Localization of a drone in a GPS-denied environment using LiDAR

Andrea Lorenzo Pallini

Master of Science Thesis

Delft University of Technology

Delft Center for Systems and Control

Localization of a drone in a GPS-denied environment using LiDAR

Master of Science Thesis

For the degree of Master of Science in Embedded Systems at Delft University of Technology

Andrea Lorenzo Pallini

April 2, 2018

CONFIDENTIAL

Faculty of Electrical Engineering, Mathematics and Computer Science (EEMCS) · Delft University of Technology

The work in this thesis was supported by Mainblades Inspections B.V. Their cooperation is hereby gratefully acknowledged.

Abstract

Mainblades Inspection is a company that specializes in the inspection of aircraft using a Micro Aerial Vehicle (MAV). The inspection is performed by capturing visual images of the airplane with a camera. The pictures are paired with the position of the drone with respect to the aircraft, making accurate localization of the MAV crucial to its operation. However, inertial sensors installed on the robot are not precise enough as they suffer from drift. Consequently, the localization can be improved using LiDAR (Light imaging, Detection, And Ranging) data. Therefore, the goal of this thesis is to *design a 3D LiDAR motion tracking system installed on a MAV, for localization in GPS-denied environment, such as in a hangar.*

The main advantage of the LiDAR technology is the ability to find correspondences between points captured at different times and estimate the movement of the robot in the time interval. Algorithms that create these correspondences are established by synthesizing scan matching algorithms from literature and verifying them using real data. The method proposed in this report aims to track the motion of the robot and additionally map the environment, for this reason has been named Tracking and Mapping (TAM).

The analysis of TAM revealed that the internal sensors of the robot provide a good initial estimate of the location, and the method is able to identify correctly the correspondences between consecutive scenes, and indeed TAM does improve the estimation result of the position. However, an error with the ground truth remains. Nevertheless, the design of the system enhances the real-time performances on the embedded system compared to other LiDAR Odometry methods.

In conclusion, TAM's algorithm can be improved by the introduction of additional filtering procedure of the point cloud and the detection of 3D features with laser scanners. Though, the possibility to execute the method in real-time onboard allows the system to be an enabling technology for an automated inspection procedure.

Table of Contents

List of Figures

List of Tables

Preface

The work presented in this report has been performed as a part of a thesis project in the scope of the Master of Science program Embedded Systems of the Electrical Engineering, Mathematics and Computer Science (EEMCS) faculty and partially the specialization Systems Control of the Mechanical, Maritime and Materials Engineering (3mE) faculty.

The project has been proposed by the founders and directors of Mainblades Inspections ir. Dejan Borota and Jochem Verboom. The thesis assignment was formulated together with them and with the input and approval of prof. dr. Joris SIjs and prof. dr. Javier Alonso Mora, professor at the Delft Center for Systems and Control (DCSC) group.

The final thesis commitee consists of the following individuals:

- Dr. ir. Bart De Schutter
- Prof. dr. ir. Arjan Van Genderen
- Prof. dr. ir. Javier Alonso Mora
- Prof. dr. ir. Joris Sijs
- Ir. Dejan Borota
- Ir. Jochem Verboom

I would like to thank all the people who have in any way contributed to this thesis and graduation process, be it by providing support, feedback, challenges, opportunities, guidance, criticism, confidence, pleasure, etc.
In particular, I would like to thank my grandmother and my mother for their tireless enthusiasm that encourage me for my entire life.

Ultimately, I dedicate this thesis to the memory of my grandfather, Gino Pallini.

Chapter 1

Introduction

This report is written as a continuation of the work reported earlier in the MSc. literature survey with the corresponding title, finalized in June 2017. That study was a broad analysis of all methods and principles used in the problem of localization and mapping of Micro Aerial Vehicles (MAVs), reviewing both the industry standard applications and state of the art research.

The progress of the project led to a specific focus on the localization problem using the integration of several sensors. In this initial chapter an introduction of the project, with a description of the background of aerial inspections and of Mainblades Inspections, are proposed. The chapter is concluded with the definition of the problem tackled in this report, the contributions and the approach of my research will be described.

1-1 Background

1-1-1 Current Aircraft inspections

Mainblades Inspections B.V introduces an innovative automated technology for the inspection of aircrafts. The inspection is executed with a MAV which is optimally designed to operate on aircraft of large dimensions to check their status.

Current situation in aerial inspections Aircraft flight control surfaces - ailerons, rudders, elevators, flaps and slats - are subject to considerable wear and tear in the course of everyday operation. Control surfaces are usually adhesively bonded composite or aluminum honeycomb sandwich structures [7]. These components can suffer from damage caused by runway debris, hailstones, lightning strikes or tools dropped during maintenance.

Each year, airlines are confronted with AOG[1] situations as a result of hail impacts on aircraft fuselages [6]. The grounded aircraft then has to undergo a detailed inspection to gauge the

[1]Aircraft on Ground or AOG is a term in aviation maintenance indicating that a problem is serious enough to prevent an aircraft from flying.

extent of the damage - a manual procedure, time-consuming and therefore costly. This is because hail impacts cannot be treated lightly. Any alteration in the airframe can lead to complex aerodynamic disruptions of the airflow that it is vital to remedy.

The shape and size of impacts can vary according to weather conditions. As a result, the recommended procedure involves examining each of the dents identified and taking exact measurements of their length, width, and depth. This is a huge task when considering that manual inspection takes 4-5 hours per square meter.

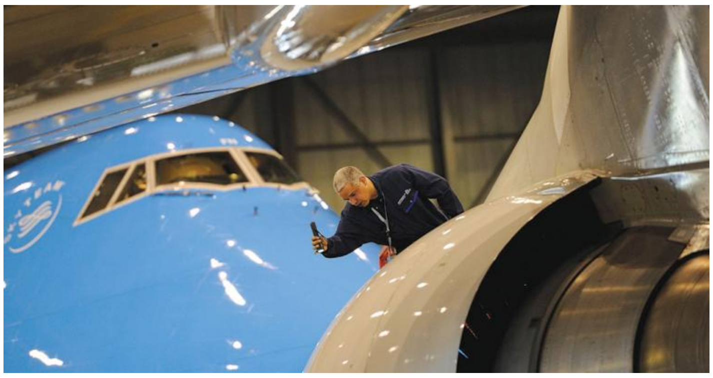


Figure 1-1: Example of manual inspectionperformed by a human.

Drone inspection Drones, or Micro Aerial Vehicles (MAVs), are probably best known for military applications but, in recent years, they have become increasingly used for commercial purposes. Large, complex structures are best surveyed from the air. It's no wonder that inspection has been one of the first MAV applications to be quickly and widely embraced world-wide. The main reason of the successful integration of this robot in the inspection field are:

- **Duration** Drones can significantly reduce the time and risk of inspections when compared to conventional methods. A trained person can easily fly a small, camera-equipped drone over the inspected large structures to capture imagery. The footage is then used to recognize possible demages in the body, with image recognition software. The main advantage of flying robot is the speed of the inspection. The drone can be deployed very quickly and capture the necessary footage in hard-to-reach places much faster than a human could.
- **Quality** The images collected are easy to share and interpret, and they provide an overview of the inspection that many prefer to a report collected by sampling individual data points through manual inspection. The images can also be combined with the position of the robot with respect to the surveyed object, which increase the amount of information during the operation.
- **Tele-operation** The drones are also able to stream what the camera captures in real-time, which allows the property manager to assess conditions from the ground by watching the drone operator's monitor.

- **Risks** Inspections may be dangerous for humans. The operators have to carefully walk around the area, and sometimes it requires to stay on proper infrastructures mounted at several meters of distance from the inspected object. This risk can be essentially reduced using new technologies, such as MAVs, to substitute humans in hazardous procedure.

These limitations of human operations compared to MAV can be applied to the aircraft inspection industry.

	Inspection with a drone	**Manual inspection**
Duration	**Low:** an inspection around an aircraft can be executed in 20 min.	**High:** The ispection requires the installation of additional structures around the aircraft.
Quantity	**High:** The robot can reach ares diffucult to inspect by a human.	**High:** in area easy to reach, very limited otherwise.
Risk	**Low:** If the drone is controlled by a certified pilot the risk of errors is really low.	**High:** The human operator has to stand and walk on dangerous structures.

Table 1-1: Summary comparison between human and robot inspections.

1-1-2 Mainblades Inspection

Mainblades Inspections (MBI) is a start-up founded in 2015. Since the beginning, the team was able to structure a clear road-map to follow in order to achieve predetermined goals. In this section, the current status of MBI will be analyzed.

Autonomous robot navigation

The current and most relevant phase for MBI is the development of an autonomous robot able to perform inspections of aircrafts. This goal is achieved by:

1. synchronizing several sensors on-board,
2. designing a software solution able to process the data captured,
3. compute in real-time an estimation of the position of the drone with respect to the aircraft,
4. determine the inspection path and avoid the obstacles,
5. ultimately provide to the final user a report that links the pictures captured with the position of them.

Navigation is a key capability of mobile robots and substantial progress has been made in the area of autonomous navigation in the past. Robots that are able to successfully navigate in the air have to cope with a series of challenges including complex three-dimensional settings and highly dynamic scenes paired with unreliable GPS information.

The process to achieve this goal in MBI has been subdivided into a set of tasks. The first item of the list is mapping and localization in real-time, this problem is commonly known as Simultaneous Localization And Mapping (SLAM). Furthermore, the robot requires the ability to autonomously determine the flight trajectory around the aircraft.

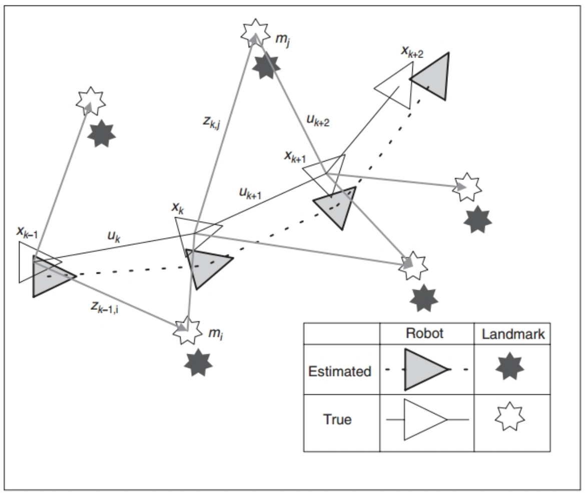


Figure 1-2: the essential SLAM problem. A simultaneous estimate of both robot and landmark locations is required. The true locations are never known or measured directly. Observations are made between true robot and landmark locations. At some instance of time k, the following quantities are defined: x_k, the state vector describing the location and orientation of the robot, u_k, the control vector which was applied at time $k-1$ to drive the robot to a state x_k at time k, m_i, a vector describing the location of the ith landmark. Landmarks are assumed to be stationary [17].

Simultaneous Localization and Mapping (SLAM) is a technique used by robots and autonomous vehicles to build a map within an unknown environment, or to update a map within a known environment, while keeping track of their current location. In Figure 1-2 the conceptual diagram of SLAM is illustrated. The mobile robot estimates landmarks and its position simultaneously. The map estimation leverages the position information and the position estimation leverages the map information.

Initially deployed drone

At the commencement of the project, MBI provided the DJI Matrice as the initial testing platform with the following equipment installed and available [2]:

- Sonar sensors and stereo cameras
- Camera Full HD able to rotate around the z and y axis.

[2]Additional details of the hardware is reported in the Appendix

- Poszyx[3] module for localization [29] .
- Global positioning system (GPS).
- Inertial measurement unit (IMU).

Visual-inspections The inspection was based on visual images captured by the Full HD camera. The images allow the operator that is controlling the drone to identify possible demages on the surface of the aircraft. The images however are not processed by any intelligent system able to detect damages on the aircraft using image recognition techniques, yet.

Remote controlled The drone is remotely controlled by an operator and the localization is computed using the Poszyx system. This localization requires additional devices placed around the aircraft, named anchors; the system is based on triangulation to determine the position of the module installed on the drone in 3D. Figure 1-3 shows an example of triangulation system for localization. Mainblades also developed a graphic user interface (GUI) able to show the position of the drone (output obtained from Poszyx).

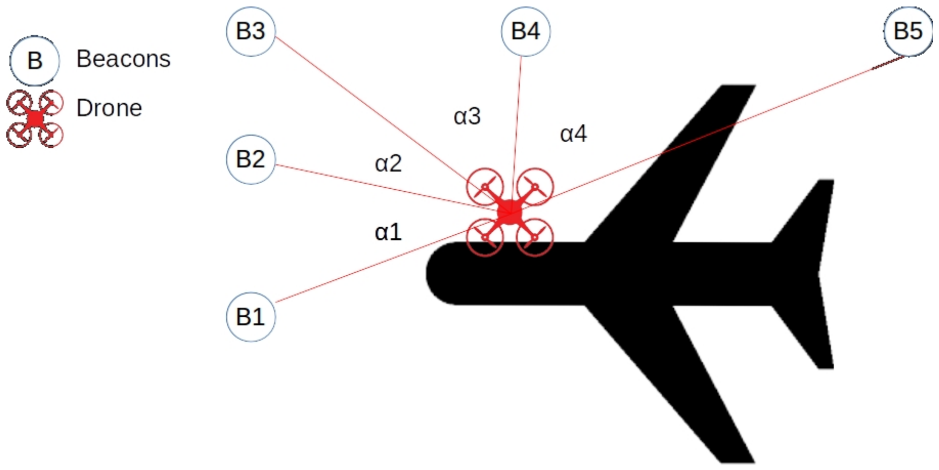


Figure 1-3: Triangulation method based on straight lines intersection uses the angles (α_{1-4}) created between the lines to compute the position of the robot [18].

[3]Pozyx is a hardware solution that provides accurate positioning and motion information. It uses ultra-wideband technology for positioning. This wireless technology works both indoor and outdoor. For 3D localization at least 5 modules are needed. One module on the drone and 4 modules that act as anchors.

1-2 Problem definition

In the previous section, an overview of the initial situation of the project has been introduced. In this section the definition of the problem will be described. As already mentioned at the beginning of the chapter, MBI is currently working on the localization of the robot and the mapping procedure using laser technologies, better known as LiDAR (Laser Imaging Detection and Ranging). The focus of this research is on the correct usage of a 2D LRF for the visualization of 3D environments and motion tracking system able to estimate the position of the drone in real-time, without using external hardware such as the Poszyx system.

Localization

In order to understand how the positioning system of the robot performs in the 3D environment, it is important to define it first.

Pose The localization of a robot in a 3D environment is described by six parameters. Three parameters are used for the position along the three axes (x, y, z) and three more parameters are necessary to represent the rotation respect to the three axes. Table 1-2 shows the relation between the terms roll, pitch, and yaw with the relative rotation axes.

Rotation	**Axes rotation**
pitch(ϕ)	x
roll(θ)	y
yaw(ψ)	z

Table 1-2: Rotational movement around the three axes.

Transformations The pose of the robot refers to the *base_link* frame depicted in Figure 1-4. Two main transformations are necessary to estimate the location of the drone with respect to the ground frame:

- odom - footprint: translation along the x and y axis plus the rotation around the z axis (yaw).
- footprint-base_link: translation along z and rotation around x znd y (pitch and roll).

Sensors As already mentioned in the previous sections, the positioning system of the robot developed by MBI cannot rely on GPS information. The only sensor used to estimate the position of the drone is the IMU which provides acceleration and velocity along the three axes at 100Hz. The integration of the data in the time interval gives an estimate of the robot movement. The estimated movement is summed to the previous position, and subsequently, the current position with respect to the initial frame is obtained. This positioning system can be improved integrating information from the LiDAR sensor, this process is named LiDAR Odometry.

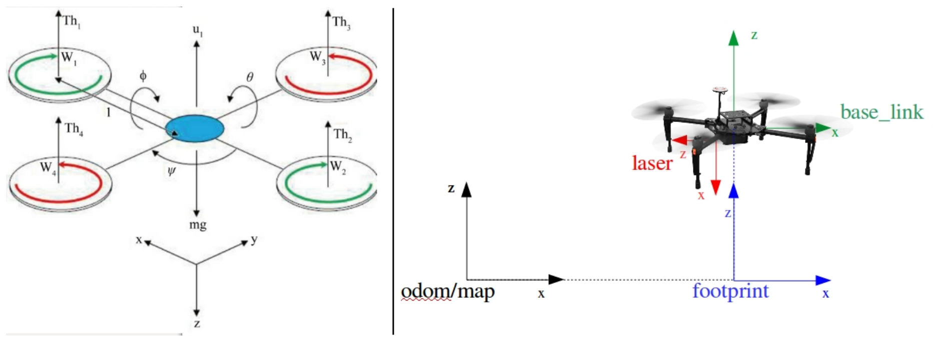


Figure 1-4: On the left the quadrotor system is illustrated. In the picture are represented the three rotational values roll, pitch and yaw with respect to the relative axis. On the right the frame transformation system is shown. The ground frame in the picture is named *odom/map*, along this report it will be refered as *odom* frame. A 2D transformation is applied between odom and *footprint*. The transformation between *footprint* frame and *base_link* is composed by translation along z, pitch and roll. The translation between *base_link* and *laser* is static, however the *laser* frame rotate around the x-axis.

LiDAR Odometry The problem to localize the robot, is connected with the ability to integrate LiDAR data with the IMU. The position estimation method using lasers is based on the recognition of correspondences between features. It is important that the system can extrapolate some information from the environment surrounding and reorganize them in data structures. The data structure must be integrated with a matching system able to identify equivalences between a new capture scan and a previous one already contained in memory. The scan matching system and the ability to identify similarities and ultimately track the motion of the robot in the 3D environment are the key-points of this research.

Laser frame The pose of the laser is expressed by the laser frame. The pose of the laser frame is computed at real-time, and it is a transformation that uses as a parent the position of the robot (*base_link*) and it is based on two components:

- static: the placement of the origin of the laser frame is static with respect to the body frame. Figure 1-4 refers to the current solution of the design where the x-axis points downward and the laser rotates around it.
- dynamic: the rotation around the x-axis on the laser frame is dynamic indeed also the transformation is dynamically computed using the information provided by the encoder of the motor.

The solution to the problem has to solve the conversion of the points captured into the laser frame by the Laser Range Finder (LRF) in other frames such as *base_link* and *odom*.

Problem Localization combines the research fields and techniques of sensing, estimation, and control while operating in both hardware and software domains. It is essential to the robot functioning, however difficult to develop due to strong competition within the industry and confidential research projects.
No localization system is yet available in the described application at a satisfactory level.

The below stated identified factors combine to form the difficulty for designing a complete solution for the positioning problem, and are used as a starting point for this thesis:

- Impossibility to use GPS signals in a hangar.
- Custom position estimation solution for a combination of data from multiple sensors, on-board and in real-time.
- Requirement of designing a reliable, maintainable and extensible software solution.
- The robot must estimate its position without need any external hardware.

1-2-1 Thesis approach

The above analysis converges to the main objective of this thesis: "Localization of a drone in a GPS-denied indoor environment using LiDAR". The report encompasses the system configuration by a step-wise design process: departing from mere knowledge of the problem to basic system requirements, to elaborate system design, to an evaluation of the working prototype. A scientific approach in reaching this objective is ensured by introducing instrumental objectives, illustrated in Figure 1-5, substantiating all design choices with arguments, and submitting the prototype to evaluation.

The thesis is divided into three parts, each representing a design step with a specified objective which contributes to reaching the main objective. The conceptual design includes the definition and discussion of the main constraints and requirements, it also covers an analysis of the related works. The conceptual design represents the first of the three in-depth analysis levels expanded. The system design encapsulates the second step in the design of the software solution, translating the proposition of optimal principles into an implementable solution. The evaluation describes the third and final design step: putting the principles into practice and serving as proof of the proposed design steps and choices. A schematic representation of the general thesis structure is given in Figure 1-5.

Contributions

The contribution of this thesis can be formulated for both the academic and industrial pillars driving the proposed design:

MAV robotics research field (accademy):

- presenting a possible solution for the localization problem of a MAVs in 3D GPS-denied environments and its evaluation.

The Mainblades Inspection robot (industry):

- overcoming the main operating limitations;
- presenting implementation and working methods which can be applied to the robot;
- enabling further development for the robot in terms of navigation, control, inspection, mapping, autonomous operation.

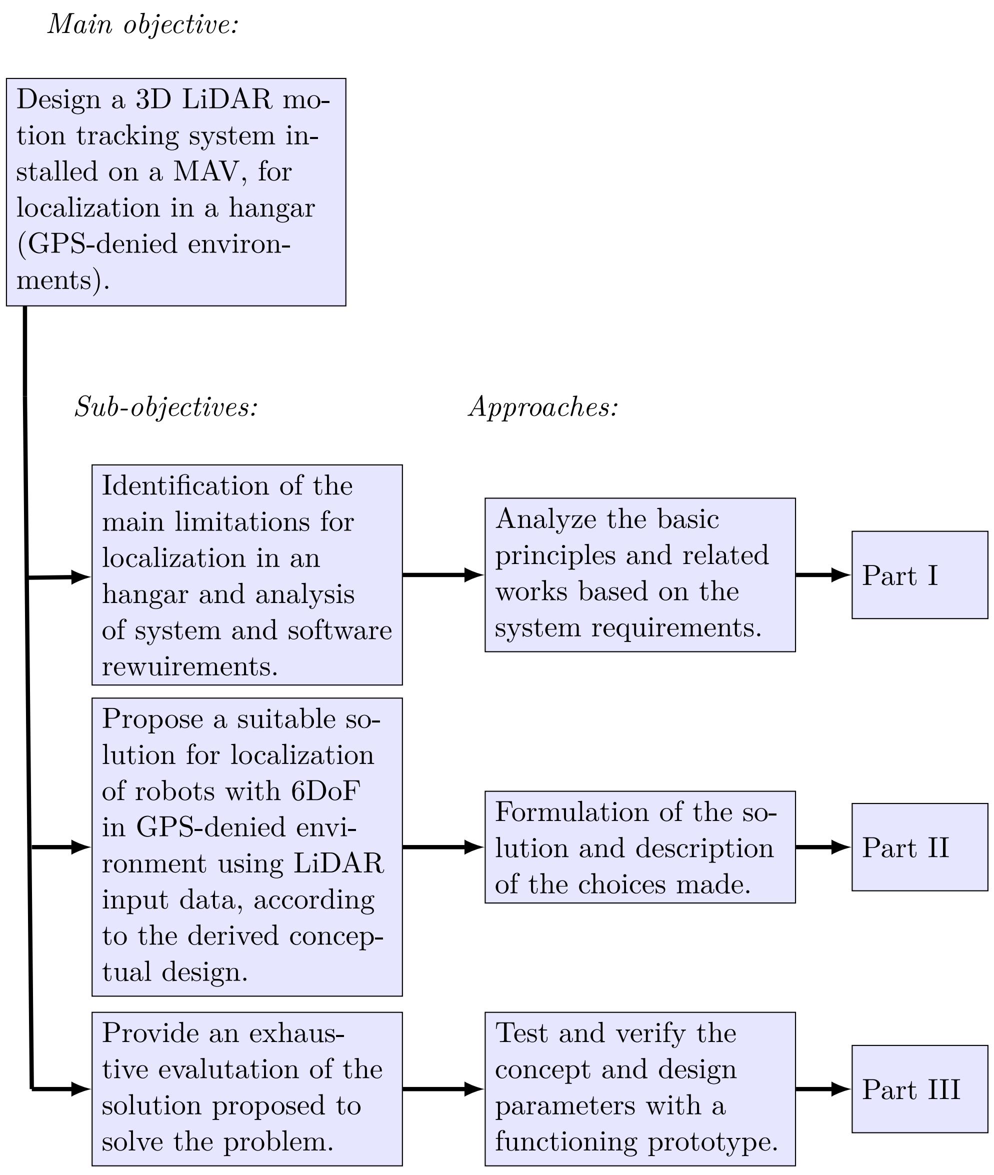
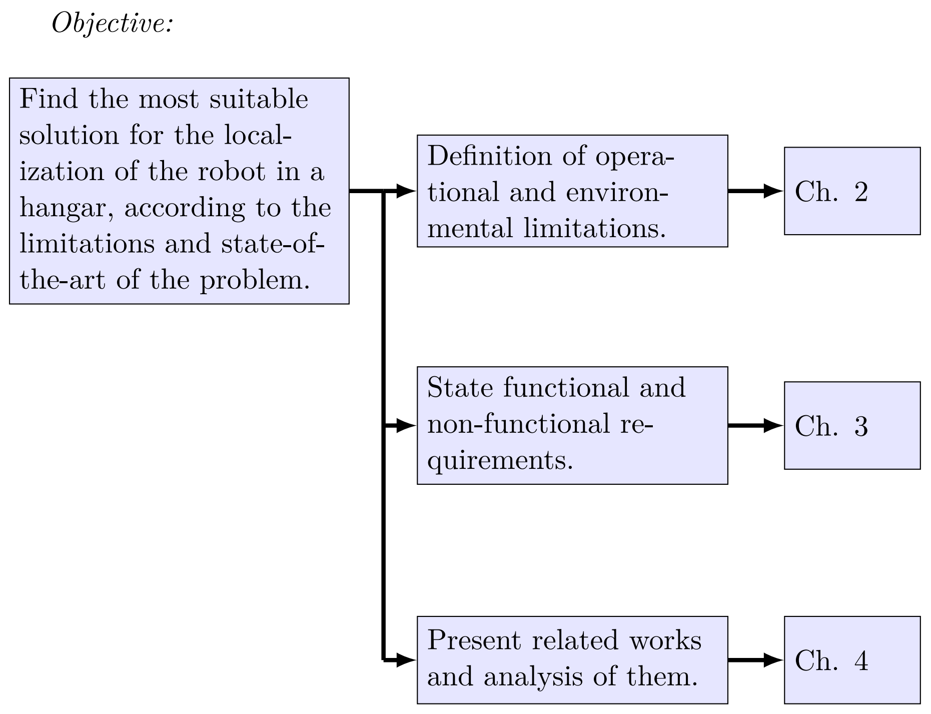

Main objective:
Design a 3D LiDAR motion tracking system installed on a MAV, for localization in a hangar (GPS-denied environments).
Sub-objectives:
Approaches:
Identification of the main limitations for localization in an hangar and analysis of system and software rewuirements.
Analyze the basic principles and related works based on the system requirements.
Part I
Propose a suitable solution for localization of robots with 6DoF in GPS-denied environment using LiDAR input data, according to the derived conceptual design.
Formulation of the solution and description of the choices made.
Part II
Provide an exhaustive evalutation of the solution proposed to solve the problem.
Test and verify the concept and design parameters with a functioning prototype.
Part III

Figure 1-5: Structure of the literature survey.

Part I

Conceptual design and related works

A conceptual design specifies the principal solution to a problem [27]. In this part of the thesis, the conceptual design approach is described, which is aimed at finding the most suitable combination of sensory elements and software processes to estimate the position of the robot. This contributes to the main design of a positioning system by proposing a solution for the software part of the system design. To this end an analysis is made of the operational constraints and the requirements which the positioning system needs to fulfill followed by the analysis of the related works. Following this line of reasoning the part is divided into chapters, with the objectives and approaches as illustrated in Figure **??**.


Objective:
Find the most suitable solution for the localization of the robot in a hangar, according to the limitations and state-of-the-art of the problem.
Definition of operational and environmental limitations.
Ch. 2
State functional and non-functional requirements.
Ch. 3
Present related works and analysis of them.
Ch. 4

Structure of conceptual design and related works part.

Chapter 2

Constraints

The evaluation of related works and their results requires the definition of parameters to be judged. In this chapter a set of constraints, that restricts or dictates the actions, are presented. Moreover, the analysis of the operating conditions is done by summing up the constraints imposed on the positioning system, which form the restricting factors for the functioning and performance. A division is made between environmental and goal-related constraints, defining where the system needs to function and what the system needs to achieve, respectively.

2-1 Environmental

Environmental constraints define the working area of the robot. It includes all the critical scenarios and circumstances that the positioning system needs to operate under.

In the hangar

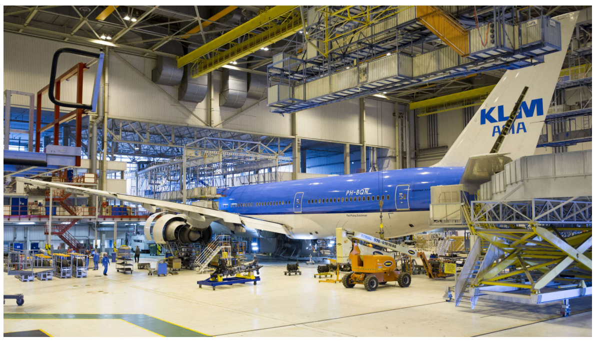


Figure 2-1: Hangar of KLM. The picture shows the obstacles and the operational structured installed around the plane.

