FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

3D Reconstruction of Civil Infrastructures from UAV Lidar point clouds

Leonardo Gomes Ribeiro

MESTRADO INTEGRADO EM ENGENHARIA ELETROTÉCNICA E DE COMPUTADORES

Supervisor at INESC TEC: Dr. Marcelo Roberto Petry

Supervisor at FEUP: Prof. Dr. António Paulo Moreira

July 27, 2021

Abstract

Nowadays, infrastructures for transportation, communication, energy, industrial production and social purpose are presented as pillars of society, being essential for its proper functioning. Coupled with this great importance within society, there is a need to ensure the safety and durability of these assets. Thus, reliable techniques should be used to assess their condition.

With technological advances and the development of new methods of data acquisition, some tasks related to civil construction, currently performed by human beings, such as inspection and quality control, become inefficient due to their time and cost. In this context, 3D reconstruction of infrastructures appears as a possible solution, presenting itself as a first step for the monitoring of infrastructures, as well as a tool for semi or completely automated inspection processes.

For the development of this thesis, a light detection and ranging (LIDAR) sensor coupled to a unmanned aerial vehicle (UAV) was used. With this equipment, it became possible to autonomously fly over real infrastructures, extracting data from all its surfaces, regardless of the difficulties that could arise to reach such regions from the ground. The data is extracted in the form of point clouds with respective intensities, filtered and used in meshing and texturing algorithms, resulting in a virtual three-dimensional representation of the target infrastructure. With these representations, it is possible to evaluate the evolution of the infrastructure during its construction or repair, as well as to evaluate the temporal evolution of certain defects present in the construction, by comparing models for the same scenario obtained from data extracted on different occasions. This approach allows the process of monitoring infrastructures to be carried out more efficiently, with lower costs and ensuring the safety of workers.

Resumo

Atualmente, as infraestruturas de transporte, comunicação, energia, produção industrial e finalidade social apresentam-se como pilares da sociedade, sendo essenciais para o seu bom funcionamento. Aliada a essa grande importância na sociedade, existe a necessidade de garantir a segurança e a durabilidade desses ativos. Assim, técnicas confiáveis devem ser utilizadas para avaliar as suas condições.

Com o avanço tecnológico e o desenvolvimento de novos métodos de aquisição de dados, algumas tarefas relacionadas com a construção civil, atualmente realizadas por seres humanos, como a inspeção e o controlo de qualidade, tornam-se ineficientes pelo seu tempo e custo. Neste contexto, a reconstrução 3D de infraestruturas surge como uma possível solução, apresentando-se como o primeiro passo para a monitorização de infraestruturas, bem como uma ferramenta para inspeções semi ou totalmente automatizadas.

Para o desenvolvimento desta tese, foi utilizado um sensor LIDAR acoplado a um UAV. Com este equipamento tornou-se possível sobrevoar de forma autónoma infraestruturas reais, extraindo dados de todas as suas superfícies, independentemente das dificuldades que possam surgir para chegar a tais regiões a partir do solo. Os dados são extraídos na forma de nuvens de pontos com respectivas intensidades, filtrados e utilizados em algoritmos de malha e texturização, resultando em uma representação tridimensional virtual da infraestrutura alvo. Com estas representações é possível avaliar a evolução da infraestrutura durante a sua construção ou reparação, bem como avaliar a evolução temporal de determinados defeitos presentes na construção. Para realizar as avaliações, será feita a comparação entre modelos, do um mesmo cenário, obtidos a partir de dados extraídos em diferentes ocasiões. Esta abordagem permite que o processo de monitorização de infraestruturas seja realizado de forma mais eficiente, com custos mais baixos e garantindo a segurança dos trabalhadores.

Acknowledgments

Since I am Portuguese, as well as my family and friends, I will write my acknowledgments in my native language.

Em primeiro lugar, e como não poderia deixar de ser, agradeço a toda a minha família, em especial aos meus pais, irmã e avós, por todo o apoio que me deram em todas as etapas da minha vida. Deixo também um agradecimento muito especial à minha namorada e aos meus amigos por tornarem estes últimos anos nos incríveis anos que foram. Agradeço-lhes também por toda a ajuda e motivação que me deram, ajuda essa que foi fundamental em todas as situações mais complicadas desta caminhada.

Agradeço aos meus orientadores, Dr. Marcelo Roberto Petry e Professor António Paulo Moreira, por toda a ajuda e conselhos que me deram durante a realização do projeto. Sem eles, este projeto não teria alcançado os resultados apresentados. Deixo também o meu agradecimento a ambos os proprietários das infraestruturas escolhidas para extração dos *datasets*, mais precisamente à Sogrape SA pela autorização na realização de levantamentos aéreos com recurso a um drone na Quinta do Seixo e a possibilidade da divulgação dos mesmos, bem como ao proprietário da moradia, pois sem as suas disponibilidades seria difícil apresentar resultados tão interessantes. Agradeço ainda a todos os professores que tive, pois foi graças a eles que consegui adquirir grande parte dos conhecimentos necessários para a realização deste trabalho.

Leonardo Ribeiro

Contents

List of Figures

List of Tables

Abbreviations and Symbols

AI	Artificial Intelligence
API	Application Programming Interface
BIM	Building Information Modeling
CDT	Construction Digital Twin
CUDA	Compute Unified Device Architecture
DoF	Degrees of Freedom
DT	Digital Twin
FoV	Field of View
GNSS	Global Navigation Satellite System
ICP	Iterative Closest Point
IMU	Inertial Measurement Unit
IoT	Internet of Things
IP	Internet Protocol
LCF	Local Consistency Factor
LIDAR	Light Detection and Ranging
MT	Maximum Flight Time
OSDK	Onboard Software Development Kit
PCA	Principal Component Analysis
PCL	Point Cloud Library
PLM	Product Lifecycle Management
PW	Payload Weight
RATIM	Robot-Aided Tunnel Inspection and Maintenance
RFID	Radio Frequency Identification
ROS	Robot Operating System
RTK	Real-Time Kinematic
SLAM	Simultaneous Localization and Mapping
SRC	Spectral Residual Clustering
SSD	Solid State Drives
TCP	Transmission Control Protocol
TSDF	Truncated Signed Distance Field
TTL	Transistor-Transistor Logic
UART	Universal Asynchronous Receiver/Transmitter
UAV	Unmanned Aerial Vehicle
UMV	Unmanned Marine Vehicle
USV	Unmanned Surface Vehicle
UUV	Unmanned Underwater Vehicle

Chapter 1

Introduction

1.1 Problem

In the area of civil construction, when a project is created information about its final state will have to be acquired in order for its construction or maintenance to be carried out successfully. Since these tasks tend to take quite a long time to be concluded, it is important that the customers and the engineers responsible for the work can monitor its evolution. Thus, it is necessary to have a data acquisition process that occurs in parallel to the work. In order to perform the data acquisition, human beings will be in charge of assessing the state of the work, obtaining information regarding the significant advances developed, problems that may have occurred, or other relevant information. These workers could either be employees hired specifically to carry out this task or simply technical staff who suspend their duties for that purpose. This process, therefore, calls for reducing the number of workers to perform their duties, which will lead to an increase in the number of hours required to complete the work, or for increasing the number of workers, leading to increased costs. In addition, the data acquisition process will have to be executed in a significantly recurring manner so that the information obtained and the information actually existing are as similar as possible.

Given the nature of the places where the buildings are located, as well as the characteristics that they present, the work of acquiring data presents itself as a task of extreme risk for the human being who performs it, a risk that is aggravated due to the need of countless repetitions throughout the process in question.

1.2 Context

To work efficiently, society depends on a set of critical infrastructures of transport, communication, energy and industrial production, as well as buildings of social purpose. The infrastructures present themselves as the core of the multiple economic activities, which is why they always must be in good condition. In order to guarantee these conditions, it is necessary that the infrastructures are constantly monitored, so that any failure is detected and subsequently repaired as quickly as

possible. In addition to the infrastructures already built, where the monitoring process substantially consists of detecting and preventing failures, the monitoring process is also applicable to infrastructures under construction. The construction of infrastructures is usually a project characterized by being time-consuming and requiring a lot of resources. Thus, the appearance of a failure could compromise the entire work, leading to very significant losses. The monitoring process is also useful to measure the quantities already used, since some contractors are paid according to these measures.

For both cases, the constant need for monitoring, combined with its strict inspection criteria, ensures the quality of the infrastructure and, consequently, its proper functioning. However, given the requirement of the process, as well as its constant need, ensuring good conditions for an infrastructure is a very expensive task.

1.3 Motivation

As a way to ensure the security and durability of assets, there is a necessity to develop reliable methods that will evaluate the state of the building during construction or maintenance processes. Traditionally, those tasks were performed mostly by human beings, making it more expensive, dangerous for the worker, and with several limitations regarding the ability of the worker to maintain quality of work overtime. In this sense, and as a way to solve this problem, robotics for infrastructures is a technological key to revolutionize Portugal's and Europe's economies in the civil construction sector, leading a new era of intelligent resource management. Robotics presents itself as the next step in the evolution of this area, providing numerous benefits compared to traditional methods, such as being able to operate in dangerous environments, continuously (without the need to rest) and at a lower cost. Robotics also brings an improvement in terms of quality of work, since they can evaluate the scenes more carefully without being influenced by distractions.

Since the construction or maintenance process of an infrastructure is characterized by being time-consuming, it is important to extract information from the infrastructure regarding its current state, allowing it to be monitored without the need to travel to the location. This need motivates the creation of a virtual representation of the infrastructure, which is updated according to the evolution of the real infrastructure.

One of the most recurrent techniques for creating virtual models from real infrastructures is photogrammetry, which consists of using local images as a starting point for the creation of the three-dimensional model. The fact that this approach is well defined in the state of the art motivated the use of a different approach for acquiring the data. Beyond this fact, the evolution of technology led to the improvement of the quality of 3D sensors such as LIDAR. With this improvement these sensors are becoming more accurate, which was also an aspect that motivates their use on the realization of this thesis.

1.4 Goals

This dissertation focuses on the development of methodologies to create a three-dimensional representation of civil infrastructure assets. These digital representations are going to be built from data acquired by an UAV equipped with a LIDAR. In each mission, the UAV will navigate through the structure autonomously, acquiring point clouds. To achieve this main goal, a set of secondary objective will have to be fulfilled, namely:

- Understand and define the steps to be used for three-dimensional reconstruction;
- Development of a robotic operating system (ROS)[1] node capable of receiving, transforming and merging different point clouds into a single registered point cloud;
- Development of an algorithm capable of generating a three-dimensional representation from a point cloud;
- Perform the comparison between meshes in order to detect differences between them;
- Development of a node capable of creating and sending a mission to the UAV, so that it performs it autonomously.

1.5 Thesis Outline

The document is divided into five chapters, these being the introduction, the literature review, the proposed approach, the results obtained and the conclusion.

In chapter one, the theme of the dissertation is presented, contextualizing the reasons that aroused the interest in this project, and the objectives intended with its development are mentioned.

In chapter two, a study is made regarding the state of development in which technologies, such as digital twins (DTs), 3D reconstruction, and the application of robots in civil construction, are. In each of these sub-chapters, the information is based on studies developed in the different areas.

In chapter three all the process used to develop this thesis is explained. The chapter starts by explaining all the methods used and developed for the acquisition of the data, and it ends presenting the developed system, which is capable of transforming a set of point clouds into a 3D model.

In chapter four the results of the different stages of the developed system are presented. In this chapter it is also explained the reasons for the obtained results.

In chapter five a conclusion is made, making a comparison of the method developed with different approaches of this problem. In this chapter the future work is also presented.

Chapter 2

Literature Review

In this Chapter it is presented a literature review on DTs, 3D reconstruction methodologies, and the presence of robots in the construction sector. In the first Section (2.1), a review of the state of the art on DTs is made, explaining their advances and utilization. In Section 2.2 the process of creating a 3D model is explained based on methods developed by different researchers. Finally, in Section 2.3 an overview of the application of robots to perform infrastructure inspections and maintenance is presented.

2.1 Digital Twins

In 2002, the concept of DTs was introduced as a tool for product lifecycle management (PLM) course called "Conceptual Ideal for PLM". This term, with the development of technology, has evolved to become a powerful tool for decision-making that is now fundamentally associated with the industrial revolution 4.0 [2].

A DT consists of a virtual copy of an asset, product, or physical process, and its main objective is to improve the characteristics of its physical twin. The development of the IoT is a major pioneer of this technology, as it allows the DT to, continuously, receive data from the sensors present in the physical component [3]. This data is processed in the DT and depending on the contained information, it can return information to the physical unit, which may interact with the real world through the physical twin actuators, thus creating a continuous cycle of information changes. Another aspect that favors the DT concept is the significant increase in the amount of information available that is associated with any product, usually called "Big Data". This significant increase in the amount of data allows the DT to simulate, in the environment where its physical twin is, the various possible scenarios, making it able to make the best decision [4]. In this way, a DT provides a system with the ability to improve autonomously.

In addition to simulation, a DT may be able to develop and verify the behavior of real equipment, being able to detect problems in the early stages of development, or even predict problems that may happen. One of the areas where this concept is gaining more and more importance is civil construction.

2.1.1 Constitution of a Digital Twin

The concept of DT, despite being arousing a lot of interest on the part of researchers, does not yet have a universal conceptual composition. However, in general, a DT is made up of domains in the areas of sensing, data connection, and learning mechanisms [3].

2.1.1.1 Sensorization

The vast majority of authors consider sensing as an essential domain of a DT, being the source of real-time data [4, 5]. Some authors point to the digital model as a great sensor used to predict behaviors of the associated physical system. In construction assets cyber-physical systems, RFID tags and scanners are used, all being interconnected to facilitate communication between the real world and the virtual model.

Starting from the data extracted by the sensors, and having a previously defined meaning and structure, the environment can be monitored, being possible to activate certain actuators when values obtained exceed limits. From the operator's point of view, the monitored values can be expressed through graphs, tables, 3D models, or simulations.

It is common knowledge that a DT should be capable of encompassing the entire life cycle of his physical twin, varying the form of implementation taking into account the application area. Through the analysis of these cycles, DTs of future generations will have greater knowledge, leading to better decision-making. In construction, there could be improvements regarding the costs of construction development and security [3].

2.1.1.2 Data Connection

Studies related to smart cities, consider that building information modeling (BIM) should become a starting point for DT, providing information related to the 3D model of the desired construction. BIM is a process that involves the creation of a 3D model and allows the control of documentation, coordination, and simulation during the life cycle of a project, making it possible to help the stakeholder to understand the format that the structure will have even before of this being built [6]. Although it was initially introduced with a collaboration improvement tool, it ended up being involved in the entire infrastructure life cycle.

The inclusion of IoT is also pointed out as an aspect that will play a crucial role in monitoring, being responsible for creating the bridge between the physical and the digital component. This technology can also be used to monitor the way the product is used by the customer, presenting itself as a benefit for both parties involved [3].

2.1.1.3 Learning Process

The DT credibility is related to how it makes decisions based on the information it receives. To improve this interaction, some mechanisms can be applied. The ability to simulate the real environment is an essential aspect of this improvement, where the quality of the simulation influences

the learning depth, based on the rigor required for the model [3]. For example, the simulation of a nuclear power plant must be more accurate, representing with more detail the reality, than a normal water tank, given the nature of the infrastructure and their respective liabilities. An important aspect to take into account, in this approach, is the quality of the sensors present in the physical twin, as significant errors can compromise the learning process.

While simulations improve the ability to make decisions according to events that have occurred, forecasting is also a crucial aspect of a DT, giving it the ability to predict events even before they happen. "Big Data" presents itself as a driver in this field, supplying large amounts of data to Machine Learning and Data Mining algorithms, which will extract and analyze information patterns making it able to predict future situations taking into account past events.

2.1.2 Application on Construction

The emergence of tools such as BIM has brought a major change in the way construction information is created and stored. Although, with the large increase in the amount of information available and the interconnection between various devices through the IoT, BIM does not become compatible with all this information, getting limited to only a small part of the entire surrounding context. Therefore, there is a need to move from using BIM to construction digital twin (CDT). This DT has different features responsible for allowing the realization of specific services, which are present during all building life-cycle, being constituted by different tools that vary along the stages [3].

Currently, the sensing of construction sites is carried out by human beings, who are responsible for managing updates and documentation. This approach compromises the DT's ability to simulate and predict the behavior of the environment, as all information received will be out of date and may contain errors. To solve this problem, it is necessary to increase the quality of data obtained by sensing the locations and guarantee a periodic flow of information between the location and the virtual model. For DT to be able to analyze the real world in great detail, it needs large amounts of information, making it difficult to guarantee the continuous flow of information and the maintenance of the various sensors necessary for the acquisition of such data. In this sense, there is a need to apply automation to monitoring. The use of 3D scanners in UAVs is an example where it is possible to obtain large amounts of information in a relatively short time. With these technologies it will be possible to improve the logistics of the site, increase its security and reduce costs in the long run.

The management of construction sites is largely based on task planning, keeping costs to values previously defined and giving better use to materials. In addition, construction sites are characterized as one of the most dangerous workplaces. Thus, a CDT will have to be able to carry out the entire management process of the site, updating the schedule of tasks and costs, taking into account the changes that arise in the environment, as well as ensuring the safety of workers, collecting data regarding the number of operators present, their respective location and the condition in which they are.

