ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Semester Project

Stereo vision in self-reconfigurable modular robots

EPFL Biorobotics Laboratory

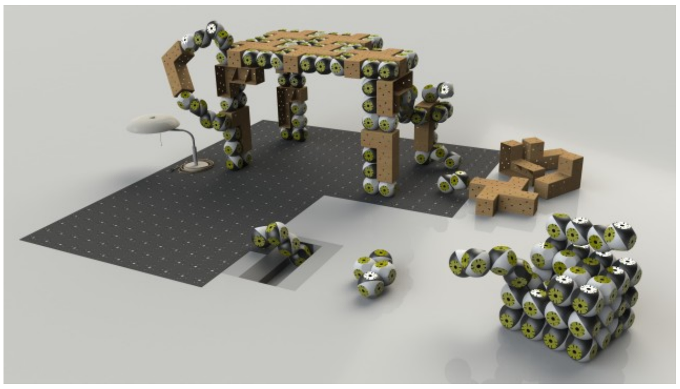
Written by :
Hugo Kohli
Assistant :
Mehmet Mutlu
Professor :
Auke Ijspeert

June 8, 2018

Contents

1 Introduction

The objectives of this study is to use the Roombots and to allow the detection and the measurement of a gap width at an unknown distance. The Roombots are a type of self-reconfigurable modular robots that are meant to be use for room layout, which includes moving, self-assembling and self-reconfiguring different types of furnitures (see Fig.1a). This kind of robot is opposed to some more classic non reconfigurable robot as it possess a greater flexibility and robustness when facing obstacles and failure. Each unit is capable of grabbing another unit thanks to connectors, so depending on the task that needs to be solved, the robot can arbitrary change its structure . The purpose of this modular robot in the organization of the layout of a room is to be able to create different types of furniture depending on the specific requirements of the room. This would typically be an application to help the people with reduced mobility.


(a) Example of Roombot application [9]

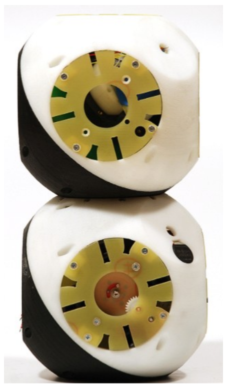

(b) Roombot module [9]

Figure 1: Présentation of the Roombots

The example of a possible collaboration between modules, which is presented in this work, is the action to cross a gap. Indeed each module is composed by two spheres with a diameter of 11 cm (see Fig.1b) which means that gaps larger that 22 cm cannot be crossed by a module working alone. However, if the module has a good knowledge of the gap's size, he can decide either to cross it alone if it is smaller than 22 cm, or assemble with one or more other modules to cross the obstacle.

1.1 Goal

At this day the Roombots do not have the ability to evaluate the size of a gap. The method described in this study is the stereo vision. Indeed, while having only one camera gives no information about the depth, integrating a second one can make it possible to measure the distance between the camera and the observed scene. The initial idea was to integrate cameras into two separated and fixed modules, that would measure the size of the possible gaps in front of another independent working module. In order to experiment the use of the stereo vision, two similar Ximea cameras, with the same lenses are used and connected to a central PC by USB cables. The processing of the images is done in the language `C++`, using functions from the library `OpenCV` which is specially designed for image processing applications. The stereo camera setup is shown on Fig.2.

The goal of this work is to evaluate the possibility of using stereo vision in this particular application by analyzing the range of use and the limitations of this method. This study is composed by a first section which explains the different theoretical aspects of the concept used, detailing each time the corresponding `OpenCV` functions used, and the second section focuses on the practical implementation.

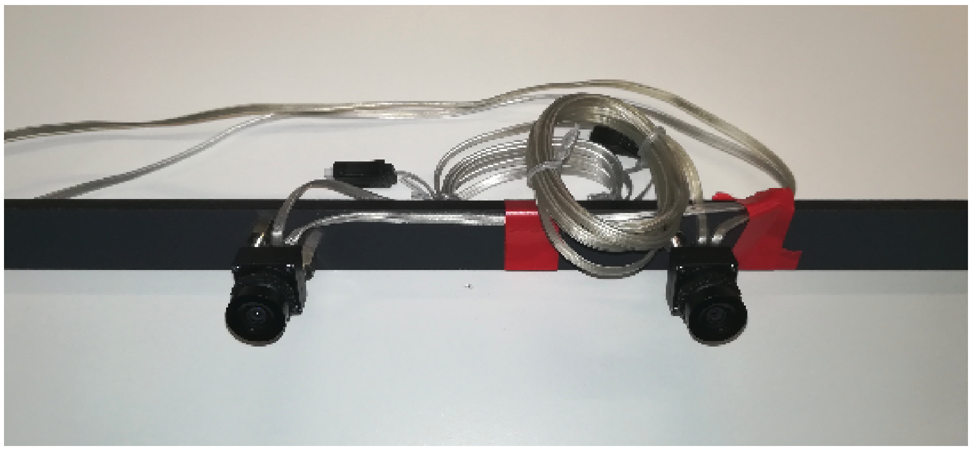


Figure 2: Stereo camera using two Ximea cameras

1.2 Motivation

Even if this project focuses on one very specific application, the use of the stereo vision can be extrapolated to concrete applications such as the mobile robotic. Demining robots, city-dweller car, drones are different examples of autonomous vehicles that could typically use the stereo vision to detect objects, to evaluate the size of obstacles or simply to measure distances.

1.3 State of the art

Nowadays the computer stereo vision is commonly use for many applications especially in robotics. As the stereo cameras return a depth map and so give an approximation of the distance between the camera and the object in the surroundings, it can be a very efficient tool for obstacle avoidance when using mobile robots. Several papers present the use of the stereo vision for path planning [1, 4, 6, 7] and terrain detection [2]. This implies a real time video processing which is made possible in stereo vision thanks to the epipolar geometry (see Sec.2.3).

The knowledge of the distance between cameras and objects is also used to have a knowledge of the positions of some parts of the robot, such as the case of a humanoid robot, presented in [8].

A specific paper has been published by S. Malssiotis and M.G.Strintzis and treats about the dimensional inspection of holes [3]. They are using stereo vision to replace expensive previous inspection methods such as the non-contact laser sensing. The principle is here to exploit CAD informations to evaluate the 3D measurements against required tolerances. The objective of including stereo vision into the Roombots is quite similar to this project as they need to measure the size of holes, excepted one big difference. In the case of the described paper, the position of the hole is known thanks to the CAD files, which is not the case for the roombots, where the edges are at unknown positions. Indeed there is no comparison possible to evaluate the result. However, similar processing parts such as the Canny edge detectors have been used for both studies.

2 Theoretical part

In the following sections are described the theoretical aspects of the different steps that are used during this study.

2.1 Calibration

The first step of this work, and more generally whenever a camera is used, is to correctly calibrate the cameras. Indeed, as the two cameras used for this work are just simple pinhole, some distortion is introduced in the image. The Fig.3 shows a raw picture taken by the Ximea camera with the presence of radial distortion. In order to correct the image given by the camera, two main factors can describe its distortion, one radial and one tangential.

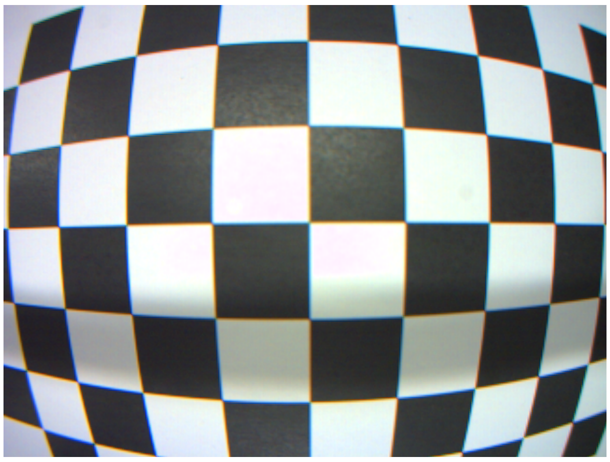


Figure 3: Radial distortion of the camera used

The radial factor appears in this first equation :

$$x_{distorted} = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$
$$y_{distorted} = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$

with x and y being the undistorted pixel and k_1, k_2 and k_3 being the radial distortion coefficients.

The tangential distortion comes from the difference of the orientation between the lenses and the imaging plane, who should be perfectly parallel in theory, and can be represented by the following formulas :

$$x_{distorted} = x + (2p_1 xy + p2(r^2 + 2x^2))$$
$$y_{distorted} = y + (p_1(r^2 + 2y^2) + 2p2xy)$$

with p_1, p_2 being the tangential distortion coefficients.

In addition to the distortion factors, the extrinsic and intrinsic parameters of the camera need to be found. The intrinsic parameters are specific to each camera, like the focal length (f_x, f_y) and the projection of the optical centers (c_x, c_y) which can all be stored in one matrix, called the camera matrix.

$$camera\ matrix = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

The transformation from the image points to the world points can be done using the following relation :

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} = camera\ matrix\ \times \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

The extrinsic parameters can vary depending on the position of the camera in the world space. They consist in two different elements. First the rotation matrix which has a size of 3×3 and allows to pass from the world coordinates to the camera coordinates (see Fig.4). Secondly the translation vector, which is composed by the components t_x, t_y and t_z containing the information of the translation between the world and the camera coordinates.

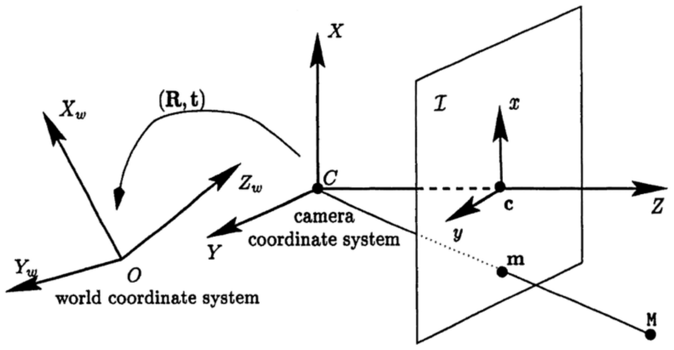


Figure 4: Rotation matrix [20]

The camera calibration consists in determining all the numerical values of the parameters mentioned previously.

