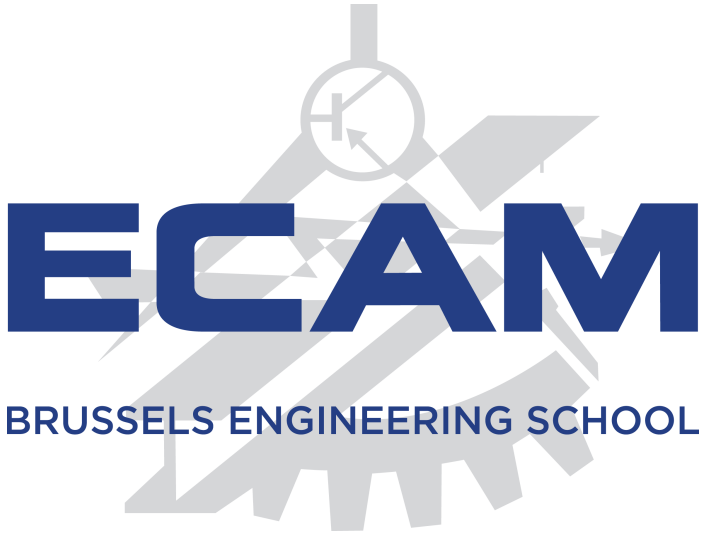


ECAM Brussels Engineering School

2021-2022

Robotic project :

Sid - 3D Scanner

Nicolas	**SAMELSON**	17288
Raphaël	**JADIN**	21295
Gabriel	**LOHEST**	16205

December 29, 2021

Contents

1 Introduction

The market of the 3D scanners being expensive, we focused ourselves on an open-source product that is affordable for end users. As the choice of the components, and their compatibility, is large, it will be possible to build with a tight budget, or aim for quality.

The scanner would be able to scan objects of maximum size of 10cm in diameter, and 20 cm in height.The device would then produce a 3D file, ready to be 3D printed or imported into any CAD software.
The scanner is equipped with a rotating plate, on which the object to be scanned is placed. The components used to scan are two cameras and a line laser. Two Raspberry Pi's will be used to do all the processing.

2 Technologies

To scan an object, there are several technologies, so we initially narrowed it down to two of them :

- The LIDAR or Light Detection And Ranging ;
- The stereovision.

The LIDAR technology uses laser beams (which are safe for the human eye) to create a 3D representation of one's surroundings. The LIDAR sends out a pulse of light and measures the time it takes for the light to return to the scanner, which enables it to calculate the surrounding depth information. It is widely used in the autonomous car industry, agriculture, biology and geology.

The principle of stereovision is the calculation of depth based on the disparity between 2 images of an object. The two images are taken at a different angle, and knowing the distance between the two cameras, we can calculate the distance of a point. For the case of a 3D scanner, we will not calculate a point, but a vertical slice of the object at a time. This slice can be highlighted with a laser.

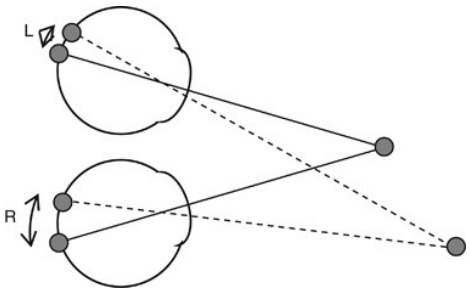


Figure 1: Principle of stereovision

It is also possible to use stereovision with a single camera, by placing two prisms in front of the camera, and 2 mirrors on either side.

