

FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO



Multi-Sensorial Simultaneous Localization and Mapping in Unmanned Aerial Vehicles

João Graça Martins

Mestrado em Engenharia Eletrotécnica e de Computadores

Supervisor: Dr. Marcelo Roberto Petry

Co-Supervisor: Prof. Dr. António Paulo Gomes Mendes Moreira

July 26, 2023

Resumo

A procura por sistemas totalmente autônomos tem crescido nos últimos anos e veículos aéreos não tripulados não são exceção. Existe um elevado número de tarefas que beneficiaria da mobilidade que os drones possuem, como a inspeção de edifícios ou zonas de perigo, a agricultura, entregas, entre outros. No entanto, para atingir a navegação autônoma é necessário assegurar uma estimativa da localização fiável e a capacidade de criar um mapa para planeamento de trajetórias e evitar colisões.

Localização e Mapeamento em simultâneo consiste em se localizar continuamente, criando um mapa. A partir do mapa criado é possível refinar a estimativa da posição levando consequentemente a alterações no mapa criado. Quando aos métodos de localização, sistemas de GNSS são uma das fontes primárias em drones, no entanto estes podem não estar disponíveis em todos os instantes. Assim, a utilização de outros sensores permite ao sistema adaptar-se consoante o ambiente em que este opera. Embora pareça complexo, a realização de localização e mapeamento em simultâneo não é o maior problema, mas sim, a sua realização quando os recursos são limitados, nomeadamente o "payload", a bateria e o poder computacional. Desta forma, é necessário ter estes fatores em consideração no momento da escolha de sensores e de algoritmos para extrair a posição e orientação do sistema e criar um mapa.

A proposta apresentada envolve a instalação de um LiDAR 3D, uma IMU externa, e um computador a um veículo aéreo não tripulado disponível. O sistema aéreo fica assim equipado com sensores, estabelecendo comunicação entre o computador e o computador de bordo. A abordagem proposta utiliza os sensores referidos para estimar a pose do drone e mapear o envolvente. Vários algoritmos com abordagem multi-sensor serão testados, analisados e adaptados a aplicações do mundo real utilizando veículos aéreos não tripulados.

Foram criados três datasets de levantamentos aéreos em um edifício de uma escola, uma torre de água também inserida na mesma escola e uma estrutura de transmissão elétrica onde existem menos características do ambiente para os algoritmos de SLAM. Os datasets contêm dados de um LiDAR 3D, uma câmara monocular, três IMUs e posicionamento recorrendo ao RTK. Os resultados mostram que mesmo em ambientes com menos características, é possível alcançar um erro no posicionamento reduzido, obtendo-se um erro médio de 0.37 a 0.72 metros no posicionamento, quando percorrendo uma distância entre 100 e 250 metros. Quanto à orientação, o erro médio RMS obtido encontra-se entre 0.4 e 5 graus por eixo, o que resulta num erro médio na orientação entre 2.9 a 5 graus. Verificou-se ainda que o erro final verificado entre os algoritmos multi sensor e o sistema de referência é muito reduzido, mantendo-se inferior a 0.48 metros, reforçando a precisão obtida com estes métodos. Adicionalmente, verificou-se que os algoritmos mantêm uma performance semelhante ao longo de todos os cenários de teste.

Abstract

The demand for fully autonomous systems has increased in recent years, and Unmanned Aerial Vehicles are no exception. A whole set of applications would benefit from the mobility of autonomous drones, such as the inspection of infrastructures or danger zones, agriculture, and delivery. Nevertheless, to achieve full autonomy, it is necessary to ensure reliable localization methods and a map for path planning and collision avoidance.

Simultaneous Localization and Mapping consist in continuously localizing itself, sensing the surroundings, and creating a map. Regarding localization, drones primarily depend on GNSS localization systems, which are unavailable in indoor or dynamic environments. Therefore, using other sensors allows the system to adapt to the environments where they are operating. Although it may sound very complex, the main problem is not performing Simultaneous Localization and Mapping but managing to do it under limited payload, battery, and computational resources. Therefore, some factors are to be considered when choosing the sensor configuration and algorithms to retrieve the pose of the system and create a map.

The proposed solution involves mounting a 3D LiDAR, an external IMU, and a computer to a commercially available Unmanned Aerial System. The Unmanned Aerial System is fully equipped with sensors, establishing communication between our computer and the drone's on-board computer. It uses the referred sensors to estimate the drone's pose and map the surrounding. Several multi-sensor Simultaneous Localization and Mapping algorithms will be tested, benchmarked, and adapted to real-world applications using Unmanned Aerial Vehicles.

Three datasets were created from aerial surveying of a school building, a water tower from the same school, and an electrical transmission infrastructure with fewer features. The datasets contain data from a 3D LiDAR, a monocular camera, three IMUs, and RTK positioning. Results show that even in low-feature environments, achieving a low root mean squared error on the positioning is possible, with an RMS error between 0.37m and 0.72m over a covered distance of 100 and 250 meters, on the positioning. On the orientation estimation, the error obtained was between 0.4 and 5 degrees per axis, resulting in a average orientation error of 2.9 to 5 degrees. The drift on the multi-sensor approaches was very low, being the final error between 0.13 and 0.48 meters. Additionally, the algorithms maintained performance across the different test scenarios.

Agradecimentos

Another journey has ended, and I could not forget to thank all those who were part of it and accompanied me, making me what I am today.

I thank my supervisors Dr. Marcelo Petry and Prof. António Paulo Moreira for all the support and help made available, particularly during the course of this dissertation.

To my friends and colleagues who accompanied me during the course, especially Maria and Zé, from study sessions to dinners and pool nights, the towers will always be a meeting place.

Will certainly not forget the 5dpo team, especially Cláudia Rocha and Ricardo Sousa, for always being available and helping. Thanks to the rest of the team for the companionship during meetings and competitions.

A hug to the staff of CRIIS, especially the guys from iilab, for all the friendliness and team spirit with which they welcomed me in the last two years.

I would also like to thank Colégio Diocesano Nossa Senhora da Apresentação for all the availability shown for creating datasets in their facilities and E-Redes for authorizing the mapping of one of the transmission towers.

Last but not least, a thank you to my parents, siblings, and other family for never failing me with anything and supporting me every step of the way.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under Grant Agreement n° 101006798.

Thank you all.

João Martins

*“Engineers like to solve problems. If there are no problems
handily available, they will create their own problems.”*

Scott Adams

Contents

1	Introduction	1
1.1	Context	1
1.2	Motivation	3
1.3	Problem	4
1.4	Objectives	4
1.5	Document Outline	5
2	Background	7
2.1	Sensors	7
2.2	Localization	8
2.2.1	Relative Localization	8
2.2.2	Global Localization	10
2.2.3	Probabilistic Localization	13
2.3	Mapping	15
3	Literature Review	17
3.1	Context	17
3.2	Single Sensor SLAM	18
3.3	Multi-Sensor SLAM	19
3.4	Machine Learning approaches	21
3.5	Conclusion	24
4	Assessment of the influence of magnetic perturbations and dynamic motions in a commercial AHRS	25
4.1	Introduction	25
4.2	Related Work	26
4.3	Tools and Methodology	27
4.3.1	Ground Truth System	27
4.3.2	Magnetometer Calibration	28
4.3.3	Experiments	29
4.3.4	Data analysis	30
4.4	Results and Discussion	31
4.5	Conclusions and Future Work	32
5	System Architecture	35
5.1	Hardware	35
5.1.1	System Components	35
5.1.2	Physical Connections	38

5.1.3	3D Printed Support	38
5.2	Software	40
5.3	Calibration of Sensors	42
6	SLAM Benchmarking to Aerial Vehicles	45
6.1	SLAM algorithms	45
6.2	Datasets	46
6.3	Localization Performance	47
6.3.1	Methodology	47
6.3.2	Results and Discussion	47
6.4	Mapping	51
6.4.1	Mapping Results	51
6.4.2	Loop closure Detection	52
6.4.3	SLAM performance	52
6.5	Conclusions	57
7	Conclusions	59
7.1	Main Conclusions	59
7.2	Contributions	60
7.3	Future Work	60
A	Hardware	61
A.1	OSDK interface cable	61
A.2	LiDAR cable	63
	References	65

List of Figures

1.1	Example of UAVs	2
2.1	Localization with a RTK GNSS system [8]	11
2.2	ICP Point Cloud registration [13]	12
2.3	Kalman Filter Diagram adapted from [23]	14
2.4	Monte Carlo Localization using <i>TurtleBot</i> robot [26]	14
2.5	Feature-based Map	16
2.6	Occupancy grid map	16
2.7	Topological Map - A pose-graph representation [30]	16
2.8	Semantic Map, adapted from [31]	16
4.1	Lab Setup with communications.	28
4.2	3D Printed Handhold support for the IMU.	28
4.3	Magnetic Calibration Results.	29
4.4	Trajectory of the tracking device for scenarios A, B and C.	30
4.5	Orientation estimation X-axis of Test 4	34
4.6	Orientation estimation Y-axis of Test 4	34
4.7	Orientation estimation Z-axis of Test 4	34
5.1	Proposed architecture and communication, adapted from [63]	36
5.2	DJI Matrice 300 RTK used in this dissertation	36
5.3	Inclination of the LiDAR on the drone	39
5.4	Model of the 3D printed support	40
5.5	ROS architecture for UAV surveying	41
5.6	Optimized robot model used in the atom calibration framework	43
6.1	Overview of datasets A and B locations	46
6.2	Dataset C - Aerial image of the school building	47
6.3	FAST-LIO2 SLAM with LiDAR and its internal IMU	50
6.4	Accelerometers of Xsens IMU	51
6.5	Accelerometers of Drone IMU	52
6.6	Accelerometers of LiDAR internal IMU	53
6.7	Mapping performance on dataset A	54
6.8	Mapping performance on dataset C	55
6.9	Mapping noise differences in dataset A	56
6.10	Loop Closure detection on LIO-SAM	57
A.1	OSDK cables connection [70]	61
A.2	Cable pinout of on-sensor receptacle [72]	63

List of Tables

3.1	Summary of 3D Single Sensor SLAM methods	19
3.2	Summary of 3D Multi-Sensor SLAM techniques	22
3.3	Summary of Machine Learning SLAM approaches	23
4.1	RMS Error of orientation estimation of internal filter	31
4.2	RMS Error of orientation estimation using Madgwick Filter	32
4.3	RMS Error of orientation estimation of both filters - Scenario D	33
5.1	Published messages types from the sensor topics	41
5.2	Mean residuals errors for Camera-IMU calibration using Kalibr	43
6.1	Accuracy of each slam algorithm in each dataset	48
6.2	CPU and RAM consumption of the multi-sensor SLAM algorithms	53
A.1	Adapter Board Cable Interface Pinout [71]	62
A.2	Ouster Interface Cable Pinout - Adapted from [72]	63

Abbreviations and Symbols

SLAM	Simultaneous Localization and Mapping
UAV	Unmanned Aerial Vehicle
RC	Remote Controller
ROS	Robot Operating System
LiDAR	Light Detection and Ranging
IMU	Inertial Measurement Unit
INESC TEC	Instituto de Engenharia de Sistemas e Computadores, Tecnologia e Ciência
ANAC	Agência Nacional de Aviação Civil
AAN	Autoridade Aeronáutica Nacional
VIS	Visual Inertial System
VO	Visual Odometry
LIS	LiDAR Inertial System
VIO	Visual Inertial Odometry
LIO	LiDAR Inertial Odometry
ML	Machine Learning
DL	Deep Learning
FOV	Field of View
ICP	Iterative Closest Point
PM	Perfect Match
MCL	Monte Carlo Localization
PF	Particle Filter
VLOS	Visual Line of Sight
API	Application Programming Interface

Chapter 1

Introduction

This chapter presents a general presentation of this dissertation, starting by contextualizing its application in real scenarios, presenting the motivation behind the use of aerial vehicles in multiple applications, and the problems and difficulties in generalizing the use of drones as autonomous vehicles. Then, the objectives to be achieved during the dissertation period are defined, finishing with the contributions achieved.

1.1 Context

Unmanned Aerial Vehicles (UAVs), commonly known as drones, are a thriving technology. Although we can associate these aerial vehicles with amusement, or tools for jobs that require cameras, UAVs have a large number of sensors to maintain stability in flight or perform autonomous takeoffs/landings. They also possess an amazing capability of performing even more challenging movements, depending on their structure, weight, and power.

The integration of autonomous systems in our lives is also becoming more common, either to help humans perform tasks that they cannot perform alone or to execute the ones where it is not possible or safe to be performed by a person. In the case of drones, they can perform some tasks in a more cost-efficient, safe, and fast way in a vast area of applications: from agriculture tasks such as spraying large terrains, to applications in large industries like mapping construction sites, and inspection of vehicles such as airplanes or ships.

According to Otto *et al.* [1] the main drone applications that could be improved to become fully autonomous are presented, ordered by their potential market value:

1. Inspection: Big industries such as energy, roads, oil, gas, and construction, have been recurring to drones to examine future and current construction sites, and periodically inspect facilities to ensure maintenance. Some UAVs can provide, alongside the video recording, high-definition 3D meshes or point clouds, allowing the users to extract measurements and go through every needed detail without having to be hanging by a rope in a wall. As an

example, a company named bladeinsight¹ develops automated aircraft inspections on wind turbine blades.

2. Agriculture: UAVs help perform analysis of the soil properties, the crop health and execute spray tasks for fertilizers and other needed products. Currently there are several reliable solutions on the market, ready to fly, for spray and spreading such as the *DJI Agras T40* (Figure 1.1a) and the *DJI Agras MG-1*.
3. Last mile delivery: Drone delivery is an alternative to reduce the price of home deliveries (DHL developed a fully-automated drone delivery service [2], see Figure 1.1b) and shorter distribution and shipping wait periods. The health sector would also benefit from drone delivery, executing urgent deliveries, like organs to be transplanted, blood and vaccines between hospitals, and medical supplies to remote places where paramedics cannot reach fast enough. Although some challenges are yet to resolve, using medical UAVs could reduce the waiting time for urgent situations.
4. Security verification: Used by insurance companies on accident and catastrophe sites to record scene information while maintaining the safety or by firefighters and police officers for intelligence gathering and situational awareness for informed decision making.
5. Entertainment and Media: Drones can carry high-definition cameras and perform all types of shots, and swarms light shows: With multi-drone coordination, it is possible to create shapes in the sky, e.g. Verge Aero a light drone show on America Got Talent².



(a) Drone for spraying tasks in agriculture³



(b) DHL prototype for drone delivery [1]

Figure 1.1: Example of UAVs

Despite the benefits of using UAVs on these applications, there are many challenges to overcome. Localization and perception represent two of the main challenges, as autonomous UAVs

¹<https://www.bladeinsight.com/>

²https://www.youtube.com/watch?v=X_jaWDs3PGg

³<https://www.dji.com/pt/t40>

require sensors with great precision and systems that estimate with accuracy the pose of the aircraft. With the increase of autonomous aerial systems, the need for more complex path planning algorithms augments as the crash between drones can represent a safety issue for the population, from people and animals to buildings and vehicles.

If, for the presented applications, the use of drones can present multiple benefits, on the other hand, it can represent a threat to the safety of goods and persons as the drones can crash or fall. Therefore, drone operations are regulated, and the legislation in the country defines their use. In Portugal and European Union, theoretical training is required to fly. Depending on the weight or class of the drone and the remote pilot license, there are different safety directives and limitations. In the case of the A1-A3 license, all flights need to be performed at visual line of sight (VLOS), and the maximum altitude from the ground needs to be below 120 meters, or even less, depending on the rules established for some areas. Generally, the only authorization required to fly is from Autoridade Aeronáutica Nacional (AAN) for capturing images, as most drones have cameras onboard. When the drone's weight is greater than 4 kg, it is considered in subcategory A3, which means it cannot fly over people, and the flight must be performed at least 150 meters from buildings and places with high frequency, such as schools, supermarkets, and parks.

In this dissertation, the work to be done relies on the perception component, such as localization and mapping of the surroundings, without yet achieving a fully autonomous system, as complex path-planning algorithms would be required. It will thus be based on the implementation and analysis of different multi-sensor configurations and Simultaneous localization and mapping (SLAM) techniques.

1.2 Motivation

The application of fully autonomous systems in society is growing every day, trying to complement humans or in cases of dangerous jobs replace them. Included in these systems are the UAVs, systems with great potential that are now being equipped with some autonomous operations such as landing, take-off, and follow me, but is not prepared to go from point A to B and be responsive for every scenario.

Simultaneous localization and mapping is a complex technique that contributes to becoming autonomous in every scenario, including unknown environments, as the robot can estimate its pose and creates a map of the surrounding in real time. However, to become autonomous it is also necessary to include path planning, to plan trajectory and deviate from obstacles, and interact with objects and humans.

The problem with most common SLAM algorithms is that they are limited by the sensors they include. In the case of UAVs, the dependency on GNSS localization systems is a problem for indoor or dynamic environments, where it is not possible to have a constant signal. This way, the integration of multi-sensor SLAM not only improves accuracy as the multiple sensors are used to mitigate their intrinsic limitations, and enhances robustness, as drift and biased sensor data can be filtered out.

1.3 Problem

The main problem with autonomous UAVs, and the majority of mobile autonomous systems, is that to estimate the vehicle pose it is necessary to know the operating environment in the form of a map, or use landmarks with predefined positions. This way, the robot cannot operate in unknown environments and the operation in dynamic environments can be difficult to achieve. Simultaneous localization and mapping is an iterative process that allows the robot to continuously update its pose estimate and refines the map based on new sensor measurements. Therefore, the localization is crucial in the creation of the map, as its accuracy relies on the robustness and accuracy of the robot's pose estimation.

For an autonomous system, the use of multiple sensors represents both a solution and a problem: a solution because it gifts the system with redundancy and increases accuracy, and a problem since it requires synchronization between the different sensors and subsystems and the intrinsic and extrinsic calibrations. The main disadvantages are the quantity of information that the system needs to process and store, needing more powerful computers, reducing considerably battery autonomy. On the other hand, the system has the advantage of having more accuracy in its pose estimation and precision in the created map. It also can give the ability to perform even on the failure of one of the sensors, or case of bias or drift on sensors measurements.

From a navigation point of view, estimating the robot's pose is crucial. In most scenarios, the map is created using SLAM to map the environment, and after that, the map is used for path planning and navigation. As some solutions require increased computational power to fuse and process data from all sensors, it became unviable to perform all the time, and since the environment do not change frequently, it is not needed.

1.4 Objectives

Motivated by the above, the purpose of this dissertation is to search, analyze, test, and compare different SLAM algorithms capable of estimating the UAV's pose while mapping the surroundings, using multi-sensor approaches, providing benchmarks between single-sensor and multi-sensor SLAM, and different algorithms.

Also, we wish to contribute with knowledge about the sensors used, the SLAM techniques, and with the datasets captured with information on the most important sensors.

Thus, we can divide this dissertation into sub-objectives:

- Setup the necessary hardware and software that are not included in the drone. Also, design a 3D print support to incorporate the computer and external sensors on flights.
- Integrate the algorithms that present good results in the literature considering the proposed architecture and sensors.
- Creation of multisensor datasets from flights on different scenarios, namely real-world applications.

- Test and compare the implemented techniques, considering the real-time application, using ground-truth systems for localization such as RTK GNSS.

1.5 Document Outline

In addition to the introductions, the remaining dissertation is split into six chapters. Chapter 2 presents an overview of the main types of sensors that can be used, the localization, and the mapping techniques available for mobile robotics. Chapter 3 focuses on analyzing and comparing the available literature, mainly on SLAM algorithms, starting with the elemental techniques and then moving to the complex multisensor algorithms. In Chapter 4 the performance of an Attitude and Heading Reference System from a commercial Inertial Measurement Unit that was later on inserted on the drone for the SLAM algorithms. Chapter 5 presents the system architecture for the SLAM dataset in the UAV as well as system setup. In Chapter 6 are presented the SLAM results from a localization perspective, where the estimated trajectories of each algorithm are compared with the ground truth system, and from a mapping perspective, where the maps are compared. Additionally, the performance of the available Inertial Measurement Units will be taken into account in multi-sensor algorithms on various datasets in order to see if the use of an external Inertial Measurement Unit improves the overall SLAM performance. Finally, Chapter 7 presents the main conclusions of this dissertation on the application of multi-sensor SLAM on UAVs.

Chapter 2

Background

This chapter presents the theoretical fundamentals for SLAM, exploring individually the localization and the mapping problems. These are common problems in mobile robotics applications, therefore multiple methodologies or a combination can be used and implemented to solve the SLAM problem. Thus, it is possible to provide an accurate pose estimation of the system and simultaneously map the surroundings.

