optimized for faster computations, within Nvidia's Isaac ROS environment, to perform two-dimensional (2-D) collision checking for the proposed planner;
- Resource-efficient online processing method of color and depth images sensed by an RGB-depth camera to identify and estimate the position of green components;
- Smart sampling method to obtain local paths that guide exploration towards detected green components.

In section II related work is presented and section III introduces the methodology developed. In section IV the setup for the flight experiments is explained and the results obtained are discussed in section V. Then, section VI follows to include various recommendations for future work. Finally, section VII draws conclusions on the research work conducted.

II. Related Work

Path planning for autonomous exploration of an unknown environment is a widely researched problem within the robotics field. Different approaches have been proposed by researchers using multi-rotors. Frontier-based and sampling-based methods are two of the most well-known methods. Frontier-based methods such as [16] direct exploration towards unknown space, as frontiers are defined as the limits between mapped and unmapped space. On the other hand, sampling based methods for exploration focus on increasing the amount of new information acquired at each planning step, considering both positioning and sensor characteristics. In their work [17] A. Bircher et al. sample "Next-Best-Views" within free configuration space as nodes of a Rapidly-exploring Random Trees (RRT). The sampled collision-free paths are evaluated at each re-planning step based on their volumetric gain, while longer paths are penalized. The authors reported better exploration rates using NBV planner compared to a frontier based implementation for larger and more complex environments.

However, M. Selin et al. [18] proved that NBV planner can get stuck exploring large scale environments, when a region that has not been explored is far from the current location and therefore outside the sampling range. As a solution, the authors propose the AE planner, combining NBV as a local planner with a frontier based global planner that generates paths when local exploration gets stuck and reports zero information gain. Node caching is implemented to estimate the gain of new sampled nodes and save high gain nodes to use in the global planner, as high gain nodes are linked with frontier regions. The results show that their planner successfully explores environments where NBV planner fails and also validate the node caching strategy, since the computational times obtained are lower compared to NBV planner.

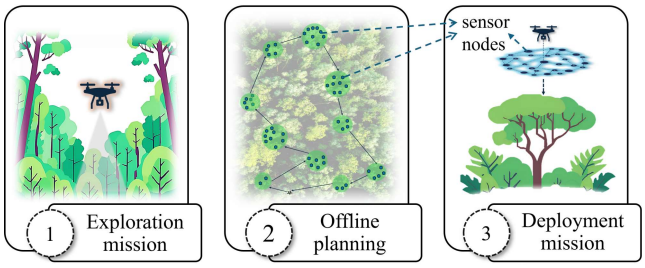


Fig. 2: Framework for the autonomous deployment of aerial sensor network using a quadrotor. We propose an initial exploration mission (left) that maps and detects suitable deployment positions, followed by an offline planning step (middle), when the characteristics of the sensor network are considered and the final placement path is computed, and finally the actual deployment mission (right), in which the quadorotor navigates to the locations and placement of the sensor nodes occurs.

Other state-of-the-art planners such as FUEL [19] and Graph based planner (GBP) [20] have proposed architectures that include both a local and a global planner. Their results highlight the advantages of planning at a global scale for exploration of large-scale environments. In the case of GBP, it was tested in two underground mine environments and their global planner also includes a return-to-home feature.

The current state-of-the-art is also moving towards including the UAV's kinodynamics - kinematic and dynamic constraints - in the sampling of new paths to enable more agile and faster exploration. In their Motion Primitives based (MPB) planner [15], Mihir Dharmadhikari et al. introduce an approach of sampling accelerations and obtaining the sampled paths through their primitives. Besides position and heading, the authors also include velocity states in their configuration definition. Therefore, sampled paths are checked for collisions but also if they are future-safe and a hover can be reached safely from the samples configurations, since a final velocity of zero is not assumed. Similarly to GBP, MPB planner was tested in two underground mines and achieved an average flight speed of $1.8m/s$ in one of the tests. REAL is a recent work by E. Lee et al. [21] that samples minimum-snap trajectories [22] to achieve faster exploration. In their paper REAL was compared and outperformed other state-of-the-art planners NBV planner, AE planner, GBP and MPB planner in a large scale environment. With regards to MPB planner, the authors attribute this result both to the sampling based approach that resulted in less smooth movements and the lack of a global planner.

III. Aerial Sensor Network Placement Mission

Fully autonomous placement of an acoustic sensor network using a quadrotor in a rainforest is conditioned by both the

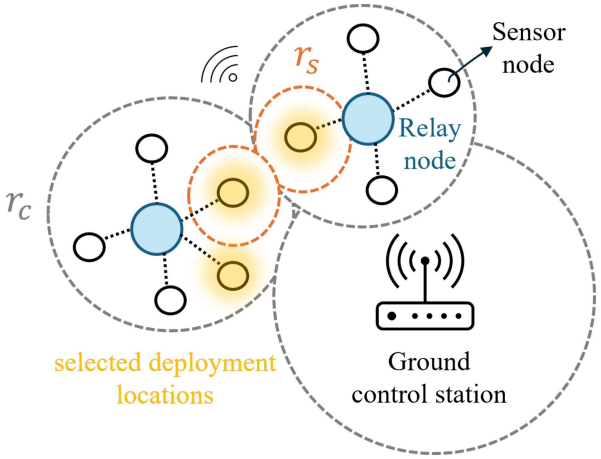


Fig. 3: Illustration of a sensor network with both sensor nodes and relay nodes. Sensor nodes connect to relay nodes, which link to other relays and the ground control station. In the proposed offline planning step, optimized deployment positions for the sensor nodes can be determined, based on the network characteristics, e.g. the number of sensor nodes, sensing radius (r_s) and communication radius (r_c) with relays and ground control station. The multirotor then deploys the nodes to the selected locations, in the final deployment mission.

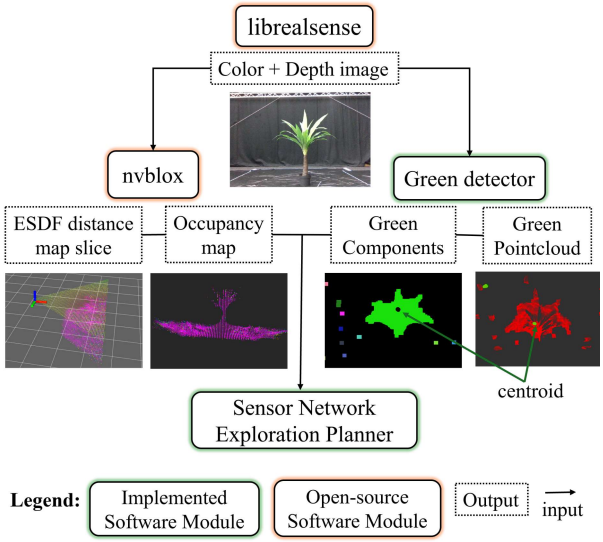


Fig. 4: Overview of the software modules related to the mapping (*nvblox*) and green detection (*Green detector*) pipeline from camera sensor data. Open source libraries are highlighted in orange amd our own code modules in green. The outputs from *nvblox* and *Green detector* are inputs to our exploration planner, namely ESDF distance map slice, green components and green pointcloud.

characteristics of the environment and quadrotors' inherent limitations, namely their limited flight time. In order to address the problem, we propose a framework to autonomously deploy as many sensor nodes as possible in an unknown environment, while covering the least distance.

A. Mission framework

We divide the autonomous placement of a sensor network in three stages: exploration mission, offline planning and deployment mission, illustrated in Figure 2.

We assume that the environment is completely unknown prior to the exploration mission. Therefore, the quadrotor must autonomously map and explore the surrounding environment, while simultaneously avoiding obstacles and detecting locations of interest for sensor placement. Sensor nodes should be placed on top of the canopy in flat and densely vegetated areas, so the exploration goal is to map such locations. After exploration, offline planning takes place, where the identified regions of interest are considered to plan the shortest deployment path for the deployment mission. The architecture and characteristics of the sensor network should be considered in this phase, for instance, the number of sensor nodes, communication radius and sensing radius, depicted in Figure 3. Finally, the deployment mission involves navigation to the selected deployment locations and local re-planning is necessary to avoid obstacles.

In this paper we propose a local exploration planner to complete the first part of the placement mission, the exploration mission. The planner explores the environment and plans safe collision-free paths to identify suitable positions for sensor placement. The detected positions for sensor placement positions are saved by our planner. They can then be used for offline planning in the following stage.

B. System Overview

Exploration path planners are highly dependent on sensing and mapping capabilities of the mobile platform. The most important sensor characteristics are the range and field of view (FoV). These are inputs to our planner, such that paths are not computed outside this region. Volumetric mapping libraries are able to perform volumetric reconstruction of an environment from sensor data. In our implementation, *nvblox* [14] is used to generate dense occupancy maps to be used by the exploration planner. *nvblox* is a state of the art library for GPU-accelerated volumetric mapping proposed by A. Millane et al. Their library provides both Truncated Signed Distance Fields (TSDFs) and Euclidean Signed Distance Fields (ESDFs). While TSDFs are used for surface reconstruction and have lower computational times, ESDFs are useful for path planning applications as they enable collision checking and selection of collision-free paths. Compared to other recent works including *Voxblox* [23], *Voxfield* [24] and *FIESTA* [25], *nvblox* provides high resolution maps and is significantly faster at both querying and distance field computations. Furthermore, a library tailored to GPU processing allows us to take advantage of the full capabilities of our hardware implementation, detailed in section IV.

We propose the architecture in Figure 4 for the mapping and detection pipeline of our system. Our local exploration planner uses the ESDF distance map slice provided by *Nvblox*

to perform collision checking. At the same time, we developed our own green detection module that obtains the green components and pointcloud from the color and depth images. The detected green components and pointcloud are also provided as inputs to our planner, in order to guide exploration towards the unmapped green components. The following section elaborates on the details of our green detector module.

C. Green detection

For the initial detection of green areas, we develop a green detector that operates on the color images from the camera sensor. As the purpose is to identify full green areas, non-green pixels are filtered out and a set of morphological operations is applied using the Open CV library. Then, the different connected components in the image are saved and their statistics are computed, namely area and centroid location. The corresponding green pointcloud and 3-D position of the centroid is computed on the GPU by leveraging functions from *nvblox*'s library.

In Figure 5 we show the green detection of the same artificial palm tree in two environments and compare the results with and without the implemented morphology operations, in two distinct settings. Clearly, the applied morphology operations are essential to obtain satisfactory results and distinguish one green component, in this case.

Our method uses color filtering in the HSV color space. This method is influenced by lighting conditions, which can lead to both missed components or false positives, which will in turn influence the planner results in the exploration tasks. It is important to mention that more complex and computational expensive methods, such as neural networks or 3-D tree detection could achieve more accurate results.

D. Exploration planner

Most exploration planners proposed in literature aim to explore the highest amount of volume possible in the least amount of time. In order to do so, paths are generated within collision-free space and then evaluated through the computation of a gain function. In MPB and GBP planners [15, 20], the authors define a gain function that includes the volumetric gain sensed by the path and additional terms to penalize longer paths and sharp changes of direction.

In our case, we also wish to direct exploration of the environment towards possible deployment locations for sensor nodes in the rainforest. Our quadrotor should fly above the canopy layer and identify suitable positions in the canopy below it. Since the canopy height is approximately constant, during the proposed exploration task we assume a fixed height for the entire mission duration, set by the operator before the start of the mission. Therefore, our planner samples horizontal paths in 2-D space. Navigation is safer above the canopy layer and few obstacles exist. However, there are also trees in the emergent layer so collision checking is necessary and performed by checking the distance map slice (at the planning height) provided by *nvblox*, as mentioned previously. We assume that GPS-guided navigation above the canopy is possible and reliable above the forest canopy, as experienced by participation in the XPRIZE rainforest competition [1].

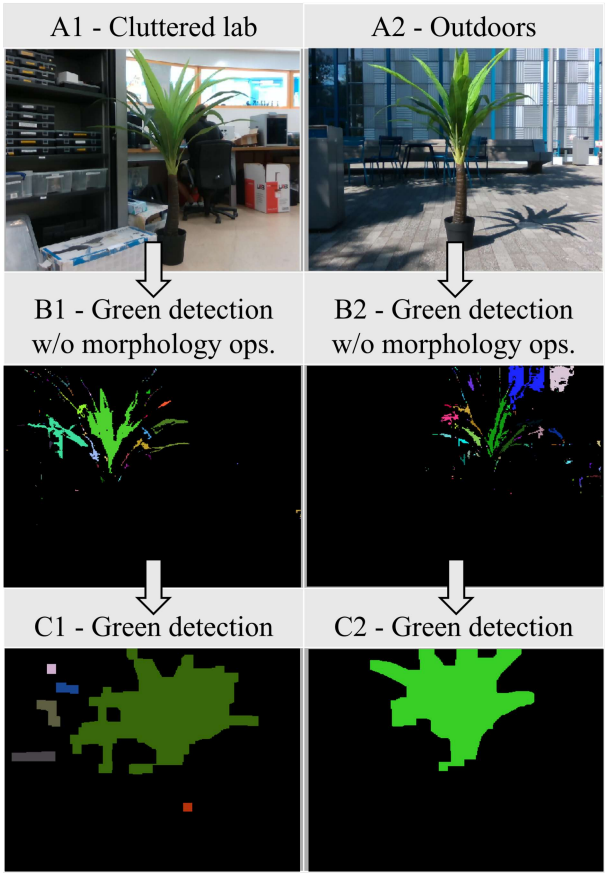
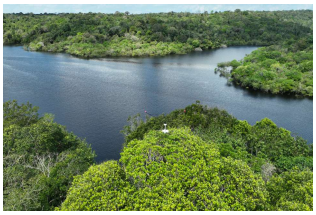


Fig. 5: Green detection results for one palm tree in two distinct environments with different lighting conditions: a cluttered lab environment (A1, B1 and C1) and outside, on a sunny day (A2, B2 and C2). The top images, A1 and A2, are the color images from the RGB-depth camera, and the middle images, B1 and B2 are obtained after masking non-green pixels in the HSV color space, using OpenCV functions. At the bottom, images C1 and C2 show the final green detection results, after applying a series of morphology operations (dilation and closing, followed by erosion and opening). These operations improve the results and clearly distinguish a compact green component, that corresponds to the palm tree.

Our proposed planner uses the green components from the green detector module to sample paths towards identified dense green areas, that would correspond to a dense region in the canopy, such as the one shown in Figure 6a. We incorporate the generation of a tree of motion-primitives based paths implemented by the authors of MPB planner [15], with our own method of sampling accelerations in the horizontal plane in the direction of the detected green components. As with any exploration path planner, it is undesirable that the planner gets stuck and is not able to plan the next path. For example,

[1] https://www.xprize.org/prizes/rainforest

(a) Canopy trees



(b) River in the rainforest

Fig. 6: Aerial views from the Amazon rainforest of Brazil, taken by Georg Strunck during the XPrize Rainforest finals. Figure 6a shows the type of dense canopy foliage that we wish to explore and detect with our planner. Figure 6b is an example of the necessity of the re-exploration step: if the quadrotor reaches the limit of the canopy, where there is the river, no paths can be sampled because no green components are detected, so the planner will fail to plan a path.

in our case, this is possible if the planner gets stuck in a corner with emergent trees surrounding it and it cannot find a collision-free path to exit or if there is clearing in the forest or a river, like in Figure 6b. We propose a re-exploration step to address this problem, which is executed when the number of consecutive runs where the planner obtains a very low gain exceeds a threshold. The pseudo-code of our planner is provided in Algorithms 1 and 2.

Algorithm 1 Sensor Network Exploration Planner

while planner node is running **do**
  ▷ Save new green components
  $greenComponent \leftarrow greenComponentCallback()$
  if $area(greenComponent) \geq area_{min}$ **then**
    if $componentNotYetExplored()$ **then**
      $currentComponents \leftarrow greenComponent$
    end if
  end if
  ▷ Build tree and get best path
  $tree \leftarrow buildTree()$
  $leafs \leftarrow findLeafNodes(tree)$
  for all $leafs$ **do**
    $path \leftarrow getPathfromLeaf(leaf)$
    $gain \leftarrow computeGain(path)$
    if $gain \geq bestGain$ **then**
      $bestGain \leftarrow gain; bestPath \leftarrow path$
    end if
  end for
  if $gain \geq gain_{min}$ **then**
    $publishPath(bestPath)$
  end if
  ▷ Check consecutive runs with low gain
  if $runsGainTooLow \geq runsGainTooLow_{max}$ **then**
    $reExploration()$
  end if
end while

Algorithm 2 Tree Generation

function $buildTree$
  $treeRoot \leftarrow currentPosition$
  while $treeSize < treeSize_{max}$ **do**
    for all $currentComponents$ **do**
      while $sample < samples_{max}$ **do**
        $a, theta \leftarrow sampleAcceleration()$
        ▷ Check if acceleration is inside FoV
        if $fov_{h_{min}} < a < fov_{h_{max}}$ **then**
          $pos \leftarrow computeMotionPrimitives()$
          ▷ Check if position is in known free space
          $status \leftarrow checkStatusInMap(pos)$
          if $status$ = Free **then**
            $insertNodeInTree()$
          end if
        end if
        $sample \leftarrow sample + 1$
      end while
    end for
  end while
end function

E. Sampling method

The proposed planner follows a sampling-based approach, where a tree of possible paths is generated. The tree generation mechanism is adapted from MPB planner [15]. The control space is sampled by sampling accelerations. Each path is obtained by computing the corresponding motion-primitives for one sampled acceleration, considering fixed time steps in a determined time interval. Each tree node is then a position in the configuration space, that has been checked to be in known free space. Algorithm 2 outlines the tree building logic.

We propose a method for sampling accelerations that directs exploration towards detected and previously unexplored green components. To reduce the probability of plannings towards false positives, a minimum area threshold on the components used for planning is set. Similarly to MPB planner, an acceleration magnitude and angle are sampled. The magnitude value is constrained by the vehicle's dynamic limits and input boundaries. The angle value is sampled within a range of the direction between the component and the current planning configuration. The sampled angle is immediately checked to belong inside the current sensor FoV or discarded and re-sampled. An illustration of the proposed acceleration sampling can be seen in Figure 7.

F. Gain function

After sampling and building a tree of possible collision free paths towards the visible green components, each path must be evaluated and compared with other paths, to find the path with the best gain. The equation for our gain function is shown in Equation 1, where each path σ was sampled towards a specified component, C. The detected area of the component is evaluated by the gain function, along with decay terms to penalize longer path distances, $dist(\sigma)$ and lack of

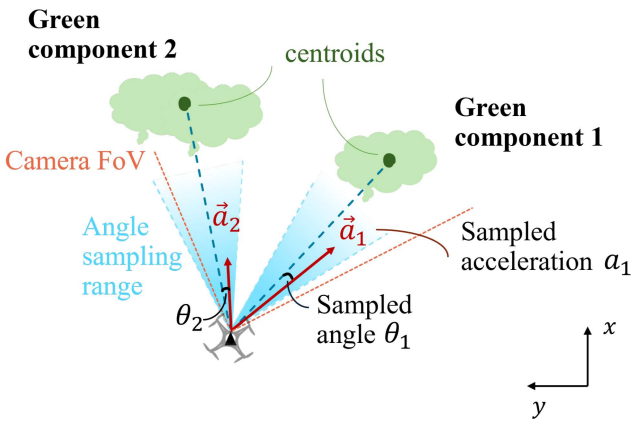


Fig. 7: Illustration of the method implemented to sample accelerations in the horizontal plane, from the current position in the direction of the identified green components. For each green component, an acceleration magnitude and angle are sampled at each sampling iteration. The magnitude a is sampled from a uniform distribution $U(a_{min}, a_{max})$, where the minimum and maximum acceleration are input constraints. The angle θ is sampled inside the camera FoV from a uniform distribution with interval $U(\alpha - \delta, \alpha + \delta)$, where α is the angle in the direction of the detected component from the current position and $[-\delta, \delta[$ is the sampling angle range.

smoothness, $S(\sigma)$. $\gamma_a, \gamma_d, \gamma_s$ are the gain function weights for the area, distance and smoothness terms, respectively, that can be tuned to increase or decrease the importance of each term. The equation for the implemented smoothness function is presented in Equation 2, which aims to penalize abrupt changes of direct by computing the sum of the yaw angle ψ difference between consecutive nodes in a path, where n is the total number of nodes and k is the current node being evaluated.

$$Gain(\sigma) = \gamma_a \cdot Area(C) \cdot e^{-\gamma_d \cdot dist(\sigma) - \gamma_s \cdot S(\sigma)} \quad (1)$$

$$S(\sigma) = \sum_{k=1}^{n} \psi(k+1) - \psi(k) \quad (2)$$

G. Re-exploration step

Our implemented planner samples the configuration space iteratively, obtaining at each run a tree of possible collision-free paths that are evaluated by means of the gain function, as discussed in the section above. However, it is possible that, in several consecutive runs, the planner fails to compute a path with a gain that is higher than the minimum desired gain. The minimum desired gain is a parameter set by the operator and the lowest value it can take is zero. This can occur in a cluttered environment where the planner is unable to sample positions within know free space (since it is very reduced). In the case of our planner, this can also take place due to our method of sampling accelerations in the direction of the green components, so if there are no green components within the sensor FoV at the re-planning moment, no path can be found.

