PONTIFICIA UNIVERSIDAD JAVERIANA

BACHELOR'S IN ENGINEERING FINAL PROJECT

Usage of 3D scanning methods in plant characterization

Author:
David F. CALLES

Director:
Juan C. GIRALDO
Co-Director:
Francisco C. CALDERÓN

A thesis submitted in fulfillment of the requirements for the degree of Bachelor's in Engineering Degree

in the

Electronics Department

December 4, 2020

Declaration of Authorship

I, David F. CALLES, declare that this thesis titled, "Usage of 3D scanning methods in plant characterization" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a Bachelor's degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date: December 4, 2020

"Always dream and shoot higher than you know you can do. Don't bother just to be better than your contemporaries or predecessors. Try to be better than yourself. "

William Faulkner

Writer and Nobel prize winner.

"We all know that hard work and good planning pay off. What not everyone believes is that not all details can be planned, sometimes it is necessary to move forward in the midst of doubt, trusting that with constant effort, the gaps will begin to close until the whole plan makes sense. "

David F. Calles

Nobody, yet.

PONTIFICIA UNIVERSIDAD JAVERIANA

Abstract

Faculty of Engineering
Electronics Department

Bachelor's in Engineering Degree

Usage of 3D scanning methods in plant characterization

by David F. CALLES

Characterizing vegetation accurately opens the door to flora species classification, development and improvement of crop growing methods, technologies, and many applications more. Nevertheless, studies involving the use of 2D imaging and manual measurements seem to lack key information in order to improve this process. In this project, three 3D reconstruction methods are evaluated for regular objects as well as for whole-plants. For this purpose, a commercial low-cost time-of-flight sensor, a high resolution plenoptic camera, and a single-laser single-camera triangulation scanner are compared. The laser triangulation scanner is completely built from scratch with open-source tools and explained in detail throughout the document. Point clouds are individually obtained from each method and a feature extraction involving dimensions and volume is performed. Obtained results direct future research work to delve mainly into laser triangulation methods and light fields using different perspectives of the object. Finally, a quantitative analysis of the results is shared, and a qualitative comparison considering results, times, costs and main strengths of each method is documented.

Acknowledgements

I couldn't have performed this final project without the help, guidance and support of many people. First, I would like to thank the electronics department of the *Javeriana University*, as they made my studies possible through a significant financial support throughout the entire program.

I would like to thank my project director Juan Carlos, co-director Francisco, and methodology professor Manuel for their guidance through the writing and design process of the whole project.

I also want to give special thanks to my mother Lorena, my father Francisco, my brother Jonathan, and my dear grandma Mercedes, whose unconditional support and advises throughout my whole academic formation made it possible for me to get to this point of my life. I wouldn't be here without you.

I would also like to acknowledge the help of the electronics lab employees Carlos, Norberto, Ivonne and professor Eduardo that helped me at different stages of my project. In addition, thanks to all the professors with whom I was fortunate to have class with, as they, semester by semester and with their own style, prepared me to achieve this project.

Last but not least, thanks to my friends and colleagues, special thanks to Cindy, Sebastián and Santiago that helped me in an early stage of the project, and all other friends that with their conversations and distractions helped me rest my mind outside my project.

Contents

Chapter 1

Project's introduction and conception

1.1 Introduction/Problem

With an unprecedented rate of population growth and the increasing demand for food, improving plant breeding efficiency and the robustness of crop production is key to meeting the food and energy demand of more than nine billion people by the year 2050 [1]. High-throughput plant phenotyping is a technology that allows us to understan how the genetics of a plant can determine its phenotypes in high-yield samples.

As the need for more effective harvesting methods becomes apparent, the ability to measure performance-related traits increases to become a fundamental asset in agriculture industries. These justify a significant amount of resources for its measurement, expressed morphologically as the amount of visible biomass, seed yield, among others.

Current manual phenotyping techniques involve subjective drawbacks, which are time-consuming and costly to the individual [2]. At the same time, 2D imaging analysis remains short to fully represent complex plant morphology and structure, especially after tillering in cereals, where occlusion becomes problematic [3]. That is why, nowadays, extensive research and development projects are being executed worldwide to correlate genetics with the plant's response to different environments, phenotyping.

The extraction of structural/morphological features from whole plant 3D modeling promises to increase high-throughput studies' potential. As a result, improvements in the precision and completeness of the measurements are obtained [2][4]. Multiple-view shooting (i.e., photogrammetry) [5], laser cameras [6], 3D digitization [2], and laser line [7] are some of the most common 3D scanning technologies. These methods have proven to be highly accurate and feasible. However, most of them are economically unattainable for most interested users. This project aims at reconstructing 3D entire-plant models without the need to move the specimens to a fully controlled environment (laboratories) by evaluating laser-based, multi-view, and time-to-flight methods to acquire point clouds of plants. To extract its useful traits such as height, width, and visible biomass, contrasting its results in terms of accuracy, time-consumption, and cost-efficient aids in the improvement of future harvesting techniques.

1.2 Requirements

1. The developed laser-based scanner implementation should have a minimum resolution of in units of millimeters.
2. The Kinect v2 montage must be set to get a resolution comparable with the two other methods. (in units of mms).
3. The plenoptic camera must be set to get a minimum point resolution and accuracy in units of millimeters.
4. Traits measurement must have a mean error below 20% compared to manual measurements.

1.3 Objectives

1.3.1 General objective

To evaluate the accuracy, resolution, and time efficiency of three different 3D scanning methods for whole single-plant modeling.

1.3.2 Specific objectives

1. To acquire a point cloud of a small-medium size plant through time-of-flight devices (Kinect v2) 3D scanning methods.
2. To acquire a point cloud of a small-medium size plant through plenoptic camera imaging.
3. To acquire a point cloud of a small-medium size plant through a laser-based stereo vision scanning method.
4. To extract plant's height, width, and visible biomass from point clouds based on pre-existing commercial/open-source 3D reconstruction software.
5. To quantify the error and time-efficiency from the morphological trait's measurements using 3D scanning methods compared to manual method's indexes.

1.4 Deliverables

1. A point cloud of a small-medium size plant obtained through a time-to-flight device with its acquisition documentation.
2. A point cloud of a small-medium size plant obtained through images from a plenoptic camera with its acquisition documentation.
3. A point cloud of a small-medium size plant obtained through a laser-based scanning method.
4. A documented software prototype based on pre-existing software to reconstruct plants and acquire morphological traits' measurements.
5. The Documentation and numerical analysis out of the results obtained from the three different cloud points extraction methods and used software.

1.5 Solution proposal

1.5.1 Blocks diagram

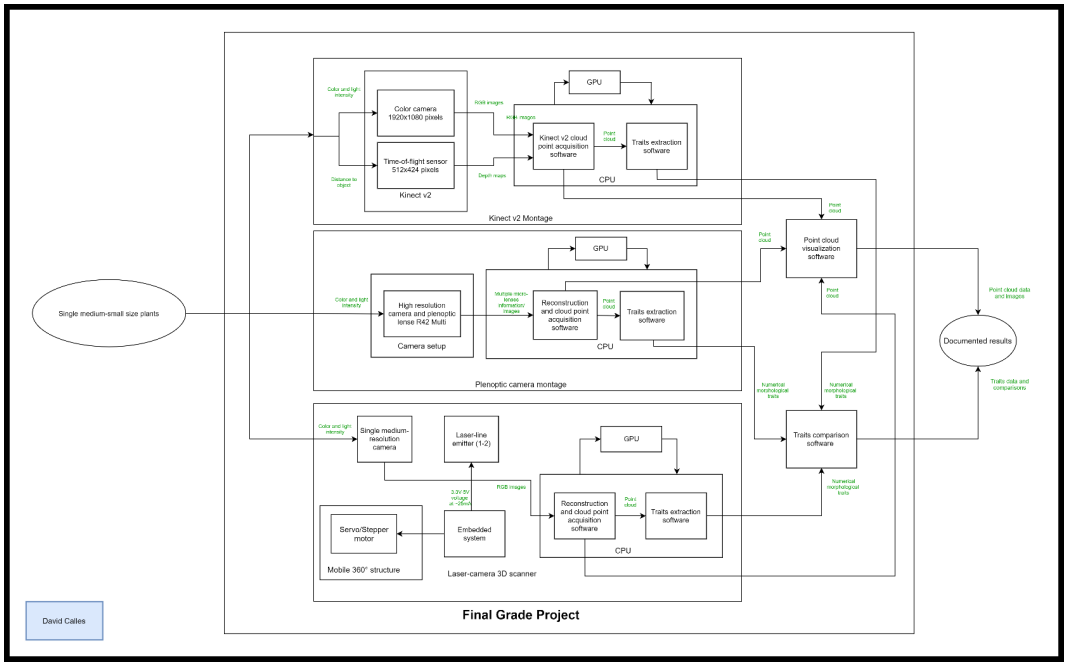

FIGURE 1.1: Final degree project's blocks diagram

The whole project consists of the montage of three different scanning methods for comparison purposes.

First, the block diagram describes the "time-of-flight" technology montage in the upper block, which uses the Kinect v2 sensor. This technology is considered the most straightforward method from the proposed ones. For this alternative, only the following are used: a computer with Microsoft's default scanning software (preferably with GPU), and the object to be scanned. All processing takes place on a laptop with an NVIDIA GTX1050 GPU to implement a traits' extraction software from pre-existing open-source software tools.

In the middle block, it is observable the plenoptic camera montage. In hardware, all the complexity can be reunited inside the camera and linked to the computer. In this case, thankfully, the vendors' software helps in the majority of the cloud point extraction process. Carefully taken pictures are needed, but the cloud point extraction software is already functional and even optimized. Again, just the traits extraction software is designed and implemented from scratch.

The most complex *macro-block* is the one at the bottom because the objects are intended not to move at all while scanning. For its part, the camera and the laser must be the ones that rotate. For this purpose, an embedded system is designed from scratch to control a motor's position and image acquisition moments, and laser energizing. A raspberry pi Linux-based system is proposed for this functionality with its current correspondent drivers to deliver the desired currents. On the other hand, data from the camera and motor's position are sent from the embedded system to the computer for general processing. The implementations of the cloud point and traits' extraction software is written in Python. Once every scanning process has

been completed, a comparison between methods and results takes place and be well documented for the final deliverables.

1.5.2 Budget

Category	Resource	Description	Hours	Whole price	Cost per hour	Total cost
Workforce	Student labor	3 credits in first semester 7 credits in second semester	480	-	15.000,00 COP	7.200.000,00 COP
	Teachers asistant	5 hours of assistance per week in first semester 2 hours of assistance per week in second semester	126	-	180.000,00 COP	22.680.000,00 COP
Equipment	Kinect v2	Time-of-Flight depth sensor	126	525.000,00 COP	5.250,00 COP	661.500,00 COP
	Plenoptic camera	Multi-lens raytrix r42 plenoptic camera	315	328.000.000,00 COP	32.800,00 COP	10.332.000,00 COP
	Tripods	Tripods to support Kinectv2 and plenoptic camera	441	200.000,00 COP	2.000,00 COP	882.000,00 COP
	Computer with GPU	General purpose computer and rough processing unit	480	2.500.000,00 COP	25.000,00 COP	12.000.000,00 COP
	Controlled DC power source	DC power source to power motor and laser	336	800.000,00 COP	8.000,00 COP	2.688.000,00 COP
	3D printer	Printing serice to print gears and other features of the mechanical model	10	-	30.000,00 COP	300.000,00 COP
	Logical states analizer	General equipment for signal visualization analysis	147	20.000.000,00 COP	20.000,00 COP	2.940.000,00 COP
	Osciloscope		147	6.000.000,00 COP	6.000,00 COP	882.000,00 COP
Components	Laser-line	5mW laser line for 3d laser based scanner	-	80.000,00 COP	-	80.000,00 COP
	Stepper motor	Stepper motor for camera and laser rotation	-	70.000,00 COP	-	70.000,00 COP
	Laser-line driver	Driver modules for stepper motor and laser line	-	20,00 COP	-	20,00 COP
	Stepper motor driver		-	50.000,00 COP	-	50.000,00 COP
	Raspberry pi camera (Element14)	Camera to be used in image acquisition for laser based scanner	-	250.000,00 COP	-	250.000,00 COP
	Raspberry pi board (3 B)	Board for embedded system mounting	-	260.000,00 COP	-	260.000,00 COP
Software	Microsoft 3D scanning software	Kinect v2 software for cloud point acquisition	126	- COP	- COP	- COP
	Raytrix scanning software	Plenoptic camera software for cloud point acquisition	315	- COP	- COP	- COP
	Microsoft Office package	General purpose organizers and text editors	480	19.000,00 COP	-	19.000,00 COP
	KiCat	PCB maker software	42	- COP	- COP	- COP
	Linux-based embedded software	Rasbian based embeded operative system for the raspberry	336	- COP	- COP	- COP
	3D modeling software	Software to model 3D parts to be printed	63	- COP	- COP	- COP
Various	Leds, resistances, etc	Various basic elements	-	50.000,00 COP	-	50.000,00 COP
	Objects to be scanned	Plants and general objects	-	100.000,00 COP	-	100.000,00 COP
					Total	61.444.520,00 COP

FIGURE 1.2: Financial quotation of final degree project

In figure 1.2, are listed the hypothetical costs of the project taking into account workforce reflected student's and advisor's wages (price per hour), software, equipment and material fees as well as PCB manufacturing/3D printing/laser cutting costs and other various costs that the project may incur on.

1.6 Tasks hierarchy and schedule

1.6.1 Schedule: Gantt diagram

Please refer to figures A.1 and A.2 in appendix A for the Gantt diagrams with the schedules of the first and second semesters of the project .

1.6.2 Work breakdown structure

A *work breakdown structure* summarizes the overall planning of the project, dividing it by simple *macro-blocks* that represent the tasks and the order in which they must be done. There is no single way of describing this hierarchy but a clear order must be specified either way. In figure 1.3, it is summarized the project's planning. The reader can find the most simple and specific tasks at the bottom of the diagram (which do not have previous requirements) and the most complex and general ones at the top. Here it must be specified that because there are three different singular montages in the project, each montage has its own set of tasks. Although the position from bottom to top of a single block can express somehow when the task must be done, there is no such relationship from one montage to another, which means

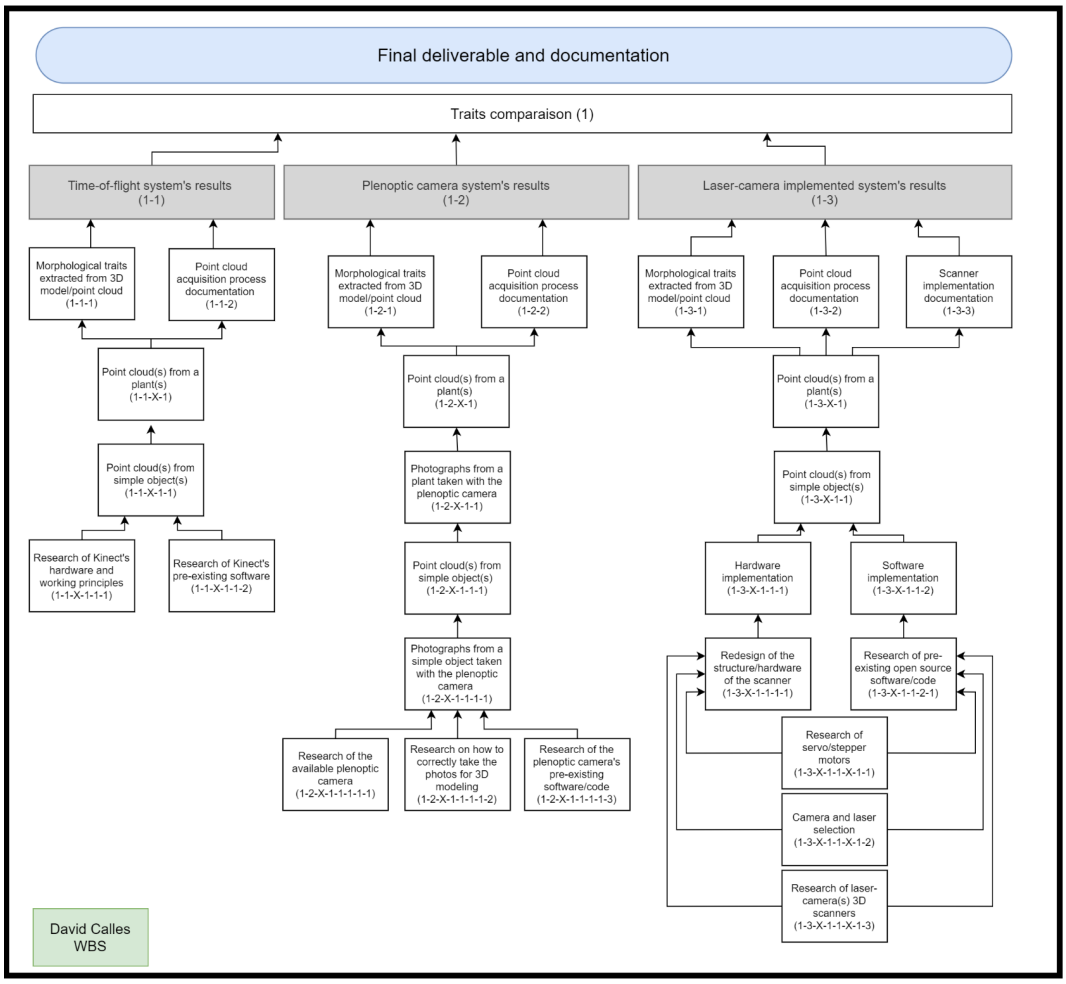

FIGURE 1.3: Project's work breakdown structure (a.k.a tasks' hierarchy)

the more straightforward tasks can all be initiated and developed in parallel. To understand this better it can be expressed in the figure that the final task is the *traits and general results comparison* which went through a step-by-step process from all the initial research for each scanning method.

Chapter 2

Theoretical framework and state-of-art

2.1 Main variables/traits in a phenotyping process

A plant phenotype is the set of structural, physiological, and performance-related traits of a genotype in a given environment. Plant phenotyping is the act of determining the quantitative or qualitative values of these traits. Plant phenomics is the study of phenomes of multiple genotypes, given that a phenome consists in principle – of the set of all possible phenotypes of a given genotype[4].

Which are the main variables/traits in a phenotyping process? The three major classes of variables are:

- **Physiological traits:** Directly related to the organizational levels' functions. Physiological traits usually evolve to help an organism fit into its ecology [8]. E.g., rate of photosynthesis, water content [4].
- **Performance-related traits:** A plant's characteristic that gives information on any production factor of the plant. E.g., Biomass, seed yield [4].
- **Structural traits:** Related to the whole plant's morphology and visible characteristics or its different organizational levels. E.g., Leaf shape, expansion rate [4].

Some measurable variables thru 3D shoot-level imaging systems are:

- Root system architecture,
- Surface area and its textures,
- Whole-plant length measurements,
- Leaf number
- Leaf length and form measurements
- Position,
- The angle of individual elements such as leaves or lateral roots.[4]

Most of them referred to as structural traits aiming at visible characteristics at a canopy or whole-plant organizational level. Figure 2.2 summarizes better these traits/characteristics.

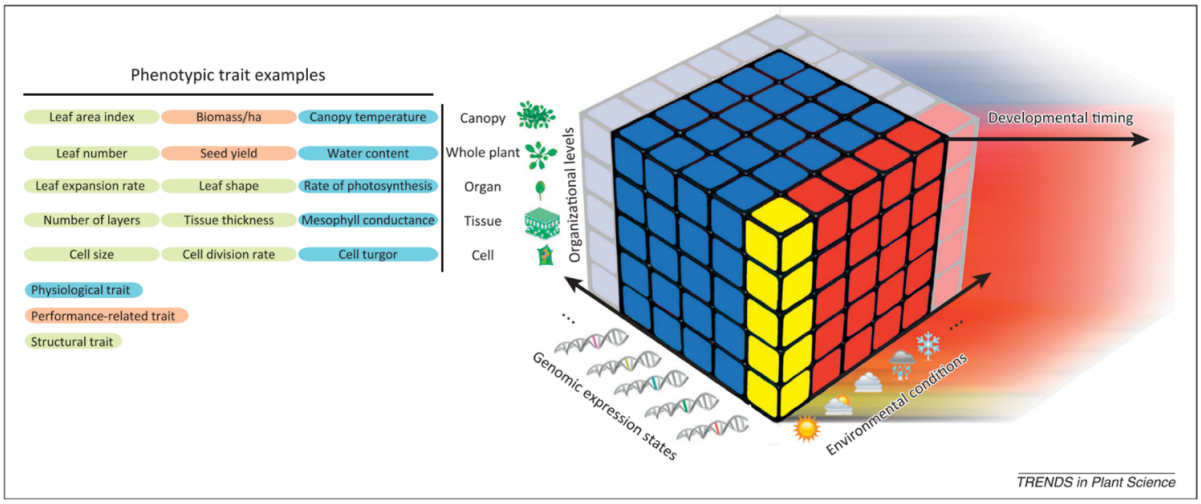

FIGURE 2.1: Traits in the process of phenotyping [4]

2.2 Measurement of phenotyping systems

In addition to these concepts, one must establish systems measurement for plant characterization. To achieve this, some key concepts are presented in figure 2.2. For this project, the throughput is the most significant aspect to consider. Further *dimensionality* and *spatial/temporal resolution/span* is not in the project range but can be the area for future work.

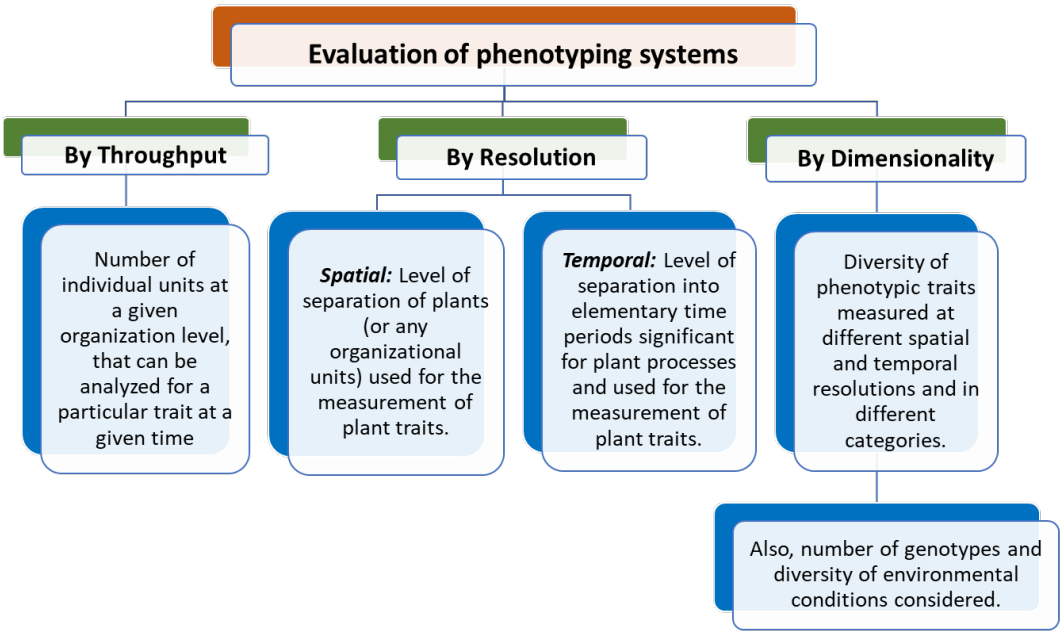

FIGURE 2.2: Key concepts for phenotyping systems. Based on [4]

2.3 State-of-art in 3D scanning methods for plants

Through the last two decades, researches have developed many applications thanks to processing power improvement with laser technologies, stereo-view, and light-based scanners [2]. Some research conducted includes method comparison [2], low-cost depth sensors/cameras applied to whole plant modeling [9], high throughput automated systems on barley plants with organ analysis [10], and most recently, segmentation software for organs/traits identification [11]. The trait identification software focuses on the detection of stems and associated leaves from the point cloud. A 3D skeletonization algorithm cuts the point cloud along the vertical direction. It links the groups between adjacent layers that separate the individual sheets [12]. Other proposed morphological segmentation software also focuses on modeling the shape of leaves and stems. This process is done by grouping triangulated surfaces related to plant locations with a high correlation to the desired traits [13]. In any case, in most of these approaches proposed in the topic, high precision results and traits such as leaf count, biomass index, stem diameter, angles related to the leaves, among others, were obtained using any of the proposed methods. These results were achieved, as long as its resolution was acceptable to scanners. A summary of the most common 3D methods is represented in figure 2.3.

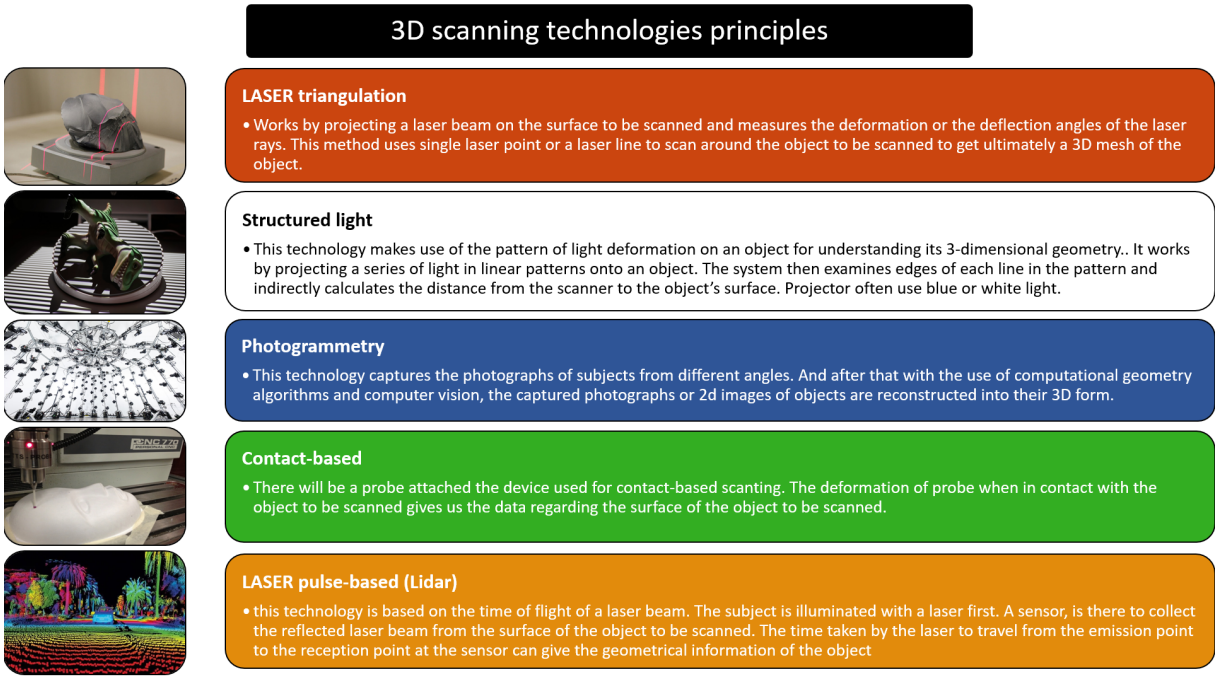

FIGURE 2.3: Most common 3D scanning methods. Based on [14]

2.3.1 Laser-based 3D Scanners

Laser triangulation systems are popular for indoor object scanning with industrial quality, which is 10μm resolution (point-to-point distance) and 40μm point accuracy [15]. In plants, a single plant's scanning process consists of several manually executed scans, mostly in controlled environments. On the other side, current traits extraction algorithms have proven to be accurate at separating leaf from stem points by making a triangle approximation for leaves and a cylinder approximation for stems

[6]. An acceptable low-cost scanner configuration consists of a single laser and a single camera that would move or rotate to have better results [16]. The principle of the triangulation technique takes into consideration the knowledge of the distance between two known points, between a pair of angles taken to a third common point. The known points correspond to the positions of the laser and the camera, pointing to the same object.

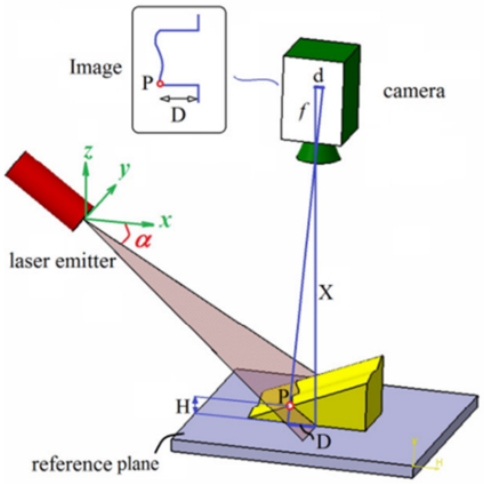

FIGURE 2.4: Traits in the process of phenotyping [16]

2.3.2 Time-of-flight Scanners

Many applications use flight time sensors, and without a doubt, the Kinect family stands out for its price-quality rate. A Kinect v2 is capable of capturing RGB images at 1920px X 1080px and 30 frames per second (fps), depth images at 512ps x 424px and 30 fps, and IR images at 512px x 424 px. The Kinect v2 sensor has a horizontal AOV of 70 and a vertical AOV of 60 with a detection range of 0.50-4.50m [17]. The acquisition algorithm is quite simple and involves taking multiple images around the object (from different points of view) at approximately 1m-4m . After filtering the color with IR data, the background of the desired object is removed. Then a cloud-to-cloud log is performed to filter some of the remaining noise, and finally, a mesh is estimated out of the entire point cloud [17]. The free Kinect Fusion software [18] is also available to use as part of the Microsoft SDK kit. This is an all-in-one "black-box" software. It uses depth data to track the sensor's 3D position and reconstruct 3D models of the physical scene in real-time, automatically turning the Kinect sensor into a relatively portable 3D scanner.