Based on the current level of development in this area, Boje et al. [3] proposed an approach where a CDT is divided into 3 generations. The authors consider that the creation of a CDT must be progressive and continues throughout the life cycle of the building. In the first generation, the CDT will have the ability to sense the environment through the sensors of its physical twin and analyze the data. The CDT is designed as an improved version of BIM, being able to predict and optimize scenarios, as well as being able to acquire information for the next stages of the life cycle. In the second generation, the knowledge base is created by enhancing monitoring platforms where all the IoT integrated devices are represented. Then, intelligence is achieved using some rules and AI algorithms. The capacity of the actuators is no longer limited to embedded emergency procedures, but also covers security, energy consumption, problem details, and recommendations for complex situations, which require verification and authentication. Finally, in the third generation, the DT is able to analyze information through Machine Learning mechanisms, becoming self-reliant, self-updatable, and self-learning. The actuation over the environment becomes autonomous but requires human supervision.

2.2 3D Reconstruction

3D reconstruction is the process of creating a digital representation of a real object, based on its shapes and appearance. To accomplish that, a set of tasks must be done, starting from the acquisition and prepossessing of the point cloud associated with the object, from that, the points are connected in a way that a mesh is created, and finally, is given some texture to the mesh, so that the final model can be, as much similar, to the real object, as possible.

2.2.1 Point Cloud Acquisition

A point cloud is a set of points in the space defined by a coordinate system that may represent the shape of a physical object. Its acquisition is a very important step when creating a 3D model since point clouds are the core of all the processes that will be done to achieve the final result, so more accurate point clouds will create better models. Thus, is very important to start by defining in which way the point cloud will be acquired. According to R. Khilar et al. [7] it is possible to divide the techniques used to extract this type of information into Passive or Active. Passive techniques use images to record the radiance reflected by the surface of different objects. These images are then used to build 3D models of the objects analyzed. These methods are characterized by having great accuracy when the environment is controlled, otherwise, the light reflection may generate some noise in the image. Active techniques, on the other hand, use sensors that emit energy, such as lasers, lights or ultrasounds, onto the object as a way to measure the distance to it, an example of these sensors are LIDARs.

2.2.1.1 Photogrammetry-based methods

Photogrammetry is an image-based 3D reconstruction method defined as “the art, science, and technology of obtaining reliable information about physical objects and the environment, through processes of recording, measuring, and interpreting imagery and digital representations of energy patterns derived from non-contact sensor systems” [8]. Being an embracing subject, it has different areas of application such as engineering, medicine, architecture, archaeology, topography, among others. Medicine benefits from this method for example when analyzing MRI and CT data [9]. In engineering, more precisely in the robotic industry, this method is used for 3D environment detection, grasping, and welding. Some approaches use a light source, as well as a camera, to increase the accuracy. This technique performs inefficiently in environments exposed to solar light. Another alternative is stereo-photogrammetry. This concept was developed based on binocular vision, which consists of perceiving depth by overlapping the field of view from each eye. The two cameras are used to simulate the human eyes [10]. Given its wide use, photogrammetry turns out to be an area in constant evolution, with several studies on the subject.

The basic algorithms on stereo matching techniques are not very complex but are computationally exhaustive. Sah and Jotwani proposed a "parallel implementation of the fast stereo matching algorithm based on correlation of multi-resolution images using CUDA" [11, p. 47]. This implementation presents a faster performance than most of the software implementations and other optimized methods in the same area [12, 13, 14, 15]. This improvement enables the usage of stereo matching in real-time.

Ivanavičius et al. developed a stereo matching algorithm called *Cyclops2*, which "produces a disparity image, provided two rectified grayscale images" [16, p. 1]. For matching, the authors use the minimization of a weight function obtained from the absolute difference in pixels intensity.

Masson [17] proposed a method that uses RGB images from monocular cameras. The author starts by trying to find features, which are patterns in the picture that have different properties relative to his local neighbors. From that point, a matching between features from different images is performed, and the posture of the images is estimated, having them as a reference. The point cloud is generated based on a template matching method.

Bici et al. [18] proposed a method where photogrammetry is used on the reconstruction of ancient statues, which are known for having metal reflective surfaces. For that, the authors proposed a photogrammetry workflow capable of overcoming some of the main issues associated with reflective surfaces. As a way to validate the developed method, the authors used a 3D scanner over the same surface and compared both results.

2.2.1.2 LIDAR-based methods

With technology improvements, LIDARs are becoming more affordable and with more capabilities. As a direct effect, they are becoming more popular in the field of 3D reconstruction. LIDAR has two ways of measuring distances. The first is time of flight, were the sensor emits laser beams into the objects around it, those beams, as soon as they hit the objects, are reflected to the sensor,

and based on the speed of light and the time passed between the emission and the reception, is calculated the distance to the object and creates a point, that represents the surface of the object. The second way for measuring distances is the phase shift, were instead of a pulsed source the sensor emits a continuous source with power modulated at constant frequency, which means that the input can be seen as a sine wave relating laser power (y-axis) with time (x-axis). Thus, knowing the phase difference in radians, the distance can be obtained[1]. Similarly, with photogrammetry-based methods, LIDAR-based methods have a wide area of applications, which is why several studies based on this technology have been carried out. Yi et al. [19] developed a method capable of creating a 3D model of an urban building based on a raw point cloud extracted directly from a LIDAR.

Besides the 3D reconstruction purpose, LIDAR information can be used in other areas. In the robotic industry, through point clouds, it is possible to obtain additional information that will help robots to perform their functions, such as the distance to objects or people, or to create a map of the world around it. Zhang and Singh [20] proposed a method that uses a 2-axis LIDAR with 6 degrees of freedom (DoF) to calculate, in real-time, the odometry and map the area surrounding a robot. For the odometry calculation, an algorithm estimates the velocity of the robot based on the LIDAR data. For mapping the environment, a second algorithm is responsible for performing the matching and registration of the point clouds.

As a way to improve both the LIDAR base and photogrammetry-based method to create a point cloud, Li et al. [21] proposed a method that uses both the point cloud generated from the images and the point cloud generated by a LIDAR. The system can be divided into two stages. In the first one, LIDAR measurements are used to improve the density of point cloud extracted from the images, and in the second phase, LIDAR measurements, from areas where the depth can not be estimated using only images, are added to the final point cloud.

2.2.2 Noise removal

LIDAR sensors data may contain noise, usually presented in the point cloud as points which are not related to the object in the analysis. These points are called outliers and can compromise all the reconstruction processes, once that the reconstruction algorithms will assume them as part of the object and will use them to create the mesh. To remove those undesired points is crucial to apply some algorithm to the point cloud so that the maximum number of outliers can be removed.

For that, Zhang et al. [22] proposed an outlier removal approach that does not have parameters and is based on density. The authors used a local consistency factor (LCF) to find points with similar local density. With this metric, the point cloud is filtered and parameters for density-based clustering are estimated. Unlike other density-based methods, the parameter estimation is calculated autonomously. Then, to enhance the outlier detection, the distance based on the fusion of spatial and color information is obtained. The outlier elimination with this method can reduce forward complexity.

[1] https://seminex.com/lidar/

Bastos et al. [23] proposed a method where the input data is reduced into a smaller set so that further optimizations can be executed quickly. To perform the reduction, only geometric operations are used, making it fast and deterministic. With this method, only the true outliers are removed in a way that the optimal solution is kept intact.

2.2.3 Surface Reconstruction

A mesh is a 3D surface that defines how the points of the point cloud will be connected. After this process, it will be possible to get a digital representation of the desired object.

Kazhdan et al. [24] proposed a Poisson method that consists of creating an indicator function X that can solve the reconstruction problem. The function defines as ones the point presented inside the model, and zeros the points found outside. The authors then obtained the reconstructed surface by extracting an isosurface. The gradient of the X function is always zero until it gets close enough to the surface where it gets the surface normal value, which enables the association of the oriented points to the gradient of the function X. This relation permits reducing the problem of finding the X function to invert his gradient. Thus, the function X that best approaches the field, defined by the oriented points, must be found. Applying it to a divergence operator, the problem turns into a Poisson problem. To discretize an adaptive octree is used. With this, it is possible to define the level of detail. Higher values generate models with more quality but will require a bigger computational capacity.

Wongwaen et al. [25] proposed an algorithm that starts by creating a cube around the point cloud and dividing it into small equal cubes. Inside each cube, there are some points of the cloud. The connection of these points creates a local mesh, named by the authors as sub mesh. To accomplish that task, it is necessary to create an initial triangle, starting by selecting two points in line, then, from the rest of the points, selecting the third one. With the first triangle formed, the rest of the algorithm continues, by associating another point to each edge of the triangle, creating new triangles. This process continues until there are no more points available. Once the process is finished, it is necessary to merge all the different sub meshes into a global mesh, for that, triangles between sub meshes frontiers are created.

Zavodny et al. [26] proposed a method capable of triangulating data acquired with a LIDAR by identifying building surfaces and applying triangulations for each individually. Since the algorithm uses input point sets for each surface individually, two planar regions will not make a close mesh. For that, the authors start by getting together regions' edges. Then, to reduce the complexity of the triangulation, some points from the cloud are removed, leaving only the minimum necessary number of points. With all the points removed, a concave triangulation is performed.

Roldão et al. [27] proposed an algorithm capable of performing a 3D reconstruction from depth sensors. The input data must be sampled by a voxel grid [28]. The number of points presented in each voxel, their mean, and covariance, are stored in some variables. These variables are a rich and compact way to represent the points inside each voxel. Fitting the data acquired on a planar surface for each voxel will lead the reconstruction to become noisier because the data have an heterogeneous density and some voxels may end up with a very reduced number of points.

To solve the problem, the authors used an adaptive neighborhood to perform approximations of the surface. This neighborhood is defined at the vertex location. Being K the level of the neighborhood, there will be $2K^3$ voxels in it. For a given neighborhood is then measured the number of points, the statistical mean, and his covariance. Those statistics are then used, according to some restrictions, on a principal component analysis (PCA) so the local planar surface could be estimated. To reconstruct a surface maintaining a high density and level of detail, from areas with originally low density, an optimal neighborhood-level must be computed for each vertex. Then, a truncated signed distance field (TSDF) is performed for every vertex of the voxel grid present in the optimal neighborhood. After the TSDF computation, the mesh is extracted from the gradient field using marching cubes [29].

For urban building reconstructions, Yi et al. [19] proposed a method that uses raw LIDAR data to create, autonomously, 3D models of urban buildings. Their approach focuses on modeling the main body of the buildings, as a consequence, some details such as windows are ignored. The authors start by detecting the points related to the ground, which will be removed from the cloud. Then, some clusters are generated spatially separated from each other. Every cluster can be regarded as a building if the number of points presented in it is bigger than a defined threshold, and if the height of the cluster is larger than another threshold, defined based on the ground height. Having the buildings separated, the next step consists in slicing the building in a set of blocks piled up, when the upwards direction is obtained by calculating a perpendicular vector to the normal of the facades, the local apexes of a histogram, created from the feature points projected on the upward direction, are extracted and consequently, the different planes of the building are separated. For each block, characterized by having the same shapes of the contours over the upward direction, the authors slice the data in a series of 2D sectional points parallel to the ground. The spectral residual clustering (SRC) algorithm is used to decompose the contour into primitive elements. Over the primitive elements, the constrained fitting will be applied for obtaining the accurate contour. With the contours extracted and the 3D models created, from the blocks with modeling operations, the final 3D model is obtained by applying the union Boolean over each 3D model.

2.2.4 Texturization

To recover the details of the objects, texture, and color must be given to the surfaces obtained in the previous process. This process is called texturization. This step requires the 3D model, some images of the target objects, and their respective poses. Knowing the position and orientation, it is necessary to find the relationship between the surfaces of the model and the pixels of the images. Thus, for each image, a relationship is created between the texture of the surfaces present in the images and the texture of the model's surfaces. Since adjacent surfaces may not be present in the same image, it is necessary to perform a color treatment. Waechter et al. [30] apply the Poisson Editing method, which compares the textures of the faces and if there are differences in lighting, a smooth transition between them will be created.

Zavodny et al. [26] propose a method capable of generating texture for triangulations. For each point in the initial cloud point, it annotated its location and color. For each surface, the authors generate an oriented bounding box which is filled with "texture using a hybrid technique that begins by splatting point onto the texture with a small radius" [26, p. 4]. In each image, all empty pixels are filled with the average value of their neighbors. To save space, just one dimension of the texture is fixed, and based on the bounding box ratio the other dimension is automatically obtained. This approach improves the consistency of the texture generated.

2.3 Robots for Building Construction and Maintenance

Increasingly, inspections and maintenance of different infrastructures are beginning to be carried out by robots, either autonomously or manually controlled. This phenomenon arises from the advancement of technology, allowing the creation of robots with robust locomotion, greater capacity for data acquisition, with more accurate sensors at a lower price. Inspections can be executed using various types of robots, from land to air. In this way, it is possible to cover all kinds of scenarios, from simple infrastructure constructions to the maintenance of places where there has been an accident. Thus, inspections carried out by robots have both safety and economic benefits. With regard to safety, this is ensured by the fact that the operator does not have to go to the inspection area, eliminating potential accidents. Inspections with robots are also cheaper, making it possible to be executed more frequently, which is also a very positive aspect when it comes to safety. In economic terms, the use of robots allows the time of each inspection to be reduced, either because it is not necessary to build structures that allow manual inspection, such as scaffolding, or because in many circumstances it is not necessary to interrupt the normal operation of some actives, such as nuclear power plants or bridges.

As explained by Lattanzi and Miller [31], the biggest obstacle to this area is the diversity of the environment in which the robots have to operate, taking into account that the structures are not uniform and are usually found outdoors. The exposure of the robots to different atmospheric conditions and the possibility of inspecting dirty places or with poor lighting conditions represents another challenge, because it may affect the performance of robots sensors and consequently the robot itself. In this way, there is no vehicle capable of performing all types of tasks, but instead, a large set of robots specified to perform certain tasks.

2.3.1 Types of Robots

There are several models of robots, each of them specialized in performing different tasks. They can be distinguished from each other in many possible ways, such as their locomotion, the type of trajectory they perform, and the type of sensors they have. In inspections, robots are chosen taking into account essentially the target mission. So locomotion, as the ability that the robot has in reaching certain areas of a structure, is an essential characteristic when making that choice. Some critical aspects to take into account are the different formats of the infrastructures, the robustness necessary for the mission, the degree of stability of the movement, and the load present on the

robot. The types of locomotion can be divided into three major areas according to the environment: terrestrial, aquatic, and aerial.

2.3.1.1 Terrestrial

This mechanism is, mainly, used when the situation requires planar movement, representing the best option since robots are more stable and balanced, but can also be used outdoors, where the surfaces are irregular. Normally, this type of robots can have either wheels, continuous tracks, or legs. Yu et al. [32] developed a robot with differential wheels configuration, capable of being at a constant distance from the wall. The robot is also equipped with a camera that is mounted on a device responsible to reduce the vibration, so the images taken from the wall would have less noise. Some authors like Choset [33], based on this mechanism, affix a manipulator on the top of the robot, giving more degrees of freedom to the robot, making it possible to reach areas that otherwise would not be reachable.

Ground-based robots are associated with higher payloads, which means more sensors can be carried by them. They are also stable, due to the fact they are in contact with a surface, and controllable platform for inspection [31]. In comparison with other mechanisms, they are less mobile, having more difficulty reaching certain points. Once they use the floor to move, depending on the inspection scene, it may be required to close the traffic to execute the inspection.

A variation to this type of locomotion is Crawling or Climbing. The big difference is that the robot will no longer have wheels, and will be in direct contact with the surface, allowing it to reach areas of difficult access in infrastructure. Normally these robots are able to walk on surfaces parallel to the floor, as well as perpendicular surfaces. This mechanism has some limitations in terms of the load it can carry, as well as the fact that it has more difficulty in avoiding obstacles. Some examples, as well as their respective uses, are presented by Schmidt and Berns [34]

2.3.1.2 Aquatic

The term unmanned marine vehicle (UMV) comprises two types of water vehicles. The unmanned surface vehicle (USV) travels at the level of the water surface, being boats an example of this type of vehicle. The unmanned underwater vehicles (UUVs) are vehicles that move underwater and normally use low transmission rate modems to communicate with the surface. Since UMVs are robots that move in the water, they become subject to currents or waves, which can remove them from the route. In addition, a large part of the sensors used in robots is not waterproof, which adds another difficulty when using these vehicles.

With regard to inspections, UMVs are mainly used for the inspection of bridges or infrastructures under the water. Murphy et al. describes a mission where three UMVs are used to map the wreckage of a collapsed bridge. One of the UMVs is equipped with an "acoustic camera for subsurface inspection and a three video cameras for viewing above the waterline" [35, p. 84]. The second is normally used for bridge inspections and is equipped with a video camera. Finally,

the third is used for environmental research, it is therefore equipped with a side-scan sonar for mapping debris fields.

2.3.1.3 Aerial

UAVs are vehicles that move through the air, commonly known as drones. There are several models for this type of vehicles, where the two main ones are, those with multiple engines such as the Tricopter, Quadcopter, Hexacopter, and Octocopter characterized by having the ability to move in all directions, and those with fixed-wing, characterized by having a flight time significative superior to those previously mentioned. The vast majority of these vehicles are not designed to carry out inspections, however, due to their great flexibility to be modified and their low prices, UAVs have been used for this purpose.

To analyze the feasibility of performing infrastructure inspections with UAVs, Khan et al. [36] developed a set of tests where a UAV equipped with an automated multispectral scanning system was used. The authors conclude that it brought several benefits mainly in the detection of deterioration of the infrastructures. Another advantage found was the fact that it was not necessary to close the traffic.

