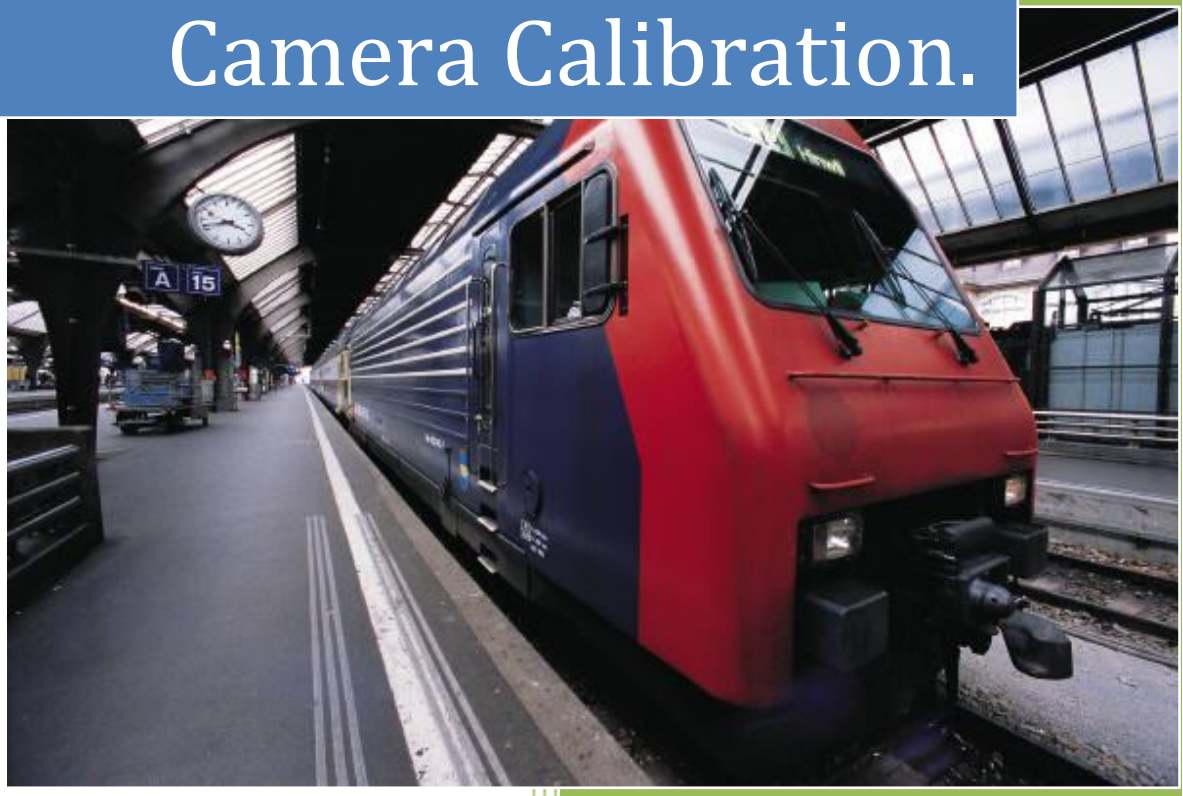
2018

Computer Vision. Practical 2 Camera Calibration.



Surajudeen Abdulrasaq and

Ezukwoke Ifeanyi Kenneth

Machine Learning and Data Mining

6/3/2018

Introduction:

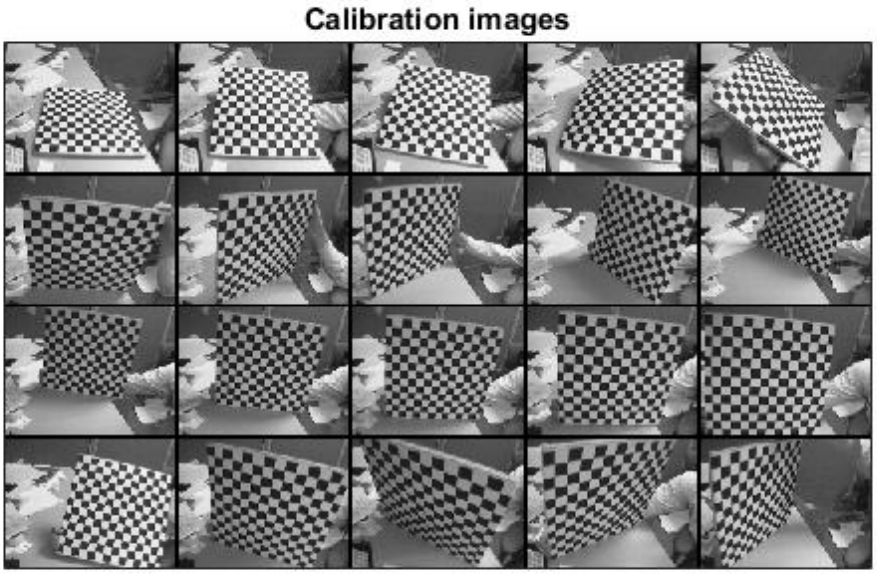
A picture is worth ten thousand words 'Fred R. Barnard, of Printers' Ink, 10 March 1927', and how do we produce picture? Camera has always been part of human for a long time, since the invention pinhole camera we have seen an evolution of advanced camera at affordable prices but this does not come without a shortcoming that's 'Distortion' therefore camera calibration is needed to overcome this phenomenon. Calibration in addition helps us to establish the relationship between the intrinsic and extrinsic parameter that exist in the camera.

