UNIVERSIDADE D
COIMBRA

Diogo José Matos Bispo Moreira Gomes

UAV NAVIGATION IN GNSS-DENIED ENVIRONMENTS

Dissertação no âmbito do Mestrado Integrado em Engenharia Electrotécnica e de Computadores, do ramo de especialização em Automação, orientada pelo Professor Doutor Lino José Forte Marques, apresentada ao Departamento de Engenharia Electrotécnica e de Computadores da Faculdade de Ciências e Tecnologia da Universidade de Coimbra.

Outubro de 2021

UNIVERSIDADE D
COIMBRA

Faculty of Science and Technology

Department of Electrical and Computer Engineering

Masters in Electrical and Computer Engineering

UAV Navigation in GNSS-denied Environments

Author

Diogo José Matos Bispo Moreira Gomes

2015263113

Supervisor

Prof. Dr. Lino Marques

Nothing can subtract hard work from success, only add to it.

Manjunath Harlapur

Abstract

Aerial robotics is a fast-growing field of robotics that is rapidly increasing in popularity. The ability of these robots to move in 3-D space brings new research challenges compared with, more common, wheeled mobile robots. Unmanned Aerial Vehicle (UAV) is being used for a wide range of indoor and outdoor applications. In outdoor common environments, these robots use Global Navigation Satellite System (GNSS) for localization. However, in indoor environments and in many closed urban spaces, this localization method is not available. In these cases, the navigation of a UAV is challenging, leading to the necessity for the drone to be self-sufficient in terms of sensors and on-board processing. Developing a drone with the ability to autonomously navigate in this type of environment between selected destinations safely and reliably while avoiding obstacles in its route, and with no prior knowledge of the environment, brings even more challenges. This dissertation attempts to take a step ahead towards this challenge, presenting a system that allows an UAV to explore autonomously in GNSS denied and unknown environments with multiple obstacles. The proposed solution integrates a novel approach for real-time pose estimation and mapping approach, which uses a 3-D Hough Transform to detect the ground floor plane. The developed system also has a high-level module capable of deciding the next destination based on certain criteria and of avoiding obstacles that might be on the way to that target location. The proposed approach was tested and validated in simulation and demonstrated with field experiments.

Keywords: UAV; GNSS-denied environments; Visual SLAM; 3-D Hough Transform; Robotic Exploration.

Resumo

A robótica aérea é um campo em rápido crescimento da robótica que está a aumentar rapidamente em termos de popularidade. A habilidade destes robôs em mover-se no espaço 3-D traz novos desafios de investigação em comparação com os robôs móveis com rodas. Em ambientes exteriores comuns, estes robôs utilizam sistemas de localização por satélite (GNSS) para se localizarem. No entanto, em ambientes interiores e em muitos espaços urbanos fechados, este método de localização não está disponível. Na ausência destes sinais ou de fontes externas de informação, a navegação de um UAV é um desafio, levando à necessidade do drone ser auto-suficiente em termos de sensores e de processamento a bordo. Desenvolver um sistema que permite um drone navegar de forma autónoma neste tipo de ambiente entre pontos seleccionados de forma segura e fiável, evitando obstáculos na sua rota, e sem qualquer conhecimento prévio do ambiente, traz ainda mais desafios. Esta dissertação tenta dar um passo em frente em direcção a este desafio, apresentando um sistema que permite a um UAV ter a capacidade de explorar autonomamente em ambientes desconhecidos, sem GNSS e com múltiplos obstáculos. A solução proposta integra uma abordagem em tempo real de mapeamento e de estimação da pose, que usa uma Transformada de Hough 3-D para detectar o plano do chão. O sistema desenvolvido tem também um módulo de alto nível capaz de decidir para onde o drone se deve deslocar com base em certos critérios e capaz de evitar obstáculos que possam estar no caminho para esse destino. A abordagem proposta foi testada e validada em simulações e demonstrada em experiências de campo.

Palavras-Chave: Veículo Aéreo Não Tripulado (UAV); Navegação em ambientes sem acesso a sistemas de localização por satélite (GNSS); Visual SLAM; Transformada de Hough 3-D; Exploração robótica.

Acknowledgments

I would like to thank all the people in my life who, in one way or another, have contributed to making this thesis possible.

I would like to thank my supervisor, Professor Lino Marques, for always guiding me in the right direction, for all the meetings we had to clarify all of my doubts, for and the knowledge transmitted, and for all the crucial recommendations.

A special thanks to Hugo Magalhães and Rui Baptista for all the knowledge transmitted, for all the doubts clarified, for all the suggestions, and for always encouraging me when things were not going well, making my mind positive. I am truly honored to have you as my lab colleagues and, especially, to have you as my friends.

I would like to thank Sedat Dogru and João Macedo for all the essential suggestions and ideas to improve my work. I am truly honored to share the lab with you.

A special thanks to my family and my girlfriend, Adriana Fernandes, who are the most important people to me, who never stopped encouraging me, always supported me, and believed in the pursuit of my dreams. All of this would not be possible without you.

Finally, to all my friends, thank you all for all your companionship and friendship, in particular, Carlos, Zé, and Cristiana, who were always with me whenever I needed them.

Contents

Acronyms

2-D Two Dimensions.

2.5-D Two and a Half Dimensions.

3-D Three Dimensions.

BFS Breadth-first search.

BRIEF Binary Robust Independent Elementary Features.

CAN Controller Area Network.

CPU Central Process Unit.

DARPA Defense Advanced Research Projects Agency.

DoF Degrees Of Freedom.

EKF Extended Kalman filter.

F2M Frame to Map.

FAST Features from Accelerated Segment Test.

FoV Field of view.

FPS frames per second.

GFTT Good Features To Track.

GNSS Global Navigation Satellite System.

GPS Global Positioning System.

GPU Graphics Processing Unit.

IMU Inertial Measurement Unit.

LiDAR Light Detection and Ranging.

LTM Long-Term Memory.

MAV Micro Aerial Vehicles.

MAVLink Micro Air Vehicle Link.

MAVROS MAVLink on ROS.

ORB Oriented FAST and rotated BRIEF.

PID Proportional–Integral–Derivative.

PNP Perspective-n-Point.

RGB-D Red Green Blue-Depth.

RMSE Root Mean Square Error.

ROS Robot Operating System.

RTAB-Map Real-Time Appearance-Based Mapping.

RTL Return To Launch.

Rviz ROS visualization.

SAR Search And Rescue.

SITL Software In The Loop.

SLAM Simultaneous localization and mapping.

STM Short-Term Memory.

TF Transformation.

TF-IDF Term Frequency-Inverse Document Frequency.

UART universal asynchronous receiver-transmitter.

UAV Unmanned Aerial Vehicle.

USB Universal Serial Bus.

VFF Virtual Force Field.

WFD Wavefront Frontier Detector.

WM Working Memory.

List of Figures

List of Tables

1 *Introduction*

The great mobility provided by this robots make then more suitable to applications such as rescue, disaster assessment, plant engineering, or other tasks that would be risky or impossible for a human to perform, then the wheeled mobile robots. In most of this applications there is no information about the environment and there is no acess to GNSS signals or other external sources of information. Therefore, it is necessary for the robot collect information about the environment, build a representation of this environment and localize itself within this representation. This task is called Simultaneous localization and mapping (SLAM). This dissertation aims to study, develop and validate a real-time method of SLAM that enables a UAV to navigate and explore unknown environments autonomously without the use of GNSS signals or other external sources of information. Light Detection and Ranging (LiDAR) and cameras are the two most commonly used sensors for this task. The technique for solving this task is based on detecting and calculating visual features in the environment and calculating the robot's pose by the movement of these. This pose is complemented with information from the ground plane, which is considered a robust and valid "feature" for the entire exploration environment. This feature is obtain through a LiDAR pointing down. By knowing the ground floor plane, the drone's height can be estimated based on the LiDAR measurements that corresponds to the ground floor plane. The navigation and exploration approach entails employing existing Pixhawk features when it has access to Global Positioning System (GPS) receiver location and substituting this information with the positioning provided by the slam strategy.

1.1 Unmanned Aerial Vehicles

This subsection 1.1 provides an overview of the history of Unmanned Aerial Vehicle (UAV). Following are explained the various applications of drones that suit the subject of this dissertation. Subsection 1.3 describes UAVs' different challenges, e. g. the lack of access to Global Navigation Satellite System (GNSS) signals. Subsection 1.4 defines the dissertation's principal goal, and the secondary objectives that compose the first objective are listed. Next, subsection 1.5 details all the main contributions of this dissertation.

Over the last decade, improvements in the field of mobile robotics have been expressly pronounced. Nowadays, robotic platforms are replacing traditional human tasks. Robots are also becoming a part of our daily routines, whether personal, business, military, or other uses. Furthermore, mobile platforms will soon become omnipresent in crowded spaces, coping better with dynamic obstacles and unstructured environments. One of the mobile platforms that fit this level is the aerial platform, specifically the UAV. An Unmanned Aerial Vehicle (UAV) (commonly known as a drone) is an aircraft without a human pilot on board. The term can refer to many types of flying vehicles, however throughout this document refers to a quadrotor.

Even though small radio-controlled quadrotors started to appear in 2000, the multi-rotor helicopter's idea is not new; it is about 90 years old. After several efforts and failures, probably the first quadrotor that was

able to fly was constructed by *Jerome de Bothezat*. In addition to four main rotors, four additional small propellers helped to control the quadrotor. In 1922, the "Flying Octopus" flew, although at low altitudes and slow forward speeds, represented in figure 1. It was not very stable, and as a result of the military's poor performance and changing interest in other projects, this project was canceled.

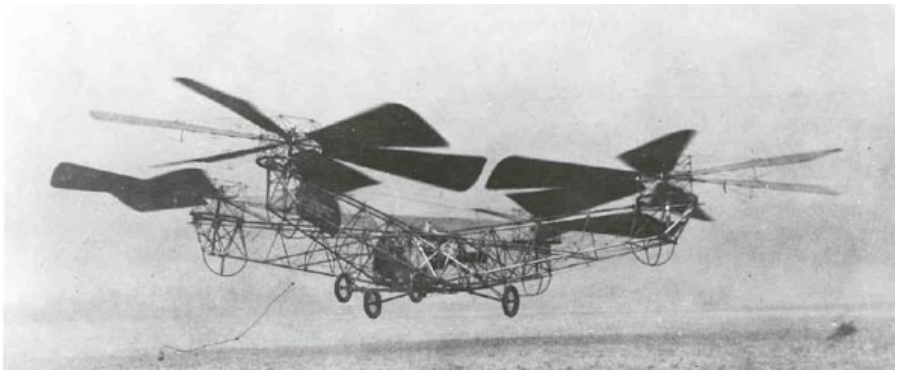


Figure 1: The first flying multi-rotor helicopter (Flying Octopus).[1]

In the past couple of decades, was an exponential growth development of technology, particularly in the development of highly efficient batteries and electronically commutated electric motors, combined with the miniaturization of sensors, and computing devices, which were the key to the success of UAV by triggering the interest of the research community. Nowadays, many companies are developing drones for the most varied applications, as Parrot and DJI, which have become fundamental for intermediate pilots and videographers by packing high-quality drone features into tiny drones. Yuneec, in 2018, launched the first-ever voice-controlled drone, the *Mantis-Q*. Kespry manufactures drones explicitly designed for capturing and analyzing aerial imagery, illustrated in the figure 2. Their customers include mining, construction, and surveying companies.

1.2 Motivation

Small-scale autonomous aerial vehicles will play a significant role soon [1]. UAVs are achieving more attention from researchers due to the numerous advantages over ground vehicles.

One of the main advantages of using a UAV is its mechanical simplicity, high maneuverability, and low cost [2], illustrated in figure 3a. Drones can fly and land vertically, hover in almost any location, and even dock to the surface. This capability allows them to efficiently work in small indoor environments, pass through windows, traverse narrow corridors, and even grasp small objects [3]. Furthermore, in buildings are multiple floors with stairwells, where aerial vehicles offer mobility and perspective advantages over ground platforms; therefore, making them the ideal vehicle type for tasks performed in this type of environment [4]. Moreover, there is a

[1] `https://en.wikipedia.org/wiki/De_Bothezat_helicopter`

[3] `https://dronelife.com/2017/01/26/kespry-announces-drone-2/`

[3] `https://www.dronebydrone.com/en/news/405/drone-dji-mavic-2-pro-with-hasselblad-camera.html`

(a) **Kespry** manufactures drones for capturing, and analyzing aerial imagery.[2]

(b) **DJI** is one of the most popular companies for videographers.[3]

Figure 2: Commercial drones for aerial imagery.

(a) UAVs in Search And Rescue (SAR) missions in tunnels.[4]

growing need for autonomous robots in applications such as rescue, disaster assessment, plant engineering, or other tasks that would be risky or impossible for a human to perform [5]. Aerial platforms allow us to easily access environments to which no humans or other ground vehicles can't get access. For the above applications, the UAV must have the capacity to operate in unstructured, unexplored, and GNSS-denied environments. These vast advantages make the Unmanned Aerial Vehicle an upcoming technology with a high purpose to modernize military applications and facilitate innovative civilian applications [6].

There are several scenarios where there is no access to GNSS, and the drone has to perform a mission in an unknown environment. These scenarios are very challenging, as they are highly complex and unpredictable and

[4] https://www.researchgate.net/publication/329492926_The_Foldable_Drone_A_Morphing_Quadrotor_That_Can_Squeeze_and_Fly/figures?lo=1

(a) UAVs in Search And Rescue (SAR) missions to search for people in remote areas.[5]

require accurate estimation of the UAV's location to execute various tasks. This category of scenarios is suited to high-level applications like military applications, e. g., spying on enemy territory [7]. Relying exclusively on the Global Positioning System (GPS) for positioning might also pose safety issues in signal jamming or platform hijacking, which is a severe problem, especially in military applications. Therefore, it is a strong motivation to achieve reliable autonomous systems that operate securely without the GPS signal [8].

Forestry applications are part of the wide range of applications that operate outside the reach of GNSS signals, e. g. forest fire monitoring, and agriculture information gathering [6]. River applications, such as mapping, monitoring, and surveillance, collect valuable information to understand an environment's topology and health. Using a UAV to navigate under the foliage of a forest [9] is advantageous since this information is often not collected from satellite imagery because tree canopy cover occludes the river from above. Furthermore, densely forested rivers are difficult to navigate by surface craft because of submerged and semisubmerged obstacles [10]. Search And Rescue (SAR) missions are a significant focus of attention for researchers [11]. One of the main challenges of the Defense Advanced Research Projects Agency (DARPA) (a high-stakes competition launched by the U.S.) is the Subterranean Challenge (exploration of caves). There are no prior maps of the environment in the disaster relief case, as the space structure may change entirely. Furthermore, this scenario is perilous for firefighters because of the unstable structure of the building, and sending a ground robot for these environments is not the most suitable option since they are often unable to overcome rubble. Therefore, a UAV is a more suited robot for this scenario. With the capability of navigating safely in a remote region and generally with no GNSS signal available, they could perform more follow-up inspections to find missing persons, deliver small supplies, or explore more of the disaster area, becoming an asset for these scenarios [12], illustrated in figure 4a.

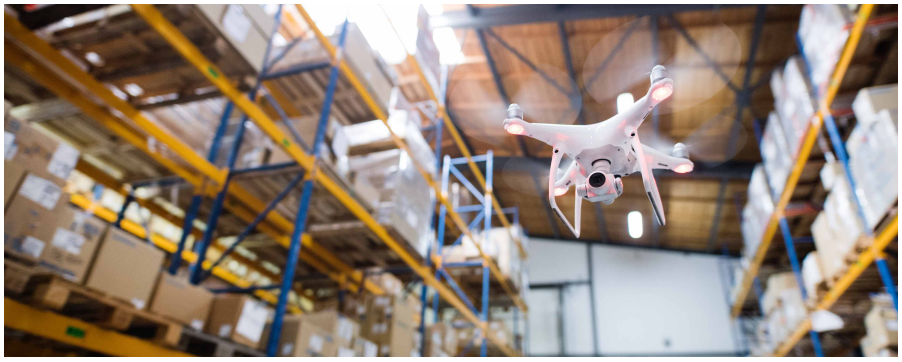


Figure 5: Drone operating in a warehouse.[8]

One promising application of UAVs for civilian missions is to improve the traffic monitoring systems deployed. The lack of traffic monitoring has become a primary weakness in providing prompt emergency services. UAV s have great potentials for providing quick and real-time aerial video images of large surface areas to the ground, both in the countryside and in the city. UAVs afford a cost-effective means to satisfy the need for a rural traffic surveillance system for this aspect [13]. To achieve this application with success is essential to develop an autonomous UAV that can operate in GNSS-denied environments due to signal blocking by skyscrapers. If the autonomous system relies solely on GPS signals for location information, a drone can collide with a building, as in the 13 December 2019 incident[6] in the United Kingdom. Moreover, a recent application related to COVID-19 pandemics uses drones to monitor body temperatures and recommended physical distance between people in the cities.

In the business context, UAVs have proven to be a significant asset in warehouse operations. Inventory management traditionally involves manual barcode scans to count the warehouse inventory and keep track of stock. Besides being an insecure and arduous task, manual counts often result in inventory data not being up to date due to disparities. The drone system can scan barcodes without human interaction and without interrupting warehouse operations, illustrated in figure 5. An example of this is that, more recently, IKEA and Verity performed successful pilot tests[7] for an automated drone solution for warehouse inventory checks, potentially reducing the many working hours required to check inventory manually.

In recent years, many researchers have focused on using a UAV to examine bridges in order to save time and money. According to *Bolourian and Hammad* [14], inspecting the second-largest bridge in Minnesota with traditional methods costs approximately $59000 taking eight days, while using a UAV would amount to around $20000 in five days. A UAV can also improve safety, accuracy, and efficiency for examination bridges, illustrated in figure 6. This concept can also be implemented on vessels by using UAVs to inspect them to

[5] https://www.theverge.com/2018/1/18/16904802/drone-rescue-australia-video-ocean
[6] https://www.uasvision.com/2020/08/12/uk-drone-crash-due-to-gps-interference/
[7] https://ikea.today/how-tech-for-show-business-can-automate-ikea-warehouses/
[8] https://www.cargo-partner.com/trendletter/issue-4/drones-in-warehouse-logistics

Figure 6: Bridge inspections performed by UAVs cost much less money and are much safer for humans.[9]

make ship inspections safer and more cost-efficient. Seagoing ships have to undergo periodic checkups, which are currently performed manually by ship surveyors, where the main cost of the operation is access to all areas. Therefore UAVs are valuable assets for these applications by reaching all vessel areas, including areas with no GNSS signal [15]. Following the same idea, an important application is the inspection of large infrastructures such as dams and penstocks. These structures are exposed to high oscillating loads for long periods, due to which continuous maintenance is vital to avoid catastrophic consequences such as the dam's collapse or water discharge tunnels. Moreover, current inspection and maintenance practices are carried manually by workers either by swinging from the scaffolds [16]. Therefore, drones' use to perform this dangerous task is one of the researchers' primary investigation focus.