To perform inspections on historical buildings, Kaamin et al. [37] used a UAV equipped with a camera capable of quickly mapping different areas. The UAV used starts at ground level and moves vertically up to the top of the building, where it reverses the direction of movement and returns to the floor. This process is repeated until the building is scanned. While performing the trajectory, images and videos are captured. These data are subsequently analyzed, accomplishing the inspection.

Inspections with these vehicles can be performed autonomously or via a remote control used by an operator. Teleoperated control is currently the most used method because it gives the operator more freedom to decide what to inspect. The biggest limitation in this method is the fact that the inspection is carried out by an operator, which makes it always more susceptible to some errors. Autonomous navigation, on the other hand, presents itself as an alternative in which the vehicle navigates by itself based on the data acquired by its sensors. This approach makes mission results much more reliable since robots are not subject to distractions or fatigue. Although these methods bring several advantages, the environment where the inspections are performed presents itself as a great challenge, because a big part of the inspections are done outdoors and it is characterized by presenting several external factors, such as weather conditions, that can influence the drone's behavior as well as compromise the entire mission.

In order to make data from UAVs more reliable, several studies have been done. To improve vehicle stabilization during inspections, Metni and Hamel [38] developed a control system based on computer vision that limits the flight orientation to keep the objects in the range of the camera. The control system also limits the orientation of the UAV to stabilize it over the target.

2.3.2 Different Scenarios

There are different types of scenarios where an inspection can take place, and given the different natures of the places in question, different approaches must be implemented to ensure the highest degree of reliability of the inspection in progress.

2.3.2.1 Bridges

Bridges are very delicate structures because, due to their shape, they may require care on their land side, as well as on their submerged supports. Sophisticated inspections, which may require traffic disruption, are therefore required as well as specialized teams to perform underwater inspections. Given these characteristics, the use of robots to carry out such tasks has advantages.

One of the first approaches used was the snooper truck. This is a vehicle equipped with a long robotic arm capable of reaching the bottom of a bridge. Based on this vehicle, several approaches were developed, all of them used cameras to detect failures. Despite being widely used, these mechanisms have some limitations related to their cost and traffic disruption. Recently, with the development of UAVs, given their flexibility and mobility, it has presented itself as a viable alternative to the inspection of these infrastructures [36].

2.3.2.2 Power Lines

Society increasingly depends on electricity in its normal functioning, and allied to this need, the importance of maintaining the proper functioning of power lines increases, as they are one of the main forms of energy transport. Given the importance associated with these infrastructures (a failure can cause great economic losses and energy failures for consumers), they always need to be in good condition, and it is very important to carry out periodic inspections and maintenance. These inspections are traditionally performed by humans, either through the ground or using helicopters (far more expensive). Since many power lines are found in places with difficult access, inspections from the ground become less efficient and more time-consuming.

Recently, UAVs have been used for this type of task, as they have characteristics that make them very appealing for this type of task, such as their small size and ease of reaching places with difficult access from the ground. Some approaches try to improve the ability of UAVs to perform these types of tasks, making them more reliable. Azevedo et al. [39] developed an algorithm capable of detecting and modulating power lines in real time. Santo et al. [40] develop a method that uses camera images on a UAV to detect power lines. This approach takes advantage of characteristics of lines such as their parallel lines as a way to improve their performance.

2.3.2.3 Tunnels

Since the tunnels are underground infrastructures, the inspectors face very uncomfortable and dangerous environments, given the lack of lighting, the dirtiness of the environment, and the possible

presence of harmful gases for their health. To mitigate this problem, robotics technology presents itself as a necessity.

Despite the need to conduct inspections autonomously, most of the approaches initially developed consisted of fully or partially teleoperated robots, which required the presence of an operator on the site itself. In addition, sometimes the data acquired contains error or is not sufficient, which demand the intervention of the operator, making him move to the place.

With the objective of making the whole process autonomous, from the detection to the fault repression, some robots were developed, such as the robot-aided tunnel inspection and maintenance (RATIM) [41]. This vehicle has a mobile platform where there is a manipulator equipped with repair tools as well as sensors for fault detection.

2.3.2.4 Vertical Structures

In vertical structures, inspections are carried out using platforms such as scaffolding that allow workers to reach such heights. This method becomes time-consuming and expensive. Then it becomes necessary to use robots to solve these problems.

The robot is required to be able to move on the surface, being able to analyze it regardless of its height. The climbing robots are designed exactly to move along vertical surfaces, being characterized by the type of adhesion they create with the same, this aspect is something that must be taken into account because different types of surfaces will require different adhesions [34].

Another type of robot capable of carrying out this type of inspection is the UAV, which has the ability to move in the air which allows them to analyze the surfaces without having to make effective contact with them. These robots are presented as being more flexible and maneuvering, making them a very interesting option.

Chapter 3

Proposed Approach

To solve the problem presented in Section 1.1, an autonomous system capable of assisting human inspectors in the task of updating the data related to the infrastructures must be used. Thus, the goal of this dissertation is to develop a system capable of extracting information related to the state of the infrastructure, namely its geometric structure and the details that its surface presents. For that, a robot equipped with a set of sensors, whose objective will be to navigate around the infrastructure, extracting points to create a cloud of points from the construction environment will be used. The data will then be analyzed and used by meshing and texturing algorithms, building, at the end, a 3D model of the current state of the infrastructure. This process can be repeated several times, making it possible to obtain several 3D models, representative of the various construction phases, throughout the entire process. These models will then be compared, making it possible to find differences between them.

With the presentation of different vehicles made on Section 2.3.1, it can be concluded that UAVs are quite versatile, and for that reason, a UAV quadcopter was used. This type of vehicles, as mentioned in Section 2.3.1.3, are capable of moving through the air, making it easy to analyze a structure with large dimensions. In addition, they present good mobility, since they can move in any direction, making it a good option to analyze surfaces while avoiding obstacles and some irregularities on the surface itself. Another important aspect is the fact that combined with these advantages, UAVs have a significantly lower price than other vehicles designed for the same purposes. Despite this, UAVs have some disadvantages, namely their flight time, which is the time they can handle in the air without having to change their batteries. The limitation in terms of payload weight is also a disadvantage. However, given the evolution of technology, these disadvantages are becoming smaller as both the capacity of your batteries and the capacity to carry a payload, are getting bigger. Taking into account these factors, it is possible to conclude that the advantages of using this vehicle outweigh its disadvantages, having therefore been chosen for this dissertation.

The sensor used to obtain the data from the structure in analysis will be a LIDAR. This sensor, as mentioned in the Section 2.2.1.2, is capable of extracting clouds of points taking into account the environment in which it finds itself. In addition to this sensor, an algorithm capable of creating

a single point cloud from several point sets obtained while the sensor moves will be used.

In this chapter, the work developed will be presented, explaining in detail, all methods developed and used. The chapter will be divided according to the subtasks used to guide the development of the project. It starts by presenting all the work related to acquisition of the data, and then it will explain the process of converting the raw point clouds into three-dimensional representation of the real scene.

3.1 Data Acquisition

The main objective of this dissertation is to create a three-dimensional model of a real infrastructure. For this, it is necessary, in the first instance, to carry out the data collection. To collect the data needed for the reconstruction process, a LIDAR sensor was used. The reconstruction pipeline is capable of operating independently of the transport system where it is inserted, but since the scenarios targeted by the dissertation are characterized by walls with some meters of height (buildings with two or more floors) and sometimes difficult access, it became necessary to use a vehicle capable of operating in these circumstances, so it was decided to use a UAV due to its ease of reaching higher altitudes and since it is a small vehicle.

3.1.1 Hardware

For the realization of this project it was decided to use a commercial UAV instead of building on from scratch. Due to technological advances, the number of manufacturers that develop UAVs increased, making it cheaper to buy one of these vehicles than to build one from scratch.

The UAV was used to perform the flight, however in this dissertation it was also intended to develop a ROS node[1] capable of creating an autonomous mission and later upload it to the vehicle. Thus, it was decided to equip the UAV with an on-board computer. The computer was also used to save the point clouds and images from the sensor. The computer has to be light and small, and needs to have an adapter that allows the exchange of information between the vehicle and the computer. All the on-board devices used, and a schema of their connection is presented on figure 3.1.

3.1.1.1 UAV and Accessories

The UAV used in this dissertation is a Matrice 300 RTK developed by DJI [42]. This vehicle presents itself as a medium-sized UAV (810 × 670 × 430mm) and has a weight of 6.3 kg considering its two TB60 batteries. These batteries allow the UAV to fly for 55 minutes considering no payload is attached. From the moment the weight of the payload increases, the flight time decreases, being reduced to 31 minutes when the payload is maximum.

According to the information provided by the manufacturer, it is possible to derive an equation describing the relationship between the payload weight in kg (PW) and the UAV Maximum flight

[1] http://wiki.ros.org/rosnode

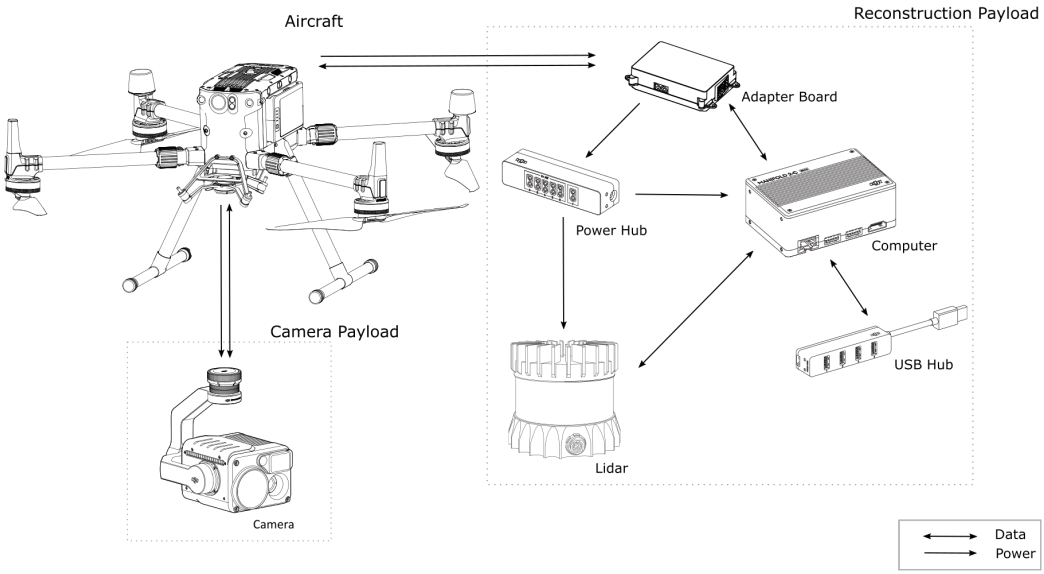


Figure 3.1: On-Board Hardware

time in minutes (MT), equation 3.1. It is important to note that this estimation does not consider the energy consumed by the payload, only his weight. Figure 3.2 presents the curve created from the equation.

$$MT = 2.988 \times PW^2 - 16.778PW + 54.581 \tag{3.1}$$

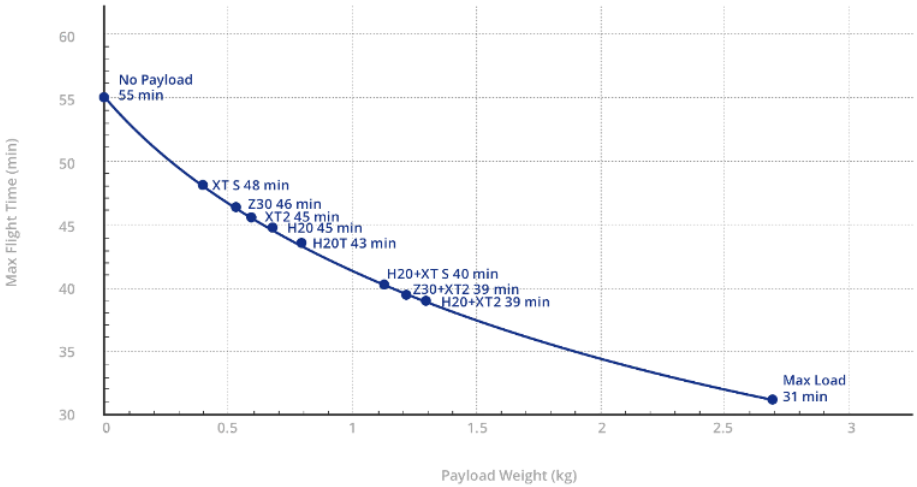


Figure 3.2: Estimated flight time based on the payload weight [2]

[2] https://www.dji.com/pt/matrice-300

The batteries are charged in the DJI Matrice 300 BS60 Intelligent Battery Station. The station has a capacity for 8 TB60 batteries and 4 WB37 batteries, but only 2 TB60 batteries and 1 WB37 can be charged simultaneously. The WB37 batteries are used by the remote control and the global navigation satellite system (GNSS) base station.

The UAV has a GNSS sensor capable of acquiring its geographic coordinates, but since the error of this system is about 2 meters, and the autonomous flight of the UAV depends on the coordinates of different points, this margin of error may be sufficient for compromise the mission and consequently the reconstruction of the models. Thus, when flying, there is a need to connect the vehicle to a base station to make its positioning more accurate. In the project, the D-RTK 2 Mobile Station[3] was used, which, as the name implies, allows to improve the accuracy of the UAV using an real-time kinematic (RTK) system. The GNSS base station, estimates the GNSS system location error and sends corrections that enable centimeter-level positioning data for improved relative accuracy, after the correction the deviations are approximately 1cm (horizontal) and 2cm (vertical).

The UAV can also be equipped with a gimbal and a camera. For this thesis, this equipment was only used on the acquisition of images and videos, which had the ultimate objective of evaluating the quality of the reconstructed models.

3.1.1.2 On-board Computer

An on-board computer was used to execute the nodes responsible for controlling the UAV and recording the point clouds. This computer, as it is placed on the vehicle, needs to have very small dimensions and weight. In view of these restrictions, it was decided to use a Manifold 2-C. This computer weighs approximately 205g, has dimensions of 91 x 61 x 35 mm, an Intel Core i7-8550U processor, 8GB of memory and a 256GB SSD. In terms of power consumption, it consumes between 5 and 60W.

The operating system used is Ubuntu 16.04, as it is a lightweight operating system that is compatible with ROS. For the Ubuntu version used, the compatible ROS version is Kinetic. Those software versions were used, taking into account the fact that the ROS package developed by the manufacturer is designed and tested with this version of the Framework.

3.1.1.3 LIDAR

As explained in Section 2.2.1.2, LIDAR is a technology in growth in the 3D reconstruction area since it has the capability of performing the measurement and creation of point clouds regardless of the lighting conditions of the scene. Having this in mind, for the acquisition of point clouds, a LIDAR sensor has been used, more precisely, an OUSTER OS1[4] with 64 beams, figure 3.3. In terms of characteristics, the sensor presents a very light model, with around 455g.

[3] https://www.dji.com/pt/d-rtk-2
[4] https://ouster.com/products/os1-lidar-sensor/

The sensor, being mechanical, has a tendency to reach high temperatures. To minimize this situation, it must be associated with a heat sink or a magnetic surface. This association will consequently lead to an increase in the total weight of the set to 985g.

In terms of data capture, it has a vertical resolution of 64 beams and can have a horizontal resolution of up to 2048 points. It has a maximum range of 120m and a vertical field of view (FoV) of approximately 45º. Its rotation frequency is configurable and can take the values of 10 or 20 hz. In this way, it is possible to capture a total of 1310720 points per second.

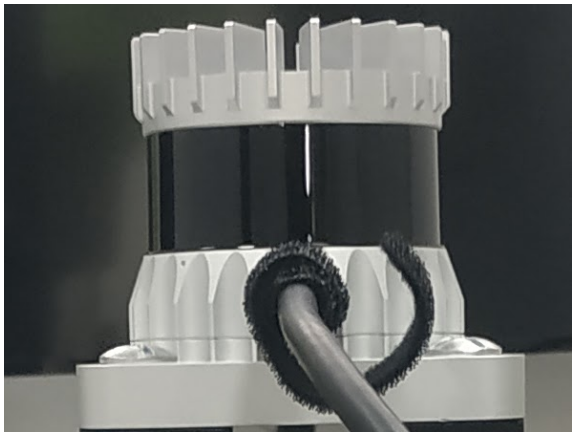
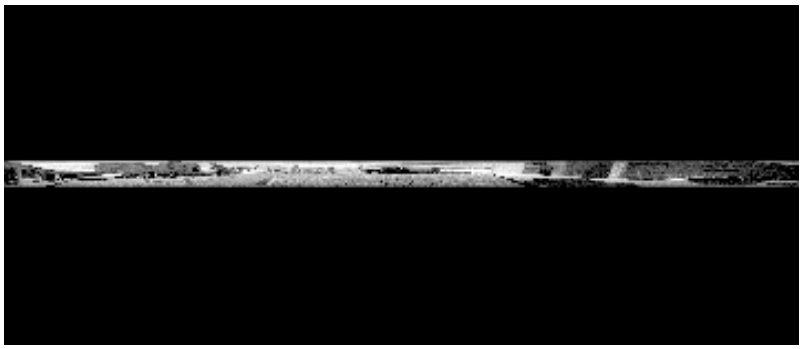


Figure 3.3: Ouster OS1

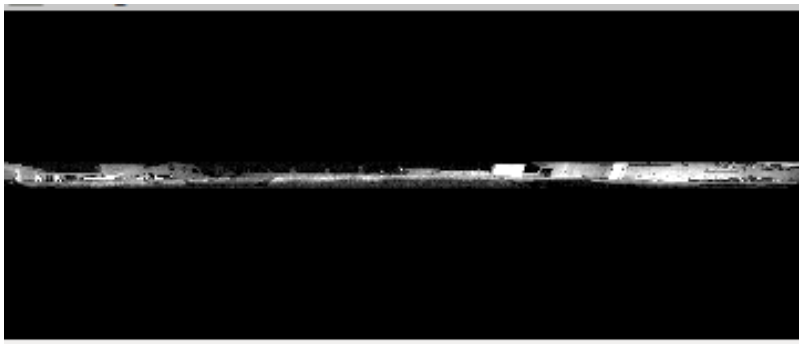
Another aspect that led to the use of this sensor is its ability to be integrated into a project in ROS. The manufacturer provides a ROS driver through which it is possible to establish communication between the sensor and the computer, configure the sensor, read information from the sensor and interact with other ROS packages. For its use, it is necessary to establish a TCP/IP communication between the sensor and a computer and execute the node from the package. This node will cause the sensor to start publishing point clouds on the topic /os_cloud_node/points, through messages in the format "sensor_msgs/PointCloud2". The node has several configurable parameters, such as horizontal resolution and frequency, parameters that allow defining the way messages are sent and the type of information they contain.