Most of the inspection's operations are executed in a hangar. Operating in the hangar of an airport means having to take into account all of the characterizations of the environment imposed by the infrastructures (area, reflective materials, field of view, etc.).

Area

The relatively confined space implies a number of boundaries of the operating area:

- working structures, such as stairs and working stations;
- ceiling, gates, and floor;
- human operators.

Some of these elements may be dynamic and they may change their position during the operation. The navigation in dynamic environment increase the difficulties during the computation of scan matching techniques as illustrated in Figure 2-2.

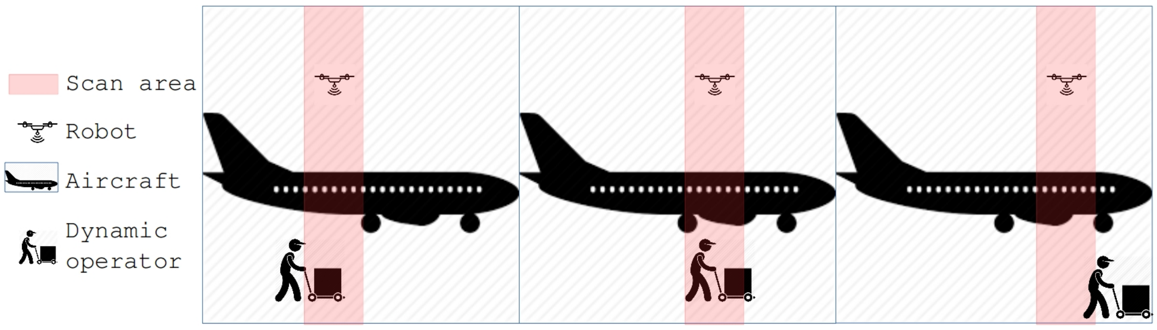


Figure 2-2: Example of a dynamic situation where an operator is moving close to the aircraft and it is included in the sweeps in different positions.

Reflective material

The quality of the data captured with the laser can be anomaly influenced by the material of the object where the laser is pointing [19]. The surface of the aircraft is shining and the reflection of the laser may be influenced and report range distance error. Furthermore, also elements such as side windows and pilot cabin's window can generate an error in the LiDAR perception.

Field of view

Two main situations, illustrated in Figure 2-3, can create problems regarding the coverage of the rotating laser range finder's field of view:

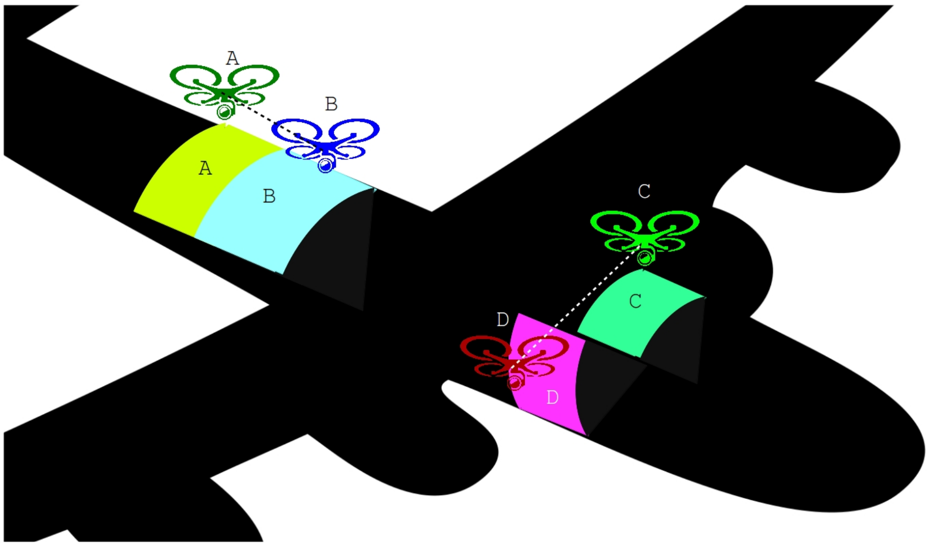


Figure 2-3: Field of view errors. On the left it is shown the movement from A to B. The scenes captured by the two sweeps can results very similar, this complicates the recognition of feature's correspondences. On the right it is shown the movement from C to D., In this case, the sweeps do not overlap due to the fast movement of the drone.

- **Similarity of consecutive sweeps** During the inspection the drone may be located very close to the aircraft surface. In this specific scenario, the field of view of the rotating LiDAR is limited. Consecutive sweeps collected in different positions may look really similar and therefore, difficult to detect the movement of the drone, using LiDAR input.
- **No overalap of consecutive sweeps** During the inspection the drone may change position relatively fast. In case the fast movement is combined with hidden spot surrounding, the consecutive sweeps may not have overlap areas, as illustrated in Figure 2-3 in the movement between C and D. This specific scenario may happen when the drone is facing the hull and all the other elements in the environment have a distance equals or higher than 30 meters.

These situations are rare since the rotation of the laser and the 270° of view will typically result in correspondences between consecutive sweeps.

2-2 Goal constraints

Next to the working environment, there are constraints posed due to positioning and inspection tasks.

2-2-1 Time constraint

The procedure of mapping is limited by the time of flight, which depends directly on the weight and the battery of the drone. To cover all the surface of the aircraft the robot needs to fly, in areas that are difficult to inspect and may require more time. Furthermore, to limit the error of the IMU and collect proper point clouds with the laser, the speed of the robot needs to be bounded. In conclusion, the limitation of resources, low speed and difficult areas where to fly, create the main constraints for the system while executing its positioning and inspection tasks.

2-2-2 Coverage

The mapping procedure needs to cover the entire surface of the plane including also hidden areas below the tail and the wings. Thus the robot has to fly around the plane with high precision. A careful analysis may explains the possibility of positioning in difficult area and mapping of these spaces. With the purpose of positioning and mapping for inspection, it is not necessary to map in detail complicated spots such as the engine internal areas.

2-2-3 Real-time processing

Real-time data processing is the execution of data in a short time period, providing near-instantaneous output. The processing starts when the data is obtained, so it needs a continuous stream of input data in order to provide a continuous output. The positioning system needs to react instantaneously to the data provided by the LiDAR and IMU. A real-time positioning system helps with a better estimation of the movement, and thereby improves the mapping and scan matching results as well.

2-2-4 On-board processing

On-board payload data processing includes the data acquisition, transfer, execution, storage and eventually transmission to an external computer. The amount of raw data generated by the sensors is large and at high frequency. Therefore, it is necessary to use various signal processing and compression techniques to reduce the amount of data and optimize their storage and execution process. It is equally important to have high-speed data links, sufficiently large on-board storage capabilities and CPUs able to handle data in the range of megabytes per second. The advantage of on-board processing is the possibility to render the robot autonomous.

2-2-5 Overview

Type	**Constraint**	**Specification**
Environmental	In the hangar	- Dynamic objects, - Surface reflections, - Hidden areas.
Goal	Time	Complete a mapping process in a single run, keeping low payload and increasing time of flight.
	Coverage	- Entire aircraft hull surface, - Fly in areas with resticted space.
	Real-time	Instantaneous data processing and position estimation
	On-board	Execute all the process with the computer on the drone

Table 2-1: Overview analysis of the constraints listed in the chapter.

Chapter 3

Requirements

In this chapter, the set of requirements that need to be met for the localization of a MAV into a simultaneously created map without GPS, are discussed. A division is made between functional and non-functional requirements. Both types of requirements are elaborated in a qualitative manner.

Functional requirements define the abstract functioning of the system - the process or the task it needs to fulfill. These requirements are used to derive possible working principles which can be used to fulfill these functions.

Non-functional requirements provide guidelines in making the choice between different working principles proposed for fulfilling a requirement. They are described in the second part of this chapter.

3-1 Functional requirements

From a hardware point of view, a set of sensory systems must be analysed which is capable to gather position measurements of the robot in GPS-denied environments.The position has to be absolute with respect to the map created simultaneously. The sensor input available are an Inertia Measurement Unit (IMU) coupled with a rotating 2D Laser Range Finder (LRF). The data captured needs to be analyzed for the creation of static features for scan matching technique. In order to classify and compare measuring principles and techniques, two main functional requirements are synthesized: measuring/sensing and data processing for localization.

3-1-1 Sensing

The positioning system relies on the ability of the sensor to track the movements of the robot in real-time. This is usually performed by motion sensors and range sensing. In this specific scenario we use IMU and LRFs.

Inertia Measurement Unit The IMU provides the acceleration and velocity of the drone along the three frame axis at 100Hz. The data need to be processed to estimate the motion of the robot. The main functionality is, indeed, the estimation of the position with respect to the map frame.

Laser Range Finders The LRF rotates 360 degrees, continuously, at constant speed; this gives the ability for the 2D LRF to view a 3D scene and consecutively the possibility to detect correspondences between points and features in consecutive scans. The ability to compute distances between features in the environment gives the ability to the system to estimate the position of the robot in the map.

3-1-2 Data Processing

The data processing functionalities are composed by three main sub-functions:

- **Data fusion** The data collected from IMU, LRF and motor need to be combined to estimate a position of the robot in the map frame.
- **Feature recognition** The point cloud needs to be examined and specific feature must be captured in the environment to perform scan matching techniques and position estimation.
- **Scan matching** The motion estimation performed by the IMU is corrected using scan matching techniques to compute the distances between corresponding points in two consecutive sweeps.

3-2 Non-Functional requirements

The set of non-functional requirements specify the behavior of the designed system. Moreover, non-functional requirements are used to judge the operation of a system. They are divided in two categories, system and software requirements. System requirements are related with the results expected by the implemented solution. Software requirements describe the software architecture organization and design principles which are to be achieved.

3-2-1 System requirements

System requirements are the configuration that a system must have in order for a hardware or software application to run smoothly and efficiently. Three main requirements are necessary to evaluate the results of the implementation: compactness, accuracy and complexity.

Compactness

Compactness is related to the size and weight of the system, or its subsystem parts. Most of the sensory elements together with some of the regulatory electronics have to fit on the (frame of) robot, so high compactness should be contemplated. Compactness is a crucial aspect on a drone.

Accuracy and precision

Both accuracy and precision are concepts that quantify the performance: accuracy is usually defined as the "closeness" of a measurement, precision as the "repeatability" [32]. The importance lies in the motion tracking precision and coverage of the mapping area.

Complexity

Complexity has multiple definitions, but is generally defined as the state of being intricate, i.e., complex or not being simple. In this setting complexity is coupled with the number of measuring methods or apparatus used. In the processing of the measured data complexity can be considered as the complexity of the algorithmic operations needed to combine data and estimate the desired values. In general holds: the more components are used in each of the subsystems, the higher the complexity. Higher complexity can be advantageous for system performance but more complicated to realize and more prone to faults. For this design a low complexity is preferred; however, a trade-off should be made between performance and complexity.

3-2-2 Software requirements

In software architecture it is necessary to define several requirements to take in consideration. In this sub-section seven factors are elaborated.

Maintainability

Maintainability can be described as how brittle the code is to change. It is important that the final solution is correctly commented and documented. The solution provided needs to be easy to interpret and any developer can sub-enter in the project and be able to modify data or set-up parameters with relatively simplicity.

Scalability

Scalability of the software is the ability to gracefully meet the demand of stress caused by increased usage. The duration of the inspection should not affect the results. It is also important that the flow of data does not increase with the time and the complexity of the algorithm is limited with the increment amount of data.

Reliability

A system is defined reliable when it does not fail. However, every electronic system may fail during the execution. It is important that the system can guarantee that the sensors continue to work during the entire procedure, or eventually communicate the error to the final user. Particular attention must be dedicated to the rotation of the motor which has to rotate continuously and at the same speed along all the inspection.

Extensibility

This factor represents points in the system that may be changed and extended. The integration of additional sensor in the positioning system is one of the causes of why this requirement is important. The software architecture need to have the possibility to extend the input interfaces number including possible extensions of the process.

Security

The security in a software system is the measure of system's ability to resist unauthorized attempts at usage or behavior modification. In this specific scenario the processes run on-board of the robot and a possible connection from external host is possible through the wireless connection installed on the drone. The LAN is private and password-protected. All the communications between hosts is within the private LAN.

Portability

Software portability is the ability to run the same applications on different platforms. The solution is tested and validated on the UP-Board (description in Appendix C). The solution is contained in a package in the ROS framework. Additional packages are required for the control of the motor and the acquisition of IMU data from external processes.

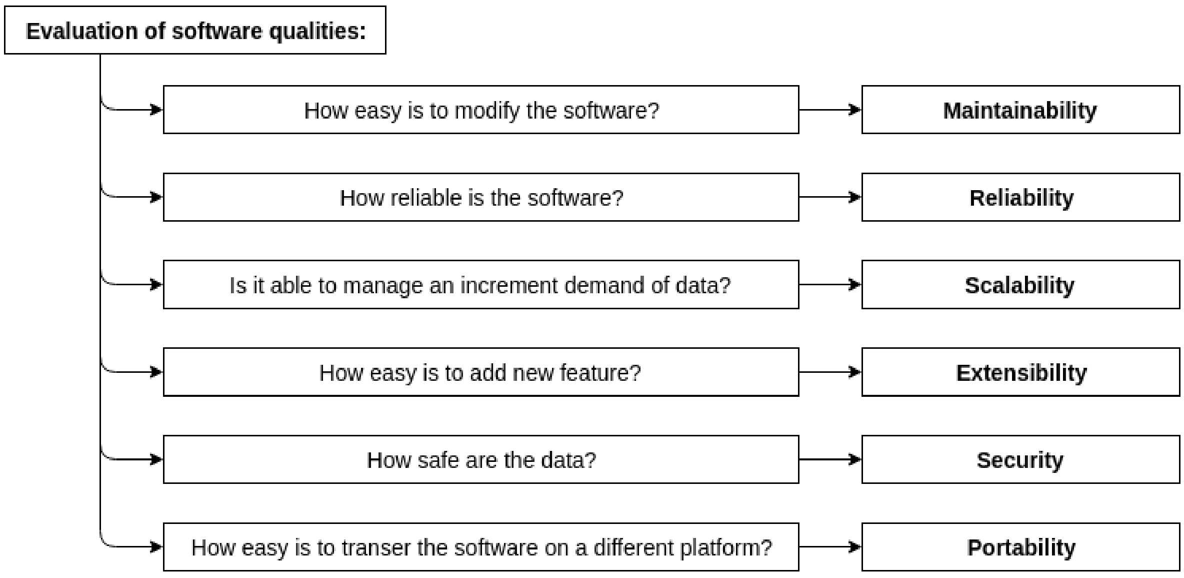

Evaluation of software qualities:
How easy is to modify the software?
Maintainability
How reliable is the software?
Reliability
Is it able to manage an increment demand of data?
Scalability
How easy is to add new feature?
Extensibility
How safe are the data?
Security
How easy is to transer the software on a different platform?
Portability

Figure 3-1: Main elements for the evaluation of software quality.

Chapter 4

Related works

The simultaneous localization and mapping (SLAM) problem asks if it is possible for a mobile robot to be placed at an unknown location and incrementally build a consistent map of this environment while simultaneously determining its location within this map [17]. A solution to the SLAM problem has been seen as a "holy grail" for the mobile robotics community as it would provide the means to make a robot truly autonomous.

The "solution" of the SLAM problem has been one of the notable successes of the robotics community over the past decade. It has been formulated and solved as a theoretical problem in a number of different forms. SLAM has also been implemented in a number of different domains from indoor robots to outdoor, underwater, and airborne systems. At a theoretical and conceptual level, SLAM can now be considered a solved problem. However, substantial issues remain in practically realizing more general SLAM solutions and notably in building and using perceptually rich maps as part of a SLAM algorithm.

This chapter aims to provide a broad introduction to this problem in robotics illustrating the related works. In the first part of the chapter, two different approaches linked with the problem will be illustrated focusing on the motion tracking of the robot. The explanation of the method is followed by the results reported by the authors. The chapter is concluded by an analysis of the approaches compared with the constraint and requirement of my project.

4-1 Approaches

Common frameworks can be implemented using basic probabilistic principles combined with sensor technologies such as Inertial Measurement Unit (IMU), LiDAR, cameras, sonar etc. Many researchers trying to solve this specific problem designed different possible solutions. A large set of theoretical implementations of the principles of SLAM into algorithms, such as EKF-SLAM, Fast-SLAM etc. can be found in the accompanying literature review [25].

The basic theories and algorithms have also been translated in practical experiments with real robots. During the first phase until the 2000's, the problem focused on ground robots with three degrees of freedom (DoF) and 2D mapping. However, due to the new technologies, the SLAM problem has been expended. Aerial robots, such as MAVs (Micro aerial vehicles) and UAVs (unmanned aerial vehicles), have been added to the problem and with their six DoF and the possibility to generate 3D maps make them the most studied robotic difficulty in SLAM research at the moment. In the following paragraphs are reported two real application that already have been presented in the Literature review but that are strongly connected with my solution to the problem that will be presented in the next parts.

Bosse and Zlot are two of the foremost pioneers in developing reliable data association techniques for SLAM. Their solution is called Atlas SLAM. They were one of the first researchers to address the online SLAM problem for large structured or unstructured environments using laser range data. More recently also Zhang et al. started to address the same problem proposing a solution called LiDAR Odometry And Mapping (LOAM) which is a real-time method for odometry and mapping using range measurement from a 2-axis LiDAR moving with six DoF [39]. These two different solutions have several points in common as well as many differences in the global approach to solve the same issues.

4-2 Robot motion estimation

4-2-1 Atlas SLAM

Bosse and Zlot demostrated that robot motion can be estimated from successive laser scans, provided that the frequency of the scans is high enough relative to the robot velocity to ensure sufficient overlap. The state of a robot is determined by position and velocity. During the iterative scan-matching process, the scans collected from the current location are compared to those from the scans collected before to update the current state of the robot. This comparison is implemented with a robust version of the iterative closest point (ICP) scan matching algorithm [13].

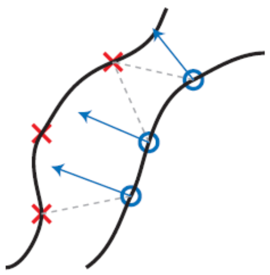


Figure 4-1: Scan surface point correspondence using normals. The dashed lines indicate the matches between the points from one scan (circles) to points in the other scan (crosses). The error between scans is that measured in the direction of the surface normals, indicated by the arrows [14].

Each iteration of the algorithm is composed of two steps. In the first step, point correspondences are found by associating each point from one scan to its closest point in the other scan. The second phase then determines a coordinate transformation that minimizes the error between the matched point correspondences. These two steps are repeated until convergence is achieved or a maximum number of iterations has occurred [14]. Since surfaces are not sampled at the same points in different scans, the Euclidean distance between points is not a good metric for evaluating the distance between scans [14]. Therefore, Bosse and Zlot use a metric that more accurately represents the error between the surfaces: the component of the vector difference between two scan points in the direction of the surface normal to the point in the current scan, Figure 4-1.

This method effectively matches surfaces rather than points, it is more accurate to call this algorithm "iterative closest surface" (ICS) rather than "iterative closest point".

4-2-2 LOAM

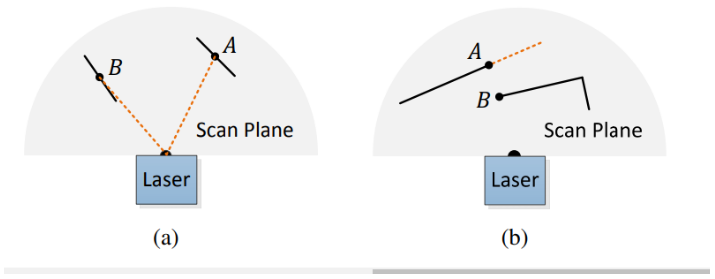


Figure 4-2: (a) The solid line segments represent local surface patches. Point A is on a surface patch that has an angle to the laser beam (the dotted orange line segments). Point B is on a surface patch that is roughly parallel to the laser beam. We treat B as a unreliable laser return and do not select it as a feature point. (b) The solid line segments are observable objects to the laser. Point A is on the boundary of an occluded region (the dotted orange line segment), and can be detected as an edge point. However, if viewed from a different angle, the occluded region can change and become observable. We do not treat A as a salient edge point or select it as a feature point [39].

The process of LOAM starts with extraction of feature points from the LiDAR cloud at time k, P_k. The feature points are extracted from P_k using only information from individual scans, with the co-planar geometric relationship. The feature points are divided into sharp edges and planar surface patches.

To estimate the robot motion two point cloud are considered:

- let P_k representing the previous sweep at time k
- let P_{k+1} representing the current sweep obtained from the laser scan.

During the selection of feature points the authors of LOAM want to avoid certain points, following these rules:

- do not insert two points that can be considered in the same neighborhood;
- do not insert points on local planar roughly parallel to the laser beams (Figure 4-2.a);

- do not insert points that are on boundary of occluded regions [24] (Figure 4-2.b).

To estimate the motion is necessary to find correspondences between these two clouds. Two different procedures are used for corner and surface points. Let E_{k+1} and H_{k+1} be the sets of edge points and planar points, respectively in P_{k+1}. The LiDAR odometry procedure will find edge lines from P_k, E_k, as the correspondences for the points in E_{k+1}, and planar patches as the correspondences for those in H_{k+1}. Note that at the beginning of the sweep $k+1$, P_{k+1} is an empty set, which grows during the course of the sweep as more points are received. The LiDAR odometry recursively estimates the 6-DoF motion during the sweep, and gradually includes more points as P_{k+1} increases. For each point in E_{k+1} and H_{k+1}, LOAM finds the closest neighbor point in P_k. Consecutively an edge line is associated with each edge point and it is represented by two points in P_k, Figure **??** shows an example and additional information will be included in the following parts of this report. A similar procedure is used to determine the correspondences for a planar point. In this case, always the closest point in P_k is searched and instead of a line, a planar patch is created by three points contained in P_k. Figure **??** shows an example of planar patch creation. With the correspondences of the feature points found, LOAM is able to derive the expressions to compute the distance from a feature point to its correspondence. The LiDAR motion is recovered by minimizing the overall distances of the feature points during the acquisition of the sweep.

4-3 Analysis related works

The constraints and requirements are the basic factors for evaluation of results. In the previous chapter, they have been clarified, specifically for the scenario of this project. In this section, Atlas-SLAM and LOAM will be analyzed and evaluated in relation to the limitations of the scenario proposed within an analysis of the experiments reported in the scientific articles published by the authors. Atlas-SLAM does not provide any open source version of the algorithm, indeed the software design analysis will be limited to the performance that they achieved in terms of real-time and onboard processing. The chapter is concluded with a reflection of these approaches and a specification of the initial approach used for the implementation of the proposed solution in this thesis project.

4-3-1 Atlas SLAM

Bentwing The most recent experiment of Bosse and Zlot with LiDAR SLAM is called Bentwing. The system does not require planar geometry common in built environments and can operate across the spectrum between natural and artificial sites without any algorithmic or parameter changes. To evaluate the accuracy of the Bentwing trajectory, the quad-copter is tracked with a Leica Viva TS12 robotic total station to produce an independent trajectory estimate.

The tracking results are illustrated in Figures 4-3. The differences between the Bentwing trajectory and total station estimates are observed to be within a few centimeters, with a smaller difference in the vertical compared to the horizontal direction. Bosse and Zlot noticed also differences in laser performances between the Hokuyo 2D laser and Faro Focus 3D

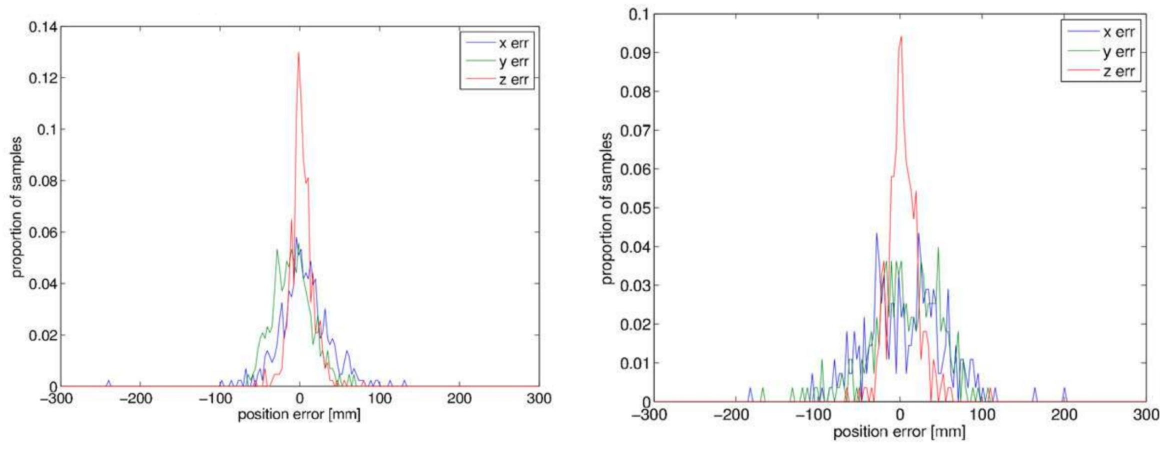


Figure 4-3: Comparison of the bentwing trajectory with the position estimated using a total station in two different environments.On the left, distribution of differences between the bentwing and total station trajectories in the global x, y,and z directions for the compound area. On the right, distribution of differences between the bentwing and total station trajectories in the global x, y,and z directions for the hill area [22].

terrestrial laser scanner(TLS). The Faro scanner has a range bias of 2mm and range precision of 0.6-2.2 mm (range, reflectivity, and configuration dependent) according to the manufacturer specification. The experiments show that areas with retroreflective tape appear mismatched, but that effect is due to differences in how the two laser scanners deal with artificially reflective materials (the Hokuyo range measurement tend to be considerably noisier on these surfaces). The distribution of the errors are represented quantitatively in Figure 4-4.

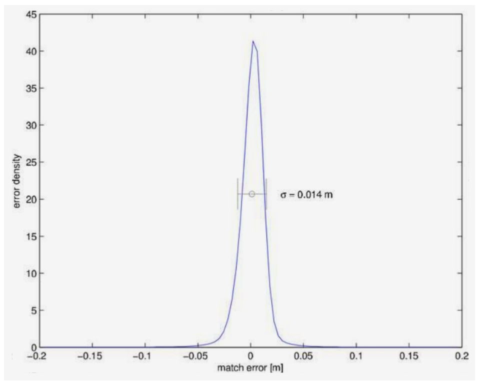


Figure 4-4: Distribution of errors of the bentwing point cloud with respect to the TLS scan data. Most points from objects that have moved have been ignored by thresholding the errors. Some points near the boundaries of these objects still remain as a result of this simple thresholding technique, which may contribute to inïňĆate the error statistics. [22].

The authors observed that the overall error has little bias (close to zero mean), and a standard deviation of 1.4 cm. This result verifies that the Bentwing point cloud accurately represents the environment (and that the visible errors discussed above truly are a small number of outliers), with the main differences between it and the TLS data occurring due to the lower precision of the Hokuyo laser scanner. Bentwing requires off-board processing to achieve real time performances.

4-3-2 LOAM

During the experiments, the algorithms run on an external computer and the odometry and mapping processes run on two different cores. The software developed by the authors and the datasets are publicly available. The author tested the application in indoor and outdoor environments to test the versatility of the solution. To evaluate local accuracy of the maps, a second set of LiDAR clouds from the same environments are collected. The laser is kept stationary and placed at a few different places. The two point clouds are matched and compared using the point to plane ICP method [30]. The density of error distributions is shown in Figure 4-5 shows the density of error distributions.

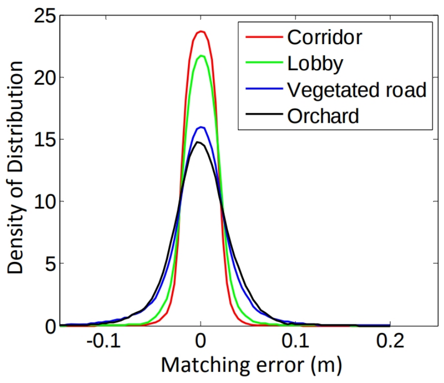


Figure 4-5: Matching errors [39].

