CAMERA GEOMETRICAL CALIBRATION COMPUTER VISION

May 12, 2017

Partners: Evelyn Paiz & Nadile Nunes

Instructor: Alain Tremeau

OBJECTIVES

Learn how to perform camera calibration using MatLab and CalTech (The California Institute of Technology) calibration toolboxes.

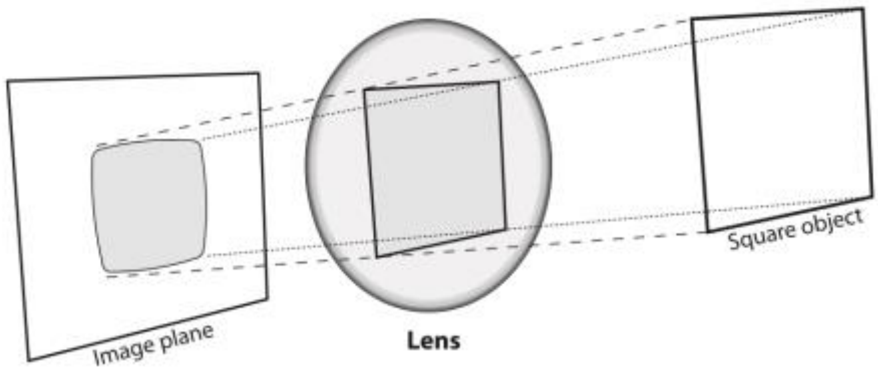
As part of this lab sessions, the following tasks were done sequentially:

- Learn how to use the camera calibration toolbox.
- Calibrate a single camera using the toolbox.
- Calibrate a stereo system using the toolbox.

INTRODUCTION

A camera is an optical system composed of a sensor and a lens. Because no optical system is perfect, an image recorded with a camera is distorted.

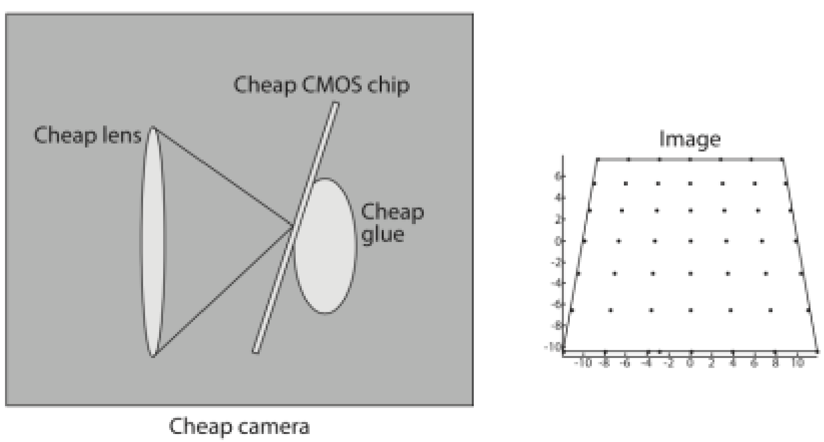
The most common type of distortion, radial distortion, is especially noticeable in images taken with cheap cameras or wide angle lens cameras (fisheye cameras for example). In function of the shape of the distortion, it can be described as pincushion or barrel. It is an optical aberration that depends only on the field of view: radial distortion is null at the center of the field, and is symmetric in x and y for a circular lens. It is usually described with 3 parameters: k_1, k_2 and k_3.



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

Figure 1: Radial distortion - rays farther from the center of a simple lens are bent too much com- pared to rays that pass closer to the center; thus, the sides of a square appear to bow out on the image plane (this is also known as barrel distortion)

The second most common distortion is tangential distortion. This distortion is due to manufacturing defects resulting from the lens not being exactly parallel to the imaging plane. It is usually described with 2 parameters: p_1 and p_2.



$$x_{\text{corrected}} = x + [2p_1 y + p_2(r^2 + 2x^2)]$$
$$y_{\text{corrected}} = y + [p_1(r^2 + 2y^2) + 2p_2 x]$$

Figure 2: Tangential distortion - lens is not fully parallel to the image plane; in cheap cameras, this can happen when the imager is glued to the back of the camera

Radial distortion arises as a result of the shape of the lens, whereas tangential distortion arises from the assembly process of the camera as a whole.

The aim of camera calibration is to determine the distortion parameters k_1, k_2, k_3, p_1 and p_2. The method developed by CalTech is based on chessboard images.

This method can also provide information about the location and orientation of two cameras that are used to take images of the same object, and is therefore suitable for stereovision system calibration.

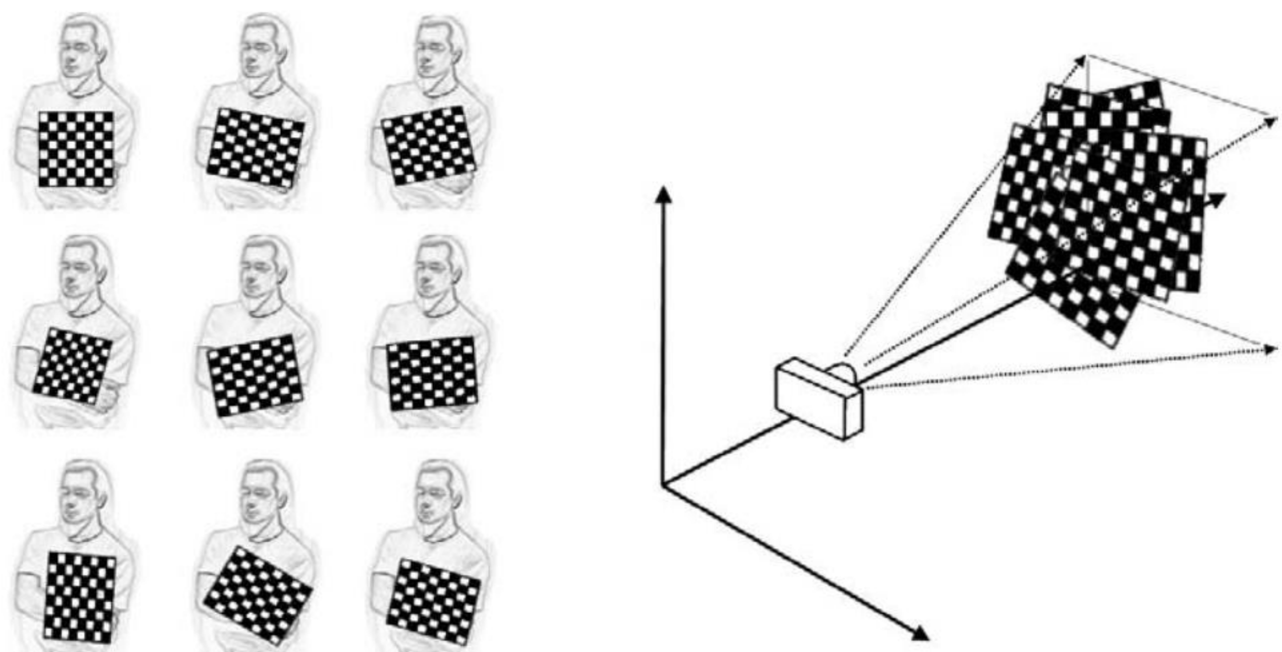


Figure 3: Calibration method based on chessboard images - Images of a chessboard being held at various orientations provide enough information to completely solve for the locations of those images in global coordinates (relative to the camera) and the camera intrinsics.

PRACTICAL SESSION

To start the practical session, the calibration toolbox was downloaded from the link provided (http://www.vision.caltech.edu/bouguetj/calib_doc/#system) and the individual matlab files (.m files) were saved into a folder (**TOOLBOX_calib**). To run the toolbox the path of this folder was added to the matlab path. All the practice was carried out using the standard memory configuration. After completing the lab session, the following results were achieved:

Part 1: Calibration of a single camera

For the first test, a calibration of a single camera was done. The figure 4 shows the main menu used for all the process. It was evaluated a set of 20 images of a planar checkboard provided by the web page (**calib_example.zip**). This approach was performed with the objective to learn how to use all the features of the toolbox.

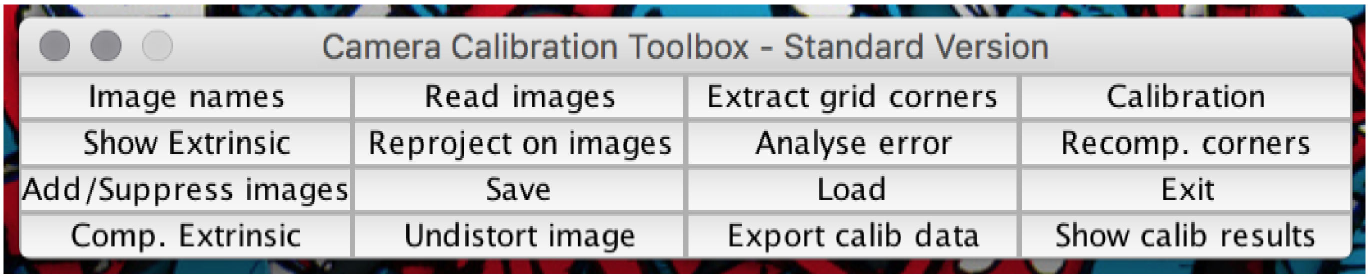


Figure 4: Main menu from the single camera calibration toolbox using the standard version of memory

The steps needed to be followed to do a complete calibration of a single camera was the following:

- Load calibration images
- Extraction of image corners
- Run of the main calibration engine
- Display of the results
- Control the accuracies
- Add and suppress images
- Undistort images
- Export calibration data to different formats

