
Politecnico di Torino

Politecnico di Torino

Laurea Magistrale in Ingegneria Meccanica

Anno accademico: 2020/2021

Sessione di Laurea: Luglio 2021

# ToF 3D Vision Algorithms in C++ for Robotic Applications

**Relatori**:

Prof. Sabri Cetin, University of Illinois at Chicago
Prof. Marco Masoero, Politecnico di Torino

**Candidato**:

Matteo Mercuri

## ACKNOWLEDGMENTS

First and foremost, I have to thank my Advisor Prof. Sabri Cetin. He helped me whenever I ran into a trouble spot or had a question about my research or writing. I would also like to thank Fariz Khandkar, R&D Engineer at Servotech, for all his help, without whom this work would not have been possible. I want to thank their hospitality as well: the door to their office was open every time I needed it and it was always a positive and constructive work environment. Thank you for all the lunches we had together. Also, I would like to thank professor Masoero for helping me and following my work as my home university's advisor and Politecnico di Torino itself, for giving me the incredible opportunity to finish my Master in Chicago with this exchange program.

Thanks to all my friends all over the world. Thanks to my friends in Varese, who even if I was absent most of the time were always ready to welcome me back every time I came home. Thanks to my friends from UGA, you are part of the greatest experience I had in my life that made me who I am today. Thanks to my Torino's and Milano's friends from the Chicago experience, who shared with me this crazy and unforgettable year. Thanks to my friends in Torino, for welcoming me after high school in a new city and making everything easy. A special thanks goes to Barbara, Max, Andi, Tommi and Dede. You welcomed me every Sunday at your house for pizza and you made me feel home every time, like in a second family. I will never forget it.

## ACKNOWLEDGMENTS (continued)

Thanks to Sara, for creating such a beautiful relationship in such a short time. You will never be alone and I will always be there for you.

Thanks to Ilaria, for being by my side this past year and throughout this experience. You supported me in my everyday life, from simple things as cooking for me to important ones, giving me precious advices.

Finally, I must express my very greatest and profound gratitude to my whole family, in particular to my parents, Federica and Fortunato, and to my sisters, Emma and Marta, for providing me with unfailing support and continuous love and encouragement throughout my years of study and through everything. All my accomplishments in life would not have been possible without them. Thank you from the bottom of my heart.

MM

## TABLE OF CONTENTS

## TABLE OF CONTENTS (continued)

## LIST OF FIGURES

## LIST OF FIGURES (continued)

**FIGURE** **PAGE**

## SUMMARY

Il presente lavoro di tesi si focalizza sullo studio dell'utilizzo dell'hardware e del software di una telecamera Time-of-Flight e successivamente sulla creazione di codici in C++ che cercano di gettare le basi per futuri programmi che sarebbero in grado di affrontare e risolvere i problemi più comuni relativi alla 3D Machine Vision. Tutti i codici e programmi presenti nella tesi sono nati e sviluppati con l'idea chiave in mente che possono costituire, in un lavoro futuro, un punto di partenza per progetti più complessi legati alla 3D Computer Vision e quindi essere implementati in applicazioni come bin-picking, guida autonoma, controllo qualità e così via. I programmi C++ sviluppati in questa tesi, insieme a questioni minori, affrontano quattro problemi principali nel campo della visione artificiale: Registrazione, Creazione di modelli CAD, Segmentazione e Riconoscimento di oggetti. La registrazione consiste nell'allineare varie nuvole di punti 3D, dette "point cloud", ottenute da diversi punti di vista dello stesso oggetto in modo da ricostruire un modello 3D completo dell'oggetto. La creazione del modello CAD è solitamente il passaggio successivo alla Registrazione e consiste nella creazione di una mesh 3D dell'oggetto registrato. Questa mesh può essere esportata in AutoCAD o in qualsiasi altro software CAD 3D per eseguire ulteriori studi e analisi dell'oggetto. La segmentazione consiste nel dividere i point cloud in più parti, che possono essere poi studiate indipendentemente. Infine, l'ultimo argomento studiato e sviluppato nella tesi è il Riconoscimento di oggetti, che consiste nella capacità di localizzare e riconoscere oggetti in un 3D point cloud.

# CHAPTER 1

# INTRODUCTION

Three dimensional (3D) machine vision has been firstly discussed in the 1960s by Larry Roberts, considered the father of Computer Vision, in his PhD thesis at MIT, where he introduces the possibility of extracting 3D geometrical information from 2D perspective views of blocks. Later, in 1978, a milestone was set by David Marr, an MIT researcher, which proposed a bottom-up processing approach to the scene study, that involves the building of an image from the smallest pieces of sensory information. Marr's work has been highly praised for years and his work is probably the single most influential work in computer vision ever. Nonetheless, some researchers tried to move away from Marr paradigm going into the opposite direction and tried a top-down processing approach, especially in autonomous vehicle applications, where it is not necessary to know the complete 3D object model, but only some partial general information, such as if an element is moving towards or away from the car. [31]

Nowadays, 3D machine vision is an evolving and maturing technology that can be used in a variety of applications. It takes life from the necessity of having more accurate, precise and meaningful information from systems previously adopting simple 2D vision technology. In recent years it has been a force in the marketplace and the proliferation of 3D components has been tremendous.

The global 3D machine vision market size has been valued at USD 1.13 billion in 2019 [1] and it is expected to grow at an annual rate of 14.7% from 2020 to 2027 reaching a value of USD

3.46 billion in 2027. Hardware accounted for more than 70% of the revenue share in 2019, while the market for software is more fragmented, being application specific. PC-based products lead the market. The product with the highest potential growth is 3D cameras, which are projected to grow at the fastest rate. Automotive industry accounts for 18% of the share of the global revenue in 2019, while the most important applications are quality assurance and inspection, holding approximately 52% share of the global revenue in 2019.

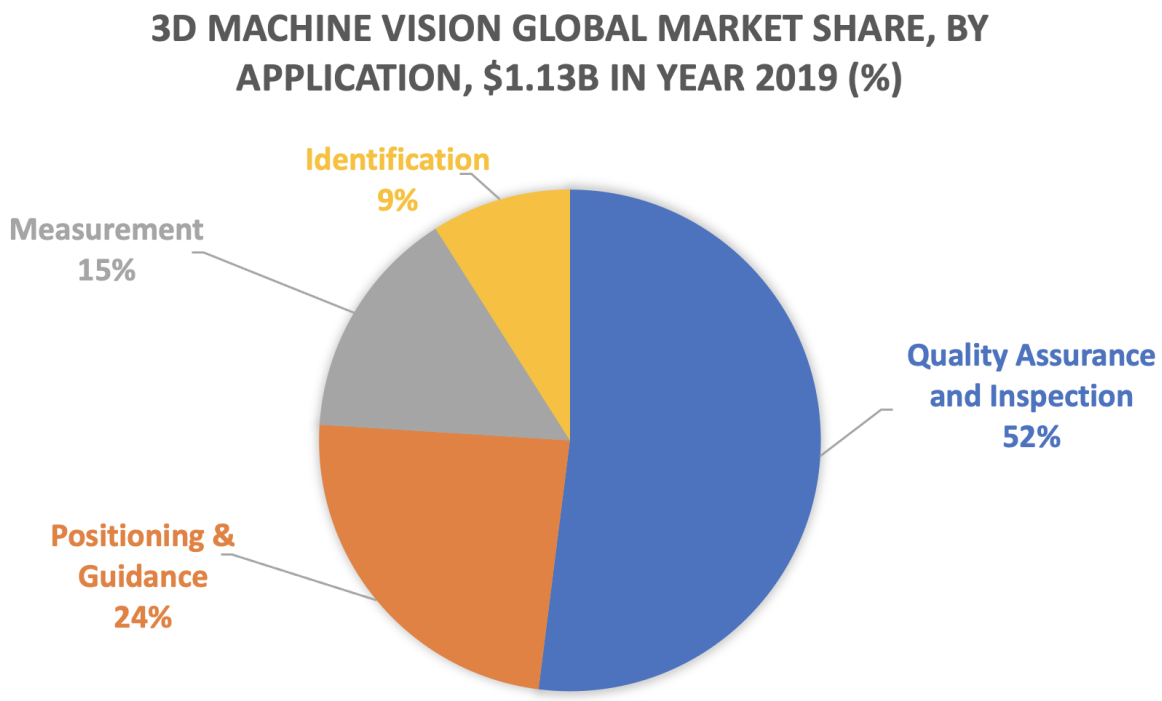


Figure 1: Pie chart representing 3D machine vision market share by application in 2019 [1]

The biggest companies in the market in the countries that lead the world in 3D vision are Cognex Corporation (USA), Keyence Corporation (Japan), Omron (Japan), Teledyne Technologies (USA), National Instruments Corporation (USA), Sick AG (Germany), ISRA Vision AG (Germany), Stemmer Imaging (Germany), LMI Technologies (Netherlands), and Basler AG (Germany) [1].

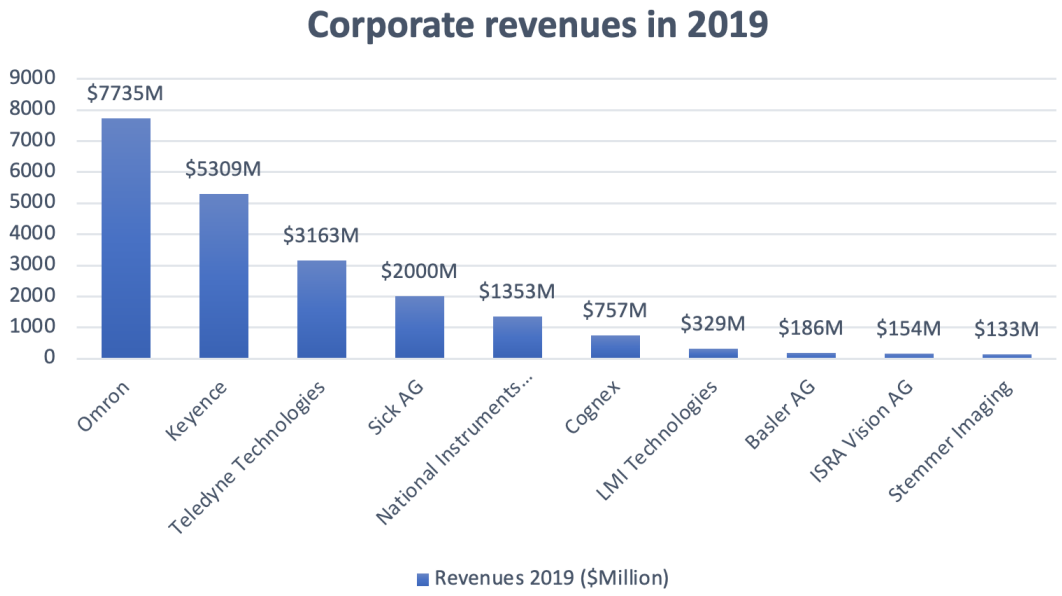


Figure 2: Revenues of leading companies in 2019 [2][3][4][5][6][7][8]

The possible applications of 3D machine vision are multiple, ranging from automotive to gaming, to military to health and robotics. In particular, this technology is increasingly being

used in the automotive and robotic sector. Automotive strives for better cameras and data to improve the performances of self-driving cars. One of the most important feature that a self-driving car needs to have is collision avoidance, mainly required after the high number of accidents due to negligence of the driver, and 3D machine vision has been proven to be useful in this area. For what concerns robotics, 3D vision is employed in quality control, assembly lines and collaborative robots. For quality control, 3D vision is helpful thanks to its ability to analyze complex surfaces at a higher rate than humans. In assembly lines 3D vision gives more flexibility, independence and reliability to the system with respect to their 2D counterparts, since it gives the ability to recognize different parts and grab them on a conveyor belt in the most correct way even if all parts do not have the same orientation. Finally, collaborative robots rely on camera systems to avoid humans around them and 3D vision helps this procedure by making it safer and thus allowing the collaborative robot to reach higher speeds and higher loads.

# CHAPTER 2

# 3D VISION

The most common technologies used in 3D machine vision are stereo vision, structured light and time-of-flight (ToF).

## 2.1 Stereo Vision

Stereo vision technology uses two 2D cameras positioned at a distance in a way that resembles the human eye. Measuring the same point on the object, the two cameras will give different positions with different angles $\alpha$ and $\beta$. Thanks to these angles the depth z can be calculated. [9] This technology has a lot of advantages, but at the same time presents important

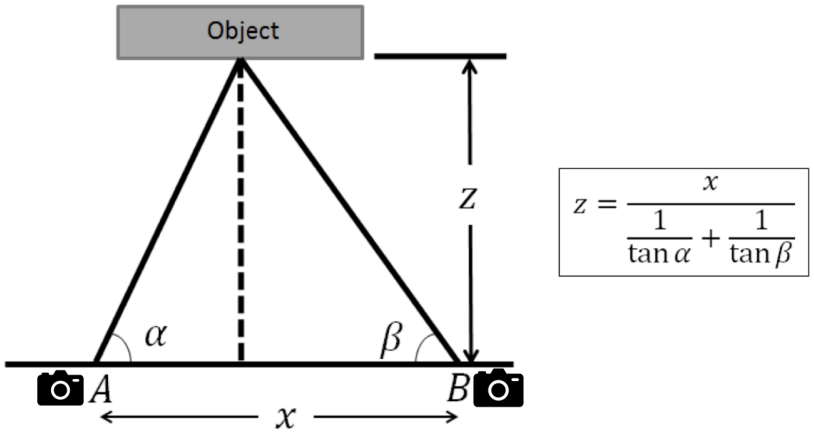


Figure 3: Stereo vision depth measurement [9]

disadvantages and challenges difficult to overcome. Stereo vision has between its advantages the fact that it is a low-cost intuitive method working with simple 2D cameras and that is capable of capturing high resolution images without any energy emission or moving part. On the other hand, there are two significant drawbacks that need to be considered: the correspondence problem and the limited Field of View (FOV). [32]

The latter is caused by the limited field of view of the arrangement of the two cameras and it becomes an issue since it is problematic to change configuration once it has been chosen for a specific application. Instead, the correspondence problem takes life from the difficulty of finding on both cameras the same exact point on the object, especially if the object does not present any special feature on the surface that could help the process. For instance, obtaining the same point of a uniformed colored wall can be quite difficult, leading to an important depth resolution error.

## 2.2 Structured Light

Structured light is considered an active method of 3D imaging technology since it uses an active illumination to project structured patterns on the object under study. The projector sends a known pattern that will be distorted by the object geometry, then a camera captures the distorted patterns and, by analyzing the differences between the know pattern and the distorted one, is able to reconstruct (x, y, z) coordinates [10] [33]. The structured light technology includes among its advantages the fact that it is a really accurate system, it is faster than laser scanning and it is incredibly compact. On the other hand, the limitations are numerous. First of all, it has a slow response time; then, it is sensitive to light conditions

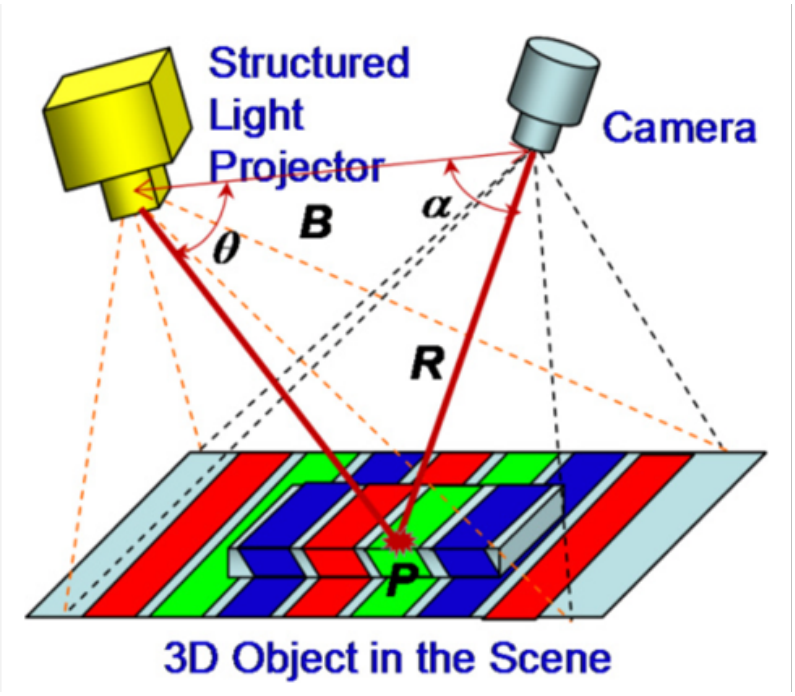


Figure 4: Structured light mechanism [10]

and to transparent or reflective materials similarly to other technologies using optical sensors, making it more suitable for indoor rather than outdoor use. Finally, it has a really high cost of material, making it not cost-effective as other technologies, such as Time-of-flight [9] [34].

### 2.3 Time-of-Flight

The final technology considered is the Time-of-Flight (ToF). Time-of-Flight cameras employ an array of laser light and observe the reflected arrays through a sensor. It can calculate the dimensions of the 3D object by analyzing the phase change of the reflected light with respect

to the source laser. In order to calculate all the object distances it is necessary to calculate the time of flight. The relation between these quantities is the following:

$$d = \frac{1}{2} \cdot c \cdot \tau \tag{2.1}$$

where d is the depth, $\tau$ is the TOF (time of flight) and c is the speed of light $(3 \cdot 10^8 m/sec)$.