The method used for this purpose is composed by the following steps. First different pictures of a known pattern are taken by the uncalibrated camera. In this case, the pattern is a 9 by 6 chessboard mounted on a rigid plate (see Fig.5) to ensure a good accuracy. Then the intersection between the squares are detected on each distorted image, and all their coordinates are stored. As the coordinates of those points are also well known in the real world space, a mathematical algorithm (see Sect.2.2.1) is able to get the distortion coefficients, the cameraMatrix, the rotational matrix and the translation vector out of all the image taken [11, 15, 19].

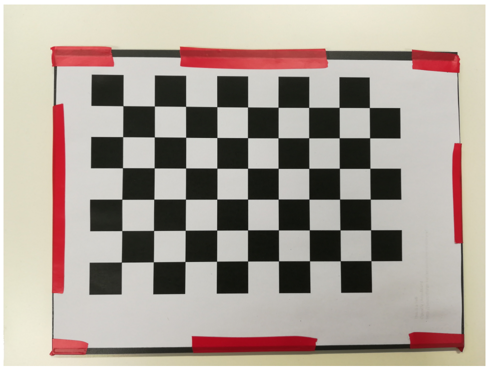


Figure 5: Pattern used for the calibration

2.1.1 OpenCV implementation

The important functions from the library `OpenCV` used for the camera calibration are explained bellow.

- **findChessboardCorners()**
  This function returns *true* each time a defined pattern is detected. It takes as an input the image and the size of the pattern to detect, and return as an ouput the coordinates of all the corners of the pattern in the image coordinate frame.

- **calibrateCamera**()
  It takes as input a vector of vectors of pattern points coordinates both in the pattern space and the image space (given by the function **findChessboardCorners()**), and return as output all the distortion coefficients, intrinsic and extrinsic parameters. The algorithm first compute the initial intrinsic parameters and estimate the initial attitude of the camera as if the intrinsic parameters have been already known, using a function called **slovelPnP()**. Once this done, an optimization algorithm (Levenberg-Marquardt) is ran to minimize the re-projection error between the projected image, computed using the previous extrinsic parameters estimated, and the actual image. The re-projection error is the sum of the squared distances between the two different images previously mentioned. It is given as an output and it is an important mean to determine the accuracy of the camera calibration. ...

2.2 Stereo calibration

Once the two camera are separately calibrated, the stereo calibration needs to be computed in order to have the relative pose of one camera referred to the other. Again the transformation between the two cameras is given by 2 matrices : R, the rotation matrix and T, the translation vector. Two more matrices are defined, E the essential matrix and F the fundamental matrix, which can also be used to find R and T.

$$E = \begin{bmatrix} 0 & -T_2 & T_1 \\ T_2 & 0 & T_0 \\ -T_1 & T_0 & 0 \end{bmatrix} \times R$$

with T_i the elements of the translation vector.

$$F = cameraMatrix2^{-T} cameraMatrix1^{-1}$$

The method is exactly the same as for the single calibration, namely multiple views of a pattern are taken, corners are detected, and the relative pose is estimated from the vectors containing the corners coordinates in the pattern and the image space [10, 11, 19].

2.2.1 OpenCV implementation

The only important function used for the stereo calibration is **stereoCalibrate()**. It takes as an input the corners coordinate in both image and pattern space for the two camera. The distortion coefficients and the camera matrices previously estimated by the calibration are also given as input. It has to be mentioned that this function allows to estimate these matrices too, but due to the high dimensionality of the parameters space and noise in the input data, the function may diverge from the correct solution, that is why it is recommended to do the calibration separately for each camera before doing the stereo calibration. As for the function

calibrateCamera(), the re-projection error is returned and is a good mean to evaluate the quality of the stereo calibration.

2.3 Depth map creation

The first step in the creation of a depth map is the rectification. Basically, the aim is to project two (or more) images onto a common image plane. As this process has several degrees of freedom, the dimensionality can be restricted using the epipolar geometry. Indeed, as shown on Fig.6, the epiline *l'* is the line which contains all the projection of the different points OX on the right plane. For example, it allows to search a matching point into the other image by searching only along this line instead of exploring every pixel. Indeed, having both camera image projected on the same plane makes all the epipolar lines parallel and simplifies the stereo correspondance. The points *e* and *e'* are the epipoles, and are also used for the rectification. They can be computed using the camera matrices, the rotation matrices and the translation vector previously estimated [16].

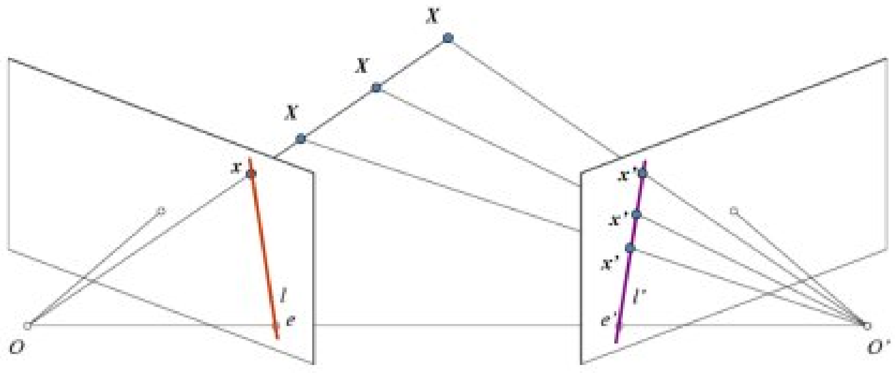


Figure 6: Epipolar geometry [16]

For the two cameras a rectification transform rotation matrix R_i that make both camera image planes the same plane and a projection matrix P_i into the rectified coordinates system which is the same for the two cameras, are computed. For horizontal stereo, the matrices $P1$ and $P2$ are shown below :

$$P_1 = \begin{bmatrix} f & 0 & cx_1 & 0 \\ 0 & f & cy & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \qquad P_2 = \begin{bmatrix} f & 0 & cx_2 & T_x * f \\ 0 & f & cy & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{1}$$

with cx_i and cy_i the center point of image i. cy is the same for both images as it is assumed that the two cameras are shifted mainly along the x-axis. The whole process is represented on Fig.7.

Once the two images are on the same plane, an algorithm is used to measure the disparity, which is the distance between two points showing the same scene on both images. Two different algorithms have been used for this work : block-matching and semi-global block-matching.

2.3.1 Block matching algorithm

This is the basic technique for finding corresponding pixels in a pair of stereo images. It consists in computing the similarity between pixels by comparing windows around pixels of interest (correlation method) as shown in Fig.8. This algorithm returns a disparity map, where each point of the map is a the distance between two corresponding pixel [13, 12].

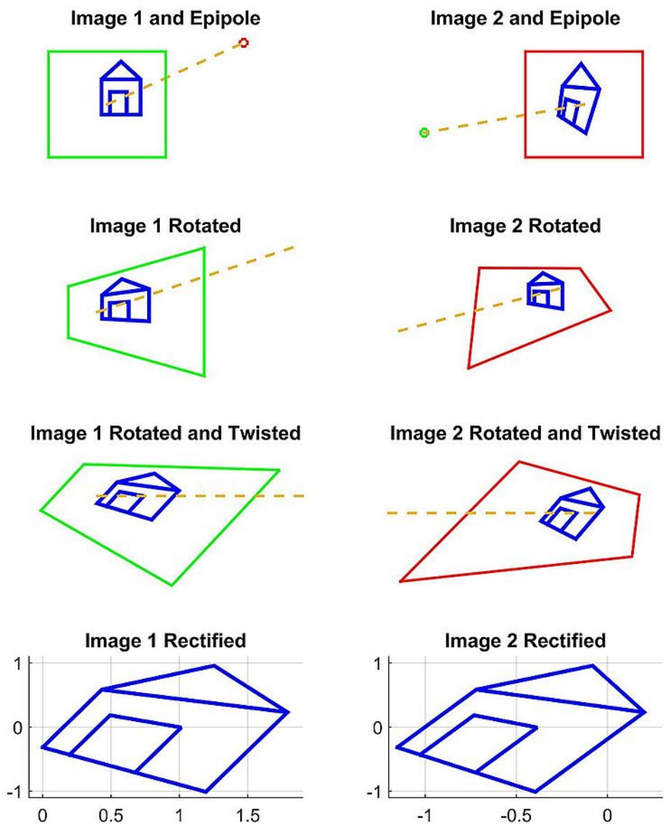


Figure 7: Rectification process [13]

2.3.2 Semi-global block matching algorithm

This technique is very close to the original one, but it differs due to some points. Some pre- and post-processing steps are included, as the uniqueness check, the quadratic interpolation, the speckle filtering and the disparity filters. It is a good trade off between accuracy and runtime and is therefore used for many practical applications. As for the previous algorithm, it returns a disparity map [21, 14].