1. Reading of the images:

From the interactive GUI (command **calib**), it was selected the **Image names** option. The basename was entered and the format of the images. All the images were loaded showing the results in the console and in a thumbnail format (figure 5).

```
>> calib

.          Image11.tif  Image15.tif  Image19.tif  Image4.tif   Image8.tif
..         Image12.tif  Image16.tif  Image2.tif   Image5.tif   Image9.tif
Image1.tif   Image13.tif  Image17.tif  Image20.tif  Image6.tif
Image10.tif  Image14.tif  Image18.tif  Image3.tif   Image7.tif


Basename camera calibration images (without number nor suffix): Image
Image format: ([]='r'='ras', 'b'='bmp', 't'='tif', 'p'='pgm', 'j'='jpg', 'm'='ppm') tif
Loading image
1...2...3...4...5...6...7...8...9...10...11...12...13...14...15...16...17...18...19...20...
Done
```

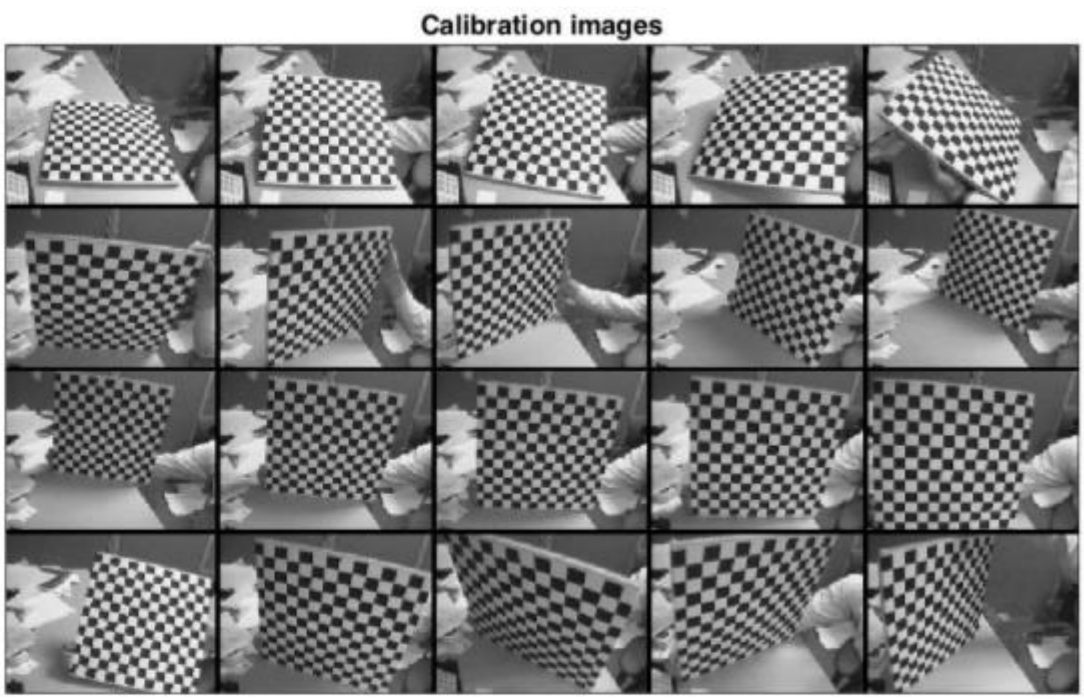


Figure 5: Image loading of 20 samples for calibration example 1

2. *Extracting the grid corners:*

Again, from the interactive GUI (main menu), it was selected the option **Extract grid corners**. It was selected all the images to evaluate and a default window size of the corner finder ($wintx = winty = 5$), leading to an effective window of size 11×11 pixels. It was selected the automatic mechanism for counting the number of squares in the grid and the four extreme corners of the image were selected (following the order: upper left, upper right, down right, down left). Then the boundary of the calibration was showed and the dX and dY in X and Y of each square in the grid was defined as $dX = dY = 30\ mm$. Finally, the prediction was showed, and if the predicted corners were close to the real image, a value for the radial distortion coefficient was not necessary. This process was repited for all the 20 images. Figure 6 shows the extraction of the first image.

```
Extraction of the grid corners on the images
Number(s) of image(s) to process ([] = all images) =
Window size for corner finder (wintx and winty):
wintx ([] = 5) =
winty ([] = 5) =
Window size = 11x11
Do you want to use the automatic square counting mechanism (0=[]=default)
  or do you always want to enter the number of squares manually (1,other)?

Processing image 1...
Using (wintx,winty)=(5,5) - Window size = 11x11      (Note: To reset the window size,
run script clearwin)
Click on the four extreme corners of the rectangular complete pattern (the first
clicked corner is the origin)...
Size dX of each square along the X direction ([]=100mm) = 30
Size dY of each square along the Y direction ([]=100mm) = 30
If the guessed grid corners (red crosses on the image) are not close to the actual
corners,
it is necessary to enter an initial guess for the radial distortion factor kc (useful for
subpixel detection)
Need of an initial guess for distortion? ([]=no, other=yes)
Corner extraction...

Processing image 2...
Using (wintx,winty)=(5,5) - Window size = 11x11      (Note: To reset the window size,
run script clearwin)
Click on the four extreme corners of the rectangular complete pattern (the first
clicked corner is the origin)...
```

...

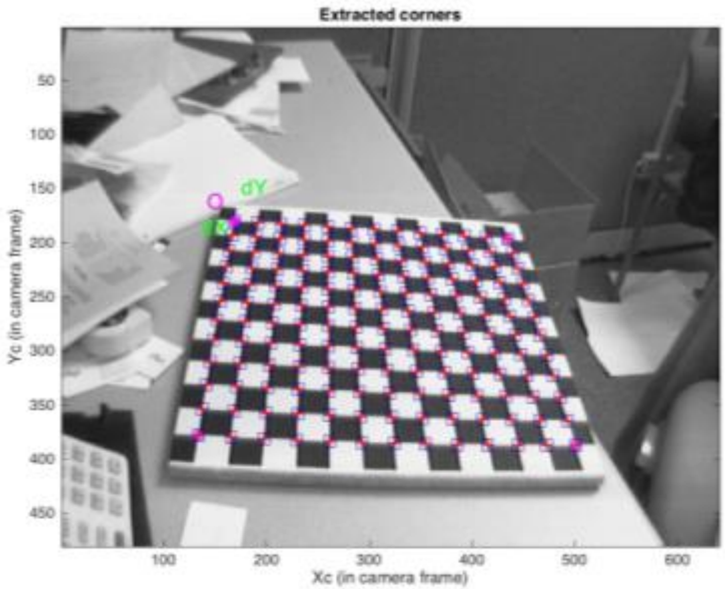


Figure 6: Corner extraction result for Image1

3. Main calibration step:

After the corner extraction, the process of calibration was performed (done in tow steps: first initialization and then nonlinear optimization). The initialization step was the one that computed the closed-form solution for the calibration parameters based not including any lens distortion. The non-linear optimization step minimized the total reprojection error (in the least squares sense) over all the calibration parameters (9 DOF for intrinsic: focal, principal point, distortion coefficients, and $6 * 20$ DOF extrinsic $\Longrightarrow 129$ parameters).

```
Aspect ratio optimized (est_aspect_ratio = 1) -> both components of fc are estimated
(DEFAULT).
Principal point optimized (center_optim=1) - (DEFAULT). To reject principal point,
set center_optim=0
Skew not optimized (est_alpha=0) - (DEFAULT)
Distortion not fully estimated (defined by the variable est_dist):
    Sixth order distortion not estimated (est_dist(5)=0) - (DEFAULT) .
Initialization of the principal point at the center of the image.
Initialization of the intrinsic parameters using the vanishing points of planar
patterns.

Initialization of the intrinsic parameters - Number of images: 20
```

```
Calibration parameters after initialization:

Focal Length:          fc = [ 670.82633   670.82633 ]
Principal point:       cc = [ 319.50000   239.50000 ]
Skew:             alpha_c = [ 0.00000 ]   => angle of pixel = 90.00000 degrees
Distortion:            kc = [ 0.00000   0.00000   0.00000   0.00000   0.00000 ]

Main calibration optimization procedure - Number of images: 20
Gradient descent iterations:
1...2...3...4...5...6...7...8...9...10...11...12...13...14...15...16...17...18...19...20...21...22...23..
.24...25...26...done
Estimation of uncertainties...done


Calibration results after optimization (with uncertainties):

Focal Length:          fc = [ 662.42717   663.65133 ] +/- [ 2.08922   2.24411 ]
Principal point:       cc = [ 304.02173   242.11286 ] +/- [ 4.03344   3.74861 ]
Skew:             alpha_c = [ 0.00000 ] +/- [ 0.00000  ]   => angle of pixel axes =
90.00000 +/- 0.00000 degrees
Distortion:            kc = [ -0.28997   0.37831   0.00130   0.00234  0.00000 ] +/- [
0.01665   0.06885   0.00094   0.00098  0.00000 ]
Pixel error:          err = [ 0.81489   0.67060 ]

Note: The numerical errors are approximately three times the standard deviations
(for reference).
```

From these results, it was possible to notice that the skew coefficient $alpha\_c$ and the 6^{th} order radial distortion coefficient (the last entry of k_c) had not been estimated. Therefore, the angle between the x and y pixel axes was 90 degrees. For our situation, this was a very good assumption.