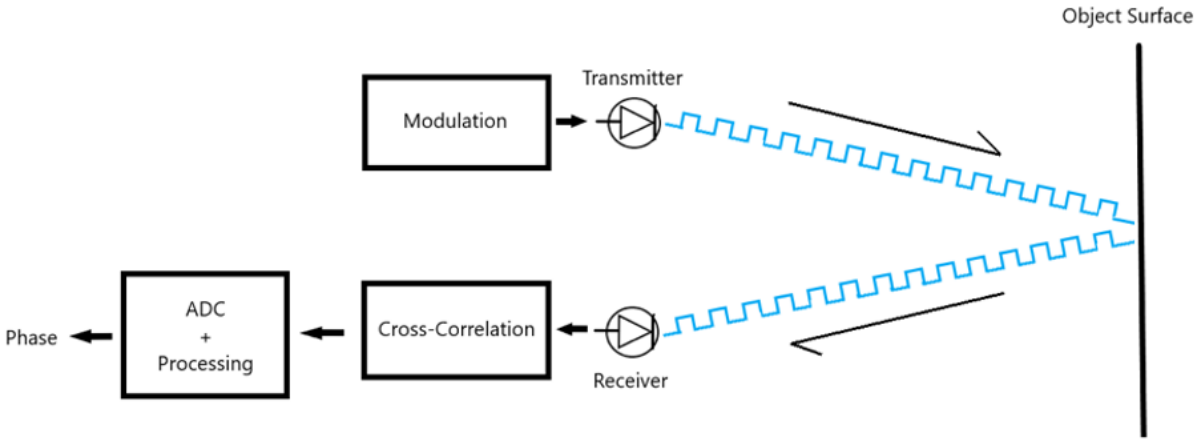


Figure 5: Time-of-Flight principle [11]

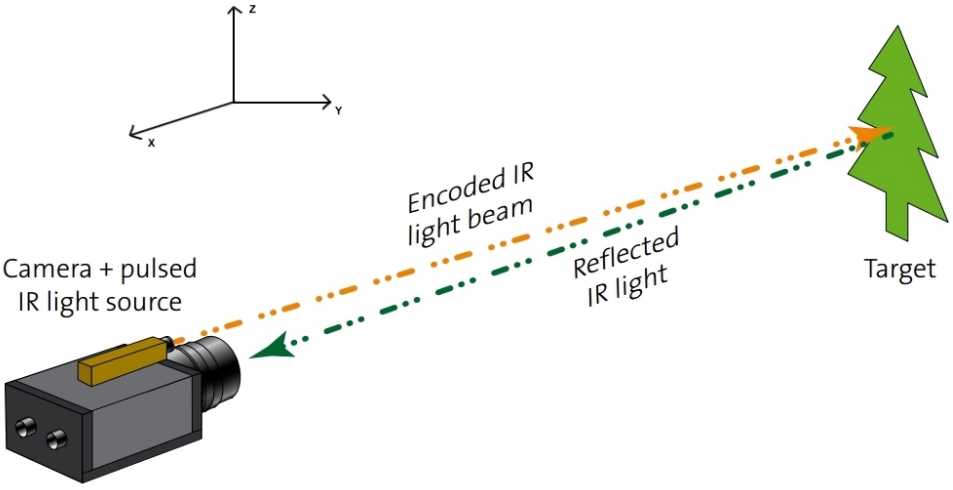


Figure 6: Time-of-Flight mechanism [12]

There are two main methods for TOF calculation: Pulse Based (PB) approach and Continuous Wave Modulation (CWM) approach. The former emits short pulses of light while the latter uses a continuous modulated signal, which can be squared or sinusoidal. In both methods, in order to increase the Signal-to-noise ratio (SNR) millions of cycles of pulses or samples are collected and the final depth is the result of an average [35].

The Pulse Based method emits short pulses of light that will be reflected and captured by a sensor, equipped with shutters, that will detect the light in one or two small temporal windows, the gates [36]. The first shutter happens at the same time the light is emitted from the source, while the second one happens immediately after the first light pulse is ended. The

electrical charge accumulated in the first gate is stored in the camera memory as $Q_1$, while the one accumulated in the second gate is saved as $Q_2$. Denoting $\Delta$t as the pulse width, the final TOF is given by:

$$\tau = \Delta t \cdot \frac{Q_2}{Q_1 + Q_2} \tag{2.2}$$

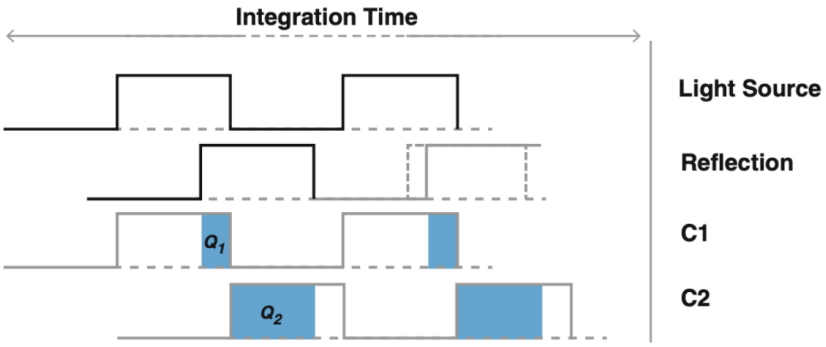


Figure 7: Pulse Based method principle [9]

Depth resolution can reach 1 mm, however it would require emitting a light pulse every 6.6 picoseconds, which is basically impossible to achieve with current silicon-based technology [9]. Today's cameras can easily reach a depth accuracy of 1 cm.

The Continuous Wave Modulation (CWM) approach is the most common method implemented in 3D ToF cameras. In this approach, the scene in continuously illuminated by a near infrared light (NIR) that will reflect on the object and will be received by a sensor. Sub-

sequently, the phase shift between emitted and reflected light will be computed in order to calculate the TOF:

$$\phi = 2\pi \cdot f \cdot \tau \tag{2.3}$$

Where f is the modulation frequency and $\tau$ is the TOF (time of flight). Multiple samples per measurement are taken, with every sample shifted of 90° degrees, with a technique similar to the pulsed light one.

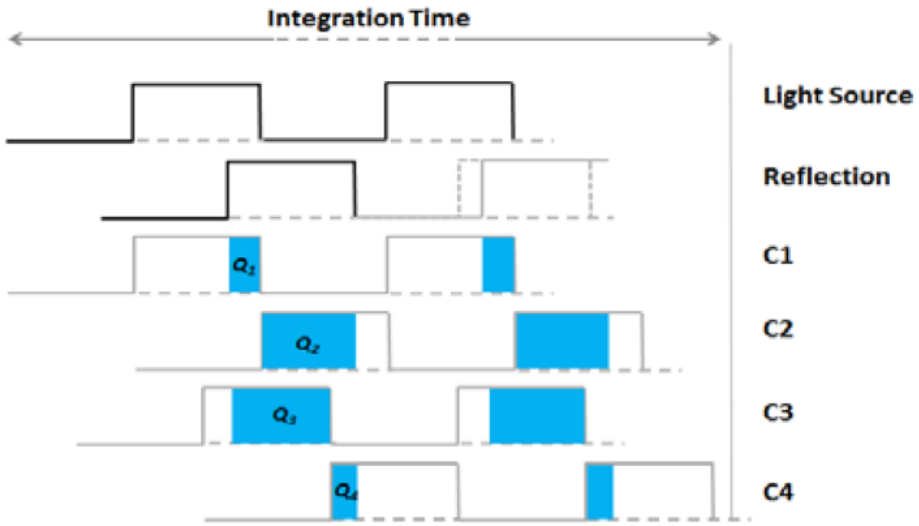


Figure 8: Continuous wave method principle [9]

The process of measurement of the phase is called cross-correlation and it helps finding the following:

$$\phi = \arctan \frac{Q_3 - Q_4}{Q_1 - Q_2} \tag{2.4}$$

Replacing $\phi$ with what has been obtained in equation (2.3), it is possible to write:

$$2\pi f \tau = \arctan \frac{Q_3 - Q_4}{Q_1 - Q_2} \tag{2.5}$$

$$\tau = \frac{1}{2\pi f} \cdot \arctan \frac{Q_3 - Q_4}{Q_1 - Q_2} \tag{2.6}$$

Using equation (2.1), it is possible to write:

$$d = \frac{c}{4\pi f} \cdot \arctan \frac{Q_3 - Q_4}{Q_1 - Q_2} \tag{2.7}$$

Which can be rewritten as

$$d = \frac{c}{4\pi f} \cdot \phi \tag{2.8}$$

Since $\phi_{max} = 2\pi$, it is immediately apparent that there is a maximum measurable distance that depends only on the speed of light and on the modulation frequency, which is $d_{max} = \frac{c}{2f}$. This distance, defined *ambiguity distance*, with a single frequency technique can only be extended

by reducing the modulation frequency. This means reducing the accuracy of the camera, as proven, after some mathematical steps, by [9]. The final formula they obtain is:

$$\sigma = \frac{c}{4\sqrt{2}\pi f} \cdot \frac{\sqrt{A+B}}{c_d A} \tag{2.9}$$

where $\sigma$ is the depth measurement accuracy, A and B are the amplitude and offset of the reflected wave. Advanced ToF cameras employ multi-frequency technologies that allow to amplify the measurable range using different frequencies: lower frequencies to identify the coarse position of the object (i.e. approximately where it is located) and higher frequencies to accurately describe it. The *beat frequency* is the frequency at which the two modulation frequency agree, that corresponds to the true location of the object under study [13].

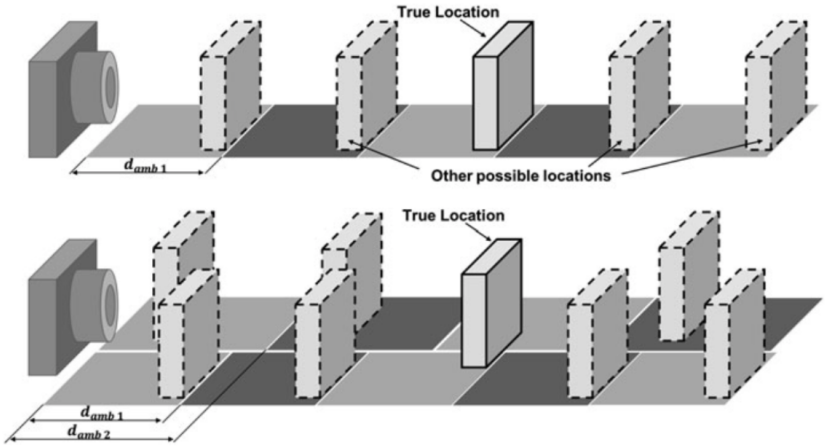


Figure 9: *Top figure*: Single frequency ambiguity problem. *Bottom figure*: Multi-frequency ambiguity problem solved. [13]

Summarizing, pulse method gives low accuracies but a higher operating range, while CMW gives higher accuracies, but it presents ambiguity problems regarding the operating range [35].

Once all the measures are performed, the camera can save and output the data in different formats. There are two main formats that the majority of cameras use: depth image and point cloud. Depth image consists in a 2D image where every point has assigned x and y values and a depth value based on the distance from the sensors. Point cloud instead collects data of the 3D camera so as to have x, y and z coordinates of every point. This method is usually able to give more information to the user, since it can show hidden objects too, that in a depth image could not be represented. Most cameras are able to provide both outputs and convert one into the other.

3D ToF cameras are a technology that still has some limitations in its possible applications. They are mainly due to problems related to lighting, changes in color or curvatures. In fact, many 3D systems show some level of "3D data drop-out", a condition where light, curvature or colors cause some voids in the 3D data.

The first main limitation is represented by unwanted reflections: bright close objects quickly scatter too much light into the lens and can lead to creating artefacts. Multiple reflections can disturb the measurement too, since a ToF cameras wants light that has been reflected only once to perform precise calculations of the waves phase difference. This is usually referred to as a "multipath error". Moreover, ambient light can cause a quick saturation of sensor pixels causing the actual reflected light to not be detected. This problem can actually be partially overcome by using a band pass filter that allows to only collect within the NIR range, cutting out every other

wavelenght. Finally, if an application requires multiple ToF cameras, it is necessary to careful study the setup since they can disturb each other in the measurement. Another limitation happens when an object under study is in motion. The motion can cause measurement errors that need to be considered. Moreover, it is difficult to predict how the materials will react to NIR illumination and these different interactions could modify and change the 3D data obtained from object to object.

## 2.4 Time-of-Flight 3D Vision Applications

Time-of-Flight cameras can be implemented in a variety of application so as to go a step forward towards a fully automated industry. In the following pages 4 different example of possible applications will be described: *Bin Picking*, *Dimensional Metrology*, *Autonomous Driving* and *3D Mapping*.

### 2.4.1 Bin Picking

Bin picking is a benchmark example in the world of robotics and 3D vision. It consists in gripping and handling objects placed on a convoy belt in order to store them or move them to another working station in the factory.

It can be classified in three different types: *Structured Bin Picking* when the parts are placed on the convoy belt in an organized manner so as to be easily recognized and picked; *Semi-structured Bin Picking* when parts are partially organized on the assembly line; *Random Bin Picking* when there is no organization on the convoy belt and the robot needs to be able to recognize the geometry and the orientation of the part and grab it correctly. The last Bin Picking category is the most interesting and challenging one and it needs to be solved employing 3D imaging

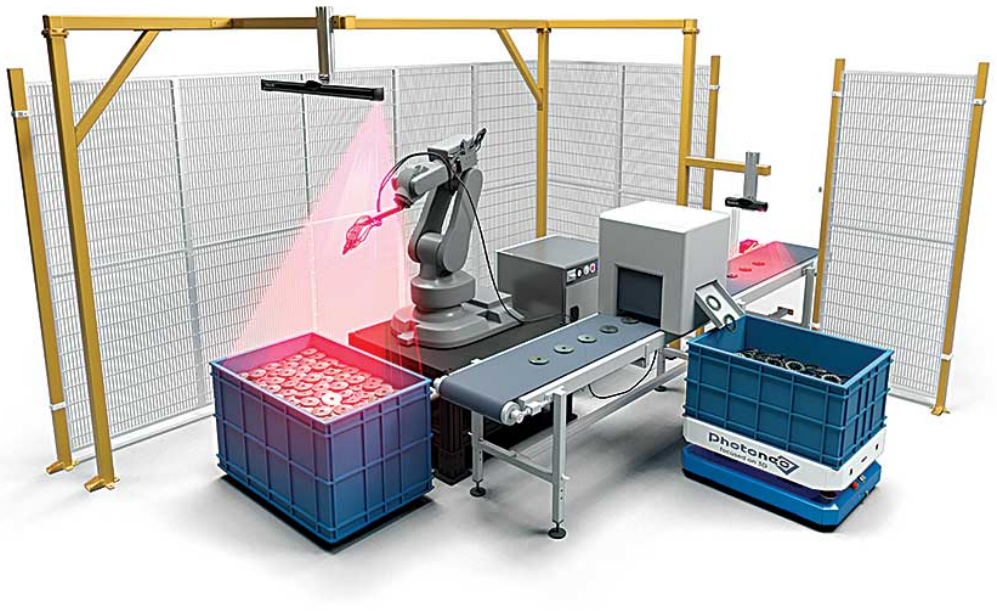

Figure 10: Bin Picking application example [14]

techniques such as Time-of-Flight cameras [37].

When used to face the bin picking problem, Time-of-Flight cameras usually implement sensors based on the technology of Photonic Mixing Devices, a particular ToF system which presents multiple error sources, but that, with proper calibrations, can reach an overall mean accuracy of 3 mm [38]. Time-of-Flight cameras have been proven to be noticeably fast in object detection, but not as precise as other 3D technologies such as 3D triangulation [39].

### 2.4.2 Dimensional Metrology

Dimensional metrology is another application where Time-of-Flight cameras can make a difference. It consists in the control of the correct size, volume and placement of objects under

study and it is a fundamental step in the quality control of the product. In some application, a simple 2D camera can perform the task, but sometimes a 3D vision system can be necessary or significantly improve the results. The Time-of-Flight camera can be placed over a convoy belt and quickly check dimensions of products certifying if the part is Good or Not Good, giving a binary output [11]. In Figure 11 it is represented a possible application for quality control. The camera in the image is not a ToF camera, but it illustrates correctly how the system would work.

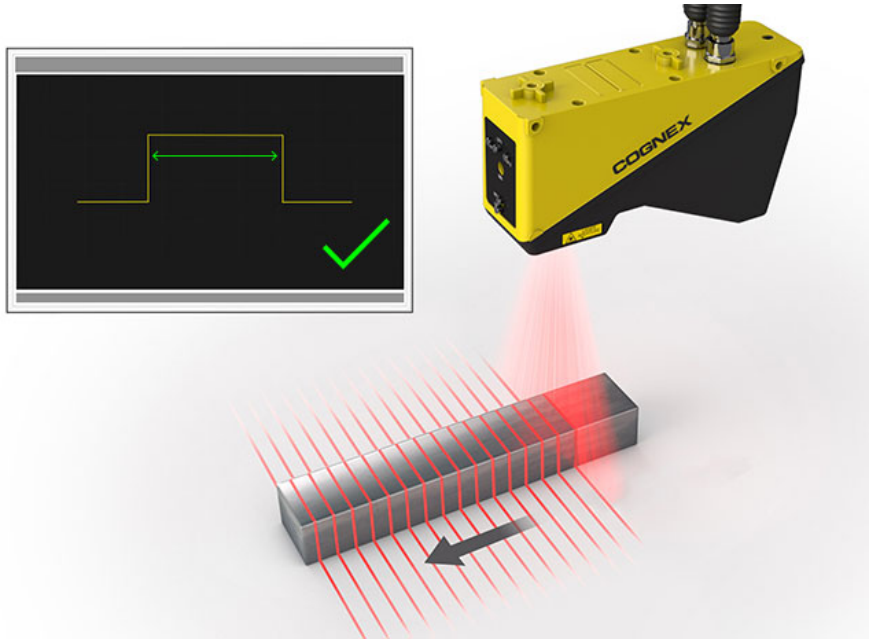


Figure 11: Quality control application [12]

Moreover, the dimension and volume control can impact the logistic choices of a factory, influencing factors such as palletizing or truck transportation. In fact, in an automated palletizing system, robots at the end of the convoy belt need to take a decision on how to place products' pallets, and 3D vision systems, in particular ToF cameras, can be helpful in speeding up the procedure. In loading trucks procedures instead, ToF cameras can be useful avoiding obstacles and in real time monitoring the volume of cargo to realize efficient transportation planning [15].

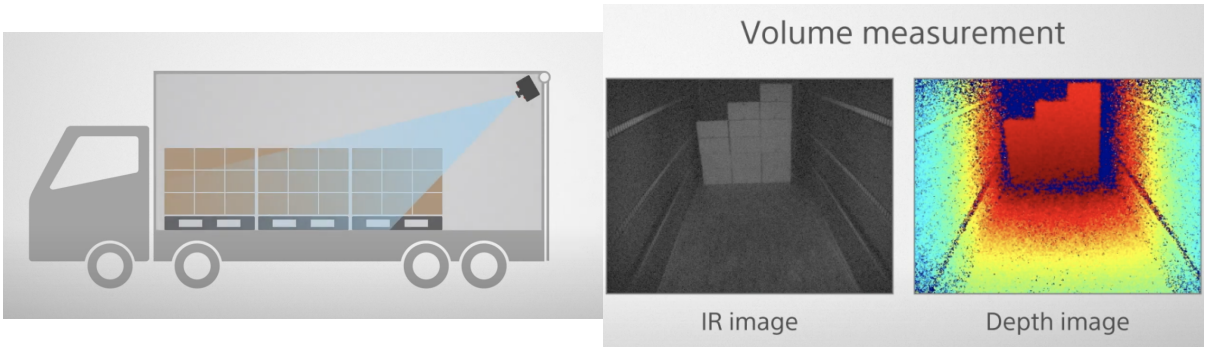


Figure 12: Truck loading application [15]

Figure 13: Palletizing application [16]

Another possible application related to the quality control field is the defect and failure inspection of cast parts. Nowadays ToF cameras do not reach the necessary accuracies requested by the market and that is mainly the reason why some other technologies such as 3D line scanning are being preferred by companies. However, the constant growth of ToF camera will eventually bring this technology to have success in the high accuracy field too.

### 2.4.3 Autonomous Driving

Another possible application where Time-of-Flight cameras have proven to be efficient is in autonomous driving systems, both for in-cabin and outdoor applications. Autonomous driving

consists in a technology that enables a vehicle to sense its environment and work without the help of human intervention.