2.3.3 Plenoptic camera-based scanners

In computer vision and 3D modeling, some researchers have considered the use of plenoptic cameras over 100 years. However, this technology remained relatively dormant due to insufficient processing power and poor lens quality [20]. These cameras use an adapted version of stereo-imaging or photogrammetry. However, instead of different sensors or different acquisition angles, just one is used with a carefully designed array of lenses behind the camera's primary lens. Many state-of-the-art approaches to this topic address the direction of light travel [21], depth from

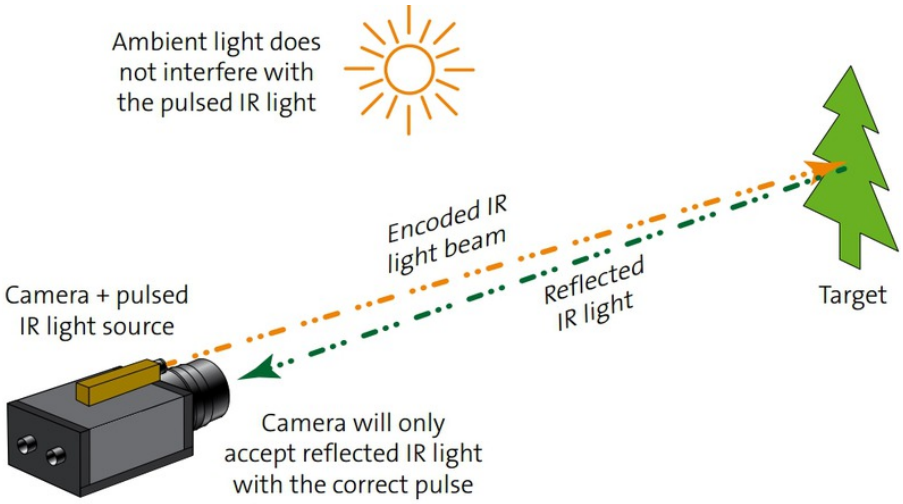

FIGURE 2.5: Time-of-flight sensor basic concept [19]

defocus, and correspondence between images [22]. Ending in the deliver of a set of 3D coordinates in absolute units within a well-defined frame in the measurable range of a given camera [20]. A significant advantage of these cameras is the amount of information they can handle, including the intensity, location of the light, and the direction in which it traveled. This data provides enough information for a 3D reconstruction using refocus and digital zooming [21]. Although this yields a set of different extra problems like Poisson noise in the light field, higher computational cost, and virtual imaging appearance, its results are promising.

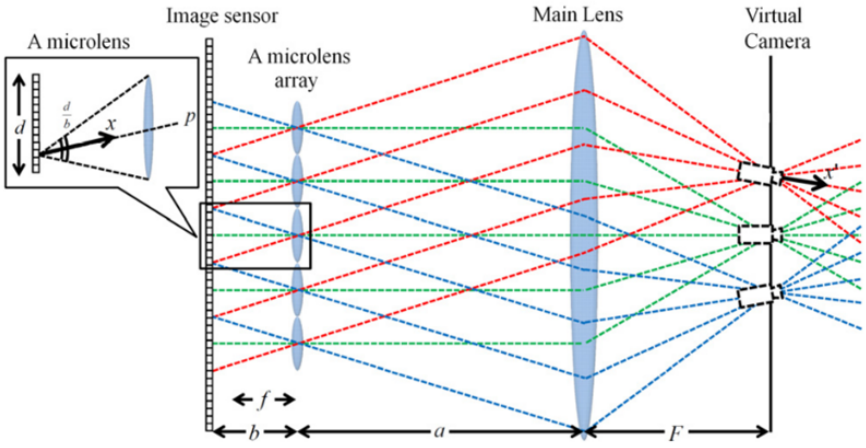

FIGURE 2.6: Plenoptic camera's lens array overview [23]

Chapter 3

Laser triangulation 3D scanner

3.1 Scanner's hardware

3.1.1 Hardware overview

An illustrative connections diagram of the hardware including the laser-line module, the stepper motor with its driver and the Raspberry Pi board can be observed in figure 3.1, as well as the definitive hardware setup used in figure 3.2.

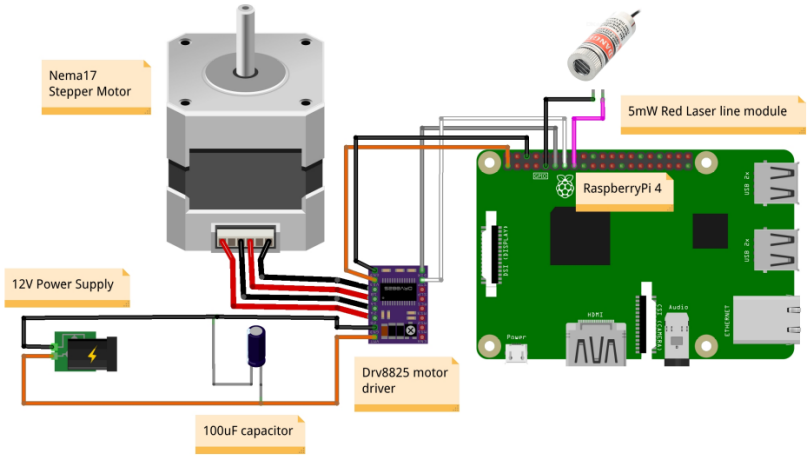

FIGURE 3.1: Hardware general connections diagram.

3.1.2 Raspberry pi

The raspberry pi board is a versatile and multi-functional device which is used to acquire image data from our USB webcam, control the position and movement of the stepper motor, toggle the power for our laser-line module, and ultimately store this information and leave it available to be processed in our main processing unit. For the project, a Raspberry Pi 4B with 2GB of RAM is used. For the tasks it manages, it has more than enough processing power, and by being able to run with a Linux OS it provides a wide variety of solutions for sharing content. The large documentation and broad community of this board was the main reason of its selection. For further development in this project, a cheaper and less powerful board should be enough.

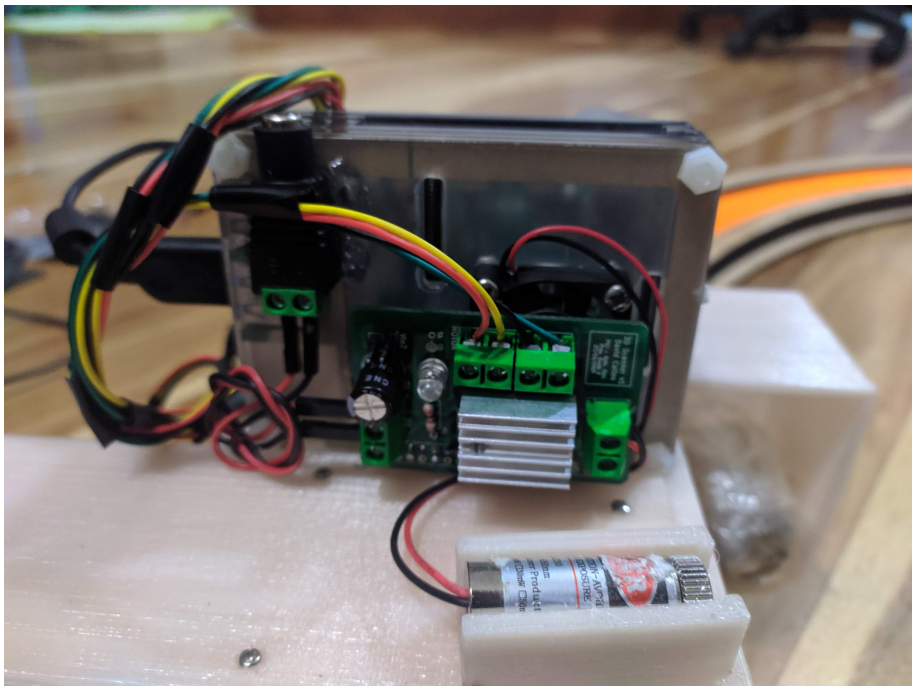

FIGURE 3.2: Hardware definitive hardware montage montage.

3.1.3 Stepper motor

In 3D scanning as well as in 3D printing applications, the motors precision is key in obtaining acceptable results, by using a stepper motor we turn all the position control in the system to a simple PWM generator. In this case for its large amount of documentation and general acceptance in the topic, a NEMA17 stepper motor was selected together with a DRV8825 stepper motor driver carrier. The selected motor has a 3.17 Kg/cm torque and 200 steps/revolution which traduces to steps of around 1.8°providing the system with high accuracy for the embedded system movement. The motor connections diagram can be observed in figure 3.3. The current limit of the driver must be set in order to avoid motor damage by adjusting a reference voltage though a potentiometer following equation 3.1. The stepper mode will be used in full-step mode, which for this motor implies using 70% of its current rating.

$$I_{limit} = 2 * V_{ref} \tag{3.1}$$

3.1.4 Laser-line module

The following characteristics where taken into account to select the appropriate laser-line module:

- **Color:** Because plants (green objects) are expected to be scanned, one should select the most easy-to-separate colour from the expected objects. In our case 650nm (red).
- **Power:** Increasing power eases the laser segmentation process, and enables the possibility of scanning objects that are further away from the laser. Having said that, the trade-off between power and price scales fast to very expensive laser modules for a small increase in power, without mentioning that more

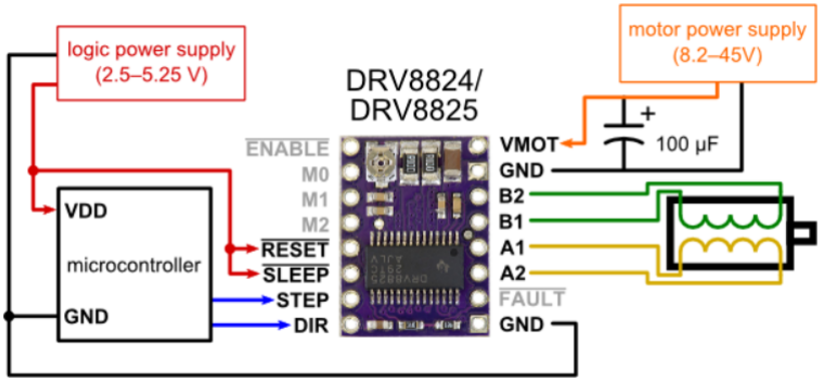

FIGURE 3.3: Basic connection diagram for a Nema17 stepper motor

powerful lasers would need of greater power sources. In our case, this decision was price-oriented to a 5mW laser line.

- **Line thickness:** Related to the quality and focus capability of the laser-module as well as the distance to the object and its material. A thinner laser line will allow a better quality of the scan, although many software techniques can be used to minimize the impact of wider laser lines.

  Very few documentation is found apart from the already mentioned basic specifications of this module. Summarizing, its working voltage range goes from 3.0V to 4.5V, with a measured current (because theoretically, it varies from one datasheet to another) of 11mA tops at 3.3V. Taking into account that the Raspberry Pi 4 has a specification of 16mA current limit per GPIO pin without exceeding 51mA (max output current from 3.3V internal voltage regulator), it is possible to directly connect the module to any available GPIO pin from our board without damaging neither of them.

  Conclusion: Based on the the information above, a 5mW 650nm red laser-line module was used in the implementation. After measuring its working current, it was decided that connecting the module directly to a GPIO pin from the Raspberry Pi could be safely done.

3.1.5 Camera evaluation and fixed variables from it

For the camera selection, resolution, field of view, and cost where the considered factors. Leaving all other aspects fixed, these three imply better scan resolution, potential to scan bigger objects and a cheaper scanner implementation respectively.

The field of view (FOV) of a lens/camera means how wide or narrow the field of vision of the device is. It will vary depending on various factors, but usually, a wider angle traduces either in a more expensive camera or in a poor image quality. Manufacturers often provide these measurements, in horizontal, vertical, and diagonal field of view (HFOV, VFOV, and DFOV), where the HFOV will almost always be wider than VFOV. In our application, the FOV is of great importance because it fixes the maximum height that the camera can record, hence reducing the tallest object that can be scanned while having the camera at a certain distance from the center of platform.

Based on research, the rice plants have an average height between 30cm and 1m. However, for this first prototype, objects of more than 70cm are not expected. The equation for calculating the necessary distance from the center of the platform to the camera lens is shown in the following figure.

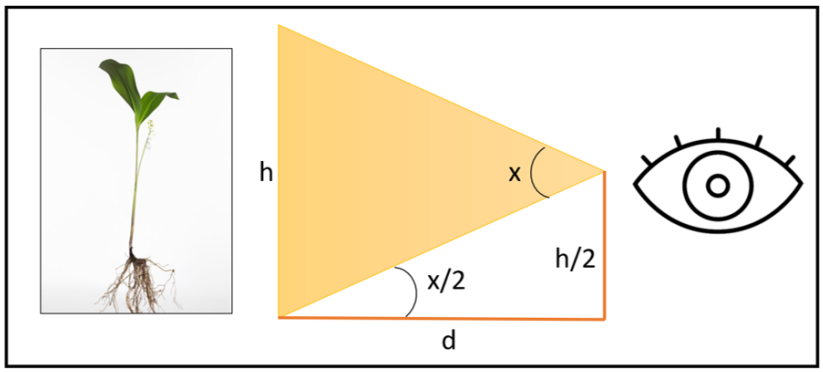

FIGURE 3.4: FOV calculations diagram

And the resulting and most simplified equation to calculate these distances is described as:

$$d = \frac{h}{2tan(\frac{x}{2})} \tag{3.2}$$

Four different cameras were evaluated for this purpose, mainly based on availability and price.

- Raspberry pi camera V1: commercially available for less than COP50.000
- Raspberry pi camera V2: commercially available for less than COP150,000
- Logitech C920 camera: commercially available for less than COP450,000
- Brio UHD Pro camera: commercially available for less than COP850,000

For each camera, the field of view, resolution, and the necessary distance for the camera to be able to capture a 100cm tall object, are summarized in the next table:

X	Rpi Cam V1	Rpi Cam V2	Logitech C920	Brio UHD Pro
Resolution	25921944px	32802464px	1920x1080px	4096x2160px
Pixel size	1.4um	1.2um	-	-
HFOV	53.5°	62.2°	70.42°	82.1°
VFOV	41.4°	48.8°	43.30°	52.2°
Radius	1.003m	0.832m	0.714m	0.575m
Horizontal limit	0.379m	0.377m	0.283m	0.282m

TABLE 3.1: Basic information of different evaluated cameras

Note: Because the HFOV is always wider than VFOV and objects won't be significantly broad, calculations were using HFOV instead of VFOV. This requirement implies a reduction in the HFOV, which is affordable in our application.

Although the most suitable camera from the evaluated ones for our application would be the *Brio UHD Pro Webcam*, it was not available for usage at the moment of the implementation, reason why the *Logitech C920* was selected.

Conclusion: After evaluating different camera options, the platform radius is initially defined to be of 80.0cm for compatibility calculations with other mechanical parts. Also, the camera to be used is the Logitech C920 webcam.

3.1.6 PCB design

A small PCB was designed and manufactured in order to integrate much more professionally and aesthetically the motor circuit and all the connections with the Raspberry Pi 4B. Specifications of the PCB include:

- All components are thru-hole type.
- All paths are no less than 0.7mm wide (just in case any peak high current output for motor occurs).
- No 90°turns.
- All connections made in a single layer.
- No ground plane (since no significant analog signals or dynamic range requirements need to be taken care of).

The PCB was entirely designed in KiCad, a completely open-source software for PCB design. Its diagram and 3D view can be seen in figure 3.5 as well as a picture of its manufactured and welded results in figure 3.6.

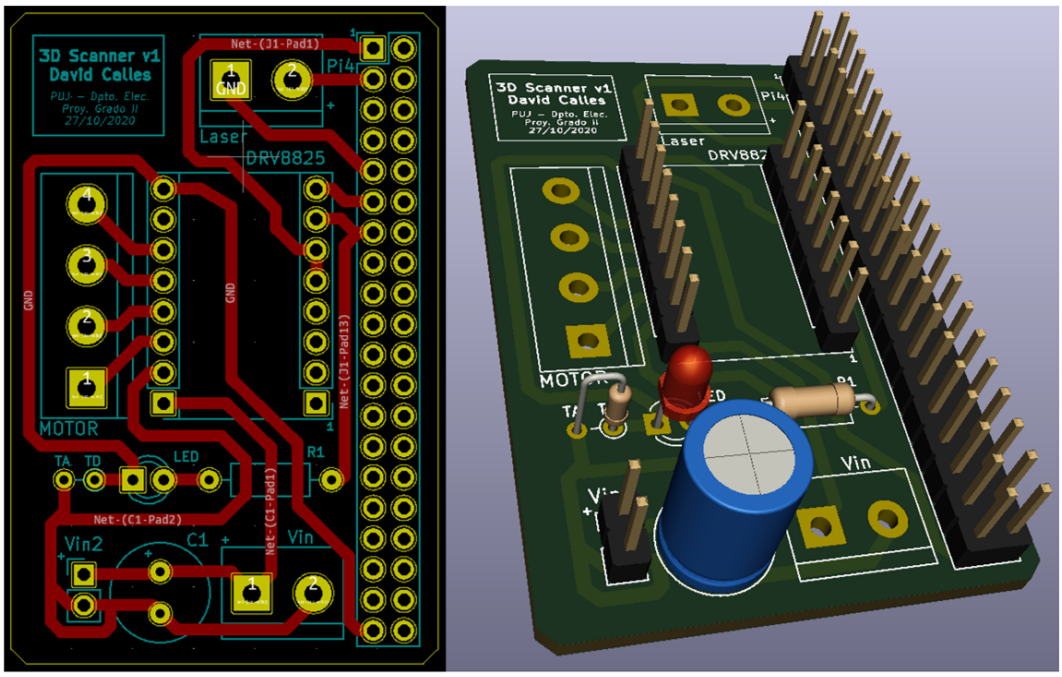

FIGURE 3.5: PCB's design diagram(left) and 3D view(right).

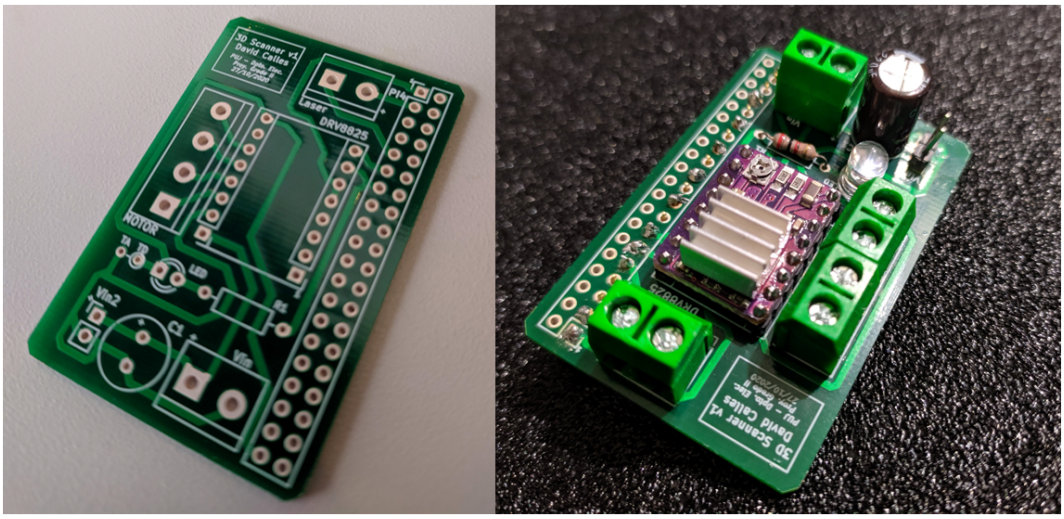

FIGURE 3.6: Manufactured(left) and welded(right) PCB.

3.2 Mechanical structure/setup

3.2.1 Setup overview

Most 3D scanners consist of a static (or various) camera and a rotating platform for the object. In our case, we want to scan objects that are either heavy or very prone to change their shape with very small movements. Which is enough reason to try to avoid moving the objects through the entire scan. In this project, a moving sensor is proposed, using the same rotary geometry, but moving the acquisition device instead of the object. This supposes a precise mechanical structure for the whole image acquisition process.

The montage proposed consists of a big round platform and a moving box/car with gears as wheels. These will contain all the devices, boards, circuitry, and sensors for the data acquisition. An graphical "idea" of the montage can be visualized in figure 3.7 where the object to be scanned is located at the center of the platform.

3.2.2 Design process

After having fixed the platform diameter, the next step was to calculate the gear/pinion parameters. For this, the following considerations were taken into account:

- Because the motor allows about 200 steps per turn, (1.8°precision), there is no specific requirement of velocity/position ratio between the gears.
- To avoid obstruction problems, it is recommended that the number of teeth is greater than 17.
- A pressure angle of 20°was implemented, as adviced in literature.
- To allow the gears to describe a circular trajectory, bevel type gears were designed.
- The mechanical montage includes two smaller gears, and the platform (bigger gear).

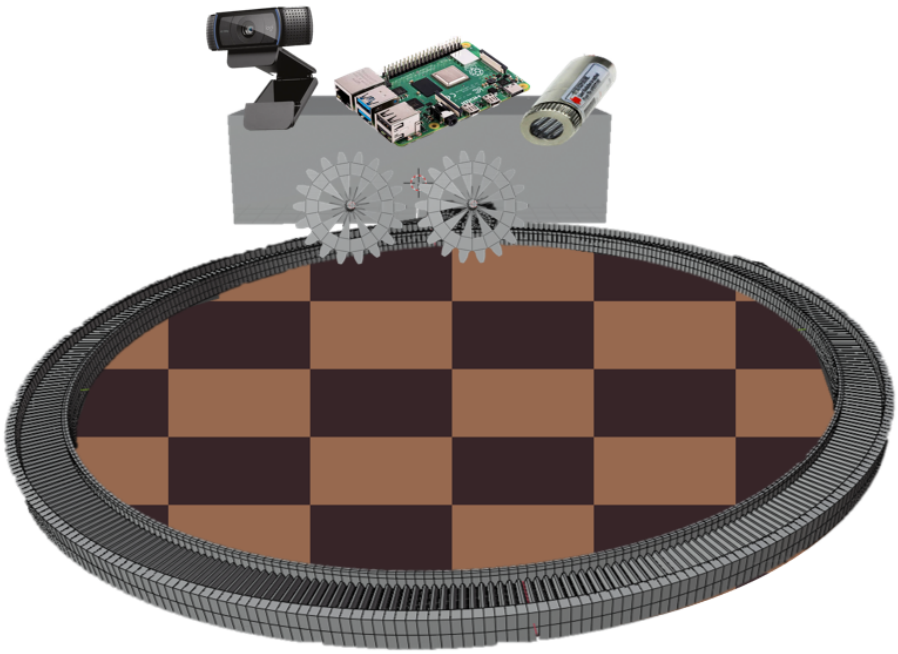

FIGURE 3.7: General "idea" of mechanical montage.

The platform (bigger gear), the smaller gears, and the support for the devices to be used were all designed in Blender 2.8. The gears were designed aided by the calculator and tutorials proposed by [24]. The support for the devices to be used, as well as the models of the devices, were designed entirely from scratch.

The gears measurements are summarized in the following table:

X	Radius	Teeth	Module	Face width	Center radius
Small pinions	4cm	18	4/9	7cm	0.4cm
Big gear	80cm	360	4/9	7cm	60cm

TABLE 3.2: Basic information of different evaluated cameras

On the other hand, the prototype of the mobile which will carry the laser-line module, the camera, the stepper motor, and the raspberry pi 4 board was modeled in blender taking into account the following specifications, as seen in figure 3.9:

- Gears size and respective spinning axis.
- Cases or space for Raspberry pi 4, Logitech C920 webcam, laser module, support for stepper motor, and support for a counterweight.
- Barriers on both sides of the platform.
- Curved surface of the platform, which translates into a curved path for the mobile.

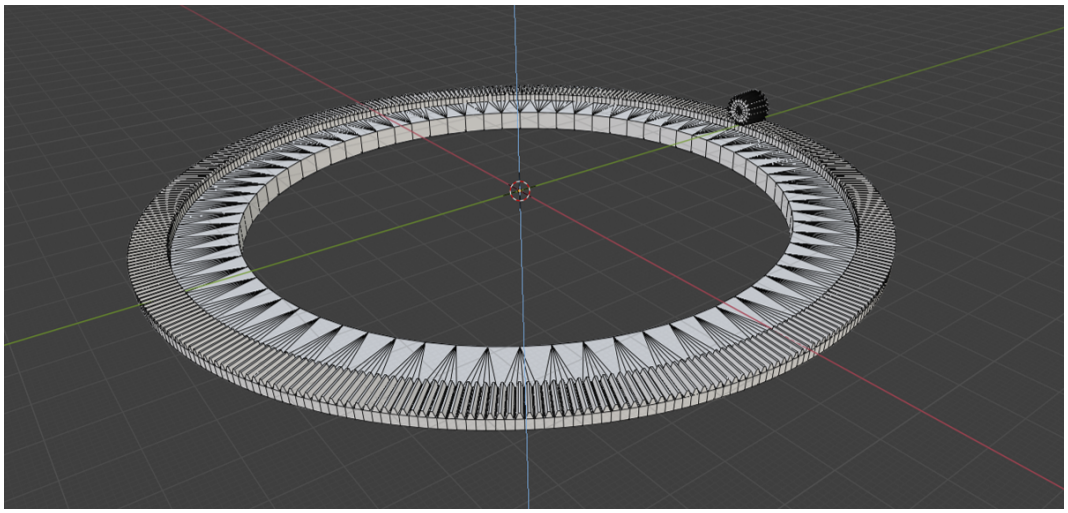

FIGURE 3.8: Gears model in Blender

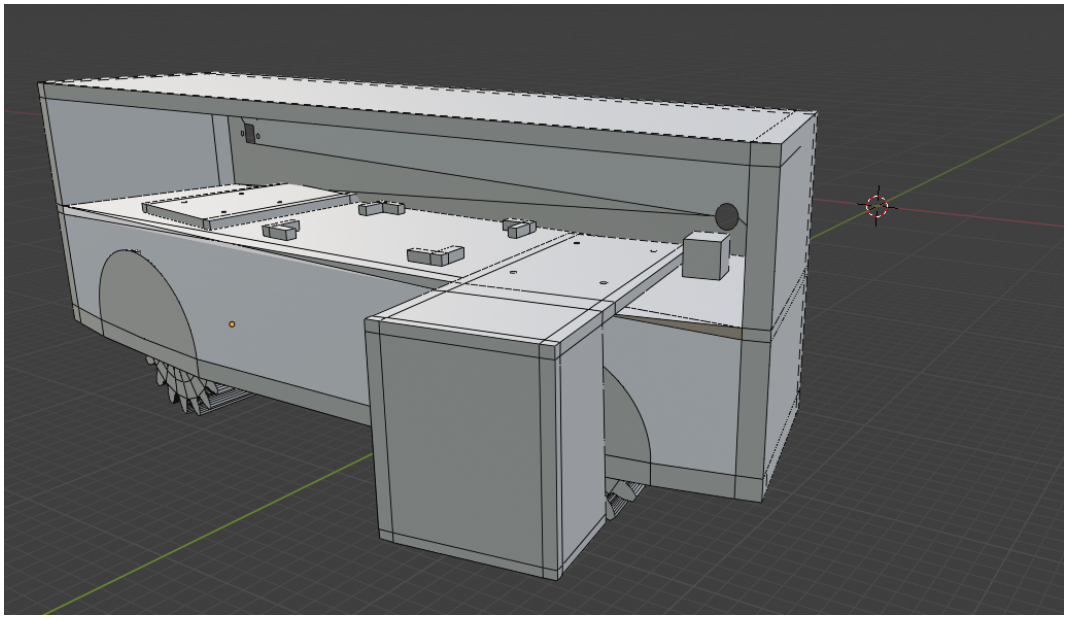

FIGURE 3.9: Blender 3D model of the mobile for 3D scanner

3.2.3 Resolution of mechanical montage

Based on these design parameters for the mechanical montage, the theoretical resolution restriction of the scanner was calculated based on the mechanical theoretical montage capabilities.

For these calculations, some useful equations are:

- Arc of a circumference

$$S = r * \theta \tag{3.3}$$

- Radians to degrees ratio

$$\pi/90 = rad/grad \tag{3.4}$$

Taking into account that the resolution of the Nema17 motor is 1.8°tops, the correspondent circumference's arc that it describes on the small gear can be calculated as shown next.

$$\begin{aligned} S_s &= r_s * \theta_m \\ &= 0.04m * \frac{\pi}{100} \\ &= 1.257mm \end{aligned} \tag{3.5}$$

Which is the same distance that will be described on the platform (big gear) so the angle will be:

$$\begin{aligned} \alpha &= \frac{S_s}{r_b} \\ &= \frac{1.257mm}{60cm} \\ &= 2.094 * 10^{-3} rad = 0.12^\circ \end{aligned} \tag{3.6}$$

If we suppose that a considerable object to be scanned will have an average radius of 20cm, the angle can be traduced into a resolution of:

$$\begin{aligned} resolution &= r_o * \alpha \\ &= 20cm * 2.094 * 10^{-3} \\ &= 418.9um \end{aligned} \tag{3.7}$$

Conclusion: The mechanical capabilities of designed mechanical setup (gear pair), provides us with the capability to have a resolution of up to 418.9um for a regular 20cm-radius object which for our application is a more than acceptable error. A better resolution could be achieved without making further changes to the hardware by using the micro-stepping capabilities of our DRV8825 driver motor which goes up to 1/32 of a step.

3.2.4 3D printing and laser cutting

Models observed in last subsections had to be mostly redesigned in order to easy its fabrication. Avoiding hanging structures, material waste and objects with any dimension having more than 18cm. To do this, the proposed modifications are listed below:

1. Holes where made through both, the small gears and the box/car.
2. The box/car was simplified by eliminating many of the hanging faces and many of the "features".
3. The small gears were chopped in halves to avoid the need of support material, aiming to join them later using any plastic adhesive.
4. The box was separated into four different pieces, aiming to join them latter using any plastic adhesive.

5. The caps for the small gears in the bottom part of the box were eliminated to avoid support material.

6. The big gear was separated into a "to 3D print" part and a "to cut with laser" part using MDF. The toothed section is destined to be printed and the structural base expected to be cut with laser.

7. Teeth from the base where separated into segments of less than 20cm. The result where 20 18°-small segments to be joined.

Models obtained after these modifications can be seen in figure 3.10.

