Universidade de Aveiro
Departamento de Engenharia Mecânica
2018

Bernardo Miguel Martins Lourenço

Reconstrução Tridimensional de Ambientes usando LIDAR e Câmara

Tridimensional Reconstruction of Scenes with LiDAR and Camera

DOCUMENTO PROVISÓRIO

Bernardo Miguel Martins Lourenço

Reconstrução Tridimensional de Ambientes usando LIDAR e Câmara

Tridimensional Reconstruction of Scenes with LiDAR and Camera

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia Mecânica, realizada sob a orientação científica de Miguel Armando Riem Oliveira, Professor Auxiliar do Departamento de Engenharia Mecânica da Universidade de Aveiro e de Paulo Miguel de Jesus Dias, Professor Auxiliar do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro.

o júri / the jury

presidente / president

Professor Doutor Vítor Manuel Ferreira dos Santos
Professor Associado com Agregação do Departamento de Engenharia Mecânica da Universidade de Aveiro

vogais / examiners committee

Doutor Eurico Farinha Pedrosa
Bolseiro do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro (arguente)

Professor Doutor Miguel Armando Riem de Oliveira
Professor Auxiliar do Departamento de Engenharia Mecânica da Universidade de Aveiro (orientador)

Palavras-chave Reconstrução 3D, Scanner Laser, Nuvem de Pontos, Calibração, Estimativa de Normais, Fusão de Cor, Registo de Cor

Resumo

Reconstrução tridimensional é uma área com múltiplas aplicações em arquitetura e robótica. Inúmera tecnologias existem para este efeito, como por exemplo a estereoscopia e luz estruturada. Contudo, muitas tecnologias carecem de precisão geométrica, que é por vezes um requisito. Outra tecnologia - LiDAR - é usada por causa dos seus resultados geométricos inigualáveis. No entanto, LiDAR é incapaz de capturar a cor dos objetos e uma solução é a integração de uma câmara com o LiDAR.
Assim, neste trabalho foram desenvolvidos um conjunto de técnicas e algoritmos direcionados para a reconstrução 3D com LiDAR e câmara. Além disso, um scanner 3D foi desenvolvido para registrar cenas reais. Em particular, um método de calibração inovador foi desenvolvido para a calibração do laser, com precisão superior a um método semelhante. Finalmente, os métodos foram testados com dados de cenas reais. A reconstrução geométrica foi bem sucedida mas o registo de cor ficou aquém do que era esperado, por causa de uma calibração pouco precisa da câmara.

Keywords 3D reconstruction, Laser scanner, Point Cloud, Calibration, Normal Estimation, Color Fusion, Color Registration

Abstract Tridimensional reconstruction is still a challenging area, that has multiple application in architecture and robotics. Several technologies are used today, like Stereoscopy or Structured Light, however, none is able to achieve precise geometric results, which are usually required. A technology, LiDAR, has evolved as the *de facto* technology for tridimensional reconstruction, being able to achieve unmatched results. Yet, this technology is unable to register the color of objects, so the usual solution is the use a camera for this.
Therefore, in this work we develop a set of algorithms and techniques for tridimensional reconstruction with a LiDAR laser scanner and a camera. Moreover, a 3D scanner was developed to register real-word scenes. In particular, a innovative calibration method was developed to calibrate the laser scanner, which performed above a similar calibration method.
Finally, six reconstruction were done to test the algorithms developed. The geometric reconstruction was very accurate but the color reconstruction was perfect, specially because of the poor calibration method of the camera.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Digital reconstruction of three dimensional scenes is a field that gained a high importance in areas like architecture, robotics, archeology and autonomous driving, and its goal is to produce high-detail and accurate models of real 3D scenes. These models can be then used, for example, in virtual reality, to provide and immersive experience, as if the used was in the real scene. Nowadays, 3D reconstruction is even available in, usually targeting Augmented Reality applications.

1.1 Motivation

3D reconstruction is still a challenge. First, real scenes have a complex geometric and objects can have small details that are hard to reproduce. Secondly, the measurements from sensors are subject to noise and errors, and are limited. For example, LiDAR lasers scanners are incapable to capture the color of the objects.

Past experience tells that a single sensor is not enough to model real environments, so currently the process lies into using multiple sensors and trying to merge the data from all the sensors, in order to capture a more realistic model.

However, this introduces a set of other problems inherent to this approach. One challenge is the registration of the different sensors, so that the data from one sensor can be accurately merged with the data from the other sensor. More specifically, the positions of the sensors need to be known accurately as well as their internal parameters. These parameters are determined using calibrations methods that need to be robust and precise.

Current reconstruction algorithms require a large amount of manual work, which means that a reconstruction require many man-hours to be completed, which is unfeasible for most applications. One of the goals of reconstruction is to develop algorithms that reduce human intervention, making it faster and more accessible.

1.2 Problem Description

Lidar laser scanners are becoming more available and more accessible, and because of their properties, as their high precision and high range, became an unmatched technology for 3D reconstruction. The LiDAR lasers are available as 2D laser scanners or 3D laser scanners, like the lasers from FARO and Riegl. Despite their immense potential, 3D laser scanners are still a very expensive solution and cheaper solutions are comprised of a cheaper 2D laser scanner

mounted on a moving frame. This solution, despite its low cost, can achieve good results, but requires a fine calibration between the laser and the moving frame.

Also, laser scanners do not register the color information, so a common practice is to pair the laser scanner with a camera to get both geometric and color data. This method also requires a fine calibration between both sensors to correctly merge the data.

1.3 Objectives

The objective is to develop a fully integrated solution for 3D acquisition using both laser scanner and a camera. This objective was divided into four main objectives:

- develop a 3D laser scanner, consisting of a laser scanner and a camera, and capable of recording data from both sensors in a fast and semi-autonomous way;
- define a methodology to record the data from the scene. In particular, it has to define the movements of the moving frame of the laser scanner and the camera and also when and where the laser scans and the images are recorded. This methodology should take into account the limitations of both sensors and the geometry of the scene;
- develop a set of methods to reconstruct the geometry of the scene reliably. The main challenge is the laser scanner to PTU extrinsic calibration, because it is fundamental for a reliable reconstruction;
- develop a set of methods to merge the image data with the geometry of the scene, to obtain a fully colorized 3D model.

The final result is, then, a point cloud with color and geometric information.

1.4 Document Outline

This dissertation is composed of eight chapters, which are arranged as follows:

Introduction The current chapter, in which the description of the problem is shown and the objectives of this work are defined.

State of Art Describes the technologies and solutions found in the field of 3D reconstruction, both commercial and academic, as well of a small technical background on this solutions.

Experimental Infraestructure Introduces all the software and hardware used to develop this work. In particular, the mobile robot for 3D scanner is described.

Methodology for Scene Capture Describes the methods and algorithms used to reconstruct the geometry of the scene, using the laser scan data recorded.

Methodology for Image Reconstruction Describes the methods and algorithms used to reconstruct the color information of the scene, using the camera images recorded.

Result and Discussion Presents and discusses the experimental results obtained in this work.

Conclusion and Future Work Summarizes the overall work developed and present possible future work.

Chapter 2

State of Art of 3D Reconstruction

2.1 3D Scanning Technologies

Many technologies exist to capture tridimensional information of the environment. The following section describes such systems and describe the basic working principle along with the pros and cons inherent to each ones. These techniques can be categorized into active and passive[**mada03**]. In particular, three technologies are described in detail: stereoscopy, structured light and LiDAR.

2.1.1 Stereoscopy

For many years, stereoscopy remained the most popular method for 3D sensing. This system uses images taken from a pair of cameras and extracts depth information using the perspective projection: the position of objects to the sensor is relatively further apart than the objects farther from the sensor. To compute depth, features from both images are extracted and matched together, which makes it a complex and computationally demanding, so it requires fast computers or dedicated software. This system has the advantage of having a good rate of acquisition and having high resolution. Also, color information is available. However, the reconstruction algorithm rely heavy on environment characteristics, like lightning conditions, texture and non-homogeneous regions [KHZ10]. This means that this method gives good results for edges and textured areas, but fails to get the depth information of continuous surfaces. Also, the geometric precision depends on the resolution of the images and degrades as objects are further apart.

2.1.2 Structured Light

In 2010, the availability of consumer grade depth sensors based on structured light lead to the development of consumer-grade small factor RGB-D cameras, started by Microsoft, with the *Kinect* and followed by other devices, like *ASUS Xtion* and *Intel RealSense*. These cameras come in small form factors, are inexpensive and are capable of capturing both color and depth information at real-time rates [Zol+18].

These appealing characteristics lead to a boost in the research and development in 3D reconstruction using this camera, culminating in the KinectFusion algorithm [New+11], capable of a fast and precise 3D reconstruction using a *Kinect* RGB-D camera and commodity GPUs. This algorithm was capable of performing real-time reconstruction, using a Iterative

Closest Point (ICP) for tracking the location of the device and for the registration of new RGB-D data. Nowadays, it is possible to achieve the same result using a phone equipped with a depth camera, like the *Lenovo Phab 2* or *ASUS Zenphone VR* and with the *Google Tango* software.

Structured light sensors work by projecting an infrared pattern onto the scene and calculate the depth via the perspective deformation of the pattern due to the different object's depth. However, this technique yields results that can be worst compared to other systems: the depth values from structured light have significant error or can be missing, specially from objects with darker colors, specular surfaces or small surfaces [SC13].

2.1.3 LiDAR

Light Detection and Ranging, or LiDAR, is one of the most precise and reliable ways to measure distances. It began being used shortly after the invention of Laser, in 1960, and it's valuable characteristics lead to the integration of it in the Apollo 15 mission, to serve as an altimeter to map the surface of the moon. Soon after, it was implemented in aircrafts to create high-precision and dense earth's surface models. Nowadays, its applications can be found everywhere where an accurate distance measurement is required, as for example in geology, archeology, geography and meteorology.

The success of LiDAR is related to the use of laser as its light source. Lasers are capable of emitting beams of light that are monochromatic, narrow and polarized. That is, lasers emit light in a narrow spectrum, so they can produce a single color of light, also known as monochromatic light. It improves the resilience against background light, making it possible to use even with sun light. Also, laser photons travel in a narrow beam that stays narrow even at large distances, with minimum scattering, therefore measuring the distance in a very small area in the surface. This improves the measurements near sharp transitions, where a bigger area of measurement can cause errors in the measurement. Moreover, lasers have a fast transition time, which is essential to reduce the error of the distance measurement, because it is directly influenced by the time between pulses, and a sharp transition reduces the error in the time measurement.

LiDAR is a Time of Flight sensor. Time of flight sensors, or ToF sensors, use the speed of light to measure the distance. A scene is illuminated by a light source and the reflected light is detected back by the sensor. The time that the light has taken to travel forth and back is then measured and the depth is calculated with this time. The measurement dependes on the type of ToF system used, that is ether *continuous* or *pulsed*. In *pulsed* systems, light is emitted in bursts with a fast shutter and the time between the emission and the reception is calculated. *Continuous* systems use a modulated light source and measure the phase-shift between the outgoing and incoming wave.

This technology has some advantages comparing to both previous approaches [Zol+18]. First, it is less computationally intensive, because the measurement is directly done by a specialized sensor. Second, it is partially independent of the lightning conditions because the light detected is emitted by the device itself. Further, it is capable of providing a dense and accurate depth values, even for continuous or irregular surfaces, unlike the stereoscopic approach. Moreover, it is much faster that any other method, capable of acquisition rates of hundreds of Hz. However, it has some disadvantages as well. The properties of the material, like the reflectivity, color and roughness can have significant effects on the accuracy of ToF sensors. Moreover, multi-path reflections are a common problem of ToF sensors,

Figure 2.1: Sick LMS511 2D laser scanner.

caused by multiple reflections of the light, causing errors in the measurements. Furthermore, interference can exist if multiple ToF sensors share the same environment. However, it is possible to mitigate this effect, by either controlling the sensor such that only one is activate at a time, or by using different modulation frequencies in their illumination source.

In recent years, LiDAR scanner became a fundamental technology for industrial and robotics applications. Their small form factor and high precision are essencial for numerous applications. In general, two types of laser scanners exist: the 2D laser scanners, the one used in this work, and the 3D laser scanners.

2D laser scanners emit a single laser beam, which is reflected by a rotating mirror to scan across a planar area. They are also the most accessible type, as their price ranges range from hundreds to tenths of thousands of Euros, depending on their characteristics. One example of this laser scanners is the SICK LMS511, shown in Figure 2.1.

This laser scanners have a large number of applications. For example, 2D laser scanners are used in autonomous robots, to provide precise 2D mapping information of the environment, which can be user afterwards for location and navigation [SPW17]. Compared to other technologies, like stereo vision, this one requires small processing power and yields accurate results, so their application is easy to implement and requires low processing power.

Another widely used application is intrusion detection. The 2D laser scanner can be placed in a room or door to detect if any object enters to the space. For example, it is used to ensure the safety of workers in industrial environment, ensuring that workers do not get close to working machines. Other example is in theft prevention in museums and banks to secure specific areas against robbery or vandalism[1].

[1]In `https://www.sick.com/ag/en/industries/building-safety-and-security/c/g288283`.

2.2 Academic Work related to 3D reconstruction

Several scientific studies can be found concerning the research and development of 3D sensors using laser scanners, and in many studies, the 2D laser is mounted on top of a moving platform and each individual laser scan is registered on a static frame of reference, in order to create a full 3D scan. The motion of the laser scanner can be classified as continuous or discontinuous. Usually a continuous motion is used for real-time systems, like autonomous vehicles, while a discontinuous motion is used when real-time is not important, like accurate 3D reconstructions of scenes. In the following paragraphs such systems are described.

In [SNH03], a mobile robot was capable of autonomous navigation, thanks to a tilting *LMS200* laser scanner, that provided a depth map of the front of the robot, with a maximum resolution of 721×256 points. However, a scan of 181×256 points took about 3.4 s, and scans with more points (361 or 721) took double or quadruple this time, making it not suitable for real-time operation. This previous system had a limited field of view, so in [Cai+05] a *LMS291* was mounted on a pan-tilt unit for generating a 3D point cloud with a parameterized field of view.

More recently, 3D laser scanner began being developed for real-time Simultaneous Mapping and Navigation, or SLAM, in autonomous robots. This was specially due to the *DARPA Grand Challenge*, a competition to award the fastest autonomous unmanned vehicle that completed a 300 miles track. In [Mau+09], a 3D laser scanner (Figure 2.2) was developed by placing two *LMS200* planar laser scanners on a rotating vertical axis, capable of generating a high-quality 3D point cloud with a 360° field of view. Several other lasers were developed by rotating the laser in a continuous motion using a turntable [NTM07], a swinging platform [Yos+11]. This sensor also became lighter, compact and modular, making it possible to integrate in multiple systems easily. One of this systems is *KaRoLa* (Figure 2.3), described in [Pfo+14]. This laser scanner was then applied to several system, specially in search and rescue robots.

Figure 2.2: 3D laser scanner developed in [Mau+09].

The 3D laser sensors are only able to reconstruct the geometry of the scene. Some sensors are also able to measure the intensity of the reflected light, returning the reflectance value for each point. This intensity is measured, of course, in the frequency spectrum of the emitted light of the sensor, which is usually infrared (950 nm). To create a textured model, one or more cameras are coupled to the sensor, and both depth and color data is merged, in a process called fusion. This is specially important for areas like architecture or archeology,

Figure 2.3: The KaRoLa 3D laser scanner.

where color information is very important. An example of this work be seen on [DMS06], where a 3D sensor like the one described in [SNH03] was paired with a camera, to generate a 3D reconstruction with color.

These techniques were applied, for example, in cultural heritage, to model important art pieces. One of the most famous examples is the Michelangelo project [Lev+00], which developed a technique to register data from a triangulation sensor and color image data to reconstruct the 3D geometry of the statue of Michelangelo's David. One of the challenges in this project was to capture the chisel marks in the surface of the status, requiring a resolution of 1/4 mm, in a statue 5 m tall.

2.3 Comercial Solutions

Many commercial solution exist for 3D reconstruction. In particular, two solution are presented that have similar characteristics and use-cases to this work: The Matterport and Faro Focus.

2.3.1 Matterport

Matterport is advertized as an all-in-one solution, capable of both 3D reconstruction and capture 4K resolution images from the scene. Their target is mostly the reconstruction of indoor scenes, more specifically, from houses. The 3D model can be used to showcase the interior of the house, using both virtual reality or panoramic photography, or to make 3D measurements and automatically generate floor plans[2].

Matterport offers two products: a 3D camera and a cloud service to process the raw data taken with the camera. The camera, as seen in Figure 2.4, consists of two pais of sensors: two structured light sensors and a two photographic cameras[3]. The structured light sensor has an advertized 99% geometric accuracy within the 4.5 m maximum range. The Photography sensor is a 4K HDR camera.

The overall process to capture a scene is fast and easy: the 3D camera is placed on a tripod and is controlled remotely. Each acquisition takes about 20 s and the result is a 3D colorized mesh with 4 million vertices and a 360° panoramic photography with 134.2 MP. To

[2]In `https://matterport.com/`.
[3]In `https://matterport.com/`.

Figure 2.4: Matterport Pro2 Camera.

scan an entire environment, an operator moves the camera to each space and make multiple new acquisitions from that space.

A set of models reconstructed from the with the Matterport solution can be found in "Matterport 3D Space Gallery"[4]. As an example, the model named "Pennsylvania Craftsman Home"[5] can be seen in Figure 2.5. This model represents the complete interior of an house, which look very detailed.

2.3.2 Faro Focus

Faro Focus[6] are a series of 3D laser scanners targeted for the sectors of architecture, engineering, construction and product design. As such, this solution is capable of fast 3D reconstructions both on outside and inside environments with great accuracy. The 3D scanner (Figure 2.6) is design for portability and is equipped with a laser scanner with a precision of ± 1 mm and a range of 0.6 m to 350 m, and a 8 MP HDR camera.

An acquisition, like in the *Matterport Pro2 Camera*, is quick and easy, but does not require any remote computer, as the scanner incorporates a touch LCD screen. All the subsequent processing is done afterwards.

Faro Focus scans are very precise, as they are used for precise measurements of the reconstructed scene. As an example, a scan obtained by the Faro Focus can be seen in Figure 2.7[7].

[4]In `https://matterport.com/gallery`.
[5]In `https://matterport.com/3d-space/pennsylvania-craftsman-home/`.
[6]In `https://www.faro.com/en-gb/products/construction-bim-cim/faro-focus/`.
[7]In `https://streambend.net/laser-scanning/`.

(a) Side view.

(b) Top view.

Figure 2.5: Matterport "Pennsylvania Craftsman Home" model.

Figure 2.6: Faro Focus 3D laser scanner.

Figure 2.7: Faro 3D scan.

Chapter 3

Experimental Infrastructure

This chapter describes in detail both the hardware and software used in this project. The hardware - a mobile robot - is described in Section 3.1 and all the software used and implemented in this robot is described in Section 3.2.

3.1 Hardware

The hardware used in this work was a 3D scanner called "lemonbot", shown in Figure 3.1. This scanner was developed to perform the acquisitions, to build a platform that was portable, mobile and did not require the presence of the operator, so that there a minimum interference with the scene (for example, no cables are required). Therefore, the robot was all packed into a tripod which which included a battery capable of powering all the systems and the control is made via a remote connection.

The robot has, in total, seven components: three of which are the essencial components for the acquisitions: the 2D laser scanner, the camera and the pan-tilt unit, or PTU and the other four components form the infrastructure: the minicomputer, the wireless router, the battery pack and finally the tripod. Each of these components are described in detail in the following lines.

2D Laser Scanner

One of the objectives of this work is to evaluate the usage of different 2D laser scanners to study the performance of the reconstruction and calibration algorithms. So, three laser scanners were chosen: the SICK LMS200, the Hokuyo UTM30LX and the Hokuyo URG04, as seen in Figure 3.2. Each of the laser scanners differ in their characteristics, like the size, price, range and error. In Table 3.1, the characteristics of the laser scanners are presented.

Camera

The camera used in this work was a PointGrey Flea3 FL3-GE-28S4 Camera (Figure 3.3), which is extensively used in industrial and traffic applications. The high quality of the images, the programming interface and it's compact size and weight makes it perfect for computer vision applications in industrial environment. The most relevant characteristics are represented on the Table 3.2.

Figure 3.1: Lemonbot mobile 3D scanner

Table 3.1: Comparison of the three laser scanners used, based on the data provided by the manufacturers.

	SICK LMS 100	Hokuyo UTM 30LX	Hokuyo URG04
Aperture angle	270°	270°	240°
Angular resolution	0.25°	0.25°	0.36°
Scanning frequency	10 Hz	40 Hz	10 Hz
Maximum range	20 m	30 m	5.6 m
Systematic error	±40 mm	not available	not available
Statistical error	20 mm	30 mm	30 mm
Dimensions (mm^3)	152 × 102 × 106	60 × 60 × 87	60 × 60 × 87
Weight	1100 g	370 g	160 g
Power consumption	<12 W	8.4 W	2.5 W

(a) SICK LMS100 (b) Hokuyo UTM30LX (c) Hokuyo URG04

Figure 3.2: Laser scanners used.

Pan Tilt Unit

Both the laser and camera are placed on top of a pan and tilt unit for their movement. The selected PTU was the FLIR PTU-D46 (Figure 3.4), which is a compact and light module. The characteristics of this PTU are presented in Table 3.3.

Figure 3.3: PointGrey Flea3 FL3-GE-28S4

Table 3.2: Characteristics of the PointGrey Flea3 FL3-GE-28S4 Camera

Resolution	1920×1448
Framerate	15 fps
Pixels	2.8 MP
Color	Yes
Interface	GigE Vision
Power	12 V to 24 V
Dimensions	29 mm × 29 mm × 30 mm

Table 3.3: FLIR PTU-D46 characteristics.