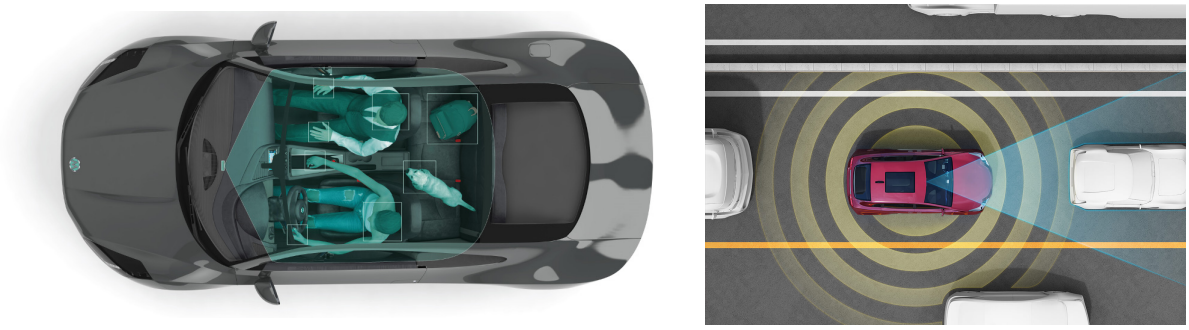

Figure 14: Indoor and outdoor possible driving applications [17] [18]

Consequently, in autonomous driving it is fundamental to obtain 3D information about the world surrounding the vehicle so as to detect and avoid obstacles. To achieve this important task, Time-of-Flight cameras are nowadays being implemented and tested, representing the future of self-driving cars. In fact, Tesla CEO Elon Musk mentioned during Tesla Autonomy Day in April 2019 that "everyone simply relying on LiDAR is doomed and that important steps need to be taken in the direction of computer vision" [40]. The solution could be to develop machine learning softwares that can be trained so as to use cameras to better understand the world around them in order to gain more information from the surroundings, such as if a pedestrian is distracted or if a car is about to turn or not. ToF cameras can be the solution

to this problem since they are able to combine the information of a LiDAR with the image sensing of a simple 2D camera. Moreover, ToF cameras are being efficiently used for in-cabin application too such as airbag pressure regulation or gesture control [41].

### 2.4.4 3D Mapping

Finally, a last application for Time-of-Flight cameras is 3D mapping of indoor or outdoor environments. 3D mapping consists in profiling of objects in three dimensions to map the objects in real world[42]. In general, ToF cameras have not been extensively used in the field of mapping since laser scanners and stereo camera systems have been proven to be highly efficient. On the other hand, ToF ease-of-use, cost effectiveness and compactness allow these types of 3D vision system to be implemented in this application field too [43]. As it has been proven [44], 3D mapping of indoor and outdoor environments is feasible with a ToF camera, after a correct calibration of the system. Moreover, ToF cameras can be implemented in drones designed for building envelope scanning, a procedure that today is done either by human testers or by technologies that present different limitations, such as scaffolding robot manipulators, mobile robots or 3D SLAM systems.

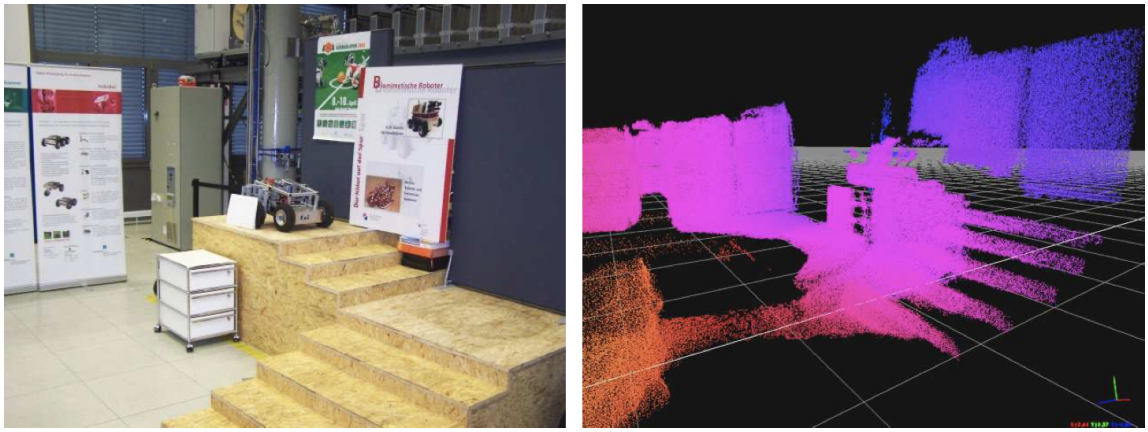

Figure 15: 3D Mapping application [19]

## 2.5 State-of-the-Art in 3D Vision Systems

There are a few companies that lead the way in the market for what concerns Time-of-Flight cameras; among them there is Basler, a German company that sells two different ToF cameras: *Basler Time-of-Flight* and *Basler Blaze*. The former is a convenient camera that offers an attractive price/performance ratio while the latter is the latest and most performing ToF camera produced by Basler. The Basler Blaze main features are the following:

- Working range: 0-10m
- Field of View: (H x V): 67° x 51°
- Accuracy: ± 5mm (0.5-5m)
- Frame rate: 30fps

- Resolution: 640px x 480px / VGA
- Dimensions: 100mm x 81mm x 64mm (L x W x H)

The average price for this camera on the market is around $1600.


Figure 16: ToF 3D vision camera by Basler [20]

Obtaining the Basler Blaze features has been possible thanks to the implementation of the latest sensor produced by Sony, the *DepthSense IMX556PLR sensor*. This sensor has a resolution of 0.3M, a pixel size of 10 $\mu$m x 10 $\mu$m with a size of only 1/2". This back-illuminated, time-of-flight 3D image sensor is perfect to capture accurate depth maps and be used in basically every application mentioned earlier. Sony is a leader in the field of sensors destined to 3D

applications and remarkably helps to meet the increasing demand of a more automated society. After releasing in 2018 IMX556, Sony will release an upgraded sensor in March 2021 called IMX570 with improved features. Among several improvements it is worth mentioning that the size is significantly reduced to 1/4.5" to allow more compact cameras, the pixel size is reduced to 5 $\mu$m x 5 $\mu$m and the modulation frequency range is broader.

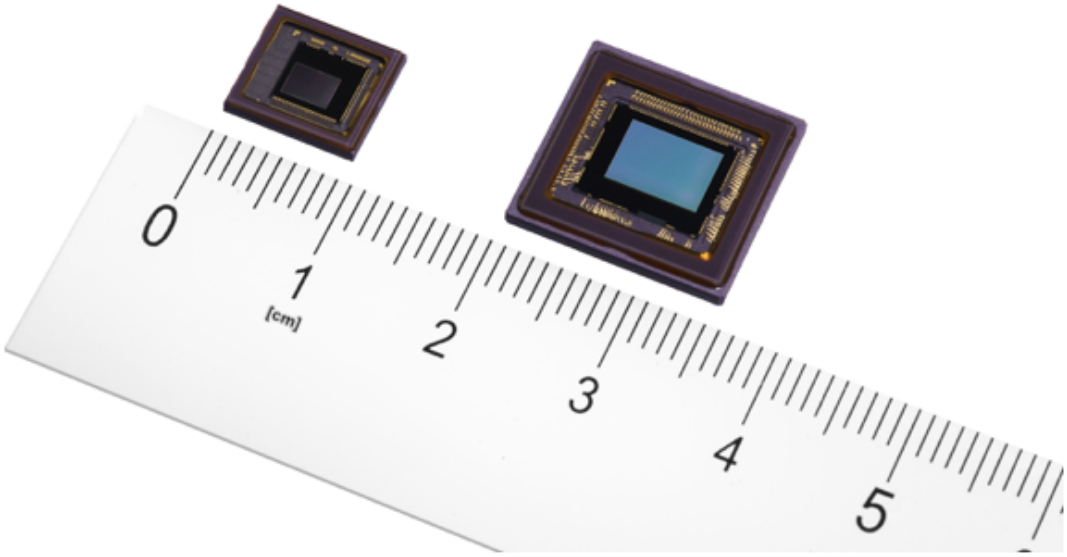


Figure 17: Sony Sensors. *Left*: IMX570. *Right*: IMX556 [21]

These Sony sensors are unique thanks to their incredible ability to capture more efficiently reflected light compared to their competitors. This is done thanks to a back-illuminated image sensor technology that allows a wider incident angle [15].

Another Time-of-Flight camera that employs the Sony DepthSense IMX556PLR sensor is *LUCID Helios2 3D camera*. Launched in July 2020, this camera is a state-of-the-art device with impressive features that make it perfect for all application previously listed. Its main features are:

- Working range: 0.3-8.33m
- Field of View: (H x V): 69° x 51°
- Accuracy: ± 4mm (0.3-1.5m)
- Frame rate: 30fps
- Resolution: 640px x 480px
- Dimensions: 60 x 60mm x 77.5mm (L x H x W)

The price for this ToF camera is $1495.

Figure 18: LUCID Helios2 3D camera [22]

On the market there are several other 3D ToF cameras that have less interesting characteristic, but more affordable prices. Some examples are: *BlasterX Senz3D*, *Microsoft Azure Kinect DK*, *DepthEye Wide* or *Turbo ToF camera*, *Mesa Imaging SR4500*, *Swift-G by Odos Imaging* , *SICK Visionary-T* and *PMD CamCube3.0*.

# CHAPTER 3

# THESIS WORK

## 3.1 Hardware and Software Description

The 3D camera used to perform all the research in the thesis is the Basler Blaze 101 Time-of-Flight (ToF) camera. The main features of the camera, such as the sensor implemented or the main specifications, have been already described at the end of the previous chapter.

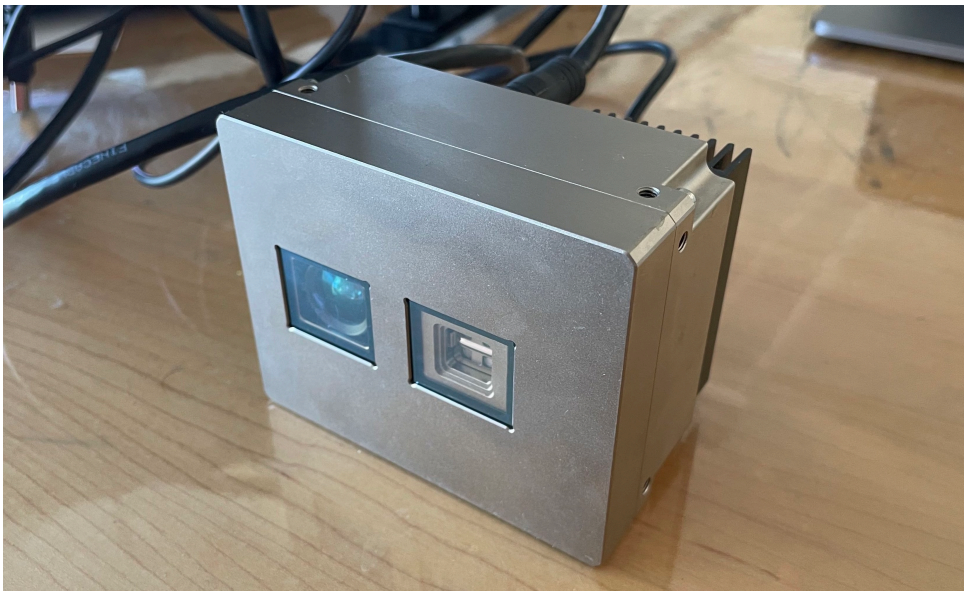

Figure 19: Basler Blaze 101

The camera, in order to be fully functional, needs to be connected to a power source able to provide 22 W and to a vision processor through an ethernet cable. It is important to point out that the camera will communicate with the vision processor thanks to the GenICam protocol, which is a global standard and thus allows the user to use the ethernet cable. There is no need for a lighting controller since the camera operates in NIR and has its own source. Subsequently, in order to access the camera functionalities, Basler provides a Software Development Kit (SDK) downloadable on its online page. The SDK contains useful folders such as "Applications", "Development" and "Documentation". The Application folder contains the Basler blaze Viewer, which helps the user in operating the camera, visualizing the point cloud and depth map live and allowing to change the camera settings to the user specific needs. It has been simply used at the beginning of the study for verification that everything was functioning correctly. Instead, in the Development folder there is a useful folder that provides different C++ prebuilt sample files that teach you how to configure and integrate the main features of the camera in future applications.

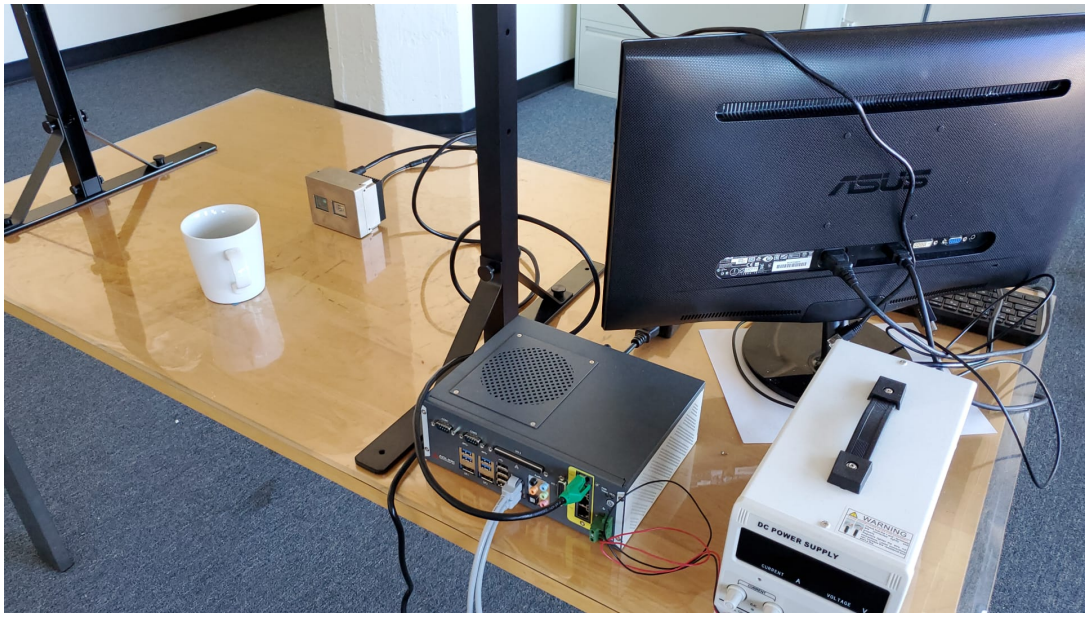


Figure 20: Fully functional setup for Basler Blaze 101 with a sample object

## 3.2 Image Acquisition

Before starting to work on any program, it is necessary to obtain point clouds from the Basler Blaze 101 3D camera. In order to do so, the program "*Blaze101*" has been created programming on C++ on the Microsoft Visual Studio 2019 platform. The code is mainly taken from the sample codes offered by Basler in the SDK folders and it has been slightly modified according to the user necessities.

The parameters that significantly affect the resolution and quality of the point cloud, and that consequently need tuning, are *Confidence Threshold*, *Operating Mode*, *DepthMin*, *DepthMax*

and *Exposure Time*. The Operating Mode can be set either as Long Range or Short Range, depending on the distance of the object under study. If the object is closer than 1.5 meters it is recommended to use the Short Range, otherwise the Long Range is more suitable.

DepthMin and DepthMax are the most intuitive parameters, since they define the depth range of the point cloud. These measures need to be input in millimeters.

Confidence threshold allows us to take into account only pixels that exceed a specified threshold. Basically, the camera calculates an internal confidence value, in a 16-bit register, between 0 and 65335. The confidence value will increase if the amount of light collected by the pixel increases as well. Having a higher confidence means that the camera system is more certain that the measured point is correct.

Finally, the exposure time is a parameter that controls for how long the photosensitive cells are exposed to light. Basler suggests an exposure time of 250 ms for the Short Range application, while an exposure time of 1000 ms for the Long Range one. The practice of reducing the exposure time is necessary only when the image is overexposed since it may reduce measurement accuracy. This parameter needs to be set in microseconds.

After tuning these parameters in order to get the best point cloud with the highest quality and resolution, the C++ code can be built and launched. It will take the picture and save it as a .PCD file in the same folder containing the program.

## 3.3 Point Cloud Library

After having obtained the points clouds, in order to build programs that can process and work with these type of file, the open source library PCL (Point Cloud Library) is used.


Figure 21: Point Cloud Library [23]

The Point Cloud Library is an open source library originally released in March 2010 by Willow Garage, a robotics research lab located in Menlo Park, California. It contains algorithms for filtering, segmentation, feature estimation and more [45]. In order to make PCL run correctly the user needs to install third-party libraries:

1. *Eigen* library to perform mathematical operations;
2. *VTK* to visualize the Point Clouds properly;
3. *Boost* for shared pointers;
4. *FLANN* for quick k-nearest neighbor search.

PCL is able to save file in the PCD format, the most common way to store 3D point cloud data, but it is able to save and load data in many other formats such as PLY, IFS, VTK, STL, OBJ or X3D.

The official website *https://pointclouds.org* offers clear and detailed instruction on how to pro-

ceed to install and run PCL on different operating systems.

The operating system used in this thesis is Windows.

## 3.4 Registration

The process of aligning various 3D point cloud views into a complete model is known as Registration. This technique finds the relative pose (position and orientation) of the different point of views so as to make the intersecting areas overlap as much as possible. The final goal is to merge all these point clouds into one single point cloud where the user can subsequently apply processing steps such as segmentation or object detection.