Besides all the previously explained features, this particular LIDAR also has the capability of generating ambient images, intensity images, and range images in real time, making them correlated and synchronised. To generate these images, the sensor has an optical system capable of collecting ambient imagery in almost any light conditions. The sensor captures data in the near infrared, giving the images a natural look. This feature is used by the texturization process to give texture to the generated mesh.

In figure 3.4 it is possible to see all the outputs of the LIDAR. The first three pictures correspond to the images sent by the sensor and the final picture to the cloud point also generated by the sensor.

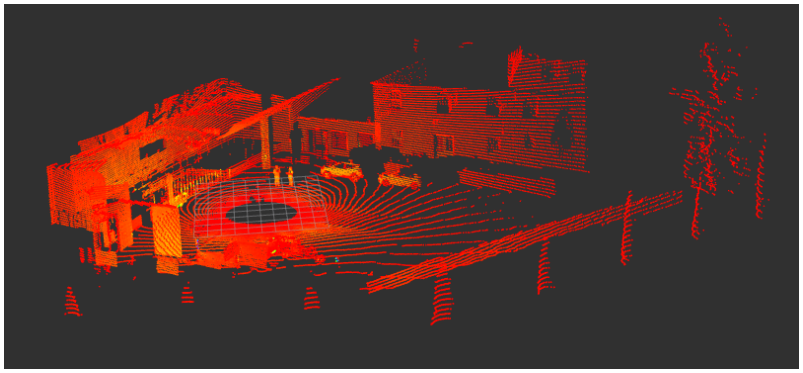


(a) Ambient Image



(b) Intensity Image

(c) Range Image



(d) Point Cloud

Figure 3.4: LIDAR Outputs

3.1.1.4 Hubs and Adapter

Given the reduced dimension of the on-board computer, upon its creation some decisions were taken by the manufacturer, one of them being the reduction of the number of ports where peripherals can be connected. The computer only has 2 USB ports, which is not enough to make all the connection needed, making it necessary to connect to one of them an USB hub, powered by the USB port itself. With this extra hub, the number of USB ports available increased from 2 to 5.

In terms of energy, the principal on-board devices (computer and LIDAR) were powered from the UAV batteries. To make the distribution of energy between the devices possible it was necessary to use a Power Distribution Unit. This unit receives the energy from the UAV batteries and splits it to 5 XT30 Female ports with 15 amperes each.

To establish the interface between the Computer and the vehicle it was necessary to use the adapter onboard software development kit (OSDK) expansion Module. All the components used, as well as all the existing connections (power and logical) are presented in the Tables A.1, A.2 and A.3.

3.1.1.5 Hardware Structure

As the necessary payload to be attached to the UAV corresponds to several components, all with reduced dimensions, a support was designed to accommodate all of them together, thus making the process of placing and removing the payload from the UAV fast, figure 3.5.

(a) Payload (b) UAV

Figure 3.5: UAV and respective Payload

3.1.2 Autonomous Navigation

The UAV is compatible with ROS and once the Framework has been used in other parts of this project, it was also used to carry out the creation of its mission. To this end, a node for creating and uploading the mission to the vehicle was created. In the development of this node some classes, made available by the manufacturer, were used to perform the autonomous control of the aircraft.

The UAV used is capable of carrying out different types of missions such as remote-controlled missions, missions defined through offsets and missions defined by waypoints. This last example

was the type of mission implemented and performed during the acquisition of the data. This type of mission consists of a trajectory defined by a set of waypoints through which the vehicle will have to traverse. Each waypoint is defined with its geographic coordinates (latitude and longitude) and its height. Thus, to create a mission it is necessary to define some parameters, such as the waypoints, the type of trajectory used between waypoints, the UAV orientation, and the UAV's speed.

- Waypoints

  The waypoints are defined using tools such as Google My Maps[5], where it is possible to select points on the world map. After choosing the points, they are exported as a .kml file. This format is very similar to a .xml file, so, in order to extract the needed values, a function based on an xml parser was developed. Since Google My Maps does not have a vertical component, the height of all points had to be estimated and entered manually in the .kml file.

- Path

  For the definition of the trajectory, it is possible to configure the UAV to perform different types of paths, such as straight or curved. For the autonomous missions, and as a way of being able to predict the path that the vehicle will take when traveling, it was decided to configure the UAV to navigate only in straight movements, always using straight lines to travel between waypoints.

- Orientation

  Another very important aspect in the configuration of the mission is to define the orientation that the vehicle must have while traveling. This parameter is very important in the flights performed because the point cloud processing algorithm will only consider the points that are within a given field of view. Thus, it is necessary that the UAV moves around the target infrastructure, always with its front pointing towards the faces of the infrastructure. For this purpose, it was necessary to define a point of interest, towards which the UAV should always be oriented, regardless of the part of the mission in which it finds itself. This point of interest is also defined in My Maps, and is also represented in the .kml file.

  Although the point of interest is defined by a set of geographical coordinates, for the vehicle to consider it as a point of interest, the point has to be defined relatively to the home point (point on the floor where the UAV is before starting its mission), so it must be represented by a pair(x, y), where the component x corresponds to the displacement to geographic North and the component y to the displacement to geographic East. This transformation had to be performed due to limitations imposed by the API provided by the manufacturer. To perform the conversion between the difference of geographic coordinates and a distance in meters, equations 3.2 and 3.3 were used [43]. *La* corresponds to the Latitude and *Lo* to the

[5] https://www.google.com/maps/d/u/0/

Longitude. The numbers associated to this variables define the points used, 1 is used for the Point of Interest and 2 for the home point.

$$\Delta x = (La_1 - La_2) \times (\frac{\pi}{180} \times 6367449 - 559.822 \times \cos(2 \times \frac{La_1 + La_2}{2}) + 1.175 \times \cos(4 \times \frac{La_1 + La_2}{2})) \tag{3.2}$$

$$\Delta y = (Lo_1 - Lo_2) \times \frac{\pi}{180} \times 6367449 \times \cos(\frac{La_1 + La_2}{2}) \tag{3.3}$$

- Velocity

  Regarding the velocity, the UAV presents 2 different types of speeds, the cruise speed and the maximum speed. Cruise speed is the speed that the vehicle displays when there is no interaction by the operator with the remote control. The maximum speed is the maximum value that the sum between the cruise speed and the speed generated by the remote control can have. As an example, a maximum speed of 10 m/s and cruise speed of 5 m/s means that the UAV will move autonomously at 5 m/s, being able to increase its speed up to 10 m/s if the operator so wishes.

It should be noted that at any time during the mission, the operator can suspend or end the mission, as well as force the UAV to move to a specific location. This happens because the commands sent by the remote control always override the autonomous control.

3.2 Data Processing

This Section is focused on creating a 3D model of a building or infrastructure, using as a starting point the raw point cloud acquired from the real building using a LIDAR scanner. To accomplish this goal, a set of different sub tasks have been performed. Those sub tasks are interconnected according to figure 3.6, which is a schema of the system developed.

3.2.1 Odometry Calculation

Like the vast majority of sensors, the LIDAR used produces raw data, which will need some processing and context so that relevant information can be extracted for the continuation of the project. In this case, all the point clouds received contain points that in reality did not exist or that were detected with wrong distances (noise), and in addition, each of the point clouds has its own reference, making it impossible to represent the real environment.

To circumvent the problem of the registration of the point clouds, it was decided to use a simultaneous localization and mapping (SLAM) algorithm. Some different approaches have been

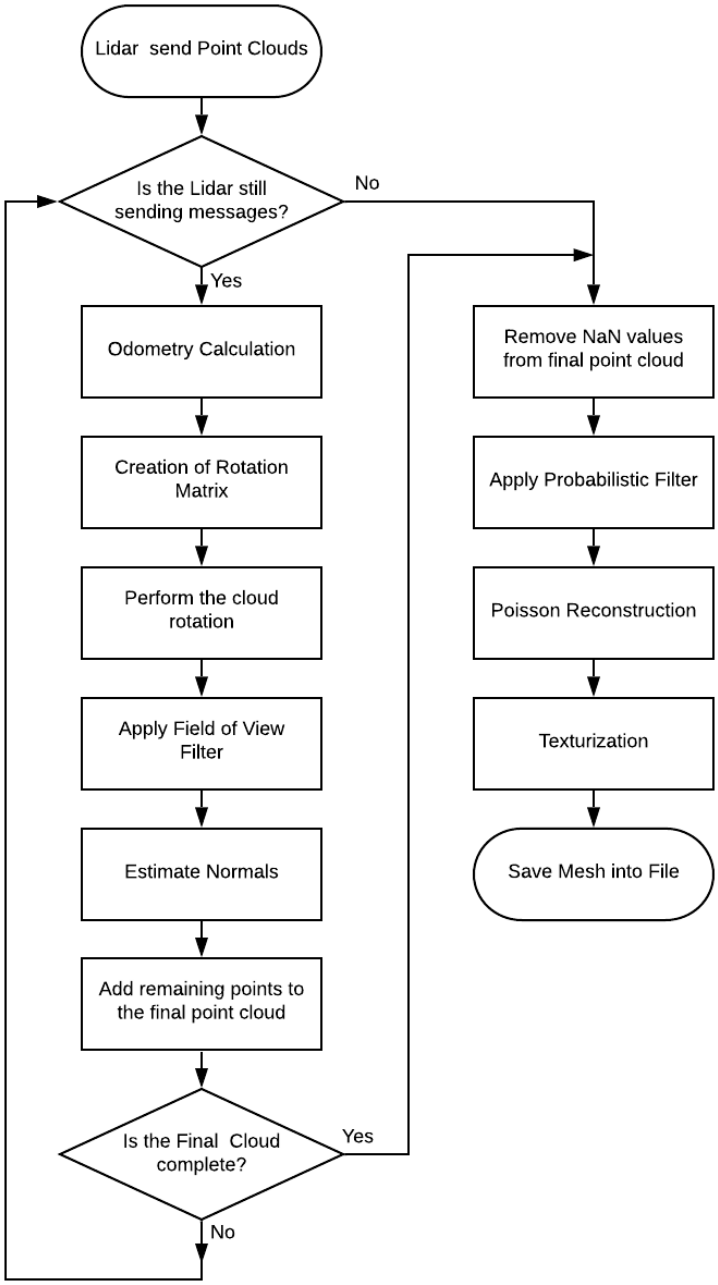


Figure 3.6: 3D Reconstruction Process

tried, such as Google Cartographer [44], LeGO-LOAM [45] and A-LOAM[6]. The first algorithm required a software version that was not compatible with the on-board computer, and was eventually excluded for that reason. The LeGO-LOAM algorithm is "lightweight and ground-optimized LOAM (LeGO-LOAM) for pose estimation of UGVs incomplex environments" [45, p. 4759]. The main characteristic of this method is the fact that it is ground-optimized, which means it uses

[6] https://github.com/HKUST-Aerial-Robotics/A-LOAM

a ground plane to perform segmentation and optimization. Due to this limitation LeGO-LOAM was also excluded. The A-LOAM algorithm, which is an advanced modification of the LOAM algorithm [46], presented very interesting results. This algorithm was developed to interact with LIDAR's data and uses the Eigen and Ceres Solver libraries to make the code structure simpler. This algorithm, in comparison with the LeGO-LOAM, is not limited to the ground, being able to work properly while the LIDAR sensor moves freely in the 3D space. In the context of the project developed, as the data received through the sensor uses the ROS, it is interesting that the SLAM algorithm, to be used, also has this interface. In the case of A-LOAM, this requirement is met.

Some parameters of this algorithm can be defined by the user. Among them, those who stand out are the minimum range and dimensions of the voxels.

- The minimum range corresponds to the minimum distance, from which the algorithm will start to consider the points. This parameter can have a great influence in relatively tight areas, such as the interior of a building, where most of the points captured are located at closer distances. On the other hand, in a larger outdoors environment, the points will already be found over greater distances. If at any time, the distance between all the points presented in the point cloud received by the algorithm and the sensor is smaller than the defined "minimum range", the algorithm will stop receiving information, becoming disoriented and compromising the rest of the project.

- With voxels it is possible to represent a point cloud in a more compact way. These are cubic representations of the environment where the points are found. Its value is obtained taking into account all the points that are inside it. The parameters "dimensions of the voxels" allow to define the size of each cube, making these cubes bigger or smaller depending on the scenario. This approach is used to simplify the required computational capacity, reducing the number of points to be processed and making the algorithm faster.

Once the algorithm is executed, it starts publishing its outputs in different topics, according to the type of information generated. The topic used for the development of this thesis were:

- /laser_cloud_map: a three-dimensional map from the visited scene, which is updated as the algorithm processes the clouds. The map is a point cloud composed of a set of points obtained after the reduction of the input point clouds into voxels. Due to this reduction, the map has a very low density of points which can lead to a 3D model with very little detail, losing important parts of the scenario.

- /velodyne_cloud_registered: input point cloud already transformed according to the scenario. For the reconstruction process this information allowed the algorithm to create very detailed three-dimensional meshes given that the point clouds generated present have almost the same density as the cloud extract from the LIDAR sensor. Due to limitations of the algorithm itself, in this topic, the point clouds published lose their intensity property, making it impossible to perform the texturization.

- /aft_mapped_to_init_high_frec: Once calculated the transformation for the input point cloud, the algorithm generates a "nav_msgs::Odometry" [7] message and publishes it on this topic. The message is constituted by a vector that defines the translation and a quaternion which defines the rotation.

Due to the limitations imposed by the different topics, the A-LOAM algorithm is used to generate a transformation for each point cloud it receives. Figure 3.7 represents the final connection model, where the LIDAR node publishes into topic /os_cloud/points, then A-loam algorithm calculates the transformation for the cloud and publishes it to the topic /aft_mapped_to_init_high_frec. The developed node receives the information presented on those topics and uses it to continue the registration process.

3.2.2 Point Cloud Registration

Having the point clouds being published from the LIDAR in one topic and the information related to their transformation, published by A-LOAM, in another topic, the synchronization between both must be done in order to associate the information. To this end, it was developed a node that integrated features of the package message_filters[8] (a package developed for ROS). In this case, the Policy-Based Synchronizer functionality was used, more specifically the ApproximateTime Policy class. Through this it is possible to choose up to 10 topics, even with different types of messages, and synchronize their messages. In this way, whenever messages are received in the various types, only one callback function will be called and will have as input parameters the data received in all the topics.

Once the association between the odometry value and the respective point cloud is completed, it is necessary to rotate and translate it so that all clouds are aligned according to the real scenario, and are represented in the same reference. To this end, the class "pcl::fromROSMsg"[9] from the point cloud library (PCL) library [47] was used. This class allows the conversion of the messages sent by LIDAR into point cloud specific variables, manipulable by functions of the PCL library. With the cloud in a variable, to perform its transformation it was only necessary to call the function "pcl::transformPointCloud"[10]. This function requires, as parameters, a point cloud and a homogeneous transformation matrix. For the calculation of the matrix, a function was developed. The function receives as input parameters the messages sent by A-loam, which contains a translation vector, defined by the xt, yt and zt values, and a quaternion defined with its w, x, y and z values ($q = (w,x,y,z)$). The translation vector is used to perform a translation on the point cloud and the quaternion is used to rotate the point cloud. Equation 3.4 contains the homogeneous

[7] http://docs.ros.org/en/kinetic/api/nav_msgs/html/msg/Odometry.html
[8] http://wiki.ros.org/message_filters
[9] https://pointclouds.org/documentation/ros_2conversions_8h_source.html
[10] https://pointclouds.org/documentation/group__common.html

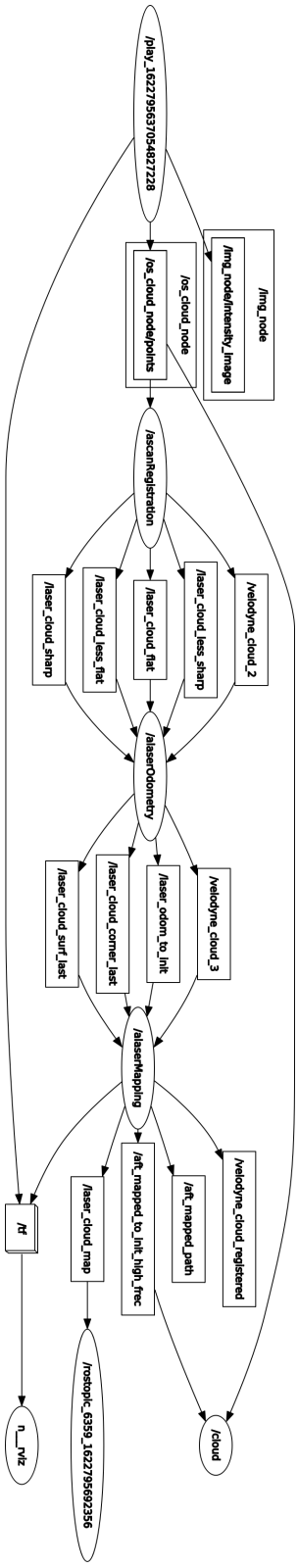


Figure 3.7: ROS nodes connections

transformation matrix used [48].

$$H = \begin{bmatrix} 2\times(w^2+x^2)-1 & 2\times(x\times y-w\times z) & 2\times(x\times z+w\times y) & xt \\ 2\times(x\times y+w\times z) & 2\times(w^2+y^2)-1 & 2\times(y\times z-w\times x) & yt \\ 2\times(x\times z-w\times y) & 2\times(y\times z+w\times x) & 2\times(w^2+z^2)-1 & zt \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.4}$$

3.2.3 Field of View filter

The LIDAR used is capable of generating point clouds with points in all directions, more precisely in 360º. This happens because the beams of light rotate around a central axis.