Thus, we implemented a re-exploration step when the number of consecutive runs where the gain was too low is higher than the maximum allowed threshold (planner parameter). A re-exploration step consists of yawing in an arbitrary direction, to search for new green components to explore. The value of the yaw angle is equally a planner parameter. After multiple consecutive re-exploration steps where the planner still could not compute a path, exploration is terminated and the landing sequence is initiated.

IV. Experiments

In order to validate the developed code, several flight experiments were carried out in a controlled indoor setting.

A. Set-up

The aerial platform employed in the flight tests and respective hardware implementation are shown in Figure 8. The quadrotor was built using a custom frame with a diameter of approximately $0.3m$. The flight controller is a *pixracer*, running the PX4 flight stack. The depth camera is an Intel RealSense D435, with a maximum range of $d_{max} = 3m$ and $FoV = 87° \times 58°$. All developed code runs onboard an Nvidia Jetson Xavier NX board. The code is ROS 2 (Humble) based and runs within Nvidia's Isaac ROS environment [2], version 2.1. Therefore, the Isaac *nvblox* package is used to make use of GPU functionality of the Jetson board, enabling faster computations.

The available testing area is approximately $10m$ x $10m$. It is equipped with an OptiTrack Motion Capture system, that provides an accurate position estimate to the path planner and flight controller during the flight tests.

Three distinct mission scenarios were considered and the set-up for of each of the scenarios A, B and C is shown in Figure 9, along with detailed explanation of the differences between each mission. For each scenario, different features of a forest environment are imitated artificially in our indoor test area. In scenario A, the drone flies at $h = 1m$ height and the artificial plants placed in the arena have a height between $1m$ and $1.6m$, such that they can be sensed by the drone's depth camera, but will simultaneously work as obstacles. On the other hand, in mission scenarios B and C, the flight height is $h = 1.5m$, the drone's camera is angled downwards at a $30°$ angle and the main goal is to explore from above the plants placed at the ground level, to mimic the exploration of canopy trees as seen in Figure 6a. In scenario C, additional obstacles and trees are arranged, to replicate the presence of emergent trees in the rainforest. In Figure 10, we present the horizontal position map of a single test run, for each scenario. The 2-D position setpoints yielded by our exploration planner during the mission are highlighted. The green components marked as

[2]Information about the Nvidia Isaac ROS environment and Isaac ROS Nvblox package can be found in https://nvidia-isaac-ros.github.io/index.html

[3]Photo taken from the pixracer page in the official PX4 Guide: https://docs.px4.io/main/en/flight_controller/pixracer.html

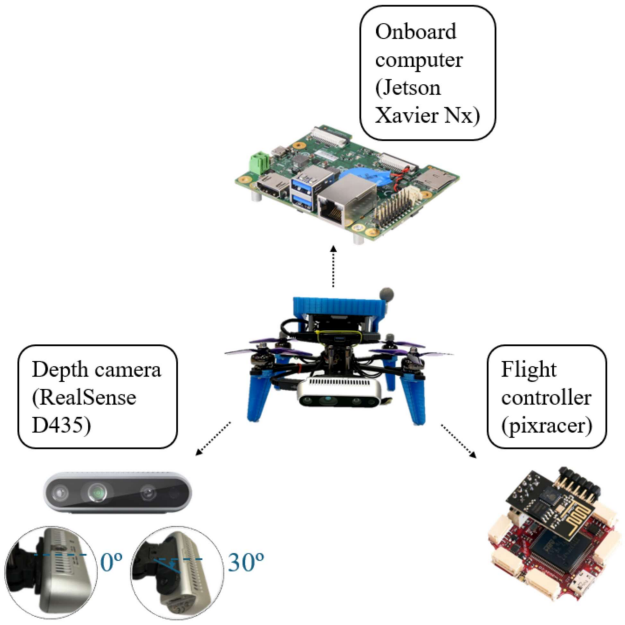


Fig. 8: Quadrotor platform used in the experimental evaluation and its components: from left to right, depth camera, onboard computer and flight controller. The depth camera is the Intel RealSense D435, the onboard computer is the Nvidia Jetson Xavier NX and the flight controller is a pixracer [3].

explored can also be observed and compared with the ground truth estimate for the actual position of each green area.

B. Scenario A: exploration "mid-canopy"

The results for scenario A mission in Figure 10a show that our planner code marked three out of the four components as explored, despite clear navigation at the end of the mission towards the fourth component on the bottom right hand side. After careful observation of the camera and green detection data from this flight, we concluded that the two bamboo plants used had thin, sparse leaves that did not make up a full green area, to be identified as an individual green component. Instead, multiple components were detected, which led to exploration in that direction, but none of the components met the planner's minimum area threshold to be identified as an explored component. Therefore, this result is actually positive, since our goal is to identify dense green areas. Additionally, there is a significant error in the position of the first explored component (the two palm trees on the left), compared with the ground truth value. We can speculate that this is due to the palm leaves having very protruding leaves, which pose a greater challenge for our method of estimating the position based on the detected centroid position.

C. Scenario B: exploration above the canopy

With regards to the mission of scenario B, Figure 10b, it is also clear that exploration occurred towards all four components, with three components identified as explored by our algorithm, of which one is an undesirable false positive. This is also likely due to the very irregular and protruding foliage of the palm leaves of component number one, seen at the top in Figure 9b. The components that were marked as explored in the mission are the largest components. The remaining two components were not identified as explored because, despite being very dense, they did not meet the minimum area threshold. These results showcase the exploration of canopy-like green areas, which is closer to the proposed exploration mission for sensor placement conducted in an actual rainforest.

As it is representative of an exploration mission in the rainforest, we present the position and yaw tracking obtained for the same test run of scenario B in Figure 12. The position controller's tracking of the desired setpoints is sufficient for our application. Several re-exploration steps took place during the mission, namely in the instances following the exploration of a component. In those moments, no unexplored green components could be found inside the camera's FoV and thus the local planner fails to plan a path. For all cases in the analyzed mission, the re-exploration approach was effective in finding new green components to explore, when the planner gain was too low.

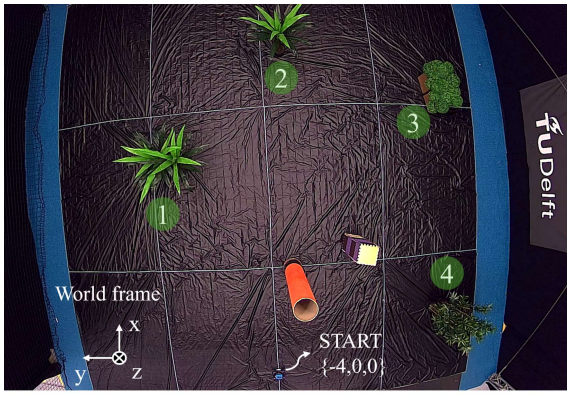
D. Scenario C: exploration above the canopy with emergent trees

Finally, the results for scenario C, in Figure 10c, show that the planner generated paths towards three out of the four total components, and all explored components were successfully identified as explored. Therefore, one component was not explored, component number three on the bottom right corner of Figure 9c, and this was common to all scenario C runs. However, this result is considered satisfactory, because the missing component is blocked by the obstacle, from the drone's position, making it harder for the planner to sample safe paths to the third component, and instead the vehicle safely proceeds to explore the remaining component.

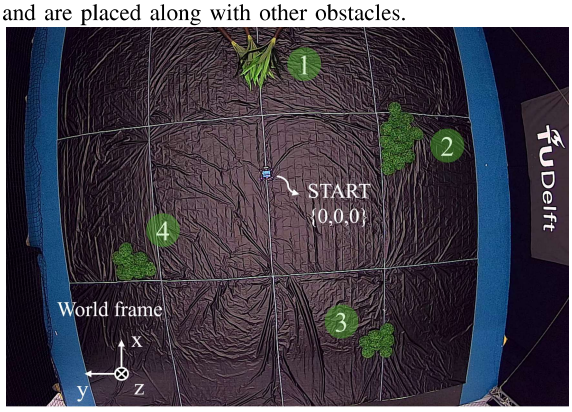
E. Flatness detection experiments

The collaboration between ETH Zurich and TU Delft in the ETH BiodivX team, that participated in the XPrize rainforest finals, led to the successful deployment of an ecology canopy raft on top of the Amazon rainforest canopy. The canopy raft carried sensor nodes and the deployment was performed in manual flight, using a commercial quadorotor. Such a platform is a possibility for future autonomous deployment of the sensor nodes, which requires not only dense green areas on the top of the canopy, but also areas that are flat enough, such that there is less risk of the raft sliding off or tipping over.

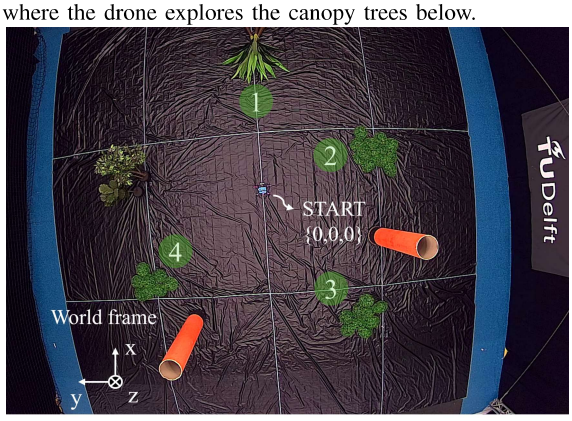
Considering this mechanical design, it is interesting to investigate how flatness detection can be included in our pipeline. A planar segmentation method using the open-source Point Cloud Library *pcl* was added to our code base. This



(a) Scenario A. The mission starts at the edge of the CyberZoo and the flight height is set to $h = 1m$. The camera is not angled. All plants have a height between 1 and $1.6m$ and are placed along with other obstacles.

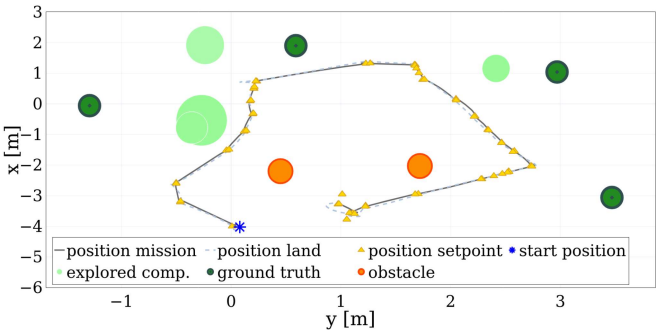


(b) Scenario B. The mission starts at the center of the CyberZoo and the flight height is set to $h = 1.5m$. The camera is angled downwards at an angle of $30°$. All plants are placed at the ground level, to mimic a rainforest mission where the drone explores the canopy trees below.

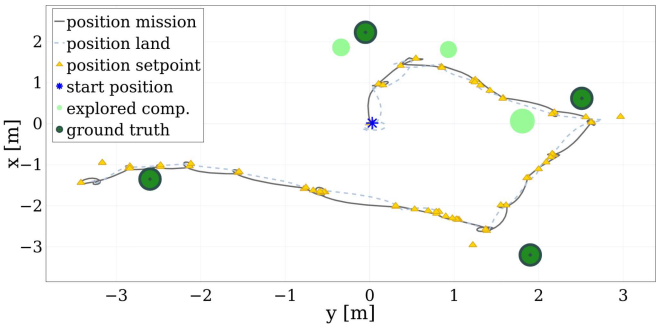


(c) Scenario C. Similar to Scenario B, with the same flight height and camera angle. Additional obstacles and two trees (on the top right) are placed, to mimic emergent trees in the rainforest that the drone has to avoid.

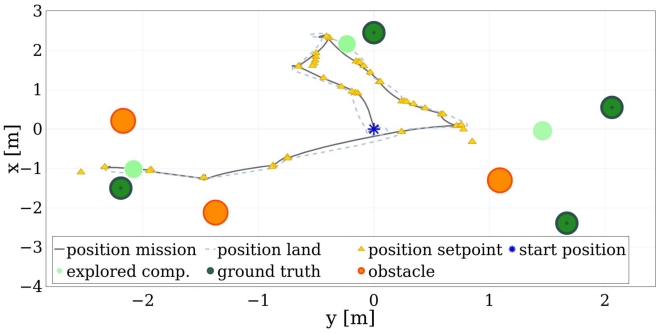
Fig. 9: Experimental setup for the 3 test scenarios. Artificial forest is assembled in indoor drone testing environment. The world frame is oriented following the ROS convention FLU (x is forward , y points left and z is up).



(a) Mission result - Scenario A

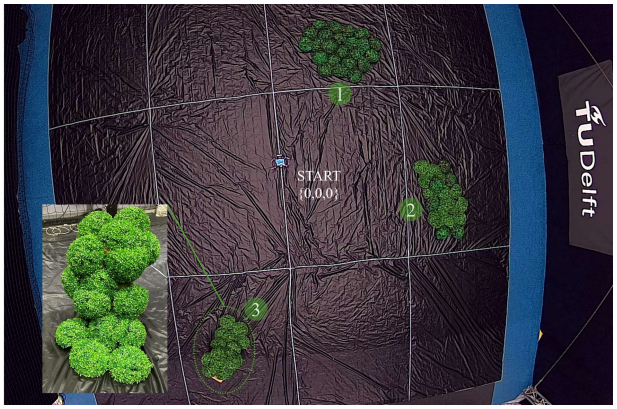


(b) Mission result - Scenario B

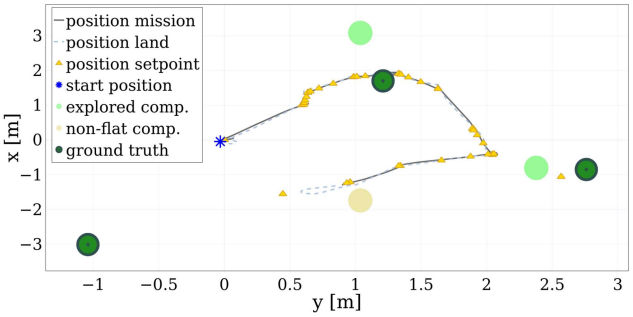


(c) Mission result - Scenario C

Fig. 10: 2-D map of one test run for each scenario, Scenario A (Figure 10a), Scenario B (Figure 10b) and Scenario C (Figure 10c). We show the position setpoints generated by our planner and the actual tracked position in the horizontal plane (-y,x). The green components marked as explored by our algorithm can be compared with the actual estimated ground truth of the position of each green component. It is clear that, in each scenario, the exploration if the environment is successfully guided towards green.



(a) Set-up used to test the flatness detection logic, with three canopy-like components. Components number one and two are placed on the ground and should be detected as "flat components", while component number 3 is not flat.



(b) 2-D position map for one flight. The non-flat component (number three) is successfully discarded by our implementation and not marked as an explored component for sensor node deployment.

Fig. 11: Flatness detection experiment results. In this experiment, explored components are subject to an additional boundary constraint compared to previous Scenarios A, B and C, related to their flatness.

method was applied to our detected green pointcloud, when the distance of the vehicle to the detected green component was below a certain threshold, such that the detected green pointcloud corresponds to a single component. The implemented method then returns and saves the percentage of pointcloud points that fit the planar model, within a desired thickness, $0.3m$ was used in this case. The goal of this implementation is that, by analyzing the values obtained for different components, we can conclude which components represent flatter surfaces, for easier and safer deployment and retrieval operations. Figure 11a displays the scenario used in the three flight tests performed, to test the flatness detection logic, and Figure 11b shows the results for one flight, where a minimum threshold of 95% is set for the percentage of "flat points", and thus only the flattest components were saved as explored. Additionally, Table I contains the results obtained for the pointcloud segmentation method, for three flight experiments, which show that the method is successful in distinguishing flatter components.

TABLE I: Average, maximum and minimum percentage of points that fit a planar model with a height of $0.3m$, for successive measurements applied to the detected pointclouds of each green components, across three flights. A difference is clearly noticeable between components 1 and 2, for which almost all points are inside the segmentated plane and component 3, with a lower percentage of detected points that are inliers.

	% detected "flat" points		
	Average	Max.	Min.
Component 1	99.9	100	96.9
Component 2	99.8	99.9	99.4
Component 3	93.4	96.3	82.9

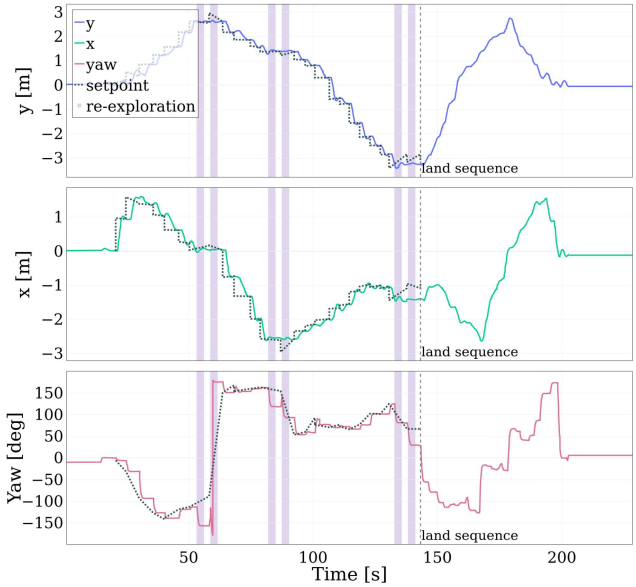


Fig. 12: Tracking of position and yaw setpoints during Scenario B mission. The position controller is able to follow the position setpoints sent by our planner module. The tracking is not perfect, but sufficient for our implementation. Additionally, six re-exploration steps were successfully executed when the planner gain was too low and no paths were sampled.

V. Discussion

A collective analysis of the results presented in the previous section reveals two key insights. First, the desired behavior of the planner to direct exploration towards dense green areas is clearly visible in all three missions, since the vehicle traveled towards the great majority of green areas. This was the main goal of our method, combining green detection with our sampling method to obtain safe paths towards the detected green areas. Second, our collision checking implementation, that checks *nvblox*' ESDF distance slice map in the sampling of collision-free paths, is equally validated, as demonstrated by the results in both scenarios with obstacles, A and C. In all flights of scenario A, no collisions were detected despite

the cluttered environment, since all plants also represented obstacles for the drone in this scenario. This is also evident in the result for scenario C, where emergent obstacles and trees were included and successfully avoided.

