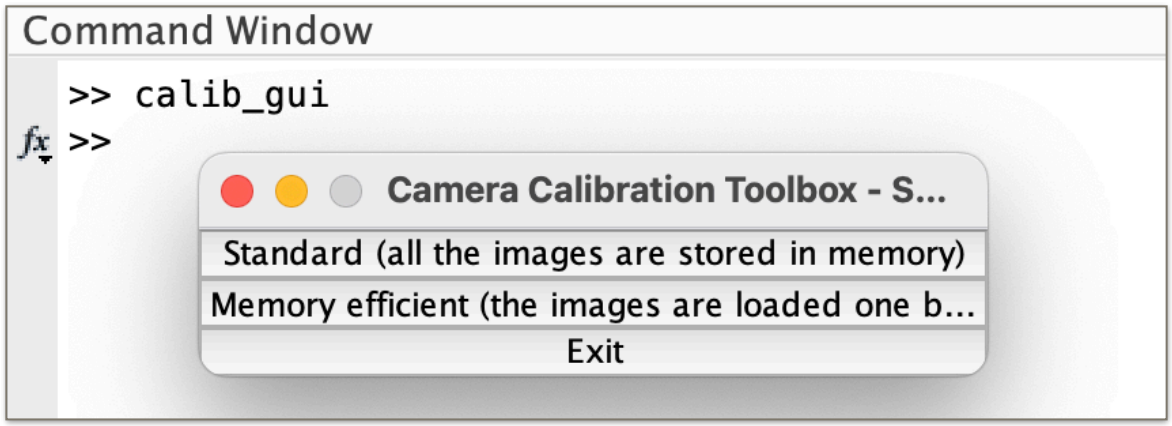
CALTECH

Camera Calibration Toolbox for MATLAB

Goktug Korkulu
27026

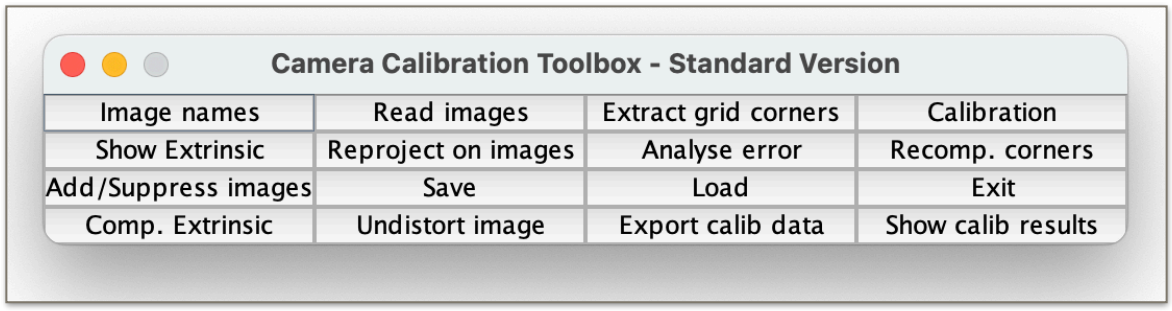
STEP 1 : initializing Camera Calibration Tool

When inside the Caltech Calibration Folder, command ***'calib_gui'*** and the Cameral calibration toolbox will show up as shown below.



Here, press ***'Standard (all the images are stored in memory)'*** option in order to proceed by storing all the images that will be entered as input to be kept in the RAM of the computer.

STEP 2 : Reading the images

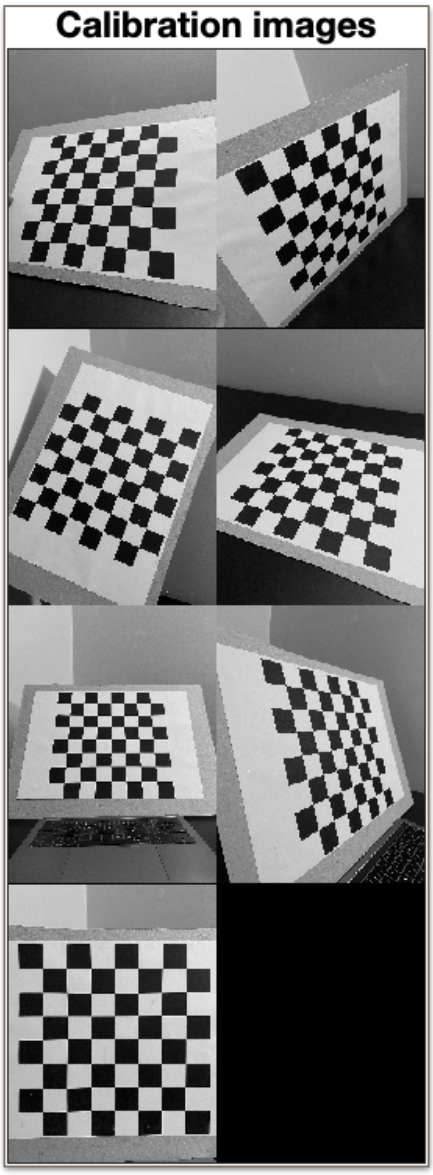


After pressing 'Standart' button, the window above will show up. From here, click on the ***'Image names'*** button in the **Camera calibration tool** window.

Enter the basename of the calibration images (**image**) and the image format (**jpg**). All the images (the 7 of them) are then loaded in memory (through the command **Read images** that is automatically executed) in the variables **I_1**, **I_2** ,..., **I_7**. The number of images is stored in the variable **n_ima (=7 here)**. The MATLAB window should look like this:

```
Basename camera calibration images (without number nor suffix): image
Image format: ([]='r'='ras', 'b'='bmp', 't'='tif', 'p'='pgm', 'j'='jpg', 'm'='ppm') jpg
Loading image 1...2...3...4...5...6...7...
done
```

The complete set of images is also shown in thumbnail format:



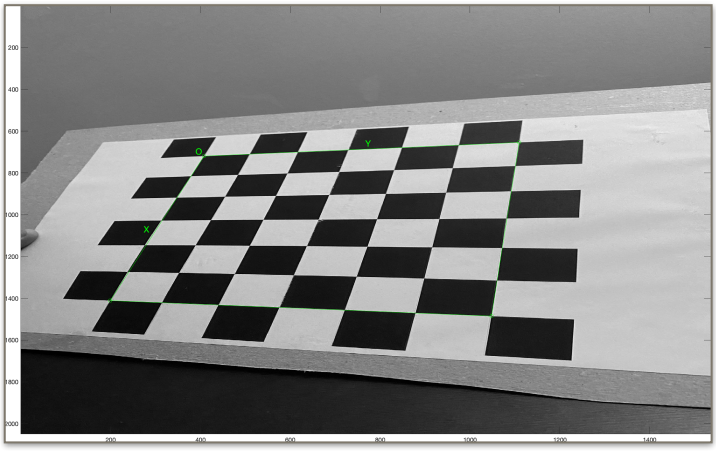
STEP 3 : Extract the grid corners

Click on the ***Extract grid corners*** button in the ***Camera calibration tool*** window. Press "enter" (with an empty argument) to select all the images. Then, select the default window size of the corner finder: **wintx=winty=21** by pressing "enter" with empty arguments to the **wintx** and **winty** question. This leads to a effective window of size 43x43 pixels.

```
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 ([] = 21) =
winty ([] = 21) =
Window size = 43x43
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)?
```

The corner extraction engine includes an automatic mechanism for counting the number of squares in the grid. This tool is specially convenient when working with a large number of images since the user does not have to manually enter the number of squares in both x and y directions of the pattern. On some very rare occasions however, this code may not predict the right number of squares. This would typically happen when calibrating lenses with extreme distortions. At this point in the corner extraction procedure, the program gives the option to the user to disable the automatic square counting code. In that special mode, the user would be prompted for the square count for every image. In this present example, it is perfectly appropriate to keep working in the default mode (i.e. with automatic square counting activated), and therefore, simply press "enter" with an empty argument.

Now, click on the four extreme corners on the rectangular checkerboard pattern. The first clicked point is selected to be associated to the origin point of the reference frame attached to the grid. The other three points of the rectangular grid can be clicked in any order.