2.3.3 From disparity to distance

To transform a disparity given for an image coordinate (x, y) into a real world 3D coordinates point (X, Y, Z), the following relation is used :

$$[X\ Y\ Z\ W]^T = Q \times \left[x\ y\ disparity(x,y)\ 1\right]^T \tag{2}$$

with Q the perspective transformation matrix, also called disparity-to-depth mapping matrix, which is computed at the rectification step.
As the origin of the real world coordinates is the point between the two cameras, the distance from the cameras can now be computed using Pythagore : $d = \sqrt{X^2 + Y^2 + Z^2}$

2.3.4 OpenCV implementation

- **stereoRectify()**
  This is the function used for the rectification step. It takes as input the camera matrices

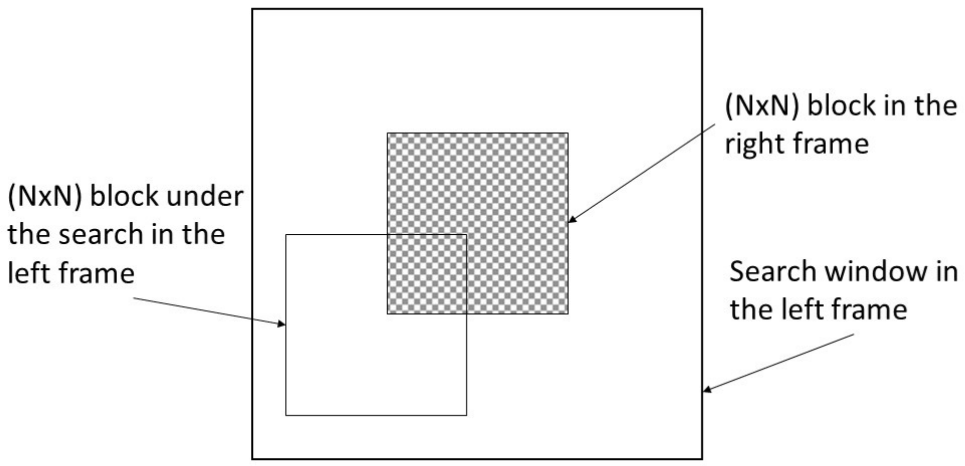

(NxN) block in the right frame
(NxN) block under the search in the left frame
Search window in the left frame

Figure 8: Block-matching algorithm [14]

and distortion coefficients of the two camera, and the rotation and translation matrices between them. As an output, it returns the rotation matrices to the new plane and the projection matrices previously described in this section. Furthermore, it returns the Q matrix, which will be used to transform the disparity into depth.

- **initUndistortRectifyMap()**
  This function computes the undistortion and the rectification transformation map, taking as input the matrices given by **stereoRectify()**. It needs to be applied to both camera, and the the output is a new camera matrix for each of them.
- **reprojectImageTo3D()**
  This function uses the matrix Q returned by **stereoRectify()** to return the location of a pixel in the 3D world from the disparity map.

2.4 Line detection

Before being able to mesure the size of a gap, the last step is the detection of the gap itself. As the gap is supposed to be rectilinear, the goal is so to detect straight lines

2.4.1 Canny edge detector

Before detecting the lines, the Canny algorithm is used to highlight the edges, which are the zones of the pictures having a high gradient. It is a very common filter used in image processing for edge detection. The first step of this algorithm is to filter out the noise, applying a Gaussian filter such as a Gaussian kernel and then to highlight the high intensity gradient of the image. This can be done in only one step, using a pair of convolution mask, the sobel filters, applied in both x and y direction.

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \tag{3}$$

The direction and the intensity of the gradient is then computed using these equations :

$$G = \sqrt{G_x^2 + G_y^2} \qquad \theta = arctan(\frac{G_y}{G_x}) \tag{4}$$

Finally, the edges are composed by all the pixels whose radiant are higher than an imposed threshold. A second smaller threshold is used for edge linking. This two threshold method is called hysteresis procedure.

2.4.2 Hough transform

To pass from edges to line, the progressive probabilistic Hough transform is applied. It is explained more in details in the section 2.4.3.

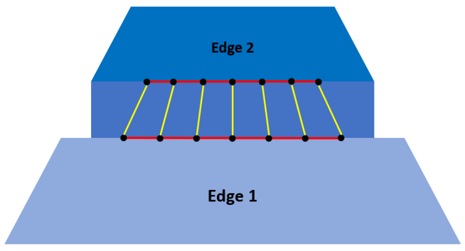
2.4.3 OpenCV implementation

- **Canny()** This is the provided function by OpenCV to perform the Canny Edge Detection. Taking the image to process, the size of the filter and the two threshold for hysteresis procedure as input, it returns a black and white image, having the same size as the initial image but containing only edges.

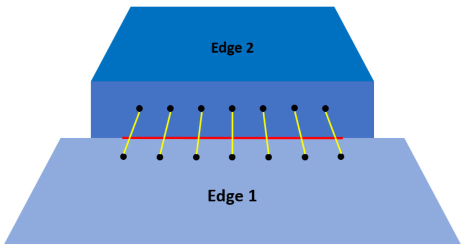
- **HoughLinesP()** This function computes a robust detection of lines based on the progressive probabilistic Hough Transform

2.5 Gap measurement

Once the edges can be correctly detected, the last step is to finally measure the width of the gap. To do so, two ideas appeared. The first one was implying to detect the two edges, then selecting some points on one line and finding the closest corresponding points on the other line. By using a re-projection of the image to 3D, simple geometry and finally an average over all the points, the distance between the two edges could theoretically be computed.
The second idea was a bit simpler. Indeed, only the first edge needed to be detected. By taking points over and just under this line, the width can be computed taking directly the corresponding distance to this points. The Fig.9 shows these two reasoning more in details. The limitations of this two ideas are explained more deeply in the Sec.3.4.



(a) First method



(b) Second method

Figure 9: Width of the gap measurement

3 Practical implementation and experiments

3.1 Calibration and stereo calibration

As explained in the introduction, the two cameras are exactly the same (Ximea) and have the same lenses and so the same focal length, which makes easier the creation of a depth map. Many trials and tests have been performed before doing the first experiments. Indeed, the calibration of the camera was a key step for the rest of this work, meaning that a bad calibration would have affected the whole process. In order to increase the quality of the calibration, more than 80 pictures of the chessboard have been taken for each camera. Then every image have been checked, selecting only the very sharp ones. All the other pictures, for example the one being blurred by some movement of the camera and the one that was overexposed have been erased (see Fig.10). In the end, each set is composed by approximatively 70 images which led to the following re-projection errors :

$$RMS_{right} = 0.209973 \qquad RMS_{left} = 0.274733$$

The same process has been respected for the stereo calibration, except that the total number of images kept for each camera is lower. Indeed, it has been observed that the RMS error was not decreasing anymore raising the number of pictures over a certain point. Therefore the set is composed by 60 stereo images, and the final re-projection error is :

$$RMS_{stereo} = 0.773306$$

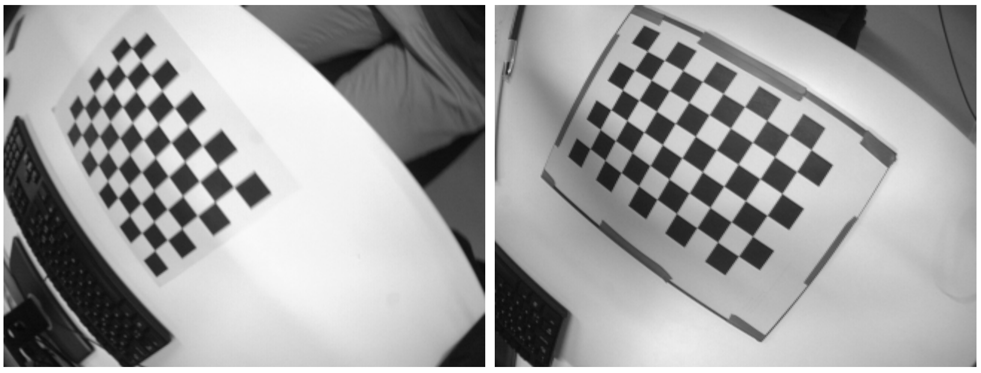
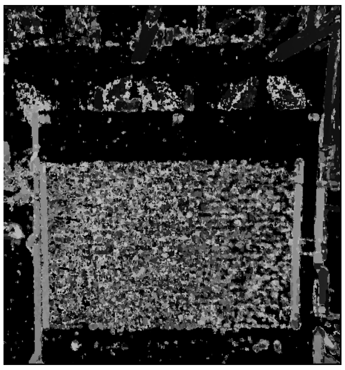

(a) Blurred calibration picture (b) Bright calibration picture

Figure 10: Selection of the good calibration pictures

The stereo calibration has been initially computed with the two cameras spaced 22 cm. The results were giving $RMS_{stereo} > 2$ which was leading to very poor quality depth map. As the cameras are meant to be integrated in the roombots which have a diameter of 11 cm, the distance between the two cameras is supposed to be a multiple of 11. The best RMS_{stereo} is the one given previously and has been obtained with the cameras spaced 11 cm, and this distance has been kept for the rest of this study.

3.2 Range of use of the distance measurement

Once the calibration and the stereo calibration have been successfully performed, the depth map can be computed as explained in Sec.2.3. Here again, before doing a proper experiment, many small tests and adjustments have been performed in order to identify the effects of the different factors. The first thing was to determine which was the block matching algorithm to use. As shown on Fig.11, the first visual impression may not be sufficient to determine which one is giving the better results.


(a) BM algorithm

(b) SGBM algorithm

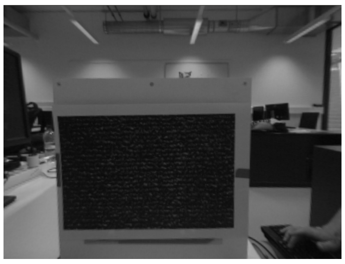
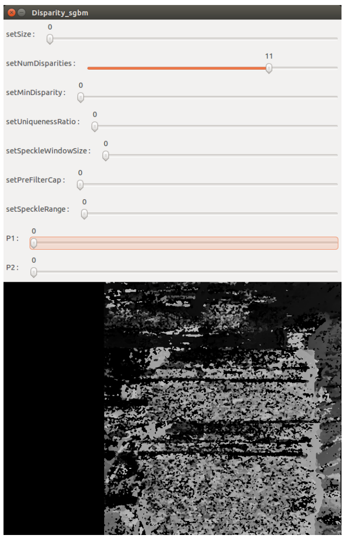

(c) Image seen by the right camera

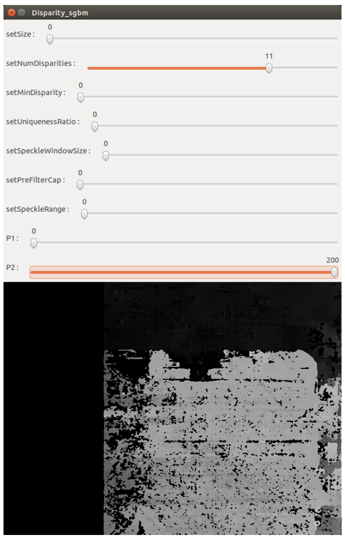
Figure 11: Differences between block matching algorithms

However, after trying to measure several distance using the simple block matching algorithm, it came out that it is was far from being accurate. Therefore the semi-global block matching as been kept and further more analyzed. For this algorithm the main parameters are the following : P1, P2, numDisparities, size of the window. As shown on Fig.12, a simple visual control allowed to identify the smoothing effect of the parameters P1 and P2. The scene is showing a flat plate covered with random pattern texture as shown on Fig.11c. Therefore, P1 have been set to its maximal value and P2 set to 20 because these parameters were showing the smoothest depth map. As this was giving very satisfying results, this two values have been kept for the rest of the experiments. However, even after hours of parameters tuning, the best depth map obtained looked still very noisy.