A total of twenty-four test runs were completed: twelve flights were performed for mission scenario A and six flights each for both scenarios B and C. In Table II we present several metrics to evaluate the planner's performance across all 24 runs. The main goal of the proposed exploration mission was to explore as many sensor node deployment locations as possible, while covering the least amount of distance, due to the limited autonomy of quadrotors. Therefore, the average distance traveled during the exploration mission is a very important metric. The average distance was highest for scenario B runs, which also have the largest variation, while both values are lowest for scenario C mission. This is easily explained because, as noted previously, in all scenario C runs, paths were generated only towards three out of the four components, due to the presence of the obstacle next to the third component. Scenario B, on the other hand, has no obstacles and more dense green components than scenario A, so more paths were planned towards all components, therefore more distance covered. Across all runs, the average distance covered was $13.41m$. In order to evaluate this performance, we can draw a simple comparison, illustrated in Figure 13, where the environment is mapped instead with a non-intelligent approach, via exhaustive search of the entire space. In this case, to explore the same volume, we estimate that the traveled distance would be approximately $49.9m$, which is more than 70% higher than our method.

Regarding the number of explored components, the average obtained is 1.96 components explored per flight, slightly below half of the total number of components. This result improves if we consider the scenario without obstacles, scenario B, with an average of 2.33 components explored per flight and also less variation between flights. Tests for scenarios A and C, which had more obstacles, had very similar results, averaging 1.83 explored components per flight. As planning towards green components occurs even if the components are not considered explored, the results on the detection rate of individual components can provide further insight.

Similarly to what was observed in the mission in Figure 10b, for most flights of scenario B components number three and four were marked explored much less often (14.3%) or not at all, respectively, due to not meeting the minimum area threshold. Nevertheless, exploration is directed towards these components successfully. It is also interesting to note that in the flights with set-up A, where different types of plants were used, component number three is detected as explored considerably more times than all other components (54.4% of the total explored components corresponded to component number three). This result is positive, as it more closely resembles the canopy foliage, reason why it was used exclusively in scenarios B and C.

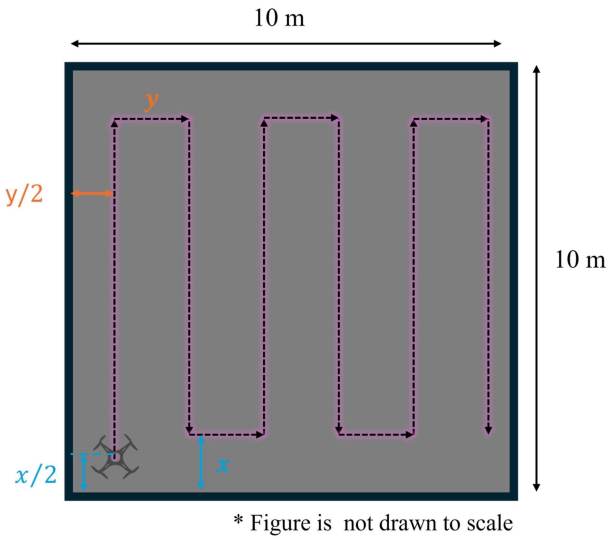


Fig. 13: Exploration of the same volume as the flight experiments, with the same depth camera, via a non-intelligent method would yield a total estimated distance traveled of $49.9m$. Mapping and detection operations would be performed offline, in post-processing. Detailed calculations to obtain distance values x and y are provided in Appendix B.

Two metrics are especially relevant to evaluate the performance of our green detection module, namely the percentage of false positives obtained and the error in the horizontal position estimate. In our case, we consider false positives the components that were saved by our algorithm that were actually false positive detections. Most cases correspond to repeated detections, i.e, the algorithm detected and saved the same component twice, in different locations. The percentage of false positives obtained is undesirably high, with an average for all test flights of approximately 27%. This value is slightly higher for scenario A (29%), compared to 26 and 21% for scenarios B and C, respectively, which is once more attributed to the protruding leaves of component number one.

Finally, our green detection module estimates the position of detected green components, by obtaining the 3-D position of the centroid detected in the color image. This method is susceptible to sources of error, such as it can be influenced by changes in lighting and perspective. Therefore, the error obtained between the estimate horizontal position of the centroid of explored components and the estimated ground truth value (obtained with the motion capture camera system), is also quite high, around $0.59m$ for all flight tests. Moreover, the RMSE for the position of individual components confirms our hypothesis that the leaves of the two palm trees that make up component number one in set-up A were more prone to detection errors, since the RMSE value obtained is 0.914, 30% higher than the average.

TABLE II: Performance metrics obtained in the test runs conducted, for each scenario and for all flights. Specifically, we considered the distance traveled during the exploration mission time (excluding the landing sequence), the number of explored components per flight (excluding false positives), the percentage of components identified as explored that are false positives and the Root Mean Square Error (RMSE) of the horizontal position of the centroid of explored green components compared with the estimated ground truth. Additionally, for each scenario, two metrics provide insight on differences between individual components: detection rate for each component and RMSE of the horizontal centroid position. Overall, the traveled distance and explored components results are very positive, while the number of false positives and error in the estimated horizontal position of the detected centroids are fairly high.

	Scenario A	**Scenario B**	**Scenario C**	**Total**
Number of flights	12	6	6	24
Travelled distance (mean ± std. dev.) [m]	13.1 ± 2.5	15.0 ± 3.5	12.5 ± 2.4	13.4 ± 2.7
Explored components / flight (mean ± std. dev.)	1.83 ± 0.83	2.33 ± 0.52	1.83 ± 0.98	1.96 ± 0.79
Percentage of false positives	29.0%	26.0%	21.4%	26.6%
Estimated horizontal position of components' centroid (RMSE | std. dev.) [m]	0.61 | 0.32	0.62 | 0.25	0.50 | 0.17	0.59 | 0.27
Detection rate of individual components (comp. 1, comp. 2, comp. 3, comp. 4)	22.7%, 18.2%, 54.4%, 4.5%	42.9%, 42.9%, 14.3%, 0.0%	45.5%, 27.3%, 0.0%, 27.3%	-
RMSE position of individual components (comp. 1, comp. 2, comp. 3, comp. 4) [m]	0.91, 0.47, 0.51, 0.17	0.66, 0.63, 0.42, -	0.37, 0.65, - , 0.54	-

VI. Future work

To the best of our knowledge, this is the first study to present a solution for rainforest canopy exploration and detection, in the context of autonomous aerial placement of a sensor network. While the current method is suited to perform additional online tests in a controlled environment, on the deployment mechanism and control logic, for example, there are several recommendations for future work to perform more complex missions in rainforest environments.

First, the green detection module can be improved. A more robust method of estimating the position of centroids would benefit planning results. The current implementation is sensitive to lighting conditions and perspective, so consecutive measurements can be fused in a more complex implementation to select suitable viewpoints and obtain more accurate results. Convolutional neural networks or 3-D tree detection methods applied to the detected green pointcloud would likely yield better results, at higher computation costs.

Alternatively, the current implementation can be used to select the desired exploration direction and approach the detected components, since it is computationally less demanding, and combined with a new method to estimate the accurate position. It would be interesting to train a neural network with aerial images of the rainforest, to perform canopy detection and centroid estimation. Data collected from manual flights above the canopy would certainly yield optimized results, compared to available forest LIDAR databases.

In relation to the exploration planner, the code developed only plans 2-D paths. Upgrading to a 3-D planner would allow to adapt to variations in the canopy height and control the distance to the canopy. It is noteworthy that this would require adapting the volumetric mapping library *nvblox* code as well, since the ESDF distance map currently provided is also 2-D. Second, including a global planner in the system architecture would significantly optimize results, for larger-scale environments, by redirecting exploration to previously detected but unexplored areas if the local planner fails, using algorithms like A* for optimal pathfinding. This global planner would also assist in return-to-home landings. Finally, a method for trajectory optimization, stacked on our path planning implementation, would allow for smoother flight.

VII. Conclusion

This work introduced a framework for the autonomous and optimized placement of an acoustic sensor network on top of the rainforest canopy, using a multi-rotor platform with limited autonomy. Based on this framework, an exploration planner is proposed to perform the initial environment exploration mission, with the goal of mapping suitable deployment locations, while minimizing the traveled distance. The implemented system architecture integrates a novel volumetric mapping library, for faster GPU-accelerated computations, and a developed green detection method to estimate the position of detected dense green components, with a local path planner, in order to guide exploration towards the detected green areas. Detected green components are approached and evaluated by the algorithm, and their estimated position is saved if the minimum area for sensor placement is observed.

Flight experiment results, conducted in an indoor environment with rainforest mimicking features, demonstrate the planner's ability to compute safe collision-free paths towards detected green components. The proposed planner covered significantly less distance compared to an exhaustive offline mapping approach, which is a promising result for larger-scale tests. Suitable deployment locations were successfully saved by the developed method. However, the error in estimating the horizontal position of these locations is noticeably high and a non-negligible percentage of false positive detections is identified. These results showcase the sampling method to direct exploration to regions of interest for sensor deployment and exhibit potential for further implementations.

References

[1] Muzafar Shah Habibullah et al. "Impact of climate change on biodiversity loss: global evidence". In: *Environmental Science and Pollution Research* 29 (2022), pp. 1073–1086. ISSN: 1614-7499. DOI: https://doi.org/10.1007/s11356-021-15702-8. URL: https://link.springer.com/article/10.1007/s11356-021-15702-8.

[2] H. J. Johnson. *Rainforest - National Geographic*. 2023. URL: https://education.nationalgeographic.org/resource/rain-forest/ (visited on 06/12/2023).

[3] Jarrod Hodgson et al. "Drones count wildlife more accurately and precisely than humans". In: *Methods in Ecology and Evolution* 9 (Feb. 2018). DOI: 10.1111/2041-210X.12974.

[4] André Farinha et al. "Optimal Sensor Launching with UAVs for Monitoring of Hazardous Environments". In: *IEEE IROS 2022 Workshop on Robotics for Nuclear Environments Exploration and Decommissioning: Challenges and Emerging Techniques*. Nov. 2022.

[5] Salua Hamaza et al. "Sensor Delivery in Forests with Aerial Robots: A New Paradigm for Environmental Monitoring". In: *IEEE IROS 2020 Workshop on Perception, Planning and Mobility in Forestry Robotics*. Oct. 2020.

[6] Salua Hamaza et al. "Sensor Installation and Retrieval Operations Using an Unmanned Aerial Manipulator". In: *IEEE Robotics and Automation Letters* PP (May 2019), pp. 1–1. DOI: 10.1109/LRA.2019.2918448.

[7] S. Sharma et al. "A Morphing Quadrotor-Blimp with Balloon Failure Resilience for Mobile Ecological Sensing". In: *IEEE Robotics and Automation Letters* 9.7 (2024), pp. 6408–6415.

[8] Z. Sheng et al. "Wireless acoustic sensor networks and edge computing for rapid acoustic monitoring". In: *IEEE/CAA Journal of Automatica Sinica* 6.1 (2019), pp. 64–74.

[9] J. Yick, B. Mukherjee, and D. Ghosal. "Wireless sensor network survey". In: *Computer Networks* 52.12 (2008), pp. 2292–2330.

[10] Stefan Kahl et al. "BirdNET: A deep learning solution for avian diversity monitoring". In: *Ecological Informatics* 61 (2021), p. 101236. ISSN: 1574-9541. DOI: https://doi.org/10.1016/j.ecoinf.2021.101236. URL: https://www.sciencedirect.com/science/article/pii/S1574954121000273.

[11] H. Cao et al. "An optimization method to improve the performance of unmanned aerial vehicle wireless sensor networks". In: *International Journal of Distributed Sensor Networks* 13.4 (2017).

[12] S. Poudel and S. Moh. "Hybrid Path Planning for Efficient Data Collection in UAV-Aided WSNs for Emergency Applications". In: *Sensors* 21.8 (2021), p. 2839.

[13] P. Corke et al. "Autonomous deployment and repair of a sensor network using an unmanned aerial vehicle". In: *IEEE International Conference on Robotics and Automation* (2004).

[14] Alexander Millane et al. *nvblox: GPU-Accelerated Incremental Signed Distance Field Mapping*. 2024. arXiv: 2311.00626 [cs.RO]. URL: https://arxiv.org/abs/2311.00626.

[15] Mihir Dharmadhikari et al. "Motion Primitives-based Path Planning for Fast and Agile Exploration using Aerial Robots". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. 2020, pp. 179–185. DOI: 10.1109/ICRA40945.2020.9196964.

[16] T. Cieslewski, E. Kaufmann, and D. Scaramuzza. "Rapid exploration with multi-rotors: A frontier selection method for high speed flight". In: *IEEE International Conference on Intelligent Robots and Systems* (2017).

[17] A. Bircher et al. "Receding horizon next-best-view planner for 3D exploration". In: *IEEE International Conference on Robotics and Automation* (2016).

[18] M. Selin et al. "Efficient autonomous exploration planning of large-scale 3-d environments". In: *IEEE Robotics and Automation Letters* 4.2 (2019), pp. 1699–1706.

[19] B. Zhou et al. "FUEL: Fast UAV Exploration using Incremental Frontier Structure and Hierarchical Planning". In: *IEEE Robotics and Automation Letters* 6.2 (2021), pp. 779–786.

[20] T. Dang et al. "Graph-based Path Planning for Autonomous Robotic Exploration in Subterranean Environments". In: *IEEE International Conference on Intelligent Robots and Systems* (2019).

[21] E. Lee et al. "REAL: Rapid Exploration with Active Loop-Closing toward Large-Scale 3D Mapping using UAVs". In: *IEEE International Conference on Intelligent Robots and Systems* (2021).

[22] D. Mellinger and V. Kumar. "Minimum snap trajectory generation and control for quadrotors". In: *IEEE International Conference on Robotics and Automation* (2011).

[23] Helen Oleynikova et al. "Voxblox: Incremental 3D Euclidean Signed Distance Fields for on-board MAV planning". In: Sept. 2017, pp. 1366–1373. DOI: 10.1109/IROS.2017.8202315.

[24] Yue Pan et al. "Voxfield: Non-Projective Signed Distance Fields for Online Planning and 3D Reconstruction". In: *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2022, pp. 5331–5338. DOI: 10.1109/IROS47612.2022.9981318.

[25] L. Han et al. "FIESTA: Fast Incremental Euclidean Distance Fields for Online Motion Planning of Aerial Robots". In: *IEEE International Conference on Intelligent Robots and Systems* (2019). URL: https://arxiv.org/abs/1903.02144.

APPENDIX A
FINITE-STATE MACHINE

Besides the software modules present in Figure 4, an additional software module named *PX4 planner* was developed. This module sends the necessary commands to the PX4 flight controller via ROS messages, using the PX4-ROS2 bridge running in a docker in the Jetson board. In order to manage communication with the exploration planner, a finite-state machine (FSM) was implemented. The code runs in the PX4 offboard mode. Figure 14 below details the different states considered for a full exploration mission.

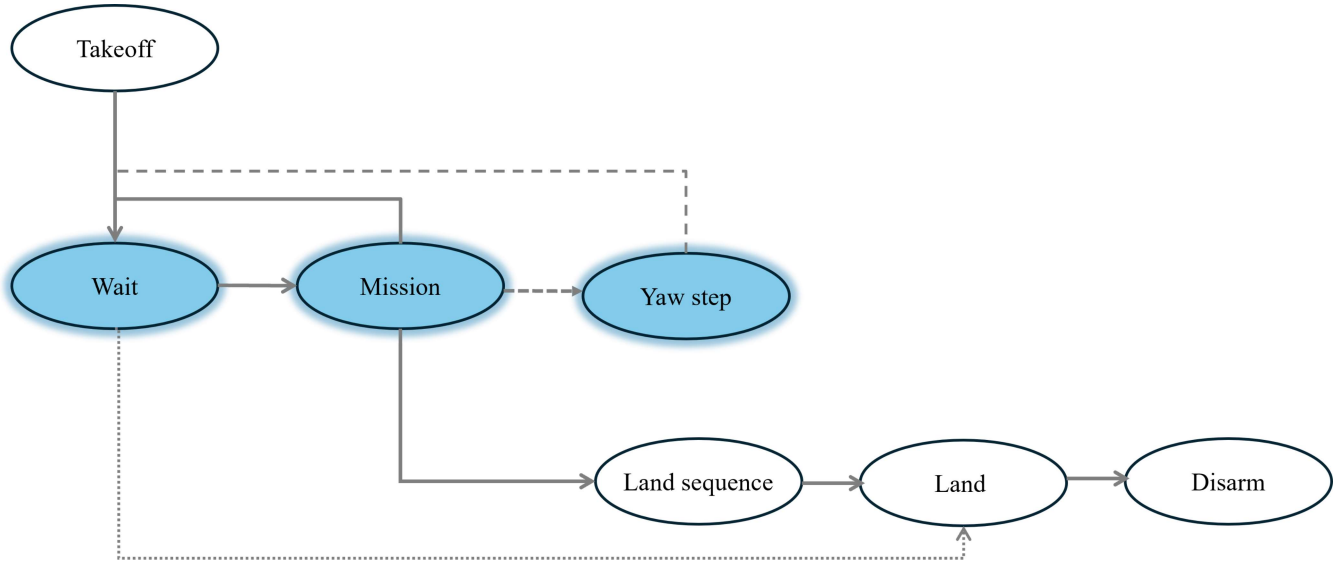


Fig. 14: Implemented FSM responsible for autonomous navigation. The generated commands, for each state, are sent to the PX4 controller via the PX4-ROS2 bridge. For the highlighted states, the FSM accepts and executes messages from the exploration planner. After taking-off to the mission height, the vehicle goes into wait state and hovers until a path is generated by the planner, activating the mission state. Otherwise, if no path is received after 30 seconds, the vehicle lands. During the mission state, a path is accepted and the yaw and position setpoints are commanded to the PX4 position controller. After a path is tracked, the wait state is activated for 2 seconds, after which time a new planner path is accepted. During the mission state, if a re-exploration message with a yaw angle command is received from the planner, the FSM enters the yaw step state to track the desired yaw. A yaw step is also followed by waiting 2 seconds, after which the vehicle is again in mission state and ready to received planner messages. Three end conditions are considered for the mission: 1) maximum mission duration is reached, 2) an out-of-bounds path is received and 3) an end of exploration message is received from the planner, due to the maximum number of re-exploration steps being reached. When the mission ends, the land sequence is activated and the previous positions are sent to the planner in reverse order, to return to the initial hover position. Finally, land and disarm commands complete the FSM logic.