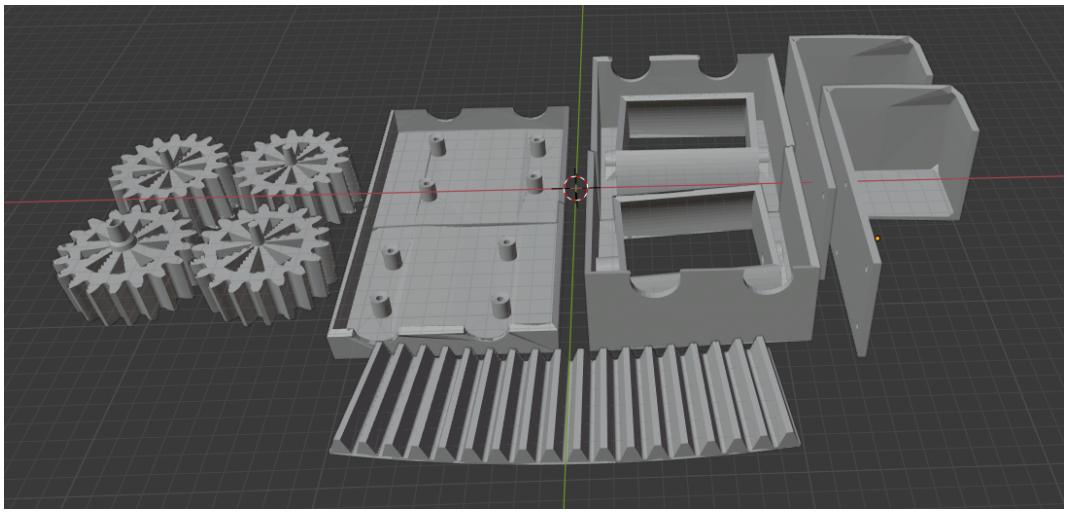

FIGURE 3.10: 3D models for 3D printing after redesign.

3.2.5 Implementation

All parts showed in figure 3.10 were printed in 1.75mm PLA material using a Tayrona Prusa i3 3D printer, as showed in figure 3.11. General configurations for the prints are the followings:

- Honeycomb fill patter at 30%
- Layer thickness of 0.25mm with first layer at 0.32mm
- Extruder nozzle temperature set to 204°C
- Heating bed temperature set to 70°C
- Speed of print set to slow (perimeter print speed at 20mm/s, infill print speed at 25mm/s, with first layer modifier of 30%)

On the other side, the base as well as the limits for the mobile where laser-cut in MDF, the base with a thickness of 5mm and the limits with a 18mm thickness, both cut as 4 equal quadrants of a whole *center-less circle*. Finally, all where glued together using generous amounts of glue for wood. The side view can be observed in figure 3.12.

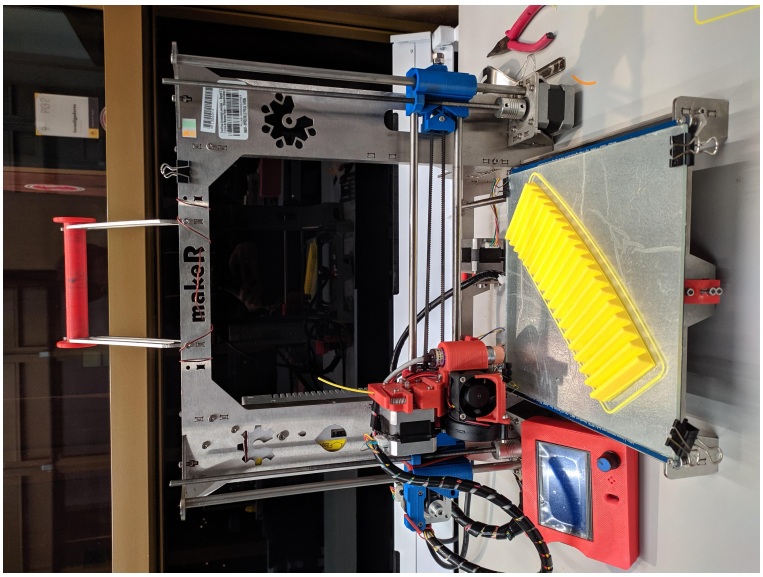

FIGURE 3.11: Tayrona prusa i3 3D printer with printed toothed segment

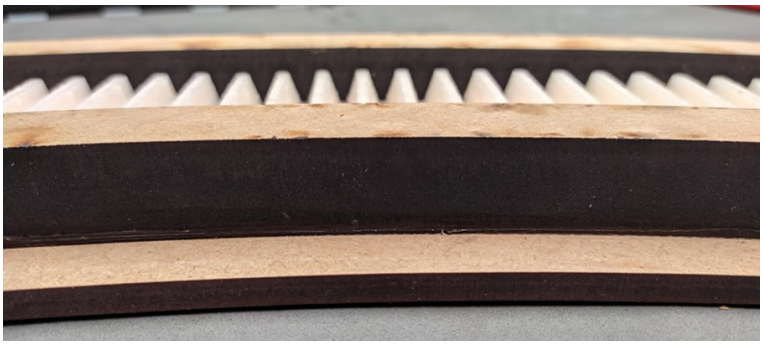

FIGURE 3.12: Ground level view of the wooden base plus toothed segment.

The 3D printed pieces meanwhile, where joined together using epoxy adhesive. And the toothed segments where also fixed in the wooden base using this same compound, figure 3.13 shows an artistic view of the 3D printed assembled "car".

Finally, figure 3.14 illustrates the final setup used for data acquisition in next sections.

3.3 Scanner's software

3.3.1 Software overview

Figure 3.15 illustrates the general picture of the developed software blocks presented in this section. In a nutshell, the software that runs in the Raspberry Pi 4B is in charge of acquiring the raw data (images) on demand, by managing the camera, the motor and the laser-line module properly. Here data is left available to the computer. On the other side, a laptop/desktop computer is where all the heavy data and image processing is executed, to ultimately achieve a set of 3D points by integrating data from all the obtained images.

FIGURE 3.13: Assembled "car".

FIGURE 3.14: Final mechanical setup for data acquisition.

3.3.2 D2D wireless connection

MQTT: Establish connection

The first part of a single scan involves establishing a connection between the scanner(RPi) and the computer. This involves the scanner to be able to *listen to* potential

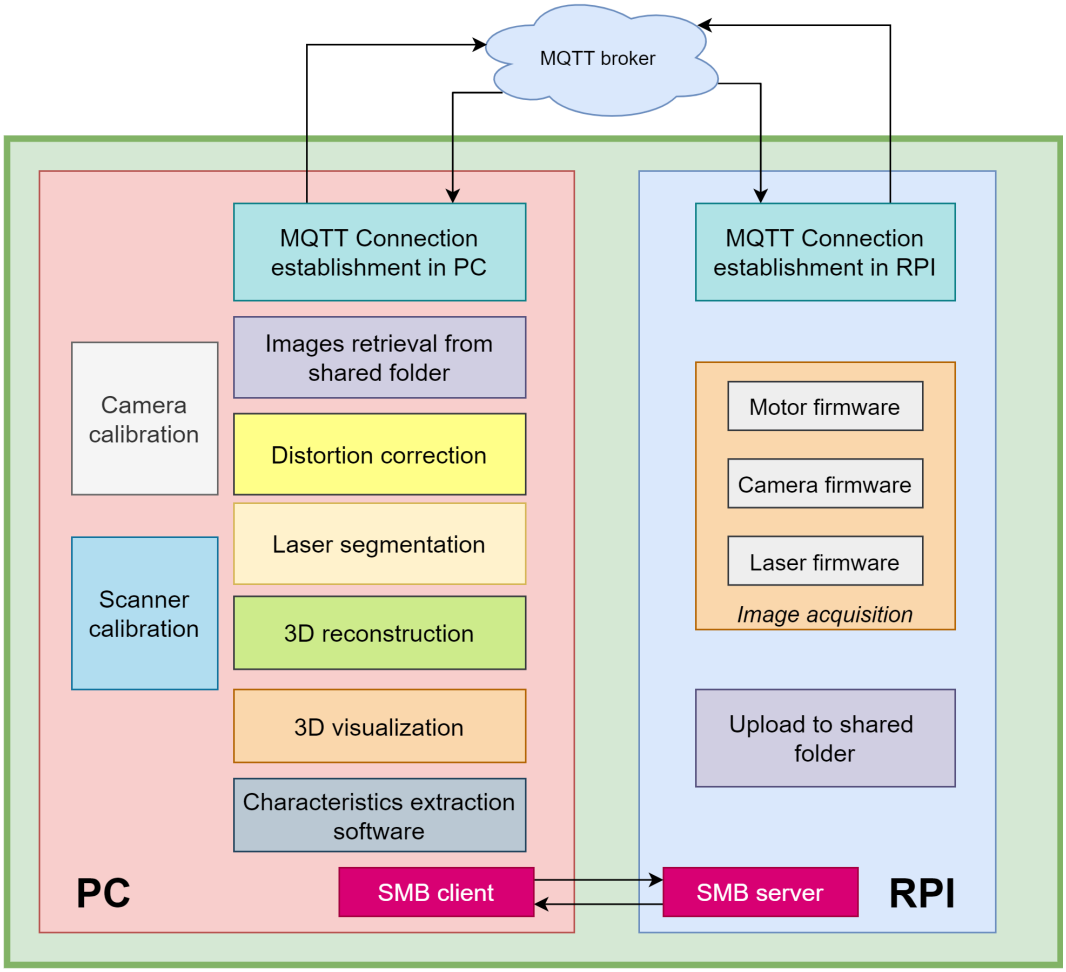

FIGURE 3.15: Laser triangulation scanner software overview

scan requests, and the computer to be able to *request* the scanner a certain amount of data. For this purpose, and due to it's popularity, reliability, and ease to implement, MQTT is used, which is TCP/IP Machine to machine communication protocol widely used in communications between *IoT devices*. In this project, both devices connect to a free-to-use *broker* and subscribe to the same *topic*. Of course, the raspberry must be already subscribed when the computer places a request.
When and if the connection is established properly, the scanner will be *listening to* the MQTT topic. From the computer, an initial message is sent and if the scanner receives it, a message exchange will begin, giving the details of how many images to take, and where to save them inside a shared folder. After all details are received correctly, the computer closes its connection and the data acquisition in the scanner initiates. Figure 3.16 illustrates in detail the messages exchange.

It's implementation was done in python 3 using a well known MQTT library called *PahoMQTT* on both devices. The broker used was *broker.hivemq.com* with a common *topic* and using a Quality of Service (QoS) of 2, which implies ensuring messages will be delivered and without duplicates.

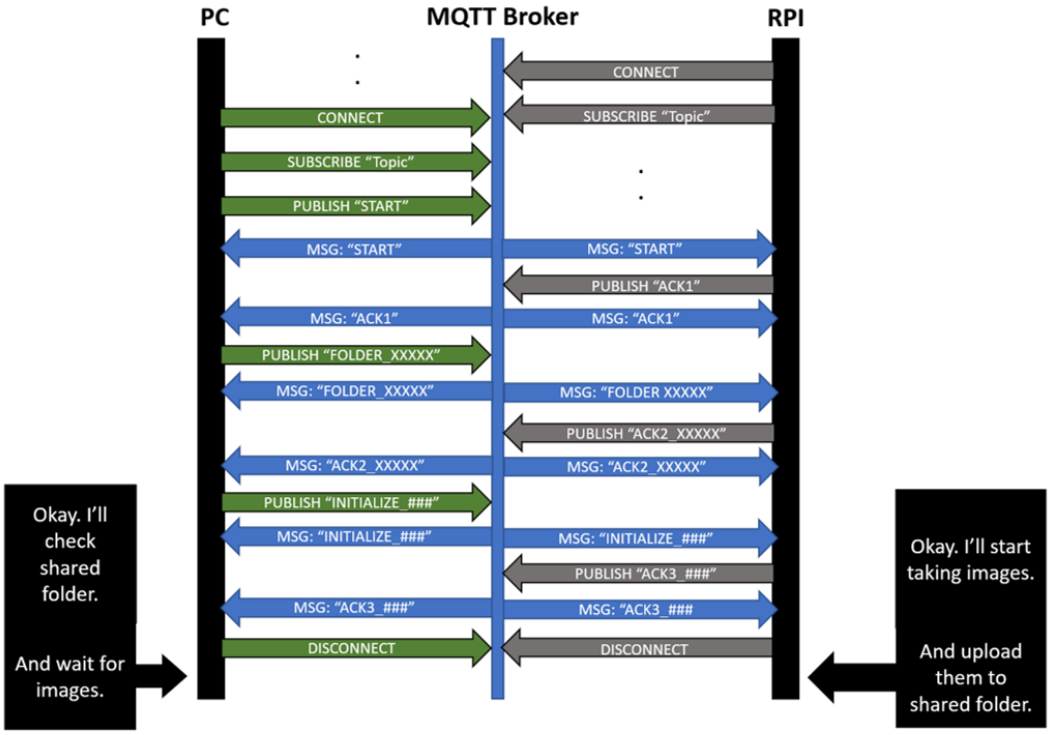

FIGURE 3.16: MQTT connection establishment (handshake)

CIFS/Samba: Share images

Aside from sharing messages between the devices, another tool is used to share the actual data, the images. For this purpose, a CIFS instance was created. CIFS is a Server-Client TCP/IP based communication protocol, previously called SMB. For Windows (the PC) there is a native implementation (as Active Directory) for this protocol, so just enabling SMB connections was enough. On Linux, on the other hand, to create a server, it was needed to install samba, create in advance the directory to be shared, and write the corresponding configuration file in the local directory *"/etc/samba/smb.conf"*.

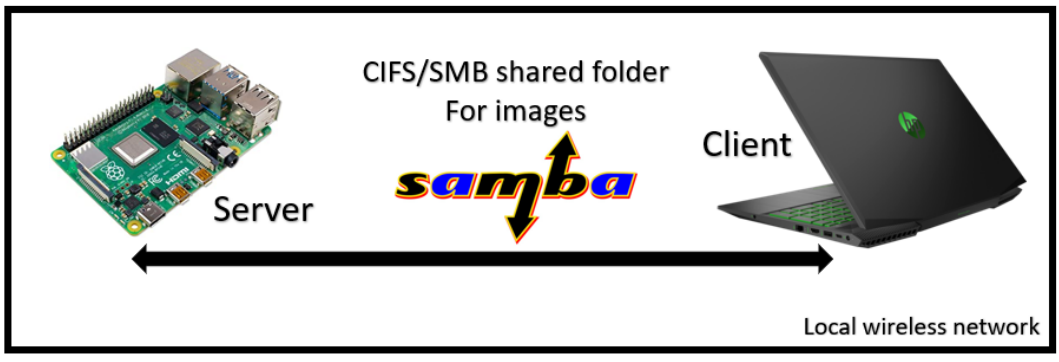

FIGURE 3.17: SMB Protocol idea

3.3.3 Embedded system's software

Motor and laser software

Both the laser and the motor (with its corresponding driver and circuitry) can now be fully controlled with 3 digital pins from the Raspberry's GPIO. One for the laser (direct value of the laser) and two for the motor (direction and step signals). Having said this, the reader should agree that no need for a diagram on the laser software is needed as it is directly connected to the Raspberry's GPIO. On the other side, for the motor, there is a need to review its usage.

Three variables are taken into account to actuate the stepper motor, these are the amount of steps it is required to move, the duration of the steps in time, and the direction in which it will move. In our application, a full step of the Nema17 it is of 1.8°, which means that a single step (rising edge plus falling edge) will cause the motor to angularly move this amount. Now, the step duration can be used to control the speed in the movement. In this application, although position control is crucial, speed control is not necessary. Thanks to this, the duration of the step was manually tuned to one that would make the car movement "feel smooth", in this case 4ms. Last but not least, the direction, being (from the motor perspective) clockwise if signal in *HIGH* (3.3V) and counter-wise when *LOW* (0V).

The bigger the amount of steps between shots, the bigger the distance between image pairs, resulting in less quality but faster scan. In this case, the amount of steps set per movement was of 10 steps, meaning a theoretical 18°angular movement between each image.

The implementation for the motor control is illustrated in figure 3.18 as a flowchart. In a nutshell, first the pins to be used are declared as outputs, then the direction pin is set to its desired value, and finally, a number of steps are generated switching the value of the step signal pin from on to off and waiting a specific time using the *GPIO.rpi* and *time* python libraries. Here it must be said that using a high level programming language as python for both writing on the GPIO registers and "waiting" time is usually not recommended, specially if accuracy in timing is needed. Nonetheless, because timing in this application would only affect the speed of the movement there is no need to go deeper into a lower level or more complicated implementation.

Image acquisition

Figure 3.19 details the general algorithm for image acquisition to be run on the Raspberry Pi through a flowchart. First, the MQTT connection is established, here it is key to add that the subroutine representing it, is a blocking function, which listens to its subscribed *topic* searching for potential scan requests. Here, both the amount of images to be taken and the destination folder are retrieved. Many cameras including the Logitech C920, can delay its normal operation when first turning them on, reason why a set of "trash" images are taken but not saved when starting the scan by turning on and off the laser module, but not the motor and thereby not moving the car. After these "trash iterations", the process is more or less monotonous until completed the demanded amount of images/iterations: turn the laser on, take an image, turn the laser off, take another image, save these images in the given directory, and move the motor a certain but fixed amounts of steps. Although it is not showed in the diagram, it must be clear that a small time is waited every iteration before taking the image and after the laser value is set, ensuring the laser is indeed in the desired state.

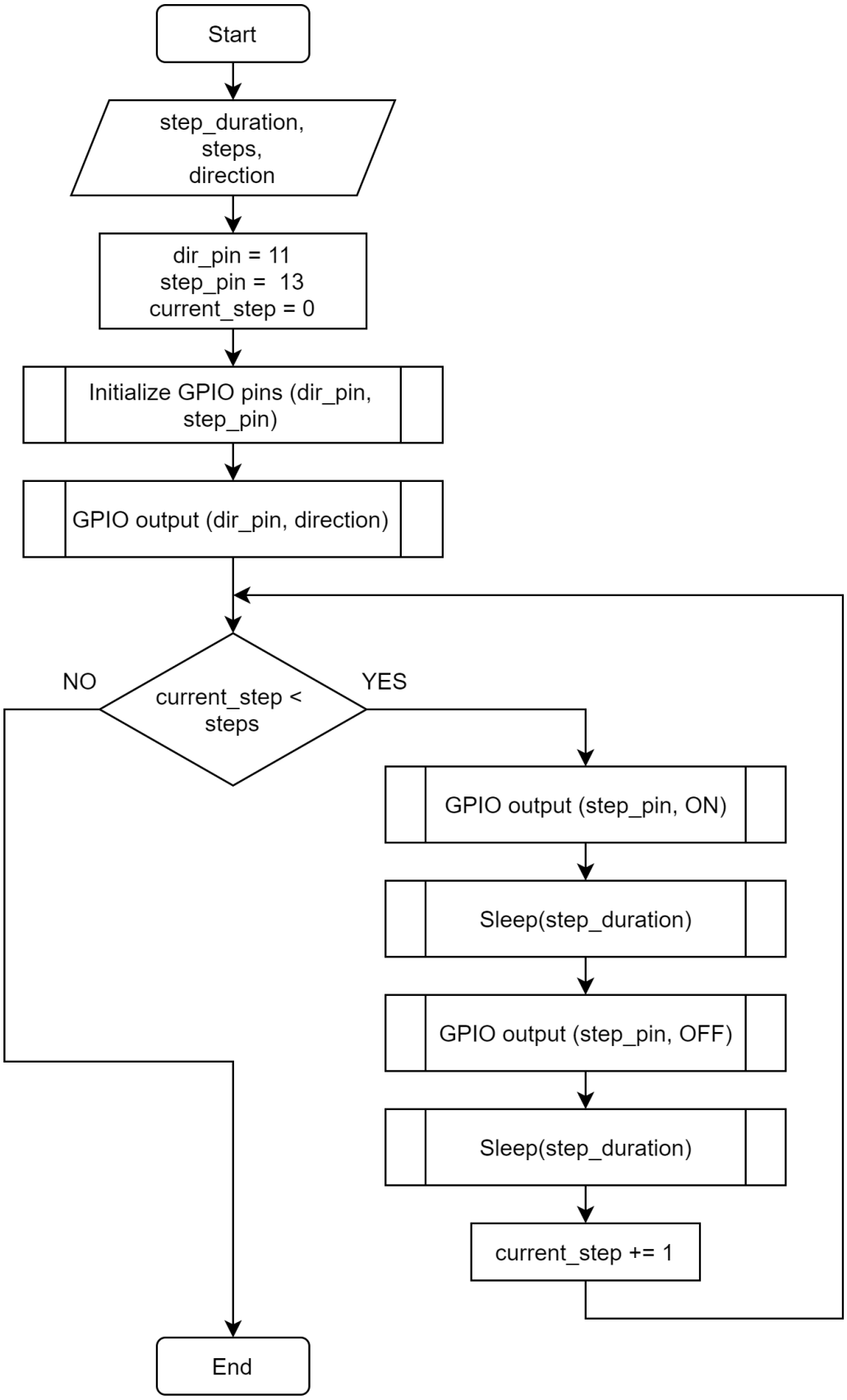

FIGURE 3.18: Motor's firmware flowchart

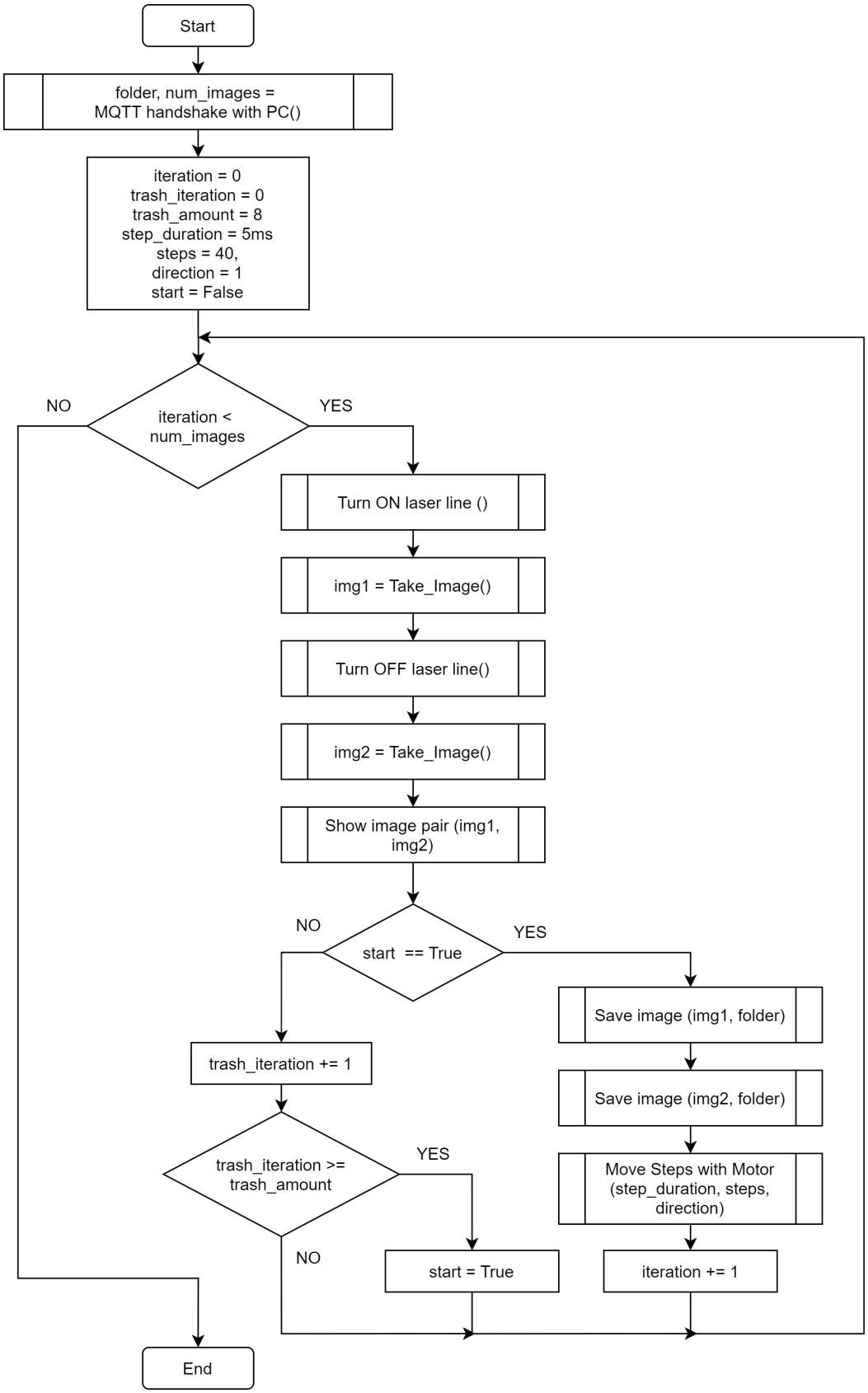

FIGURE 3.19: Flowchart for image acquisition firmware

3.3.4 Camera calibration software

"Camera calibration" is the process of estimating the parameters of a camera. This process is required to accurately set a relationship between the 3D point in the real world and its corresponding 2D projection in the image plane. Figure 3.20 illustrates the different coordinate frames used in the camera calibration. Here, we can observe the world's coordinate system with a w sub-index and the camera's coordinate system with a c sub-index. For a detailed description of the camera calibration theoretical background, the reader is encouraged to refer to appendix B.

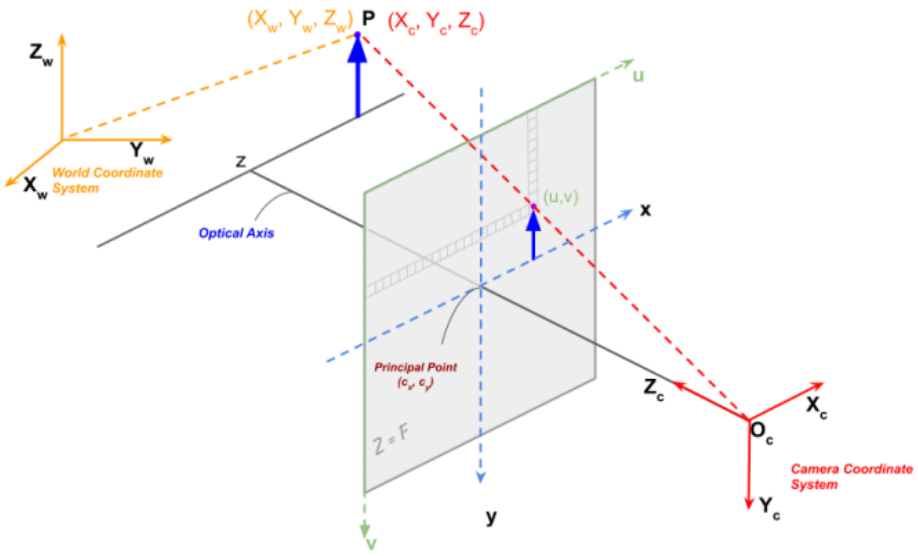

FIGURE 3.20: Coordinates for image formation in a camera [25]

Camera calibration process with OpenCv

The process of calibrating a camera has two main goals. The first goal is to estimate both the intrinsic and extrinsic parameters of a camera. The second one is to calculate the distortion provided by the camera. OpenCv facilitates a set of different functions both in C++ and Python for the calibration process. After estimating all the mentioned parameters, the distortion correction is possible because its effects are a constant related entirely to the camera.

In the calibration process with OpenCv, the two main calibration patterns widely used are *rectangular Chessboards* or *circular references*. In the exposed implementation, the first one is used. The following steps define a simple camera calibration process given a fixed set of images (of at least 10):

1. **Prepare an image set with the camera to be calibrated:** Reference pattern must have as few imperfections as possible and must be a rectangular (not squared) chessboard. Knowing the square size is also required. In our case, a set of 100 1920x1080px images of a rectangular 7x9 chessboard were taken.

2. **Create 3D object points:** Here, we wanted to establish which will be the real-world coordinates we will assign to each corner we find in the image (assuming Z=0 to ease calculations). If the sizes of the squares of a chessboard are

unknown, one could have assumed each square measured a single unit (not recommendable). In our case each square was 20mmx20mm so the test point were: [(0,0,0); (0,20,0); (0,20,20); (0,40,20)...] and so on.

3. **Find corners in reference pattern:** Here, the *findChessboardCorners* OpenCv's function was used to detect the reference pattern. If the detection was successful, corners measurements were refined with the *cornerSubPix* function, which allows us sub-pixel resolution measurements.

4. **View corner detection results:** Although this step is not mandatory, it surely helps to confirm everything went just fine. For each image, the *drawChessboardCorners* was used to draw the detected pattern. The result of this step can be observed in figure 3.21.

FIGURE 3.21: Border detection result

5. **Check calibration results:** The correction is observed both visually and mathematically. Visually by using the functions *getOptimalNewCameraMatrix* to get the new corrected matrix and *undistort* to generate the corrected version of a single image out of the new matrix. Mathematically the calibration results are evaluated by comparing the detected corners in world measurements with the projections of them. Then average magnitude of the error between them is calculated in pixel units, also known as *reprojection error*. The distortion correction results can be observed in figure 3.22.

In the proposed calibration, a set of 100 images where taken. After the first iteration of the calibration, 89 pictures were accepted by the corner detection software, meaning 11 where discarded. To found a more optimal estimation of the camera's parameters, blocks of 20 images where randomly chosen from the set without replacement, and the whole calibration process was executed. After 200 iterations, the attempt with smallest error was found, and the corresponding images and estimated parameters were stored. More precisely, the parameters obtained from the 200 iterations are (visually rounded to 4 meaningful digits):

FIGURE 3.22: Original image (left), image after correcting distortion (center), cropped corrected image (right).

$$\mathbf{K} = \begin{bmatrix} 1.383 * 10^3 & 0 & 9.335 * 10^2 \\ 0 & 1.3856 * 10^3 & 5.281e * 10^2 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.8}$$

$$Dist = \begin{bmatrix} 0.1111 & -0.1862 & 0.0006593 & -0.002097 & 0.01723 \end{bmatrix} \tag{3.9}$$

$$\begin{aligned} maxerror &= 0.06747 \\ minerror &= 0.03288 \end{aligned} \tag{3.10}$$

Distortion correction

Once our camera's intrinsic parameters and distortion coefficients are estimated accurately enough, an important initial step when using a camera as a distance sensor is to remove the already known distortion. OpenCv, presents 3 different ways of doing this. The first 2 are used to remove the distortion from the whole image, either by using the all-in-one function *undistort()* that creates de transformation map and also applies it to the image, or by doing it as separate steps with *initUndistortRectifyMap()* and *remap()* functions correspondingly. The third way is done with *undistortPoints()* which returns the equivalent undistorted points of a given array of points in the original image. Although the whole image all-in-one undistortion function was chosen, the last mentioned method was also tested during the development process of the project, but many compatibility problems were found.