Procedure:

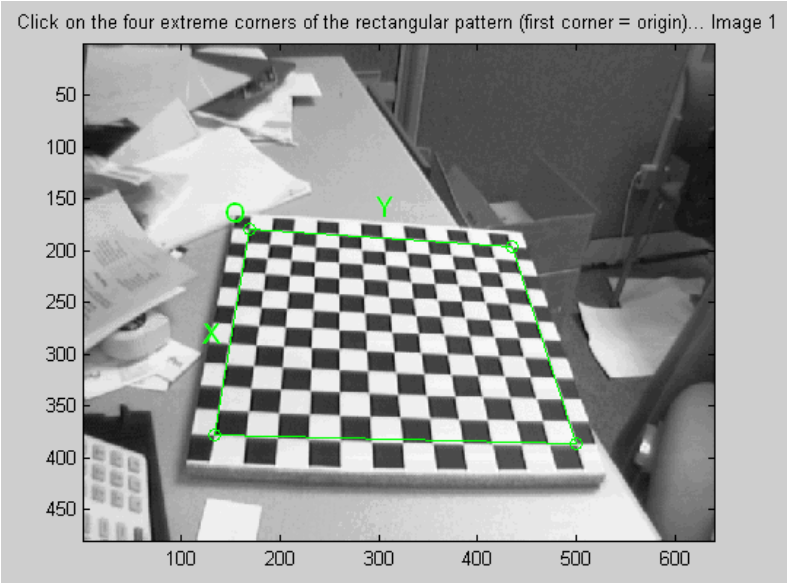
This experiment was carried out using the existing toolbox in Matlab "CalTech camera calibration toolbox" and the entire step taking in this lab are depicted below.

Corner extraction, calibration, with additional tools:

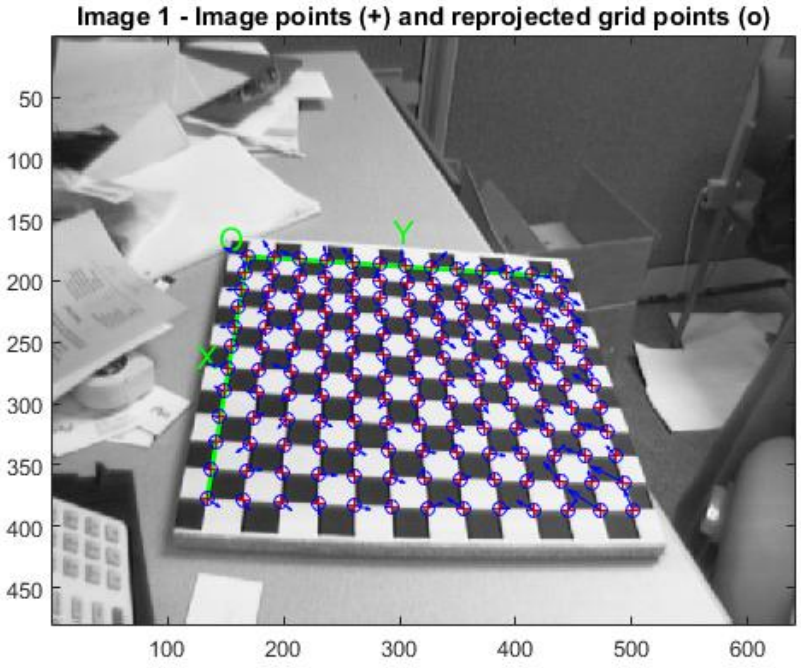
Load the images: we use the 20 images provided for this lab by the instructors (I did this to check if my results will be slightly different from the original results performed by matlab). I loaded the images and run the *mosaic* to visualize the 20 images.