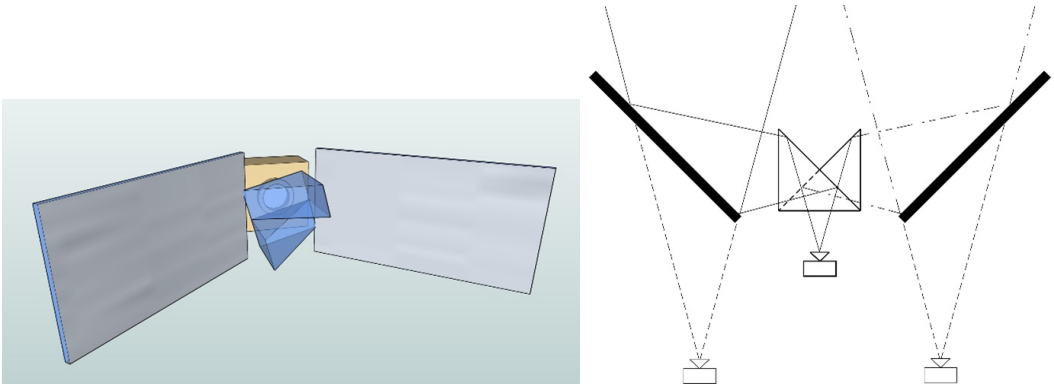


Figure 2: Stereovision with mirrors

2.1 Comparison and choice of the technologies

The LIDAR technology is the easiest to implement. In fact, it requires little math, and no calibration (unlike the stereovision method) to get distances. However, a sensor which can accurately detect objects within a range of less than one meter is difficult to find at an affordable price.

Therefore, we looked into stereovision technology.

For the cameras to maintain a high level of accuracy in dimly lit environments, we can use a laser. Moreover, it makes it easier to determine the region to be studied. In this regard, it is possible to use one or two cameras in several ways.

- One camera and two prisms and two mirrors;
- One camera and one laser;
- One camera and two lasers;
- Two cameras and one laser.

The first way is quite complex to set up, and makes it expensive to get a good image quality, due to the prisms and mirrors that come into play.

Then, it is possible to do it with a camera and a laser. However, this requires an additional stepper motor, in order to have two "false" angles of view, thanks to the stepper that moves the laser on the object at a precise angle.

It is also possible to use a camera and 2 lasers instead of one laser and an additional stepper. This allows, compared to the previous solution, to be faster, because the 2 lines of lasers are found on each image. In addition, it has one less moving part, therefore it should be more accurate.

Finally, stereovision using two cameras and a laser is advantageous over a single camera and a laser, thanks to the speed and the quality of the scan. It is also easier to do the processing in order to recover the depths with two images than with one. However, the total amount of the project will be more expensive.

2.1.1 Choice of components

Component	**Name**	**Advantages**	**Disadvantages**	**Price** [euro]
Camera	Raspberry pi cam High res	12MP Adjustable focal	Need additional lens	50
Camera	Raspberry pi cam module V2.1	8MP known FOV	fixed focal	26
Stepper motor	Nema 17 10:1	Torque of 3.5Nm Step angle of 0.18°	power supply of 8V and 35V	47
Stepper motor	28BYJ1	5V	Torque: 0.034Nm Less precise	3

Table 1: Choice of components

For this project, we chose two Raspberry Pi 4, which will each have a camera plugged in. Both computer boards will be interconnected with a RJ-45 cable, and set up in a master-slave configuration. This allows the master to easily get the data from the slave, and then compute the stereovision.

We then chose an LCD touch screen to display an interactive user interface, which allows to set up, to start the scan and to save the 3D files.
We're also taking two Raspberry Pi camera Modules V2.1, because they are affordable and have a sufficient resolution for our project.

The stepper motor 28BYJ-48 has a lot of steps, but the torque is very weak. Therefore, we chose the Nema 17, which has a much higher torque, and a stepping angle of 0.18°.

Finally, a 5mW laser, with a lens of divergence of minimum 60° in line will be able to cover the entire object.

2.1.2 Power consumption research

To power all the components, we would need two different voltage sources. The first one, in 5V, is used to power the raspberries, cameras, the display, the laser and the A4988 driver. This results in a consumption of 19.6W (Table 2).
On the other hand, the stepper motor, will be connected to the driver, and would need more voltage. We chose 12V as the second voltage, because it is easy to find an AC/DC converter that outputs those two voltages.