2.1 Sensors

Finding reliable results depends on carefully selecting suitable sensors for a given application. The market offers an extensive range of sensors, each with distinct working principles, accuracy levels, sizes, form factors, weights, and communication protocols. Unmanned Aerial Vehicles are already equipped with a significant number of sensors to enable various autonomous functionalities, such as altitude and attitude control, executing predefined flight paths, and performing mapping and localization. These sensors traditionally include but are not limited to Inertial Measurement Units (IMUs), Global Navigation Satellite Systems (GNSS), and cameras.

Depending on the UAV, it is possible to access its sensor data via some Software Development Kit (*SDK*) provided by the UAV's manufacturer. For example, DJI¹ not only provides multiple Development Kit's² to interact with the drone but also different payloads such as LiDAR lasers, high resolution, zoom and infrared cameras. Since SLAM can be performed by resorting to different types of sensors, it is essential to understand what sensors can perform better in dynamic situations and achieve higher accuracy. Below are summarized the principal sensors and some essential characteristics.

Inertial sensors, such as Inertial Measurement Units (IMUs), use inertia and forces to provide information to the system. Most IMUs combine a set of different sensors: an accelerometer, a gyroscope and a magnetometer. Depending on the model, there are a set of important characteristics to consider, such as the range of the sensors (for gyroscope and accelerometer), in-run bias

¹<https://www.dji.com/pt>

²<https://developer.dji.com/>

stability, commonly known as drift, noise density of the data and sensibility of the sensor, factory calibration of the sensor. Finally, another add-on to an IMU is a filter to fuse the available data and estimate the orientation of the sensor.

Cameras can be used as data sources for computer vision algorithms to extract colors, textures, shapes, and features (corners, edges, points in planes, among others) present in the environment. Although cameras may look like one of the best sensors because they can track lots of features at the same time, typically, will require more memory and processing power, and the main downside is its sensitive to light. In dynamic environments, where UAVs operate most of the time, the process of adapting to light conditions changes, can lead to frames where it is not possible to extract any usable feature.

LiDAR's sensors on the other hand do not have the same problem as cameras when it comes to light. However, like any other sensor, it has its own disadvantages: LiDARs do not work properly in the existence of fog or dust as it blocks the light from passing through. LiDAR stands for Light Detection and Ranging, where laser beams are emitted and the sensor measures distances by measuring the time for the reflected light to return to the receiver. There are a good amount of options, some are only 2D with a rotating laser, others 3D, some are 360 degrees, and others have a limited Field of View (FOV). These sensors play an important part in the safety of the emerging autonomous driving technology by detecting people, cars, bicycles, and others, and activating emergency braking if needed. Typically, LiDARs are also included as an option to mount on some drones, however, their high price and weight, must be considered.

Global Navigation Satellite System (GNSS) receptors are also essential sensors usually integrated into the UAVs, using methods such as trilateration to determine the position of the aircraft. The characteristics of a GNSS system are the accuracy, precision, sensitivity, and repeatability of the sensor, the sensitivity, and the output rate.

2.2 Localization

The present section presents the main localization methods used in all types of autonomous systems, operating on the ground, air or even underwater. The localization module includes both relative and global localization. The combination of these methods or the usage of some filters allows the pose estimation of the system.

2.2.1 Relative Localization

2.2.1.1 Dead Reckoning

Dead reckoning is a technique used to estimate the pose (position and orientation) of a robot based upon estimated speeds over elapsed time and course [3]. It relies on integrating sensor measurements, such as wheel encoders or inertial sensors (accelerometers and gyroscopes), to estimate changes in position and orientation over time.

Dead reckoning is typically used when an absolute localization method is not accurate enough or is unavailable, like GNSS on indoor environments or beacon-based environments with increased perturbation. Although it does not improve the overall accuracy, it can reduce the noise covariance. It has also proven itself very useful for providing a higher rate localization is obtained with a simple method, proper to feed the control loop when the global localization method has a lower output rate. However, this technique is prone to error due to sensor noise, motion blur, and light or ground irregularity changes, which can accumulate error over time [3] and lead to drifts in the pose estimation. The Dead Reckoning typically resorts to algorithms such as Kalman Filters to estimate the pose displacement according to the motion model of the system.

2.2.1.2 Visual Odometry

Visual Odometry (VO) is a vision-based method that processes images from single or multiple cameras to estimate the robot's relative pose. Niestér *et al.* [4] explained that this 2-step technique involves feature detection, detection of distinctive features in the environment, and feature matching where feature points are matched between a pair of frames. This way, it is possible to estimate the displacement of the camera and therefore the displacement of the robot.

It is an important method as it can be used both indoors and outdoors or be applied to scenarios where most of the common sensors and methods do not work, like underwater. However, it has some limitations, the motion blur and the light conditions have an important role, and also, the environment depends on distinctive features distributed on the image, as planes and repetitive patterns decrease significantly the algorithm performance. To mitigate these errors, Visual Odometry can be combined with information from other sensors such as IMUs or laser scanners. This method is usually employed on Unmanned Autonomous Vehicles (UAVs) and Autonomous Underwater Vehicles (AUVs), and other non-wheeled systems and is used for navigation and obstacle avoidance.

2.2.1.3 LiDAR Odometry

LiDAR Odometry [5] uses data from a LiDAR sensor that resorts to lasers to scan the environment and generate a point cloud. Then, feature points that are on sharp edges and planar surface patches are selected and used to find feature point correspondence [5].

Similarly to Visual Odometry, LiDAR odometry typically uses algorithms such as feature matching, to identify and track features on the point clouds, or point-cloud registration, which compares and estimates the transformation between two point clouds. By comparing consecutive point clouds, it is possible to estimate the displacement of the LiDAR sensor and therefore the relative motion of the system. One of the main advantages of using LiDAR over cameras is that LiDAR Odometry is relatively robust to motion blur and does not depend on the environment light conditions. However, it presents limitations since the iterative comparison of point clouds can be computationally heavy for denser point clouds.

2.2.2 Global Localization

2.2.2.1 Landmark methods

Landmark-based localization [6] is one of the most used techniques for known environments. It involves the use of beacons in different forms: active or passive. Active beacons transmit a signal using a variety of technologies, including Radio Frequency (RF), Infrared (IR), or ultrasound. On the other hand, passive beacons do not emit any signal and might not require any equipment. Widely used in the industry, these beacons can have a reflective coating, for sufficient contrast relative to the background [6]. Then, localization techniques such as triangulation or trilateration, are used to estimate the vehicle pose. This requires knowing the position of each beacon in the environment, considering a predefined coordinate system.

This localization method is relatively simple to implement and can be used in a wide range of robots. However, it is inherently intermittent, since when moving the source of information is constantly switching between the beacons that are being detected, and the trajectories are limited by the landmarks and the initial position needs to be well-defined, and a "database of landmarks and their location in the environment must be maintained" [6]. Also, landmark localization depends on a structured environment in the case of artificial landmarks, and its use in dynamic environments with many obstacles can be challenging, as it might be difficult to detect beacons or find an accurate solution.

Global Navigation Satellite Systems (GNSS) extends landmark localization, that use GNSS satellites as active beacons. This method implies the computation of the distance to three or more satellites (the more, the better) to perform trilateration, that can provide the robots pose. However, due to the challenges of using GNSS systems in urban areas where there are signal reflections on tall buildings, causing non-line-of-sight measurements, there is a need for more robust GNSS localization methods. Thus, Pfeifer *et al.* [7] introduced a novel sensor fusion algorithm to parametrize the actual distribution online and approximate it with a Gaussian mixture model, achieving superior localization accuracy.

Using GNSS systems for pose estimation in outdoor applications is a common practice [7]. Moreover, its usage on UAVs is even more present since drones use the GNSS signal for most of the autonomous functions, such as Return to Home, since it is a robust and reliable technology that is easy to integrate, provides real-time measurements, and has global coverage. GNSS extends different constellation of satellites, such as Europe's Galileo, the USA's NAVSTAR Global Positioning System (GPS), Russia's Global'naya Navigatsionnaya Sputnikovaya Sistema (GLONASS), China's BeiDou Navigation Satellite System, and India's Navigation with Indian Constellation (NavIC).

In some applications, the standard GNSS localization is not sufficient, as the error can be around some meters. The usage of Real-Time Kinematic GNSS (RTK) can improve the accuracy of the localization. RTK is a high-precision and high-accuracy system that resorts to a fixed ground station to receive the GNSS signal. This way, using this differential system, it is possible to reduce

the error to the sub-centimeter, as the moving system is capable of applying a correction factor to the previous estimation. This operation mode is explained in Figure 2.1.

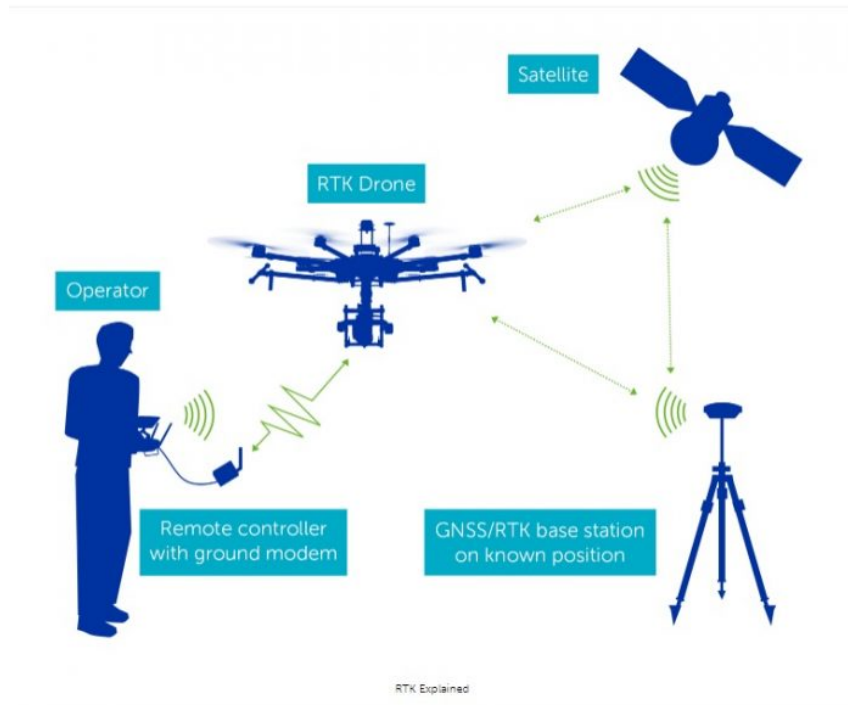


Figure 2.1: Localization with a RTK GNSS system [8]

2.2.2.2 Map Matching methods

Map matching is the process of matching the information being acquired by a system to a digital map to retrieve the system pose. Point cloud localization methods can be used to improve the accuracy and reliability of the pose estimation, particularly where other techniques, such as odometry or beacon-based, are ineffective. These methods can be used to estimate the position of a robot by finding the best match between a point cloud obtained from the robot sensors and the known map. These methods include:

1. **Feature Matching:** identify distinctive features such as corners, edges, and planes, and match them to the known environment (map of the local). Comparing the position of the matched features, it is possible to find the homogenous transformation (position and orientation) that relates to both point clouds.
2. **Point Cloud Registration:** consists of aligning the point cloud data with the known map by minimizing the difference between point clouds. It can be used with resort to Iterative Closest Point (ICP) and the Levenberg-Marquardt ICP method [9–11].

3. Perfect Match: Consists in resolving an optimization problem where it needs to find the best relationship between an existing gridmap and a set of points of a detected line extracted from a vision or range system.

The classical ICP algorithm [9, 11, 12] tries to iteratively find the transformation between a point cloud and a reference, minimizing the error between points, until the matching error is below a threshold, a maximum of iterations is reached or the transformation between consecutive point clouds is out of predefined limits. It is based on gradient descent optimization, which can quickly converge to a local minimum if the initial estimation of the overlapping of the two point clouds is sufficiently close.

Figure 2.2 illustrates the operation of the ICP, wherein in the initial step there are two point clouds misaligned, on the left, and the solution found, on the right.

To solve the above issue, a new implementation - Levenberg-Marquardt ICP method [9, 11, 12] that is based on the same algorithm but uses a different optimization method, the Levenberg-Marquardt to optimize the mean squared error between the pair of point clouds. Wu *et al.* [9] show that this implementation can perform even for low overlapping ratios.

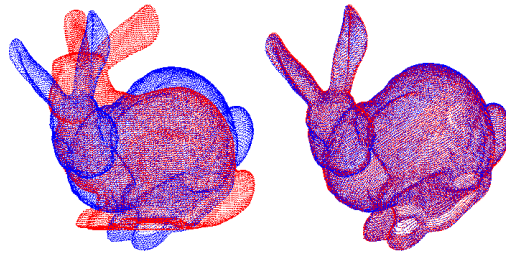


Figure 2.2: ICP Point Cloud registration [13]

Perfect Match [14] was developed as a camera-based localization method for the Middle Size League (MSL) robot soccer competitions. This approach tries to find the best local match between the camera image and the field model which enables robust localization in robots that require high frequency decision and control. The popularity of this technique in MSL led to the implementation of this method even in industrial robots such as the RobVigil presented in [15, 16], where the Perfect Match (PM) was adapted from 2D to 3D using Laser Range Finder data, while maintaining the low computational requirements.

Both PM and ICP methods are very popular in the robotics community as map matching techniques for robot self-localization, but they are very different from one another. Sobreira *et al.* [17, 18] presented a comparison between the two methods on several aspects of robot localization, on two different implementations of ICP, the original version available in PCL and a second version that used lookup tables inspired in the PM approach. Results show that ICP requires fewer iterations to converge when stationary or in a straight line, but the performance degrades when rotating, needing more iterations. The accuracy of both algorithms was similar, with Perfect Match being much lighter and more efficient.

However, map matching methods are not perfect and have some disadvantages in their use. The map matching methods rely on a map, that can be given previously and can be outdated. When the map is being created on the go, there is the problem of accuracy, as loop closure and optimizations to the map were not yet executed. Thus, the use of this map limits the pose estimation accuracy to the same extent and can introduce a matching break that is caused by trajectory outliers, two observations may lead to a disconnected route [19].

2.2.3 Probabilistic Localization

2.2.3.1 Kalman Filters

Kalman Filter [20–22], is a probabilistic algorithm that provides an estimation of some unknown variables given the measurements observed over time. Usually, they have a relatively simple form and require small computational power that can be implemented in real-time systems. Therefore, it is widely used in multiple applications, including robotics.

The Extended Kalman Filter (EKF) [21, 22] is, probably, one of the most used estimation algorithms for nonlinear systems. It is a variant of the Kalman Filter that consists in resolving nonlinear problems by performing linear approximations. However, these approximations may lead to divergence in the estimation and they cannot guarantee optimal results.

The Unscented Kalman Filter (UKF) [21] is another variant of the Kalman, designed for highly nonlinear systems. It achieves better results due to the usage of the Unscented transform (UT), which was developed to address the problems of linearization by providing a more direct method to transform mean and covariance information.

In robotics, one of the main uses of these algorithms is localization or pose estimation. The algorithm and main equations are available as an example in Figure 2.3. Starting with an initial estimation of the system state, for every iteration, a new pose is calculated from the sensors and system available information. Alongside the pose estimation, a Gaussian distribution of the errors is also estimated in the form of mean and covariance. When observations on the system state are available, the Update or correction part is executed.

2.2.3.2 Monte Carlo Localization

Monte Carlo Localization (MCL) [20, 24, 25], or Particle Filter (PF), is a probabilistic robust and efficient approach that solves the global localization and kidnapped robot problem. It can even perform under arbitrary noise distributions and non-linearity [24].

It is based on the idea of using a set of samples (particles) to evaluate the likelihood of the possible hypothesis. The density of particles represents the probability of the solution and can represent also the uncertainty. This technique starts with a random distribution of a set of particles over the robot's known map. Over time, based on the perception of the system, the particles are being redistributed. The robot's pose will be defined by the density of particles. Figure 2.4 shows the application of a Monte Carlo Localization using a *TurtleBot*. However this method requires a

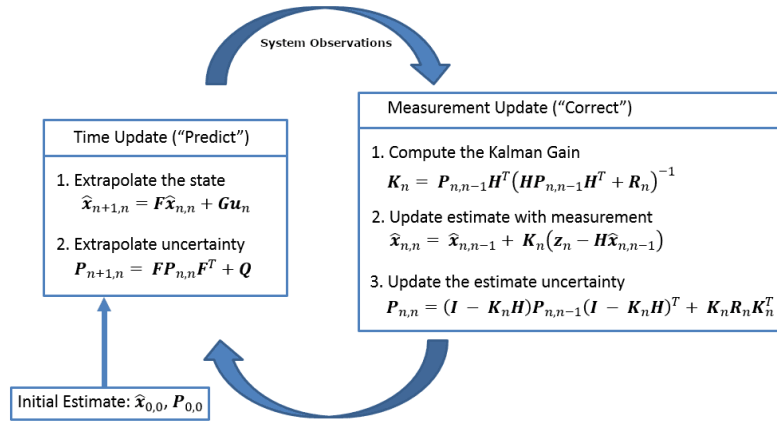
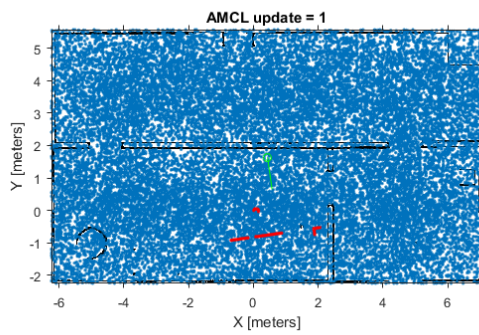
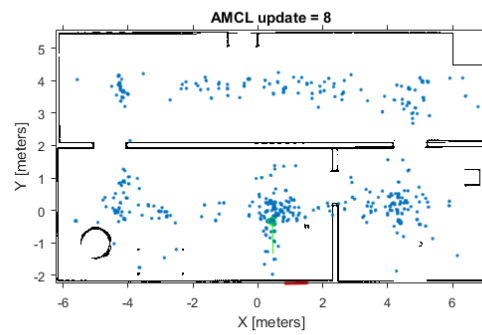


Figure 2.3: Kalman Filter Diagram adapted from [23]

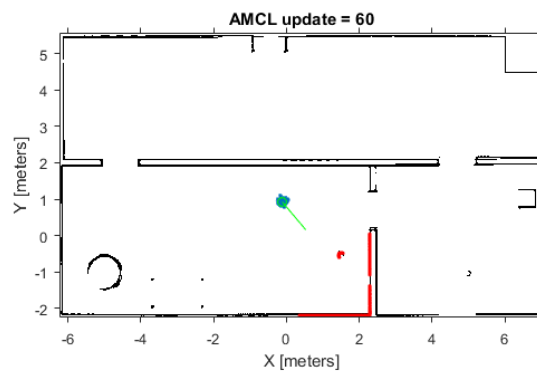
large number (in the hundreds or so) of particles in order to achieve accurate estimations, which cannot be implemented in every system, especially where computational resources are limited.



(a) Initialization of Monte Carlo Algorithm



(b) Monte Carlo Algorithm after 8 iterations



(c) Monte Carlo Algorithm after 60 iterations - found solution

Figure 2.4: Monte Carlo Localization using *TurtleBot* robot [26]

Arulampalam *et al.* [25] have conducted tests using different Particle Filters algorithms to compare the RMSE between the solutions. They also include a detailed description of the algorithms and their implementation.

2.3 Mapping

Mapping is the process of creating a representation of the environment in which the robot operates. It typically involves collecting sensor data from the robot's onboard sensors, and processing that data to build a map. Therefore, it is important to discuss the main mapping methods and in which way they are better suited for some applications.

In robotics, different types of maps can be used for robot perception, navigation, and interaction. Maps are usually classified in four categories: metric maps, topological maps, semantic maps, and hybrid maps [27, 28]: metric maps, topological maps, and semantic maps.