APPENDIX B
SENSOR VIEW DISTANCE CALCULATION

As discussed in the paper, our method can be compared with an exhaustive search method, where the vehicle scans the entire available space and mapping and detected operations are conducted offline, via post-processing of the collected color and depth images. The total distance is obtained with horizontal and vertical distances y and x, respectively. These are the maximum distances in the horizontal plane that are sensed by the camera at each moment and depend on its characteristics, as shown in section B. Below Equation 3 and Equation 4 show the calculations to obtain x and y, for a mission at a height of $h = 1.5m$. Equation 5 is the complete calculation of the total distance traveled using the exhaustive search method.

$$x_1 = h \cdot \tan(31^\circ) = 0.9013m; \quad x_2 = \sqrt{d_{max}^2 - h^2} = 2.598m; \quad x = x_2 - x_1 = 1.697m \tag{3}$$

$$\frac{y}{2} = x_1 \cdot \tan(\frac{87^\circ}{2}) = 0.854 \Rightarrow y = 1.708m \tag{4}$$

$$dist_{total} = 5 \cdot y + 10 - x + 5 \cdot (10 - 2 \cdot x) = 49.87m \tag{5}$$

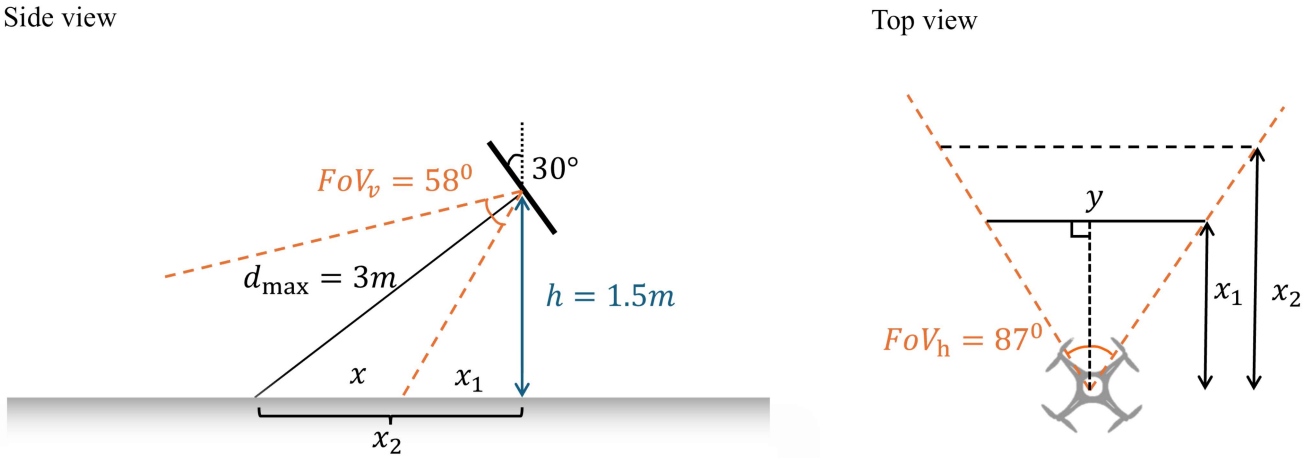
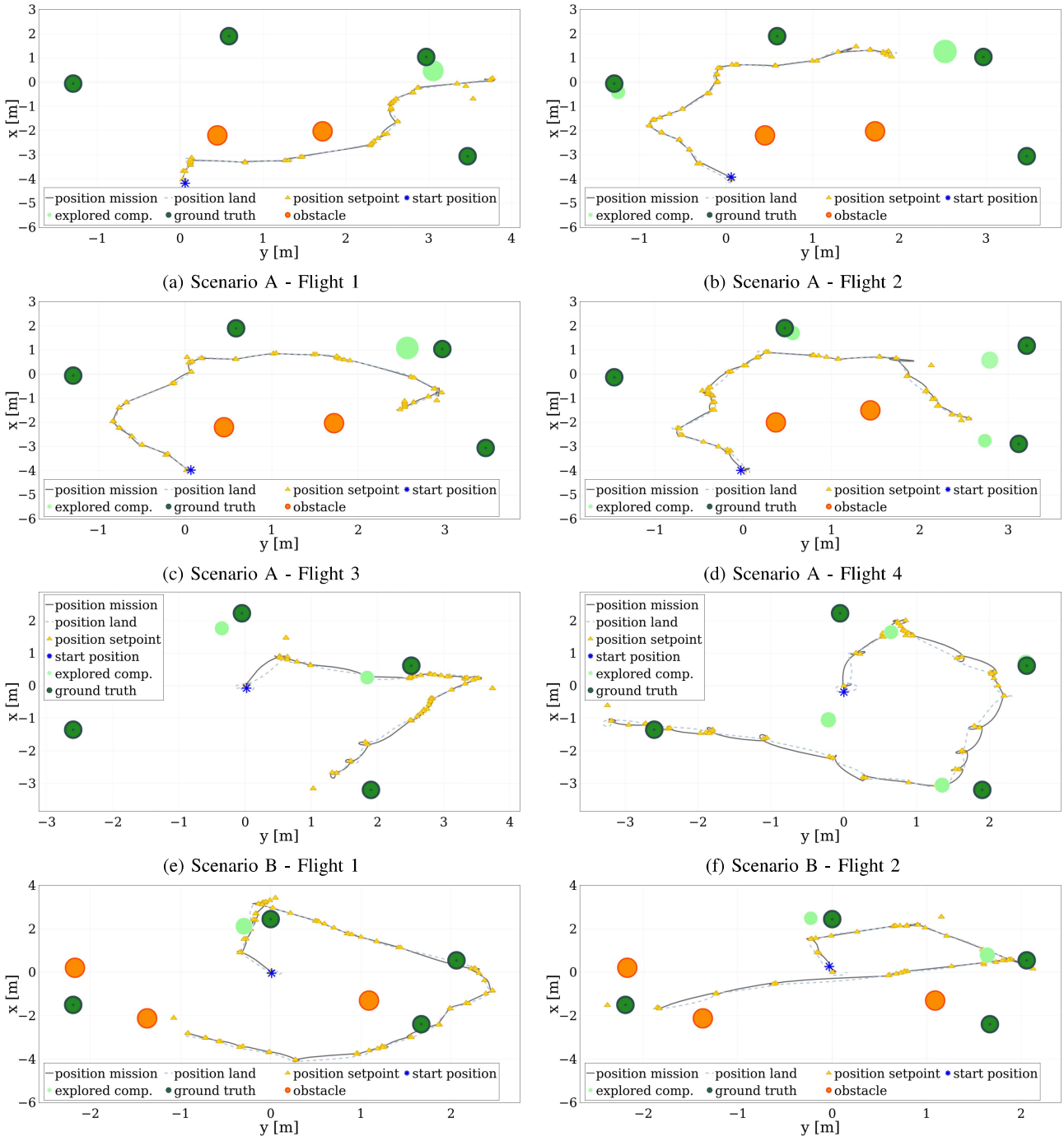


Fig. 15: Schematics used to obtain horizontal y and vertical x distances used in the exhaustive search approach in Figure 13. As can be seen, these distance values depend on the flight height and the depth camera range d_{max} and horizontal and vertical fields-of-view, FoV_v and FoV_h.

APPENDIX C
ADDITIONAL EXPERIMENTAL RESULTS



(a) Scenario A - Flight 1

(b) Scenario A - Flight 2

(c) Scenario A - Flight 3

(d) Scenario A - Flight 4

(e) Scenario B - Flight 1

(f) Scenario B - Flight 2

(g) Scenario C - Flight 1

(h) Scenario C - Flight 2

Fig. 16: Representative flight experiments for the three scenarios.

[This page is intentionally left blank]

Part II

Literature Review

*This part has been assessed for the course AE4020 Literature Study.

[This page is intentionally left blank]

UAV path planning for the deployment of an aerial sensor network

AE4020: Literature Study

Rita Santos Raminhos

Delft University of Technology

TU Delft

UAV path planning for the deployment of an aerial sensor network

by

Student Name	Student Number
Rita Santos Raminhos	5607817

Supervisor: S. Hamaza
Project Duration: April - June, 2023
Faculty: Faculty of Aerospace Engineering, Delft

TU Delft

Contents

Nomenclature iii

1 Introduction 1
1.1 Research question . . . 1
1.2 Content Overview . . . 2

2 Aerial sensor network 3

3 Path planning algorithms 5
3.1 Path planning and trajectory optimisation . . . 5
3.1.1 Configuration space . . . 5
3.2 Path planning algorithms . . . 6
3.2.1 Sampling-based algorithms . . . 7
3.2.2 Node based optimal algorithms . . . 8
3.2.3 Mathematic model based algorithms . . . 9
3.2.4 Bio-inspired algorithms . . . 10
3.2.5 Multi-fusion based algorithms . . . 11
3.2.6 Comparative analysis . . . 11

4 UAV path planning applications 12
4.1 Sensing . . . 12
4.2 Volumetric mapping . . . 13

5 Exploration path planners 15
5.1 Frontier based exploration . . . 15
5.2 Sampling based exploration . . . 15
5.3 Receding horizon method . . . 16
5.4 Gain function . . . 16
5.5 State-of-the-art planners . . . 16
5.5.1 NBVPlanner . . . 17
5.5.2 AEPlanner . . . 17
5.5.3 GBPlanner . . . 17
5.5.4 MPBPlanner . . . 18
5.5.5 FUEL . . . 19
5.5.6 REAL . . . 19
5.5.7 Considerations . . . 20

6 Goal-oriented path planners 22
6.1 Gradient-based trajectory optimisation . . . 22
6.1.1 B-splines parameterisation . . . 22
6.2 State-of-the-art planners . . . 23
6.2.1 FASTER . . . 23
6.2.2 EGO-Planner . . . 23
6.2.3 Bubble Planner . . . 23

7 Conclusion 25

References 27

List of Figures

2.1 Layer division in the rainforest and the three sensor delivery strategies (direct placement on tree trunks, impulsive launch in the canopy layer and perching on tree branch) proposed by S. Hamaza et al. [12]. 4
2.2 Representation of path planning for precise deployment of the aerial sensor network. . . 4

3.1 Expansion of mapped obstacle with the dimension of the robot's radius [17]. 6
3.2 Taxonomy proposed by L. Yang et al. to classify UAV path planning algorithms [19]. . . 6
3.3 Example of a roadmap in two-dimensional configuration space with obstacles (in grey) obtained in the first phase of PRM algorithm [18]. A very simple planner is used to connect the nodes with straight lines. 7
3.4 Representation of how RRT grows in two-dimensional configuration space [24]. 8
3.5 Path obtained with RRT (left) and RRT* (right) algorithms ran with 20000 samples in a simulation environment with obstacles (red) and goal (purple) [23]. 8
3.6 Simple example of Dijkstra's algorithm with 5 nodes [30]. 9
3.7 Method of mathematic model based algorithms [19]. 10

4.1 Point cloud obtained in a forest environment with a UAV-mounted LiDAR [47]. 13
4.2 Volumetric mapping progress in exploration path planning application [27]. 14

5.1 Frontier region (in green) update process based on new sensor data [59]. 16
5.2 Representation of cached points (left) and estimated gain vaues (right) in AEP [61]. . . 18
5.3 Architecture proposed for GBP method for exploration [28]. 18
5.4 Representation of sampling of motion primitives to obtain a configuration tree (left) and selection of collision-free and future-safe paths (right) in MBP [63]. 19
5.5 Representation of viewpoint refinement stage of FUEL path planning method [59]. . . . 20
5.6 Architecture of REAL [48]. 20
5.7 Representation of REAL path planning method [48]. 21

6.1 Representation of the convex hull property. It can be seen that the control points are marked in gray and the B-spline curve is clearly contained inside the dotted rectangle that can represent the free space for trajectory planning [54]. 23

List of Tables

3.1 Comparison between the different categories of path planning algorithm adapted from [19], including relevant examples for each category and evaluation of time complexity, type of environment (S - Static or D - Dynamic) and real-time processing (online vs offline). 11

5.1 Comparison between the different path planning methods for autonomous exploration. . 21

Nomenclature

Abbreviations

Abbreviation	Definition
ASN	Aerial Sensor Network
ESDF	Euclidean Signed Distance Field
FoV	Field of View
LiDAR	Light Detection and Ranging
MILP	Mixed Integer Linear Programming
NBV	Next-best-view
RRT	Rapidly-exploring random tree
RRG	Rapidly-exploring random graph
ToF	Time of Flight
TSDF	Truncated Signed Distance Field
UAV	Unmanned Aerial Vehicle
WSN	Wireless Sensor Network

Symbols

Symbol	Definition	Unit
$c(\sigma)$	Path cost function	m
$\mathcal{C}$	Configuration space	
$\mathcal{C}_{free}$	Free configuration space	
$\mathcal{F}$	Frontier vector	
$g_{vol}(\xi)$	Volumetric gain	
$g_{exp}(\xi)$	Exploration gain	
$\vec{p} = (x, y, z)$	Position vector	[m]
r_{map}	Radius of a voxel	m
V	Bounded volume in $\mathbb{R}^3$	m³
V_{un}	Part of V that is unknown	m³
V_{free}	Part of V that is free	m³
V_{obs}	Part of V that is occupied by obstcales	m³
V_{res}	Residual volume in V that cannot be perceived	m³
ψ	Yaw angle	rad
σ	Path	
σ'	Optimal path	
ξ	Configuration vector	

1 Introduction

In recent decades, technology for unmanned aerial vehicles (UAVs) has seen big advancements. The European Commission estimates that by 2035 the European drone market will be worth more than 10 billion euros per year and employ more than 100,000 people [1]. Multirotors are lightweight and agile UAVs that are easy to deploy. For this reason, they are used in a number of applications ranging from agriculture [2], surveillance [3], emergency response [4] and communications [5] to environmental research and monitoring [6, 7].

Technological progress in Micro-Electro-Mechanical Systems (MEMS) enabled the sensing industry to produce cheap, small and light sensors with some level of processing, communication and localization capabilities depending on the application. Sensor nodes can be combined straightforwardly to form a wireless sensor network (WSN) [8]. Such a network offers great potential for environment and biodiversity monitoring and survey applications [9] in locations such as rainforests.

Rainforest are very rich and diverse ecosystems despite representing only six percent of the Earth's surface - they are inhabited by more than half of known plant and animal species [10]. However, deployment of WSN in these environments is challenging due to a number of factors, including their remoteness, dense vegetation and humidity levels. Recent developments in aerial manipulators onboard multirotor platforms open the doors to more possibilities. In this field solutions and frameworks have been proposed for the deployment of sensors in forest environments using multirotors[11, 12].

The purpose of this report is to establish a research gap in the field of path planning for sensor deployment in the rainforest. Current research will be analysed to determine whether there are prior path planning implementations for the precise deployment of a sensor network with a UAV or how the current state-of-the-art can be adapted for this goal.

1.1. Research question

In the context of this project, the research question below is formulated. Several additional sub-questions are also identified.

How to design an efficient path planner for an aerial sensor network, with one hundred sensor nodes, to map and navigate to precise deployment locations in an obstacle dense environment such as the rainforest using a quadrotor?

- How can we develop an exploration planner to explore the rainforest environment and map positions of interest for possible deployment locations?
- What are the parameters and constraints to select possible deployment locations?
- After possible deployment locations have been identified, should path planning be conducted offline? Which algorithm is more efficient to plan the actual deployment locations?
- Which method is suitable for online replanning towards the precise deployment locations while avoiding dynamic obstacles in the safest and fastest way?

1.2. Content Overview

This literature study report is divided into sequential chapters. First, chapter 2 defines aerial sensor network and elaborates on the context of the thesis project. Second, chapter 3 distinguishes path planning and trajectory optimisation and introduces the configuration space. Different path planning algorithms are classified and the most commonly used algorithms are discussed and compared. chapter 4 focuses on UAV path planning applications and covers important concepts related to sensing and volumetric mapping used by most planners. chapter 5 follows to elaborate on current state-of-the-art exploration planners. Next, chapter 6 briefly presents common approaches for trajectory optimisation and their relation with goal-oriented planners. Finally, the conclusion in chapter 7 brings back research question to bridge the findings and the proposed thesis project.

2

Aerial sensor network

The thesis project aims to develop a path planner for the deployment of an aerial sensor network in a rainforest environment. Therefore, this chapter starts by introducing the rainforest structure to be able to explain our concept of aerial sensor network. After presenting previous works from the literature that propose path planning for WSNs, we discuss the constraints and challenges for developing a planner for the deployment of an aerial sensor network.

Rainforests have a characteristic layered structure, as shown in Figure 2.1. Layers differ in the amount of water, sunlight and air circulation therefore plant and animal species vary in the different layers. Above the forest floor, there is the lower layer - understory layer. Above it, the canopy layer is found. This is the most densely vegetated layer. Finally the top layer is the emergent layer where taller trees are found [10].

We consider an aerial sensor network (ASN) a set of sensor nodes, including relay nodes, placed on top of the forest canopy with a quadrotor. After placement, the main goal of the ASN is to survey biodiversity. Regarding the deployment strategy, S. Hamaza et al. proposed the three methods [12] shown in Figure 2.1 in their work. Our method would be similar to their impulsive launch, except that it take place on the top of the canopy.

Depending on the sensor type, the data collected will naturally differ. For surveying in the rainforest, some sensors are more adequate than others. For instance, acoustic sensors are strong candidates given the lower dimensions of the data. Placed on top of the canopy, they will be able to capture the sounds of birds, bats and gliders in the emergent layers. Also the success of solutions for the post-processing of the acoustic data is increasing [13].

In the literature, UAVs have been used for aerial collection of data from WSNs. Several works have proposed path planners for this end [14, 15]. However, they assume random deployment of sensor nodes. In [16] an autonomous helicopter is used and a solution is proposed for deployment of sensors but in a uniform way. Therefore, following extensive literature research, path planning that takes into account a map of the environment to place the sensors and the whole network precisely in positions of interest has not been implemented to the best of our knowledge.

Figure 2.2 below shows an aerial sensor network as described. The path is planned towards previously mapped deployment locations. We consider that planning the deployment position of the elements of an aerial sensor network is not trivial and presents a number of challenges. For optimal placement sensor characteristics such as range should be taken into account. At the same time, the whole network has to be considered and there are connectivity constraints with the relay nodes.

We consider that the architecture of the sensor network is outside the scope of the thesis project. Hence, parameters such as the number of sensors and relay nodes will be inputs to the path planner. Another possibility is including bio-environmental factors into planning such as proximity to bodies of water.

In addition, challenging factors are mostly related to the environment characteristic and the UAV platform. The limited endurance of lightweight drones is one of the main difficulties of path planning in a large-scale environment. For this reason, the efficiency of the developed planner is important.

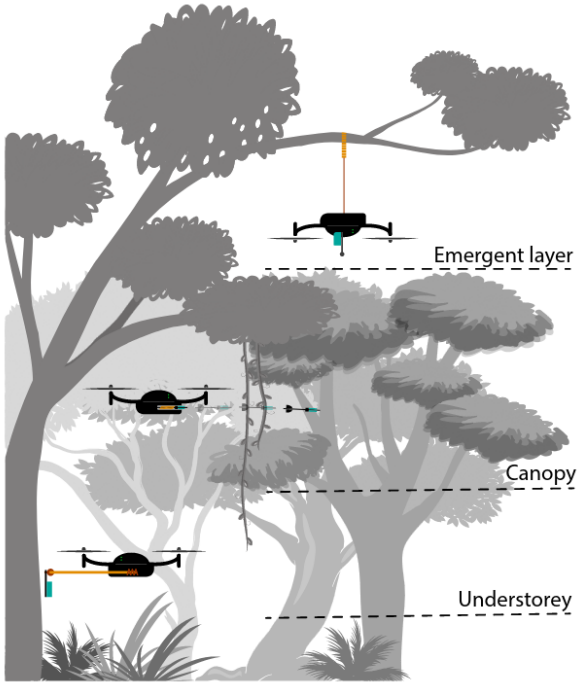


Figure 2.1: Layer division in the rainforest and the three sensor delivery strategies (direct placement on tree trunks, impulsive launch in the canopy layer and perching on tree branch) proposed by S. Hamaza et al. [12].