As we can see on Figure 3, the driver has 2 power connections : VDD in 5V and VMOT between (8-35V). VDD is used to power the logic components and drains up to 1.75W.
On the other hand, VMOT is used to power the stepper motor. The max current the driver can drain is 2A per coil, thus 4A in total. We are going to connect an power supply of 12V, resulting in 48W in power consumption in the worst case.
However, we will set the driver to drain a maximum of 1A per coil, resulting in a consumption, for our use case, of 24W.

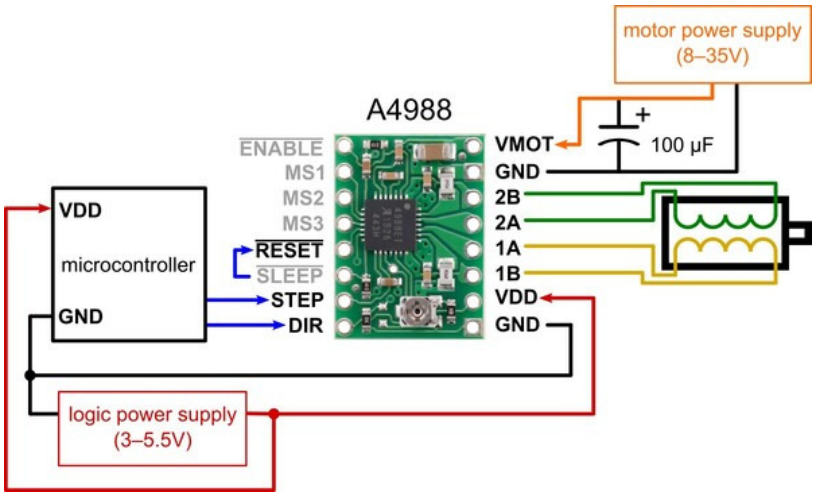


Figure 3: A4988 driver

Thanks to those results, we can choose the power supplies we would need for this project. We thus chose an AC/DC power supply with 2 outputs: one at 5V (5A) and the other at 12V (2A). This allows us to have a bit of headroom for the components connected in 5V, as well as the maximum power the driver would require (See BoM at Appendix C).

Component	Consumption
Raspberry pi 4	2 * 5.5W
Raspberry pi cam v2	2* 1.4W
11" DSI display	4W
A4988 driver	1.75W
Red line	5mW
TOTAL (in 5V)	19.6W
A4988 driver (max)	48W
Use case	24W
TOTAL (in 12V)	24W

Table 2: Consumption

2.1.3 Sizing

Having chosen our components, we are able to dimension the prototype.

First, we use cameras whose maximum field of view is 62.2° (if positioned vertically). By trigonometry, we find that the camera must be at least 166mm from the object, if it is 200mm high.
To have a margin and ensure to take the whole object, the cameras will be placed at 200mm from the closest point of the object.

Then, the horizontal field of view being 48.8°, the cameras can be off-centered by a maximum of 90mm (beyond that, the red laser line is no longer in the field of view). The more they are off-centered, the more precise we can be to the calculation the depth of the points. The cameras will therefore be off centered by 75mm each.

The turntable having a diameter of 200mm, the cameras at 200mm from it and the Raspberry Pis and the screen behind the cameras, our prototype would measure about 500*200*150mm.

3 Measurement logic

3.1 Theory

In order to compute the depth, we are going to use the principle of triangulation.
The depth of the point P is the distance z which we can express as $z = f + d$. We already know f from our camera's specifications (the focal length) but we need to compute the distance d.
In order to do that, we will use the angle θ_l and θ_r. Indeed, as we know the distance b we will be able (thanks to the tangent of the angles θ_l and θ_r) to find the distance d. We now need to find the angles θ_l and θ_r. To find them we use the field of vision of the camera.