A positive aspect that makes this capability almost mandatory is the fact that the A-LOAM algorithm needs points to be able to create the map, namely it needs points to detect common features among the clouds in order to estimate the transformation. Since the sensor has a wider viewing angle it is expected to create point clouds containing information about the all scenario around it, improving the final result of the algorithm. Another very positive aspect is the ability to acquire point clouds from an indoor environment without having to rotate the sensor. An example could be the use of a LIDAR inside a warehouse, as it will be able to generate points in all directions of the warehouse from a central point.

Despite presenting some positive features, it also generates a large amount of data. This is a very important aspect as point clouds can contain millions of points, leading to quite large sizes. Another negative aspect is associated with obtaining point clouds of objects from outdoors. In this type of acquisitions, the main objective is to be able to extract only the points associated with the surface of that infrastructure, however, and given that the vast majority of infrastructures are not completely isolated, when creating the point cloud, it will contain points related to other infrastructures of no interest, leading to an increase in the size of the cloud.

To take advantage of the best that can be acquired with the sensor and taking into account limitations in terms of memory and further processing, it was decided to develop a function that, given a point cloud, is responsible for removing all points that are outside a given Field of View. This function is based on the class pcl::FrustumCulling[11], and in order to work properly, it needs to have aligned with the point cloud it receives. Since the point cloud was transformed by the homogeneous transformation, this same transformation is used on the alignment.

With this approach, the SLAM algorithm runs with the complete cloud, maximizing the quality of the registration. Cloud filtering is performed only after the point cloud transformation has been calculated and applied. It should be noted that the filter is always applied from the front of the LIDAR, thus, as an example, a 60º filter corresponds to 30º clockwise and another 30º counterclockwise.

Figure 3.8 represents the applied filter, where the green line represents the region with valid points, and the blue dot represents the LIDAR sensor.

[11] https://pointclouds.org/documentation/classpcl_1_1_frustum_culling.html

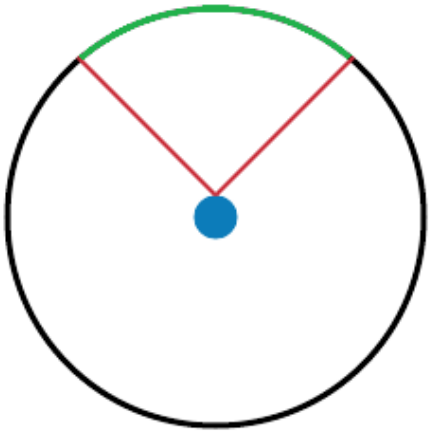


Figure 3.8: Field of View

Since the project was developed with the intention of processing clouds obtained from the exterior, this set of processes allows to create a point cloud oriented, lighter and with only the useful part of the environment.

3.2.4 Normal Estimation

So far, point clouds are made up of points characterized by their X, Y, Z coordinates and their intensity. All of these values came from the sensor and, in the registration operation, the values of X, Y and Z underwent changes. However, for most surface reconstruction algorithms, each point must contain information about the normal vector of the surface to which the point will belong, making it possible to encode the local orientation of surfaces.

To perform the estimation, PCL library provides 2 classes capable of making this estimation. Both classes consider the "problem of determining the normal to a point on the surface is approximated by the problem of estimating the normal of a plane tangent to the surface, which in turn becomes a least-square plane fitting estimation problem" [49, p. 45]. The class used was "pcl::NormalEstimationOMP"[12], which differs from the other one by the fact it can use threads, making the process much quicker. Some arguments of this class can be defined according to the user needs, being them:

- Number of Threads: This value, as the name implies, allows the user to define the number of threads to be used to calculate the normals. The higher the number, the faster the algorithm. It should be noted that the maximum number of threads is limited to the number of threads the computer can create. In the project 8 threads were used;

[12] https://pointclouds.org/documentation/classpcl_1_1_normal_estimation_o_m_p.html

- Point Cloud: The variable that holds the points, for which the algorithm will be executed, estimating their normals. In the project, this variable keeps the cloud received from LIDAR after undergoing orientation;
- Neighborhood radius: Value that defines the size of an imaginary circle that will be defined around each point in order to consider the nearest neighbors for the calculation of normals, the points that are within the circle;
- View Point: It is a point characterized by values in X, Y and Z, and works as a point of view, making all the calculated normals point in their direction. To estimate the coordinates of this point, a cloud centroid was firstly calculated, based on the average of all the cloud's constituent points. The method works well when the cloud has a homogeneous distribution of its points and when they are distributed around the sensor, however, not all clouds will have a 360º Field of View, much less be homogeneous, so this approach ends up showing unsatisfactory results. In a second approach, odometry data is used, more specifically the values of the translation of the cloud. After the orientation process, the central point of the cloud coincides with the value of the translation performed, being this point a good estimate for the Point of View.

Once the parameters were provided, it was still necessary to remove all points with coordinates (0,0,0) from the point cloud because the class used can not estimate normals for these points and ends up returning an error. Following the requirements of the class, the normal vectors were estimated, passing the points of the clouds to be characterized not only by their coordinates and intensity value, but also by their normal vector.

3.2.5 Saving the Final Cloud

After the filtering and registration process for each point cloud is finished, the processed point cloud is added to the final point cloud. This point cloud contains all previously processed point clouds, which means that whenever the processing of a point cloud is finished, all its remaining points are added to the final point cloud. Since this method is iterative the increase in the number of point clouds will lead to a final point cloud with more points and, consequently, with a higher density, which is a very important aspect for the reconstruction of surfaces.

Once the final point cloud contains all the desired points from the scene, it must be saved onto a file so that later can be visualized on a proper software such as MeshLab [50]. For this, a ROS Service was developed, and whenever it is called, the function responsible for saving the cloud to a file is executed.

Due to limitations imposed by ROS, it is impossible to send a message with more than 1GB of information to a topic. Since the size of the final cloud can easily exceed this value it was decided that the cloud must be saved in the same node. Thus, the service callback function is responsible for saving the cloud in a .pcd format file. Since the process of writing large amounts of information in memory is time consuming, this function is executed in a different thread from

the rest of the node, because, otherwise, the node would be blocked in this task, failing to receive the clouds from the LIDAR, ending the SLAM algorithm to exhibit unpredictable behavior.

3.2.6 Statistical Filter

Despite all the processing done to each cloud, individually, after the process of creating the final cloud there are some badly framed points in the general panorama, outliers. These points arise due to small errors resulting from odometry, as they always have a margin of error. These points, when inserted in the reconstruction algorithm, will influence the entire process, leaving the construction of the mesh compromised due to the appearance of surfaces that do not correspond to the scene.

In order to minimize the problem, a statistical filter was applied to the point cloud before it was used by the reconstruction algorithm, "pcl::StatisticalOutlierRemoval"[13]. The class has some configurable arguments such as the number of neighbors to consider and the standard deviation multiplier. The multiplier allows the user to decide how many times the deviation value of each point must be, taking into account the overall value, so that this same point is still considered valid. In the project, it was considered that the neighborhood was made up of 50 points and that the standard deviation of each point could not be greater than the global value (multiplier equal to 1).

This process could have been carried out immediately on the point clouds obtained from the sensor, however, as they were not yet transformed and had a density much lower than the final cloud, it was decided to perform the statistical filtration only at this stage of the process.

3.2.7 Surface Generation

After processing the point clouds, the phase of converting the clouds into meshes, creating surfaces from the points is performed. For this step, a Poisson reconstruction [24] based algorithm was used, more specifically the class "pcl::Poisson"[14] from PCL library. As a Poisson based algorithm, this class requires that all points used have a normal vector, hence the estimation of normals when processing clouds.

During the execution of the algorithm an octree is formed from the analyzed points. The depth of the tree is configurable *apriori*, and the greater its value, the greater the computational capacity required, which consequently leads the model to have a higher number of surfaces and detail. As the number of surfaces increases, the size of the model also increases, requiring more space to store the result. In addition, with higher depths, points become more influential in the reconstruction process. This behavior can leave original smooth surfaces to become rough, because points on the same region of the scenario present small changes between their coordinates (noise from the clouds orientation process, which has been quite reduced, but not completely eliminated). Taking into account these small details, a trade-off between the quality of the final reconstruction, the size of the model and the processing time of the algorithm must be made.

[13] https://pointclouds.org/documentation/classpcl_1_1_statistical_outlier_removal_3_01pcl_1_1_p_c_l_point_cloud2_01_4.html

[14] https://pointclouds.org/documentation/classpcl_1_1_poisson.html

3.2.8 Texturization

The mesh created by the reconstruction algorithm presents itself as a 3D model of the scene, however, despite already containing some characteristics of the real object, the model still presents low detailed surfaces, lacking a fundamental characteristic which is the texture. Without this characteristic present in the model it is harder for humans to recognize parts of the infrastructure that are more subtle, such as window frames, windows, doors, among others.

Texturing, as the name implies, consists of imbuing texture on the model already built, thus managing to create a final model, where details initially discarded by the 3D reconstruction can be recovered, culminating in a final representation closer to the real scene. Peachey defines Texturing as "an effective method of simulating surface detail at relatively low cost" [51, p. 279].

In the work developed, the texturing process consists of associating color to the various regions of the model. Firstly an association between points of the point cloud and the surfaces of the mesh were done, taking into account their respective positions on the global referential. In order to do that, a kd tree was built, creating a search index on the point cloud. Those indexes were then used to associate the points with the meshes regions. Since for the same region of the mesh there are lots of points, to colourize the meshes the average intensity value of the closest 5 points was used. It should be noted that only the information present in LIDAR was used to carry out this process, and taking into account that the intensity provided by the sensor is a value that varies between 0 and 255, the final model will present a color scheme limited to greyscale.

3.2.9 Mesh Alignment

One of the objectives of the dissertation is to be able to detect changes that may arise in infrastructures, and given that data from the same scenario can be collected in different moments, it is necessary to perform the alignment between the data sets, using its common parts as a reference. Once aligned, the data becomes comparable.

In this thesis the alignment was performed through the iterative closest point (ICP) algorithm. The algorithm is able to align different sets of data, minimizing the distance between the two. Considering two point clouds, the algorithm starts by going through all the points present in one of them and calculates, for each one, the shortest distance to the second cloud. Then, based on the values obtained, a transformation is generated and applied to one of the clouds, bringing them closer. The process is repeated until some condition is met.

This alignment algorithm was applied on the generated meshes because their number of vertices are significantly smaller than the number of points in the original point cloud, making the process faster. For that, instead of using the points of point clouds, the algorithm uses the vertices of meshes.

Despite having a smaller set of data to be aligned, the meshes still have a large amount of information, so it is necessary to provide the algorithm with a good initial estimate of the orientation of both models, minimizing the processing time and preventing the method from converging to a

local minimum. To generate this initial estimation the CloudCompare[15] tool was used. It is an open source tool that offers several functions for the manipulation of point clouds and meshes. The specific function used requires 4 common points in each mesh to be selected, then it generates a transformation to one of the meshes so that the points in both models coincide.

3.2.10 Mesh Comparison

In order to make the differences between the already aligned models notorious, it was decided to create a heat map using the distance between the input models. A heat map is a technique that allows the visualization of the magnitude of data through colors. In this section, the heat map corresponds to a point cloud, where its points have colours according to the distance between the models. This heat map is generated by a function from CloudCompare which receives as input data a point cloud and a mesh, and for each point it calculates the distance to the closest surface. Since the objective is to detect differences between two meshes, the most detailed of them is the one converted into a point cloud. This new point cloud should be quite dense in order to contain all the information presented on the mesh, and consequently for the heat map to have information about all regions of the mesh, making it possible to detect small differences present in the models. If the most detailed mesh is dense enough (have a high number of vertices), it is not necessary to generate a point cloud, and instead the vertices of the mesh will be used as input.

Having obtained the distances for all points, it is possible to view that the resulting point cloud contains its points with a color scheme proportional to the calculated distance value. Thus, using this heat map, it is possible to detect inconsistencies between the models, by simply detecting on the map regions with colors different from the standard.

[15] https://www.danielgm.net/cc/

Chapter 4

Results

This chapter presents the results obtained during the process of creating the 3D model presented in Section 3.2. This process, as presented above, is constituted by three phases: data processing, surface reconstruction and texturing. It is also going to present some results of the similarities between meshes. Beyond the 3D model process, this chapter will also present the result of the mission creation node. For the execution of the developed algorithms, a computer with the following characteristics was used:

- Processor: Intel Core i7-8550U
- Memory: 8GB 64 bit, DDR4 2400 MHz
- Operating System: Ubuntu 16.04

The results obtained from the algorithm are point clouds or three-dimensional models, and given the existing limitations for presenting this type of results in the document in question, the results presented consist of images extracted from these models. The evaluation of the results will be made using the comparison between images, because it is not trivial to carry out a quantitative evaluation between the 3D model and the scenario from the dataset. Thus, the images of the three-dimensional models will be accompanied by images of the real scenario, obtained when acquiring their datasets, allowing a qualitative assessment to be executed.

4.1 Datasets

To avoid having to perform missions whenever it is necessary to access the data from the LIDAR, ROS [1] has a package called ROS Bags[1], which allows the user to record in a file all messages that have been sent to the topics, chronologically. In this way, the creation of a bag when acquiring the data allows that later, while analysing the data, the mission can be reproduced in the exact same way as that happened in the real mission, but this time without the need for the hardware.

[1] http://wiki.ros.org/rosbag/

As input for the reconstruction algorithm, two datasets referring to infrastructures were used. Each dataset consists of a ROS bag containing all messages acquired by the UAV and the payload set while performing its mission. As scenarios for the missions, a house (Gondomar - dataset A), figure 4.1, and the main building of Quinta do Seixo (Valença do Douro, Tabuaço - dataset B), figure 4.2, were chosen. Both datasets, besides containing the target infrastructure, are also formed by its surrounding vegetation. Table 4.1 presents more details about the datasets.

(a) Left View (b) Front View (c) Right View

Figure 4.1: House - Dataset A

(a) Left View (b) Front View (c) Right View

Figure 4.2: Quinta do Seixo - Dataset B

Table 4.1: Datasets Characteristics

Dataset	**Number of messages**	**Size**	**Duration**
A	2,518,317	67.5GB	14.39 minutes
B	2,772,787	73.5GB	15.57 minutes

To choose the scenarios, some restrictions have been taken into account. Since the UAV is framed in CE class C3, under subclass A3 it is not possible to fly with uninvolved people present in the area of the flight, and fly within 150 horizontally of unauthorized residential, commercial, industrial, or recreational areas. Both selected places were isolated (without surrounding buildings), and the reduced number of people present there was aware of the flight, removing them from the uninvolved people category. In addition to the characteristics of the scenes, in all flights, the UAV was carefully controlled in order to eliminate any undesired risk.

4.2 Data Processing

In this section the results related to the point cloud processing are going to be presented. The algorithm receives messages from the ROS bags, filters them and produces as output a single point cloud constituted by points aligned according to the scenario. The point cloud will later be used to perform the surface reconstruction and the texturization.

The performed process can be divided in two steps where the first one consists on the alignment of all point clouds received, and the second step is responsible for reducing the amount of points presented on the point cloud, allowing the desired points (points associated with the surfaces of the building) to be used on the reconstruction algorithm, without the influence of the unwanted points.

4.2.1 Registration

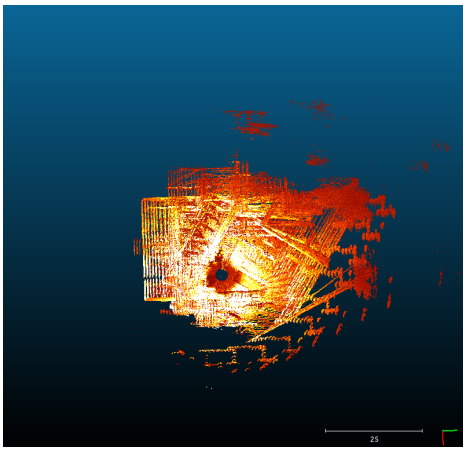
Originally, the point clouds sent by the LIDAR are all in the LIDAR local reference frame, so, if the sensor is moving in any direction, all the clouds will be put on top of each others, resulting on a final point cloud containing is points disoriented, figures 4.3a and 4.3b. With this point cloud it is impossible to find any similarity with the infrastructure scanned, and consequently it is impossible to perform the surface reconstruction.

After the process of registration is performed, all point clouds will be transformed, so that all of them are correctly aligned according to the scene. Figures 4.3c and 4.3d present the result of the registration process over the previously presented images.