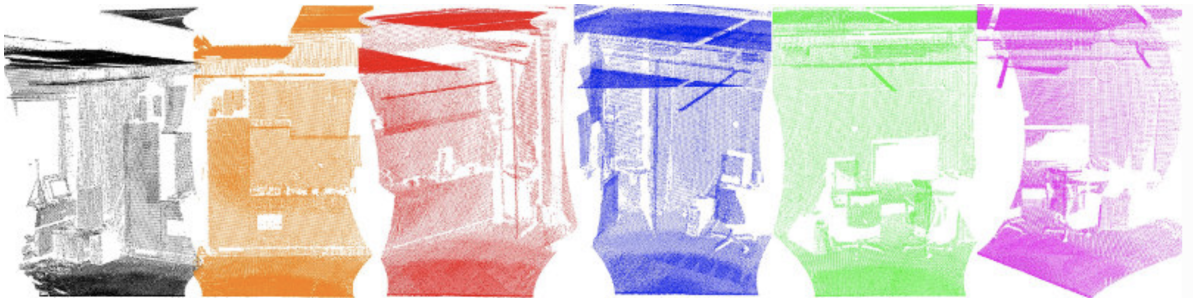

Figure 22: 6 different Point Clouds taken from different view points [24]

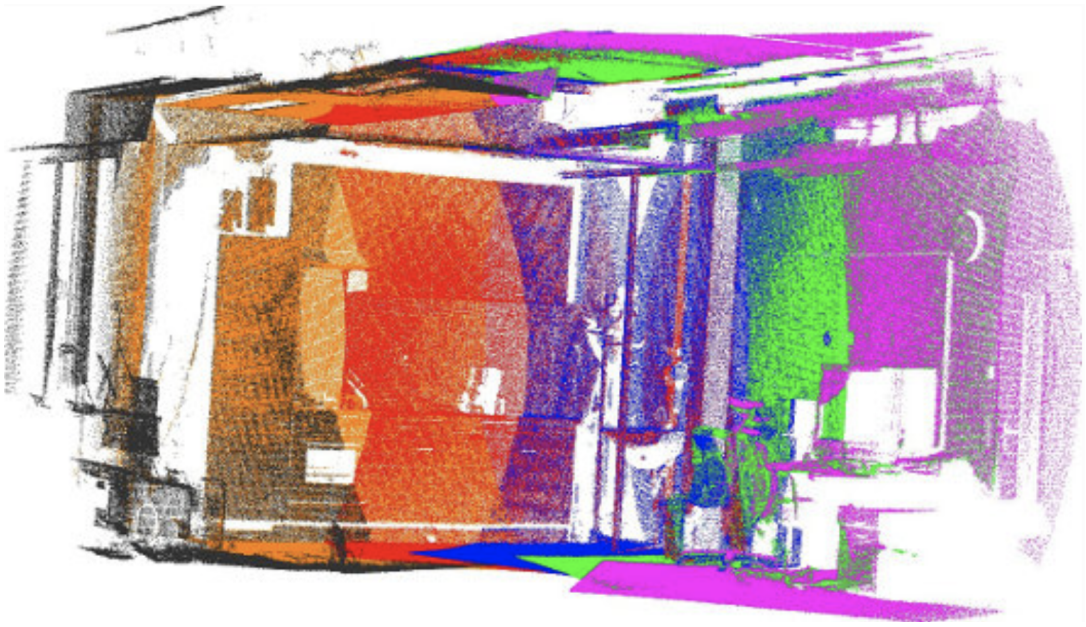

Figure 23: Registered Point Cloud model [24]

The particular 3D Registration technique applied in this thesis is called *Pairwise Registration of 3D point clouds*, that consists in aligning two point clouds at a time. The output of the algorithm is typically a rigid transformation matrix (4x4) which represents the translation and rotation that needs to be applied to one point cloud (selected as *source*) to perfectly match and overlap the second point cloud (selected as *target* or *model*).

The pipeline used to perform pairwise registration is illustrated in Figure 24.

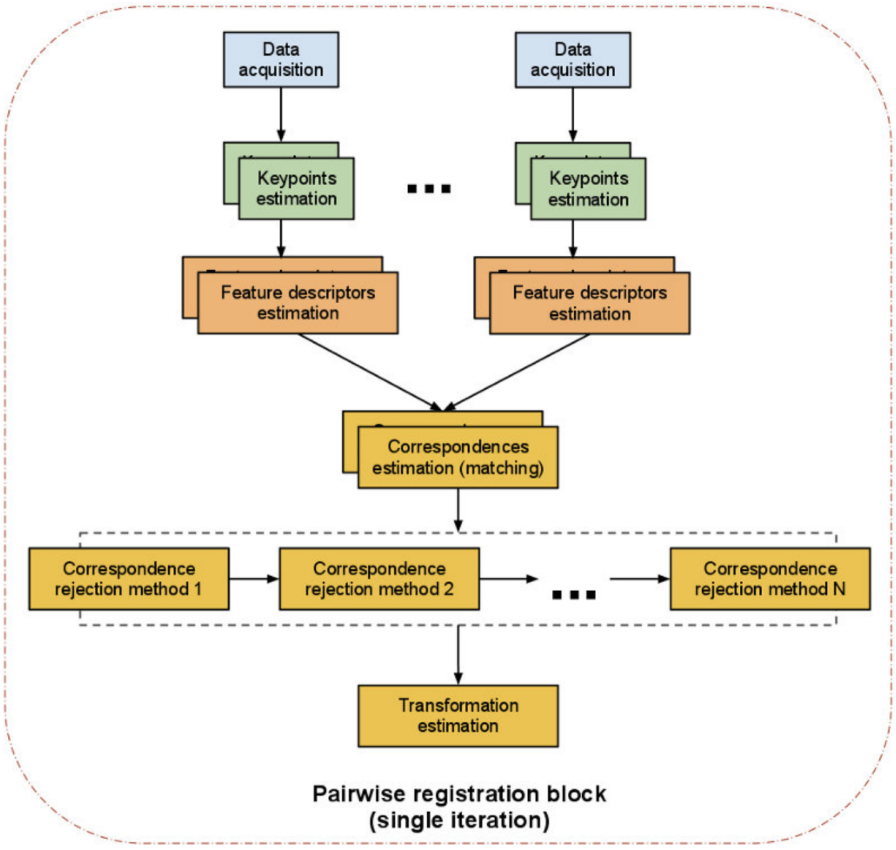


Figure 24: Registration pipeline [24]

### 3.4.1 Data acquisition

The first step is the data acquisition. It consists in obtaining correct, precise and clean point clouds of the object under study. First of all, it is necessary to utilize the Basler Blaze 101 3D ToF camera to get the pictures that will be used later in the registration process. The program "Blaze101" is used and the PCD file is saved in a folder selected by the user.

Subsequently, the PCD file needs to be *downsampled* so as to make it less heavy for the computer for future post-processing operations. In order to downsample the file, the program "*voxel_grid*" is created. The program takes as input the original full file and outputs the new PCD file downsampled, saving it to the same folder of the program. The most important parameter is the *LeafSize*, which needs to be tuned according to the user necessities. *LeafSize* takes three values, one for each of the axis x, y and z. It basically creates voxels (i.e. 3D boxes) with the specified dimensions, and all the points present inside these boxes will be approximated (i.e. downsampled) with their centroid.

Finally, so as to properly proceed with the registration process, it is important to remove everything that the user does not want to register, such as close objects or the table on which the object is lying on. In order to accomplish this task, the program "*remove_table*" has been created. It takes the downsampled file as input and it outputs the new PCD file. The first thing that the program does is create a pcl::SACSegmentation object, setting the model and method type. The model that the algorithm needs to look for is a plane, and the method is RANSAC. Random sample consensus (RANSAC) is an iterative method whose goal is to determine parameters of a mathematical model (the plane) from a dataset (the point cloud) [46]. In this part of the code, it is also possible to specify the distance threshold, which determines how close a point must be to the model to be considered an inlier. The code will find all points belonging to the biggest plane model, which is the table, and will remove them from the point cloud. Subsequently, the code implements a passthrough filter on the X axis and on the Z axis. These filters allow to remove all the points that were not removed by the

plane removal algorithm and that do not belong to the object under study. Finally, a statistical and radius filter can be applied if the result is still not satisfying.

### 3.4.2 Keypoints estimation

All the following steps are implemented in the program called "*Registration*".
The second step in the registration process is finding keypoints. A keypoint is a special point, also called "interest point", that best represents the scene in the dataset. PCL offers several ways to find keypoints such as with NARF, SIFT and FAST keypoints algorithms. In this thesis, the keypoints estimation step has been carried out through a uniform sampling of the cloud. This choice reflects the intent of the program to make the process as generalizable as possible, i.e. not relying on specific keypoints that work better on certain cloud datasets rather than other. This has been done in the code by using UniformSampling class, which creates a local 3D grid over a given PointCloud. In each 3D box, all the points present will be approximated with the closest point to the center of the voxel [47]. It can be considered a random way to get uniformly distributed keypoints. In this part of the code, it is necessary to tune the *radius_keypoint* parameter, which represents the nearest neighbors search radius for each point.

### 3.4.3 Feature descriptors estimation

The third step is to estimate feature descriptors for each of the found keypoints. In fact, using only keypoints will not allow the user to perform a proper registration since they do not carry enough information with themselves: only their XYZ coordinates; on the other hand, feature descriptors will help distinguish points since they will have similar values when calculated

for the same area in different point clouds.

The normals of each keypoint are an example of a very simple feature. Feature descriptors, instead, consist of a "more detailed signatures of points" [48] that will help later on to determine correspondences between the two point clouds.

There are two types of descriptors: *local* and *global descriptors*. A global descriptor describes an entire object or a whole point cloud, while the local descriptor is a representation of a point's local neighborhood and encodes a lot about the surrounding geometry, thus becoming suitable and useful when it comes to identify and describe various points and match them [30].

After the algorithm calculates the necessary values for the descriptor, the result is binned into a histogram so as to reduce the descriptor size. Each value range of each descriptor's variable is divided into n subdivisions, and the number of occurrences in each subdivision is counted [49].

The Fast Point Feature Histograms (FPFH) descriptors are the local descriptors that have been used in the code. FPFH is born as a simplification of the Point Feature Histogram (PFH) formulation and it reduces the computational complexity of the algorithm from $O(nk^2)$ to $O(nk)$ [25]. The PFH algorithm and theoretical aspects will be explained first and then the improvements brought by FPFH will be described.

#### 3.4.3.1 Point Feature Histograms (PFH)

In order to estimate PFHs it is necessary to start with having 3D coordinates and surface normals of each keypoint. The subsequent steps that need to be followed are:

1. All the points enclosed in a sphere of a given radius $r$ and centered in point $p$ are selected (k-neighborhood);

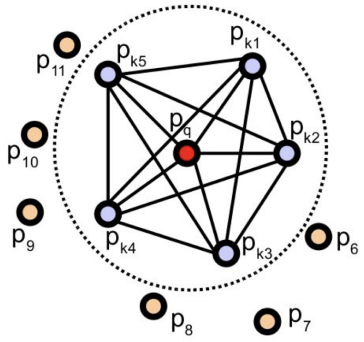


Figure 25: K-neighborhood of point $p$ [25]

2. For every pair of points $p_s$ and $p_t$ in the k-neighborhood of $p$ a $uvw$ frame is defined and the difference between the two normals $n_s$ and $n_t$ are calculated thanks to their angular variations [25].

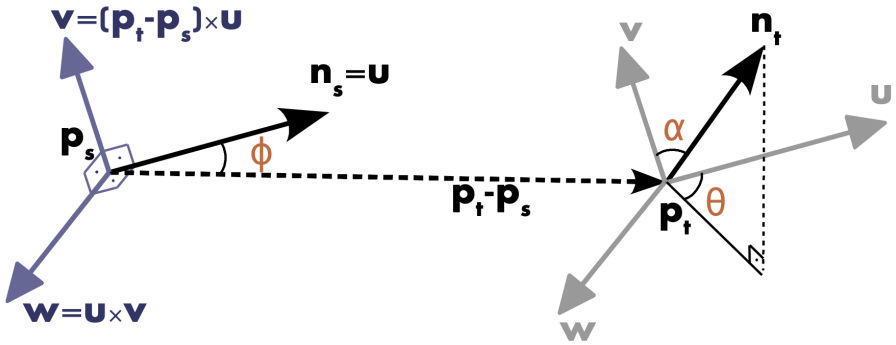


Figure 26: Graphical representation of points $p_s$ and $p_t$ with their normals and their angular PFH features [26]

The frame is centered at $p_s$ and is defined as:

$$\begin{cases} u = n_s \\ v = u \times \frac{(p_t - p_s)}{||p_t - p_s||_2} \\ w = u \times v \end{cases} \tag{3.1}$$

While the angles are:

$$\begin{cases} \alpha = v \cdot n_t \\ \phi = (u \cdot (p_t - p_s))/||p_t - p_s|| \\ \theta = arctan(w \cdot n_t, u \cdot n_t) \end{cases} \tag{3.2}$$

#### 3.4.3.2 Fast Point Feature Histograms (FPFH)

The simplification introduced with the FPFH reduces the computational complexity of the algorithm, but manages to keep most of the discriminative power of the PFH. The procedure for FPFH is the following:

1. For each point $p$ calculate the relationships only between itself and its neighbors. This is then called the Simplified Point Feature Histogram (SPFH).
2. For each neighbor calculate their SPFH and finally use these values to weight the final histogram for the original point $p$. The final formula that summarizes this concept is the following:

$$FPFH(p) = SPF(p) + \frac{1}{k}\sum_{i=1}^{k}\frac{1}{w_k}\cdot SPF(p_k) \tag{3.3}$$

where the weight $w_k$ represents the distance between point $p$ and a neighbor $p_k$ [25].

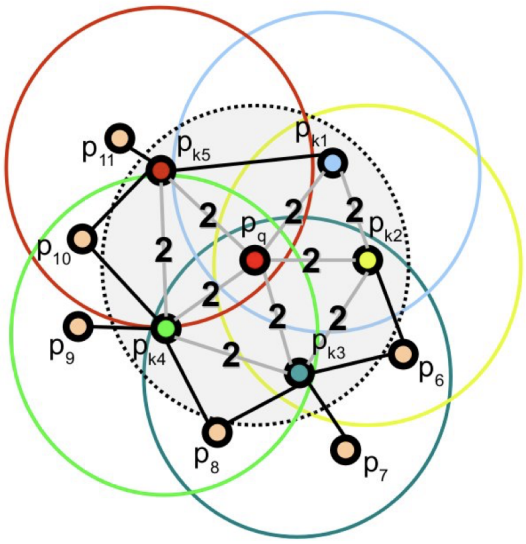


Figure 27: New region of influence of point $p$ in FPFH algorithm [25]

In the code, the FPFH function requires the keypoints and their normal as input as well as the sphere radius that is used for determining the nearest neigbhours used for feature estimation (*radius_fpfh*). The normals are computed thanks to the *NormalEstimation* class, where the parameter *radius_normals* is tuned. The class *FPFHEstimation* then proceeds in calculating

the feature descriptors. The parameter that needs to be tuned is *radius_fpfh*, which represents the nearest neighbors search radius for each feature descriptor.

### 3.4.4 Correspondences estimation (matching)

The fourth step in the registration process consist in finding correspondences between the keypoints in the two point clouds. This is made using the *CorrespondenceEstimation* class, which is going to determine all reciprocal correspondences. In order to work it has to receive the source and target FPFH descriptors of the two point clouds as an input. The algorithm simply associates each keypoint on one cloud to its nearest neighbor keypoint on the other cloud based on the fact that their descriptors are in the same descriptor space [30].

It is clear that not all the areas of the two point clouds will be overlapping and this is why, after having found all the correspondences, it is necessary to reject the bad ones. This is made through the *CorrespondenceRejectorDistance* class. This class implements a simple correspondence rejection method based on thresholding the distances between the correspondences. Correspondences more distant than the parameter *maxDistanceCorrespondences* will be removed. Finally, the code is now ready to calculate the 6DOF transformation of the source file to match the target one. This is done by the *TransformationEstimationSVD* class, which takes the good correspondences, the source and the target keypoints as inputs.

### 3.4.5 ICP

Iterative closest Point (ICP) is an algorithm used to minimize the distance between points of the source and the target point clouds. It can be considered a "brute force method" since it pairs every point from one cloud to its "twin" and tries to reduce their distance [50].

ICP has been used as final step in the pairwise registration process to try to refine the results previously obtained with the 6DOF matrix. In the code the inputs are, first, the source cloud where the transformation matrix has already been applied and, secondly, the target point cloud. The algorithm will then proceed to [51]:

1. For each point in the source cloud finds the closest match in the target cloud based on distance;
2. Transform the source point cloud minimizing a root mean square point to point distance metric;
3. Iterate the previous step until the metric used reaches a global minimum

### 3.4.6 Remarks on Registration

ICP is the last step of the registration process adopted in this thesis and is implemented in the program "*icp_simple_live*", which provides a live visualization of the point clouds aligning. The source and target point cloud are now correctly aligned. Since the registration is a *Pairwise Registration* method, the program aligned only two of the multiple point clouds taken of the object. It is instead necessary to iterate this process until all views of the object are integrated in the final model. Consequently, the two aligned point clouds are merged into one that will become the new source point cloud for the following registration iteration. After finishing this process for all the points of view the user will have a complete 3D model of the selected object.

## 3.5 Geometry evaluation of point clouds

After acquiring a complete point cloud model of an object, the user may want to get basic geometry information out of it, such as length, height and width. In order to accomplish this task, the program "*Geometry*" has been created.

A quick method to obtain reasonable accurate geometry measures is to build a bounding box around the object. After having built a bounding box the user can obtain information on the width, depth and height of the box: these measures can be used to describe the enclosed object too. The bounding box needs to be correctly oriented so as to be the one with the smallest possible volume and so as to be the one that correctly describes the dimensions of the object. The bounding box that respects these characteristics is called *Oriented Bounding Box (OBB).* In PCL there is a class that is able to extract the necessary information to properly build an OBB: the *MomentOfInertiaEstimation* class. After using this class, the procedure is quite simple since the user only needs to call the function "*getOBB*" to get all the necessary data to build the box.

Instead of using the pre-built functions in PCL, the user can manually obtain the same results following some easy steps. The procedure originates from user Nicola Fiorario on the PCL forum [52]. The first step would be to calculate the centroid and the normalized covariance matrix of the point cloud. Subsequently a Principal Component Analysis (PCA) is performed: the three eigenvectors of the normalized covariance matrix are found and they are used to form a new reference system. The new reference system will be ($e_0$, $e_1$, $e_0 \times e_1$) (note: $e_0 \times e_1 = \pm e_2$). The eigenvectors are used to transform the point cloud to the origin point

with the eigenvectors corresponding to the axes of the space. Subsequently, for all axis, the maximum point, the minimum point and the middle of the diagonal between these two points are calculated for the transformed cloud. Finally, the user will build a box centered at the origin with size $(max\_pt.x - min\_pt.x, max\_pt.y - min\_pt.y, max\_pt.z - min\_pt.z)$. In order to enclose correctly the point cloud the box needs to receive a rotation and a translation too, that are calculated thanks to the eigenvectors. The rotation is in quaternions and is ($e_0$, $e_1$, $e_0 \times e_1$), while the translation is obtained through: Rotation $\times$ center of diagonal + centroid.

## 3.6 AutoCAD model creation

If a more precise and accurate study on the object's geometry is required, it is possible to create an AutoCAD model from the final registered point cloud and perform additional studies on the CAD software. The programs "*Triangulation*" and "*AutoCAD*" have been created to solve this problem.

The algorithm used to perform this task and create the mesh models is the *Greedy Projection Triangulation* algorithm.

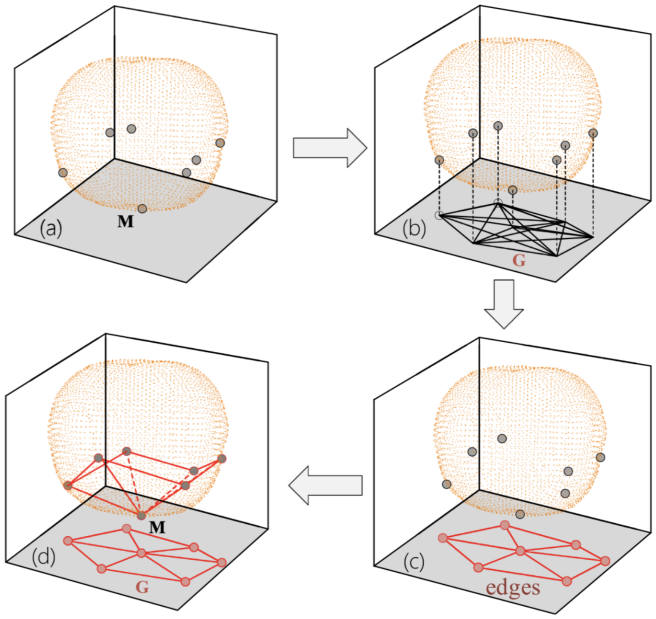


Figure 28: Greedy projection triangulation algorithm [27]

The process can be divided into 4 steps [27][53]:

1. Nearest neighbor search: Given a reference point M in the point cloud, the program looks for all the closest neighbors, with a maximum number given as input by the user.