It indicates smaller matching errors in indoor environments than in outdoor. The result is reasonable because feature matching in natural environments is less exact than in manufactured environments [39]. None of the experiments is executed with a drone. Recently the author published videos online proving great results with the usage of LOAM with MAVs, however, a scientific article has not been published yet, with the latest version of the application.

4-3-3 Conclusion

The results obtained by both the research fulfill some of the constraints and functional requirements listed in Chapter 2 and 3. Atlas-SLAM is not open source, instead, LOAM is a public ROS package.

After installing the package of LOAM in the project's working space, it was noticed that many of the software requirements were missing. The code was badly maintained, difficult to extend, and was not simple to test the reliability of it. A new software design and implementation of LiDAR odometry and mapping has proven necessary. In the following chapters, an elaboration of the solution proposed for this project is given, which synchronizes the sensors and produces an estimation of the odometry using IMU and a rotating 2D LRF; the method has been named Tracking and Mapping (TAM).

Part II

System design

System design is the process of defining the elements of a software system such as the architecture, modules and components, the different interfaces of those components and the data that goes through the system. It is meant to satisfy specific needs and requirements of a business or organization through the engineering of a coherent and well-running system [36]. In this part of the thesis, the system design approach is described, which is aimed to explain the decisions made during the development of the software. To this end an explanation of the sensing system is described in Chapter 5, followed by a more detailed explanation of the software and all the techniques, formulas and technologies used. Particular attention will be invested on the main differences and divergences from TAM to the related works mentioned in the previous part. Following this line of reasoning the part is divided into chapters, with the objectives and approaches as illustrated in Figure 4-6.

Objective:

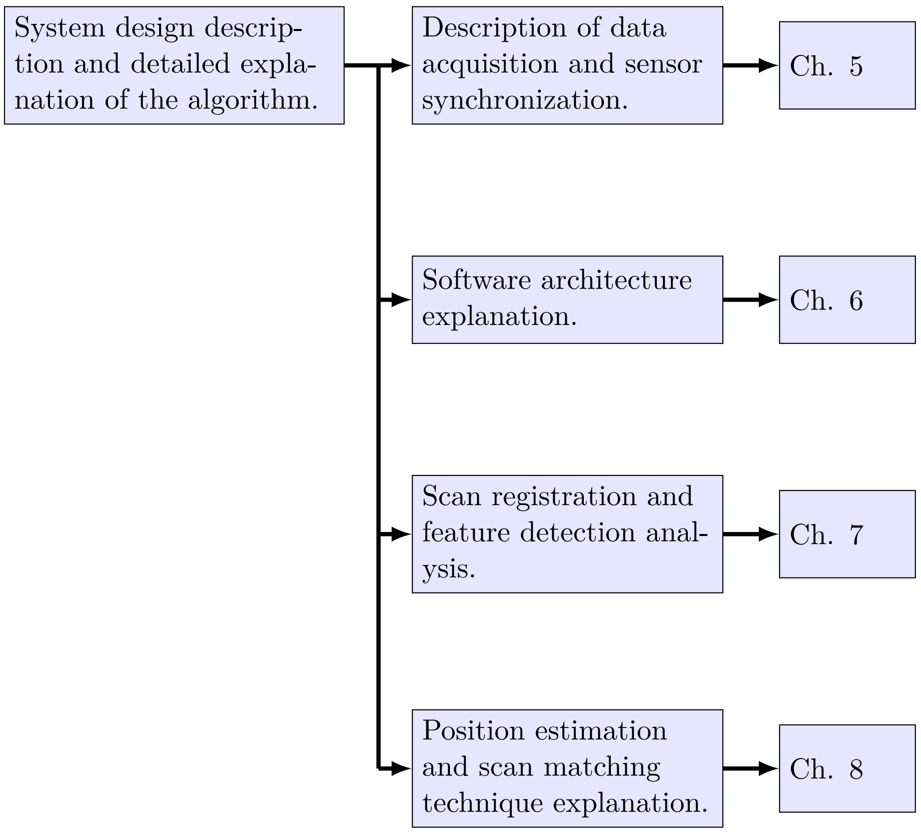

System design description and detailed explanation of the algorithm.
Description of data acquisition and sensor synchronization.
Ch. 5
Software architecture explanation.
Ch. 6
Scan registration and feature detection analysis.
Ch. 7
Position estimation and scan matching technique explanation.
Ch. 8

Figure 4-6: Structure of conceptual design and related works part.

Chapter 5

Sensing system

Sensor fusion combines several sensor measurements and mathematical models in order to obtain an enhanced state estimate. The association and integration of sensor data requires time and position synchronization [21]. TAM is composed of three main components. Figure 5-1 shows the three main input in the positioning and mapping systems:

- **Motor,** which acts as an actuator and its encoder as a sensor.
- **LRF,** laser range finder.
- **Drone,** with several sensors incorporated such as IMU, barometers etc.

In this chapter, the three different sensors are described in terms of data-processing and synchronization between them and the ROS framework.

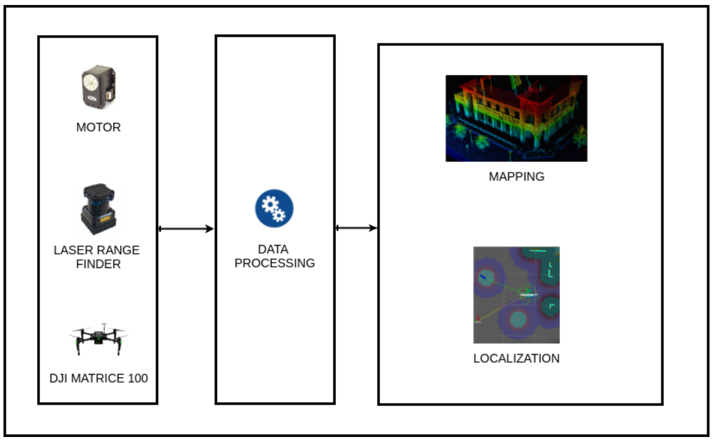


Figure 5-1: Overview of the sensing system

5-1 Internal sensors of the MAV

As mentioned in the introduction of this report the drone provided by Mainblades Inspections is manufactured by DJI, Matrice 100. Inside the drone, several sensors are located such as a compass, Real Time Kinematic (RTK) GPS positioning and IMU (the related specification of the hardware is reported in Appendix C). The IMU's state information are a combination of gyroscope and accelerometer.

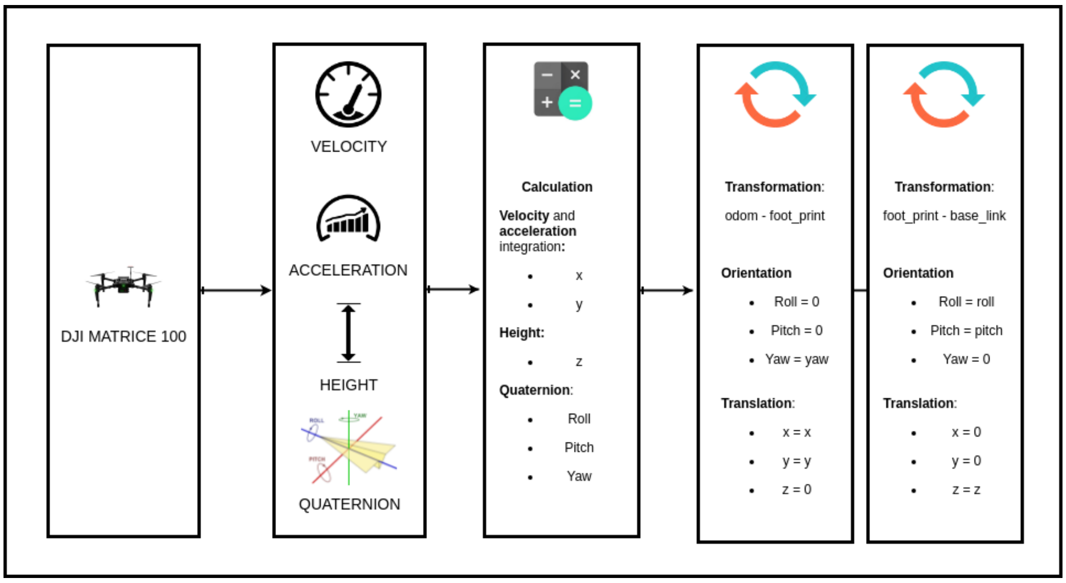


Figure 5-2: Data flow for estimation of the position with DJI's IMU.

Figure 5-2 shows an overview of the system. The drone has a software interface for the tuning of certain parameters that affect the information received as input. The values received from the IMU (velocity and acceleration) have the possibility to be expressed in two different modes:

- *Raw* mode, this means that the value referred to the body frame of the robot.
- *Fusion* mode where the acceleration and velocity are transformed into the ground frame with an internal process of the robot.

After an analysis of the performance of the two different configurations, the "fusion" mode is used for this project.

Receiving the information already transformed in the ground frame simplifies the procedure of motion estimation. The procedure integrates the acceleration according to the basic rules of body dynamics:

$$v = \int a dt \tag{5-1}$$

$$p = \int v dt \tag{5-2}$$

where v is the velocity of the body, a its acceleration and p the position. It follows:

$$p \approx \int v dt + \frac{1}{2} a(dt^2) + p_0 \tag{5-3}$$

where dt is the time interval considered. The altitude of the robot is directly obtained by the DJI autopilot. The data obtained from the calculations (x, y, z, roll, pitch, and yaw) are then published in the *tf* framework [1].

5-2 Motor - Dynamixel

The motor selected for the implementation of the design is MX-28T Dynamixel Robot Servo Actuator. It is equipped with a contact-less magnetic encoder, and up to 3mpbs using the new TTL 2.0 bus. The servo has the ability to track its speed, temperature, shaft position, voltage, and load.

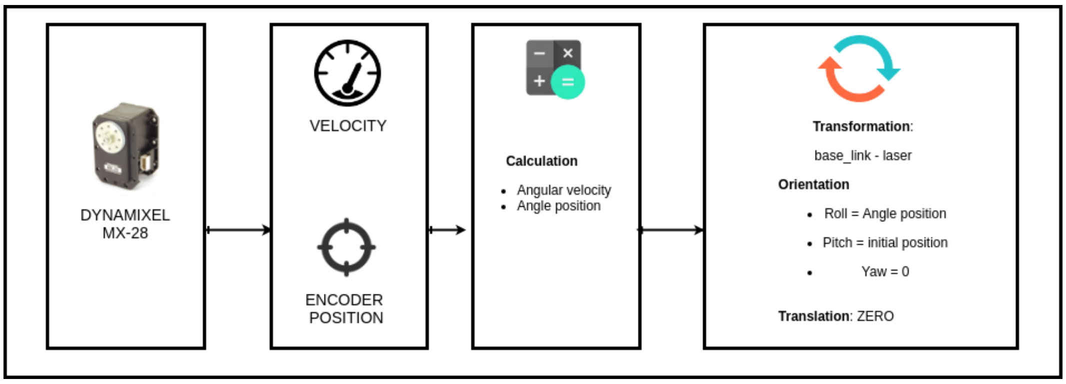


Figure 5-3: Data flow for estimation of the laser position.

The data used by our application are the velocity and the position from the encoder, as illustrated in Figure 5-3. The two data are then processed to compute angle position and angular velocity:

$$v_a = v_m \cdot rpm_c \cdot \frac{2\pi}{60s} \tag{5-4}$$

$$x_{rot} = 2\pi \cdot \frac{p_m}{p_{max}} + p_{init} \tag{5-5}$$

where v_a is the angular velocity (expressed in degrees per second) and x_{rot} is the rotational angle around the x *axis* of the laser frame (expressed in degrees), rpm_c is a constant value (0.11rpm [5]) that express the repetition per minute of the single unit of v_m, v_m and p_m are the velocity (expressed in unit) and the position captured from the motor, p_{max} is the maximum value of the position obtained by the motor, p_{init} is the initial position of the laser, what we call zero-position, which required to be tuned and checked before the flight test.

[1] *tf* keeps track of all these frames over time. More information about the *tf* framework are reported in the Appendix B.

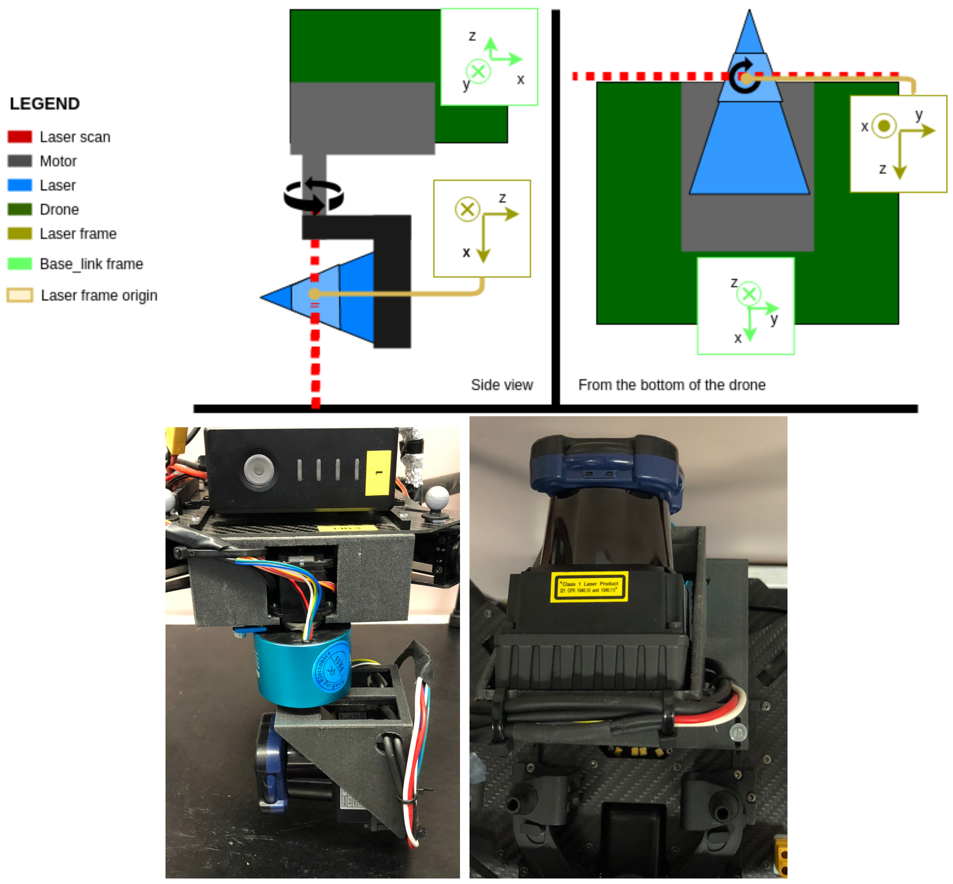


Figure 5-4: Motor position and LRF rotation. The picture on the left shows the position of the laser and motor with respect to the drone from a side view. The laser frame is positioned along the red line in the center of the laser. The image wants to show that the rotation axis is centered in the scan plane, indeed the movement of the laser is only a rotation along the x-axis. **NB: The frame showed in the picture is not centered in their origin but they only show the orientation of the axis.** The yellow arrow connect the laser frame to an estimated position of its origin.

The reason why the motor rotates around the x axis is because the initialization of the laser frame has been rotated with respect to the *base_link* frame. Figure 5-4 shows the position of the motor respect to the drone. The initialization of the *laser* frame is a translation from *base_link* along x and z *axis* and 90 degrees rotation around y. Furthermore it is important to set a reference position for the angle position. This parameter has been tuned to align the z *axis* of the *laser* frame with the x *axis* of *base_link*.

5-3 Laser range finder

The last element of the sensing system is the laser range finder.The sensor used for this application is the Hokuyo UTM30LX, further technical information can be found in Appendix C. The field of view covers 270° area, scanning range with 0.25 degrees angular resolution and it operates at 40Hz. The message provided by this sensor contains: time stamp, the reference frame, and the list of ranges collected. At this step, the integration between the three sensors can be computed. The ranges are translated into points using the transformation described above between *laser* frame and *odom*. The procedure is illustrated in Figure 5-5. At the end of the procedure, a point cloud is created and the scan registration procedure is ready to start.

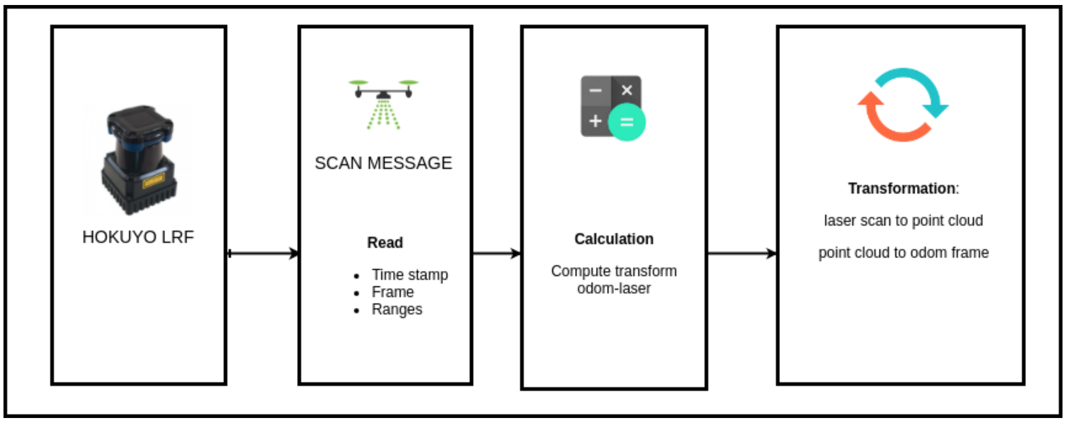


Figure 5-5: LRF data processing schema.

Scans and sweeps The point cloud generated by the single input of the LRF is composed by points relying in the same 2D plane; this point cloud is named *scan*. During the rotation motion of the laser several consecutive scans are generated. Once the laser complete half of the rotation it scans already all the 360° environment surrounding. The *scans* collected during this motion are grouped into a larger point cloud which is called *sweep*. The current configuration of the system set the rotation speed to $1\pi/s$; due to the frequency of the LRF, the number of *scans* per *sweep* is around 40.

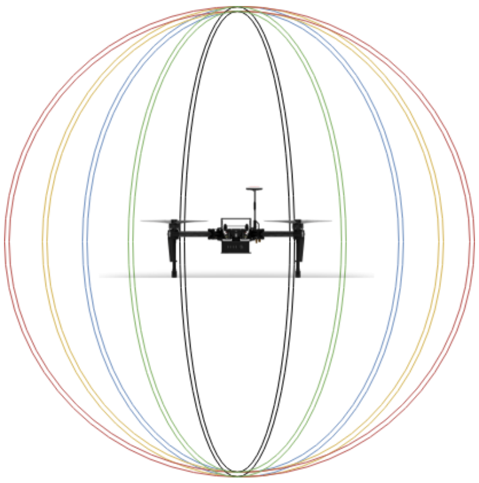


Figure 5-6: A single colored line represents a *scan*. All the scans grouped together are a *sweep*.

5-4 Overview

The sensing system represents the set of hardware component involved in the solution to solve the problem for localization of a MAV in a 3D GPS-denied environment. The analysis is summarized in Table 5-1. The three main components provide three different information sources that must be processed to transform the data acquired into useful data for localization and mapping purposes. The update rate for communication between the different devices creates one of the main issues for the synchronization of them. In the next chapters, a deep explanation of how the data are used and processed in TAM is reported.

Sensing system		
Component	**Outputs**	**Usage**
DJI drone's sensors	Velocity Acceleration Height Orientation	Position estimation
Motor - dynamixel	Speed Position	Compute position of the LRF frame in real time
LRF	Laser scans	3D point cloud Localization corrections

Table 5-1: Overview of the components and their main output and usage

Chapter 6

Architecture and implementation

All the elements and processes present in TAM have been developed using ROS (Robotic Operative System). The infrastructure that connects all the different elements grew from the beginning of the project till the end. It is important to define interfaces, communication protocols, and rules between the different processes to allow the system to achieve the software requirements mentioned in Chapter 3. In this chapter the Software Architecture of the system is analyzed. Furthermore, an introduction to the libraries used and a detail description of the point cloud data structure is included.

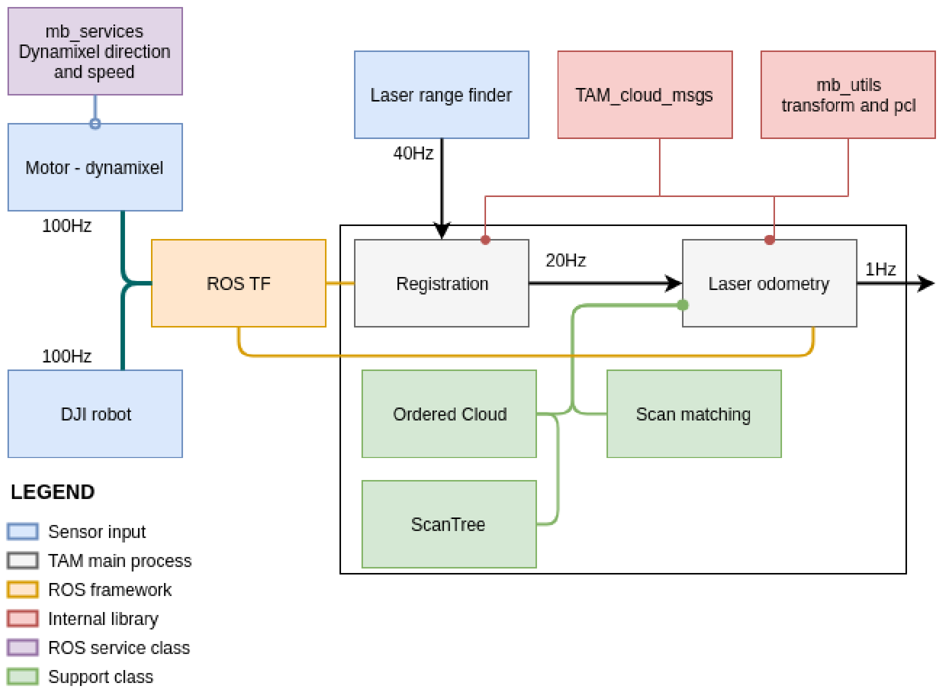


Figure 6-1: Software architecture of TAM and external processes. The graph shows a clear overview of all the elements included to solve the SLAM problem.

Architecture

In the previous chapter, the technologies and the data flow of the different sensors used in TAM to perform SLAM have been introduced. They represent the input in the system and are shown in Figure 6-1 in the blue boxes. All of them are connected to *Registration* which is the initialization process of TAM. The output of *Registration* is the point cloud dived into groups based on point feature and it is directly communicated to *Laser odometry* at a speed that can vary between 10 to 40 Hz. The *Laser odometry* process uses external classes such as *Ordered Cloud* and *Scan Matching* for point cloud organization and data processing. Odometry produces in output the position of the robot and a point cloud containing the last *sweep*. The *Registration*, *Laser odometry* process, and *Scan matching* class will be described in detail in the next chapters. In this chapter other relevant elements of the architecture are presented such as the communication between different processes, the organization of the point cloud in *Ordered Cloud* and *ScanTree* and the internal and external library used, such as *TAM_cloud_msgs* and *mb_utils* illustrated in Figure 6-1 in the red blocks.

6-1 ROS communication

All the communications between the main blocks of TAM (represented in grey in Figure 6-1) use the ROS protocol publisher/subscriber. The concept of publishers and subscribers is a basic principle of the ROS framework. It allows the synchronization and the communication between different components of the system that can also be located on different devices and communicate within the ROS network. TAM is currently installed entirely onboard and it is not expected to have communications with external devices, at the moment. Despite the onboard processing, the ROS communication rules (publisher/subscriber) are used to transfer the information between the different blocks. The reason for this choice is the possibility to distribute the system between different devices in the future.

6-2 Point cloud organization

The point cloud is initially organized into a single list of points. However, the Laser odometry node treats points differently based on their feature. Having this in mind an attempt was made to elaborate a data structure that can easily separate points based on their feature and furthermore simplify the research of points in consecutive scans. Indeed several sweeps are reorgaized in a data structure that would support these characteristics.

The structure is mainly composed of a vector of Ordered Cloud Objects, and it is illustrated in Figure 6-2. Where each Ordered Cloud contains all the information of a sweep. It is composed by a vector of ScanTree, where each element expresses a single scan and one more ScanTree Object which contains the entire sweep. A ScanTree Object contains four point clouds and a time stamp. The number of point clouds is four because the points are divided into two groups and each of these two groups is saved into two different structures: pcl::KdTreeFLANN and pcl::PointCloud. The main difference between these two data structure is the organization of the points. In fact, the structure provided by FLANN (Fast Library for Approximate Nearest Neighbors) results faster in the research of the closest point in the cloud. On the other side, the list structure of the point cloud allows us to access a specific point using an index (in FLAAN is not possible). The Ordered Cloud structure occupies more memory than a single list because each point is replicated four times however it speeds up in the processing after the registration process. It is important to consider that this structure has a constant size that does not increase during the execution of the process because once the vector is full and a new sweep is generated the older sweep is removed from the list and replaced with the new one.

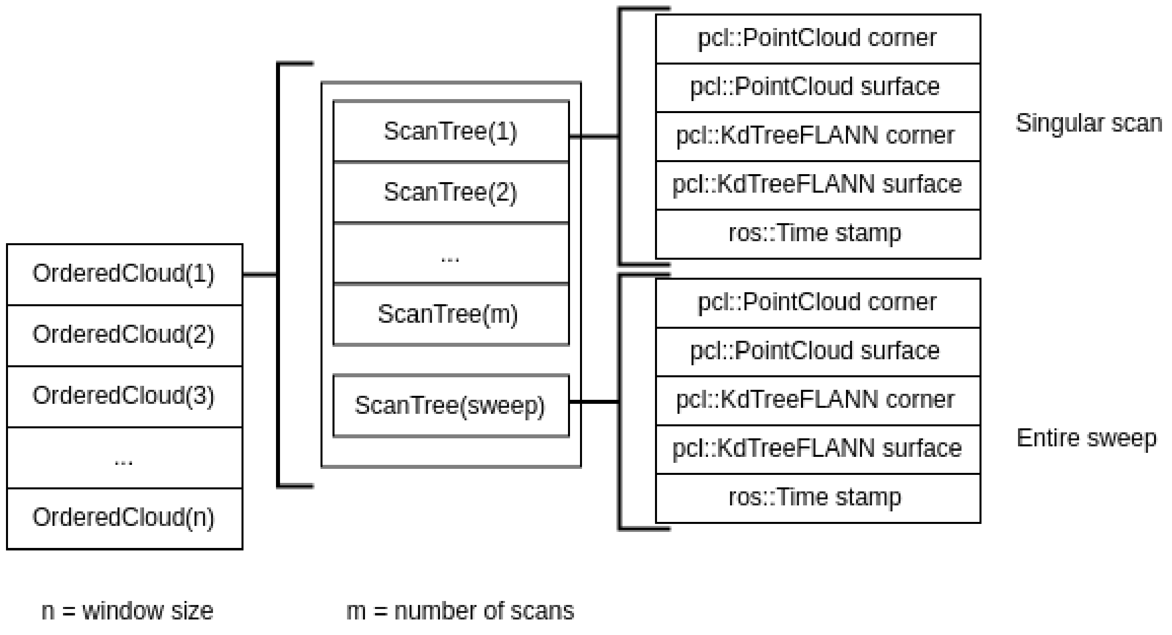


Figure 6-2: Internal organization of the point cloud. The point cloud is divided into Ordered Cloud objects, where each element of the array represent a sweep. The Order Cloud object is composed by a vector of ScanTree, where each element represents a single scan. This structure helps the research of correspondences between consecutive sweeps.

6-3 Libraries and packages

During the development of TAM many operations have been organized into software methods and grouped into libraries. Furthermore also external libraries to solve specific problems have been largely used in TAM. In this section a brief introduction of internal and external libraries is reported.

6-3-1 Internal libraries

During the development of the software system many functions were used in several different processes. To simplify the code various internal libraries have been created that are used only inside the TAM node.

ROS msgs The first one is related with the communication protocol between the different blocks. In fact, a specific typology of messages has been designed to facilitate the communication. It is contained in the *TAM_cloud_msgs* library. The message contains four point clouds that are the output of the feature selection during the Registration procedure.

Funtion libraries Another internal library is mb_utils which is composed of three files:

- pcl: it is used to transform a PointCloud object into a sensor_msgs::PointCloud2 object, adding a time stamp and the frame_id to the message's header.
- Eigen rotations: contains all the methods related to rotation transformation and their derivatives. The rotation matrices are used before the optimization during the scan matching process by Mapping and Odometry nodes.
- Transform: it contains a method that simplifies the lookup for transformation contained in the *tf* framework. It returns an error in case the transform is not available.