3.3.5 Laser segmentation

The laser segmentation algorithm proposed in this project consists of a step-by-step process which is described as follows:

1. **Original Images:** Note that the process of laser segmentation described in this project works with the existence of two images for each perspective taken, one of the object with the laser stripe projected in it and one with out it. This allows to detect the color of the object, as well as provides more information for the problem's solution. As an example, figure 3.24 shows a pair of images used in the testing stage.

2. **Red component extraction:** Because the laser-line module used in the montage has a known red 650nm wavelength, the most suitable option to perform

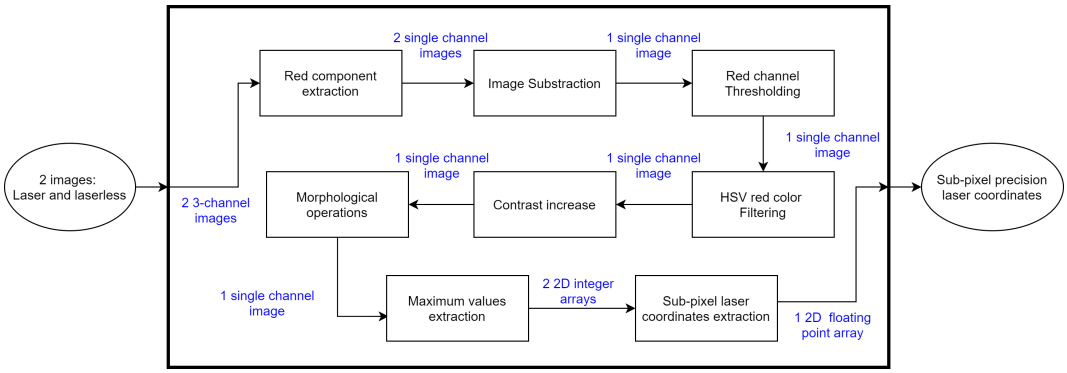

FIGURE 3.23: Laser segmentation algorithm's blocks diagram

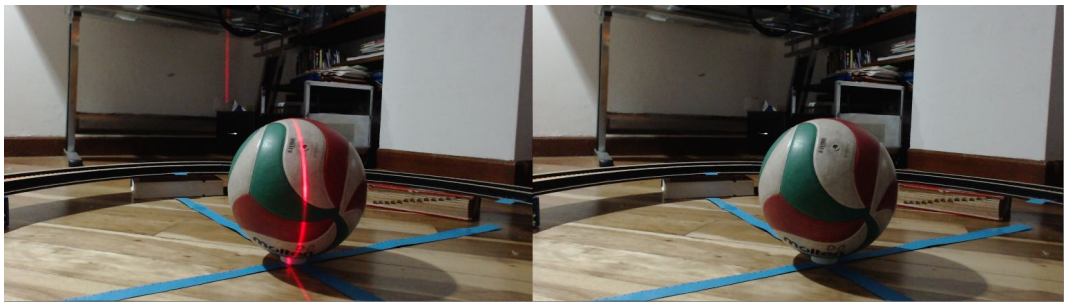

FIGURE 3.24: Original pair of images for laser segmentation

an initial segmentation of the laser pixels is from color. Two options are evaluated, an HSV range filter and a red component thresholding from the BGR image. The first showed to highly depend on the amount of light in the image, causing no simple HSV filter to provide enough confidence. On the other hand, an acceptable assumption can be made, this is to consider the laser as the brightest red element in the image. Having this in mind, only the red component from the RGB image is extracted. An example of the result of this stage is showed in figure 3.25.

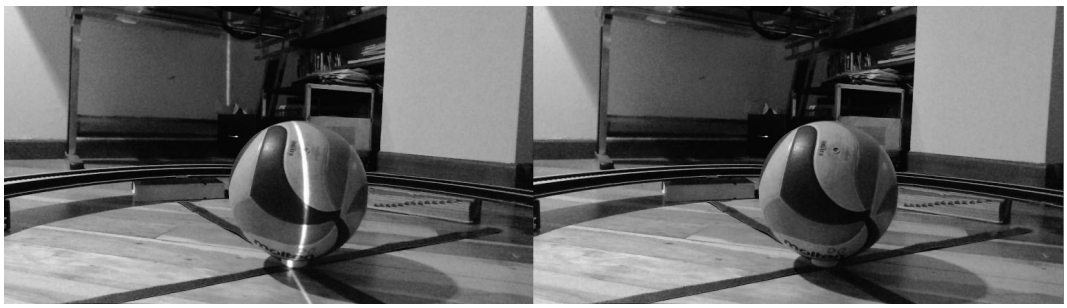

FIGURE 3.25: Red channel of original images for laser segmentation

3. **Image subtraction:** Although there are many algorithms which not rely on having a laser-less image for every lasered image, having both and applying an image subtraction is one of the most practical solutions to approach this problem. In this case, a pixel-wise subtraction of the red components of both

image is implemented. An example of the result of this stage is showed in figure 3.26.

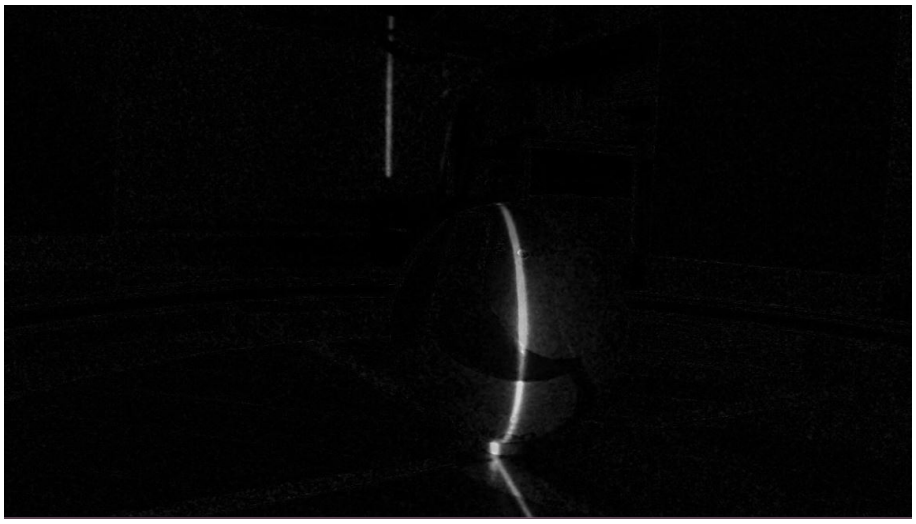

FIGURE 3.26: Red component image subtraction

4. **Red channel Threholding:** Initially an Otsu's threshold operation was implemented but periodically, a simple threshold was tuned manually with many examples to get better results, the final threshold was established in 14% of the maximum value of the image (a very flexible and low one). The result here is a single channel image in gray-scale instead of a binary image. An example of the result of this stage is showed in figure 3.27.

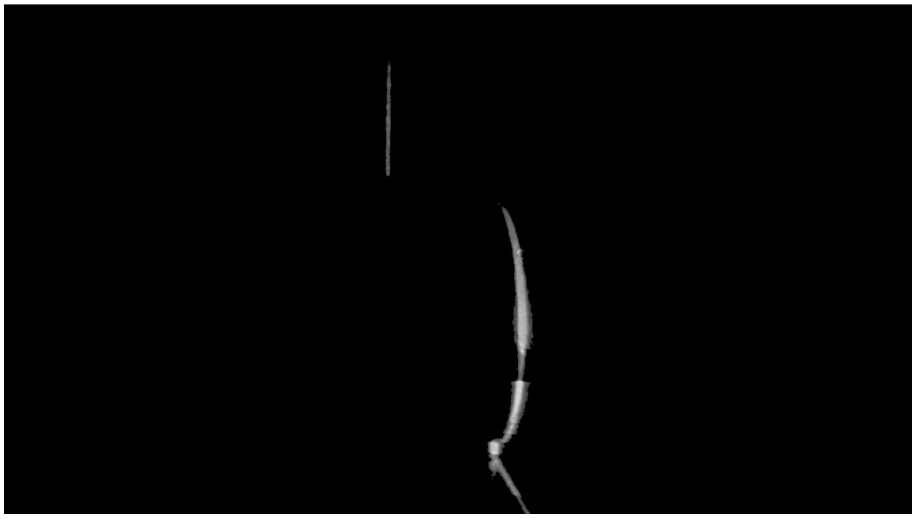

FIGURE 3.27: Red channel threshold implementation

5. **HSV Red color filtering:** In addition to the thresholding, a soft filter is applied to the image, this is a red range filter in the HSV color space (H:32-335, S:30-255, V:30-255). Because this type of filter also presents significant variations, it was design to be slightly permissive. It's objective more than detecting the laser's location, is to establish regions in the image where it is not possible to find the laser. Often allowing much more colors than just simple red. An example

of a resulting mask for an image in this stage is showed in figure 3.28 where white means "pixel allowed". This stage may not cause any modification in the image, reason why it should only be considered as an additional measurement to reduce possible noisy pixel detection product of light variations between the two images subtracted on the first steps.

FIGURE 3.28: HSV filter (mask)

6. **Contrast increase:** A contrast and/or brightness change is a linear transformation to the intensity of each pixel, with the form of $y = ax + b$ where y is the new value for each pixel, x the original value and a the . For this stage, the transformation applied is $y = 1.2x$. This helps separate each peak from the background value and helps the parabola approximations described later in this section. The appearance of this transformation will be the same image but brighter for the most part. An example of the result of this stage is showed in figure 3.29.

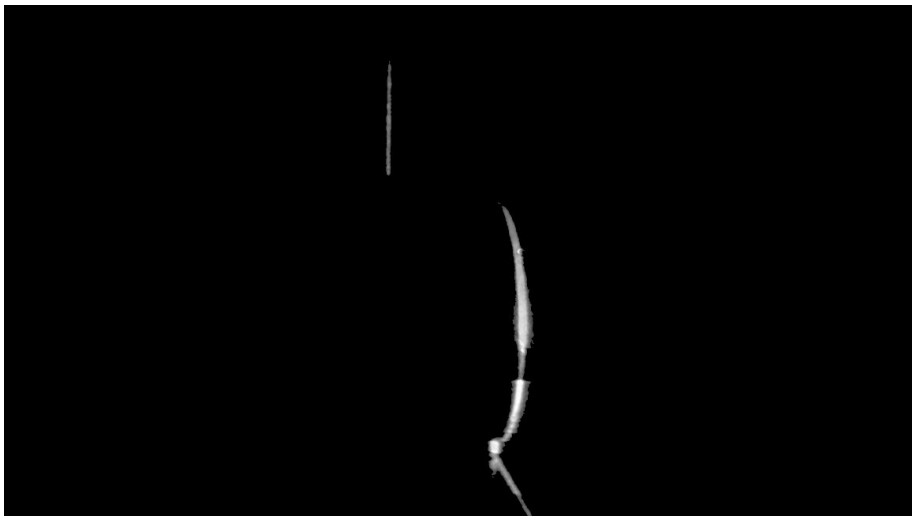

FIGURE 3.29: Contrast linear transformation y=1.2x

7. **Morphological operations:** Two morphological operations are used. First, a *Closing* operation with a vertical (8x6) rectangular structuring element is applied, helping get a more homogeneous laser line and removing small black holes through the laser section. Next, an *Opening* operation with a horizontal (6x8) rectangular structuring element was applied to the closed image, removing noisy isolated white points across the image that for the most part are not related to the laser area. The chosen structuring elements were determined in part by trial and error and taking into account the type of shapes that were supposed to be kept (vertical lines) and to be removed (horizontal lines). An example of the result of this stage is showed in figure 3.30.

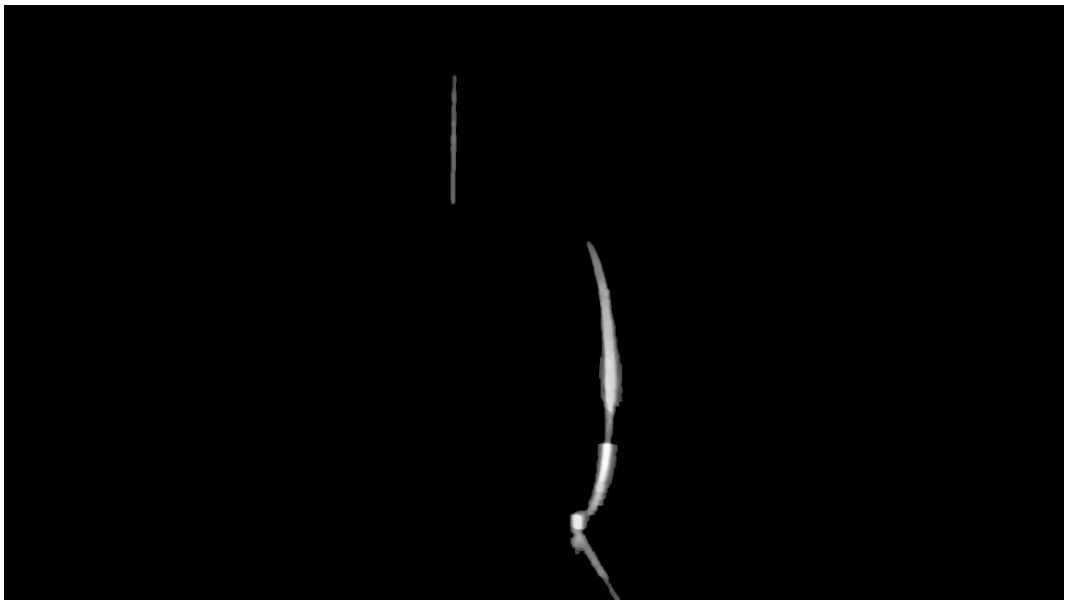

FIGURE 3.30: Resulting image from morphological operations.

8. **Maximum values extraction:** A very simple way to summarize the information from the laser (tens of pixels wide) would be to select the maximum value for each row. Although being an acceptable first approximation, it does not provide good accuracy for the center of the laser unless a single pixel peak was to be found, which is not an assumption one should make. However, gathering the information for the maximum of each row is an important task before obtaining more accurate results. The outcome from this stage is the maximum from each row where the laser was detected. Most of the times it was found that there are more than one "peak-pixels" with the same exact brightness value. For this reason, two maximums where found for each row, one searching from left to right and the other one from right to left. A generated image to illustrate this stage is showed in figure 3.31. Please note that only one maximum is painted.

9. **Sub-pixel laser coordinates extraction:** With access to the maximums for each row of the laser, two cases can be found, the most infrequent one is when there is only a single peak in the row (the two maximums are exactly the same point), and the most common one is when there are two different maximum values at two different columns in the image.
   Work done by Molder and their team [26] showed that fitting this maximum

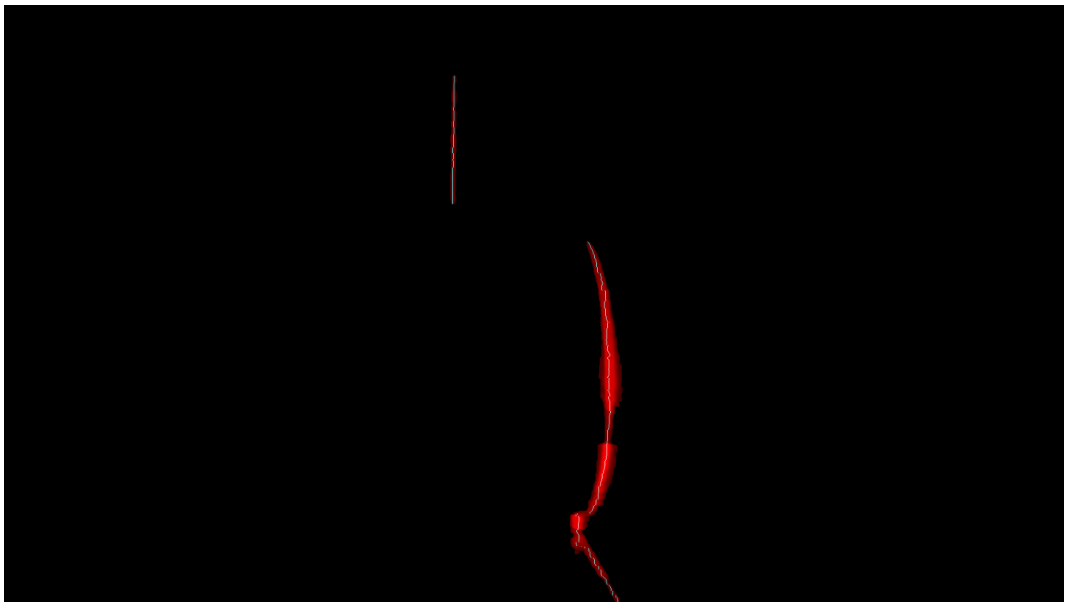

FIGURE 3.31: Image after morphology operations with coloured maximums (cyan) and wide laser-line (red).

to a parabola showed increased overall accuracy in the laser line detection, so this approach was initially implemented. Producing a floating point precision maximum location, which could be called as with *sub-pixel accuracy*. This approach is fairly better, and works perfectly for the first case we mentioned in the paragraph above, but not for the second case, where only one maximum pixel would be taken into account and the detected center even with the maximum of the parabola would be slightly but clearly shifted to that side.
Having stated this problem, an even more complete solution is proposed. A double parabola fitting for the second case. Here, a parabola is fitted for each of the two maximums found, then the maximums of each parabola are computed, and the center point between these maximums (found as a mean) is declared as a much more precise approximation of the center. Leaving the single parabola fitting solution for the first case only.
The implementation of the parabola fitting was tested on two ways, the first one, is by solving with 3 points for a set of 3 equations of the type $y = Ax^2 + Bx + C$. And the second one with much more summarized equations, is through the calculation of a Lagrange Polynomial, which takes an N number of points and generates the analytic equation of order $N - 1$ that best fits these points. This last one was used in the implementation of this stage because it allows scalability to higher polynomial orders (in future improvements), has fewer operations per fitting to be performed, and an implementation for python in the *Scypy* library is already available. For each parabola calculated (each row) the data used considers the index of the maximum value and it's two closest neighboring points. In figure 3.32 the reader can observe an overall of the center approximation process, by visualizing the fitted parabolas for each maximum and its mean value. This graph is for a single row of a single image. On the other hand, figure 3.33 shows all the calculated sub-pixel resolution points in comparison to their pixel resolution maximums.

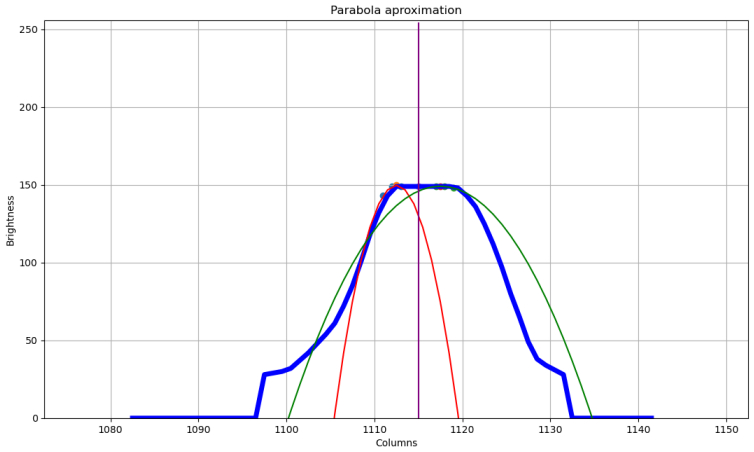

FIGURE 3.32: Double parabola center approximation for a single row pre-processed image.

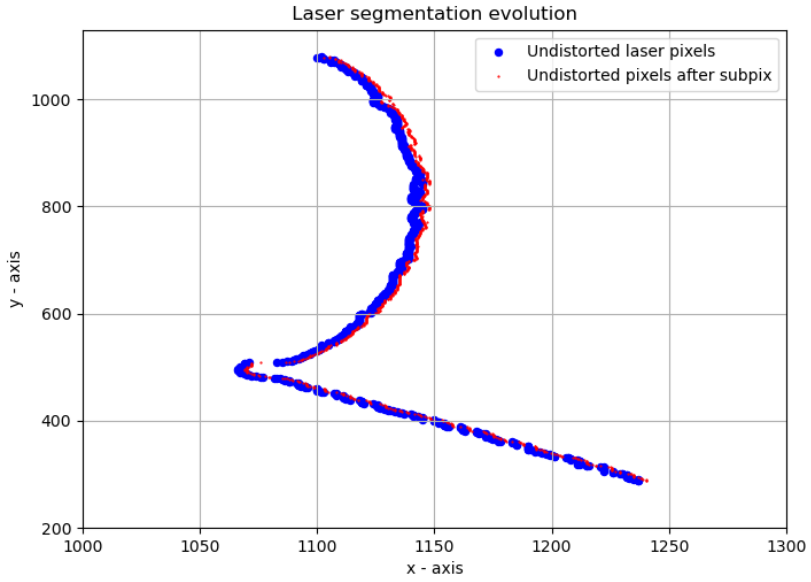

FIGURE 3.33: Comparative from pixel resolution indexes to sub-pixel resolution indexes.

3.3.6 Scanner calibration

The scanner calibration is the process that allows the whole system to convert 2D image coordinates (row,column) to 3D object coordinates (X,Y,Z). This can be seen as the whole setup metric calibration, relating pixel units from an image, to world units (e.g millimeters, inches, etc). One can find many research papers from the last decade on different techniques for this procedure, some involving very precisely constructed mechanical setups and thus reconstructing points from arithmetic modeling, and others, requiring complex calibration objects or reference points inside the scene (needed at all times during the scanning process).

Theoretical background

Two main principles are applied for the extraction of 2D-3D correspondences of the laser plane. These are the complete quadrangle of a trapezoid and the cross-ratio of a pencil of lines.

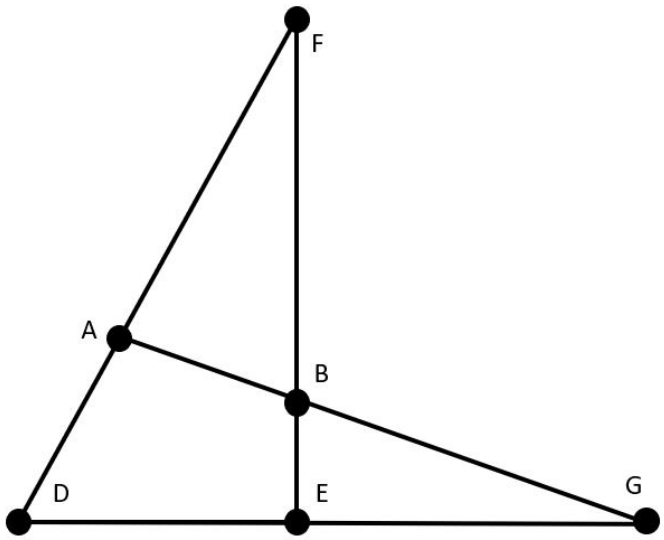

FIGURE 3.34: Complete quadrangle from a trapezoid.

Figure 3.34 illustrates this principle. Given a trapezoid of coordinates A,B,E,D, the complete quadrangle points G and F can be calculated by intersecting the line pairs A-B, D-E and D-A, E-B respectively. The only assumption it is made here is that all the points lay in the same plane.

Figure 3.35 on the other hand, shows the basis behind the cross-ratio of a pencil of lines in a 1D space. Given a center of projection point P, and whichever pair of lines S, r. One can project another line with origin in P that crosses each one of those lines, creating a respective set of points X and their projection on line S, X'. Now, it can be demonstrated that a relation between four points in line r, let say A', B', C', D' will keep a relation fixed with their unique projected points A, B, C, D. This relation is called the cross ratio and can be calculated using equation 3.11 or 3.12. With XY being the euclidean distance between points X and Y and $*$ being a scalar product.

$$CrA, B, C, D = \frac{AC * BD}{AD * BC} \tag{3.11}$$

or having $\lambda_2 = AB$,$\lambda_1 = AC$ and $\lambda_3 = AD$:

$$CrA, B, C, D = \frac{\lambda_2(\lambda_3 - \lambda_1)}{\lambda_3(\lambda_2 - \lambda_1)} \tag{3.12}$$

Also, by having 3 of those 4 points and their cross-ratio, one can calculate by clearing let's say λ_1 as:

$$\lambda_1 = \frac{(Cr - 1)\lambda_2\lambda_3}{Cr\lambda_3 - \lambda_2} \tag{3.13}$$

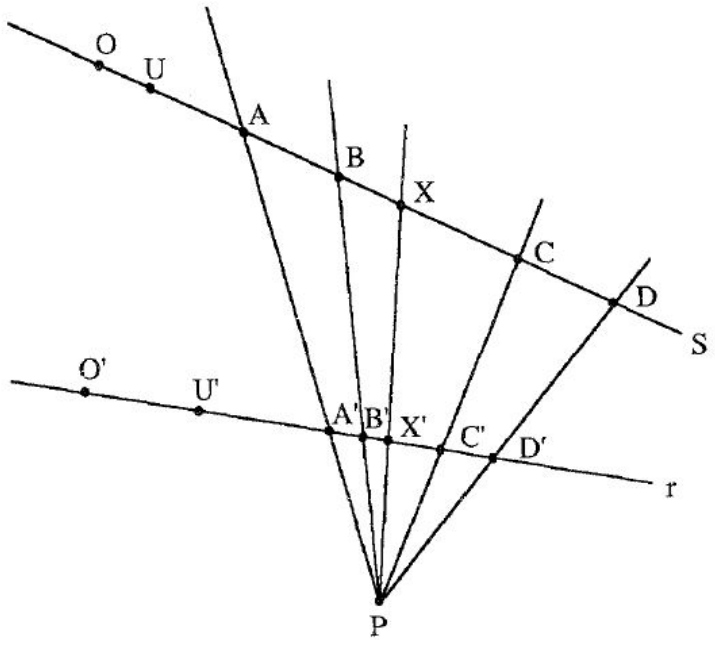

FIGURE 3.35: Cross-ratio of a pencil of lines.

Scanner calibration process

In this project, the whole scanner calibration process is detailed in figure 3.36. In a nutshell, this process requires obtaining pictures of a known-size planar object (a chessboard pattern) with the laser line visible in it. Both the image coordinates of the chessboard corners and the laser are recorded. Using the first ones and by knowing the size of the chessboard pattern, the pose in space respect to the camera is estimated (a transformation between our arbitrary world coordinate system and the camera's coordinate system). Parallel to this, a trapezoid made out of 4 non-collinear points is derived from the chessboard, its complete quadrangle coordinates are calculated, and by using line intersections and the cross-ratio of a pencil of lines, the projection of the laser points is estimated in an arbitrary world coordinate system.
Using the same transformation from pose estimation, these laser projections are translated to the cameras coordinate system. These 2D-3D point correspondences are acquired for different depths respect to the camera by locating the chessboards in different postures. Then an over-defined system is proposed with this correspondences and its minimum squared error solution is used in later sections for 3D coordinates estimation. The way this model is calculated is completely independent from the physical montage, meaning no information is needed respect to the position of the laser, nor the camera. No measurements of the angles are needed and no relative orientation or location between the two is necessary as longs as it remains fixed for both the calibration and future scans. For the remaining of the section, because a pin-hole camera model is required for this type of calibration, all images unless specified have been passed trough an "undistorting" step before being used here.

The main steps exposed in the diagram from figure 3.36 are detailed as follows:

1. **Images data set:** For the scanner calibration, a total of 106 image pairs are

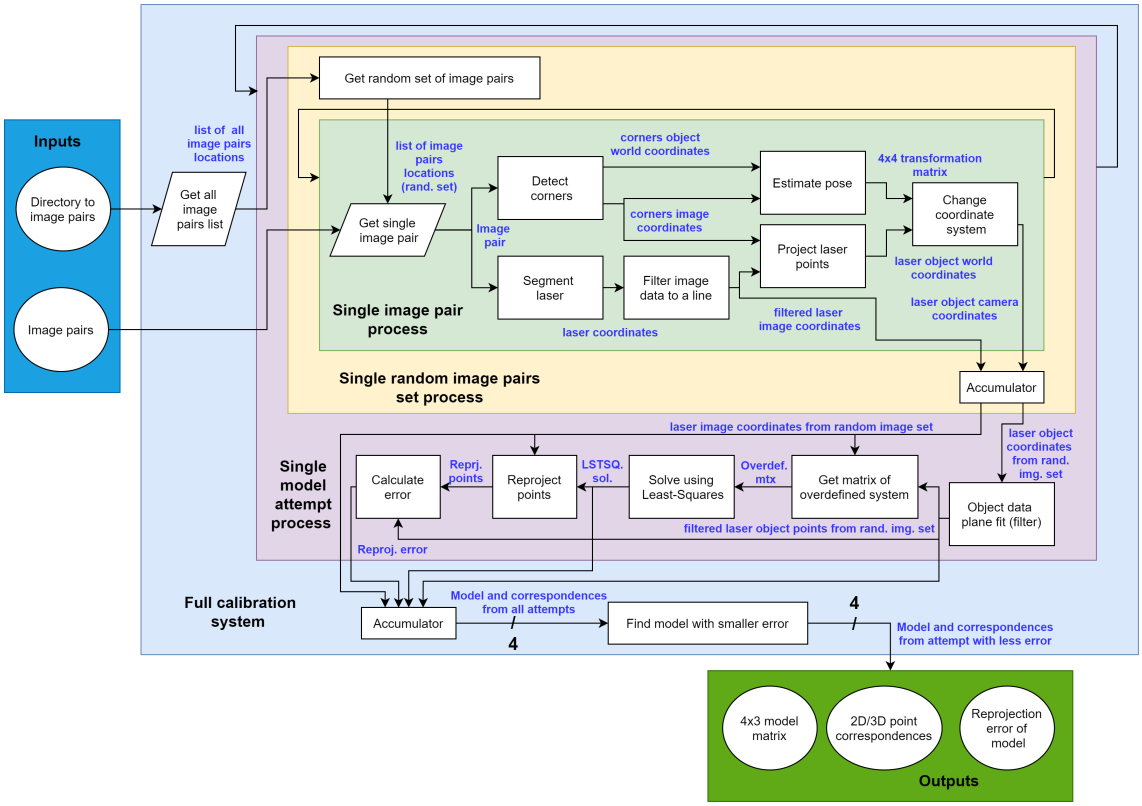

FIGURE 3.36: Scanner calibration process

used. Each single image pair has a "lasered" and a "laser-less" image of the chessboard pattern. In addition, all image pairs must cover a wide variety of depths to the camera. Here, radial and tangential distortion has been removed from all image pairs for the scanner calibration. An example of image pair is showed in figure 3.37.