Pan range	±159°
Tilt range	−47° to 31°
Maximum payload weight	4 kg
Angular resolution	0.0032°
Communication	serial interface
Size	small

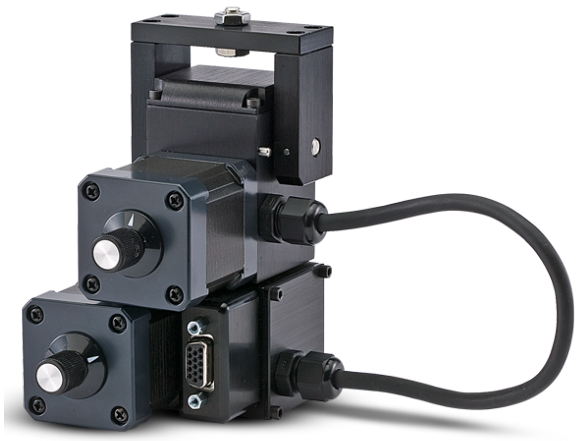


Figure 3.4: FLIR PTU-D46

3.2 Software

This section describes the software used in this work. Two different applications were developed. One application was the software that constitutes the mobile 3D scanner, which was developed in a framework called Robot Operating System, described in Section 3.2.1. The other one is the software used to process the data recorded by the 3D scanner, which is described in Section 3.2.2.

3.2.1 Robot Operating System

Robot Operating System, or ROS, is a software architecture for robot development, providing a collection of tools, libraries and conventions to simplify the development of complex robotic systems. It was originally created at stanford University in the mid-2000s and new is widely adopted as the standard framework for robotics by most research communities.

Its design principles follow the one of a distributed system. In ROS, a system is composed by multiple nodes that have just one task and communicate between them by message passing. To achieve this, ROS, in its core, has the infrastructure responsible for the:

Orchestration . ROS runs, stops and monitors all nodes, so in the case of failure, for example, ROS is capable of restarting the node.

Communication . ROS provides both the pipelining to distribute messages as well as the standard serialization specification.

Configuration . ROS provides a key-value store for parameters that are accessible for nodes. These parameters can be specified when the node is created and also changed dynamically at runtime.

Discovery . Each node in the environment can inspect it, such as finding other nodes and finding topics.

This architecture has many advantages, such as:

Fault-tolerant . A failure in one node does not affect other nodes, so there is not a overall system crash, unlike monolithic systems. Usually, because errors are transitive and not very frequent, a restart-on-failure policy is used to keep the downtime low.

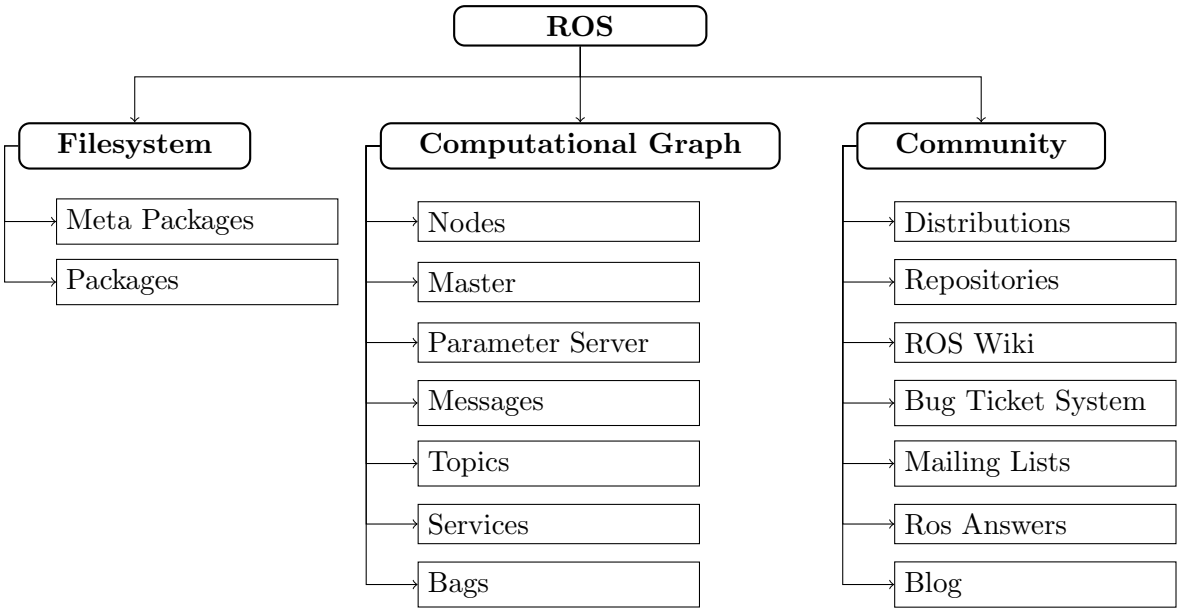
More generic . Because each node has a single responsibility, they tend to be more generic and detached to a single project, so integration in a new project can be easy. This is fundamental to reduce the need to "reinvent the wheel", therefore reducing the development cost and time of this complex systems.

Easier to develop . Each component can be developed be separate developers, with different languages and independent release cycles, because they do not have any dependency between each over and only a message specification needs to be agreed on between developer teams, making collaborative development possible. Debugging is also easier, because each node can be unit-tested separated from the "real environment".

Large Community . ROS is open source, which incentivises different research groups to share packages. Also, ROS is widely adopted so multiple packages for numerous tasks are already programmed, and can be integrated easily into new systems. One of this examples are drivers, which almost never require to be developed, because most robotic hardware already has a developed driver.

ROS is split into 3 levels, according to [Fer15]: the *filesystem* level, the *the computational graph* level and the *community* level. Each level is composed of several core components, that make the whole system work, as seen in Section 3.2.1. Some components that were required for this work are explained next.

Figure 3.5: ROS architecture overview.



Messages

Messages are the communication element and are just structures of data composed by primitive types such as integers, floating point numbers, strings and other messages. All messages follows a schema, which is required to encode and decode the message. Messages are serialized in a binary format before and after the exchange, so messages are small and efficient. Moreover, all messages have a header, which contains a timestamp and the source of the message.

Topics

Topics are named buses over which nodes exchange messages. Topics follow the publisher/subscriber paradigm, so nodes can both subscribe to receive messages or publish messages to the topics. The exchange of data is done anonymously, so nodes are not aware which nodes are publishing or subscribing to a topic. This way of exchanging data is well suited for streaming data, such as sensor data.

Launch Files

Launch files are xml files that describe the steps to launch multiple node, as well as setting parameters. Launch files also support composition, so a launch file can invoque other launch files. Launch files were used in this project extensively, to launch the drivers of the mobile robot and to launch the acquisitions.

Bags

A bag is a file format for storing ROS messages, and have a myriad of tools to store, process, visualize and analyze them. During runtime, bags can be used to store the messages published in multiple topics, so data can be analysed later. Also, messages in bag files can be republished back into the system for testing or visualization purposes.

In this work, bag files played a very important role, as they were responsible to store the sensor data and also the transformation graph of the 3D scanner.

RViz

RViz is a 3D visualizer for ROS for displaying sensor data, like laser scans and point clouds, and the representation of the robot state, like the position of the coordinate frames and the joints. RViz can be a indispensable debug tool, for example, by comparing the real environment with the displayed environment shown in RViz.

TF

TF is a package that keeps track of multiple coordinate frames and maintains the relationship between coordinate frames in a tree structure, called the transformation graph. This transformation graph can be queried to obtain the transformation between two frames at any point in time. Also, tf can work in distributed systems, just like ROS, so any node can publish transformations and the transformations can be obtained in any node. TF is also responsible to interpolate between the discrete transformations and handle transformations with different sampling rates.

URDF

Unified Robot Description Format, or URDF, is a format to represent a robot model, like the joints and links configuration and the geometry of the joints. This file is loaded at runtime and the transformations are published according to the joint state.

3.2.2 Processing Application

This application required a wide spectrum of libraries, frameworks, file formats and graphical programs, which are described next.

Libraries and Frameworks

The software developed in this work that implements the processing algorithms was done using the Python programming language and some libraries to provide both data structures and common algorithms. Both the language and libraries are described next.

Python is a general purpose programming language that become popular for its syntax and small learning curve. It is also the defacto language for science, along with MATLAB. However, inlike MATLAB, it is s open source, has large community and has plenty of libraries that provide many algorithms and efficient data structures. Also, it is a dynamic language, which facilitates the process of testing and debugging the code developed.

Numpy is a library that contains an implementation of *nd*-arrays, as well as algorithms to manipulate them. This library was fundamental for this work to store and process the point data. The main advantage of this library is that it is implemented in compiled languages like C and Fortran to implement high performance and optimized data structures and algorithms, available through a clean interface in Python.

Pandas is a library that provides a fundamental data structure which was extensively used in this work: the DataFrame. DataFrames store data in columns, which is perfect to store tabular data. This is a common way to store point information, because point clouds are, fundamentally, tables, where each property are stored as a column, like x, y, z for position and r, g, b for color. Also, because it relies on numpy arrays to store the data, it is still very high performance.

PIL , or Python Image Library, is a library for image loading and manipulation, and was used to read and write the images recorded by the camera.

Jupyter provides an interactive interfaces, called Jupyter notebooks, which provides interactive documents with embedded code. This notebooks are extremely useful and were used to document and explore the code that was used in this work.

File Formats

AVRO is a binary data serialization format that is user to store collections of structured data. This format was chosen to store the laserscans and the image metadata. AVRO relies on schemas, which describe the data in the file is stored with the data. Therefore, an AVRO file is self-describing and data can be read and write without much overhead. Moreover, this format is implemented in Python and has numerous tools for inspection and conversion of the data.

PLY , or Polygon File Format is one of the most used and supported file formats to store three dimensional data, like point clouds and meshes. It was originally developed and used in the Stanford University to store data from 3D scanners. It supports a wide number of properties, like color, transparency, surface normals and texture coordinates. It also supports the storage of custom properties, which were required for this work, for example in the segmentation for the calibration. Moreover, it supports binary encoding, so files are small and fast to read and write.

JPEG is a commonly used format for images and was used to store the recorded images.

YAML is an human readable format that was used to store the parameters of the acquisitions, suck as the the extrinsic calibration of the sensors. The advantage of this format is that files are very easy to read and modify by the user.

Graphical Software

CloudCompare is a software to render, process and manipulate 3d point clouds. It includes many algorithms, like point cloud registration, re-sampling, handling scalar fields, and automatic or interactive segmentation. It can also render point clouds using different shaders and support point cloud decimation, which is a technique that allows manipulation of large point clouds without a decrease in performance. A screenshot of this software can be seen in Figure 3.6.

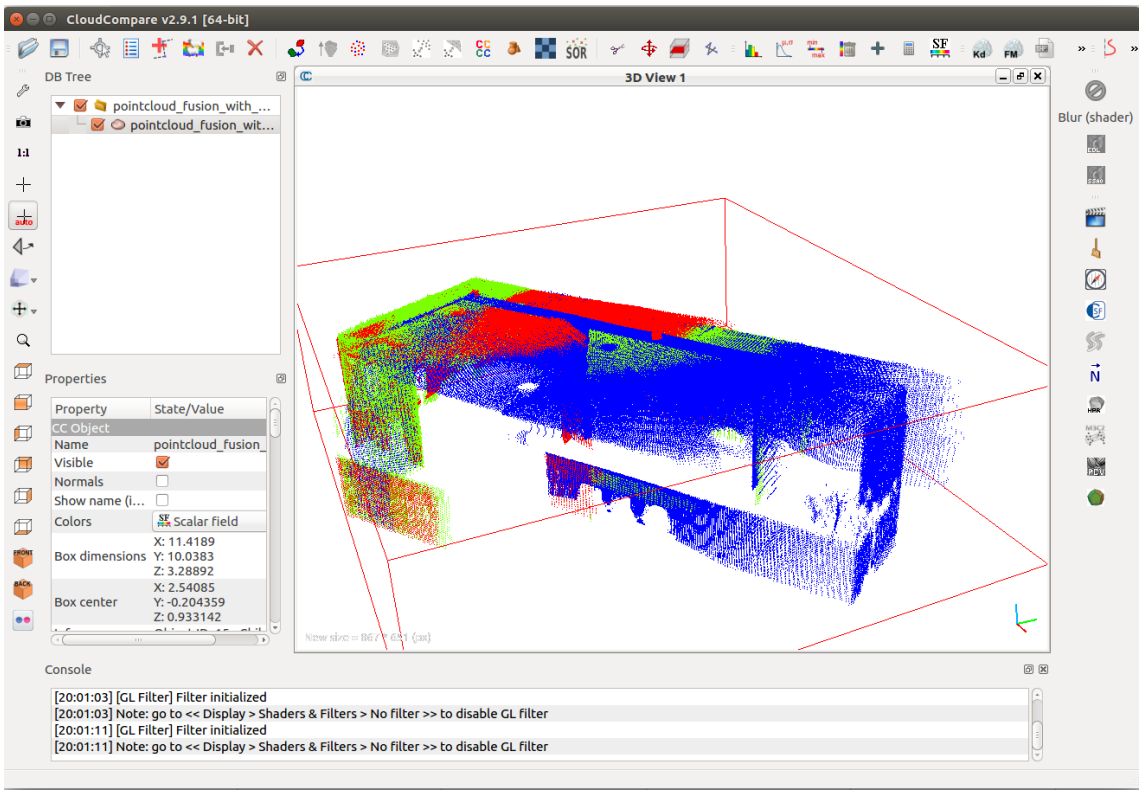


Figure 3.6: Cloud Compare screenshot.

Chapter 4

Methodology for 3D Scene Acquisition

This chapter describes the concepts and methodology around a capture of a scene. This methodology is split into two levels: the acquisition and the capture level. An acquisition is a collection of sensor data recorded from a specific position of the scene and a capture is a collection of acquisitions taken from the scene scene, but from different positions.

The reason behind these two level has to be because there is limited information about the scene in each acquisition. These limitations are the occlusion of different parts of the scene by objects, the hardware limitation of sensors, like the small aperture angle of the laser scanner or the field of view of the camera, and the environment factors, for example, the lightning conditions and the reflectivity of the object's surfaces. This effects are show in Figure 4.1. To overcome these limitations, multiple acquisitions required, however, this also comes with some challenges, for example, how to merge all the acquisitions and how to handle the redundant data.

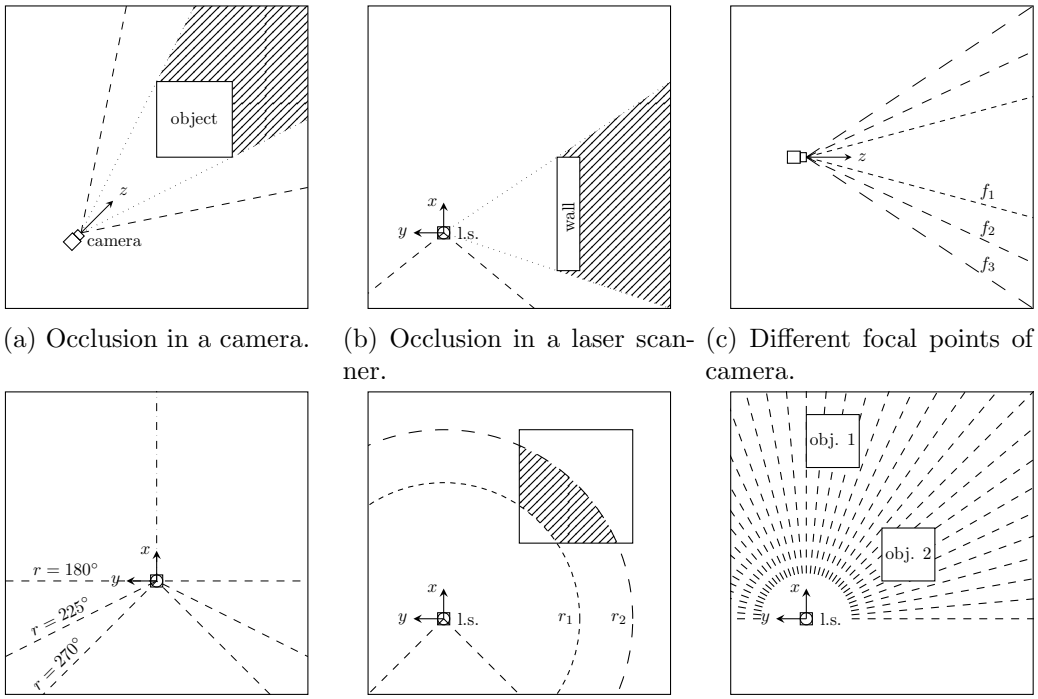
So, acquisitions and captures are different levels and each one has a different method and objectives. In an acquisition level, the focus is on how to operate the scanner and define how the data is recorded. In a capture level, the focus is on how to plan multiple acquisitions so a good reconstruction is possible. In the following sections, both acquisitions and captures are further explained.

4.1 Acquisition

An acquisition is a collection of sensor data (laser scans and images) collected by the sensors in the scanner. Both sensors sample only a small subset of the whole environment: the laser scans only have points from a planar region of the space and cameras are limited by their field of view. To overcome this limitation, both sensors are moved to different poses in space to cover a wider space. In this case, the movement of the sensors is controlled by the movement of the joints of the PTU.

4.1.1 Movement Programming

To program the motion of the PTU joints, a list of waypoints are defined and the joints move from waypoint to waypoint. The waypoints are defined in a grid in the joint-space and



(a) Occlusion in a camera. (b) Occlusion in a laser scanner. (c) Different focal points of a camera.

(d) Radial aperture of a laser. (e) Range limitations. (f) Radial resolution.

Figure 4.1: Limitations of a single acquisition.

the movement between waypoints is the one that defines the shortest path possible and most of the movement is done in pan. So, each acquisition is parameterized with the following parameters: the range (minimum and maximum angle) of pan/tilt, the speed of each joint and the number of waypoints in pan/tilt. An example of this parameterization can be seen in Figure 4.2.

Once the movement of the scanner was defined, the next step is to define when to record the laser scans and the images, according to it. Because of the nature of both sensors, it was established that laser scans are captured continuously during the pan movement between the waypoints and images are captured at every waypoint.

4.1.2 Parameterization Considerations

This methodology has numerous implications: first, the pan and tilt range is only limited by the PTU capabilities, but it beneficial to use the maximum range possible, in order to get as much data as possible. In this work, most data collected is redundant, for example if multiple tilt angles are used. However, despite not being required, it can be beneficial for the final reconstruction if the point density is high.

Second, the number of laser scans recorded is going to depend on the pan speed and the frequency of scanning of the 2D laser scanner. So, it is expected that a laser scanner with a lower scanning frequency to require a slower speed compared to a faster one, to collect the collect the same amount of data.

Third, the camera used in this work did not have stabilization so, to get sharp images, a complete immobilization was required in each waypoint. This was achieved by setting a time

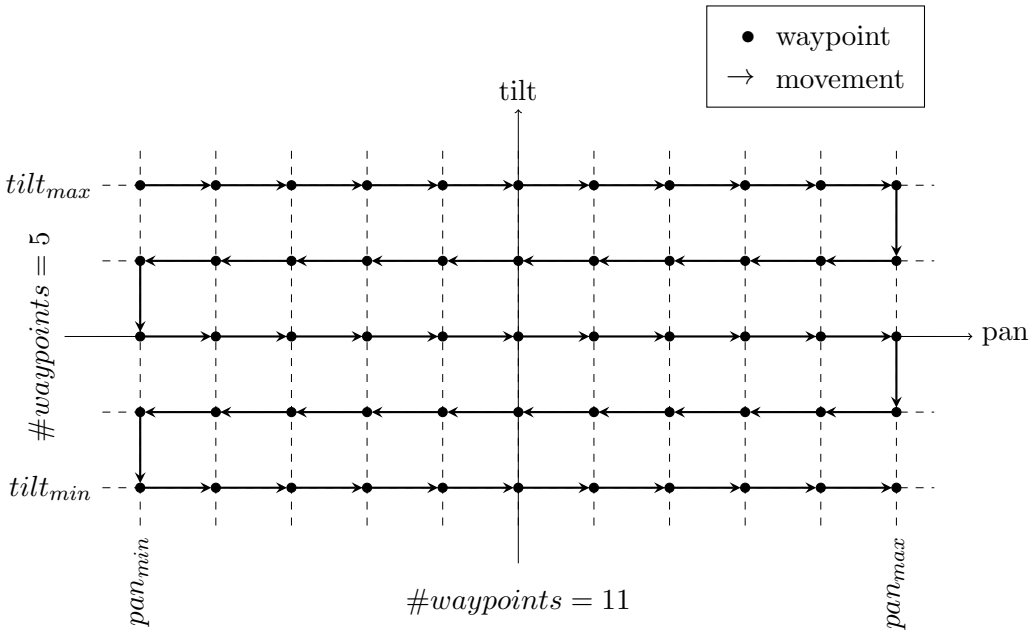


Figure 4.2: Waypoints and movements in the pan/tilt joint space.

between the stop of all joints and the capture of the image by the camera. In this work, a time of 1.5 s was empirically defined.

Last, the waypoints' angle increment has to be enough so that part of the previous image appear in the next image, so that every observable part of the scene is seen at least once. This depends heavily on the focal point of the camera: the bigger the focal point, the least area it captures and more waypoints are required.

4.1.3 Acquisition node

To implement this functionality, a ROS node was developed according to the previously defined specifications. This node, called *single_acquisition_node* is present in the *lemonbot_acquisition* package and the way it is implemented is the following: the PTU movement is controlled and the selected messages are published into a new topic. For convenience, all the acquisition topics are republished into the */acquisition* namespace. So, during an acquisition, two topics can be found, each one corresponding to each sensor, inside this namespace: the laser scans are in */acquisition/laserscans* and the images are in */acquisition/images*. This idea of publishing back all the important messages greatly improved the acquisition organization, so all the topics that were required were also published into this namespace. This topics were the */acquisition/camera_info*, containing the intrinsic parameters of the camera, and */acquisition/tf* and */acquisition/tf_static*, containing all the transformations of the robot.