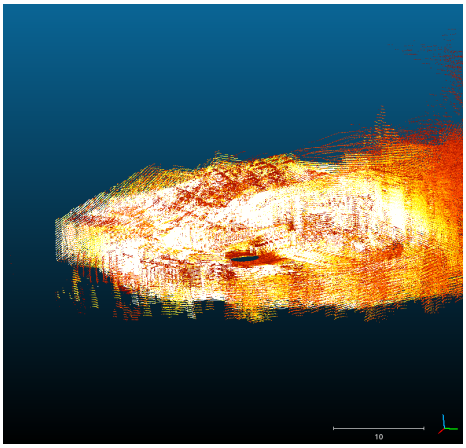
4.2.2 Point Cloud Noise Removal

The LIDAR used is capable of collecting points up to a distance of 120 meters and in all directions (360º). With these characteristics, the point clouds generated will contain far more non-target points than points which actually correspond to the building. With this in mind, a FoV filter was applied in order to remove the unwanted points. The filter has a configurable parameter which makes it possible to control the range of points considered as valid. In figures 4.4 and 4.5 it is possible to understand the difference between the clouds containing points all around the sensor, and the ones which only contain the wanted portion of the scenario.

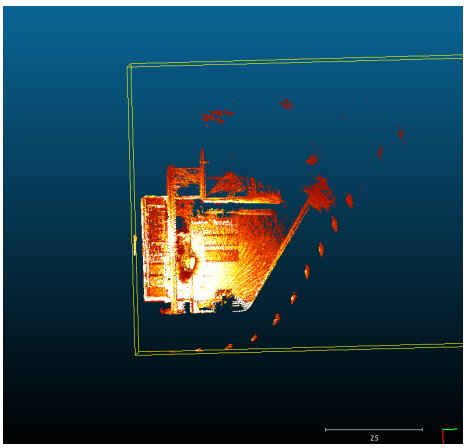
While performing the registration of the point clouds, the algorithm stops receiving the messages from the LIDAR, so, from the same dataset (ROS bag), different computers with different computational capabilities will end up with point clouds with different densities. In order for the algorithm to use all the information presented on the ROS bags, in this specific computer, it is not possible to run the ROS bag at its normal speed, making it mandatory to slow it down. Despite not containing all the information from the ROS bag, the built point clouds have enough density for the reconstruction step. The angle of the field of view is another aspect that can have a big influence on the point cloud size, since it is responsible for reducing the amount of point present in the cloud. Table 4.2 contains the number of points present in each point cloud.



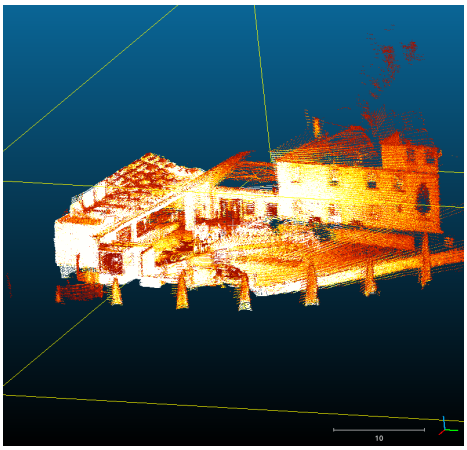
(a) Before Registration - Top View



(b) Before Registration - Front View

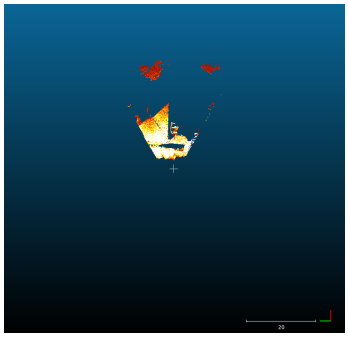


(c) After Registration - Top View

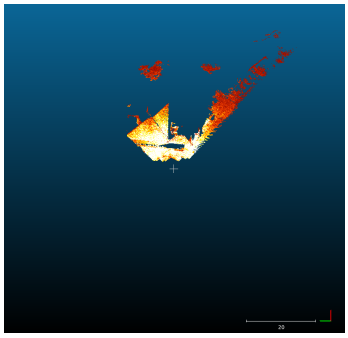


(d) After Registration - Front View

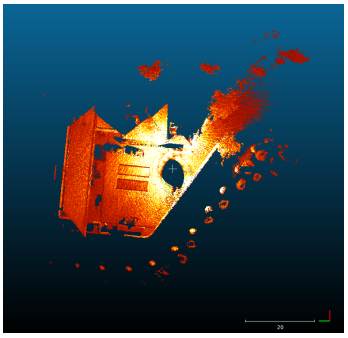
Figure 4.3: Cloud Registration



(a) 60º Field of View



(b) 90º Field of View



(c) 360º Field of View

Figure 4.4: FoV Filter - Top View

4.3 Surface Reconstruction

In this section, the results obtained from the reconstruction of the 3D mesh will be presented. The algorithm receives as input data the point clouds generated using the method previously explained, section 3.2.7.

(a) 60º Field of View (b) 90º Field of View (c) 360º Field of View

Figure 4.5: FoV Filter - Front View

Table 4.2: FoV Point Clouds Constitution

Point Cloud	**Number of Points**
60 FoV	1,283,665
90 FoV	1,880,710
360 FoV	3,878,873

Due to the large size of the point clouds and the time it will take to perform the reconstruction, only the point clouds submitted to a 90º field of view filter were used. This value was also used because with a bigger angle, the point clouds began to contain too many non-building points. Figure 4.6 represents the point clouds obtained on the previous step, using both dataset A and B.

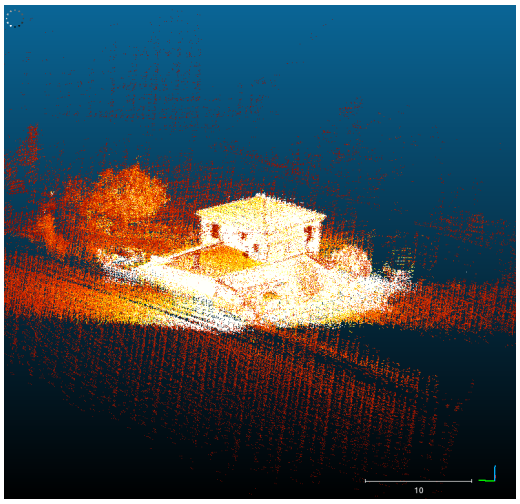
4.3.1 Depth Influence

For Poisson reconstruction, the depth parameter has a big impact on the final 3D model, as well as in the time the algorithm takes to perform the reconstruction. Bigger values for depth will make the result more detailed, containing a higher number of vertices and surfaces. This increase will lead the reconstruction to be more time consuming.

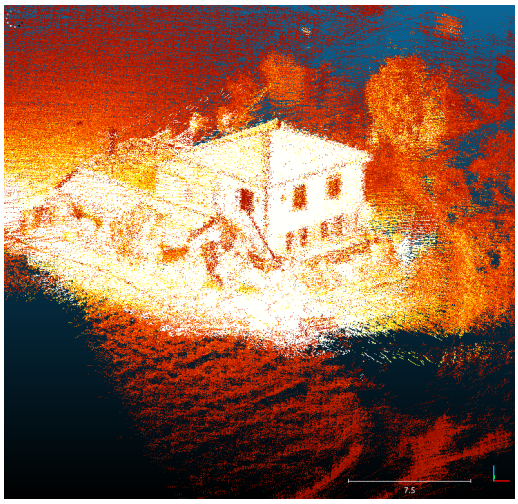
On figures 4.7 and 4.8 it is possible to compare the detail presented in the different meshes based on the depth used to create them. It is important to emphasise that the meshes, from the same scenario, were generated from the same point cloud. Only the depths used for the reconstructions were different.

For both dataset, the influence of the depth in the Poisson Reconstruction algorithm is notorious. With a depth of 7, it is almost impossible to differentiate the dataset because of the low detail presented on the meshes. All the surfaces are completely flat, removing parts of the building such as the windows and doors. With a depth value of 9 the mesh starts to have some detail, namely, it is possible to detect the windows and doors positions as well as the roof. Finally, with a depth of 11, the meshes have enough detail, making it possible to detect subtle details, such as the cars in front the building in figure 4.8, or the pillars in figure 4.7.

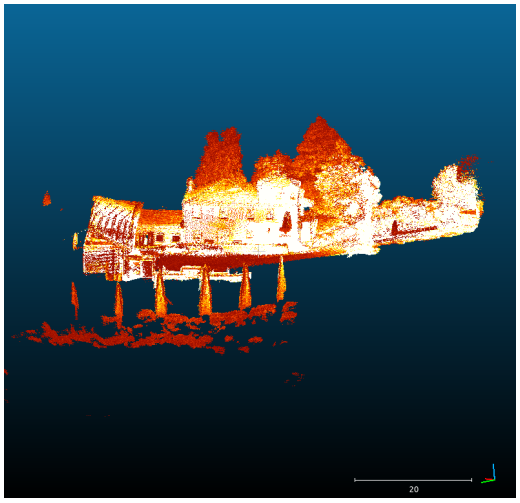
For the datasets used, a depth value bigger than 11 will make the process far more slow. Beyond the time, the generated meshes will not have significant improvements, in fact, with bigger



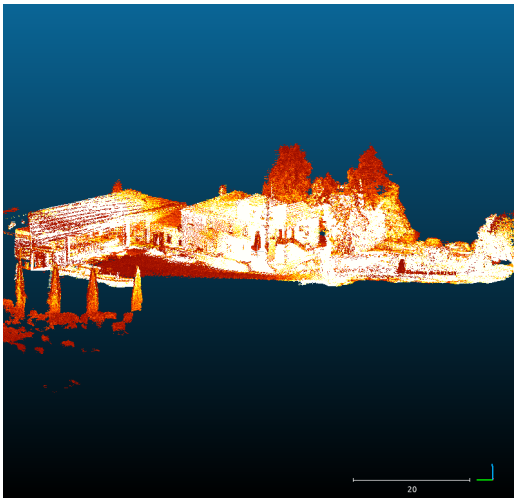
(a) House - Front-Left View



(b) House - Front-Right View

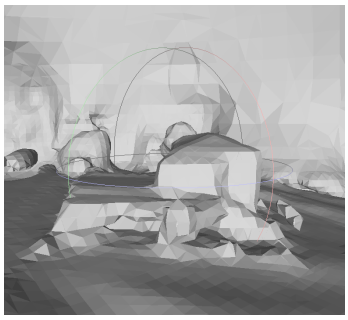


(c) Quinta do Seixo - Front-Left View



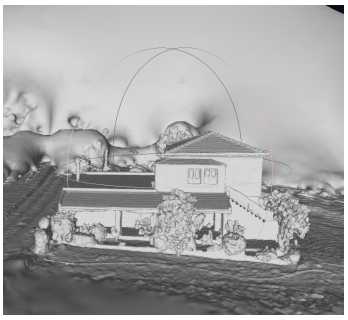
(d) Quinta do Seixo - Front-Right View

Figure 4.6: Point Clouds used on Reconstruction



(a) Depth 7

(b) Depth 9



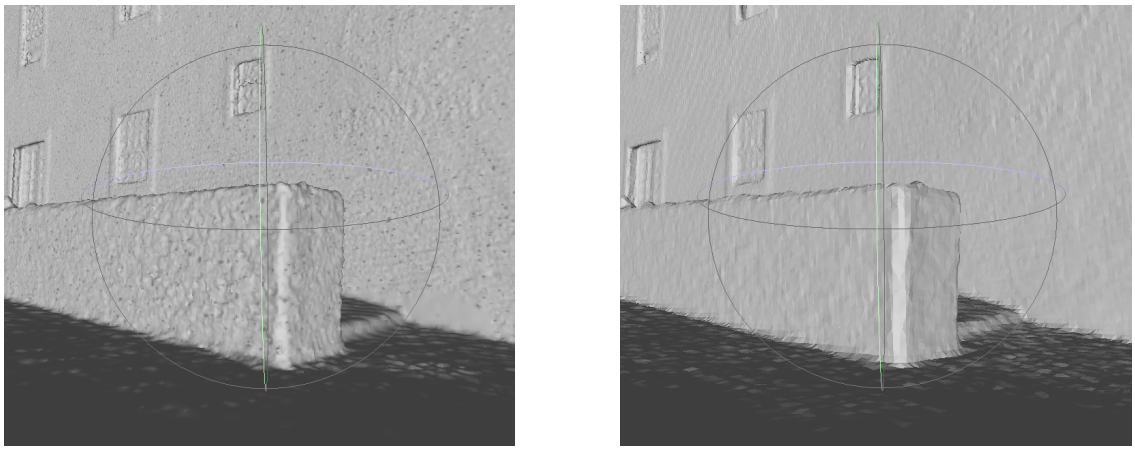
(c) Depth 11

Figure 4.7: Reconstruction Depth - House

depth values surfaces which are flat on reality will appear with some roughness, due to the presence of some noise, figure 4.9 illustrates the phenomenon.

(a) Depth 7 (b) Depth 9 (c) Depth 11

Figure 4.8: Reconstruction Depth - Quinta do Seixo



(a) Depth 12 (b) Depth 11

Figure 4.9: Roughness from High Depths

4.3.2 Statistic Filter Influence

In the Poisson Reconstruction algorithm, as presented in section 3.2.7, the increase in the applied depth makes it so that each constituent point of the cloud will have a greater influence on the result. With this, the cloud becomes more detailed, however, the influence of noise becomes more noticeable. Thus, when applying the reconstruction to a high depth, a filter can be applied to the cloud in order to reduce the number of outliers found in it. In the figure 4.10 it is possible to compare the differences between the result using the filter, and without using it. In the image on the left, due to the lack of the filter, the mesh presents a lot of irregularities on its surfaces. In the image on the right, and once the filter has been applied, it is possible to see that the irregularities have disappeared, leaving the mesh with a more realistic appearance. It is important to notice that the filter did not remove details from the structure, as seen in the roof tiles.

The filter considers as outlier areas of the point cloud where its density is reduced, so it is necessary to be careful in its use as it may eventually remove important parts of the scenario. In the datasets used, as the density of points in the regions associated with the building surfaces is quite high, the filter did not remove important parts of the cloud.

(a) Without Filter (b) With Filter

Figure 4.10: Statistic Filter Impact on Reconstruction

In Table 4.3 the characteristics of both, the obtained meshes and the original point clouds are presented, making it possible to compare the different processing times of the algorithm with the number of vertices and surfaces of the resulting meshes. For the reconstructed meshes the statistical filter was used in order to remove undesired surfaces. While calculating the time values, only the reconstruction process was running in the computer, in order to make the value fair.

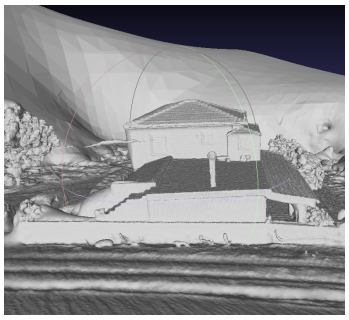
Table 4.3: Generated Meshes characteristics

Dataset	Original Cloud	Depth	Reconstruction Time	Mesh Vertices	Mesh Surfaces
A	25,652,635 points	7	21.081s	11,559	22,942
A	25,652,635 points	9	35.485s	123,724	247,518
A	25,652,635 points	11	179.442s	1,090,146	2,181,306
B	27,841,984 points	7	23.441s	13,967	27,755
B	27,841,984 points	9	40.745s	161,582	322,930
B	27,841,984 points	11	224.518s	1,831,643	3,663,510

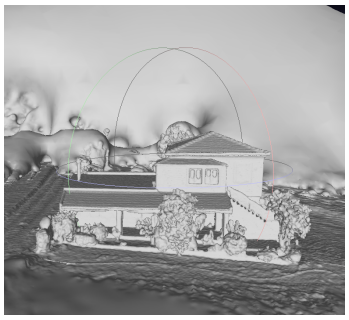
4.4 Texturization

In this section, the meshes extracted from the texturization process are presented. The algorithm on this section uses the intensity values from the point cloud to colorize the mesh acquired on the previous step. The resulting meshes will only have a gray scale scheme of colours, because the intensity values on the point are obtained by the LIDAR and are all integers values from 0 to 255. For the texturization process, only the meshes generated with depth value of 11 and its respective point clouds were used.

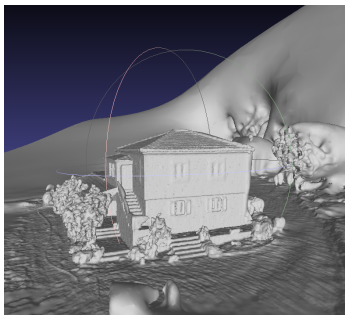
The figure 4.11 shows the result of applying the texture to the mesh from dataset A. It is possible to verify that only the surfaces relative to the house present texture, as they are the only areas where the point cloud and the mesh coincide. With texturing, it is now possible to clearly detect the windows of the building, as well as it is possible to observe the contrast between the darker and lighter areas (zone of separation between the roof and the faces of the house).