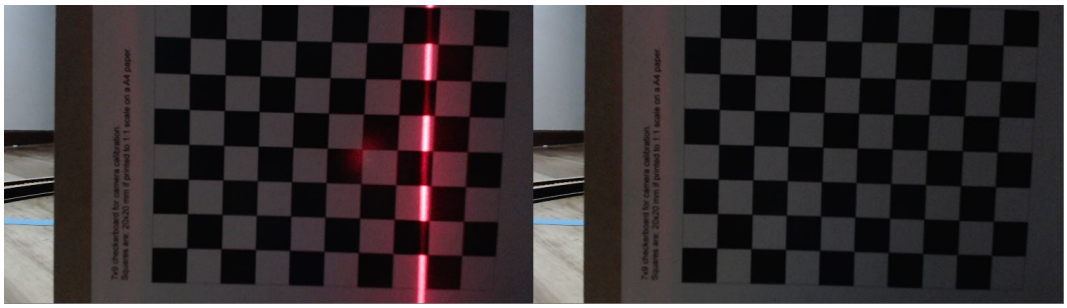

FIGURE 3.37: Single image pair

2. **Detect corners:** In this step, the image coordinates of the corners from the chessboard pattern are recorded using the "laser-less" image. The reader can refer to the *"Camera calibration process with OpenCv"* subsection to know more about the corner detection process. An example of the detected corners is illustrated in figure 3.38. Please note that although all corners are showed in the "lasered" image, in reality, these are calculated from the "laser-less" image to avoid detection problems due to the laser line. This supposes that there is

no movement between the two images. Change in light conditions should not affect this step.

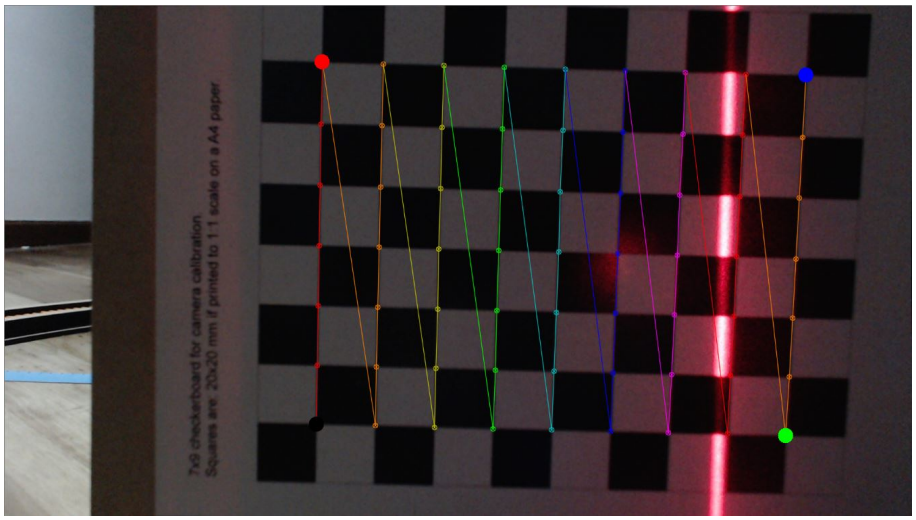

FIGURE 3.38: Extracted chessboard corners.

3. **Segment laser:** Using the algorithm explained in subsection *"Laser segmentation"*, the image coordinates of the laser line are recorded with a sub-pixel resolution. From this process one can summarize the image processing step and the calculation of the center laser in figure 3.39.

FIGURE 3.39: Image processing plus center calculation from single image pair.

4. **Filter image data to a line:** Because it is presumed that the chessboard pattern is flat, without losing generality, the recorded laser images are fitted to a line to reduce any possible noise from the laser segmentation algorithm. The resulting data of this step is calculated as 95% of the exact point in the fitted line and 5% of the original value of the data.
5. **Project laser points:** Please refer to figure 3.41 for better understanding in this step. From the recorded corners, 4 non-collinear points are selected and an

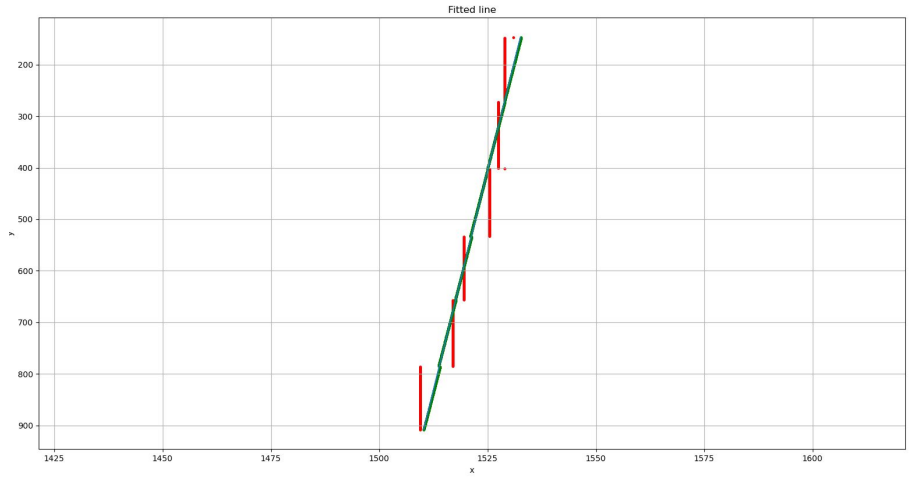

FIGURE 3.40: Fitted line to single image pair segmented laser coordinates. Original data(red), fitted data(green)

imaginary trapezoid is derived with **a**,**b**,**e**,**d** image coordinates and **A**,**B**,**E**,**D** world coordinates.
Then, the complete quadrangle coordinates are calculated for both the image coordinates as well as for the established arbitrary world coordinates of the trapezoid. In the image plane, **f** point is calculated from intersection of lines **a-d** and **b-e**, and **g** point is calculated from lines **a-b**, **d-e**. The same process can be applied to points in the world coordinate system as all its 4 corners are known. To this point, all the world and image coordinates of the four corners of the trapezoid and the complete quadrangles are known.
Now, the intersections of the laser with the trapezoid (**h** and **j**) are found in the image, as an evaluation process of which point is the closest to the polygon's lines. Now, to get their corresponding locations in the world coordinate system, one can see that taking **O** as reference, lines **a-A**, **h-H**, **b-B**, and **g-G** form a pencil of lines. Because 7 out of the 8 values are known, the missing one can be calculated using equations 3.12 and 3.13 (cross-ratio of a pencil of lines). The same for the pencil of lines **d-D**, **j-J**, **e-E**, **g-G**.

Next step, is taking a random amount of points **n** from the left line of the trapezoid in the image plane (**a-d**) and projecting them into the world coordinate system as before, now using the pencil **d-D**, **n-N**, **a-A**, **f-F**.
Finally, the laser plane correspondences of this points can be found as intersections between the lines **n-g**, **h-j** for the image plane, and **N-G**, **H-J** for the object plane. From this step, a series of 2D-3D point correspondences in the arbitrary world coordinate system are found.

6. **Estimate pose:** Using an iterative method based on Levenberg-Marquardt optimization, A 4x4 transformation matrix is estimated such that the points in the arbitrary world coordinate frame can be translated to the camera's coordinate frame. This is completely implemented in OpenCv as *cv2.solvePnP()*. Note that the arbitrary world coordinate frame depends strictly on the chessboard size. For our project, the upper left corner of the chessboard was taken as the (0,0,0) point and each square had a side of 20mm.

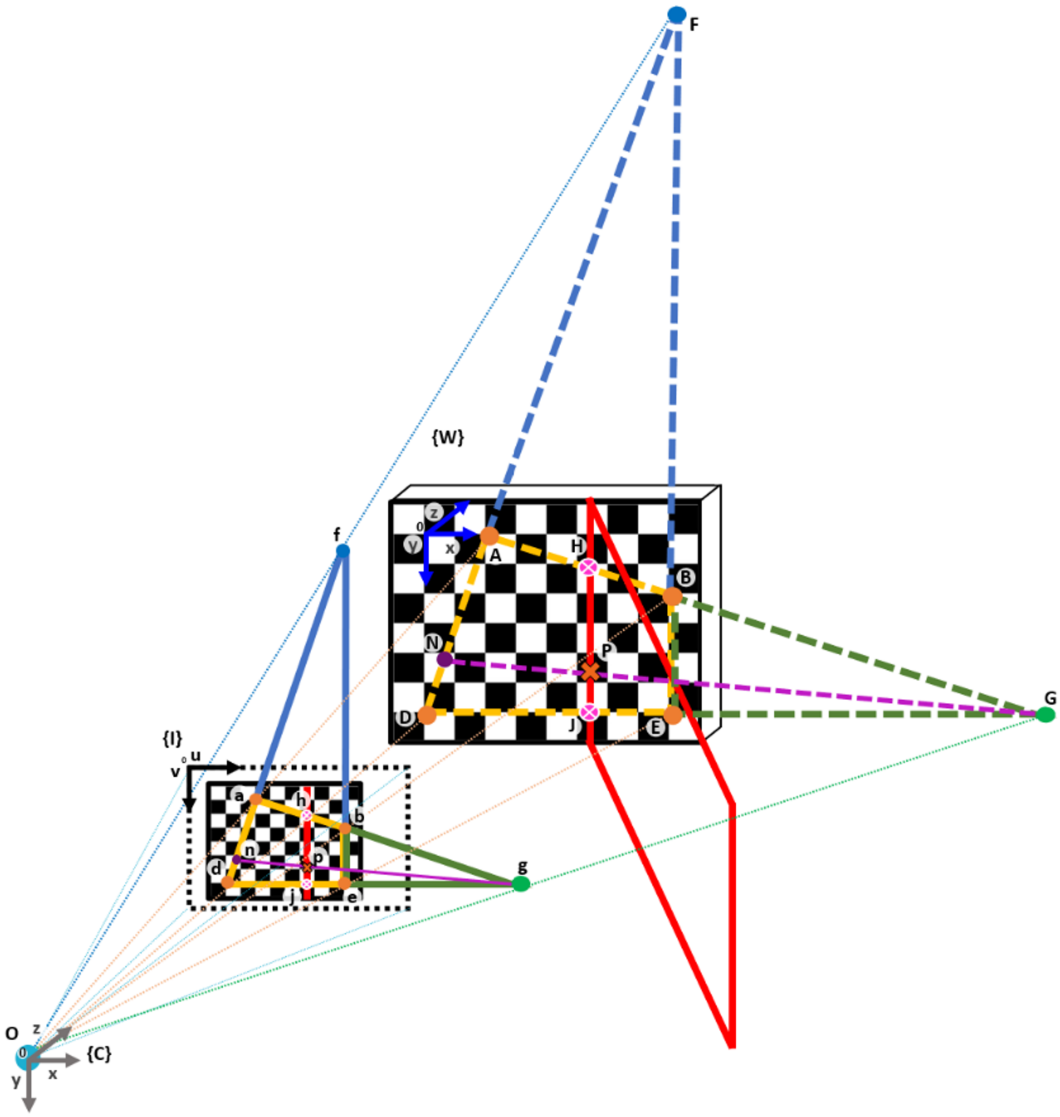

FIGURE 3.41: Projective model used in scanner calibration

7. **Change coordinate system:** Using the transformation matrix calculated in the last step, both the projected laser points and the corners (just for reference) where translated for each image. The resulting points although correctly situated in a reasonable camera's coordinate frame, where inverted respect to the *centroid* of the corners through the X axis. This in contrast with the locations one could, by eye, observe in real life. Reason why a reflection across the X coordinate of the *centroid* of the chessboard was performed. Finally, 2D-3D correspondences have been successfully obtained. Figure 3.42 shows the 2D points, and figures 3.43 their 3D respective projections in the cameras coordinate frame as seen by a person in real world.

8. **Fit plane to object data:** Although it may be some accumulated error through all the steps, it is clear that in theory, all projected laser points when located in the camera coordinate system correctly, should all be aligned in a perfect plane. Reason why all the object points are fitted to a plane. As showed in figure 3.44. Resulting points are fitted 95% a plane, maintaining 5% of their

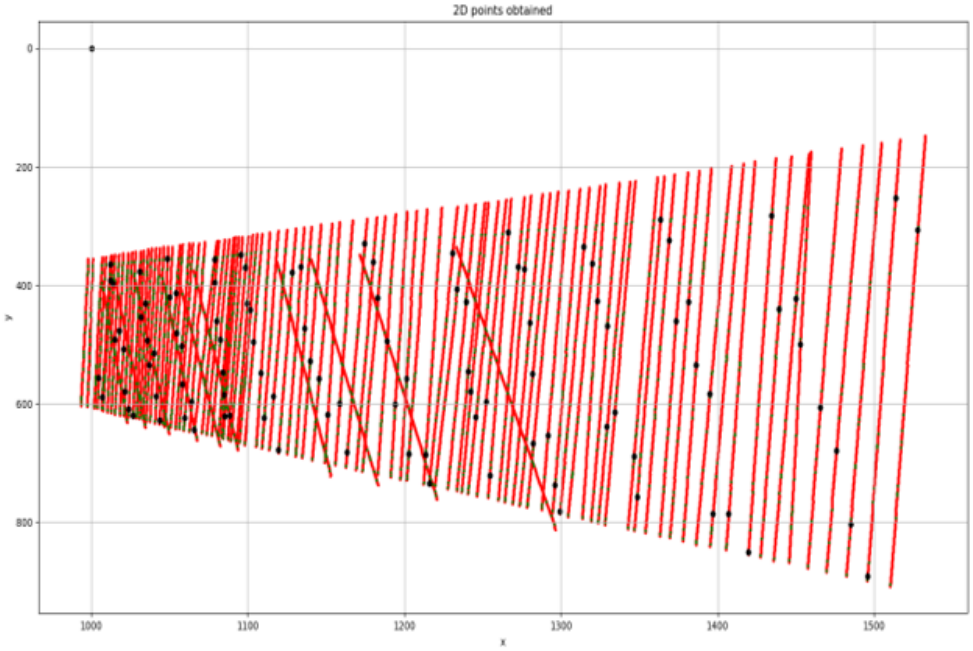

FIGURE 3.42: Laser image points after "Change coordinate system" step.

deviation.

9. **Get matrix of over-defined system:** The transformation that we are ultimately after in the scanner calibration is a 3x2 matrix (ideally) receiving 2 image coordinates and returning 3 object coordinates in the adequate coordinate system. However, for this problem, homogeneous coordinates are required, reason why a 4x3 matrix is the objective as showed in equation 3.14.
Each of the parameters in this matrix needs to be calculated or "tuned", in our case using the 2D-3D correspondences we have estimated. In addition, the system just mentioned should be in the form $A * x = b$, avoiding B to be a complete zeros vector in to get rid of the trivial zeros solution. The equations system expressed in matrix formed can be observed in equation 3.15. And knowing coefficient t_{43} can be established as 1, the solvable system looks as in equation 3.16. Take into account that each single correspondence point gives out 3 equations, reason why 4 "perfect" points should be enough to completely define our rank 12 system. Nevertheless because there exists some noise in every correspondence, our best approach to solve this problem is by generating much more than 12 equations. Expecting to solve for the coefficients vector x by minimizing some error/cost function such as the squared error.

$$\begin{bmatrix} sX \\ sY \\ sZ \\ s \end{bmatrix} = \begin{bmatrix} t_{11} & t_{12} & t_{13} \\ t_{21} & t_{22} & t_{23} \\ t_{31} & t_{32} & t_{33} \\ t_{41} & t_{42} & t_{43} \end{bmatrix} * \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \tag{3.14}$$

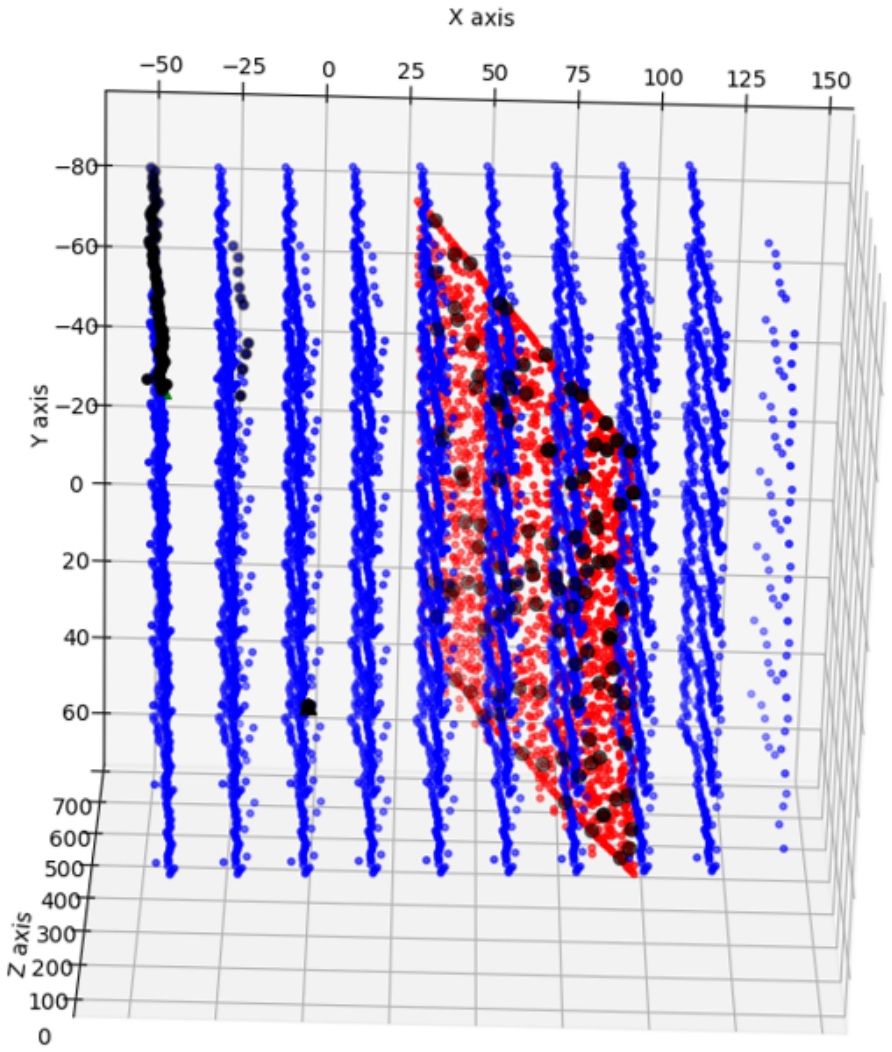

FIGURE 3.43: Laser projected and transformed object points after the "Change coordinate system" step (red) and respective corners of chessboards(blue)

$$\begin{bmatrix} \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ u_i & v_i & 1 & 0 & 0 & 0 & 0 & 0 & 0 & -u_iX_i & -v_iX_i & -X_i \\ 0 & 0 & 0 & u_i & v_i & 1 & 0 & 0 & 0 & -u_iY_i & -v_iY_i & -Y_i \\ 0 & 0 & 0 & 0 & 0 & 0 & u_i & v_i & 1 & -u_iZ_i & -v_iZ_i & -Z_i \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \end{bmatrix} * \begin{bmatrix} t_{11} \\ t_{12} \\ t_{13} \\ t_{21} \\ t_{22} \\ t_{23} \\ t_{31} \\ t_{32} \\ t_{33} \\ t_{41} \\ t_{42} \\ t_{43} \end{bmatrix} = \begin{bmatrix} \dots \\ 0 \\ 0 \\ 0 \\ \dots \end{bmatrix} \tag{3.15}$$

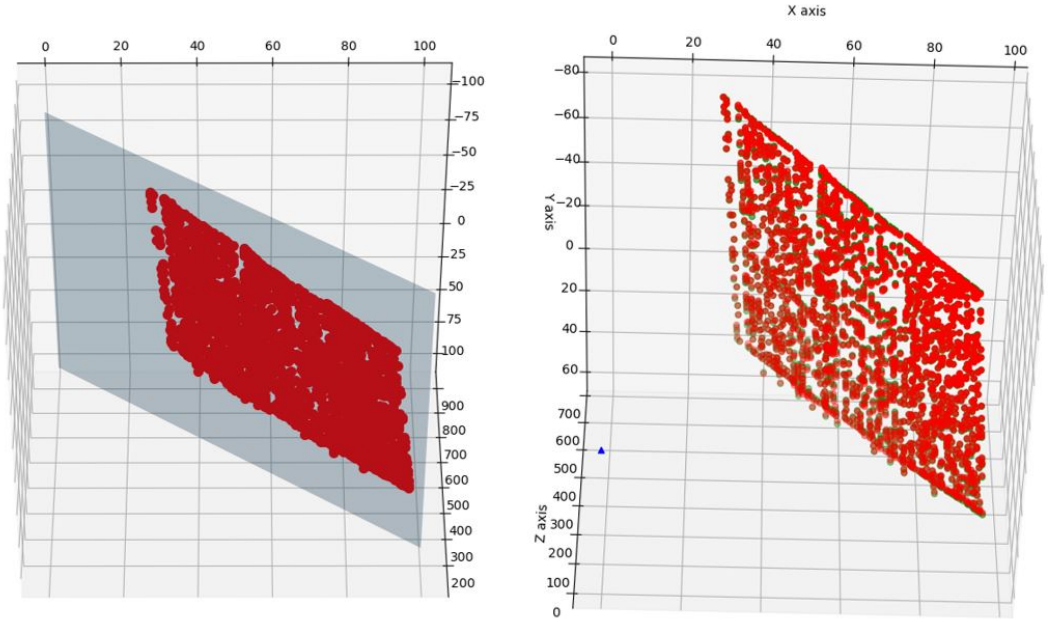

FIGURE 3.44: Plane fitted to 3D laser points.

$$\begin{bmatrix} \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ u_i & v_i & 1 & 0 & 0 & 0 & 0 & 0 & 0 & -u_iX_i & -v_iX_i \\ 0 & 0 & 0 & u_i & v_i & 1 & 0 & 0 & 0 & -u_iY_i & -v_iY_i \\ 0 & 0 & 0 & 0 & 0 & 0 & u_i & v_i & 1 & -u_iZ_i & -v_iZ_i \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \end{bmatrix} * \begin{bmatrix} t_{11} \\ t_{12} \\ t_{13} \\ t_{21} \\ t_{22} \\ t_{23} \\ t_{31} \\ t_{32} \\ t_{33} \\ t_{41} \\ t_{42} \end{bmatrix} = \begin{bmatrix} \cdots \\ X_i \\ Y_i \\ Z_i \\ \cdots \end{bmatrix} \quad (3.16)$$

10. **Solve over-defined system and calculate error:** With the raised system, *Numpy* least-squares solver is used to estimate the coefficients vector by minimizing the squared error function. The solution is then used to re-project the used 2D correspondences and compare its value to the original 3D correspondence values. Next, the average error in X,Y and Z coordinates is calculated. The re-projected points using all the correspondence points is showed in figure 3.45. The respective errors using the full set of 106 calibration image pairs is $Error = [X, Y, Z] = [0.398623330.381165492.91289025]mm$.

11. **Minimize reprojection error:** Although is it highly recommendable to use a large number of calibration images (and correspondence points) to improve the results, it does not necessarily mean the more points the better results. Here, quality of the points takes precedence before quantity. To improve this aspect, an iterative process of randomly selecting sets of image pairs and calculating the solution with those correspondences is implemented. 300 iterations were rehearsed selecting 25 different calibration image pairs each time.

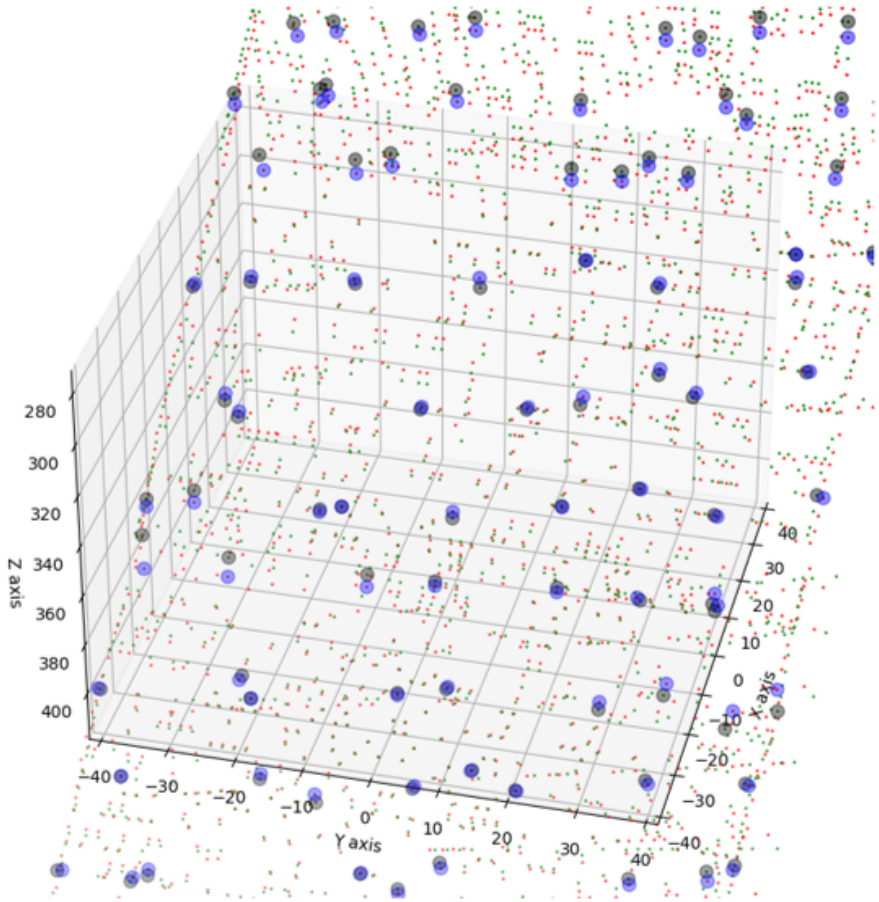

FIGURE 3.45: Re-projected points. Originals (green/black), Re-projected (red/blue)

After estimating the matrix parameters and calculating the error, only the coefficients obtained in the iteration with the smallest error magnitude (as the L2 norm) is ultimately used.

The results are summarized as follows:

$$Mtx_{4x3} = \begin{bmatrix} -1.72593161 * 10^{-}1 & -7.38268002 * 10^{-}3 & 1.64347601 * 10^{2} \\ 5.93355454 * 10^{-}3 & -1.73152785 * 10^{-}1 & 8.55596171 * 10^{1} \\ 1.75047107 * 10^{-}2 & 3.71031233 * 10^{-}3 & -2.62764483 * 10^{2} \\ -1.37392065 * 10^{-}3 & 2.04547513 * 10^{-}6 & 1 \end{bmatrix} \tag{3.17}$$

$$Error_{xyz} = [0.203916690.293095841.48587243] ||Error_{xyz}|| = 1.528170167296689 \tag{3.18}$$

3.3.7 Points cloud extraction

Figure 3.46 illustrates the followed step by step process for acquiring a point cloud. Its inputs are the already segmented laser coordinates with sub-pixel resolution, and

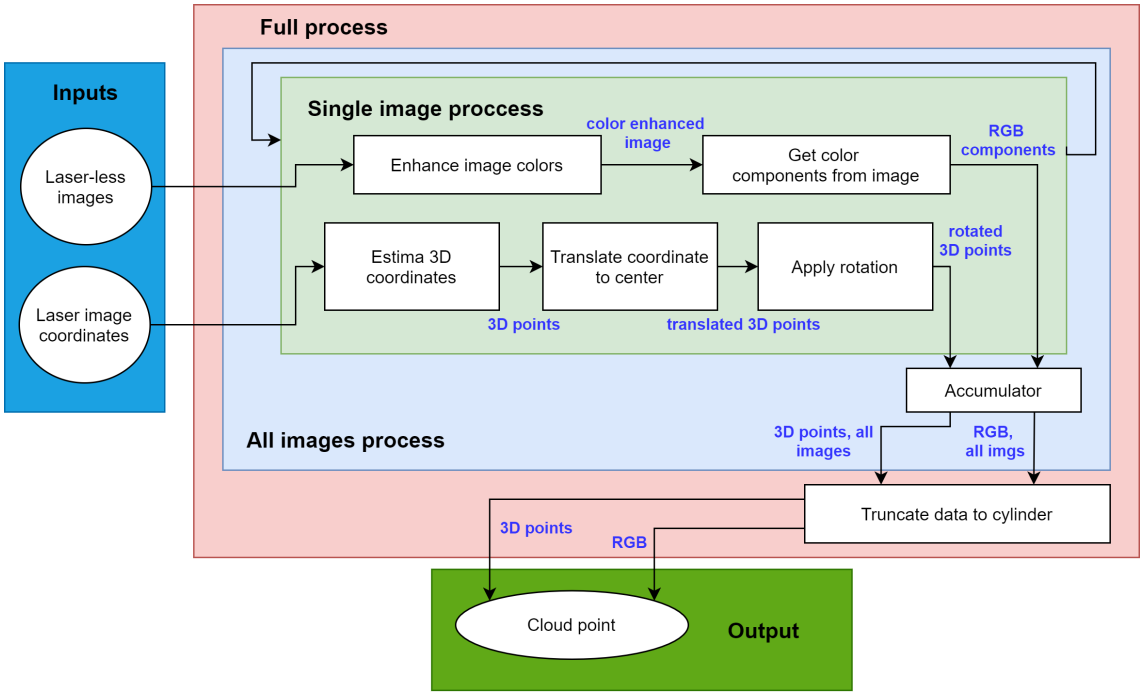

FIGURE 3.46: Blocks diagram of the point cloud extraction software/process.

their respective laser-less images (for voxels color extraction).