1. Metric Maps: A symbolic structure that encodes the geometry of the environment [28]. Depending on the sensor input data, and the dimension of the mapping procedures [27], maps can be represented in several ways. This representation is one of the easiest ways to locate a robot on the created map, as the environment geometry is preserved.
 - Feature-based mapping: In this method, the system extracts unique features of the environment, such as planes, corners, and beacons [27], and uses them to create a map. The matching of the identified features with the ones on the map can determine the robot's location. On Figure 2.5 is represented a feature map obtained from INESC TEC Robotics Navigation Stack. On yellow there are represented the landmarks that are currently being detected by the robot, in blue the not detected beacons that exist in the database, and in red the points of the 2D LiDAR in the map.
 - Occupancy grid maps: This method is typically used for creating 2D maps, and consists of dividing the environment into a grid of cells. Commonly, each cell represents the probability of being occupied [29]. In the case of 3D, we can extrapolate the occupancy grid and add one more dimension, the *Octrees*. The main advantage of using this representation is the fact that it is easy to perform obstacle avoidance and robot pose estimation on the map. A representation of the floor -1 of the FEUP DEEC building is present in Figure 2.6.
2. Topological Maps: this approach consists of creating a graph-based representation of the environment, in which nodes represent the different locations on the "world" and edges represent the possible paths between these locations. Figure 2.7 represents this approach from a detailed tutorial on graph-based SLAM [30].
3. Semantic Maps: Consists of creating associations between geometric entities on the map, detected by the robot, with high-level information, such as shape, color, and size, to enhance

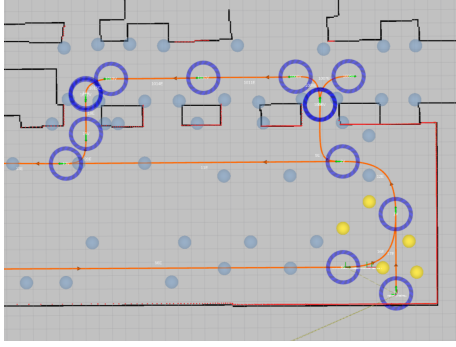


Figure 2.5: Feature-based Map

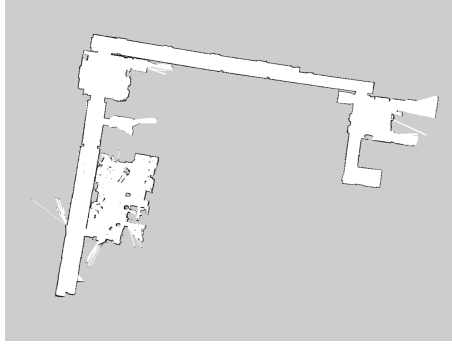


Figure 2.6: Occupancy grid map

the robustness and autonomy of the system as it "facilitate more complex tasks (e.g. avoid muddy-road while driving)" [28]. Figure 2.8 illustrated a semantic representation of the environment where different scene features, such as walls, doors, floors of a building are determined by a semantic labeling and added to the map [31].

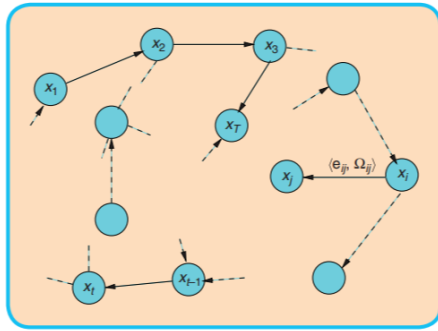


Figure 2.7: Topological Map - A pose-graph representation [30]

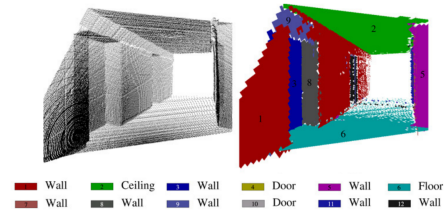


Figure 2.8: Semantic Map, adapted from [31]

4. Hybrid maps: Combine multiple types of representations to create a comprehensive map that includes both geometric and semantic information. Hybrid maps enable robots to have a holistic understanding of their surroundings and support a wide range of robotic tasks. For example, the usage of semantic information on a topological map can be used for a fully autonomous robot, as the semantic information for the decision and task allocation, and the use of topological for navigation.

Chapter 3

Literature Review

This chapter presents the literature review, starting from the beginning of SLAM with the most common and known algorithms, to Multi sensor SLAM algorithms, using various technologies, sensors, and created for different applications. As the objective is to perform SLAM using a UAV, the techniques found will be discussed in order to identify which ones are more suitable and can be used in real-time systems, weighing the sensors used and the computational power. It also presents state-of-the-art SLAM algorithms that introduced some of the methods that are now applied in accurate multi-sensor SLAM techniques, and the ones that, being based on a single-sensor approach, are still able to perform and have created new research lines towards more accurate techniques. Also, some Machine Learning approaches, that emerged on the past few years, to perform SLAM and can resolve some problems of traditional SLAM techniques. Tables 3.1, 3.2, 3.3 resume the main slam methods resorting to single-sensor, multi-sensor and machine learning techniques, respectively, highlighting the sensors that can be used, the strengths, and the weakness of each algorithm.

3.1 Context

SLAM algorithms are in constant change. The first versions were based mainly on one sensor and the features that could be extracted. Later, as new and different localization and fusion methods were published, SLAM saw significant evolutions, creating different approaches, such as lidar and vision-based, inertial, and even machine learning.

In the past few years, with the popularization of 3D LiDARs due to their strong environmental awareness and the increasing improvement of the computational power of embedded processors, 3D LiDAR-based positioning estimation methods have advanced rapidly. With numerous robust and efficient point cloud matching algorithms, and 3D LiDARs providing high-density point clouds [32], it is possible to match consecutive frames to extract the displacement on the sensor/system pose. Furthermore, it can be fused with other sensors, including cameras. However, it comes with the price of high processing demand for applications with real-time sensors.

In the last years, it is possible to encounter a good amount of SLAM reviews ([28] and [32] for example) comparing different algorithms and their performance, and identifying the key modules such as "front-end matching", "closed-loop detection", "back-end optimization", and "mapping" [32]. The problem found with the referred reviews is that they are not as recent and therefore do not include the algorithms from the past two years. Additionally, they are not directed to drones, including algorithms that can only be used in ground robots.

We can classify multisensor SLAM algorithms on loosely and tightly coupled systems according to the types of sensors and fusion techniques employed [32], and extract the pros and cons of each solution. The loosely coupled systems first process the data from each sensor individually, and subsequently, try to fuse all the pre-processed information in a filter that does not give much weight to the current state information in order to estimate the next state. The tightly coupled systems, on the other hand, attempts to merge the data as input of the system globally, combining the characteristics and observations from each sensor to achieve a more accurate pose estimation.

In summary, the loosely coupled system has a simpler architecture and implementation and is computationally easier to produce the results, compromising, however, the overall accuracy. In contrast, a tightly coupled system requires more computational power as its implementation is not trivial, but it can provide an accurate estimation even in the most complex and dynamic conditions.

3.2 Single Sensor SLAM

One of the main contributions to the LiDAR-based 3D SLAM was done by Zhang and Singh in 2014 with its original work "LiDAR Odometry and Mapping in Real-Time" (LOAM) [5], used as the basis for multiple loosely coupled system algorithms. This approach extracts information such as edge and plane feature points and applies scan-matching techniques to estimate odometry at high frequency (10Hz), and point cloud registration for adding the LiDAR scan to the existing map, which is updated every second.

Shan and Englot presented LeGO-LOAM (Lightweight and Ground-Optimized Lidar Odometry and Mapping) [33], as an extension of LOAM [5], designed to be more lightweight and efficient for use on ground robots on a low-power embedded system. It uses the ground plane on the segmentation of the point cloud to filter out the noise and extract distinctive planar and edge features. Then the Levenberg-Marquardt optimization method is used to estimate the 6-DOF transformation between consecutive scans. An Advanced modification of the LOAM (A-LOAM) [5] that uses libraries such as Eigen and Ceres Solver to simplify the code structure was made available. Compared to LeGO-LOAM which is ground-optimized, A-LOAM can potentially work better when the ground plane is not always available and the sensor is moving freely in 3D space.

Although LiDARs are a popular solution for SLAM algorithms, the use of cameras became also popular, mainly when it was possible to use them in real-time on computationally lower systems. ORB-SLAM [34], proposed by Mur-Artal *et al.*, is a real-time feature-based monocular SLAM, for both indoor and outdoor environments, and is resorts of ORB (Oriented FAST and Rotated BRIEF) [35] to extract unique features from the surroundings. It is designed to be lightweight

and efficient, making it suitable for resource-constrained devices, but can even perform with motion clutter and varying illumination conditions.

Table 3.1: Summary of 3D Single Sensor SLAM methods

Name	Sensors Used	Strength	Problems
LOAM (2014) [5]	LiDAR	Low-drift. Low-computational complexity.	Does not include loop closure or backend optimization.
LeGO-LOAM (2018) [33]	LiDAR	Ground segmentation. Two-stage optimization.	Low accuracy in loop closure detection. Complex terrain failure.
ORB-SLAM (2015) [34]	Monocular cameras	Pose-graph optimization using Bundle Adjustment (BA) Fast and accurate initialization	Low-textured environment failure. Low performance compared to more advanced feature descriptors.

3.3 Multi-Sensor SLAM

In the past few years, the multi-sensor slam approach has gained popularity as the need for more accuracy and robustness both in the localization and mapping among robots has increased. Furthermore, the development of tightly coupled algorithms has increased due to the improved accuracy and computational efficiency they offer.

Qin *et al.* proposed "LINS: A Lidar-Inertial State Estimator for Robust and Efficient Navigation" [36]. This algorithm is based on the LeGO-LOAM [33], using an external IMU that is fused with the 3D LiDAR as a tightly coupled system for the LiDAR Inertial Odometry (LIO) module. It uses an Iterated error-state Kalman Filter (ESKF) that not only corrects the measurements but also overperforms state-of-the-art algorithms, deals with the linearization error of the extended Kalman Filter, and is responsible for the LINS superior computational speed. The presented results also show that LINS has low drift over long periods of time, resulting in a real-time, robust, and high-precision system.

Shan *et al.*, authors of LeGO-LOAM, released their LiDAR Inertial Work, LIO-SAM [37]. This tightly-coupled LiDAR Inertial Odometry via Smoothing and Mapping builds the LIO on a factor graph and relative and absolute measurements, including closed loops, are incorporated as factors. To ensure high performance in real-time, old scans are marginalized for pose optimization instead of performing matching of LiDAR scans on the global map. In addition, GNSS positioning can be used to perform long-term drift correction.

LVI-SAM [38] was proposed by Shan *et al.*, and succeeds LeGO-LOAM and LIO-SAM adding the Visual-Inertial System (VIS) to the LiDAR-Inertial System (LIS) designed in a tightly-coupled manner in which VIS leverages LIS estimation to facilitate initialization. The accuracy of the VIS is improved by the depth information of LIS and VIS provides initial guesses for the LIS to perform scan-matching. Also, loop closure is used as a combination of both systems, identified by VIS and refined by LIS. The use of the two systems offers robustness to the system in both texture-less and feature-less environments and is able to perform even in case of failure of one of the two systems.

In 2021, Campos *et al.* [39] proposed ORB-SLAM3 as a "feature-based tightly-integrated visual-inertial SLAM system that fully relies on Maximum-a-Posteriori (MAP) estimation" [39] to perform on visual, visual-inertial and multimap SLAM on monocular, stereo, and RGB-D cameras. These methodologies also perform loop correction when a common area is recognized. Unlike ORBSLAM that only supported monocular cameras, this new version includes visual-inertial, stereo and RGB-D cameras.

Zhang *et al.*, authors of LOAM, proposed "Visual-lidar Odometry and Mapping" (V-LOAM) [40]. An online method that uses visual odometry at high frequency to estimate ego-motion and then a scan matching based odometry from the LiDAR scans to refine the motion estimation and point cloud registration at low frequency.

Wang *et al.* [41] proposed a Visual-Inertial-Laser SLAM (VIL-SLAM) system based on V-LOAM [40] and VINS-MONO [42], that introduced a nonlinear sliding window estimator that tightly integrates pre-integrated IMU measurements with visual observations. A method that starts with visual-inertial to estimate motion and then uses scan matching to further improve the anticipated motion and register the point cloud on the map. To enhance the robustness, the system is designed with modularity to upon sensor degradation, i.e. when illumination changes or texture-less environments are found, or similarly structure-less environments for scan matching, the system adjusts itself and the remaining modules will operate taking on the task of motion estimation.

Xu *et al.* [43] proposed a completely different approach for the LiDAR-inertial Odometry (LIO) estimation, "FAST-LIO2: Fast Direct LiDAR-inertial Odometry" [43]. It consists of registering the scanned points directly on the map without performing any feature extraction. Another novelty was introduced by recurring to an incremental k-dimensional (k-d) tree data structure for maintaining the map. This data structure achieves superior performance and enables incremental updates (insertion and removal of points) and dynamic rebalancing. FAST-LIO2 is faster than the state-of-the-art LIO algorithms while achieving highly competitive accuracy. However, FAST-LIO2 may drift over long distances and does not include loop closure or back-end optimization.

Later, also in 2022, Bai *et al.* presented "Faster-LIO: Lightweight Tightly Coupled Lidar-Inertial Odometry Using Parallel Sparse Incremental Voxels" [44] that was built on top of FAST-LIO2 introducing a new mapping approach based on Incremental Voxels (iVOX). Results show that Faster-Lio can achieve 2 to 3 times faster due to its more efficient mapping technique that has better search and update efficiency.

In 2022, Lin *et al.* [45] proposed R³LIVE, based on their previous work, that consists of

two subsystems: LiDAR-Inertial odometry (LIO) and Visual-Inertial odometry (VIO). The LIO subsystem (FAST-LIO2) resorts on inertial and LiDAR sensors to create the 3D map using an ikd-Tree as a mapping data structure, while the VIO subsystem uses Visual-Inertial sensors to perform texture mapping (i.e., associate color with the 3D points). Moreover, VIO "fuses the visual data directly and effectively by minimizing the frame-to-map photometric error" [45].

3.4 Machine Learning approaches

Machine learning (ML) techniques have been rapidly emerging in multiple applications. Robotics and autonomous systems are no exception, the application of learning techniques on is multiple forms is growing exponentially and providing innovative solutions for multiple challenges. In SLAM, the use of Deep Learning (DL), is competing with the traditional approaches, as this algorithms can now learn to extract different features and being generalized for multiple scenarios, and being a solution for many of the traditional SLAM systems failures, such as lost feature tracks, optimization divergency and drift accumulation [46]. The use of machine learning models, have also been used to process sensor measurements and estimate the trajectory. Although the application of ML in SLAM is still recent and most of the work is not multi-sensor oriented, it is, however, worth highlighting some of the work that is competing with classical algorithms or that, could be integrated into a SLAM algorithm in real-time autonomous systems.

In 2020, Wang *et al.* presented "Tartan VO: A Generalizable Learning-based VO" [47], the first learning-based visual odometry (VO) model. This method is able to learn representations from visual input and use it to estimate camera motion. It aims to being scalable in multiple scenes and camera setups, taking into account two problems: the scale ambiguity from recovering the camera pose from a sequence of monocular images, and take into account the camera intrinsic parameters, without the need of retraining. It is based on two modules, the first one for feature extraction and matching on two consecutive undistorted images, and a second, the Pose network which uses the feature matching results and the camera intrinsic parameters. The problem with this approach is that to achieve scalability for multiple camera setups and environments, it needs a large ammount of data and training.

Nubert *et al.* [48] presented "Self-supervised Learning of LiDAR Odometry for Robotic Applications", which involves machine learning techniques to learn the relationships between LiDAR scans and the robot's pose in an unsupervised manner, which means it does not require ground truth data for the training, to estimate LiDAR odometry. This algorithm then uses the LOAM [5] mapping technique and can operate in real time on a mobile-class CPU, as it is computationally lightweight. One of its main advantages over others learning techniques is that it can be used where ground truth data is difficult to obtain once it uses unsupervised learning. However, it does not perform loop closure with may lead to drift over time.

In 2022, Teed and Deng proposed DROID-SLAM [46]: a Deep Learning-based approach to perform SLAM, in particular, Visual SLAM for Monocular, Stereo and RGB-D Cameras. This novel approach aims to solve the limitation of traditional SLAM approaches by using DL to

Table 3.2: Summary of 3D Multi-Sensor SLAM techniques

Name	Sensors Used	Strength	Problems
LINS (2020) [36]	LiDAR, IMU	Lightweight and high computational speed. Robust and with high accuracy even in challenging environments.	It performs badly as the number of points in the map grows large.
LIO-SAM (2020) [37]	Lidar, IMU, GNSS	Sensor measurements can be incorporated into the framework easily due to the factor graph.	Low loop closure performance.
LVI-SAM (2021) [38]	Lidar, Camera, IMU, GNSS	Sensor measurements can be incorporated into the framework easily. VIS and LIS complement each other, offering robustness and accuracy.	Low loop closure performance.
ORB-SLAM3 (2021) [39]	Monocular, stereo, and RGB-D cameras	Pose-graph optimization using Bundle Adjustment (BA) Fast and accurate initialization	Low-textured environment failure. Low performance compared to more advanced feature descriptors.
V-LOAM (2015) [40]	Lidar, Camera	Still one of the bests algorithms on the benchmarks in terms of average translation and rotation errors.	Low correlation between vision and LIDAR.
VIL-SLAM (2019) [41]	LiDAR, Camera, IMU	Ability to handle sensor degradation - case of failure	Loop closure is unavailable in the case of failure of VIO
FAST-LIO2 (2022) [43]	LiDAR, IMU	High computational speed. Stable even in high angular speed - reliable for fast robots such as UAVs	Does not include loop closure nor backend optimization.
Faster-LIO (2022) [44]	LiDAR, IMU	High computationl speed due to faster map search and insertion	Does not include loop closure.
R ³ LIVE (2022) [45]	LiDAR, Camera, IMU	Capable of performing on LiDAR degenerated and visual texture-less environments. The overall accuracy, as the system can close the loop itself even without back-end optimization.	Real-time usage can be a problem with computationally low-power computers.

achieve higher accuracy and robustness while being able to generalize for multiple scenarios and applications. It uses deep neural networks to extract and process visual features from camera images and simultaneously estimates the pose of camera and reconstructs the environment. The pre-

sented results shown that DROID-SLAM outperformed TartanVO [47] and other DL approaches, and ORB-SLAM [34], ORB-SLAM3 [39], and other classical methods. However, to run in real-time, the system had 2 3090 GPUs achieving 20 to 30 frames per second (fps) depending on the resolution, failing in some datasets with faster camera motion due to low fps number.

Also in 2022, there was two different works related with SLAM that used Deep Learning techniques: "SHINE-Mapping: Large-Scale 3D Mapping Using Sparse Hierarchical Implicit Neural Representations" [49] by Zhong *et al.*, and "LoGG3D-Net: Locally Guided Global Descriptor Learning for 3D Place Recognition" [50] by Vidanapathirana *et al.*. Although they are not SLAM algorithms, they can be used on that area to resolve some of the problems associated with SLAM and its usage on several applications. SHINE-Mapping "enables large-scale incremental 3D mapping" [49] by using sparse hierarchical representations that aims to reduce memory consumption while maintaining accuracy on the environment representation. As SLAM is the process of localization and mapping, this algorithm can be used to map efficiently the surroundings from range sensors such as LiDAR, requiring however the pose to be known as input. The pose can come from different sources, or even be a fusion of multiple sensors and pose estimation techniques. One of the simplest implementation is by resorting to ICP methods, such as KISS-ICP [51] by Vizzo *et al.*, an odometry estimation approach that relies in point-to-point ICP combined with other techniques that perform matching, motion compensation and point cloud subsampling. On the other hand, LoGG3D-Net does not perform mapping, but run-time analysis has proven that can be integrated into SLAM systems as an loop closure detection module [50].

Table 3.3: Summary of Machine Learning SLAM approaches

Name	Sensors Used	Strength	Problems
Tartan VO: A Generalizable Learning-based VO (2020) [47]	Monocular Camera	Ability to use different cameras without retraining	Needs large amount of data and training to be scalable between different cameras and scenarios.
Self-supervised Learning of LiDAR Odometry for Robotic Applications (2021) [48]	LiDAR	Can perform where ground truth is difficult to obtain.	Doesn't use loop closure.
DROID-SLAM: Deep Visual SLAM for Monocular, Stereo, and RGB-D Cameras (2022) [46]	Monocular, stereo, and RGB-D cameras	Strong Generalization - It is trained with monocular input but can run stereo or RGB-D input for higher accuracy without retraining.	It requires multiple high-cost GPUs to run in real-time.

In summary, ML in SLAM represent a new research line and try to solve some of the problems with classical approaches, and advance in the field of visual-based localization and mapping that

is present in multiple applications such as robotics, autonomous navigation and augmented reality. However, it comes with the cost of requiring very high computational power in the form of one or more powerful GPUs, mostly for training but also to run in some of the cases.

3.5 Conclusion

As the scope of this dissertation is to apply multi-Sensor SLAM to UAVs, it is important to test and compare these algorithms against state-of-the-art approaches, understanding if the use of more sensors enables more robustness and accuracy. Furthermore, it is important to consider the limited computational power as the UAVs already have low flight time, and the added weight and power consumption are never desired.