Next the reprojection error was evaluated using the **Reproject on images** on the camera calibration tool menu:

```
Number(s) of image(s) to show ([] = all images) =
Pixel error:      err = [0.81489   0.67060] (all active images)
```

The result was too large, this was because the manual corner extraction was not done properly so an automatically corner extraction was done. It was achieved using the

Recomp. corners option in the menu with a window size again of $wintx = winty = 5$ and all the 20 images.

```
Re-extraction of the grid corners on the images (after first calibration)
Window size for corner finder (wintx and winty):
wintx ([] = 5) =
winty ([] = 5) =
Window size = 11x11
Number(s) of image(s) to process ([] = all images) =
Use the projection of 3D grid or manual click ([]=auto, other=manual):
Processing image
1...2...3...4...5...6...7...8...9...10...11...12...13...14...15...16...17...18...19...20...
done
```

Finally, the **Calibration** option was selected again to obtain the final and improved results:

```
Aspect ratio optimized (est_aspect_ratio = 1) -> both components of fc are estimated
(DEFAULT).
Principal point optimized (center_optim=1) - (DEFAULT). To reject principal point,
set center_optim=0
Skew not optimized (est_alpha=0) - (DEFAULT)
Distortion not fully estimated (defined by the variable est_dist):
    Sixth order distortion not estimated (est_dist(5)=0) - (DEFAULT) .

Main calibration optimization procedure - Number of images: 20
Gradient descent iterations:
1...2...3...4...5...6...7...8...9...10...11...12...13...14...15...16...17...18...19...done
Estimation of uncertainties...done


Calibration results after optimization (with uncertainties):

Focal Length:          fc = [ 657.39513   657.76277 ] +/- [ 0.34704   0.37125 ]
Principal point:       cc = [ 302.98122   242.61705 ] +/- [ 0.70573   0.64577 ]
Skew:             alpha_c = [ 0.00000 ] +/- [ 0.00000  ]   => angle of pixel axes =
90.00000 +/- 0.00000 degrees
Distortion:            kc = [ -0.25584   0.12759   -0.00021   0.00003  0.00000 ] +/- [
0.00271   0.01076   0.00015   0.00014  0.00000 ]
Pixel error:          err = [ 0.12672   0.12610 ]

Note: The numerical errors are approximately three times the standard deviations
(for reference).
```

As seen, this time the error for x and y, respectively, was reduced to 0.12672 and 0.12610. Again, the **Reproject on images** option was selected on the camera calibration tool menu, to show the reprojections of the grids onto the original images. These projections were computed based on the intrinsic and extrinsic parameters that have been previously calculated (figure 7). Also, the result of the reprojection error was calculated (figure 8).

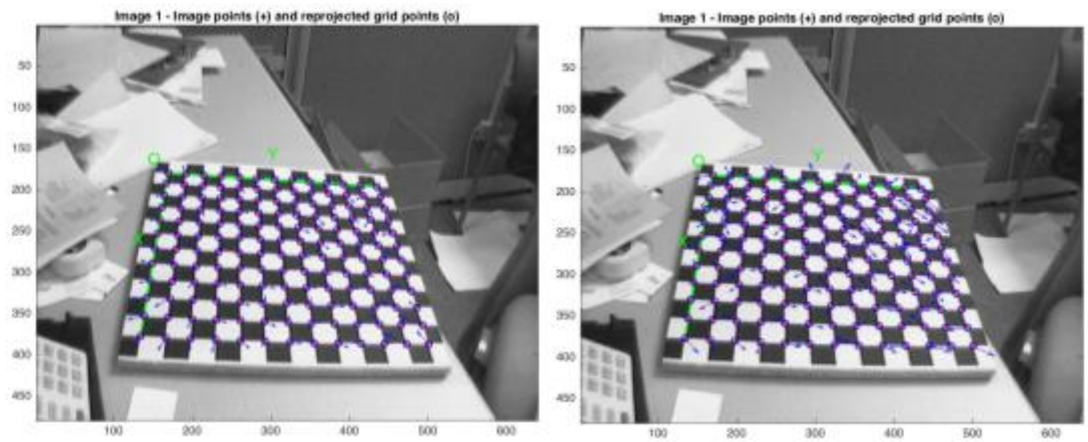


Figure 7: First reprojection result (left) and improved reprojection result (right) for Image1

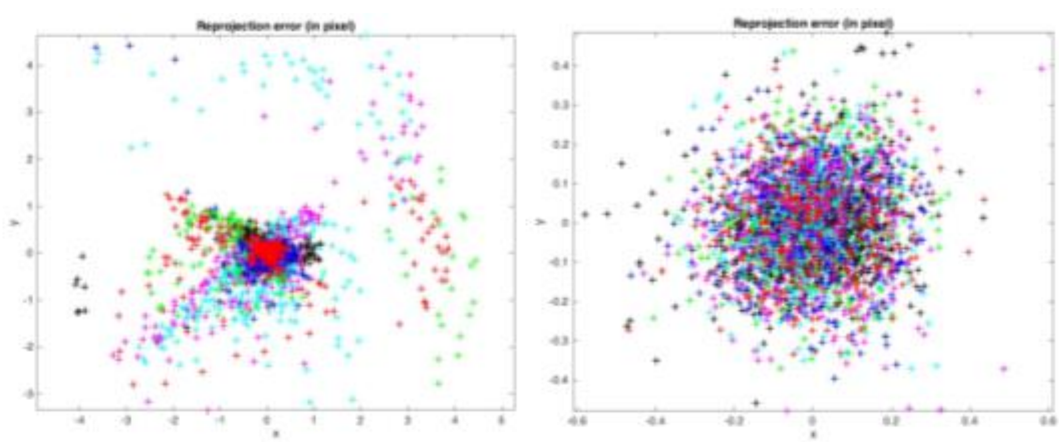


Figure 8: First Reprojection error (left) and improved reprojection error (right)

Finally, the extrinsic parameters were showed using the option **Show Extrinsic**. This was a 3D representation of relative positions of the grids with respect to the camera.

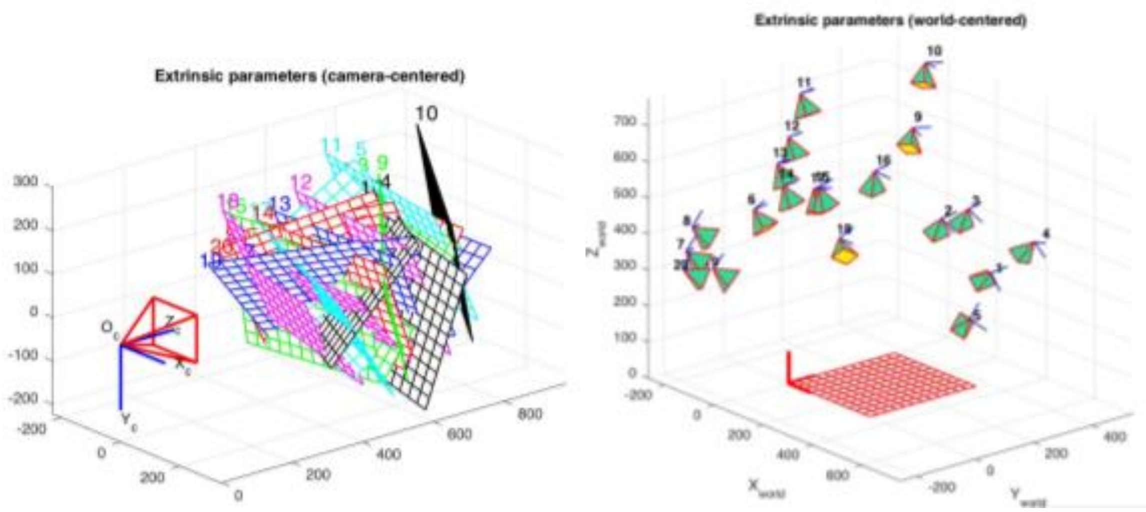


Figure 9: Extrinsic parameters of a single camera (3D plot)

As an additional tool, it's possible to use the **Analysis error** option to continue minimizing the error. This allow to see the images that have large error, so it is possible to re-do the automatically corner extraction with a bigger window size for that images. For example, if it is re-computed the image corners using ($wintx = winty = 7$) for images 5, 7, 8 and 19; ($wintx = winty = 8$) for image 18; and ($wintx = winty = 9$) for the rest, the calibration result is:

```
Calibration results after optimization (with uncertainties):

Focal Length:          fc = [ 657.46302   657.94686 ] +/- [ 0.31819   0.34046 ]
Principal point:       cc = [ 303.13685   242.56964 ] +/- [ 0.64682   0.59218 ]
Skew:             alpha_c = [ 0.00000 ] +/- [ 0.00000  ]   => angle of pixel axes =
90.00000 +/- 0.00000 degrees
Distortion:            kc = [ -0.25403   0.12144   -0.00021   0.00002  0.00000 ] +/- [
0.00248   0.00986   0.00013   0.00013  0.00000 ]
Pixel error:          err = [ 0.11689   0.11500 ]
```

Optimizing the error to 0.11689 and 0.11500 for x and y respectively.

Part 2: Calibration of a stereo system

For the second calibration test, it was used the file **stereo_example.zip**, which contained 14 pairs of corresponding left and right images. The file also had two individual calibration results files: **Calib_Results_left.mat** and **Calib_Results_right.mat**. They were produced after separately calibrating the two cameras using the procedure described in the part 1. The main difference was that after each calibration, the calibration result file **Calib_Results.mat** had been renamed to either **Calib_Results_left.mat** for the left camera or **Calib_Results_right.mat** for the right camera. Figure 7 shows the interactive GUI (command **stereo_gui**) for this procedure:

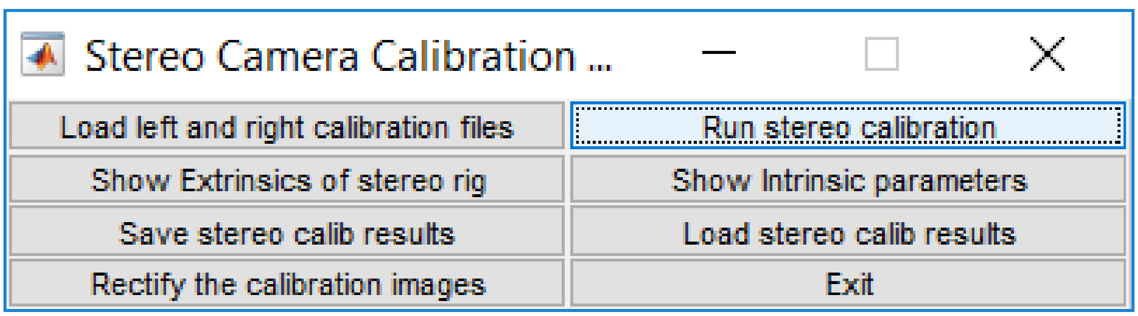


Figure 10: Main menu from the stereo camera calibration toolbox using the standard version of memory

1. *Loading calibration files*

From the main menu, indicated on figure 10, the option **Load left and right calibration files** was selected, then the names were entered of the calibration files.