Disparity_sgbm

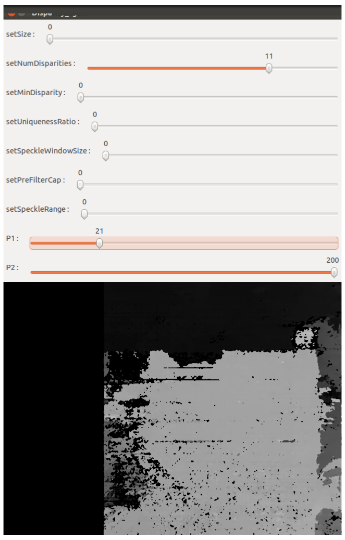
setSize : 0
setNumDisparities : 11
setMinDisparity : 0
setUniquenessRatio : 0
setSpeckleWindowSize : 0
setPreFilterCap : 0
setSpeckleRange : 0
P1 : 0
P2 : 0

(a) No P1 and no P2


Disparity_sgbm

setSize : 0
setNumDisparities : 11
setMinDisparity : 0
setUniquenessRatio : 0
setSpeckleWindowSize : 0
setPreFilterCap : 0
setSpeckleRange : 0
P1 : 0
P2 : 200

(b) No P1 and maximal P2


setSize : 0
setNumDisparities : 11
setMinDisparity : 0
setUniquenessRatio : 0
setSpeckleWindowSize : 0
setPreFilterCap : 0
setSpeckleRange : 0
P1 : 21
P2 : 200

(c) High P1 and maximal P2

Figure 12: Effect of the smoothness parameters P1 and P2 on the depth map

The goal of the first experiment was to determine the range of distances that the stereo cameras were able to calculate, and which value of "Maximal disparity value" was giving best results. As explained in the introduction, the length of the gap is assumed to take discrete values which are multiples of 11 cm. This length is computed measuring two distances, which are the distances from each edge to the cameras. The difference measured between this two distances is the width of the gap. If it is bigger than $\alpha \cdot 11 + \frac{11}{2}$, the gap will be considered to be $(\alpha + 1) \cdot 11$ cm wide (see Fig.13). Therefore, a distance can be measured with a precision of $\pm\frac{11}{4}$ and still be valid as shown in the following example :

$$d_1 = 22 - \frac{11}{4} \qquad d_2 = 33 + \frac{11}{4} \qquad gap = d_2 - d_1 = 35.75 - 19.25 = 16.5$$

In this example, the discrete length of the gap calculated is still 11 cm.

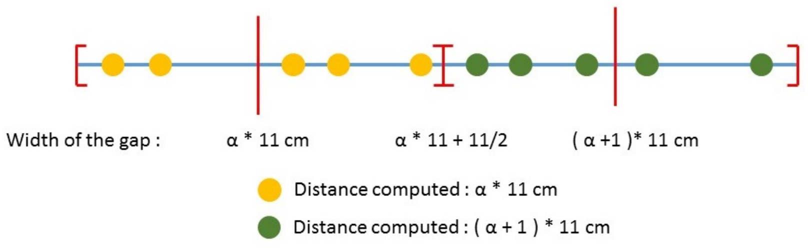

Width of the gap :
α * 11 cm
α * 11 + 11/2
(α +1)* 11 cm
● Distance computed : α * 11 cm
● Distance computed : (α + 1) * 11 cm

Figure 13: Discrete widths explanation

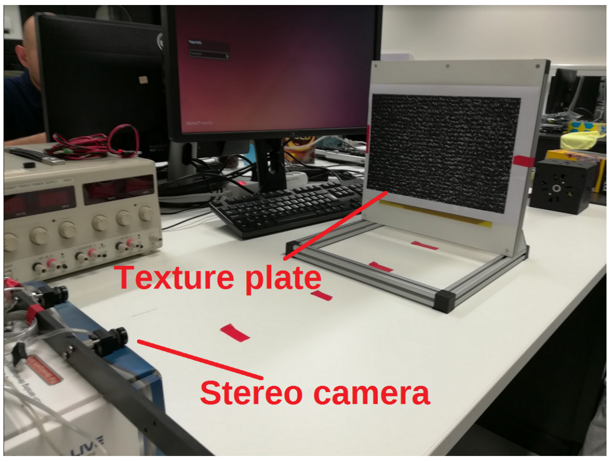


Figure 14: Setup for the first experiment

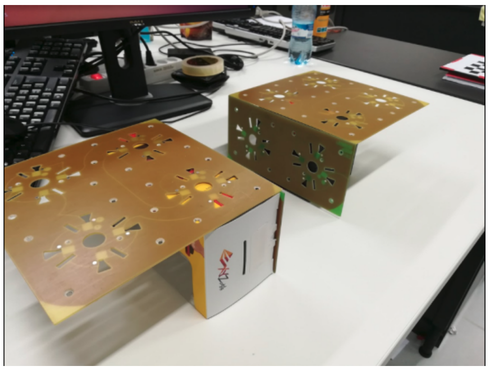
The setup of this experiment is shown on Fig.14. A texture has been applied on the measuring surface making it easy for the algorithm to find matching points and so calculate their disparity. In order to remove some noise, a closing morphological operation has been applied before doing the measurement (see Sec.3.3). For each maximal disparity value (which muss be dividable by 16 by definition of the OpenCV function), distance measurements have been done from distance between 11 cm and 110 cm and being multiples of 11, doing each measurement ten times. The results are shown on Fig.23 in the annexe. The x axis of these graphs represents the distance at which the measuring plate was set, as the y axis shows the means of the distances computed for each distances. The measurements are shown with their standard deviation.

The first thing that came out of these graphs is that the stereo camera is not giving satisfying results for distances bigger than 90 cm whatever the maximal disparity value chosen is. Thereby the upper bound of the range of validity of the stereo camera is assumed to be at 90 cm. The second observation that has been made is about the lower bound. It seems clear that the smaller the maximal disparity value is, the worst the measurement is for short distances, which can be explained easily. Indeed, a small disparity means that the scene observed is far from the camera. If a too small maximal disparity value is imposed, some errors will occur when observing scenes that are too close and so that have a bigger disparity than the maximal value imposed. However when the maximal disparity is set to a bigger value than 144 pixels, distances under 30 cm can be measured.

To determine which maximal disparity value was giving the most accurate results, the root means square error between the real distance and the measures have been computed for every maximal disparity value, and for distances between 22 and 89 cm. The results are shown in Fig.24 in the annexe. The smallest RMS value has been found for a maximal disparity value of 176 pixels, therefore this value has been kept for the rest of this work. Furthermore, it can be observed that a black square is present on the disparity map, and becomes bigger when raising the maximal number of disparity value. This is due to the fact that the map represents the pixels on the scene that are visible from both cameras, and the black border represents the pixels that are visible from only one camera, so for a large maximal disparity value, the scene observed by both cameras is smaller.

3.3 Detection of the gap

As explained in the Sec.2.4.1, the detection of straight lines in one image is theoretically simple, using Sobel filters and Hough Transform. However in practice, the implementation of an accurate line detection algorithm implies more challenges. The biggest limitation was here the quality of the depth map. Indeed, the first attempts were done using two Roombot yellowish plates having a size of 22×22 cm to create a gap, as shown on Fig.15. As for the distance measurement experiment, it has been observed that due to the uniformity of the colors on the plate, the SGBM algorithm was not able to effectively couple enough matching points from the two cameras to create an accurate depth map. A first improvement has been to cover the plates with some texture showing random patterns as shown on Fig.16.


(a) Gap

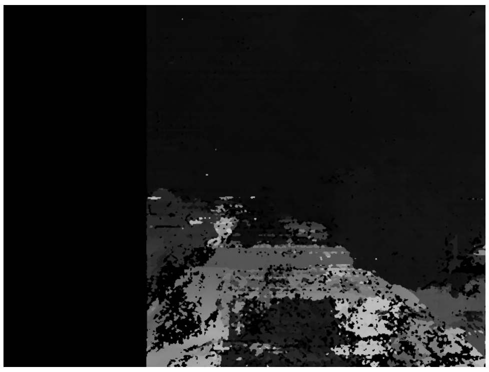
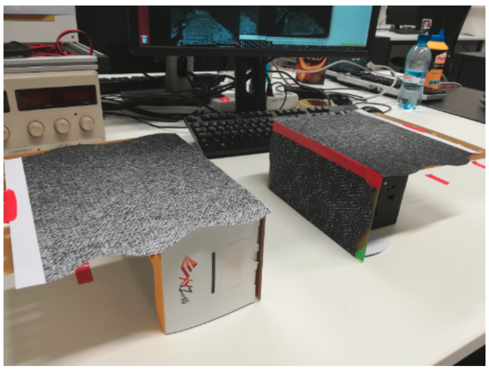

(b) Corresponding depth map

Figure 15: Roombots plate without texture


(a) Gap

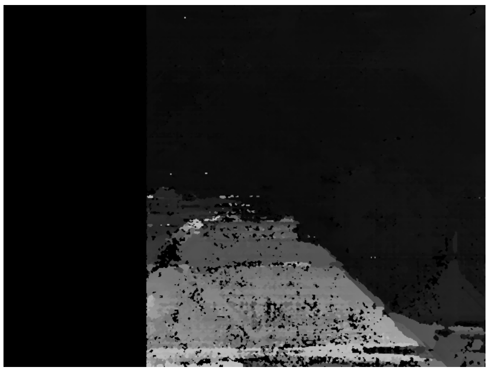

(b) Corresponding depth map

Figure 16: Roombots plate with random pattern texture

To smooth the output depth map, a morphological operation has been applied. The closing operator was the most adequate, given that this is the most efficient operation for noise removal, and the size of the Kernel was set to be relatively big compared to the size of the image (9 to 21 pixels). The choice of a rectangular morphological elements allowed in the same time to enhance the horizontal lines. Thanks to this improvement the algorithm was sometimes able to detect the correct edges. However the results were still not acceptable, the lines being too many times unseen and incorrect lines were selected instead. This is understandable given the appearance of the smoothed depth map (see Fig.17).