From the analysis of the SLAM algorithms discussed in this dissertation, a few stood out, either by their impressive accuracy results or by the resulted map and their multi-modal application, being suitable for indoor and outdoor environments:

- FAST-LIO2 and Faster-LIO are the LiDAR inertial that show the most promising results. They claim to handle high angular velocities that is expected from a UAV while being computationally light.
- VIL-SLAM presents to achieve good results while maintaining robustness when one of the sensors degenerate due to its modularity.
- R³LIVE claims to achieve higher robustness and accuracy than the others, even in sensor-degenerated situations. Also, the texture mapping applied to the map by the VIO system can lead to map reconstruction that can be very similar to the real environment.
- LIO-SAM claims to achieve high accuracy, using techniques such as loop closure. It also includes the possibility to use GNSS data. Contrary to others, LIO-SAM outputs odometry at high frequency (same as the IMU) which can be useful to feed a control loop.
- LVI-SAM also shows promising results. The implemented architecture based on a factor graph allows the incorporation of more sensors into the system, using the LiDAR information as depth information for the visual system.

Other algorithms, even if not designed for drones, can be implemented and tested. As UAVs are flying, only algorithms that are ground optimized can be discarded, as might occur in situations when the ground cannot be detected by the onboard sensors, leading to errors or drifts that hurt the accuracy of the overall solution.

Chapter 4

Assessment of the influence of magnetic perturbations and dynamic motions in a commercial AHRS

Inertial Measurement Units (IMUs) are a crucial sensor in almost every drone. Contrary to ground robots, the propelling systems cannot provide information about the displacement of an aerial vehicle. Therefore, inertial sensors allow high-rate information that can be used to feed the control loops. On this thought, this chapter presents the work on the published paper [\[52\]](#) that analyzes the performance of Attitude and Heading Reference Systems (AHRS) from a commercial IMU.

4.1 Introduction

In the past few years, the use of Micro-Electro-Mechanical Systems (MEMS) Inertial Measurement Units (IMUs) has been increasing in multiple applications, such as autonomous operation, and motion analysis. An IMU usually consists of accelerometers, gyroscopes, and magnetometers which are used to measure linear acceleration, angular velocity, and the magnetic field or magnetic dipole moment, respectively. These sensors can be integrated with a filter to estimate the orientation of a rigid body, however, the raw data provided are often corrupted by various types of noise and drift [\[53\]](#), leading to errors in the estimation. Depending on the IMU used, a calibration of the gains and offsets are also a key factor to achieve good accuracy. Moreover, the sensors are presented in different price ranges that may vary accordingly to the accuracy, sensibility, noise, or the fact that they are factory calibrated.

The field of Inertial sensors has seen significant advancements in recent years, with a high demand for sensors for accurate motion tracking for multiple applications such as autonomous vehicles or wearables devices. Therefore, not only new and better sensors have been introduced into the market, but a few already include sensor fusion filters for orientation estimation, with Attitude and Heading Reference Systems (AHRS).

In the case of complex systems with multiple sensors and high dynamics, the sampling rate of the system can have an impact on the estimation. Moreover, depending on the fusion algorithm, the sensors can be integrated in a tight or loose way. In summary, an IMU can be integrated into a complex system as a complement to other sensors, by providing an orientation estimation or the raw data to be integrated into a more complex filter that merges the information of every sensor in a tight way.

This work was motivated by the need to include an IMU in an Unmanned Aerial Vehicle in order to perform multi-sensor simultaneous localization and mapping. As the onboard computational power is limited, and the Attitude and Heading Reference Systems process the data on the sensor itself consuming fewer resources, the use of the internal sensor fusion that provides an orientation estimate would be preferred, freeing resources for point cloud and image processing. However, it may come with the price of loss of accuracy, which is also undesired. This way, the presented work evaluates how the chosen IMU filter behaves in multiple situations compared to a state-of-the-art algorithm, to understand how to utilize the available resources in a more efficient way.

The proposed study analyzes and compares the accuracy of orientation estimation using the Xsens Fusion Filter of the Xsens MTi-630 AHRS, an AHRS of an intermediate price range with factory calibration of both accelerometer and gyroscope, over a Madgwick filter¹ available for ROS. The reason behind the use of this filter over the Extended Kalman Filter or other approaches, was based on the results shown by Madgwick [54], where the proposed approach presented better results in both static and dynamic test scenarios, and due to the fact that the Madgwick was already implemented and tested. The proposed assessment compares the results of the Madgwick filter that uses the IMU raw data and the Xsens internal AHRS assess which one can perform better in scenarios with high dynamics, such as estimating the orientation of Unmanned Aerial Vehicles, and with perturbations of the magnetic field. The two filters will be compared with a ground-truth system with a high-accuracy system with an operational area of 11.5 x 10.5 x 3.5 meters.

4.2 Related Work

In this section, the authors analyze the studies available, including the methodology and form of result assessment.

Harindranath and Arora [53] presented a comparison of orientation algorithms in a simulation environment to analyze the impact of introducing different noise inputs in different filters: Madgwick, Mahony, EKF, and a Two-stage extended Kalman Filter algorithm with process covariance matrix Q update (TKF-Q). In their results, the EKF presented the worst performance due to the constant noise assumption, while the TKF-Q is efficient and has a small mean error angle. The other filters in the study had a good performance that fits in between the other two.

Lages and Henriques [55] had compared an external Kalman filter with the filter of a Bosch BNO055 IMU [56]. The results achieved had shown that a generic EKF is capable of matching the

¹http://wiki.ros.org/imu_filter_madgwick

IMU estimation filter. Hu *et al.* [57] presents a two-phase orientation estimation using magnetic, angular rate, and gravity sensors (MARG) with online calibration tested on a low-cost MEMS IMU. In the experimental setup, a strong magnet is introduced to test the online calibration and sensor fusion algorithm. The results show that even for low-cost sensors, a good estimation was achieved with a small error in the heading estimation. However, the conducted study is performed with the Z-axis aligned with the gravity, estimating only the heading angle. Tong *et al.* [58] also presented a comparative analysis between different filters, including a proposed novel adaptive extended Kalman filter, and the Xsens Kalman Filter (internal filter) under different situations. Madgwick *et al.* [54] proposed a gradient descent algorithm for IMU and MARG orientation estimation. The conducted experiments compared the RMS error of the Euler angles in both dynamic and static situations. Ludwig *et al.* [59] presented a study where different filters were compared in a drone, where a Mahony filter outperformed an EKF and a Madgwick filter. The test does not use any AHRS, nor magnetic perturbations are discussed.

The presented study is conducted similarly to the presented in [55] and [58] where a set of trajectories is performed, in this case by hand, but with the addition of a ground truth system that is not present on the mentioned studies. Although the related work analyzes the accuracy of orientation estimation from the IMU or different filters, and the impact of the magnetometer calibration on the estimation, to the best of the author's knowledge, there is not any study that included magnetic field perturbations and tracked the behavior of the Xsens internal filter for orientation estimation under the conditions presented in this case study. The goal meets the work of [57] but analyzes the error in every axis and the movement is not limited.

4.3 Tools and Methodology

This work focuses on measuring and evaluating the accuracy and robustness of the AHRS in the form of the self-contained sensor fusion filter over another filter available in the literature. To achieve the proposed goal, a set of tools were used: *Optitrack* as the ground-truth system, ROS as the interface to record the datasets, and MATLAB for data analysis and plotting graphics.

4.3.1 Ground Truth System

To evaluate the performance of the two filters and compare them, it is necessary to have an accurate tracking system to provide an accurate and robust estimation of the rigid body pose. For this effect, as shown in Fig. 4.1, a setup that includes twelve Prime^X 22 cameras and Motive 3.0 from Optitrack running on the Mocap Computer was used, the tracked pose is sent via Ethernet to the router. The router connects the two computers also via Ethernet. The AHRS is connected to the ROS Master via a USB cable. The PoE switch is used both for communication and powering the cameras.

For tracking, a 3D-printed handheld device to hold the AHRS in place was designed. It contains six passive markers to create a rigid body with a unique spatial relationship, as can be observed in Fig. 4.2. On the calibration of the Optitrack, the mean ray error was of 0.82 mm and the

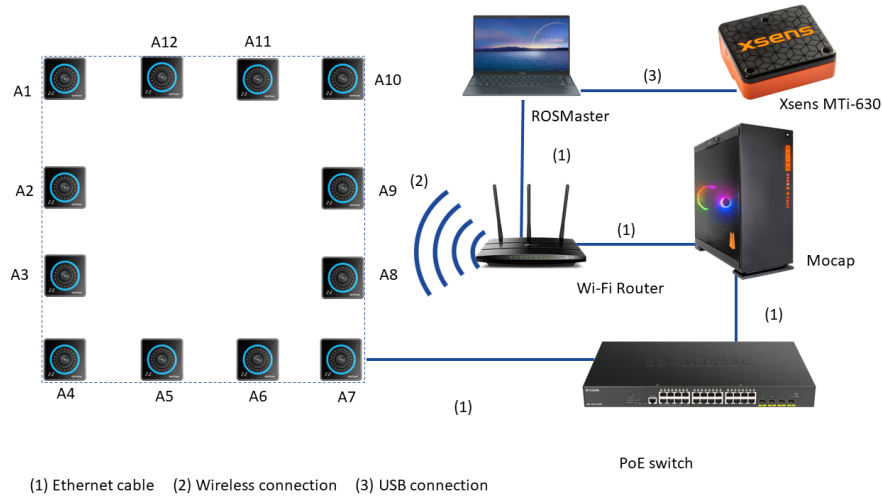


Figure 4.1: Lab Setup with communications.

mean wand error of 0.15 mm. For more information on the values regarding calibration, consult the Optitrack documentation².

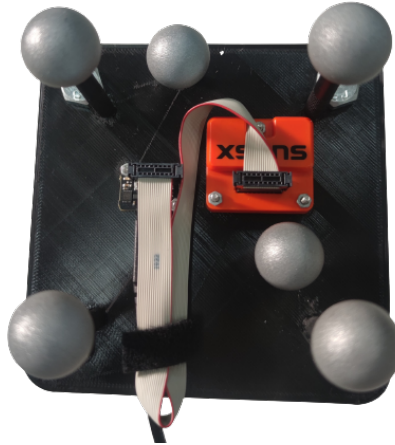


Figure 4.2: 3D Printed Handhold support for the IMU.

4.3.2 Magnetometer Calibration

In order to perform a calibration of the magnetometer, it is necessary to understand the magnetometer model, how the external influences affect the magnetometer measurements, and the calibration procedure. To obtain an accurate estimate, errors in sensing the magnetometer field that comes from onboard hardware and electronics, and external objects with magnetic properties must be mitigated.

The magnetometer influences can be divided into two categories: hard and soft iron. The hard iron corresponds to a permanent deviation on the magnetic field [60] that can be mapped

²<https://docs.optitrack.com/motive/calibration>

as an offset; on the other hand, the soft component is caused by induced magnetization due to the interaction of the permeability of ferromagnetic compounds [60], and affects the intensity and direction of the magnetic field.

Therefore, the calibrated magnetometer values are written as follows, according to the model in [61]:

$$\vec{H}_t = C \cdot S^{-1} \cdot (\vec{H}_m - \vec{b}), \quad (4.1)$$

where $\vec{H}_t$ is the calibrated magnetic field, C for non-orthogonality corrections, S a diagonal matrix containing a scale factor, and $\vec{b}$ as the bias vector, for Hard Iron corrections.

The presented model consists in rotating the IMU containing as many directions of the magnetic field vector as possible and approximating the mapped points to the surface of a 3D sphere with a unitary norm. As we are using a commercial IMU from Xsens, a magnetic field mapper and calibration tool are already available to use, and the full procedure is described in [62]. As it

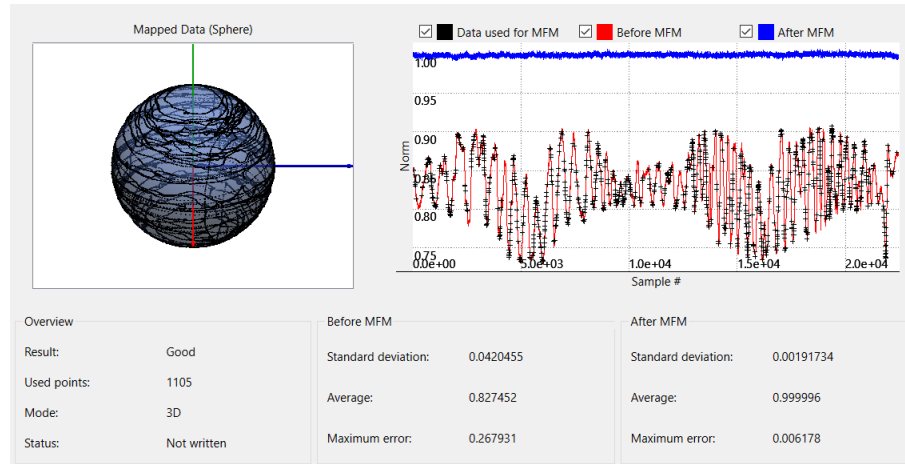


Figure 4.3: Magnetic Calibration Results.

is possible to observe in Fig.4.3, the points of the magnetic field before calibration had a mean relative value equal to 0.827 and a maximum error of 0.268. After the calibration process, the same points were distributed with a unitary mean value and a maximum error of 0.006. This model and the calibration process disregard the value of the magnetic field strength, considering only that the measurements are relative to the local magnetic field intensity.

4.3.3 Experiments

The goal of the presented study is to evaluate the accuracy of the orientation estimation using a generic filter at a high frequency and the Xsens sensor fusion filter with a lower frequency, even under the influence of external magnetic fields that can disturb the results. Therefore, a pre-defined trajectory with multiple rotations is required, in different environments.

For these experiments, four different test scenarios were defined:

- Scenario A - Low external magnetic influences without calibration of the magnetometer.

- Scenario D - A dynamic scenario for rotations in multiple axis with increasing speeds.

4.4, where the rotations are mostly on the Z-axis. It is composed of straight lines and slow 90 degrees rotations. For case D) the goal is to perform rotation in more than one axis at the same

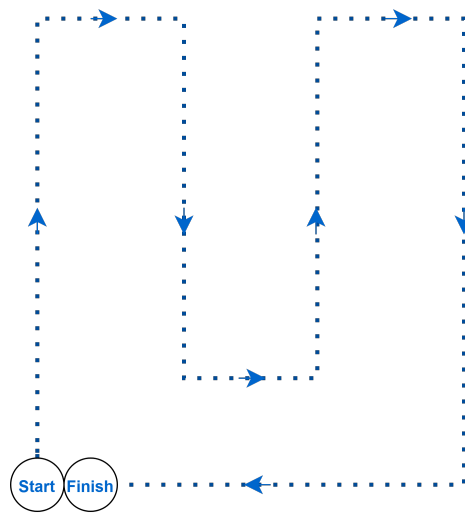


Figure 4.4: Trajectory of the tracking device for scenarios A, B and C.

time and introduce more speed into the rotations. Therefore, for this scenario, the person should walk on a straight line while handling the tracking device in the air with the shape of an 8. These rotations will introduce changes in the measured magnetic field and linear acceleration vectors. For this scenario, the main goal is to understand if dynamics can lead to loss of estimation and if the filters can recover. This way, the movement of a UAV is simulated in order to understand if the application of the AHRS is viable. Each test will be executed two times to ensure the consistency of the results and not draw false conclusions based on a bad run.

4.3.4 Data analysis

For data analysis and to compute comparative metrics, Matlab was used. Since the sensors have different sampling rates, in this case, Opitrack at 360Hz and both the AHRS and external filter at 400Hz, a linear interpolation of the measurements was used in order to ensure that the data being compared was evaluated at the same instant. This way, the processed measurements were presented at 400Hz for direct comparison.

After synchronizing the measurements, it is necessary to transform the measurements to the same referential frame, so they can be compared directly. The internal filter from the AHRS provides orientation referenced to the Earth North-pole; on the other hand, the Madgwick filter

creates its referential based on its orientation on the initialization process. Since Optitrack also has a referential that is defined by its calibration, the retrieved data suffered a rotation defined by the initial value of the orientation quaternion in relation to the Optitrack frame. The mean of the first five measurements of the quaternion from all systems was used in order to find the rotational matrix that relates the two frames, and then transforming all measurements by applying equations (4.2) and (4.3).

$$R_{xsens}^{opt} = R_{orig}^{opt} \cdot R_{xsens}^{orig} = R_{orig}^{opt} \cdot (R_{orig}^{xsens})^{-1} \quad (4.2)$$

$$R_{filt}^{opt} = R_{orig}^{opt} \cdot R_{filt}^{orig} = R_{orig}^{opt} \cdot (R_{orig}^{filt})^{-1}, \quad (4.3)$$

where R_A^B , represents the Rotation of A in relation to B, and the rotation matrix can be obtained by converting the quaternion as follows:

$$R(q) = \begin{bmatrix} 2(q_0^2 + q_1^2 - 1) & 2(q_1q_2 - q_0q_3) & 2(q_1q_3 - q_0q_2) \\ 2(q_1q_2 + q_0q_3) & 2(q_0^2 + q_2^2) - 1 & 2(q_2q_3 + q_0q_1) \\ 2(q_1q_3 - q_0q_2) & 2(q_2q_3 + q_0q_1) & 2(q_0^2 + q_3^2) - 1 \end{bmatrix}$$

The RMS error for the Euler angles was calculated using the following equation:

$$E = \sqrt{\frac{1}{n} \cdot \sum_{i=1}^n |Z_i - X_i|^2}, \quad (4.4)$$

where Z_i is the observation value of the euler angles (using the ZYX sequence) obtained from the ground-truth system on iteration i and X_i is the estimated value of the orientation.

4.4 Results and Discussion

Tables 4.1 and 4.2 summarize the results for every scenario, presenting for each run the root mean squared error in degrees between the filter and the ground-truth system.

Table 4.1: RMS Error of orientation estimation of internal filter

Scenario	Test	RMS Error		
		$\phi_x(deg)$	$\phi_y(deg)$	$\phi_z(deg)$
Scenario A	1	5.13	4.86	3.87
	2	8.86	3.44	3.03
Scenario B	1	7.92	3.45	5.55
	2	3.23	2.41	3.22
Scenario C	1	1.75	2.95	1.98
	2	1.63	1.90	1.71

Table 4.2: RMS Error of orientation estimation using Madgwick Filter

Scenario	Test	RMS Error		
		$\phi_x(deg)$	$\phi_y(deg)$	$\phi_z(deg)$
Scenario A	1	4.68	4.54	3.79
	2	4.88	3.92	4.02
Scenario B	1	5.02	3.26	4.95
	2	3.56	2.31	3.22
Scenario C	1	2.13	3.14	2.14
	2	3.99	4.28	3.88

To begin with, the results are quite satisfactory, and both filters had shown to perform and deliver good results. One of the main goals was to analyze how the magnetometer calibration and influences such as motion dynamics and magnetic field could perturb the results. As it is possible to see in both tables, the differences between scenario A and the others, where the magnetometer was calibrated, are not as high as expected. This can be justified by the existence of the validation module to determine if the magnetic measurements are used or not in the case of the Xsens internal filter and if the magnetic distortion compensation is performing as expected, limiting significantly the impact of distorted measurements. Even in scenario C, where rotations are performed near ferromagnetic objects, the results in both filters had not suffered on the orientation estimation. In test 2 of scenario B it is possible to observe the effect of calibration with RMS errors between 2.41 and 3.22 degrees in the internal filter and 2.31 and 3.56 degrees for the Madgwick. In Scenario C, the results of both tests are better than the results of scenario B, with an error between 1.71 and 2.95 for both tests in the AHRS, and between 2.13 and 4.28 for the external filter. The expectation was to obtain similar or worse values in scenario C than the ones obtained in B. However, when there were valid measurements from the magnetometer, the filters could correct the estimations having slight or none influence on the rotation estimation.

Finally, the last scenario of these experiments consists of performing multiple movements with an eight-shape. Four different tests were executed, where the velocity and acceleration were increased. The RMS error results of this test scenario are registered in Table 4.3. In the performed tests, it was not detected any orientation estimation loss, although there was some error at the point of rotational direction reversal, as it is possible to see in Fig. 4.5, 4.6 and 4.7. Compared to the other scenarios where the movements are slower and more stable, the error in scenario D has increased a few degrees, which is not substantial considering that it is performing multiple rotations.