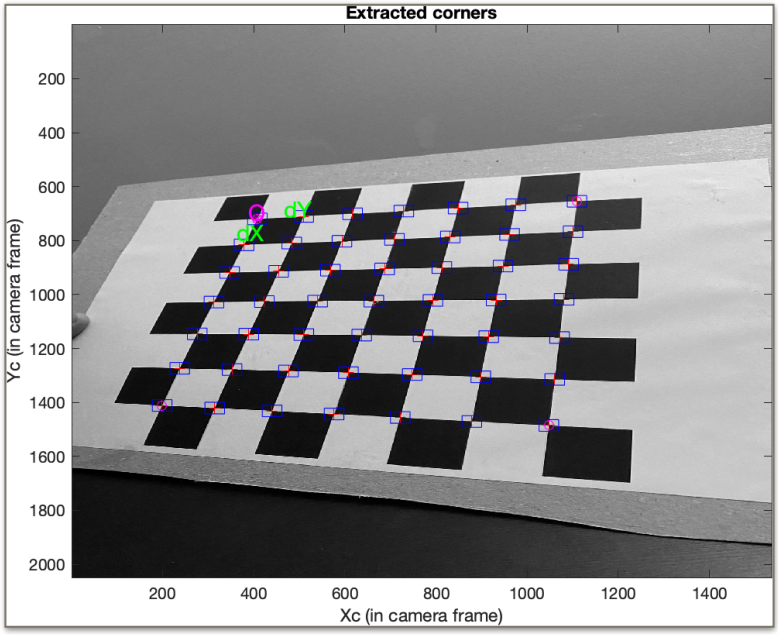


Enter the sizes **dX** and **dY** in X and Y of each square in the grid (in this case, **dX=dY=36mm**). Note that you could have just pressed "enter" with an

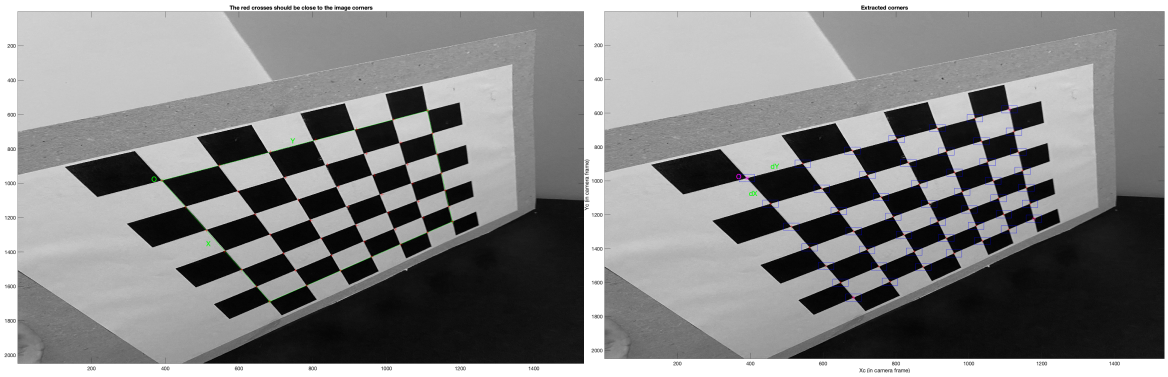
empty argument to select the default values. The program automatically counts the number of squares in both dimensions, and shows the predicted grid corners in absence of distortion. If the predicted corners are close to the real image corners, then the following step may be skipped (if there is not much image distortion). This is the case in that present image: the predicted corners are close enough to the real image corners. Therefore, it is not necessary to "help" the software to detect the image corners by entering a guess for radial distortion coefficient. Press "enter", and the corners are automatically extracted using those positions as initial guess.

```
Processing image 1...
Using (wintx,winty)=(21,21) - Window size = 43x43      (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) = 36
Size dY of each square along the Y direction ([]=100mm) = 36
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...
```

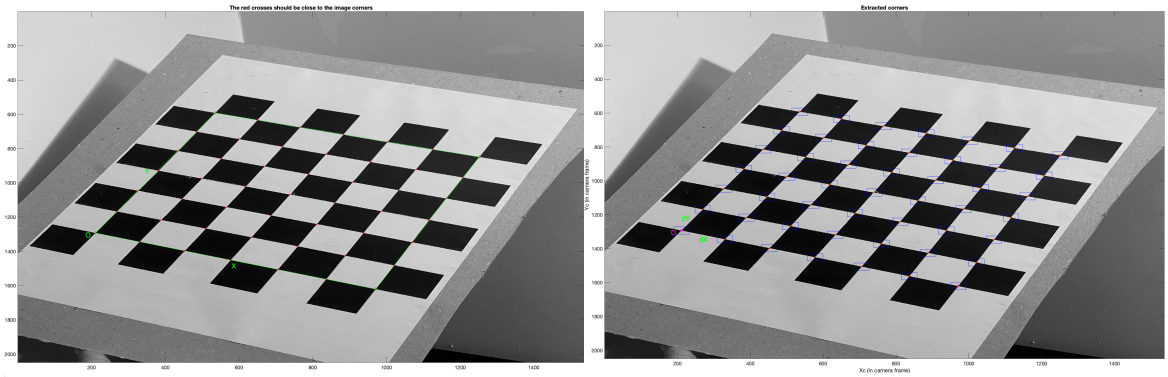
The image corners are then automatically extracted, and displayed on the figure below (the blue squares around the corner points show the limits of the corner finder window).The corners are extracted to an accuracy of about 0.1 pixel.



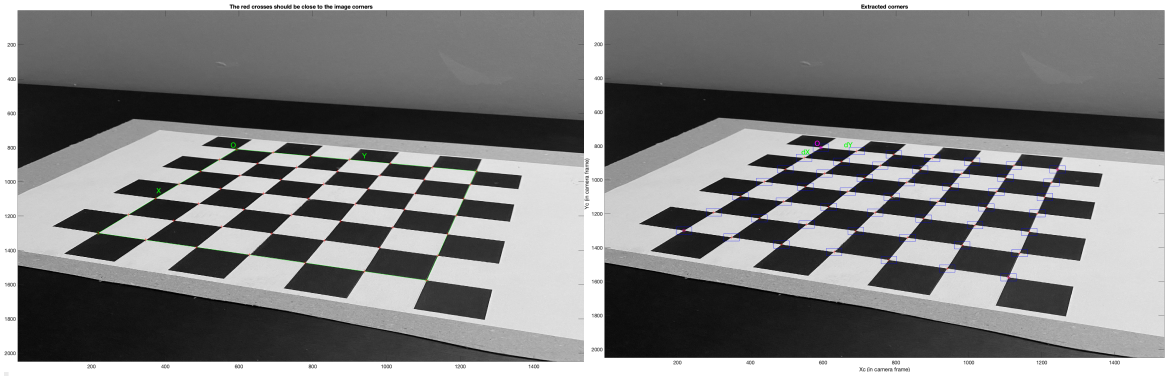
Follow the same procedure for the 2nd, 3rd, ... , 7th images. Observe the square dimensions **dX**, **dY** are always kept to their original values (36mm) since the predicted corners are quite close enough to the real image corners to allow for an effective corner extraction.