After many trials to measure the width of a gap with the described setup, it has been decided to modify the setup in order to improve the quality of the depth map. The main weakness of the previous setup was the fact that the length of each edge was really short (only 22 cm) compared to the scene observed. So in order to facilitate the edge detection, longer edges have been made out of expanded polystyrene and covered with random pattern textures (see Fig.18). As they were covering almost the whole field of view of the camera, the results were expected to be better. A picture showing the quality of the depth map before and after applying the closing element is shown on Fig.19

Once the closing is applied to the raw depth map image, a Canny edge detector is performed to isolate the edges of the gap. The Fig.20 shows the effect of the threshold on the detection. It can be seen that even if the quality of the raw depth map is not so good because of the

Figure 17: Depth map after closing with a kernel size of 9

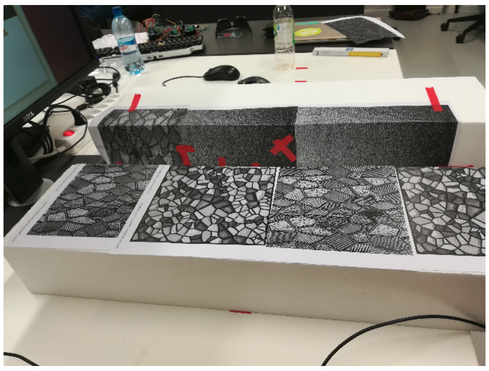


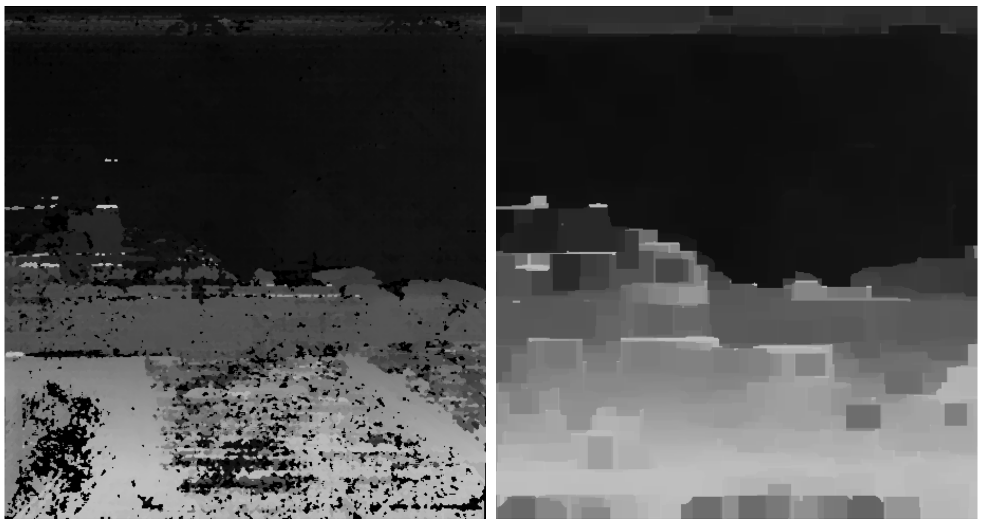
Figure 18: Final setup with long edges

high presence of noise, the first edge is highlighted and will possibly be detectable by the Hough transform. However, the furthest edge of the gap is hardly detectable. At this point, it had to be decided which of the two ideas described in the Sec.2.5 was fitting the best for this application. The first method was implying to detect correctly the two edges of the gap, which seemed to be way more difficult to obtain while offering no greater accuracy. For this reason, the second method described in the Sec.2.5 has been kept, implying this time to detect only the closest line.

3.4 Width of the gap

As mentioned in the previous section the method to measure the size of the gap is the one shown in the Fig.9b. The biggest challenge is to detect the correct line among all the other ones detected. As shown on Fig:21a, the noisy nature of the depth map leads to the detection of many undesirable lines. If the values of the thresholds of both the canny edge detector and the Hough transform are set too high, too much information is loss and the correct edge cannot be detected anymore. In order to make a first sort, an assumption has been made that the edge that the camera sees was close to be horizontal, given the fact that the edges where parallel to the line formed by the alignment of the two stereo cameras. So all the lines having an angle $| \alpha |> \pm 5°$ have been deleted.

This allowed to have a remaining set with less than 10 lines approximatively. As the gap can



(a) Raw depth map (b) Depth map after closing (K = 19)

Figure 19: Effect of the closing on the depth map

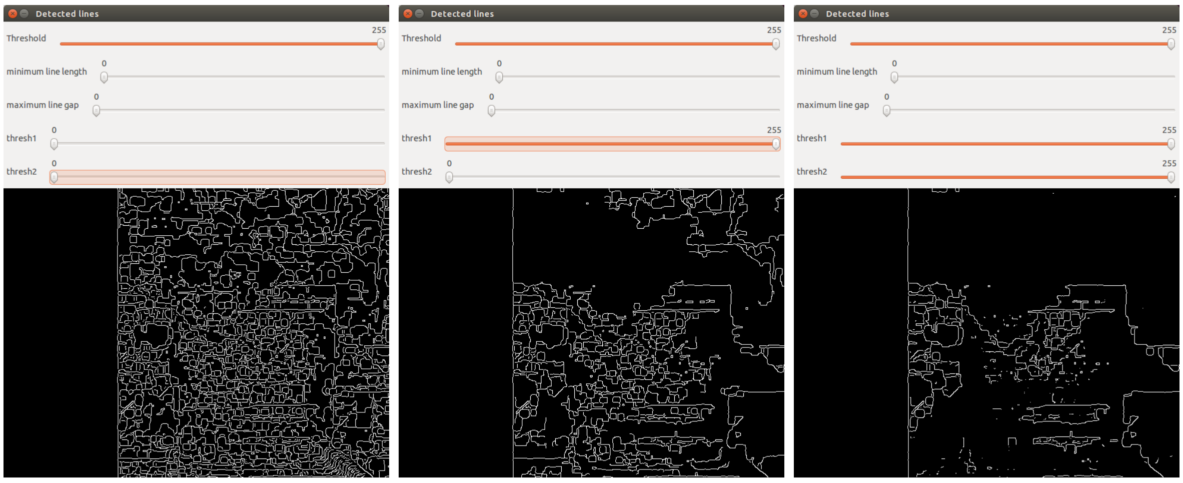
be detected in all the field of view of the camera, it is not possible to use a position condition to select the correct line among the remaining ones. The idea that came up was to keep only the longest line, because the edge being big enough to cover the whole field of view of the camera, the longest horizontal line should always be the correct edge. However in practice, this line detection algorithm has a very low detection rate of 31% as explained in the following section. In order to be able to evaluate the quality of the distance measurement even when the line detected was not the correct one, an option has been added, allowing to select the correct line with the mouse in case of a wrong output of the line detection algorithm.

An experiment has been made to evaluate the quality of the measurements of the gap width. The goal was in the same time to see if the line detection algorithm was giving satisfying results. The first edge has been placed at different distances from the stereo cameras and measurements have been made for widths between 11 and 55 cm. The Fig.18 shows the setup used for this tests. As for the previous experiment, each distance has been measured 10 times, and it is mentioned if the line has been automatically detected or if the use of the mouse was necessary.

The height of the camera had to be precariously chosen. Indeed, at a too low position, the camera did not allow to detect correctly the edge due to the fact that the two edges were barely distinguishable, and on the other hand a too high camera leaded to undesirable results too. By performing different tests, the height of the camera have been set for this experiment to 25.5 cm, and the cameras are tilted down a 5 degree angle.

Some practical limitations had to be taken care of. Indeed, as the expanded polystyrene edges is only 9 cm high, the cameras were seeing the table instead of the furthest edge when the width of the gap was too long (see Fig.22). To solve this, the closest edge had to be raised for some measurement preventing the cameras to detect the table. This allowed to measure the performances of the stereo cameras without having biased results.

The Fig.25 (in the annexe) shows the results of the experiment meant to evaluate the measurement quality of the gap width. The x coordinate of the graphs represents the distance of the second edge from the camera, while the distance to the first edge is given for each graph. The y


Detected lines
Threshold 255
minimum line length 0
maximum line gap 0
thresh1 0
thresh2 0

Detected lines
Threshold 255
minimum line length 0
maximum line gap 0
thresh1 255
thresh2 0

Detected lines
Threshold 255
minimum line length 0
maximum line gap 0
thresh1 255
thresh2 255

(a) Both treshold at zero (b) High treshold 1 (c) High threshold 1 and 2

Figure 20: Effects of the thresholds on the Canny edge detector

coordinate represent the measured width of the gap, compared each time to the acceptable error range. As discussed before, the possible widths of the gap are assumed to be discrete multiples of 11, that is why an error of ±6.5 cm is acceptable for each measure. All the points shown in red on the graphs represent the lines that needed to be selected manually while the green ones are showing the lines detected automatically. The mean of each width gap measurements is also represented by a black square. Many informations can be drawn from this graph. The first thing that can be observed is the detection rate of the line detection algorithm. The overall success rate is 31% which is far from enough to qualify the detection as robust. However, some differences can be observed depending on the distance of the stereo camera to the first edge.