For all the applications mentioned above, there is an enormous necessity to achieve autonomous UAVs that can operate safely and with precision in GPS-denied environments [1] and, that is the aim of this work.

1.3 Challenges of aerial vehicles

For all robotic platforms, there is a general navigation problem. This problem is broken down into three main questions that are connected and illustrated in figure 7, regardless of the robot's environment:

- Where am I?
- Where am I going?
- How do I get there?

The first question is directly related to the drone's location concerning the environment. Unlike ground vehicles, air vehicles cannot measure odometry directly, which most SLAM algorithms require to initialize

[9] `https://www.aerovfinancial.com/civil-infrastructure/`

Figure 7: Navigation problem in robotics.

or complement their estimation of the vehicle's motion between time steps. Even though double-integrating accelerations can obtain odometry, Inertial Measurement Unit (IMU) often provide high noisy measurements [17], thus not being the best option. The most common method is receiving signals from GNSS and performing the calculation to acquire the UAV's position. However, indoor environments and many parts of closed urban spaces remain without access to these signals. In the absence of GNSS or external sources of information and unreliable communications, the navigation of a UAV is challenging since it depends entirely on onboard IMUs [6], cameras, and scanning lasers, leading to the necessity for the drone to be self-sufficient without off-board processing [12]. To resolve this problem, SLAM algorithms build a map of the vehicle's surroundings from onboard sensors while simultaneously using it to estimate its position [18], since the UAV will operate in an unknown environment, not being aware of the map a priori.

Once the drone has located itself in the environment, the second question arises, directly related to a decision-making block. Based on observations of the environment in which it is inserted, it is necessary to decide where to go based on the best satisfaction of the proposed objectives. This decision-making capability can produce some heavy computational calculations, which can be challenging due to the drone's limited payload. UAVs must generate sufficient vertical thrust to remain airborne, limiting the available payload. Due to this weight constraint, it is necessary to rely on lightweight laser scanners, cameras, lower-quality MEMS-based IMUs, and single-board computers. Generally, Red Green Blue-Depth (RGB-D) cameras and range finders provide data with high resolution and at high sampling rates. Processing such information is computationally expensive and requires massive computing resources [5], thus allocating most of the drone's resources to process this data. Furthermore, UAV platforms' control systems are usually embedded devices with limited resources, low performance, and little memory. Consequently, it is a challenge to achieve a decision-making capability in a UAV.

Finally, after deciding the goal for the drone to reach, it is necessary to know how to get there. The unpredictability of an unknown environment increases the difficulty in reaching the chosen target due to the possibility of existing static and dynamic obstacles. For this, it is necessary to implement reactive approaches

capable of dealing with possible objects that have not been mapped.

In summary, we must address the problems of mapping, localization, planning, and control, given these system requirements [4], to develop a complete system capable of autonomous navigation and exploration in unknown GNSS-denied environments [17].

1.4 Objectives

The work's main objective is to develop a real-time method that enables a UAV to navigate and explore unknown environments autonomously without the use of GNSS signals or other external sources of information. It will be necessary to complete the following secondary objectives to achieve the principal purpose of this dissertation:

- To study and understand the operation of the various technologies needed for implementing the work, namely the Pixhawk motion controller for quadrotor platforms, a scanning LiDAR, vision cameras, inertial systems, and different types of single-board computers.
- Develop and implement an algorithm that is constantly estimating the UAV's pose and motion in real-time.
- Develop and implement a mapping method for accurate environment representation allowing the creation of maps of unknown environments.
- Develop and implement a strategy to explore the environment to provide the capability of navigating in unknown environments.
- Develop and implement an algorithm for local path planning with a reactive module to avoid unmapped obstacles.
- Test and evaluate the performance of the implemented algorithms.

1.5 Contributions

To the best of the author's knowledge, this dissertation presents the first work to develop a 3-D Hough Transform to detect the ground floor plane with UAVs, being, for this reason, the most relevant contribution of this dissertation. This contribution is directly related to estimating the vehicle's altitude, which involves determining the global height in relation to a fixed ground. By calculating the ground floor plane, it is meant to estimate the height of the drone with respect to this ground floor. The developed system allows an UAV to explore autonomously in GNSS denied environments with multiple obstacles. Considerable experimental validation of the proposed solution has been conducted and evaluated on a realistic simulator and real hardware, conceding an extensive analysis and discussion of the developed work.

2 *UAV Navigation in GNSS-denied Environments*

The research towards UAV navigation in GNSS-denied environments has been increasingly active over the past decade, resulting in significant progress in the field. When operating in unknown environments, this area of research is directly related to Simultaneous localization and mapping (SLAM) applied to UAVs, using exteroceptive sensor(s) used to obtain information about the vehicle's state regarding the environment. This section overviews the achievements directly related to UAV navigation in GNSS-denied environments while pointing to the detailed related work and state of the art in related fields.

2.1 LiDAR-based Navigation

Several research groups have worked on 2-D laser range finder-based navigation, mapping exploration, and control. One of the first groups to use a laser range finder in UAVs for position control loop was the group of *Nicholas Roy* [19], who also presented a planning approach that considers the localization limitations of the range of the sensor. This work was constrained by that time's technology and the helicopter's non-detailed model. Not much later, the same group completed a complex indoor navigation scenario, relying only on sensors onboard the vehicle mainly by laser-based navigation, aided by [20]. They managed to incorporate stereo vision and laser odometry in a quadcopter, thus achieving a stable platform capable of fully autonomous exploration in unstructured and unknown GNSS-denied environments, relying solely on sensors onboard the vehicle [21]. The same incorporation occurs in the work of [11], who presented a brilliant work combining visual or laser odometry to perform an autonomous flight from an indoor to an outdoor environment through a 1m-wide window, driven by an exploration mission to enter and exit a building. Since low illumination conditions and a lack of environmental features make indoor situations unsuitable for a visual odometry system, laser odometry was used. On the other hand, in outdoor environments, sunlight interferes with measurements. Therefore, visual odometry was used. Because the UAV was transitioning between two scenarios, it was required to change the odometry source, either visual or laser. The shift is made by measuring the magnitude of laser and visual odometry covariances. The switch is triggered when the current odometry source has a higher covariance than the one not in use. Shen [4] presented a system design and strategy that enables autonomous navigation with real-time performance on a 1.6 GHz Atom processor in an indoor two-floor environment utilizing just onboard sensors. The computation occurs on the robot without the requirement for external infrastructure, communication, or human intervention. The implemented system uses a Hokuyo UTM-30LX to estimate the drone's position through the Iterative Closest Point algorithm. When a floor transition is identified, a new layer in the multi-layered occupancy grid is created, allowing the UAV to be located on multiple floors. Grzonka [22] presented a navigation system that enables a small-sized quadrotor system to autonomously operate in indoor environments by employing a 2-D laser range finder on a UAV; however, it relies on the 2.5-D assumption of the environment to work correctly. Another constraint of this system is that only a fraction of the software runs

onboard because of some relatively high computational cost; the other runs off-board on a laptop computer.

2.2 Vision-based Navigation

LiDARs are highly accurate and provide data at high rates, however they were too heavy and use too much power for being used in UAVs. As a consequence, vision sensors seemed quite appealing. However, cameras require external light and a lot of computing power to derive meaningful navigational data. Celik [23] presented a real-time vision navigation method for SLAM using monocular vision, assuming a GNSS denied unknown environment. Applies an approach for gathering useful landmarks from a monocular camera for SLAM application through corner-based feature points. This system is limited by the camera's capabilities, the availability of good corners, and the computational power, which affects the real-time quality of the algorithms involved. The same author [24] successfully adapted the system to various situations, such as changing the drone's height and navigating an environment with obstacles. One of the most significant advantages of the monocular camera vision SLAM approach is that it does not require initialization procedures, having the ability to start at an arbitrary point. Forster [25] applied a semi-direct approach that eliminates the need for costly feature extraction and robust matching techniques for motion estimation by operating directly on pixel intensities, resulting in subpixel accuracy at high frame rates. This method leads to a precise and high frame-rate motion estimation with increased robustness in repetitive scenes and high-frequency textures.

Cameras other than monocular cameras have been developed as technology has progressed. RGB-D, stereo, and event cameras become more incorporated into UAVs as the interest in UAV navigation in GNSS-denied areas grows. The works of [26] and [27] were among the first to performed Autonomous Flights using only an RGB-D camera in GNSS-denied Environments. Their method involves detecting and matching features between consecutive frames and estimate their motion to determine vehicle position. Their system can plan complex 3-D paths in cluttered environments while retaining a high degree of situational awareness. Mohta [2] presented a paper intending to navigate target locations autonomously, quickly, and reliably while avoiding obstacles in its path and, with no prior knowledge of the operating environment, and with no access to GNSS signals. This work was limited in two main aspects, the drift in visual odometry and not using all Pixhawk characteristics. Due to low depth estimates of some features, the visual odometry could not detect the correct scale of the motion between frames, which led to a drift in the estimates and consequently caused a failure to reach the goal in some cases. Similar to [28], the UAV's orientation and angular velocities are controlled by an inner-loop, while its position and linear velocities are controlled by an outer-loop. The Pixhawk controls the inner loop, while the Intel NUC computer controls the outer loop. The position controller receives the desired position, velocity, acceleration and computes the desired force, orientation, and angular velocities which, are sent to the orientation controller. The attitude controller running on the Pixhawk accepts these commands and converts them into motor speeds. This system's architecture is a disadvantage because it does not take

advantage of all of Pixhawk's features since it can control the UAV's six degrees of freedom with the correct configuration. It's also worth noting that event cameras are becoming increasingly popular and promising in the UAV area. A very specific case is the outstanding work of Davide Scaramuzza on 6-DoF pose tracking and with trajectory planning of a UAV. The first onboard perception system for 6-DoF localization during high-speed maneuvers employing an event camera was created by [29]. An event camera only transmits pixel-level brightness changes at the instant they occur with microsecond resolution, thus, allowing for the creation of a perception pipeline with minimal latency relative to the robot's dynamics. This work exhibited reliable motion tracking during quadrotor flips with angular speeds up to 1200 ^{o}/s. Davide Scaramuzza [30] also proposed a method in 2020 with an overall latency of only 3.5 milliseconds, which is enough for the UAV to avoid several objects of various sizes and shapes at relative speeds of up to 10 meters per second, both indoors and outdoors. This was accomplished by using the temporal information in the event stream to distinguish between static and dynamic objects to avoid oncoming obstacles. These two works, in particular, yielded remarkable results, resulting not only in a massive advancement in the field of computer vision but also in the navigation of UAVs in GNSS-denied environments.

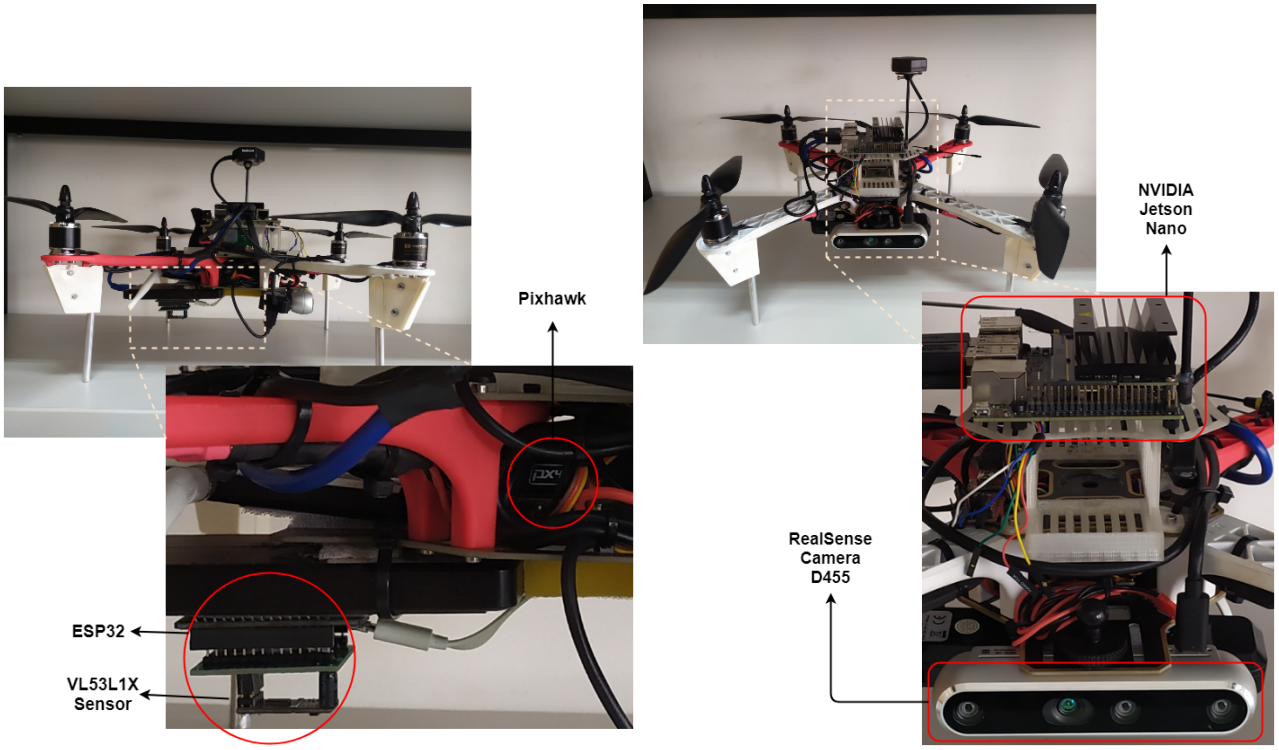
3 *Materials and Methods*

This section overviews the system design regarding the hardware elements utilized to build the drone and overviews the software methods by explaining the frameworks and the software packages used.

3.1 UAV

This subsection is divided into two parts. The first part describes the system design, where is explained the approach taken to build the drone and the choice of the hardware used. The second part describes the roles performed by the different components and how they are integrated into the system.

Extensive survey of many depth sensors was made to choose the component that provides the drone's location. Several factors were taken into account (price, weight, dimensions, among others) to know which is the most advantageous in the system integration. It was chosen the RealSense D455 Camera which, is cheap, has a longe range and provides data up to 90 frames per second (FPS). Ending up building the drone represented in the Figure 8 that depicts all the components that were chosen to integrate the system.



(a) Side of the drone.

(b) Front of the drone.

Figure 8: Drone used to perform the field experiments.

The **Pixhawk** (depicted in Figure 9) is a Professional Autopilot which can control all kind of vehicles,

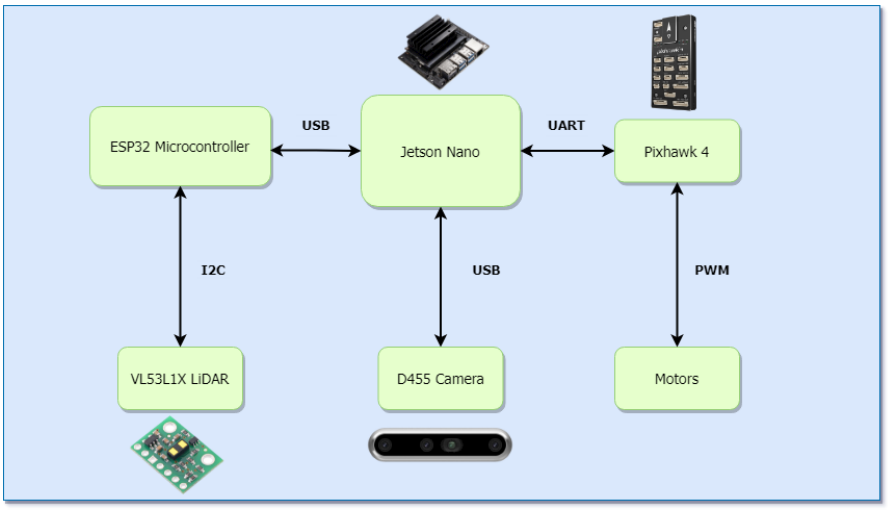


Figure 9: Schematic of the hardware.

Table 1: NVIDIA Jetson Nano's features.

	NVIDIA Jetson Nano
Central Process Unit (CPU)	64-bit Quad-core ARM A57 @ 1.43GHz
Graphics Processing Unit (GPU)	128-core NVIDIA Maxwell @ 921MHz
Memory	4GB 64-bit LPDDR4 @ 1600MHz | 25.6 GB/s
USB	4x USB 3.0
Storage	MicroSD card
I/O	(3x) I2C | (2x) SPI | UART

from racing and cargo drones to ground vehicles and submersibles. It uses an STM32F765 microcontroller, two IMUs, a barometer, a magnetometer, a microSD card slot, six universal asynchronous receiver-transmitter (UART) plus USB, four I2C ports, and two Controller Area Network (CAN) ports. Pixhawk's responsibility is to control and stabilize the drone. The Pixhawk utilizes an Extended Kalman filter to estimate several state variables such as position, velocity, and angular orientation, and these sensors are essential for this task. Moreover, it uses various Proportional–Integral–Derivative (PID) controllers to control the linear and angular velocities and the correct power needed to guide the drone. All computing is done onboard, so it is necessary to have a small computer on the drone. The **Jetson nano**, represented in figure 9, will perform this task, having enough capacity to run all the algorithms onboard without interruptions. Table 1 shows the Jetson Nano's characteristics that make it a small computer capable of having enough memory and processing speed to run heavy and complex algorithms in real-time.

The camera is a fundamental part of the system since it is the principal source of the drone's location. Among the several depth cameras investigated and represented in table 7, the **Intel RealSense Depth**

Camera D455 was chosen for its low price, small size, small weight, and for providing RGB-D data with high resolution. The **STMicroelectronics VL53L1X Time-of-Flight Distance Sensor**, represented in figure 9, will provide measurements pointing downwards, contributing to a better perception of what is underneath the drone and higher confidence in the drone's height. The VL53L1X laser-ranging sensor was chosen for its low price, tiny size, small weight, and for offering fast and accurate ranging up to 4 m and fast ranging frequency up to 50Hz. In the Figure 9 depicts the schematic of the hardware of the drone of the Figure 8. The Jetson Nano powers the ESP32 Microcontroller and the camera and consequently receives data from the Light Detection and Ranging (LiDAR) and camera for processing. Then the Jetson Nano communicates with the Pixhawk via UART, sending, for example, waypoints for the drone to reach.

3.2 Software

This subsection describes the frameworks and the software packages used to assemble the overall system.