- **Enhance image colors:** One of the most perceivable downsides of laser triangulation scanning with a common red coloured laser, is its dependence on low ambient light intensity throughout the data acquisition process.
  In other words, there is a trade-off between how well the laser can be detected and how well illuminated is the environment. Imagine a totally closed room with no sources of light other than the laser emitter itself, this would make the software detect the laser coordinates with the highest precision and missing the fewest points, but also, all vertices would have a black color, and the opposite when having a well illuminated object. Although it is not one of the main objectives of this project to acquire point clouds with high color fidelity, it just didn't seemed right to acquire just black or color-less point. Reason why, this step aims at enhancing the overall colors in each image when low light is used. For this purpose, the BGR image is converted to the LAB or CIELAB color space ("L" for lightness, "a" for green-red, and "b" for blue-yellow).This color space in few words is designed to approximate human vision, specially the L component which closely matches human perception of lightness. Then, an adaptive histogram equalization with clip size of 9.0 and tiles of 4x4 is used through the L component of the image. Finally, the image is then reconverted to its BGR components. A sample result of this process is illustrated in figure 3.47.

  Note: Take into account that this step is totally optional and that because one has access to the original images, one could certainly recover the original colors.

- **Get color components from image:** Having both the laser coordinates and the filtered laser-less image from the last step, an array of the colors of this coordinates is stored.

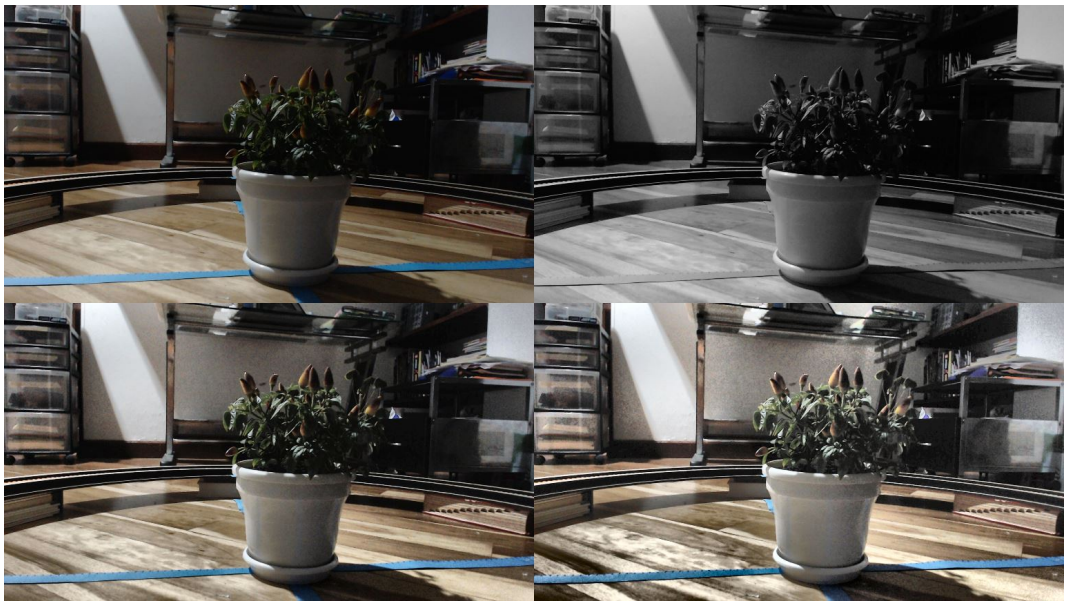

FIGURE 3.47: Color enhance example Original (left top), L component (right top), After equalization with clip size=2.0 (left bot), After equalization with clip size=9.0 (right bot).

- **Estimate 3D coordinates:** All the hard work for this step has already been done in the scanner calibration steps. Here, one simply uses the obtained 4x3 coefficients matrix and applies is to all the segmented laser points of a given single image. Obtaining a set of 3D points of the laser line in the cameras coordinate system. The obtained 3D points of a single image are illustrated in figure 3.48.

- **Translate coordinates to center:** The location of the estimated 3D coordinates are respect to the camera's position, meaning the origin point [0,0,0], is in fact the camera location. To join all the images together into a single point cloud, and because we know the nature of the acquisition setup, a change in the center is performed. The value of this center of course needs to be calibrated. For this purpose, an image was taken with the camera having the laser emitter on and with no other object than the platform itself. The image is visualized and the mouse position is recorded in image coordinates (pixels), as shown in figure 3.49.
  The manually selected coordinate of the center has to, of course be illuminated by the laser line. Then, the 3D coordinate of this point is estimated using the already calibrated 4x3 coefficient matrix. Finally, all the extracted 3D points from each single image pair is then translated to this center point.

- **Apply rotation:** With the translated coordinates to a new [0,0,0] origin, a rotation is applied to the points of each image, and the result is attached to the complete point cloud. The angle of rotation is calculated depending on the amount of image pairs taken, for example, if 360 image pairs are taken, then the angle of rotation would be 1°(counted clockwise if seen from a top view. Reviewing the coordinate frame of the camera from figure 3.41, the rotation is applied across the Y axis.

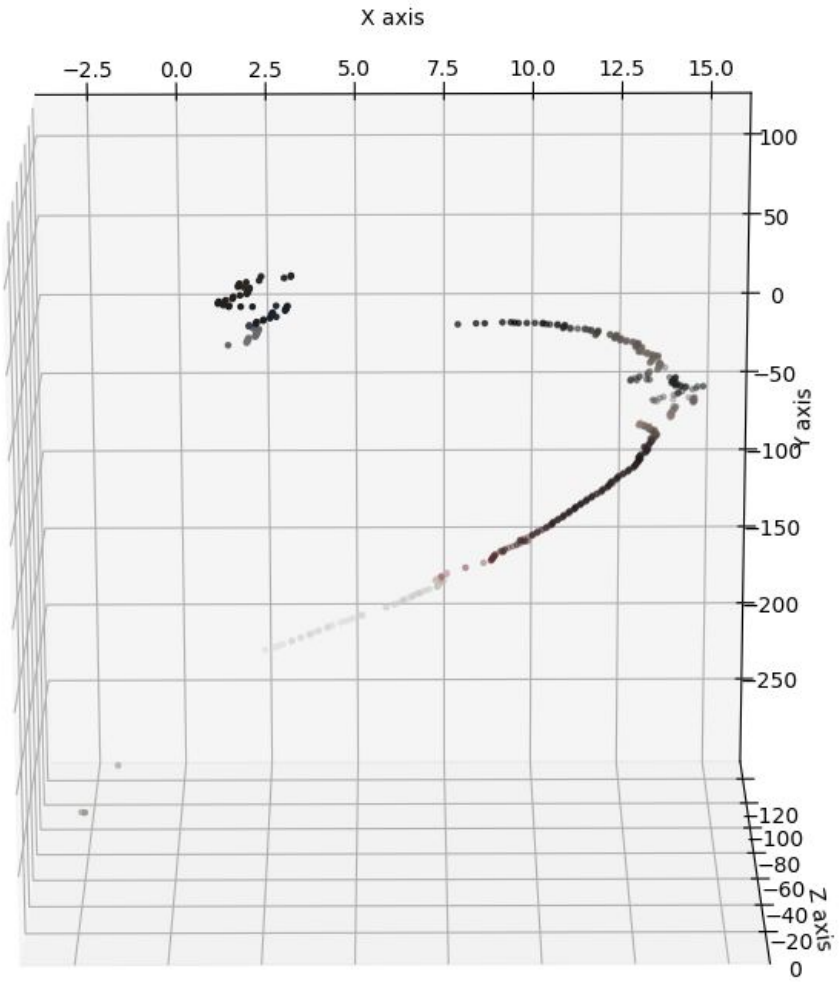

FIGURE 3.48: Estimated 3D points from a single image pair

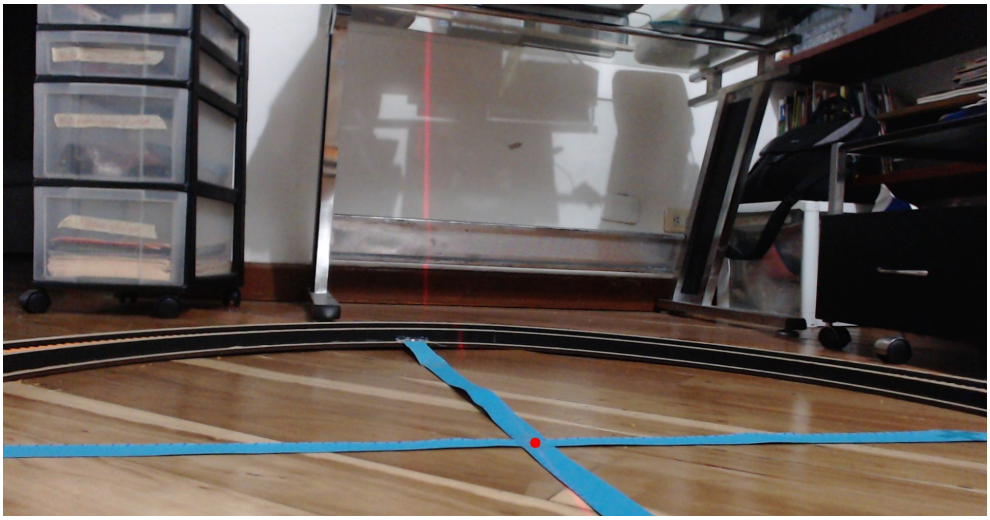

FIGURE 3.49: Calibrated center position with single image

- **Truncate data to cylinder:** Occasionally, some irrelevant points are detected by the laser segmentation algorithm, lets say another object outside the acquisition setup that was hit by the laser. In this case, the obtained cloud points would have some irrelevant points at further distances from the rest of the point cloud. To make up for this issue, just the points inside a fixed-size cylinder are accepted as part of the scan. This can be considered as the first filter applied to the point cloud.

3.3.8 3D points Visualization and processing

Once 3D coordinates have been successfully calculated, a software must be implemented to make this results visible and to turn the current point cloud into a mesh (meaning a solid surface). For this purpose a 3D object processing library was used, Open3D. Which allows to perform the following tasks on a series of 3D points:

1. **Visualize raw data with pixel RGB color:** A set of [X,Y,Z,R,G,B] points can be plotted in an interactive 3D environment (at dedicated GPU is highly recommended).

2. **Outliers removal:** Two methods are available to remove "noisy" points in the point cloud, these are the *statistical* and *radius-based* methods. The first one calculates the average distance between all vertices and eliminates points that are further away from a given number of neighbors. The second one, filters vertices by defining a sphere in space centered at each vertex, and eliminates all points that have less than a certain amount of neighbors within that sphere. Both algorithms showed to work correctly at eliminating undesired points product of noise in our laser segmentation algorithm.

3. **Estimate normal vector of pixels:** Using a K-NN (nearest neighbors) method, depending on their position in space, a normal vector can be estimated for each vertex. This step apart from providing information on the location of each point in space, is also required to generate a mesh.

4. **Calculate a bounding volume and a convex hull from point cloud:** Given a point cloud of any shape (susceptible to noise) it is possible to calculate the coordinates of an (equivalent to bounding box) axis-oriented or object oriented bounding volume. This allows to extract the width, height, and depth of the point cloud. Also, a convex hull can be estimated, and using it as an acceptable low-resolution approximation of the point cloud, the volume (in m^3) can be calculated.

5. **Calculate and visualize a mesh from point cloud:** Calculating a mesh implies creating a series of quadrilaterals and/or triangles that can represent the surface of the points cloud. To solve this problem, two methods are tested in our project, these are the *Pivoting ball* and *Poisson surface* methods. The first one defines a set of spheres with different radii that would "roll" on the surface of the point cloud, defining in its path each face. The second one, is more a mathematical approach to creating the surface, this one although inevitably smoothing up the surface, generates with more ease a much more homogeneous surface. However, with point clouds with high frequency changes, none of these showed accurate results.

6. **Export point cloud and mesh in various commonly used formats:** This opens the possibility of using a large amount of software for further processing. Formats available are xyz, xyzn, xyzrgb, pts, ply, pcd (for a point cloud) and ply, stl, obj, off, gltf/glb (for a mesh).

Most of this tools are used for acquiring the feature measurements from the point clouds, this process is explained in the next section. Also, most results that are showed in the **Results** section are visualized using this library.

3.4 Feature estimation

3.4.1 Point cloud cleaning/filtering

Before acquiring any measurements in the object, the point cloud needs to be completely clean. This means that although there is likely some overall error throughout the scan, any point too far from the object needs to be removed. Mainly, three steps for achieving this are implemented in the following order:

1. **A narrower cylinder for the ROI:** This process is entirely done by trial and error, and it depends on each scan, but it is the most deterministic way of getting rid of undesired noisy points. Only points which reside inside a defined-size cylinder are taken into account for the output point cloud, this also helps to split the object 3D points from the ground.

2. **Outliers removal:** Two methods for removing outliers are tested and a combination of both is used (in different proportions) for removing undesired points from the point cloud. These are an statistical and a neighbors amount-based outliers removal algorithms. The statistical way for removing outliers consists of calculating the standard deviation of a point and its **N** nearest neighbors. Then, only accept those points which are located within **n** times the calculated standard deviation. It's implementation is already in the Open3D library. The neighbors amount-based method consists of defining a sphere in space of radius **r**, and setting up a minimum number of neighbors that a point must have inside this sphere (with center in the evaluated point) in order to be accepted as part of the point cloud. Depending on the type of noise found for each point cloud, one or both of them are implemented across all points. An example of the results obtained in this step can be seen in figure 3.50. Some parameters across this implementations are manually tuned for each single points cloud.

An example of the differences of the point cloud before and after all these filtering steps is showed in figure 3.51.

3.4.2 Width, depth, and height estimation

After having a relatively clean point cloud, the process of acquiring the width, depth, and height values can take place. By calculating an axis oriented bounding box for the point cloud, all this values can be estimated. The implementation provided by Open3D is used for this purpose. An example of this is showed in figure 3.52. With the calculated bounding box, the width, depth, and height are directly taken as the difference between two corners of the same line.

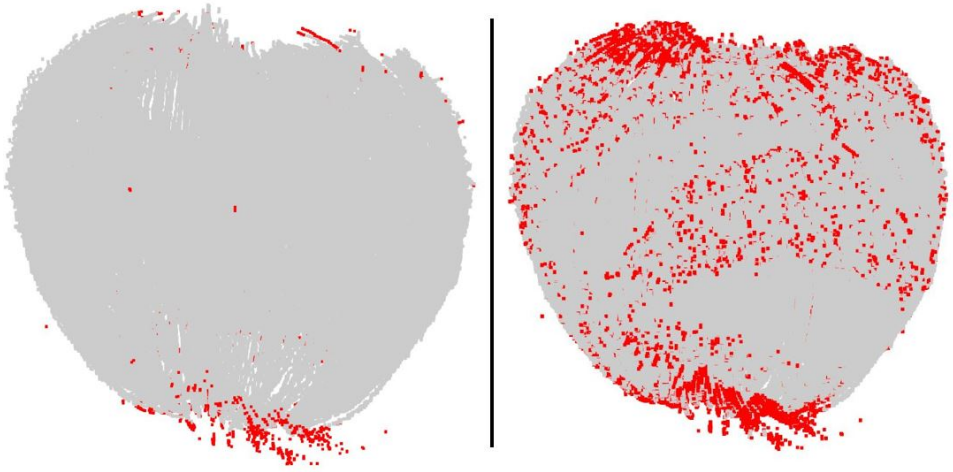

FIGURE 3.50: Outliers removal results inliers(gray), outliers(red). Mild filter (left), Aggressive filter (right)

FIGURE 3.51: Point cloud cleaning/filtering results. Original(left), Filtered(right)

3.4.3 Volume/Visible mass estimation

For acquiring the volume of a given point cloud, the convex hull of the object is calculated. A convex hull is a mesh calculated from the least amount of points that can enclose a given point cloud. Because a convex hull of an object will always be a waterproof and watertight mesh, its volume can be calculated as proposed by C. Zhang and T. Chen, implementing a sum of tetrahedrons[27]. An example of the calculated convex hull of a point cloud is illustrated in figure 3.53.

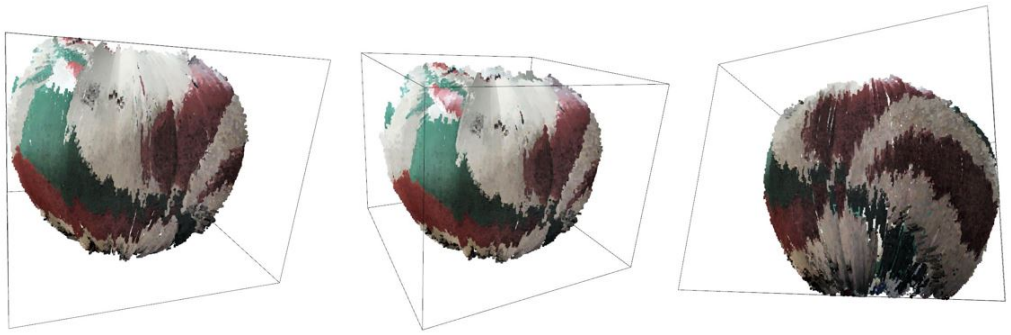

FIGURE 3.52: Bounding box of a point cloud from different perspectives.

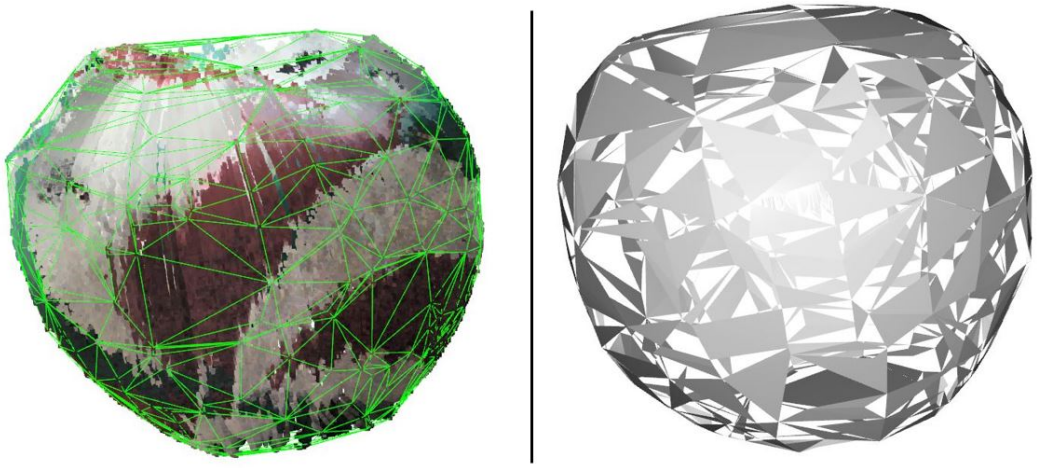

FIGURE 3.53: Convex hull of point cloud. Edges of convex hull (left), Original convex hull mesh (right)

3.5 Results

Please note that results showed in this section correspond to visually most relevant scans. Where a trade-off between color and position accuracy was considered. Also note that aside from the ball (regular object) where a theoretical volume value can be calculated, no manual measurements of the volumes were performed for the irregular objects. Also note that the error displayed is calculated as the difference between the manual measurement and the measurement acquired from the point cloud, divided by the manual measurement. All expressed as percentage.

X	Width	Depth	Height	Volume
Volleyball ball	203.46	204.09	185.33	4,226,491.91
Chili plant	240.97	214.29	236.96	5,533,012.00
Chili plant biomass	240.97	214.29	103.76	2,831,797.13

TABLE 3.3: Features results from laser triangulation scanner acquisitions (units in mms/mm^3)

X	Width	Depth	Height	Volume
Volleyball ball	205	205	205	4,510,868.90
Chili plant	220	210	280	-
Chili plant biomass	-	-	-	-

TABLE 3.4: Features results from manual measurement (Laser triangulation) (units in mms/mm^3)

X	Width	Depth	Height	Volume
Volleyball ball	0.75	0.44	9.59	6.30
Chili plant	9.53	2.04	15.37	-
Chili plant biomass	-	-	-	-

TABLE 3.5: Error between manual measurements and 3D model measurements(units in %)

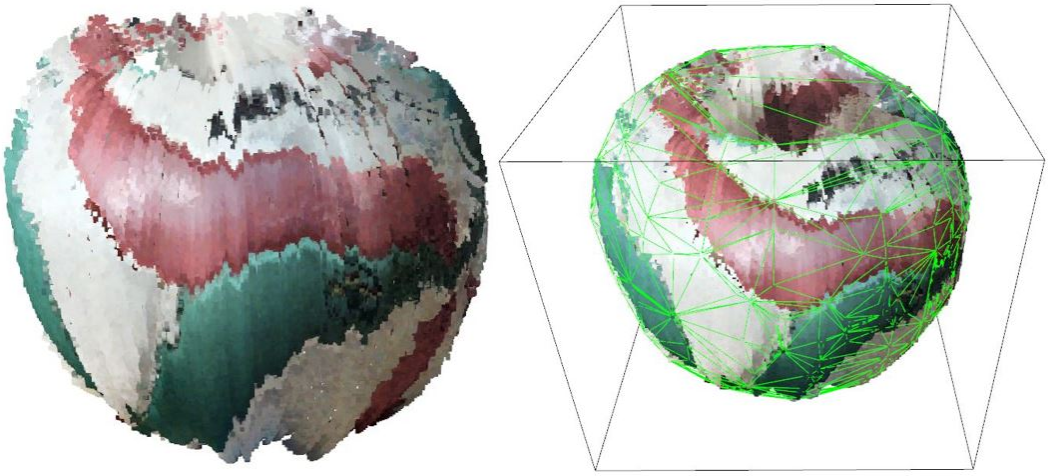

FIGURE 3.54: Laser triangulation scanner results: Acquired volleyball ball point cloud

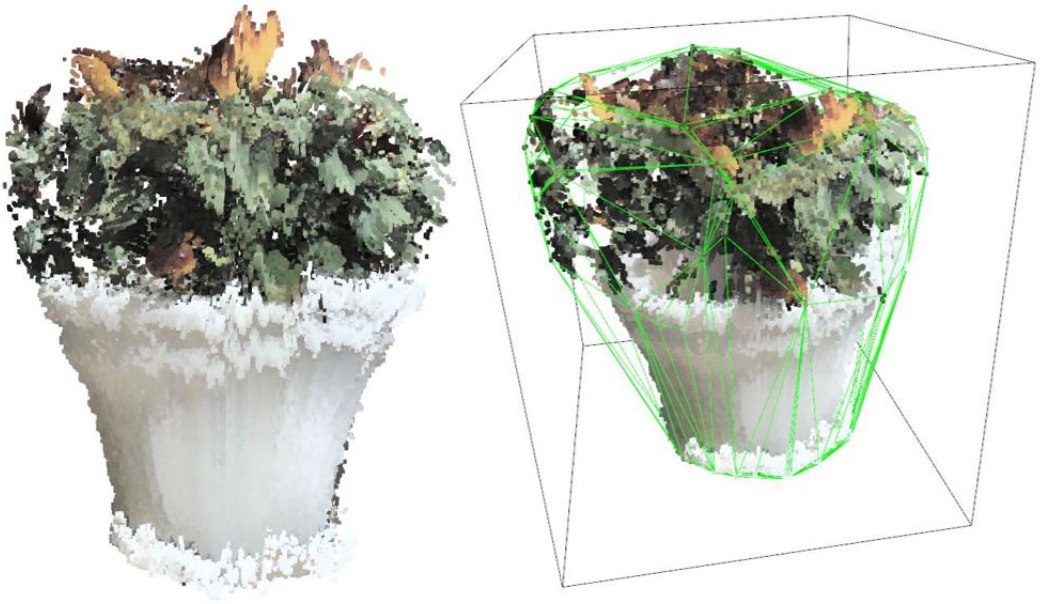

FIGURE 3.55: Laser triangulation scanner results: Acquired chili point cloud, whole plant

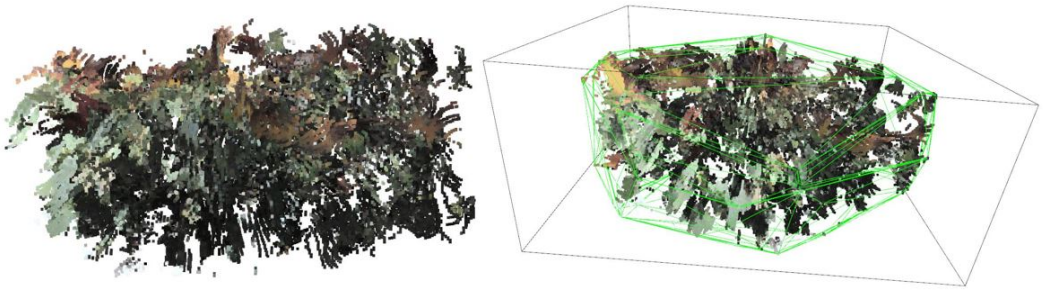

FIGURE 3.56: Laser triangulation scanner results: Acquired chili point cloud, just biomass

Chapter 4

Kinect v2 as a 3D scanner

4.1 Kinect version 2 overview

4.1.1 Hardware

The Kinect sensor 2, was initially manufactured for the XBOX ONE gaming console in 2013, to enable computer vision applications for a much better in-game experience. However, it has been widely tested for 3D reconstruction an other low-cost computer vision solutions through the last couple of years. The device can be seen in figure 4.1, and its hardware consists of:

FIGURE 4.1: Kinect 2.0 picture

- An RGB camera of 1920x1080 px.
- A Time-of-flight camera of 512x424 px using an IR sensor.
- A Microphone array.

Both cameras having a 70x60 FOV and working at a rate of 30fps.

4.1.2 Software

Although some open-source solutions can be found for acquiring data with the Kinect sensor, in this project, only the Microsoft Software Development Kit (MS SDK) was used along with the Microsoft 3D Scan Software.
The first one includes the *Kinect Studio 2.0* tool, which allows us to record data from both the RGB camera and the IR sensor, and acquire depth data as a colour-coded point cloud of a single shot. Its disadvantage is than only the points of a single perspective can be obtained at each single shot. On the other hand, the MS 3D Scan tool does not allow to retrieve this raw information, but returns a complete 3D model from a video taken with the Kinect sensor. The file format used by this software is "3MF", a 3D file format containing 3D information as well as obtained color and texture, but with low usage in the scientific community.

4.2 Data acquisition

4.2.1 Experiments' setup

Although one cannot access the source code of the 3D Scan software, it is assumed that to find correspondences, rotations and translations between frames, any sort of feature extraction software (corner extraction) must be performed. Because of this, a non-uniform background was considered. In addition, research done by E. Lachat and their team showed that a warm time for the sensor of 20 minutes is recommended before acquiring data [17]. With the Kinect sensor, brightness is a noise factor that must be considered, and although the Kinect sensor can work under sunny days conditions, it is not desirable as it will affect the IR sensor measurements considerably.

Now, for the actual data acquisition, the sensor is manually carried around the object, and using the "Microsoft 3D Scan" software, frame are captured for a certain amount of time ranging from less than 10 seconds to around 3 minutes.

The figures 4.2, 4.3 and 4.4 illustrate the differences in the obtained mesh when varying the data acquisition duration. This meshes are being visualized in "Microsoft 3D Builder".

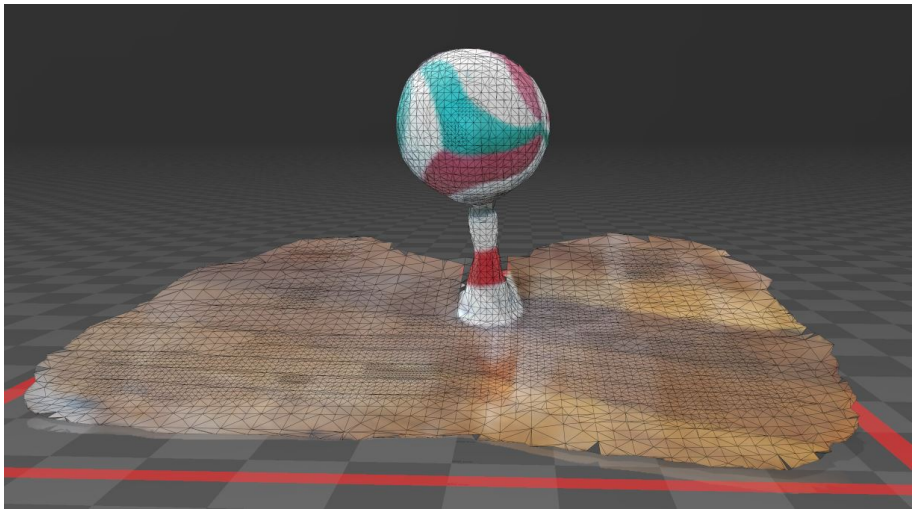

FIGURE 4.2: Kinect data: Mesh of a volleyball ball with a 3 seconds acquisition time

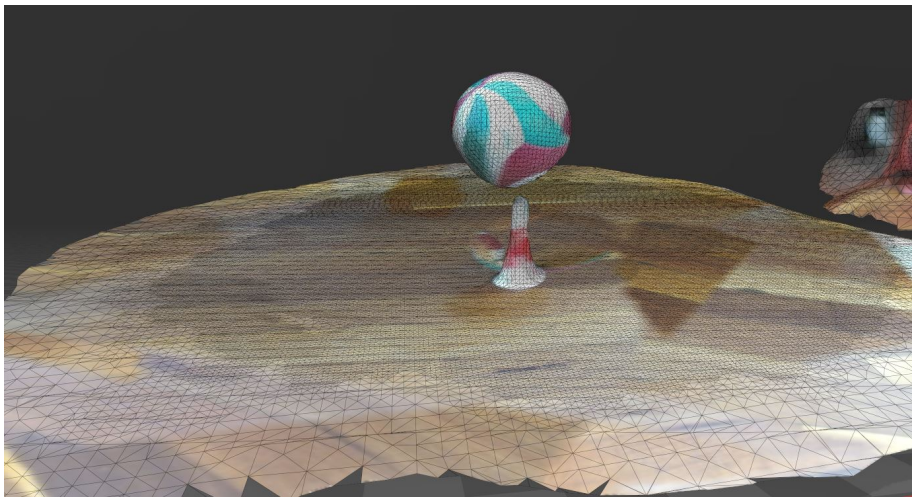

FIGURE 4.3: Kinect data: Mesh of a volleyball ball with a 60 seconds acquisition time

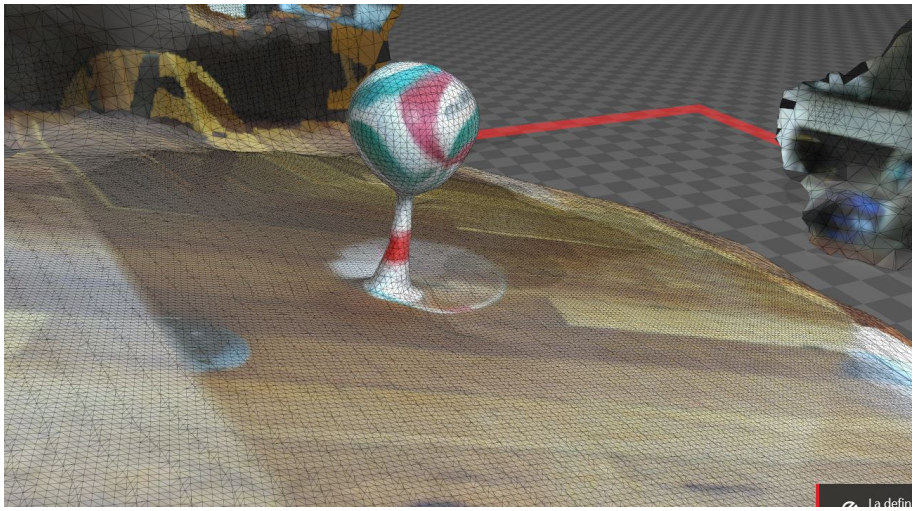

FIGURE 4.4: Kinect data: Mesh of a volleyball ball with a 150 seconds acquisition time

4.3 Features measurements

4.3.1 Mesh cleaning

Having acquired the data, one can observe that the Kinect sensor allows us to obtain information of the whole scene where the object is located. In many applications this can be a considerable advantage. Nonetheless, for the purpose of this project, just the object is needed. To split the object from the rest of the scene, an **STL** file is exported out of 3D builder, and the cleaning process is manually performed in Blender 2.8. Figure 4.5 shows the results of this process. It most be noted that in this process, the texture of the object is lost, although none of the proposed measurements requires it.