4.5 Conclusions and Future Work

This section presents a study conducted in order to assess how the Xsens internal filter performs under different scenarios including near magnetic fields, and track the orientation in dynamics

Table 4.3: RMS Error of orientation estimation of both filters - Scenario D

Test	Max ang. velocity	Filter	RMS Error		
			$\phi_x(deg)$	$\phi_y(deg)$	$\phi_z(deg)$
1	2.51 rad/s	Internal	7.76	2.92	5.52
		Madgwick	10.22	4.37	9.35
2	4.76 rad/s	Internal	11.10	3.83	11.62
		Madgwick	11.18	5.08	11.47
3	5.83 rad/s	Internal	6.80	4.85	6.73
		Madgwick	7.05	5.20	6.32
4	7.27 rad/s	Internal	7.70	3.86	7.93
		Madgwick	8.33	6.04	9.78

scenarios and compare with an external filter.

Upon analysis of the experimental results, it became clear that the orientation error found in an indoor scenario with perturbations of the magnetic field is in a dynamic scenario between 5 and 10 degrees. Although it is not desired for every application, considering that is an off-the-shelf device, it becomes easy to implement in any application that needs to estimate its orientation. In the case of a UAV, that can achieve high dynamics, it is important to have a robust algorithm capable to resist even the most adversary conditions.

It is possible to conclude that the internal AHRS tested has shown better results than one state-of-the-art algorithm, the Madgwick filter. For the problem raised, in which the IMU is going to be added to a drone, we can conclude that the AHRS system cannot be used as a unique source for orientation in a system that requires an accurate orientation. However, this AHRS does not require computational power from the onboard computer, which can be used to provide initial estimates of the orientation for other sensors/algorithms and be used on platforms where computational power is already limited. Moreover, in the Xsens datasheet, there is no differentiation in the power consumption of the model with and without AHRS, which indicates that the increased power consumption of the AHRS is not significant considering the computational reduction.

As future work, the rotations could be realized in a controlled way, using a robotic manipulator, maintaining the testing conditions equal in the various tests, and/or a UAV to really measure the error in really high rotation speeds. Also, more datasets with different timeframes could be acquired to analyze the AHRS drift, a more significant batch of tests for each experiment to discard possible outliers and a comparative analysis of more algorithms.

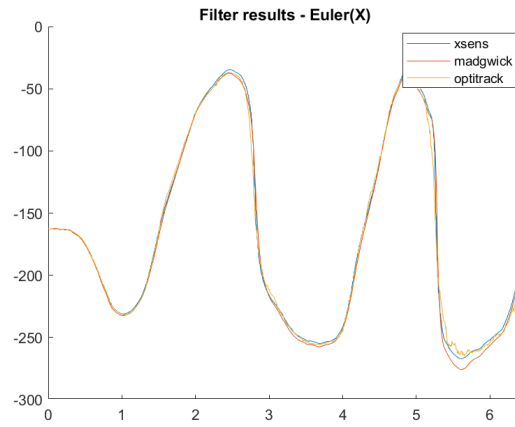


Figure 4.5: Orientation estimation X-axis of Test 4

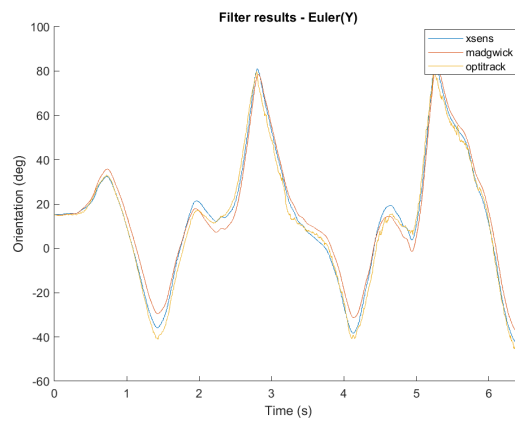


Figure 4.6: Orientation estimation Y-axis of Test 4

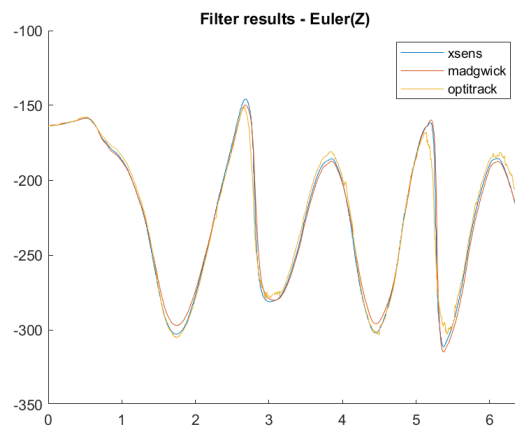


Figure 4.7: Orientation estimation Z-axis of Test 4

Chapter 5

System Architecture

This chapter presents the proposed Unmanned Aerial Vehicle surveying system approach. Starting from the system architecture and integrating the multiple sensors, then presenting the hardware and software that allows using a commercial drone with external sensors. This chapter also presents the tools and frameworks that allow estimating both the sensor's intrinsic and extrinsic parameters.

5.1 Hardware

5.1.1 System Components

The goal of this dissertation was to perform multi-sensor SLAM with an UAV. Developing an aerial platform for UAV surveying will allow to map locals of interest. To accomplish this objective it is required a drone that either have some of the sensors with and API to extract the data or that is capable to flight with the additional weight that will be added to the aircraft. Then, all the sensors needed for the SLAM, such as LiDARs, cameras, IMUs, GNSS receptors, between others, and a computer to perform SLAM online or record the data for offline processing after the flight. The proposed platform is presented on Figure 5.1, integrating the following components:

1. Drone

The drone is the most fundamental constituent of this platform, once it is capable to flight, composed of the engines that sustain it in the air, embedded sensors, and communication and navigation systems. The drone used is a DJI Matrice 300 RTK¹, an enterprise solution from DJI with a max takeoff weight of 9 Kg, including the drone that weights 6.3 Kg. It has two batteries to provide a total time of 55 minutes without payload and 25 to 30 minutes with the configuration of Figure 5.2. It also provides the necessary power to all external components and allow to include different payloads and carry other sensors, and an Application Programming Interface (API) for developers to access telemetry data and camera stream, to be discussed on Section 5.2 with more detail.

¹<https://www.dji.com/pt/matrice-300>

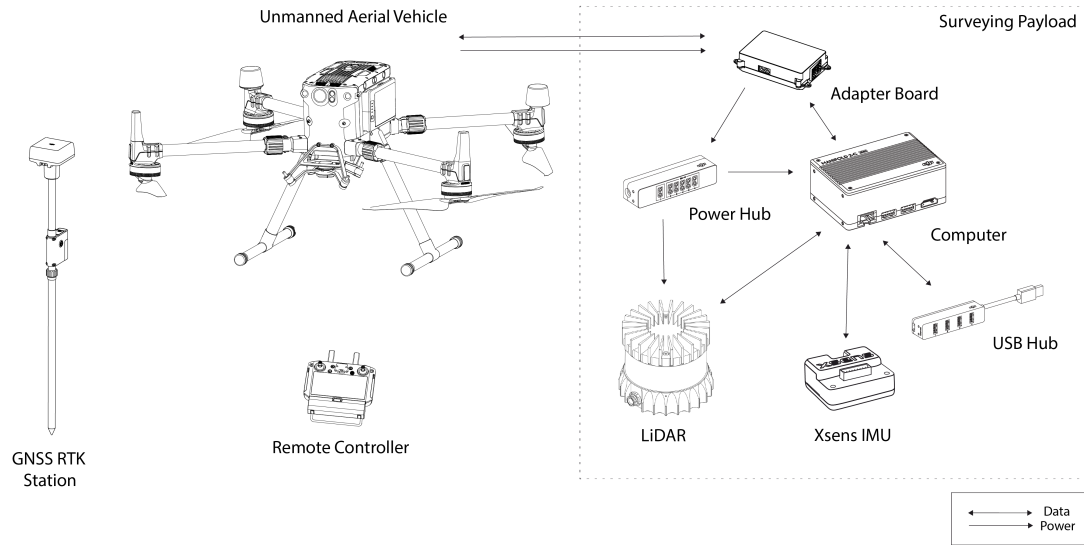


Figure 5.1: Proposed architecture and communication, adapted from [63]



Figure 5.2: DJI Matrice 300 RTK used in this dissertation

2. LiDAR

The use of LiDARs in robotics has been increasing mainly due to their ability to retrieve information about the surroundings regardless of the lighting conditions, reconstructing the environment. However, the weight and power consumption are the main disadvantages that are even more relevant when using this sensor on UAVs since it can significantly affect flight time. On this case, as the UAV allows to carry the weight of a LiDAR and the flight time is

still adequate, it was included a 360° 3D LiDAR from Ouster², with a vertical Field-of-View (FOV) of 45°, and a 10 Hz sampling rate on two versions:

- *OS1-64 RevC* - A 64 beams version with a maximum range of 90 meters on high reflectivity conditions and about 45 meters on low reflectivity such as dark roads and vegetation.
- *OS1-128 Rev7* - A 128 beams version with double of the range of the RevC: 180 and 90 meters on high and low reflectivity conditions, respectively.

3. IMUs

Since the objective of this dissertation is also to discuss the impact of different sensors when performing SLAM with an UAV, it was also included an external IMU in order to access if the use of an intermediate price range could improve the results. On this particular case, since it is possible to access the IMU data from both the drone and the LiDAR internal IMU, comparing the results. For that, an Xsens MTi-630 AHRS³ will be included on the surveying payload. This sensor has a scale that can detect angular velocities within $\pm 2000 \text{ deg/s}$ and accelerations of $\pm 10 \text{ g}$, providing data at 400 Hz. The others IMUs onboard are the drone internal sensor of which we have no information regarding the specifications, only that has an output rate equal to the Xsens, and the Lidar internal sensor that outputs only 100 Hz.

4. RGB Cameras

When it comes to drones, it is fairly easy to associate them with cameras since they usually have cameras mounted in every direction. The DJI M300 is no different, and it has thirteen cameras that are part of the drone vision system, and the gymbal cameras that can be attached. From those thirteen, twelve constitute the pairs of the cameras stereo all around for obstacle sensing and one First Person View (FPV) for navigation, that is transmitted into the remote controller. From all these cameras, we can only access the data of the FPV camera that has a 960p resolution at 30 frames per second, with a FOV of 145°, as an limitation of the DJI API.

5. GNSS Localization

The UAV is equipped with GNSS and RTK localization systems in order to perform autonomous missions that follows waypoints lists and provide high accuracy telemetry. The drone can use the GNSS in two modes, the GNSS enabled, where it only uses the information received on the drone from the satellites and RTK enabled. On the last mode, it is possible to retrieve information with an positioning accuracy of $\pm 1 \text{ cm}$ in horizontal and $\pm 1.5 \text{ cm}$ on vertical.

6. OSDK Expansion Module

²<https://ouster.com/products/scanning-lidar/os1-sensor/>

³<https://www.xsens.com/mti-630>

Trying to create a development tool, DJI has released the DJI Onboard Software Development Kit (OSDK)⁴ and an expansion module that allows to connect to the aircraft both for communication and power, making use of onboard sensors and telemetry data, and eliminating the need to carry extra batteries.

7. Onboard computer and accessories

The onboard computer allows to communicate with the drone, and retrieve data, and record or process the data in real-time. The computer used is a DJI Manifold 2-C⁵ with an Intel Core i7-8550U, 8GB of RAM memory, and 256GB SSD. This computer together with the adapter board and USB and power hubs allow to develop and communicate using Linux native APIs, wrapped in the ROS environment (as it is being currently used), or in systems that support serial communications. Using the OSDK kit allows to access features such as the aircraft state data, flight control, live video feed, camera and gimbal control, between others.

5.1.2 Physical Connections

To access the UAV and LiDAR data, and at the same time power the components, it is necessary to make two cables. The first one connects the UAV to the computer and the power hub via the OSDK port available on the top of the drone, delivering power to the other components, and to communicate with the onboard computer. The second one is a cable to connect the LiDAR to the computer's Ethernet port and to the power hub.

1. OSDK interface cable

The OSDK cable consists of a Type-C female connector with polarity on the UAV side and two USBs (one standard USB-A for data and an FTDI that converts the serial (TTL) to USB, since the Manifold does not have serial port), and an XT60 connector on the payload side. The OSDK port pinout and OSDK interface cable schematics are presented on the Annex A.1.

2. LiDAR cable

In order to use the LiDAR on the drone with only one cable, without the use of the interface box, it was nailed an RJ45 connector for communication and a XT30 to connect into the Power hub to power the sensor. The detailed schematics and pinouts are available on the Annex A.2.

5.1.3 3D Printed Support

As stated initially, one of the goals was to design a new 3D-printed support that could be attached to the drone and hold all the sensors and electronics, improving the existing one. For the design there

⁴<https://developer.dji.com/onboard-sdk/>

⁵<https://www.dji.com/pt/manifold-2/>

were some considerations to be taken, such as keep the IMU the furthest from the power hub for fewer perturbations in the magnetic field, and the fixing points available on the drone. Mounting the support below the drone allows to have a better weight distribution since it is centered while the top mounting points are more at the front of the drone. Considering that one of the main applications of UAV surveying is to map buildings and infrastructures without an abundance of features, as most of the time these infrastructures are on open fields, the capability of the LiDAR to reach the ground is of great relevance. So, in the support design, the LiDAR must be tilted to ensure it is still picking up the ground plane, even at some altitude. However, this tilt must still consider the ability to map what is in front of the drone and the top of the object being mapped.

Considering the ranges that were presented during the LiDAR descriptions in Section 5.1.1, in order to make the best use of LiDAR data and ensure that it is possible to detect enough features from the surroundings or the ground plane, an analysis of the inclination of the sensor in the support was made: Taking by example a 30 m height structure, and that the UAV is 7 m away and at that altitude, what should be the angle that ensures an equal ammount of light beams for mapping the top of the structure and to reach the ground? One of the possibilities is to tilt the lidar 11.25° , wich corresponds to a quarter of the vertical FOV, where at the front of the UAV it is possible to map 11.25° below de horizontal line and the rest above as it is possible to see on Figure 5.3. A higher tilt would ensure to always reach the ground, but would be less effective mapping the front and back of the drone. The resulting 3D printed support model can be seen in Figure 5.4.



Figure 5.3: Inclination of the LiDAR on the drone

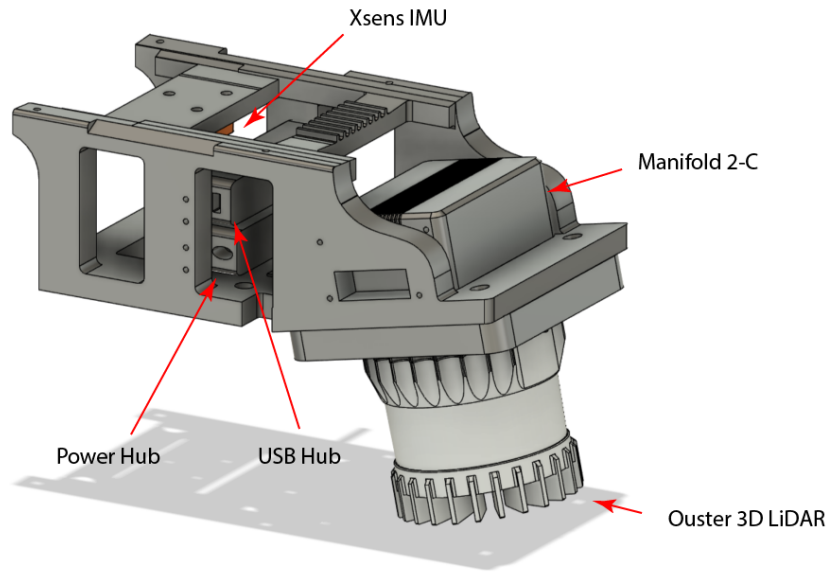


Figure 5.4: Model of the 3D printed support

5.2 Software

With all the sensors mounted and with the needed cables, the software plays an important role, as it is necessary to extract the information from each sensor. For this, ROS is an essential framework, as it facilitates communication. This way, if the SLAM algorithms are developed in ROS, they only need to subscribe the data published by the sensors. Figure 5.5 illustrates the nodes that are running when performing a UAV surveying:

- DJI ROS OSDK⁶ that publishes most of the drone telemetry, such as GPS and RTK position and velocity, camera images, imu, between others.
- Ouster ROS SDK⁷ publishes the point clouds, the internal IMU and the available frames. -
- Xsens ROS node⁸ that publishes imu data and its internal filter orientation estimation.

In Table 5.1 are specified the message types of the topics that are being published by the above nodes. Figure 5.5 represents the software architecture, where it is highlighted the most important information that is being exchanged. The SLAM block represented corresponds to a generic SLAM algorithm that subscribes the topics from the sensors and publish, among other information, the Odometry that contains the drone pose estimation and the map that is being created and incrementally updated, typically in the form of a point cloud.

⁶<https://github.com/dji-sdk/Onboard-SDK-ROS>

⁷<https://github.com/ouster-lidar/ouster-ros>

⁸https://github.com/xsens/xsens_mti_ros_node

Table 5.1: Published messages types from the sensor topics

/imu	sensor_msgs/Imu
/rtk_position	sensor_msgs/NavSatFix
/fpv_camera_images	sensor_msgs/Image
/points	sensor_msgs/PointCloud2
/Odometry	nav_msgs/Odometry
/map	sensor_msgs/PointCloud2

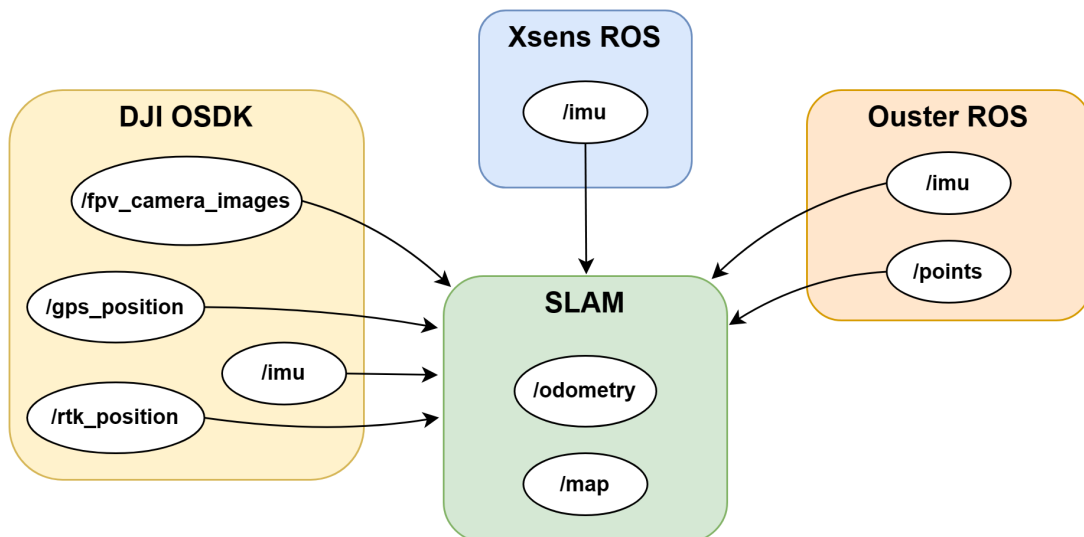


Figure 5.5: ROS architecture for UAV surveying

Using `rosbag`⁹, it is possible to record data from the system and sensors during a flight and reproduce it later to use and test different algorithms and configurations.

5.3 Calibration of Sensors

Sensor calibration is also one of the most important topics during setup in order to perform multi-sensor SLAM. Depending on the sensor, there are different calibrations to be performed:

- **Intrinsic:** Process of determining the internal parameters of a sensor, such as the focal point or lens distortion parameters for cameras, and offset or gain for other sensors such as IMUs.
- **Extrinsic:** Refers to the spatial relationship between two or more sensors, i.e., the relative position and orientation of one frame concerning others.

In our case, the intrinsic calibration refers to the camera, where the focal length, principal point, lens distortion, and other parameters are determined, and the IMU in the form of the noise density and random walk. The extrinsic calibration needs to be performed to estimate the transformation between the LiDAR, camera and IMUs.

In this dissertation it was necessary to use two calibration tools since there is none free/open-source calibration package that can calibrate LiDARs, IMUs and cameras. The first one, Kalibr¹⁰, is able to estimate not only the camera intrinsics but also the camera-IMU extrinsic, which is a major part of our calibration problem. This tool is a two stage process where it starts by estimating the time offset between the different sensors. Then, it takes advantage of defined calibration pattern that makes it possible to express the position of each point in the world coordinate frame, the initial guess of the sensors extrinsic and the intrinsic parameters given as input to solve the transformation between the sensors coordinate frames (camera and IMU) in a continuous estimation optimization problem [64]. Later, the work was extended in order to include extrinsic calibration of multiple IMUs [65]. The only missing estimation was the camera-LiDAR relationship. The second tool used is ATOM¹¹, a calibration framework that can be used for complex systems with multiple sensors, including multiple cameras, LiDARs, and depth cameras, by optimizing the provided robot model. It makes use of a topological representation of the existing coordinate frames in order to calculate the pose of the sensors and of the calibration pattern, resorting to optimization methods, making it applicable to multiple calibration problems [66].