3.2.1 Frameworks

The proposed and implemented system consists of six ROS nodes, which perform different functions, complementing each other. All work was developed in C++ and python languages. The entire system was developed in a **ROS** environment because this framework carries several advantages to the development of robotics works. The core base of ROS doesn't necessitate enormous space and resources. therefore can also be used on embedded computers. The ROS framework is used for the development of drones due to an especial ROS 'node', the **MAVLink on ROS (MAVROS)**, that can convert between ROS topics and Micro Air Vehicle Link (MAVLink) messages allowing ArduPilot vehicles to communicate with ROS.

Another advantage is that ROS is open-source and has great simulation and visualization tools, such as **Gazebo** and **Rviz**. Working with a drone can cause problems in carrying out indoor experiments, namely due to its high instability, speed of execution, and the unpredictability of the code implemented on the drone. With Gazebo, the same indoor experiments can be performed in simulation with similar outcomes as the real drone flying in a real environment.

Another main framework of this work is the **ArduPilot**, a open source autopilot software capable of controlling autonomous drones. The **Software In The Loop (SITL)** is used to run the ArduPilot in simulation together with Gazebo, without any special hardware.

The frameworks **Mission Planner** and **QGroundControl** were used to download mission log files and analyze the results of the indoor experiments. These two frameworks can configure the vehicle's settings and select mission commands, providing full flight control and mission planning for almost any drone.

The ESP32 uses **FreeRTOS** to process and send synchronized data from the sensors to the UART node.

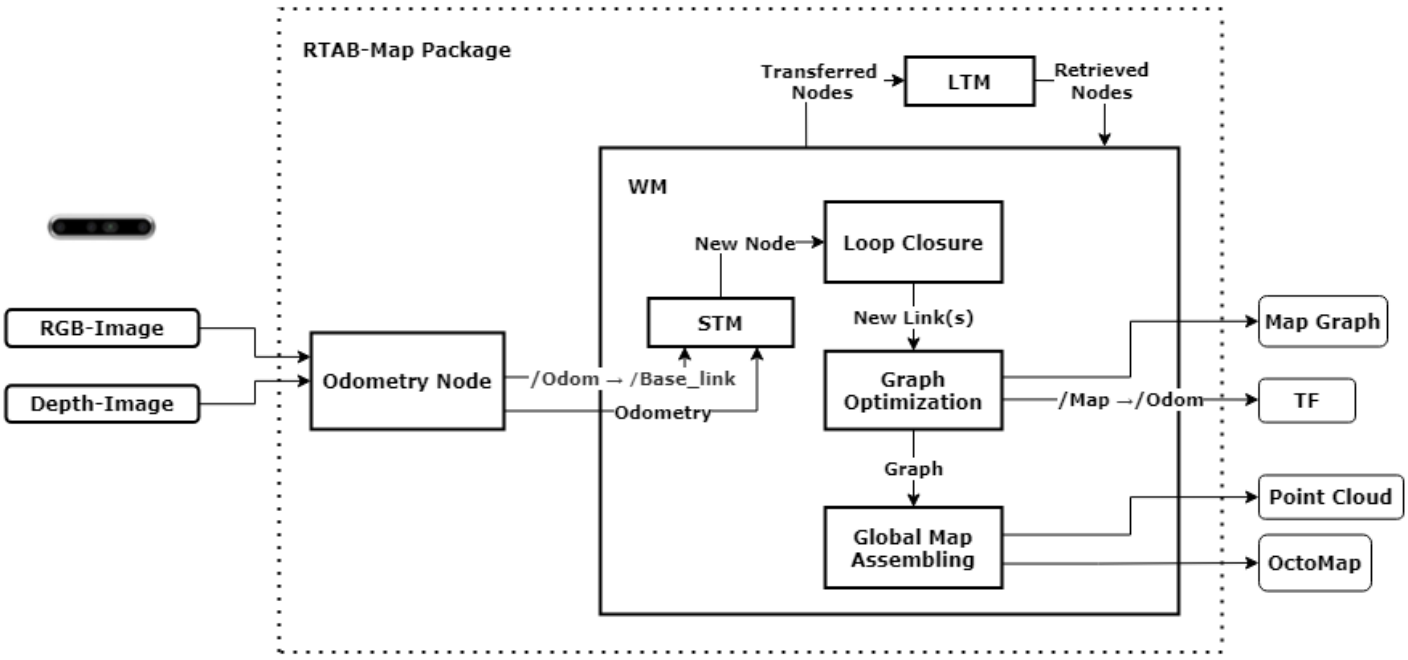


Figure 10: Block diagram of RTAB-Map package.

3.2.2 Software Packages

Real-Time Appearance-Based Mapping (RTAB-Map) was integrated into the system, providing a position and orientation of the drone. This package has a fundamental role in the proposed solution since it performs a large part of the drone's location, extremely important for navigation and its block diagram is represented in figure 10. This package was chosen because it uses RGB-D data, thus taking advantage of all the camera's features. Another advantage of this package is the fact that it has incorporated several different approaches of visual odometry, visual-inertial odometry, and graph optimization, providing a better adjustment to the environment in which the drone is inserted. The RTAB-Map will be explained in more detail in subsection 4.1.

The **Octomap** package performs a 3-D occupancy grid mapping method, presenting data structures and mapping the environment in which the drone is operating. This package was chosen due to its capability to construct a dynamic multi-resolution map, taking into account dynamic environments and its incorporation into RTAB-Map software package. Furthermore, it can define free, unknown, and occupied spaces being essential for safe robot navigation. Another advantage of this package is that the map can model arbitrary environments without prior assumptions about them.

4 *Proposed Solution*

Before presenting the proposed solution, the problem must be formalized and some underlying assumptions must be identified. The mission is for an UAV to be able to explore and map an unknown environment without the need of GNSS signals or other external sources of information, as well as without human intervention. It is assumed that the environment in which the drone operates is an indoor scenario, which can be dynamic or static, composed of multiple walls and objects, and structured with a horizontal floor.

This section explains in detail the proposed solution described in figure 11, where all the main software blocks and their connections are represented. Everything starts at the 'uart_node', responsible for receiving the data sent by ESP32 in a synchronized way through the UART port and transforming that data, publishing them in synchronized ROS topics. This node provides LiDAR data represented in the figure 11. These topics will be received by the node that estimates the drone's height and the ground floor plane. The Visual SLAM calculates the position of the drone and created a 2-D and 3-D map of the environment, simultaneously. Subsequently, the data fusion node receives the estimated drone's pose provided by the Visual SLAM node and fuse it with the estimated drone's height. Consequently, resulting in an estimated drone's pose with higher certainty. At this point, with the 2-D occupancy map created and the resulting pose of the data fusion block, the drone has the necessary conditions to navigate and explore the environment. Finally, local path planning is performed to avoid obstacles while reaching the desired goal. If there is a dynamic obstacle, the reactive approach is activated.

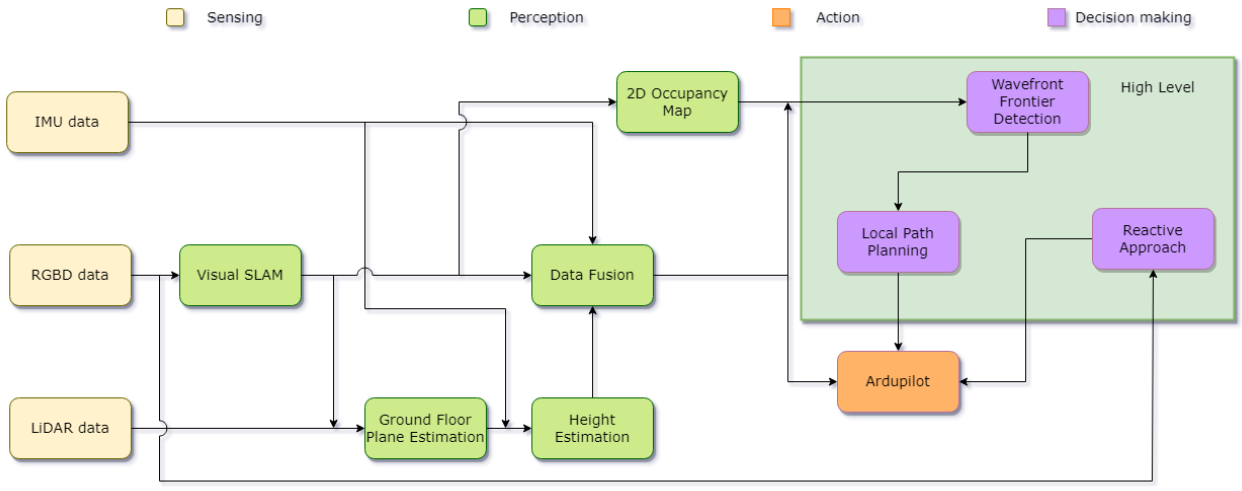


Figure 11: Architecture of the proposed solution in block diagram.

Figure 12 exhibits the Transformation (TF) tree of the complete system, with each reference frame being a 3-D coordinate frame. These TF's make it possible to keep track of multiple coordinate frames over time.

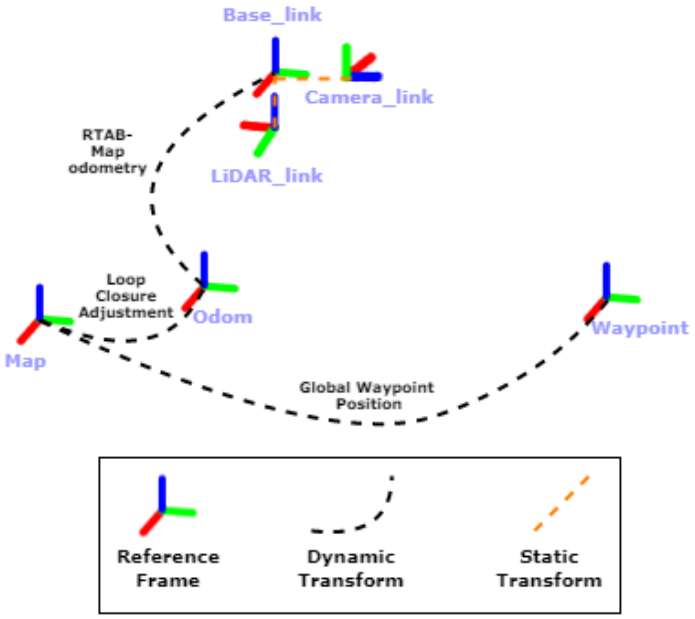


Figure 12: The transformation tree of the reference frames used in the proposed solution.

The entire movement of the base of the drone (base_link) is done with respect to the reference map.

4.1 Visual SLAM

Visual SLAM is the incremental process of creating a map and localizing in that map using information captured by vision sensors. This task is performed through the RTAB-Map package, which is quite complete, incorporating several approaches of Visual Odometry, Visual-Inertial Odometry, and Graph Optimization. Moreover, this package can provide a 2-D and 3-D occupancy grids of the environment in which the drone is operating. This package is represented in figure 10 and is divided into several blocks. The tasks of each block and the connections will be explained in detail in the following subsubsections.

4.1.1 Visual Odometry

RTAB-Map contains the implementation of different approaches of Visual Odometry, such as: Frame to Map (F2M), Fovis, ORB_SLAM 2, Frame-to-Frame (F2F). The Frame to Map (F2M) [31] implementation was chosen for this work. The chosen strategy will be explained very briefly. The **F2M** approach is depicted in figure 13 and can be described by the following steps:

- **Feature Detection**: This approach uses Good Features To Track (GFTT) features [32], an extension of Harris Corner Detector, with a mask to avoid extracting features with invalid depth. The GFTT feature

detector was used to facilitate parameter adjustment and get uniformly identified features across different light intensities.

- **Feature Matching**: The F2M performs the feature matching by comparing Oriented FAST and rotated BRIEF (ORB) [33] descriptors of the extracted features, which are rotation invariant, against those in the feature map.
- **Motion Prediction**: A motion model is used to predict where the features should be in the current frame, based on the previous motion transformation. This implementation restricts the search window for feature matching giving, more suitable matches, particularly in environments with non-static objects.
- **Motion Estimation**: After the Feature Matching, the Perspective-n-Point (PNP) RANSAC [34] implementation in OpenCV is used to compute the transformation of the current frame accordingly to features in the Feature Map (structure that keeps the visual features).
- **Local Bundle Adjustment**: The resulting transformation is refined using local bundle adjustment on features of all key frames in the feature map.
- **Pose Update**: In this block, the pose and the transformation between the *odom* and the *base_link* are updated.
- **Key Frame and Feature Map update**: If the number of inliers computed during motion Estimation is below a fixed threshold, the Feature Map is updated. For the F2M method, the feature Map is updated by adding the unmatched features of the current image and updating the refined matched features position by the Local Bundle Adjustment module.

4.1.2 Memory Management

Memory management limits the size of the graphs, and it is fundamental to perform a long-term online SLAM in vast environments. Without this algorithm, the processing time for modules like Loop Closure, Graph Optimization, and Global Map Assembling can eventually exceed real-time constraints [35]. The memory is composed of Short-Term Memory (STM), Working Memory (WM), and Long-Term Memory (LTM), as shown in figure 14.

When new data received, new nodes are added to the graph in the STM that have a fixed size S. When this size is reached the oldest node is transferred to WM. For loop closure detection and graph optimization, just the WM is taken into account. The WM dimension is determined by a time limit, T. When the time required to process the new data exceeds T, the nodes that are less significant for loop closure (i.e. with less rare visual words) are transferred from WM to LTM, resulting in a nearly constant WM size. When a loop closure occurs, the graph's neighbor nodes in the LTM can be brought back to the WM by a process called Retrieval.

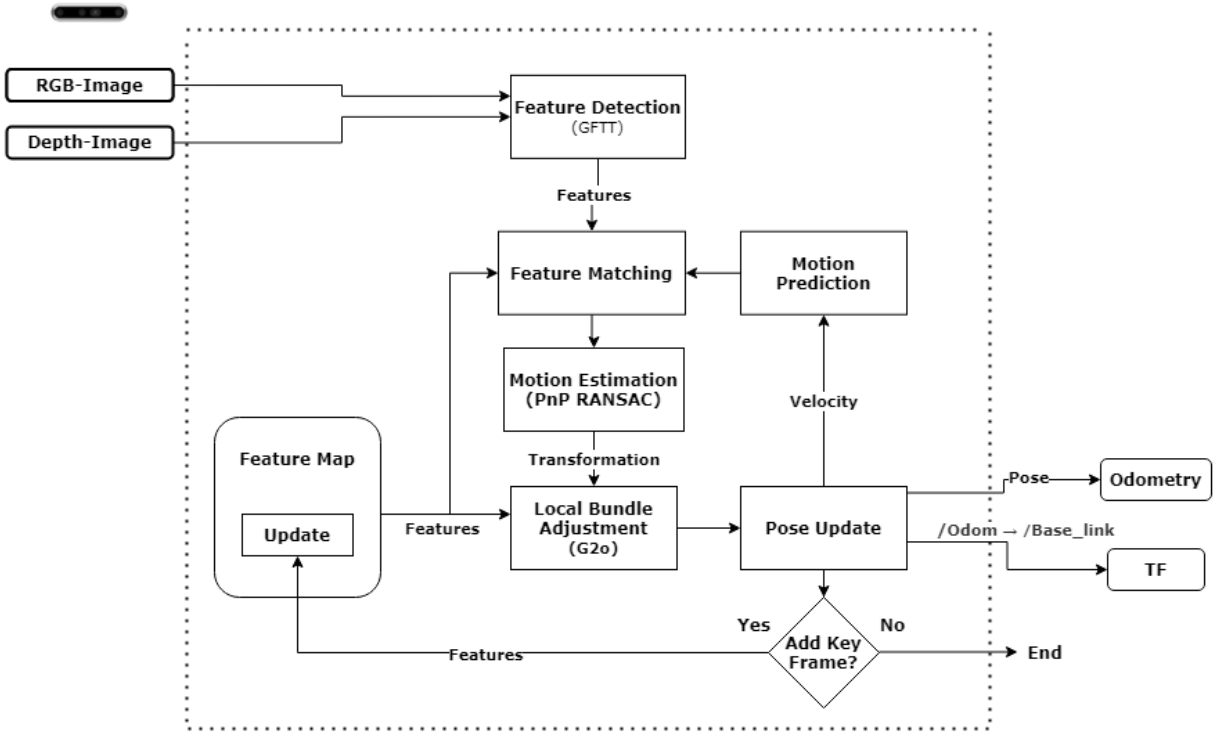


Figure 13: Block diagram of RGB-D odometry with the Frame to Map (F2M) approach.

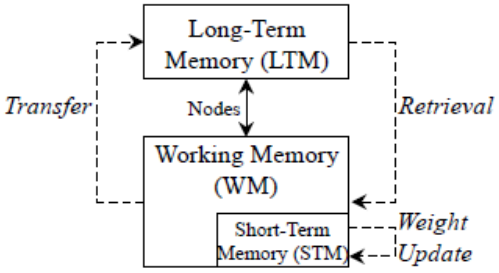


Figure 14: Memory management model (adapted from [35]).

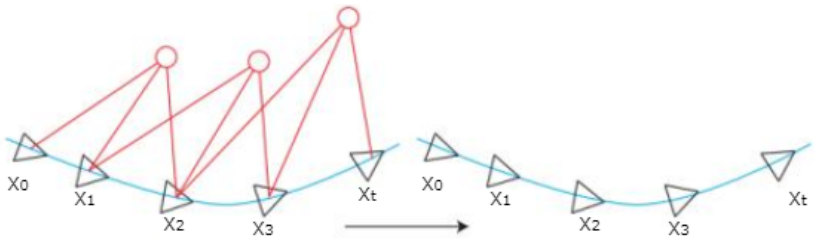


Figure 15: Example of a graph showing multiple features observed from sequential poses (x_i) along a trajectory.

4.1.3 Loop Closure

The technique of correctly declaring that a vehicle has returned to a previously visited area is known as loop closure detection. For loop closure detection, a bag-of-words approach is used, where a set of visual words can describe an image. A visual word is often a "generalized" feature descriptor with the mean value of a cluster of similar visual features. When the STM creates a new node, it extracts SURF features from the frame and quantizes them to generate a histogram containing the amount of occurrences of these visual words.

Since some visual words are good for detecting loop closures while others are less expressive because of how frequently they appear in images, the Term Frequency-Inverse Document Frequency (TF-IDF) approach is used. The TF-IDF reweights every visual word in a histogram, reducing the relevance of "uninformative" visual words (i.e., features that appear in many images) while increasing the importance of rare visual words. When the news histograms are compared, a loop closure is recognized and a transformation is computed if two photos are found to be identical. The change is carried out using the same Motion Estimation approach as in the 4.1.1 subsubsection, and the new link is added to the graph if it is accepted.

4.1.4 Graph Optimization

When a loop closure is detected or some nodes are transferred, a graph optimization approach is utilized to reduce errors in the trajectory estimation and, consequently, in the map. RTAB-Map integrates three graph optimization approaches TORO, g2o, and GTSAM [31]. The g2o and GTSAM approaches converge faster and with a better optimization quality than TORO for a single map. The **g2o** method has great performance and is used in many projects that perform SLAM and thus is the strategy used.