The first graph (Fig.25a) is the one showing the most accurate results. It gives the width measurements when the first edge is placed at 27 cm from the cameras. In this case the detection rate goes up to 70%. Secondly it can be noticed that almost all the points are inside the tolerance zone for gap width between 11 and 44 cm, which means that the width of the gap could have been correctly computed each time (only one point is outside of the zone, which makes a correct measurement rate of 97.5%. A gap of 55 cm imply to measure distances at 83 cm from the cameras, which is close to the range of validity of the stereo camera as explained in the previous experiment. This can be an explanation to the the poor results obtained in this case (0% of correct measurements).

Analyzing then the measurement when the first edge is place 38 cm from the cameras (Fig.25b), it can first be observed that the detection rate is much worse than for the previous measurement (10%). There are also some measurement that are outside of the acceptance zone, which makes a total of 90% of correct distance computed. The same observations can be made for the third graph (Fig.25c). It has the same detection of rate of 10%, with some measurement that are outside of the acceptable zone (95% of correct distances computed).

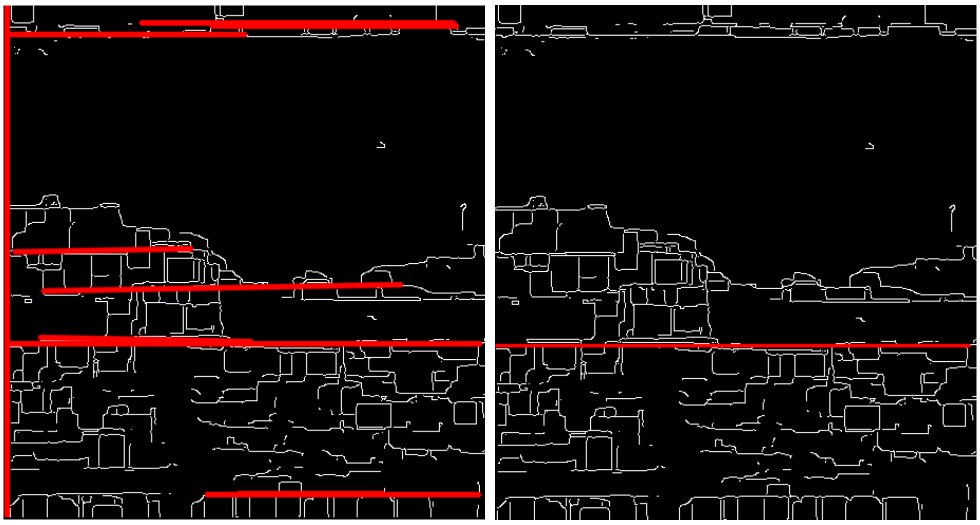

(a) Horizontal line detected (b) Correct line detected

Figure 21: Line detection algorithm

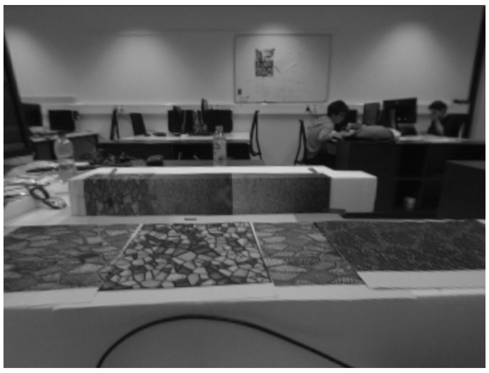


Figure 22: Physical limitations of the edges

The overall measurement success rate is 87.8%. This proves that even with a bad quality disparity map, the width of an edge can be computed quiet accurately. Looking at the means values, they are all inside the acceptable tolerances, except for the big misclassification explained on the first graph. This is extremely promising because this means that by taking ten pictures of the gap and then making an average of the computed distances would give a success rate of 100%.

4 Discussion

The first thing that can be drawn from this study is that it is possible to detect an edge using stereo vision. However, in order to end up with decent results, many assumptions had to be made.

The first one was to assume that the width of the gaps was taking discrete values, which were the multiple of 11. As explained before, 11 is not taken randomly, but it is the diameter of a roombot module. This has been assumed in order to give the measures a large error margin. However, in practice if the gap takes random width values, the calculations made in this work will not be valid.

A second important assumption has been to consider that the edges were seen by the camera as horizontal in order to eliminate some wrong detected lines. Here again in practice it will not be always true. More generally, as explained in the Sec.3.4, the line detection algorithm was not giving accurate results, which means that the edge had to be selected manually for each measurement. This is really dependent to the quality of the depth map. As it can be seen in every depth map figure of this study the disparity map has an important noisy nature, and presents some unexplained shadow zones, which makes it more complicated to detect straight lines.

The initial idea was to integrate the two cameras inside two separated Roombot modules, and to connect them to embedded PC (in this case the NanoPi NEO Plus2) which would send the data to a central PC to process the images. Due to a lack of time, this work has been focused on trying to improve the quality of both the depth map and the line detection.

4.1 Outlook

The next steps of this work would consist first in exploring new solutions that could improve the quality of the depth map, such as working in a more neutral environment or trying different textures. Indeed, as mentioned several times before, the automation of the line detection and so of the gap measurement step would be consequently improved with a better disparity map. Then the following steps would be to adapt the existing code for the NanoPi NEO Plus2, and so make it possible the integration of the cameras into Roombots modules.

However, through this study, it has been shown that stereo vision can be an efficient tool for this particular width measurement application, and more generally for any robot needing obstacle avoidance or distance measurement. But as many computer vision works, it was custom designed to work under some very specific application scenarios. An important improvement would surely be to adapt this work to some more general conditions. For example to allow the detection of gaps with different shapes which would be more close to a natural environment.

5 Conclusion

Being the first project of this magnitude I've ever done on my own, this work allowed me to go deep into the different aspects of a scientific study, from the learning of a programming language and its library, to the creation of a valid processing program and the creation of experiments to evaluate the quality of the processed images. It made me aware of certain limitations, such as the gap between the theoretical concepts and their implementation into a `C++` code and the coding time, which has been consequent, especially at the beginning when I was not yet comfortable with the many functions of the `OpenCV` library. This project also allowed me to practice some basic image processing concepts such as the morphological operations or the Canny edge detector, acquired during last semester courses and which have been very useful for this work.

References

[1] K. Sabe, M. Fukuchi, J.-S. Gutmann, T. Ohashi, K. Kawamoto, T. Yoshigahara *Obstacle avoidance and path planning for humanoid robots using stereo vision.* Published in: Robotics and Automation, 2004

[2] R. Mandelbaum, L. McDowell, L. Bogoni, B. Reich, M. Hansen *Real-time stereo processing, obstacle detection, and terrain estimation from vehicle-mounted stereo cameras* Published in: Applications of Computer Vision, 1998

[3] K. Toyama, G.D. Hager, J. Wang *Servomatic: a modular system for robust positioning using stereo visual servoing* Published in: Robotics and Automation, 1996

[4] D.J. Kriegman, E. Triendl, T.O. Binford *Stereo vision and navigation in buildings for mobile robots* Published in: IEEE Transactions on Robotics and Automation, 1989

[5] Sotiris Malassiotis, Michael G. Strintzis *Stereo vision system for precision dimensional inspection of 3D holes* Published online 2003

[6] D. Burschkal, S. Lee, G. Hager *Stereo-based obstacle avoidance in indoor environments with active sensor re-calibration*

[7] Don MurrayJames J. Little *Using Real-Time Stereo Vision for Mobile Robot Navigation* Published in Autonomous Robot, Volume 8, Issue 2, pp 161–171, 2000

[8] Yi Jin, Ming Xie *Vision guided homing for humanoid service robot* Published in: Pattern Recognition, 2000

[9] BioRob website (*last consulted 02.06.18*)
`https://biorob.epfl.ch/roombots`

[10] Wikipedia : Computer stereo vision (*last consulted 18.05.18*)
`https://en.wikipedia.org/wiki/Computer_stereo_vision`

[11] Wikipedia : Calibration de caméra (*last consulted 11.05.18*)
`https://fr.wikipedia.org/wiki/Calibration_de_caméra`

[12] Wikipedia : Essential matrix (*last consulted 12.05.18*)
`https://en.wikipedia.org/wiki/Essential_matrix`

[13] Wikipedia : Image rectification (*last consulted 12.05.18*)
`https://en.wikipedia.org/wiki/Image_rectification`

[14] Wikipedia : Block-matching algorithm (*last consulted 12.05.18*)
`https://en.wikipedia.org/wiki/Block-matching_algorithm`

[15] Opencv documentation : camera calibration (*last consulted 17.05.18*)
`https://docs.opencv.org/3.0-beta/doc/tutorials/calib3d/camera_calibration/camera_calibration.html`

[16] Opencv documentation : epipolar geometry (*last consulted 17.05.18*)
`https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_calib3d/py_epipolar_geometry/py_epipolar_geometry.html`

[17] Opencv documentation : Canny edge detector (*last consulted 21.05.18*)
https://docs.opencv.org/3.0-beta/doc/tutorials/imgproc/imgtrans/canny_detector/canny_detector.html

[18] Opencv documentation about typical function used in stereo vision(*last consulted 21.05.18*)
https://docs.opencv.org/3.4/d9/d0c/group__calib3d.html