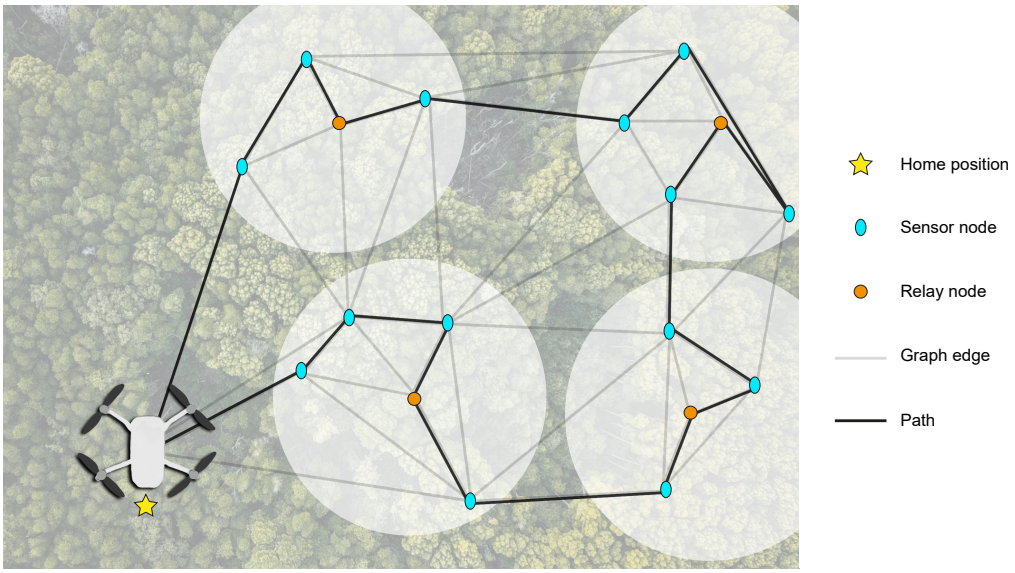


Figure 2.2: Representation of path planning for precise deployment of the aerial sensor network.

3

Path planning algorithms

The current chapter begins by defining path planning and trajectory optimisation in section 3.1 and the configuration space in subsection 3.1.1. Then in section 3.2 path planning algorithms are classified and relevant algorithms are introduced and compared.

3.1. Path planning and trajectory optimisation

Studies of motion planning commonly distinguish two aspects of the problem: front-end discrete path planning and back-end continuous trajectory optimisation [17]. Although many research works propose a solution for both the front- and back-end, there are different methods for each.

Path planning algorithms aim to find, for a certain iteration, the next best position within a configuration space to achieve a certain goal. This means that path planning algorithms do not consider the time variable and they return a geometric solution - the path.

When this path is parameterised in time a trajectory is obtained [18]. Trajectory optimisation algorithms compute a trajectory that satisfies feasibility constraints (kinematic, dynamic and safety constraints for example) while optimising for certain factors such as smoothness [17]. When the trajectory is parameterised as a twice-differentiable polynomial it is simple to obtain the velocity and acceleration from its first and second derivative, respectively. Thus, the desired next state (position, velocity, acceleration) is known.

3.1.1. Configuration space

To give a more complete definition of path planning it is important to define the configuration space. Similarly to the literature, in this work we consider a bounded volume V in three-dimensional space ($\mathbb{R}^3$). We further define three categories of V: unknown space V_{un}, free space V_{free} and occupied space V_{obs}. V_{free} is the part of V that we are certain contains no obstacles and is safe for the robot to transverse. The subsets of V that contain obstacles are classified as V_{obs}. V_{un} is the volume for which we don't have sufficient information to classify as V_{free} or V_{obs}.

For path planning the configuration space $\mathcal{C}$ is generally considered [18]. The configuration space is the "set of possible transformations that could be applied to robots" [17]. Naturally, these possible robot configurations are contained in V. Then the free configuration space $\mathcal{C}_{free}$ is the set of possible configurations within V_{free}. A common practice in path planning applications is to expand the obstacles in occupied configuration $\mathcal{C}_{obs}$ space with the radius of the robot. Subsequently the robot can be represented as a point in space which simplifies collision avoidance computations. This is illustrated in Figure 3.1.

Adopting a similar definition to the ones given in [18, 19], path planning involves finding a path $\sigma : [0, T] \rightarrow V$ where $\sigma(0) = \xi_{init}$ and $\sigma(T) = \xi_{goal}$ such that $\sigma(\tau) \in \mathcal{C}_{free}$ for all $\tau \in [0, T]$, if such a path exists. σ_{init} and σ_{goal} are the initial and goal configurations, respectively and both belong to $\mathcal{C}_{free}$.

L. Yang et al. [19] further define optimal path planning as the process to find the optimal path σ' by minimising a cost function $c : \sum \rightarrow R \geq 0$ where $\sum$ is the set of solutions to the original path planning problem, so $c(\sigma') = min\{c(\sigma)\}$.

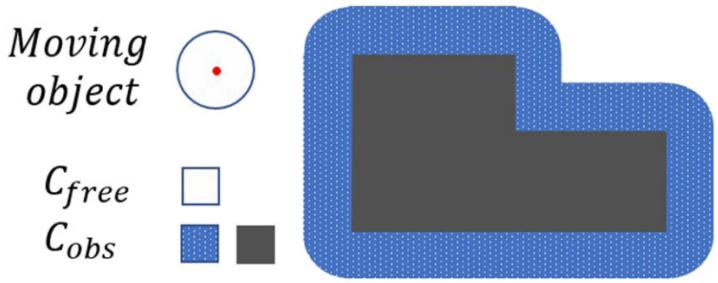


Figure 3.1: Expansion of mapped obstacle with the dimension of the robot's radius [17].

3.2. Path planning algorithms

Algorithms for path planning have been proposed for a number of applications using different methods. The most common classifications for path planning algorithms are based on their approach [17, 19, 20], complexity [21] and application [17].

In his extensive book on path planning algorithms S. M. LaValle [21] distinguishes sampling-based motion planning and combinatorial motion planning. The author makes this distinction based on the concept of completeness. To be considered complete an algorithm must be able to present a solution within finite time or communicate that one does not exist. Algorithms that match these requirements are called combinatorial or exact algorithms. Sampling-based algorithms aren't complete since they sample the configuration space in a determined way and can get stuck if there is no solution.

Sampling-based algorithms randomly sample the configuration space to obtain a graph of possible paths to the goal. Starting from an initial configuration of the robot, at each iteration sampling occurs and the random result is evaluated, for instance, to check for collisions. If it is accepted, the sampled nodes or edges are added to the graph. If the sampling density is sufficiently high, the probability that the algorithm finds a solution converges to one [21]. However, the solution found might not match the optimal solution. On the other hand, combinatorial algorithms consider the entire configuration space in their search for the optimal solution. For large-scale environments employing these methods is computationally expensive.

In their literature review work, L. Yang et al. [19] propose the taxonomy shown in Figure 3.2 to classify 3D path planning algorithms for UAVs. They distinguish five different approaches to planning: sampling-based algorithms, node based optimal algorithms, mathematic model based algorithms, bio-inspired algorithms and multi-fusion based algorithms. Their classification scheme will be adopted in this paper. For each category the characteristics will be elaborated below as well as more detail on individual methods.

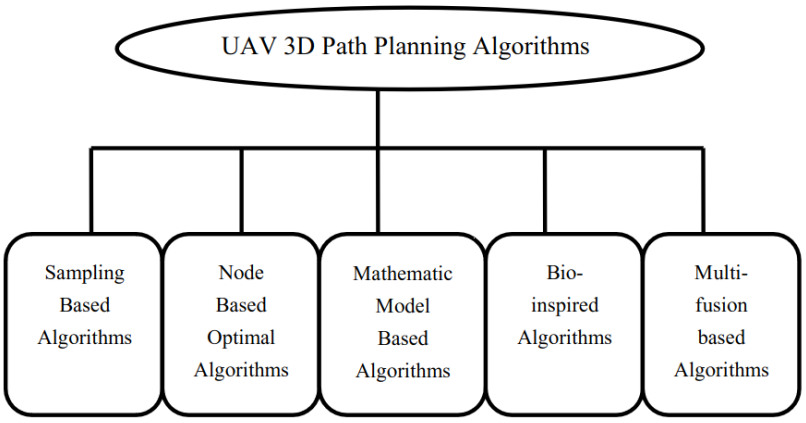


Figure 3.2: Taxonomy proposed by L. Yang et al. to classify UAV path planning algorithms [19].

3.2.1. Sampling-based algorithms

The approach of sampling-based algorithms has been described earlier. Probabilistic roadmaps (PRM) and rapidly-exploring random trees (RRTs) are sampling-based algorithms.

PRM series

The PRM algorithm [22] works in two stages. First, random nodes are sampled in the configuration space and checked for feasibility (if they are collision free, i.e. in the free configuration space). Then they are connected by a local planner to build a graph, as can be observed in Figure 3.6 The second stage is the query stage. It involves using a graph search algorithm to obtain the shortest path from the initial configuration to the goal. Although it can be used as a single query method, PRM was developed to answer multiple queries given different starting configurations as inputs. More advanced versions of the PRM algorithm have been implemented, including PRM* [23].

PRM based algorithms have proved efficient for holonomic systems [22], i. e. systems where the number of total degrees of freedom matches the number of controllable degrees of freedom. However, they can't be directly applied to the path planning of non-holonomic robots. Three-dimensional and complex environments with narrow passages are also challenging for PRM algorithms to solve [21]. RRTs were proposed to fill this gap [24].

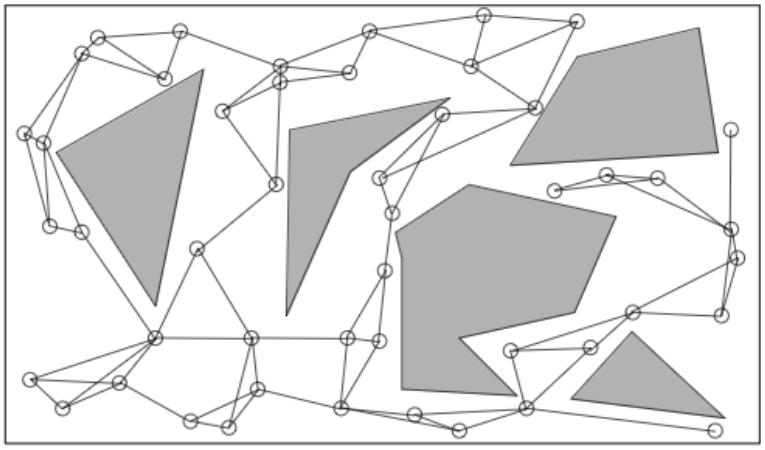


Figure 3.3: Example of a roadmap in two-dimensional configuration space with obstacles (in grey) obtained in the first phase of PRM algorithm [18]. A very simple planner is used to connect the nodes with straight lines.

RRT series

RRTs [25] is an algorithm that incrementally builds a random search tree in the configuration space. The initial configuration is the root of the tree and at each iteration a new edge or branch is added. This involves randomly sampling the neighbouring space to select a node. Then the connection between this node and the nearest tree node is evaluated. If it is feasible i.e. if it is collision-free, the new node is added to the tree graph and the process is repeated until the goal is reached. It has been shown that the algorithm is initially biased to rapidly explore the configuration space and ultimately converges to uniform coverage [25]. This is illustrated in Figure 3.4.

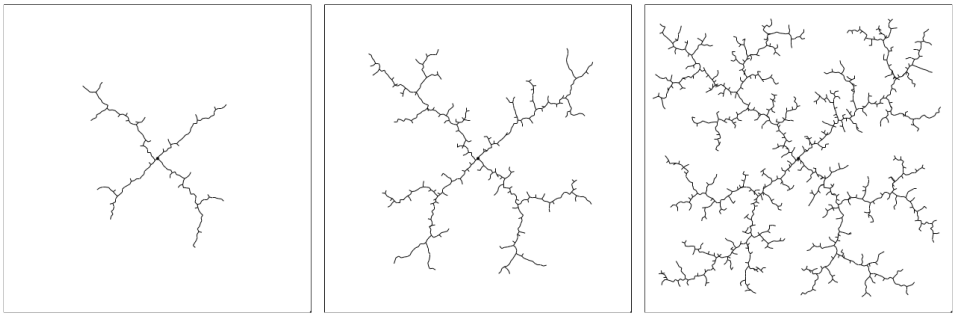


Figure 3.4: Representation of how RRT grows in two-dimensional configuration space [24].

Rapidly-exploring random graph (RRG) [26] is a variation of RRT where sampled nodes are also connected to other nodes within a set range. The path is searched from the obtained graph. It has been proven that when the number of samples approaches infinity the solution proposed by the algorihtm will aproach the optimal solution [23] - asymptotic optimality.

RRT* [23] is another algorithm derived from RRT that is also able to find a minimum cost path. RRT* works in a similar way, but after connecting a new node with the closest tree node it also evaluates the connection between the new node and all other tree nodes within a defined radius r. If a shorter path from the root can be obtained from rewiring the connections, the initial edge is deleted and the new tree structure is linked. Figure 3.5 shows an example of the differently constructed trees and the paths obtained with RRT and RRT*.

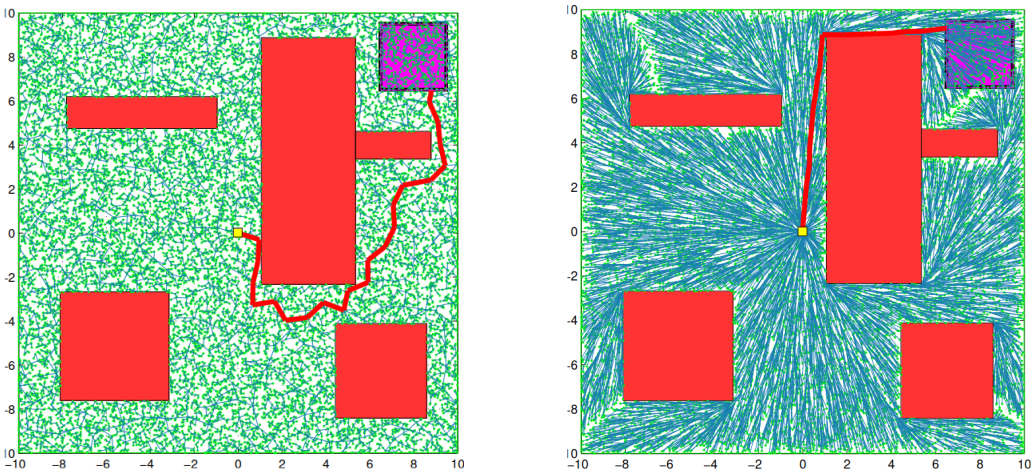


Figure 3.5: Path obtained with RRT (left) and RRT* (right) algorithms ran with 20000 samples in a simulation environment with obstacles (red) and goal (purple) [23].

Over the last decades since their introduction RRT and derived algorithms have been applied in numerous works in robotics, including as path planning modules for quadrotor applications [27, 28].

3.2.2. Node based optimal algorithms

Node based optimal algorithms approach the path planning problem as a graph search problem, making it simpler to find the optimal path. These algorithms define a cost function and the nodes in the prebuilt graph are searched to find the minimum-cost path. Dijkstra's algorithm and A star (A*) search are examples of widely used node based optimal algorithms.

Dijkstra's algorithm

Dijkstra's algorithm was proposed in 1959 [29] and is still one of the most representative algorithms for graph search.

At each run of the algorithm, a node is visited and a distance is calculated as the minimum found cost of reaching the current node from the starting node. Initially the starting point is set to 0 and the distance for all other nodes is infinity. The neighbouring nodes are analysed and the one associated with the lowest cost is chosen and its distance can be set equal to the calculated value. Then it is set as the next node to be visited and once again its neighbours will be analysed and the algorithm will follow the minimum cost path onto the next unvisited node. Each time the distance of visited nodes also has to be checked since it is possible that a lower cost path now exists through the newly visited node. Termination occurs when the goal node is visited.

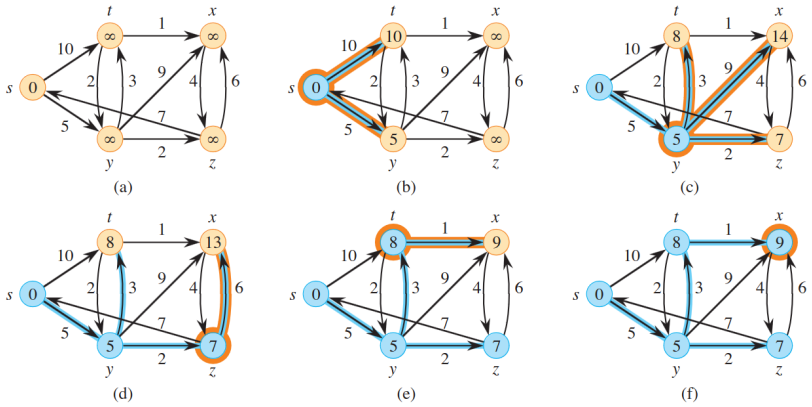


Figure 3.6: Simple example of Dijkstra's algorithm with 5 nodes [30].

A* series

A* evolved from Dijkstra's algorithm and incorporated heuristics to achieve optimality [31]. Based on a map grid, A* receives as inputs an initial and a goal position or node and returns the minimum cost path.

The algorithm defines an evaluation function $f(n) = g(n) + h(n)$ that represents the actual cost of a path from starting node n_s to the goal node n_g passing through a node n. Then $g(n)$ is the actual cost to go from n_s to node n and $h(n)$ is the cost of the path from n to n_g. If $\hat{g}(n)$ is an estimate of $g(n)$ and $\hat{h}(n)$ is an estimate of $h(n)$ then we can obtain an estimate for $\hat{f}(n) = \hat{g}(n) + \hat{h}(n)$. Now $\hat{h}(n)$ is a heuristic function that estimates the cost of the path from n to the goal n_g. The goal of the algorithm is to obtain the path that minimises $\hat{f}(n)$.

The algorithm defines two classes of nodes: open and closed nodes. Closed nodes are contained in an obstacle or belong to the optimal path. Open nodes are all other nodes. It starts by computing $f(n_s)$ for each of the possible successor nodes. The open node n for which $\hat{f}$ is smaller is selected and ties are resolved randomly. This node is then closed and $\hat{f}(n)$ is calculated for each successor of n. Each possible successor should be an open node including any closed node that now has a lower value of $\hat{f}(n)$ when recalculated. Once again the node corresponding to the lowest value of $\hat{f}(n)$ is selected and closed and the process can be repeated. When the goal is reached the algorithm is terminated.

Choosing an appropriate heuristic function is important. If $\hat{h}(n) \leq h(n), \forall n$ that means that $\hat{h}(n)$ never overestimates the real cost to reach the goal. In this case it has been proven that A* will return an optimal plan [32]. A simple example is provided by LaValle [21] considering a 2-D map with obstacles: calculating the distance by going in a straight line from a node to the goal is an underestimate of the cost function. The actual value will naturally be higher if the path has to go around obstacles.

The Jump Point Search (JPS) [33] algorithm is derived from A* for uniform cost grids, achieving a computation time around one order of magnitude faster than A* using graph pruning.

3.2.3. Mathematic model based algorithms

Mathematic model based algorithms implement methods such as Linear Programming, Nonlinear Programming (NLP) [34], Optimal Control [35] or others. Linear Programming includes Mixed Integer Linear Programming (MILP) [36]. Although these methods can be considered sampling-based algorithms,

the authors decided to include them in a separate category due to the different planning processes and inherent computational complexity [19].

These algorithms model the environment and the robot and apply a set of kinematic and dynamic (kinodynamic) constraints. A cost function is defined to find the optimal path. Figure 3.7 from their paper represents this method.

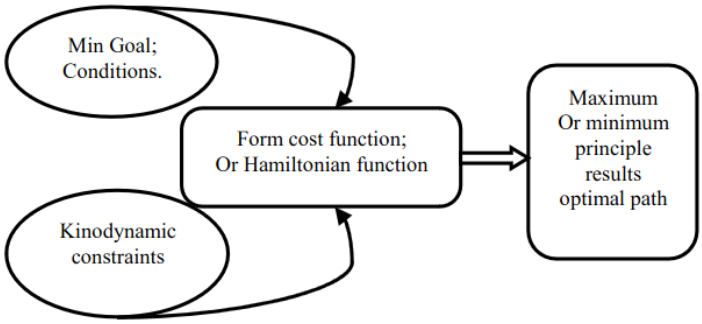


Figure 3.7: Method of mathematic model based algorithms [19].