6-3-2 External libraries

Besides the internal libraries and the code developed for Tracking and Mapping, also additional dependencies are included in the project:

- PCL Point Cloud Library. The library contains state-of-the-art algorithms for filtering, feature estimation, surface reconstruction, registration, model fitting, and segmentation. PCL is supported by an international community of robotics and perception researchers (http://pointclouds.org) [31]. It is used in TAM for point cloud organization and operation with points such as filtering and transformation of entire point clouds.
- Eigen. The main template library for linear algebra operations: matrices, vectors, numerical solvers, and related algorithms. It is a versatile, fast and reliable library. This library is used for matrix and vector operations internally in TAM.
- Ceres. Library developed by Google to solve nonlinear optimization problems. It is used alternatively to OpenCV (used in the original version of LOAM) to solve the optimization problem in Odometry and Mapping. Additional information of this library are reported in Appendix B and in the Odometry Chapter.

Chapter 7

Registration

The registration process captures the data from the laser and transforms them into a 3D point cloud. Furthermore, it divides the point cloud into different groups based on the feature recognition procedure. In this chapter, the analysis of the feature recognition technique and also of the interfaces in input and output of the scan registration process are presented. The chapter starts with an explanation of the data in input, it continues with an analysis of the feature selection process and it is concluded with the explanation of the output provided.

7-1 Input interface

The scan registration is the first of the four main processes involved in TAM. It receives in input the scan message from the laser and it uses the information collected from the TF framework to remap the points in the 3D scene. The input interface is composed by a ROS::subscriber object which is associated with a callback which triggers whenever a new message from the laser is published. The messages from the current LRF installed on the drone are published with a frequency of 40Hz.

7-2 Feature selection

The feature selection procedure associates a tag to each point in the scan. There are four different type of tags:

- extreme corners: are the points at the most extreme of a surface. As illustrated in Figure 7-1, they are accumulated in narrow areas between two surfaces, such as ceiling-wall or wall-floor.
- corners: can be a neighbour of a extreme point or a point collected in presence of other corners with lower smoothness value.

- extreme surfaces: are the points relying on the center of a surface. The floor usually contains many extreme surface points.
- surfaces: are all the remaining points.

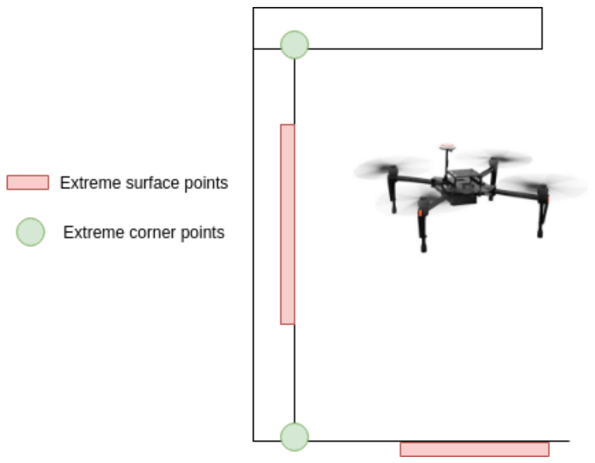


Figure 7-1: The image illustrate the displacement of the extreme points in the environment.

The division is based on the smoothness factor assigned to each point. The smoothness factor is computed using this formula:

$$c = \frac{1}{|S| \cdot ||X^L_{(k,i)}||} || \sum_{j \in S j \neq i} (X^L_{(k,i)} - X^L_{(k,j)}) || \tag{7-1}$$

where S is a window of points surrounding the selected point $(X^L_{(k,i)})$, $X^L_{(k,j)}$ are the points in the window. The formula computes an average of the distances between the selected points and the points in the window. The process continues with the division of the scan into sub-scans, each sub-scan is a set of consecutive points; all the sub-scans have the same number of points. Into each sub-scan, the points are ordered based on their smoothness value. The points with the maximum c's are classified as extreme corners. The points that are included in a defined interval of smoothness are also classified as corners but not extreme. The points with the minimum c's are classed as extreme surfaces, all the remaining points are inserted into the surface set. The trade-off between sub-scans division and number of extreme points is an important element to evaluate, especially because the scan matching technique relies on the correspondences creation between extreme points from different sweeps.

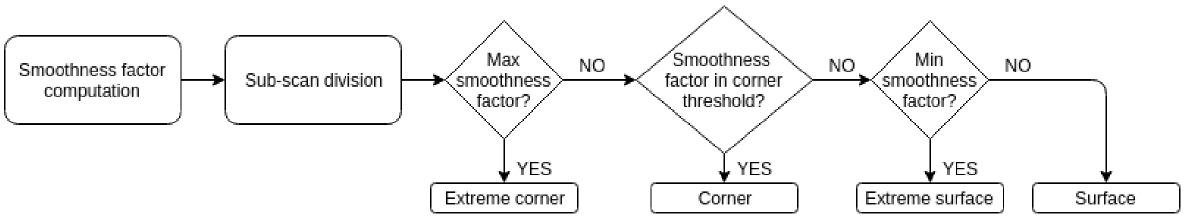


Figure 7-2: Data processing in registration process.

7-3 Output interface

One of the main differences between TAM and LOAM is the publication of the points. A specific ROS message has been defined in *TAM_msgs_cloud* library, where a single message groups the four different point clouds computed during the feature recognition method. Instead, LOAM published a single point cloud containing all the points. In LOAM the differentiation between the points based on their feature is made using a variable contained in the point object. Using the solution of LOAM, it is necessary to divide the point cloud into corner and surface (operation used multiple times in the further processes Odometry and Mapping) requires an iteration between all the point and a check condition on the specific variable to identify the type of the point. This iteration is not necessary anymore in TAM.

Difference between sweeps and scans Each scan is processed independently during the feature recognition process. At the end of the feature recognition method, the output message can be prepared and published. Not all the scans are published immediately, but instead, multiple scans are collected and published together. Indeed, there are three different case scenarios at this point:

- scan ready but does not reached the number of scans to send;
- scan completed and reached the number of scans to send;
- sweep completed.

In the first case, the points are saved respectively into the four point clouds but no messages are published. In the second case the points are saved into the four point clouds, however only the extreme points are saved into a message and published. After the publication of the message, the point cloud containing the extreme point is cleared. In the last scenario, the points are saved respectively again in the appropriate point cloud and the message containing all the four point clouds is published. In this scenario, the value corner and surface of the message contain all the points of the sweep, instead, the extreme ones only contain the points collected from the last publish action of the second scenario. The message published in the second scenario is named *update message* and the message published at the end of the sweep is named *sweep message*. The *update* message is published with a frequency varying between 10 to 40Hz and depends on the number of consecutive scans collected into the message, the frequency remains constant along all the process. Instead, the *sweep* message is published constantly at 1Hz (same frequency of half rotation of the motor).

Once the feature recognition procedure is concluded the points are accumulated into four different point clouds. These point clouds are published, only once the entire sweep is completed. Besides this process, also other update messages are published at a higher frequency. The update messages contain only the extreme points. Later in the explanation of the odometry process will be clarified why only the extreme points are contained in the update message.

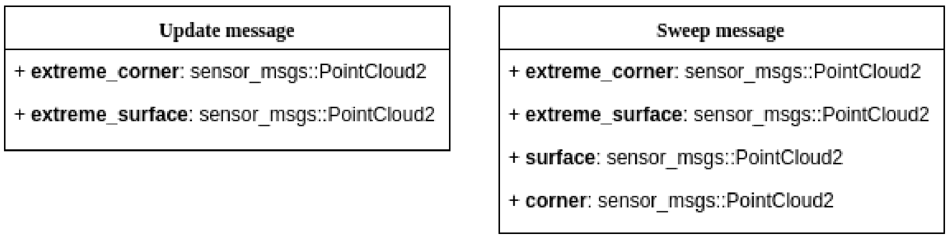
Chapter 8

Laser odometry

The information captured by the LiDAR can be used for position estimation, as have been stated already in the previous chapters. In this chapter, the procedure for localization of the robot using the laser sensor is explained. The messages published by the registration process are captured in input by the odometry node. Then the position estimation is computed and the points previously received are corrected and are registered into the data structure. The chapter is concluded with the explanation of the output interface which is influenced by the mapping process.

8-1 Input interface

The input messages in the process are two, and they are both from the scan registration node. As mentioned already in the previous chapter, they are *update* messages and *sweep* message.



Update message

+ **extreme_corner**: sensor_msgs::PointCloud2

+ **extreme_surface**: sensor_msgs::PointCloud2

Sweep message

+ **extreme_corner**: sensor_msgs::PointCloud2

+ **extreme_surface**: sensor_msgs::PointCloud2

+ **surface**: sensor_msgs::PointCloud2

+ **corner**: sensor_msgs::PointCloud2

Figure 8-1: Message structure in input in the odometry node.

- The *update* message containes extreme corner and extreme surface and it is received at a constant frequency. For reasons of efficiency not all the scans are streamed as soon as they are captured, but they are grouped and the *update* message contains a set of them. According to the structure and the data processing of the odometry node, the accumulation of several scans into a single message does not affect the performance results. This is due to the limit number of extreme points filtered per scan. It is better to accumulate them and send a larger amount of points to improve the performances in terms of time and computational efficiencies.

- The *sweep* message contains all the points collected during one rotation of the laser, separated into four different point clouds. The corner and surface elements are organized in vectors where each element represents a single scan. This structure helps the storage of the points into the Ordered Cloud data structure. The sweep message is received with a frequency of 1Hz. Whenever a *sweep* message occurs all the points are inserted per scan in a temporary Ordered Cloud object, that at the end of the procedure is stored in the vector of the latest sweep.

8-2 Point cloud processing

The process is composed of two main steps: point validation and transformation estimation. Not all the extreme points are inserted in the transform estimation algorithm, they have to be filtered and selected. The selection of points and the transform estimation are explained in the next sections.

Point validation

The validation algorithm consists of the identification of points that have clear correspondences with points contained in the previous sweep. These correspondences are computed through the creation of feature-shapes in the previous sweep where the points of the new sweep are supposed to rely on. This shape identification algorithm is different for corner and surface points.

Corner correspondences The validation of a corner point (p_c^i) starts with the research of the closest point in the previous sweep (p_1^i). In case the distance is not bounded by a certain threshold the point is discarded. Instead, if the distance is valid the closest point is inserted into a point cloud, *correspondence_points*. Consecutively one more point is researched and pushed into *correspondence_points* which is the closest point to p_1^i in the consecutive or previous scan, p_2^i. The two points, inserted in *correspondence_points*, are then used for the identification of a straight line where the corner point is supposed to rely on. The computation of a line passing between two points is calculated with the equation [8]:

$$line = \begin{bmatrix} x_{p1} + (x_{p2} - x_{p1})t \\ y_{p1} + (y_{p2} - y_{p1})t \\ z_{p1} + (z_{p2} - z_{p1})t \end{bmatrix} \tag{8-1}$$

where t is a parameter, $line$ is the vector that describes the line in 3D, x_{p1}, y_{p1}, z_{p1}, x_{p2}, y_{p2}, z_{p2} are the coordinates of p_1^i and p_2^i. In addition, also the error is computed. The error is the distance between $line$ and the corner point p_c^i. The error is computed with the equation [8]:

$$error = \frac{|(p_1^i - p_c^i) \times (p_2^i - p_c^i))|}{|(p_1^i - p_2^i)|} \tag{8-2}$$

The four values computed (three values identify the line and one for the distance) per point are inserted in a vector, named *coefficient*.

Surface correspondences The validation of the surface points is similar to the validation of corner points, the only difference is the creation of a surface instead of a line where the point relies on. The surface point (p_s^i) requires three additional points from the previous sweep to create a 3D plane. The first point is the closest point to p_s^i in the previous sweep (p_1^i). The other two are the closest points to p_1^i in the previous or consecutive sweep, p_2^i and p_3^i. Once the three points have been selected the plane and the error are computed using the formula [9] [3]:

$$plane = \frac{(p_2^i - p_1^i) \times (p_3^i - p_1^i)}{|(p_2^i - p_1^i) \times (p_3^i - p_1^i)|} \tag{8-3}$$

$$error = \frac{plane}{|plane|} \cdot p_s^i \tag{8-4}$$

The four values (three values identify the line and one for the distance) computed per point are inserted in the same *coefficient* vector of the corner points.

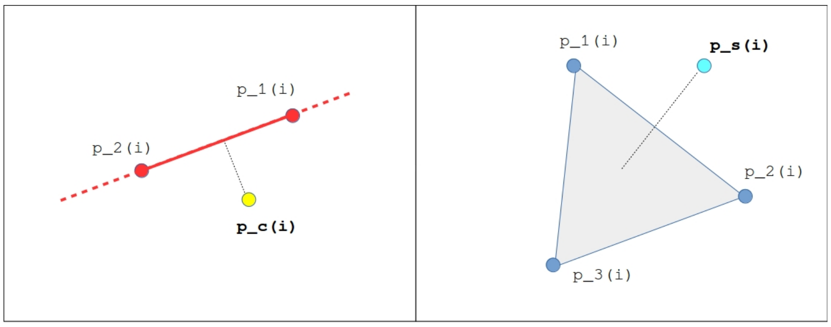


Figure 8-2: On the left is shown the line created with two points from the previous sweep and the yellow point own to the current sweep. The algorithm identifies that the point is supposed to rely on the line and indeed identify the distance point-line as an error of the odometry estimation of the robot. On the right side, a similar concept is illustrated. In this case, the light blue point is a surface point in the new sweep and the three blue points own to the previous sweep. The three points are used to create a plane where the light blue point is supposed to rely on. Using the same technique from the corner point the error is computed.

The correspondence research is facilitated by the data organization implemented (Ordered Cloud). In fact, the research of points in previous sweeps, the identification of the scan where a specific point has been captured and the research of closest points in consecutive scans, is significantly simplified.

8-3 Transformation estimation

The core of the odometry process relies on the estimation of the transformation from the previous position to the current one. As already mentioned in Chapter 5, this action is performed with the synchronization of the inertial measurement unit and LRF. The transform estimation procedure is composed of four sequential steps, and they are illustrated in Figure 8-3.

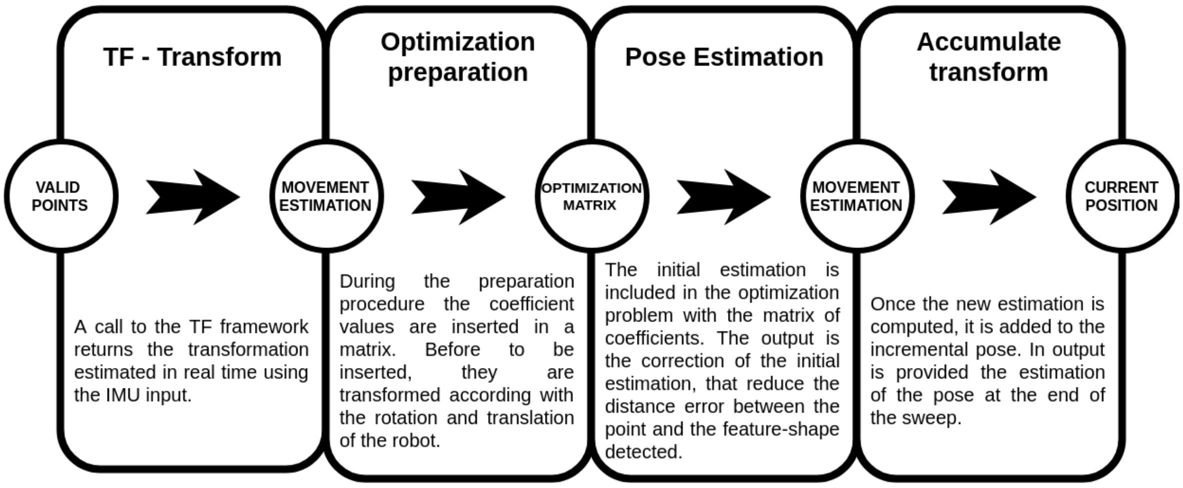

TF - Transform
Optimization preparation
Pose Estimation
Accumulate transform
VALID POINTS
MOVEMENT ESTIMATION
OPTIMIZATION MATRIX
MOVEMENT ESTIMATION
CURRENT POSITION
A call to the TF framework returns the transformation estimated in real time using the IMU input.
During the preparation procedure the coefficient values are inserted in a matrix. Before to be inserted, they are transformed according with the rotation and translation of the robot.
The initial estimation is included in the optimization problem with the matrix of coefficients. The output is the correction of the initial estimation, that reduce the distance error between the point and the feature-shape detected.
Once the new estimation is computed, it is added to the incremental pose. In output is provided the estimation of the pose at the end of the sweep.

Figure 8-3: The diagram shows the flow of the Odometry process. It is a sequential procedure composed of four steps: *tf* -transform, Optimization preparation, Pose estimation and Accumulate transform.

8-3-1 Get the data from *tf* framework

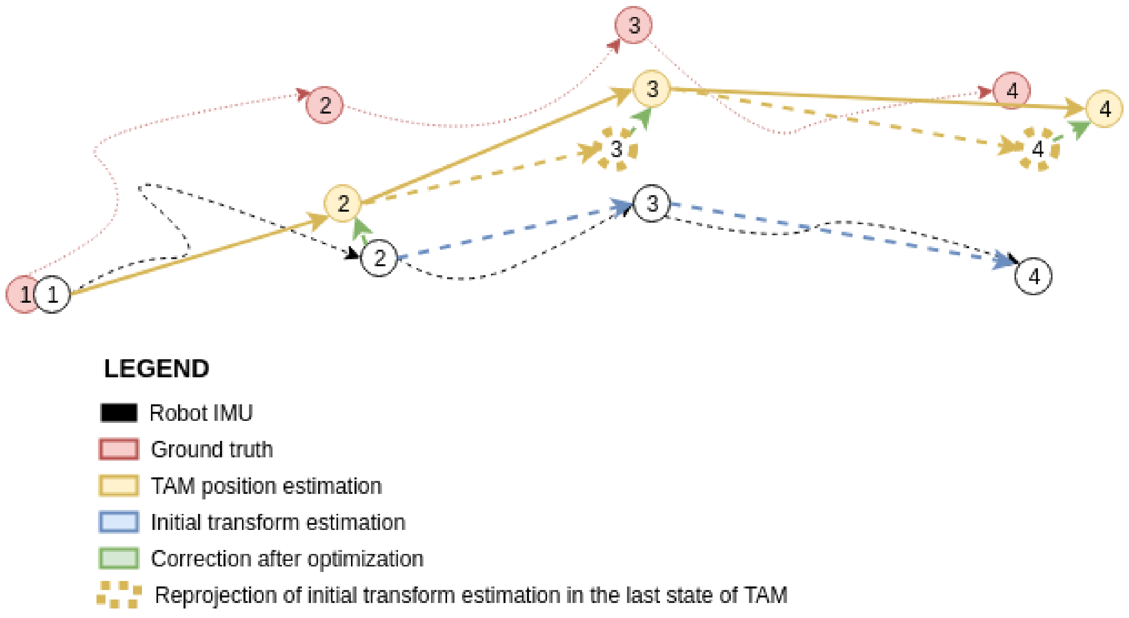

1
1
2
2
2
3
3
3
3
4
4
4
4
LEGEND
Robot IMU
Ground truth
TAM position estimation
Initial transform estimation
Correction after optimization
Reprojection of initial transform estimation in the last state of TAM

Figure 8-4: The image shows in red the ground truth, which is the real movement of the robot, in black is shown the motion estimated using IMU data, In Yellow the motion computed using the LiDAR information is represented, the blue arrow shows the estimated movement between the last position and the current position of the drone, and finally in green the correction computed with an integration of LiDAR and IMU informations.

The initial estimate of the positioning of the robot is received in input from the IMU data processing. The difference between the IMU position at the end of the sweep and the last position estimated by TAM results in an initial estimation of the movement of the drone during the sweep. Figure 8-4 illustrates clearly the several transformations involved in the process. The data that the *transformation estimation* algorithm requests to the *tf* framework are the black circles in Figure 8-4. The motion estimation is then computed subtracting the transformation of TAM at the end of the previous sweep from the current position of computed with the IMU and published in *tf*.

8-3-2 Optimization preparation

An important assumption used during the estimation of the correction of the pose is that the motion is modeled with constant angular and linear speed during a sweep. This allows to linear interpolate the pose transform within a sweep for the points that are received at different times. Let t be the current time, and t_{start} is the starting time of the sweep. T_k is the transform between $[t_{start}, t]$, and it is a 6 DoF transformation. Given a point p_i, with timestamp t_i and let $T_{k,i}$ be the pose of the robot at time t_i, and can be computed with interpolation:

$$T_{k,i} = \frac{t_i - t_{start}}{t_i - t} T_k \tag{8-5}$$

Using this transform is derived:

$$p_i = R_i \widetilde{p_i} + T_{k,i}(1:3) \tag{8-6}$$

were R_i and $T_{k,i}(1:3)$ are rotation and translation of $T_{k,i}$, and $\widetilde{p_i}$ is the corresponding point in the previous sweep. R_i is a rotation matrix defined by the Rodrigues formula [26].

The transformation is applied to the coefficient computed in the previous steps. This lead to the creation of the optimization problem which aims to minimize the error.

$$f(p_i, T_{k,i}) = d \tag{8-7}$$

each row of f corresponds to the feature point, and d contains the corresponding error. Before to insert the function in the optimization problem the Jacobian with respect to $T_{k,i}$ is computed

$$J = \frac{\partial f}{\partial T_{k,i}} \tag{8-8}$$

The function can be solved through nonlinear iteration by minimizing d to zero.

8-3-3 Optimization problem and solver

The problem explained in the previous subsections is then a nonlinear problem which needs to be minimized. The solution gives in output a transformation. The transformation represents a correction of the position to minimize the error the correction to the current pose provided by the *tf* framework. There are different libraries that propose possible solvers for nonlinear optimization problems. For this project Ceres is used (developed by Google). Ceres can solve bounds constrained robustified non-linear least squares problems [2].

The first step to prepare the solver is to write a functor that will evaluate the function. The important thing to note here is the template method, which constructs the problem receiving inputs and computing outputs. The residual function is then created:

$$residual = (x_i \cdot x_e)^2 + (y_i \cdot y_e)^2 + (z_i \cdot z_e)^2 + (\theta_i \cdot \theta_e)^2 + (\phi_i \cdot \phi_e)^2 + (\psi_i \cdot \psi_e)^2 - d \tag{8-9}$$

where x_i, y_i, z_i, are the translation value and θ_i, ϕ_i, ψ_i are respectively roll, pitch and yaw of $T_{k,i}$, in the preparation step; x_e, y_e, z_e are the translation value and θ_e, ϕ_e, ψ_e are respectively roll, pitch and yaw of the position estimated by the IMU and d is the distance error.

Once we have a way of computing the residual function, it is now time to construct a non-linear least squares problem using it and use Ceres to solve. To do so, The AutoDiffCostFunction which takes a CostFunctor as input is used. It automatically differentiates the ConstFuctor and gives it a CostFunction interface. After the differentiation the cost function is inserted in the problem with the AddResidualBlock method of the ceres::Problem object. This operation is repeated for all the valid points, and finally the problem is ready to be solved.

8-3-4 Accumulate transformation

The solution to the problem is a correction of the estimation computed in real-time with the information provided by the odometry sensor of the robot. This correction is then applied to the initial transformation which is broadcasted to the other nodes.

8-3-5 Points re-transformation

Once a new estimation of the final position of the robot is computed all the points need to be updated according to the new estimated movement. Also, in this case, the movement is considered at a constant velocity during the sweep, and consecutively the transformation is distributed with interpolation to all the points. Once the point cloud is updated it is ready to be published to the Mapping node which will align the updated points to the previous map generated in the previous steps.

8-3-6 Sweep registration

The sweep process in the odometry node is concluded with the storage of the point cloud into the Ordered Cloud structure. As illustrated in Chapter 6 the points must be stored subdivided into scans. The temporary Ordered Cloud object is already correctly structured, indeed it only requires to update the ScanTree object that collects the entire sweep, with the new relocation of the points and then it is pushed in the vector containing the latest sweep. In case the vector is full the older sweep is removed from the data structure.

8-4 Output Interface

The entire process is executed only once per sweep. Once the computation is over, the new point cloud is retransformed to the origin of the *odom* frame and together with the current pose estimated of the drone they are published using on the ROS network. The frequency of the output is 1Hz.

The output of TAM differs from LOAM for two main reasons:

- the points are divided into different point clouds for corner and surface, as well as in the registration;
- the frequency of the output in LOAM is 10Hz. The difference is that LOAM publishes multiple times the odometry of the robot during a sweep. This means that the point cloud evaluated during the updates are not spread equally along the 3D scene but they represent only a limited space, because of the rotation of the laser. In the current version of TAM it is possible to increase the frequency of the output, however, requires an evaluation of the performances of the odometry estimation with a limited field of view. Another possible solution that can be introduced in TAM to increase the output frequency will be discussed in the future works section at the end of the report.

8-5 Overview

In this chapter, the process of the robot's odometry estimation using data fusion between the IMU and the LRF have been explained. The procedure is organized in sequential steps well defined and structured. The process organizes the data in a data structure to facilitate the execution of the scan matcher. The scan matching procedure is translated into an optimization problem to reduce the error between the points in the current sweep and the corresponding points obtained in the previous scans. The solution of the problem leads to an improvement of the robot position estimated and consecutively a correction of the point cloud. The results are then published into a ROS topic accessible by all the processes on the ROS network.

System design In this part the entire system design has been illustrated from the sensor-actuator system until the optimization procedure to improve the motion estimation of the robot.

The system has three main components that provide input data. The input is processed by the Registration process that transforms the ranges into a point cloud and assigns a *tag* to each point. The point cloud is ultimately used to compute the positioning error of the IMU and minimizing it, providing in output a corrected point cloud and pose of the robot.

Part III

Evaluation

In this section of the report, the results obtained from the dataset collected by the drone during the research are articulated. In Chapter 9, the clarification of important aspects related to hardware, software and data-set scenario used for the data registration are explained. In Chapter 10, an evaluation of the Registration process is illustrated, analyzing the effects of the feature selection technique adopted. In Chapter 11, the examinations of the motion tracking system compared with ground truth system are presented. Following this line of reasoning, the part is divided into chapters, with the objectives and approaches as illustrated in Figure 8-5.

Objective:

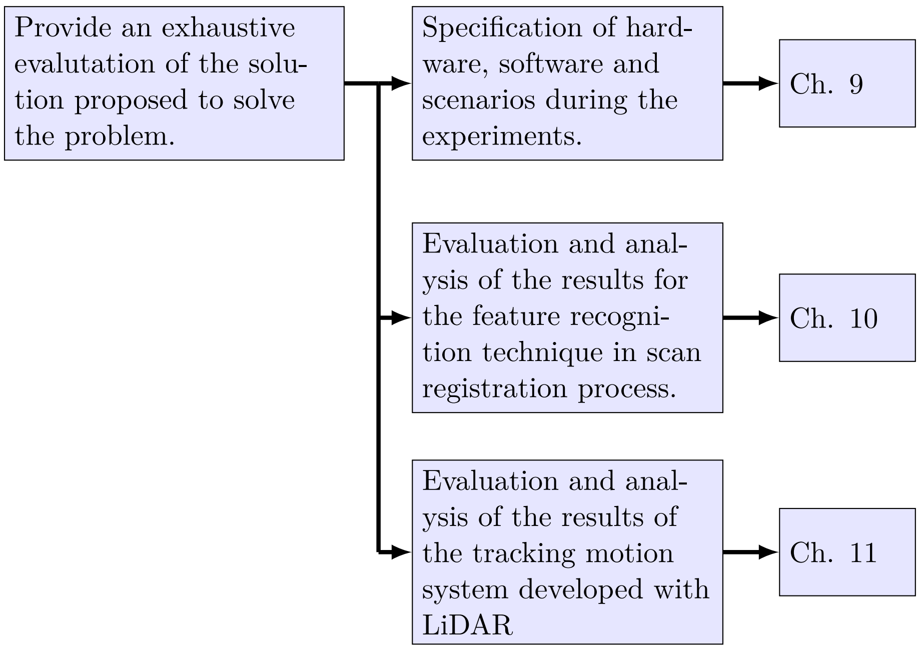

Provide an exhaustive evalutation of the solution proposed to solve the problem.
Specification of hardware, software and scenarios during the experiments.
Ch. 9
Evaluation and analysis of the results for the feature recognition technique in scan registration process.
Ch. 10
Evaluation and analysis of the results of the tracking motion system developed with LiDAR
Ch. 11

Figure 8-5: Structure of Evaluation part.