2. Point cloud projection: Point M and its k-neighborhood are projected to the tangent plane which is perpendicular to the normal vector generated by point M. All points belonging to the k-neighborhood are projected to this plane and they form the so-called point set {G}.

3. Triangulation in plane: The greedy triangulation takes place on the tangent plane. It follows a greedy schema, which consists in adding every edge starting from the shortest one in increasing order by length, with the condition that an edge can not cut an already existing edge [54].

4. Triangulation re-projection: After obtaining the mesh on the tangent plane, the triangulation is returned to the 3D space. Every step of the algorithm is then repeated for all reference point in the cloud until the full mesh model is ready.

In the code there are some parameters that need to be set and tuned to specify the wanted features and to obtain the best possible mesh:

- *setMaximumNearestNeighbors*: It defines the maximum number of neighbors to consider when performing the greedy triangulation;
- *setMu*: It sets the multiplier of the nearest neighbor distance to obtain the final search radius. It is used in order to adapt to point clouds with different densities;
- *setSearchRadius*: It defines the maximum distance between connected points. Consequently it sets the maximum triangle edge length;
- *setMinimumAngle* and *setMaximumAngle*: They set the maximum and minimum angles of triangles;
- *setMaximumSurfaceAngle* and *setNormalConsistency*: If normal are not oriented consistently, the former sets the maximum angle between normals to still consider points for triangulation. It is useful when there are sharp edges or corners.

After reconstructing all the triangles of the model, it is likely that some holes are still present in the mesh. In order to overcome this problem, the *fillHolesFilter* class is helpful. The user can input the maximum area of the missing triangle and the program will proceed to fix the holes.

Since the Point Cloud Library is not able to provide a file format that is readable by AutoCAD, the *Visualization and Computer Graphics Library (VCG)* has been used. VCG is an "open source portable C++ templated library for manipulation, processing and displaying with OpenGL of triangle and tetrahedral meshes" [55]. This library allows to receive the .PLY file, output by PCL, and export it into a .DXF file, which can be easily opened on any of the latest AutoCAD versions.

## 3.7 Segmentation

Segmentation consists in the process of dividing a point cloud in multiple segments or pieces, which take the name of *clusters*. The goal is to separate objects in a scene in order to perform different studies or analysis on them indipendently. There are several techniques that can be used to perform segmentation on a point cloud, such as based on point distance, point normals or even texture [56]. The method used in this thesis is one of the simplest: the *Euclidean Segmentation*, which is based on the distance between two points. The program "*Segmentation*" has been created to perform this task.

First of all, the program starts by eliminating possible tables, planes and points that the user does not need to keep in his final file. This is made in a really similar way it has been done at the beginning of the registration process. This time the "*remove_table*" code is implemented in

a while loop so as to find multiple planes and not just only the biggest one. Following this step, the Euclidean segmentation algorithm analyzes the distances between points, and if it is lower than a specified threshold, they are considered belonging to the same cluster. The algorithm goes on following this step until no more points can be added. It then creates a new cluster and the procedure starts again until all points in the cloud have been considered [56].
In the code there are some parameters that need to be set and tuned to specify the wanted features and to obtain the best possible segmentation:

- *setClusterTolerance*: It defines the search radius for neighbors;
- *setMinClusterSize* and *setMaxClusterSize*: They define the boundaries for the number of points in the cluster.

The program also contains a section where the point cloud is filtered and the eventual plane (table) is removed from the scene. After finally running the program, the clusters found will be saved in the same folder of the program with different names.

## 3.8 Object Recognition

3D Object Recognition is the technique through which the user is able to locate and recognize objects from a 3D point cloud. Basically, the problem consists in finding the object (model) in a scene and being able to locate it. In this thesis, correspondence grouping algorithms, contained in the pcl_module, have been used to cluster the correspondences found in the model that are present in the scene. The program "*Object Recognition*" has been created to perform this task. The process can be seen as quite similar to the registration process previously described in

the thesis since it uses the local shape feature-based matching paradigm too. In the object recognition application, the local feature-based matching proceeds finding keypoints and feature descriptors of the cloud, then generating an initial set of correspondences between the model and the scene. However, there will be a high amount of false positive due to multiple reasons such as noise, clutter, occlusions and overlaps in the scene point cloud [28]. At this point, the correspondence grouping algorithm is used to filter and find the correct correspondences.

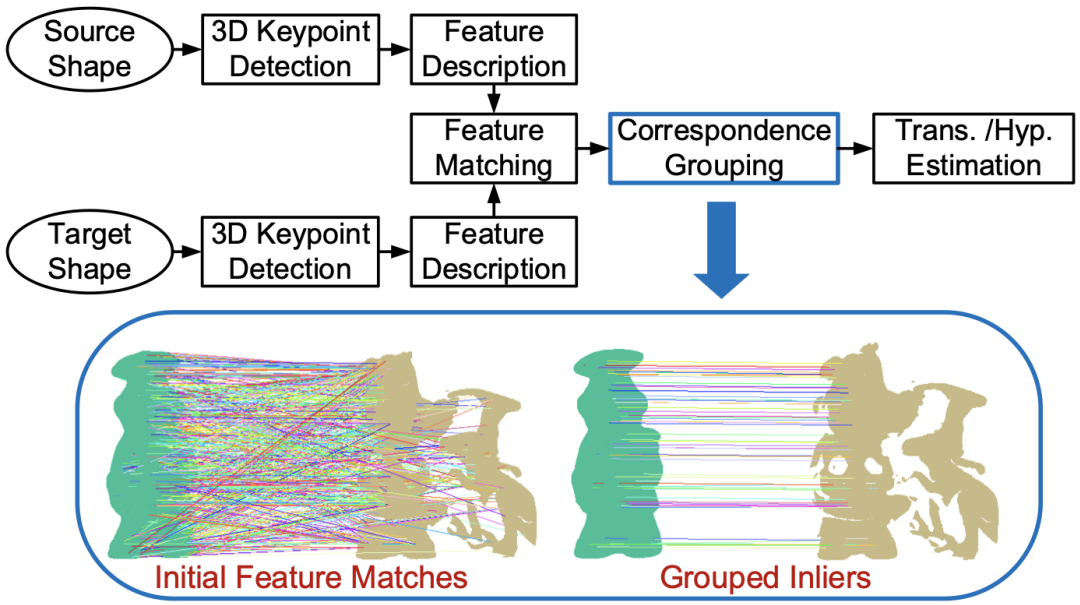


Figure 29: 3D Object Recognition algorithm [28]

The correspondence clustering algorithm used in the program takes the name of Hough3DGrouping and it is based on a 3D Hough voting scheme described in [29]. In the algorithm proposed by Tombari and Di Stefano each correspondence casts a vote in 3D Hough space.

For every $i$th correspondence denoted by $c_i = \{p_i, p_i^{'}\}$, the vector between the source centroid $C_S$ and the model feature point $p_i$ is calculated in the coordinates of the global reference system of the source as:

$$\mathbf{V}_{i,G}^{S} = C_S - p_i \tag{3.4}$$

then it is transformed in the local reference system (LRF) of $p_i$ as:

$$\mathbf{V}_{i,L}^{S} = \mathbf{R}_i^S \cdot \mathbf{V}_{i,G}^{S} \tag{3.5}$$

where $\mathbf{R}_i^S$ is the rotation matrix where each line is a unit vector of the LRF of the feature $p_i$. The LRF is an independent coordinate system created around the surface of the keypoint and gives the vector invariance to rigid transformation.

Given that $p_i$ and $p_i'$ are a good correspondence, vectors $\mathbf{V}_{i,L}^{S}$ and $\mathbf{V}_{i,L}^{S'}$ should coincide. Consequently we can proceed to calculate $\mathbf{V}_{i,L}^{S'}$ for $p_i'$. Finally, the vector $\mathbf{V}_{i,L}^{S'}$ is transformed in the global reference system of S' as:

$$\mathbf{V}_{i,G}^{S'} = \mathbf{R}_i^{S'} \cdot \mathbf{V}_{i,L}^{S'} + p_i' \tag{3.6}$$

now the feature $f_i'$ can vote for the centroid position in a tiny 3D Hough space thanks to vector $\mathbf{V}_{i,G}^{S'}$. Consequently, the presence of a correct centroid (and consequently of a particular object) can be confirmed by analyzing the peaks of the Hough space [28][29].

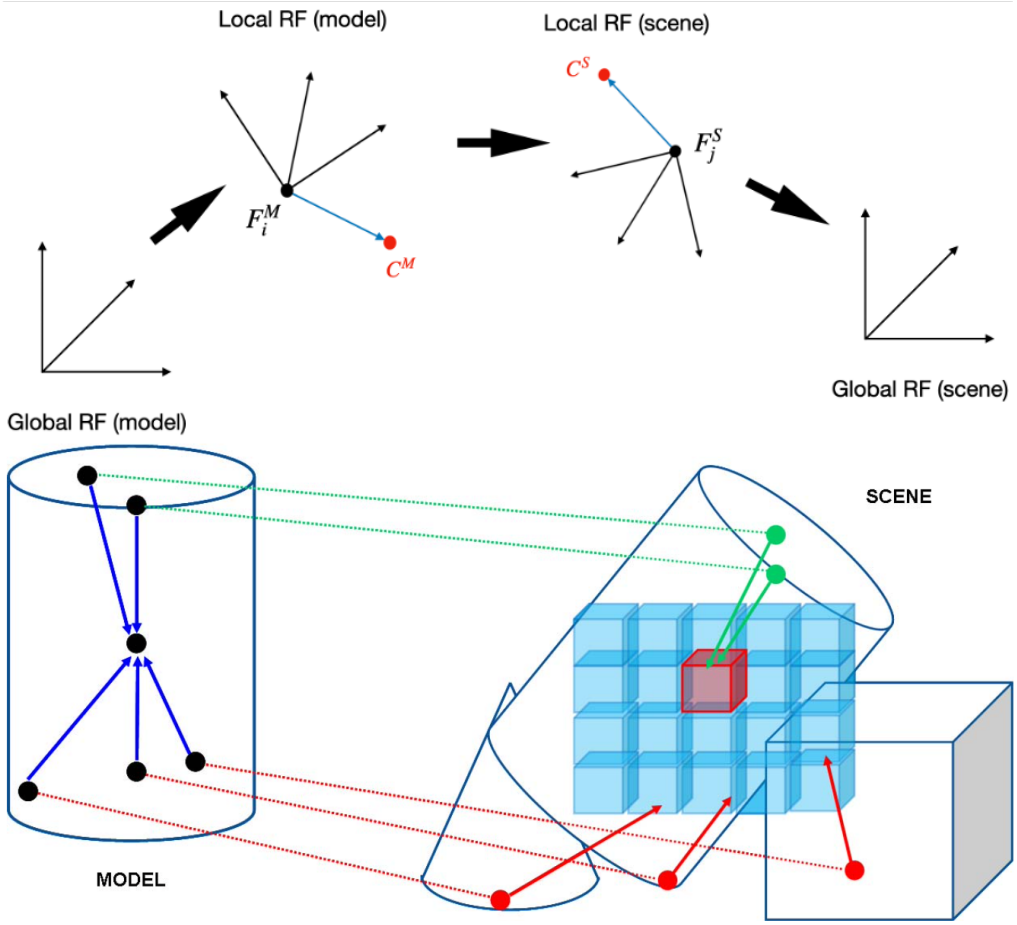


Figure 30: Reference systems transformations (*top*) and 3D Hough Voting scheme [29] (*bottom*)

Figure 30 can give a visual idea of the algorithm and of the voting process. The top figure illustrates the different changes in the reference system, while the bottom figure shows the voting process. The green vectors will cast a vote in the correct red 3D Hough space of the centroid. The red wrong vectors will cast votes too, but, at the end, the amount of votes in the correct hough space will be higher consequently forming a peak in the position of the correct centroid. This is how the object is found.

Another important thing to mention about the object recognition algorithm is that the feature descriptors used are not the FPFH anymore, as in the registration process, but the SHOT descriptor has been implemented.

The SHOT descriptor is a 3D descriptor that successfully combines the robustness to noise of histograms with the discriminative power of the descriptor [57].

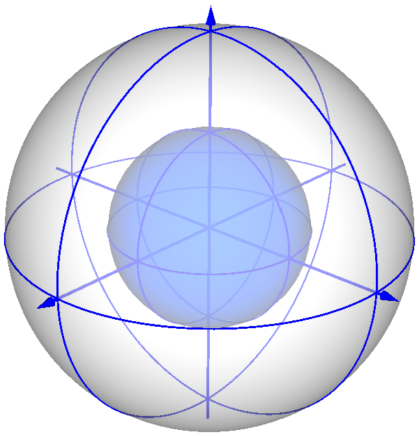

Figure 31: SHOT Descriptor [30]

The SHOT descriptor relies on a 3D grid centered on the keypoint and oriented according to a unique LRF. In every section of the grid, the angles between the keypoint normal and the normal of each point inside that sector is accumulated to build a histogram. The final descriptor is obtained by putting together all obtained histograms [30].

Finally, there are some parameters in the code that needs tuning in order to get the best possible result:

- *model_ss* and *scene_ss*: they represent the nearest neighbors search radius of the sphere used to determine keypoints in the uniform sampling part of the code.
- $rf\_rad$: It is the LRF support radius. It is used to find the local reference system of the keypoints.
- *descr_rad_*: It is the SHOT descriptor radius. It is used when calculating the feature descriptors of the keypoints.
- *cg_size*: It defines the spatial length of each Hough bin. In order to be set correctly it should be enough large to solve the problem of noise of the 3D Hough votes but not too large to create too many useless peaks in the Hough space.
- *cg_thresh_*: It represents the threshold for the number of votes received of each object hypothesis.

After giving as input the model and the scene, the program will calculate the position of the model and it will superimpose it in red the scene point cloud. There are further options for the visualization of results, such as the option to see keypoints, correspondences or even the

connections between the model and scene correspondences. How to use all these options is well explained in the code comments.

# CHAPTER 4

# RESULTS

## 4.1 Registration Results

The objects that have been studied and registered are a shoe and a little model of a caterpillar excavator.

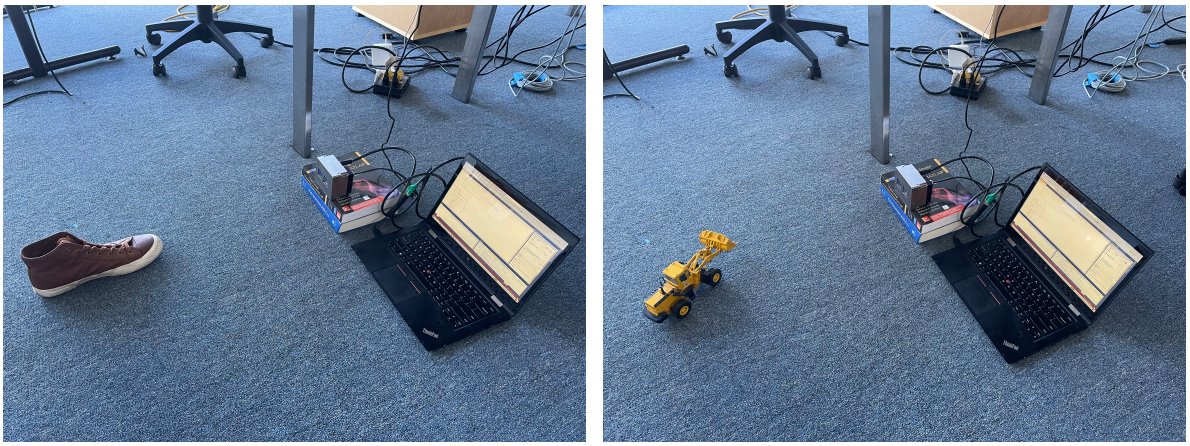

Figure 32: Setup for shoe and caterpillar excavator Registration process

The downsampling process for all the objects in the thesis has been usually done with a *LeafSize* of (5,5,10) mm. An example of the results obtained for the first step of downsampling the multiple views of the object is the following:

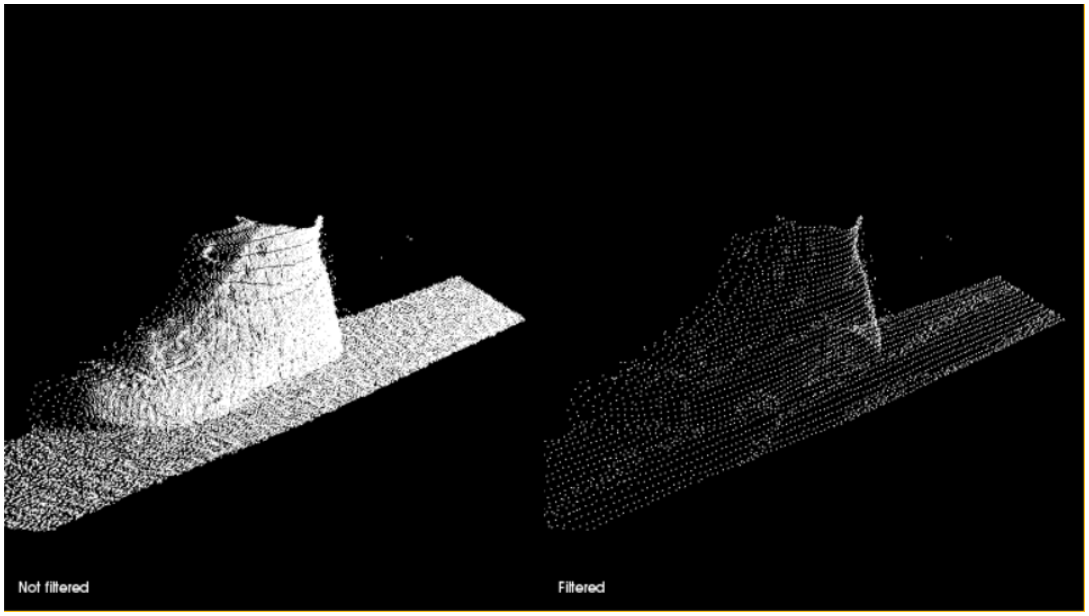


Figure 33: Left: original point cloud, Right: downsampled point cloud

Subsequently, the plane has to be removed together with everything that is not related with the object the user wants to register. Together with the plane removal, the cloud is also filtered and corrected by removing outliers present because of possible ToF camera imprecisions.
In Figure 34 the red points have been removed and identified as a plane, while the green points are saved as a new final point cloud of the object. The grey points have been removed by the statistical filters applied to the cloud.