3.2.4. Bio-inspired algorithms

Bio-inspired algorithms apply techniques based on biological evolution to solve the path planning problem. L. Yang et al. [19] subdivide this category into two subcategories: Evolutionary Algorithms and Neural Network algorithms. For path planning some of the most used evolutionary algorithms are genetic algorithms (GA) [37], ant colony optimisation (ACO) [38], particle swarm optimisation (PSO) [39], artificial bee colony (ABC) [15].

Genetic Algorithms

In GA for path planning chromosomes are used to represent paths where each gene that constitutes the chromosome is a random feasible node. GAs start by initialising a population (set of chromosomes) that is then evaluated based on a defined fitness function. The fitness function considers criteria such as proximity to goal or obstacle positioning. To obtain the next generation only the fittest parents are considered. At this stage crossover and mutation of the parents genes take place. The whole process is repeated until the termination criteria is achieved.

Ant colony optimisation

ACO algorithm is inspired by ants that release pheromones when they find food. Then other ants can follow the ”marked” shortest path to the food source. In the same way, ACO algorithms explore the environment and record potentially good positions. Compared to other evolutionary algorithms, improved versions of the basic ACO method are able to incorporate multiple objectives and continuous planning [40].

Particle swarm optimisation

PSO is inspired by flocks of birds that are able to share information about food sources by flying in a group. Similarly, PSO algorithms explore the environment with particles that know their best location and the swarm’s best location. This is taken into account in successive planning steps.

Artificial bee colony

ACO algorithms imitate the different roles of bees in food search activities. There are three types of bees: employed bees that keep track and search previous food sources; onlooker bees that evaluate these sources to select the best one and scout bees that are responsible for exploration to find new food sources. In path planning applications implementing ABC logic promotes both local and global exploration [15].

3.2.5. Multi-fusion based algorithms

The final classification proposed by the authors is multi-fusion based algorithms. These are path planning algorithms that combine of multiple algorithms to obtain a desired optimal path. This is a very common approach for applications in the UAV field as will be seen in the next chapters.

3.2.6. Comparative analysis

Table 3.1 adapted from the paper by L. Yang et al. [19] compares sampling based, node based, mathematical model based and bio-inspired algorithms in terms of time complexity, type of environment (static or dynamic) and processing in real-time (online vs offline). Compared with the approaches of sampling based and node based algorithms, bio-inspired methods have significantly higher computational times which means they are unsuitable for most online planning applications. Mathematic model based algorithms also have higher computational requirements. However, as we will observe in chapter 6 they can be employed online for local planning.

To compare sampling based and node based approaches, C. Zammit and E. van Kampen [41] studied the performance of RRT and A*. Their results confirm the theoretical properties of each method. While RRT explores the space uniformly but doesn't find an optimal solution, A* covers only a part of the space to converge to the optimal path. The authors propose that A* is more suited for online 3D path planning environments with static and dynamic obstacles in quadrotor applications subject to battery constraints, given its optimality and low computational times. On the other hand, RRT will perform better for exploration of uniform spaces. Nonetheless they find that if sampled nodes are checked in terms of obstacle positioning and dynamics, RRT can produce faster and optimal results.

Table 3.1: Comparison between the different categories of path planning algorithm adapted from [19], including relevant examples for each category and evaluation of time complexity, type of environment (S - Static or D - Dynamic) and real-time processing (online vs offline).

Method	Examples	Time complexity	Environment	Real-time
Sampling based	PRM [22], RRT [25], RRG [26], RRT* [23]	$\mathcal{O}(n \log n) \leq T \leq \mathcal{O}(n^2)$	S and D	Online
Node based	Dijkstra's [29], A* [31], JPS [33]	$\mathcal{O}(m \log n) \leq T \leq \mathcal{O}(n^2)$	S and D	Online
Math. Model Based	MILP [36], NLP [34], Optimal Control [35]	Depends on polynomial eq.	S and D	Oflline
Bio-inspired	GA [37], ACO [38], PSO [39], ABC [15]	$T \geq \mathcal{O}(n^2)$	S	Offline

4

UAV path planning applications

In their literature survey, L. Quan et al. [17] synthesize the main UAV applications where path planning algorithms play an important role. In a similar way, we distinguish different planners for UAVs: goal-oriented planners, exploration planners, uncertainty-aware planners and planners for collaborative robotic systems.

Goal-oriented path planning is the process of reaching a goal configuration autonomously from a starting configuration. It is the most basic form of path planning necessary for most UAV applications [17].

Additionally, a number of applications relates to the information collected by the UAV while flying. This is the case for exploration path planners that explore and map a previously unknown environment. Another possibility uncertainty-aware planning that aims to ”reduce the uncertainty or improve the accuracy of ego-motion estimation” [17].

Collaborative robotic systems are multiple UAVs or UAVs and ground robots that work together to perform a variety of tasks [42, 43].

Finally, multi-objective path planners have also been proposed [44, 45]. These planning algorithms weight multiple goals or adapt them according to the environment characteristics.

For the thesis project, goal-oriented and exploration planners are most relevant. Therefore the state-of-the-art exploration planners will be covered in chapter 5 and in goal-oriented planners chapter 6. The remaining of the current chapter will cover some concepts related to sensing (section 4.1) and volumetric mapping (section 4.2) used by these path planners.

4.1. Sensing

UAVs are equipped with multiple sensors that measure different variables. An Inertial Measurement Unit (IMU) is composed of accelerometers that measure linear accelerations and gyroscopes to measure rotational accelerations. A GPS allows the drone to know its current position and velocity. These are essential sensors in lightweight UAVs for most applications. Obstacle avoidance or exploration also require visual sensing of the environment and information on the distance to obstacles (depth information). These algorithms also have to account for the field of view (FoV) and range covered by the sensor.

As defined previously and similarly to the literature, volume V is a bounded volume in 3D space. V can be divided into unknown space V_{un}, free space V_{free} and occupied space V_{obs}. Cameras and Light Detection and Ranging (LiDAR) systems are examples of sensors used to perceive V.

While mono cameras use a single lens to capture a 2D image of the environment, stereo cameras integrate the images from two or more lens. In this case depth data can be extracted using triangulation. Another possibility are RGB-Depth (RGB-D) cameras that include a depth sensor to produce both color and depth data.

Time of Flight (ToF) sensors emit a set of pulses per unit time and capture the reflected signals. The range to the object is determined by measuring the time between emission and detection. LiDAR systems are ToF sensors that work with one or more laser beams. Multiple LiDAR measurements are

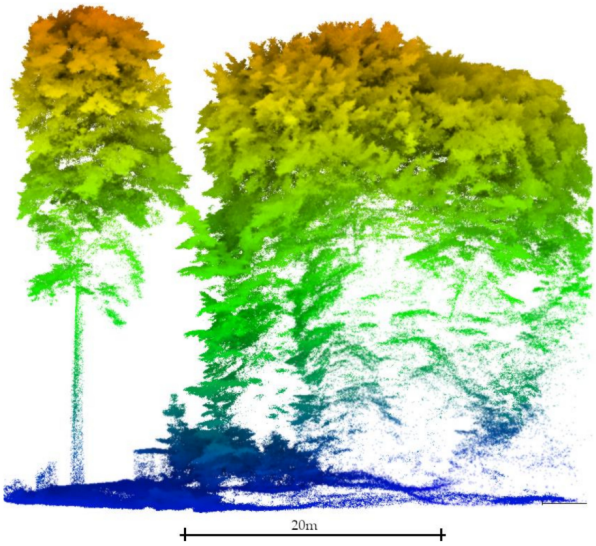


Figure 4.1: Point cloud obtained in a forest environment with a UAV-mounted LiDAR [47].

combined to form a point cloud from which the environment can be mapped in 3D. Other types of ToF sensors exist for example using infrared light but these have lower range and FoV [46]. Figure 4.1 shows an example of a point cloud obtained with a UAV's LiDAR.

Visual-Inertial Odometry (VIO) systems combine visual and IMU data for estimation of the drone's pose. These systems are necessary in GPS-denied environments and when testing path planning implementations in an indoor setting. Hence, VIO systems are implemented in several of the path planning solutions that will be presented in the next chapters

It is important to note that other sensing solutions exist but it is impossible to cover all of them. The sensor types mentioned are the ones most commonly used in the implementations researched for autonomous navigation and exploration.

4.2. Volumetric mapping

A 3D representation of the environment is necessary for the applications covered in this literature study. A common approach in UAV path planning applications is volumetric mapping [48, 49] that divides the volume into cubic units named voxels with radius r_{map}. Depending on the chosen mapping system, voxels can be classified as unknown, free or occupied to create an occupancy map or contain information on the distance to occupied voxels. This information is based on the integration and treatment of sensor data.

Multiple frameworks for 3D mapping have been proposed and applied in online UAV path planning implementations. Octomap [50], Voxblox [51] and FIESTA [52] are widely used for autonomous exploration and their source code is available open source. Figure 4.2 shows how a volumetric map is progressively built using an exploration path planner.

Octomap builds an occupancy map using a hierarchical data structure for 3D spatial subdivision called *octree*. Their method is applied to fused sensor data to calculate the probability of occupancy for the sensed voxels. Because of the hierarchical way in which mapping data is stored, the tree can be queried for occupied voxels at different resolutions, up to a maximum value.

Euclidean Signed Distance Fields (ESDFs) are an alternative way of representation to occupancy maps. An ESDF map is divided in voxels and each voxel includes the Euclidean distance to the closest occupied voxel. This facilitates online collision checking and choosing collision-free paths necessary in a number of path planning applications [53]. For gradient-based planning methods such as [54] occupancy maps aren't enough since they don't provide any information on the gradient to obstacles. ESDF representations are then useful for these methods, since a gradient computation occurs naturally.

TSDF (Truncated Signed Distance Field) is a representation developed in 2011 [55] and used in computer graphics and surface reconstruction applications. Compared to the ESDF, the difference is the

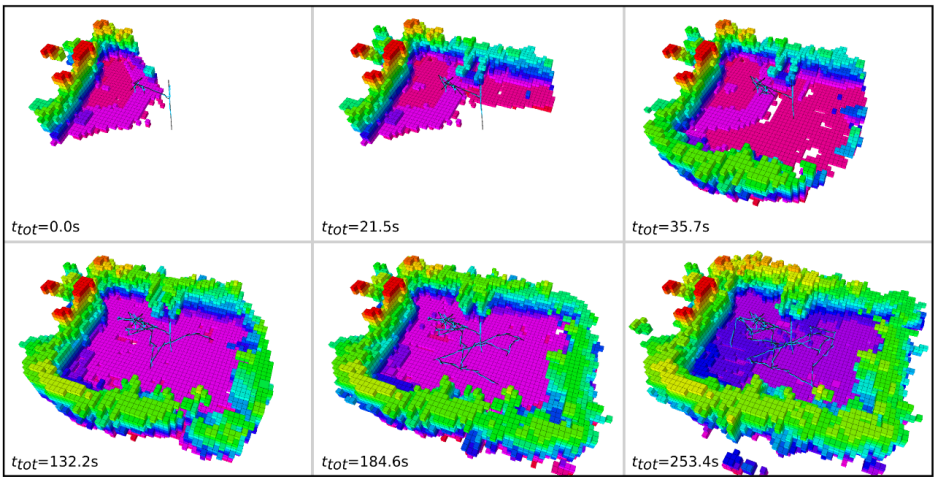


Figure 4.2: Volumetric mapping progress in exploration path planning application [27].

way of calculating the distance for each voxel. TSDFs compute the distance along the direction of the casted sensor ray from the sensor center to the nearest occupied voxel [53]. This distance calculated along one dimension is called *projective* distance.

The papers that propose Voxblox and FIESTA construct ESDFs. Voxblox [51] works by building TSDFs and using the distance values to update voxels and compute their distances in an ESDF map. Fiesta [52] also propose a method to update the ESDF map incrementally based on their own data structures and an algorithm to handle and optimize map updates.

The authors of Voxblox acknowledge two sources of error of their system because of building the ESDF from TSDFs. First, the TSDF projective distance can overestimate the real Euclidean distance to the closest obstacle despite this error decreasing with the number of observations. The second type of error is related to the calculation of quasi-Euclidean distances (that is the distance is only measured along horizonal, vertical and diagonal line segments) to build the ESDF map for faster computation purposes. Safety margin factors are defined in their paper to account for these errors.

The authors of Fiesta claim that their approach allows for building the ESDF directly without these errors [52]. They also compared the performance of FIESTA and Voxblox in experiments with real-world datasets and report an improvement for both accuracy and performance metrics of around one order of magnitude.

Very recently Y. Pan et al. [56] proposed and validated a new framework named Voxfield. Their system is a TSDF-based method to build an ESDF representation in a similar approach to Voxblox. They implement a new way of calculating non-projective distances on a TSDF map combined with an efficient ESDF update algorithm. A complete comparison with state-of-the-art mapping systems including Voxblox [51] and Fiesta [52] is presented where the authors claim better accuracy and computation times. Compared to Voxblox they achieve lower error values on TSDF maps and significantly lower error values on the ESDF map in tests with different datasets. Regarding the ESDF map, their results show 15% higher accuracy than FIESTA. The authors explain this result with the different way of computing the Euclidean distance. In FIESTA's implementation the distance is computed from the center of the origin voxel to to center of the closest voxel that is occupied while in their implementation the distance is considered up to the surface of the obstacle.

5

Exploration path planners

The exploration path planning problem consists of finding a safe (collision-free) path that maximizes the exploration and mapping of unknown volume V while satisfying kinodynamic constraints. For any path planning algorithm for exploration initially $V = V_{un}$. The goal is to fully determine $V = V_{free} \cup V_{obs}$ and eliminate all unknown space $V_{un} = \varnothing$ or reduce it to the residual volume $V_{un} = V_{res}$ that is the volume that cannot be perceived due to the environment characteristics (such as stretches that are too narrow for the multirotor to navigate).

Path planning starts from an initial configuration ξ_{init}. It is common to define the configuration as the flat state with position and yaw $\xi = (x, y, z, \psi)^T$. Then each planning step computes a path σ_{k-1}^k to reach ξ_k from the previous configuration ξ_{k-1}. This path has to be collision-free and respect the UAV's dynamic constraints. The path cost is defined $c(\sigma_{k-1}^k)$. A gain function can be used to determine the best path for exploration among all collision-free paths. More detail will be provided in section 5.4.

Different strategies have been proposed for exploration. Frontier based and sampling based exploration are representative methods. While they were proposed by earlier works, they are still used in recent applications and are introduced in section 5.1 and section 5.2 below. The receding horizon approach is also presented in section 5.3 and the gain function in section 5.4. Finally, section 5.5 analyses state-of-the-art exploration planners proposed for multirotor applications.

5.1. Frontier based exploration

Frontiers were first defined in [57] as the border region between mapped space and unknown space. The path planning algorithm presented uses frontiers to accomplish the exploration goal by repeatedly directing the robot towards these unexplored areas of space. Improved versions of this method have been proposed recently for exploration with UAVs such as Rapid by T. Cieslewski, E. Kaufmann, and D. Scaramuzza [58].

In current 3D path planning applications the frontier region usually refers to the set of unknown voxels that is adjacent to "known" voxels belonging to V_{free} or V_{obs}. It can be defined mathematically as:

$$\mathcal{F} = \{\vec{v_f} | \vec{v_f} \in V_{free}, \;\; \exists \vec{v_u} \in V_{un}, \;\; ||\vec{v_f} - \vec{v_u}|| = r_{map}\}, \tag{5.1}$$

where $\vec{v_f}$ and $\vec{v_u}$ are the centroid positions of a free and unknown voxel respectively. For better understanding Figure 5.1 highlights frontier regions in 2D.

5.2. Sampling based exploration

In the literature, sampling based exploration is often related to sampling of "next-best-views". While earlier mentions exist, the concept of "next-best-views" (NBV) is well introduced in [60] as the best sensor view among all candidates for the next planning step. The determination of the NBV depends on the amount of new information that can be sensed as well as positioning (e.g. obstacles in the way) and sensing constraints (range and FoV of the UAV's camera).

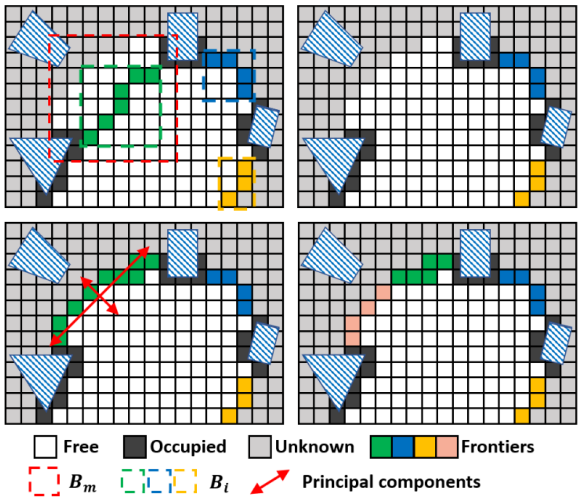


Figure 5.1: Frontier region (in green) update process based on new sensor data [59].

Among the current state-of-the-art of exploration path planning algorithms, the implementation by A. Bircher et al. [27] samples NBVs within the free configuration space as nodes of an RRT. Their work will be analysed in more detail in subsection 5.5.1.

5.3. Receding horizon method

A number of state-of-the-art algorithms for exploration path planning use a receding horizon (RH) strategy. This approach is inspired by model predictive controllers [35] where at each step a sequence of future control inputs is obtained based on the mathematical model and the current system state. Receding horizon means that only the first calculated control signal is implemented and the optimization process is repeated in the next step.

This concept has been applied successfully to UAV path planning problems [27, 48, 49]. In these implementations at each planning step the best future configurations are computed but only the next configuration is used and the process is repeated for that configuration with new information.

5.4. Gain function

Most exploration planners define an exploration gain in order to select the best path. This gain is computed based on the volumetric gain, while penalizing longer paths (paths with higher cost c(σ)).

The volumetric gain is the amount of volume i.e. the number of voxels that can be perceived from a certain configuration taking into account the sensor characteristics and the environment geometry. For example, if the sensor range and FoV cover a part of the environment from the current position, the volumetric gain is higher than when there is an obstacle in the sensor's FoV. The volumetric gain is estimated from the limited knowledge at that moment, since the volumetric map $\mathcal{M}$ is built incrementally when new sensor data is available.

Different planning methods propose different mathematical definitions for the exploration gain. This will be further discussed in section 5.5 but we can present a general definition for the exploration gain g_{exp} of a candidate configuration:

$$g_{exp}(\xi_k) = g_{vol}(\mathcal{M}, \xi_k) \cdot e^{-\lambda c(\sigma^k_{k-1})}, \tag{5.2}$$

where g_{vol} is the volumetric gain and λ is the tuning factor for penalizing paths with longer distances.

5.5. State-of-the-art planners

This section will analyse in detail six state-of-the-art UAV path planning implementations for autonomous exploration. In a growing research field with a continuous influx of publications the selection process

was not an easy one. Besides considering the impact of these works within the academic community, our primary selection criteria was choosing papers that validate their method through real-world experiments, since one of the main goals of the thesis project is to implement a robust method and test it in real-world conditions.

5.5.1. NBVPlanner

A. Bircher et al. [27] proposed a Receding Horizon "Next-Best-View" Planner that we will designate NBVP. Their sampling based exploration method uses RRT to generate collision-free paths that are then evaluated based on their volumetric gain. At each planning step a tree is built from the current configuration with a set number of nodes. The gain of each tree node is calculated with the volumetric gain. Full paths up to the maximum dimension of the tree are analysed so the gain of a node is always summed with the gain of the previous node. Longer paths are also penalised. The branch or path to the node with the best gain is chosen but the receding horizon approach means that at each step only a part of the path is executed. In the next step, a new tree is constructed including the previous best path so that possible good solutions aren't lost.