Chapter 9

Conditions for the evaluation

The performances of a robotic system is strongly influenced by the scenario where the tests are executed. In this chapter, the specifications of the data-set is explained, it will be used in the next chapter to evaluate the achievements. The chapter starts with the specification of the equipment and the hardware used for testing and evaluation, it continues with the software specification and its concluded by a explanation of the experiments executed with a description of the setup and the goals set.

9-1 Equipment specification

The hardware installed on a robot can make the differences between bad and good results. The possibility to use really good hardware, reliable and robust, improve the quality of the system. The MAV used for this project is the first prototype from Mainblades and indeed it is important to mention that the current hardware design is not yet optimal to perform aerial inspections. As already mentioned several times during the report, the 3D SLAM problem is tackled using a 2D LRF rotating continuously at a constant speed. During the execution of the data-set presented in this evaluation the speed of the laser has never been changed, I always used angular speed equal to 3.14 rad/s (180°/s). This lead to a field of view of 360° every second. Because the laser frequency is equal to 40Hz, two consecutive scans are captured with a distance of 4.5°. In fact single scan has a field of view of 270° with 0.25° of angular resolution. The continuous rotation of the motor also influenced the single scans that are not perfectly straight but are curved. This solution even with some drawback is still the one that fits better in this scenario, for two main reasons:

- **Weight:** the entire system with Laser, Dynamixel motor, and slipring is lighter than the current 3D laser on the market, as reported in Table 9-1.

	Velodyne VLP-16 3D LiDAR	**MBI hardware solution**		
		Dynamixel	Slipring	Hokuyo UTM-30LX
Weight	830 g	72g	97g	370g
		539g		

Table 9-1: Weight comparison of the MBI hardware design and a 3D LRF.

- **Field of view:** The possibility to capture 3D scene losing only a cone on the top part is the wider field of view solution available at the moment. The 3D lasers on the market are developed mostly for ground robots (cars) and for this reason, they have a limited vision on the z-axis (for instance Velodyne 3D laser have only 40° field of view on the z-axis [37]).

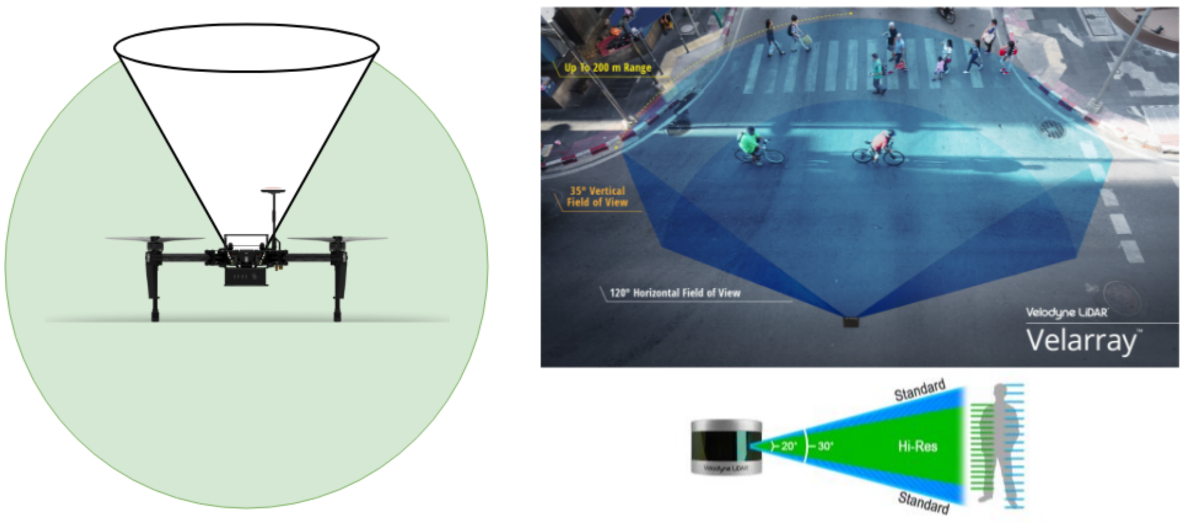


Figure 9-1: Field of view comparison between a 3D laser and the MBI hardware design. On the left is shown the current MBI design, where the green area is the field of view and it is a sphere where only the cone in white is the hidden area. As visible in the image the 3D sensor is not able to cover the same area, and it results much limited on the z-axis.

The field of view influences the odometry result but even more the mapping result. In fact, with the current design, the objects that are poitionated above the robot are not visible, for example, ceilings. The odometry procedure is less affected by this limitation. The field of view with the current design is also limited by the legs of the robot. In order to remove this errors, a limit distance threshold has been added equals to 0.5 meters.

Equipment The robot used for the evaluation of the performances is the DJI Matrice 100. It is equipped with sonar sensors and stereo cameras in the bottom part to help the stabilization of the robot in the air. Two batteries are placed on the robot, one used to power up only the drone and all the DJI system and the second is used for the board, laser, motor, and wireless acess point.

Communication security The access point allows the communication between the drone and external devices on the private network. The LAN is secured by a secret code, and also only specific MAC address can connect to the LAN for two main security reasons: first, to not allow a man in the middle that can corrupt the communication between the two devices and second to protect the access to the code.

9-2 Software specification

The software used has been explained in the previous chapters. It is a refactored version of the original LOAM software. The software has been rewritten completely in the process of registration and odometry and it has been customized to fit the software requirements.

The drone itself provides a Software Development Kit (SDK) which contains all the sensor information provided by the autopilot of the drone. The main limitation of the SDK is that it is not easy to debug the internal behavior and some "fusion" settings are not clear how they are computed. The SDK used for the data-set in this evaluation is the version 3.3 however recently we upgraded to 3.6.

During the flight test the data provided by the DJI SDK, LRF and motor are saved in memory using a ROS functionality called *rosbag*. In real time also the odometry information captured with the IMU is published into the *tf* framework. *tf* is a ros topic and it is included in the set of data saved in the rosbag. The evaluations are executed on my personal laptop HP ZBook Studio G4 [20] using the data-set collected during the test flights. The evaluation of the performances on-board are the only results reported that are executed on the UP boards that are installed on the drone.

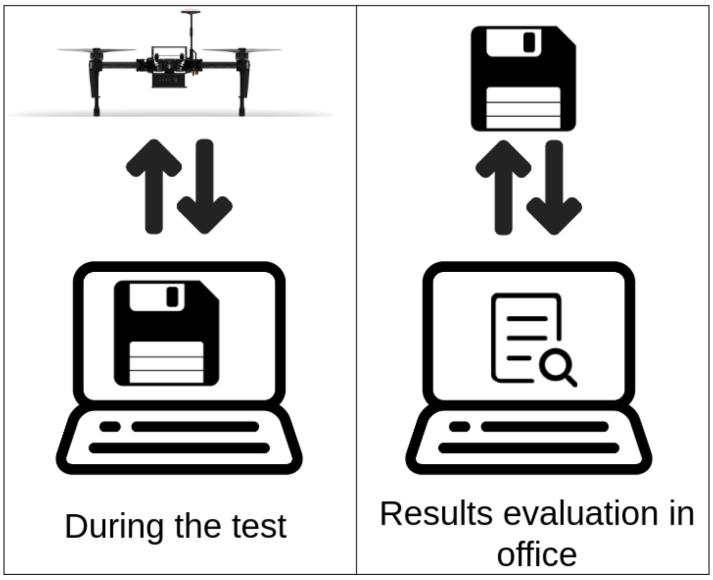


Figure 9-2: During the test the data from the drone are saved locally with the rosbag service of ROS. All the evaluation procedure are executed offline using my laptop.

9-3 Experimental test setup and goals

For the evaluation of the results, different setups are necessary for Registration and Odometry.

9-3-1 Registration

The data-set used for the evaluation of the performances during the registration procedure is a 40 seconds execution of the system with the drone in a static position, placed in an indoor environment.

The goals of this evaluation are:

- Synchronization of the motor-encoder and LRF. To evaluate this synchronization, it is important to check where the points of several consecutive sweeps are placed. If the points represent the same environment, for example, they all clearly define walls and ceiling at the same distance, it follows that the synchronization is successfully designed.
- Feature points recognition. Consecutive sweeps generated in the same state with the same conditions must place the feature points in the same area. In fact, even if the robot is statically positioned, there is a small offset between the first scan of the sweep at time t and the first scan of the sweep at time $t + 1$, and consecutively all the others. What is important to detect is that even if the offset is present the corner and surface extreme points are placed on the same line or plane. This lead to conclude that while the robot is moving the extreme points will be accumulated always in certain areas of the space. The results are valid if the extreme corner points are displaced along a specific line and the surface points are displace on flat surfaces.
- on-board performances. Evaluation of the CPU usage on the embedded system installed on the MAV during the execution of the test. The data-set contains a flight test of three minutes. The processor is monitored during the entire test and the average and max usage of the CPU are reported in the evaluation.

9-3-2 Odometry

For the evaluation of the odometry system, the robot has to fly in an indoor environment, however, during the development of the project, the possibility to collect data-sets of flying motion of the drone has always been a limitation.

The tests performed for the evaluation of TAM have been executed at the Cyber ZOO Laboratory of the Technical University of Delft. The laboratory is equipped with an optical motion capture system, named OptiTrack.

The OptiTrack has multiple synchronized cameras installed around the target capture volume, and 2D images are captured from each camera. 2D positions are calculated, and the overlapping position data are compared to compute the 3D positions via triangulation. The 3D location of markers can be resolved with sub-millimeter accuracy with an optimal capture volume size and camera configuration [19]. Due to the high accuracy, the data of the

OptiTrack are used as ground truth during the evaluation of TAM. The data-set used contains sprints, rotation around yaw and other stress behaviors that are not the common flight pattern that will be used for the inspection of an aircraft.

The goals of this evaluation are:

- Qualitative analysis. Evaluation of the movement of the drone in the 3D space.
- Quantitative analysis. Computation of the error following mathematics benchmark to examine the different methods.

In addition, an evaluation of the on-board performances and software requirements are illustrated. The examination evaluates the CPU usage of the odometry process in percentage with the data-set, and examines the software architecture proposed.

Chapter 10

Scan registration analysis

The registration procedure collects the ranges provided by the LRF and transforms them into a 3D point cloud. Furthermore, it assigns a specific feature to each of the points in the point cloud. In this chapter, an evaluation of the synchronization motor-encoder and LRF is given. The evaluation of the registration process continues with an analysis of the parameters and feature selection in a static map. The last part of the chapter focuses on the real-time on-board performances of the system compared with the original LOAM version.

10-1 Sensor processing and integration

One of the pillars of the entire system is the synchronization between the LRF and the motor-encoder. The rotation requires an accurate matching procedure of every single scan into the correct position. Figure 10-1 shows the mapping of an indoor environment.

Consecutive scan distance As it is visible in the picture the scans are displaced at a regular distance. The distance between two consecutive scans of the same sweep is due to the speed of the motor, and it is equal to 4.5°.

Not visible areas In the figures are also visible two diagonal lines that have no points, this area are situated behind the leg of the drone and indeed the laser is not able to see them. Furthermore the drone is placed close to the wall indeed all the points behind it are not visible. Moreover, the cone on top of the drone does not have points because of the 270° field of view.

Results The synchronization between motor and LRF is prosperous, because, the wall behind the drone is straight and also the ceiling and the wall on the opposite side of the drone.

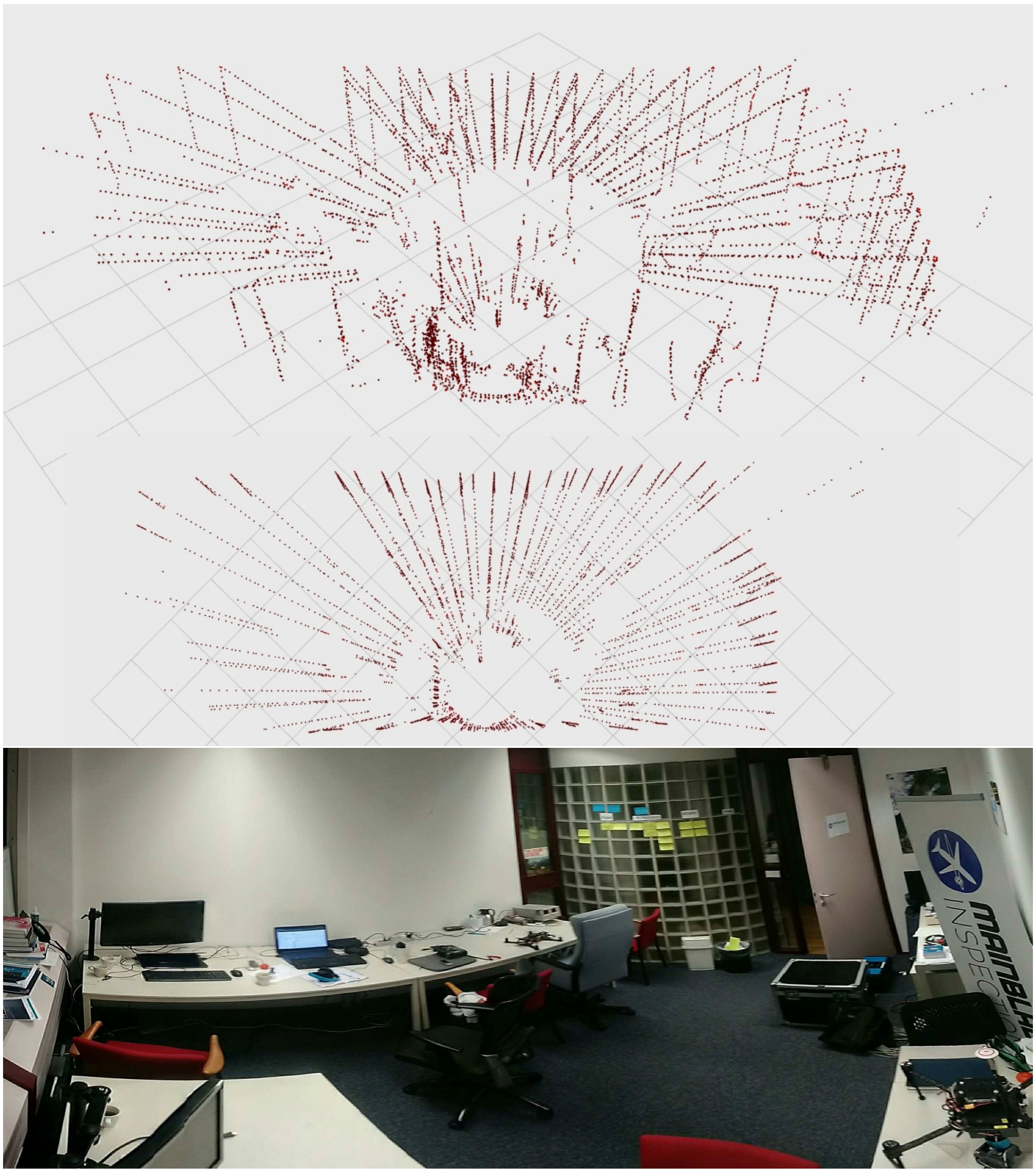


Figure 10-1: On top, the point cloud generated by three consecutive sweeps. On the bottom, an image of the real scenario and the displacement of the drone in the indoor environment.

10-2 Feature selection

The feature selection algorithm is responsible for the division of the point cloud between corner and surface points. The algorithm has many parameters that can be tuned and their values directly affect the output of this procedure. The most relevant parameters where I focused my attention are:

- Number of sub-scans. As mentioned in the System Design part, every scan is sub-divided into n sub-scans. The points in each of this sub-scans are ordered by their smoothness value and based on this ranking each of the points is appointed to a determined feature. Higher is the number of sub-scans, larger is the number of extreme points in each scan and more noise points are registered. According to this statement, the number of sub-scans for the evaluation is bounded between 1 and 4.
- Number of extreme points. This parameter refers to the number of extreme points per sub-scan. The goal in the evaluation of this parameter is to highlight the displacement of the extreme points. It is important to evaluate a trade-off between correctly allocated points and noise points. The number of extreme points influences more the corner point selection procedure instead of the surface selection, because the corner area is strictly limited in an edge, rather the surface area is wider and easier to place extreme points rightly.
- Smoothness window size. The smoothness windows define the number of points evaluated during the feature selection algorithm. This value linearly increases the complexity of the algorithm. The goal of the evaluation of this parameter is the identification of a small value that can identify with sufficient accuracy the edge features in an environment and its ability to displace the surface points of consecutive scans always in the same area of the plane.

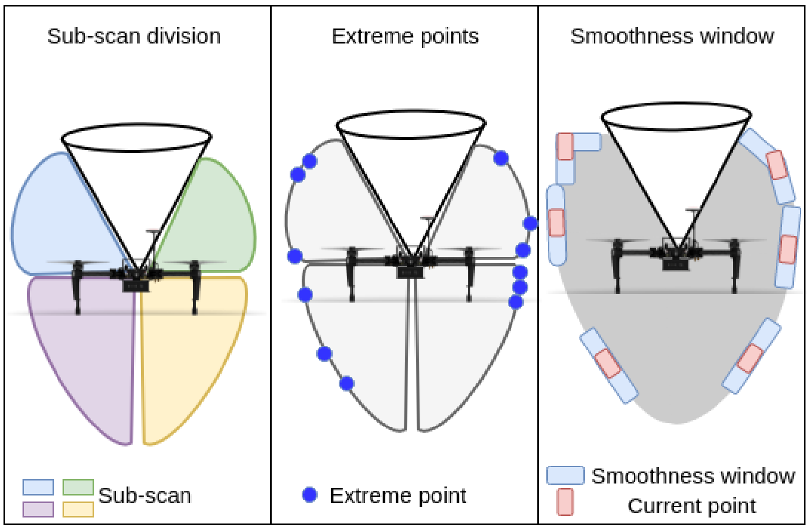


Figure 10-2: Visualization of the meaning of the three parameters evaluated in section 10-2.The first picture on the left show the division of the entire point cloud into 4 sub-scans. The figure in the middle represent the displacement of corner point in the environment, the exaple shows the numbver of extreme points equals to three. The last image on the right shows the smoothness window surrounding the single point for the computation of the smoothness factor.

10-2-1 Point cloud division and extreme points detection

The sub-scan creation and the extreme points detection analysis are evaluated separately between corner and surface points.

Corner

The corner points have been defined previously as the points with higher smoothness factor (refer to Chapter 7 Equation 7-1).

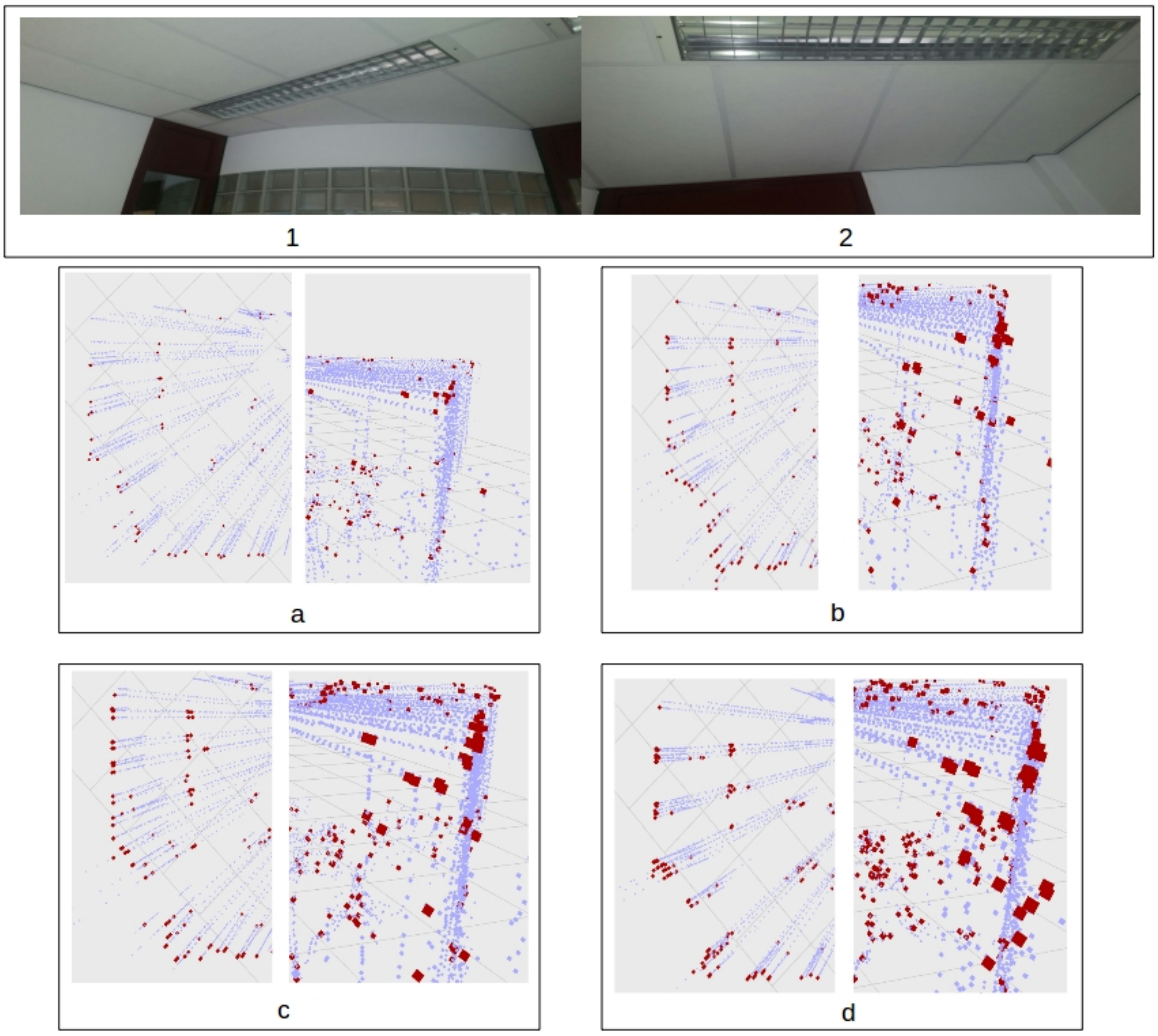


Figure 10-3: The picture shows on top the corner between wall and ceiling that is represented in the point cloud. The points have been registered keeping the drone statically positioned on a table. The four images show how the detection of extreme points can vary when the parameters are tuned differently. The parameter used per picture are reported in Table 10-1

Figure 10-3 shows several experiments executed with different amount of sub-cloud and number of extreme corner points. All the images reported show five consecutive sweeps. The

Parameters	**a**	**b**	**c**	**d**
number sub-clouds	1	2	4	4
number extreme corners	2	2	2	6

Table 10-1: Parameter setting for evaluation of the extreme corner points. The letter refers to the experiments reported in Figure 10-3

data-set is the same shown in Figure 10-1, however, I focus the attention on a sub-part of the map that is the corner between wall and ceiling in the top-right of Figure 10-1. The extreme points are dark red and increased in size for highlighting purposes, the light purple points are all the other points registered during the five sweeps.

Feature recognition The several experiments showed that in all the scenario the extreme points are correctly mapped at the extreme of the map. It is also noticeable that the light-grid placed several centimeters from the edge on the left of the image, is also considered as a corner feature.

	a	b	c	d
Number of extreme corner in the examined area per sweep	7 (46%)	10 (51%)	17 (59%)	36 (43%)

Table 10-2: Number of extreme points visible in Figure 10-3 per sweep. The letter refers to the related image in Figure 10-3. The percentage value express the correctness of the point identified compared to the amount of extreme points in the sub-area.

The number of extreme corners increases according to the number of sub-clouds sets, as reported in Table 10-2. In particular, in Figure 10-3 (a) the number of points is really limited compared to Figure 10-3 (c). It is correct that in all the cases the points are accumulated around the corner line. The number of sub-clouds has been bounded to four because the author decided to limit the number of areas per scan in order to restrict the number of total extreme points. However, it is evident that the edge is better identified when the number of sub-clouds is higher. This result is consequences of the division of the area in restricted spaces where is easier to displace the extreme points and identify environment feature, such as edges.

Noise points The illustrations on the right part of Figure 10-3 (a), (b), (c) and (d) show several edge points displaced in the environment not close to *corner lines*. These noise points are due to several not-geometric objects present in the room during the acquisition of the data-set, such as chairs, jackets, people. The noise points can create a high danger during the scan matching procedure that follows the scan registration algorithm.

However not all the noise points are errors, but many of them may detect other 3D objects in the environment that present edges, such as desks, or chairs. Indeed, an additional filtering element may be necessary but also further examination are required to defined precisely error points patterns. Additional reflection on the point analysis technique during the registration are suggested in the final chapter of this report.

Before have been stated that the edge between the wall and the ceiling is better recognized when the number of sub-clouds increase, however, a high number of sub-clouds leads to an increment of noise points.

Number of extreme corner points Figure 10-3 (c) and (d) show two different experiments where the number of extreme corner points is set differently. It is evident that a higher level of this parameter does not influence the displacement of the points in the scene. However, the density of extreme points accumulated in the same area raises.

This concludes that the neighbors of the extreme corner points displayed in Figure 10-3 (c) have a smoothness factor similar to the corner point itself. This is due to the overlap of the smoothness window area.

It is important to consider that also the quantity of noise points increase for the same reason, indeed a limited number of extreme points is preferred compared to a larger one.

Overview The evaluation of the organization of the extreme corner points in the scene shows that a larger number of sub-cloud helps to identify better the features in the environment, but at the same time, they increase what the author called the noise points. It is important to find a trade-off between the two. The most suitable solution is to keep the number of sub-clouds larger and limit the number of extreme points per sub-clouds. This decision lead to a distribution of extreme points within all the scan and at the same time limit the size of the noise points.

Surface

The definition of surface point (refer to Chapter 7) says that the points with lower smoothness value are classified as extreme surface, and all the points that are not inserted in any group (extreme corner, corner, and extreme surface) are inserted into the surface set.

In this section, an evaluation to determine the displacement of the extreme surface points in the scene for different value of sub-scans is presented.

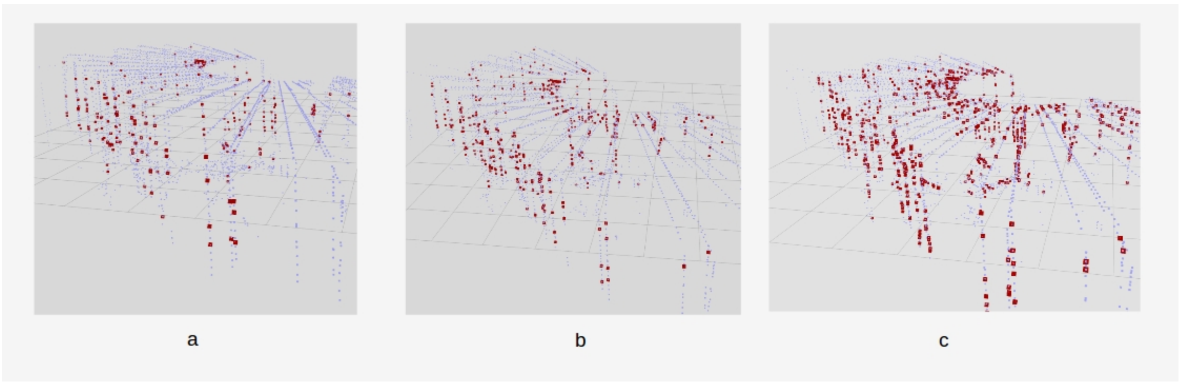


Figure 10-4: The parameter used per picture are reported in Table 10-3

Parameters	**a**	**b**	**c**
number sub-clouds	1	2	4

Table 10-3: Parameter setting for evaluation of the extreme surface points. The letter refers to the experiments reported in Figure 10-4.

Feature recognition Figure 10-4 shows three different experiments where the number of sub-clouds is varying, the values are reported in Table 10-3.

As already mentioned in the previous section, the number of sub-clouds is limited to four to guarantee the displacement around a limited number of features in the space.

The three experiment executed show how the extreme surface points are displaced in the scene.

	a	b	c
Number of extreme surface in the examined area per sweep	26	58	113
Number of extreme surface on the ceiling per sweep	7 (27%)	20 (35%)	61 (54%)
Number of extreme surface on the wall per sweep	19 (73%)	38 (65%)	52 (46%)

Table 10-4: Number of extreme points visible in Figure 10-4 per sweep. The letter refers to the related image in Figure 10-4. The percentage represent the sitribution of the poin t between the two surfaces. A larger number of sub-clouds lead to a different distribution of the points.