With Kalibr it was possible to estimate the camera intrinsics such as lens distortion parameters, focal length and the optical center. The intrinsic parameters allow to transform the points in the camera coordinate frame and project it into a 2D plane, in the pixel coordinates. Then using the visual-inertial calibration it was possible to compute H_{cam}^{xsens} and $H_{imuDrone}^{xsens}$ as the homogenous transformation between the camera and the Xsens IMU, and between the two IMUs coordinate frames, with the residual errors presented in Table 5.2.

⁹<http://wiki.ros.org/rosbag>

¹⁰<https://github.com/ethz-asl/kalibr>

¹¹<https://github.com/lardemua/atom>

Table 5.2: Mean residuals errors for Camera-IMU calibration using Kalibr

Reprojection error (camera)	0.31767 px
Gyroscope error (Xsens)	0.00914 rad/s
Accelerometer error (Xsens)	0.06850 m/s^2
Gyroscope error (Drone IMU)	0.00624 rad/s
Accelerometer error (Drone IMU)	0.07847 m/s^2

Finally, a URDF model was created, an XML format for representing a robot model, with an initial estimate of the sensors position based on the 3D model of the drone and mounted support. This model serves as input to Atom which outputs an optimized version of the same URDF file from where it was possible to extract the homogenous transform that relates the LiDAR and camera coordinate frames H_{cam}^{LiDAR} . With the three homogenous transformations mentioned, it is possible to compute the transformation between any of the sensors involved in the system. The optimized URDF model can be seen in Figure 5.6, being the FPV camera and 3D LiDAR in the 3D printed suport represented and at its origin the coordinate frames of each sensor. The optimization algorithm converged with average residuals of 0.1327 px for the camera, and 0.0252 m for the LiDAR.

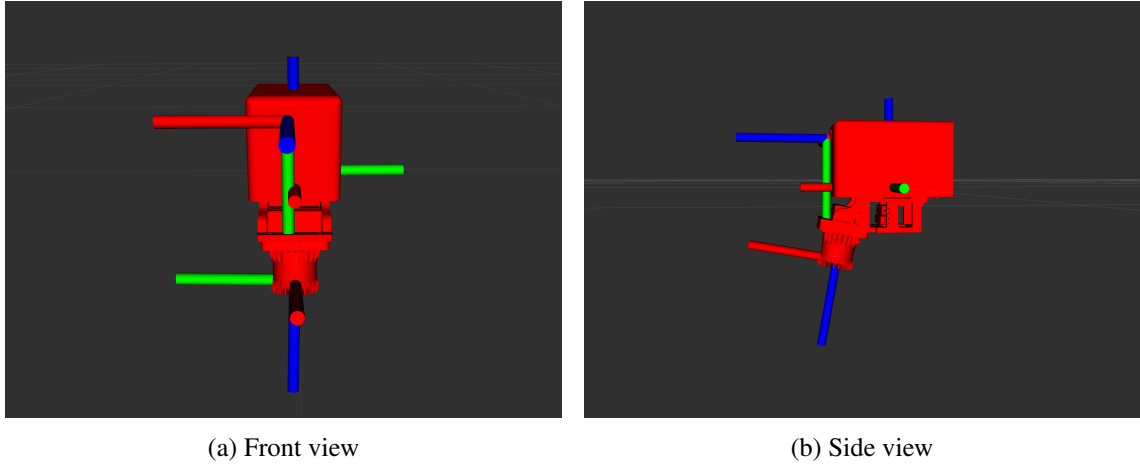


Figure 5.6: Optimized robot model used in the atom calibration framework

Chapter 6

SLAM Benchmarking to Aerial Vehicles

6.1 SLAM algorithms

After analyzing the algorithms presented in the literature review, it is necessary to understand the limitations of the algorithms for the aerial surveying system. One of the goals is to understand the impact of including other sensors. Therefore it is fundamental to test them over state-of-the-art single-sensor SLAM methodologies which are the foundation for most LiDAR multi-sensor approaches. A-LOAM¹ as an advanced implementation of LOAM [5], it will be tested together with Kiss-ICP [51] in order to assess the performance and possible challenges in the use of single sensors with a UAV. Although Kiss-ICP is not a SLAM algorithm, it can estimate pose by matching the current scan and the registered map.

As for the multi-sensor SLAM approaches, some algorithms stood out in particular, FAST-LIO² and Faster-LIO³ showed high potential as they are fast and computationally light, and were developed to handle high rotation velocities being a drone one of their test scenarios. Then, LIO-SAM⁴ that offered a high rate odometry (same rate as the IMU). The three algorithms mentioned are LiDAR-Inertial approaches that resort to IMU for providing an initial estimation for the LiDAR Odometry.

Algorithms such as the VIL-SLAM and the V-LOAM were impossible to test since they are neither open-source nor free to use. Others, such as LeGO-LOAM and LINS, are ground-optimized and therefore are required to detect the ground plane and that the LiDAR is on the horizontal, and for this reason, were not considered in the experiments. *R³LIVE* have an assumption on its implementation that the IMU coordinate frame is coincident with the LiDAR coordinate frame, which does not apply to our scenario. Other algorithms that included a monocular camera, such as ORB-SLAM3 and LVI-SAM, were tested but did not produce valid results. One of the possible reasons for the invalid results is the camera's large FOV (145°). When motion blur occurs

¹<https://github.com/HKUST-Aerial-Robotics/A-LOAM>

²https://github.com/hku-mars/FAST_LIO

³<https://github.com/gaoxiang12/faster-lio>

⁴<https://github.com/TixiaoShan/LIO-SAM>

due to sudden movements, no features are detected for a few frames, leading the vision system to reset, also resetting the pose estimation.

6.2 Datasets

As stated initially in the contributions, three different datasets were created, and made public to the research community. They include data from an Ouster OS1-128 Rev7, the three IMUs, image from the FPV camera, and position and velocity from the GPS and RTK. The datasets are UAV surveys in infrastructures, in this case an electrical transmission tower (dataset A) and a water tower (dataset B) and building (dataset C) from a school. The reason behind having multiple datasets is to analyse the behavior of a few SLAM approaches in different environments.

Dataset A [67] was surveying an electric transmission tower. The surroundings (Figure 6.1a) are mostly cultivated fields and vegetation, with a road and some houses at some distance, which is a poorly structured environment in order to infer if the use of more sensors is actually beneficial and can improve the system overall performance.

In dataset B [68], the mapped structure is a water tower inside a school. Therefore, it is also possible to detect the buildings and trees nearby (Figure 6.1b). This dataset had the shortest distance traveled, of only 100m. Figure 6.1 shows a satellite and aerial image of datasets A and B surveying sites, respectively.

Dataset C [69], represents a flight with a bigger distance, where the drone takes off, and do a full turn around a building (Figure 6.2) and lands near the the take off position. The idea behind this larger dataset is to test the requirements on mapping a larger environment, and access the influency on the performance, as map matching by itself would be computationally heavy. The total distance covered on this dataset was 250m.



(a) Dataset A - Satellite image of electrical tower surroundings



(b) Dataset B - Aerial image of water tower and its surroundings

Figure 6.1: Overview of datasets A and B locations



Figure 6.2: Dataset C - Aerial image of the school building

6.3 Localization Performance

6.3.1 Methodology

On Section 5.2 it was presented all the data that is available during a flight in the form of the ROS messages presented in Table 5.1. As it was already discussed, the RTK GNSS positioning system provides data with an accuracy of $\pm 1 - 1.5\text{cm}$, which can be used as ground-truth for comparing the accuracy of the SLAM algorithms. The goal is to run each algorithm presented on Section 6.1 on the different sensor configurations and compute the root mean squared error of the position and orientation estimations from each algorithm with RTK data as ground-truth.

Concerning the algorithm chosen, the main conclusions to be taken are:

- The addition of an IMU over LiDAR-only algorithms can improve the results in terms of accuracy and robustness?
- Can a external IMU be attached to a UAV provide relevant data and help to produce good results?
- In which conditions can the IMU data be used? What is the impact of a lower sampling rate on the pose estimation?

6.3.2 Results and Discussion

6.3.2.1 Comparing different algorithms

The following results were obtained using the LiDAR and the drone IMU for the LiDAR-inertial algorithms. For each dataset, the RMS error is computed for every algorithm, according to Equation 6.1 and 6.2, in order to understand how the dataset conditions (more or less structured environment) and the sensor configuration influence the accuracy of the UAV pose estimation.

$$E_{pos} = \sqrt{\frac{1}{n} \cdot \sum_{i=1}^n (x_i - \bar{x}_i)^2 + (y_i - \bar{y}_i)^2 + (z_i - \bar{z}_i)^2}, \quad (6.1)$$

where $X_i = \begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix}$ is the position observation obtained from the ground-truth system on iteration i and $\bar{X}_i = \begin{bmatrix} \bar{x}_i \\ \bar{y}_i \\ \bar{z}_i \end{bmatrix}$ is the estimated value of drone's position.

$$E_{ang} = \sqrt{\frac{1}{n} \cdot \sum_{i=1}^n |\theta_i - \bar{\theta}_i|^2}, \quad (6.2)$$

where θ_i is the observation value of euler angles (using the ZYX sequence) obtained from the ground-truth system on iteration i and $\bar{\theta}_i$ is the estimated value of the orientation. Finally orientation error, as the euclidean norm of the individual RMS error, where $\theta = \begin{bmatrix} \theta_x \\ \theta_y \\ \theta_z \end{bmatrix}$.

The results are presented on Table 6.1. The red lines on the table indicate the results obtained until the moment when the algorithm gets lost, at which it stops providing valid estimates without recovering. On the other hand, the green ones highlight the algorithm that had higher accuracy on positioning and orientation.

Table 6.1: Accuracy of each slam algorithm in each dataset

Dataset	Algorithm	Position Error (m)	$\phi_x(deg)$	$\phi_y(deg)$	$\phi_z(deg)$	orientation error (deg)
A	A-LOAM	3.8430	7.66	10.91	3.30	13.73
	Kiss-ICP	1.9704	8.01	10.70	7.11	15.14
	Fast-LIO2	0.3776	2.54	2.17	0.40	3.36
	Faster-LIO	0.3702	3.14	2.25	0.39	3.88
	LIO-SAM	1.2012	2.15	1.44	3.93	4.71
B	A-LOAM	1.3511	7.83	11.92	1.46	14.34
	Kiss-ICP	0.6640	7.94	12.09	1.61	14.55
	Fast-LIO2	0.4207	1.21	1.03	0.45	1.66
	Faster-LIO	0.3668	1.51	1.66	0.43	2.28
	LIO-SAM	0.7170	1.75	1.17	2.28	3.10
C	A-LOAM	0.5599	7.80	14.39	11.73	20.14
	Kiss-ICP	0.5276	7.67	13.59	8.17	17.61
	Fast-LIO2	0.3626	1.72	3.06	0.41	3.53
	Faster-LIO	0.3679	2.50	4.93	0.38	5.54
	LIO-SAM	0.5481	1.34	0.91	2.50	2.98

Most of the results reinforce the thesis that the use of multi-sensor improves the accuracy results, with both positioning and orientation results being more accurate on these algorithms. The difference between dataset A and B is that in the scenario B, the surroundings are structured and all the walls, trees and roofs can be used for matching, obtaining a better estimation of the UAV pose. On the other hand, dataset A is represented by the tower, the ground, and a few trees or houses at the distance. Thus it was expected that the single-sensor algorithms could fail, which happened from both (A-LOAM and Kiss-ICP). It is also worth pointing that before getting lost the accuracy was already bad, jumping between solutions on the pose estimation and creating multiple towers on the map.

When it comes to multi-sensor algorithms, all of them were capable to provide a continuous estimation without failing. It is possible to state that Faster-LIO and FAST-LIO2 provide better results than LIO-SAM. Additionally, since FAST-LIO2 is the base for Faster-LIO, changing mainly the mapping approach, it was expected that the accuracy of both algorithms would be similar.

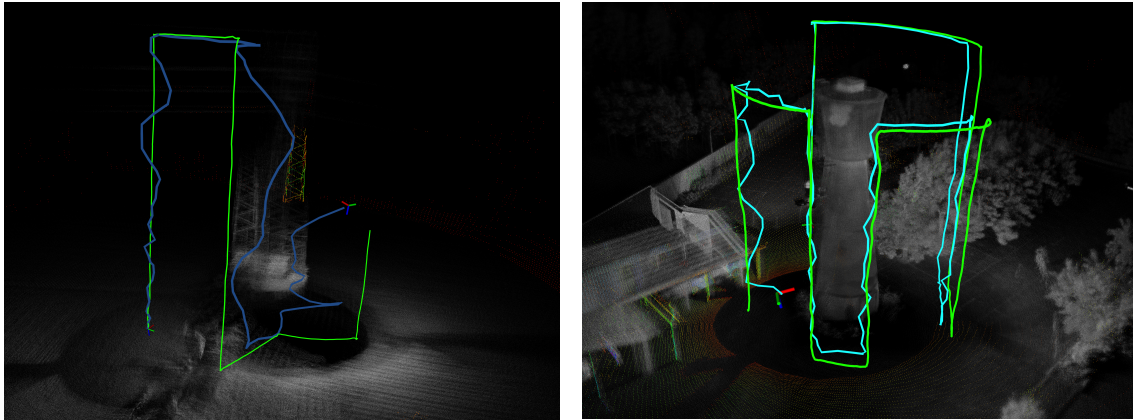
On dataset C, as there is a bigger area to map and a greater distance traveled, it was expected that the system accuracy could suffer, which did not happen. On the positioning accuracy, all the algorithms had similar performance on this dataset, however, the algorithms that included the IMU had significantly better results on the orientation accuracy.

In general, it was also possible to achieve a low drift, as the last positioning error was considerably low. FAST-LIO2 and Faster-LIO achieved a final error on dataset A of 0.1307 and 0.1271 meters, respectively, while LIO-SAM had a higher drift: a final error of 4 meters. In dataset C, the errors were 0.1764 m using FAST-LIO2, 0.2100 m with Faster-LIO, and 0.4728 m with LIO-SAM. In conclusion, FAST-LIO2 and Faster-LIO were more robust to the occurrence of drift, maintaining a higher accuracy across the test scenarios.

6.3.2.2 Comparing different IMUs

One of the goals of this dissertation was to compare and assess if including an IMU in a drone to perform multi-sensor SLAM improves the pose estimation. After running the datasets with different sensor configurations with the FAST-LIO2 algorithm, it was visible that the SLAM with the LiDAR IMU was not as good as the others. This difficulty by the LiDAR internal IMU can be justified by the lower sampling rate, which is only 100 Hz on the LiDAR IMU and 400 Hz on the other two. This difficulty was more present in dataset A since the LiDAR odometry had fewer features, and the SLAM algorithm needed more IMU than in dataset B. On dataset B, it is still possible to see that the LiDAR IMU results are inaccurate. However, since there were more features for the LiDAR odometry estimation, it was still possible to estimate the trajectory, although with more error. In this case, the predicted trajectory had an accuracy of 0.9283 meters which was visible on the resulting map. It was impossible to keep up on dataset A, and the algorithm diverged immediately. It can be seen in Figure 6.3, where the FAST-LIO2 was used with the LiDAR internal IMU on dataset A on the left and dataset B on the right, where in green it is represented the actual trajectory obtained from the RTK system, and in blue the pose estimation

from the algorithm. On the other hand, the Xsens IMU could keep up with the drone IMU in both scenarios.



(a) Dataset A - the algorithm drifts and cannot estimate the trajectory for long

(b) Dataset B - the algorithm can estimate the trajectory with considerable error

Figure 6.3: FAST-LIO2 SLAM with LiDAR and its internal IMU

When testing the same sensors but with LIO-SAM, neither the Xsens nor the LiDAR internal IMU could be used as the algorithm started drifting. As stated before, LIO-SAM takes advantage of the IMU to provide a higher rate of odometry estimation that uses the IMU on every measurement updated with the LiDAR odometry at a lower rate. Figures 6.4, 6.5 and 6.6 shows the acceleration measured by each IMU with respect to the drone coordinate frame during take-off. As it is possible to see, all sensors have their measurements with small variance. Then, when the drone takes off, the sensor measurements are now disturbed by an external source of noise, the vibrations due to the propellers rotation. DJI claims that the DJI M300 RTK contains dual industrial grade IMU providing vibration and temperature stability in harsh environments, indicating that there is mechanical absorption or digital filters in the DJI IMU to cut the vibrations frequencies.

In conclusion, it is possible to use an external IMU by itself, however, a mechanical absorption kit would be recommended in order to get the best out of the sensor. Also, DJI announces that the drone can have rotations up to $300deg/s$ in roll and pitch and $100deg/s$ in yaw. After analysing the gyroscope of the IMUS, the maximum angular velocity was below $100deg/s$. Thus, there is no need to include IMUs with a scale for $2000deg/s$ with the cost of losing sensitivity. Additionally, the higher the sampling frequency, the best it is possible to detect the high rate dynamics of a drone.

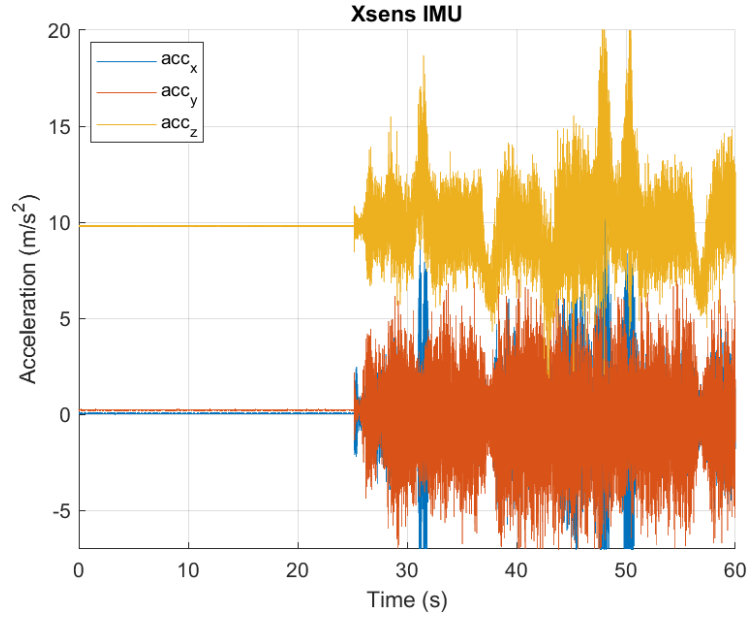


Figure 6.4: Accelerometers of Xsens IMU

6.4 Mapping

6.4.1 Mapping Results

As it is well known, when performing SLAM, the mapping performance is directly dependent on the pose estimation accuracy, in order to transform the current scan to the global map. Considering the results presented above, it is expected that the mapping process was also well succeeded. Figures 6.7 and 6.8 present, respectively, the mapping results on dataset A and dataset C for FAST-LIO2, Faster-LIO and LIO-SAM.

From the mapping results, it is possible to see that the results of all algorithms in dataset A are very similar. However, the mapping from LIO-SAM looks sharper and with less noise. In particular, when zooming in the electrical lines, FAST-LIO2 and Faster-LIO have some points outside the lines which does not happen in LIO-SAM, which can be seen in Figure 6.9.

On dataset C, around the school building, it is possible to see the opposite. The map from FAST-LIO2 has conserved more properties of the building, visible on the windows, revealing also a denser point cloud. However, the map resulting from FAST-LIO2 and Faster-LIO have the motion blur of the remote pilot that was walking, which did not happen on the LIO-SAM. This phenomenon should be mitigated as those algorithms evaluate if the points are moving before adding them to the map. Furthermore, since it is being used a 360° 3D LiDAR, the pilot can be in frame multiple times so ensure the safety of the flight, which is not desirable to appear in the map.