The authors tested their algorithm in different simulation setups and a real-world experiment with a hexacopter. The exploration progress was measured as the amount of mapped volume in m^3 over time and compared with a frontier based planner. Other measured metrics include total exploration time and total computation time (from which average computation time for each planning step can be calculated). The results show that the performance of both algorithms is similar in a simple environment, while NBVP is more suited for exploring more complex and larger environments than the frontier based approach, due to significantly lower computation times.

However NBVP has been proven to get stuck in larger scale environments if an unexplored region is far from the current location, so the sampling method does not reach this area and no higher gain nodes are found [48, 61]. Having verified this behaviour of NBVP but also its advantages M. Selin et al. proposed AE planner.

5.5.2. AEPlanner

AE Planner [61] combines NBVP with a frontier based approach as a global planner. The authors implement node caching for the sampled nodes from the RRT step in NBVP, as can be seen in Figure 5.2. The nodes cached in previous iterations are used in two ways. First, they can be used to estimate the gain of newly sampled nodes. Second, since nodes with high gain values are linked with frontier regions these nodes can be used to direct exploration when the local planner can't find high gain nodes nearby. Therefore, exploration only ends when both the local and global planner report zero information gain.

The authors also implement a way of computing the best yaw. At each planning step, before building the RRT, ray casting is done to determine the best yaw angle that maximizes the potential information gain from that configuration. This decreases the sampling space for the RRT.

Results from both simulations and a real-world experiment are presented in their paper. Similarly to [27], the metrics considered are exploration progress (volume explored in m^3 over time), total exploration time and computational time for gain estimation and collision checking. NBVP and AEP are compared in a large-scale simulated environment of a maze. AEP completes the exploration time in significantly less time while NBVP only covers the full space when the tuning factor is low, meaning that local exploration could be compromised in more complex environments. The results also validate the gain estimation from cached nodes, since computational times for this component are lower compared to NBVP. However, the calculation of best yaw is not completely validated since their method with this feature disabled is actually shown to perform faster exploration. This is explained due to RRT's random sampling while the best yaw method can get stuck locally for longer times.

5.5.3. GBPlanner

Graph-based planner (GBP) [28] is an autonomous exploration path planner designed for subterranean conditions in the context of the DARPA Subterranean Challenge [62]. These environments are characteristically large-scale and constricted in space. The implemented planner incrementally builds two

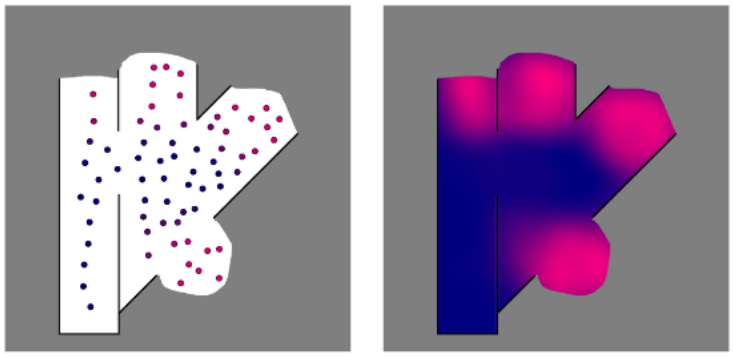


Figure 5.2: Representation of cached points (left) and estimated gain vaues (right) in AEP [61].

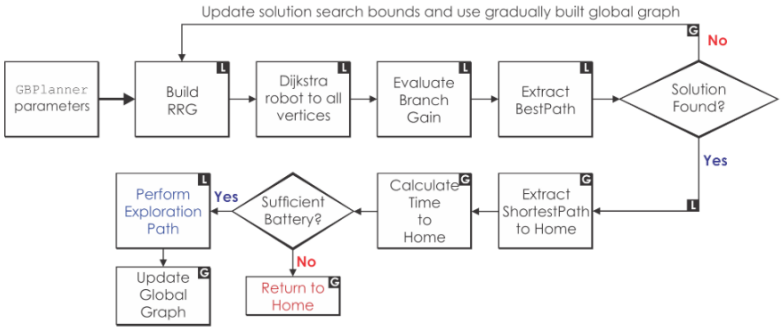


Figure 5.3: Architecture proposed for GBP method for exploration [28].

graphs: one for local exploration and another for the global planner. This architecture is represented in Figure 5.3.

The first step for the local planner is to build an RRG in V around the current configuration. Each sampled configuration is checked and if it belongs to V_{free} it is connected to nearby nodes. After the building step, Dijkstra's algorithm is used to obtain the shortest paths from the graph. The volumetric gain is then computed for each path. In this case the authors an additional weight functions from the usual cost function related to the Euclidean distance: a function that computes the distance between the path considered and a path in a straight line following the current estimated exploration direction, to penalize sharp changes of direction. As usual, the path with the highest calculated gain is selected. If the gain is below a set threshold the global planner takes over.

The global planner includes a global graph and a return-to-home feature. After each local planning step, the global graph is updated with the highest gain path and other high gain paths from the local planner. The shortest path back to the home position is recomputed. In case the local planner was unable to provide a solution, the global graph is used so the search space is increased to discover unexplored regions.

Compared to other methods, GBP planner has been tested more extensively in real-world conditions. For instance, two experiments took place in underground mines therefore the UAV was actually subjected to the difficult lighting conditions of these environments.

5.5.4. MPBPlanner

Also motivated by the DARPA Subterranean Challenge, a motion primitives-based path planner (MPB) [63] was proposed. A local planner that samples the control space is proposed to achieve faster exploration.

Besides the position and heading considered by the previous planners, the authors of MPB include velocity states in the configuration definition. The first step for the local planner is to build a tree of possible new configurations from the current one. This works by randomly sampling acceleration control

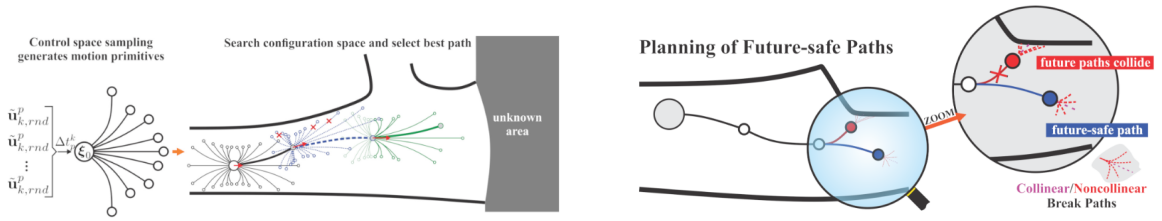


Figure 5.4: Representation of sampling of motion primitives to obtain a configuration tree (left) and selection of collision-free and future-safe paths (right) in MBP [63].

inputs that correspond to the new tree configurations if they are collision-free and future-safe. This last concept is introduced by the authors since zero velocity is not assumed for the new configuration unlike other planners. Therefore, sampled configurations that don't have null velocity values are checked for a collision-free path that is able to reach a final position with null velocity (hover), within the system's kinodynamic limits. Figure 5.4 from their paper illustrates the planning process.

The second step is calculating the volumetric gain of the tree of collision-free and future-safe paths. The calculation is identical to the one described for GBP. Then the path with the highest gain can be selected and tracked by the controller.

Comparably to GBP, MPB was tested in two distinct underground mines. One of the tests took place in a mine $165m$ long and an average flight speed of $1.8m/s$ was achieved. Exploration rate in m^3/s and computational cost per iteration in s are the metrics considered by the authors.

5.5.5. FUEL

FUEL [59] is a frontier-based methodology for exploration of unknown environments. The authors propose a hierarchical planner that exploits information on the frontiers using their own data structure the Frontier Information Structure (FIS).

Most algorithms for frontier-based exploration consider the center of the frontier region [61]. The authors of FUEL create FIS to support their claim that more information of the frontiers can be obtained and used by exploration path planning algorithms. In their algorithm frontier regions are regularly updated based on new sensor information. For each frontier region or cluster a set of viewpoints is evenly created around the center. The coverage quality is evaluated and the viewpoints that reach a set threshold are considered while the rest are discarded.

Then, path planning occurs in three stages for each planning step. In the first stage global planning occurs between all detected frontier regions such that there is a path that passes through one viewpoint of each frontier. Second is what the authors call viewpoint refinement where only a section of the global path is considered and paths between multiple viewpoints are searched with a graph search algorithm, Figure 5.5. A local path is obtained with a cost function that penalizes longer distances and changes of direction. The third step performs back-end trajectory optimization with dynamically feasible B-splines based on the method in [64].

The authors compared their exploration framework with the classic frontier method [57], Rapid [58] and NBVP [27] in simulation. In a large-scale maze simulation FUEL achieved four times faster exploration on average due to the more efficient and smooth paths. However, this was at the cost of a higher computation time. The results show the advantages of planning on a global scale for large-scale environments.

5.5.6. REAL

The final autonomous exploration path planner that will be discussed is REAL, a recent work by E. Lee et al. [48]. Their architecture shown in Figure 5.6 includes both a global and a local exploration planner and active loop-closing to improve pose estimation accuracy in GPS-denied environments.

The local planner works with Peacock Trajectory [65]. Peacock Trajectory is a set of minimum-snap trajectories [34] contained in the sensor's FoV from the initial position with a determined length l_{traj}.

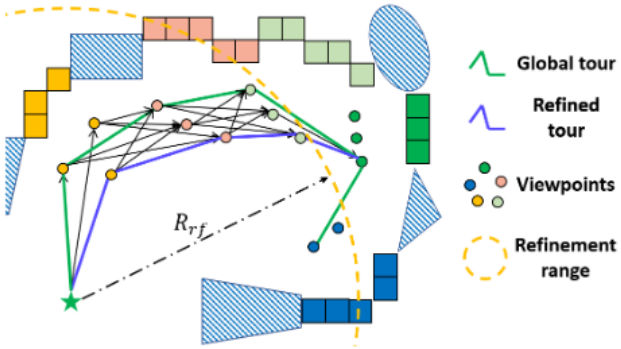


Figure 5.5: Representation of viewpoint refinement stage of FUEL path planning method [59].

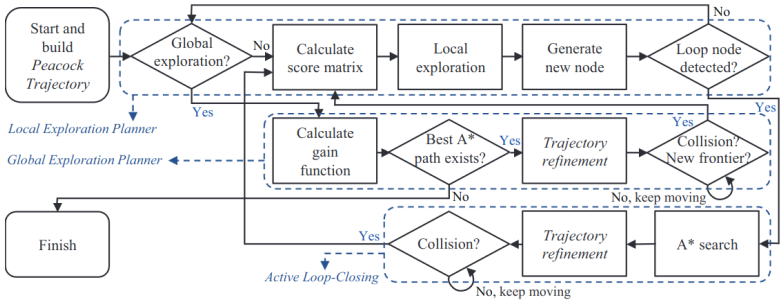


Figure 5.6: Architecture of REAL [48].

Constant velocity v_{max} is considered to enable faster exploration. The coefficients for Peacock Trajectory are computed at the beginning of the exploration process. For each local planning step, one of the trajectories is selected based on collision avoidance requirements and the amount of frontiers. Only the first part of the trajectory is actually implemented in a RH approach.

Regarding the global exploration planner it is used when local exploration struggles to find a path if a frontier is not visible or a collision cannot be avoided. A map of all frontier regions that is built incrementally from the local exploration progress is used. A* search is then applied to find the shortest collision-free path to an unexplored region. From the path, a minimum-snap trajectory is obtained. The whole planning process is illustrated in Figure 5.7.

The authors present comprehensive simulation results to benchmark their method and compare with state-of-the-art planners: NBVP [27], GBP [28], MPB [63] and AEP [61]. REAL achieved the best results in terms of average exploration times in both a small-scale and a large-scale environment. Sampling based approaches in NBVP and MPB resulted in less accurate movements so longer exploration times and got stuck in the large-scale scenario. GBP was able to cover the small-scale environment in lower time on average and complete the large-scale one in a few runs. This was attributed to the inclusion of a global planner. Among the tested planners, AEP was the fastest after real for the small-scale environment but was still slowed down by its high computational complexity. Yet it could not complete the exploration of the large-scale environment.

5.5.7. Considerations

Table 3.1 summarises the main characteristics, benefits and drawbacks of the exploration path planning algorithms that were presented.

Based on the methods and results present by the different planners, we conclude that exploration planners that take into account the kinodynamic constraints of the multirotor when sampling the configuration space achieve faster exploration speeds. It can also translate in higher computational efficiency.

Another important takeaway is that having a global graph improves exploration results in large-scale environments. It is also useful for return-to-home functionality or to keep a topological graph of important ”landmarks” which will likely be implemented in the thesis project.

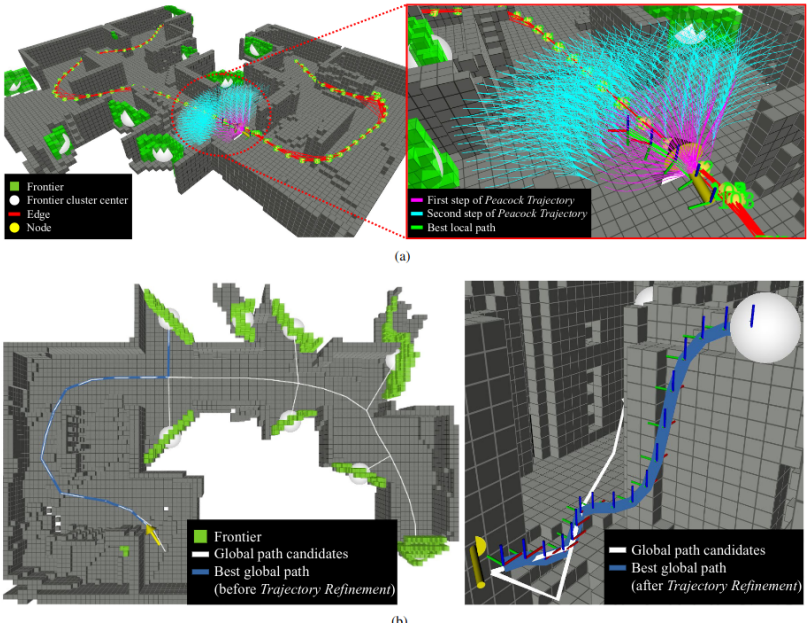


Figure 5.7: Representation of REAL path planning method [48].

Table 5.1: Comparison between the different path planning methods for autonomous exploration.

Planner	Architecture	Characteristics	Advantages	Limitations
NBVP	Local	- RRT to sample NBV - Volumetric gain function to choose best path	Lower computation times	Stuck in large-scale environments
AEP	Local + global	- NBVP as local planner - Frontier based global planner	Cached nodes to estimate gain	- Computational complexity - Kinodynamics not considered
GBP	Local + global	Sampling based with graph search	Field-tested	Kinodynamics not considered
MBP	Local	- Control space sampling to build path tree - Future-safe paths	-Kinodynamics considered -Field-tested	No global approach
FUEL	Hierarchical (3 stages)	- Multiple viewpoints considered	Exploits frontiers	Higher computation times
REAL	Local + global	- Peacock Trajectory - Loop closing	Trajectory optimisation	Not open source

6

Goal-oriented path planners

Goal-oriented planners plan safe paths in an unknown environment to reach a target position. To enable faster navigation, current research uses trajectory optimisation techniques that allow for higher flight speeds. A common approach is gradient-based trajectory optimisation that will be introduced below in section 6.1. Then section 6.2 will elaborate on the current state-of-the-art.

6.1. Gradient-based trajectory optimisation

As mentioned previously path planning algorithms provide a geometric safe path from which a trajectory can be obtained. The most common method is formulating the problem as a trajectory optimisation problem that minimises a cost function. Hard-constrained methods such as minimum-snap trajectory generation [34] set safety and kinodynamic constraints. On the other hand, the approach of soft-constrained methods is to penalise these constraints directly in the cost function [17].

Ratliff et al. were the first to propose gradient-based trajectory optimisation using ESDF gradient information with their method CHOMP [66]. By calculating the gradient descent the trajectory can be moved away from obstacles.

Regarding trajectory optimisation, CHOMP [66] obtains trajectories in discrete-time. H. Oleynikova et al. [67] adapted it to generate continuous time trajectories suitable for UAV applications using polynomial splines. V. Usenko et al. [68] propose the use of uniform B-splines that can accomplish faster optimisation. Due to their convex hull property and smoothness they require less variables and constraints in the optimisation problem.

6.1.1. B-splines parameterisation

Three-dimensional B-spline curves are guided by a set of control points. Uniform B-splines have a fixed time interval between consecutive control points. The curve does not pass through the control points. However, if the control points of a B-spline are contained inside a polyhedron so are all the points of the spline curve. This is the convex hull property and it is very useful for trajectory optimisation problems. Free space can be divided into polyhedra using convex decomposition. Then it is sufficient to check if the control points of a spline are contained in these polyhedrons to ensure that the whole trajectory is safe. Figure 6.1 illustrates this property.

Bézier Curves are specific cases of B-splines for which the polynomial degree is the number of control point minus one. Therefore, they still enjoy the convex hull property.

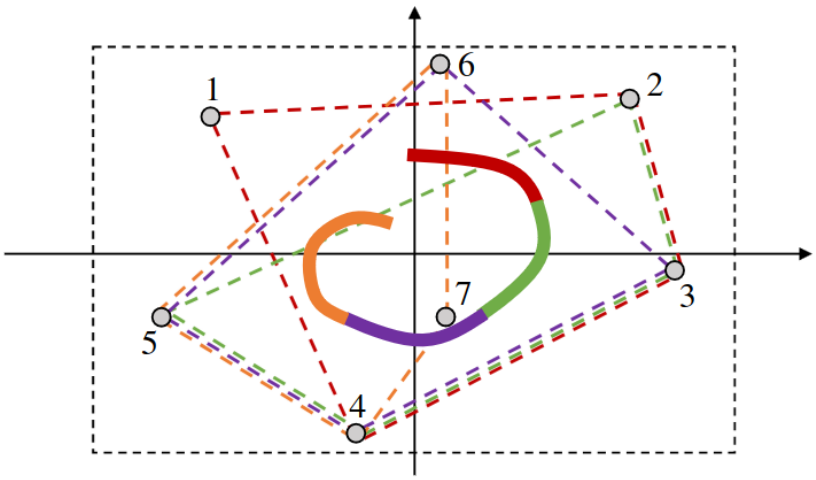


Figure 6.1: Representation of the convex hull property. It can be seen that the control points are marked in gray and the B-spline curve is clearly contained inside the dotted rectangle that can represent the free space for trajectory planning [54].

6.2. State-of-the-art planners

In this section three goal-oriented path planners will be discussed: FASTER [69], EGO-Planner [54] and Bubble Planner [49]. FASTER and EGO-Planner are widely cited and recognized implementations. Bubble Planner was proposed more recently with two novel strategies and achieving higher speeds. All of them were validated in real-world experiments.

This analysis is naturally limited due to the scope of the thesis project. The current state-of-the-art is more extensive and planners such as [70] are noteworthy, although its complexity doesn't make it useful for the purpose of the project.

6.2.1. FASTER

FASTER or Fast and Safe Trajectory Planner [69] implements a new strategy for achieving faster flight speeds. Their method combines a global and a local planner. The global planner uses JPS to find the shortest path to the goal in the uniform voxel map. Locally trajectories are planned in both free and unknown configuration space. Simultaneously a safe trajectory within free space is always available to be executed when obstacles are detected in the first trajectory.

Trajectories are represented by Bézier curves and obtained by solving a Mixed Integer Quadratic Program (MIQP). Time allocation is done heuristically considering the time interval for which a solution was obtained in the previous step.