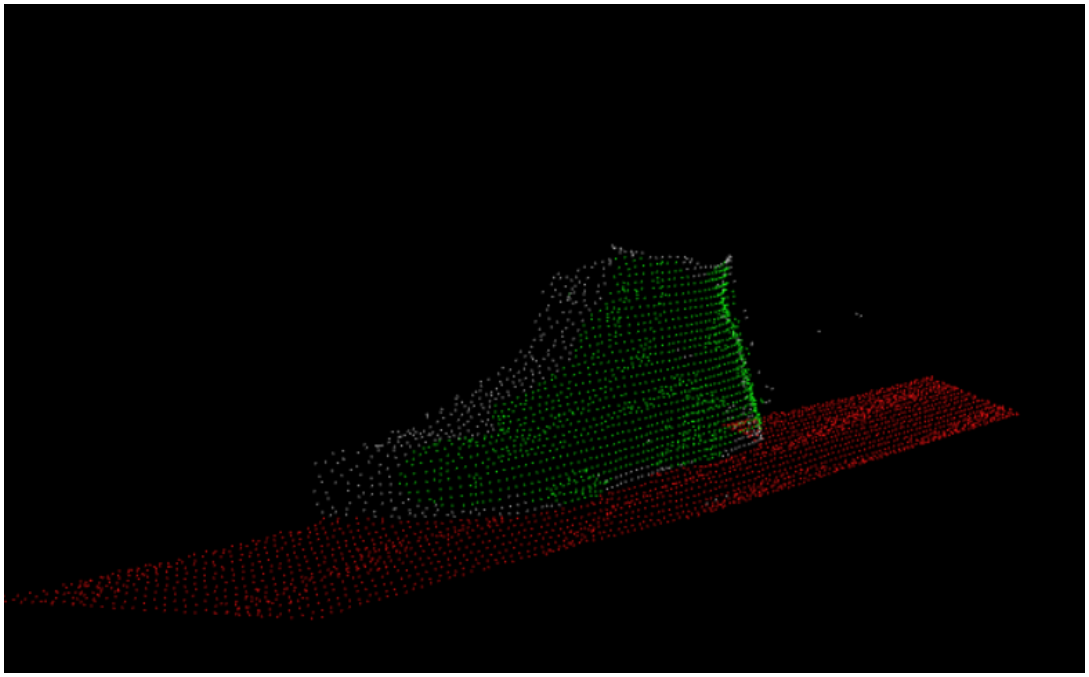

Figure 34: Example result of the plane removal algorithm

After applying these pre-processing steps to all the clouds for all viewpoints of both the shoe and the excavator, the actual registration process and ICP final iteration takes place. As it can be seen in Figure 35, the two point cloud are aligned by the initial registration algorithm. However, they are not perfectly aligned on the back of the shoe, this is why the user needs to use the ICP process to refine the obtained results. In fact, after the ICP process the results are clearly more acceptable as shown in Figure 36.

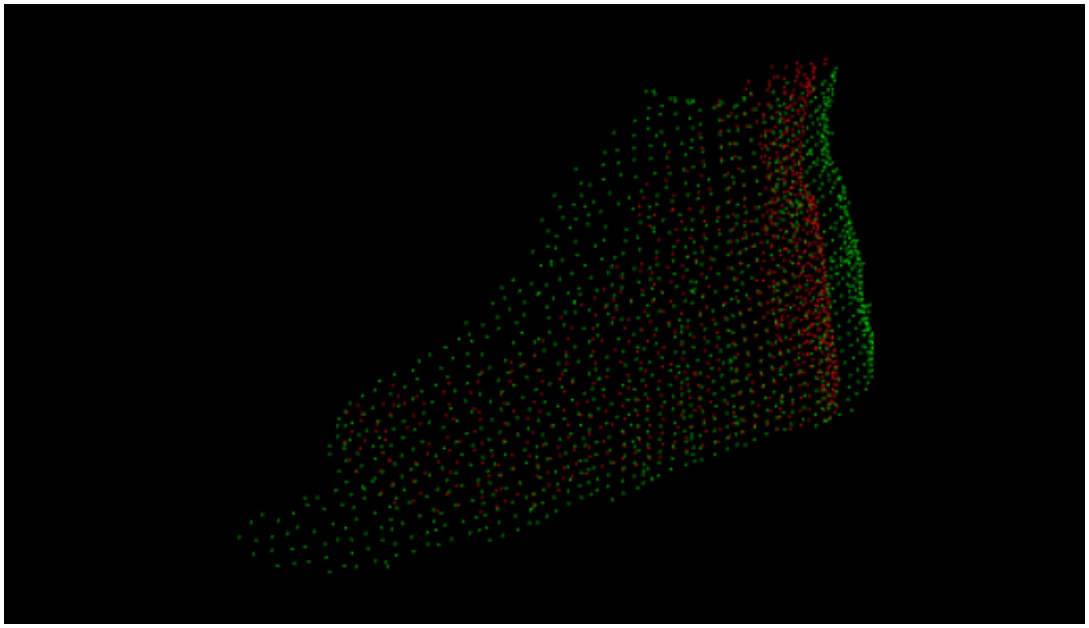

Figure 35: Example result of two shoe point cloud registered

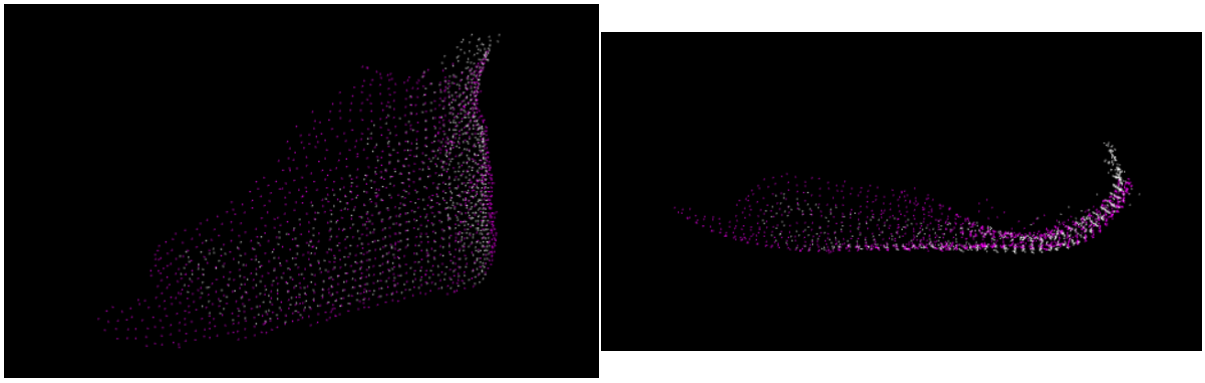

Figure 36: Example result of two shoe point cloud registered after ICP

The final results obtained for the shoe and the excavator have been successful and they are shown below:

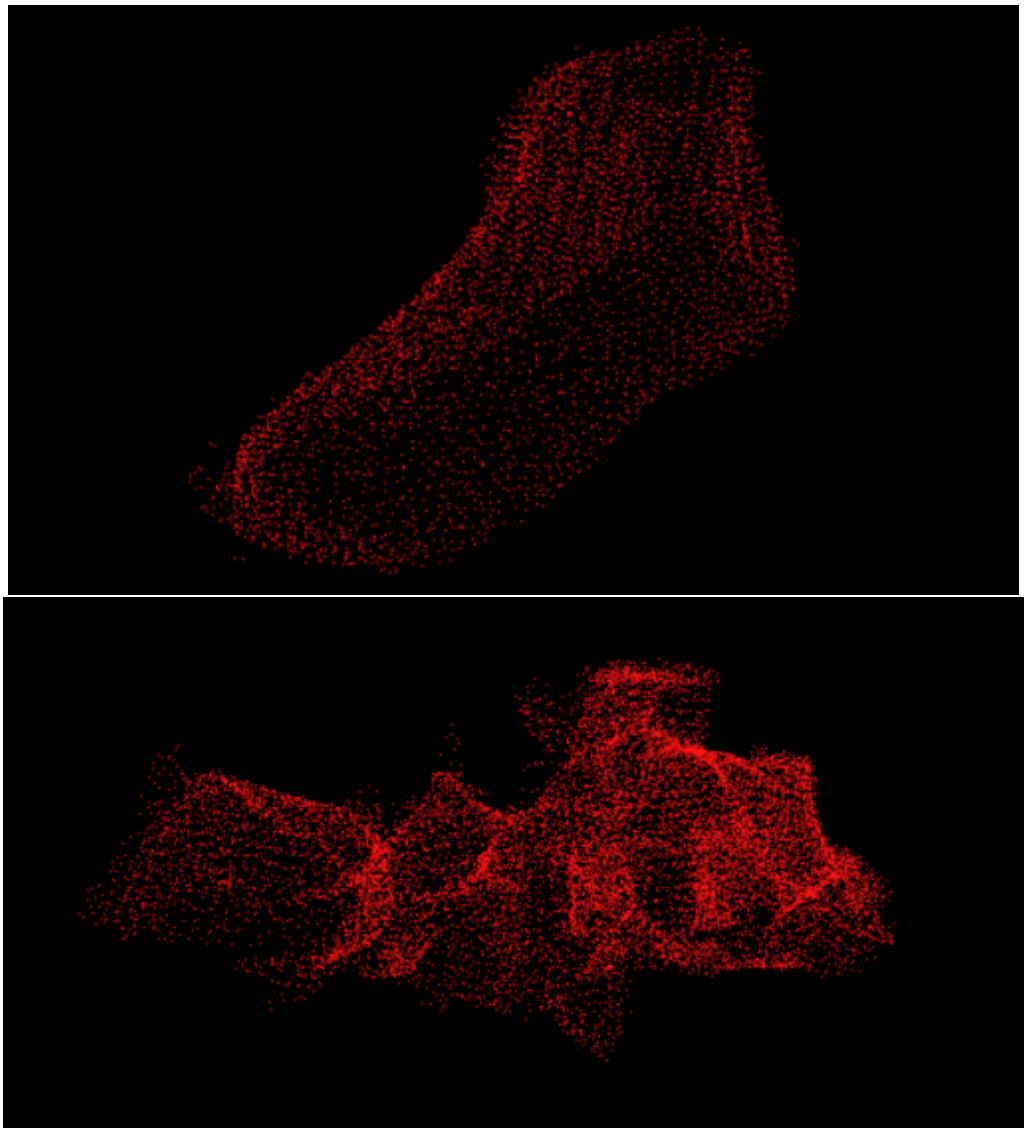

Figure 37: Final registered objects

## 4.2 Geometry evaluation Results

After having obtained the final registered objects, we gave them as inputs to the *Geometry* program, that, building an Oriented Bounding Box around them, has been able to output their width, height and depth.

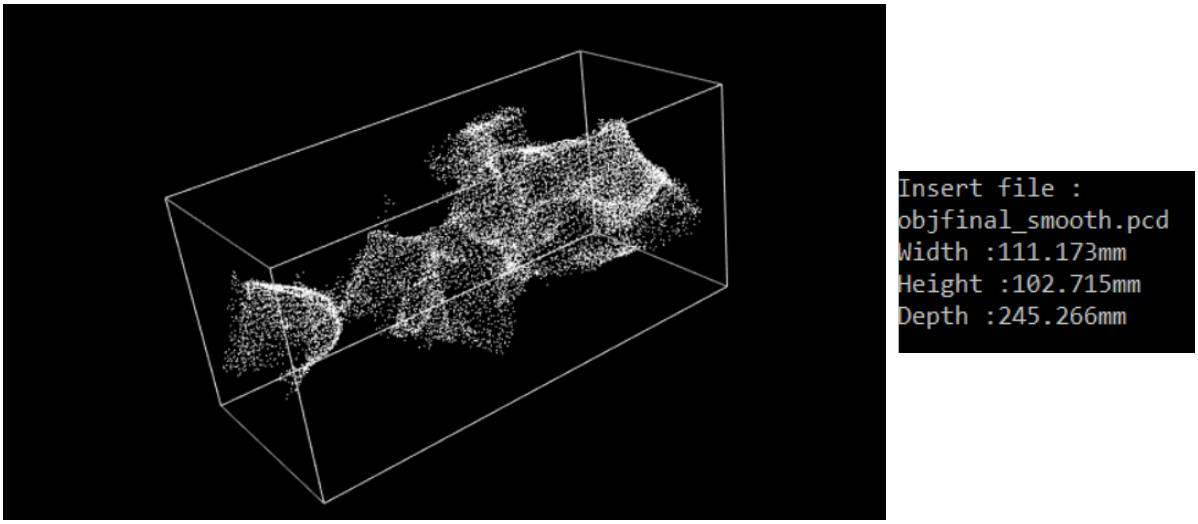


Figure 38: OBB for the Caterpillar excavator and related measures

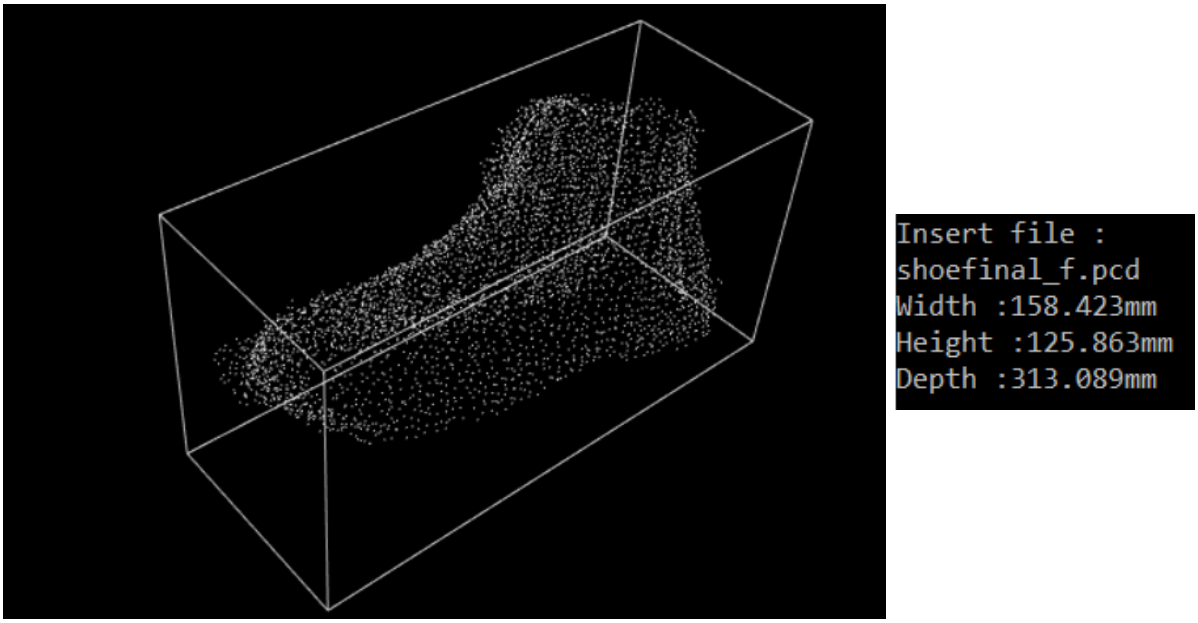


Figure 39: OBB for the shoe and related measures

## 4.3 AutoCAD model creation Results

The first result for the AutoCAD model creation is obtained after creating a mesh with the PCL Library. The file format is .VTK and can be correctly be visualized through the program *pcl_viewer*. As it can be seen in the figures below the holes have been successfully closed almost everywhere in the cloud. Then, after using the VCG Library, the .VTK file can be easily converted first in a .DXF file that can be read in AutoCAD.