Calib_Results_left.mat   Calib_Results_right.mat

```
Loading of the individual left and right camera calibration files
Name of the left camera calibration file ([]=Calib_Results_left.mat):
Name of the right camera calibration file ([]=Calib_Results_right.mat):
Loading the left camera calibration result file Calib_Results_left.mat...
```

```
Loading the right camera calibration result file Calib_Results_right.mat...

Stereo calibration parameters after loading the individual calibration files:

Intrinsic parameters of left camera:
Focal Length:          fc_left = [ 533.00371   533.15260 ] ± [ 1.07629   1.10913 ]
Principal point:       cc_left = [ 341.58612   234.25940 ] ± [ 1.24041   1.33065 ]
Skew:             alpha_c_left = [ 0.00000 ] ± [ 0.00000  ]   => angle of pixel axes =
90.00000 ± 0.00000 degrees
Distortion:            kc_left = [ -0.28947   0.10326   0.00103   -0.00029  0.00000 ] ±
[ 0.00596   0.02055   0.00030   0.00037  0.00000 ]

Intrinsic parameters of right camera:
Focal Length:          fc_right = [ 536.98262   536.56938 ] ± [ 1.19786   1.15677 ]
Principal point:       cc_right = [ 326.47209   249.33257 ] ± [ 1.36588   1.34252 ]
Skew:             alpha_c_right = [ 0.00000 ] ± [ 0.00000  ]   => angle of pixel axes =
90.00000 ± 0.00000 degrees
Distortion:            kc_right = [ -0.28936   0.10677   -0.00078   0.00020  0.00000 ] ±
[ 0.00488   0.00866   0.00027   0.00062  0.00000 ]

Extrinsic parameters (position of right camera wrt left camera):
Rotation vector:             om = [ 0.00611   0.00409  -0.00359 ]
Translation vector:           T = [ -99.84929   0.82221  0.43647 ]
```

As a result, the initial values for the intrinsic camera parameters were obtained. In addition, also an estimate for the extrinsic parameters om and T, characterizing the relative location of the right camera with respect to the left camera. The intrinsic parameters were (for left and right cameras respectively):

- **Focal length:** fc_left & fc_right
- **Principal point:** cc_left & cc_right
- **Skew coefficient:** $alpha\_c\_left$ & $alpha\_c\_right$
- **Distortions:** kc_left & kc_right

The parameters om and T were defined such that if considering a point P in $3D$ space, its two coordinate vectors X_L and X_R in the left and right camera reference frames respectively were related to each other through the rigid motion transformation $X_R = R * X_L + T$, where R is the $3{\times}3$ rotation matrix corresponding to the rotation vector om. The relation between om and R was given by the rodrigues formula $R = rodrigues(om)$.

2. Global stereo optimization

From the main menu, the option **Run stereo calibration** was selected, recalculating the intrinsic and extrinsic parameters.

```
Recomputation of the intrinsic parameters of the left camera
(recompute_intrinsic_left = 1)
Recomputation of the intrinsic parameters of the right camera
(recompute_intrinsic_right = 1)

Main stereo calibration optimization procedure - Number of pairs of images: 14
Gradient descent iterations: 1...2...3...done
Estimation of uncertainties...done

Stereo calibration parameters after optimization:
Intrinsic parameters of left camera:
Focal Length:        fc_left = [ 533.52331   533.52700 ] ± [ 0.83147   0.84055 ]
Principal point:     cc_left = [ 341.60377   235.19287 ] ± [ 1.23937   1.20470 ]
Skew:           alpha_c_left = [ 0.00000 ] ± [ 0.00000  ]   => angle of pixel axes =
90.00000 ± 0.00000 degrees
Distortion:          kc_left = [ -0.28838   0.09714   0.00109   -0.00030  0.00000 ] ±
[ 0.00621   0.02155   0.00028   0.00034  0.00000 ]

Intrinsic parameters of right camera:
Focal Length:        fc_right = [ 536.81376   536.47649 ] ± [ 0.87631   0.86541 ]
Principal point:     cc_right = [ 326.28655   250.10121 ] ± [ 1.31444   1.16609 ]
Skew:           alpha_c_right = [ 0.00000 ] ± [ 0.00000  ]   => angle of pixel axes =
90.00000 ± 0.00000 degrees
Distortion:          kc_right = [ -0.28943   0.10690   -0.00059   0.00014  0.00000 ] ±
[ 0.00486   0.00883   0.00022   0.00055  0.00000 ]

Extrinsic parameters (position of right camera wrt left camera):
Rotation vector:          om = [ 0.00669   0.00452  -0.00350 ] ± [ 0.00270   0.00308
0.00029 ]
Translation vector:        T = [ -99.80198   1.12443  0.05041 ] ± [ 0.14200   0.11352
0.49773 ]

Note: The numerical errors are approximately three times the standard deviations
(for reference).
```

After the recalculation, all intrinsic and extrinsic parameters were recomputed, together with all the uncertainties to minimize the reprojection errors on both cameras for all calibration grid locations. The uncertainties on the intrinsic parameters (especially that of the focal values) for both cameras were smaller after stereo calibration. This was because the global stereo optimization performed over a minimal set of unknown parameters. In particular, only one pose unknown (6 DOF) is considered for the location of the calibration grid for each stereo pair. This insures global rigidity of the structure going from left view to right view. The values for the stereo calibration were saved on the file **Calib_Results_stereo.mat** using the menu option **Save stereo calib results**.

Also, the spatial configuration of the two cameras and the calibration planes were displayed in a form of a 3D plot by clicking on the option **Show Extrinsics of the stereo rig** in the stereo toolbox (figure 11).

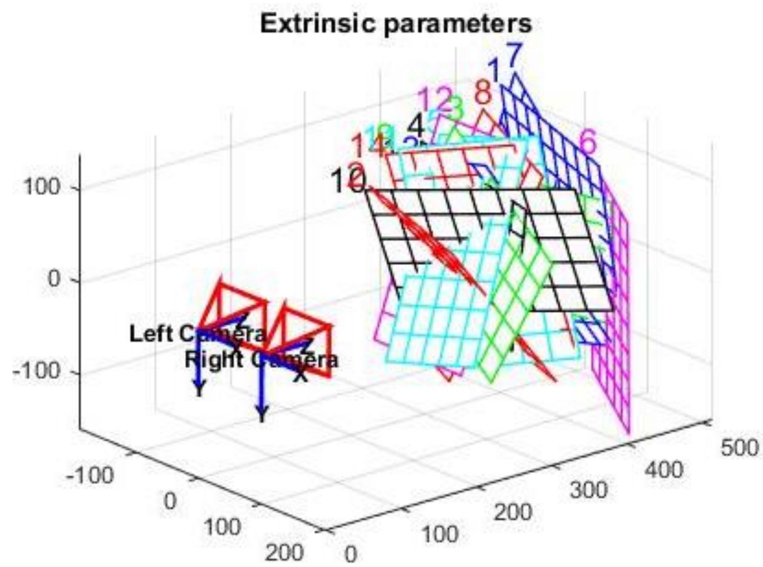


Figure 11: Extrinsic parameters of a stereo camera (3D plot)

The stereo images used for calibration were rectified by clicking on **Rectify the calibration images**. All 14 pairs of images were then stereo rectified (with epipolar lines matching with the horizontal scanned lines).

> Calculating the rotation to be applied to the right and left images in order to bring the epipolar lines aligned with the horizontal scan lines, and in correspondence...
>
> Saving the *NEW* set of intrinsic and extrinsic parameters corresponding to the images *AFTER* rectification under Calib_Results_stereo_rectified.mat...

Pre-computing the necessary data to quickly rectify the images (may take a while depending on the image resolution, but needs to be done only once - even for color images)...

Rectifying all the images (this should be fast)...

Loading image left01.jpg...
Image warping...
Saving image under left_rectified01.bmp...

Loading image right01.jpg...
Image warping...
Saving image under right_rectified01.bmp...

Loading image left02.jpg...
Image warping...
Saving image under left_rectified02.bmp...

Loading image right02.jpg...
Image warping...
Saving image under right_rectified02.bmp...

...

Loading image left13.jpg...
Image warping...
Saving image under left_rectified13.bmp...

Loading image right13.jpg...
Image warping...
Saving image under right_rectified13.bmp...

Loading image left14.jpg...
Image warping...
Saving image under left_rectified14.bmp...

Loading image right14.jpg...
Image warping...
Saving image under right_rectified14.bmp...

Figure 12: Rectified images for the left camera of example 5

Figure 13: Rectified images for the left camera of example 5

The toolbox also included a function **stereo_triangulation.m** that computed the 3D location of a set of points given their left and right image projections. This process is known as stereo triangulation. The website gave the following exercise:

Apply the triangulation function on a simple example: re-compute the 3D location of the grids points extracted on the first image pair {left01.jpg, right01.jpg}. After running through the complete stereo calibration example, the image projections of the grid points on the right and left images are available in the variables x_left_1 *and* x_right_1*.*

```
[Xc_1_left, Xc_1_right] = stereo_triangulation(x_left_1, x_right_1, om, T, fc_left, cc_left, kc_left, alpha_c_left, fc_right, cc_right, kc_right, alpha_c_right);
```

The function had as parameters the intrinsic parameters for the cameras, the extrinsic parameters rotation and translation and the image projections.