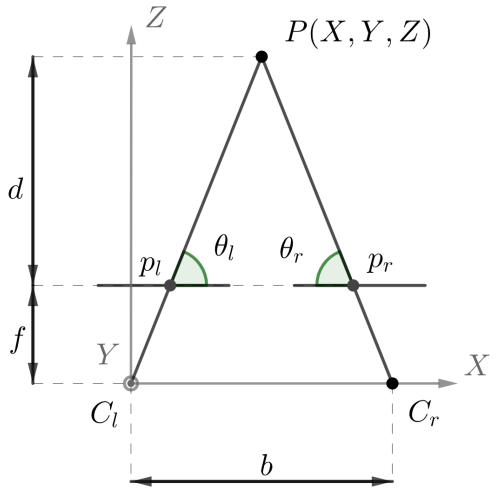


Figure 4: Principle of triangulation

Let's define d_{im} as the distance f, expressed in pixels. If our frame is of size (W_{frame}, H_{frame}) where W_{frame} and H_{frame} are given in pixels, and if we define ϕ as being the camera's field of view, we can say that $\phi_1 = \phi_2 = \frac{\phi}{2}$.

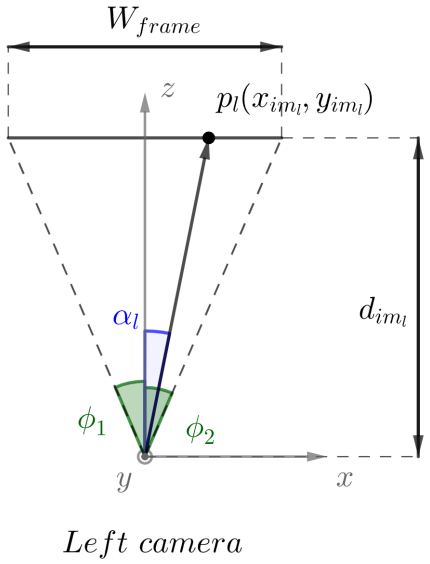


Figure 5: Field of vision of the camera c1 (left one)

We can find the distance d_{im} thanks to the following relation :

$$tan(\phi_2) = \frac{\frac{W_{frame}}{2}}{d_{im}} \Leftrightarrow tan(\phi_2) = \frac{W_{frame}}{2*d_{im}} \Leftrightarrow d_{im} = \frac{W_{frame}}{2*tan(\phi_2}$$

Now that we know the distance d_{im} we can compute the angle α for every point $p_{im} = (x_{im}, y_{im})$. Indeed we can say that :

$$tan(\alpha) = \frac{W_{frame} - x_{im}}{d_{im}} \Leftrightarrow \alpha = atan(\frac{W_{frame} - x_{im}}{d_{im}})$$

We can now find the angle angle θ_l and θ_r. If we define α_l and α_r as being α for the left and right cameras respectively, we can see that :

$$90 = \alpha_l + \theta_l \Leftrightarrow \theta_l = 90 - \alpha_l$$

$$90 = \alpha_r + \theta_r \Leftrightarrow \theta_r = 90 - \alpha_R$$

With the same method, we can find the angle θ_r. Now that we have θ_l and θ_r, we can compute the distance d. Looking at Figure 4, we can then say that:

$$b = \frac{d}{tan(\theta_l)} + \frac{d}{tan(\theta_r)} \Leftrightarrow b = \frac{d*\theta_r + d*\theta_l}{\theta_l * \theta_r} \Leftrightarrow b = \frac{d*(\theta_l + \theta_r)}{\theta_l * \theta_r} \Leftrightarrow d = \frac{b*\theta_l*\theta_r}{\theta_l + \theta_r}$$

Having found d, we can now compute z which is :

$$z = f + d \Leftrightarrow z = f + \frac{b*\theta_l*\theta_r}{\theta_l + \theta_r}$$

By replacing θ_l and θ_r by their expressions with α_l and α_r we find :

$z = f + \frac{b*(90-\alpha_l)*(90-\alpha_r)}{(90-\alpha_l)+(90-\alpha_r)}$

By replacing α_l and α_r by they expressions with the coordinates of the points $p_{im_l} = (x_{im_l}, y_{im_l})$ and $p_{im_r} = (x_{im_r}, y_{im_r})$ we can say :

$$z = f + \frac{b*(90-atan(\frac{W_{frame}-x_{im_l}}{d_{im_l}})*(90-atan(\frac{W_{frame}-x_{im_r}}{d_{im_r}})}{(90-atan(\frac{W_{frame}-x_{im_l}}{d_{im_l}})+(90-atan(\frac{W_{frame}-x_{im_r}}{d_{im_r}})}$$

By applying the same method, we are able to compute the Y coordinates of the point, which is why it will not be shown here.