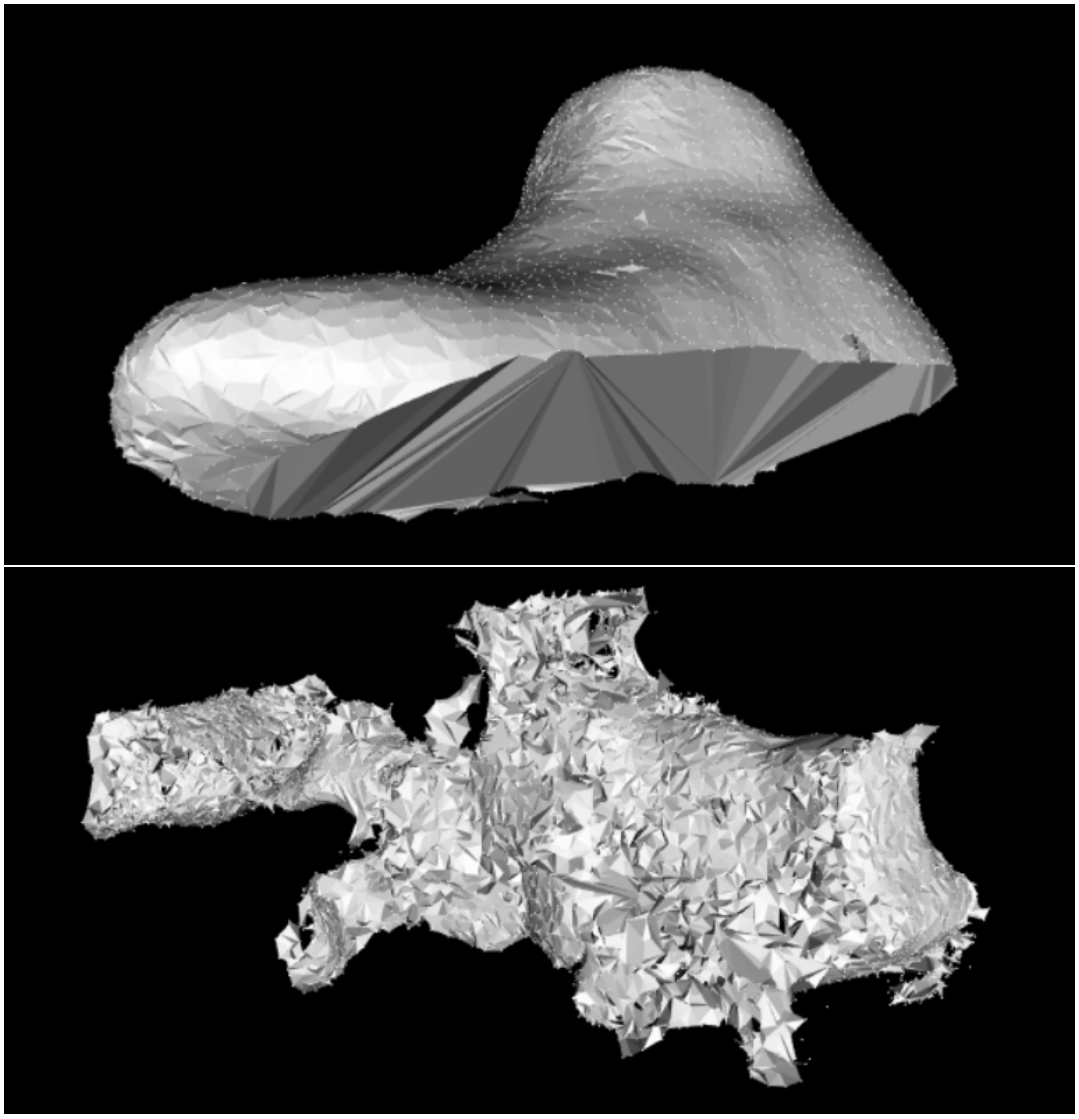

Figure 40: VTK meshes

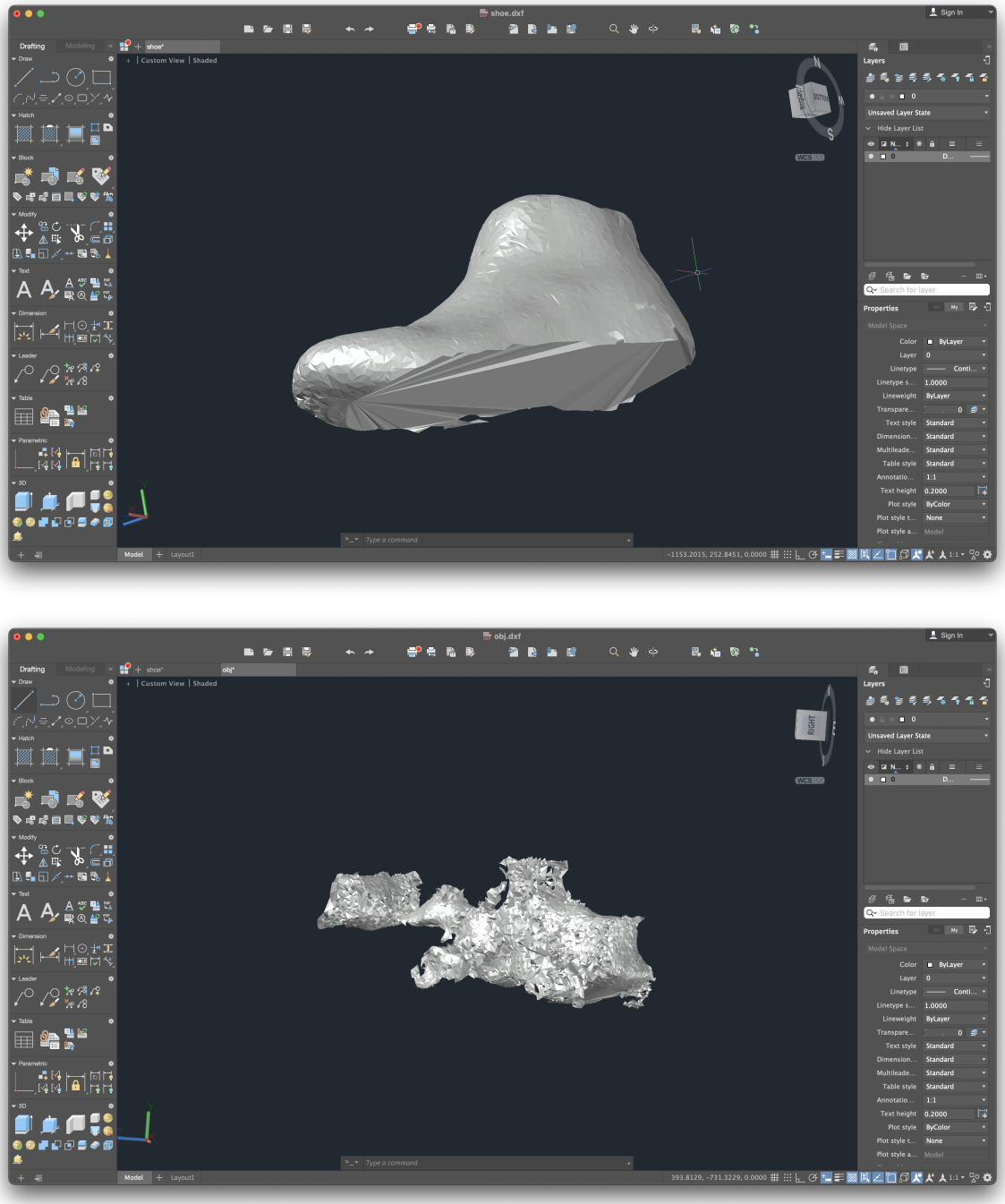


Figure 41: Autocad models

## 4.4 Segmentation Results

The scene on which segmentation has been performed is shown in Figure 42 and it is composed by two simple objects:

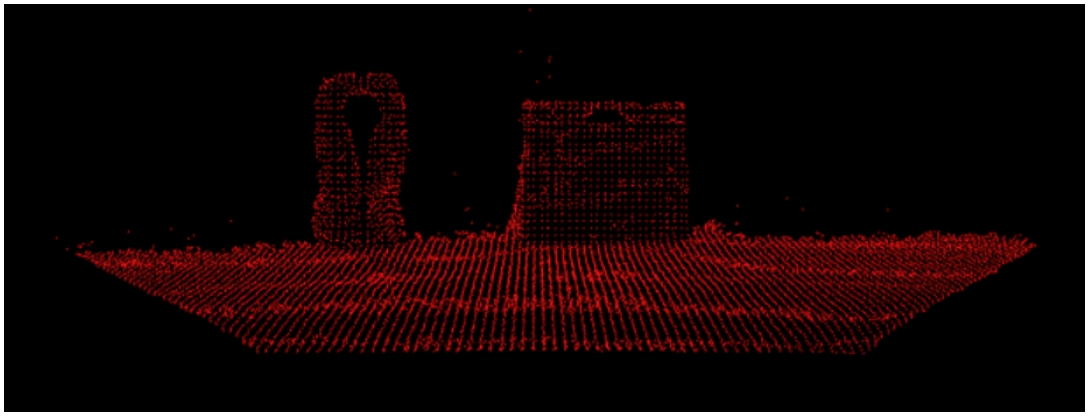

Figure 42: Scene point cloud

The program was able to eliminate the table and to find correctly two objects, which have been saved as two different point clouds in the same folder of the program. The two objects can be visualized on the same visualizer, with different colors.

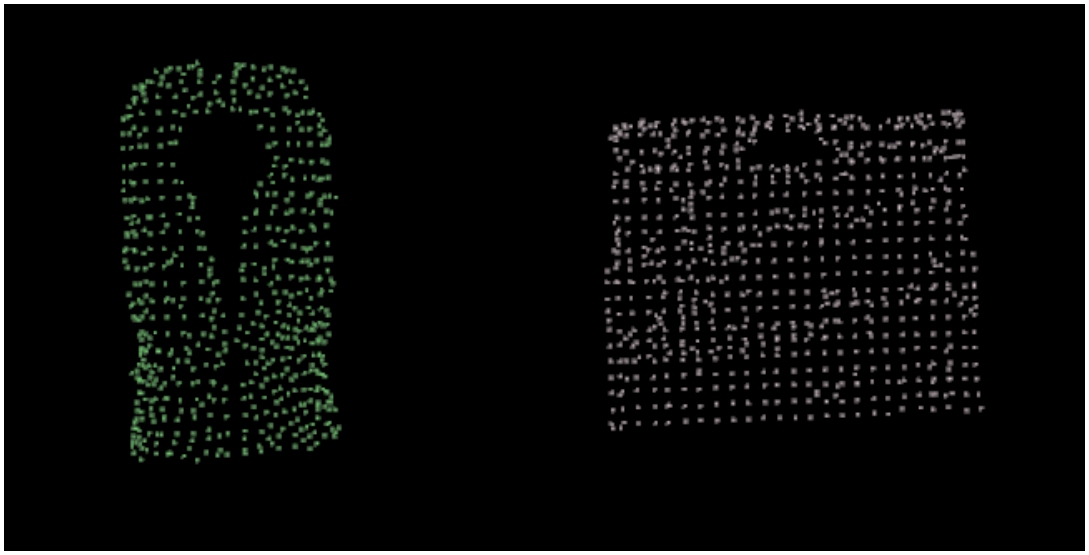

Figure 43: Two different object segmented, shown in different colors

After trying several times, it is important to point out that the Euclidean segmentation algorithm has an important limit: when the two objects are too close together (usually when they are touching), their points are closer than the threshold given as input and the algorithm will consequently mistakenly recognize them as one big object, as shown in Figure 44.

Figure 44: Two different object wrongly segmented

## 4.5 Object Recognition Results

The first step to do, before even running the program, is to create scenes with multiple object present and even possible occlusions. In Figure 45 there are examples of created scenes. After creating the scenes it is possible to play around with the algorithm and test wether it can recognize different models (mug, shoe or bug spray) and locate them correctly in the scene.

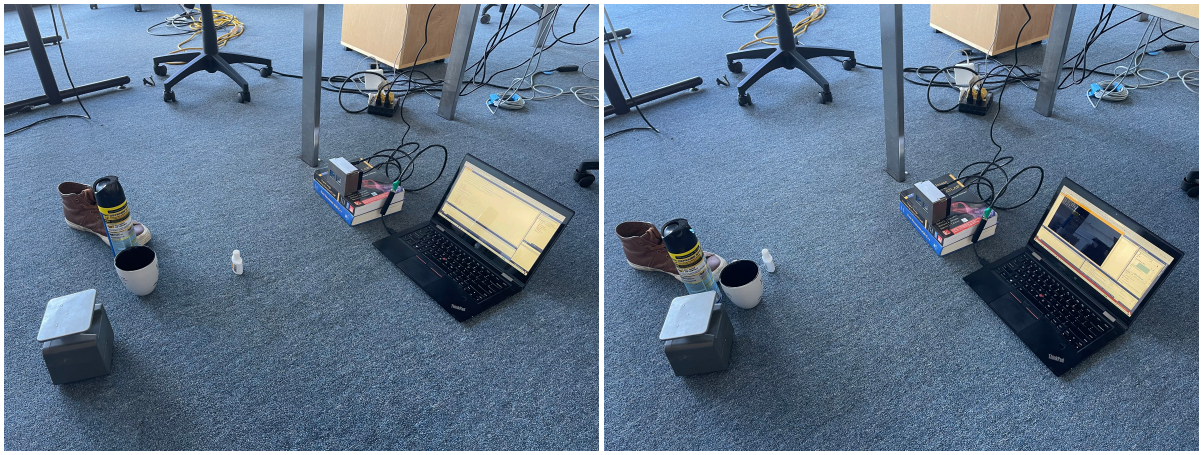

Figure 45: Examples of occluded scenes for object detection application

Starting with the mug, it is possible to see that the algorithm is able to successfully locate the object in the scene. The program output can also illustrate the correspondences with the model, shown on the side, drawing lines connecting them. In Figure 46 the scene chosen is a simple one, where the mug is quite visible in front. Instead, in Figure 47 and Figure 48, it has been tried to hide the mug a bit behind the bug spray, making the handle and some of the cup itself not visible in the cloud. This has been done to test the robustness of the algorithm and the results show that the program is still able to correctly locate the object.

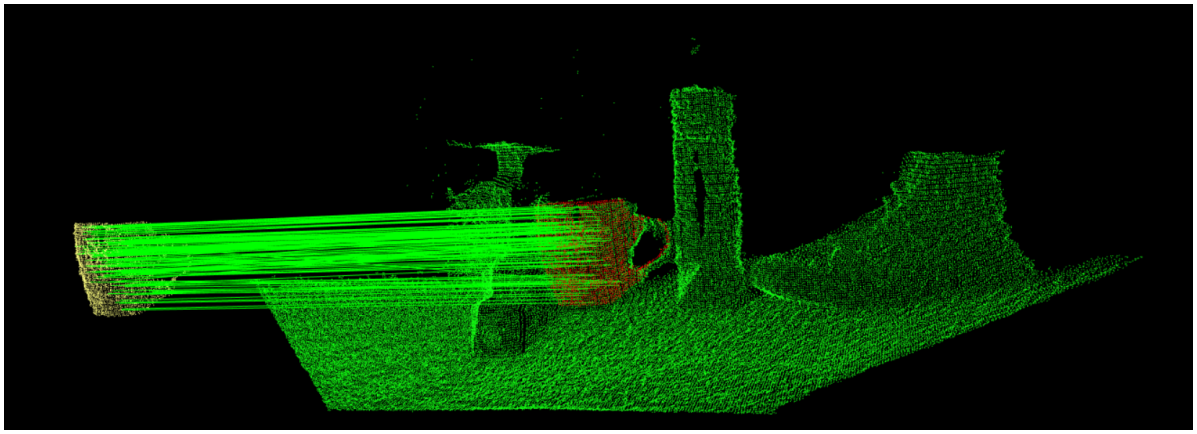

Figure 46: Mug correctly recognized in a simple scene

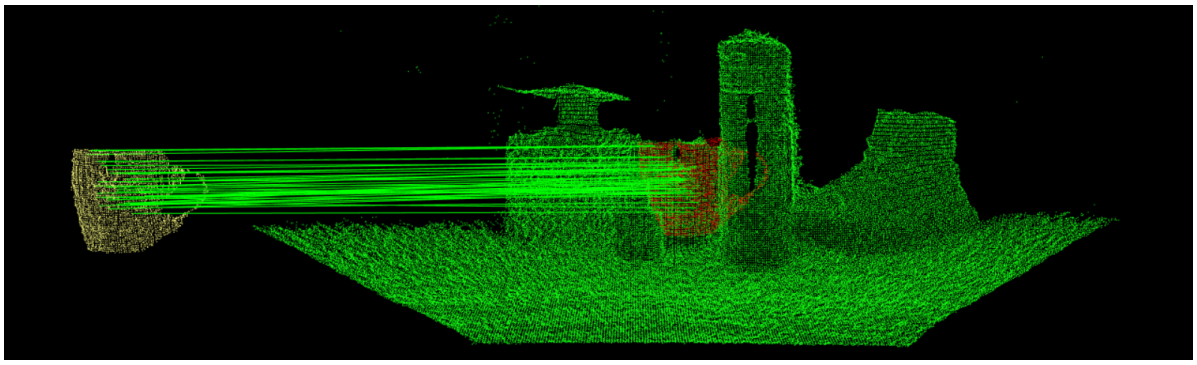

Figure 47: Mug correctly recognized in an occluded scene

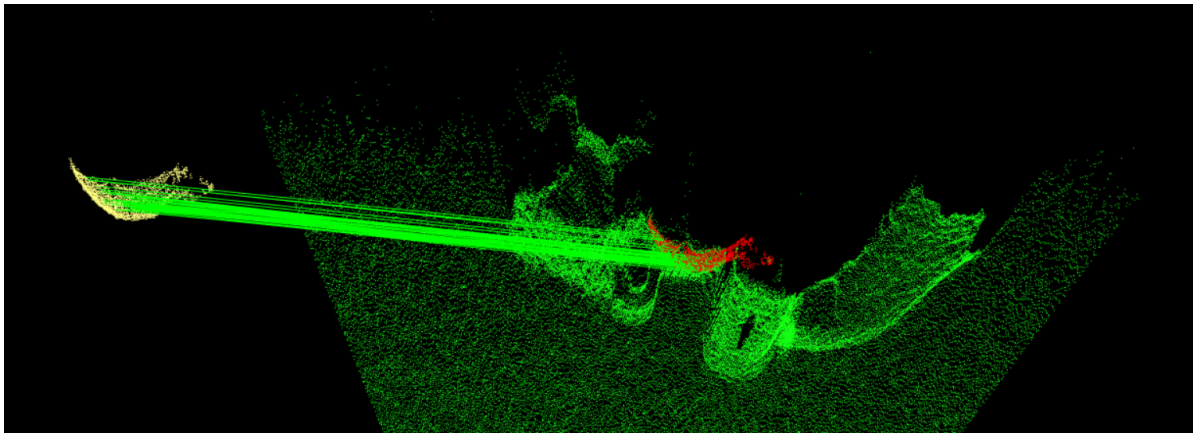

Figure 48: Another point of view of the mug correctly recognized in an occluded scene

The other two objects that have been fed to the algorithm are the bug spray and the shoe. As it can be seen, the program is able to recognize them successfully too in the scene. Moreover, in Figure 51 and Figure 52, the algorithm has been tested again in a difficult occluded situation and it managed to locate correctly the shoe even though almost half of it was covered by other objects. This proves again the robustness of the algorithm and its positive results.