The output variables Xc_1_left and Xc_1_right were the 3D coordinates of the points in the left and right camera reference frames respectively. The variables were related to each other through the rigid motion equation $Xc\_1\_right = R * Xc\_1\_left + T$. After that the "intrinsic" geometry of the calibration grid from the triangulated structure Xc_1_left was re-computed by undoing the left camera location encoded by Rc_left_1 and Tc_left_1:

```
X_left_approx_1 = Rc_left_1' * (Xc_1_left - repmat(Tc_left_1,[1 size(Xc_1_left,2)]));
```

The output variable $X\_left\_approx\_1$ was then an approximation of the original 3D structure of the calibration grid stored in X_left_1. For all three coordinates the difference between them were small. To analyze the difference, it was calculated the Euclidian distance between each point. Figure 14 shows a graph with the equivalent distance to each point.

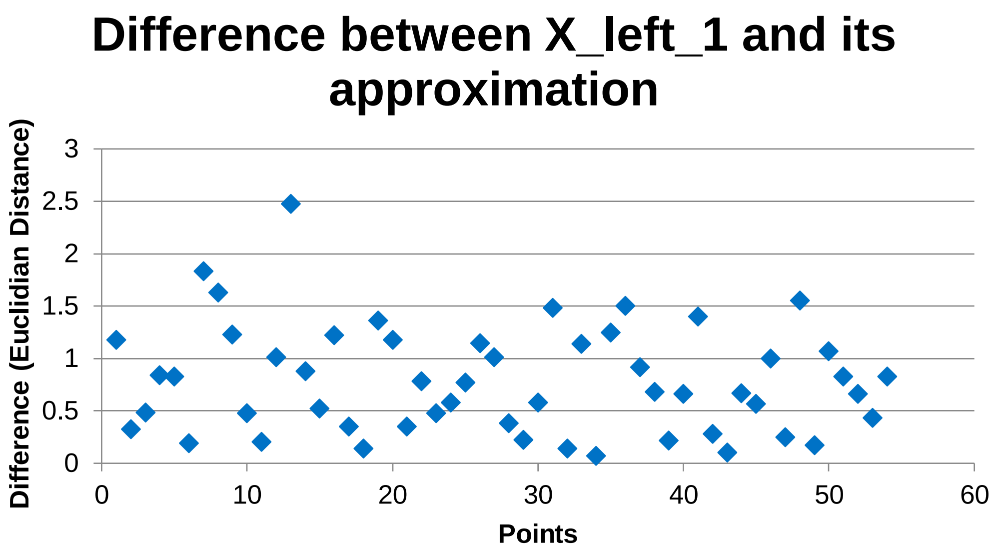


Figure 14: Euclidian distance between X_left_1 and X_left_approx_1

Part 3: Complete calibration

As the last point, the whole process since the calibration of two single cameras to a stereo calibration was performed with data provided from the CV course laboratory. Because the large number of images, only 9 samples were taken from the data set.

1. *Left camera single calibration:*

First the whole process of the single camera calibration explained in part 1 was done with the data of the left camera.

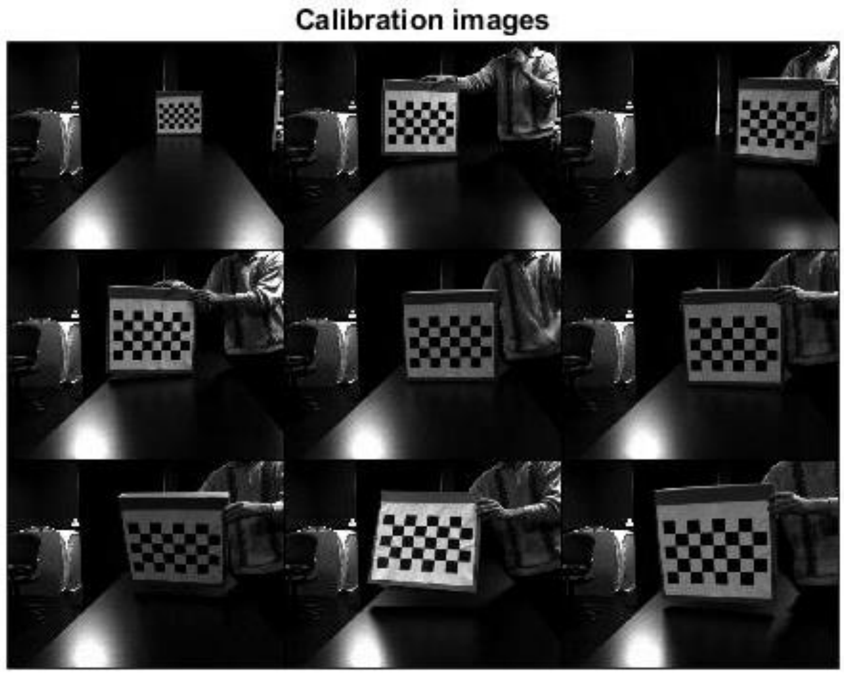


Figure 15: Image loading of 9 samples for single left camera calibration

The manual corner extraction was performed with a windows size of ($wintx = winty = 5$) and $dX = dY = 30\ mm$.

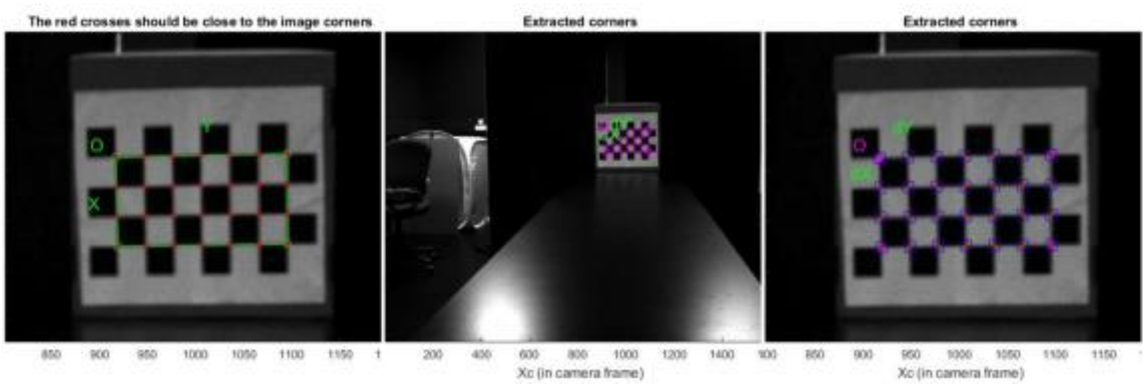


Figure 16: Corner extraction result for image left1

Followed the main calibration step was done (with the manual selection of the corners). Then, the corner extraction was computed automatically for all the images and the calibration was performed again.

Aspect ratio optimized (est_aspect_ratio = 1) -> both components of fc are estimated (DEFAULT).

Principal point optimized (center_optim=1) - (DEFAULT). To reject principal point, set center_optim=0

Skew not optimized (est_alpha=0) - (DEFAULT)

Distortion not fully estimated (defined by the variable est_dist):

Sixth order distortion not estimated (est_dist(5)=0) - (DEFAULT) .

Initialization of the principal point at the center of the image.

Initialization of the intrinsic parameters using the vanishing points of planar patterns.

Initialization of the intrinsic parameters - Number of images: 9

Calibration parameters after initialization:

Focal Length: fc = [1698.84133 1698.84133]

Principal point: cc = [811.50000 611.50000]

Skew: alpha_c = [0.00000] => angle of pixel = 90.00000 degrees

Distortion: kc = [0.00000 0.00000 0.00000 0.00000 0.00000]

Main calibration optimization procedure - Number of images: 9

Gradient descent iterations:

1...2...3...4...5...6...7...8...9...10...11...12...13...14...15...16...17...18...19...20...21...22...23...24...25...done

Estimation of uncertainties...done

Calibration results after optimization (with uncertainties):

Focal Length: fc = [2069.80912 2179.19239] +/- [239.13233 297.80903]

Principal point: cc = [914.01428 1173.94760] +/- [76.99455 204.91597]

Skew: alpha_c = [0.00000] +/- [0.00000] => angle of pixel axes = 90.00000 +/- 0.00000 degrees

Distortion: kc = [0.21959 -0.60682 0.07931 0.00505 0.00000] +/- [0.20782 0.59163 0.04352 0.00795 0.00000]

Pixel error: err = [0.93814 0.87026]