Now, data from these topics need to be saved permanently, so this was done using a ROS tool called *rosbag*, that saves all the data from a predefined set of topics into a binary file called a *bag* file. This was a easy and powerful solution, because it allows the acquisition to be reproduced again, by republishing all the messages back into the system. To save a set of topics, a node called *record* from the *rosbag* package is run with the list of topics that required to be recorded into disk. In this case, the required topics are all the topics inside

```
path:        acquisition_2018-09-07-16-01-46.bag
version:     2.0
duration:    4:53s (293s)
start:       Sep 07 2018 16:01:47.11 (1536332507.11)
end:         Sep 07 2018 16:06:40.96 (1536332800.96)
size:        87.0 MB
messages:    6690
compression: none [16/16 chunks]
types:       sensor_msgs/CameraInfo [c9a58c1b0b154e0e6da7578cb991d214]
               sensor_msgs/Image      [060021388200f6f0f447d0fcd9c64743]
               sensor_msgs/LaserScan  [90c7ef2dc6895d81024acba2ac42f369]
               tf2_msgs/TFMessage     [94810edda583a504dfda3829e70d7eec]
topics:      camera_info    953 msgs    : sensor_msgs/CameraInfo
               images          10 msgs    : sensor_msgs/Image
               laserscan     2788 msgs    : sensor_msgs/LaserScan
               tf            2938 msgs    : tf2_msgs/TFMessage
               tf_static        1 msg     : tf2_msgs/TFMessage
```

Figure 4.3: Example of a recorded bag file info.

the */acquisition* namespace.

To streamline the acquisition process, all these components (the acquisition node, the topic republisher nodes and the rosbag record node) can be all launched through a *launch file*. A set of all the parameters required for each acquisition can be override over the default parameters. Therefore, running an acquisition just requires a single command:

```
roslaunch lemonbot_acquisition single_acquisition.launch \
    pan_min:=-90 pan_max:=90 pan_vel:=10 pan_nsteps:=25 \
    tilt_min:=-15 tilt_max:=15 tilt_nsteps:=5
```

In conclusion, running the previous command will run an acquisition and in the end, a bag file will be present, with the topics *images*, *laserscan*, *camera_info*, *tf* and *tf_static*, therefore all the information relevant for the reconstruction.

To have a better insight in the bag file, a tool called *rosbag info* can be used. All the details about when the calibration took place, how long it took as well as how many messages it contains are printed. An example of this information is shown in Figure 4.3

4.1.4 Data Serialization

Despite their potential, bag files are not the best way to store the acquisition data for the reconstruction pipeline, because some limitations of bag files were found: the most noticeable is that the full transformation graph is stored, while in fact only the transformations between the start and end frame of the PTU are needed, as well as the transformations between the PTU mount link and each one of the sensors, which are static. Also, this transformation messages are not synchronized with the laser scans and image messages, which means an interpolation has to be performed each time the data is read. Another drawback is that bag files stores messages in it's own format, which hinder reading and inspecting the data with external tools, which can be helpful to check if an acquisition was successful. For example,

```
{
    "ranges" : [ ... ],
    "limits" : {
        "min" : 0.100000001490116,
        "max" : 29
    },
    "timestamp" : 1536174204611117487,
    "angles" : {
        "min" : -2.35619449615479,
        "max" : 2.35619449615479
    },
    "transform" : {
        "rotation" : [ ... ],
        "translation" : [ ... ]
    }
}
```


Figure 4.4: Example of laser scan row.

the images are serialized into a ROS message, instead of being in a file with a known format, like *JPG*, which would allow for easier access and inspection.

To solve these issues, a preprocessing of the bag files was performed, to convert and extract all the important information into well known and useful formats. Each laser scan was stored in a *AVRO* file row that contains the timestamp (when it was taken), the minimum and maximum angle (aperture of the laser scan), the minimum and maximum ranges that the laser can capture, the transformation of the PTU and the list of all the measured ranges. An example of such row is show in Figure 4.4, obtained using the *avro cat* command. Each image was stored in a separate *JPEG* file and it's timestamp and transformation was stored in a row, again in a *AVRO* file. The parameters inherent to the acquisition, such as the name of the bag, the extrinsic and intrinsic calibration of the camera used and the extrinsic calibration of the laser was stored in a *YAML* file. The transformations in both the images and laser scans are stored as vector for translation and quaternion for rotation. An example of this parameters can be seen in Figure 4.5.

4.2 Capture

As seen before, acquisitions only capture a subset of the scene geometry and color, so multiple acquisitions are required. This problem can be partially solved by recording multiple acquisitions instead of one. Therefore, a capture is a collection of acquisitions of the same scene and its goal is to collect enough data to create a fully 3D reconstruction. However, this raises some challenges, on how to plan and execute the multitude of acquisitions and how to merge the data from all of the acquisitions (discussed in Section 5.4).

Planning determines where should the 3D scanner be placed in each acquisition and the sequence of the acquisitions. In this work this was done with the objective to maintain a minimum point density on all surfaces, capture color information of as much surfaces as possible and minimize the processing errors. Each one of this problems and its solutions are explained in more detail hereupon.

```
bag: acquisition_2018-09-05-20-02-46.bag
camera:
    extrinsic:
    translation: [ ... ]
    rotation: [ ... ]
    intrinsic:
    principal_point: [ ... ]
    height: 1448
    focal_lenghts: [ ... ]
    width: 1928
    distortion_coef: [ ... ]
    distortion_model: plumb_bob
laser:
    extrinsic:
    translation: [ ... ]
    rotation: [ ... ]
    limits:
        max: 29
        min: 0.1
    angles:
        max: 2.356194
        min: -2.356194
```

Figure 4.5: Example of the parameters YAML file.

To begin with, occlusion and range limitations restrict the covered area of an acquisition to a subset of the scene, which is dependent of the position and orientation of the 3D scanner in the scene.

Secondly, the point density decreases with the distance of the object to the sensor, which can influence the reconstruction, specially if small objects exist. For example, a wall does not need a high point density, but a smaller object such as a chair or table should have a higher one. Therefore, the position and orientation of the acquisitions should regard this, such that the point density is adequate to the dimensions of the objects.

At last, the acquisition registration requires that between each acquisition there is enough overlap between the point clouds, so enough correspondent points exists to compute the registration between acquisitions. So, between each acquisition there should be a maximum distance, such that this registration is possible. Also, this registration requires a good initial estimate for the transformation, otherwise it is not able to find a correct transformation. The solution proposed is to define a sequence of acquisitions such that each subsequent acquisition is near to the previous one and the relative rotation is small.

In conclusion, a good capture planning requires that key acquisitions are made to minimize occlusion and maintain a adequate point density and multiple acquisition have to be made, connecting the key points, and each acquisitions should be close enough to the previous one, such that the registration between acquisitions are possible. In this work, we determined this sequence of acquisitions by determining a path inside the scene. This process, however, can be very subjective and dependent of the user, and the evaluation of the capture is all done afterward, because no feedback exists during the capture, which is a disadvantage in comparison with other reconstruction systems like the *Google Tango*.

Chapter 5

Methodology for Geometry Reconstruction

This chapter presents the methodology using for the reconstruction of the geometry of a scene. In Section 5.1, the laser scans are transformed into point clouds. In Section 5.2, two calibration methods are described to obtain the extrinsic calibration of the laser scanner to the PTU. Section 5.3 describes a method to estimate the normals, based on the structure of the point cloud. In Section 5.4, a method to register the multiple acquisitions is described, to merge the acquisitions into one point cloud. In Section 5.5, three point cloud filters used in this work are described.

5.1 Point Registration

Each laser scan is a collection of points in polar coordinates, so each range point (r_i, θ_i) is transformed to a point in the laser frame of reference according to Equation (5.1). The angles are uniform distributed between a minimum and maximum angle, θ_{min} and θ_{max}, respectively, so $\theta_i = \{\theta_{min},\ \ldots,\ \theta_{max}\}, i = 1 \ldots N$. The index i is defined as the range index of each laser scan.

$$\begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} = \begin{bmatrix} r_i \cos(\theta_i) \\ r_i \sin(\theta_i) \\ 0 \end{bmatrix} \tag{5.1}$$

Further on, each point p_{ij} is registered in the referencial of the acquisition. According to the transformation graph (see Figure 5.1), there are two transformation from the acquisition frame and the laser scanner frame: the transform from the acquisition frame to the PTU frame ${}^{ptu}_{acq}T$, which is dynamic and depends on the PTU position for each laser scan, and the transformation from the PTU frame and the laser scanner frame ${}^{laser}_{ptu}T$, which is static. This two transformations can be chained together to obtain the point in the acquisition frame, according to:

$$p_{i,j} = \begin{bmatrix} x_{i,j} \\ y_{i,j} \\ z_{i,j} \\ 1 \end{bmatrix} = {}^{ptu}_{acq}T \cdot {}^{laser}_{ptu}T \cdot \begin{bmatrix} r_i \cos(\theta_i) \\ r_i \sin(\theta_i) \\ 0 \\ 1 \end{bmatrix} \tag{5.2}$$

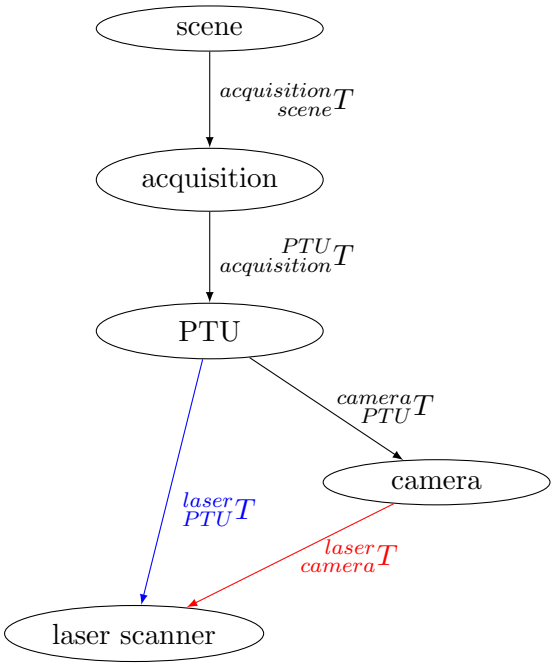


Figure 5.1: Transformation graph.

At this phase, each point has 2 indexes, one for the laser scan index $j = 1 \dots L$ and another for the range index $i = 1 \dots N$, relative to the each laser scan. Therefore, at this stage, each point can be indexed with a pair of (i, j) indexes. This point clouds are called structured point clouds.

This reconstruction phase depends heavily on the transformation from the PTU to the laser scanner. This transformation is obtained by a calibration process and is commonly referred to as the extrinsic calibration of the laser scanner.

In conclusion, for each acquisition results a point cloud with $L \times N$ points, where L are the number of laser scans and N the number of range values in each laser scan. Each point can be indexed in a bidimensional space, which is useful for subsequent algorithms.

5.2 Laser Extrinsic Calibration

The key for a good geometric reconstruction is the laser scanner extrinsic calibration, which has to be accurate, so that every point is correctly located. Therefore, two calibration methods are here presented: the RADLOCC camera-laser calibration (Section 5.2.1) and a new method developed in this work (Section 5.2.2), that aims to achieve better results than the latter.

5.2.1 Robust Automatic Detection in Laser of Calibration Chessboards Method

In [ZP04], a method for automatic calibration of a camera with respect to a 2D laser scanner is presented. This method, known as Robust Automatic Detection in Laser Of Calibration Chessboards, or RADLOCC, uses information from both sensors and tries to find

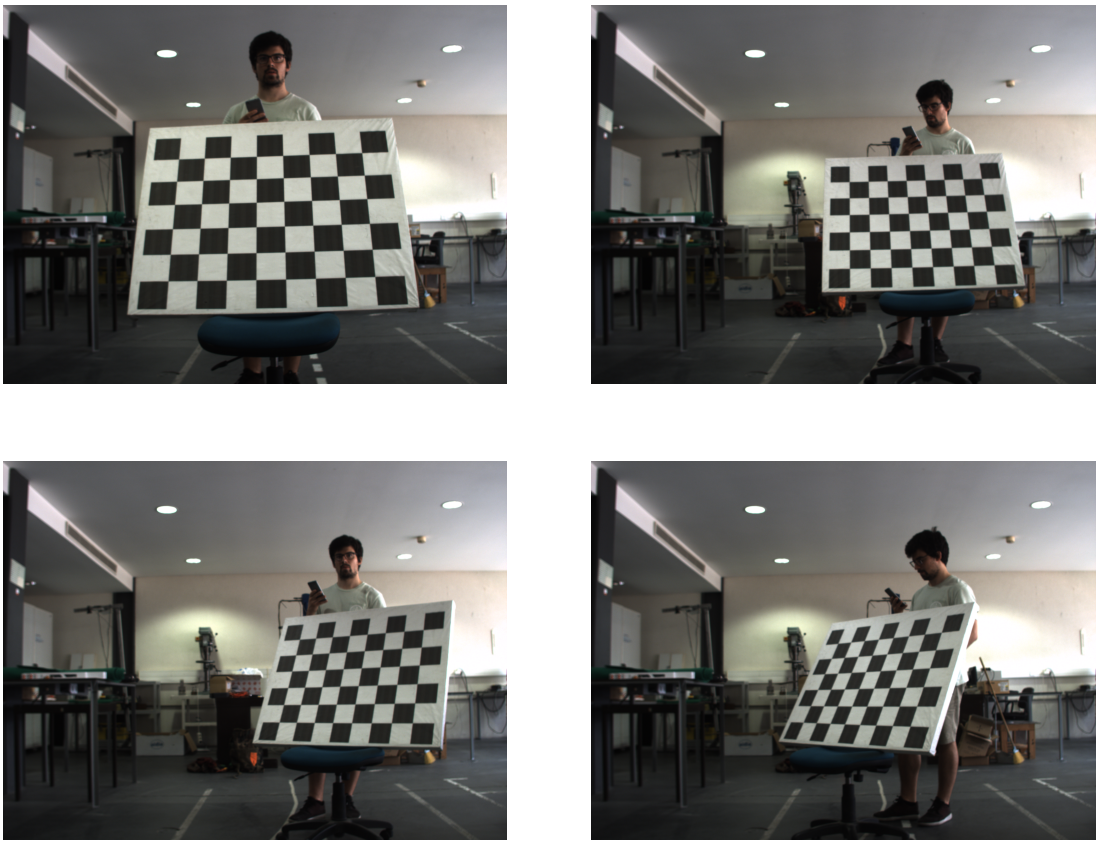


Figure 5.2: Images captured for RADLOCC.

point correspondences to compute the calibration. In this work, this method was used together with the extrinsic calibration method (explained in Section 6.1.3), to obtain the full extrinsic laser calibration. As a result, this calibration obtains the transformation marked in red in Figure 5.1.

To use this method, the user has to obtain a calibration dataset, which is a set of synchronized images and laser scans containing a chessboard in multiple poses. The chessboard serves as the calibration object, which is the link between the two sensors. In this work, a ROS package was developed to handle this capture and to convert between the ROS messages and the RADLOCC format. The laser scanner was positioned such that the laser scans are horizontal and about 20 to 30 images were taken per dataset. Such images are shown in Figure 5.2.

First, a chessboard extraction algorithm finds both the intrinsic calibration of the camera, as well as the poses of each chessboard in the camera coordinate frame. Then, laser scans are segmented into board and background, and all the points measured on the board are extracted, as seen on Figure 5.3.

Then, the reprojection error of the laser scans points, obtained in the segmentation, to the chessboard plane,obtained by the pose estimation, are calculated, depending of the transformation from the camera to the laser scanner. This reprojection error is minimized during the calibration optimization, and the transformation from the camera to the laser if found.

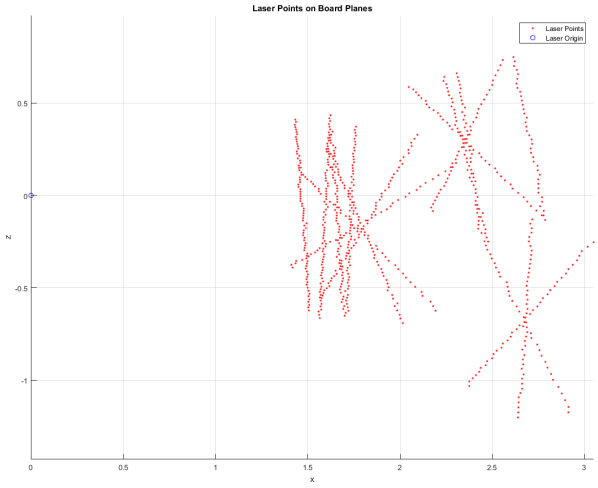


Figure 5.3: Radlocc laser scans chessboard extraction.

Finally, the extrinsic calibration can be calculated as:

$$T_{calibration} = {}^{laser}_{PTU}T = {}^{camera}_{PTU}T \cdot {}^{laser}_{camera}T. \tag{5.3}$$

This calibration method, however, was not able to obtain an accurate result.

5.2.2 Planar Based Calibration

A alternative method, the Planar Based Calibration, was developed in this work to calibrate the laser scanner in this system. One of the key differences to the previous method (in Section 5.2.1) is that the calibration is done only using the laser range data, and the calibration from the PTU to the laser scanner is directly found (the transformation marked in blue in Figure 5.1), so no camera is required. This method supposes that, in a good calibration, the deviation of a point set is minimal. In other words, in a point set representing a planar surface, the deviation from the points to the planar surface should be the lowest, if the extrinsic calibration is correct.

This method is, therefore, an optimization problem. For each extrinsic calibration transformation $\mathcal{T}$, corresponds a point cloud $\mathcal{P}$, following the method shown on Section 5.1. This point cloud is evaluated by a cost function, which determines quantitatively how good each generated point cloud is. Finally, an optimizer will find the transformation T that minimizes the loss function. Each one of these steps is described in detail next.

Segmentation

This calibration method uses an acquisition as its dataset, which is a significant advantage, since no calibrations patterns and no special apparatus is required, like chessboards or other markers. Also, a point cloud has to be generated using a estimation of the calibration transformation. This point cloud does not have to be geometrically accurate but the geometry should be perceivable for the plane segmentation, which is done manually prior to the calibration. In this work, the software CloudCompare was used to segment the point cloud into multiple planes, and the data was saved as a scalar index in each point. An example of

a segmentation can be seen in Figure 5.4, where each cluster is represented with a different color.

The segmentation was done manually because most segmentation algorithms, for example the RANSAC algorithm, were not capable of achieving a reliable segmentation for the initial estimate, because the point cloud had significant deformation. In addition, manual segmentation is easy to do and accurate, considering that it is a one-time process.

During the optimization, this segmentation serves as a blueprint for all the segmentations. Each point cloud is generated in the same way, so the sequence of points is always the same. Therefore, it is always possible to match any point on the generated point cloud to the point in the segmented point cloud, and get the corresponding cluster index for all the points.

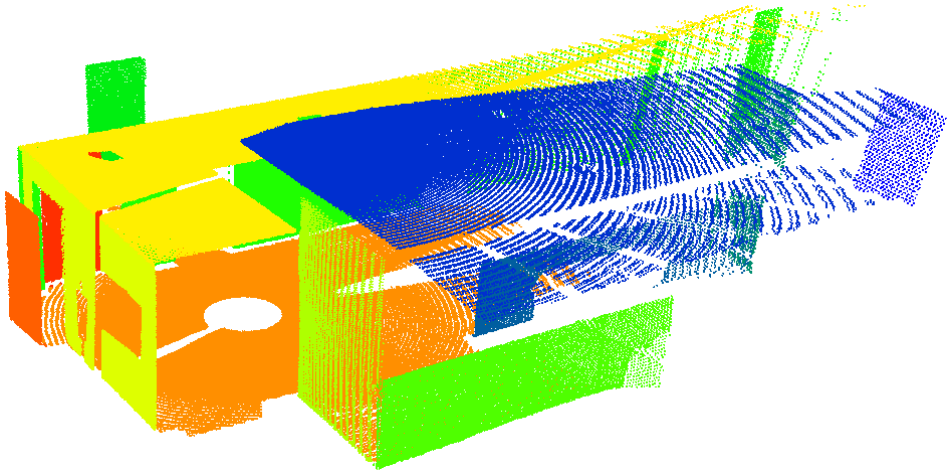


Figure 5.4: Example of a plane segmentation, where each color represents a cluster.

Cost Function

The cost function is a measure used in optimization that compares the result of a model with its expected result, and returns a value that describes the dissimilarity between the two. More concretely, in this calibration the cost function has two steps: the cost is computed for each cluster and then the cost of all the clusters is combined into a single value, which is the final cost of the point cloud.

In an inicial step, the plane equation for each cluster is computed, using the Principal Component Analysis method, or PCA. First the centroid $\bar{p}$ of each plane is found, which is the same as the mean value of all the points $p : (x, y, z) \in \mathbf{R}^3$:

$$\bar{p} = \sum_i p_i. \tag{5.4}$$

Then, the covariance matrix $\mathcal{C}$ is calculated:

$$\mathcal{C} = \sum_i (p_i - \bar{p}) \otimes (p_i - \bar{p})^1. \tag{5.5}$$

[1] $\otimes$ is the outer tensor product.

Then, the principal axes of the plane is find by an eigen decomposition of the covariance matrix. The smallest eigenvalue λ_3 will be the variance σ^2 of the cluster. In other words, σ^2 is the mean square of the orthogonal distance of all points in the cluster to the plane. So, σ^2 can be a quantitative factor to measure the cost or each cluster. Formally, let us admit that the σ^2 has two components: the statistical error of the laser sensor σ^2_{sensor}, which is not affected by the calibration and a second component σ^2_{calib}, which depends of the calibration error. Thus, the idea is that, by minimizing σ^2, a exact calibration can be obtained. For this calibration, however, the value σ was used instead of σ^2, which is known as the Root Mean Square Deviation, or RMS. Therefore, the loss of each cluster will be the σ value.