The number of extreme points increases linearly with the number of sub-clouds. The number of extreme points per sweep is reported in Table 10-4. In Table 10-4 are also reported the percentage of the displacement of the surface points between the two flat areas (ceiling and vertical wall). It is visible how a higher number of sub-cloud helps in a better distribution of the points along all the scene, without concentrating them in a single spot. In fact, the experiment in Figure 10-4 (c) shows an equal distribution of points per sweep between the two flat surfaces.

Overview The evaluation of the organization of extreme surface points demonstrates that a larger number of sub-clouds lead to a higher coverage of the space. The number has been bounded to four in order to follow the rules introduced in the evaluation of corner points.

10-2-2 Smoothness window

The smoothness value attributed to each point is strongly related to the amount of points contained in the window used to estimate this value. In Chapter 7 have been introduced the term *smoothness window*, which denotes the number of points considered during the computation of the smoothness factor in Equation 7-1.

The goal of this evaluation is to determine how the windows size influence the displacement of extreme corner and surface points in the edge and along the walls.

The value considered for this examination are reported in Table 10-5. the smoothness window includes the point itself plus the neighbours, indeed the sizes evaluated are all odd numbers, in order to have the same amount of points on the two side of the evaluated point which is placed in the middle, as illustrated in Figure 10-2. Furthermore, the factor is bounded between 3 and 7 for two reasons: 1 is discarded because it would be just the point itself; the size of a window larger than 7 would include in the computation points relatively far from the point examined. Moreover, a short window size reduces the computation necessary for the calculation of the smoothness factor for each point.

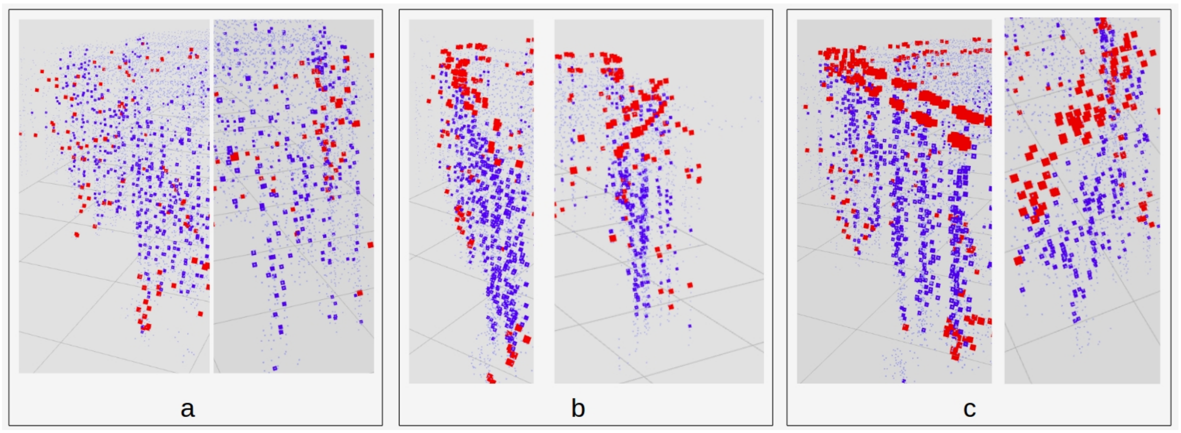


Figure 10-5: Evaluation of the smoothness coefficient. The three boxes show two walls in the data-set and the displacement of the extreme corner and extreme surface points in several consecutive scans. In red are shown the extreme corner points and in blue the extreme surface points, in light purple and with smaller size are drawn all the other points. The parameter used per picture are reported in Table 10-3

Parameters	a	b	c
Size smoothness window	3	5	7

Table 10-5: Parameter setting for evaluation of the smoothness coefficient computation. The letter refers to the experiments reported in Figure 10-5.

Figure 10-5 shows the graphical results for three values of the *smoothness window* size reported in Table 10-5. All the three boxes in the figure are composed of two images; the two images represent a similar scene (a wall and an edge) but in different spots of the area.

Results The amount of extreme points per sweep in the three experiments reported in Figure 10-5 is the same, what changes is just their distribution and location in the space. With a very small window size, only a very limited number of extreme points are displaced in the space considered for the examination. Furthermore, the corner points (red) in Figure 10-5 (a) are not well placed in the area. Instead, a larger size leads to better concentration of the points on the edge and on the two walls. In Figure (c), the left image shows two lines that represent the edge between the wall and the ceiling and the edge in the ceiling frame.

The possibility to identify edge lines and surfaces head to consider achieved the goal of the feature selection algorithm with a smoothness size higher or equal to five. This evaluation uses several assumptions made by the author that require additional qualitative verification. Based on the results achieved on this evaluation, for future experiments (reported in the next chapter) a windows size value equals to seven will be used.

10-3 On-board performances

An embedded system is an electronic device with limited computational power and limited resources. The evaluation of the performance of an algorithm running on an embedded system is very important to judge the quality of the method. The performances are expressed by the usage of the CPU during a test.

The embedded systems utilized for this evaluation are two: UP and UPsquare. The main difference between the two boards is the CPU, both of them are Quad-core but the burst frequency for the UP is 1.92GHz instead it reach up to 2.5GHz in UPsquare. Additional information regarding the two embedded systems are detailed in Appendix C.

The goal of this evaluation is to compare the CPU consumption utilizing the same data-set (described in Chapter 9) between two different methods: TAM and the original version of LOAM. To execute the method in real-time it is necessary that the CPU does never reach the 100% of usage because in that case, it happens there is the possibility to skip some tasks in the real-time execution.

Comparison The results of the four executions are shown in Table 10-6. The outcome exhibits a reduction of CPU usage with the LOAM method. It reduces the usage of almost 50% on both the boards. The cause of this result can be attributed to the initial transformation from raw laser data to point cloud. In fact, this procedure is diverse between the two methods. In TAM is used the *tf framework* which requires a query to the data stored in the data-set and consecutively may require more time and computational resources.

	Avarage CPU %		Max CPU %	
	UP board	UPsquare board	UP board	UPsquare board
TAM	69.5	36.6	76.6	45.2
LOAM	50.8	28.3	59.6	34.8

	Memory usage in MB (% of total memory)
TAM	4095 (0.5%)
LOAM	3276 (0.4%)

Table 10-6: Performances of TAM with respect to LOAM on the two boards during the registration method.

In Table 10-6 are also reported the memory usage values for the two methods. TAM and LOAM have a static usage of memory during the registration procedure. The result is equal on both the UP boards tested because both of them have 4GB of RAM. LOAM uses less memory for the registration of the points and feature recognition compared to TAM. This result can also influence the CPU usage, in fact, operations of read and write in memory increase the usage of the processor.

Result TAM shows a downgrade in terms of on-board performances compared to LOAM. However, both the methods achieve the goal required by the problem which is to keep the consumption of the CPU bounded below its limit. For better performances of the execution, it is suggested to use the UPsquare board, this limit the risk to reach high CPU usage and complete the method correctly in real-time.

TAM also requires more memory during the execution with respect to LOAM, however both the methods use a limited amount of memory compared to the total RAM available on both the boards.

10-4 Conclusion

In this chapter, a broad evaluation of the Registration process has been presented. The evaluation has been divided into three sections:

- The first section examines the synchronization between laser and motor encoder, the visualization of several consecutive sweeps obtained during a static indoor test demonstrate the correct behavior of the laser system.
- The second section examines the feature selection algorithm. The number of sub-cloud, the number of extreme points and the size of the smoothness window have been tuned to obtain the best setting based on graphical results. The evaluation produces in output the most coherent configuration based on the results obtained.
- In the third, and final, section the performances of the registration procedure have been shown. The tests demonstrate that the original version of LOAM have better results, but also TAM doesn't use entirely the CPU and consecutively can be executed on-board in real-time.

In conclusion, a proper configuration of the parameter of the registration process has been discovered and furthermore, the ability of the method to be executed in real-time have been confirmed.

Chapter 11

Odometry

The odometry estimation of the robot is fundamental for mapping and navigation purposes. In this report, a tracking system to solve the localization problem have been described. This system is applied to a real scenario to ultimately estimate the motion of the robot in 3D with 6DoF.

The chapter starts with an explanation of the method and formulas used for the evaluation, it continues with an analysis of the results obtained, and concludes with an evaluation of the on-board performances and software requirements achieved.

Importance of the odometry results An accurate estimation of the pose of the robot in real-time helps to create correct maps. The importance of this step in the SLAM problem is really significant. An evaluation of the error during the odometry estimation can give a feedback on how the map will look like at the end of the entire process. This chapter focuses on the motion estimation's error, comparing the result between TAM and LOAM with the OptiTrack system (ground truth for my data sets).

11-1 Evaluation procedures

The evaluation process starts with the registration of a rosbag[1] that execute TAM and LOAM and record at the same time the four frames considered in this evaluation: OptiTrack estimation (OT), the robot odometry computed with internal sensors (RO), the tracking fusion of RO with LiDAR (TAM) and the original LOAM that receives RO as input (LOAM).

Motion analysis method The data analysis, in the next section, starts with a representation of the motion of the four frames along the three axes. All the frames are initialized at the beginning of the procedure and they share the same initial origin, which the author decided to make it corresponds to the OT origin. The operations to compute this transformation are:

[1]rosbag is a set of tools for recording from and playing back to ROS topics. It is intended to be high performance and avoids de-serialization and re-serialization of the messages.

- computation of the difference between the two frame

$$diff = GT^{-1} * P \tag{11-1}$$

where GT is the ground truth, P is the pose of the frame, and $diff$ is a third transformation that represents the difference between the two; $GT \in SE(3)$ [2], $P \in SE(3)$, $diff \in SE(3)$.

- Apply the computed transformation to the initial pose

$$P_{corrected} = diff * P \tag{11-2}$$

where $P_{corrected}$ is the pose of the frame overlapping the initial frame of the ground truth, $P_{corrected} \in SE(3)$.

Error analysis method The benchmark employed for the evaluation of the error in SLAM system is described in [34]. It is mainly composed by two main error evaluations:

- **Relative pose error** The relative pose error (RPE) measures the local accuracy of the trajectory over a fixed time interval dt. Therefore, the relative pose error corresponds to the drift of the trajectory which is in particular useful for the evaluation of odometry systems [34]. The relative pose error is defined as:

$$E_i := (GT_i^{-1} GT_{i+dt})^{-1} (P_i^{-1} P_{i+dt}) \tag{11-3}$$

where GT is the ground truth system and P is the pose of the tracking system evaluated, and $E \in SE(3)$, $GT \in SE(3)$, $P \in SE(3)$.

From a set of consecutive poses and relative error it is possible to compute the root mean square error (RMSE) over all the time indices of the translation components:

$$RMSE(E_{i:n}) := (\frac{1}{n} \sum_{i=1}^{n} trans(E_i)^2)^{\frac{1}{2}} \tag{11-4}$$

where n is the number of poses evaluated and $trans(E_i)$ is the translation vector of the error pose. Together with the root mean square error also the mean square error will be evaluated which gives less influence to outliers.

- **Absolute pose error** For SLAM systems, additionally the global consistency of the estimated trajectory is an important quantity. The global consistency can be evaluated by comparing the absolute distances between the estimated and the ground truth trajectory [34]. As stated previously all the trajectories are specified in the same global frame and indeed do not need to be realigned, but the only difference is the origin point. Indeed, the absolute error is computed with a formula similar to RPE but instead of the difference between two consecutive poses, in the problem is inserted the current pose and the origin:

[2]SE(3) is the Special Euclidian space in 3D, it contains information regarding translation and rotation of the pose.

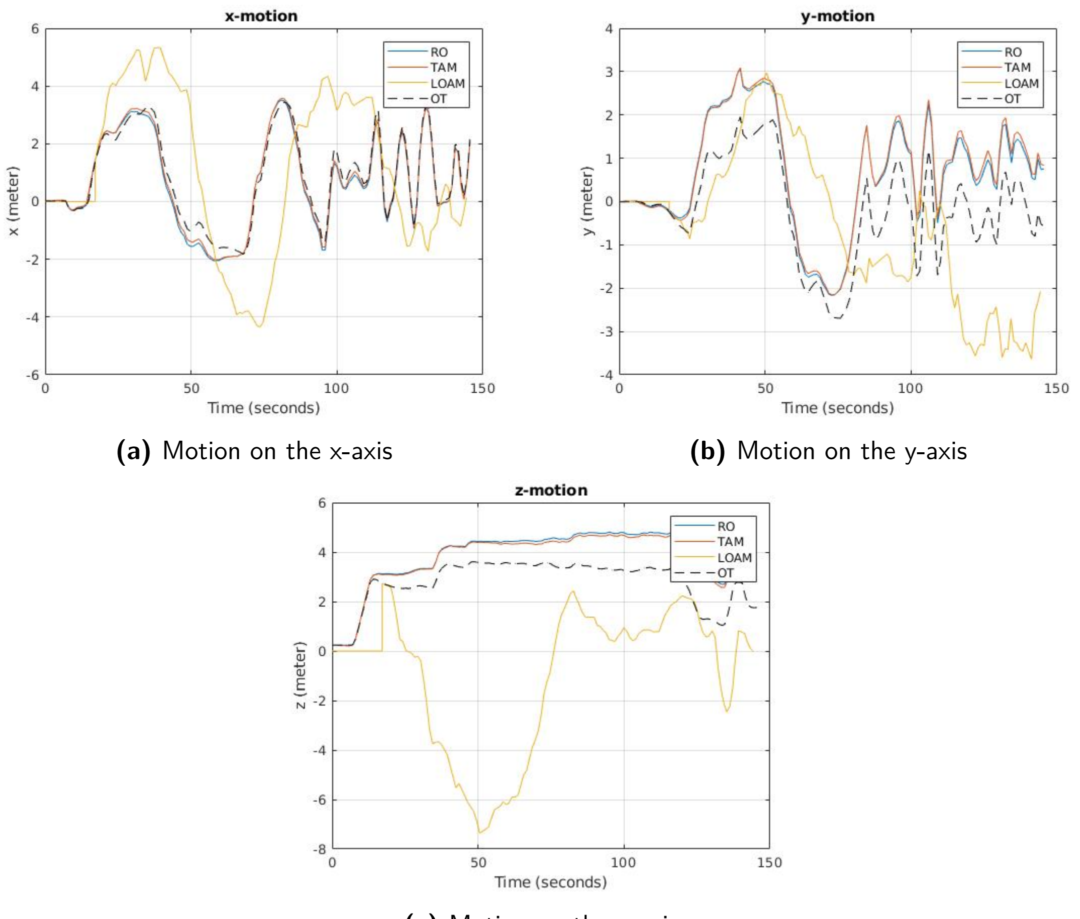
$$E_t := (OT_0^{-1} OT_t)^{-1}(P_0^{-1} P_t) \tag{11-5}$$

where E_t is the absolute pose error at time t, OT_0 and P_0 are respectively the origin of the body frame in recorded with the OptiTrack or the current motion system evaluated, and ultimately OT_t and P_t are the poses of the ground truth and of the motion system at time t.

11-2 Measurement and results

To evaluate the efficiency of the algorithm, described in the previous chapters, a measurement and results analysis of TAM is reported in this section.

11-2-1 Motion analysis



(a) Motion on the x-axis **(b)** Motion on the y-axis

(c) Motion on the z-axis

Figure 11-1: 3D motion of TAM and LOAM compared to the OptiTrack data in the dataset.

The motion results of the experiment executed are reported in Figure 11-1. LOAM has some delay in the initialization of the system, it is visible in all the three graphs in the first 20 seconds of the execution.

Altitude The results show a low accuracy in the estimation of the altitude by TAM, even if it improves the estimation computed by RO. LOAM after the initial delay is initialized at the same altitude of the OT frames, however, instead of following the ground truth it behaves differently and the frame moves down and only at the end of the execution the method seems able to correct the error and accompanying better the correct behaviour, having an error approximately equals to zero at 125 seconds.

x and y motion TAM estimates with high accuracy the movement of the robot along the x-axis. Instead along the y-axis it accumulates a large error in the first 20 seconds and maintains the offset error along all the execution. LOAM instead behave differently and accumulates error along all the test.

Conclusion The motion analysis shows the results of TAM compared to LOAM using a ground truth system in a qualitative way.

TAM outperformed when compared to LOAM, however, is still accumulates errors of more than one meter along z-axis and y-axis.

TAM improves the value received by the robot's IMU, reducing up to 10% the error on the z-axis.

11-2-2 Error analysis

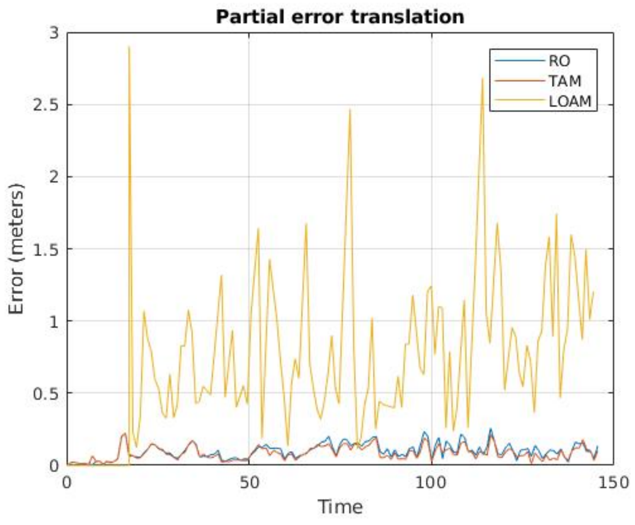


Figure 11-2: RPE data-set without executing LOAM.

Figure 11-3: Relative pose error per frame. The relative motion of the ground truth is compared with the relative motion of the three methods and the results are reported in the two graphs.

The graph, shown in Figure 11-3, represents the relative pose error (RPE) of the three frames (TAM, RO, and LOAM) compared to the OT between two consecutive frames.

The results confirm the good performances of TAM compared with LOAM. The RPE of TAM has its maximum peak equals to 23.8cm. In several steps of the execution, TAM is able to reduce the error computed during the initial estimation by RO.

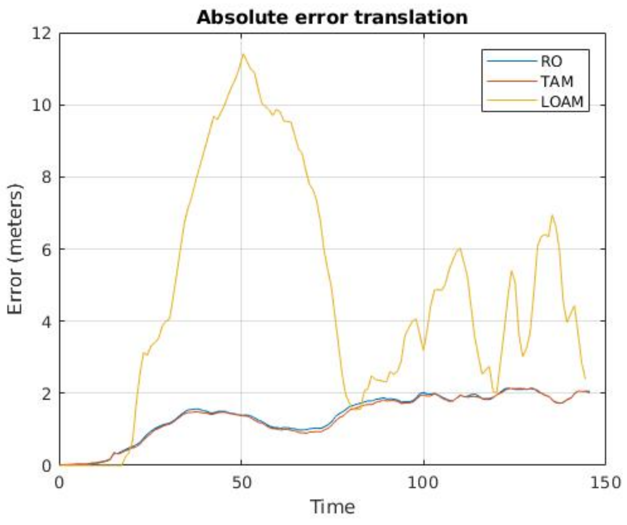


Figure 11-4: APE data-set without executing LOAM.

Figure 11-5: Absolute pose error per frame. The pose of the three frames is compared to the absolute motion of the robot captured with the ground truth.

The Figure 11-5 shows the absolute error of the three frames compared to the ground truth. TAM is able to improve the results of RO for the large part of the execution with an average of 8% in the reduction of the error between 50 to 100 seconds.

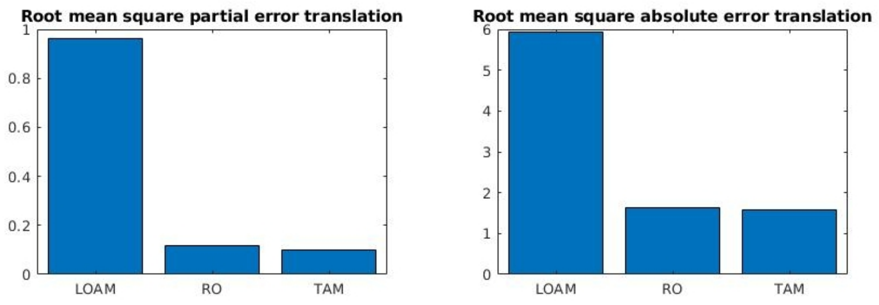


Figure 11-6: Root mean square error and mean square error of RPE and APE of the data-set without executing LOAM.

Figure 11-6 shows the RMSE of the RPE and APE results showed in the previous graphs, the exact results of the calculation are reported in Table 11-1. The chart shows the large error of LOAM compared to the other two methods in all the errors evaluated, with a 960%

	LOAM	**RO**	**TAM**
RMSE RPE	0.96047	0.11786	0.10062
RMSE APE	5.9453	1.6254	1.598

Table 11-1: Value of the root mean square error for relative and absolute position errors for the three methods evaluated with respect to the ground truth.

increment of the average relative error compared to TAM. TAM improves the results of RO of more than 14% in the RPE evaluation and by 1.6% the absolute error.

Overview The error analysis confirms in a quantitative way the results already observed above. In fact, TAM can improve the performances of the IMU and this benefit is driven by the LiDAR odometry algorithm.

The performances of the original LOAM are confirmed to be negative with the proposed data-set.

11-3 Onboard performances

As already mentioned in the previous chapter, to complete the evaluation of a software design for an embedded system it is important to evaluate the on-board performances. The boards used for testing are the same already introduce in Chapter 10.

Comparison The on-board performances obtained by TAM are compared with the original version of LOAM.

The average and maximum usage of a single core in the CPU is reported in Table 11-2. The results show that TAM requires less calculation to compute the localization of the robot with respect to LOAM. One of the main reasons for this result is certainly the iteration of the optimization procedure in the original LOAM which is substituted by a more straight and direct estimation in TAM.

The low CPU usage allows the TAM algorithms to be executed in real-time. The results show also an impossibility of real-time processing by LOAM with the embedded system available.

	Avarage CPU %		Max CPU %	
	UP board	UPsquare board	UP board	UPsquare board
TAM	35.0	23.2	43.8	30.4
LOAM	92.5	98.8	103.3	102.3

	Memory usage in MB (% of total memory)
TAM	1638 (0.4%)
LOAM	2457 (0.6%)

Table 11-2: On-board erformances of TAM with respect to LOAM. The table on top shows the CPU usage, and the table of the bottom illustrate the memory usage expressed in megabytes.

The quantity of memory occupied by the two software solutions are static and are reported in Table 11-2. LOAM uses more memory compared to TAM; indeed, the data structure with redundancy presented in Chapter 6 is a better solution compared to the point cloud organization used in LOAM.

Results TAM has a lower CPU usage when compared to LOAM. This result lead to real-time computation of the odometry during the execution. Furthermore, the data structure used reduces the amount of memory needed. Both the methods have static memory usage value and both of them occupy a low amount of the total memory of the boards.

11-4 Software requirement achievement

In Chapter 3 several requirements related to the software architecture have been declared.

During the testing TAM demonstrates to be **maintanable**, because the software is easy to be adjusted due to its well structure and because it is well commented and documented. TAM is also **reliable** with the data-set examined because it never fails. Furthermore, it can be **extended** adding additional operational processes, such as mapping. The results computed by TAM are published as ROS topics that can be obtained in input from any ROS node, this renders the system **scalable**. Ultimately, the algorithm has been **exported** on different embedded systems (the two boards) to prove the portability of the method.

In conclusion, all the requirements have been matched.

11-5 Conclusion

The chapter illustrated the results obtained with TAM to solve the localization problem of a drone in a 3D GPS denied environment. The results show that the motion tracking system proposed does improve the results obtained with the internal sensors of the robot. However also another well-known system, LOAM, have been compared with TAM. TAM operates better than the original version of LOAM in terms of onboard performances and also have a lower relative and absolute error compared to the ground truth. Finally, the software architecture of TAM respects all the software requirement stated by the problem.

The accuracy of the tracking system has large margins of improvement for precise inspection of large objects. The system can use the data provided by the sensors and by TAM to execute in parallel additional operations to improve the current results, such as estimation of the distance robot-floor, the position with respect to 3D object, loop-closure etc.

Chapter 12

Closing remarks

The final chapter concludes to what degree the set objectives are achieved. By clarifying the knowledge gained from the design process and the evaluation of the designed system, the recommendations provide a critical view of the obtained results and define the focus for future work.

12-1 Conclusion

The main objective of the thesis, i.e., to *design a 3D LiDAR motion tracking system installed on a MAV, for localization of the robot in GPS-denied environments, such as a hangar*, has emerged from an early stage of identifying the problems regarding localization indoor with MAVs.

In order to achieve the set objective, the design process is divided into three parts with separate sub-objectives, of which the conclusions are formulated first.

Partt I: Conceptual design The conceptual design aims to *define the main limitations, for indoor localization with a MAV.* The positioning system needs to fulfill certain demands. An analysis of the related shows how the problem can be approached and which difficulties were already faced by other researchers. The open source LOAM method results to fit well in the scenario described, however, it does not meet several software requirements, such as extensibility and maintainability, and indeed need to be refactored, introducing the usage of external libraries and simplifying the structure of the code, for a better usage of the inspection system.

Part II: System design The objective of the system design is to *propose a suitable solution for localization of robots with 6DoF in GPS-denied environments using LiDAR input data.* The found combination of principles of the scan matching techniques, when subjected to criteria for implementation, leads to the application-specific system design.

The operation of location estimation is composed of three main phases:

- Synchronization of the LRF, IMU, and motor-encoder. The capacity to couple the ranges generated by the LRF with the position of the laser obtained by the motor encoder and IMU.
- Scan registration and feature detection. A *tag* is assigned to each point, which defines the character of the specific points. The ability to correctly group the points in different collections, edge, and surface, supports the execution of scan matching.
- Odometry estimation. The method is concluded with the estimation of the position of the robot. Using correspondences between points collected in different sweeps, it is possible to identify an error in the displacement of the robot in the 3D space. The reduction of this error leads to an improvement in the motion tracking system.

Part III: Evaluation To *provide an evaluation of the solution which solves the problem*, the method has been tested. Results obtained from the test settings, which are performed in an indoor environment without GPS signal, demonstrate the functionalities of the system. The synchronization of the laser position and feature detection shows good results with static maps. Furthermore, the scan matcher, to improve the position estimated with IMU sensor, is able to detect correctly the correspondences between points and track the movement of the robot in the space. Due to the limited possibility to execute the tests only a limited amount of data is available. The performance of TAM on-board has better results in comparison with the original version of LOAM. The odometry process requires one-fifth of computational power compared to the original one.

Main conclusion With the first two parts, the main objective of this thesis is partially accomplished. In the third part, the designed system is implemented to assess the practical issues and validate the concept. The main conclusion is that the combination of hardware components and software is able to create local maps and recognize features in a scan, furthermore, the scan matcher improves the result obtained by the IMU, but it is not able to remove entirely the error to solve accurately the positioning problem of the robot.

The proposed design in this thesis serves as the blueprint and the platform developed is the basic module for future research and engineering improvements. Further development will lead to a unique motion tracking method for MBI's inspection system.

12-2 Recommendations

For a complete design and better performances, further development has to continue building upon the established platform: (1) improvement of the scan matcher for the correspondences detection of points and initialization of the sensors, (2) improvement of the hardware design, (3) incorporating the mapping procedure with the addition of loop closure detection and feature recognition of 3D elements.

The suggested main research directions focus on 3D feature detection and loop closure to improve the current results of localization using LiDAR. The recommendations are divided into several groups: improvement of measurement and estimation techniques, further robot development and additional software functionalities.

Improvement of measurement and estimation techniques

1. LRF initialization. The setup procedure should be enforced by an initialization of the components position. Between two consecutive inspections, it may happen that the position configuration of the LRF may be changed. Currently, the setup check is manually made, this procedure is not precise. The method demands an automatic procedure to initialize the LRF position with respect to the drone when the execution starts.

2. Tests for evaluation. So far the volume of tests completed is not enough for an exhaustive evaluation of the performances. For example, the rotation speed of the laser has been constant in all the tests performed, however, decreasing the speed may lead to an improvement of the correspondences detection between points and ultimately better results of the method.

3. Noise points. Some of the points obtained by the laser are placed next to the *cutoff area* due to the short or high range. These points need a different treatment because they may be false-extreme and consecutively they become noise points for the method. Additional filtering and verification on each point are required during the acquisition procedure.