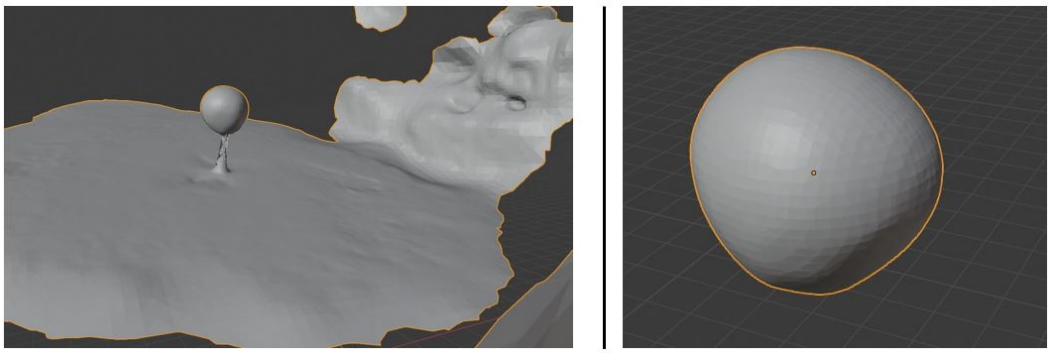

FIGURE 4.5: Cropped data from the acquired scene. Whole scene (left), region of interest (right)

4.3.2 Width, depth and height and volume measurements

The same procedure followed for the point clouds acquired with the laser triangulation method scanner is applied to each cleaned mesh here. Which basically consists of extracting width, depth, and height by calculating a bounding box of the mesh. A convex hull is calculated as well, and volume is estimated using a tetrahedrons sum. Both result of this process can be observed in figure 4.6.

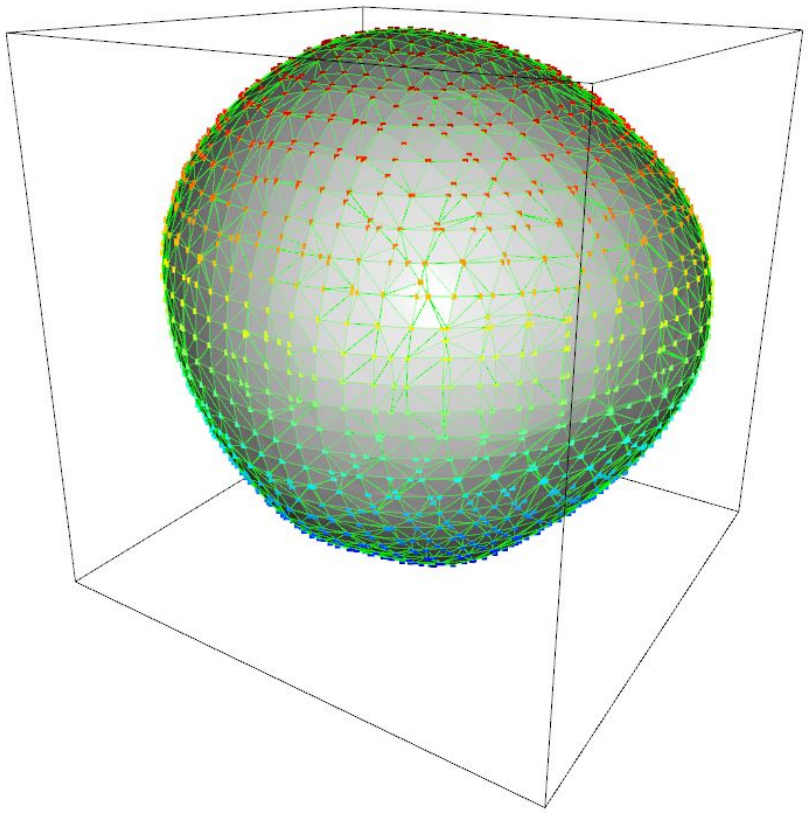

FIGURE 4.6: Bounding box and convex hull of the acquired and cleaned ball's mesh

4.4 Results

From acquired data, one can observe that taking a larger amount of frames from a scene, does not yields to better 3D models. In this sections, the visually best results obtained for each object are showed. As well as their respective features measurements.
Note: Error is calculated as the difference between the manual measurement and the measurement from mesh divided by the manual measurement. All expressed as percentage. Also note that aside from the ball, no measurements can be easily made manually for the objects' volumes.

X	Width	Depth	Height	Volume
Volleyball ball	189.35	196.93	200.82	3,616,641.90
Chili plant	190.89	207.96	252.15	5,463,119.84
Chili plant biomass	186.43	202.03	111.58	1,959,958.95

TABLE 4.1: Features results from Kinect acquisitions (units in mms/mm^3)

X	Width	Depth	Height	Volume
Volleyball ball	205	205	205	4,510,868.90
Chili plant	195	215	280	-
Chili plant biomass	-	-	-	-

TABLE 4.2: Features results from manual measurement (Kinect) (units in mms/mm^3)

X	Width	Depth	Height	Volume
Volleyball ball	7.63	3.94	2.03	19.82
Chili plant	2.11	3.27	9.95	-
Chili plant biomass	-	-	-	-

TABLE 4.3: Error between manual measurements and 3D model measurements(units in %)

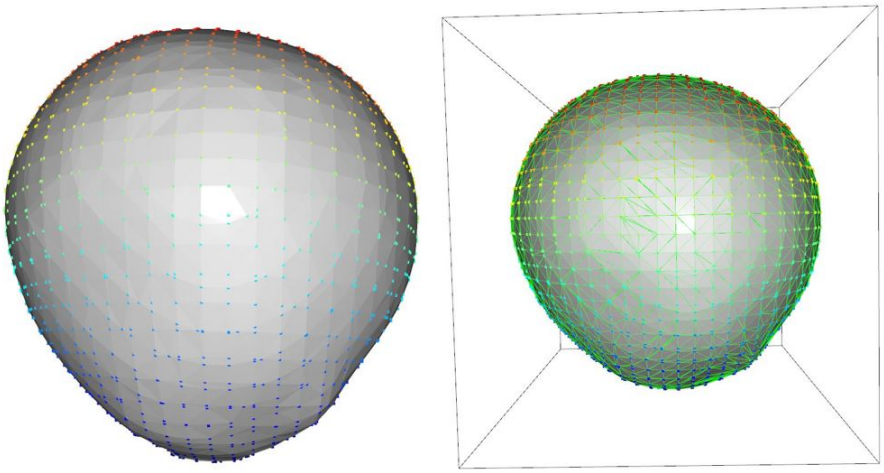

FIGURE 4.7: Kinect results: Acquired volleyball ball mesh

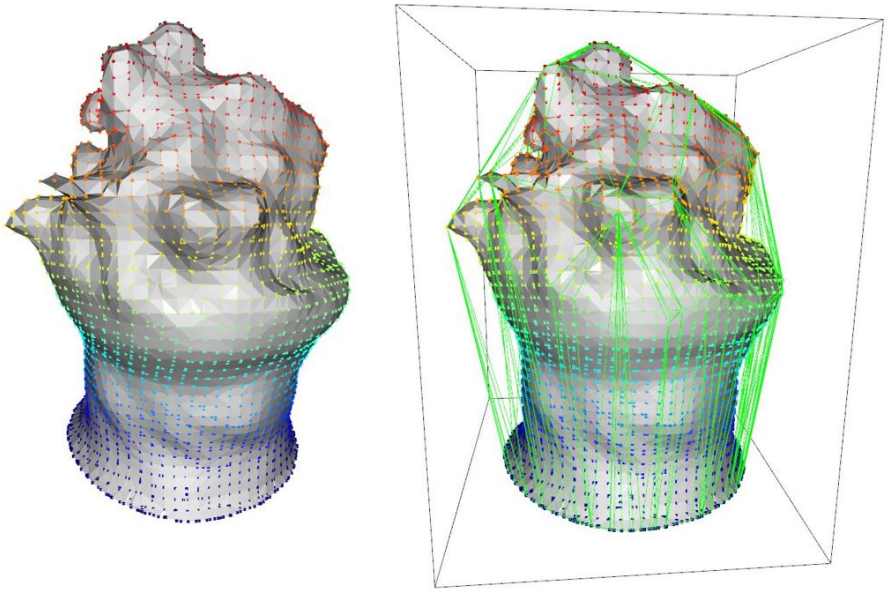

FIGURE 4.8: Kinect results: Acquired chili mesh, whole plant

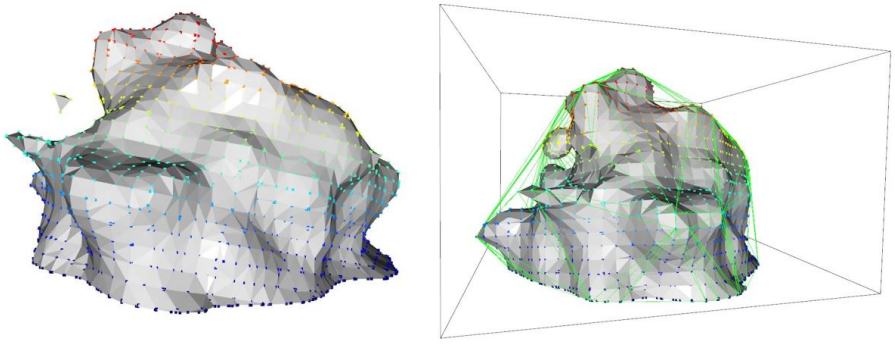

FIGURE 4.9: Kinect results: Acquired chili mesh, just biomass

Chapter 5

Plenoptic camera as 3D scanner

5.1 Plenoptic camera overview

5.1.1 Hardware

The available plenoptic camera used in this chapter is showed in figure 5.1, and has the following characteristics:

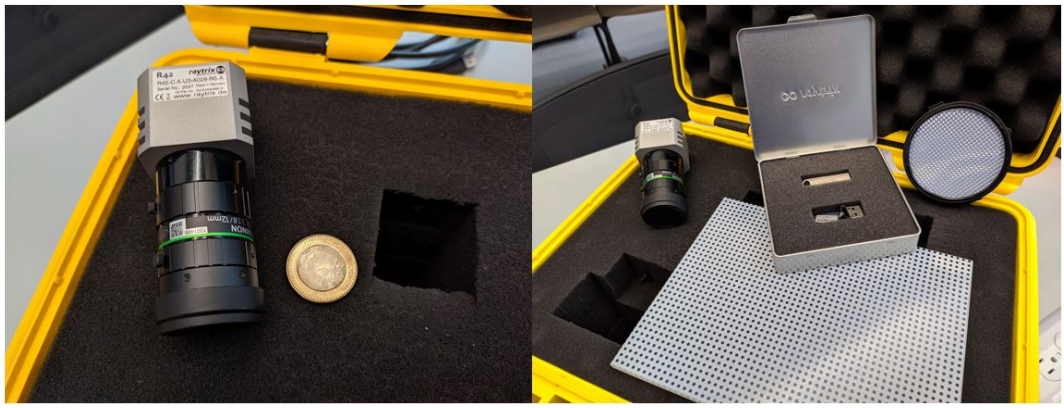

FIGURE 5.1: Plenoptic camera R42. Alone (left), with package, gray filter, calibration pattern and USB software (right)

- **Camera reference:** Raytrix R42
- **Camera type:** C-mount
- **Max frame rate:** 7 fps
- **Lens reference:** Fujinon HF1218-12M 2/3"
- **Lens equivalent resolution:** 12 MP
- **Focal length:** 12mm (actual 12.3)
- **Lens aperture:** F1.8 to F2.2

5.1.2 Software

The software used for retrieving data from the R42 camera was the *RxLive 5.0.57.1* (last version due November 2020). The software was run in a Laptop with 16GB RAM with an Nvidia GTX1050 GPU of 4GB. According to the minimum hardware requirements to run the software, this specifications are as low as they can be for this purpose.

5.2 Data acquisition

5.2.1 Experiments' setup

Contrary to both the Kinect and the laser triangulation setups, the plenoptic camera remains in a fixed position, and the respective point clouds are estimated by the software for each single frame. Because of this, the type of point clouds that can be obtained can not describe a whole object but a part of it. To obtain complete object models and acquire the missing information, moving the camera or the object is advised. The dynamic approach for the plenoptic camera goes beyond the scope of this project and an additional setup and software's license would be required.

In regard to the adjustable parameters of the camera, the shutter speed for the camera can be varied by software to adjust the light field acquisition to different light conditions. The same way, the frame rate can also be adjusted. For this setup it was fixed 1 fps to avoid hardware requirements beyond the basic ones. Figure 5.2 illustrates the used setup, consisting of a solid structure, a source of light, and the camera.

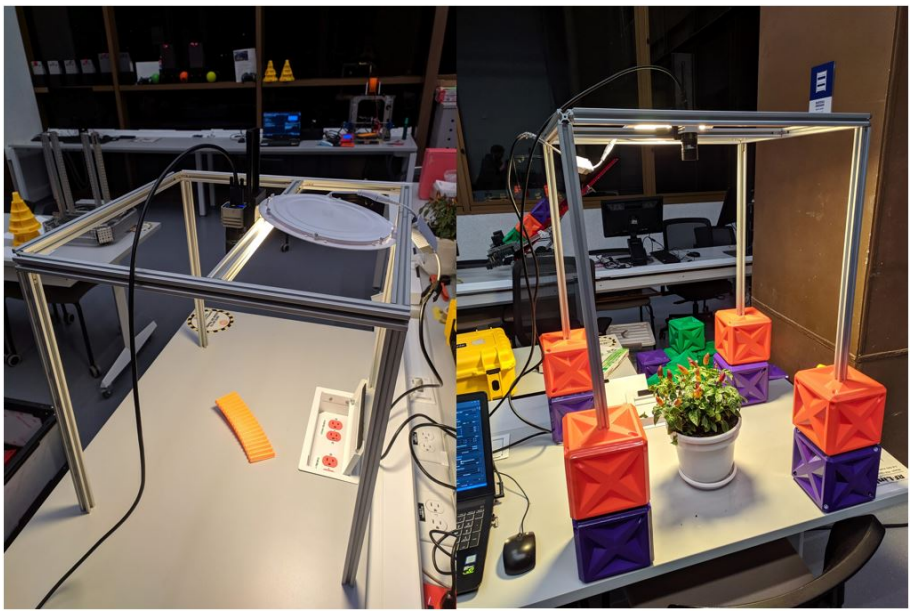

FIGURE 5.2: Setup for data acquisition with R42 camera at different heights

5.2.2 Plenoptic camera calibration

Two main calibrations are needed for acquiring depth maps and point clouds with the R42 camera. These are the MLA (micro lens array) calibration and the metric calibration. RxLive5.0 has many features for user to improve the quality of the calibrations, such as histograms graphs, and light exposure indicators for over and under exposed regions in the images.

MLA calibration

The purpose of this calibration is to align the micro lenses of the camera by using the calibration filter showed on the right image from figure 5.1. This will eliminate

FIGURE 5.3: Plenoptic camera. With attached calibration filter for MLA calibration (left), with no calibration filter (right).

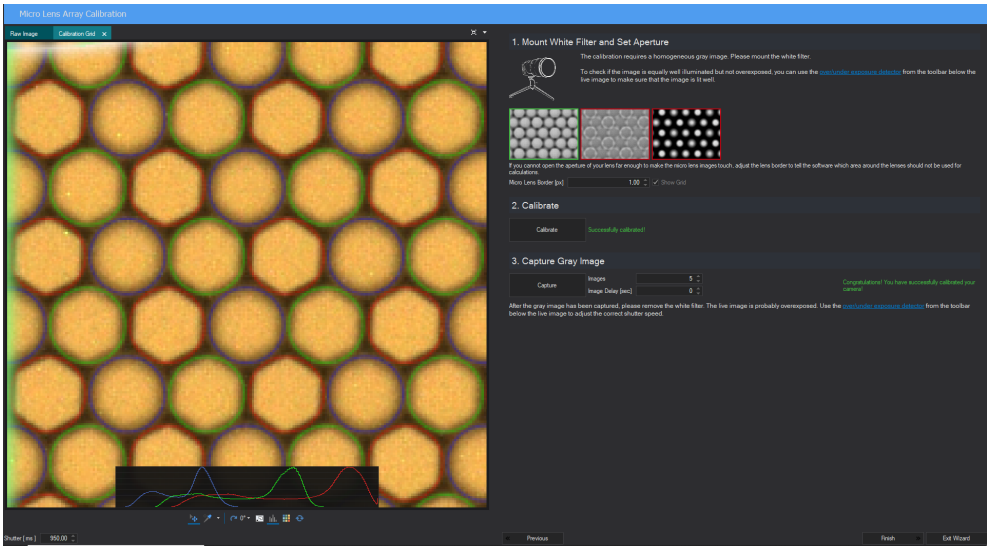

FIGURE 5.4: Calibration wizard in *RxLive5.0* for MLA calibration

the "honeycomb" patter from the acquired data and will allow the software to reconstruct images with highest precision.
The objective in this process is to variate the aperture of the lens in order to make each lens be as close as possible from its neighbors without overlapping with them. All the steps are explained in the same calibration wizard. In this calibration the illumination should be over 90%. Reason why a very high (near 1sec) shutter speed was fixed. Figure 5.3 shows how the camera is attached to the calibration filter for this step. Also, figure 5.4 shows the result of this calibration. Take into account that after making this calibration, the aperture and focus of the camera must not be altered.

Metric calibration

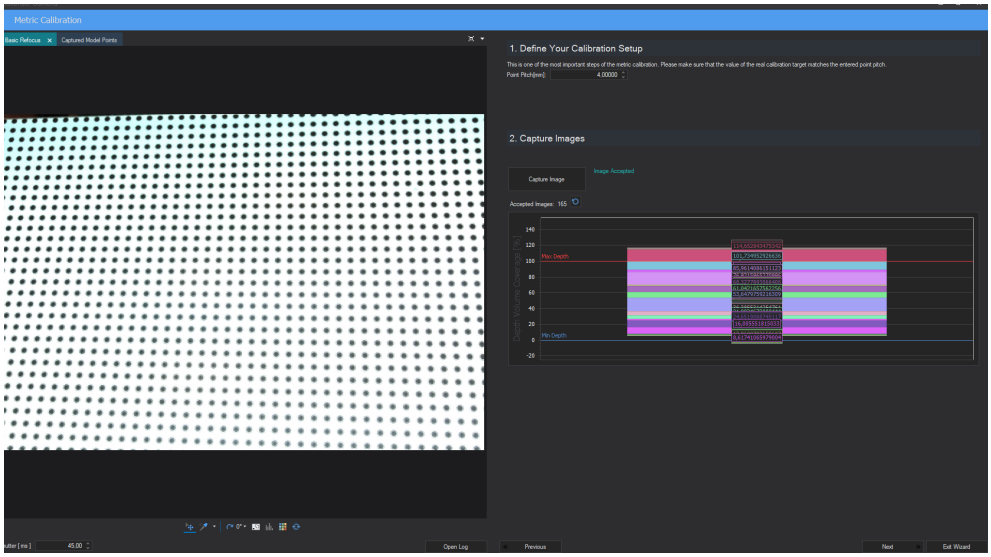

FIGURE 5.5: Calibration wizard in *RxLive5.0* for Metric calibration

The metric calibration is the process of creating the mathematical model for 3D reconstruction from 2D images. First, a set of photos are taken of an object of known dimensions. This object is called the calibration pattern and can be seen in the right image of figure 5.2. In order to estimate the best possible model, images of the pattern must be taken at different heights and orientations respect to the camera. As a feature, the calibration wizard indicates at which distances the calibration patterns have already been acquired. After estimating the model with a given set of images, it also gives a 0 to 5 stars indicator on how "accurate" the model is.

In our experiments, 3 different metric calibrations where conducted, the first one with less than 20 images, the second one with 45 images, and the third one with 165 images. Point cloud acquisitions with each calibration showed that although the 3 calibrations had a "5-star" rank, the bigger the image set, the better the obtained point clouds. For this reason, only results obtained with the 165 images calibration are showed in the results section of this chapter. The calibration wizard with the result of this process is illustrated in figure 5.5.

5.3 Traits' measurements

Although not specified, it can be seen that the *RxLive5.0* software attempts to subtracts the background before yielding an observable point cloud. This reduces significantly the pre-processing needed for features estimation. However, it still has some undesired noisy points that need to be eliminated. For this purpose, the same filtering/cleaning steps used for the triangulation scanner are used in these point clouds. Listed as: Cropping the region of interest of the point cloud with a cylinder, statistical outliers removal, and neighbor-based outliers removal. Please refer to subsection 3.4.1 for more information.
In the same way, width, depth, height, and volume can be estimated by calculating a bounding box and performing a tetrahedron sum over the convex hull of the point

cloud respectively. Please refer to subsections 3.4.2 and 3.4.3 for more information. An example of the cleaning and feature extraction results are showed in figures 5.6 and 5.7 respectively.

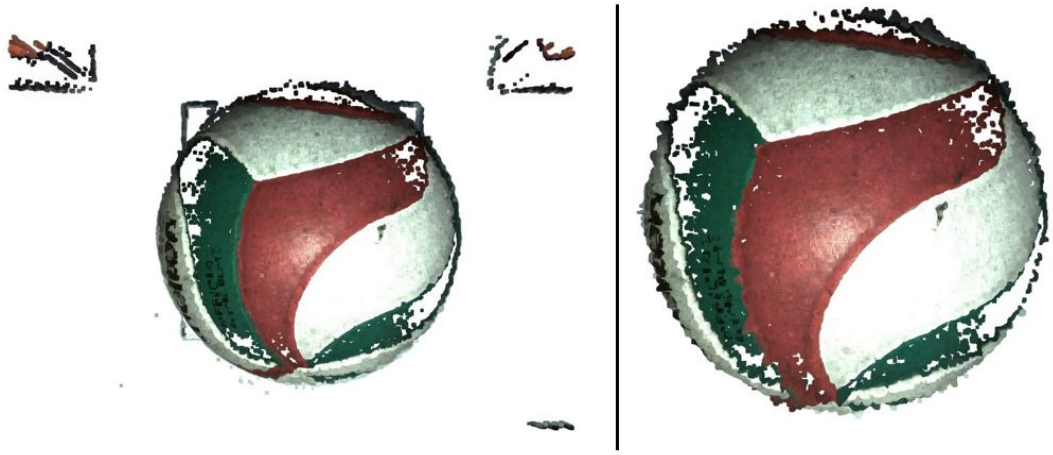

FIGURE 5.6: Cleaning/Filtering of a point cloud obtained from the plenoptic camera. Original (left), filtered (right).

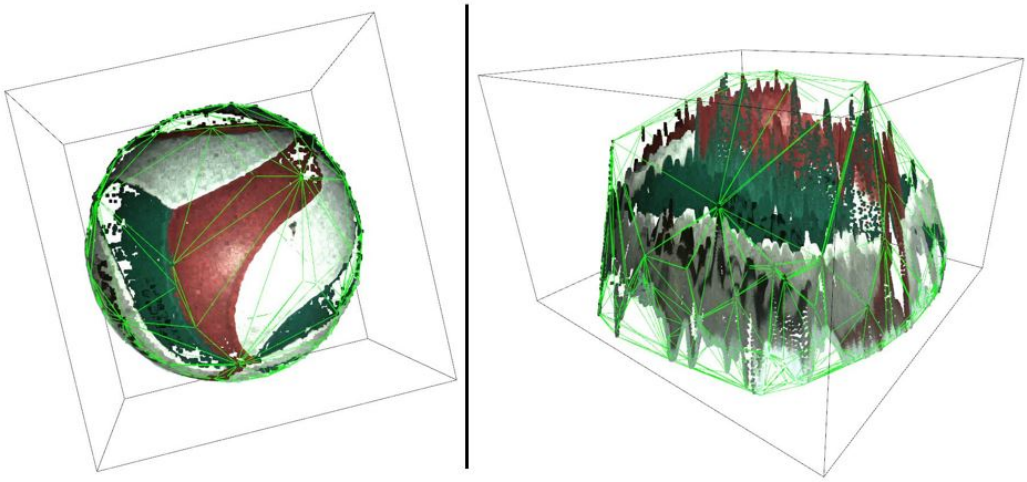

FIGURE 5.7: Example of bounding box and convex hull calculated for feature extraction

5.4 Results

In this sections, the visually best results obtained for each object are showed. As well as their respective features measurements.
Please note that the results obtained from the plenoptic camera have comparable data for the X and Y axis respect to the other two methods. However, given that only a part of the object can be reconstructed, it makes no sense comparing height or volume values. For this reason, although all this values will be presented in the section, this two does not reflect the overall performance of this method. ***Note: Error is***

calculated as the difference between the manual measurement and the measurement from the acquired point cloud, divided by the manual measurement. All expressed as percentage.

X	Width	Depth	Height	Volume
Volleyball ball	204.81	209.35	133.88	2,664,009.73
Chili plant biomass (top)	234.08	220.75	163.41	4,577,935.98
Chili plant (side)	284.99	199.81	161.87	4,874,097.23

TABLE 5.1: Features results from R42 camera acquisitions (units in mms/mm^3)

X	Width	Depth	Height	Volume
Volleyball ball	205	205	205	4,510,868.90
Chili plant biomass (top)	230	215	280	-
Chili plant (side)	280	215	230	-

TABLE 5.2: Features results from manual measurement (Plenoptic) (units in mms/mm^3)

X	Width	Depth	Height	Volume
Volleyball ball	0.093	2.12	34.69	40.94
Chili plant biomass (top)	1.77	2.67	41.64	-
Chili plant (side)	1.78	7.06	29.62	-

TABLE 5.3: Error between manual measurements and 3D model measurements(units in %)

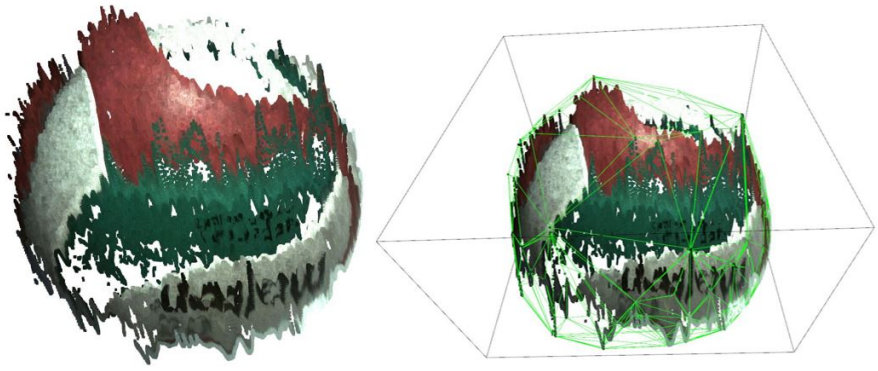

FIGURE 5.8: Plenoptic camera results: Acquired point cloud of volleyball ball

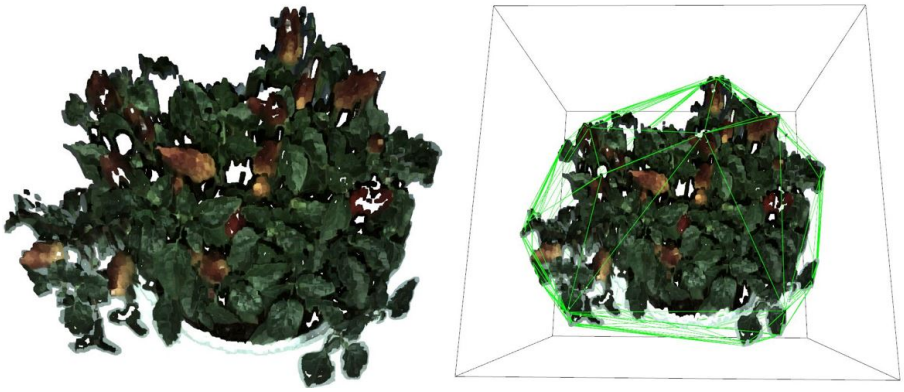

FIGURE 5.9: Plenoptic camera results: Acquired point cloud of Chili Plant, top view

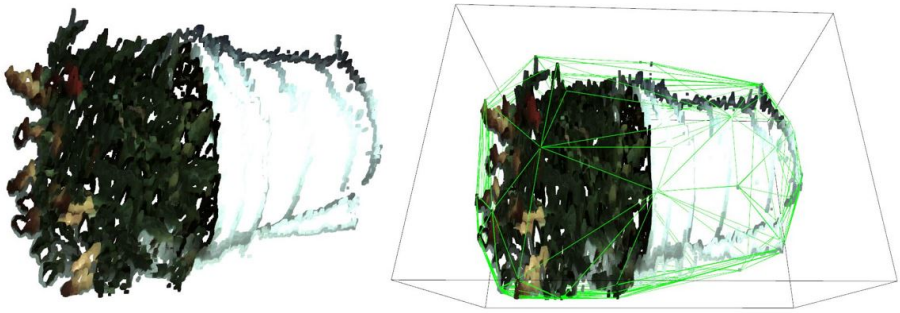

FIGURE 5.10: Plenoptic camera results: Acquired point cloud of Chili Plant, side view

Chapter 6

Results analysis and discussion

This chapter aims to analyze both the process and results obtained from three exposed methods. The analysis examines quantitative and qualitative results. Please note that the results obtained for the laser triangulation, Kinect sensor, and plenoptic camera are shared in sections 3.5, 4.4, and 5.4 respectively.