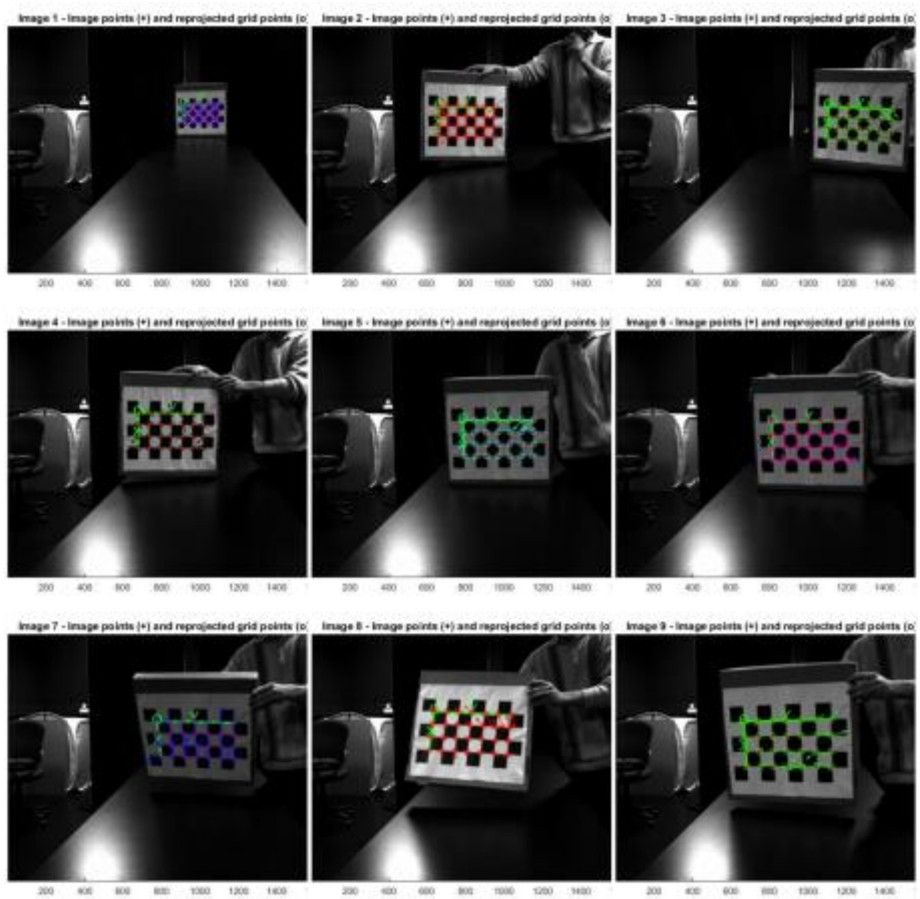


Figure 17: Reprojection result for all images of left camera

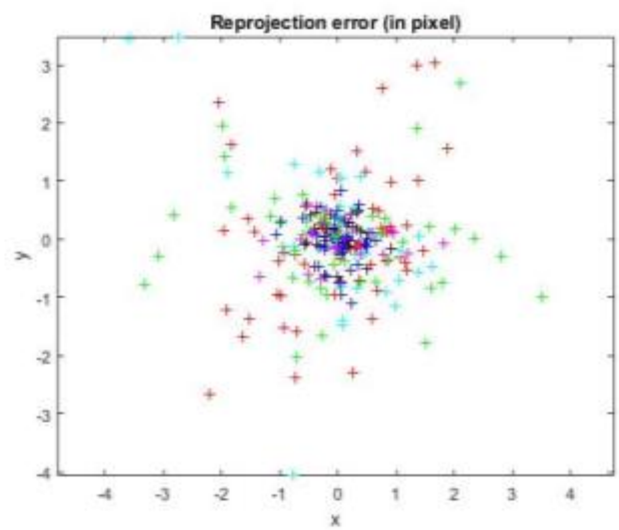


Figure 18: Reprojection error for left camera

Finally, the extrinsic parameters were showed as a 3D representation of relative positions of the grids with respect to the camera.

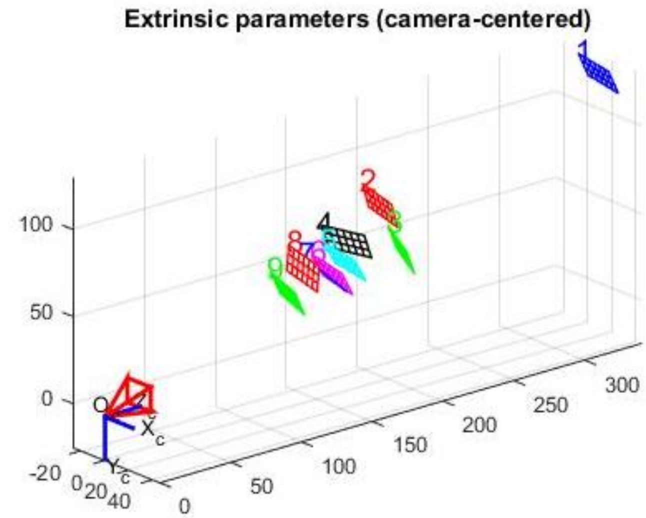


Figure 19: Extrinsic parameters of the left camera (3D plot)

2. *Right camera single calibration:*

Next the same process of the previous section (left camera) was performed on the right camera.

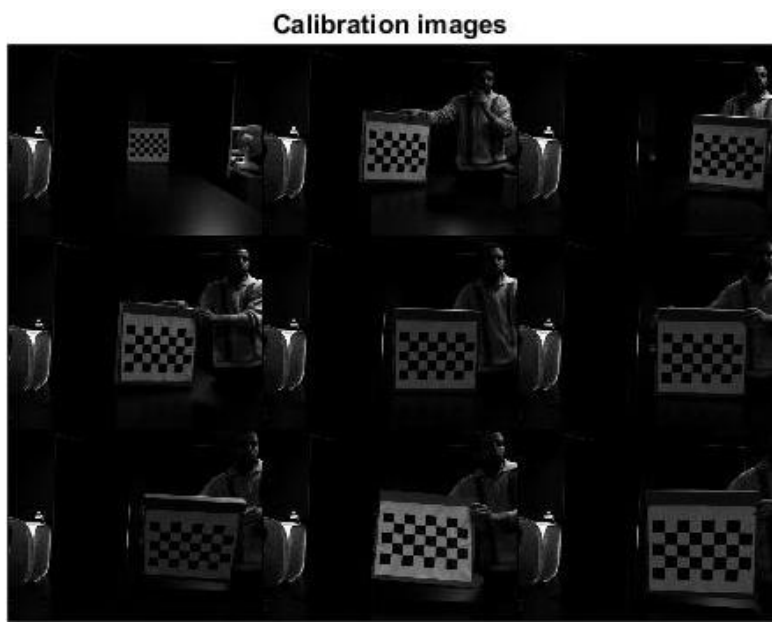


Figure 20: Image loading of 9 samples for single right camera calibration

The manual corner extraction was performed with a windows size of $(wintx = winty = 5)$ and $dX = dY = 30\ mm$.

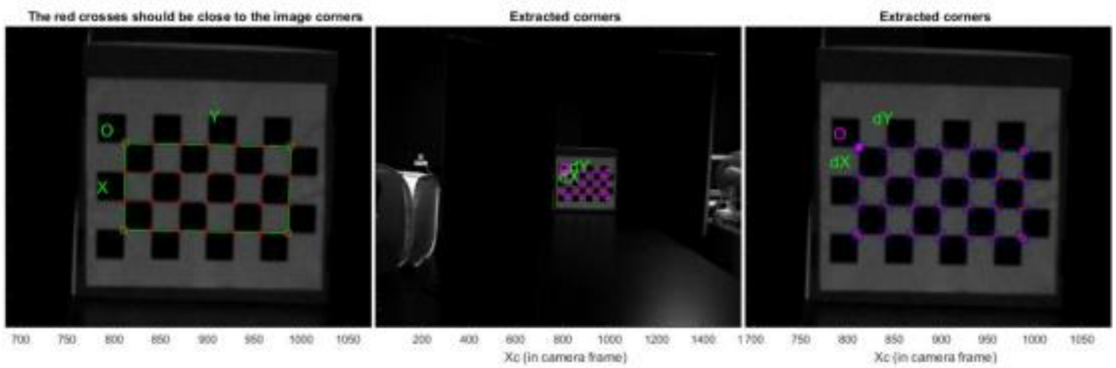


Figure 21: Corner extraction result for image right1

Followed the main calibration step was done (with the manual selection of the corners). Then, the corner extraction was computed automatically for all the images and the calibration was performed again.

```
Aspect ratio optimized (est_aspect_ratio = 1) -> both components of fc are estimated
(DEFAULT).
Principal point optimized (center_optim=1) - (DEFAULT). To reject principal point,
set center_optim=0
Skew not optimized (est_alpha=0) - (DEFAULT)
Distortion not fully estimated (defined by the variable est_dist):
     Sixth order distortion not estimated (est_dist(5)=0) - (DEFAULT) .
Initialization of the principal point at the center of the image.
Initialization of the intrinsic parameters using the vanishing points of planar
patterns.

Initialization of the intrinsic parameters - Number of images: 9
Calibration parameters after initialization:
Focal Length:          fc = [ 965.48252   965.48252 ]
Principal point:       cc = [ 811.50000   611.50000 ]
```

Skew: alpha_c = [0.00000] => angle of pixel = 90.00000 degrees

Distortion: kc = [0.00000 0.00000 0.00000 0.00000 0.00000]

Main calibration optimization procedure - Number of images: 9

Gradient descent iterations:

1...2...3...4...5...6...7...8...9...10...11...12...13...14...15...16...17...18...19...20...21...22...23...24...25...26...27...28...29...done

Estimation of uncertainties...done

Calibration results after optimization (with uncertainties):

Focal Length: fc = [1222.13085 1247.86570] +/- [184.50002 174.54162]

Principal point: cc = [623.94095 836.17236] +/- [155.16486 89.78247]

Skew: alpha_c = [0.00000] +/- [0.00000] => angle of pixel axes = 90.00000 +/- 0.00000 degrees

Distortion: kc = [-0.26499 0.12426 0.00304 0.03344 0.00000] +/- [0.14109 0.09260 0.00801 0.03435 0.00000]

Pixel error: err = [1.09770 1.07444]