Next, the scores of the clusters are combined into a scalar value, which is the error of the point cloud. The method found was to, again, calculate the RMS of the values of the partial losses loss_i, according to Equation (5.6). This value is expected to be minimal when all the partial losses are minimal which, according to this hypothesis, corresponds to a correct calibration.

$$RMS = \sqrt{\sum_{i}^{N} \text{loss}_i^2} \tag{5.6}$$

Paramerization

The parameters in this calibration are six values that define a geometric transformation in space, which is, in the end, a transformation matrix T (Equation (5.7)). This transformation can be decomposed into two components, a translation and a rotation. The translation can be represented as the vector $t = (t_x, t_y, t_z)$, and the rotation can be represented as a 3×3 rotation matrix R. Since a rotation matrix has only $3 \times 3 = 9$ elements but only 3 degrees of freedom, another parameterization has to be used to represent a rotation. Popular parameterization for rotations are euler angles, quaternions and axis/angle representation.

$$T = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.7}$$

However, not all representations are suitable for an optimization. In fact, in [HT99] the term fair parameterization was introduced: a parameterization is called fair, if it does not introduce more numerical sensitivity than the one inherent to the problem itself. Therefore, fair parameterization are a requirement for optimizations, as it increases the chances of convergence. For example, euler angles, which are probably the most used angle parameterization, are not suitable for optimizations [SN01], because they do not yield smooth movements, each rotation is non-unique and, most notably, there are singularities, so-called *gimbal-lock* singularities, where one degree of freedom is lost [SN01]. Also, quaternions are not suitable for optimizations, because quaternions have 4 components which are constrained to an unitary length. Despite being a fair parameterization, quaternions introduce some complexity in the algorithm to handle this constrain, so they are not usually used for optimizations [SN01].

The axis/angle parameterization is the most widely used to represent a rotation in an optimization, as it is a fair parameterization and has only three components. Any rotation can be represented as a rotation around an axis a, by an angle θ. Since a only represent the

direction of the rotation (hence only has 2 degrees of freedom), it can be combined with the angle θ into a single vector $\omega = (\omega_1, \omega_2, \omega_3)$, as in Equation (5.8).

$$\begin{aligned} \theta &= |\omega| \\ a &= \frac{\omega}{|\omega|} \end{aligned} \tag{5.8}$$

Computing the rotation matrix from ω is done using the Rodrigues' formula (Equations (5.9) and (5.10)) [SN01]:

$$R = I + \frac{\sin\theta}{\theta}[\omega] + \frac{1 - \cos\theta}{\theta^2}[w] \tag{5.9}$$

$$[w] = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix}, \tag{5.10}$$

where I is the 3×3 identity matrix, θ is the angle and ω_1, ω_2 and ω_3 are the components of ω.

In conclusion, the parameter vector will have 6 values: 3 representing the translation (t_1, t_2, t_3) and 3 representing the rotation in the axis/angle representation (r_1, r_2, r_3). So, the parameter vector is shown in Equation (5.11).

$$P = \{t_1, t_2, t_3, r_1, r_2, r_3\} \tag{5.11}$$

5.2.3 First Guess

This optimization was quite robust to the initial parameters, so the first guess was always a null translation and the rotation was done doing a visual inspection of the laser scanner, using angles multiples of 90°.

Optimizer

The optimization is performed using the Powell's method, described in [Pow64]. This method finds a local minimum of a multi-dimensional unconstrained function, and does not require the gradient of this function (is unknown in this problem), which fits this particular optimization. This method is implemented in the python scientific library SciPy[2].

Overview

To sumarize, the overview of the entire steps of the calibration is shown in Figure 5.5.

[2]See the Scipy reference in `https://docs.scipy.org/doc/scipy/reference/optimize.minimize-powell.html`.

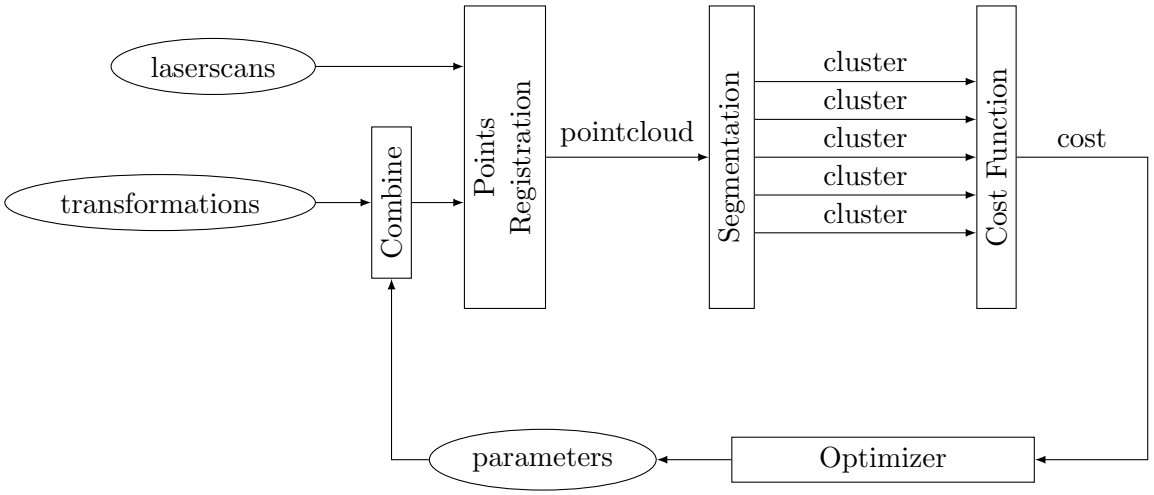


Figure 5.5: Calibration Overview.

5.3 Normal Estimation

As part of the reconstruction work, it is common to create a mesh using the point cloud, through a process called triangulation. Despite that in this work, this was not performed, it can be done in posterior work. Surface normals are a important property of geometric surfaces and are a requirement for most triangulation algorithms. Also, normals are required for lightning calculation, which can improve the rendering of a point cloud model[3]. As an example, the Stanford Bunny model[4] rendered with and without lightning are show in Figure 5.6. As can be seen, lightning can improve the perception of the geometry of the point cloud model.

Normal estimation is quite trivial for surfaces, but for point clouds the process is quite not as easy. Usually there are two ways to estimate the normals: either by meshing the surface first, and then calculate the normals for the mesh, or using the point cloud itself to infer the normals. However, most meshing algorithms already require the normals to achieve a good result, so the latter option is more effective.

The most common solution is, for each point, to find the k closest points, defined as the k-neighborhood of a point, and calculate the normal of the best-fitting plane formed by this points. However, finding the k-neighborhood of all the points in a point cloud has a time complexity $O(N \log N)$, so it can become quite slow for point clouds with a large number of points. In this work, an alternative solution was used to find the closest points, exploiting the bidimensional structure of the point cloud. This solution has a linear time complexity $O(N)$, which makes it a valuable solution for large point clouds.

The solution uses the fact that each point in the point cloud resulting from Section 5.1 has two indexes, one for the range index i and one for the laser scan j. For each laser scan, each point p_i has a neighborhood $p_{i-k}, \ \ldots, \ p_{i+k}$, because each subsequent point has an increasing angle to the previous point. Between successive laser scans each point has an increasing angle (the pan angle) to the previous one. Therefore, for this algorithm, the neighborhood of each

[3]See "Estimating Surface Normals in a PointCloud" in `http://pointclouds.org/documentation/tutorials/normal_estimation.php`.

[4]From `https://www.cc.gatech.edu/~turk/bunny/bunny.html`.

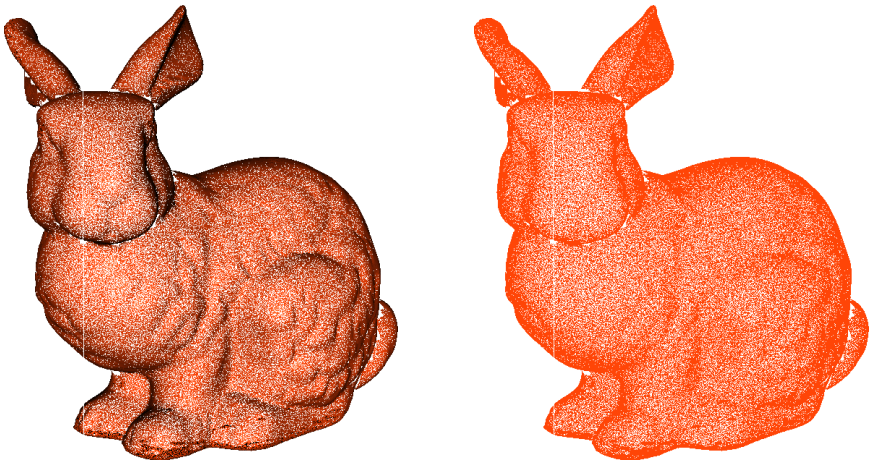


Figure 5.6: Stanford rabbit rendering with lightning (on the left), using the normals information, and without lightning (on the right).

point:

$$neighborhood(p_{i,j}, k_1, k_2) = \{p_{i-k_1,j-k_2},\ \ldots,\ p_{i+k_1,j+k_2}\}. \tag{5.12}$$

The value of k_1 and k_2 have to be adjusted for a better result, because if the values are large, fine details are going to disappear and edges are going to be smeared, and on the other hand if the values are small, the surface will appear as too noisy. In this work, the value of k_1 and k_2 was 3, so the neighbor has 9 points.

Then, for each point, the tangent plane that fits the neighborhood is calculated, which in turn is a least-square plane fitting problem. This is usually solved by an analysis of Principal Component Analysis, as explained in Section 5.2.2. This method will compute the direction of the normal n for each point.

Then, the orientation of the point has to be defined, because the result of the PCA is ambiguous, which may lead to inconsistent normals in the point cloud. In this case, the solution found was to orientate the normals towards the frame of the 3D scanner, which for each acquisition in the origin of the coordinate system. Therefore, each normal has to satisfy:

$$n \cdot p < 0. \tag{5.13}$$

5.4 Registration of Acquisitions

During a capture, multiple acquisitions are performed and to each one corresponds a transformation (position and orientation) to the scene referencial. In this section, a method is described to find each one of this transformations, so all the acquisitions are merged into a single point cloud. The method chosen is the Iterative Closest Point, or ICP. This method is capable of aligning two point clouds, the reference and the target point cloud, by finding the transformation between the second to the first one. This is also known as point cloud registration.

5.4.1 Iterative Closest Point

Iterative Closest Point, or ICP, is a method which finds the transformation between two pont clouds: a reference point cloud and a target point cloud, by minimizing the distance between correspondent features, as points, lines or edges, found in both point clouds. A further explanation of the process can be found in [BM92].

This method can be divided in two steps: the correspondence or matching steps, which finds the common features between the point clouds and the iterative optimization step, which find the transformation though an optimization algorithm. Multiple improvements can be made to the algorithm, by choosing different optimizations or correspondent algorithms. A common modification, for example, is to remove the outliers features which have no correspondence.

A common problem of this method is an imperfect correspondence, which compromises the final result. This is greater if the point clouds are not dense and if there is small overlap between them. In general, point-to-point correspondent is not very robust. To overcome this problem, it is common to use more robust features, like visual features, for example markers, or geometric features, for example corners.

In this work, a simple ICP method with a point-to-point correspondence algorithm with outlier removal was used.

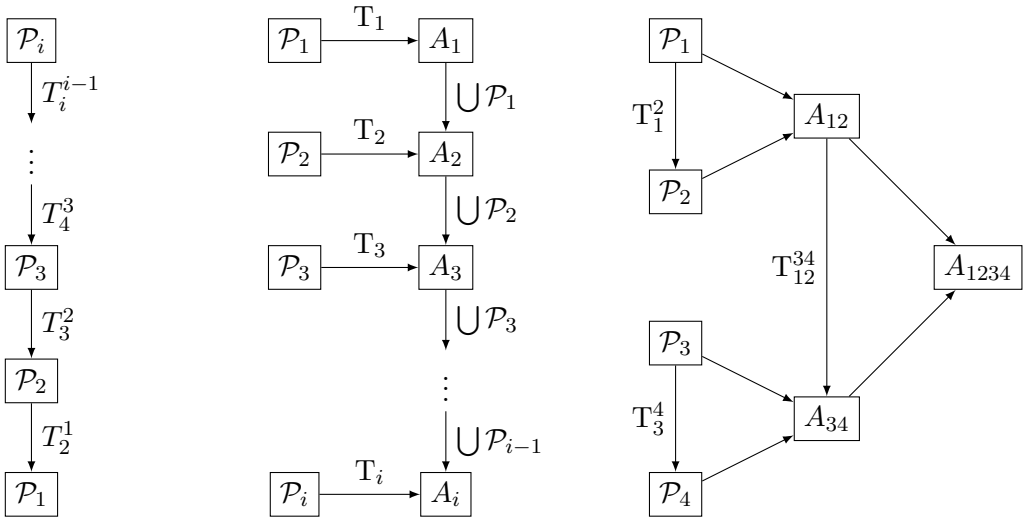
5.4.2 ICP for Multiple Point Clouds

ICP can only register pairs of point clouds, whereas this work requires a registration of N point clouds, corresponding to n acquisitions. So, a technique has to be found so that the ICP algorithm can be used with n point clouds. Three of this techniques are now described, ordered by their complexity:

- The first approach and the easier one to implement is to register each point clouds sequencially. In other words, this method registers the point cloud $\mathcal{P}_i$ to the point cloud $\mathcal{P}_{i-1}$ and the transformation T_{i-1}^{i} is found. The final accumulated point cloud is assembled using the Equation (5.14). This method is the one that requires less overall registration, but has the disadvantage that the accumulative transformation errors increases for each successive point cloud. This approach is shown in Figure 5.7.

$$\mathcal{P} = \bigcup \left(\mathrm{T}_1^2 \circ \mathrm{T}_2^3 \circ \cdots \circ \mathrm{T}_{i-1}^{i}\right)(\mathcal{P}_i) \tag{5.14}$$

- The next approach is widely used in robotics for Simultaneous Location and Mapping , or SLAM. This method holds an accumulated point cloud A in memory, and each new incoming point cloud $\mathcal{P}$ registers to the accumulated point cloud. Afterwards it is merged into A, which is then used for the next iteration, as shown in Figure 5.7. It has the advantage that each new registration is done against a wider point cloud so there is more overlapping between the point clouds. Also, at each iteration the current pose of the 3D scanner is obtained, which is used as an initial estimate for the next iteration. However, the accumulated point cloud grows at each iteration, and some filtering has to be performed to maintain the number of points bounded. In conclusion, each iteration can be calculated as:



(a) First method. (b) Second method. (c) Third method.

Figure 5.7: Multiple Point Cloud ICP approaches.

$$\mathrm{T}_i = \mathrm{ICP}(A, \mathcal{P}_i, \mathrm{T}_0 = \mathrm{T}_{i-1}), \tag{5.15}$$

$$A_{i+1} = Ai \bigcup \mathrm{T}_i(\mathcal{P}_i). \tag{5.16}$$

- The last approach is the most complex. The idea of this approach is to minimize the number of transformation combinations, to minimize the propagation of the error. In particular, the registrations for the N points clouds are done pairwise and are merged together to create a new point cloud. Then, this process is done recursively until an unique point cloud is obtained. This way, the maximum number of transformation combinations are equal to the number of levels of the tree, which is $\log_2(N)$, instead of N combinations in the first approach. This algorithm is formalized in Equations (5.17) and (5.18), for a list of point clouds $S = \{P_1, P_2, \ \ldots, \ P_n\}$. At each level l a new list of point clouds ${}^lP$ and transformations ${}^lT$ are computed, as shown in Figure 5.7:

$${}^l\mathrm{T} = \left\{ \mathrm{ICP}({}^{l-1}\mathcal{P}_1, {}^{l-1}\mathcal{P}_2), \ \ldots, \ \mathrm{ICP}({}^{l-1}\mathcal{P}_{n-1}, {}^{l-1}\mathcal{P}_n) \right\}, \tag{5.17}$$

$${}^l\mathcal{P} = \left\{ {}^{l-1}\mathcal{P}_1 \bigcup {}^l\mathrm{T}_1({}^{l-1}\mathcal{P}_2), \ \ldots, \ {}^{l-1}\mathcal{P}_{n-1} \bigcup {}^l\mathrm{T}_{n/2}({}^{l-1}\mathcal{P}_n) \right\}. \tag{5.18}$$

In conclusion, three methods are possible to extend the ICP algorithm to multiple point clouds, and the three methods were used in this work and compared. After this registration the point clouds are assembled into the final point cloud. There is, however, a limitation of all this methods, because all of them have the principle that every point cloud is close to the previous one, which can be false. In this work, this was ensured in the capture methodology.

5.5 Filters

The final point cloud, after the assembly from every acquisition's point cloud, can have unnecessary or redundant information, which can make the point cloud size very large. A common solution is to use filters to remote unnecessary points and downsample the point cloud.

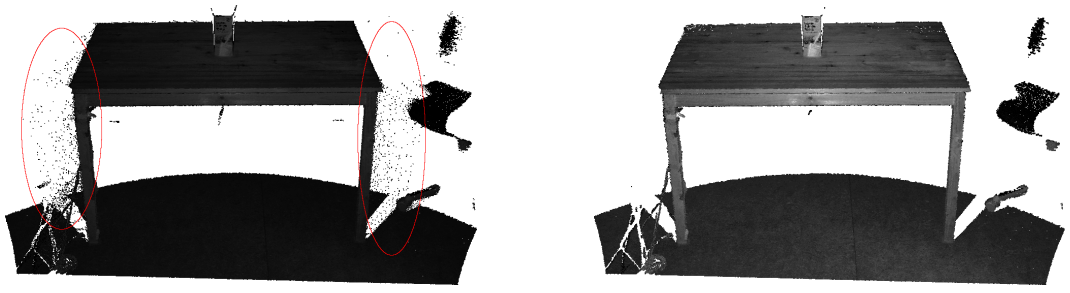
5.5.1 Not a Number Removal

The first filter is the Not a Number removal, or NaN removal. In the first steps of the reconstruction, the point cloud is stored as a dense point cloud, or structured point cloud. To maintain this structure, NaNs are used to mark the missing values, which are usually originated by measurement errors. When the structure of the point cloud becomes irrelevant, its dimensions are collapsed into one. After, the NaNs become irrelevant and are removed from the point cloud.

In the acquisition, any range that is not measured is stored as a NaN, to signal that they are missing. During the point registration phase, all this missing ranges remain as NaN, and should be removed, because their information is irrelevant and take as much space as a real value. So, each point that contains a NaN value is removed from the final point cloud.

5.5.2 Statistic Outlier Removal

Usually point clouds contains different point densities, dependent on the distance of the object to the sensor. Also, measurement errors also occur next to edges or corners. As a result, point clouds tend to have outliers that can affect subsequent algorithms, like segmentation or registration algorithms. A usual solution is to perform an statistically analysis on each point, removing the points that do not reach a certain criteria. In particular, the mean distance of each point to its neighbors is computed, and if this distance is above or below an interval centered in the mean of all the distances, then it is removed. An example can be seen in Figure 5.8.



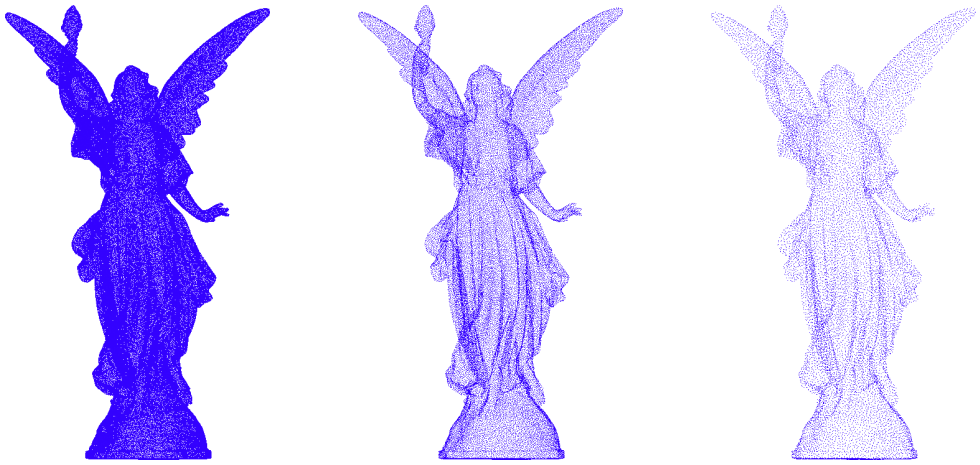
(a) Before SOR. (b) After SOR.

Figure 5.8: SOR filter in a point cloud, processed by the software *CloudCompare*.

5.5.3 Voxel Grid Downsampling

This method downsamples, that is, reduce the number of points of a point cloud, using a voxel grid. A voxel is a cubic volumetric volume and is the element of a tridimensional grid.

So, each point in the point cloud belongs to some voxel. Then, in each voxel, all the points are represented by their centroid. This is an effective and fast method to downsample a point cloud. The level of detail can be parameterized with the voxel leaf-size (the size of each voxel in the x, y, z direction). A smaller leaf-size maintains more details but generates a larger point cloud. A larger leaf-size does not keep as much detain but generates a smaller point cloud. As an example, Figure 5.9 shows the Stanford Lucy model[5] after a voxel grid downsampling with different leaf size values: Figure 5.9a with 2 mm (288.000 pts), Figure 5.9b with 5 mm (55.000 pts) and Figure 5.9c with 2 mm (18.000 pts).



(a) Leaf size of 2 mm. (b) Leaf size of 5 mm. (c) Leaf size of 8 mm.

Figure 5.9: Stanford Lucy scan after a voxel grid downsampling with different leaf sizes.

[5]From "Stanford Scanning Repository" in `graphics.stanford.edu/data/3Dscanrep/`.