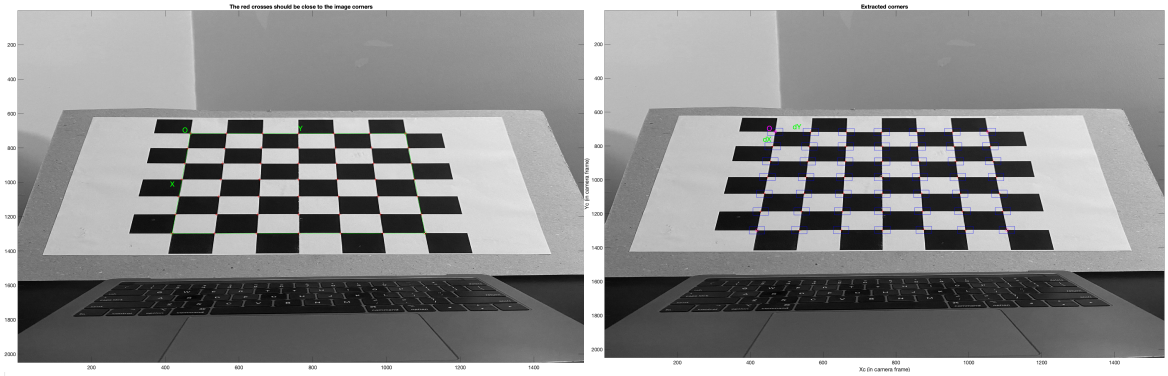

```
Processing image 2...
Using (wintx,winty)=(21,21) - Window size = 43x43      (Note: To reset the window size, run script clearwin)
Click on the four extreme corners of the rectangular complete pattern (the first clicked corner is the origin)...
Size of each square along the X direction: dX=36mm
Size of each square along the Y direction: dY=36mm   (Note: To reset the size of the squares, clear the variables dX and dY)
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...
```



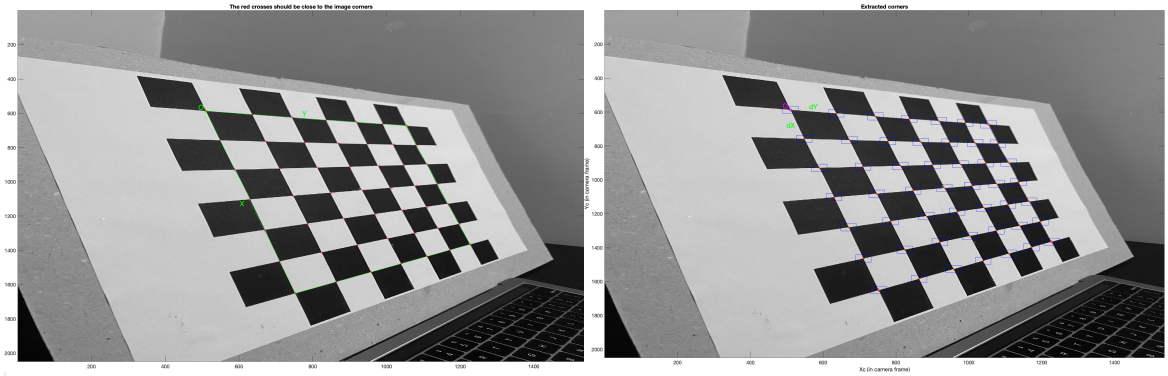

```
Processing image 3...
Using (wintx,winty)=(21,21) - Window size = 43x43      (Note: To reset the window size, run script clearwin)
Click on the four extreme corners of the rectangular complete pattern (the first clicked corner is the origin)...
Size of each square along the X direction: dX=36mm
Size of each square along the Y direction: dY=36mm   (Note: To reset the size of the squares, clear the variables dX and dY)
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...
```



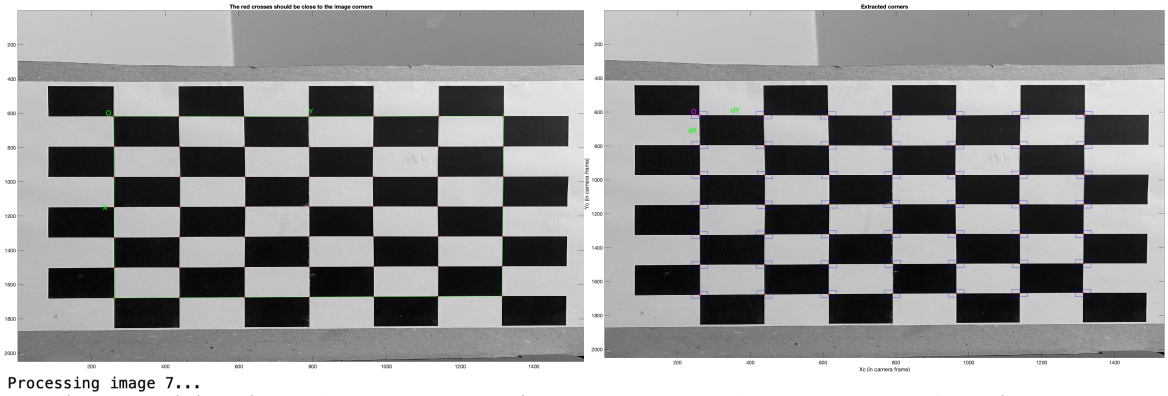

```
Processing image 4...
Using (wintx,winty)=(21,21) - Window size = 43x43      (Note: To reset the window size, run script clearwin)
Click on the four extreme corners of the rectangular complete pattern (the first clicked corner is the origin)...
Size of each square along the X direction: dX=36mm
Size of each square along the Y direction: dY=36mm   (Note: To reset the size of the squares, clear the variables dX and dY)
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 5...
Using (wintx,winty)=(21,21) - Window size = 43x43      (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 of each square along the X direction: dX=36mm
Size of each square along the Y direction: dY=36mm   (Note: To reset the size of the squares, clear the variables dX and dY)
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...
```

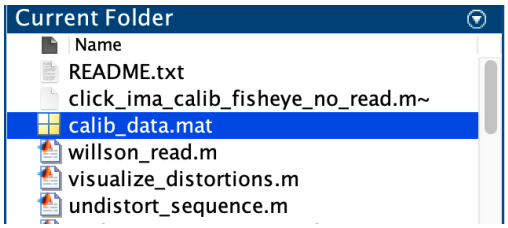



```
Processing image 6...
Using (wintx,winty)=(21,21) - Window size = 43x43      (Note: To reset the window size, run script clearwin)
Click on the four extreme corners of the rectangular complete pattern (the first clicked corner is the origin)...
Size of each square along the X direction: dX=36mm
Size of each square along the Y direction: dY=36mm   (Note: To reset the size of the squares, clear the variables dX and dY)
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 7...
Using (wintx,winty)=(21,21) - Window size = 43x43      (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 of each square along the X direction: dX=36mm
Size of each square along the Y direction: dY=36mm   (Note: To reset the size of the squares, clear the variables dX and dY)
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...
done
```

After corner extraction, the MATLAB data file **calib_data.mat** is automatically generated. This file contains all the information gathered throughout the corner extraction stage (image coordinates, corresponding 3D grid coordinates, grid sizes, ...). This file is only created in case of emergency when for example MATLAB is abruptly terminated before saving. Loading this file would prevent you from having to click again on the images.



STEP 4 : Main Calibration step

After corner extraction, click on the button **Calibration** of the **Camera calibration tool** to run the main camera calibration procedure.

Calibration is done in two steps: first initialization, and then nonlinear optimization. The initialization step computes a closed-form solution for the calibration parameters based not including any lens distortion (program name: **init_calib_param.m**). The non-linear optimization step minimizes 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*7 DOF extrinsic => 53 parameters).

The Calibration parameters are stored in a number of variables. After calibration, the list of parameters may be stored in the MATLAB file **Calib_Results** by clicking on **Save**.

The list of intrinsic parameters:

- **Focal length:** The focal length in pixels is stored in the 2x1 vector **fc**. Observe that **fc(1)** and **fc(2)** are the focal distance (a unique value in mm) expressed in units of horizontal and vertical pixels. Both components of the vector **fc** are usually very similar. The ratio **fc(2)/ fc(1)**, often called "aspect ratio", is different from 1 if the pixel in the CCD array are not square. Therefore, the camera model naturally handles non-square pixels.
- **Principal point:** The principal point coordinates are stored in the 2x1 vector **cc**.
- **Skew coefficient:** The skew coefficient defining the angle between the x and y pixel axes is stored in the scalar **alpha_c**. The coefficient **alpha_c** encodes the angle between the x and y sensor axes. Consequently, pixels are even allowed to be non-rectangular. Some authors refer to that type of model as "affine distortion" model.
- **Distortions:** The image distortion coefficients (radial and tangential distortions) are stored in the 5x1 vector **kc**.

```
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: 7


Calibration parameters after initialization:

Focal Length:          fc = [ 1542.45357   1542.45357 ]
Principal point:       cc = [ 767.50000   1023.50000 ]
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: 7
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...done
Estimation of uncertainties...done


Calibration results after optimization (with uncertainties):

Focal Length:          fc = [ 1534.95853   1535.89377 ] +/- [ 7.94207   7.84857 ]
Principal point:       cc = [ 782.99760   1040.79770 ] +/- [ 8.48403   9.63315 ]
Skew:             alpha_c = [ 0.00000 ] +/- [ 0.00000  ]   => angle of pixel axes = 90.00000 +/- 0.00000 degrees
Distortion:            kc = [ 0.17648   -0.47868   0.00326   0.00173  0.00000 ] +/- [ 0.02203   0.08657   0.00274   0.00244  0.00000 ]
Pixel error:          err = [ 0.84669   0.66660 ]

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

The list of extrinsic parameters:

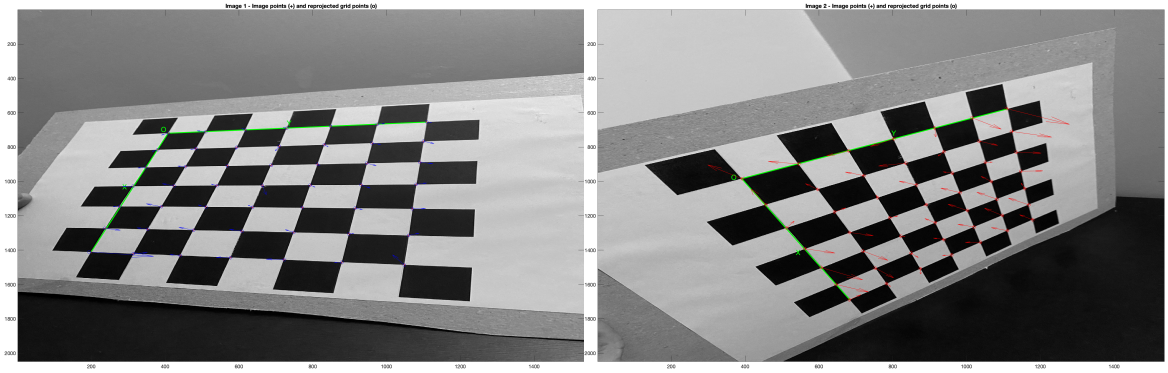
- **Rotations:** A set of **n_ima** 3x3 rotation matrices **Rc_1**, **Rc_2**,.., **Rc_7** (assuming **n_ima=7**).
- **Translations:** A set of **n_ima** 3x1 vectors **Tc_1**, **Tc_2**,.., **Tc_7** (assuming **n_ima=7**).

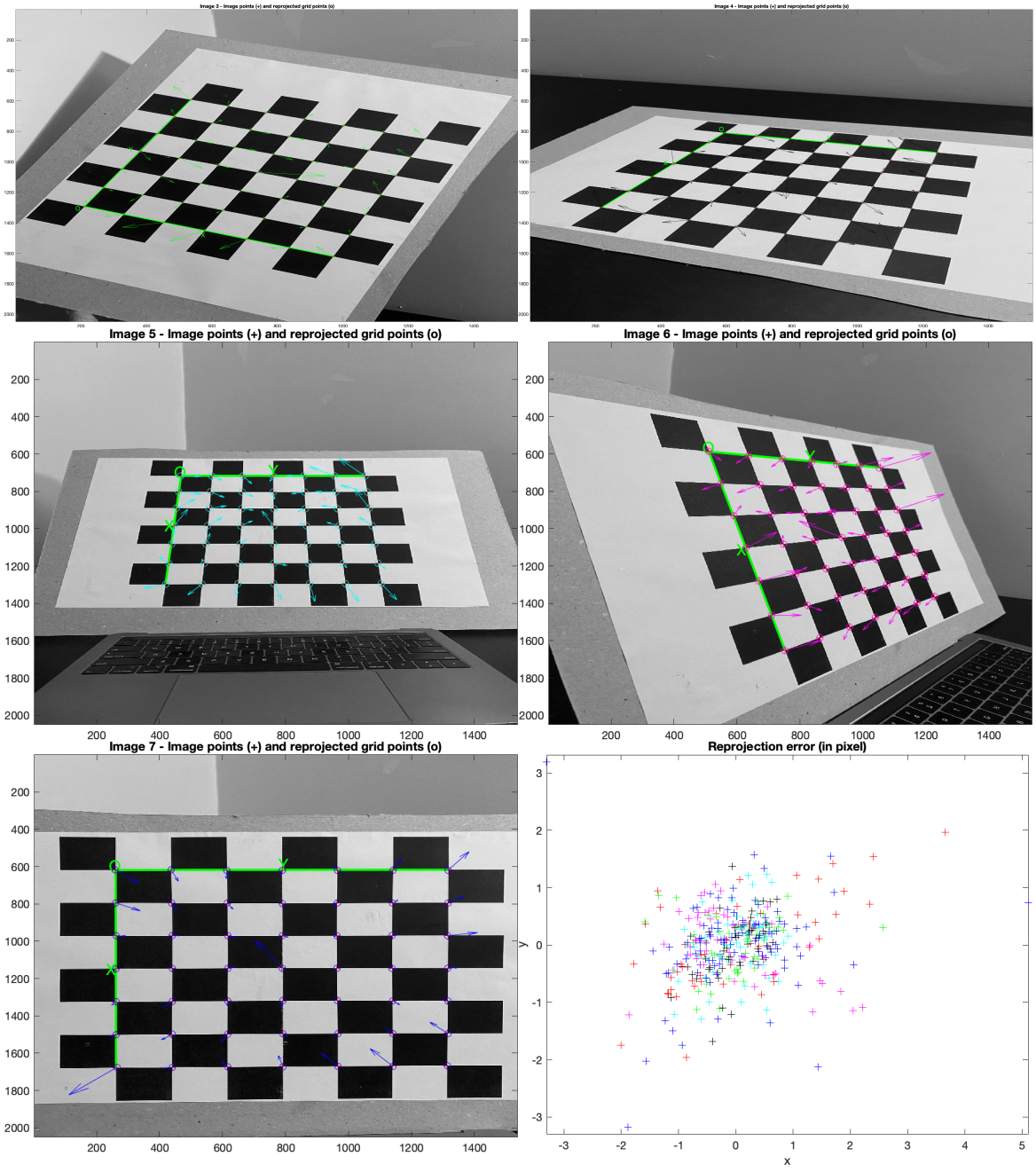
Rc_1	[-0.1463,0.9394,-0.3100;0.8952,-0.0076,-0.4456;-0.4209,-0.3427,-0.8399]
Rc_2	[0.2187,0.7200,0.6586;0.8226,-0.4991,0.2724;0.5248,0.4821,-0.7015]
Rc_3	[0.9254,0.3757,-0.0492;0.3605,-0.9130,-0.1910;-0.1167,0.1590,-0.9804]
Rc_4	[-0.3331,0.9428,-0.0114;0.6537,0.2222,-0.7234;-0.6795,-0.2484,-0.6904]
Rc_5	[0.0081,0.9998,0.0164;0.9260,-0.0013,-0.3775;-0.3774,0.0183,-0.9259]
Rc_6	[0.2283,0.6409,0.7328;0.9713,-0.0982,-0.2167;-0.0669,0.7613,-0.6450]
Rc_7	[0.0016,0.9998,0.0183;0.9999,-0.0018,0.0136;0.0136,0.0182,-0.9997]
Tc_1	[-117.2163;-100.7670;484.1489]
Tc_2	[-73.4901;-10.6441;290.7578]
Tc_3	[-153.1323;66.8233;416.2011]
Tc_4	[-64.6190;-79.0859;523.0127]
Tc_5	[-116.4133;-119.8137;569.6159]
Tc_6	[-55.7511;-92.6117;315.2129]
Tc_7	[-106.8462;-86.5578;315.9154]
Tc_error_1	[2.6586;3.0669;2.2239]
Tc_error_2	[1.6322;1.8386;1.9136]
Tc_error_3	[2.3215;2.6662;2.2570]
Tc_error_4	[2.8621;3.2907;2.1996]
Tc_error_5	[3.1530;3.5789;2.9524]
Tc_error_6	[1.7682;1.9685;1.9612]
Tc_error_7	[1.7631;2.0232;1.8157]

For this particular input images and calculations, the extrinsic parameters for each image are shown above. Note that **Tc_error_1, ..., Tc_error_7** are error magnitudes in translations parameter.

STEP 5 : Reprojection

Click on **Reproject on images** in the **Camera calibration tool** to show the reprojections of the grids onto the original images. These projections are computed based on the current intrinsic and extrinsic parameters. Input an empty string (just press "enter") to the question **Number(s) of image(s) to show ([] = all images)** to indicate that you want to show all the images.



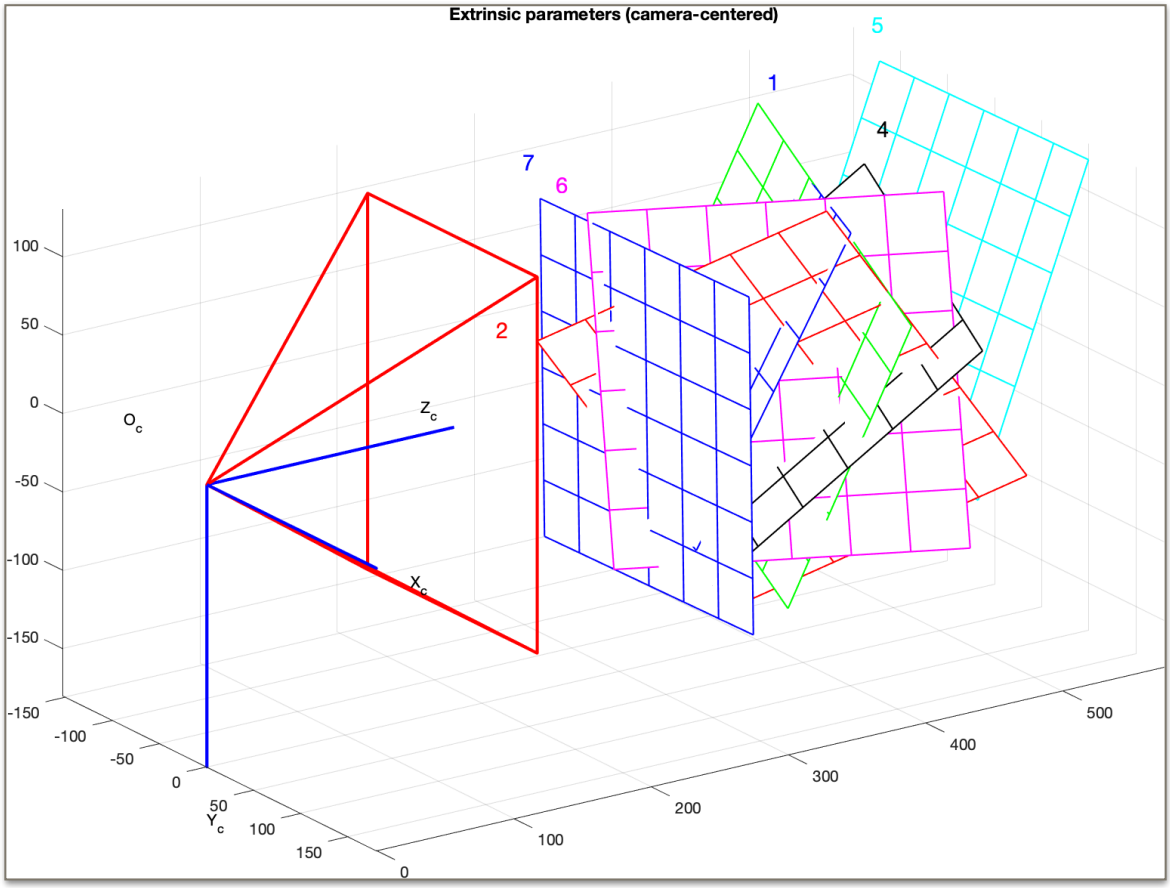


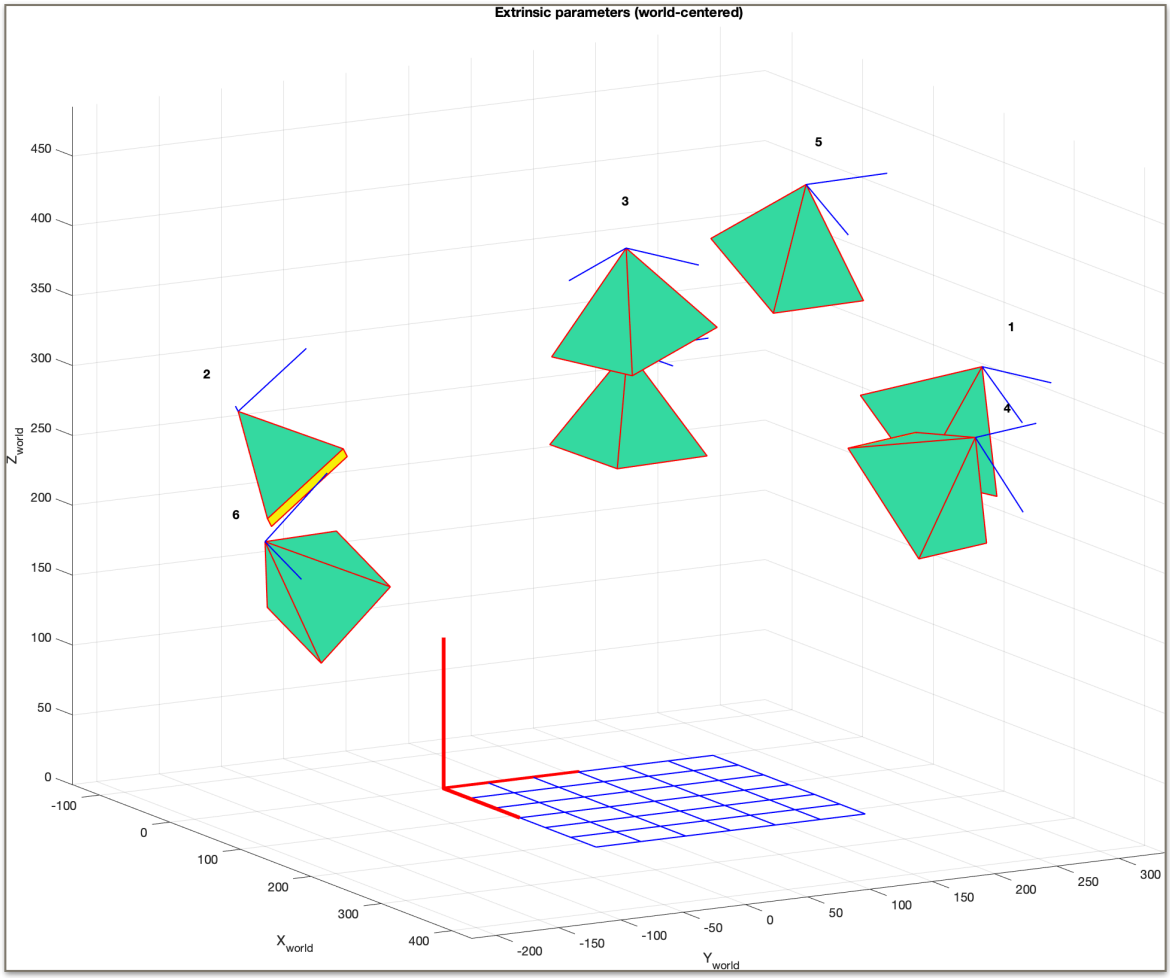
First 7 images are reprojections of the grids onto the original images that are computed based on the current intrinsic and extrinsic parameters. They show seven input images with the detected corners (red crosses) and the reprojected grid corners (circles). And the last image is the error plot.

STEP 6 : Show Extrinsic

Click on **Show Extrinsic** in the **Camera calibration tool**. The extrinsic parameters (relative positions of the grids with respect to the camera) are then shown in a form of a 3D plot.

On this figure, the frame **(O_c,X_c,Y_c,Z_c)** is the camera reference frame. The red pyramid corresponds to the effective field of view of the camera defined by the image plane. To switch from a "camera-centered" view to a "world-centered" view, just click on the **Switch to world-centered view** button located at the bottom-right corner of the figure.





On this new figure, every camera position and orientation is represented by a green pyramid.

STEP 7 : Recomp. corners

Looking back at the error plot, notice that the reprojection error is large across a large number of figures. The reason for that is that we have not done a very careful job at extracting the corners on some highly distorted images (a better job could have been done by using the predicted distortion option). Nevertheless, we can correct for that now by recomputing the image corners on all images automatically. Here is the way it is going to be done: press on the **Recomp. corners** button in the main **Camera calibration tool** and select a corner finder window size of **wintx = winty = 5** (the default values).