A graph contains two types of components, nodes (vertices) and constraints (edges), as the figure 15 exhibits. The vertices of the graph, depicted with nodes, denote drone poses x_0 to x_t. Every edge between two nodes corresponds to a spatial constraint between subsequent poses and the measurements of landmarks and keypoints [36]. The graph optimization optimizes the camera orientation R and position t, minimizing the reprojection error between matched 3-D points in world coordinates and keypoints in the image to correct all keyframes and the map [37].

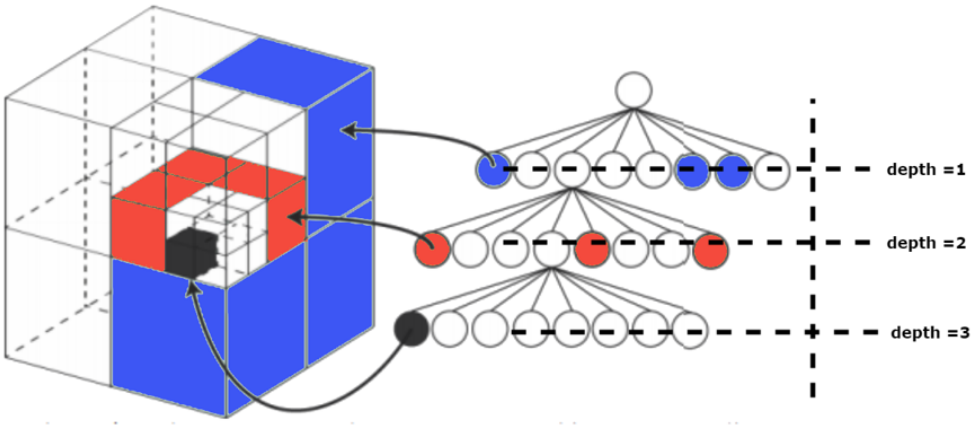


Figure 16: Example of an octree storing free (white) and occupied (colored) cells.

4.1.5 Global Map Assembling

Lastly, for each estimated pose is created its corresponding local occupancy grid. When a new node is added to the map, the occupancy grid is updated, clearing or adding obstacles. Then, it is created a 3-D grid through an octree, which is a hierarchical data structure for spatial subdivision in 3-D. Each node in an octree represents the area contained in a cubic volume named voxel [38]. This volume is recursively subdivided into eight sub-volumes until a given minimum voxel size is reached, which determines the resolution of the octree, as illustrated in figure 16.

The voxels can be defined as free, occupied, and unknown cells. Moreover, the voxels are arranged in an efficient tree structure that leads to a compact memory representation, a faster query of the map, and allows parts of the map at different resolutions.

4.2 Height Estimation

Estimating the vehicle's altitude in an indoor environment means determining the global height relatively to a fixed ground [39]. Since the ground floor is a fixed plane in the considered environments, it is essential to calculate the vehicle's height accurately to achieve accurate local path planning and control the UAV. If the estimation on the z-axis is not precise, it will allow an accumulation of errors in the relative location of the vehicle and, therefore, in the local trajectory planning, which could lead to the UAV crashing into an object.

The Visual Odometry explained before produces odometry with a few errors that can affect the relative position on the z-axis. Therefore, an approach is proposed using a downward pointing LiDAR providing height observations filtered by the 3-D Hough Transform to mitigate the problem addressed previously, providing greater confidence in the drone's localization. The 3-D Hough Transform allows the estimation of the ground

floor plane. The measurements that belong to this plane are inserted in a Kalman Filter to estimate the drone's height. The measurements that are above this plane by a certain threshold are discarded.

4.2.1 Kalman Filter

A Kalman filter is used for calculating with precision the current height of the vehicle comparable to the present ground floor level, based on the observations given by the downward pointing LiDAR, and the measurements of the speed and acceleration on the z-axis provided by the IMU.

The goal is to estimate the drone's altitude, and for this reason, the process model for this axis is the following equation:

$$z = z_0 + \dot{z} \times t + \frac{1}{2} \times \ddot{z} \times t^2 \tag{1}$$

If there are objects underneath the drone, the sensor measurements are not relative to the height of the drone but to the distance from the drone to the object. So it is necessary to filter these measurements and understand which measurements correspond to the height of the drone and which correspond to the objects under the drone. To accomplish this, the 3D Hough transform was used, which can estimate the ground plane and determine whether or not the measurements refer to this same ground plane, which in turn refers to the height of the drone.These measurements, referring to the drone's height, enter as new observations into a Kalman filter that estimates the drone's height based on these observations. This process is illustrated by the diagram 17.

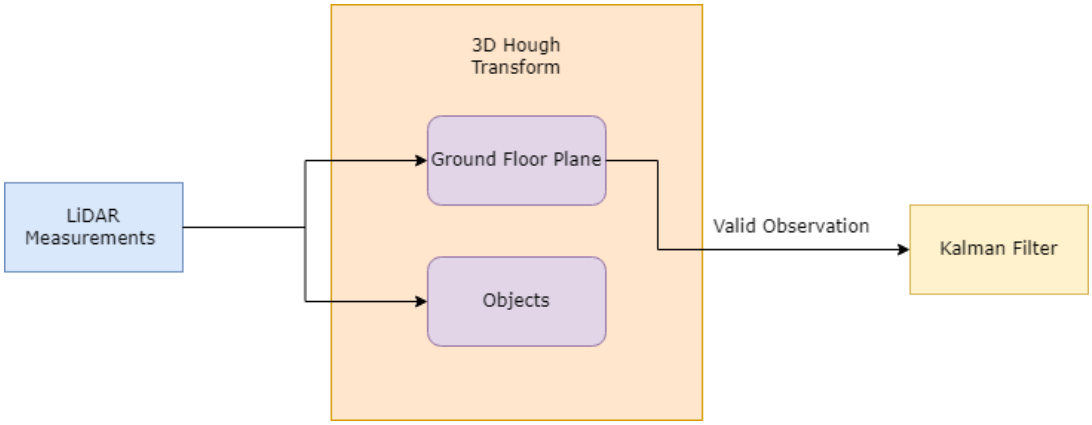


Figure 17: Height estimation process.

4.2.2 Ground Floor Plane Estimation

The main goal of the 3-D Hough Transform is to filter the measurements of the LiDAR by finding the ground floor plane and consequently which measurements are part of it. By knowing the ground floor plane and the

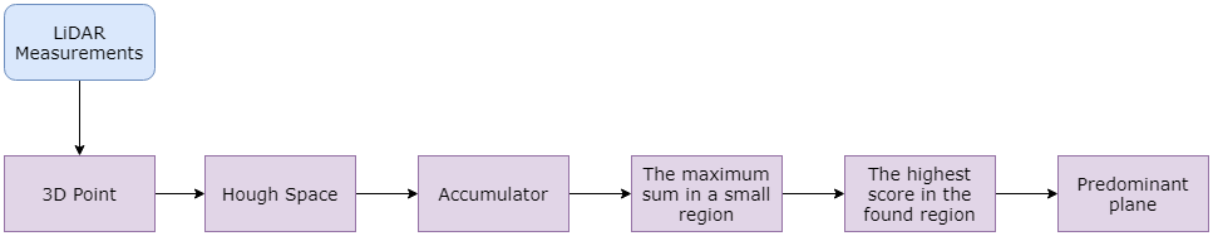


Figure 18: 3-D Hough Transform process.

LiDAR's measurements it is easy to discover the drone's height concerning the same ground floor. The Figure 18 represents the entire 3-D Hough Transform process.

First, when a measurement arrives from the LiDAR passes to the 3-D Hough Transform process. It is necessary to have the drone's position to have a 3-D point. These 3-D points correspond to the x and y coordinates of the drone and, the z coordinate is the difference between the drone's height and the measurement from the LiDAR. This difference prevents wrong estimations when the drone passes over objects.

The next step is to parameterize the 3-D point into the Hough Space. For that, it is necessary to realize what a plane is in three-dimensional space. A plane in 3-D space can be described using a slope-intercept equation, where k_x is the slope in the x-direction, k_y is the slope in the y-direction, and b is the intercept on the z-axis:

$$z = k_x\, x + k_y\, y + b \tag{2}$$

The equation (2) can simply parameterize a plane as (k_x, k_y, b). However, values of k_x, k_y, b are unbounded, which can cause problems due to infinite slopes when trying to represent vertical planes. Therefore, to avoid these problems, another usual definition is the Hesse normal form, that uses normal vectors [40]. Considering the angles between the normal vector and the coordinate system, we can write the plane equation as the following:

$$\rho = x\, \cos(\theta) \sin(\phi) + y\, \sin(\theta) \sin(\phi) + z\, \cos(\phi) \tag{3}$$

with ρ representing the distance between the plane and the origin, assuming the plane is between -1 and 3 meters from the initial reference, being limited $\rho \in [-1, 3]$ meters. The variable θ represents the angle of the normal vector of the plane and the x-axis, $\theta \in [0, 360]$ and ϕ represents the angle between the normal vector of the plane and the z-axis, $\phi \in [0, 180]$. These define the 3-dimensional Hough Space (θ, ϕ, ρ) such that each point in the Hough Space corresponds to one plane in $\mathbb{R}^3$. The Figure 19 graphically represents these 3

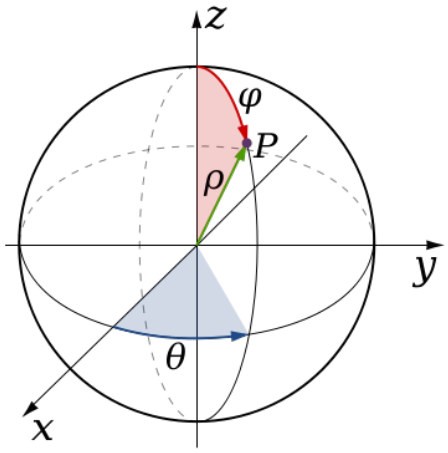


Figure 19: Point P described by polar coordinates.[10]

variables. Therefore, by the equation 3 it is clear that the normal vector n of a plane surface in 3-D Hough space is defined as follows:

$$n = (\cos(\theta)\sin(\phi), \sin(\theta)\sin(\phi), \cos(\phi)) \tag{4}$$

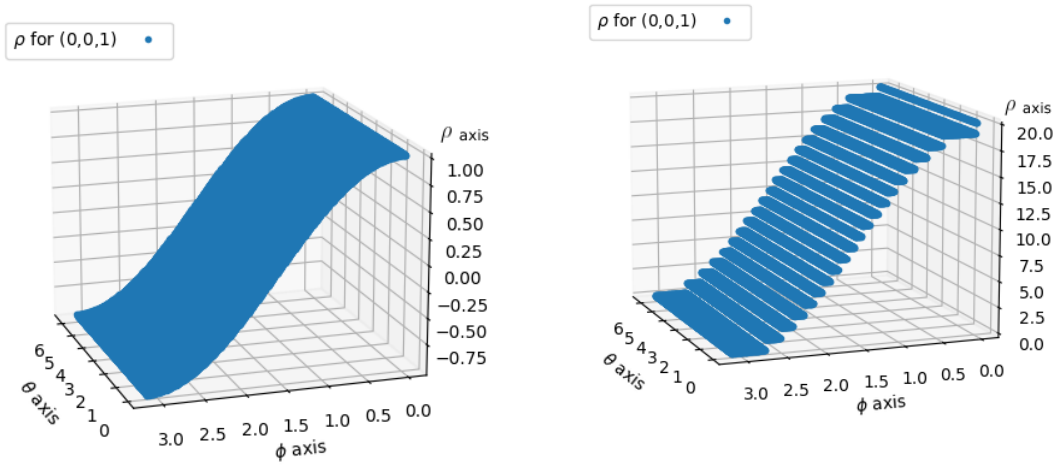
To calculate the Hough Transform for the point in Cartesian coordinates, we have to find all planes the point lies on. For that reason, for all θ and ϕ, it is found all ρ that satisfy the equation (3). Marking these points in the Hough Space leads to a 3-D sinusoid curve, as shown in Figure 20a. For this example was used the Cartesian point (0,0,1).

The intersections of two curves in Hough Space denote the planes found around the line built by the two points. Consequently, the intersection of three curves in Hough Space corresponds to the plane in polar coordinates formed by the three points, showed in Figure 21. For this example was used the three Cartesian points (1,0,0), (0,1,0) and (0,0,1). The more curves intersect on the same point in the Hough Space, the more cartesian points lie on the plane represented by the point (θ, ϕ, ρ) in the Hough Space, and the higher is the probability that this plane is the predominant plane.

To know where is the point with the most intersections, it is necessary to discretize the Hough Space, where $[\theta, \phi, \rho]$ is now a cell of a data structure. As already said, $\rho \in [-1, 3]$ with intervals of 0.1 meters; $\theta \in [0, 360]$ with intervals of 1° and $\phi \in [0, 180]$ with intervals of 1°. This discretization is one limitation of this process because, adds some errors to the data, where the ρ can have a maximum error of 0.1 meters, θ, and ϕ can have a maximum error of 1°. Figures 21b and 20b show the discretization process in two different situations. This data structure, called the accumulator, is needed to store the score of all cells.

For each point in the cartesian system, the accumulator incrementents by one unit all cells $[\theta, \phi, \rho]$ that

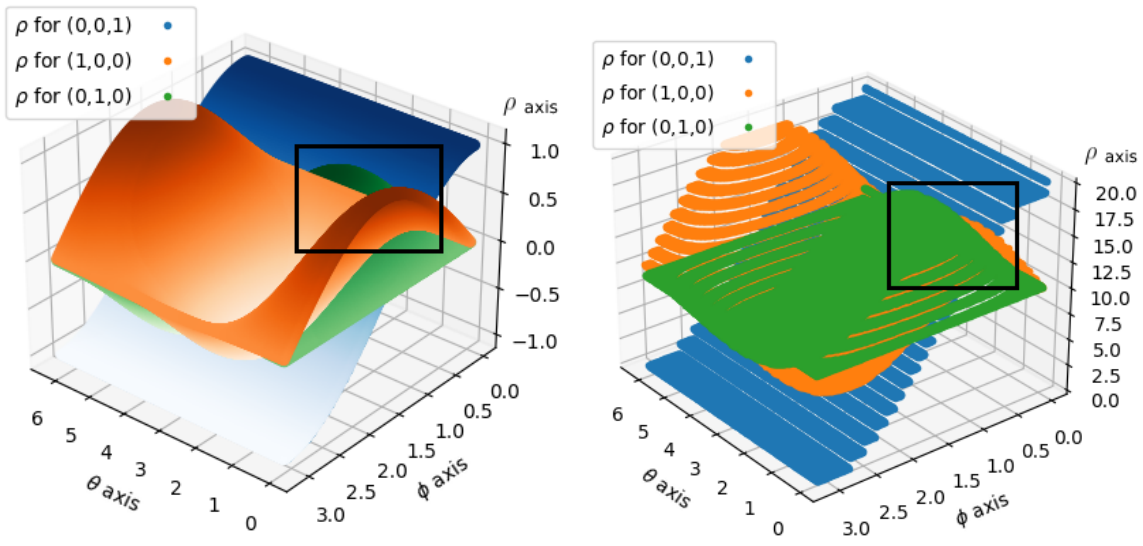
[10] `https://pt.wikipedia.org/wiki/Coordenadas_polares`



(a) Continuous ρ.

(b) Discrete ρ.

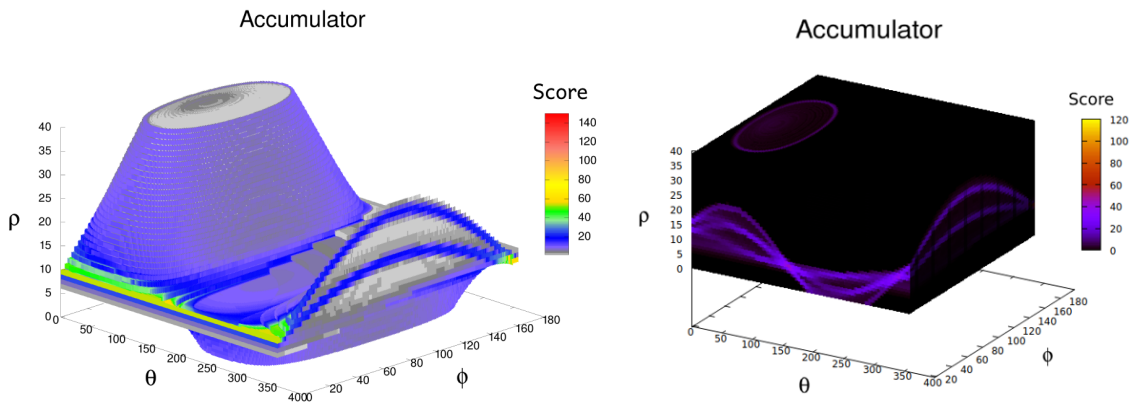
Figure 20: Transformation of one point in Cartesian coordinates into Hough Space (θ, ϕ, ρ).



(a) Continuous ρ.

(b) Discrete ρ.

Figure 21: The intersection of the curves (marked in black) depicts the plane formed by the three cartesian points.



(a) Example of an accumulator array.

(b) Accumulator array sliced.

Figure 22: The accumulator is needed to store the score of all cells $[\theta, \phi, \rho]$.

represent a plane defined by this point. The cells with the highest values denote the most intersections of possible planes described by the cartesian points, depicted in the Figure 22b. Therefore, these cells represent the most prominent planes on the point cloud. The accumulator is depicted in the Figure 22a.

One problem related to the accumulator beyond the additional errors of the measurements is the fact that the Hough Space is enormous (the proposed approach has 2592000 cells), which makes it very difficult to visualize the accumulator and have a good perception of the most prominent plane.

It is necessary to find the cell with maximum value to discover this most prominent plane. However, it is more advantageous to search not only for the cell with the highest score but, to search for the maximum sum in a small region of the accumulator due, to the discretization process of the Hough Space. For this reason, it was implemented a sliding window procedure with the size of 4 cells that searches for this small region.

Unlike the paper [40] that considers that the most prominent plane corresponds to the center point of the sliding window in the Hough Space with a maximum sum of accumulation values, in this case, it is searched for the highest value inside the search window. The cell found is considered the most predominant plane and with the most probability to be the ground floor plane based on the LiDAR's measurements and the drone's position.

The last step in the process is using the equation (3) with the parameters $[\theta, \phi, \rho]$ of the plane that most likely is the ground floor plane and the coordinates x and y of the drone, as the inverse process, targeting plane parameters in a Cartesian coordinate system [41], finding the z coordinate of the plane for the inputted x and y coordinates. All the cartesian points above 0.3 meters of the z coordinate of the ground floor plane are discarded and do not enter on the Kalman Filter.