4 Resolution and error on depth estimation

4.1 Resolution

The output resolution of the scan can be seen as the number of points of the 3D object model. Thus, the resolution depends on the number of pixels we are able to compute for each step, as well as on the number of steps in a full scan rotation. We can calculate this as follows:

$R = s * (W_{laser} * H_{laser})$

where

- R is the resolution, in number of points
- s is the number of steps in a full scan rotation
- W_{laser} is the width of the laser line, in pixels
- H_{laser} is the height of the laser line, in pixels

4.2 Error computation on depth estimation

To compute the error on the depth measurement we use the propagation of error method. We can say :

$\Delta z = |\frac{\partial z}{\partial x_l}| + \Delta x_l + |\frac{\partial z}{\partial x_r}| + \Delta x_r$

$$\begin{aligned}\Delta z &= \left|\frac{b*d*(tan^{-1}(\frac{W_{frame}-x_r}{d_{im}})-90)^2}{(d^2+(W_{frame}-x_l)^2*(-tan^{-1}(\frac{W_{frame}-x_l}{d_{im}})-tan^{-1}(\frac{W_{frame}-x_r}{d_{im}})+180)^2}\right| + \Delta x_l + \\ &\quad \left|\frac{b*d*(tan^{-1}(\frac{W_{frame}-x_l}{d_{im}})-90)^2}{(d^2+(W_{frame}-x_r)^2*(-tan^{-1}(\frac{W_{frame}-x_r}{d_{im}})-tan^{-1}(\frac{W_{frame}-x_l}{d_{im}})+180)^2}\right| + \Delta x_r\end{aligned}$$

The calculations for Y are similar, since the Y coordinates are computed in a similar fashion.

5 Post-processing

Once we have the world coordinates of each point, we need to bring them together in order to build a point cloud. We will subsequently convert this point cloud into a polygon mesh that can be used for 3D printing.

To build a point cloud, we need to go through all the shots resulting from a full rotation. Each shot around the object will produce an array of relevant pixels. We then have to process those pixels, and for each one of them, we need to calculate the corresponding point in the *obj* coordinate system, whose origin will be at the center of the turntable.

For a point $P = (X, Y, Z)$, whose coordinates Y and Z have previously been evaluated thanks to stereovision, we have to establish its coordinates according to the *obj* space, whose origin is the center of turntable.
In order to compute the angular value of the polar coordinates, we need to know the number of preceding steps since the start of the rotation, as well as the angle covered by one step of the motor. We can define N_s as the number of steps performed by the motor since the start of the rotation, and ϕ_s as the angle covered by one step of the motor.

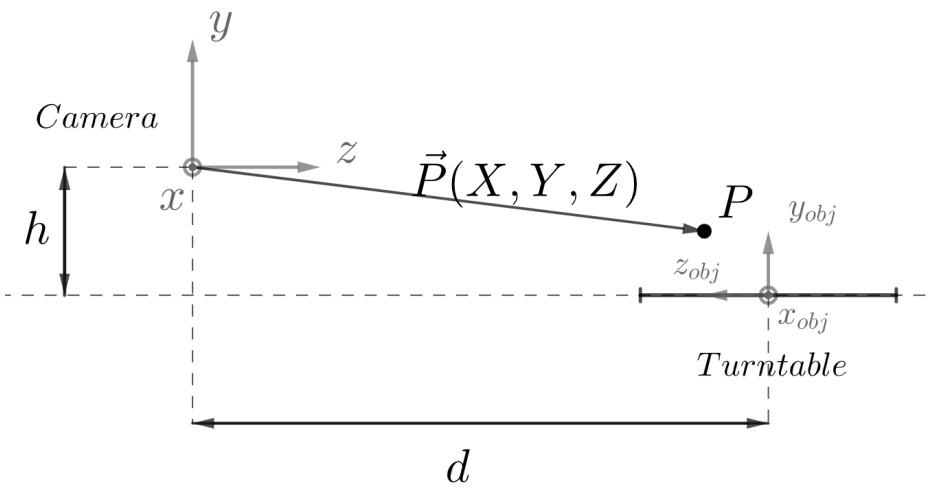


Figure 6: object coordinate change

We can then define P', as the corresponding point to P, in the *obj* coordinate system:

$P' = (r, y_{obj}, \theta)$

where

- $r = d - z$
- $y_{obj} = h - Y$
- $\theta = N_s * \phi_s$

We can convert the polar coordinates to Cartesian coordinates as follows:

$$P'_{cart} = \begin{pmatrix} x_{obj} & = & r * sin(\theta) \\ y_{obj} & = & y_{obj} \\ z_{obj} & = & r * cos(\theta) \end{pmatrix}$$

Once we have the point cloud, we need to convert it into a polygon mesh. In order to achieve that, we will use the Python library PyMeshLab.
After that, the output should be able to be run into any file converter, so that the scanned object can be expressed into a wide range of 3D file types. The PyMeshLab library also provides several ways to enhance the object model, such as smoothing functions and other useful post-processing features, like the possibility to eliminate a portion of the data, in order to find a good compromise between quality and file size.

References

[1] Aish Dubey. "Stereo vision—Facing the challengesand seeing the opportunities for ADAS applications". In: (2020). URL: https://www.ti.com/lit/wp/spry300a/spry300a.pdf (visited on 12/28/2021).

[2] Nicholas Ayache and C. Hansen. "Rectification of images for binocular and trinocular stereovision". report. INRIA, June 1988. URL: https://hal.inria.fr/inria-00075694 (visited on 12/28/2021).

[3] Clayton Darwin. *Distance (Angles+Triangulation) - OpenCV and Python3 Tutorial - Targeting Part 5*. 2018. URL: https://www.youtube.com/watch?v=sW4CVI51jDY (visited on 12/26/2021).

[4] Cosmetic bag. *DIY 3D Laser Scanner Using Arduino - Make:* Make: DIY Projects and Ideas for Makers. Nov. 30, 2001. URL: https://makezine.com/2013/05/24/new-project-diy-3d-laser-scanner-using-arduino/ (visited on 12/28/2021).

[5] DroneBot Workshop. *Getting Started with LIDAR*. 2018. URL: https://www.youtube.com/watch?v=VhbFbxyOI1k (visited on 12/20/2021).

[6] dtrewren. *Ciclop 3D Scanner (BQ & Horus)*. Instructables. URL: https://www.instructables.com/Ciclop-3D-Scanner-BQ-Horus/ (visited on 12/29/2021).

[7] FU-Fighters. "Stereo vision with a single camera". In: (2004). URL: http://www.inf.fu-berlin.de/lehre/SS06/Robotik/stereo_vision_newest.pdf (visited on 12/20/2021).

[8] Enric Meinhardt Gabriele Facciolo Carlo de Franchis. "MGM: A Significantly More Global Matching for Stereovision". In: (2015). URL: http://dev.ipol.im/~facciolo/mgm/mgm.pdf (visited on 12/29/2021).

[9] *In-Depth: Control 28BYJ-48 Stepper Motor with ULN2003 Driver & Arduino*. Last Minute Engineers. Dec. 22, 2019. URL: https://lastminuteengineers.com/28byj48-stepper-motor-arduino-tutorial/ (visited on 12/20/2021).

[10] Saad Merrouche et al. "Objective Image Quality Measures for Disparity Maps Evaluation". In: *Electronics* 9.10 (Oct. 2020), p. 1625. URL: https://www.mdpi.com/2079-9292/9/10/1625 (visited on 12/23/2021).

[11] *Object Files (.obj)*. URL: http://paulbourke.net/dataformats/obj/ (visited on 12/29/2021).

[12] *Objet 3D (format de fichier)*. In: *Wikipédia*. Page Version ID: 171551342. June 1, 2020. URL: https://fr.wikipedia.org/w/index.php?title=Objet_3D_(format_de_fichier)&oldid=171551342 (visited on 12/20/2021).