Extract the grid corners: we extracted the grid corners using the default window size of the corner finder: *wintx=winty=5* which effectively transformed the window to size *11x11* pixels.



we choose the default sizes ***dX*** *and* ***dY*** in X and Y of each square in the grid (in this case, ***dX=dY=30mm***=default values), which automatically transform the image corners.



The procedure was repeated for all the 20 images and the results are shown below:

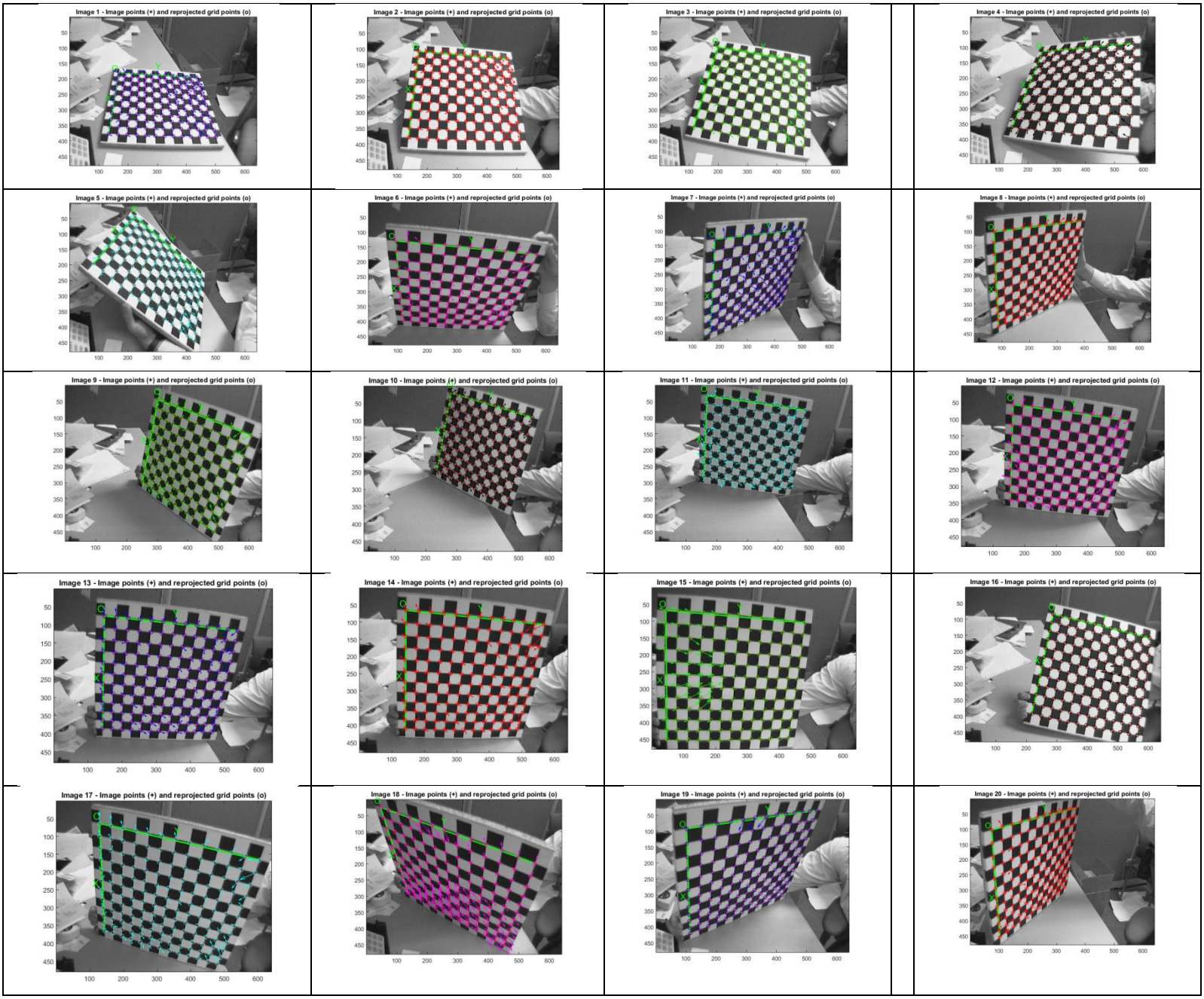
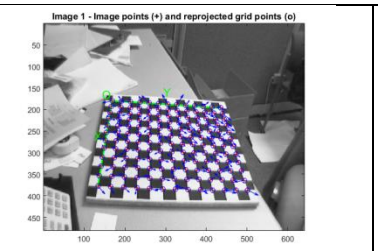
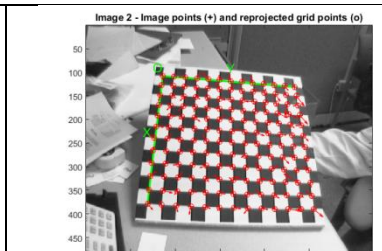
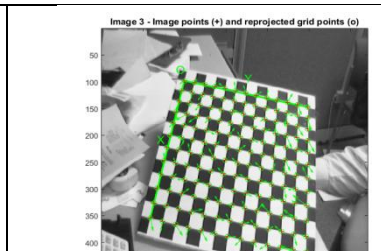