4. Initial transformation of the points. Currently, all the points are transformed into the *odom* frame. This transformation carries the error related to the estimation obtained from the robot IMU. The points should not include this error during their initialization.

5. No-motion signal. Whenever the drone is not moving but it is placed in a static position for several seconds the system should recognize this status and not perform any operation of localization and mapping, because it is unnecessary and may only add additional error.

6. Sweep-window. Currently, the motion estimation is executed every second, at the end of every sweep. However, the frequency can be increased by inserting a sweep-window that evaluates the scans collected in the last second and compares them with the scans generated previously. This design would produce the motion between two consecutive scans and would increase the output frequency up to the laser scan frequency (40Hz).

Further robot development

1. Sensors redundancy. Employing redundancy by adding multiple IMUs can increase system reliability, performance, and robustness. Further research is required for exactly determining the performance-complexity trade-off. However, easy/cheap integration as infrastructure and methodology are already available/implemented, suggests only a marginal increase in complexity.

2. Hardware design. The attachment of the LRF to the drone is a temporary solution (shown in Figure 5-4). The frame where the LRF is attached should be statically connected to the motor with a better-designed hardware.

Additional software functionalities

1. Mapping and navigation. The positioning system should be expanded with navigation and mapping modules. This will lead to the ability to couple the position with the surface mapping features of the robot and future inspection system.
2. Feature detection. Due to the 3D field of view of the robot, it is a limitation to generate features only using a single scan. It is possible to determine features in the 3D scene, and it is even possible to determine the position of the laser with respect to the object scanned [38]. The usage of this additional information may lead to an important improvement in the motion estimation of the robot.
3. Localization in existing maps. The aircraft inspection procedure must reuse maps created in advance where the robot needs to localize itself and add a correction in the map in case the environment dynamically changed.

Appendix A

Related and future works

A-1 Basic principles of probabilistic SLAM

The SLAM problem have been addressed for many years and it has been translated in a probabilistic form, where the problem is a simple probability distribution

$$P(x_k, m|Z_{0:k}, U_{0:k}, x_0) \tag{A-1}$$

which requires to be computed for all times k. Where x_k is the state vector describing the location of the robot, m is the vector describing the location of the ith landmark whose true location is assumed time invariant, $Z_{0:k}$ is the set of landmark observation, $U_{0:k}$ is the history of the control inputs and x_0 is teh initial position of the robot.

This probability distribution describes the joint posterior density of the landmark locations and vehicle state (at time k) given the recorded observations and control inputs up to and including time k together with the initial state of the vehicle. In general, a recursive solution to the SLAM problem is desirable.

Starting with an estimation for the distribution $P(x_{k-1}, m|Z_{0:k-1}, U_{0:k-1})$ at time $k-1$, the joint posterior, following a control u_k and observation z_k , is computed using Bayes theorem [1]. This computation requires that a state transition model and an observation model are defined describing the effect of the control input and observation respectively. The observation model describes the probability of making an observation z_k when the vehicle location and landmark locations are known and is generally described in the form

$$P(z_k|x_k, m) \tag{A-2}$$

It is reasonable to assume that once the vehicle location and map are defined, observations are conditionally independent given the map and the current vehicle state. The motion model

[1]Bayes' theorem offers a natural way to unfold experimental distributions in order to get the best estimates of the true ones. Additional information of this theorem can be found in [35]

for the vehicle can be described in terms of a probability distribution on state transitions in the form

$$P(x_k|x_{k-1}, u_k) \tag{A-3}$$

That is, the state transition is assumed to be a Markov process in which the next state x_k depends only on the immediately preceding state x_{k-1} and the applied control u_k and is independent of both the observations and the map.

The SLAM algorithm is now implemented in a standard two-step recursive (sequential) prediction (time-update) correction (measurement-update) form:

Time update

$$P(x_k, m|Z_{0:k}, U_{0:k}, x_0) = \int P(x_k|x_{k-1}, u_k) \times P(x_{k-1}, m|Z_{0:k-1}, U_{0:k-1}, x_0)dx_{k-1} \tag{A-4}$$

Measurement update

$$P(x_k, m|Z_{0:k}, U_{0:k}, x_0) = \frac{P(z_k|x_k, m)P(x_k, m|Z_{0:k-1}, U_{0:k}, x_0)}{P(z_k|Z_{0:k-1}, U_{0:k})} \tag{A-5}$$

Equations A-4 and A-5 provide a recursive procedure for calculating the joint posterior $P(x_k, m|Z_{0:k}, U_{0:k}, x_0)$ for the robot state x_k and map m at a time k based on all observations $Z_{0:k}$ and all control inputs $U_{0:k}$ up to and including time k. The recursion is a function of a vehicle model $P(x_k|x_{k-1}, u_k)$ and an observation model $P(z_k|x_k, m)$.

A-2 Related approaches

Atlas SLAM Bosse and ZLot divided the process into two steps, as illustrated in Figure A-1. During the process, first, local maps from scans produced by one or more asynchronous lasers are constructed. The map building algorithm is based on a Kalman filter that maintains a collection of robot poses and estimates vehicle motion through an iterative laser scan matching technique [14]. Bosse and Zlot developed a system independent from IMU measurements, though these additional measurements may be incorporated into the filter if available, to improve the precision of the results. During the second step the local maps are matched to recognize regions that have been previously explored. Using this data association technique, Bosse and Zlot were able to close large loops, thereby solving one of the fundamental challenges of the SLAM problem [14].

LOAM Figure A-2 shows the software system overview of LOAM where the LiDAR odometry process and the LiDAR mapping are separated. $\hat{P}$ is the point received by the laser scanner; it is registered in the local map of the sensor {L}. During each sweep multiple different points are collected in P_k. The LiDAR odometry takes the point cloud and computes the motion of the LiDAR between two consecutive sweeps [39]. The outputs are further processed by LiDAR mapping, which generates a map from the undistorted point cloud at a lower

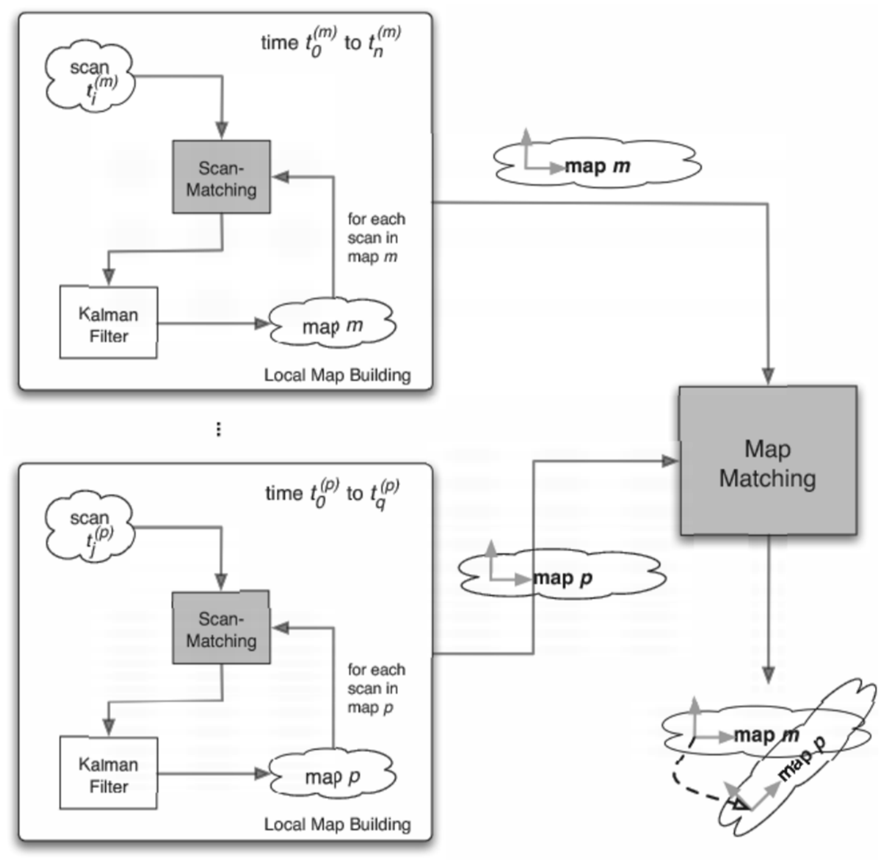


Figure A-1: Data flow in the mapping process. Map building (lefthand side) iteratively aligns laser scans using the scan matching algorithm to form a local map. Completed maps are then matched with existing maps (right-hand side) in order to determine their relative alignment if they overlap [14].

frequency. The process is completed by the transform integration step, which integrates the output from the two different algorithms to generate a transform output, defining the position of the robot with respect to the map.

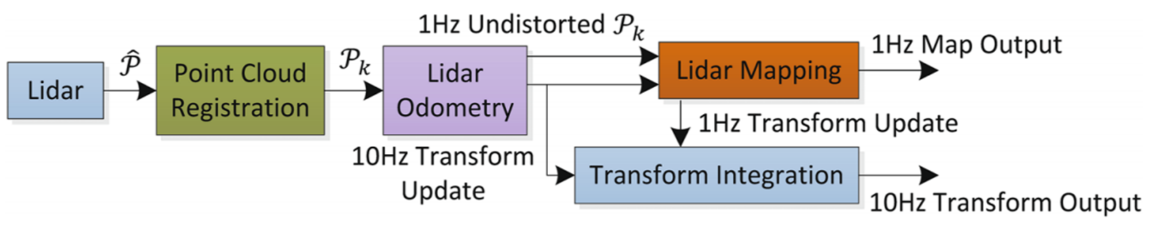


Figure A-2: Block diagram of the LiDAR odometry and mapping software system [39].

In the next paragraphs are compared these two works focusing on scan-matching techniques for robot motion estimation. The chapter is concluded with an evaluation of the works, analyzing constraints and requirements of the problem. In Appendix A are presented additional information of these two research.

A-2-1 Mapping

Mapping algorithm can be categorized by the type of the map. Two main map topologies are metrical and topological. Metrical maps represent exact distances of the environment, such as volumetric environment maps in 3D. Furthermore, SLAM approaches can be classified by the number of degrees of freedom of the robot pose, as mentioned above a quadrotor can move within six degrees of freedom.

Several techniques exist for three-dimensional mapping. In this project, I will focus on a 6D pose estimation using IMU (Inertia Measurement Unit) combined with a rotating 2D LiDAR sensor installed on the MAV. Table A-1 summarizes possible mapping techniques.

Sensor data	**Axes rotationDimensionality of pose representation**	
	3D	6D
2D	**Planar 2D mapping** 2D mapping of planar sonar and laser scans [Thrun 2002]	**Slice-wise 6D SLAM** 3D mapping using a precise localization, considering the x, y, z-position and the roll yaw and pitch angle.
3D	**Planar 3D mapping** 3D mapping using a planar localization method and, e.g. an upward looking laser scanner or 3D scanner.	**Full 6D SLAM** 3D mapping using 3D laser scanner or (stereo) cameras with pose estimates calculated from the sensor data.

Table A-1: Overview of the dimensionality of SLAM approaches

A-2-2 Atlas SLAM

Once the alignment between consecutive laser scans is estimated, it is necessary to determine the correspondences on a large scale - between local maps. This feature facilitates the detection of loop closure and instances of repeated traversal including approaches to the same region from different directions [14]. During this matching process, it is crucial to determine if two different local maps overlap, and also how they overlap. To quickly align two maps, each local map requires a compact representation of its salient characteristics which distinguish a local map from the others. The map matching representation, proposed by Bosse and Zlot [14], consists of an orientation histogram of the scan normals plus a set of weighted projection histograms created from the orthogonal projections of scan points onto lines from discrete orientations. Strong correlations between histograms can be used to deduce potential map matches: first, the orientation histograms can be used to resolve angular offset, followed by the projection histograms associated with that direction to resolve the translational component [14].

A-2-3 LOAM

The mapping algorithm is executed only once per sweep. It generates an undistorted point cloud, $\bar{P}_{k+1}$, and a pose transform, $\mathbf{T}^{L}_{k+1}$, containing the motion information of the laser during the sweep. The point cloud generated is registered into the world map {W}.

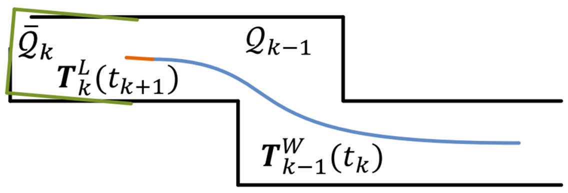


Figure A-3: Mapping process [39].

Figure A-3 illustrates the mapping process. The blue colored curve represents the LiDAR pose on the map, $\mathbf{T}_k^W$, generated by the mapping algorithm at the sweep k. The orange color curve indicates the LiDAR motion during sweep $k+1$, $\mathbf{T}_{k+1}^L$, computed by the odometry algorithm. With $\mathbf{T}_k^W$ and $\mathbf{T}_{k+1}^L$, the undistorted point cloud published by the odometry algorithm is projected onto the local map, denoted as $\bar{Q}_{k+1}$ (the green colored line segments). During the mapping process, the results obtained from the local point cloud are matched with the existing global map, it is represented by Q_k (the black colored line segments) in Figure A-3.

Bosse and Zlot's method [40] requires batch processing to allow loop closure and develop accurate maps; therefore are unsuitable for application where maps are needed in real-time; LOAM can produce maps that are qualitatively similar [39].

A-2-4 Additional analysis

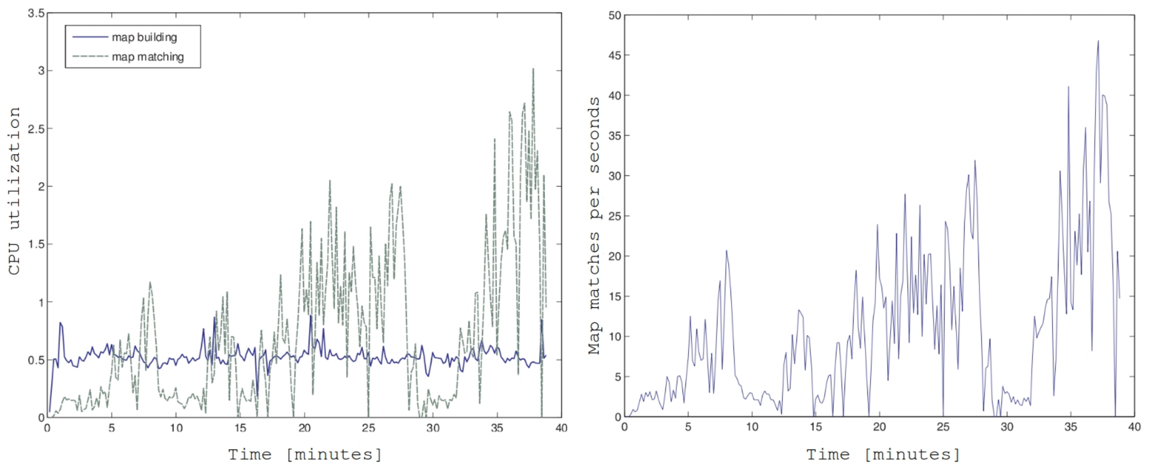


Figure A-4: Performance of the mapping framework for the Kenmore data set. On the left he number of seconds to process 1 second of incoming data for the map building process and the map matching process. On the right the average number of map matches per second. Note how the map matching CPU time is correlated with the number of map matches per second [14].

Application 1 One of the first implementations of their solution was an iterative scan matching technique which processes the data of a 2D LRF with EKF to maintain all the poses in a local map [14]. The experiments computed without odometry and GPS sensors show an error in laser odometry that grows unboundedly over the distance traveled [14]. The experiment is executed outdoor, indeed no environment constraints can be applied. However, the real-time performance analysis is shown in Figure A-4, where the contributions from the map building and map matching processes are given. That map building requires essentially a constant amount of computation time per measurement, whereas the map matching CPU utilization is highly dependent on the number of map matches considered (Figure A-4). The experiment is executed with a car and static 2D LRF, indeed time constraints and functional evaluation are not related to the project for this specific initial version of Atlas-SLAM.

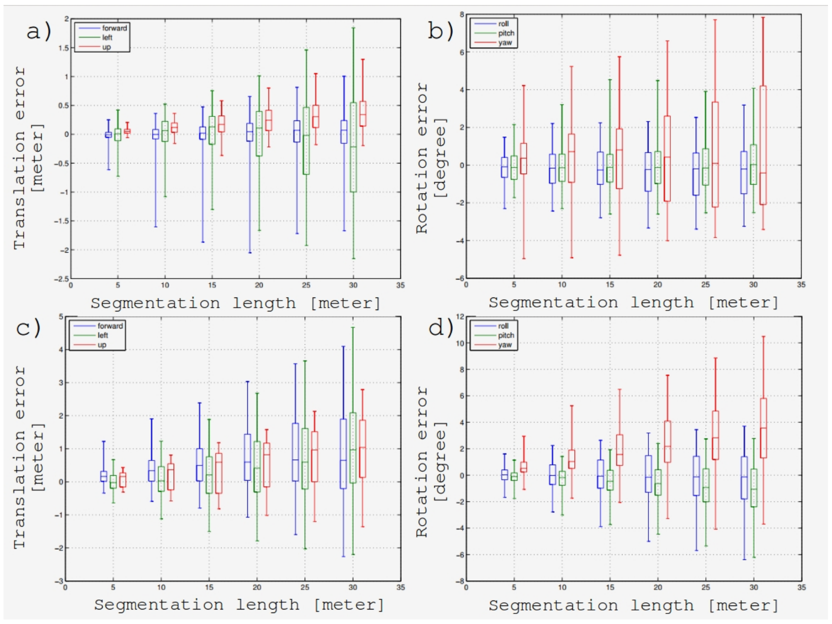


Figure A-5: Box-plots of trajectory errors for the industrial (a),(b) and off-road (c) environments,(d) when comparing trajectory segments of various lengths. The whiskers of the box-plots extend to the 3 rd and 97th percentiles, whereas the box covers the 25th to the 75th percentile, and the mid-line is at the median. For the industrial environment experiment, ground truth for z-displacements and roll and pitch angles can be assumed to be near zero, as the vehicle operates on approximately flat ground. The box-plots are taken at 5 m intervals, and are spread out into groups of three close to the window size for visualization purposes [15].

Application 2 Another experiment from Bosse and Zlot is the implementation of a spinning 2D LRF on a car [15]. The results of the estimated trajectory of the vehicle are depicted in Figure A-6.

The results of the experiment demonstrate that the growth of translational errors is generally expected to be less than 5% of distance traveled, and rotational errors less than 0.3 degrees per meter. The most difficult motions to accurately recover are fast spot-turns when portions of the sweeps might not be sufficiently constrained. The authors explain that for example, for the part of the sweep where the laser is vertical and only senses the ground, yaw rotations are momentarily nonobservable, because no IMU is used. This limitation is related to the field of view of the laser and the possibility to miss information in part of the scan. The vehicle also had to pass through off-road environments. The growth of trajectory errors was a bit larger in the off-road experiment as compared to the industrial environment, see Figure 9[**?**].

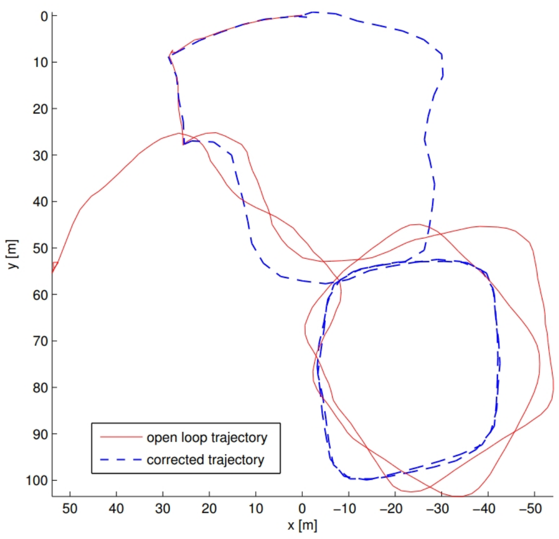


Figure A-6: A comparison between the trajectory estimated by the 3D sweepmatching algorithm and the trajectory after "closing-the-loop" in 6DoF for the off-road environment [15].

In Figure A-6d we can see that there is a rotational bias of about 0.1 degrees per meter; however, the translational errors depicted in Figure A-6c are generally less than 1% of distance traveled, and the algorithm is still able to construct a locally accurate map and trajectory. The maximum linear and angular velocities allowed in this specific approach explained in [15] are limited by the amount of overlap between consecutive sweeps and on the richness of the environment. In the experiments presented here, the vehicle traveled up to speeds of 6 meters per second and turn rates of 20 degrees per second.

A-3 Bentwing place recognition

Kaul with the support of Bosse and Zlot, recently developed an airborne 3D mapping system implemented on a drone named bentwing [22]. The system can recreate an accurate dense map of the environment using a rotating lightweight 2D laser scanner. The measurements obtained from the sensor are input in the SLAM solution, which is an extended version of Atlas. A key point in the solution proposed in [22] is the place recognition solution for loop closure detection.

The place recognition problem addresses the estimation of identifying regions in the environment re-observed at different times. It can be used to recognize loop closure, global relocation, and data-sets merging. The place recognition solution provides a coarse alignment that must be further refined by the non-rigid registration algorithm. Figure A-7 shows an example of the map evolution with the application of the non-registration and place recognition technique.

The place recognition algorithm, used in bentwing, divide the point cloud into a discrete set of *places*. Each place is represented by a set of descriptor vectors, each of which encodes the geometry of the neighborhood. The point descriptors are based on a 3D generalization of features, which encode the height distribution of points falling in radial bins surrounding the keypoints [22].

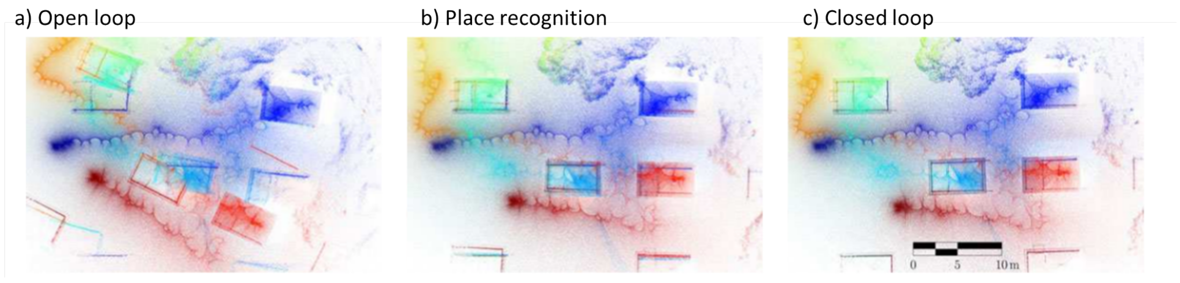


Figure A-7: Example of the non-rigid registration and place recognition framework applied to an individual dataset scan. (a) A portion of the point cloud generated during the initial open-loop trajectory generation phase. (b) The same area in the point cloud after the place recognition phase. (c) The point cloud after global non-rigid registration using the coarsely aligned place recognition solution as input. [22]

Place recognition uses this new data structure organized in places to compute similarities between descriptors using Euclidian distances. Upon the search is completed for all places, vote scores over a threshold are retained and verified using consistency check based on RANSAC. The matches remained are then organized into a pose graph. Ultimately the pose graph is optimized updating the trajectory with the identified places matched.

A-4 Mainblades Inspections

In this chapter I wanted to give more information about the company with a look into the future in Aircraft inspections.

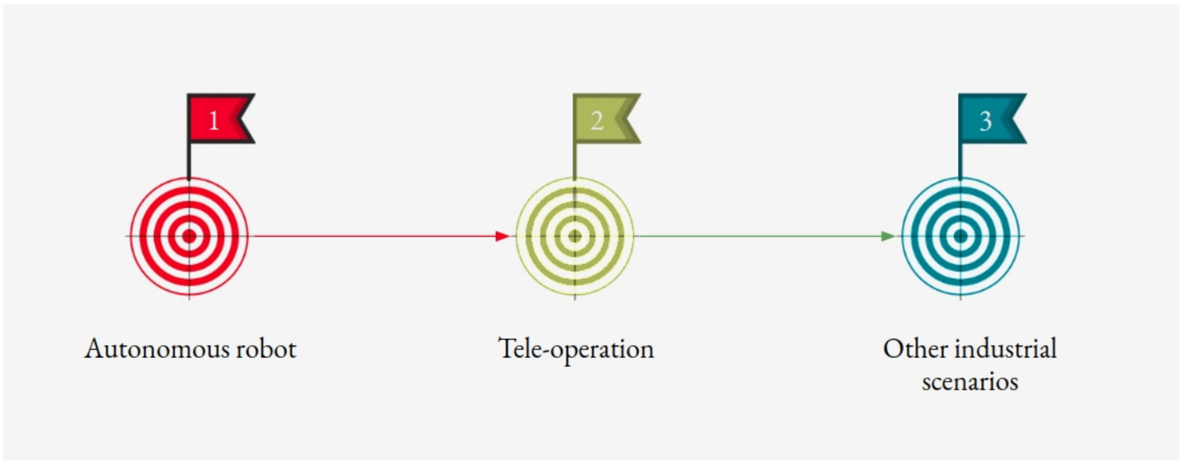


Figure A-8: Milestones

The majority of navigation systems developed thus far, however, focuses on navigation in indoor environments, through rough outdoor terrain, or based on road usage [23]. Only few systems have been designed for MAV navigation, for example in 2015, a team at the Massachusetts Institute of Technology demonstrated their autonomous drone navigating through forested areas at 30 miles per hour [11].

Localization in existing map. The basic idea for this particular task is receiving in input a global point cloud map, and then use Iterative Closest Point (ICP) [28] or Normal Distributions Transform (NDT) [33] based methods to match the current point cloud to global map.

Path planning. Path planning and trajectory planning are crucial issues in the field of Robotics and, more generally, in the field of Automation. Path planning algorithms generate a geometric path, from an initial to a final point, passing through pre-defined via-points, either in the joint space or in the operating space of the robot, while trajectory planning algorithms take a given geometric path and endow it with the time information.

Obstacle avoidance The objective is to move a vehicle around the aircraft free of collisions with the obstacles detected by the sensors during motion execution. The advantage of reactive obstacle avoidance is to compute motion by introducing the sensor information within the control loop, used to adapt the motion to any possible contingency in the hangar incompatible with initial plans. The right image in Figure 1-2 shows an example of path planning and obstacle avoidance combination.

Tele-operation at the airport

Once the team will be able to achieve all the goals described in the previous section will be also necessary to create an infrastructure for the operators to control and manage the robot remotely. Indeed, the autonomous robot need to communicate with a remote station that in real time can monitor and eventually correct the motion of the robot. This tele-operation system allow the remote operator to have control of the inspection. Furthermore Mainblaides aim to furnished to the final user the possibility to look into the history of the inspection and possibly compared multiple of them for damage comparison with previous similar situations. This system requires also communication with internet and remote servers where all the information can be stored and saved and accessible at everytime from everywhere.

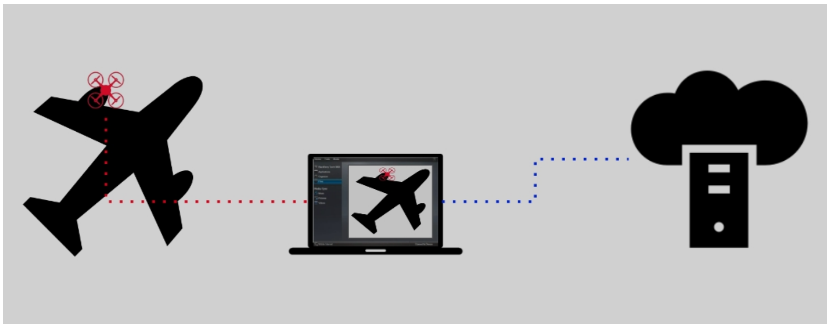


Figure A-9: Tele-operation system

Different type of inspections

Once the system is ultimated, the same inspection procedure and material can be reused in many other industrial scenarios. As mentioned also in the previous section the quantity of industries involved in this technologies is large. Mainblades aim to design and develope a solution scalable and agile to fit different industrial applications.

Appendix B

Software

B-1 TF framework

tf is a package that lets the user keep track of multiple coordinate frames over time. tf maintains the relationship between coordinate frames in a tree structure buffered in time, and lets the user transform points, vectors, etc between any two coordinate frames at any desired point in time.