[13] *OpenCV: Depth Map from Stereo Images*. URL: https://docs.opencv.org/4.x/dd/d53/tutorial_py_depthmap.html (visited on 12/29/2021).

[14] M. Pesce et al. “A Low-cost Multi Camera 3D Scanning System for Quality Measurement of Non-static Subjects”. In: *Procedia CIRP*. 3rd CIRP Global Web Conference - Production Engineering Research 28 (Jan. 1, 2015), pp. 88–93. ISSN: 2212-8271. DOI: 10.1016/j.procir.2015.04.015. URL: https://www.sciencedirect.com/science/article/pii/S2212827115002772 (visited on 12/23/2021).

[15] *Pololu - A4988 Stepper Motor Driver Carrier*. URL: https://www.pololu.com/product/1182 (visited on 12/26/2021).

[16] Stephen Se and Piotr Jasiobedzki. “Stereo-vision based 3D modeling and localization for unmanned vehicles”. In: *International Journal of Intelligent Control and Systems* 13 (Mar. 18, 2008). URL: https://www.researchgate.net/publication/228618177_Stereo-vision_based_3D_modeling_and_localization_for_unmanned_vehicles.

[17] Super Make Something. *DIY 3D Scanner (Arduino, 3D Printing, PCB Design, Stepper Motors, IR Sensing) - Super Make Something*. Mar. 14, 2016. URL: https://www.youtube.com/watch?v=-qeD2__yK4c (visited on 12/28/2021).

[18] The Coding Lib. *Depth Estimation Using Stereo Vision - Computer Vision Project with OpenCV C++ Code*. 2021. URL: https://www.youtube.com/watch?v=snJVyfl9ZMg (visited on 12/20/2021).

[19] *The FabScan Project - Media Computing Group - RWTH Aachen University*. URL: https://hci.rwth-aachen.de/fabscan (visited on 12/29/2021).

[20] *What is a stereo vision camera? — Camera blog*. URL: https://www.e-consystems.com/blog/camera/camera-board/what-is-a-stereo-vision-camera/amp/ (visited on 12/20/2021).

[21] William Lovegrove. *Single-camera stereo vision for obstacle detection in mobile robots*. 2007. URL: https://spie.org/news/0911-single-camera-stereo-vision-for-obstacle-detection-in-mobile-robots?SSO=1 (visited on 12/20/2021).