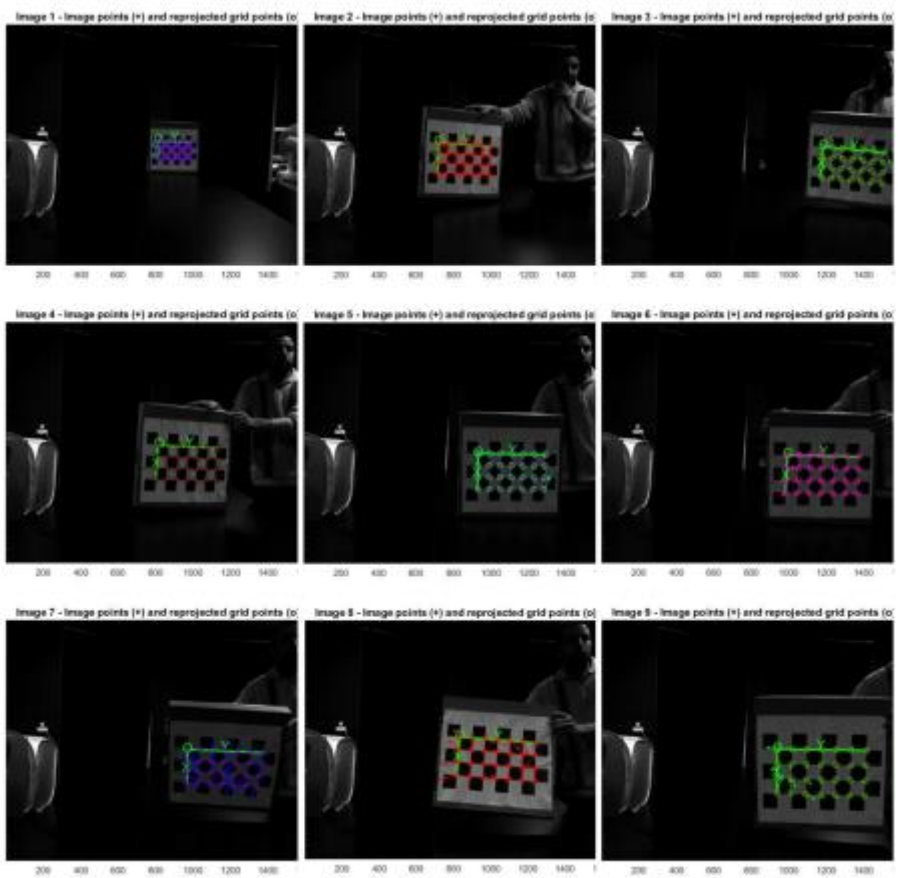


Figure 22: Reprojection result for all images of right camera

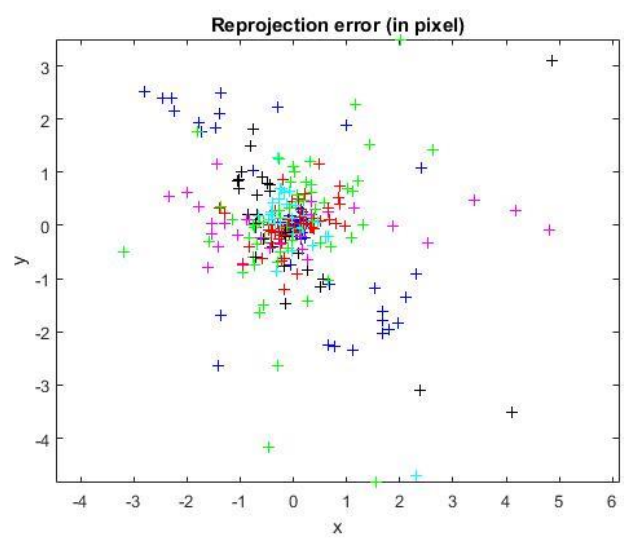


Figure 23: Reprojection error for right camera

Finally, the extrinsic parameters were showed as a 3D representation of relative positions of the grids with respect to the camera.

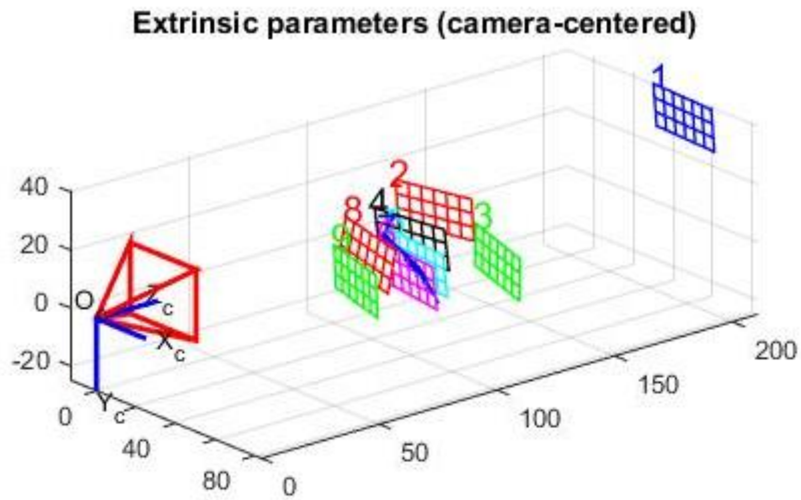


Figure 24: Extrinsic parameters of the right camera (3D plot)

3. *Stereo calibration:*

The stereo calibration for the images provided followed the same steps that were executed for the part 2. First, the calibration files created in the previous sections (for each camera) were loaded:

Stereo calibration parameters after loading the individual calibration files:

Intrinsic parameters of left camera:

Focal Length: fc_left = [1920.56548 1981.73927] ± [147.22398 162.39946]

Principal point: cc_left = [815.43965 926.18687] ± [82.33442 118.10416]

Skew: alpha_c_left = [0.00000] ± [0.00000] => angle of pixel axes = 90.00000 ± 0.00000 degrees

Distortion: kc_left = [-0.06852 0.09689 0.03561 0.00078 0.00000] ± [0.10642 0.28627 0.01946 0.00676 0.00000]

Intrinsic parameters of right camera:

Focal Length: fc_right = [1867.68045 1884.88522] ± [67.90937 69.34424]

Principal point: cc_right = [830.92643 911.77216] ± [47.37088 40.05330]

Skew: alpha_c_right = [0.00000] ± [0.00000] => angle of pixel axes = 90.00000 ± 0.00000 degrees

Distortion: kc_right = [-0.13479 0.16312 0.02180 0.00518 0.00000] ± [0.06948 0.20839 0.00609 0.00635 0.00000]

Extrinsic parameters (position of right camera wrt left camera):

Rotation vector: om = [-0.12241 -0.18020 -0.00157]

Translation vector: T = [39.15644 0.00637 -5.35164]

After that, the global stereo optimization procedure was done and the results saved. Figure 25 shows the spatial configuration of the two cameras.

```
Recomputation of the intrinsic parameters of the left camera
(recompute_intrinsic_left = 1)
Recomputation of the intrinsic parameters of the right camera
(recompute_intrinsic_right = 1)

Main stereo calibration optimization procedure - Number of pairs of images: 9
Gradient descent iterations: 1...2...3...4...5...6...done
Estimation of uncertainties...done

Stereo calibration parameters after optimization:
Intrinsic parameters of left camera:
Focal Length:       fc_left = [ 1911.45048   1920.77673 ] ± [ 47.72844   50.28532 ]
Principal point:    cc_left = [ 731.78733   819.74352 ] ± [ 78.15146   49.81733 ]
Skew:           alpha_c_left = [ 0.00000 ] ± [ 0.00000  ]   => angle of pixel axes =
90.00000 ± 0.00000 degrees
Distortion:         kc_left = [ -0.18911   0.28031   0.00531   -0.00130  0.00000 ] ±
[ 0.07608   0.20462   0.00774   0.00573  0.00000 ]

Intrinsic parameters of right camera:
Focal Length:       fc_right = [ 1934.85374   1950.89190 ] ± [ 46.25807   48.39339 ]
Principal point:    cc_right = [ 888.10947   856.32613 ] ± [ 60.63102   52.83045 ]
Skew:           alpha_c_right = [ 0.00000 ] ± [ 0.00000  ]   => angle of pixel axes =
90.00000 ± 0.00000 degrees
Distortion:         kc_right = [ -0.00298   -0.09571   0.02207   0.00110  0.00000 ] ±
[ 0.11282   0.56405   0.00887   0.00942  0.00000 ]

Extrinsic parameters (position of right camera wrt left camera):
Rotation vector:        om = [ -0.08262   -0.26134  0.00074 ] ± [ 0.02714   0.04690
0.00633 ]
Translation vector:      T = [ 40.01750   2.42852  2.21168 ] ± [ 0.14953   0.16109
1.51290 ]
```

Note: The numerical errors are approximately three times the standard deviations (for reference).

Saving the stereo calibration results in Calib_Results_stereo.mat...

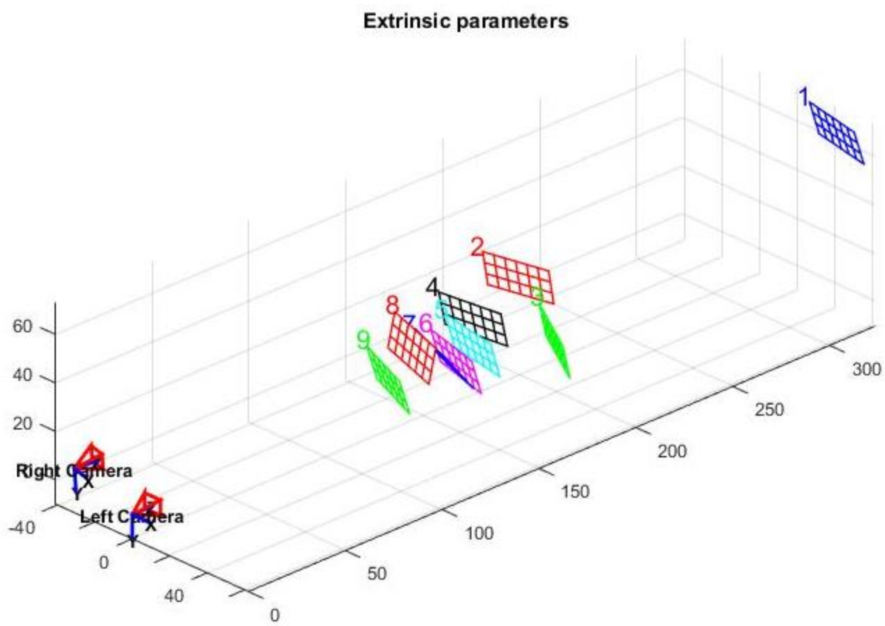


Figure 25: Extrinsic parameters of a stereo camera (3D)

The stereo calibration was followed by rectification of the stereo images used:

Calculating the rotation to be applied to the right and left images in order to bring the epipolar lines aligned with the horizontal scan lines, and in correspondence...