How TF is used A robotic system typically has many 3D coordinate frames that change over time, such as a odom frame, base_link frame, base_footprint frame, ground truth frame, etc. tf keeps track of all these frames over time. tf can operate in a distributed system. This means all the information about the coordinate frames of a robot is available to all ROS components on any computer in the system. There is no central server of transform information.

Anyone using tf will need to listen for transforms:

Listening for transforms - Receive and buffer all coordinate frames that are broadcasted in the system, and query for specific transforms between frames. To extend the capabilities of a robot is required to start broadcasting transforms.

Broadcasting transforms - Send out the relative pose of coordinate frames to the rest of the system. A system can have many broadcasters that each provide information about a different part of the robot.

Additional information of this framework are available at [10].

B-2 Ceres

Ceres Solver is an open source C++ library for modeling and solving large, complicated optimization problems. It can be used to solve Non-linear Least Squares problems with bounds constraints and general unconstrained optimization problems. It is a mature, feature rich, and performant library that has been used in production at Google since 2010.

Additional information of this library are available at [2].

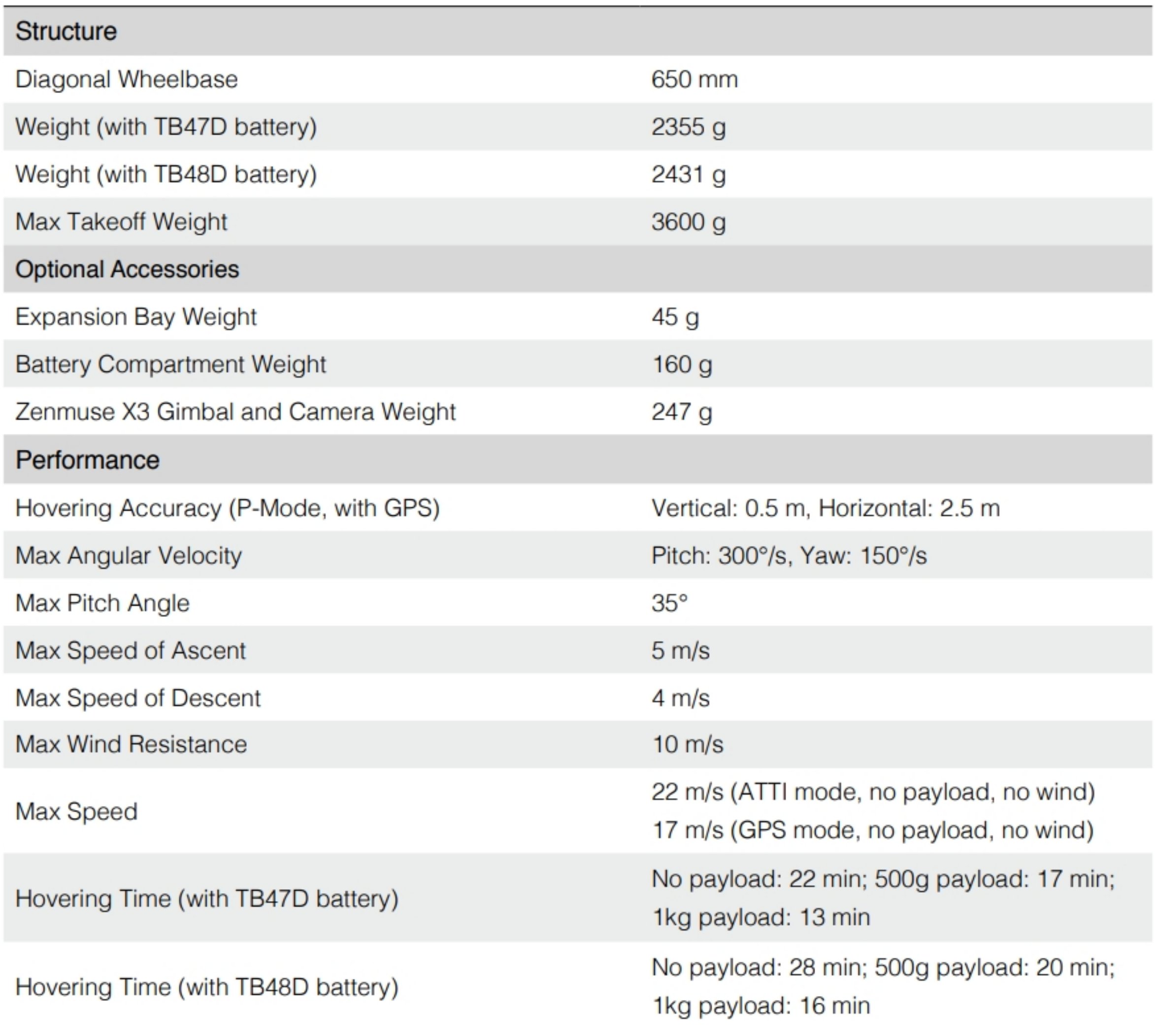
Appendix C

Hardware

C-1 DJI matrice 100

Complete flight platform The Matrice 100 platform has all of DJI's easy-to-fly technology built in, and includes the flight controller, propulsion system, GPS, DJI Lightbridge, a dedicated remote controller, and a rechargeable battery. This system automatically manages the most complex tasks required for flight [4].

Fully programmable it is possible to customize and tailor the flight platform by using the DJI SDK. In the project have been necessary to develope a position estimation based on acceleration and velocity values. This operation is executed directly inside the SDK provided by DJI. Figure C-1 shows additional hardware details of the robot.



Structure	
Diagonal Wheelbase	650 mm
Weight (with TB47D battery)	2355 g
Weight (with TB48D battery)	2431 g
Max Takeoff Weight	3600 g
Optional Accessories	
Expansion Bay Weight	45 g
Battery Compartment Weight	160 g
Zenmuse X3 Gimbal and Camera Weight	247 g
Performance	
Hovering Accuracy (P-Mode, with GPS)	Vertical: 0.5 m, Horizontal: 2.5 m
Max Angular Velocity	Pitch: 300°/s, Yaw: 150°/s
Max Pitch Angle	35°
Max Speed of Ascent	5 m/s
Max Speed of Descent	4 m/s
Max Wind Resistance	10 m/s
Max Speed	22 m/s (ATTI mode, no payload, no wind) 17 m/s (GPS mode, no payload, no wind)
Hovering Time (with TB47D battery)	No payload: 22 min; 500g payload: 17 min; 1kg payload: 13 min
Hovering Time (with TB48D battery)	No payload: 28 min; 500g payload: 20 min; 1kg payload: 16 min

Figure C-1: DJi technical details. Additional information in [4].

C-2 Hokuyo UTM-30LX

Figure C-2: Hokuyo UTM-30LX

Hokuyo UTMX-30LX is one of the most sophisticated 2D sensor produced by Hokuyo and is used in many applications because of its performance. Hokuyo UTM-30LX provides scanning range of 270° and 30 meters with an angular resolution of 0.25° per step. One full scan cycle lasts for 25 ms, which results in 40 Hz measurement frequency. Data transfer to host is realized through USB 2.0 interface, using dedicated SCIP 2.0 protocol in communication with the device.

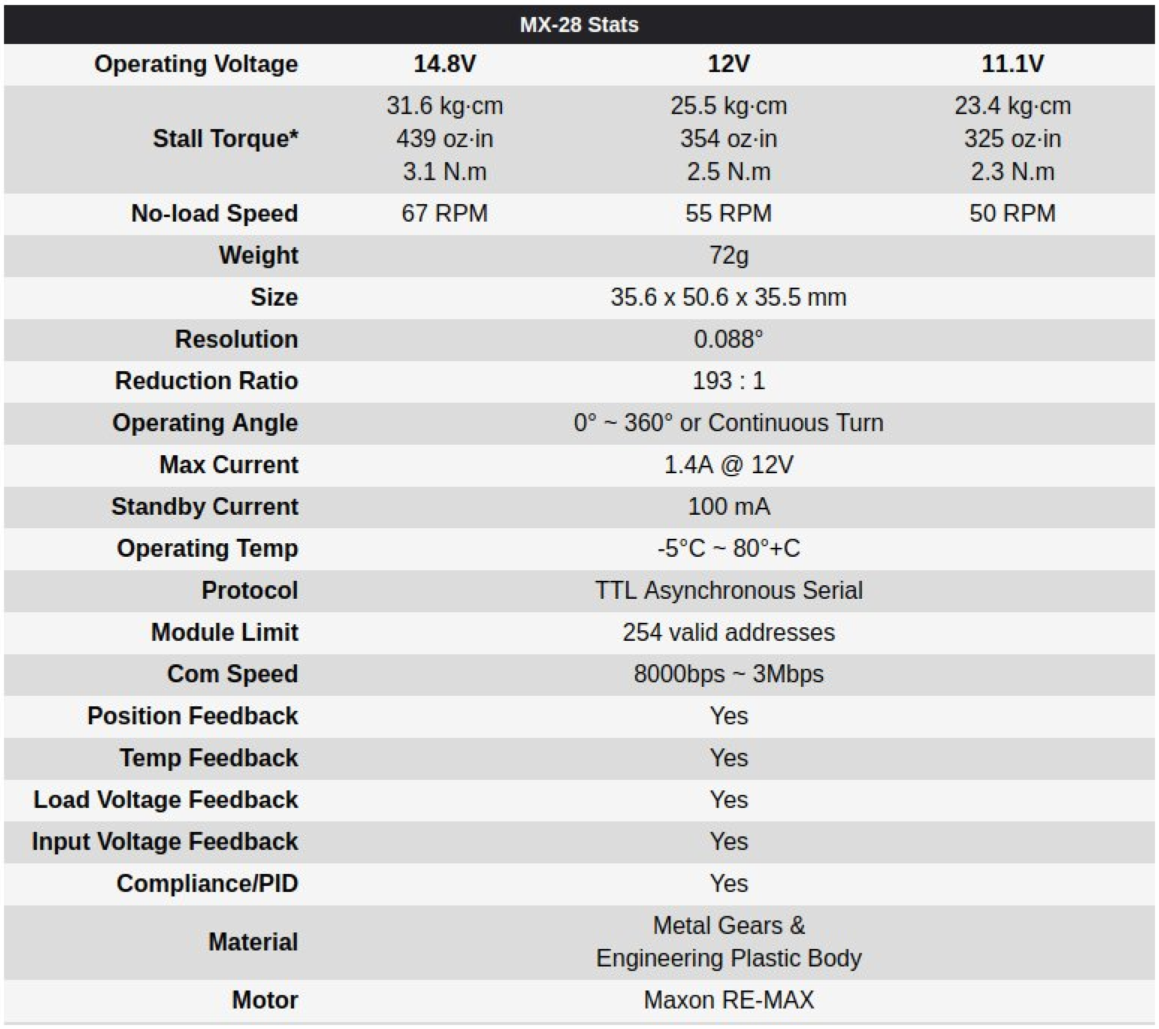
Performance The measurement accuracy, according to the producer [16], is ±30 mm at 0.1 to 10 m range, and ±50 mm at 10 to 30 m range. Precision, defined as a standard deviation of samples, is less than 10 mm at 0.1 to 10 m range, and less than 30 mm at 10 to 30 m range. The usage of laser range finder requires many tests to prove its real performance in certain environments. In [?] a great performance test of Hokuyo UTM-30LX is reported, the authors tested the laser range for the drift effect, influence of target distance, surface brightness, color and material (Table C-1), and the sensor orientation. The results show that Hokuyo UTM-30LX can be used for distance scanning, targeting different materials, colors, and brightness with similar effects. The relative errors of measurements were within ±1% of the real distance and the standard deviation was less than 5 mm. Caution must be taken during measurements of highly reflective surfaces, because the result gave higher error and larger standard deviation.

Table C-1: Summary of the measurements of different materials

	Absolute error(mm)	**Relative Error (%)**	**Standard Deviation (mm)**
Aluminium	-6.74	-0.44	4.75
Steel	7.70	0.38	5.03
CD	27.39	1.82	9.13
Cotton	12.75	0.63	4.73
Wood	4.24	0.21	4.76
Plastic	7.25	0.36	5.10

C-3 Motor -Dynamixel - MX 28

The MX-28T Dynamixel Robot Servo Actuator is the newest generation of Robotis Dynamixel actuator; equipped with an onboard 32bit 72mhz Cortex M3, a contact-less magnetic encoder with 4x the resolution over the AX/RX series, and up to 3mpbs using the new TTL 2.0 bus. Each servo has the ability to track its speed, temperature, shaft position, voltage, and load. As if this weren't enough, the newly implemented PID control algorithm used to maintain shaft position can be adjusted individually for each servo, allowing a control of the speed and strength of the motor's response. All MX Series servos use 12v nominal voltage, so AX MX Dynamixels can be mixed without having to worry about separate power supplies. All of the sensor management and position control is handled by the servo's built-in microcontroller [5].

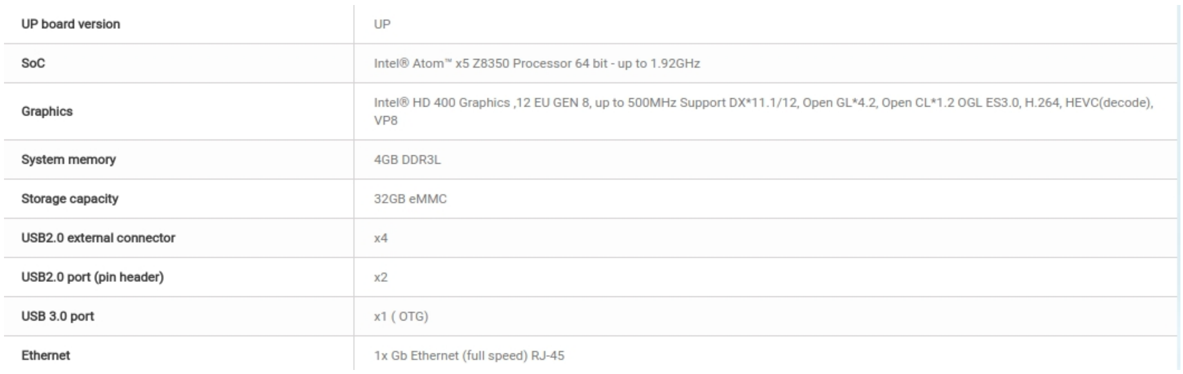


MX-28 Stats			
Operating Voltage	**14.8V**	**12V**	**11.1V**
Stall Torque*	31.6 kg·cm 439 oz·in 3.1 N.m	25.5 kg·cm 354 oz·in 2.5 N.m	23.4 kg·cm 325 oz·in 2.3 N.m
No-load Speed	67 RPM	55 RPM	50 RPM
Weight	72g		
Size	35.6 x 50.6 x 35.5 mm		
Resolution	0.088°		
Reduction Ratio	193 : 1		
Operating Angle	0° ~ 360° or Continuous Turn		
Max Current	1.4A @ 12V		
Standby Current	100 mA		
Operating Temp	-5°C ~ 80°+C		
Protocol	TTL Asynchronous Serial		
Module Limit	254 valid addresses		
Com Speed	8000bps ~ 3Mbps		
Position Feedback	Yes		
Temp Feedback	Yes		
Load Voltage Feedback	Yes		
Input Voltage Feedback	Yes		
Compliance/PID	Yes		
Material	Metal Gears & Engineering Plastic Body		
Motor	Maxon RE-MAX		

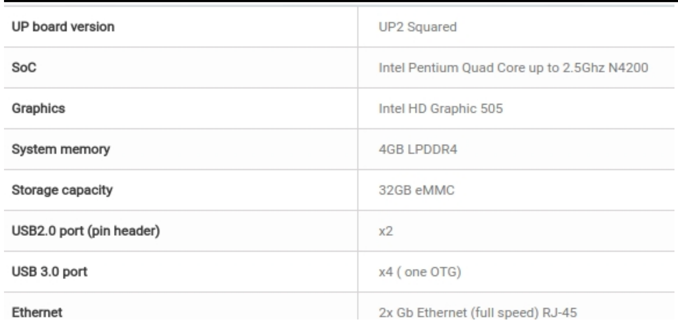
Figure C-3: Dynamixel technical details. Additional information in [5]

C-4 UP boards

The extensive features included within UP make it the perfect solution for applications such as Robotics, Drone, Machine Vision, Smart Home, Education, Digital Signage, Intelligent Cars and Internet Of Things (IoT) solutions. UP is produced by Aaeon, industrial embedded company of Asus group [12].



UP board version	UP
SoC	Intel® Atom™ x5 Z8350 Processor 64 bit - up to 1.92GHz
Graphics	Intel® HD 400 Graphics ,12 EU GEN 8, up to 500MHz Support DX*11.1/12, Open GL*4.2, Open CL*1.2 OGL ES3.0, H.264, HEVC(decode), VP8
System memory	4GB DDR3L
Storage capacity	32GB eMMC
USB2.0 external connector	x4
USB2.0 port (pin header)	x2
USB 3.0 port	x1 (OTG)
Ethernet	1x Gb Ethernet (full speed) RJ-45

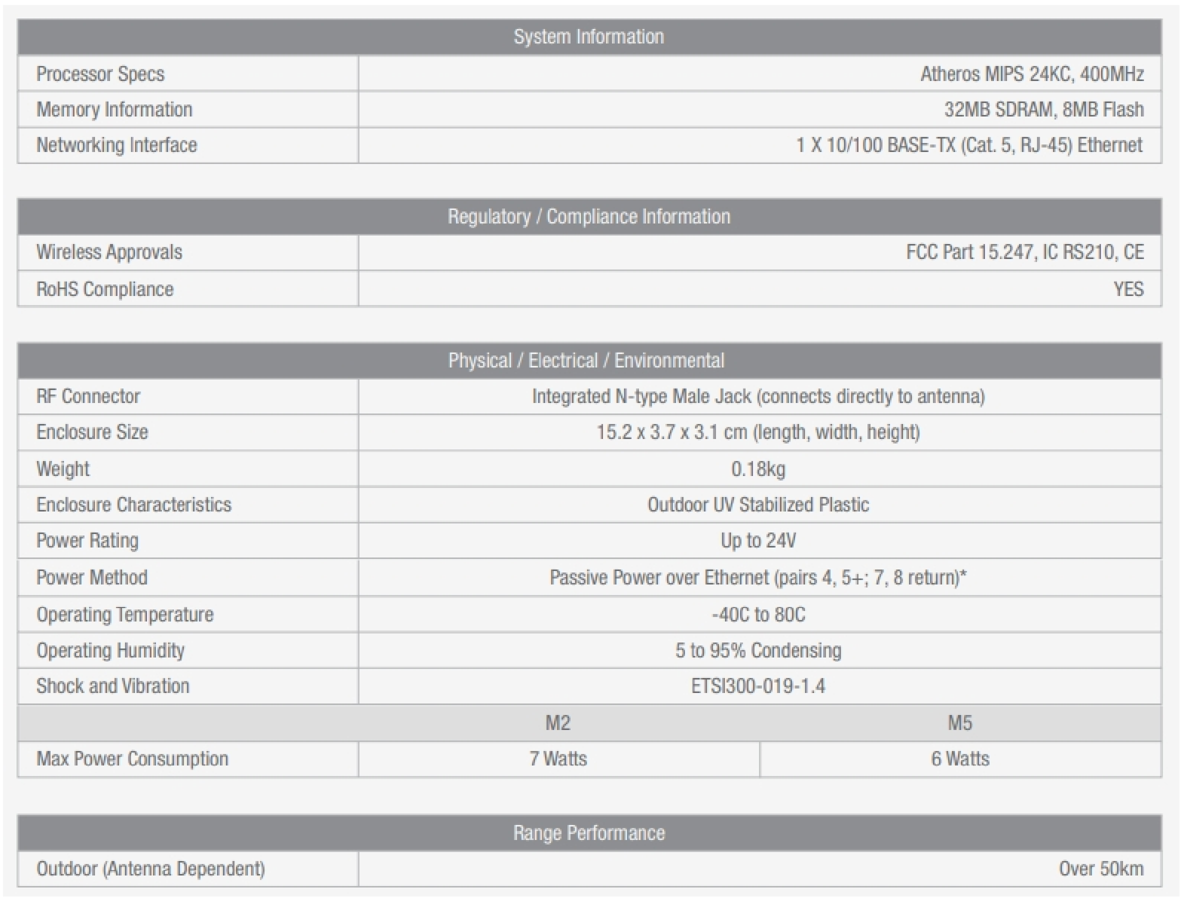


UP board version	UP2 Squared
SoC	Intel Pentium Quad Core up to 2.5Ghz N4200
Graphics	Intel HD Graphic 505
System memory	4GB LPDDR4
Storage capacity	32GB eMMC
USB2.0 port (pin header)	x2
USB 3.0 port	x4 (one OTG)
Ethernet	2x Gb Ethernet (full speed) RJ-45

Figure C-4: UP boards technical details. On top UP and on the bottom UPsquared. Additional information in [12]

C-5 Access point

Bullet M is the latest version of the popular Ubiquiti Bullet. Like its predecessor, Bullet M is a wireless radio with an integrated Type N RF connector that can be directly plugged in to any Antenna to create a powerful and robust outdoor Access Point, Client, or Bridge. The Bullet M features a signal strength LED meter for antenna alignment, a low-loss integrated N-type RF connector, and a robust weatherproof design. With up to 600mW of power and enhanced receiver design, the Bullet M is ideal for long-distance links, capable of 100Mbps+ real TCP/IP speeds over multi-km distances [1].



System Information		
Processor Specs	Atheros MIPS 24KC, 400MHz	
Memory Information	32MB SDRAM, 8MB Flash	
Networking Interface	1 X 10/100 BASE-TX (Cat. 5, RJ-45) Ethernet	

Regulatory / Compliance Information	
Wireless Approvals	FCC Part 15.247, IC RS210, CE
RoHS Compliance	YES

Physical / Electrical / Environmental		
RF Connector	Integrated N-type Male Jack (connects directly to antenna)	
Enclosure Size	15.2 x 3.7 x 3.1 cm (length, width, height)	
Weight	0.18kg	
Enclosure Characteristics	Outdoor UV Stabilized Plastic	
Power Rating	Up to 24V	
Power Method	Passive Power over Ethernet (pairs 4, 5+; 7, 8 return)*	
Operating Temperature	-40C to 80C	
Operating Humidity	5 to 95% Condensing	
Shock and Vibration	ETSI300-019-1.4	
	M2	M5
Max Power Consumption	7 Watts	6 Watts

Range Performance	
Outdoor (Antenna Dependent)	Over 50km

Figure C-5: Access point technical details. Additional information in [1]

Bibliography

[1] Bullet m - access point. https://www.ubnt.com/airmax/bulletm/. Accessed: 2018-03-16.

[2] Ceres solver. http://ceres-solver.org. Accessed: 2018-2-16.

[3] Cross product. http://tutorial.math.lamar.edu/Classes/CalcII/CrossProduct.aspx. Accessed: 2018-2-16.

[4] Dji - matrice 100. https://www.dji.com/matrice100. Accessed: 2018-03-16.

[5] Motor dynamixel mx28. http://www.trossenrobotics.com/dynamixel-mx-28-robot-actuator.aspx. Accessed: 2017-12-16.

[6] The mro lab - 3d scanning. http://www.afiklmem.com/AFIKLMEM/en/g_page_standard/MediaRelation/BlueLink_21_June2015/BlueLink_21_MROLab_3DScanning.html. Accessed: 2017-12-16.

[7] New technology for mature flight control inspections. http://www.mro-network.com/engineering-design/new-technology-mature-flight-control-inspections. Accessed: 2017-12-16.

[8] Point-line distance–3-dimensional. http://mathworld.wolfram.com/Point-LineDistance3-Dimensional.html. Accessed: 2018-2-16.

[9] Point-plane distance. http://mathworld.wolfram.com/Point-PlaneDistance.html. Accessed: 2018-2-16.

[10] Ros tf. http://wiki.ros.org/tf. Accessed: 2018-03-16.

[11] Self-flying drone dodge through tree branches at 30 miles per hour. https://www.theverge.com/2015/11/2/9659706/pushbroom-stereo-drone-uav-autonomous-sense-and-avoid. Accessed: 2017-12-16.

[12] Up boards. http://www.up-board.org. Accessed: 2017-3-16.

[13] P. J. Besl, N. D. McKay, et al. A method for registration of 3-d shapes. *IEEE Transactions on pattern analysis and machine intelligence*, 14(2):239–256, 1992.

[14] M. Bosse and R. Zlot. Map matching and data association for large-scale two-dimensional laser scan-based slam. *The International Journal of Robotics Research*, 27(6):667–691, 2008.

[15] M. Bosse and R. Zlot. Continuous 3d scan-matching with a spinning 2d laser. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 4312–4319. IEEE, 2009.

[16] D. Droeschel, Holz, and S. Behnke. Scanning range finder utm-30lx product page. hokuyo automatic co., ltd.,. May 2017.

[17] H. Durrant-Whyte and T. Bailey. Simultaneous localization and mapping: part i. *IEEE robotics & automation magazine*, 13(2):99–110, 2006.

[18] J. M. Font and J. A. Batlle. Mobile robot localization. revisiting the triangulation methods. *IFAC Proceedings Volumes*, 39(15):340–345, 2006.

[19] D. A. Haner and R. T. Menzies. Reflectance characteristics of reference materials used in lidar hard target calibration. *Applied optics*, 28(5):857–864, 1989.

[20] Hp. Hp zbook studio g4 mobile workstation. http://store.hp.com/. Accessed: 2018-03-10.

[21] N. Kaempchen and K. Dietmayer. Data synchronization strategies for multi-sensor fusion. In *Proceedings of the IEEE Conference on Intelligent Transportation Systems*, pages 1–9, 2003.

[22] L. Kaul, R. Zlot, and M. Bosse. Continuous-Time Three-Dimensional Mapping for Micro Aerial Vehicles with a Passively Actuated Rotating Laser Scanner. *J. Field Robotics*, 33(1):103–132, Jan. 2016.

[23] R. Kümmerle, M. Ruhnke, B. Steder, C. Stachniss, and W. Burgard. Autonomous robot navigation in highly populated pedestrian zones. *Journal of Field Robotics*, 32(4):565–589, 2015.

[24] Y. Li and E. B. Olson. Structure tensors for general purpose lidar feature extraction. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 1869–1874. IEEE, 2011.

[25] P. A. Lorenzo. *3D LiDAR SLAM with a MAV for autonomous aircraft inspections.* TUDelft, 2017.

[26] R. M. Murray, Z. Li, S. S. Sastry, and S. S. Sastry. *A mathematical introduction to robotic manipulation.* CRC press, 1994.

[27] G. Pahl and W. Beitz. *Engineering design: a systematic approach.* Springer Science & Business Media, 2013.

[28] F. Pomerleau, F. Colas, R. Siegwart, and S. Magnenat. Comparing icp variants on real-world data sets. *Autonomous Robots*, 34(3):133–148, 2013.

[29] Pozyx. Website. https://www.pozyx.io. Accessed: 2018-02-03.

[30] S. Rusinkiewicz and M. Levoy. Efficient variants of the icp algorithm. In *3-D Digital Imaging and Modeling, 2001. Proceedings. Third International Conference on*, pages 145–152. IEEE, 2001.

[31] R. B. Rusu and S. Cousins. 3d is here: Point cloud library (pcl). In *Robotics and automation (ICRA), 2011 IEEE International Conference on*, pages 1–4. IEEE, 2011.

[32] M. W. Spong, S. Hutchinson, M. Vidyasagar, et al. *Robot modeling and control*, volume 3. Wiley New York, 2006.

[33] T. Stoyanov, M. Magnusson, H. Andreasson, and A. J. Lilienthal. Fast and accurate scan registration through minimization of the distance between compact 3d ndt representations. *The International Journal of Robotics Research*, 31(12):1377–1393, 2012.

[34] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A benchmark for the evaluation of rgb-d slam systems. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 573–580. IEEE, 2012.

[35] R. Swinburne. Bayes's theorem. 2008.

[36] Techopedia. System design. http://www.techopedia.com/definition/29998/system-design. Accessed: 2018-02-14.

[37] Velodyne. Website. http://velodynelidar.com/vlp-32c.html. Accessed: 2018-03-10.

[38] S. Winkelbach, S. Molkenstruck, and F. M. Wahl. Low-cost laser range scanner and fast surface registration approach. In *Joint Pattern Recognition Symposium*, pages 718–728. Springer, 2006.

[39] J. Zhang and S. Singh. Low-drift and real-time lidar odometry and mapping. *Autonomous Robots*, pages 1–16, 2016.

[40] R. Zlot and M. Bosse. Efficient large-scale 3d mobile mapping and surface reconstruction of an underground mine. In *Field and service robotics*, pages 479–493. Springer, 2014.