```
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...
done
```

To the question **Number(s) of image(s) to process ([] = all images)** press "enter" with an empty argument to recompute the corners on all the images. Enter then the mode of extraction: the automatic mode (auto) uses the re-projected grid as initial guess locations for the corner, the manual mode lets the user extract the corners manually (the traditional corner extraction method). In the present case, the reprojected grid points are very close to the actual image corners. Therefore, we select the automatic mode: press "enter" with an empty string. The corners on all images are then recomputed.

Run then another calibration optimization by clicking on **Calibration.**

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

Focal Length:          fc = [ 1534.99623   1536.05385 ] +/- [ 7.33429   7.26339 ]
Principal point:       cc = [ 780.15729   1037.36403 ] +/- [ 7.93053   9.00333 ]
Skew:             alpha_c = [ 0.00000 ] +/- [ 0.00000  ]   => angle of pixel axes = 90.00000 +/- 0.00000 degrees
Distortion:            kc = [ 0.17242   -0.45281   0.00230   0.00071  0.00000 ] +/- [ 0.02022   0.07884   0.00257   0.00227  0.00000 ]
Pixel error:          err = [ 0.78040   0.62125 ]

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

Observe that no initialization step was performed (the optimization started from the previous calibration result). And also compare the new err value and previous err value, you will see there is an improvement in terms of error values.

After optimization, click on **Save** to save the calibration results (intrinsic and extrinsic) in the MATLAB file **Calib_Results.mat**.

Once again, click on **Reproject on images** to reproject the grids onto the original calibration images. Click on **Analyse error** to view the new reprojection error (observe that the error is slightly smaller than before).

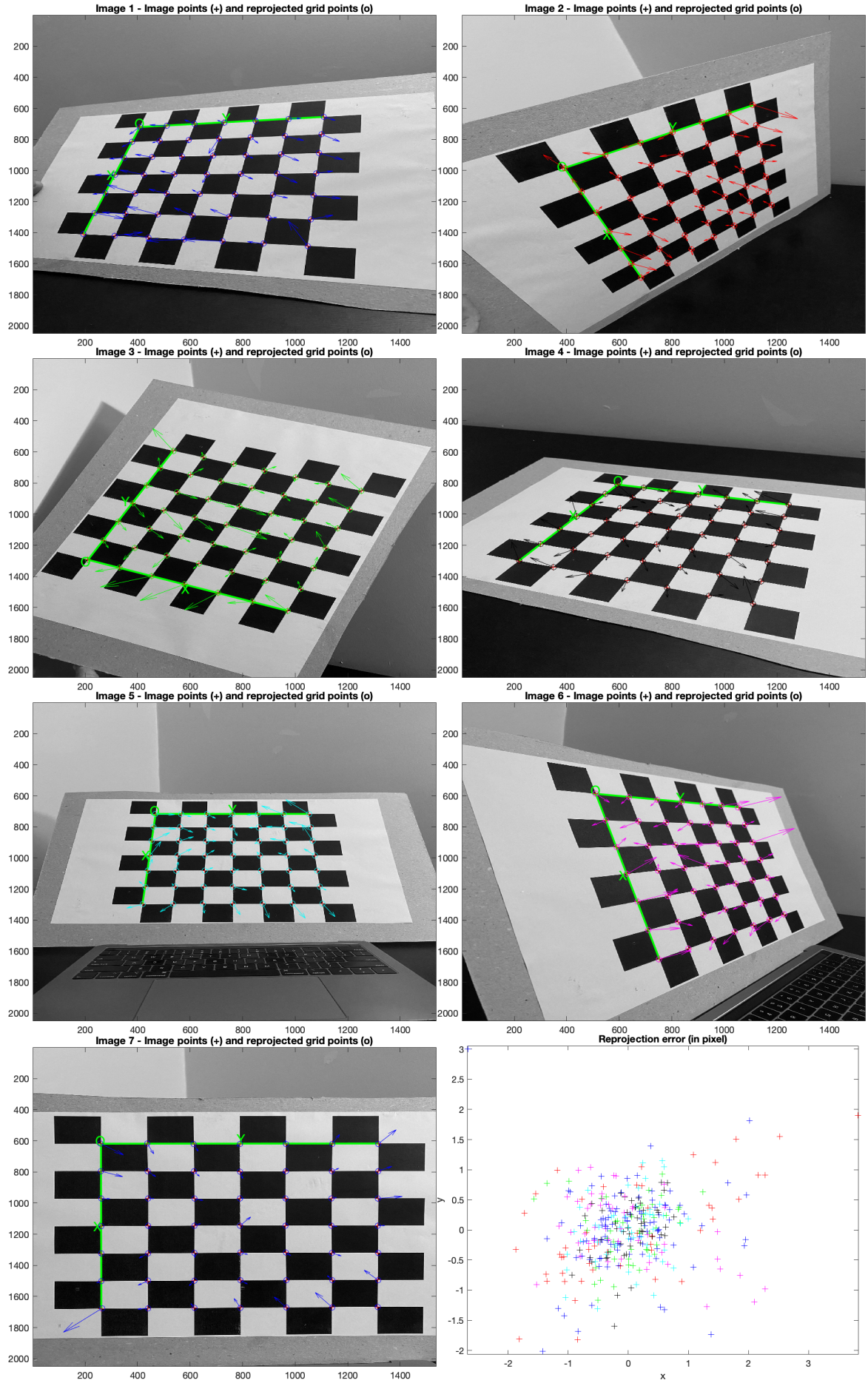


Image 1 - Image points (+) and reprojected grid points (o)
Image 2 - Image points (+) and reprojected grid points (o)
Image 3 - Image points (+) and reprojected grid points (o)
Image 4 - Image points (+) and reprojected grid points (o)
Image 5 - Image points (+) and reprojected grid points (o)
Image 6 - Image points (+) and reprojected grid points (o)
Image 7 - Image points (+) and reprojected grid points (o)
Reprojection error (in pixel)
x
y

The tool **Analyse error** allows you to inspect which points correspond to large errors. Click on **Analyse error** and click on the figure region. After clicking the point at the most right-top, the following information appears in the main Matlab window.