Chapter 6

Methodology for Image Registration

This chapter describes the methodology for image registration, that is, the process that colorizes (defines the colors) the point cloud based on the images taken in the acquisitions. This method can be split into two parts: the Color Registration (Section 6.1), where the process is described per-image and each image colorizes a portion of the point cloud, and the Color Fusion (Section 6.2), where all the colorized point cloud are merged into the final colorized point cloud. The pixel registration relies on a camera calibration, both the intrinsic calibration and also the extrinsic, so two methods are shown to obtain this calibration (Sections 6.1.2 and 6.1.3).

6.1 Color Registration

This method describes how to colorize a point cloud based on a single image, using projection principle based on the camera model. As an overview, each point in the point cloud can be projected as a ray in the camera perspective, which is basically the path from the eye point to the point. This ray can be used to retrieve the original color of the point from the image. However, this process is not so straightforward, because the position and orientation of the camera has to be very precise and occlusion has to be considered.

6.1.1 Point Projection

To start with, each point has to be transformed, because the original point is registered in the scene coordinate frame (p_{scene}) and has to be transformed into the camera coordinate frame (p_{camera}). So, the transformations ${}^{acquisition}_{scene}T$, ${}^{ptu}_{acquisition}T$, ${}^{camera}_{ptu}T$ can be used according to Equation (6.1). The ${}^{acquisition}_{scene}T$ transformation is obtained in Section 5.4, ${}^{ptu}_{acquisition}T$ is the transformation of the PTU and ${}^{camera}_{ptu}T$ is the extrinsic calibration of the camera and the method to obtain it is in Section 6.1.3. The transformation graph can be seen in Figure 6.1.

$$p_{scene} = {}^{acquisition}_{scene}T \cdot {}^{ptu}_{acquisition}T \cdot {}^{camera}_{ptu}T \cdot p_{camera} \tag{6.1}$$

Next, each point was transformed into pixel coordinates (u, v), using the pinhole camera model. This model defined how a light ray in projected in the image sensor of a camera and has two parameters: the focal length $f = (f_x, f_y)$ and optical center $(c = (c_x, c_y))$. This

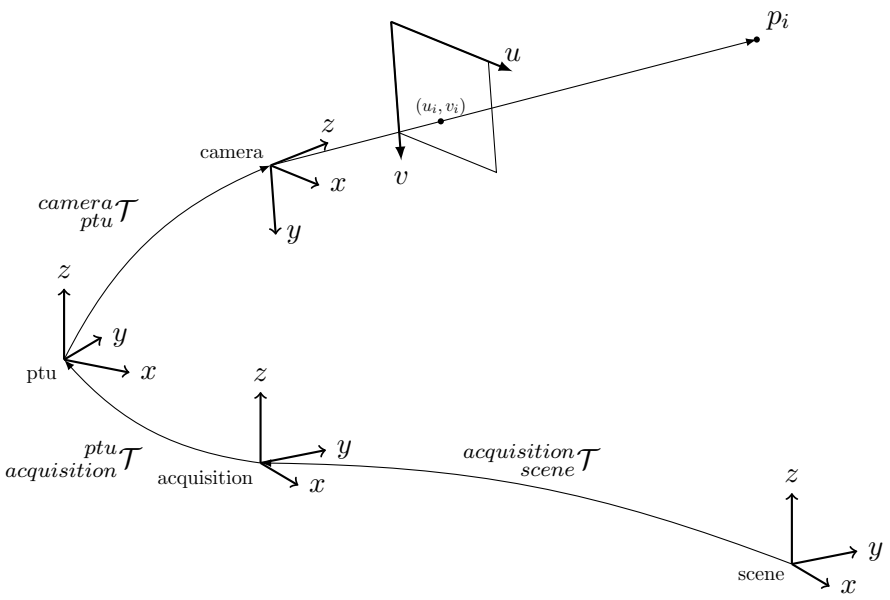


Figure 6.1: Color registration for a single point.

parameters are obtained in the intrinsic calibration of the camera (Section 6.1.2). According to this model, each point is projected as pixel coordinates (u, v) to a plane located a unit distance from the camera eye point, using the perspective projection matrix in Equation (6.2), according to:

$$\mathcal{P} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}, \tag{6.2}$$

$$\begin{pmatrix} uz \\ vz \\ z \end{pmatrix} = \mathcal{P} \cdot \begin{pmatrix} x \\ y \\ z \end{pmatrix}. \tag{6.3}$$

The pinhole camera model does not regard the distortion caused by the lens, which is not negligible for most cameras. The two sources of distortion are radial and tangential distortion. Radial distortion makes straight lines appear curved, known as the barrel distortion and pincushion distortion. This distortion is highly noticed in images taken with fish-eye lenses, as seen in Figure 6.2. This distortion can be solved by transforming the (u, v) with Equation (6.4). Similarly, tangential distortion is caused by a misalignment of the lens to the imaging plane, which causes areas in the image to appear closer than expected. This deformation can be solved with the Equation (6.5). In brief, to undistort the image five parameters need to be determined, also known as the distortion coefficients: $\{k_1, k_2, p_1, p_2, k_3\}$, which are obtained in the camera intrinsic calibration method, described in Section 6.1.2.

$$\begin{pmatrix} u \\ v \end{pmatrix}_{calibrated} = (1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \begin{pmatrix} u \\ v \end{pmatrix}. \tag{6.4}$$

$$\begin{pmatrix} u \\ v \end{pmatrix}_{calibrated} = \begin{pmatrix} u \\ v \end{pmatrix} + \begin{pmatrix} 2p_1 uv + p_2(r^2 + 2u^2) \\ p_2(r^2 + 2v^2) + 2p_2 uv \end{pmatrix}. \tag{6.5}$$

Figure 6.2: Barrel distortion in fish eye lens.

6.1.2 Camera Intrinsic Calibration

The intrinsic calibration determines the intrinsic parameters of the camera. The calibration procedure used in this work is a standard procedure for cameras with low distortion and is known as the chessboard camera calibration. This method calibrates a monocular camera with fixed focus using a sequence of images taken from a chessboard with known dimensions.

In order to improve the calibration results, the chessboard should rotate and move, in order to occupy the entire image size and the chessboard poses should be enough and be well distributed spatially. Also, the calibration is more accurate if the corners of the chessboard are well defined in the image, so the chessboard should have an appropriate size

In the end, the accuracy of the calibration should be measured for new images, with the re-projection error. This value should be as low as possible and, as a rule of thumb, a value less than 0.01 is acceptable.

In ROS, this calibration is easily obtained with the *cameracalibrator.py*, which includes a graphical interface, and provides feedback about the corner detection and the state of the calibration. The interface is shown on Figure 6.3. In this system, this data is first saved into a ROS *camera_info* file. Then, this file is also saved in each capture in the parameters file.

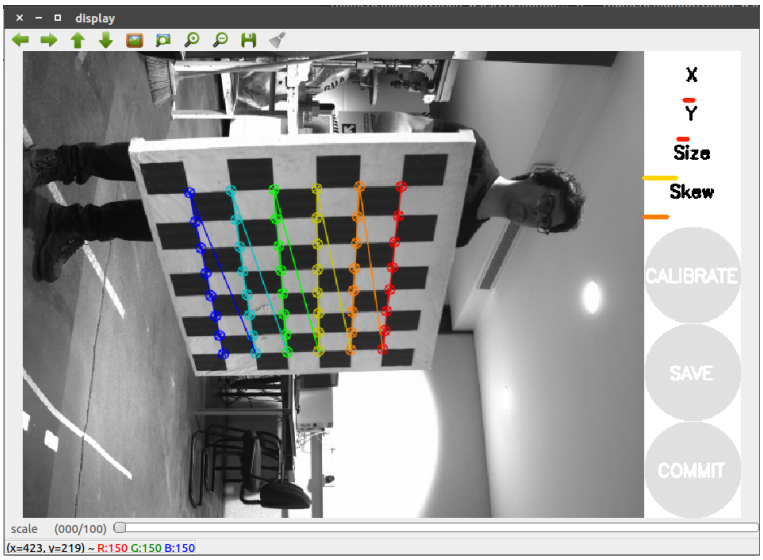


Figure 6.3: Interface for the *cameracalibrator* node.

6.1.3 Camera Extrinsic Calibration

The calibration method used to determine the extrinsic parameters is known as the eye-in-hand calibration, described in [HD95].

This calibration relies on a static calibration object, whose pose can be estimated in the camera frame. Hence, four coordinate frames and four transformation exist. The four frames are the *camera* frame, the *world* frame, the *PTU* frame and the *object*. The four transformations are the extrinsic transformation of the camera, or the *PTU* to the *camera* transformation ${}^{camera}_{ptu}T$, which is static and unknown, the *camera* to *object* transformation ${}^{object}_{camera}T$, which is obtained by the object pose estimation algorithm, the *world* to *PTU*, which is known and, finally, the *world* to *object* transformation, which is static and unknown. The overall transformation graph is shown is in Figure 6.4, with the unknown transformations in red and the known transformations in green.

The inspection of the transformation graph determines an equality, because there are two possible ways to transverse the graph from one node to another, which yields the Equation (6.6). This equality is the base of this optimization: ${}^{camera}_{ptu}T$ can be obtained from multiple pairs of synchronized ${}^{object}_{world}T$ and ${}^{object}_{camera}T$ transformations.

$$ {}^{object}_{world}T = {}^{ptu}_{world}T \cdot {}^{camera}_{ptu}T \cdot {}^{object}_{camera}T \tag{6.6} $$

In this work, the object used for detection was an ArUco marker, which is comprised of a pattern which can be detected and also allows for precise pose estimation, as seen in Figure 6.5. One of the biggest advantages over other markers is that the implementation for detection and pose estimation is already implemented in the ROS package *aruco_detect*. The calibration is also implemented in the ROS package *visp_hand2eye_calibration*, as a node that receives multiple transformations in the topics */world_effector* and */camera_object*, which correspond respectively to the ${}^{ptu}_{world}T$ and ${}^{object}_{camera}T$ transformations. To publish the transformations on this topics, a node was developed, the *hand2eye_simple_client*, which publishes both the transformations synchronously at the keypress of the user. The control of the PTU

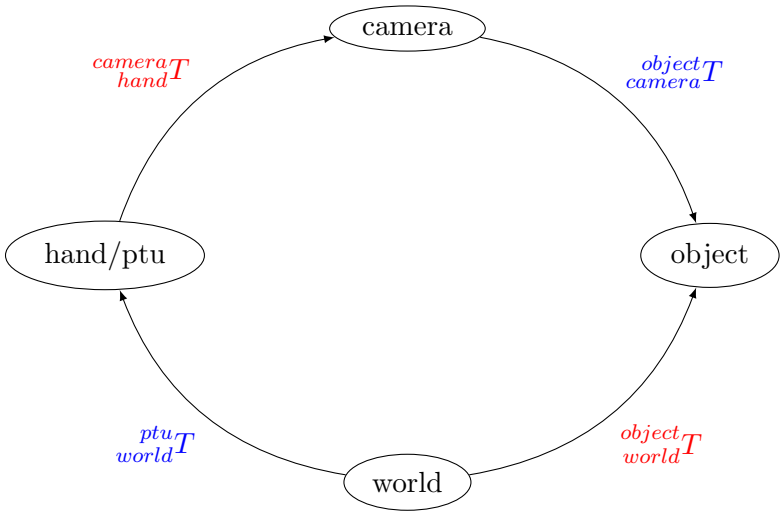


Figure 6.4: Hand-in-eye transformation graph.

was also manual.

6.1.4 Point filtering

Not all points are eligible for the color registration, based on it's location and camera properties, so two filtering steps were used: the first filter removes the points outside the field of view and the second removes the occluded points.

Field of View Removal Filter

The field of view is defined as the region of space that is captured by the camera sensor, which for pinhole cameras has a pyramid geometry, as seen in Figure 6.6. The sides of the pyramid are limited by the size of the sensor, so the points that lie outside the rectangle defined by the points $(0, 0)$ and $(width, height)$ are excluded, as:

$$\begin{aligned} 0 &< u < width \\ \wedge\ 0 &< v < height. \end{aligned} \tag{6.7}$$

Hidden Point Removal Filter

Not all points that lie on the frustum of the camera are seen by the camera, because some of this points are occluded by nearer objects, so they need to be removed. A fast and straightforward solution is to use the point cloud resulting from the same acquisition as the image, because the sensor are considered close together. However, this is not the best solution, as it would be better if the point cloud obtained after the acquisition registration was used.

In [KTB07], an simple and fast operator, the Hidden Point Removal, or HPR, determines the visibility of point sets, viewed from a given viewport. This method is easily implemented and has a asymptotic complexity of $O(n \log n)$, where n is the number of points in the point cloud. Moreover, this method works well for both sparse and dense point clouds.

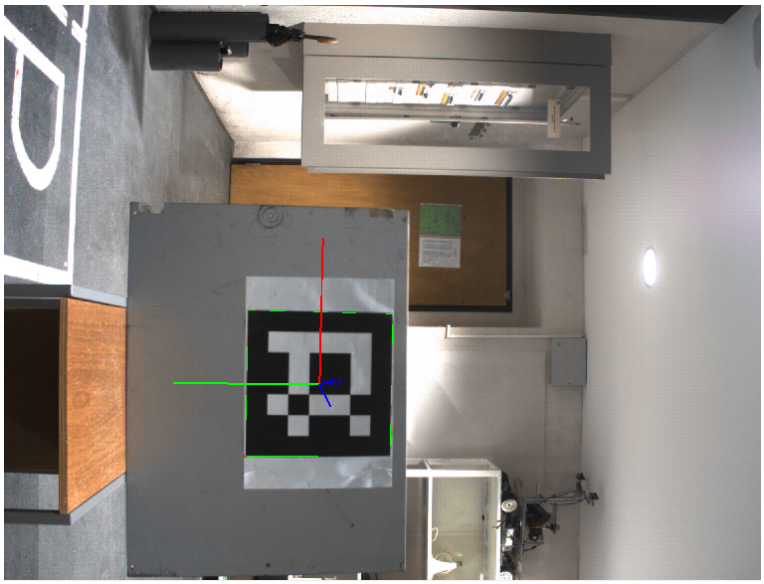


Figure 6.5: ArUco maker detection and pose estimation.

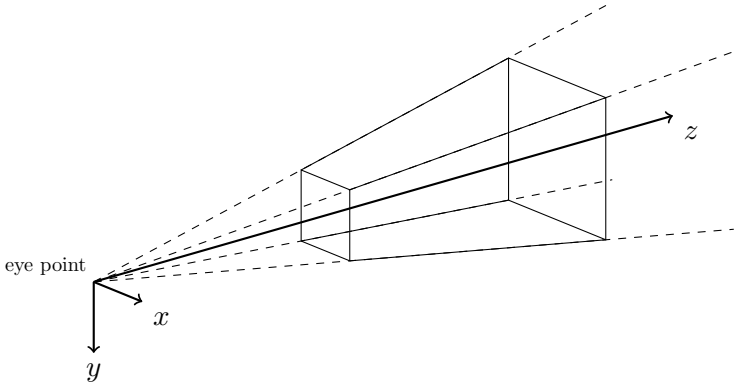


Figure 6.6: Representation of the visual frustum of the camera.

The HPR operator operates on a set of points $\mathcal{P} = \{p_i | i = 1 \dots n\}$, and the goal is to determine whether p_i is visible from a viewpoint $\mathcal{C}$. In this application, C is the origin of the point cloud. The algorithm consists of two steps: the inversion and the construction of the convex hull.

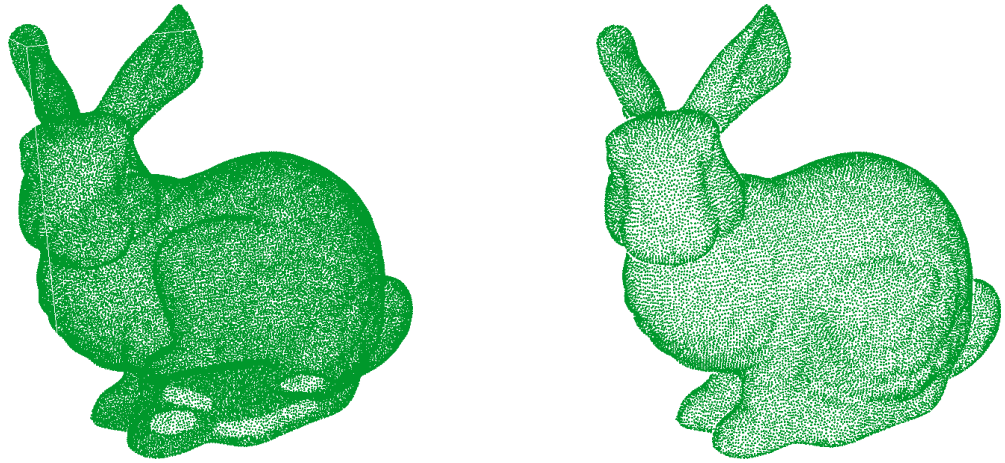
The inversion step maps each point p_i along the ray from $\mathcal{C}$ to p_i, such that $|p_i|$ is monotonically decreasing. There are multiple ways to perform the inversion, but in [KTB07] the *spherical flipping* was used. Spherical flipping reflects a point p_i with respect to a sphere of radius R to the new point $\hat{p_i}$ by applying:

$$\hat{p_i} = p_i + 2(R - |p_i|)\frac{p_i}{|p_i|}. \tag{6.8}$$

Afterwards, the convex hull of $\hat{\mathcal{P}} \bigcup \{\mathcal{C}\}$, where $\hat{\mathcal{P}}$ is the transformed point set and $\mathcal{C}$ is the center of the sphere, is computed. Finally, the points that lie in the convex hull are the visible points of the point set.

This algorithm only has a parameter, which is the radius R of the sphere used for the spherical flipping, which influences the amount of false positives of the algorithm. In general, R is determined based on the maximum point length $max(|p_i|)$ and a exponential factor α, such that $R = max(|p_i|) \times 10^{\alpha}$. In this application, a factor of $\alpha = 3$ was empirically selected.

As an example, the HPR operator was used in the Stanford Bunny[1] point cloud, as seen in Figure 6.7 and, as seen, Figure 6.7b only presents the points that are visible, as opposed to Figure 6.7a.



(a) Before the HPR. (b) After the HPR.

Figure 6.7: Result of the HPR operator in the Bunny point cloud.

6.1.5 Color Attribution

Finally, the color is selected from the image at the pixel coordinates (u, v) and saved for the correspondent pixel. Because images are discrete, the color is interpolated using a bilinear interpolation, which uses the neighbor pixels to interpolate the color C at (u, v) in an image I according to Equation (6.9) (the ceil and floor operators are, respectively, $\lceil \cdot \rceil$ and $\lfloor \cdot \rfloor$). The interpolation can be visualized in Figure 6.8.

$$\begin{aligned} C(u,v) = &(u - \lceil u \rceil)\,(v - \lceil v \rceil)\; I_{\lfloor u \rfloor, \lfloor v \rfloor} \\ &+ (u - \lceil u \rceil)\,(v - \lfloor v \rfloor)\; I_{\lfloor u \rfloor, \lceil v \rceil} \\ &+ (u - \lfloor u \rfloor)\,(v - \lceil v \rceil)\; I_{\lceil u \rceil, \lfloor v \rfloor} \\ &+ (u - \lfloor u \rfloor)\,(v - \lfloor v \rfloor)\; I_{\lceil u \rceil, \lceil v \rceil} \end{aligned} \tag{6.9}$$

6.2 Color Fusion

In a capture with N_a acquisitions, each one with N_i images, the total number of images account to $N_a \times N_i$. Each one of this images will yield a partial colorized point cloud, according to Section 6.1, and the point clouds need to be merged into a final point cloud. More specifically, each point p_i has multiple correspondent colors, one for each registered

[1] From https://www.cc.gatech.edu/ turk/bunny/bunny.html.

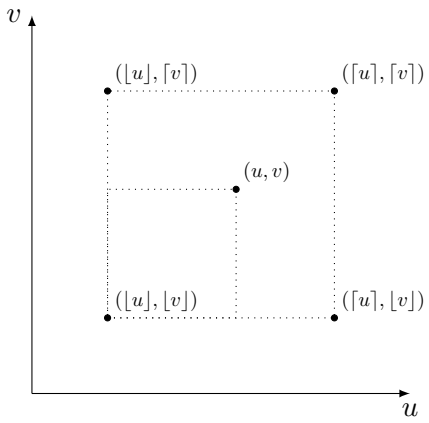


Figure 6.8: Bilinear interpolation in an image.

image. The method here described determines the final color in a point-wise fashion and does not account for the neighbor points.

Let us admit that the point p has a set $C = \{c_i | i = 1 \dots k\}$ of k registered colors. The final color of this point c should be a combination of the colors in C.

The first approach is to colorize the point with one image only, for example, the first or last image. This method is the easiest and the faster, but does not consider the other images for the colorization.

The second approach is to average the colors to obtain the color c, as seen in Equation (6.10). However, this is a poor heuristic as it considers that all colors have the same error, which is not true. For example, an image taken closer to an object is more precise than one taken away from it.

$$c = \frac{1}{k} \sum_{i}^{k} c_i \tag{6.10}$$

A common solution for the mean limitation is to use an weighted mean, shown in Equation (6.11). The w_i are the weights for each color and should reflect the quality of each color, because colors with larger weight have a larger influence in the final color.

$$c = \frac{\sum_i^k w_i c_i}{\sum_i^k w_i} \tag{6.11}$$

In this work, the quality measurement was determined based on an heuristic that depends on two factors, that are obtained in the color registration phase (Section 6.1).