(a) No Texture - Left View

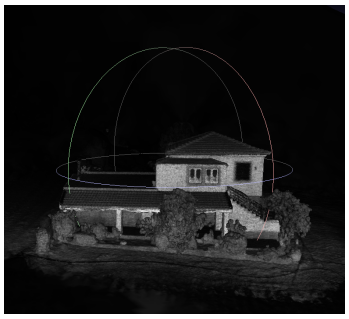


(b) No Texture - Front View

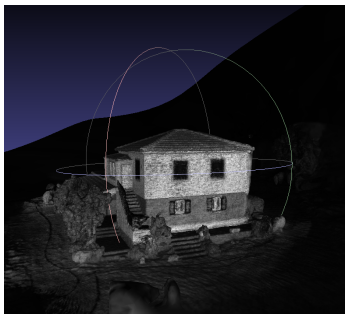


(c) No Texture - Right View

(d) Textured - Left View

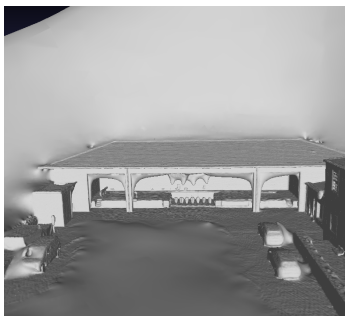


(e) Textured - Front View



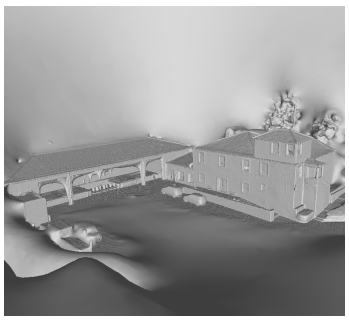
(f) Textured - Right View

Figure 4.11: House Meshes

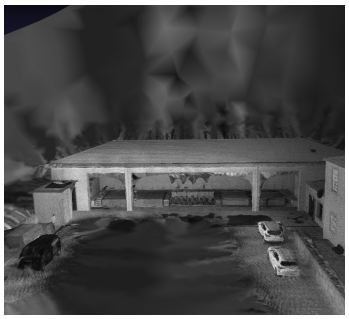


(a) No Texture - Left View

(b) No Texture - Front View



(c) No Texture - Full View



(d) Textured - Left View

(e) Textured - Front View

(f) Textured - Full View

Figure 4.12: Quinta do Seixo Meshes

Similar to what happens in figure 4.11, in figure 4.12 (dataset B) it is possible to detect details that were not present in the original model. A clear example can be seen in the image 4.12e,

where it is possible to see the building's trademark image on the surface of the house, something impossible only with the reconstruction process. In addition to this detail, in the same image, it is also possible to detect other parts of the building more easily, such as the windows and respective contours, doors, roof and patio benches. The mesh region associated with the cars also presents a very significant improvement, as it becomes possible to distinguish its wheels and windows.

With the set of figures presented above it is possible to understand the influence of texture on the meshes. It can be concluded that meshes without texture, despite having some level of detail, can not be compared with the textured ones because they miss important characteristics of the scenario that can only be obtained through texturing. Beyond the big improvement on the mesh, the time associated with the texturization process is significantly smaller than the time required to perform a reconstruction, Table 4.4.

Table 4.4: Reconstruction and Texturization Times

Dataset	Reconstruction Time	Texturization Time
A	179.442s	12.154s
B	224.518s	14.430s

4.5 Final 3D Models

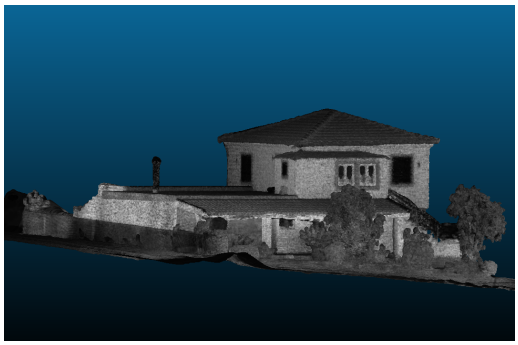
As explained in the beginning of this chapter, all the results must have a context, so that they can be evaluated. In this section, images and meshes will be put side to side, making it possible to compare the pictures extracted from the reconstructed meshes with the images from the real infrastructures. In order to have a comparison with reality as close as possible, the mesh regions that do not correspond to the buildings were removed manually using the CloudCompare [52] tool.

4.5.1 House

Through the figure 4.13 it is possible to make the comparison between the created mesh and the real infrastructure. The images on the left side are both extracted from the same mesh. The angles for capturing the images were chosen so that the region from the mesh was as close as possible to the associated real image.

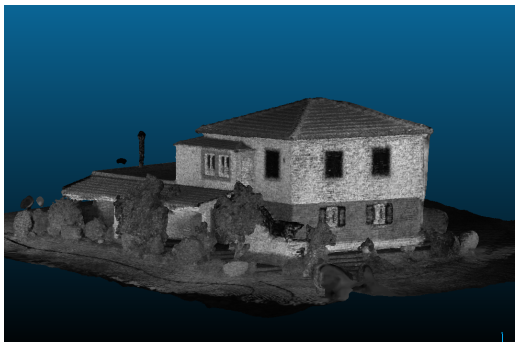
It is possible to confirm that in terms of geometry, the three-dimensional model is very similar to the real infrastructure. The windows and roofs present a lot of detail, being possible through the figure 4.13c to verify that during the mission the shutters of the windows were open. In terms of details, it can be seen that in the 3D model, the chimney, despite being insulated and quite thin, was well rebuilt. The pillars are also well defined, even though it is a lower point density zone. Regarding the plants that are found around the house, as expected, they do not have as much definition (during the acquisition, due to the wind, they ended up moving). As for the texture present in the model, it can be seen that the lighter regions of the model correspond to the areas of

the house whose surfaces were lighter and the darker areas correspond to the roof, stone walls and windows.



(a) Mesh Front View

(b) Building Front View



(c) Mesh Right View

(d) Building Right View

Figure 4.13: Meshes vs Reality - House

4.5.2 Quinta do Seixo building

In this subsection and similarly to the previous one, the results obtained at the end of the three-dimensional model creation process are presented. In figure 4.14, it is possible to compare the result from dataset B with the original building.

Regarding the form that the created model presents, it is possible to conclude that this consists of a structure very similar to the structure of the real infrastructure. The model managed to present a very interesting level of detail as the windows of the building, its roofs, the door and even the cars that are in front of it can be seen with good definition. Similar to what happens in the other dataset, it is possible to observe that the model managed to reconstruct isolated and very thin parts of the scenery, as is the case of the lightning rod existing on the building's roof. Regarding the texture present in the model, what stands out the most is effectively the building's logo found on the front facade of the house. Furthermore, through the colors of the model it is also possible to differentiate regions such as the faces of houses and roofs in the model. In the figure 4.14c the interior of the infrastructure was not very well reconstructed due to the low density of points present in the region (the UAV did not fly inside the building).

(a) Mesh Front View

(b) Building Front View

(c) Mesh Left View

(d) Building Left View

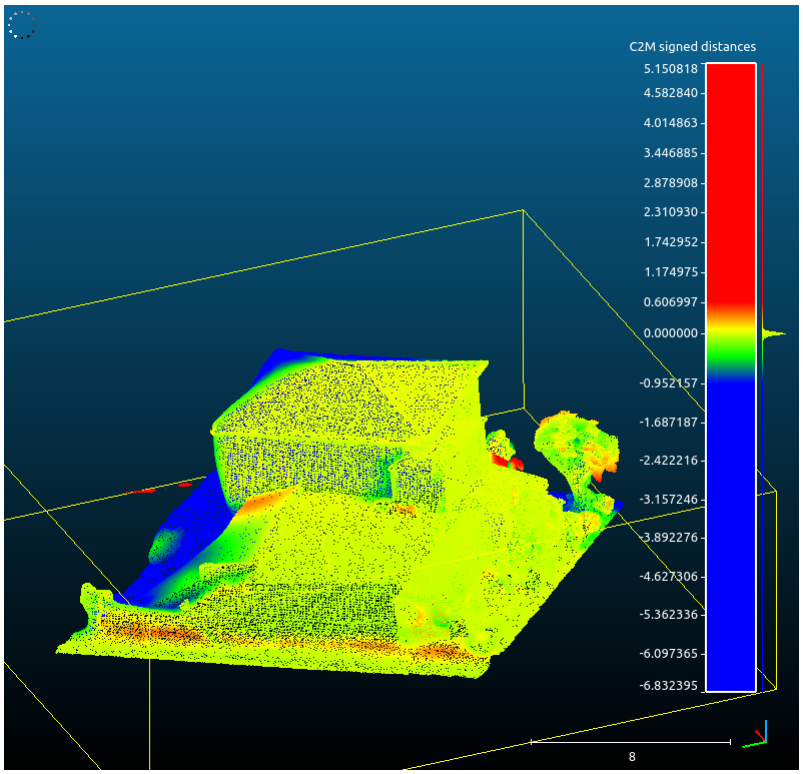
Figure 4.14: Meshes vs Reality - Quinta do Seixo

4.6 3D Model Comparison

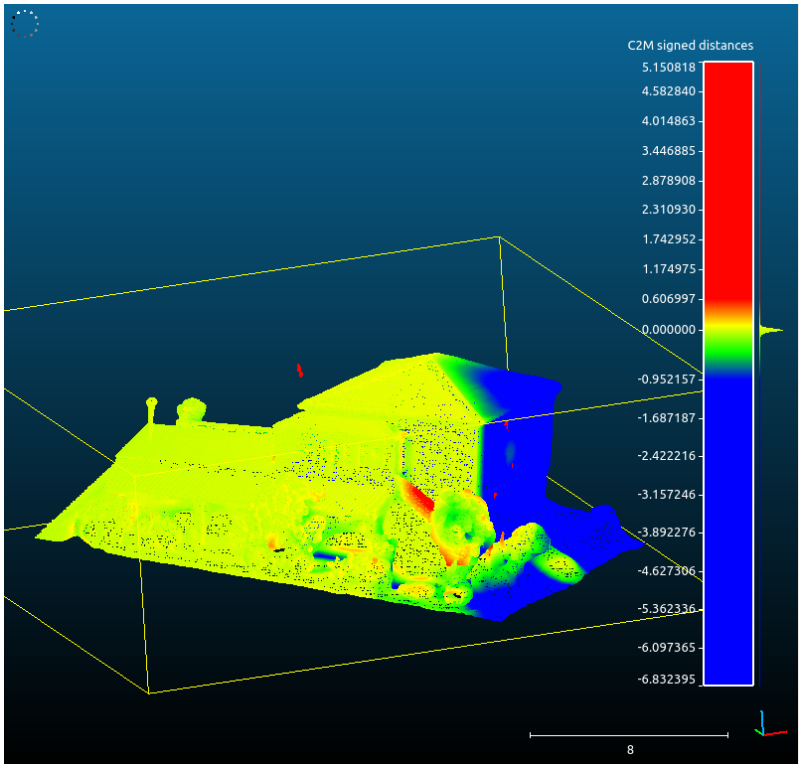
This section contains the results obtained while trying to detect differences between meshes from the same scenario. For this process, information extracted from two flights on the House scenario were used. One of the flights covers 2 faces of the house and the second one covers all the 3 faces. To assess the differences between the models, we chose to use a heat map, where the colors of the map will be directly related to the distances between the models. For the creation of the heat map presented on figure 4.15 the open source software Cloud Compare was used. With this representation it is possible to understand that all the surfaces with the colour yellow can be found in both models, while the surfaces with colors red or blue only exist in one of them. Thus, with this heat map it becomes easy to find differences between the models.

4.7 UAV Mission Creation

As explained in the section 3.1.2, in this project, it was developed a ROS node capable of receiving as input a .kml file and from it creating and uploading a mission to the UAV. The deployment of the mission, more precisely, the control of the vehicle while it carries out the mission, is completely autonomous and uses the classes developed by the manufacturer specifically for this aircraft.



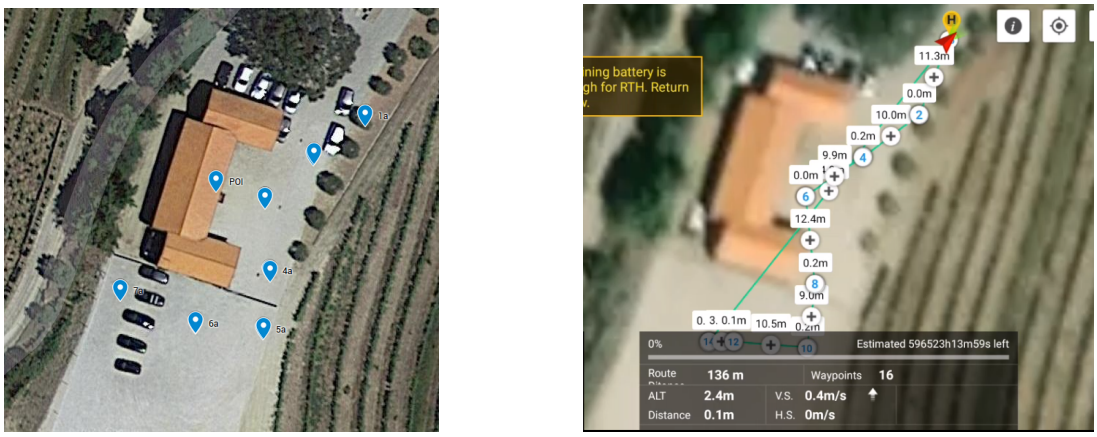
(a) Left View



(b) Right View

Figure 4.15: Meshes Comparison

Regarding the creation of the mission, in the figure 4.16a it is possible to observe the choice of points made in the Google My Maps tool, where each point is defined by its geographic coordinates. The POI point consists of the point chosen as the point of interest, i.e, the point where the drone has its front always pointed. The figure 4.16b shows the mission that was sent to the drone. This image was captured in the controller of the UAV, while it was carrying out the mission. Using the images presented, it is possible to observe that the path taken by the drone is very similar to the path defined in the .kml file.



(a) Points defined on Google My Maps

(b) Mission sent to UAV

Figure 4.16: Mission Trajectory

Chapter 5

Conclusion

The results obtained through the developed system are presented as being very satisfactory. It is possible to reach this conclusion by analyzing the figures 4.11 and 4.12. Making the comparison between the images of the models and the scenarios, it is possible to conclude that the models are very similar, since their geometry were reconstructed in a very identical way to the target infrastructure.

Regarding the various stages of the system, in the point clouds registration stage, the developed method presents very interesting results, as it is possible to obtain, only with the A-LOAM algorithm, the necessary transformations to be applied in the point clouds in order to align them according to the scenario. Although this algorithm presents quite good results, it is limited to using point clouds as input data. This aspect, together with the error present in the data acquisition by LIDAR, means that, despite the correct alignment applied to the point clouds, it will also contain some noise. This noise will have more influence on the final model as the depth of the algorithm used increases.

Regarding the cloud filtering process, the developed filter allows only the points related to the target surfaces to be acquired, thus resulting in a final cloud with much less noise (points related to vegetation and outliers resulting from the alignment). The only condition that has to be taken into account is that for the filter to work properly, the UAV and consequently the LIDAR must always be pointed towards the infrastructure. This filtering had a very positive influence on the final result, as it made the point cloud used lighter, maintaining a high density in the regions associated with the faces of the infrastructures, which consequently allowed the reconstruction algorithm to work faster and get better results.

In the reconstruction part, very interesting results were obtained considering some parameters used. The point clouds obtained were quite dense, which was a crucial aspect for the good performance of this step. With the use of low-density point clouds, and given that the algorithm tends to round straight surfaces, parts of the scenarios such as pillars or even roofs tend to be rounded, losing their similarity with the real infrastructure. An example of this phenomenon can be seen in figure 4.12a. What caused the phenomenon was the fact that the sensor did not capture points on the inside of the building's roof, causing the density of points in that region to be very low,

ending up generating these irregular surfaces. In this specific scenario, what could have been done in order to improve this result would have been to fly the UAV over these regions, but given the dimensions of the vehicle and the proximity it would have to be to the walls and other objects, it was decided to do not perform these risky maneuvers as the cost could be far higher than the benefit.

In the texturing stage, the results obtained were very interesting. With the applied texturing, it was possible to embed in the models several features, lost in the reconstruction stage, as is the case of the design present on the building's facade in the 4.12e image or even other details that end up making the constituents of the meshes more clear.

With the models created, the difference detection step presents itself as a very interesting feature for carrying out inspections. Through the presented results, figure 4.15, it is possible to detect in the models, through the generated heat map, the lack of small components such as the staircase handrail (shown in red in the Right View), the lack of one of the facades of the house (blue in Right View) or the small roof (also shown in blue Left View). Some of the meshes an point clouds generated on this dissertation can be found at: https://drive.inesctec.pt/s/mkmmHrsoAJ5rYeQ

5.1 Limitations of the system

The developed system allows to fulfill the objectives presented in the chapter 1.4, however despite its promising results some restrictions were imposed by the various technologies used.

- The sensor used has a precision between 1.5 and 3cm which makes it impossible to be used in some type of inspection where the small details are very important. Taking into account that the data coming from the sensor is directly applied in the A-loam algorithm, this one, despite being able to estimate the necessary transformation to align the point clouds, will end up generating some noise. As a way of trying to minimize the effect of this noise, several filters were applied throughout the process.

- As the sensor used is capable of generating point clouds at a very high frequency, the accumulation of the various point clouds received by the algorithm will make the point cloud used for the reconstruction contain an extremely high number of points. To manipulate the point clouds, classes from the PCL library were used. Despite the good functioning of the library for small point clouds, in the case of large clouds, the same has not happened. The library's problem for manipulating the granting clouds arises from the fact that the classes use a 32-bit variable to store the various points of the cloud. Since this variable has a finite size, it means that the number of points constituting the final point cloud also has to be. From the results obtained, a cloud with the number of points close to the limit will present a size of approximately 10GB (file in ASCII format). As a way to solve this problem, the node responsible for receiving the point clouds from LIDAR is able to discard a set of clouds, this value being defined by the user.