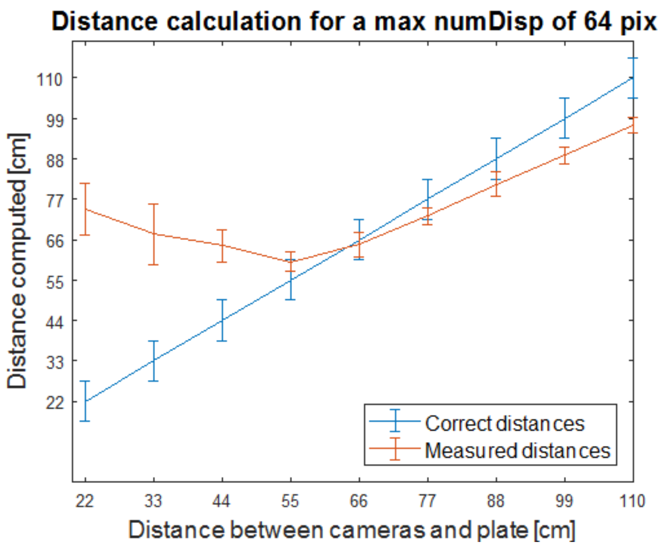
[19] Mathworks : what is camera calibration (*last consulted 21.05.18*)
https://ch.mathworks.com/help/vision/ug/camera-calibration.html

[20] Benjamin Pichler *HDR Light Field.* 2012

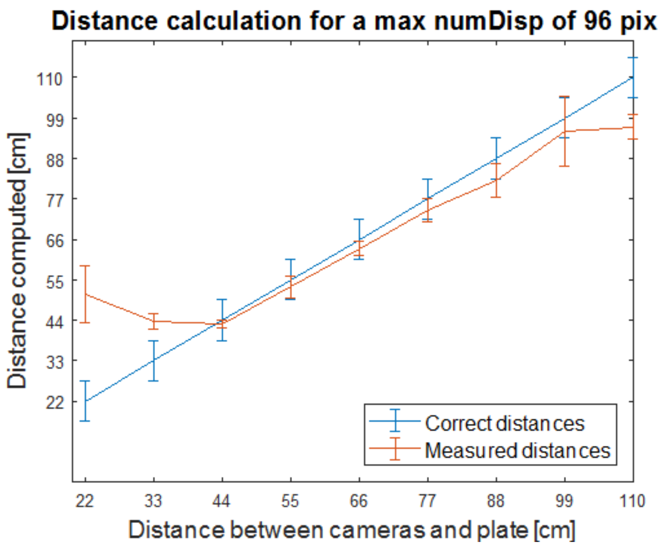
[21] Heiko Hirschmüller *Semi-Global Matching.* 2011

[22] Alexander Sprowitz, Soha Pouya, Stephane Bonardi, Jesse van den Kieboom, Rico Möckel, Aude Billard, Pierre Dillenbourg, A.J. Ijspeert *Roombots: Reconfigurable Robots for Adaptive Furniture* Published in IEEE Computational Intelligence Magazine, 2010