```
Selected image: 2
Selected point index: 1
Pattern coordinates (in units of (dX,dY)): (X,Y)=(0,6)
Image coordinates (in pixel): (1110.21,576.35)
Pixel error = (3.82285,1.90082)
Window size: (wintx,winty) = (5,5)
```

This means that the corresponding point is on image 2, at the grid coordinate (0,6) in the calibration grid. The error inspection tool is very useful in cases where the corners have been badly extracted on one or several images.

STEP 8 : Skew Coefficient calculation

Let us now run a calibration by including the skew factor **alpha_c** describing the angle between the x and y pixel axes. For that, set the variable **est_alpha** to one (at the MATLAB prompt). Let us also fit the radial distortion model up to the 6th order (up to now, it was up to the 4th order, with tangential distortion). For that, set the last entry of the vector **est_dist** to one. i.e. ;

```
>> est_dist = [1;1;1;1;1];
>> est_alpha = 1;
```

Then, run a new calibration by clicking on **Calibration.**

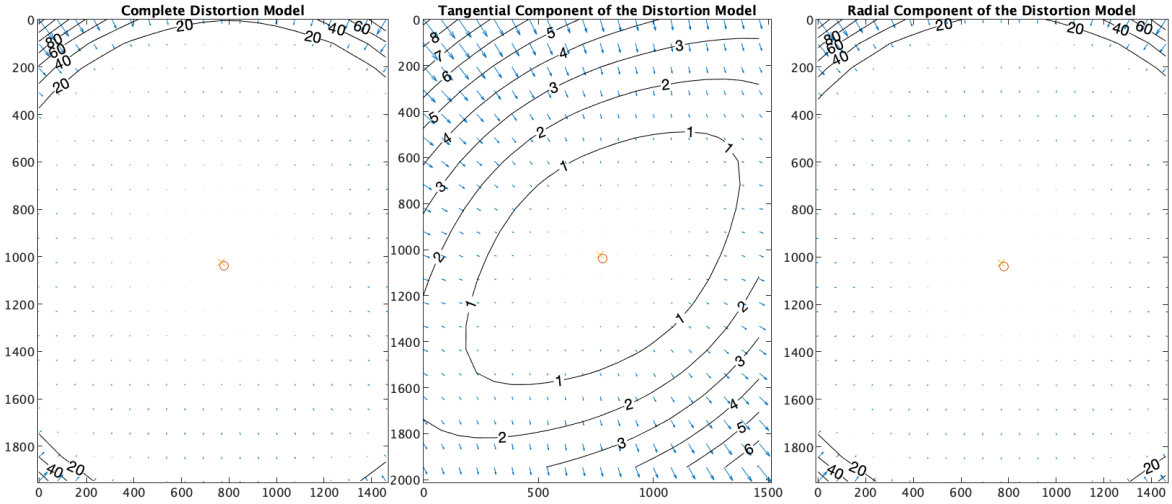
```
Calibration results after optimization (with uncertainties):