Saving the *NEW* set of intrinsic and extrinsic parameters corresponding to the images *AFTER* rectification under Calib_Results_stereo_rectified.mat...

Pre-computing the necessary data to quickly rectify the images (may take a while depending on the image resolution, but needs to be done only once - even for color images)...

Rectifying all the images (this should be fast)...

Loading image left1.jpg...
Image warping...
Saving image under left_rectified1.bmp...

Loading image right1.jpg...

```
Image warping...
Saving image under right_rectified1.bmp...
...

Loading image left9.jpg...
Image warping...
Saving image under left_rectified9.bmp...

Loading image right9.jpg...
Image warping...
Saving image under right_rectified9.bmp...
```

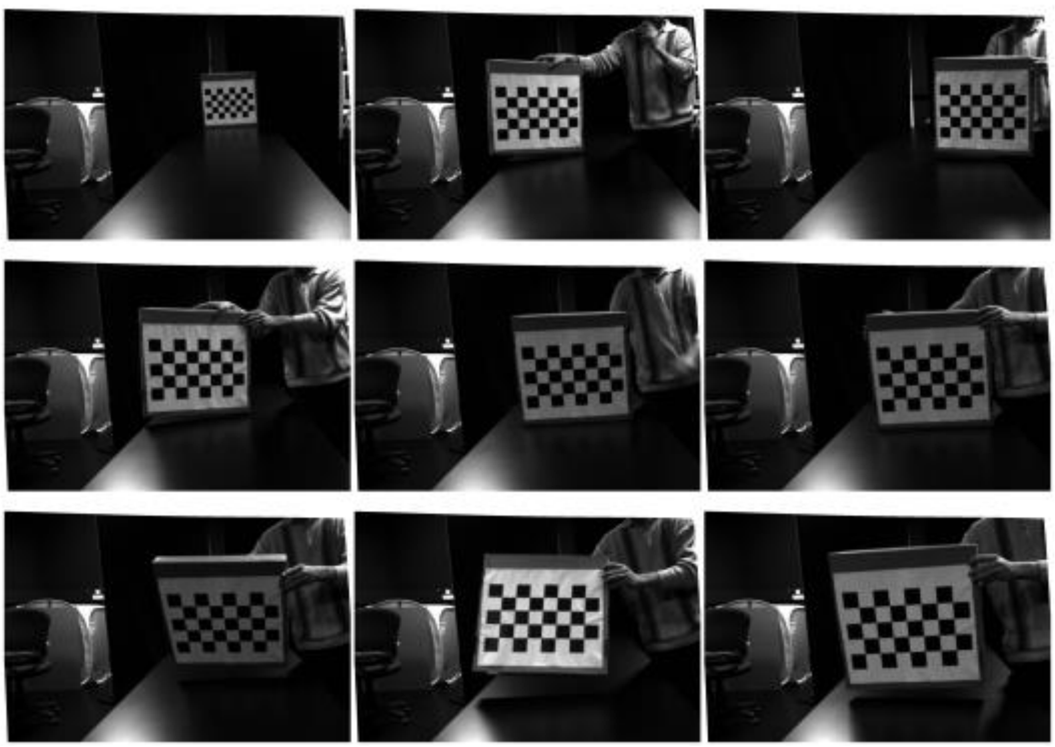


Figure 26: Rectified images for the left camera

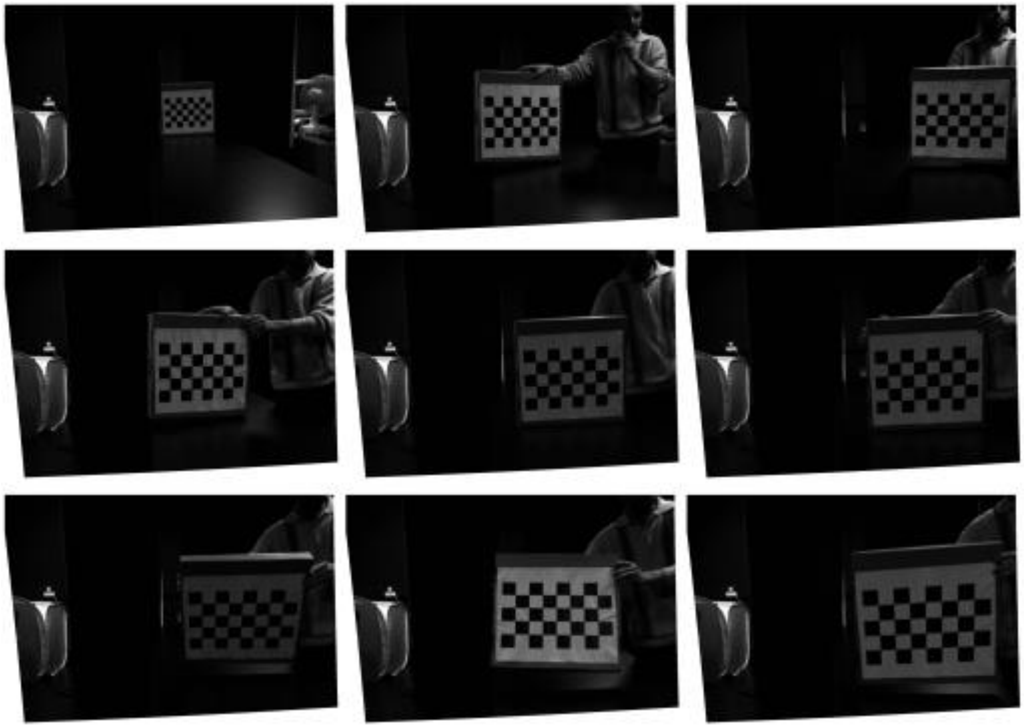


Figure 27: Rectified images for the right camera

The 3D coordinates of the points of the left and right camera reference frames (Xc_1_left and Xc_1_right, respectively) and approximation of X_left_1 were also calculated. Figure 28 shows the Euclidian distance between the points and his approximated value.

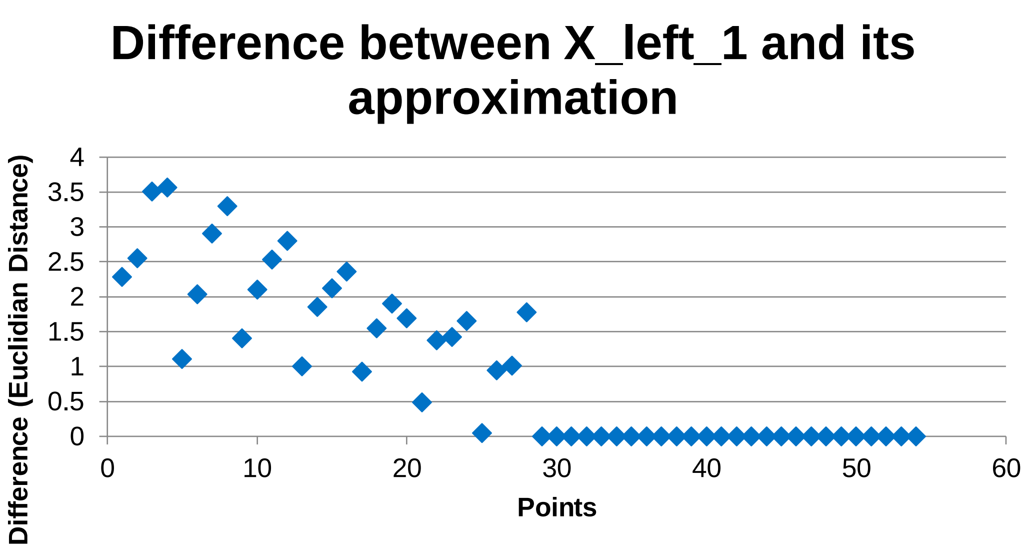


Figure 28: Euclidian distance between X_left_1 and X_left_approx_1

Different from the graph on Figure 14, some points have larger distance between them (between 1 and 4), but also around half of the points have a complete match with the approximation calculated.

CONCLUSIONS

The conclusion of the second lab session were the following:

- ✓ Using the CalTech tool was possible to learn and performed a single camera calibration and a stereo calibration.
- ✓ It was hard to compare the results from two data sets. Both were from different situations, resolution, etc., therefore giving different results to each stage of the calibration.
- ✓ Some difficulties were faced to mark the corners for the second data set. The chessboard was too far from the camera, which made it complicated to have accuracy on the position for the corners.
- ✓ The rectification results were not so easy to understand. Perhaps the final shape of the images depended a lot from the position of the object related to the camera. For example: the rectification for the first data was almost the same for left and right, but for the second data the images were on different shapes.

REFERENCES

[1] Gary Bradski & Adrian Kaehler, *Learning OpenCV: Computer vision with the OpenCV library*, O’Reilly Media, Inc., 2008.