4.3 Data Fusion

This subsection describes the fusing process of the data from Visual SLAM with the data provided by height estimation and its integration into Pixhawk's EKF. This Kalman filter is very similar to that implemented in Section 4.2, except for the state vector, which is larger since this kalman filter estimates 6 DoFs. The state vector's size is associated with the tracking of both the position and the orientation. The position can be tracked with the x, y, z coordinates and their first and second derivatives (linear velocity and acceleration). On the other hand, the rotation can be tracked through three Euler angles (roll, pitch, yaw) together with their first and second derivatives (angular velocity and acceleration). The state vector is represented as the following:

$$X_k = \begin{pmatrix} x & y & z & \dot{x} & \dot{y} & \dot{z} & \ddot{x} & \ddot{y} & \ddot{z} & \psi_k & \theta_k & \phi_k & \dot{\psi_k} & \dot{\theta_k} & \dot{\phi_k} & \ddot{\psi_k} & \ddot{\theta_k} & \ddot{\phi_k} \end{pmatrix} \tag{5}$$

The drone's pose was obtained through the Visual SLAM and the Height Estimation. Linear and angular velocities and accelerations were obtained through Pixhawk's IMU measurements. It was required to transform the quaternions given by the IMU into the Euler angles. It was necessary to apply the following formulas to make this transformation:

$$\psi = \arctan(2(q_y q_z + q_w q_x), q_w^2 - q_x^2 - q_y^2 + q_z^2) \tag{6}$$

$$\theta = \arcsin(-2(q_x q_z - q_w q_y)) \tag{7}$$

$$\phi = \arctan(2(q_x q_y + q_w q_z), q_w^2 + q_x^2 - q_y^2 + q_z^2) \tag{8}$$

The predicted state Xp_k relates the state at a previous time k_{-1} with the process model, is represented by equation 9. Matrix A is the state transition matrix, the matrix Bu_k is the control variable matrix and w_k represents the predicted state noise matrix.

$$Xp_k = AX_{k-1} + Bu_k + w_k \tag{9}$$

The predicted state can be rewritten as the following:

$$Xp_k = \begin{pmatrix} A_T & 0 \\ 0 & A_T \end{pmatrix} X_{k-1} + w_k \tag{10}$$

with

$$A_T = \begin{pmatrix} 1 & 0 & 0 & \Delta t & 0 & 0 & \frac{1}{2}(\Delta t)^2 & 0 & 0 \\ 0 & 1 & 0 & 0 & \Delta t & 0 & 0 & \frac{1}{2}(\Delta t)^2 & 0 \\ 0 & 0 & 1 & 0 & 0 & \Delta t & 0 & 0 & \frac{1}{2}(\Delta t)^2 \\ 0 & 0 & 0 & 1 & 0 & 0 & \Delta t & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & \Delta t & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & \Delta t \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

The observations made by Visual SLAM and Height Estimation are injected into the Kalman filter through the Measurement Model:

$$\begin{pmatrix} x_k \\ y_k \\ z_k \\ \psi_k \\ \theta_k \\ \phi_k \end{pmatrix} = \begin{pmatrix} I_{[3\times3]} & 0_{[3\times3]} & 0_{[3\times3]} & 0_{[3\times3]} & 0_{[3\times3]} & 0_{[3\times3]} \\ 0_{[3\times3]} & 0_{[3\times3]} & 0_{[3\times3]} & I_{[3\times3]} & 0_{[3\times3]} & 0_{[3\times3]} \end{pmatrix} X_k + V_k \tag{11}$$

where, X_k is the state at time k and V_k is the measurement noise.

The Kalman Filter provides the final position and orientation of the drone. These 6 Degrees Of Freedom (DoF) are injected into the Pixhawk EKF, replacing GPS measurements, allowing position control modes like Loiter, PosHold, Guided, Return To Launch (RTL) and Auto to work. This integration is done by publishing a specific topic called */mavros/vision_pose/pose*. For Pixhawk to accept the topic */mavros/vision_pose/pose*, it was necessary to configure some parameters of ArduPilot[11], allowing it to receive an external source of odometry and fuse it into the EKF. The EKF estimation system of the Pixhawk is a 24 state extended Kalman filter that estimates states like altitude, velocity, position, wind velocity, and many others, providing a stable control of the UAV. This integration with EKF also allows autonomous navigation by sending waypoints, for example, to achieve the goals of a mission.

4.4 Frontier-based Exploration

Exploration of an unknown environment is a fundamental problem in mobile robotics which, carries out the localization of the robot while creating a map of the undiscovered area. This problem appears in many

[11] `https://ardupilot.org/dev/docs/ros-vio-tracking-camera.html`

applications, such as monitoring and mapping forests, inspecting civil infrastructures, or search and rescue scenarios.

Integrating exploration algorithms into robots concedes the potential to remove the human from the loop, making the robot independent of any human action. Therefore, the main task in autonomous exploration is to take a local perception of the environment and extract targets for the vehicle to travel without human interference. The fundamental question in autonomous exploration is: "Given the present knowledge about the environment, where should I move to get the most information?" This question can be answered by the concept of Frontiers.

Has been integrated a frontier-based exploration approach, the Wavefront Frontier Detector (WFD), wherein frontiers are regions on the boundary between unexplored and explored space [42]. The selection of a frontier-based exploration approach was due to its being computationally light, which makes it easier to operate onboard. Furthermore, this method extracts frontiers fast, which makes it more suitable for lightweight and energy-constrained platforms. This approach needs the current position of the drone (from data fusion block) and the current 2-D map (occupancy grid) created by the RTAB-Map package. It is only possible to use the 2-D map to navigate in 3-D, assuming a 2.5-D environment where the 2-D map extends vertically.

The Wavefront Frontier Detector is an iterative method that performs a graph search for frontier detection over already-visited map points, resulting in no need to scan the entire map [43]. The UAV uses a 2-D occupancy-grid map representation in the exploration process, which is described with several concepts:

- **Unknown Region** is the area that has not been reached yet by the drone's sensors.
- **Open-Space** is the acknowledged area that does not have an obstruction.
- **Occupied-Space** is a known region that contains an obstacle.
- **Frontiers** are a set of unknown cells that each has at least one open space neighbor, presented in the figure 23.

The WFD is compounded by two Breadth-first search (BFS), and each BFS has its own queue. Only cells that have at least one open-space neighbor can be added to the queues. Furthermore, to avoid rescanning the same map cell and the same frontier, WFD marks the map cells with four identifications:

- **Map-Open-List**: cells of the 2-D occupancy grid map that have already been enqueued by the first BFS.
- **Map-Close-List**: cells of the 2-D occupancy grid map that have already been dequeued by the first BFS.
- **Frontier-Open-List**: cells of the 2-D occupancy grid map that have already been enqueued by the second BFS.

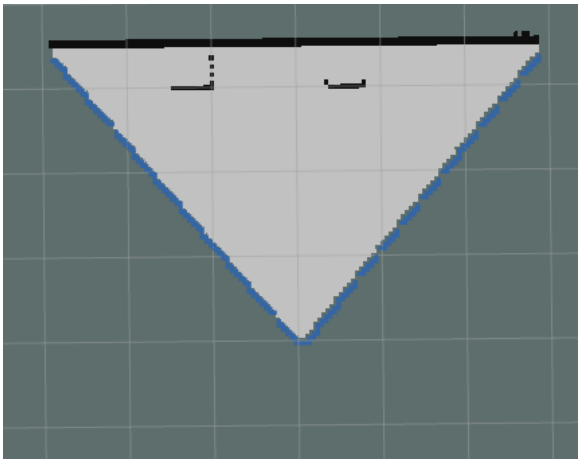


Figure 23: The Unknown Region is represented in dark grey, Occupied-Space in black, Open-Space in white, and Frontiers in blue.

- **Frontier-Close-List**: cells of the 2-D occupancy grid map that have already been dequeued by the second BFS.

The first BFS aims to find all the frontier points contained in the current 2-D occupancy grid map. A frontier point is an open-space cell that has at least one unknown cell. If a frontier point is discovered, the second BFS is performed to extract its valid frontier. Each calculated border is a potential goal for the drone to travel. Therefore, it is necessary to choose the best possible boundary to maximize the mission objectives, which, in this case, is the exploration of the environment where the drone is inserted. This feature acts as a decision-making tool that guides the drone to the next exploration point [44].

The selection of the frontier is based on three criteria:

- A system of quadrants is created to push the drone to navigate in a clockwise direction, trying to reduce the angular rotations of the drone, as these are the principal causes of loss of visual odometry. Moreover, this system allows the drone not to randomly explore the environment and not explore areas that have already been explored. The 2-D occupancy grid map is created in the origin of the drone and is divided into four quadrants depicted in the figure 24. All frontiers that have ($x > 0$, $y < 0$) will be part of the first quadrant, those with coordinates ($x < 0$, $y < 0$) will be part of the second quadrant, and so on. Only when there are no more accessible boundaries in the current quadrant, the drone starts looking for available frontiers in the next quadrant.

- The same frontier can only be chosen once since the objective is to explore areas that have not yet been explored.

- Lastly, the border closest to the last one sent is chosen, with a minimum distance of one meter, as the

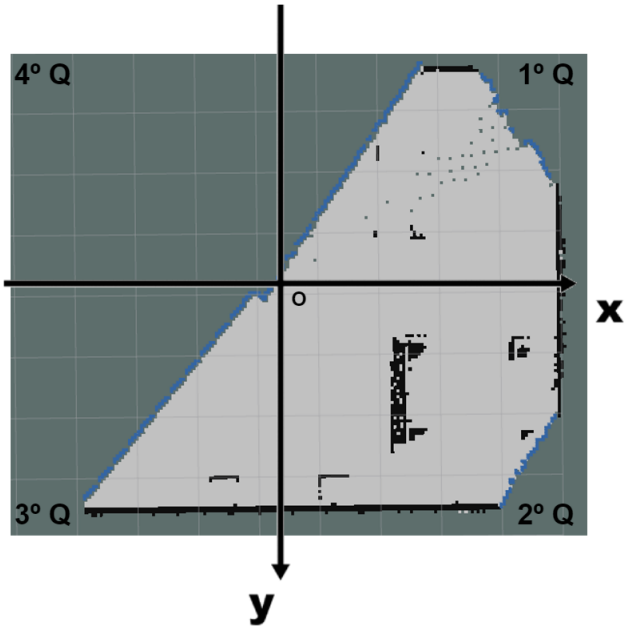


Figure 24: A system of quadrants is created, dividing the 2-D occupancy map into four quadrants.

area around the latest frontier must already be known, based on the range given by the camera.

By flying to successive frontiers, the drone can continually expand its knowledge of the environment and update new information to its map. When there are no new bourders left to explore, the exploration is considered complete.

4.5 Local Path Planning

UAVs are being integrated into a wide range of indoor applications. From this perspective, performing safe and collision-free navigation is essential. These complex applications require autonomous navigation control systems, which must guarantee safe navigation, regardless of the uncertainties present in the environments in which they will operate [45]. For this reason, these autonomous systems must have an acceptable interpretation of the environment in which they are inserted, allowing a decision-making capability. Furthermore, this cognition layer provides the ability to determine whether or not the goal is reachable. The occupied voxels within a specific radius of the target are used to determine whether or not the goal is reachable. If the number of occupied voxels in this radius exceeds a certain threshold, the target is considered inaccessible, and a new target is assigned. On the other hand, if the number of voxels occupied in this radius is less than a certain threshold, the goal is determined to be reachable. After having an accessible target, the force field is generated based on the current map. That being said, it was implemented a reactive module with a decision-making capability to provide collision-free navigation in real-time when there are one or more obstacles in the UAV's

flight path. The local path planning module uses **Virtual Force Field** algorithm, which applies Artificial Potential Fields to avoid static and dynamic obstacles. This algorithm considers the drone as a particle that moves immersed in a potential field caused by the target to reach and by the obstructions present in the drone's surroundings. The goal creates an attractive potential, while each obstacle generates a repulsive potential [46]. Let q represent the position of the drone, considered as a particle moving in a n-dimensional space R_n. Using the full capabilities of the drone is almost mandatory to use a 3-D reactive aproach, and for that reason, q represents a 3-D position in space. The combination of the attractive force and the repulsive forces away from the obstacles is represented as:

$$F(q) = F_{att}(q) + F_{rep}(q) \tag{12}$$

The force $F(q)$ in 12 is the resulting vector that guides the drone in a safe and controlled way. The drone is waypoint-controlled. The attractive force $F_{att}(q)$ is a vector proportional to the difference from q to q_{goal}, and is represented as:

$$F_{att}(q) = -k_{att}(q - q_{goal}) \tag{13}$$

where q is the current drone's position, q_{goal} is position of the goal, and k_{att} is a scaling factor. When the drone is far from the target, the magnitude is high because it is proportional to the distance from the drone to the goal. The repulsion force keeps the drone away from the obstacles and results from the sum of the repulsive effect of all the obstructions, and can be represented as the following:

$$F_{rep}(q) = \sum_i F_{rep_i}(q) \tag{14}$$

The repulsive force of each obstacle is represented as the following:

$$F_{rep_i}(q) = \begin{cases} k_{obst_i} \left(\frac{1}{d_{obst_i}(q)} - \frac{1}{d_0} \right) \frac{1}{d^2_{obst_i}(q)} \frac{q - q_{obst}}{d_{obst_i}(q)} & if \quad d_{obst_i}(q) < d_0 \\ 0 & if \quad d_{obst_i}(q) \geq d_0 \end{cases} \tag{15}$$

where $d_{obst_i}(q)$ is the minimal distance from q to the obstacle i, k_{obst_i} is a scaling constant and d_0 is the obstacle influence threshold. When the drone is too close to an object, the magnitude of the repulsive vector can be very high, causing an enormous repulsion force, sending the drone to huge distances relative to the obstacle. Therefore is necessary to normalize the repulsive vector, keeping the orientation but making the magnitude of the resulting vector smaller enough to avoid objects. This approach has the advantage of being able to be used for real-time obstacle avoidance for any type of robot with onboard sensors during its motion, regardless of

the environment in which it is found. The VFF algorithm has the advantage of being computationally simple as it takes multiple source information and condenses the data into a single resultant vector. Furthermore, as UAV platforms have limited resources with low performance and little memory, this algorithm is well suited to these platforms. One drawback of this algorithm is to find a local minimum, which means that the vector $F(q)$ resulting from the sum of the attractive and repulsive forces is equal to zero, thus leaving the drone in the same place. This issue arises due to the symmetry of the environment, such as hallways. Lastly, the system needs to have a reactive approach since the built map may have errors and may not be updated due to dynamic environments. And consequently, wrong forces can be generated based on faulty maps. Therefore, was implemented a safety measure that allows checking very close obstacles in front of the drone by projecting the point cloud onto the image. If any object is detected in this situation, the drone stops and waits for a new target to reach.

5 *Experiments*

Drones are robots with high maneuverability, which need high stability and control, and often these characteristics are highly-risky when it comes to developing autonomous drones. Therefore, numerous experiments were performed to validate the algorithms described above, both in simulated and in real environments.

The simulations were performed in Gazebo to certify the Height Estimation block and the whole complete system. Subsequently, field experiments were performed to demonstrate the localization block and the height estimation block under the current conditions.

5.1 Metrics

A set of metrics are proposed to evaluate the work created toward the given objectives.

It was used two metrics to evaluate the accuracy of the **Pose estimation** provided by the data fusion block described in Subsection 4.3:

- **Root Mean Square Error (RMSE)**: Align the complete estimated trejectory with ground truth (compute RMSE over all poses):

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n} e_i^2}{n}} \tag{16}$$

- Computation time of the data fusion block.

Since the **Height Estimation** block, detailed in Subsection 4.2, is the most relevant contribution of this dissertation, it is essential to do a complete and detailed evaluation of the implemented algorithm. Hence, it was applied the following metrics:

- Computation time of Ground Plane Estimation.
- Distance between the origin of Normal vector of the Ground Truth Plane and the Estimated Ground Plane.
- Angle between the Normal vector of the Ground Truth Plane and the Estimated Ground Plane represented in Figure 25.
- Computation time of Height Estimation.
- The absolute error of the Height.

To evaluate the **Navigation** and **Exploration** of the drone, including the path-planner algorithms and the obstacles avoidance algorithms it was used a metric with the following characteristics:

- **Navigation characteristics**:

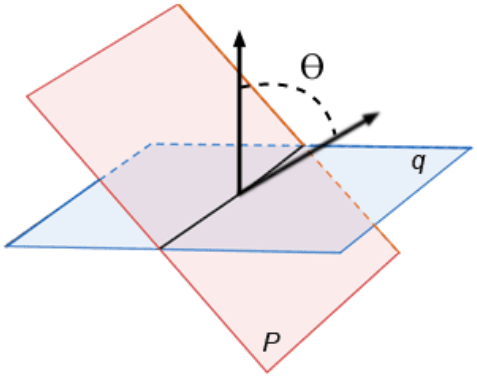


Figure 25: The angle between two normal vectors of two different planes.

- Memory usage.
- Execution Time.
- Central Process Unit (CPU) usage.
- Crashed: The number of times the drone crashed.
- Computation time of the VFF block.

- **Exploration characteristics**:
  - Time of exploration.
  - Completeness: if the drone explored 95% of the environment.
  - Quality of the 2-D occupancy map: For this purpose, an image similarity metric was used [47]. The similarity function is interpreted as the average minimum Manhattan distance of the same color pixels of two images and is given by equation:

$$\Psi(a, a') = \sum_{c \in C} (\psi(a, a', c) + \psi(a', a, c)) \tag{17}$$

where

$$\psi(a, a', c) = \frac{\sum_{a[p_1]=c} \min(md(p_1, p_2) \mid a'(p_2) = c)}{\#c(a)} \tag{18}$$

where C characterizes the set of colors of the image, which is the occupancy value in the case of a grid map, a and a' symbolizes the two images, p_1 and p_2 represents the 2-D image indices and, md(p_1,p_2) is the Manhattan distance between pixels p_1 and p_2. The smaller the coefficient, the greater the similarity between the maps.

  - Computation time of the Wavefront Frontier Detection block.

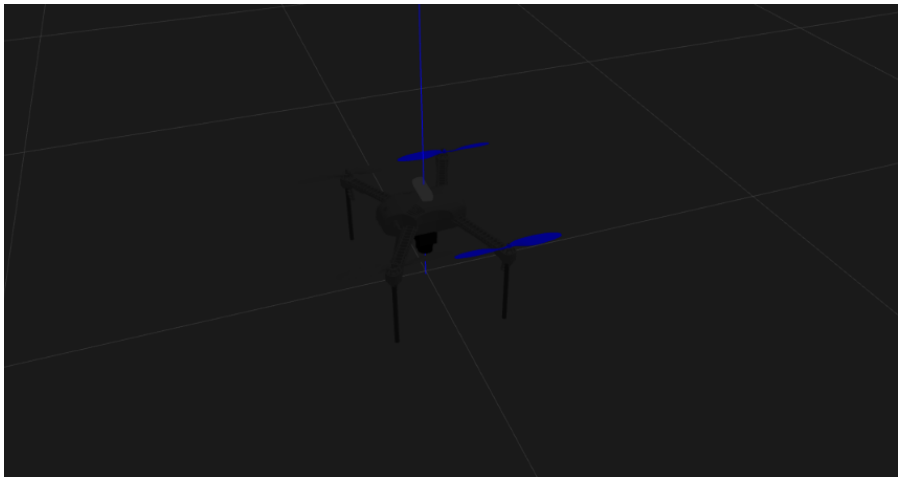


Figure 26: Simulated drone in Gazebo.