The first factor f_1 depends on the distance d from the camera to the point and on the optimal focus point d_f. f_1 is smaller the bigger the distance between d and d_f. The function used was the gaussian centered on d_f. The second factor f_2 depends on the distance from the pixel coordinates (u, v) to the center of the optical center (c_x, c_y). Again, a gaussian distribution was used to calculate f_2, and a bigger distance also yields a smaller f_2. In brief, both factors f_1 and f_2 are calculated according to Equations (6.12) and (6.13). The parameters α and β determine how wide the gaussian function is, so points farther from the peak point influence more or less.

$$f_1 = \exp\left(-\frac{(d-d_f)^2}{2\alpha^2}\right) \tag{6.12}$$

$$f_2 = \exp\left(-\frac{(u-c_x)^2+(v-c_y)^2}{2\beta^2}\right) \tag{6.13}$$

$$\tag{6.14}$$

The two factors are then combined into the weight w factor of the color, based on a linear combination, dependent on a parameter s, which determines the influence of each factor, as seen on Equation (6.15).

$$w = sf_1 + (1-s)f_2 \tag{6.15}$$

In conclusion, for each point p_i the color c_i is attributed, based on the registered colors of each image. The fusion of all this colors is based on a weighted mean, where the weight of each color is determined by an heuristic that considers the location of the color in pixel coordinates and the distance of the point to the camera, in order to benefit points that have a better quality in the measurement, for example, points that are in focus or points that are closer to the camera center. This process is repeated for all the points of the point cloud until every point has a color (however, some points have no color registered, because no color was registered before).

Chapter 7

Results

In order to evaluate the proposed methods, multiple acquisitions were taken from the department of Mechanical Engineering at the *Universidade de Aveiro*.

7.1 Dataset Description

In total, six captures were taken from three hallways of the department, using all the three lasers installed in the 3D scanner, one at each time. The hallways were chosen because of their large area and structured environment, with big and flat surfaces, making it easier to inspect the geometric quality of the scans. Also, small objects, like chairs, tables and doors can be found, as well as unstructured objects, like trees, can be found, which can be required to evaluate the color registrations. All the six captures can be found in Table 7.1.

Table 7.1: Captures obtained to test the proposed methods.

	Scene	Laser	#Acquisitions
1	Second Floor Hallway	Sick LMS100	7
2	Third Floor East Hallway	Sick LMS100	12
3	Third Floor East Hallway	Hokuyo URG04	9
4	Third Floor West Hallway	Hokuyo URG04	10
5	Second Floor Hallway	Hokuyo UTM30	6
6	Third Floor East Hallway	Hokuyo UTM30	5

7.2 Geometric Reconstruction

The geometric reconstruction is the first part of the 3D reconstruction and uses the laser scanner data and the PTU transformation to obtain the non-colorized point cloud. This reconstruction relies on the extrinsic calibration of the laser to register the laser scans precisely, which is described in detain in Section 5.2. Moreover, a new method to obtain the normals of the points was developed (Section 5.3), as well as the methodology to register spatially

the multiple acquisitions (Section 5.4). At the end of this registration, a point cloud resulted from all the acquisitions should be obtained.

7.2.1 Extrinsic Laser Calibration

The extrinsic calibration of the laser scanner is one of the main factors that influenced the geometric registration, because a bad calibration results in a deformed point cloud, as seen in Figure 7.1. In this work, two calibration methods were used: one pre-existing method called RADLOCC and one method developed in this work, which attempts to be more accurate than the previous method.

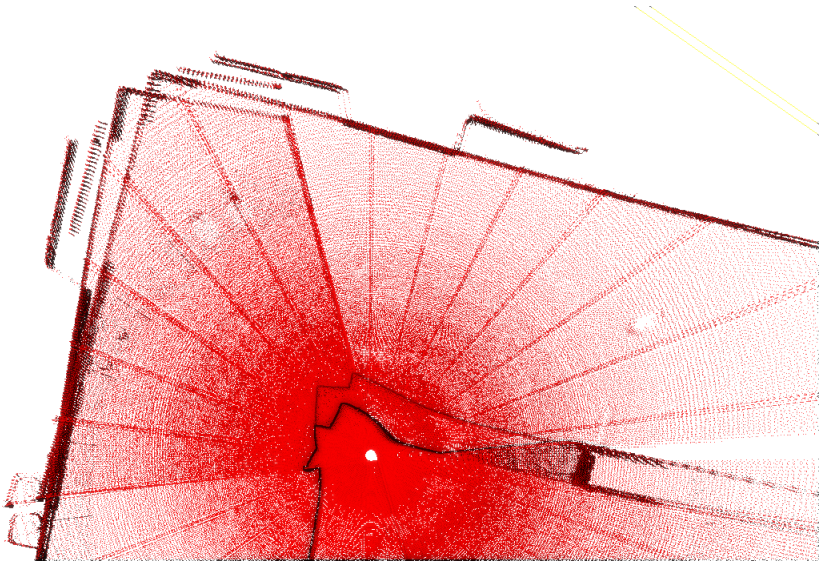


Figure 7.1: Uncalibrated point cloud of the capture 3.

RADLOCC

The RADLOCC calibration, described in Section 5.2.1, was the first method used to obtain the extrinsic calibration of the laser scanner. The evaluation of this calibration can be done by the re-projection of the laser scans onto the images, where the edges of the laser scan should be coincident with the edges of the chessboard, as seen in Figure 7.2. In total, six calibration datasets were obtained using the SICK LMS100 sensor, which have around 20 to 40 images and laser scans pairs. The results obtained are shown in table Table 7.2.

It was expected to see similar results along the calibration, because the datasets were taken with similar conditions. However, there are large variations: for example, the translation on the x axis between the dataset 2 and 3 has a difference of around 0.07 m. This is not a negligible difference and, in the end, this can affect the geometry of the point cloud.

In conclusion, the resulting transformations have a large deviation between calibration, both in rotation and translation. This, associated with the fact that the full extrinsic calibration requires the extrinsic calibration of the camera, which also has significant error, justifies that this method is not suitable or capable for this application.

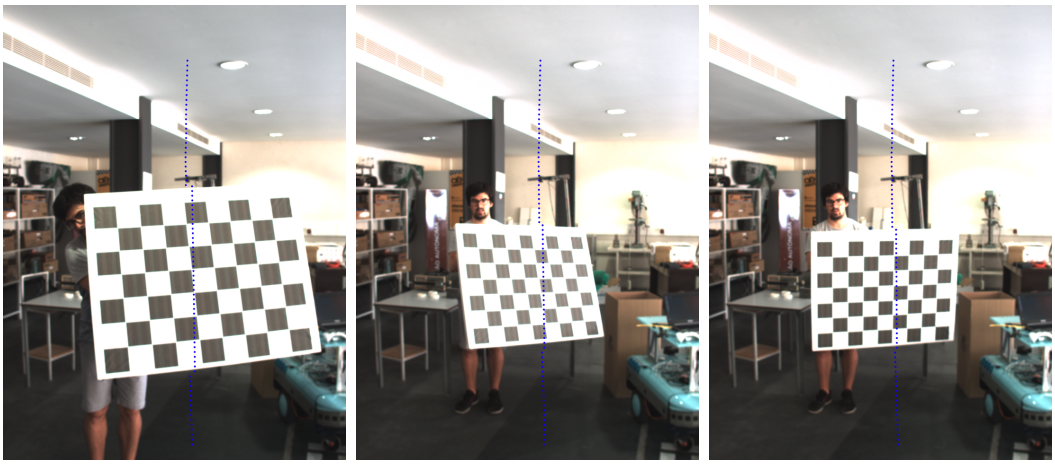


Figure 7.2: Reprojection of the laser scans in the images obtained in the RADLOCC calibration method.

Planar Based Calibration Method

The proposed method for the calibration uses the same data as the one used for the geometric reconstruction. The first step is the manual segmentation of the planes. At the end of this step, each point has a correspondent integer index, which is called the cluster index. In figure Figure 7.3, a segmented point cloud can be seen, correspondent to the capture 3. Specifically, the deformation of this point cloud results can be as seen in Figure 7.4.

After the segmentation, the calibration score is minimized to obtain the extrinsic calibration of the laser scanner, as described in Section 5.2.2. For the initial estimate, a rough transformation was obtained via visual inspection. The result of a single acquisition is shown in Figure 7.6. The initial score was 3.9×10^{-3} and after 10 steps the score was reduced to about 1.1×10^{-4}. The resulting point cloud at each step can be seen in Figure 7.5. The time required to this calibration is about 20 minutes.

The resulting calibration does indeed improve the geometry of the point cloud, shown before. In Figure 7.7, the resulting point cloud (in red) can be compared to the previous point cloud, obtained by the initial estimate. As can be seen, the deformation visible before are no longer present in the calibrated point cloud.

In general, this calibration method successfully yields similar results for different acquisitions and captures. In other words, the calibration is repeatable. For example, the transformations obtained for a set of acquisitions of the capture 5 has similar results across the acquisitions, as shown in Table 7.3.

As seen, the standard deviation σ in this calibration is much less that in the RADLOCC calibration. For example, in the translation in the x axis, in RADLOCC $\sigma \approx 0.02\,\text{m}$, while in this calibration it was $\sigma \approx 0.002\,\text{m}$, which is approximately ten times less.

Therefore, the proposed calibration can be a reliable method to obtain the extrinsic calibration of laser scanners mounted in motion platforms, as the PTU, because transformation obtained has the accuracy required for the geometric registration of the laser scanners for this work and the results have repeatability and reproductivity. The advantages of this method, compared to the previous method, are:

1. The method directly obtains the extrinsic transformation of the laser scanner, instead

Table 7.2: Resulting extrinsic calibration obtained using the RADLOCC method.

		translation			rotation			
Dataset	#Images	x	y	z	x	y	z	w
1	28	-0.0145	0.0435	-0.0385	0.7129	-0.0024	0.7008	-0.0236
2	36	-0.0242	0.0521	-0.0926	0.7153	-0.0082	0.6987	0.0059
3	38	0.0493	0.1823	-0.0538	0.7113	0.0252	0.7005	0.0497
4	52	0.0190	0.0388	-0.0561	0.7111	0.0005	0.7030	-0.0077
5	15	-0.0058	0.0850	-0.0739	0.7183	0.0032	0.6956	-0.0016
6	19	0.0072	-0.0192	-0.0334	0.7291	0.0247	0.6834	-0.0245
7	14	0.0009	0.0699	-0.0692	0.7126	0.0003	0.7013	-0.0130
8	22	0.0212	0.0373	-0.0641	0.7190	0.0074	0.6948	-0.0131
μ[1]		0.0066	0.0612	-0.0602	0.7162	0.0063	0.6973	-0.0034
σ[2]		0.0217	0.0539	0.0180	0.0056	0.0115	0.0059	0.0223

[1] μ is the mean of the results.
[2] σ is the standard variation of the results.

Table 7.3: Extrinsic calibration obtained using multiple acquisitions.

	translation			rotation			
Acquisition	x	y	z	q_x	q_y	q_z	q_w
1	-0.00157	0.1195	0.1053	-0.5015	-0.5151	-0.5056	0.4770
2	0.00401	0.1103	0.0932	-0.5006	-0.5175	-0.5035	0.4776
3	0.00301	0.1222	0.0982	-0.5001	-0.5168	-0.5042	0.4782
4	0.00434	0.1321	0.0988	-0.5014	-0.5158	-0.5059	0.4761
μ	0.00245	0.12103	0.09888	-0.50090	-0.51630	-0.50480	0.47723
σ	0.00237	0.00777	0.0043	0.00058	0.00092	0.00099	0.00078

of a partial transformation (to the camera). This also decreases the error that would come from the intrinsic and extrinsic calibration of the camera.

2. This calibration uses exclusively the laser scanner data, which is more accurate than images, because it is a direct measurement, as opposed to the transformation resulting from a pose estimation algorithm. Moreover, the number of laser scans used are also greater than in the RADLOCC method, which can explain the smaller error found in this calibration method.

3. This calibration acknowledges and benefits from the movement of the PTU, as opposed to the RADLOCC, which was designed for static laser scanners. The laser scanner has to be static in RADLOCC, so that the background segmentation of the laser scans is possible.

4. The method proposed uses the data from the acquisitions directly, so the calibration process can be done even after the capture is made, or if the laser scanner is not available. This can be useful if the prior calibration did not have the accuracy required

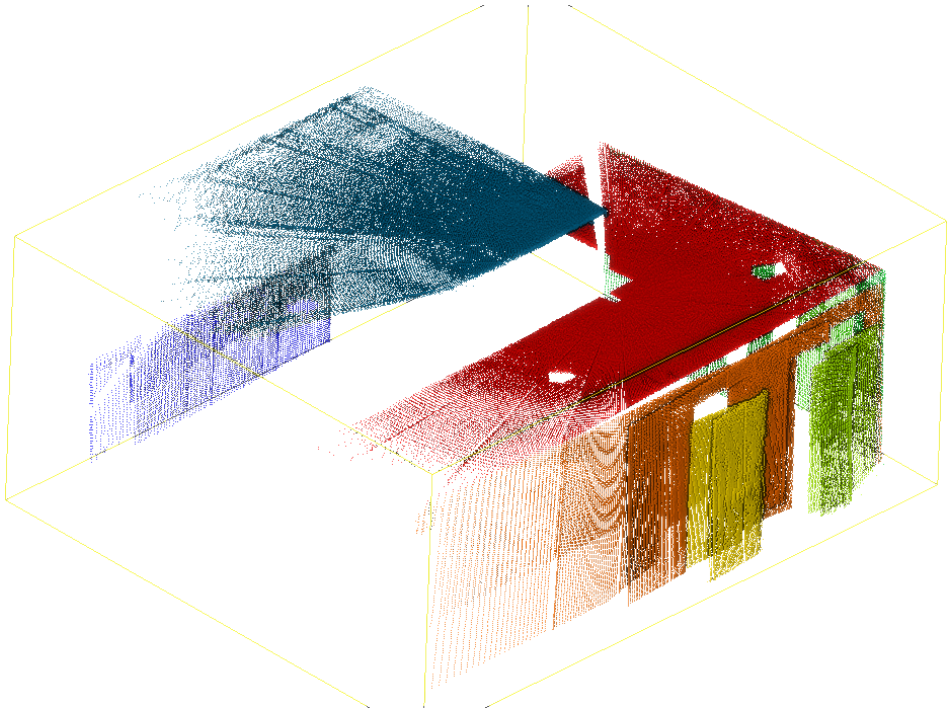


Figure 7.3: Segmented point cloud from capture 3.

for the acquisition, for example, if the calibration was done using a smaller scene, or if the equipment has changed or is not available. The only requirement is that there are enough planes in the scene for the calibration.

7.2.2 Normal Estimation

In this section, the results obtained from the Normal Estimation method described in Section 5.3 are shown and compared to the more traditional method using the k-nearest-Neighbors. Both methods have similar results, as shown in Figure 7.8, however, there are differences:

1. The proposed method relies on a continuous movement in pan, while recording the laser scans. However, this was not the case for the acquisitions, because the movement was interrupted to record the images. This can explain why some points have the wrong normals. However, this limitation can be surpassed if the movement of the PTU is continuous.

2. This method can only be used for each acquisition and can not be generalized for any point cloud, because it required that the point cloud is structured in a 2D structure. The other method, however, works for any point cloud.

3. The computational complexity of the proposed method is $O(N)$, while the complexity of the method using the k-Neighbors is $O(N \log N)$, which has an increasing impact for large point clouds. For example, in a point cloud with 5 million points (for example, in captures 5 and 6), the time required to calculate the normals using the proposed

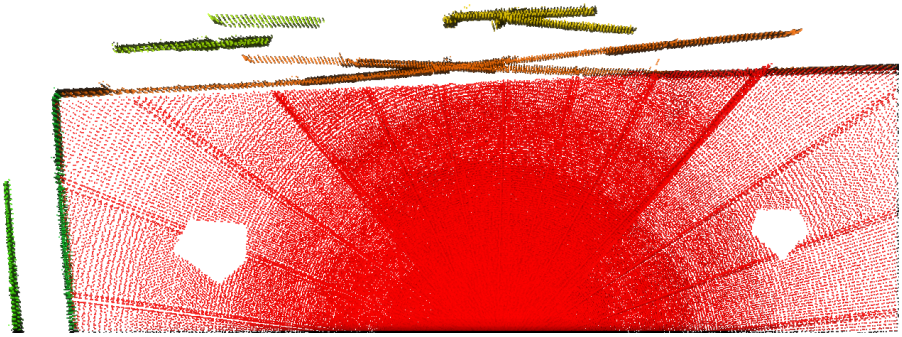


Figure 7.4: Detail of the segmented point cloud from capture 3.

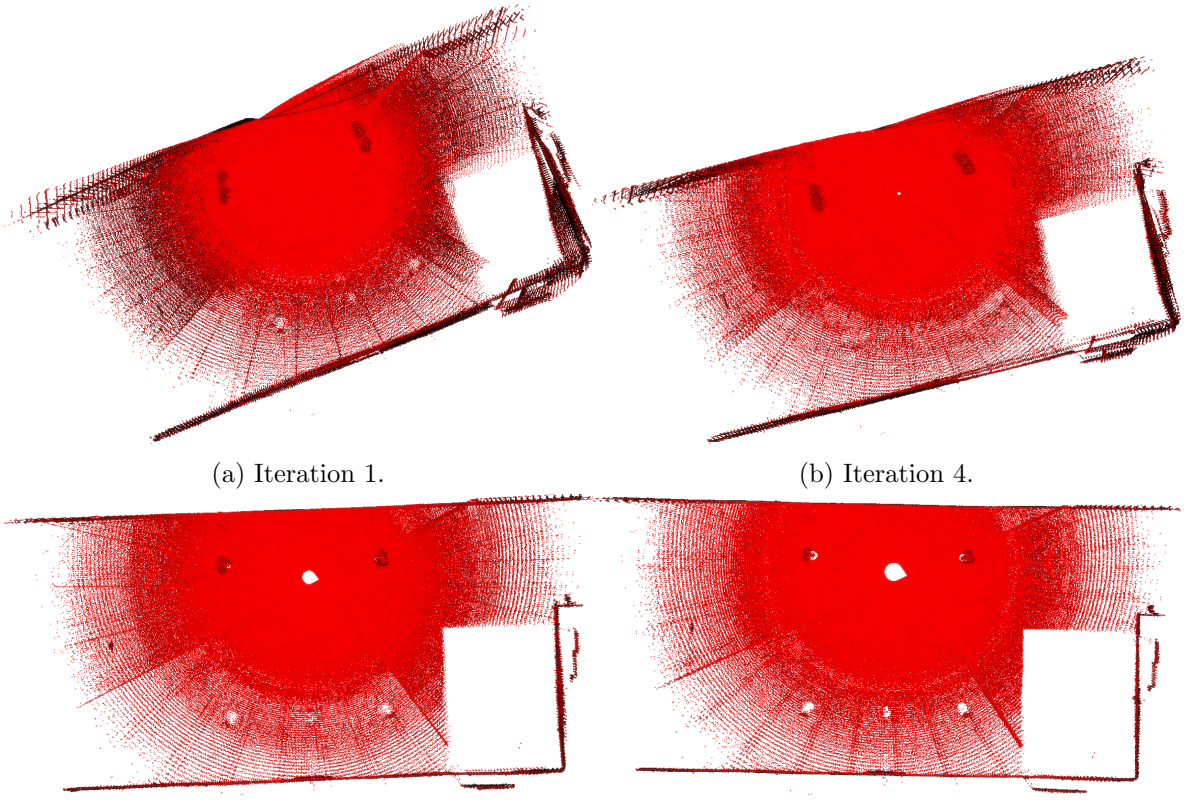
method is 10 s (using a implementation in python, which is regarded as a slow language for numerical computations) and using the k-neighbors method the time required is about 10 min.

7.2.3 Acquisition Registration

The method for the acquisition registration uses the ICP algorithm, which find the transformation between two point clouds by the minimization of the distance between the point clouds. However, finding the closest points in the two point clouds is a difficult task and can be wrong if two point clouds are far apart. In particular, the point clouds transformation is the distance between the acquisition poses, which is in the order of meters. The ICP method, however, is only successful if the manual alignment is already very close. Any noticeable misalignment results in a wrong registration, as seen in Figure 7.9. The solution found was to manually align the point clouds first, and then use the ICP algorithm as a fine alignment. This solution is sub-optimal, as it is not automatic, thus requiring manual work. These shortcomings in the acquisition registration can be solved by either enhancing the ICP algorithm to improve the registration for large differences, or by providing the inicial estimate for the transformation between each acquisition.

Another disadvantage of the ICP algorithm is that the registration is pairwise. In this work, multiple point clouds were registered, and the inability of the ICP to register more than two point clouds at once implies that the registration does not use the full spectrum of data available, and is constrained to only the two point clouds at each registration. This suggests that a registration which is done between two point clouds with small overlap, which results in a weaker registration, could be avoided. If the algorithm used the entire data set, the registration would be done with the most overlap possible, and number of common points is maximized. To circumvent this problem, the three alternative methods are shown in Section 5.4.

However, only the second method was proven to be effective in this work. This method registers the point clouds to an accumulated point cloud resulted by the fusion of the prior



(a) Iteration 1. (b) Iteration 4.

(c) Iteration 5. (d) Iteration 7.

Figure 7.5: Resulting point cloud at each iteration in the optimization process.

registered point clouds. The success of this method can be explained by the increasing overlap existent in further registrations. The other methods fail because there are point clouds that have so small overlap that the ICP method always fail. For example, in the Figure 7.10, two registrations are done, where the target point cloud is colorized in red. As seen, the registration fails in the pairwise registration (against other point cloud in green), but is successful when the reference point cloud is the accumulated point cloud (in green).

In conclusion, the ICP algorithm is not sufficient for an automatic registration of the acquisitions. However, with manual intervention as a first registration, the ICP algorithm can be used to find the fine registration of the acquisitions. Using this method, the registrations were possible, as shown in Figure 7.9.

7.2.4 Influence of the different laser scanners

In this work, three different laser scanners were used to study the influence of the different characteristics of each one on the reconstruction process.