In conclusion, LIO-SAM can deliver accurate maps that visually appears to have low noise. One of the differences between LIO-SAM and the rest, is that LIO-SAM performs de-skewing of the point clouds using inertial measurements in order to undistort the point clouds, caused

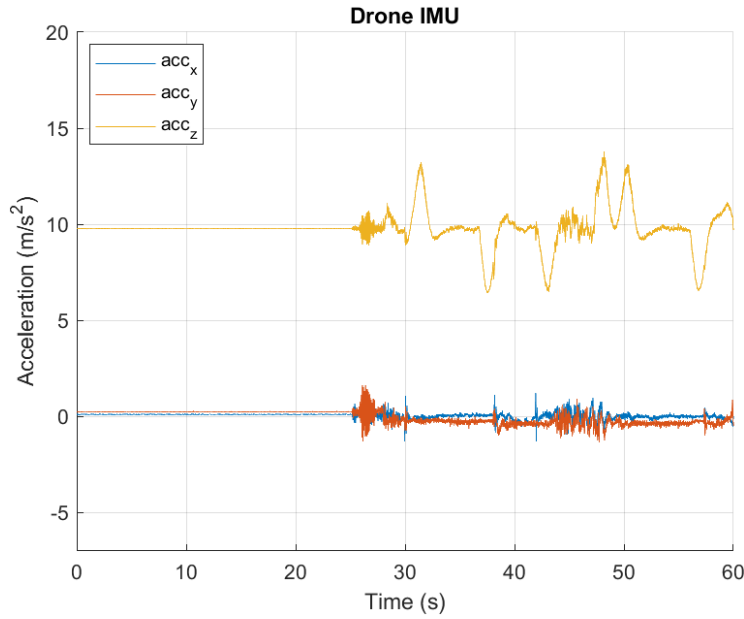


Figure 6.5: Accelerometers of Drone IMU

by ego-motion when mapping the surrounding area. This undistortion may justify the difference between having or not points between the electric lines on the tower.

6.4.2 Loop closure Detection

In dataset C a full turn around of a building was performed, and it ended near the take-off point. Since the multi-sensor SLAM algorithms were able to maintain a low error on its pose estimation and a low drift, the impact of loop closure it is not as significant as situations with more drift. Nevertheless, LIO-SAM shows on the rviz interface the loop closure detections, where loop-closure factors are added to the graph. The authors state that the loop closure factors are useful to correct the drift mainly on the altitude. Figure 6.10 shows two moments of loop closure detection in dataset C, where Figure 6.10a depicts the detection that connects the first and last nodes on the factor graph, and Figure 6.10b the loop closure when the drone changed its altitude.

6.4.3 SLAM performance

Considering that SLAM might be performed in real-time in a real UAV, it is essential to analyze the computational resources demanded by each algorithm from both CPU and RAM. FAST-LIO2 was originally tested in a drone using a DJI Manifold 2 (Intel Core i7-8550U, 8GB of RAM), the same computer used in this dissertation. However, the LiDAR used in FAST-LIO2 has limited horizontal FOV, while on our datasets, a 360° LiDAR was used, which requires more than 8GB of RAM in all datasets, due to its large number of points. Therefore, the presented results were obtained using a desktop with an Intel Core i5-11600 2.80GHz and 48GB of RAM DDR4. Table 6.2 resumes

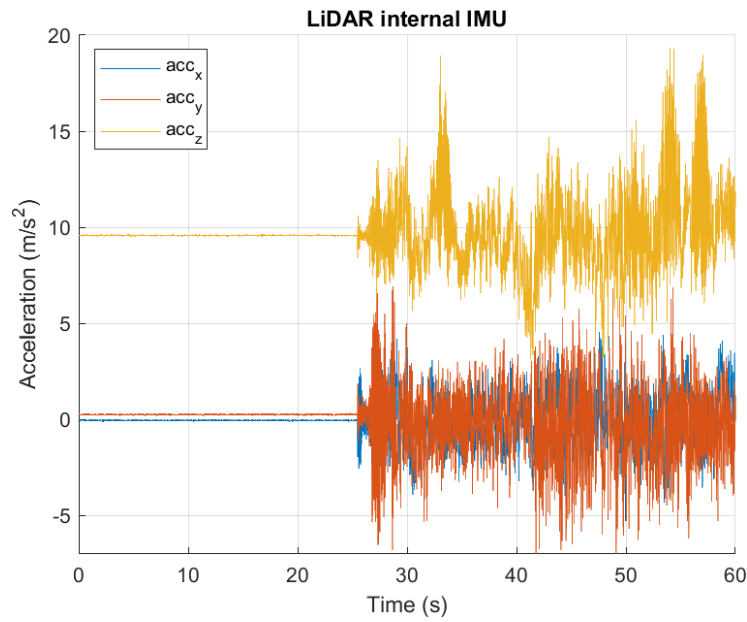


Figure 6.6: Accelerometers of LiDAR internal IMU

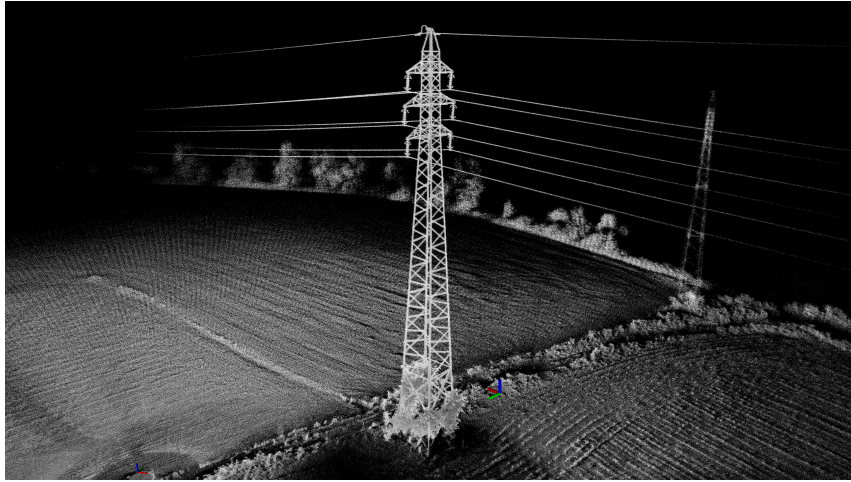
the average CPU and RAM consumption at the end of the mapping process, both in dataset A and dataset C. The objective is to assess which multi sensor algorithm consume fewer resources.

Table 6.2: CPU and RAM consumption of the multi-sensor SLAM algorithms

Dataset		FAST-LIO2	Faster-LIO	LIO-SAM
A	CPU(%)	20.60	23.55	31.35
	RAM (GB)	12.60	12.70	9.40
C	CPU(%)	12.80	14.32	34.80
	RAM (GB)	23.90	23.80	17.40

In the CPU performance, FAST-LIO2 shows better results than the others, being the algorithm that consumes less computational resources, with Faster-LIO being behind by only 2 to 3 %. On the other hand, LIO-SAM shows to be computationally heavier. On the other hand, when considering RAM consumption, it is the opposite. LIO-SAM consumes fewer resources, while the rest have similar results but are less optimized.

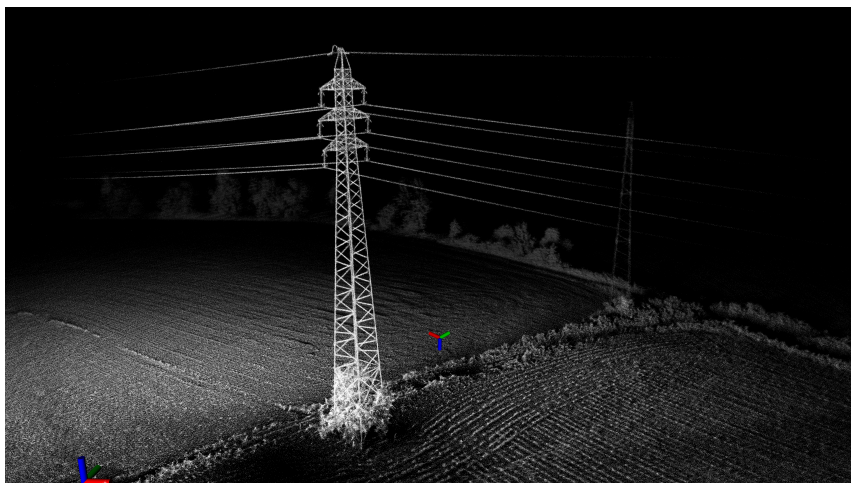
In conclusion, FAST-LIO2 or Faster-LIO have superior performance and are more indicated to apply where computational power is limited. However, to perform online mapping with high-resolution sensors, it is necessary to have a computer with more RAM, which results in higher energy consumption. Otherwise, it may not be possible to create maps from long flights.



(a) FAST-LIO2



(b) Faster-LIO



(c) LIO-SAM

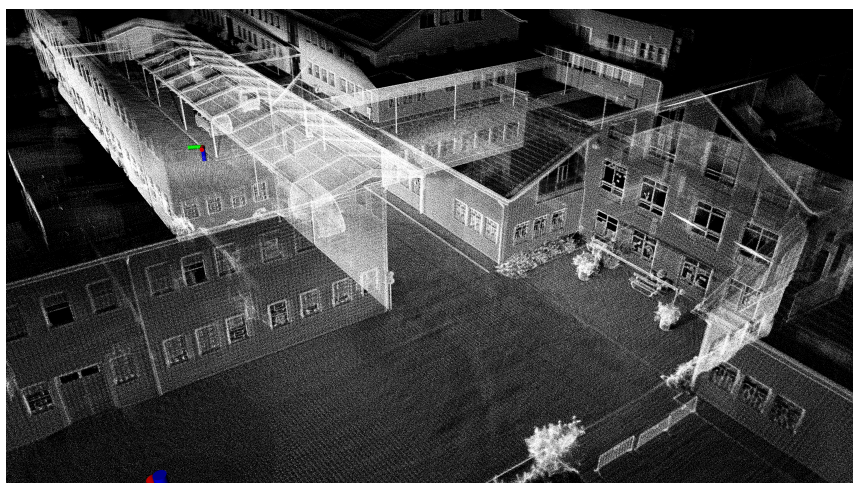
Figure 6.7: Mapping performance on dataset A



(a) FAST-LIO2

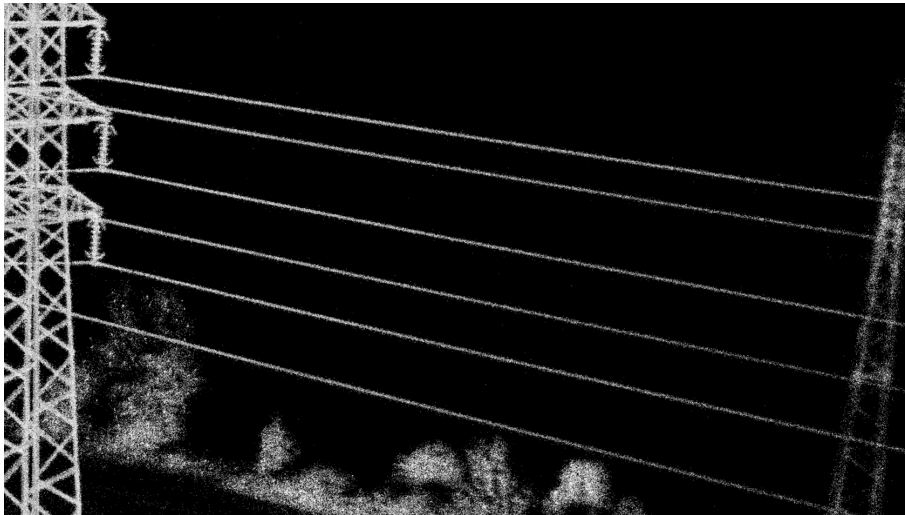


(b) Faster-LIO



(c) LIO-SAM

Figure 6.8: Mapping performance on dataset C



(a) FAST-LIO2



(b) LIO-SAM

Figure 6.9: Mapping noise differences in dataset A

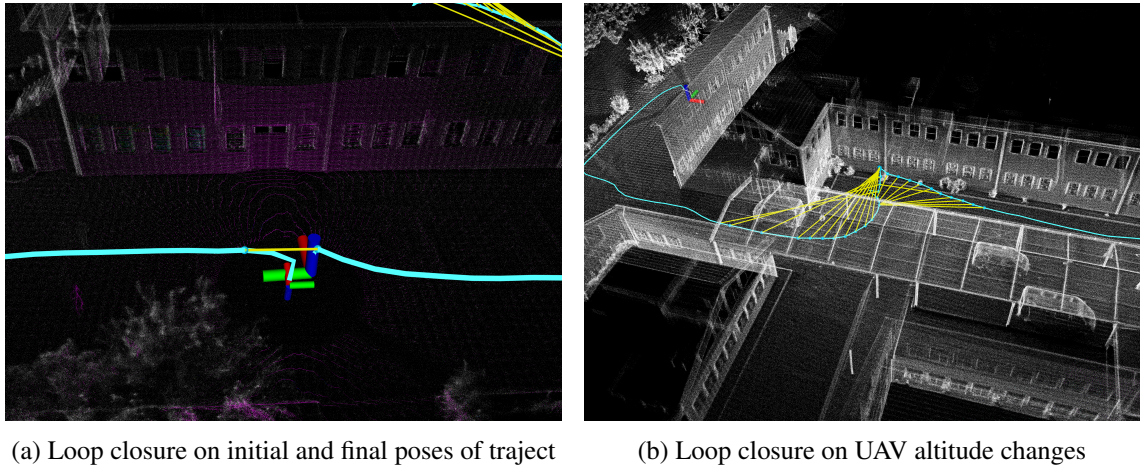


Figure 6.10: Loop Closure detection on LIO-SAM

6.5 Conclusions

This Chapter presents the results of the performance of five SLAM algorithms, that resorted to sensors such as LiDAR and IMUs to estimate the pose of the drone while mapping the surroundings. From the accuracy analysis, it was possible to validate the premise that the use of multiple sensors in the SLAM indeed contributes to improve performance, which translates into pose accuracy and map quality, providing the systems with robustness. On the three different datasets it was possible to obtain a dense point-cloud, that represent the environment to be used in simulation environments or to perform some types of inspection if the resolution or precision required allows.

In the results presented, two algorithms stood out for their performance, FAST-LIO2 and Faster-LIO that besides having the accuracy presented, also can handle the high dynamics of a UAV, and be computationally lighter than the others. In the mapping, all three show good results when visually analyzed.

On the inclusion of external IMUs in drones that does not provide IMU data, installing one sensor with an absorption kit may improve the results of some algorithms, and allow for a higher rate pose estimation, since LiDARs have an output rate between 10 and 20 Hz which can be low for systems with high dynamics or to feed a system control loop.

Chapter 7

Conclusions

7.1 Main Conclusions

Nowadays, the use of robotics is even more present in our society. As robots can now interact with humans and assist them in multiple tasks, the need for more accurate and robust systems is required. Furthermore, the robots also need to be able to operate in unknown and dynamic scenarios, and therefore, SLAM is constantly evolving to perform in more and more adverse conditions. The use of multiple sensors can increase the accuracy and the robustness to error, being able to include redundancy for these undesired situations.

This document presents an overview of possible solutions or key points to achieve an accurate solution for Simultaneous Localization and Mapping on UAVs. It addresses the background required to understand the topic, the challenges, and the requirements, including the different sensors and main characteristics, the localization methods for both global and relative pose estimation, and the different mapping approaches. Then, a literature review presents state-of-the-art SLAM algorithms using a single and a multi-sensor approach, as well as the newest approaches that resort to Machine Learning instead of traditional techniques.

Analyzing the related works, it became clear that the problems associated with perception, navigation, and mapping are familiar, and numerous approaches exist to solve these issues. However, applying these techniques in UAVs, which requires the pose estimation of 6-DoF of a very dynamic vehicle with accuracy and robustness, is still challenging. Furthermore, drones typically have less computational power, making it harder to perform localization and mapping in real-time.

An aerial platform capable of performing UAV surveying in multiple applications was developed, recording three datasets on infrastructures to test and benchmark different SLAM approaches. The results show that it is possible to obtain an accurate pose estimation, with a root mean squared error lower than 0.37 m and less than 4 degrees on the orientation, and a low drift on the trajectory, with the final positioning around 0.13 meters. Furthermore, it is also possible to obtain a low noise and high density map in the form of a point-cloud of the mapped structure and its surroundings.

7.2 Contributions

The main contributions of this dissertation are:

- A review of 3D SLAM algorithms, from single to multi-sensor.
- A benchmark of multiple SLAM algorithms for UAV applications: building and infrastructures inspections.
- Creation of multi-sensor UAV datasets that include LiDAR, inertial and visual data, and GPS and RTK information that can be used for ground truth.
- An assessment of the influence of magnetic perturbations and dynamic motions in a commercial AHRS, which resulted in a scientific paper [52] presented at ICARSC 2023.
- The design, 3D printing and assembly of a support for multiple sensors and hardware on multiple configurations.

7.3 Future Work

Considering the work presented, some points were not addressed, and that could be done in the near future, starting by understanding if the use of a camera can improve the accuracy and robustness of SLAM using drones since it was not possible to obtain valid results in this dissertation, or there is no advantage in having a higher computational power for processing image data. Also, evaluate whether the newest machine learning approaches can be applied to the high dynamics present in an UAV, and if those approaches can perform in the drone's embedded computer.

Appendix A

Hardware

A.1 OSDK interface cable

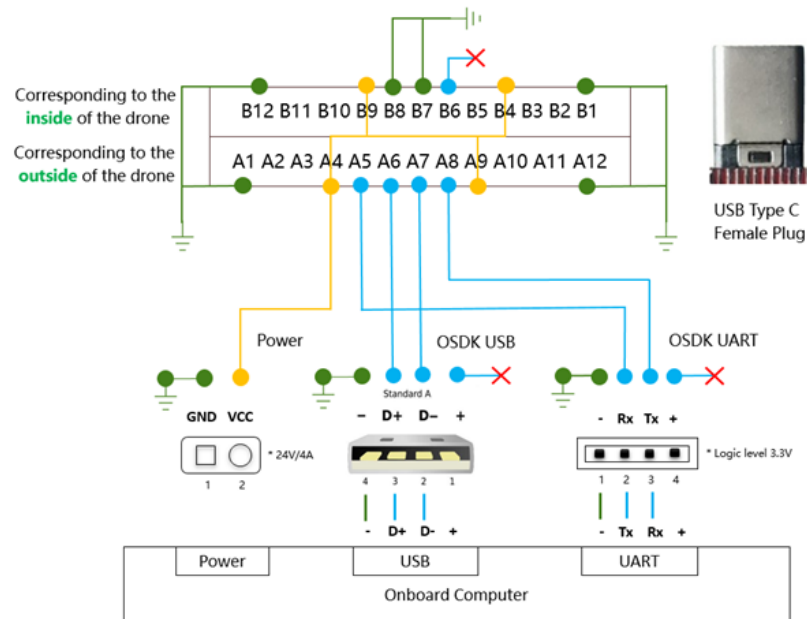


Figure A.1: OSDK cables connection [70]

Table A.1: Adapter Board Cable Interface Pinout [71]

Side A	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12
Name	GND	NC	NC	VCC	UART_RX	DP	DM	UART_TX	VCC	PWM0	PWM1	GND
Side B	B12	B11	B10	B9	B8	B7	B6	B5	B4	B3	B2	B1
Name	GND	NC	NC	VCC	ON_DET	RSVD2	RSVD1	SYNC(pps)	VCC	PWM3	PWM2	GND

Table A.2: Ouster Interface Cable Pinout - Adapted from [72]

				RJ45	Power
Function	Pin #	Wire Color	Type-3, 12V	Pin #	Pin
VCC	1	Red	18 AWG	-	+
Ground	7	Black	18 AWG	-	-
MULTI-PURPOSE_IO	3	Purple	28 AWG	NC	NC
SYNC_PULSE_IN	2	Yellow	28 AWG	NC	NC
Ethernet BI_DA+	5	White/Orange	28 AWG	1	NC
Ethernet BI_DA-	4	Orange	28 AWG	2	NC
Ethernet BI_DB+	8	White/Green	28 AWG	3	NC
Ethernet BI_DB-	6	Green	28 AWG	6	NC
Ethernet BI_DC+	9	Blue	28 AWG	4	NC
Ethernet BI_DC-	10	White/Blue	28 AWG	5	NC
Ethernet BI_DD+	12	White/Brown	28 AWG	7	NC
Ethernet BI_DD-	11	Brown	28 AWG	8	NC

A.2 LiDAR cable

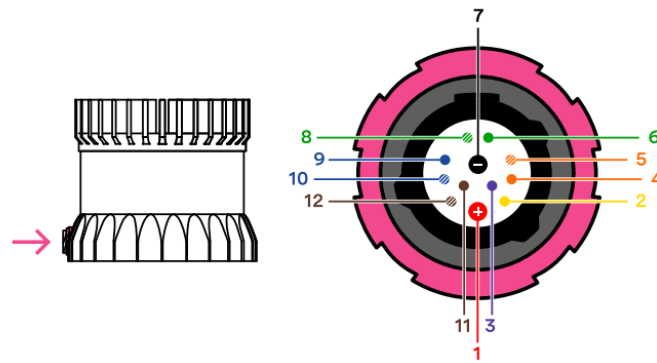


Figure A.2: Cable pinout of on-sensor receptacle [72]

References

- [1] Alena Otto, Niels Agatz, James Campbell, Bruce Golden, and Erwin Pesch. Optimization approaches for civil applications of unmanned aerial vehicles (uavs) or aerial drones: A survey. *Networks*, 72:411–458, 12 2018. doi:10.1002/NET.21818.
- [2] DHL. Dhl express launches its first regular fully-automated and intelligent urban drone delivery service, 2019. Accessed on December 5, 2022. URL: <https://www.dhl.com/global-en/home/press/press-archive/2019/dhl-express-launches-its-first-regular-fully-automated-and-intelligent-urban-drone-delivery-service.html>.
- [3] Qing-Li Zhou, Youmin Zhang, Yao-Hong Qu, and Camille-Alain Rabbath. Dead reckoning and kalman filter design for trajectory tracking of a quadrotor uav. In *Proceedings of 2010 IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications*, pages 119–124, 2010. doi:10.1109/MESA.2010.5552088.
- [4] David Nistér, Oleg Naroditsky, and James Bergen. Visual odometry. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1, 2004. doi:10.1109/CVPR.2004.1315094.
- [5] Ji Zhang and Sanjiv Singh. Loam: Lidar odometry and mapping in real-time. In *Proceedings of Robotics: Science and Systems (RSS '14)*, July 2014.
- [6] J. Borenstein, H. R. Everett, L. Feng, and D. Wehe. Mobile robot positioning: Sensors and techniques. *Journal of Robotic Systems*, 14(4):231–249, 1997. doi:[https://doi.org/10.1002/\(SICI\)1097-4563\(199704\)14:4<231::AID-ROB2>3.0.CO;2-R](https://doi.org/10.1002/(SICI)1097-4563(199704)14:4<231::AID-ROB2>3.0.CO;2-R).
- [7] Tim Pfeifer and Peter Protzel. Incrementally learned mixture models for gnss localization. In *2019 IEEE Intelligent Vehicles Symposium (IV)*, pages 1131–1138, 2019. doi:10.1109/IVS.2019.8813847.
- [8] Drone Engr. Is rtk the future of drone mapping, rtk drone solution, 2019. Accessed on January 3, 2023. URL: <https://www.droneassemble.com/is-rtk-the-future-of-drone-mapping/>.
- [9] Y. F. Wu, W. Wang, K. Q. Lu, Y. D. Wei, and Z. C. Chen. A new method for registration of 3d point sets with low overlapping ratios. *Procedia CIRP*, 27:202–206, 1 2015. doi:10.1016/J.PROCIR.2015.04.067.
- [10] Carlos Costa. Robot self-localization in dynamic environments. Master’s thesis, Faculty of Engineering, University of Porto (FEUP), 3 2015. URL: <https://repositorio-aberto.up.pt/handle/10216/78472>.