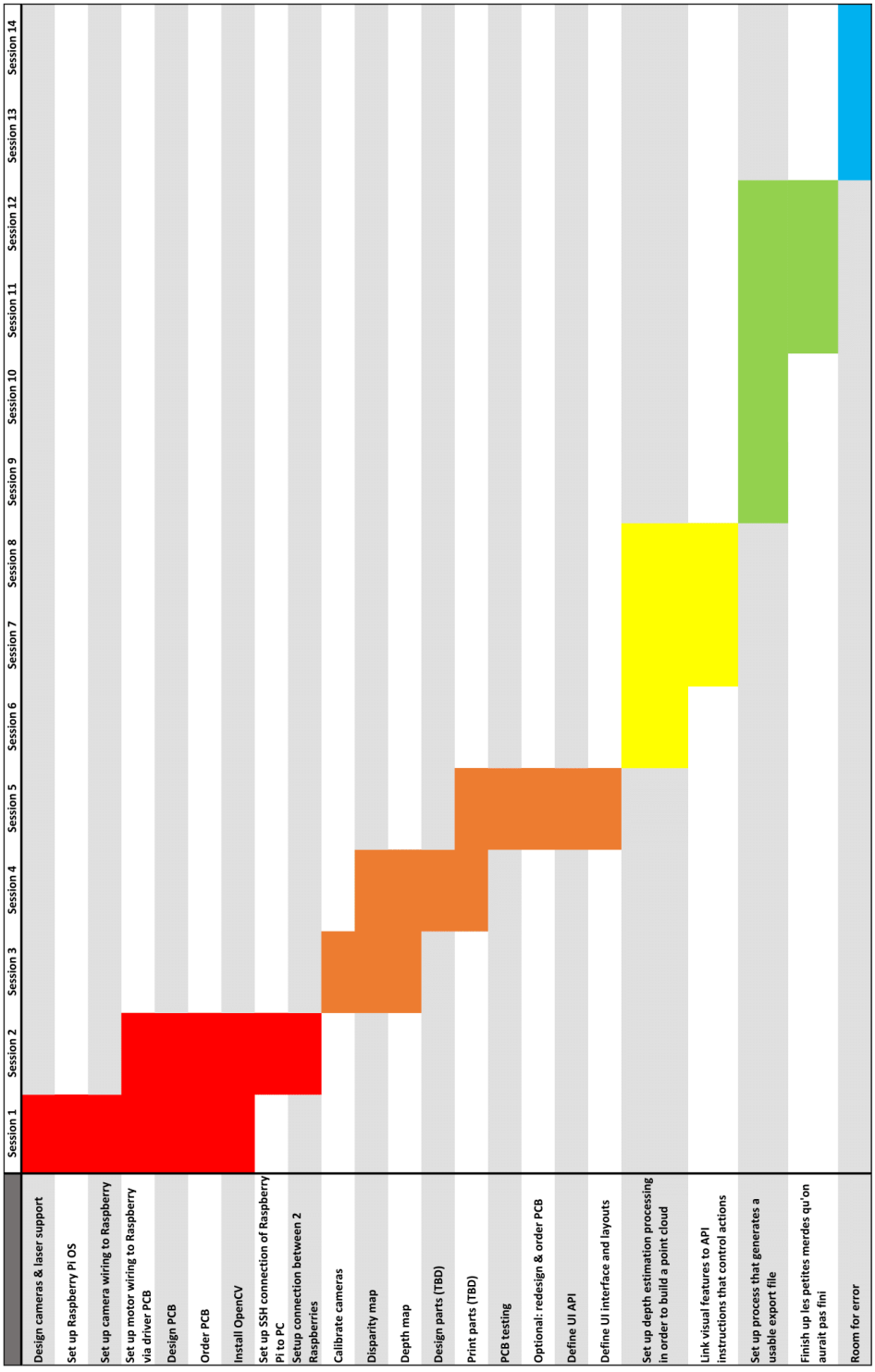
[22] Zhiyi Zhang Zhihua Lv. “Build 3D Scanner System based on Binocular Stereo Vision”. In: (2012). URL: https://cie.nwsuaf.edu.cn/docs/20170614173931092961.pdf (visited on 12/20/2021).

Appendices

Appendix A Raci Matrix

Project activity/deliverable	Raphael	Gabriel	Nicolas
3D design: camera & laser support	I	R	A
Rasperry: Setup Raspberry pi OS	A	R	I
Rasperry: Setup motor wiring to Raspberry via driver PCB	A	R	I
Rasperry: Setup camera wiring to Raspberry	A	I	R
Design PCB	A	I	R
Order PCB	I	A	R
Rasperry: Install OpenCV and other libraries on the Raspberry	R	A	I
Raspberry: setup connection between the raspberries	R	I	A
Rasperry: Setup SSH connection of Raspberry	R	A	I
Stereovision: calibrate cameras	R	A	I
Stereovision: disparity map	R	I	A
Stereovision: depth map	R	A	I
3D design: design parts (TBD)	I	R	A
3D design: print parts (TBD)	I	R	A
PCB: PCB testing	A	I	R
PCB: (Optional) redesign and order PCB	A	I	R
UI: design API	A	I	R
UI: define interface layout and controls	R	I	A
Image postprocessing: setup depth estimation processing	R	A	I
Image postprocessing: build point cloud	A	R	I
UI: link visual features to API instructions	A	I	R
3D exporting: setup process to generate 3D file	R	A	I
Adjustments and correcting	I	A	R

Appendix B Gantt Diagram



Appendix C Bill of Materials

Name	**Price** [euro]
Raspberry Pi Camera Module V2.1	54
Stepper Nema 17 10:1	47
driver a4988	7
Red line 5mW HLM1230	9
Alimentation AC/DC 5V et 12V	27
Raspberry pi 4 * 2	104
Display 11”	72
PCB	30
TOTAL	350