The aperture of the laser and the range are important factors in the capture process. As an example, the Hokuyo URG04 has a small range of 5.6 m, compared to the larger range of the two other lasers, which both have a range of 30 m. The resulting point clouds are, then, much smaller in the case of the Hokuyo URG04, which means that more acquisitions are required to capture a scene, and the distance between each acquisition should not be larger that the range of the laser scanner. Otherwise, there is no overlap between acquisitions so

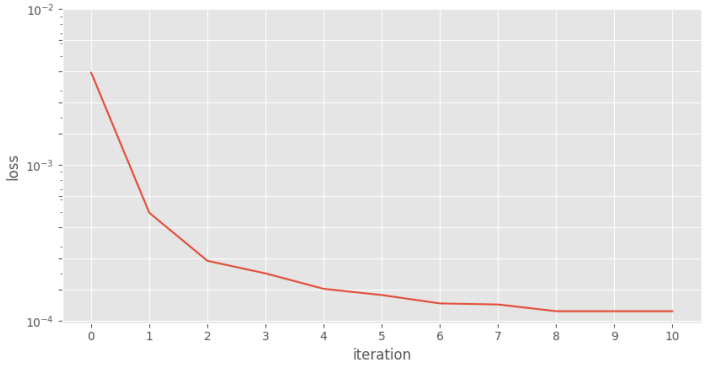


Figure 7.6: Calibration iteration results for the third acquisition of the capture 4.

the acquisition registration would fail. The same can be applied to the aperture of the laser scanner.

Also, the response of the same surfaces differ from laser to laser, mostly due to the wavelength and energy of the laser used. As an example, the floor surface, which was made of tiles with reflective material, was only properly registered in the SICK LMS100 laser scanner, which is the laser with the most power output. It is expected to be due to the high reflection of the surface, only a small fraction of the laser beams is reflected back to the laser scanner. If the laser emitted does not have considerable power, the reflected light could not have sufficient energy to be detected by the sensor.

Moreover, the scanning rate of the laser scanner can be important to obtain dense point clouds, without requiring a large acquisition time. The Hokuyo UTM30 was the laser scanner with the higher scanning rate (40 Hz), which produced point clouds with about 4 times the number of points obtained by the two other lasers, without sacrificing the acquisition time.

Lastly, a problem was identified in the SICK LMS100 laser which made it unsuitable for this application, because the laser scans obtained were not accurate, compared to the other sensors. The laser scans of planar surfaces appear as deformed in this laser scanner. This can be seen in Figure 7.12, where both the floor and the root are not shown as curved lines. Therefore, the resulting point clouds were always deformed, even after many extrinsic calibrations. This is most probably a problem of this laser scanner, and it is suspected to be due to a damaged laser scanner. Therefore, the acquisitions of this laser are not considered further on.

7.2.5 Overall Results

The results of the geometric registration are satisfactory, mostly because of the new calibration method of the laser, which ensured the success all the geometric algorithms used further on, such as the normal estimation and the acquisition registration. The overall results are shown in Figures 7.13 and 7.14.

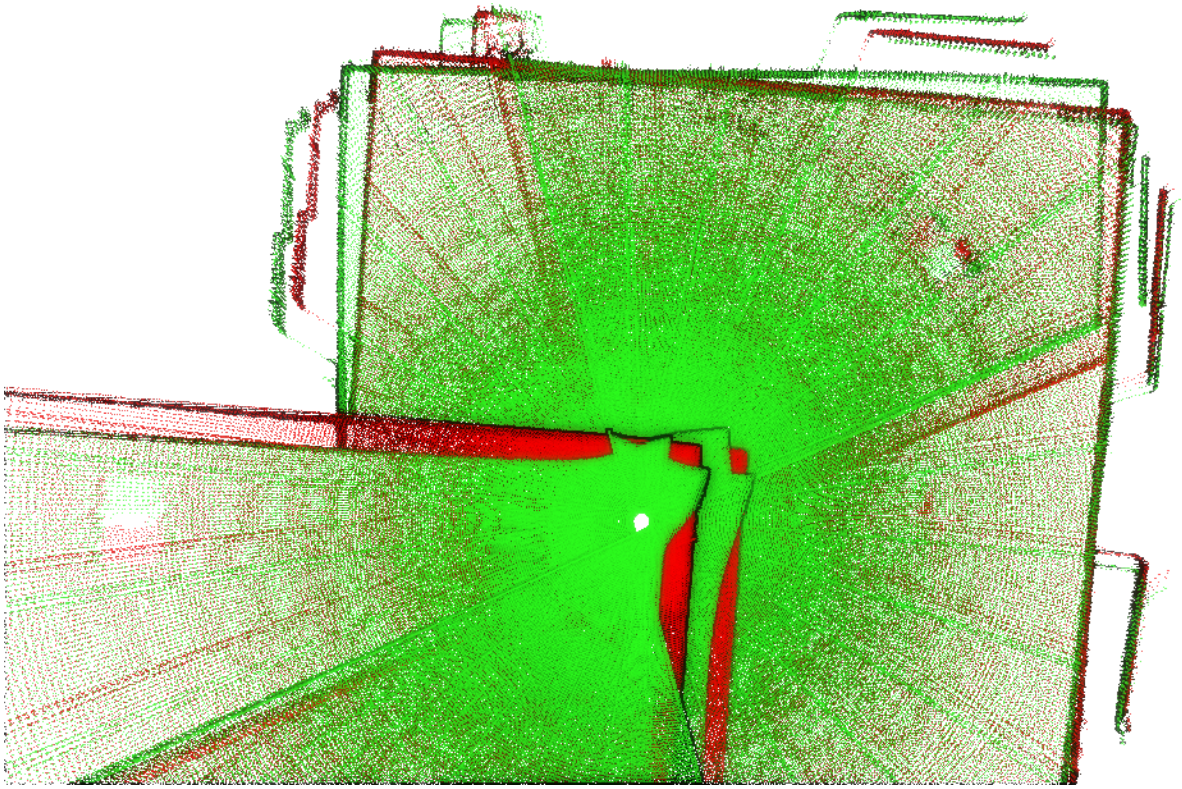


Figure 7.7: Comparison between the calibrated point cloud (in red) and the non-calibrated point cloud (in green).

7.3 Color Reconstruction

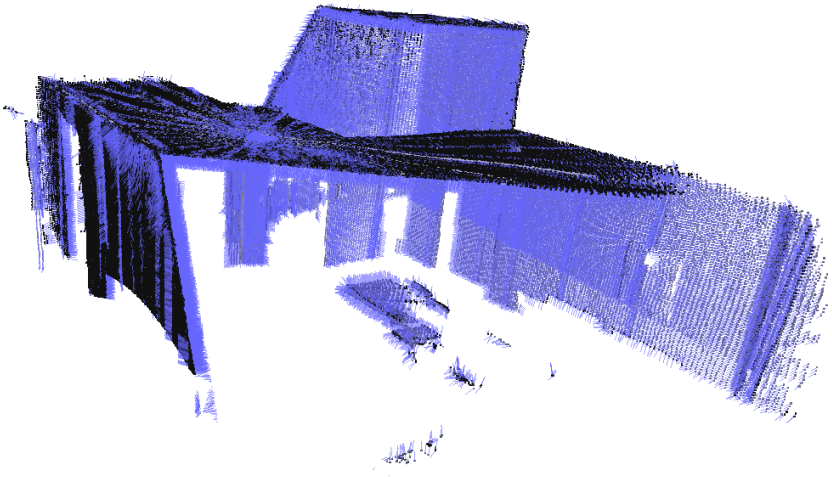
The color reconstruction is the final part of the reconstruction, which uses the images taken from the camera and registers the color into the point cloud obtained in the geometric reconstruction and merges the colors to obtain a colorized point cloud. This methods require both the intrinsic and extrinsic calibration of the camera, to register the images correctly.

7.3.1 Camera Intrinsic Calibration

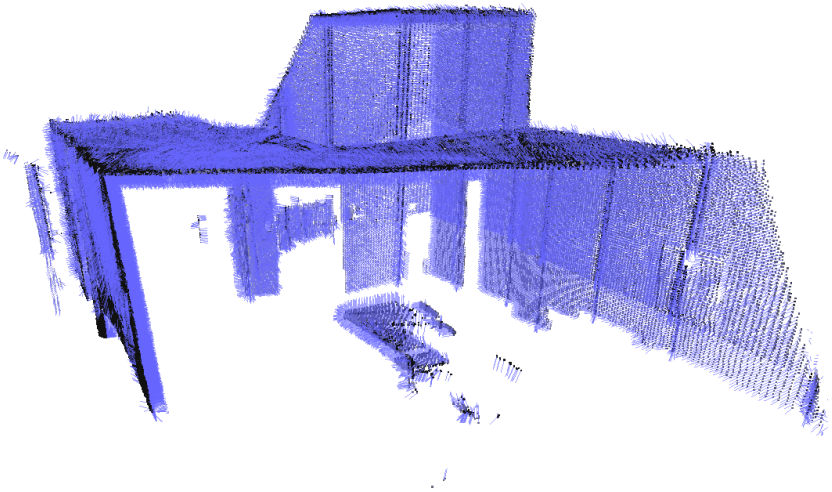
This calibration obtains the intrinsic parameters of the camera, as the focal length and center point. This calibration method used is widely used so it is expected that the results are accurate.

7.3.2 Camera Extrinsic Calibration

The method used to obtain the extrinsic calibration of the camera was the hand-in-eye calibration method. In this work, an ArUCO marker with a side of 200 mm was placed about 2 m away from the camera. The PTU was then manually controlled to capture different images of the marker in different poses of the camera. After about 30 poses, the calibration was done. The number of calibration done using this method were 3, and the results are shown in Table 7.4. The results are very close between calibrations: for example, the standard deviation in the translation is in the order of 0.001 m.



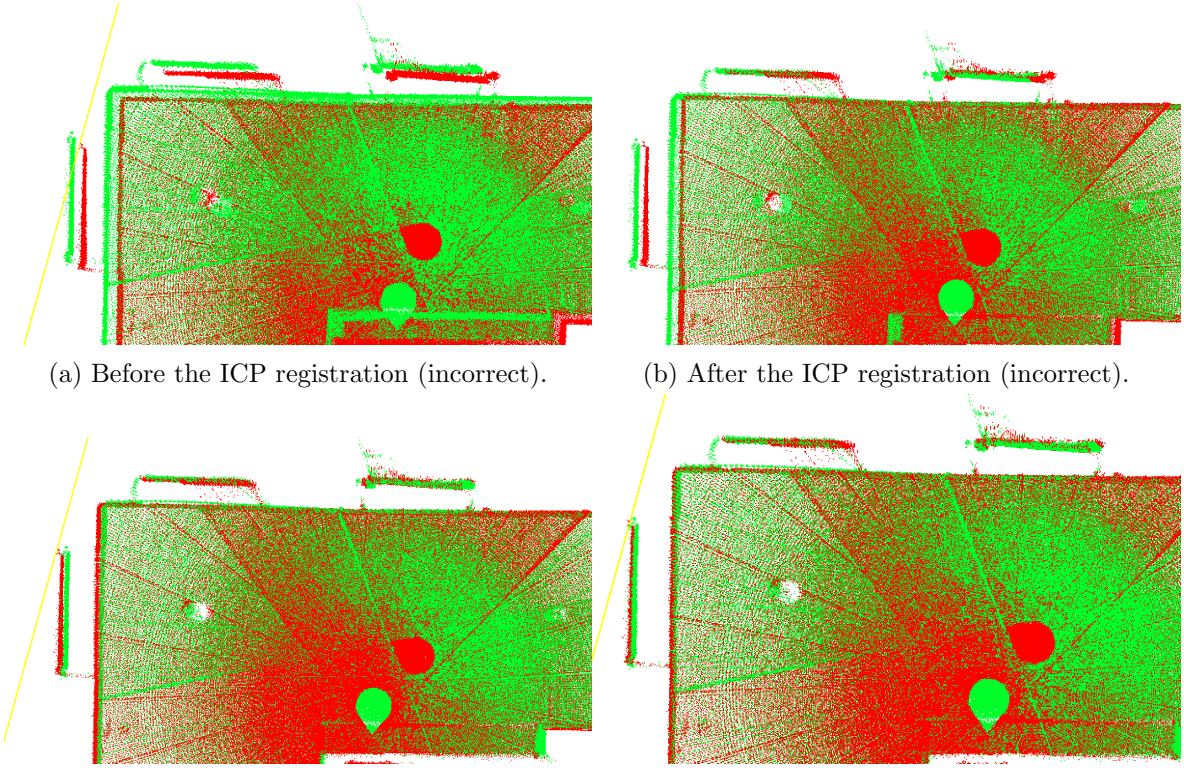
(a) Normal estimation result using the proposed method.



(b) Normal estimation result using the common method.

Figure 7.8: Comparison of both Normal estimation methods (the blue arrows represent the normal vector).

However, this calibration lacks the required precision for the color registration, because the colors are noticeable misaligned with the geometry, as seen in the next section. The failure of this calibration can be related to multiple shortcomings in this calibration method. First, this calibration uses an ArUCO as the marker for the pose estimation. It is possible that the error associated with the pose estimation is large and be the source of the error of the calibration. Second, the limited field of view of the camera limits the range of the movements of the PTU for a small interval of angles in pan and tilt. This limitation in space can result in a limited space of angles registered, which can therefore impact the accuracy of the result. It is possible that, by using more markers, wider angles cloud be possible. Hence, this calibration fails to obtain an accurate extrinsic calibration of the camera.



(a) Before the ICP registration (incorrect).

(b) After the ICP registration (incorrect).

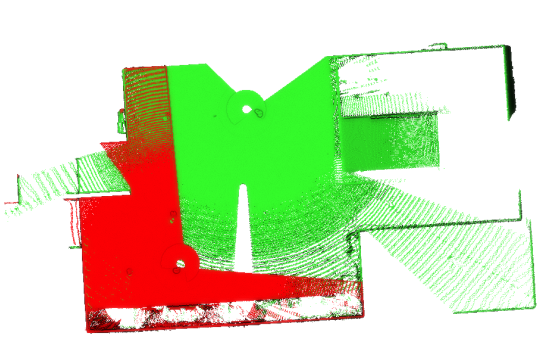
(c) Before the ICP registration (correct).

(d) After the ICP registration (correct).

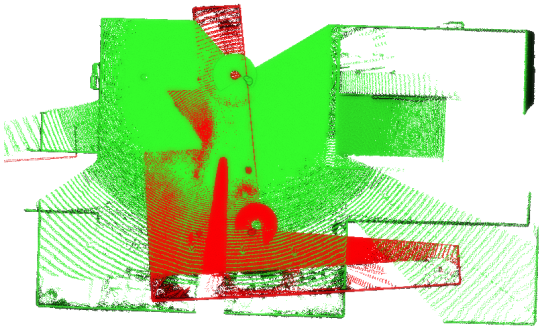
Figure 7.9: Comparison of the results regarding the ICP registration for different initial estimates. Two registration with a small difference result (Figures 7.9a and 7.9c) can yield two different outcomes (Figures 7.9b and 7.9d).

Table 7.4: Results of the extrinsic calibration of the camera.

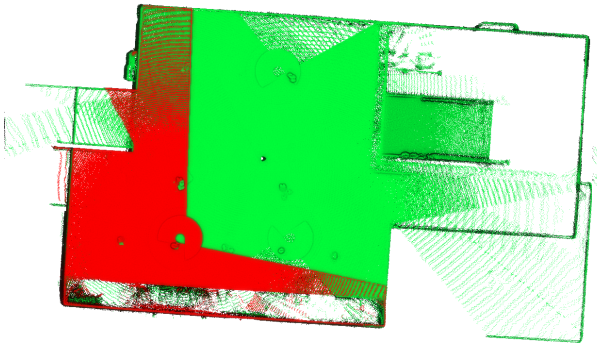
	translation			rotation			
calibration	x	y	z	q_x	q_y	q_z	q_w
0	-0.103524	-0.086451	0.050509	0.027264	-0.702682	-0.024205	0.710570
1	-0.100458	-0.088829	0.049478	0.030372	-0.701046	-0.027414	0.711941
2	-0.098975	-0.084319	0.046249	0.033105	-0.702234	-0.031446	0.710481
3	-0.101184	-0.089208	0.052215	0.029564	-0.702891	-0.025002	0.710243
μ	-0.10104	-0.087202	0.049613	0.030076	-0.702213	-0.027017	0.710808
σ	0.001643	0.001971	0.002174	0.002088	0.000714	0.002817	0.000665



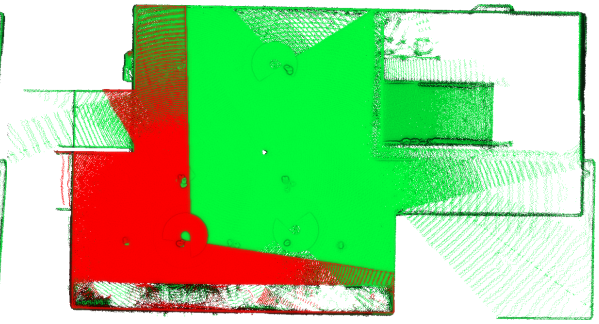
(a) Pairwise registration (before).



(b) Pairwise registration (after).



(c) Accumulated registration (before).



(d) Accumulated registration (after).

Figure 7.10: Comparison of the ICP registration between two point clouds or between the accumulated point cloud and another point cloud.

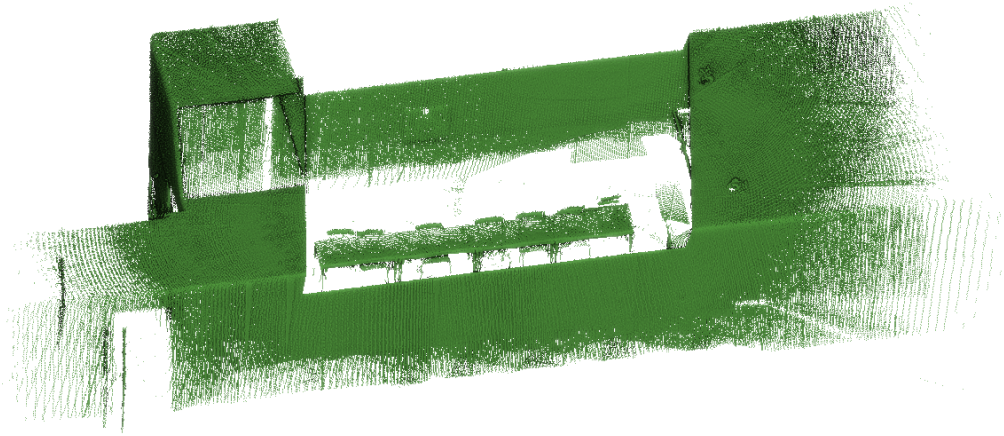


Figure 7.11: Resulting fusion of all point clouds obtained in the capture 3, after the acquisition registration.

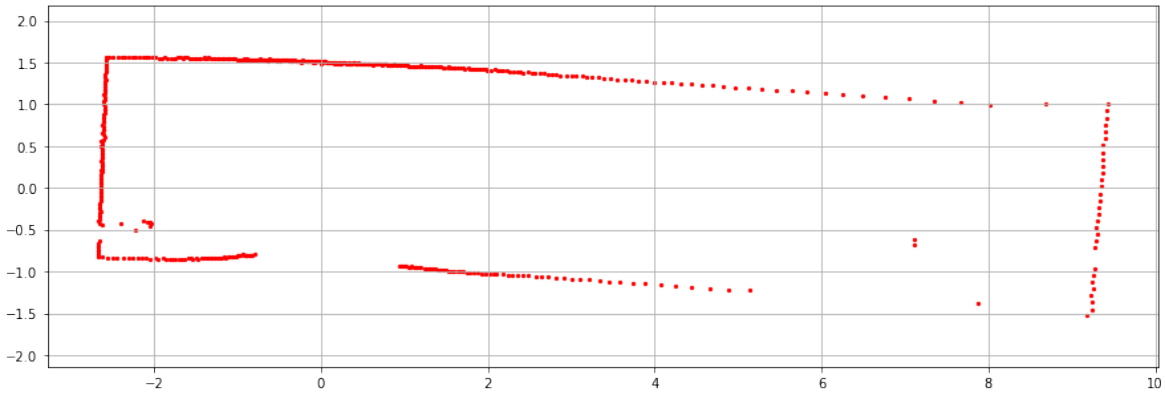


Figure 7.12: Deformed laser of the LMS100 laser scanner.

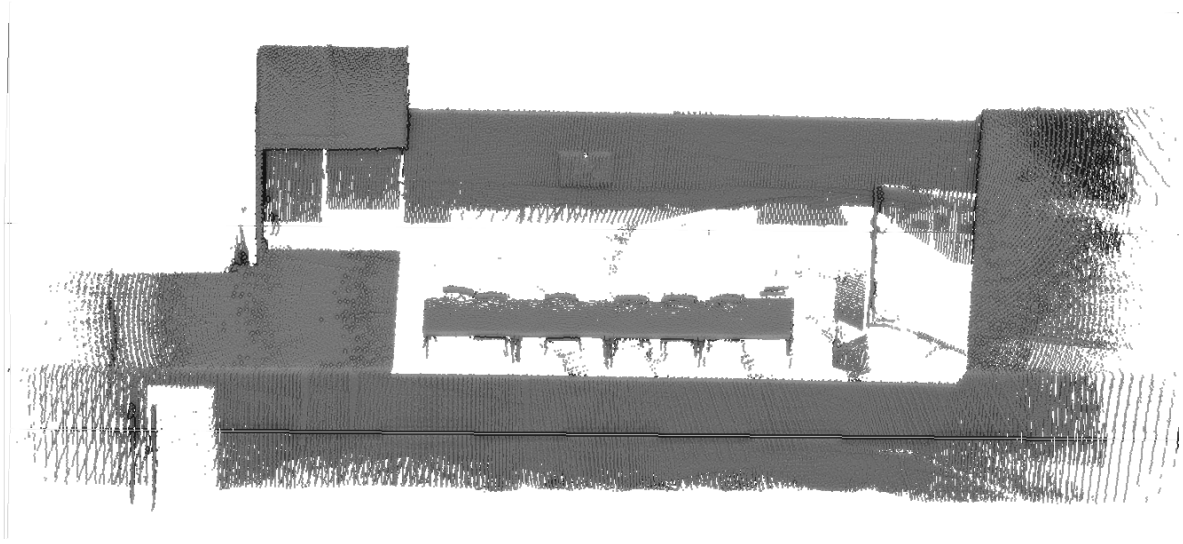


Figure 7.13: Result of the geometric reconstruction of the capture 5.