- Since the intensity used by texturing is obtained by LIDAR, and given the limitations of the spectrum of values present in it (0-255), the final models can only be represented in a gray scale, making some details only distinguishable through the color to be lost.

- Regarding the system used to detect differences, this also has some limitations, namely the fact that it can only detect differences between the built models, and only consider the surfaces and not the texture present on them. If the meshes used do not have enough definition, it will be impossible to know if any changes have occurred in small details, as could be the case of a small crack in a wall. Despite these limitations, and since the reconstruction of the meshes manages to preserve the general structure of buildings, it is possible to conclude that more significant changes in infrastructure, such as the fall of a roof, or even the evolution of a construction, can be detected by the developed system.

5.2 Comparison with Photogrammetry

As previously mentioned, in the area of three-dimensional reconstruction, a widely explored technique is photogrammetry. For this technique, cameras are used, in order to enable the capture of data related to the same scenario, but with different angles. One of the aspects that must be taken into account when using photogrammetry is the synchronization between the images obtained. This aspect is very important, because the scenario should remain the same in all images in order to be able to relate features. In the case of rebuilding infrastructures, this aspect is not very problematic as infrastructures are static objects.

Photogrammetry has some limitations that can be overcome by the proposed approach, namely, the inability to acquire data under adverse lighting conditions and the impossibility of performing reconstructions from textureless surfaces. This last limitation is very relevant because, through photogrammetry, it is complicated to reconstruct a flat surface since it does not have enough features, making it difficult to relate the different images acquired. The main advantages of photogrammetry over the 3D scanners, and consequently the developed system, is the price of the equipment, since cameras are usually less expensive than 3D sensors like LIDARs. Given these advantages and disadvantages of the system, the use of one or the other technique should be carefully analyzed, so that with the characteristics that each one presents, the user is able to decide which one will be more advantageous.

5.3 Future Work

The developed system is modular, which means that the different stages of the entire process are well divided, making it possible to perform changes in some of the parts and the entire process will continue to work properly. Therefore, future work should sensibly focus on trying to overcome the existing limitations of the system and presented in section 5.1.

- In order to reduce noise from the sensor and the SLAM algorithm, other sensors and algorithms should be tested. For sensors, price is a parameter that affects the quality of the data obtained, and the use of a sensor with a higher resolution and precision will make the noise obtained much lower. Regarding the SLAM algorithm, other variations should be tested, namely variations whose input data are not limited to point clouds. Since the UAV has sensors that were not explored, like an IMU, those sensors can be used to improve the SLAM algorithm, helping with trajectory estimation.

- One possibility to overcome cloud size limitation is to change the variable type on the library code. This approach is not the best one, since every user will have to perform this alteration in order to run the programs. Taking this into account, other libraries should be explored. These new libraries should also be able to reduce the processing and reconstruction time of point clouds, so that greater depths can be used.

- Regarding the texture of the 3D model, images of the scenes must be acquired, using the cameras from the UAV, alongside with the point clouds. Having both types of information it will be possible to associate an RGB value to each point from the point clouds. To perform this association, a calibration process between the LIDAR and the camera must be done. The change in the points information will allow the final 3D model to contain colors from the visible light spectrum.

- To improve the detection of differences between models, other techniques as well as different 3D manipulation programs should also be explored.

Appendix A

Hardware used and Connections

Table A.1: Components

Component	Model	Qt	Type	Weigth[Kg]
Aircraft	Matrice 300 RTK	1	Onboard	3.6
Batteries	TB60	2	Onboard	1.35
Multi-sensor payload	Zenmuse H20T	1	Onboard	0.83
Onboard computer	Manifold 2-C	1	Onboard	0.205
Lidar	OS1-64	1	Onboard	0.455
Power distribution Unit	MA1P6A	1	Onboard	-
USB 3.0 Hub	MA1H4A	1	Onboard	-
Computer adapter board	-	1	Onboard	0.096
Controller	Smart Controller PJ001	1	Offboard	0.630
GNSS mobile station	D-RTK 2	1	Offboard	2.4
Battery station	BS60	1	Offboard	8.4

Table A.2: Power Connections

Component	Port	Component	Port	Cable
H20T	Gimbal connector	M300	PSDK.1	-
M2C	PWR.1	PDU	PWR.2	XT30 Female to XT30 Male
PDU	PWR.8	CAB	PWR	XT60 Female to XT30 Male
USBH	USB-A cable	M2C	USB-A	USB-A Male to USB-A Female
CAB	USB-C cable	M300	OSDK.1	-
OS1	OPC	PDU	PWR.3	OPC2 to XT30 Male

Table A.3: Logical Connections

Component	**Port**	**Component**	**Port**	**Cable**
H20T	Gimbal connector	M300	PSDK.1	-
USBH	USB.4	CAB	USB	USB-A to USB-A
USBH	USB.3	CAB	UART	USB-A to TTL
USBH	USB-A cable	M2C	USB.2	-
CAB	USB-C cable	M300	OSDK	-
OS1	OPC	M2C	ETH	OPC to RJ45 Male

References

[1] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Ng. "ROS: an open-source Robot Operating System". volume 3, 01 2009.

[2] Z. M. Cinar, A. A. Nuhu, Q. Zeeshan, and O. Korhan. "Digital Twins for Industry 4.0: A Review". In Fethi Calisir and Orhan Korhan, editors, *Industrial Engineering in the Digital Disruption Era*, pages 193–203, Cham, 2020. Springer International Publishing.

[3] C. Boje, A. Guerriero, S. Kubicki, and Y. Rezgui. "Towards a semantic Construction Digital Twin: Directions for future research", jun 2020. doi:10.1016/j.autcon.2020.103179.

[4] Q. Qi and F. Tao. "Digital Twin and Big Data Towards Smart Manufacturing and Industry 4.0: 360 Degree Comparison". *IEEE Access*, 6:3585–3593, 2018. doi:10.1109/ACCESS.2018.2793265.

[5] F. Tao, H. Zhang, A. Liu, and A. Y. C. Nee. "Digital Twin in Industry: State-of-the-Art". *IEEE Transactions on Industrial Informatics*, 15(4):2405–2415, 2019. doi:10.1109/TII.2018.2873186.

[6] Autodesk Inc. "What are the benefits of BIM?". autodesk.com. https://www.autodesk.com/solutions/bim/benefits-of-bim (Accessed on Jan. 9, 2021).

[7] R. Khilar, S. Chitrakala, and S. SelvamParvathy. "3D image reconstruction: Techniques, applications and challenges". In *2013 International Conference on Optical Imaging Sensor and Security (ICOSS)*, pages 1–6, 2013. doi:10.1109/ICOISS.2013.6678395.

[8] J. I. Ebert. "Chapter 3 - Photogrammetry, Photointerpretation, and Digital Imaging and Mapping in Environmental Forensics". In Brian L. Murphy and Robert D. Morrison, editors, *Introduction to Environmental Forensics (Third Edition)*, pages 39–64. 2015. doi:https://doi.org/10.1016/B978-0-12-404696-2.00003-5.

[9] A. Tsai, J. W. Fisher, C. Wible, W. M. Wells, J. Kim, and A. S. Willsky. "Analysis of Functional MRI Data Using Mutual Information". In C. Taylor and A. Colchester, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI'99*, pages 473–480, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg.

[10] P. N. B. Do and Q. C. Nguyen. "A Review of Stereo-Photogrammetry Method for 3-D Reconstruction in Computer Vision". In *2019 19th International Symposium on Communications and Information Technologies (ISCIT)*, pages 138–143, 2019. doi:10.1109/ISCIT.2019.8905144.

[11] S. Sah and N. Jotwani. "Stereo Matching using Multi-resolution Images on CUDA". *International Journal of Computer Applications*, 56:47–55, 10 2012. doi:10.5120/8947-3119.

[12] A. Geiger, M. Roser, and R. Urtasun. "Efficient Large-Scale Stereo Matching". In Ron Kimmel, Reinhard Klette, and Akihiro Sugimoto, editors, *Computer Vision – ACCV 2010*, pages 25–38, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[13] R. Kalarot and J. Morris. "Comparison of FPGA and GPU implementations of real-time stereo vision". In *IEEE Computer Vision and Pattern Recognition*, pages 9–15, 2010. doi:10.1109/CVPRW.2010.5543743.

[14] S. Forstmann, Y. Kanou, J. Ohya, S. Thuering, and A. Schmitt. "Real-Time Stereo by using Dynamic Programming". In *2004 Conference on Computer Vision and Pattern Recognition Workshop*, page 29, 2004. doi:10.1109/CVPR.2004.428.

[15] Y. Zhao and G. Taubin. "Real-time stereo on GPGPU using progressive multi-resolution adaptive windows". *Image and Vision Computing*, 29(6):420–432, may 2011. doi:10.1016/j.imavis.2011.01.007.

[16] A. Ivanavičius, H. Simonavičius, J. Gelšvartas, A. Lauraitis, R. Maskeliunas, P. Cimmperman, and P. Serafinavičius. "Real-time CUDA-based stereo matching using Cyclops2 algorithm". *EURASIP Journal on Image and Video Processing*, 2018(1):12, 2018. doi:10.1186/s13640-018-0253-2.

[17] J. Masson. *"Desenvolvimento de um algoritmo para a reconstrução 3D a partir de imagens RGB"*. Ph.d. dissertation, UFSC, 2018.

[18] M. Bici, F. Gherardini, F. Campana, and F. Leali. "A preliminary approach on point cloud reconstruction of bronze statues through oriented photogrammetry: the "Principe Ellenistico" case". *IOP Conference Series: Materials Science and Engineering*, 949, 2020. doi:10.1088/1757-899X/949/1/012117.

[19] C. Yi, Y. Zhang, Q. Wu, Y. Xu, O. Remil, M. Wei, and J. Wang. "Urban building reconstruction from raw LiDAR point data". *CAD Computer Aided Design*, 93:1–14, dec 2017. doi:10.1016/j.cad.2017.07.005.

[20] L. Zhang and S. Singh. "LOAM: Lidar Odometry and Mapping in Real-time". 07 2014. doi:10.15607/RSS.2014.X.007.

[21] Z. Li, P. C. Gogia, and M. Kaess. "Dense Surface Reconstruction from Monocular Vision and LiDAR". *2019 International Conference on Robotics and Automation (ICRA)*, 2019. doi:10.1109/ICRA.2019.8793729.

[22] B. Zhang, B. Xiang, and L. Zhang. "Parameter-free outlier removal of 3D point clouds with large-scale noises". *2017 17th International Symposium on Communications and Information Technologies (ISCIT)*, 2017. doi:10.1109/ISCIT.2017.8261207.

[23] A. P. Bustos and T. Chin. "Guaranteed outlier removal for point cloud registration with correspondences". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(12), dec 2018. doi:10.1109/TPAMI.2017.2773482.

[24] M. Kazhdan, M. Bolitho, and H. Hoppe. "Poisson Surface Reconstruction". In *Proceedings of the Fourth Eurographics Symposium on Geometry Processing*, SGP '06, pages 61–70, Goslar, DEU, 2006. Eurographics Association.

[25] N. Wongwaen, S. Tiendee, and C. Sinthanayothin. "Method of 3D mesh reconstruction from point cloud using elementary vector and geometry analysis". *Proceedings - ICIDT 2012, 8th International Conference on Information Science and Digital Content Technology*, 1, jun 2012.

[26] A. Zavodny, P. J. Flynn, and X. Chen. "Textured mesh generation of extracted regions from urban range-scanned LIDAR data". pages 1–6, 2011. doi:10.1109/ICME.2011.6012230.

[27] L. Roldão, R. de Charette, and A. Verroust-Blondet. "3D Surface Reconstruction from Voxel-based Lidar Data". *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, 2019. doi:10.1109/ITSC.2019.8916881.

[28] L. Roldão, R. de Charette, and A. Verroust-Blondet. "A Statistical Update of Grid Representations from Range Sensors". Technical report. arXiv:1807.08483v2.

[29] W. Lorensen and H. Cline. "Marching Cubes: A High Resolution 3D Surface Construction Algorithm". *ACM SIGGRAPH Computer Graphics*, 21:163–, 08 1987. doi:10.1145/37401.37422.

[30] M. Waechter, N. Moehrle, and M. Goesele. "Let There Be Color! Large-Scale Texturing of 3D Reconstructions". In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 836–850, Cham, 2014. Springer International Publishing.

[31] D. Lattanzi and G. Miller. "Review of robotic infrastructure inspection systems". *Journal of Infrastructure Systems*, 23(3), sep 2017. doi:10.1061/(ASCE)IS.1943-555X.0000353.

[32] S. Yu, J. Jang, and C. Han. "Auto inspection system using a mobile robot for detecting concrete cracks in a tunnel". *Automation in Construction*, 16(3):255–261, may 2007. doi:10.1016/j.autcon.2006.05.003.

[33] H. Choset. *"Bridge inspection with serpentine robots"*. Transportation Research Board, IDEA Program, 2002.

[34] D. Schmidt and K. Berns. "Climbing robots for maintenance and inspections of vertical structures - A survey of design aspects and technologies". *Robotics and Autonomous Systems*, 61(12):1288–1305, dec 2013. doi:10.1016/j.robot.2013.09.002.

[35] R. R. Murphy, E. Steimle, M. Hall, M. Lindemuth, D. Trejo, S. Hurlebaus, Z. Medina-Cetina, and D. Slocum. "Robot-Assisted Bridge Inspection". *Journal of Intelligent & Robotic Systems*, 64(1):77–95, 2011. doi:10.1007/s10846-010-9514-8.

[36] F. Khan, A. Ellenberg, M. Mazzotti, A. Kontsos, F. Moon, A. Pradhan, and I. Bartoli. "Investigation on Bridge Assessment Using Unmanned Aerial Systems". *Structures Congress 2015. Proceedings*, 2015.

[37] M. Kaamin, N. A. Idris, S. M. Bukari, Z. Ali, N. Samion, and M. A. Ahmad. "Visual inspection of historical buildings using micro UAV". *MATEC Web of Conferences*, 103, 2017. doi:10.1051/matecconf/201710307003.

[38] N. Metni and T. Hamel. "A UAV for bridge inspection: Visual servoing control law with orientation limits". *Automation in Construction*, 17(1):3–10, nov 2007. doi:10.1016/j.autcon.2006.12.010.

[39] F. Azevedo, A. Dias, J. Almeida, A. Oliveira, A. Ferreira, T. Santos, A. Martins, and E. Silva. "LiDAR-Based Real-Time Detection and Modeling of Power Lines for Unmanned Aerial Vehicles". *Sensors*, 19(8), 2019. doi:10.3390/s19081812.

[40] T. Santos, M. Moreira, J. Almeida, A. Dias, A. Martins, J. Dinis, J. Formiga, and E. Silva. "PLineD: Vision-based power lines detection for Unmanned Aerial Vehicles". In *2017 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, pages 253–259, 2017. doi:10.1109/ICARSC.2017.7964084.

[41] J. G. Victores, S. Martínez, A. Jardón, and C. Balaguer. "Robot-aided tunnel inspection and maintenance system by vision and proximity sensor integration". *Automation in Construction*, 20(5):629–636, 2011. doi:https://doi.org/10.1016/j.autcon.2010.12.005.

[42] DJI. "M300_RTK_User_Manual_EN". dl.djicdn.com. https://dl.djicdn.com/downloads/matrice-300/20200507/M300_RTK_User_Manual_EN.pdf (Accessed on Jun. 15, 2021).

[43] JJones. "How to convert latitude or longitude to meters?". stackoverflow.com. https://stackoverflow.com/questions/639695/how-to-convert-latitude-or-longitude-to-meters (Accessed on May. 14, 2021).

[44] W. Hess, D. Kohler, H. Rapp, and D. Andor. "Real-Time Loop Closure in 2D LIDAR SLAM". In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1271–1278, 2016.

[45] T. Shan and B. Englot. "LeGO-LOAM: Lightweight and Ground-Optimized Lidar Odometry and Mapping on Variable Terrain". In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4758–4765. IEEE, 2018.

[46] J. Zhang and S. Singh. "LOAM: Lidar Odometry and Mapping in Real-time". In *Robotics: Science and Systems*, 2014.

[47] R. B. Rusu and S. Cousins. "3D is here: Point Cloud Library (PCL)". In *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 9-13 2011. IEEE.

[48] Automatic Addison. "How to Convert a Quaternion to a Rotation Matrix". automaticaddison.com. https://automaticaddison.com/how-to-convert-a-quaternion-to-a-rotation-matrix/ (Accessed on Apr. 6, 2021).

[49] R. Rusu. "Semantic 3D Object Maps for Everyday Manipulation in Human Living Environments". Ph.D. Dissertation, Institut für Informatik, TUM, Munich, Germany, 2009. [Online]. Available: http://mediatum.ub.tum.de/doc/800632/941254.pdf.

[50] P. Cignoni, M. Callieri, M. Corsini, M. Dellepiane, F. Ganovelli, and G. Ranzuglia. "MeshLab: an Open-Source Mesh Processing Too"l. In Vittorio Scarano, Rosario De Chiara, and Ugo Erra, editors, *Eurographics Italian Chapter Conference*. The Eurographics Association,

2008. doi:10.2312/LocalChapterEvents/ItalChap/ItalianChapConf2008/129-136.

[51] D. R. Peachey. "Solid Texturing of Complex Surfaces". *SIGGRAPH Comput. Graph.*, 19(3):279–286, July 1985. doi:10.1145/325165.325246.

[52] CloudCompare. "CloudCompare - 3D point cloud and mesh processing software". danielgm.net. https://www.danielgm.net/cc/ (Accessed on May. 27, 2021).