6.1 Quantitative analysis

In this section, numerical results from all three scanning methods are analyzed. Due to the different nature of the point clouds/meshes obtained, the analysis is based on measurements of width, height, depth, and volume of modeled objects where these analyses make sense. For each method, the models of two reference objects are obtained, a regular object (volleyball ball) and a plant (chilli plant). For the second one, also the measurements corresponding to biomass are estimated.

6.1.1 Laser triangulation scanner

For the laser triangulation scanner results, the lowest errors respect to manual measurements of the object were obtained for the width and depth of the object, being as low as 0.75% and 0.44% for the regular object, respectively and a bit higher for the plant 9.53% and 2.04%, which an overall bigger object. In the case of the height measurement for both the plant and the regular object, a more significant error is obtained, of 15.37% and 9.59%, respectively. Nevertheless, this increasing error in this axes was expected.
The reason behind it is the location of the camera. The physical montage allows the camera and the laser to move along 2 of the 3 axes in space (although dependent one from the other), but the third axis (vertical one) remains fixed for a single scan, leaving some spots outside the field of view of the sensor unseen, in particular, the points at the very top and bottom of the object. Taking this into account, but without knowing the exact coordinates where the camera stops detecting the laser, a reasonable height of the "almost complete" object is obtained. In addition, the volume error for the ball relies almost entirely on this limitation for the height measurement, being a 6.30% for the original object. About the measurements without manual comparison values, as the volume for the plant or the plant biomass, one can argue by observing the original object that a volume of about half the total mass of the plant is reasonable under the assumption that the plant has a very dense amount of biomass.

6.1.2 Kinect sensor as 3D scanner

By analysing the Kinect sensor results, one can argue that general measurements of the objects are relatively accurate. But the axis of error depends entirely on the object.

In regular object case, the highest error corresponds to the width measurement, of 7.63% respect to manual measurements. The lowest error corresponds to the height measurement, with 2.03% error. On the other side, for the plant mesh, the feature with least error corresponded to the width, with 2.11%. The feature with the largest amount of error corresponded to the height, with 9.95%. Depending on the object's amount of detail, and after many attempts of generating the best model, it can be argued, that the best results are achieved when bigger objects are scanned. With a larger lost of information where objects have small details and high frequency textures. For the volume measurement, a considerably high error of 19.82% is obtained, related to the deformation added to the 3D model even when scanning a regular, opaque and low detailed object such as a volleyball ball. Although there is no access to the algorithms run in the background by the software, it is argued that this error is not due to the sensor itself but due to the inaccuracies of the reconstruction software.

6.1.3 Plenoptic camera R42 as 3D scanner

The plenoptic camera poses a different set of results with a totally different set of drawbacks. In the case of the model's width and depth measurements of the object, the best and most consistent overall results out of the three methods are obtained, with errors of 0.093% and 2.12% for the regular object, and 1.77% and 2.67% for the plant. Here, the highest error is obtained for the height of the object (basically distance to camera), rounding a 35% to 42%. Also, observing the resulting scans, not just the general feature measurements but also single points where similar height should be estimated have a high deviation from a reasonable value. Because of this lack of coherent height values, a volume comparison with manual measurements makes no sense here. In the case of the plant, the acquired information of the "whole plant" seen from an upper view, can be taken as a not so accurate acquisition of the biomass only, where very good texture and dimensions measurements can be expected for the width and height of the specimen. Here it must be added that this method where both the main sensor and the object remain fixed works with much less information on the object, having much more blind spots overall.

6.2 Qualitative analysis

After observing and analysing quantitatively the results obtained in last section, here some of the advantages and disadvantages of each method are exposed, referencing its portability, time-cost efficiency, and their main issues. Although some categories may not be applicable, every appreciation is also compared to a manual measurement procedure. Table 6.1 summarizes some elements taken into account for each method:

1. **Calibration time:** The Kinect sensor works with no calibration needed, which reduces the setup considerably for acquiring data. The laser triangulation method requires both a camera calibration and a metric (scanner) calibration. The plenoptic camera requires an MLA calibration and a metric calibration as well.

2. **Data acquisition time:** Once calibrated, the plenoptic camera can take images right away. There is a considerably low time between one take and another. On its side, the Kinect requires a warming time to acquire more reliable data, and the acquisition per se can take from a few seconds to a couple of minutes.

-	Laser triangulation	Kinect sensor	Plenoptic camera	Manual measurements
Calibration time	<2 hours	NR	<2 hours	NR
Data acquisition time	<1 hour	20 min(warm) + <3 min	1 min	2 min
Data processing time	<30 min	<20 min	1 min	NR
Data size	1.5GB	20MB	1.5GB	<1KB
Hardware requirements	Medium-low	Medium-low	Very-High	NR
Working distance range	<1m and 0.7m height	1.5m - 4.5m	<0.7m <0.7m	NA NA
Ease of use	Difficult	Easy	Intermediate	Very easy
Portability	Low	Medium-High	Low	NA
Open-source software	Yes	Partially	No	- -
Data export ease	High(P.C) NA(Mesh)	Low(P.C), High(Mesh)	High(P.C), High(Mesh)	NA(P.C) NA(Mesh)
Reconstructed object fidelity	Very Good	Good	Poor	None
Color/texture accuracy	Acceptable	Good	Best	None
Features accuracy	Good	Poor	Very Good	Best
Measurable features	3 axis + volume	3 axis + volume	along 2 axis	along 3 axis axis
Cost (USD)	300	400	106,000	3

TABLE 6.1: Qualitative comparison between the 3D scanning methods exposed throughout this project. *(NR="Not required", NA="Not applicable", PC="Point cloud"). (Best=5/5, Very Good=4/5, Good=3/5, Acceptable=2/5, Poor=1/5, None=0/5)*

The laser triangulation scanner takes images from every angle, but requires a considerable time to gather it all.

3. **Data processing time:** The Kinect sensor and the laser triangulation methods have a comparable data processing time, of around 30 minutes with big amounts of data. The plenoptic camera on its side can almost instantly process the acquired data (which is a single frame) to generate a point cloud.

4. **Data size:** The Kinect sensor software (3D scan) discards all the data from the acquired frames and makes only available the mesh file, which is no bigger than 20MB for most objects. The laser triangulation scanner on its side, leaves all image pairs available (near 750) in addition to the point cloud file. The reason why a much bigger space is required to store it. Finally, the plenoptic camera saves not only the point cloud but also other additional files. Like the 3D model, some depth map images, the processed image, the raw image, the light rays file, and many other *middle steps* images. Requiring almost 1.5GB of space to store the data from a single shot.

5. **Hardware requirements:** Here the laser scanner and the Kinect sensor have comparable hardware requirements for running their respective software, where a low-end GPU (like a laptop's Nvidia GTX1050 3GB) is more than enough to run the algorithms and graphical interfaces smoothly. The plenoptic camera's software (RxLive5.0) on its side, has as minimum GPU requirement a Nvidia GTX1080 4GB, running the program very slowly and allowing to visualize no more than 2 or 3 outputs of the data at the same time.

6. **Working distance range:** The most flexible solution for this category is without a doubt the Kinect sensor. This method can acquire data from small objects or big scenes from distances between 1.5m-4.5m, and under some conditions even less. The plenoptic camera is in the middle point as it "can" acquire information of all kinds of objects at different ranges depending on its calibration. However, it is limited to the setup where it is mounted. Finally, the laser scanner has been designed for objects of no more than 0.7m tall and as big as 1m for their depth and width.

7. **Ease of use:** The Kinect sensor is what one would call a plug and play device. With a Windows OS computer, the device is automatically detected, its drivers are installed, and data can be acquired instantly. Because both the triangulation scanner and the plenoptic camera require a setup and a calibration to be used, their difficulty is comparable. Nevertheless, because the laser scanner has no graphical interface yet, it is considered to have a greater difficulty.

8. **Portability:** None of the three solutions exposed in this project is completely portable as these are all wired devices. But because the plenoptic camera and the laser scanner have a physical montage, their ease of transport is reduced.

9. **Open-source software:** The laser triangulation algorithms have all been developed from scratch or using open source tools, making it free to use and modifiable. The Kinect sensor software on its part is free to use, having a MS Windows OS. Still, although some of its code can be found in the Software development kit from Microsoft, many of its algorithms are unknown. Last but not least, the Raytrix RxLive software for the Plenoptic camera is a completely proprietary software. Every single functionality is sold as a separate package, with a cost of several thousand dollars each.

10. **Data export ease:** The plenoptic camera can export 3D models in STL and PLY files, which are widely used, as well as the point clouds in XYZ ASCII format or PCD binary format, which can also be read by most point cloud visualization tools or using python libraries. For the laser scanner on its side, algorithms are also implemented for exporting XYZ and PCD formats. But in the case of the 3D Scan software for the Kinect scanner, only the 3MF file format is available for export. This has poor documentation and no other tools than the developed by Microsoft for its processing. Relying on other software for type conversions often resulting in data loss.

11. **Reconstructed object fidelity:** This category can be understood as the qualitative resemblance of the original scanned objects to their respective point clouds or 3D models. For the laser triangulation scanner, a great overall model of each object is achieved, with the most qualitative resemblance to the original objects. The Kinect sensor, for its part, obtains models that are quite similar to the actual objects. However, when having to reconstruct the entire object, it

can sometimes produce highly distorted objects or omit parts of it. Last, the plenoptic camera generates the object models with the greatest resemblance to original objects, but only of seen from the camera's point of view. When visualizing the object from other perspectives (visualizing the estimated height), the object presents the most deformations out of the three methods. The point clouds can be seen as with high frequency deformations in it.

12. **Color/texture accuracy:** In this category, the plenoptic camera yields the best results, with the acquired colors being the most similar to their originals, expressed as vertex colors. The Kinect sensor has its own downsides in this subject, as its colors are expressed with a general texture instead of vertex colors. In addition, some of the textures from the background are miss-placed from where the object points are. Finally, the laser triangulation scanner does, in fact, a great job at retrieving vertex colors, but because scans must be performed with fewer ambient light, darker point clouds are often achieved even with the color enhancing algorithm.

13. **Features accuracy:** Overall, acquired features reflect that all three methods need their improvements in the object reconstruction process. Nevertheless, the results in quantitative analysis show that the plenoptic camera has the best results, but only for the X and Y axis, yielding totally incoherent values for the Z-axis. The Kinect sensor, on its side, will often yield smaller models respect to the original objects, and with variable errors in each axis, mainly dependent on the object's shape. Finally, the laser triangulation scanner often produces bigger models than the original ones. Although acceptably maintaining the aspect relation, for the most part, this yields larger X, Y, and Z measurements than the ones of the actual objects.

14. **Measurable features:** Both the Kinect and the laser scanner allow the measurement in the three axis and volume calculation, obtaining coherent values. The plenoptic camera on its side, yields no useful information on the Z-axis, as the object can not be seen from different angles.

15. **Cost(USD):** Both the Kinect sensor and the laser triangulation scanner have prices of comparable magnitude. However, it must be taken into account that the Kinect sensor for its price is sold as a plug and play device. In contrast, the laser triangulation scanner was manufactured for this project with all pieces costing its price. The plenoptic camera on its side, has a very high price, causing it to be unaffordable for must purposes other than the academic one.

Chapter 7

Conclusions and future direction

7.1 Conclusions

Each one of the methods has it own strengths. While the laser triangulation method performs well at acquiring detailed objects in low light conditions at the lowest cost, the time-of-flight sensing method performs better at scanning low detailed bigger objects in environments with a wider variety of light conditions. The plenoptic camera on its side, stands out for its high accuracy in color/texture information acquisition from the object, as well as for working with much less information than the other two methods, and still giving out the best results for measured width and depth of an object. As weaknesses on the other hand, the plenoptic camera has its considerably higher price along with its much more noisy height estimations (distance to camera). The Kinect sensor, on its side, has an smoothing effect on all surfaces along with its overall deformation of the objects when data from several perspectives is gathered. Finally, the laser triangulation scanner has its inaccuracies due to its mechanical setup movements along with its much more narrower requirements respecting to light conditions. After having analyzed qualitatively and qualitatively all methods, it is concluded that the designed laser triangulation scanning method, although away from perfection, can obtain the best results for acquiring high detailed point clouds of plants at a whole-plant level, being as well, the cheapest of the presented methods.

7.2 Aspects to improve

The laser triangulation scanner has many aspects that can be improved, starting from the mechanical setup to the software where:

- Vertical movements should be taken into account in the movement of the device for avoiding small magnitude errors in the reconstruction.
- A higher resolution camera should be used to get more dense and accurate point clouds.
- A laser with greater intensity should be used to allow high resolution scans in a wider variety of light conditions.
- Self calibration of rotation and center should be implemented to improve accuracy in object sizes.
- A portable power source should be evaluated to upgrade the data acquisition system from wired to wireless.

- Different acquisition algorithms and cameras should be tested to decrease data acquisition time.

7.3 Areas of further research

- The laser triangulation scanner proposed in this project takes into account different viewpoints along the two axes corresponding to the horizontal plane, but with one dependent from the other as it performs a circumference-type movement.
  Adding the vertical degree of freedom to the method should be considered. Also, eliminating the dependency between the two axis of the horizontal plane should be evaluated as it would yield more information on the object.

- In this project, point clouds were acquired from a static plenoptic camera with a single perspective of an object. Although poor measurements were found in the height axis, multiple viewpoints should be tested as this methods has performed better in many aspects where key features can be retrieved from a plant.

- As many of these methods have individually performed better under certain circumstances, data/sensor fusion between these, and other 3D reconstruction methods should be evaluated.

- This project does not cover the modeling of the specimens over time for the case of plants. However, the measurement of growth in biomass and other plant features using 3D scanning, should yield meaningful results.

Appendix A

Schedule: Gantt diagram

Figure A.1, andA.2 illustrate the proposed schedule at an early planning stage of the project. Main tasks are listed and tasks for the first and second semesters are showed.

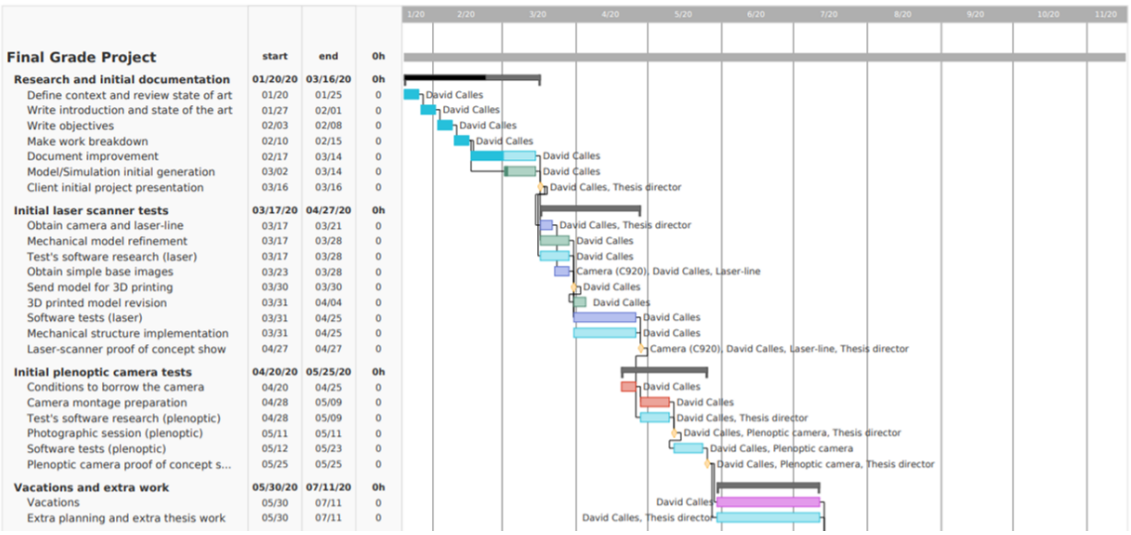

FIGURE A.1: Project's Gantt diagram for first semester

FIGURE A.2: Project's Gantt diagram for second semester

Appendix B

Camera calibration: Theoretical background

Extrinsic parameters

Both *world's coordinate* system and *camera's coordinate* system are related by a rotation **R** and a translation **t** often represented as vector or matrix for each of this parameters to then turn it into the so-called Extrinsic matrix **P**. The relationship between these two is described as follows:

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = \mathbf{R} \begin{bmatrix} X_w \\ Y_w \\ Z_w \end{bmatrix} + \mathbf{t} \tag{B.1}$$

Or represented in an other way:

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = [\mathbf{R}|\mathbf{t}] \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \tag{B.2}$$

Where this new gathered matrix is called the extrinsic matrix **P** of the camera.

Intrinsic parameters

Once the exact place in the space where the camera coordinate system is with respect to that of the world has been defined, an image plane is located at the focal length F of the camera. The parameters of the camera that must be estimated are the following ones:

1. The different focal lengths due to possible variations between the horizontal and vertical size of a pixel. (fx, fy)
2. The optical center of the camera that may not coincide with the center of the image coordinate system. (cx, cy)
3. Possible skew between the x and y axes of the camera's sensor (γ)

These variables are used to get the intrinsic matrix **K**, from the equations that describe the geometric relationship between each 3D point (Xc, Yc, Zc) in the world and the corresponding (x, y) projection in the image plane.

$$\mathbf{K} = \begin{bmatrix} f_x & \gamma & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{B.3}$$

Finally, because in **OpenCv**, we work with images starting from the top left corner, the image plane is translated from (x, y) to (u, v). Ending with the following equation:

$$\begin{bmatrix} u' \\ v' \\ w' \end{bmatrix} = \begin{bmatrix} f_x & \gamma & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} \tag{B.4}$$

Coordinate systems

Although the extrinsic parameters allow establishing a relationship between the world coordinate system and the camera coordinate system, the intrinsic ones allow us to define the other necessary relationship, which is the one that exists between the camera coordinate system and the projection of the image in the coordinates of the image plane. Gathering both of them together, we can finally get a full equation that relates both the 3D coordinates in the real world and the observed image over the image plane in the following equation:

$$\begin{bmatrix} u' \\ v' \\ z' \end{bmatrix} = \mathbf{P} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \tag{B.5}$$

Where this new matrix P is obtained from both the extrinsic and intrinsic matrix to describe camera characteristics and position respect to the world adequately.

$$P = K * \begin{bmatrix} \mathbf{R} | \mathbf{t} \end{bmatrix} \tag{B.6}$$

Distortion effects and coefficients

In the 20th century, cheap pinhole cameras were introduced as an everyday device in our lives, although this cheapness generally comes with a highlighted drawback, which is significant distortion. Two main types of distortion can be identified.

- **Radial distortion:** he most common distortion type, where image magnification decreases or increases with the distance from the optical axis origin.

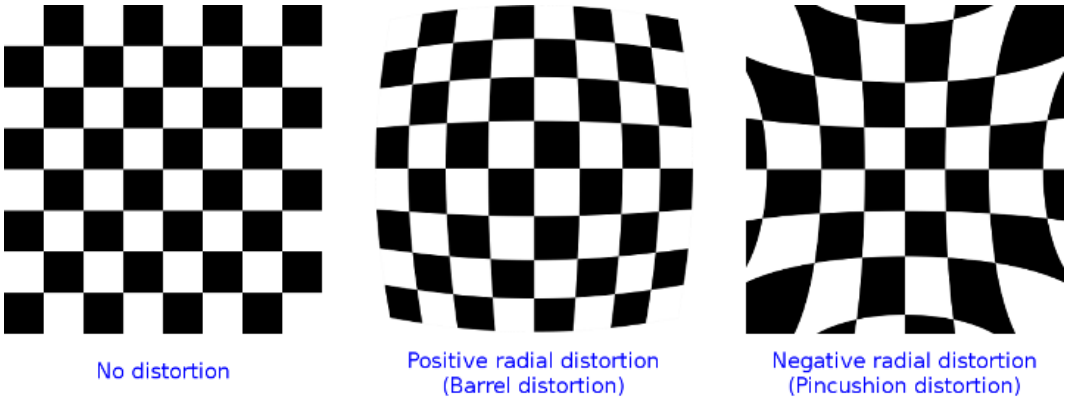

FIGURE B.1: Visual effects with radial distortion [28]

This type of distortion can be corrected from the distortion parameters following equation B.7.

$$\begin{aligned} x_{(corrected)} &= x(1 + K_1 r^2 + K_2 r^4 + K_3 r^6) \\ y_{(corrected)} &= y(1 + K_1 r^2 + K_2 r^4 + K_3 r^6) \end{aligned} \tag{B.7}$$

- **Tangential distortion:** This other type of distortion, although it is less visible, it may also affect the resolution desired in many applications. It occurs when the camera lens and the camera sensor are not parallel.

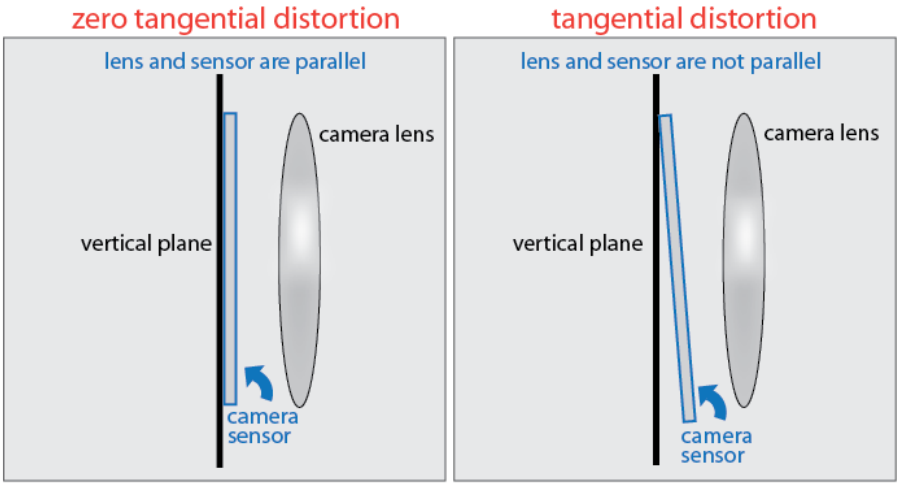

FIGURE B.2: Graphic representation of tangential distortion [29]

This other type of distortion can also be corrected from the distortion parameters following equation B.8.

$$\begin{aligned} x_{(corrected)} &= x + [2p_1 xy + p_2(r^2 + 2x^2)] \\ y_{(corrected)} &= y + [2p_2 xy + p_1(r^2 + 2y^2)] \end{aligned} \tag{B.8}$$

The two types of distortion shown above are both fully described by the distortion vector of the form:

$$DistortionCoefficients = \begin{bmatrix} k_1 & k_2 & p_1 & p_2 & k_3 \end{bmatrix} \tag{B.9}$$

Bibliography

[1] Max Roser. *Future Population Growth*. 2014. URL: https://ourworldindata.org/future-population-growth (visited on 09/02/2020).

[2] Yongjian Wang et al. "Maize plant phenotyping: Comparing 3D laser scanning, multi-view stereo reconstruction, and 3D digitizing estimates". In: *Remote Sensing* 11.1 (2019). ISSN: 20724292. DOI: 10.3390/rs11010063.

[3] Robert T. Furbank and Mark Tester. "Phenomics - technologies to relieve the phenotyping bottleneck". In: *Trends in Plant Science* 16.12 (2011), pp. 635–644. ISSN: 13601385. DOI: 10.1016/j.tplants.2011.09.005. URL: http://dx.doi.org/10.1016/j.tplants.2011.09.005.

[4] Stijn Dhondt, Nathalie Wuyts, and Dirk Inzé. "Cell to whole-plant phenotyping: The best is yet to come". In: *Trends in Plant Science* 18.8 (2013), pp. 428–439. ISSN: 13601385. DOI: 10.1016/j.tplants.2013.04.008.

[5] Juan Carlos Perez-Cortes et al. "A system for in-line 3D inspection without hidden surfaces". In: *Sensors (Switzerland)* 18.9 (2018), pp. 1–25. ISSN: 14248220. DOI: 10.3390/s18092993.

[6] Stefan Paulus et al. "High-precision laser scanning system for capturing 3D plant architecture and analysing growth ofcereal plants". In: *Biosystems Engineering* 121 (2014), pp. 1–11. ISSN: 15375110. DOI: 10.1016/j.biosystemseng.2014.01.010.

[7] Stefan Paulus. "Measuring crops in 3D: Using geometry for plant phenotyping". In: *Plant Methods* 15.1 (2019), pp. 1–13. ISSN: 17464811. DOI: 10.1186/s13007-019-0490-0. URL: https://doi.org/10.1186/s13007-019-0490-0.

[8] Kenneth W. Michael Wills. *Differences Between "Physical" and "Physiological"*. 2018. URL: https://sciencing.com/the-difference-between-craniology-phrenology-12759816.html (visited on 02/02/2020).

[9] Yann Chéné et al. "On the use of depth camera for 3D phenotyping of entire plants". In: *Computers and Electronics in Agriculture* 82 (2012), pp. 122–127. ISSN: 01681699. DOI: 10.1016/j.compag.2011.12.007.

[10] Stefan Paulus et al. "Automated analysis of barley organs using 3D laser scanning: An approach for high throughput phenotyping". In: *Sensors (Switzerland)* 14.7 (2014), pp. 12670–12686. ISSN: 14248220. DOI: 10.3390/s140712670.

[11] Anthony Paproki et al. "A novel mesh processing based technique for 3D plant analysis". In: *BMC Plant Biology* 12 (2012), pp. 1–13. ISSN: 14712229. DOI: 10.1186/1471-2229-12-63.

[12] Lirong Xiang et al. "Automated morphological traits extraction for sorghum plants via 3D point cloud data analysis". In: *Computers and Electronics in Agriculture* 162.January (2019), pp. 951–961. ISSN: 01681699. DOI: 10.1016/j.compag.2019.05.043. URL: https://doi.org/10.1016/j.compag.2019.05.043.

[13] Tino Dornbusch, Peter Wernecke, and Wulf Diepenbrock. "A method to extract morphological traits of plant organs from 3D point clouds as a database for an architectural plant model". In: *Ecological Modelling* 200.1-2 (2007), pp. 119–129. ISSN: 03043800. DOI: 10.1016/j.ecolmodel.2006.07.028.

[14] Edge. *3D SCANNING: METHODS, APPLICATIONS AND ADVANTAGES - EDGE 3D*. 2019. URL: https://www.edge3d.io/what-3d-scanning-types-applications-advantages-and-methods-laser-lidar-contact-based-photogrammetry-triangulation-structured-pulse-light/ (visited on 02/02/2020).

[15] Vision Components. *ARM Laser Triangulation*. 2020. URL: https://www.vision-components.com/en/products/oem/3d-systems/arm-laser-triangulation/ (visited on 02/16/2020).

[16] S.M. Emam, S. Khatibi, and K. Khalili. "Improving the Accuracy of Laser Scanning for 3D Model Reconstruction Using Dithering Technique". In: *Procedia Technology* 12 (2014), pp. 353–358. ISSN: 22120173. DOI: 10.1016/j.protcy.2013.12.498.

[17] Elise Lachat et al. "Assessment and calibration of a RGB-D camera (Kinect v2 Sensor) towards a potential use for close-range 3D modeling". In: *Remote Sensing* 7.10 (2015), pp. 13070–13097. ISSN: 20724292. DOI: 10.3390/rs71013070.

[18] Shahram Izadi et al. "KinectFusion". In: (2011), p. 559. DOI: 10.1145/2047196.2047270.

[19] STEMMER IMAGING. *3D time of flight cameras*. 2020. URL: https://www.stemmer-imaging.com/en-se/knowledge-base/cameras-3d-time-of-flight-cameras/ (visited on 05/02/2020).

[20] Rangappa Shreedhar. "USING LOW-COST LIGHT FIELD". In: (2018).

[21] Jiaye Zhao, Zhanwei Liu, and Baoqiao Guo. "Three-dimensional digital image correlation method based on a light field camera". In: *Optics and Lasers in Engineering* 116.December 2018 (2019), pp. 19–25. ISSN: 01438166. DOI: 10.1016/j.optlaseng.2018.12.008. URL: https://doi.org/10.1016/j.optlaseng.2018.12.008.

[22] Michael W. Tao et al. "Depth from combining defocus and correspondence using light-field cameras". In: *Proceedings of the IEEE International Conference on Computer Vision* (2013), pp. 673–680. DOI: 10.1109/ICCV.2013.89.

[23] Mehrdad Teratani Panahpourtehrani et al. "3D Imaging system using multi-focus plenoptic camera and tensor display". In: *2018 International Conference on 3D Immersion, IC3D 2018 - Proceedings* (2019), pp. 1–7. DOI: 10.1109/IC3D.2018.8657863.

[24] URL: http://otvinta.com/bevel.html.

[25] Satya Mallick. *Geometry of image formation*. Feb. 2020. URL: https://www.learnopencv.com/geometry-of-image-formation/.

[26] A. Molder et al. "Laser line detection with sub-pixel accuracy". In: *Elektronika ir Elektrotechnika* 20.5 (2014), pp. 132–135. ISSN: 13921215. DOI: 10.5755/j01.eee.20.5.7114.

[27] Computer Engineering and Forbes Avenue. "IN MESH REPRESENTATION Cha Zhang and Tsuhan Chen". In: *Virtual Reality* (), pp. 1–4. URL: http://chenlab.ece.cornell.edu/Publication/Cha/icip01%7B%5C_%7DCha.pdf.

[28] n.a. *Camera Calibration and 3D Reconstruction*. n.a. URL: https://docs.opencv.org/2.4/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html?highlight=calibratecamera.

[29] n.a. *Distortion types*. n.a. URL: https://es.mathworks.com/help/vision/ref/cameraintrinsics.html.