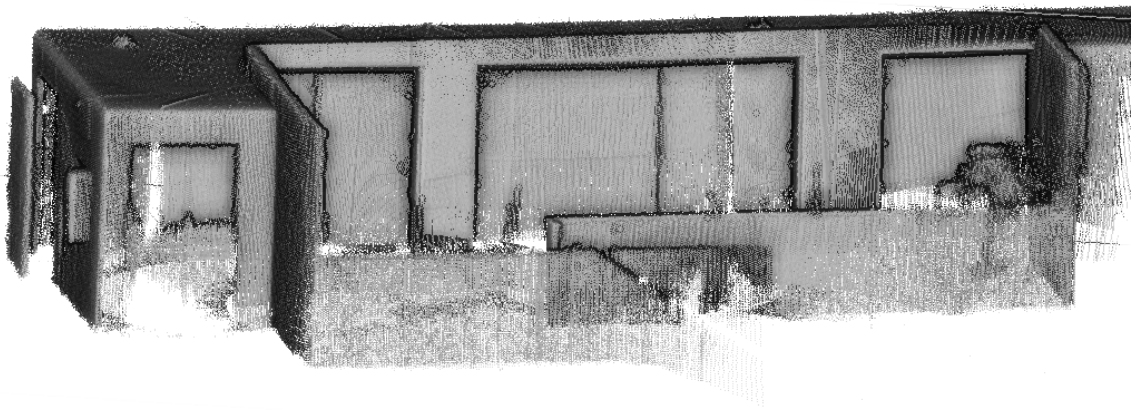


Figure 7.14: Result of the geometric reconstruction of the capture 6.

7.3.3 Color Registration and Fusion

In the color registration and fusion process, the color is attributed to each point in the point cloud. In short, the color registration defines how each point is colorized, based on an image, and the fusion process defines what is the resulting color, after all the images are registered. In total, six methods were tested in this work for the color fusion: the first three methods choose a color from all the registered color and the last two methods calculate the average of all the colors, though a simple mean, or a weighted mean.

The registration process was, however, not perfect, mostly due to a imperfect calibration of the camera. Therefore, the color are not registered in their correct positions, as seen in Figure 7.15. In this image, the colors from different object are registered in wrong objects, for example, the elevator door. This is a common problem and can be seen, for example, in the elevator door.

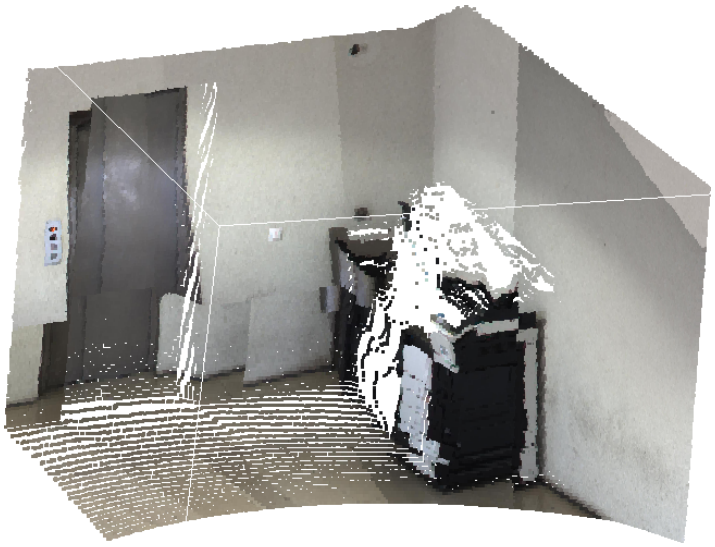


Figure 7.15: Example of the inaccurate color registration.

Also, the images taken have different color values for the same objects. This is due to illumination differences at the capture of each image. This difference is also noticed in the colorized point clouds. This problem is hard to solve, but can be mitigated by ensuring an uniformly illuminated scene. Even so, direct sunlight or shadows negatively affect the colorization. This problem can be shown is the images in Figure 7.16, where the differences in color are very noticeable. The resulting colorization has, therefore, the same variation in color as the one present in the images, as shown in Figure 7.17.

The color fusion techniques also have a great effect on the final result. The colors look sharper if the chosen method is the first or last color fusion method, but discontinuities caused by the imperfect registration are more noticeable. On the other hand, mean methods results in a blurring effect on the colors, but the discontinuities are less noticeable. In general, the mean fusion produces the best result overall, but the small details are smudged. The closest color fusion produces the sharpest result, so the small details are visible, however, it yields the worst quality overall. The two results can be seen in Figures 7.18b and 7.19a.

Moreover, the mean method still shows some discontinuities and edges. To improve the blending of the colors, this work proposes a weighted mean, based on two factors: f_1 and f_2.

Figure 7.16: Two images taken on the same acquisition with different colors.

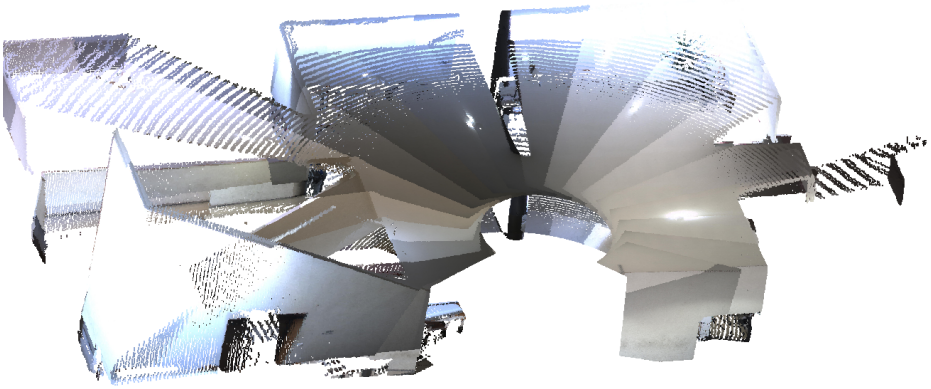
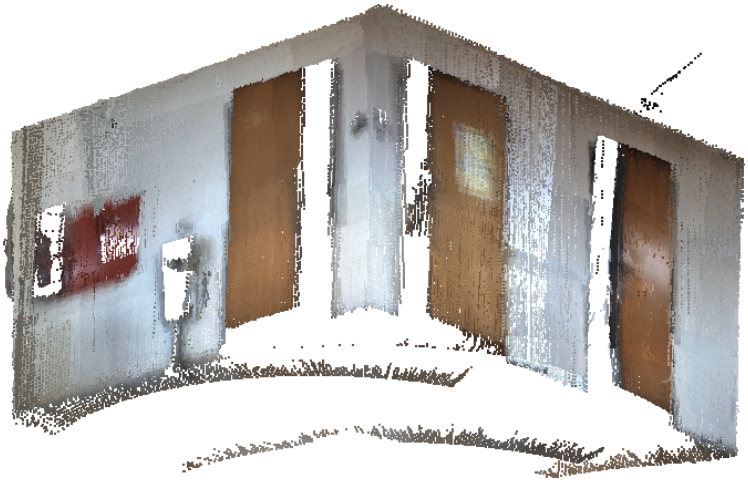


Figure 7.17: Illumination issues in the point cloud.

This factors prioritize colors that are taken closer to the point or taken closer to the center point of the camera. In Figures 7.19b and 7.19c, this two color fusion methods are compared against the closest color method, present in Figure 7.19a. As seen, both methods have overall better results, specially in the blending of the different colors, and reducing the transitions between images.

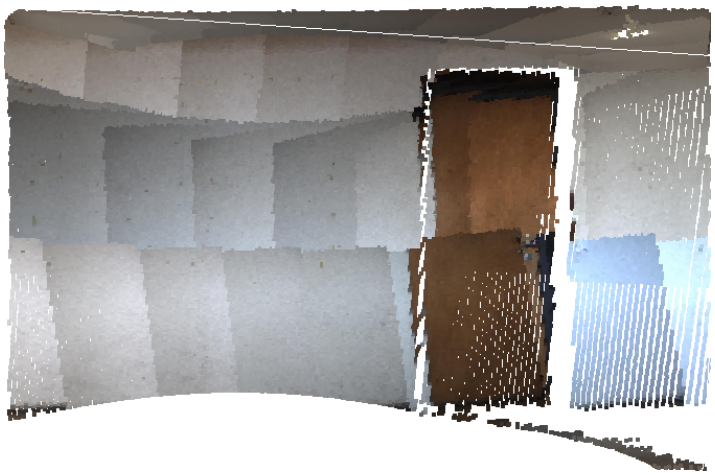
Finally, the color reconstruction method successfully colorized the point clouds, as seen in Figure 7.20. However, the results are far from perfect, due to the inconsistent color in different images and the sub-optimal calibration of the camera.



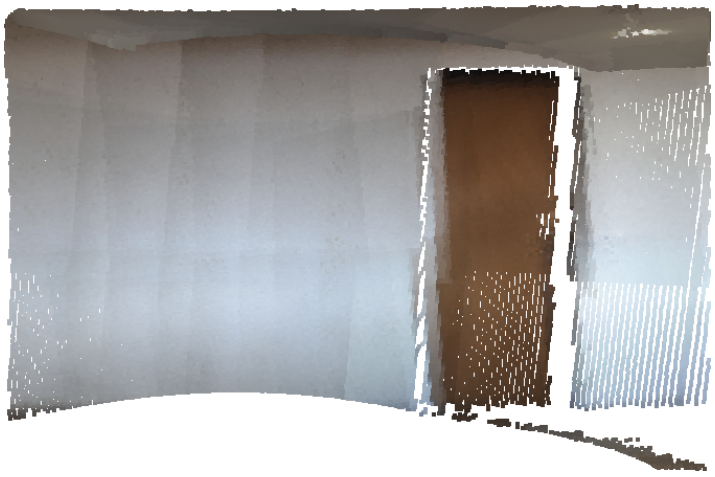
(a) Mean color fusion method.

(b) Closest color fusion method.

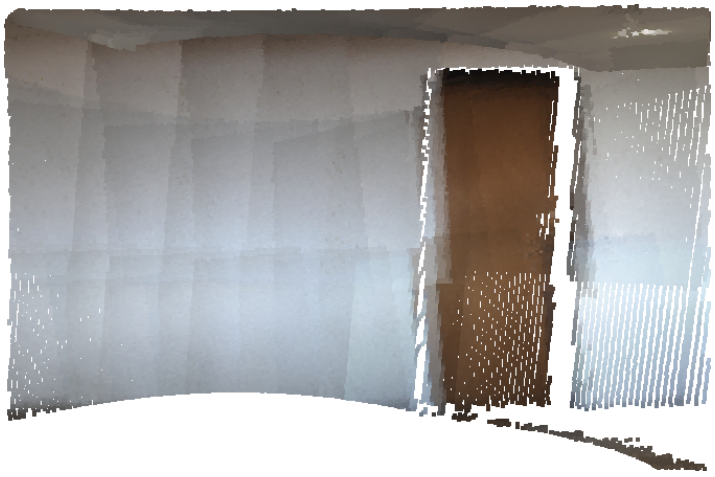
Figure 7.18: Comparison of two color fusion methods.



(a) Last color fusion method.



(b) Weighted mean with factor f_1 color fusion method.



(c) Weighted mean with factor f_2 color fusion method.

Figure 7.19: Comparison of two weighted mean color fusion methods.

Figure 7.20: Full color registration of one point cloud.

Chapter 8

Conclusions and Future Work

3D reconstruction of real world scenes has been and still is an important field today with many applications. The most promising technology relies on LiDAR laser scanners and cameras to capture both the geometry and color information of the scene. However, the fusion of the data of both sensors still has some issues. This work presents a methodology for both the capture process, as well as the algorithms to reconstruct the geometry and the color information from the laser scanner and camera data.

First, a 3D scanner was developed, in order to obtain the data required. This 3D scanner is composed, at its core, by a pan and tilt unit, a 3D laser scanner and a camera. In particular, three laser scanners were used, to test the validity of the reconstruction algorithms and also to study the influence of the laser scanner in the final result.

The scene capturing methodology describes a method to minimize the technical limitations of the sensors, so that the scene capture is not influenced by them. Hence, each capture is composed by several acquisitions, that are taken from different poses in the scene. In each acquisition, the PTU joints moves though multiple waypoints and multiple images and laser scans are recorded during this movement. The pose and number of acquisitions should be adequate such that every part of the scene is recorded. However, this planning is done a priori, so it is not always certain that the scene is totally captured, as seen in the results.

The first reconstruction is the geometric reconstruction, which uses the laser scanner data and the PTU transformations to reconstruct the geometry, resulting in a point cloud. This reconstruction relies on an accurate calibration of the laser scanner to the PTU frame, which is called the extrinsic laser calibration. The first attempt of this calibration was with the RADLOCC calibration. This was, however, unsuccessful, because the calibrations yield by it still produced point clouds with visible deformation. Therefore, a new method was developed which has superior results and the point cloud produced by it does not have noticeable deformations. Furthermore, a normal estimation method was developed that exploits the bi-dimensional structure of the point cloud obtained to estimate the normals. This method is faster than the more standard method using the k-neighbors without sacrificing the overall result. However, this method is sensible to inconsistencies in the movement of the PTU, specially at the interruptions using to capture the images. Finally, the registration of multiple acquisitions was done using the ICP method. However, this method was sub-optimal, because the registration failed if the initial estimate of the transformations were not close to the correct transformation. To bypass this problem, the initial estimate was done manually first. Also, the ICP method is limited to a pairwise registration, so three methods were tried to apply

the ICP to multiple point clouds. Overall, the geometric registration was successful and the resulting point cloud is dense and accurate.

The next stage in the reconstruction was the color reconstruction, which uses the image taken from the camera to attribute colors to the point cloud previously obtained. This reconstruction was separated into two steps. First, each image is registered in the point cloud, by re-projection of the points in the image surface. Then, the points that are outside the visual frustum of the camera and the points that are occluded are removed. Finally, the color correspondent to each projected point is obtained using a bilinear interpolation. After this step, each image has a correspondent partial colorization of the point cloud. This step relies on the projection matrix and extrinsic transformation of the camera to register the points in the image, which are obtained via the intrinsic calibration and extrinsic camera calibration. Then, each point has multiple colors associated, but an unique color has to be chosen for each point, which is done in the color fusion step. Multiple fusion techniques were tried, as choosing the first, last or closest color from all the registered colors, or by calculating the average color of the registered colors. Overall, the color reconstruction is not optimal due to several factors. First, the images vary in hue or intensity, due to factors during the capture, like the lightning of the scene. This creates inconsistencies in the colorization point cloud that are really noticeable in the final results. Secondly, the extrinsic of the camera is not accurate, so the color registration is imperfect.

Overall, the results are satisfactory, specially the geometric registration. There are, however, numerous limitation on both registrations that require to be solved, for example, a better extrinsic camera calibration, or an automatic acquisition registration method.

8.1 Future Work

The methodologies explored in this work still leave much room for exploration and development. Some algorithms and methods have shortcomings in their performance and usability. Hereby, some possible solutions and thoughts that could be explored and developed in the future work are described. In particular, possible solutions for the acquisition registration, color normalization and camera extrinsic calibration are discussed.

The acquisition registration can be improved in two approaches. The first approach could be the integration of some localization device in the robot, as for example a inertial measurement unit, which can be used to obtain an approximate transformation between the acquisitions. This approximate solution could replace the initial estimate done manually in this work. The second approach world be to improve or modify the ICP algorithm, because of its disadvantages. Firstly, the new algorithm should be able to register multiple point clouds at once, instead of just two. This algorithm world probably have a better accuracy than the methods used in this work. Also, the distance between point clouds, which is the heuristic of the ICP algorithms should be modified, because most times it is incorrect and is only successfully if the two point clouds are very close. Instead, features could be extracted from the point clouds, for example, edges and corners, and their distance should be used instead. Because this features are sparse, it is possible that this correspondence is more accurate.

Moreover, the images obtained should have a similar color for the same object, which is not the case, due to the different illumination. So, a normalization of the images should be done prior to any color registration phase. Also, the use of high dynamic range photography should be used to improve this normalization. As a result, the colors of the same objects should be

the same across all the images, and this color should be independent of the lightning of the scene.

Furthermore, a new calibration method to obtain the extrinsic calibration of the camera should be developed. The principal limitations of the hand-in-eye algorithms, is the fact that only a small portion of the angle interval of the joint angles are used for the calibration, and the ArUCO pose estimation has an high error. Recently, in the master thesis of Filipe Costa [Cos18], a bundle calibration method of ArUCO markers and camera was developed. With small adjustments, this algorithm can be modified to calibrate a camera mounted on a PTU. More specifically, this calibration obtains a precise pose of the camera and all the ArUCO markers, which could be considered to fed the original hand-in-eye algorithms, with the change that now there are multiple markers instead of just one. It is possible that this method could achieve better results that the original hand-in-eye method used in this work.

Finally, some abnormalities were found in the laser scans produced by the 2D laser scanner SICK LMS100, which undermined its use. The source of the problem was not identified in this work, but it should be found.

Bibliography

[BM92] P. J. Besl and N. D. McKay. "A method for registration of 3-D shapes". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 14.2 (Feb. 1992), pp. 239–256.

[Cai+05] Zi-xing Cai et al. "A 3-D perceptual method based on laser scanner for mobile robot". In: *2005 IEEE International Conference on Robotics and Biomimetics - ROBIO*. 2005, pp. 658–663.

[Cos18] Filipe Costa. "ATLASCAR2 Sensors Calibration by Global Optimization". 2018.

[DMS06] Paulo Dias, Miguel Matos, and Vitor Santos. "3D Reconstruction of Real World Scenes Using a Low-Cost 3D Range Scanner". In: *Comp.-Aided Civil and Infrastruct. Engineering*. Sept. 2006, pp. 486–497.

[Fer15] Enrique Fernandez. *Learning ROS for Robotics Programming - Second Edition*. Packt Publishing, 2015.

[HD95] Radu Horaud and Fadi Dornaika. "Hand-Eye Calibration". In: *The International Journal of Robotics Research* 14.3 (1995), pp. 195–210.

[HT99] Joachim Hornegger and Carlo Tomasi. "Representation Issues in the ML Estimation of Camera Motion". In: *Proceedings of the IEEE International Conference on Computer Vision*. Vol. 1. Feb. 1999, pp. 640–647.

[KHZ10] D. Klimentjew, N. Hendrich, and J. Zhang. "Multi sensor fusion of camera and 3D laser range finder for object recognition". In: *2010 IEEE Conference on Multisensor Fusion and Integration*. Oct. 2010, pp. 236–241.

[KTB07] Sagi Katz, Ayellet Tal, and Ronen Basri. "Direct visibility of point sets". In: *ACM Transactions on Graphics*. Vol. 26. July 2007.

[Lev+00] Marc Levoy et al. "The Digital Michelangelo Project: 3D Scanning of Large Statues". In: *Proceedings of the ACM SIGGRAPH Conference on Computer Graphics*. June 2000, pp. 131–144.

[Mau+09] F. Maurelli et al. "A 3D laser scanner system for autonomous vehicle navigation". In: *2009 International Conference on Advanced Robotics*. 2009, pp. 1–6.

[New+11] Richard Newcombe et al. "KinectFusion: Real-time dense surface mapping and tracking". In: *2011 10th IEEE International Symposium on Mixed and Augmented Reality* (Sept. 2011), pp. 127–136.

[NTM07] Z. Nemoto, H. Takemura, and H. Mizoguchi. "Development of Small sized Omni directional Laser Range Scanner and Its Application to 3D Background Difference". In: *IECON 2007 - 33rd Annual Conference of the IEEE Industrial Electronics Society*. Nov. 2007, pp. 2284–2289.

[Pfo+14] Lars Pfotzer et al. “Development and calibration of KaRoLa, a compact, high-resolution 3D laser scanner”. In: *2014 IEEE International Symposium on Safety, Security, and Rescue Robotics (2014)* (2014), pp. 1–6.

[Pow64] M. J. D. Powell. “An efficient method for finding the minimum of a function of several variables without calculating derivatives”. In: *The Computer Journal* 7.2 (1964), pp. 155–162.

[SC13] Ju Shen and Sen-ching Cheung. “Layer Depth Denoising and Completion for Structured-Light RGB-D Cameras”. In: *Proceedings / CVPR, IEEE Computer Society Conference on Computer Vision and Pattern Recognition. IEEE Computer Society Conference on Computer Vision and Pattern Recognition.* June 2013, pp. 1187–1194.

[SN01] Jochen Schmidt and Heinrich Niemann. “Using Quaternions for Parametrizing 3-D Rotations in Unconstrained Nonlinear Optimization”. In: *Vision, Modeling, and Visualization.* Jan. 2001.

[SNH03] Hartmut Surmann, Andreas Nuchter, and Joachim Hertzberg. “An autonomous mobile robot with a 3D laser range finder for 3D exploration and digitalization of indoor environments”. In: *Robotics and Autonomous Systems.* Vol. 45. Dec. 2003, pp. 181–198.

[SPW17] Prarinya Siritanawan, Moratuwage Diluka Prasanjith, and Danwei Wang. “3D feature points detection on sparse and non-uniform pointcloud for SLAM”. In: *2017 18th International Conference on Advanced Robotics* (2017), pp. 112–117.

[Yos+11] T. Yoshida et al. “3D laser scanner with gazing ability”. In: (2011), pp. 3098–3103.

[Zol+18] Michael Zollhöfer et al. “State of the Art on 3D Reconstruction with RGB-D Cameras”. In: *Comput. Graph. Forum* 37 (2018), pp. 625–652.

[ZP04] Qilong Zhang and R. Pless. “Extrinsic calibration of a camera and laser range finder”. In: *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems.* Vol. 3. 2004.