5.2 Simulation

The Gazebo framework is an excellent tool to develop and test new algorithms due to its capability to simulate the physics of the drone and the environment. Consequently, the same indoor experiments can be performed in simulation with the same outcome as the real drone in the real world.

The drone is represented in figure 26 to simulate the experiments that were created. This drone is trying to replicate the drone used in the field, depicted in figure 8. For this, it was designed a drone with the same characteristics. First, the Iris QuadCopter model, which is very similar in terms of weight and diameter, was employed. It was placed the Hokuyo LiDAR under the drone to simulate similar measurements to VL53L1X Time-of-Flight Distance Sensor. Finally, a Realsense 435i camera was embedded on top of the drone, being the main difference between the simulated drone and the real one. The Realsense D455 camera of the physical drone is inverted compared to the Realsense 435i camera of the simulated drone. However, this difference does not affect the visual odometry because it is invariant to camera rotation.

The simulation machine is equipped with an Intel (R) HD Graphics 4600, an Intel (R) Core (TM) i7-4700MQ CPU, and 8GB of RAM. This laptop has acceptable performance, which is critical for the tests because the Gazebo simulator and all of the algorithms are computationally expensive.

5.2.1 Height Estimation

As already stated, the Height Estimation block is the most relevant contribution of this dissertation. It is essential to do a complete and detailed evaluation of the implemented algorithm. With this purpose, a set of tests was designed with multiple configurations to evaluate the behavior of the proposed algorithm in different situations and gather relevant information to analyze the performance of each simulation.

It was decided to assess this response in a zig-zag flight of 30 meters from its origin. In all configurations, the drone makes the same path and the same distance. Five tests were carried out. For each test, 30 simulations were performed. Therefore, 150 simulations were accomplished to evaluate the performance of the algorithm. The following tests were performed:

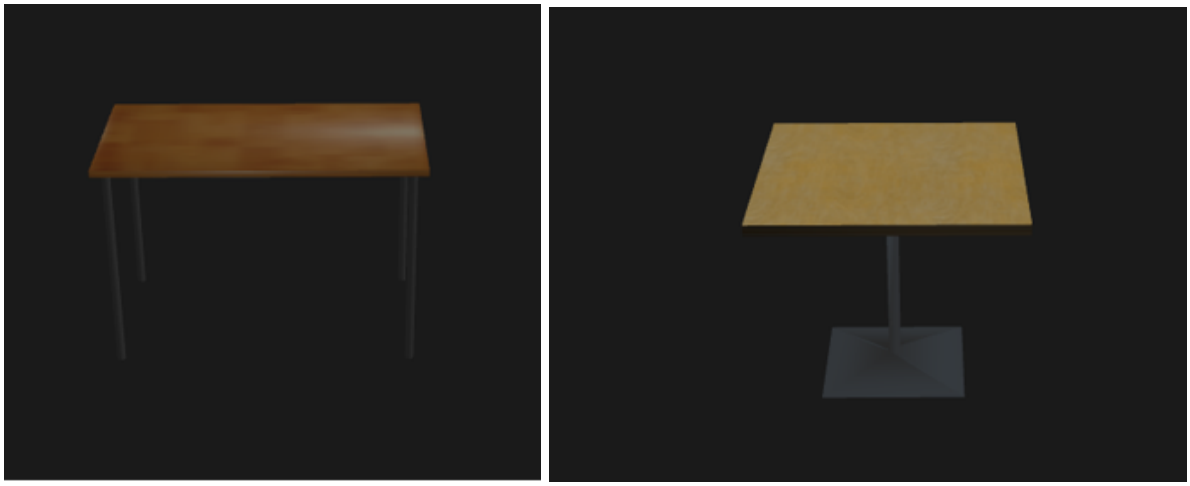
- Test 1: The drone executed a flight with constant height during the entire flight, the environment contained no objects on the floor.
- Test 2: The drone completed a flight incrementing the altitude during the entire flight, the environment contained no objects on the floor.
- Test 3: The drone performed a similar flight of Test 1, with the main difference being the existing objects on the floor.
- Test 4: The drone performed a similar flight of Test 2, with the main difference being the existing objects on the floor.
- Test 5: The drone achieved an aggressive flight, in which it executed drastic changes in its height, existing objects on the floor.

Test 1 intends to understand the behavior of the algorithm in a flight as simply as possible. On the other hand, test 2 aims to examine the behavior of the algorithm in constant height changes. Tests 3 and 4 intend to analyze the algorithm's response in the presence of objects on the floor. Finally, Test 5 plans to analyze the behavior of the algorithm in extreme conditions.

The obstacles used are represented in figure 27. The table model and cafe_table model were used because they are usually found in indoor environments, in which there are no GNSS signals. The table model has a height of 1 meter and, the cafe_table model has a height of 80 centimeters.

The estimated altitude is compared to the ground truth to obtain the absolute error of the estimated height. The GPS is used for ground truth, as it is the closest measure to the actual drone position.

Table 2 shows the results of the tests performed above described, where the 'Time Ground' represents computation time of the ground plane estimation and the 'Time Height' represents the computation time of the Height Estimation. The 'Error' is the absolute error of the height estimation $\pm$ std, 'Distance' represents the distance between the origin of the normal vector of the ground truth plane and the estimated ground plane $\pm$ std and 'Angle' represents the angle between the normal vector of the ground truth plane and the estimated ground plane. It was used the metrics described in subsection 5.1 to evaluate the height estimation block. It was observed that the compute time of the Ground Plane Estimation block is between 165 to 205 ms. This time varies depending on the measurements that are filtered. When are filtered measurements, the computation time is reduced, since they do not enter in the Hough Transform. Height Estimation computation



(a) Table model. (b) Cafe_table model.

Figure 27: Obstacles detected by LiDAR measurements.

time is based on the computation time of the Ground Plane Estimation block, and the height is only updated when the Ground Plane Estimation block is completed. As a result, Height Estimation takes 0.001 seconds to update the altitude. Analyzing the absolute error of height estimation reveals that from test 1 to test 4, there is an average error of 1.5 cm, which is quite accurate. In test 5, the mean absolute error was increased to 7.3 cm. This occurrence occurred as a result of the update rate not being rapid enough to keep up with the drone's sudden ascension. The same is true for all experiments' take-off, which is seen in figure 29 and is discussed more below.

Lastly, the last two lines of the table refer to the floor plane estimation. It is observed that in all experiments, the angle between the normal vector of the ground truth plane and the estimated ground plane, represented in figure 25, is about 1.5°. To assess the ground plane estimation, it is also necessary to evaluate the distance between the origin of the normal vector of the ground truth plane and the estimated ground plane. This distance differs between 6.7 and 7.5 cm in the tests performed. It can be concluded from these two statistics that detecting objects under the drone or changing its height does not interfere with the estimation of the ground plane.

Figure 28 depicts one of the experiments in the test 4 where the drone performed a flight with different altitudes with obstacles under the drone. The drone takes off to 2 meters in elevation, then slowly increment the height. During the flight, the measurements of the LiDAR (represented in yellow) follow the drone's altitude until it detects the objects where it gets the distance from the drone to the obstacle. These measurements do not correspond to the drone height, as a result, do not refer to the ground plane, hence they are excluded from the drone height estimation.

In all experiments, a maximum error of 1 meter in the height estimation was observed in the take-off. This

Table 2: Comparison of the results of the Height Estimation block in different tests.

	Test 1	Test 2	Test 3	Test 4	Test 5
Time Ground (s)	0.204	0.203	0.174	0.167	0.205
Time Height (s)	0.205	0.204	0.175	0.168	0.206
Error (cm)	1.402 ± 0.17	1.452 ± 0.10	1.457 ± 0.21	1.704 ± 0.23	7.380 ± 0.49
Distance (cm)	6.734 ± 0.40	6.864 ± 0.34	7.056 ± 0.51	6.998 ± 0.48	7.416 ± 0.32
Angle (∘)	1.519	1.559	1.602	1.557	1.764

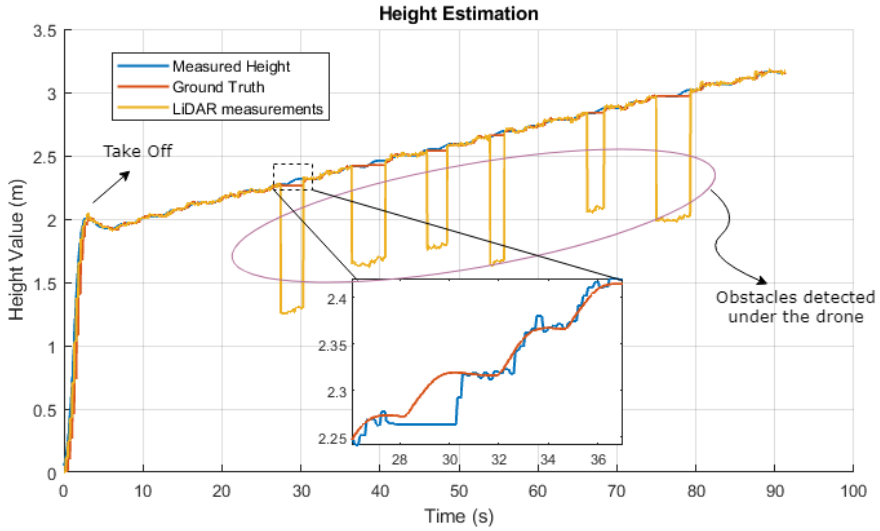


Figure 28: Height estimation in respect of the LiDAR measurements and the ground truth of the test 4.

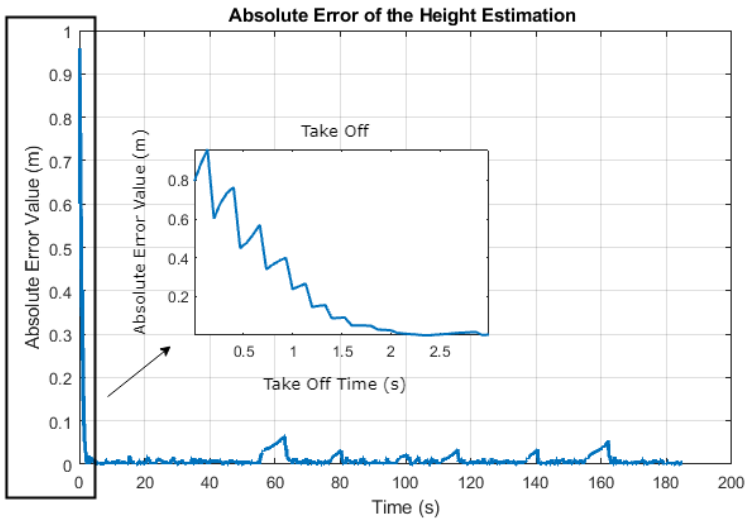


Figure 29: Absolute Error of the Height estimation of the test 4.

error is compensated in about 2 seconds after the take-off, as shown in Figure 29. This event occurs because the algorithm updates every 200 ms, and consequently, the output from the algorithm do not keep up with the drone's ascent rate. By looking at Figure 28, it is clear that the drone is filtering out measurements that do not correspond to the ground plane but correspond to object detection. This event occurs since the LiDAR observations are of much more importance in estimating the drone's height due to their greater certainty in the measurements than the velocity and acceleration states in the z-axis. Thus, resulting in an increase in the absolute error of the drone's height estimation until obtaining a new measurement corresponding to the ground plane, as shown in figure 29.

5.2.2 Navigation without GNSS signals

The simulations with the proposed system are described in this subsection. The drone described in subsection 5.2 was utilized to evaluate the performance of the complete approach. Two distinct scenarios were created to assess the capabilities of the proposed approach.

The created environment (scene1) is represented in figure 30a and, it is intended to simulate an indoor environment in which it has four walls, a door, and many objects that usually are present in a room. The created scenario is 7.5 meters wide and 12 meters long.

Scenario 2 was created to be more complex to assess the drone's ability to avoid obstacles to explore the environment. Scenario 2 is illustrated in figure 30b. Similar to scenario 1, scenario 2 intend to simulate an indoor environment with multiple objects but in an "L" shape. The created scenario is between 6 to 10 meters

(a) Scene 1.

(b) Scene 2.

wide and 17.5 meters long.

Thirty experiments were carried out in the two scenarios described above to assess the proposed solution. In all experiments, the drone takes off 1 meter, makes a 360-degree turn to collect as much information as possible from the scene in which it is located, and explores the room completely autonomously, choosing the frontiers to which it will move, consequently building the map of the environment.[12]

It was also carried out thirty experiments in scene 2 but without the height estimation block to verify how much robustness the method gives to the system.

In both scenarios, when the visual odometry is lost, the drone receives a signal to land on the last know position, thus creating a safety mechanism capable of landing the drone safely.

The results of the metrics concerning the height estimation block are presented in table 3. Where 'Error' is the absolute error of the height estimation ± std, 'Distance' represents the distance between the origin of the normal vector of the ground truth plane and the estimated ground plane ± std and 'Angle' represents the angle between the normal vector of the ground truth plane and the estimated ground plane. Observing the table's results and comparing them with the results of subsection 5.2.1, it can be concluded that the algorithm can estimate the ground floor plane with a low error for any trajectory the drone makes, estimating the height of the ground floor plane at 3.99 cm in scenario 1 and 6.58 cm in scenario 2. However, looking at the angle of inclination of the ground floor plane in scenario 2, it can be seen that it is much higher than that seen in

[12] `https://youtu.be/tlgvyLzCDcM`

Table 3: Results of the Height Estimation block in Scene 1 and Scene 2.

	Error (cm)	Distance (cm)	Angle (º)
Scene1	2.60 ± 0.41	3.99 ± 2.94	3.02
Scene2	0.18 ± 0.02	6.58 ± 4.63	21.52

subsubsection 5.2.1 and scenario 1. This angle value is a consequence of often a difference between the altitude given by the visual odometry and the LiDAR reading, resulting in observations further away from the actual value of the ground plane height, increasing the slope of the estimated ground plane. Lastly, the drone height estimation is also estimated very accurately, having only an error of an average of less than 3 cm in both scenarios.

The computational time for each block of the approach is represented in figure 31. In this figure, it can be observed that altitude estimation and 3-D Hough transform takes the longest time on average to complete an iteration. Both algorithms behave the same as verified in subsubsection 5.2.1. The difference between both scenarios is since, in scenario 2, the drone passes over one of the tables, thus rejecting the observations, not needing to update the accumulator, thus saving some computational time.

The computation time of visual odometry depends on the frame rate achieved by the RGB-D data. Under current conditions, on average, these data arrive at a rate of 7 Hz, which leads to the visual odometry calculation time shown in Figure 31. This 5.5 Hz (scene 1) and 6.9 Hz (scene 2), corresponding to visual odometry computation time, is a small value for real-time systems. It can lead to errors and even loss of odometry. Therefore, these simulations need to be run on a computer that can publish RGB-D data even faster. The difference between the scenarios can be due to the higher number of detected features in scene 1 since it has more objects and different textures than scenario 2, thus resulting in more computational time.

The VFF algorithm has to be fast enough to avoid obstacles in real-time through the force field. Through figure 31, it can be concluded that the VFF is quite fast, needing only an average of 30 milliseconds to generate the potential field of the current map. The Data Fusion block is very fast at merging the visual odometry and the estimated height, and it takes an average of 2 milliseconds to do this task. The Wavefront Frontier Detector has a low computation time thus, being able, with the current map, to calculate the borders and send the chosen frontier very quickly. The difference between scenarios is because scenario 2 is sizeable larger, having more frontiers to calculate. These experiments were performed on the same computer described in subsubsection 5.2.1. On average, the computer's in-use memory was 65.5% (scene 1) and 72.18% (scene 2) when running the 30 simulations. The laptop's CPU averaged was 92% and 83.3% when running the 30 experiments. With these data, it can be concluded that it is necessary to run the simulations on a computer with better specifications to lower the CPU percentage and increase the camera's frame rate.

Table 4 demonstrates that the X-axis has a higher error than the Y and Z axis. The main reason this

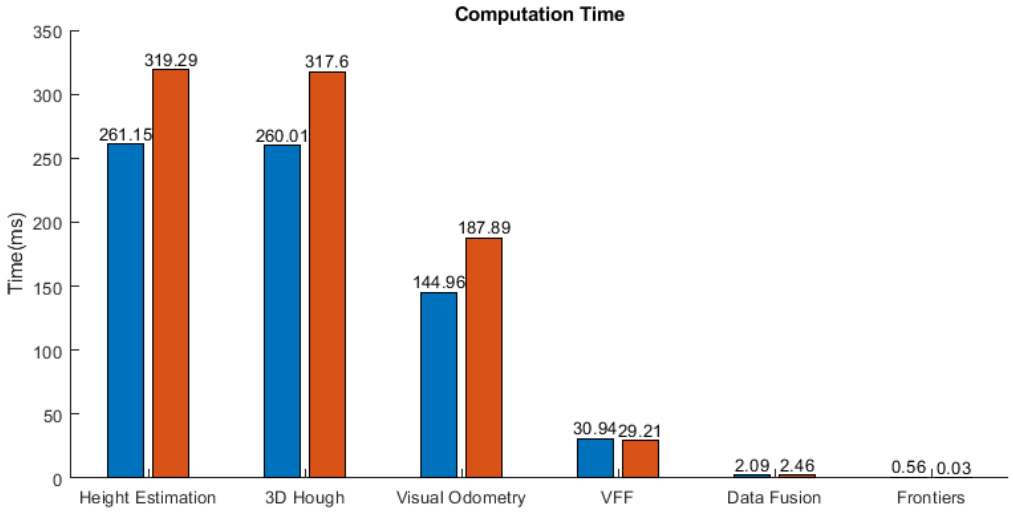


Figure 31: Computation time of all system components of both scene 1 (red) and 2 (blue).

happens is that the camera is 10 cm away from the drone's center of gravity on the X-axis. The camera's position makes the drone less stable on the X-axis and increases the odometry error on this axis. Due to the implementation of the height estimation block, the Z-axis error is the lowest of the three-axis relative to the position. Regarding the drone's orientation error, only the yaw has an average error of 2 degrees. This error is mainly due to the low frame rate of the visual odometry, having a small latency relative to the actual drone orientation. This behavior occurs in both scenarios. In order to test the robustness of the height estimation block, thirty tests were run in scene 2 without it. When table 4 and table 5 are compared, it is clear that the system behaves identically, with the X axis having the biggest error due to the camera's position, which makes the drone less stable on the X-axis, increasing the odometry error on this axis. Furthermore, the orientation error in the two tables is nearly identical. However, without the height estimation block on the system, the error in the Z axis increases slightly. As a result, it is possible to deduce that the height estimation block reduces the inaccuracy in the z axis.