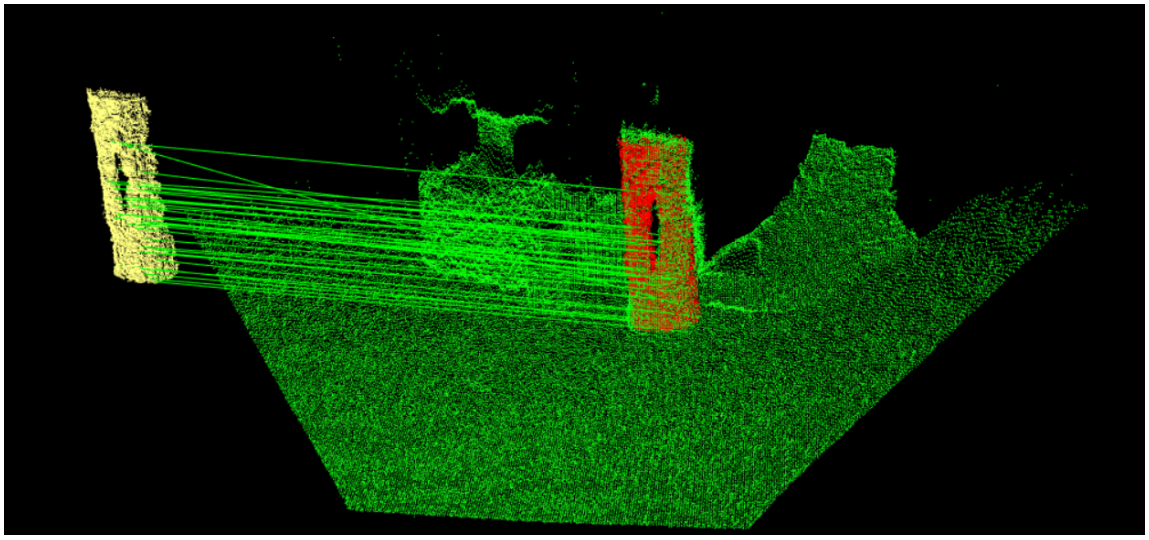

Figure 49: Bug spray correctly recognized

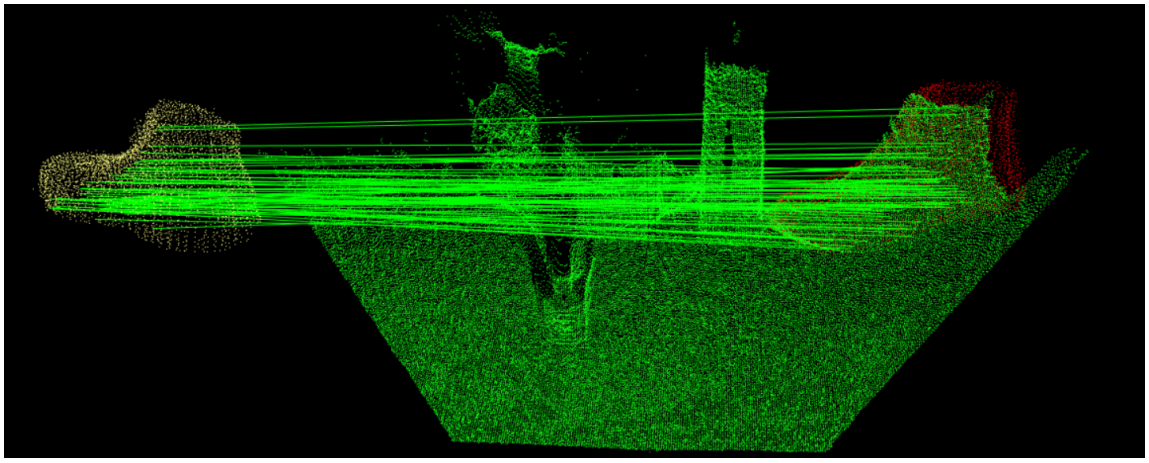

Figure 50: Shoe correctly recognized

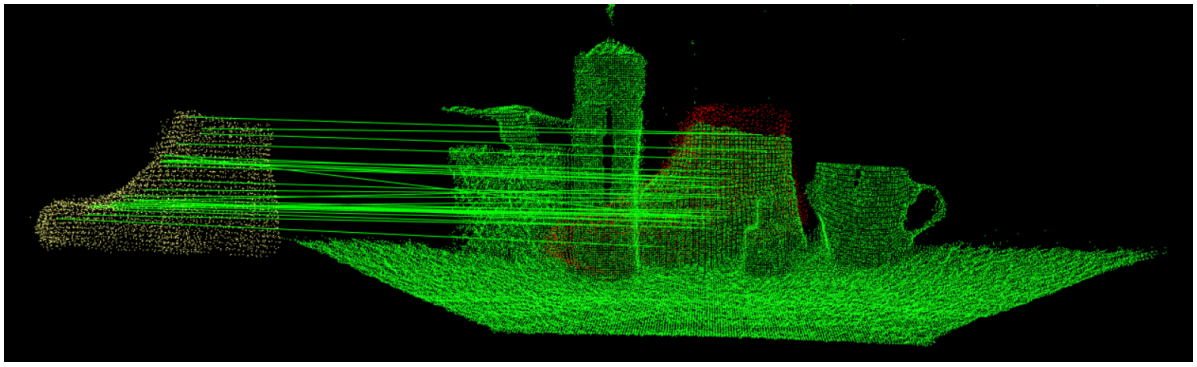

Figure 51: Shoe correctly recognized in an occluded scene

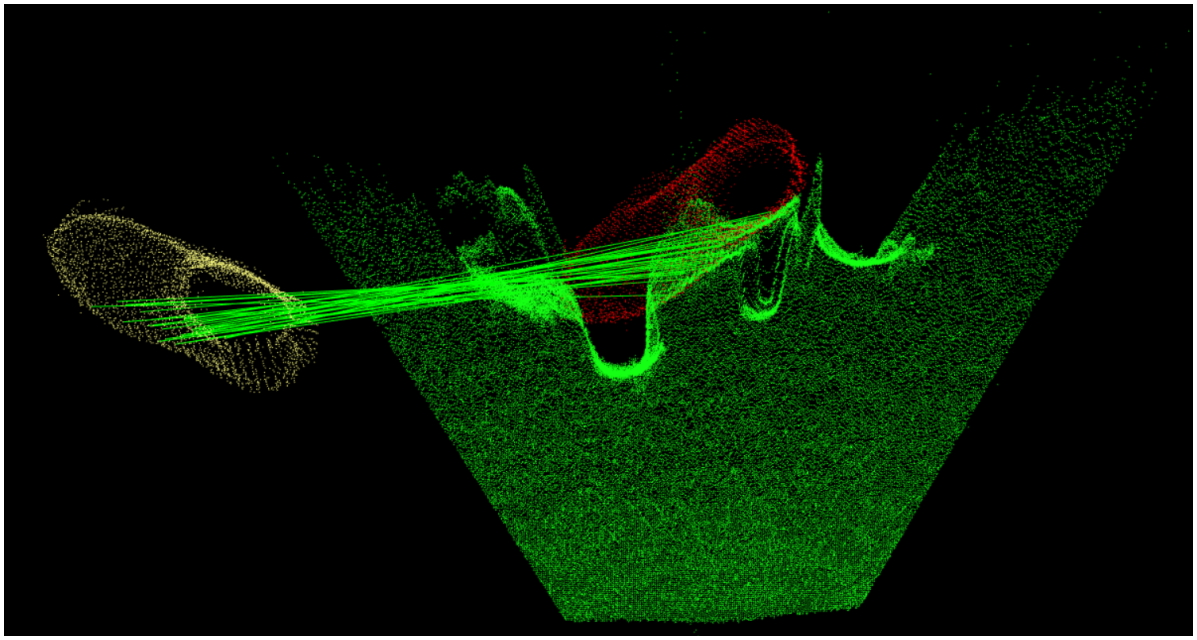

Figure 52: Another point of view of the shoe correctly recognized in an occluded scene

# CHAPTER 5

# CONCLUSION

The goal of this thesis was to learn how to work with one of the latest 3D ToF cameras on the market, to understand its hardware and software, and to develop C++ programs that lay the basis for future work in the robotics applications world.

The results highlight that this has been possible and the goals have been successfully reached. In fact, the C++ codes work smoothly and some of the developed programs can be implemented in robotics application. For instance, every program developed in this thesis can be useful in a bin-picking application.

Further improvements to this work consist in using all the software developed in the thesis in an actual robotic application to test its quality. Moreover, another applicable improvement is to use other libraries together with the PCL library so as to improve the results and increase the potential of what can be done with a 3D ToF camera.

# CITED LITERATURE


1. Grand View Research: Global 3D Machine Vision Market Size Report, 2020-2027. Grand View Research, 2020.

2. Keyence Corporation: Financial statements. https://www.dnb.com/business-directory/company-profiles.keyence.corporation.811c5c36acad190d0a4dea8d28cc5eab.html, [Online; accessed on 2021-2-6].

3. Owler: Annual revenue. https://www.owler.com/company/cognex, [Online; accessed on 2021-2-6].

4. MarketScreener: ISRA Vision. https://www.marketscreener.com/quote/stock/ISRA-VISION-AG-435932/news/Isra-Vision-concludes-financial-year-2018-2019-with-high-profit-margins-despite-stagnating-growth-29731313/, [Online; accessed on 2021-2-6].

5. LMI: LMI Concludes Successful 2019, Positions for Further Growth with "LMI 2025" Strategy. https://www.lmi.org/press-release/ar2019, [Online; accessed on 2021-2-6].

6. Macrotrends: National Instruments Revenue 2006-2020, NATI. https://www.macrotrends.net/stocks/charts/NATI/national-instruments/revenue, [Online; accessed on 2021-2-6].

7. Macrotrends: Omron Revenue 2006-2020 , OMRNY. https://www.macrotrends.net/stocks/charts/OMRNY/omron/revenue, [Online; accessed on 2021-4-28].

8. Macrotrends: Teledyne Technologies Revenue 2006-2020 , TDY. https://www.macrotrends.net/stocks/charts/TDY/teledyne-technologies/revenue, [Online; accessed on 2021-4-28].

9. Li, L.: Time-of-flight camera - an introduction. Technical Report SLOA190B, Texas Instruments, Dallas, Texas, January 2014.

## CITED LITERATURE (continued)

10. Geng, J.: Structured-light 3d surface imaging: a tutorial. Adv. Opt. Photon., Jun 2011. Reprinted as Vol. A of *Computers & Typesetting*, 1986.

11. Servotech Inc.: Vision systems in robotic motion control. Technical Report 115, Servotech Inc., Chicago, Illinois, 2020.

12. Stemmer Imaging: 3D time of flight cameras. https://www.stemmer-imaging.com/en/knowledge-base/cameras-3d-time-of-flight-cameras/, [Online; accessed on 2021-5-10].

13. Giancola, S., Valenti, M., and Sala, R.: A Survey on 3D Cameras: Metrological Comparison of Time-of-Flight, Structured-Light and Active Stereoscopy Technologies. Springer, June 2018.

14. Sodavision: Photoneo Bin Picking Solutions. https://www.sodavision.com/product/phot-oneo-bin-picking-solutions/, [Online; accessed on 2021-5-10].

15. Time of flight image sensor for industry. https://www.sony-semicon.co.jp/e/products/IS/industry/technology/ToF_technology.html, [Online; accessed on 2021-1-17].

16. Basler: Palletizing. https://www.baslerweb.com/en/products/cameras/3d-cameras/3d-applications/palletizing/, [Online; accessed on 2021-2-6].

17. Melexis: Enabling the full potential of automotive 3D ToF imaging. https://www.melexis.com/tech-talks/enabling-potential-automotive-3d-tof-imaging, [Online; accessed on 2021-6-29].

18. AMS: Automotive. https://ams.com/automotive, [Online; accessed on 2021-6-29].

19. May, S., Droeschel, D., Fuchs, S., Holz, D., and Nüchter, A.: Robust 3d-mapping with time-of-flight cameras. In 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 1673–1678, 2009.

20. Basler: Basler blaze - time-of-flight (tof) camera. https://www.baslerweb.com/en/products/cameras/3d-cameras/basler-blaze/, [Online; accessed on 2021-5-11].

## CITED LITERATURE (continued)


21. Sony: Back-illuminated time of flight (tof) 3d image sensors. https://www.sony-semicon.co.jp/e/products/IS/industry/product/ToF_products.html, [Online; accessed on 2021-5-11].

22. Lucid Vision Labs: Helios2 time of flight (tof) ip67 3d camera. https://thinklucid.com/product/helios2-time-of-flight-imx556/, [Online; accessed on 2021-5-11].

23. Point Cloud Library: Point cloud library. https://pointclouds.org/, [Online; accessed on 2021-5-13].

24. Point Cloud Library: The pcl registration api. https://pcl.readthedocs.io/projects/tuto-rials/en/latest/regitration_api.html#registration-api, [Online; accessed on 2021-5-14].

25. Rusu, R. B., Blodow, N., and Beetz, M.: Fast point feature histograms (fpfh) for 3d registration. In 2009 IEEE International Conference on Robotics and Automation, pages 3212–3217, 2009.

26. Rusu, R.: Semantic 3d object maps for everyday manipulation in human living environments. KI - Künstliche Intelligenz, 24, 11 2010.

27. Guo, N., Zhang, B., Zhou, J., Zhan, K., and Lai, S.: Pose estimation and adaptable grasp configuration with point cloud registration and geometry understanding for fruit grasp planning. Computers and Electronics in Agriculture, 179:105818, 12 2020.

28. Yang, J., Xian, K., Xiao, Y., and Cao, Z.: Performance evaluation of 3d correspondence grouping algorithms. In 2017 International Conference on 3D Vision (3DV), pages 467–476, 2017.

29. Tombari, F. and Di Stefano, L.: Object recognition in 3d scenes with occlusions and clutter by hough voting. In 2010 Fourth Pacific-Rim Symposium on Image and Video Technology, pages 349–355, 2010.

30. Holz, D., Ichim, A. E., Tombari, F., Rusu, R. B., and Behnke, S.: Registration with the point cloud library: A modular framework for aligning in 3-d. IEEE Robotics Automation Magazine, 22(4):110–124, 2015.

31. Epic Systems: Quick history of machine vision. https://www.epicsysinc.com/blog/machine-vision-history, [Online; accessed on 2021-1-14].

## CITED LITERATURE (continued)


32. Hussmann, S., Ringbeck, T., and Hagebeuker, B.: A performance review of 3d tof vision systems in comparison to stereo vision systems. Laser Focus with Fiberoptic Technology, Nov 2008.

33. SongZhang: Optics and lasers in engineering. Elsevier, 2018. Purdue University.

34. Flynt, J.: Structured light 3d scanning: What is it and how does it work? https://3dinsider.com/structured-light-3d-scanning/, [Online; accessed on 2021-1-15].

35. Laukkanen, M.: Performance evaluation of time-of-flight depth cameras. Technical Report AS-84, Aalto university, school of electrical Engineering, November 2015.

36. Sarbolandi, H., Plack, M., and Kolb, A.: Pulse based time-of-flight range sensing. Sensors, 18(1679), April 2018.

37. The future of automated random bin picking. https://www.robots.com/blogs/the-future-of-automated-random-bin-picking, [Online; accessed on 2021-1-19].

38. Fuchs, S., Haddadin, S., Keller, M., Parusel, S., Kolb, A., and Suppa, M.: Cooperative bin-picking with time-of-flight camera and impedance controlled dlr lightweight robot iii. Technical report, 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems, October 2010.

39. Buchholz, D.: Bin-picking – new approaches for a classical problem. Technical report, Technischen Universitat Carolo-Wilhelmina, January 2015.

40. McFarland, M.: Most self-driving companies say this tech is crucial. elon musk disagrees. https://www.cnn.com/2019/06/17/tech/lidar-self-driving-tesla/index.html, [Online; accessed on 2021-1-17].

41. Tof 3d image sensors for automotive. https://www.infineon.com/cms/en/product/sensor/tof-3d-image-sensors/tof-3d-image-sensors-for-automotive/, [Online; accessed on 2021-1-18].

42. The Geospatial: What is 3d mapping. https://medium.com/@thegeospatialnews/what-is-3d-mapping-5fb86944a96, [Online; accessed on 2021-1-18].

43. Cui, Y., Schuon, S., Chan, D., Thrun, S., and Theobalt, C.: 3d shape scanning with a time-of-flight camera. Technical report, Stanford University, Jun 2010.

## CITED LITERATURE (continued)


44. May, S., Droeschel, D., Holz, D., Fuchs, S., Malis, E., Nüchter, A., and Hertzberg, J.: Three-dimensional mapping with time-of-flight cameras. Journal of Field Robotics, 26(11-12):934–965, 2009.

45. Wikipedia: Point cloud library. https://en.wikipedia.org/wiki/Point_Cloud_Library, [Online; accessed on 2021-5-13].

46. Wikipedia: Random sample consensus. https://en.wikipedia.org/wiki/Random_sample_co-nsensus, [Online; accessed on 2021-5-18].

47. Point Cloud Library (PCL): pcl::uniformsampling< pointt > class template reference. https://pointclouds.org/documentation/classpcl_1_1_uniform_sampling.html, [Online; accessed on 2021-5-19].

48. Grupo de Robòtica, University of Leòn: Pcl/openni tutorial 2: Cloud processing (basic): Feature estimation. https://robotica.unileon.es/index.php?title=PCL/OpenNI_tut-orial_2:_Cloud_processing_(basic)#Feature_estimation, [Online; accessed on 2021-5-31].

49. Grupo de Robòtica, University of Leòn: Pcl/openni tutorial 4: 3d object recognition (descriptors). https://robotica.unileon.es/index.php?title=PCL/OpenNI_tutorial_4:_3D_object_recognition_(descriptors)#FPFH, [Online; accessed on 2021-5-31].

50. Grupo de Robòtica, University of Leòn: Pcl/openni tutorial 3: Cloud processing (advanced). https://robotica.unileon.es/index.php?title=PCL/OpenNI_tutorial_3:_Cl-oud_processing_(advanced)#Registration, [Online; accessed on 2021-5-31].

51. Wikipedia: Iterative closest point. https://en.wikipedia.org/wiki/Iterative_closest_point, [Online; accessed on 2021-6-02].

52. Point Cloud Library (PCL) Users mailing list: Finding oriented bounding box of a cloud. http://www.pcl-users.org/Finding-oriented-bounding-box-of-a-cloud-td4024616.html, [Online; accessed on 2021-6-5].

53. Liu, J., Bai, D., and Chen, L.: 3-d point cloud registration algorithm based on greedy projection triangulation. Applied Sciences, 8:1776, 09 2018.

54. Wikipedia: Greedy triangulation. https://en.wikipedia.org/wiki/Greedy_triangulation, [Online; accessed on 2021-6-3].

## CITED LITERATURE (continued)


55. VCG Library: The vcg library. http://www.vcglib.net, [Online; accessed on 2021-6-3].

56. Grupo de Robòtica, University of Leòn: Pcl/openni tutorial 3: Cloud processing (advanced). https://robotica.unileon.es/index.php?title=PCL/OpenNI_tutorial_3:_Cl--oud_processing_(advanced)#Segmentation, [Online; accessed on 2021-6-3].

57. Tombari, F., Salti, S., and Di Stefano, L.: Unique signatures of histograms for local surface description. volume 6313, pages 356–369, 09 2010.

58. Huang, T. S.: Computer vision: Evolution and promise. Technical report, University of Illinois at Urbana-Champaign, Urbana, Illinois, 1996.

59. Luan, X.: Experimental investigation of photonic mixer device and development of tof 3d ranging systems based on pmd technology. Technical report, University of Siegen, Siegen, November 2001.

60. Garcia, F.: Machine vision systems: Industrial applications rise, but trade is hard to track. U.S. International Trade Commission, November 2019.

61. Wyant, J. C.: Interferometric optical metrology-basic principles and new systems. Laser Focus with Fiberoptic Technology, 18(5):65–71, 1982.

62. Riordan, A. O., Newe, T., Toal, D. J. F., and Dooly, G.: Stereo vision sensing: Review of existing systems. International Conference on Sensing Technology, Dec 2018. University of Limerick.

63. Cognex: IN-SIGHT LASER PROFILER. https://www.cognex.com/products/machine-vision/3d-machine-vision-systems/in-sight-laser-profiler, [Online; accessed on 2021-5-10].

64. Point Cloud Library (PCL): 3d object recognition based on correspondence grouping. https://pcl.readthedocs.io/en/latest/correspondence_grouping.html#id5, [Online; accessed on 2021-6-8].

# VITA

| | |
|---|---|
| NAME | Matteo Mercuri |
| EDUCATION | |
| | Master of Science in "Mechanical Engineering, University of Illinois at Chicago, May 2021, USA |
| | Specialization Degree in "Propulsion of Land Vehicles ", Jul 2021, Polytechnic of Turin, Italy |
| | Bachelor's Degree in Mechanical Engineering, Jul 2019, Polytechnic of Turin, Italy |
| LANGUAGE SKILLS | |
| Italian | Native speaker |
| English | Full working proficiency |
| | 2018 - IELTS examination (7.5/9) |
| Spanish | Intermediate level |
| SCHOLARSHIPS | |
| Fall 2020 | Italian scholarship for final project (thesis) at UIC |
| Fall 2020 | Italian scholarship for TOP-UIC students |
| Fall 2017 | Merit scholarship based on university GPA given by Polytechnic of Turin, Italy |
| Fall 2017 | Merit scholarship based on university GPA given by Camplus College |
| Fall 2016 | Merit scholarship based on high school GPA given by Camplus College |

| | |
|---|---|
| WORK EXPERIENCE | |
| 07/16 - 09/20 | Summer Assistant Manager, MEM Finance |
| 06/15 - 07/15 | Summer Assistant Manager, Globe Cafè Varese |