Focal Length:          fc = [ 1525.91716   1527.19562 ] +/- [ 7.11392   7.02875 ]
Principal point:       cc = [ 782.03292   1035.53289 ] +/- [ 7.35199   7.80418 ]
Skew:             alpha_c = [ -0.00275 ] +/- [ 0.00096  ]   => angle of pixel axes = 90.15735 +/- 0.05520 degrees
Distortion:            kc = [ 0.27949   -1.42904   0.00164   0.00232  2.35131 ] +/- [ 0.04197   0.34536   0.00221   0.00210  0.79951 ]
Pixel error:          err = [ 0.72468   0.53641 ]

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

Observe that after optimization, the skew coefficient is very close to zero (**alpha_c = 0.00096**). This leads to an angle between x and y pixel axes very close to 90 degrees (90.15735 degrees). This justifies the previous assumption of rectangular pixels (**alpha_c = 0**). In addition, notice that the uncertainty on the 6th order radial distortion coefficient is very large (the uncertainty is much larger than the absolute value of the coefficient). In this case, it is preferable to disable its estimation. In this case, set the last entry of est_dist to zero by [1;1;1;1;0].

```
>> est_dist = [1;1;1;1;0]
est_dist =
     1
     1
     1
     1
     0

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 optimized (est_alpha=1). To disable skew estimation, set est_alpha=0.
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: 7
Gradient descent iterations: 1...2...3...4...5...6...7...8...9...10...11...12...13...14...15...16...17...18...19...20...done
Estimation of uncertainties...done


Calibration results after optimization (with uncertainties):

Focal Length:          fc = [ 1530.39437   1531.61322 ] +/- [ 7.19434   7.10978 ]
Principal point:       cc = [ 778.97736   1038.30573 ] +/- [ 7.56256   8.56396 ]
Skew:             alpha_c = [ -0.00280 ] +/- [ 0.00101  ]   => angle of pixel axes = 90.16022 +/- 0.05786 degrees
Distortion:            kc = [ 0.16937   -0.44271   0.00210   0.00185  0.00000 ] +/- [ 0.01918   0.07436   0.00244   0.00219  0.00000 ]
Pixel error:          err = [ 0.74779   0.58732 ]

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

Judging the result of calibration satisfactory, let us save the current calibration parameters by clicking on **Save**.

In order to make a decision on the appropriate distortion model to use, it is sometimes very useful to visualize the effect of distortions on the pixel image, and the importance of the radial component versus the tangential component of distortion. For this purpose, run the script **visualize_distortions** at the MATLAB prompt. The three following images are then produced.



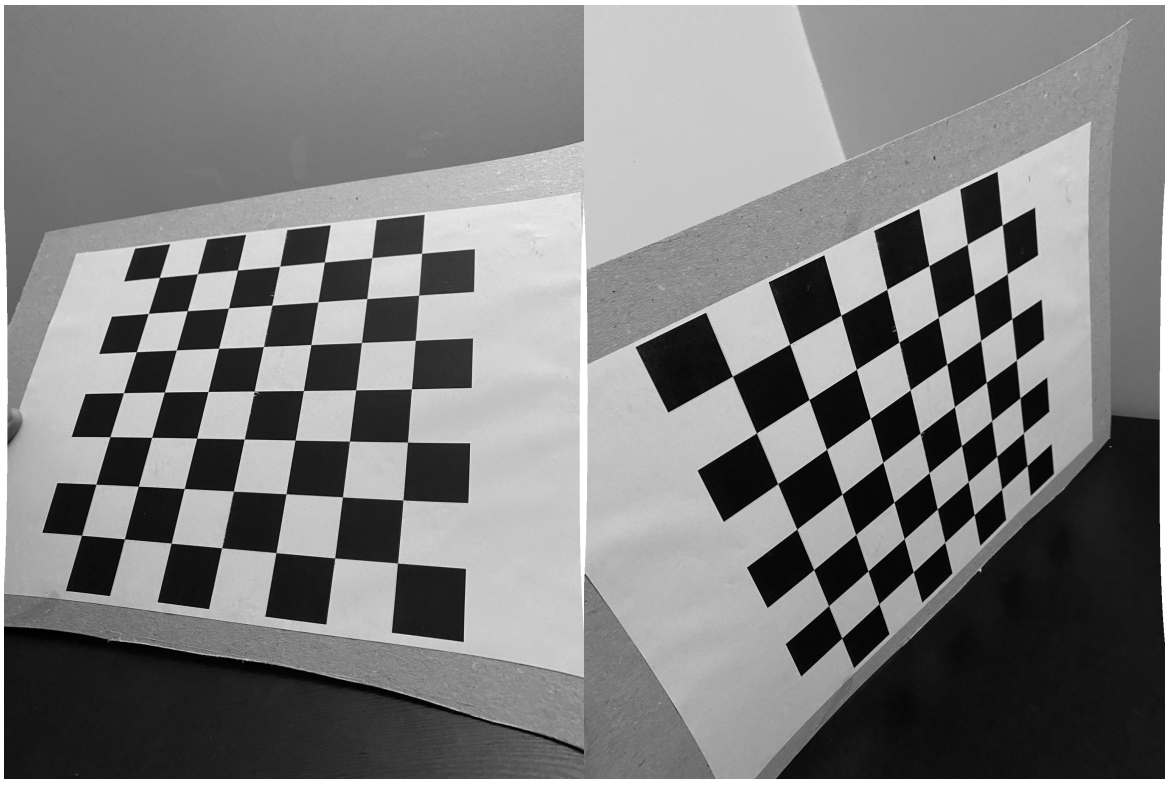
The first figure shows the impact of the complete distortion model (radial + tangential) on each pixel of the image. Each arrow represents the effective displacement of a pixel induced by the lens distortion. Observe that points at the corners of the image are displaced by as much as 80 pixels. The second figure shows the impact of the tangential component of distortion. On this plot, the maximum induced displacement is 8 pixel. Finally, the third figure shows the impact of the radial component of distortion. This plot is very similar to the full distortion plot, showing the tangential component could very well be discarded in the complete distortion model. On the three figures, the cross indicates the center of the image, and the circle the location of the principal point.

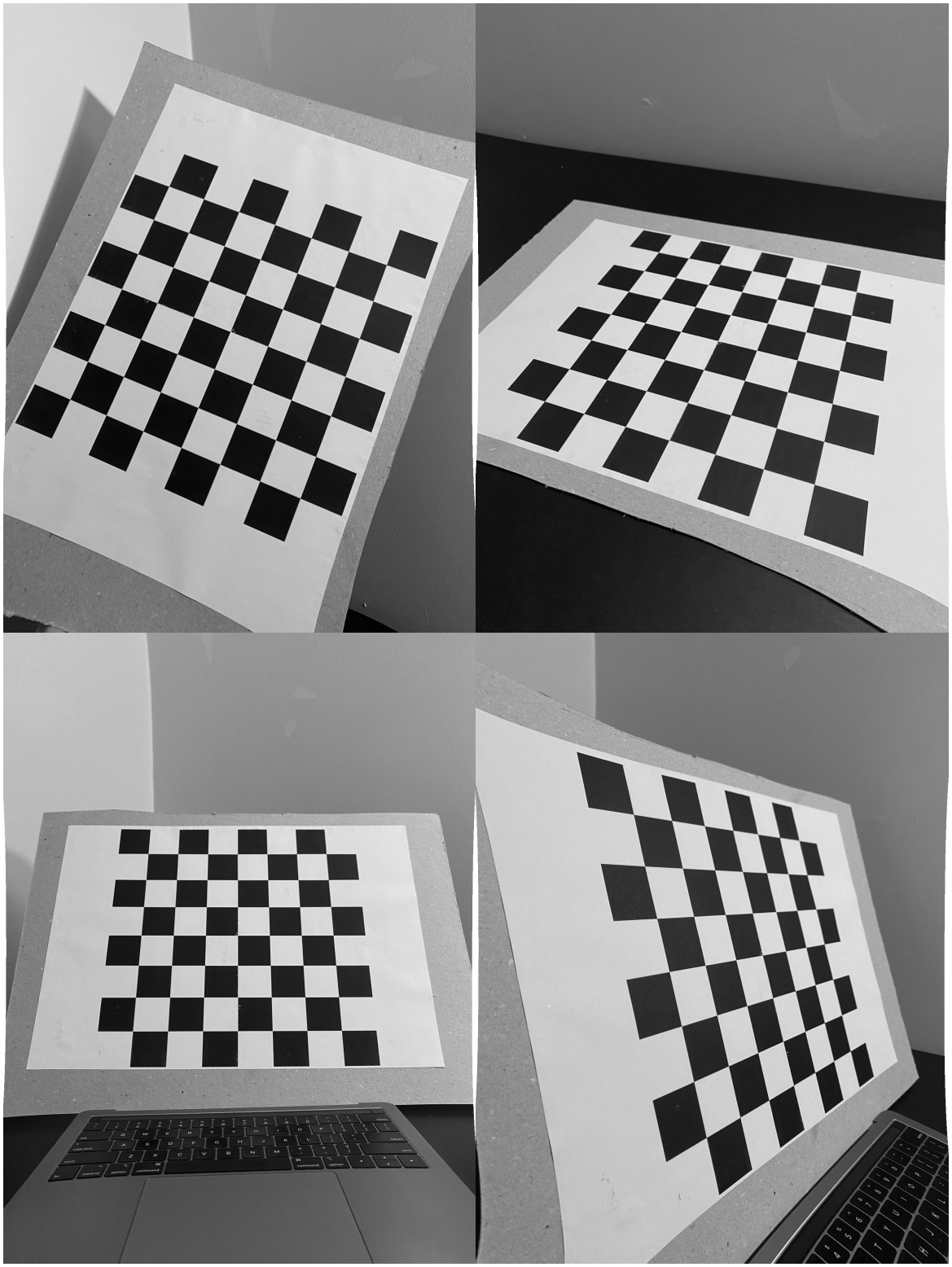
STEP 9 : Undistort images

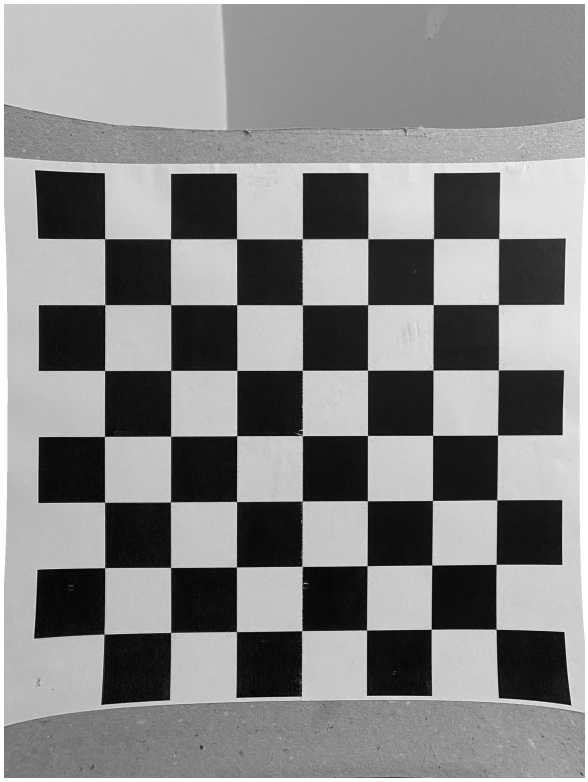
This function helps you generate the undistorted version of one or multiple images given pre-computed intrinsic camera parameters. Let us now undistort the complete set of calibration images. Click on **Undistort image** in the **Camera calibration tool**, and enter an empty argument to the first question. All the calibration images are then undistorted and saved onto disk under **image_rect1.bpm**, **image_rect2.bpm**, ..., **image_rect7.bpm.**