Regarding the exploration part, the various metrics defined are evaluated in subsection 5.1. Whenever we talk about exploration, we need to assess the time the robot spends exploring the environment. On average, the drone took 164.76 seconds to explore the scene 1 environment and 889.21 seconds to explore scene 2. The difference between scenarios is because scenario 2 is sizeable larger, therefore having more frontiers to verify if they are accessible and having more area to cover. It is significant to note that the system is considered safe and stable, as the drone **never crashed** in all the experiments performed on both scenarios. Next, it is required to analyze and compare the 2-D occupancy maps generated throughout the exploration with the Ground Truth of the 2-D Occupancy Map of the scene 1 to assess the quality of the exploration, using the

Table 4: RMSE of the drone's pose in Scene 1 and Scene 2.

	Scene 1	Scene 2
RMSE of X axis ± std (cm)	16.8 ± 5.95	20.64 ± 16.35
RMSE of Y axis ± std (cm)	9.1 ± 3.24	11.18 ± 6.88
RMSE of Z axis ± std (cm)	4.6 ± 4.78	7.76 ± 5.06
RMSE of Roll ± std (^{o})	0.02 ± 0.01	0.02 ± 0.01
RMSE of Pitch ± std (^{o})	0.03 ± 0.01	0.03 ± 0.01
RMSE of Yaw ± std (^{o})	2.77 ± 0.30	2.70 ± 0.27

Table 5: RMSE of the drone's pose in Scene 2 without the height estimation block.

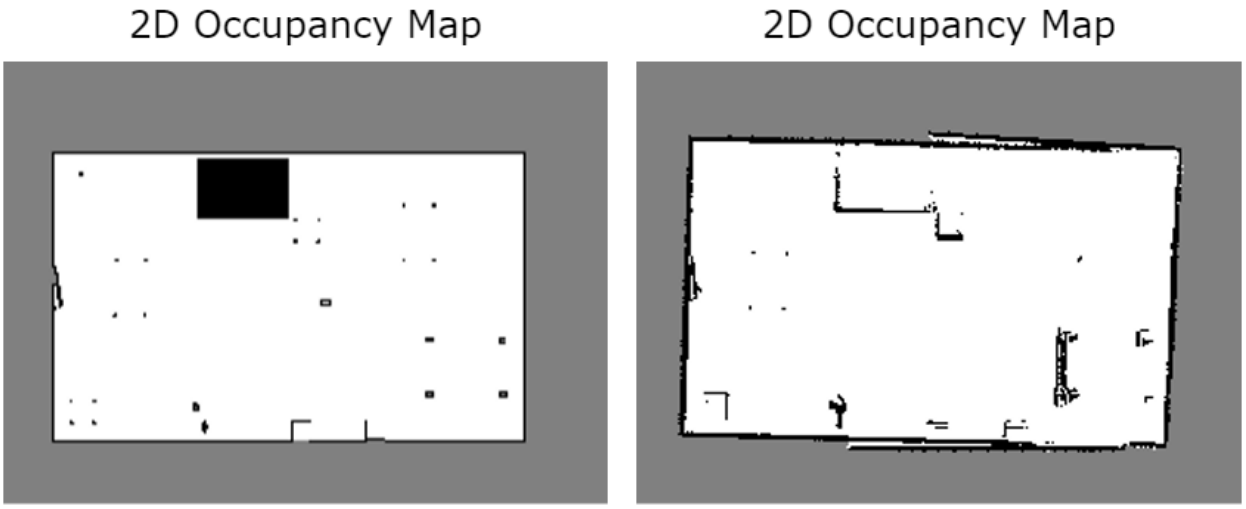
	Scene 2
RMSE of X axis ± std (cm)	15.45 ± 24.1
RMSE of Y axis ± std (cm)	15.40 ± 11.2
RMSE of Z axis ± std (cm)	11.10 ± 3.3
RMSE of Roll ± std (^{o})	0.02 ± 0.05
RMSE of Pitch ± std (^{o})	0.02 ± 0.03
RMSE of Yaw ± std (^{o})	2.77 ± 0.21

image similarity metric described in subsection 5.1.

The *gazebo_ros_2Dmap_plugin*[13]was used to create a accurated 2-D occupation map of a simulated environment at a specific height. Figure 32a shows the map generated at the height of 1 meter, which is the height of the selected edges. For 30 experiences, the average similarity between the image of the Ground Truth of the 2-D Occupancy Map of scene 1 and the generated map is 18.2022. In figure 32b, the map with the best similarity coefficient of the 30 experiments performed is represented. In comparison between the two images, it can be concluded that the map generated by the autonomous exploration inclines approximately 1.71 degrees relative to the ground truth of the 2-D occupancy map. As shown in Table 4, it can be concluded that this event occurs due to a slight positional error of 2.77 degrees for yaw. For the 30 experiences, the average similarity between the image of the Ground Truth of the 2-D Occupancy Map of scene 2 and the generated map is 96.1744. The difference between the coefficients of the different scenarios is since the map is sizable larger in scenario 2. Therefore, there are more pixels to compare, and, consequently, the error will be higher, as the mapping is not perfect. In figure 33b, the map with the best similarity coefficient of the 30 experiments performed is represented.

It is beneficial to perceive if the 3-D map created through the exploration is complete, therefore allowing

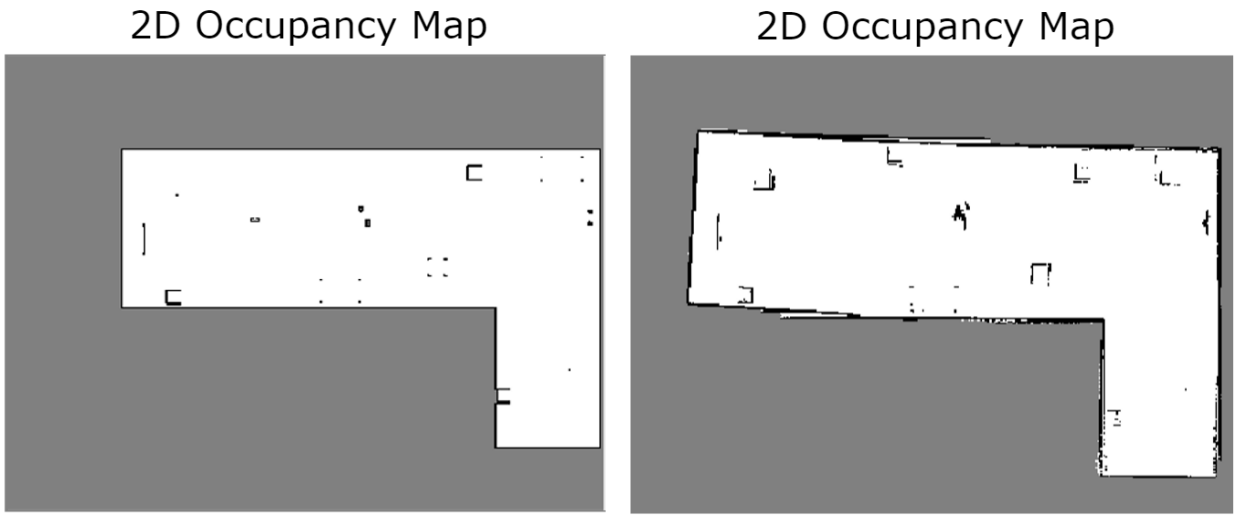
[13]`https://github.com/marinaKollmitz/gazebo_ros_2Dmap_plugin`



(a) Ground Truth 2-D Occupancy Map of the presented scene 1.

(b) Best 2-D Occupancy Map generated from the exploration with an image similarity of 15.9233.

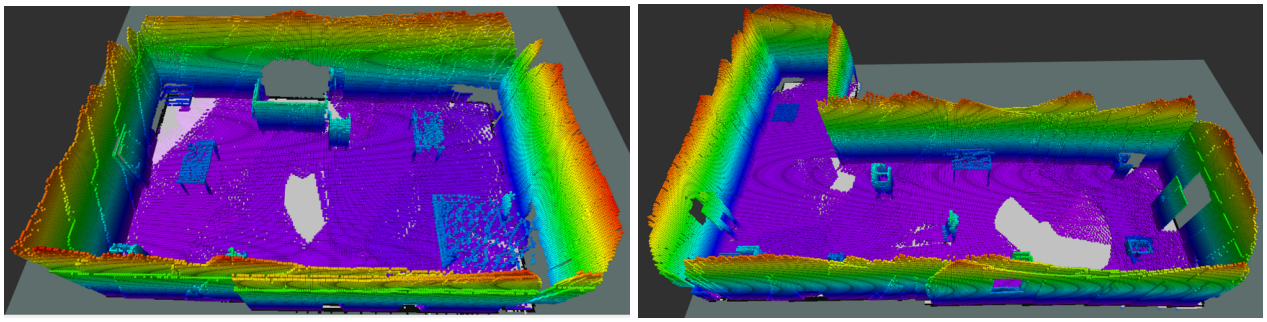
Figure 32: Comparison between the best 2-D Occupancy Map generated from the exploration of the scene 1 and the Ground Truth of the 2-D Occupancy Map.



(a) Ground Truth 2-D Occupancy Map of the presented scene 2.

(b) Best 2-D Occupancy Map generated from the exploration with an image similarity of 90.7571.

Figure 33: Comparison between the best 2-D Occupancy Map generated from the exploration of the scene 2 and the Ground Truth of the 2-D Occupancy Map.



(a) 3-D map generated from autonomous exploration of the scene 1.

(b) 3-D map generated from autonomous exploration of the scene 2.

Figure 34: 3-D maps resulting from the exploration of the two scenarios.

to comprehend if the exploration was successful. For this, the completeness metric was created, considering that a map is complete if its percentage is equal to or greater than 95% of occupied cells relative to the ground truth of the 3-D map of the environment. For scenario 1, 20 of the 30 experiments performed have a percentage equal to or greater than 95% of occupied cells. While for scenario 2, 28 of the 30 experiments performed have their full 3-D map. The fact that there are fewer experiments in scenario 1 where the number of occupied cells is equal to or greater than 95% is due to the camera not having enough range in specific trajectories that the drone takes near the center of the map to create sufficient 3-D features to be compared to the feature map, thus losing odometry. In scenario 2, this event does not occur because there are objects in the middle of the map, allowing the creation of 3-D features and, consequently, the estimation of the drone's position.

One of the main reasons to simulate scene 2 was to observe the behavior of the system when the environment has objects to avoid, to explore the environment. Figure 35 shows the scenario where it is necessary to avoid obstacles. Figure 36 depicts the path taken by the drone to reach the three green points in scene 2, which are the chosen frontiers. It is possible to observe that the drone navigates safely and is able to avoid obstacles. It is then possible to conclude that the drone can explore an environment regardless of whether there are obstacles along the way or not.

5.3 Field Experiments

To apply the proposed solution in the field, it is first necessary to test the *alt_hold* mode to see if the drone can maintain the altitude autonomously. After doing some tests and some Pixhawk parameter adjustments, the drone was able to successfully maintain the altitude autonomously by injecting the z-position of the drone provided by the proposed location system. The next step was flight in Loiter mode, in order to allow the drone to control the position and orientation autonomously, by injecting its pose from the proposed approach. Multiple tests were performed to achieve this, but without success. Even with the drone at the same position

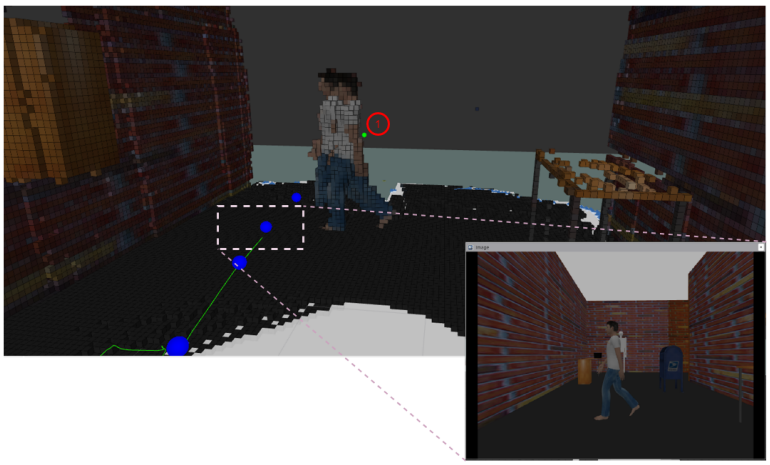


Figure 35: The blue dots represent the waypoints provided by the VFF algorithm in order to reach the green dot, which is the choose frontier. The image on the right is the output image from the drone's camera.

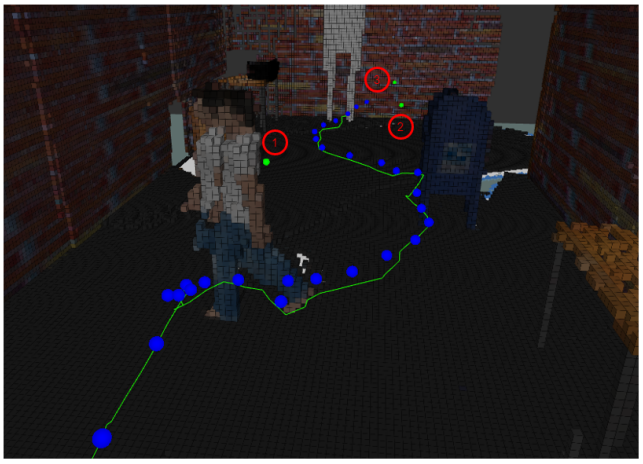


Figure 36: The numbers represent the sequence of the chosen frontiers (green dots). The blue dots represent the waypoints provided by the VFF algorithm and, the green line is the drone's path.

and injecting its correct position, the drone did not maintain its position. After analyzing multiple Pixhawk logs and rosbags, it was determined that the problem lies in the lack of processing to process the camera data, which significantly reduces the FPS of the ROS topics that contain the camera data. In view of the processing problem, the image resolution and fps given by the camera were reduced as much as possible. There was also an attempt to establish the best balance between the RTAB-Map parameters and the processing required to ensure the drone's odometry. The size of the local map (feature map) that stores 3-D features with descriptors of the last keyframes was reduced. The maximum features extracted from the images were also reduced and the minimum distance between detected features was increased to reduce the calculation time for feature matching and hence the estimation of the drone pose, but still without success. In a last attempt, it was tried to run only the node of Pixhawk, of the camera, and of the LiDAR on the Jetson Nano and run in a laptop all the algorithms needed for the drone to be able to navigate autonomously. The communication between devices was via wi-fi. However, the real-time communication was also not possible due to the amount of data that was transmitted between devices (mainly due to the images and the pointcloud). As a result, the only alternative found and available at the time was to connect the drone to a laptop via cable and navigate with the drone by hand. The goal was to put the proposed system's localization to the test and compare the estimated position of the drone with the GPS position. However, in order to receive GPS data from the Pixhawk, the drone must be armed. Since the drone is attached to a laptop via cable, it cannot be armed because the propellers must rotate for the drone to remain armed. Therefore, it was only demonstrated that the localization provided by the system behaves as expected by placing markers on the ground and measuring the distance between them, thus knowing their location, as is depicted in figure 37. Moreover, it was possible to demonstrate the height estimation block as well as the 3-D Hough Transform block.

Two experimental tests were made to test the height estimation block as well as the 3-D Hough Transform block under diferent conditions. Both tests were repeated three times, and the trajectory followed was nearly identical. In both tests, the reference trajectory was start at the same marker (0,0,0), "take-off" one meter and go to the next marker (5,0,0) at the same height, and finaly go to the last marker (5,-5,0). After reach this last marker, it was performed a 180 degrees turn and was performed the same trajectory until the first marker (0,0,0) is reached. The figure 37 shows the experimental test 1, where the estimated position is represented by the yellow line and the reference trajectory by the blue line. It can be observed that the estimated position is almost overlaped to the reference trajectory, which means that the algorithm is working has expected.

Regarding the 3-D Hough Transform block, and looking at the table 6, the floor plane is estimated to be on average 9.9 cm from the real floor plane, while on average the estimated floor plane slope is 11.65 degrees, having a similar behavior to the simulated tests. Experimental test 2 was developed to test the behavior of the algorithm when there are objects under the drone. In figure 38 is represented by the blue line, which represents the estimation of the drone's height over time. The red line is the ground plane observation, which means that

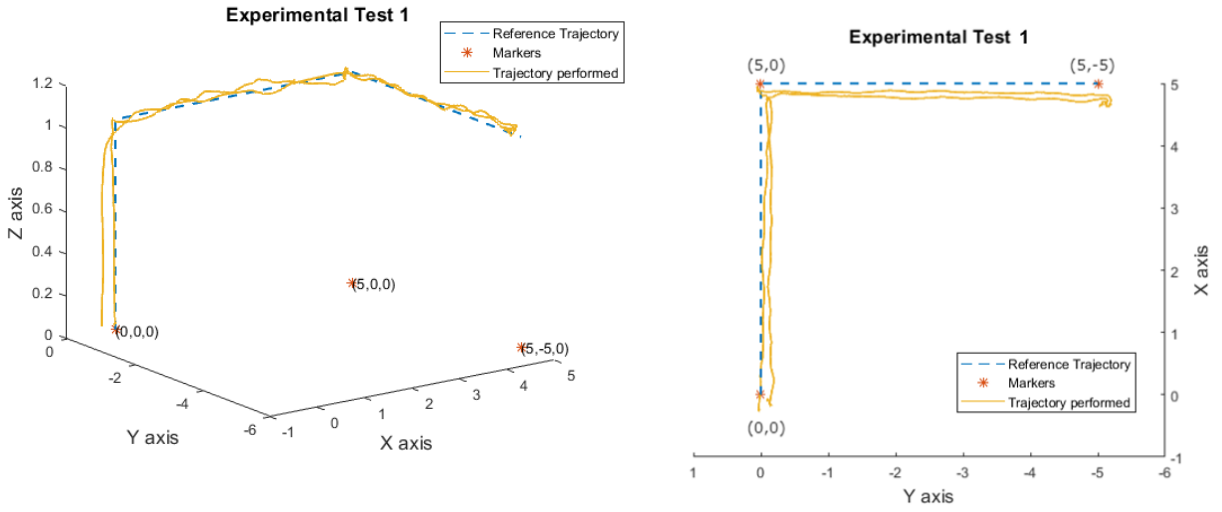


Figure 37: Experimental test 1 to demonstrate the localization of the proposed solution.