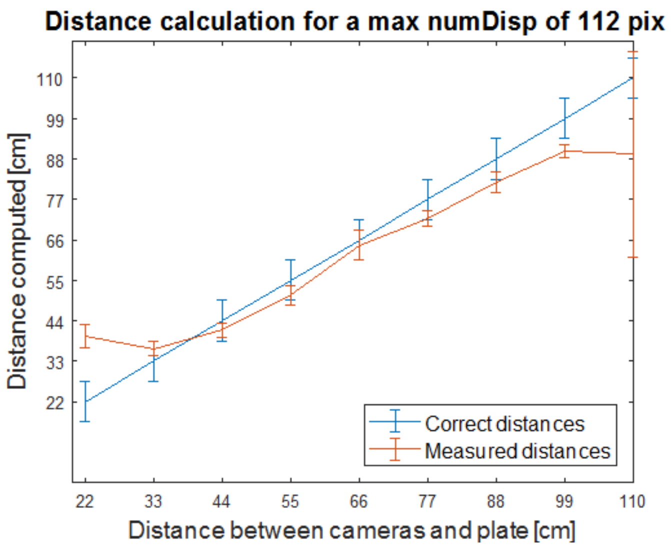
6 Annexe



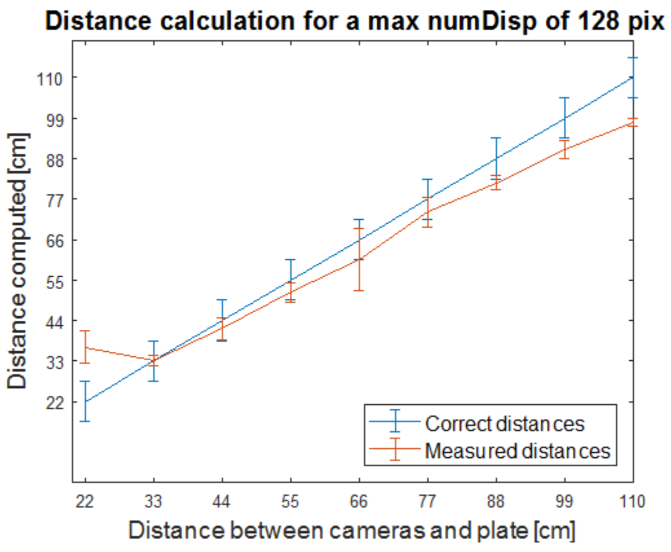
(a) Maximal disparity value = 64 pix



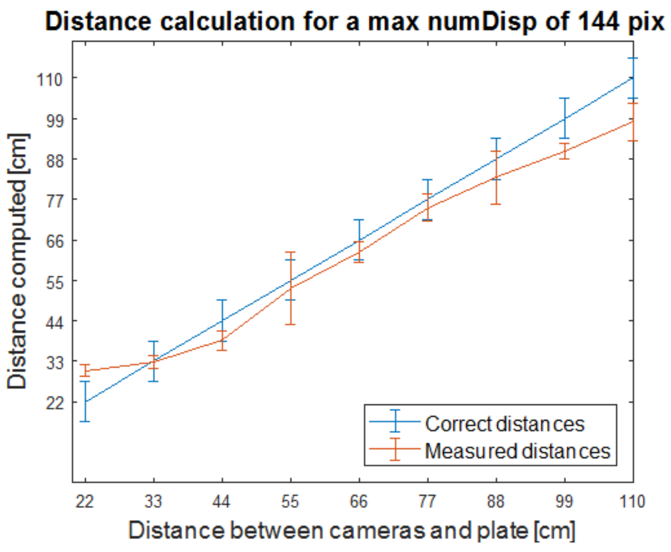
(b) Maximal disparity value = 96 pix



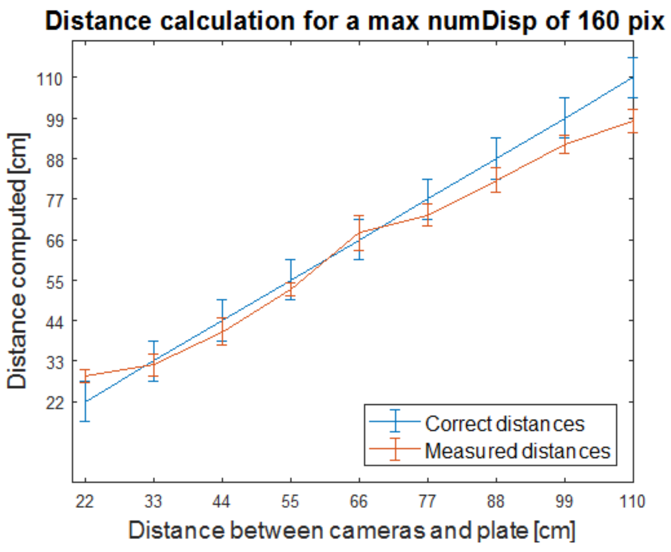
(c) Maximal disparity value = 112 pix



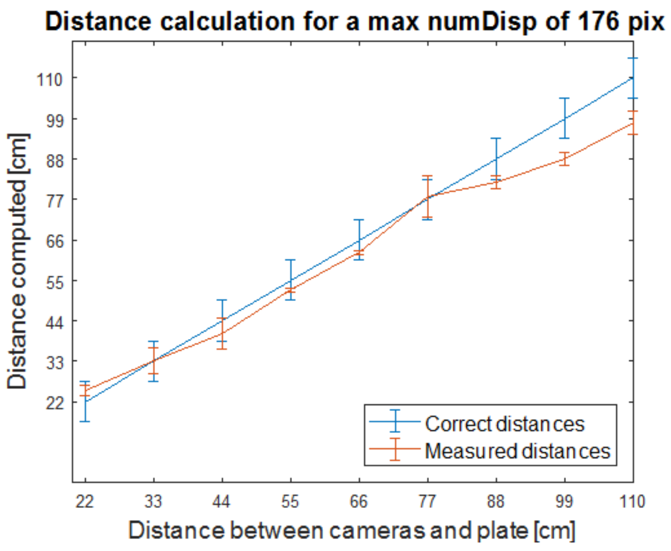
(d) Maximal disparity value = 128 pix



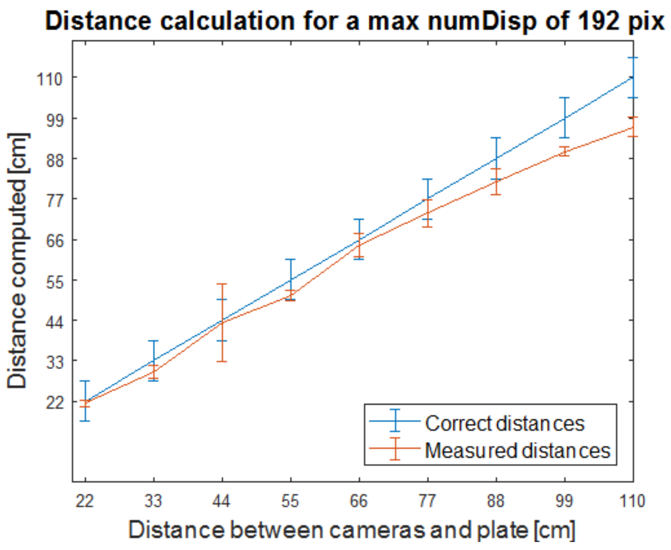
(e) Maximal disparity value = 144 pix



(f) Maximal disparity value = 160 pix

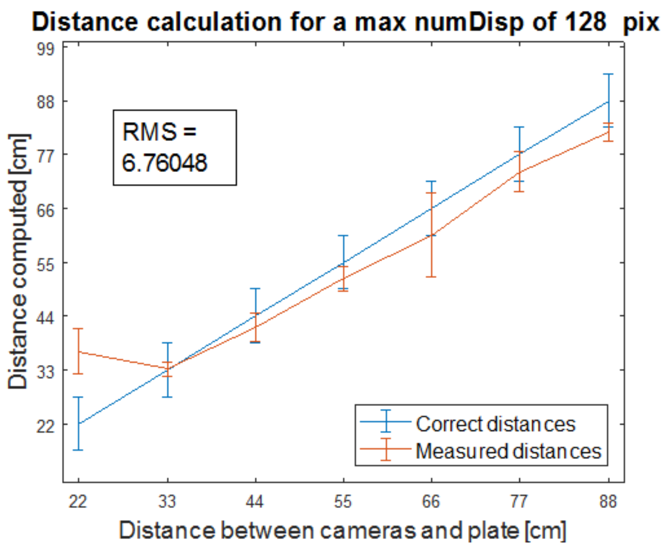


(g) Maximal disparity value = 176 pix

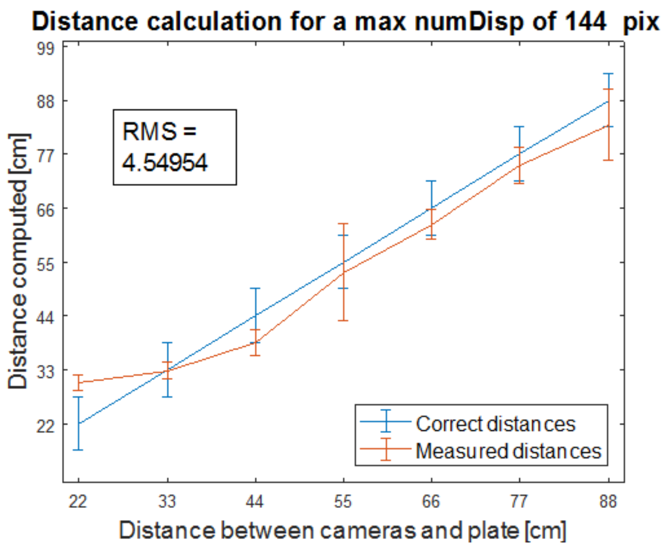


(h) Maximal disparity value = 192 pix

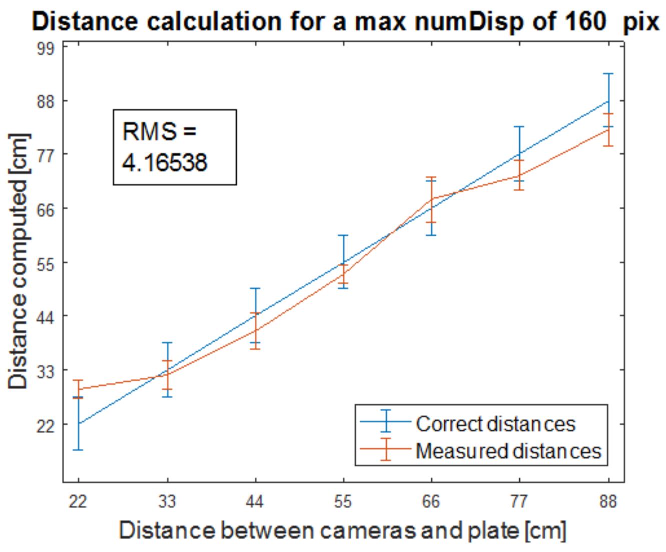
Figure 23: Distance measurement for different values of maximal disparity



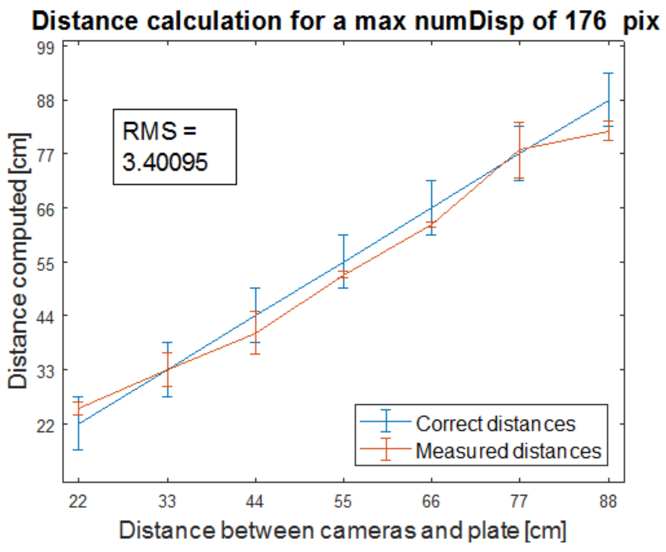
(a) Maximal disparity value = 128 pix



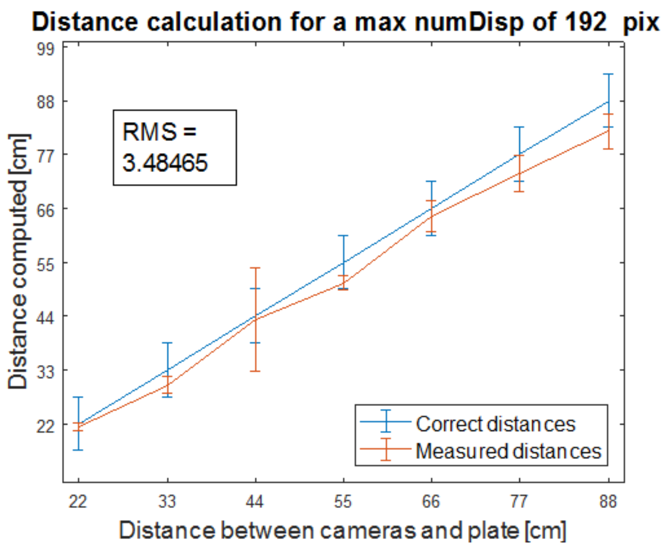
(b) Maximal disparity value = 144 pix



(c) Maximal disparity value = 160 pix

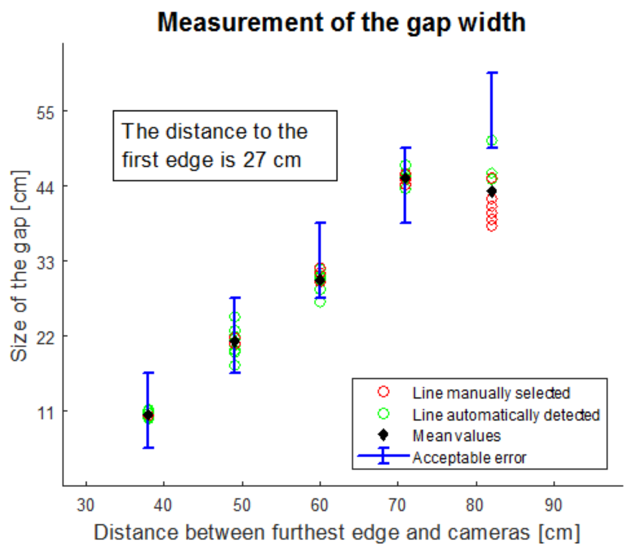


(d) Maximal disparity value = 176 pix

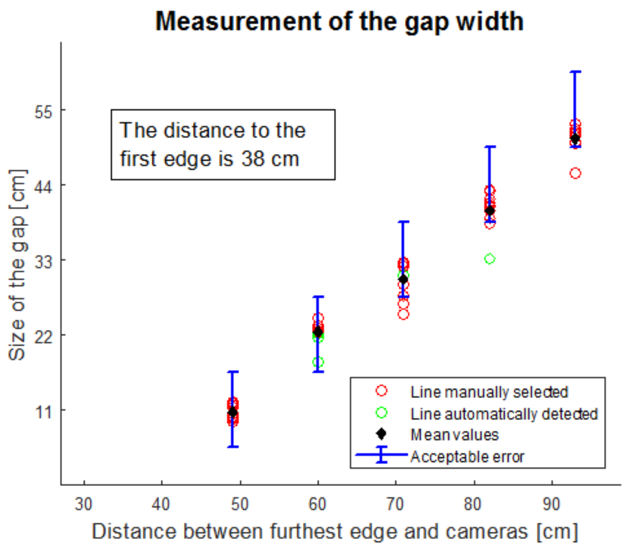


(e) Maximal disparity value = 192 pix

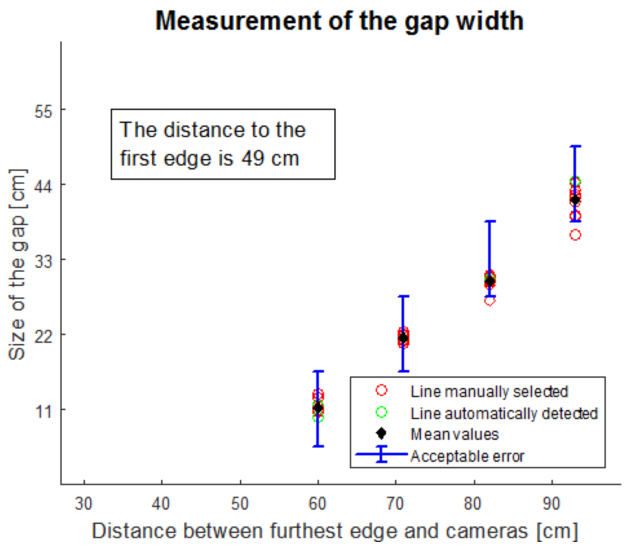
Figure 24: Distance measurement in the correct range



(a) Distance to the first edge : 27 cm



(b) Distance to the first edge : 38 cm



(c) Distance to the first edge : 49 cm

Figure 25: Results of the measurements of the gap width