```
Program that undistorts images
The intrinsic camera parameters are assumed to be known (previously computed)

Do you want to undistort all the calibration images ([],0) or a new image (1)?
Saving undistorted image under image_rect1.bmp...
Saving undistorted image under image_rect2.bmp...
Saving undistorted image under image_rect3.bmp...
Saving undistorted image under image_rect4.bmp...
Saving undistorted image under image_rect5.bmp...
Saving undistorted image under image_rect6.bmp...
Saving undistorted image under image_rect7.bmp...
done
```

Below, undistorted images are shown respectively.







INTRODUCTION

In this part of the laboratory, I will be exploring the Caltech Camera Calibration Toolbox for MATLAB in order to calibrate my camera and obtain the intrinsic/extrinsic parameters.

Specifically, I will be using 7 different images of the SAME 2D calibration object whose picture has been captured with different angles but same camera. The brief aim of this laboratory is to calculate the intrinsic parameter of the camera that is used and then calculate the extrinsic parameters of each image that was captured by this camera. For instance, at the end, there will be 2 different representation of frames (called word-centered and camera-centered) and those frames will only be implemented by the non-changing intrinsic parameters of the camera and changing extrinsic parameters of each image.

There will be 9 steps which will be going through in order to investigate the input images and therefore estimate the internal and external parameters of projection matrix. They are namely;

STEP 1 : initializing Camera Calibration Tool

STEP 2 : Reading the images

STEP 3 : Extract the grid corners

STEP 4 : Main Calibration step

STEP 5 : Reprojection

STEP 6 : Show Extrinsic

STEP 7 : Recomp. corners

STEP 8 : Skew Coefficient calculation

STEP 9 : Undistort images

Basically, what will be done is that; first, the input images' grid corners will be extracted that were read by MATLAB at previous steps. Then, intrinsic and extrinsic parameters will be estimated in the 4. Step. Right after that, I will reproject the points on the image by utilizing the estimated projection

matrix and compare the error amount between real corner points and reprojected corners on each image. In between recomputing the corners by the error amount and previous step, we will show two different frames emerged due to the extrinsic parameters of the input images. Then, I will recompute the parameters by initializing the optimization from the previous calibration results, so that will obtain better result, lesser error amount. Finally, I will also calculate the skew coefficient that I wasn't considered in the beginning and finally show the undistorted versions of the inout images. I hope you will have fun during reading the report.

CONCLUSION

What we have done?

As I have mentioned what we were going to do in INTORDICTION part, let me briefly recap what we have done again from different aspect. Basically, we had 7 input images that we were going to utilize in order to obtain first intrinsic parameters of the camera, then extrinsic parameters of each image. For the very first process, I have extracted 49 grid corners from each input image. Thus, 7*49 = 343 points which helped the algorithm to estimate the projection matrix first, then estimate intrinsic and extrinsic parameters from it. Later on, I used them to calibrate my camera. At that step, I had intrinsic and extrinsic parameters. However, when I reprojected the points calculating them by intrinsic & extrinsic parameters, I have realized that error amount is a little bit much. Then, I have recomputed the corners by performing some stuff so that I improved my estimation and lowered the error amount. During those processes, I also showed both world-centered and camera-centered frames using extrinsic parameters of each image, which were so much fun. Finally, I have estimated the skew coefficient which I used to ignore during the parameter calculation due to the fact that it was so small to be considered and show that as well. As the most final thing, I have retrieved the undistorted images for visualization.

What is the motivation of this lab?

The motivation of this lab is to extract the projection matrix by not just one single image of 3D calibration object, but estimating it by utilizing several 2D calibration object images captured with different angles and aspects. Therefore, at then end, the motivation has been accomplished and the difference of estimating the projection matrix by Caltech and 3D Calibration object method will be presented at the end of this entire lab report.