Table 6: Results from the 3-D Hough Transform block in the field experiments.

	Distance (cm)	Angle (^{o})
Experimental Test 1	9.90 ± 0.06	11.65
Experimental Test 2	10.10 ± 0.03	16.02

the value represented by this line is the difference between the height provided by the visual odometry and the LiDAR measurements. When this discrepancy rises, it indicates that an item is beneath the drone, and the readings are no longer relative to the ground plane. Figure 38 shows that the object under the drone does not influence the estimation of the drone height, since measurements that are not relative to the first floor plane are discarded.

Regarding the 3-D Hough Transform block, and looking at the table 6, where 'Distance' represents the distance between the origin of the normal vector of the ground truth plane and the estimated ground plane ± std and 'Angle' represents the angle between the normal vector of the ground truth plane and the estimated ground plane. The results are identical to the results from the experimental test 1. From the results shown, it can be concluded that the height estimation is not influenced by objects under the drone, and the ground plane is detected with a small and acceptable error, overcoming the height estimation problem when there are objects under the drone.

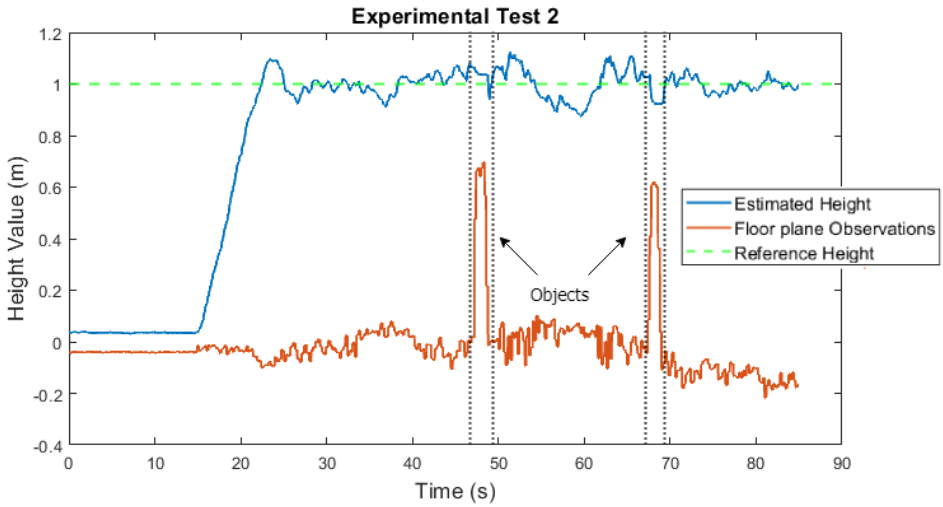


Figure 38: Experimental test 2 was developed to test the behavior of the algorithm when there are objects under the drone.

6 *Conclusion and Future Work*

The research area of UAV navigation in GNSS-denied environments still has some challenges and barriers to tackle. In this dissertation, a step was taken in that direction by presenting an unique approach that provides the drone with the ability to autonomously navigate and explore unknown environments without access to GNSS signals. This chapter summarizes the developed system during the course of this dissertation, bringing the major conclusion of the work to a close. The focus of this project was to create a system that could extract the drone's pose while also mapping the environment using an RGB-D camera, allowing the drone to navigate through GNSS-denied and unknown environments. The approach presented in this dissertation combines the pose provided by the RTAB-Map package with the height estimation of the drone through a LiDAR pointing down. As the drone can fly over objects, the measurements in this situation would be the distance between the drone and the object rather than the drone's height. To address this issue, a 3-D Hough Transform was used to detect the ground floor plane and, as a result, filter observations that are not in respect to the predominant plane, the ground floor plane. As far as the author's knowledge, this dissertation is the first to apply a 3-D Hough Transform for this purpose in a UAV. It is essential to map the drone's surroundings and navigate without colliding with objects in order to fly safely in an unknown environment. As a result, a high-level module was developed that identifies frontier points and decides the next destination based on specific criteria, while also avoiding potential obstacles that may present on the path to that target location.

Two simulation scenarios were created to test the proposed complete system. The simulations were performed to validate the algorithms developed and to demonstrate that the drone can safely navigate without

human intervention in unknown areas without GNSS while also exploring and mapping the environment. The height estimation block, as well as the 3-D Hough Transform block, worked as expected in field experiments. The height estimation is not influenced by objects under the drone, and the ground plane is detected with a small and acceptable error, overcoming one of the previously mentioned problems.

For future work, it would be interesting to implement the micro-ROS framework. The micro-ROS is the robotic framework that connects the gap between resource-constrained and robotic applications. It brings the integration and portability of ROS-based software to microcontrollers. As a result, the system is no longer dependent on a block such as the 'UART node,' which reads data packages byte by byte and then distributes this information in ROS topics. This transformation allows the single computer board to allocate the computing resources allocated to this block to other algorithms. In brief, the Micro ROS allows the microcontroller to publish the ROS topics, making them immediately available for all nodes, accomplishing a more scalable system.

After analyzing the results of the Height Estimation, it is possible to understand that the algorithm can be improved. It was observed that when the measurements are discarded, the height estimation does not follow the ground truth. This event occurs because the LiDAR measurements are more important than the process model and the error of the process model is too high. Therefore, to reduce the error in height estimation, it is necessary to switch the importance of LiDAR observations with the process model when measurements do not correspond to the ground plane.

By interpreting the section 5, it can be noticed that one of the drawbacks of this work is the lack of mechanisms to recover odometry. It is mandatory to create an autonomous process to recover the visual odometry and continue to explore the environment safely and correctly, without human intervention. It is also essential to change the hardware to allow a better frame rate from the camera and better processing. By improving the single board computer on the drone, the camera data will be processed faster and at a higher frame rate, thus decreasing computational time in visual odometry.

References

[1] S. Weiss, D. Scaramuzza, and R. Siegwart, "Monocular-SLAM–based navigation for autonomous micro helicopters in GPS-denied environments," *Journal of Field Robotics*, vol. 28, no. 6, pp. 854–874, 2011.

[2] K. Mohta, M. Watterson, Y. Mulgaonkar, S. Liu, C. Qu, A. Makineni, K. Saulnier, K. Sun, A. Zhu, J. Delmerico, *et al.*, "Fast, autonomous flight in GPS-denied and cluttered environments," *Journal of Field Robotics*, vol. 35, no. 1, pp. 101–120, 2018.

[3] D. Scaramuzza, M. C. Achtelik, L. Doitsidis, F. Friedrich, E. Kosmatopoulos, A. Martinelli, M. W. Achtelik, M. Chli, S. Chatzichristofis, L. Kneip, *et al.*, "Vision-controlled micro flying robots: From system design to autonomous navigation and mapping in GPS-denied environments," *IEEE Robotics & Automation Magazine*, vol. 21, no. 3, pp. 26–40, 2014.

[4] S. Shen, N. Michael, and V. Kumar, "Autonomous multi-floor indoor navigation with a computationally constrained MAV," in *2011 IEEE International Conference on Robotics and Automation*, IEEE, 2011, pp. 20–25.

[5] G. A. Kumar, A. K. Patil, R. Patil, S. S. Park, and Y. H. Chai, "A LiDAR and IMU integrated indoor navigation system for UAVs and its application in real-time pipeline classification," *Sensors*, vol. 17, no. 6, p. 1268, 2017.

[6] G. Balamurugan, J. Valarmathi, and V. Naidu, "Survey on UAV navigation in GPS denied environments," in *2016 International conference on signal processing, communication, power and embedded system (SCOPES)*, IEEE, 2016, pp. 198–204.

[7] M. A. Ma'Sum, M. K. Arrofi, G. Jati, F. Arifin, M. N. Kurniawan, P. Mursanto, and W. Jatmiko, "Simulation of intelligent unmanned aerial vehicle (UAV) for military surveillance," in *2013 international conference on advanced computer science and information systems (ICACSIS)*, IEEE, 2013, pp. 161–166.

[8] F. J. Perez-Grau, R. Ragel, F. Caballero, A. Viguria, and A. Ollero, "An architecture for robust UAV navigation in GPS-denied areas," *Journal of Field Robotics*, vol. 35, no. 1, pp. 121–145, 2018.

[9] S. W. Chen, G. V. Nardari, E. S. Lee, C. Qu, X. Liu, R. A. F. Romero, and V. Kumar, "SLOAM: Semantic LiDAR odometry and mapping for forest inventory," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 612–619, 2020.

[10] S. Nuske, S. Choudhury, S. Jain, A. Chambers, L. Yoder, S. Scherer, L. Chamberlain, H. Cover, and S. Singh, "Autonomous exploration and motion planning for an unmanned aerial vehicle navigating rivers," *Journal of Field Robotics*, vol. 32, no. 8, pp. 1141–1162, 2015.

[11] T. Tomic, K. Schmid, P. Lutz, A. Domel, M. Kassecker, E. Mair, I. L. Grixa, F. Ruess, M. Suppa, and D. Burschka, "Toward a fully autonomous UAV: Research platform for indoor and outdoor urban search and rescue," *IEEE robotics & automation magazine*, vol. 19, no. 3, pp. 46–56, 2012.

[12] H. Oleynikova, C. Lanegger, Z. Taylor, M. Pantic, A. Millane, R. Siegwart, and J. Nieto, "An open-source system for vision-based micro-aerial vehicle mapping, planning, and flight in cluttered environments," *Journal of Field Robotics*, vol. 37, no. 4, pp. 642–666, 2020.

[13] K. Ro, J.-S. Oh, and L. Dong, "Lessons learned: Application of small UAV for urban highway traffic monitoring," in *45th AIAA aerospace sciences meeting and exhibit*, 2007, p. 596.

[14] N. Bolourian and A. Hammad, "LiDAR-equipped UAV path planning considering potential locations of defects for bridge inspection," *Automation in Construction*, vol. 117, p. 103 250, 2020.

[15] M. Eich, F. Bonnin-Pascual, E. Garcia-Fidalgo, A. Ortiz, G. Bruzzone, Y. Koveos, and F. Kirchner, "A robot application for marine vessel inspection," *Journal of Field Robotics*, vol. 31, no. 2, pp. 319–341, 2014.

[16] T. Özaslan, K. Mohta, J. Keller, Y. Mulgaonkar, C. J. Taylor, V. Kumar, J. M. Wozencraft, and T. Hood, "Towards fully autonomous visual inspection of dark featureless dam penstocks using MAVs," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2016, pp. 4998–5005.

[17] M. Achtelik, A. Bachrach, R. He, S. Prentice, and N. Roy, "Autonomous navigation and exploration of a quadrotor helicopter in GPS-denied indoor environments," in *First symposium on indoor flight*, 2009.

[18] A. G. Bachrach, "Autonomous flight in unstructured and unknown indoor environments," Ph.D. dissertation, Massachusetts Institute of Technology, 2009.

[19] R. He, S. Prentice, and N. Roy, "Planning in information space for a quadrotor helicopter in a GPS-denied environment," in *2008 IEEE International conference on Robotics and Automation*, IEEE, 2008, pp. 1814–1820.

[20] M. Achtelik, A. Bachrach, R. He, S. Prentice, and N. Roy, "Stereo vision and laser odometry for autonomous helicopters in GPS-denied indoor environments," in *Unmanned Systems Technology XI*, International Society for Optics and Photonics, vol. 7332, 2009, p. 733 219.

[21] A. Bachrach, S. Prentice, R. He, and N. Roy, "Range–robust autonomous navigation in GPS-denied environments," *Journal of Field Robotics*, vol. 28, no. 5, pp. 644–666, 2011.

[22] S. Grzonka, G. Grisetti, and W. Burgard, "A fully autonomous indoor quadrotor," *IEEE Transactions on Robotics*, vol. 28, no. 1, pp. 90–100, 2011.

[23] K. Celik, S.-J. Chung, and A. Somani, "MVCSLAM: Mono-vision corner SLAM for autonomous micro-helicopters in GPS denied environments," in *AIAA Guidance, Navigation and Control Conference and Exhibit*, 2008, p. 6670.

[24] K. Celik, S.-J. Chung, M. Clausman, and A. K. Somani, "Monocular vision SLAM for indoor aerial vehicles," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2009, pp. 1566–1573.

[25] C. Forster, M. Pizzoli, and D. Scaramuzza, "SVO: Fast semi-direct monocular visual odometry," in *2014 IEEE international conference on robotics and automation (ICRA)*, IEEE, 2014, pp. 15–22.

[26] A. S. Huang, A. Bachrach, P. Henry, M. Krainin, D. Maturana, D. Fox, and N. Roy, "Visual odometry and mapping for autonomous flight using an RGB-D camera," in *Robotics Research*, Springer, 2017, pp. 235–252.

[27] A. Bachrach, S. Prentice, R. He, P. Henry, A. S. Huang, M. Krainin, D. Maturana, D. Fox, and N. Roy, "Estimation, planning, and mapping for autonomous flight using an RGB-D camera in GPS-denied environments," *The International Journal of Robotics Research*, vol. 31, no. 11, pp. 1320–1343, 2012.

[28] T. Dang, C. Papachristos, and K. Alexis, "Autonomous exploration and simultaneous object search using aerial robots," in *2018 IEEE Aerospace Conference*, IEEE, 2018, pp. 1–7.

[29] E. Mueggler, B. Huber, and D. Scaramuzza, "Event-based, 6-DOF pose tracking for high-speed maneuvers," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2014, pp. 2761–2768.

[30] D. Falanga, K. Kleber, and D. Scaramuzza, "Dynamic obstacle avoidance for quadrotors with event cameras," *Science Robotics*, vol. 5, no. 40, 2020.

[31] M. Labbé and F. Michaud, "RTAB-Map as an open-source LiDAR and visual simultaneous localization and mapping library for large-scale and long-term online operation," *Journal of Field Robotics*, vol. 36, no. 2, pp. 416–446, 2019.

[32] T. Mouats, N. Aouf, D. Nam, and S. Vidas, "Performance evaluation of feature detectors and descriptors beyond the visible," *Journal of Intelligent & Robotic Systems*, vol. 92, no. 1, pp. 33–63, 2018.

[33] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," in *2011 International conference on computer vision*, Ieee, 2011, pp. 2564–2571.

[34] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[35] M. Labbe and F. Michaud, "Online global loop closure detection for large-scale multi-session graph-based SLAM," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2014, pp. 2661–2666.

[36] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "G2o: A general framework for graph optimization," in *2011 IEEE International Conference on Robotics and Automation*, IEEE, 2011, pp. 3607–3613.

[37] R. Mur-Artal and J. D. Tardós, "ORB-SLAM2: An open-source SLAM system for monocular, stereo, and RGB-D cameras," *IEEE transactions on robotics*, vol. 33, no. 5, pp. 1255–1262, 2017.

[38] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "Octomap: An efficient probabilistic 3D mapping framework based on octrees," *Autonomous robots*, vol. 34, no. 3, pp. 189–206, 2013.

[39] S. Grzonka, G. Grisetti, and W. Burgard, "Towards a navigation system for autonomous indoor flying," in *2009 IEEE International Conference on Robotics and Automation*, IEEE, 2009, pp. 2878–2883.

[40] D. Borrmann, J. Elseberg, K. Lingemann, and A. Nüchter, "The 3D hough transform for plane detection in point clouds: A review and a new accumulator design," *3D Research*, vol. 2, no. 2, pp. 1–13, 2011.

[41] Y. Tian, W. Song, L. Chen, Y. Sung, J. Kwak, and S. Sun, "Fast planar detection system using a gpu-based 3D hough transform for LiDAR point clouds," *Applied Sciences*, vol. 10, no. 5, p. 1744, 2020.

[42] Y. Tian, K. Liu, K. Ok, L. Tran, D. Allen, N. Roy, and J. P. How, "Search and rescue under the forest canopy using multiple UAVs," *The International Journal of Robotics Research*, vol. 39, no. 10-11, pp. 1201–1221, 2020.

[43] M. Keidar, E. Sadeh-Or, and G. A. Kaminka, "Fast frontier detection for robot exploration," in *International Conference on Autonomous Agents and Multiagent Systems*, Springer, 2011, pp. 281–294.

[44] A. Batinovic, T. Petrovic, A. Ivanovic, F. Petric, and S. Bogdan, "A multi-resolution frontier-based planner for autonomous 3D exploration," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 4528–4535, 2021.

[45] E. Burgos and S. Bhandari, "Potential flow field navigation with virtual force field for UAS collision avoidance," in *2016 International Conference on Unmanned Aircraft Systems (ICUAS)*, IEEE, 2016, pp. 505–513.

[46] J. Borenstein and Y. Koren, "Real-time obstacle avoidance for fast mobile robots," *IEEE Transactions on systems, Man, and Cybernetics*, vol. 19, no. 5, pp. 1179–1187, 1989.

[47] I. Varsadan, A. Birk, and M. Pfingsthorn, "Determining map quality through an image similarity metric," in *Robot Soccer World Cup*, Springer, 2008, pp. 355–365.

Appendices

A *Depth Cameras*

The table below describes some of the investigated and compared sensors to be integrated into the drone. Some of the characteristics evaluated for the choice of sensors were the sensor dimensions, frame rate, price, maximum range, among others.

Table 7: Comparison of different depth cameras.

	Depth F.R. (Fps)	Depth Res. (Pixels)	RGB Res. (Pixels)	Max. range (m)	Dimensions (cm)	Weight (g)	Price (Euro)	Website
Kinect 360	30	320 × 240	640 × 480	3.5	28 × 6.5 × 6.5	680	20.57	[1]
Kinect One	30	320 × 240	640 × 480	4.5	25 × 6.5 × 6.5	1400	66.08	[2]
Xtion PRO LIVE	30	640 × 480	1280 × 1024	3.5	18 × 3.5 × 5	540	784.99	[3]
Carmine 1.08	60	640 × 480	640 × 480	3.5	18 × 2.5 × 3.5	226	60.16	[4]
RealSense SR305	60	640 × 480	1920 × 1080	1.5	13.9 × 2.6 × 1.2	70	69.3	[5]
RealSense D415	90	1280 × 720	1920 × 1080	10	9.9 × 2 × 2.3	72	122.99	[6]
RealSense D455	90	1280 × 720	1280 × 800	10	12.4 × 2.6 × 2.9	389	197.28	[7]
ZED Mini Camera	100	1344 × 376	1344 × 376	15	12.4 × 3 × 2.6	62.9	323	[8]
RealSense L515	30	1024 × 768	1920 × 1080	9	6.1 × 6.1 × 2.6	100	287	[9]
Tara 3D Stereo	60	752 × 480	752 × 480	3	100 × 30 × 35	80.5	82.99	[10]
MYNT EYE	60	752 × 480	752 × 480	18	165 × 31 × 30	184	128.36	[11]