6.2.2. EGO-Planner

EGO-Planner is a gradient-based method proposed by X. Zhou et al. [54]. The authors propose a novel algorithm to achieve lower computational complexity compared to ESDF-based methods.

Without using an ESDF map, collision-free trajectories are obtained by taking advantage of B-spline properties. Gradient information is obtained directly from obstacles and control points are penalized in the cost function if they are within a certain distance to an obstacle. In case a solution does not satisfy dynamic constraints, the authors propose re-allocation of the time length of the uniform B-spline followed by trajectory refinement to adjust smoothness and guarantee safety. The results for computational replanning time in simulation validate the ESDF-free approach with an average of $0.81ms$.

6.2.3. Bubble Planner

Bubble Planner is a recent work by Y. Ren et al. [49]. Their paper proposes a planning front-end that generates sphere-shaped flying corridors within free space and a back-end for trajectory optimisation based on the MINCO framework [71].

Compared to other methods that use safe flight corridors, they include a RH approach for generating the spheres which allows the quadrotor to continue following the first part of the previous trajectory thus promoting higher flight speeds. The sphere center points are sampled from a distribution around a guide point and a score function is computed to obtain the best sphere in terms of total volume and overlapped volume with the previous one.

However, reusing corridors from previous steps does not guarantee collision avoidance since most recent sensor data was not used. The authors suggest implementing a strategy similar to FASTER [69] to always have a back-up safe trajectory.

Bubble Planner was tested in several experiments in a forest. The highest speed achieved in autonomous navigation of a quadrotor in an unknown environment is reported for one of the flights, around $13.7m/s$. Compared to EGO-Planner, the computational times obtained are higher taking an average of $4.69ms$ in simulation and $13ms$ using the onboard setup for each planning step.

7

Conclusion

In this literature review the most important topics related to UAV path planning were covered. First, widely recognized path planning algorithms such as RRT and A* that are used as modules of recent multirotor path planning implementations were introduced. UAV path planners were then classified into four categories: goal-oriented planners, uncertainty-aware planners, exploration planners and collaborative robotic systems. In addition, state-of-the art exploration and goal-oriented planners were presented and compared.

The thesis project aims to research how path planning can be implemented to place an aerial sensor network in a rainforest environment taking into account the network architecture and the quadrotor's limited endurance. After extensive investigation we could not find a prior implementation of path planning for the placement of a sensor network in precise mapped deployment locations. We could also conclude that our proposed concept of an aerial sensor network is original. Having identified the research gap, we can recall our research question: **How to design an efficient path planner for an aerial sensor network, with one hundred sensor nodes, to map and navigate to precise deployment locations in an obstacle dense environment such as the rainforest using a quadrotor?**. To answer the research question we will approach the path planning problem in three layers, considering the state-of-the-art exploration and goal-oriented planners that were analysed.

The first step is to develop an exploration planner and include a global planner in its architecture. Similarly to the state-of-the-art, local exploration will be guided by volumetric gain estimations and computing safe paths (collision avoidance). In adding a global module, we can construct a topological graph incrementally to map ”good positions” for sensor placement. In the literature we found that having a global planner is more suited for large-scale exploration preventing the local planner from getting stuck. It also enables return-to-home functionality.

In a large-scale environment such as a rainforest, termination of exploration will likely be determined within the algorithm by a threshold on the number of positions of interest for deployment or when the drone's battery is running low. Also, in such an environment the drone cannot simply land in case of critical battery. This would require a specific landing sequence and the drone would be subject to unknown conditions until retrieval. It is therefore essential to be able to navigate back to the take-off position where the operator is hence the implementation of the global planning module with return-to-home feature.

Moreover, before implementation of the exploration planner it is necessary to define parameters for determination of positions of interest or ”good placement positions”. Since we propose to place the sensors above the canopy, vegetation density is the main factor. Other biological factors can be considered as well.

In the second planning layer the topological graph built during the exploration phase is used. Not all mapped locations for deployment will actually be suited for sensor node placement according to the network's connectivity constraints and the sensor range. Another relevant factor will be the distance that has to be travelled by the drone. Therefore, this stage will likely involve offline path planning to actual placement positions.

Regarding the third and final stage, the deployment drone runs a goal-oriented planner to navigate to the defined sensor placement positions. The planned deployment path in the previous planning phase is adapted online to avoid dynamic obstacles.

In conclusion, the developed path planner should propose a robust solution given the harsh conditions of the rainforest and the battery limitations of quadrotors. While validation in simulation will be important for the three planning stages, it remains one of the ambitions of this project to test the implementation in real-world conditions as representative as possible to the actual rainforest.

References

[1] SESAR. “European Drones Outlook Study”. In: (2016).

[2] S. Bibi D. C. Tsouros and P. G. Sarigiannidis. “A Review on UAV-Based Applications for Precision Agriculture”. In: *Information* 10 (2019), p. 349.

[3] J. Gu et al. “Multiple Moving Targets Surveillance Based on a Cooperative Network for Multi-UAV”. In: *IEEE Communications Magazine* 56.4 (2018), pp. 82–89.

[4] Y. Wan et al. “An Accurate UAV 3-D Path Planning Method for Disaster Emergency Response Based on an Improved Multiobjective Swarm Intelligence Algorithm”. In: *IEEE Transactions on Cybernetics* 53.4 (2023), pp. 2658–2671.

[5] S. A. R. Naqvi et al. “Drone-Aided Communication as a Key Enabler for 5G and Resilient Public Safety Networks”. In: *IEEE Communications Magazine* 56.1 (2018), pp. 36–42.

[6] J. A. Gonçalves and R. Henriques. “UAV photogrammetry for topographic monitoring of coastal areas”. In: *ISPRS Journal of Photogrammetry and Remote Sensing* 104 (2016), pp. 101–111.

[7] R. H. Fraser et al. “UAV photogrammetry for mapping vegetation in the low-Arctic”. In: *Artic Science* 2.3 (2016), pp. 79–102.

[8] J. Yick, B. Mukherjee, and D. Ghosal. “Wireless sensor network survey”. In: *Computer Networks* 52.12 (2008), pp. 2292–2330.

[9] Z. Sheng et al. “Wireless acoustic sensor networks and edge computing for rapid acoustic monitoring”. In: *IEEE/CAA Journal of Automatica Sinica* 6.1 (2019), pp. 64–74.

[10] H. J. Johnson. *Rainforest - National Geographic*. 2023. URL: `https://education.nationalgeographic.org/resource/rain-forest/` (visited on 06/12/2023).

[11] S. Hamaza et al. “Sensor Installation and Retrieval Operations Using an Unmanned Aerial Manipulator”. In: *IEEE Robotics and Automation Letters* 4.3 (2019), pp. 2793–2800.

[12] S. Hamaza et al. “Sensor Delivery in Forests with Aerial Robots: A New Paradigm for Environmental Monitoring”. In: *IEEE International Conference on Intelligent Robots and Systems* (2020).

[13] S. Kahl et al. “BirdNET: A deep learning solution for avian diversity monitoring”. In: *Ecological Informatics* 61 (2021), p. 101236.

[14] H. Cao et al. “An optimization method to improve the performance of unmanned aerial vehicle wireless sensor networks”. In: *International Journal of Distributed Sensor Networks* 13.4 (2017).

[15] S. Poudel and S. Moh. “Hybrid Path Planning for Efficient Data Collection in UAV-Aided WSNs for Emergency Applications”. In: *Sensors* 21.8 (2021), p. 2839.

[16] P. Corke et al. “Autonomous deployment and repair of a sensor network using an unmanned aerial vehicle”. In: *IEEE International Conference on Robotics and Automation* (2004).

[17] L. Quan, L. Han, and B. Zhou. “Survey of UAV motion planning”. In: *IET Cyber-systems and Robotics* 2.1 (2020), pp. 14–21.

[18] H. Choset et al. *Principles of robot motion: theory, algorithms and implementations*. The MIT press, 2005.

[19] L. Yang et al. “A literature review of UAV 3D path planning”. In: *Proceedings of the World Congress on Intelligent Control and Automation (WCICA)* (2015), pp. 2376–2381.

[20] S. Aggarwal and N. Kumar. “Path planning techniques for unmanned aerial vehicles: A review, solutions, and challenges”. In: *Computer Communications* 149 (2020), pp. 270–299.

[21] S. M. LaValle. *Planning algorithms*. Cambridge University Press, 2006.

[22] L. Kavraki et al. “Probabilistic Roadmaps for Path Planning in High-Dimensional Configuration Spaces”. In: *IEEE Transactions on Robotics and Automation* 12.4 (1996), pp. 566–580.

[23] S. Karaman and E. Frazzoli. “Sampling-based Algorithms for Optimal Motion Planning”. In: *The International Journal of Robotics Research* 30.7 (2011), pp. 846–894.

[24] S. M. LaValle. “Rapidly-exploring random trees: A new tool for path planning”. In: (1998).

[25] S. M. LaValle and J. J. Kuffner. “Rapidly-exploring random trees: Progress and Prospects”. In: *Algorithmic and Computational Robotics: New Directions* (2001), pp. 293–308.

[26] S. Karaman and E. Frazzoli. “Incremental Sampling-based Algorithms for Optimal Motion Planning”. In: *Conference: Robotics: Science and Systems 2010* (2010).

[27] A. Bircher et al. “Receding horizon next-best-view planner for 3D exploration”. In: *IEEE International Conference on Robotics and Automation* (2016).

[28] T. Dang et al. “Graph-based Path Planning for Autonomous Robotic Exploration in Subterranean Environments”. In: *IEEE International Conference on Intelligent Robots and Systems* (2019).

[29] E. Dijkstra. “A note on two problems in connexion with graphs”. In: *Numerische Mathematik* 1.1 (1959), pp. 269–271.

[30] T. H. Cormen et al. *Introduction to Algorihtms*. 4th ed. The MIT press, 2022.

[31] P. E. Hart, N. J. Nilsson, and B. Raphael. “A Formal Basis for the Heuristic Determination of Minimum Cost Paths”. In: *IEEE Transactions on Systems Science and Cybernetics* 4.3 (1968), pp. 100–107.

[32] R. Dechter and J. Pearl. “Generalized Best-First Search Strategies and the Optimality of A*”. In: *Journal of the Association for Computing Machinery* 32.3 (1985), pp. 505–536.

[33] D. D. Harabor and A. Grastien. “Online Graph Pruning for Pathfinding On Grid Maps”. In: *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence* (2011).

[34] D. Mellinger and V. Kumar. “Minimum snap trajectory generation and control for quadrotors”. In: *IEEE International Conference on Robotics and Automation* (2011).

[35] E. F. Camacho and C. Bordons. *Model Predictive Control*. 2nd ed. Springer London, 2007.

[36] C. Schumacher et al. “UAV Task Assignment with Timing Constraints via Mixed-Integer Linear Programming”. In: *AIAA 3rd ”Unmanned Unlimited” Technical Conference* (2004).

[37] M. Samadi and M. F. Othman. “Global Path Planning for Autonomous Mobile Robot Using Genetic Algorithm”. In: *2013 International Conference on Signal-Image Technology Internet-Based Systems* (2013).

[38] S. Konatowski and P. Pawłowski. “Ant colony optimization algorithm for UAV path planning”. In: *2018 14th International Conference on Advanced Trends in Radioelecrtronics, Telecommunications and Computer Engineering (TCSET)* (2018).

[39] S. Shao et al. “Efficient path planning for UAV formation via comprehensively improved particle swarm optimization”. In: *ISA Transactions* 97 (2020), pp. 415–430.

[40] L. Yang et al. “Survey of Robot 3D Path Planning Algorithms”. In: *Journal of Control Science and Engineering* (2016).

[41] C. Zammit and E. van Kampen. “Comparison between A* and RRT Algorithms for 3D UAV Path Planning”. In: *Unmanned Systems* 10.2 (2021), pp. 129–146.

[42] X. Zhou et al. “Swarm of micro flying robots in the wild”. In: *Science Robotics* 7.66 (2022), eabm5954.

[43] M. Kulkarni et al. “Autonomous Teamed Exploration of Subterranean Environments using Legged and Aerial Robots”. In: *IEEE International Conference on Robotics and Automation* (2022).

[44] M. Dharmadhikari et al. “Hypergame-based Adaptive Behavior Path Planning for Combined Exploration and Visual Search”. In: *IEEE International Conference on Robotics and Automation* (2021).

[45] S. Hayat et al. “Multi-objective UAV path planning for search and rescue”. In: *IEEE International Conference on Robotics and Automation* (2017).

[46] V. V. Estrela et al. *Imaging and Sensing for Unmanned Aircraft Systems*. The Institution of Engineering and Technology, 2020.

[47] R. Neuville, J. S. Bates, and F. Jonard. “Estimating Forest Structure from UAV-Mounted LiDAR Point Cloud Using Machine Learning”. In: *Remote Sensing* 13.2 (2021), p. 352.

[48] E. Lee et al. “REAL: Rapid Exploration with Active Loop-Closing toward Large-Scale 3D Mapping using UAVs”. In: *IEEE International Conference on Intelligent Robots and Systems* (2021).

[49] Y. Ren et al. “Bubble Planner: Planning High-speed Smooth Quadrotor Trajectories using Receding Corridors”. In: *IEEE International Conference on Intelligent Robots and Systems* (2022).

[50] A. Hornung et al. “OctoMap: an efficient probabilistic 3D mapping framework based on octrees”. In: *Autonomous Robots* 34.3 (2013), pp. 189–206.

[51] H. Oleynikova et al. “Voxblox: Incremental 3D Euclidean Signed Distance Fields for On-Board MAV Planning”. In: *IEEE International Conference on Intelligent Robots and Systems* (2017).

[52] L. Han et al. “FIESTA: Fast Incremental Euclidean Distance Fields for Online Motion Planning of Aerial Robots”. In: *IEEE International Conference on Intelligent Robots and Systems* (2019).

[53] H. Oleynikova et al. “Signed Distance Fields: A Natural Representation for Both Mapping and Planning”. In: *RSS 2016 Workshop: Geometry and Beyond - Representations, Physics, and Scene Understanding for Robotics* (2016).

[54] X. Zhou et al. “EGO-Planner: An ESDF-free Gradient-based Local Planner for Quadrotors”. In: *IEEE Robotics and Automation Letters* 6.2 (2021), pp. 478–485.

[55] R. A. Newcombe et al. “KinectFusion: Real-time dense surface mapping and tracking”. In: *2011 10th IEEE International Symposium on Mixed and Augmented Reality* (2011), pp. 127–136.

[56] Y. Pan et al. “Voxfield: Non-Projective Signed Distance Fields for Online Planning and 3D Reconstruction”. In: *IEEE International Conference on Intelligent Robots and Systems* (2022).

[57] B. Yamauchi. “A frontier-based approach for autonomous exploration”. In: *Proceedings 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation CIRA’97. ’Towards New Computational Principles for Robotics and Automation’* (1997), pp. 146–151.

[58] T. Cieslewski, E. Kaufmann, and D. Scaramuzza. “Rapid exploration with multi-rotors: A frontier selection method for high speed flight”. In: *IEEE International Conference on Intelligent Robots and Systems* (2017).

[59] B. Zhou et al. “FUEL: Fast UAV Exploration using Incremental Frontier Structure and Hierarchical Planning”. In: *IEEE Robotics and Automation Letters* 6.2 (2021), pp. 779–786.

[60] J. Irving Vasquez-Gomez et al. “Volumetric Next-best-view Planning for 3D Object Reconstruction with Positioning Error”. In: *International Journal of Advanced Robotic Systems* 11.10 (2014), p. 159.

[61] M. Selin et al. “Efficient autonomous exploration planning of large-scale 3-d environments”. In: *IEEE Robotics and Automation Letters* 4.2 (2019), pp. 1699–1706.

[62] DARPA. *DARPA Subterranean Challenge*. 2023. URL: `https://www.subtchallenge.com/` (visited on 06/14/2023).

[63] M. Dharmadhikari et al. “Motion Primitives-based Path Planning for Fast and Agile Exploration using Aerial Robots”. In: *IEEE International Conference on Robotics and Automation* (2020).

[64] B. Zhou et al. “Robust and efficient quadrotor trajectory generation for fast autonomous flight”. In: *IEEE Robotics and Automation Letters* 4.4 (2019), pp. 3529–3536.

[65] E. Lee et al. “Peacock Exploration: A Lightweight Exploration for UAV using Control-Efficient Trajectory”. In: *arXiv:2012.14649* (2019).

[66] N. Ratliff et al. “CHOMP: Gradient optimization techniques for efficient motion planning”. In: *IEEE International Conference on Robotics and Automation* (2009).

[67] H. Oleynikova et al. “Continuous-time trajectory optimization for online UAV replanning”. In: *IEEE International Conference on Intelligent Robots and Systems* (2016).

[68] V. Usenko et al. “Real-Time Trajectory Replanning for MAVs using Uniform B-splines and 3D Circular Buffer”. In: *IEEE International Conference on Intelligent Robots and Systems* (2017).

[69] J. Tordesillas, B. Lopez, and J. How. “FASTER: Fast and Safe Trajectory Planner for Flights in Unknown Environments”. In: *IEEE International Conference on Intelligent Robots and Systems* (2019).

[70] A. Loquercio et al. “Learning High-Speed Flight in the Wild”. In: *Science Robotics* 6.59 (2021), eabg5810.

[71] Z. Wang et al. “Geometrically Constrained Trajectory Optimization for Multicopters”. In: *IEEE Transactions on Robotics* 38.5 (2022), pp. 3259–3278.

[This page is intentionally left blank]

References

[1] Vitek Jirinec et al. “Morphological consequences of climate change for resident birds in intact Amazonian rainforest”. In: *Science Advances* 7.46 (2021), eabk1743. DOI: 10.1126/sciadv.abk1743. eprint: https://www.science.org/doi/pdf/10.1126/sciadv.abk1743. URL: https://www.science.org/doi/abs/10.1126/sciadv.abk1743.

[2] Çağan H. Şekercioğlu et al. “The effects of climate change on tropical birds”. In: *Biological Conservation* 148.1 (2012), pp. 1–18. DOI: https://doi.org/10.1016/j.biocon.2011.10.019. URL: https://www.sciencedirect.com/science/article/pii/S0006320711003880.

[3] S. Sharma et al. “A Morphing Quadrotor-Blimp with Balloon Failure Resilience for Mobile Ecological Sensing”. In: *IEEE Robotics and Automation Letters* 9.7 (2024), pp. 6408–6415.

[4] Tzu-Hao Lin et al. “Computing biodiversity change via a soundscape monitoring network”. In: *2017 Pacific Neighborhood Consortium Annual Conference and Joint Meetings (PNC)*. 2017, pp. 128–133. DOI: 10.23919/PNC.2017.8203533.

[5] Rachel T. Buxton et al. “Efficacy of extracting indices from large-scale acoustic recordings to monitor biodiversity”. In: *Conservation Biology* 32.5 (2018), pp. 1174–1184. DOI: https://doi.org/10.1111/cobi.13119. URL: https://conbio.onlinelibrary.wiley.com/doi/abs/10.1111/cobi.13119.

[6] C. Chalmers et al. “Modelling Animal Biodiversity Using Acoustic Monitoring and Deep Learning”. In: *2021 International Joint Conference on Neural Networks (IJCNN)*. 2021, pp. 1–7. DOI: 10.1109/IJCNN52387.2021.9534195.

[7] S. Poudel et al. “Hybrid Path Planning for Efficient Data Collection in UAV-Aided WSNs for Emergency Applications”. In: *Sensors* 21.8 (2021), p. 2839.

[8] P. Corke et al. “Autonomous deployment and repair of a sensor network using an unmanned aerial vehicle”. In: *IEEE International Conference on Robotics and Automation* (2004).