- [11] Jorge Ferreira. Mapping, localization and trajectory planning for quadruped robots in highly uneven terrains. Master's thesis, Faculty of Engineering, University of Porto (FEUP), 7 2022. URL: <https://repositorio-aberto.up.pt/handle/10216/142110>.
- [12] Juyong Zhang, Yuxin Yao, and Bailin Deng. Fast and robust iterative closest point. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44:3450–3466, 7 2020. doi: [10.1109/TPAMI.2021.3054619](https://doi.org/10.1109/TPAMI.2021.3054619).
- [13] Maks Ovsjanikov. Ps 1 - rigid alignment. Accessed on January 7, 2023. URL: http://www.lix.polytechnique.fr/~maks/Verona_MPAM/TD/TD1/.
- [14] Martin Lauer, Sascha Lange, and Martin Riedmiller. Calculating the perfect match: An efficient and accurate approach for robot self-localization. In Ansgar Bredenfeld, Adam Jacoff, Itsuki Noda, and Yasutake Takahashi, editors, *RoboCup 2005: Robot Soccer World Cup IX*, pages 142–153, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.
- [15] Miguel Pinto, Héber Sobreira, A. Paulo Moreira, Hélio Mendonça, and Aníbal Matos. Self-localisation of indoor mobile robots using multi-hypotheses and a matching algorithm. *Mechatronics*, 23(6):727–737, 2013. doi: <https://doi.org/10.1016/j.mechatronics.2013.07.006>.
- [16] Miguel Pinto, António Paulo Moreira, Aníbal Matos, Héber Miguel Placido Sobreira, and Filipe Neves dos Santos. Fast 3d map matching localisation algorithm. *Journal of Automation and Control Engineering*, 1:110–114, 2013.
- [17] Héber Sobreira, Luis Rocha, Carlos Costa, Jose Lima, Paulo Costa, and A. Paulo Moreira. 2d cloud template matching - a comparison between iterative closest point and perfect match. In *2016 International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, pages 53–59, 2016. doi: [10.1109/ICARSC.2016.13](https://doi.org/10.1109/ICARSC.2016.13).
- [18] Héber Sobreira, Carlos M. Costa, Ivo Sousa, Luis Rocha, José Lima, P. C.M.A. Farias, Paulo Costa, and A. Paulo Moreira. Map-matching algorithms for robot self-localization: A comparison between perfect match, iterative closest point and normal distributions transform. *Journal of Intelligent and Robotic Systems: Theory and Applications*, 93:533–546, 3 2019. doi: [10.1007/s10846-017-0765-5](https://doi.org/10.1007/s10846-017-0765-5).
- [19] Pingfu Chao, Yehong Xu, Wen Hua, and Xiaofang Zhou. A survey on map-matching algorithms. In Renata Borovica-Gajic, Jianzhong Qi, and Weiqing Wang, editors, *Databases Theory and Applications*, pages 130–131, Cham, 2020. Springer International Publishing. doi: [10.1007/978-3-030-39469-1_10](https://doi.org/10.1007/978-3-030-39469-1_10).
- [20] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic robotics*. MIT Press, Cambridge, Mass., 2005.
- [21] Simon J. Julier and Jeffrey K. Uhlmann. Unscented filtering and nonlinear estimation. *Proceedings of the IEEE*, 92:401–422, 3 2004. doi: [10.1109/JPROC.2003.823141](https://doi.org/10.1109/JPROC.2003.823141).
- [22] Youngjoo Kim, Hyochoong Bang, Youngjoo Kim, and Hyochoong Bang. Introduction to kalman filter and its applications. *Introduction and Implementations of the Kalman Filter*, 11 2018. doi: [10.5772/INTECHOPEN.80600](https://doi.org/10.5772/INTECHOPEN.80600).
- [23] Alex Becker. Kalman filter: Summary, 2017. Accessed on January 6, 2023. URL: <https://www.kalmanfilter.net/multiSummary.html>.

- [24] Sebastian Thrun, Dieter Fox, Wolfram Burgard, and Frank Dellaert. Robust monte carlo localization for mobile robots. *Artificial Intelligence*, 128(1):99–141, 2001. doi:[https://doi.org/10.1016/S0004-3702\(01\)00069-8](https://doi.org/10.1016/S0004-3702(01)00069-8).
- [25] M. Sanjeev Arulampalam, Simon Maskell, Neil Gordon, and Tim Clapp. A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *IEEE Transactions on Signal Processing*, 50(2):174 – 188, 2002. doi:[10.1109/78.978374](https://doi.org/10.1109/78.978374).
- [26] MathWorks. Localize turtlebot using monte carlo localization algorithm - matlab and simulink. Accessed on January 8, 2023. URL: <https://www.mathworks.com/help/nav/ug/localize-turtlebot-using-monte-carlo-localization.html>.
- [27] André Silva Aguiar, Filipe Neves dos Santos, José Boaventura Cunha, Héber Sobreira, and Armando Jorge Sousa. Localization and mapping for robots in agriculture and forestry: A survey. *Robotics*, 9(4), 2020. doi:[10.3390/robotics9040097](https://doi.org/10.3390/robotics9040097).
- [28] Cesar Cadena, Luca Carlone, Henry Carrillo, Yasir Latif, Davide Scaramuzza, Jose Neira, Ian Reid, and John J. Leonard. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Transactions on Robotics*, 32:1309–1332, 12 2016. doi:[10.1109/TRO.2016.2624754](https://doi.org/10.1109/TRO.2016.2624754).
- [29] Ieee standard for robot map data representation for navigation. *1873-2015 IEEE Standard for Robot Map Data Representation for Navigation*, pages 1–54, 2015. doi:[10.1109/IEEESTD.2015.7300355](https://doi.org/10.1109/IEEESTD.2015.7300355).
- [30] Giorgio Grisetti, Rainer Kummerle, Cyrill Stachniss, and Wolfram Burgard. A tutorial on graph-based slam. *IEEE Intelligent Transportation Systems Magazine*, 2(4):31 – 43, 2010. doi:[10.1109/MITS.2010.939925](https://doi.org/10.1109/MITS.2010.939925).
- [31] Andreas Nüchter and Joachim Hertzberg. Towards semantic maps for mobile robots. *Robotics and Autonomous Systems*, 56(11):915–926, 2008. Semantic Knowledge in Robotics. doi:<https://doi.org/10.1016/j.robot.2008.08.001>.
- [32] Changhui Jiang, Qian Meng, Bing Xu, Wang Gao, Panlong Wu, Lianwu Guan, Zeyu Li, Xiaobin Xu, Lei Zhang, Jian Yang, Chenfei Cao, Wen Wang, Yingying Ran, Zhiying Tan, and Minzhou Luo. A review of multi-sensor fusion slam systems based on 3d lidar. *Remote Sensing 2022, Vol. 14, Page 2835*, 14:2835, 6 2022. doi:[10.3390/RS14122835](https://doi.org/10.3390/RS14122835).
- [33] Tixiao Shan and Brendan Englot. Lego-loam: Lightweight and ground-optimized lidar odometry and mapping on variable terrain. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4758–4765, 2018. doi:[10.1109/IROS.2018.8594299](https://doi.org/10.1109/IROS.2018.8594299).
- [34] Raul Mur-Artal, J. M. M. Montiel, and Juan D. Tardos. ORB-SLAM: A versatile and accurate monocular SLAM system. *IEEE Transactions on Robotics*, 31(5):1147–1163, oct 2015. doi:[10.1109/tro.2015.2463671](https://doi.org/10.1109/tro.2015.2463671).
- [35] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. In *2011 International Conference on Computer Vision*, pages 2564–2571, 2011. doi:[10.1109/ICCV.2011.6126544](https://doi.org/10.1109/ICCV.2011.6126544).
- [36] Chao Qin, Haoyang Ye, Christian E. Pranata, Jun Han, Shuyang Zhang, and Ming Liu. Lins: A lidar-inertial state estimator for robust and efficient navigation, 2019. doi:[10.48550/ARXIV.1907.02233](https://doi.org/10.48550/ARXIV.1907.02233).

- [37] Tixiao Shan, Brendan Englot, Drew Meyers, Wei Wang, Carlo Ratti, and Daniela Rus. Lio-sam: Tightly-coupled lidar inertial odometry via smoothing and mapping. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5135–5142, 2020. doi:[10.1109/IROS45743.2020.9341176](https://doi.org/10.1109/IROS45743.2020.9341176).
- [38] Tixiao Shan, Brendan Englot, Carlo Ratti, and Daniela Rus. Lvi-sam: Tightly-coupled lidar-visual-inertial odometry via smoothing and mapping, 2021. doi:[10.48550/ARXIV.2104.10831](https://doi.org/10.48550/ARXIV.2104.10831).
- [39] Carlos Campos, Richard Elvira, Juan J. Gomez Rodriguez, Jose M. M. Montiel, and Juan D. Tardos. ORB-SLAM3: An accurate open-source library for visual, visual-inertial, and multimap SLAM. *IEEE Transactions on Robotics*, 37(6):1874–1890, dec 2021. doi:[10.1109/tro.2021.3075644](https://doi.org/10.1109/tro.2021.3075644).
- [40] Ji Zhang and Sanjiv Singh. Visual-lidar odometry and mapping: low-drift, robust, and fast. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2174–2181, 2015. doi:[10.1109/ICRA.2015.7139486](https://doi.org/10.1109/ICRA.2015.7139486).
- [41] Zengyuan Wang, Jianhua Zhang, Shengyong Chen, Conger Yuan, Jingqian Zhang, and Jianwei Zhang. Robust high accuracy visual-inertial-laser slam system. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6636–6641, 2019. doi:[10.1109/IROS40897.2019.8967702](https://doi.org/10.1109/IROS40897.2019.8967702).
- [42] Tong Qin, Peiliang Li, and Shaojie Shen. Vins-mono: A robust and versatile monocular visual-inertial state estimator. *IEEE Transactions on Robotics*, 34(4):1004–1020, 2018. doi:[10.1109/TRO.2018.2853729](https://doi.org/10.1109/TRO.2018.2853729).
- [43] Wei Xu, Yixi Cai, Dongjiao He, Jiarong Lin, and Fu Zhang. Fast-lio2: Fast direct lidar-inertial odometry. *IEEE Transactions on Robotics*, 38(4):2053–2073, 2022. doi:[10.1109/TRO.2022.3141876](https://doi.org/10.1109/TRO.2022.3141876).
- [44] Chungge Bai, Tao Xiao, Yajie Chen, Haoqian Wang, Fang Zhang, and Xiang Gao. Faster-lio: Lightweight tightly coupled lidar-inertial odometry using parallel sparse incremental voxels. *IEEE Robotics and Automation Letters*, 7(2):4861–4868, 2022. doi:[10.1109/LRA.2022.3152830](https://doi.org/10.1109/LRA.2022.3152830).
- [45] Jiarong Lin and Fu Zhang. R3live: A robust, real-time, rgb-colored, lidar-inertial-visual tightly-coupled state estimation and mapping package. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 10672–10678, 2022. doi:[10.1109/ICRA46639.2022.9811935](https://doi.org/10.1109/ICRA46639.2022.9811935).
- [46] Zachary Teed and Jia Deng. Droid-slam: Deep visual slam for monocular, stereo, and rgb-d cameras, 2022. arXiv:[2108.10869](https://arxiv.org/abs/2108.10869).
- [47] Wenshan Wang, Yaoyu Hu, and Sebastian Scherer. Tartanvo: A generalizable learning-based vo, 2020. arXiv:[2011.00359](https://arxiv.org/abs/2011.00359).
- [48] Julian Nubert, Shehryar Khattak, and Marco Hutter. Self-supervised learning of lidar odometry for robotic applications, 2020. doi:[10.48550/ARXIV.2011.05418](https://doi.org/10.48550/ARXIV.2011.05418).
- [49] Xingguang Zhong, Yue Pan, Jens Behley, and Cyrill Stachniss. Shine-mapping: Large-scale 3d mapping using sparse hierarchical implicit neural representations, 2023. arXiv:[2210.02299](https://arxiv.org/abs/2210.02299).

- [50] Kavisha Vidanapathirana, Milad Ramezani, Peyman Moghadam, Sridha Sridharan, and Clinton Fookes. Logg3d-net: Locally guided global descriptor learning for 3d place recognition. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 2215–2221, 2022. doi:10.1109/ICRA46639.2022.9811753.
- [51] Ignacio Vizzo, Tiziano Guadagnino, Benedikt Mersch, Louis Wiesmann, Jens Behley, and Cyrill Stachniss. Kiss-icp: In defense of point-to-point icp – simple, accurate, and robust registration if done the right way. *IEEE Robotics and Automation Letters (RA-L)*, 8(2):1029–1036, 2023. doi:10.1109/LRA.2023.3236571.
- [52] João G. Martins, Marcelo R. Petry, and António Paulo Moreira. Assessment of the influence of magnetic perturbations and dynamic motions in a commercial ahrs. In *2023 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, pages 175–180, 2023. doi:10.1109/ICARSC58346.2023.10129577.
- [53] Aparna Harindranath and Manish Arora. Mems imu sensor orientation algorithms-comparison in a simulation environment. In *2018 International Conference on Networking, Embedded and Wireless Systems (ICNEWS)*, pages 1–6, 2018. doi:10.1109/ICNEWS.2018.8904029.
- [54] Sebastian O. H. Madgwick, Andrew J. L. Harrison, and Ravi Vaidyanathan. Estimation of imu and marg orientation using a gradient descent algorithm. In *2011 IEEE International Conference on Rehabilitation Robotics*, pages 1–7, 2011. doi:10.1109/ICORR.2011.5975346.
- [55] Walter Fetter Lages and Renato Ventura Bayan Henriques. Performance evaluation of data fusion for orientation estimation in an intelligent inertial measurement unit. In *2019 Latin American Robotics Symposium (LARS), 2019 Brazilian Symposium on Robotics (SBR) and 2019 Workshop on Robotics in Education (WRE)*, pages 150–155, 2019. doi:10.1109/LARS-SBR-WRE48964.2019.00034.
- [56] Bosch Sensortec. Data sheet bno055: Intelligent 9-axis absolute orientation sensor, 2019. Accessed on January 25, 2023. URL: <https://datasheetspdf.com/pdf-file/848780/Bosch/BNO055/1>.
- [57] Jwu-Sheng Hu and Kuan-Chun Sun. A robust orientation estimation algorithm using marg sensors. *IEEE Transactions on Instrumentation and Measurement*, 64(3):815–822, 2015. doi:10.1109/TIM.2014.2359815.
- [58] Xin Tong, Zhaofeng Li, Guowei Han, Nan Liu, Yan Su, Jin Ning, and Fuhua Yang. Adaptive ekf based on hmm recognizer for attitude estimation using mems marg sensors. *IEEE Sensors Journal*, 18(8):3299–3310, 2018. doi:10.1109/JSEN.2017.2787578.
- [59] Simone A. Ludwig and Kaleb D. Burnham. Comparison of euler estimate using extended kalman filter, madgwick and mahony on quadcopter flight data. In *2018 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 1236–1241, 2018. doi:10.1109/ICUAS.2018.8453465.
- [60] Valérie Renaudin, Muhammad Haris Afzal, and Gérard Lachapelle. New method for magnetometers based orientation estimation. In *IEEE/ION Position, Location and Navigation Symposium*, pages 348–356, 2010. doi:10.1109/PLANS.2010.5507301.

- [61] S. Salehi, N. Mostofi, and G. Bleser. A practical in-field magnetometer calibration method for imus. In *IROS Workshop on Cognitive Assistive Systems: Closing the Action-Perception Loop*, pages 39–44, 2012.
- [62] Xsens Technologies. Magnetic calibration manual, 2019. Accessed on January 20, 2023. URL: https://www.xsens.com/hubfs/Downloads/Manuals/MT_Magnetic_Calibration_Manual.pdf.
- [63] Ltd SZ DJI Technology Co. Manifold 2 series - product information, 2019. Accessed on April 12, 2023. URL: https://dl.djicdn.com/downloads/manifold-2/20190528/Manifold_2_Production_Information_v1.0_Multi.pdf.
- [64] Paul Furgale, Joern Rehder, and Roland Siegwart. Unified temporal and spatial calibration for multi-sensor systems. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1280–1286, 2013. doi:10.1109/IROS.2013.6696514.
- [65] Joern Rehder, Janosch Nikolic, Thomas Schneider, Timo Hinzmann, and Roland Siegwart. Extending kalibr: Calibrating the extrinsics of multiple imus and of individual axes. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4304–4311, 2016. doi:10.1109/ICRA.2016.7487628.
- [66] Miguel Oliveira, Eurico Pedrosa, André Pinto de Aguiar, Daniela Ferreira Pinto Dias Rato, Filipe Neves dos Santos, Paulo Dias, and Vitor Santos. Atom: A general calibration framework for multi-modal, multi-sensor systems. *Expert Systems with Applications*, 207:118000, 2022. doi:<https://doi.org/10.1016/j.eswa.2022.118000>.
- [67] João G. Martins, Marcelo R. Petry, and António Paulo Moreira. UAV Surveying - Electrical Transmission Infrastructure, June 2023. doi:10.5281/zenodo.8092105.
- [68] João G. Martins, Marcelo R. Petry, and António Paulo Moreira. Uav surveying - water tower inside a school, June 2023. doi:10.5281/zenodo.8099333.
- [69] João G. Martins, Marcelo R. Petry, and António Paulo Moreira. Uav surveying - school building, June 2023. doi:10.5281/zenodo.8092129.
- [70] Xiao Long. How to make an adapter cable/board for dji m300 rtk osdk interface, 2020. Accessed on April 15, 2023. URL: <https://forum.dji.com/thread-219723-1-1.html>.
- [71] DJI. Onboard sdk - device connection, 2020. Accessed on April 15, 2023. URL: <https://developer.dji.com/onboard-sdk/documentation/quickstart/device-connection.html#using-m300-rtk>.
- [72] Ouster. Os1 hardware user manual - for rev7 sensors, 2023. Accessed on May 23, 2023. URL: <https://data.ouster.io/downloads/hardware-user-manual/hardware-user-manual-rev7-os1.pdf>.