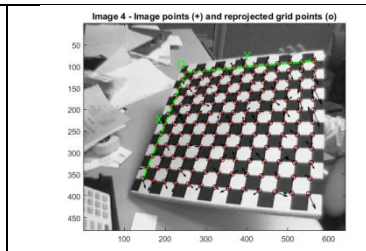
Image distortion: sometimes some distortions do occur, when this happen we choose distortion coefficient kc **= 0.3**

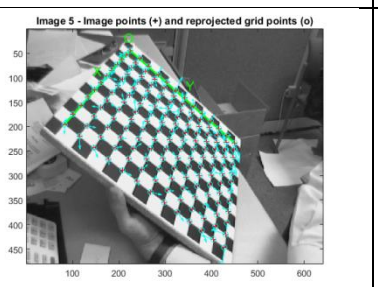
Main Calibration step: After corner extraction, we did the camera calibration and this is done in two steps: first initialization, and then nonlinear optimization, then re-projects them on the images. The results are dissipated below.

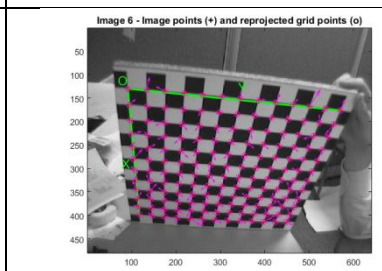


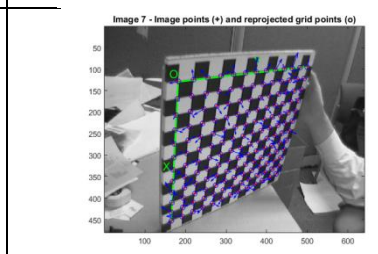


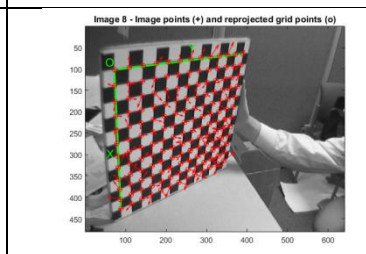


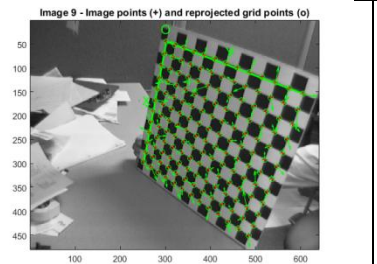


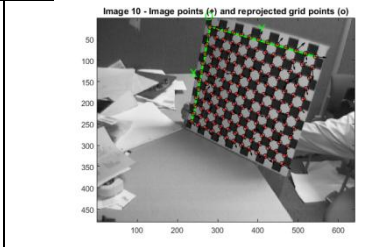


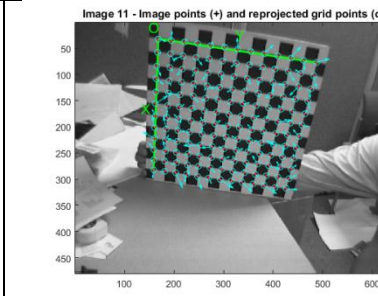


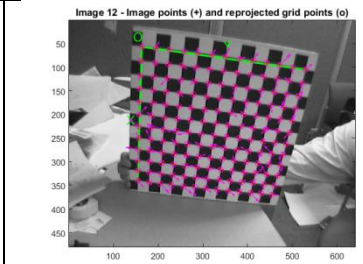


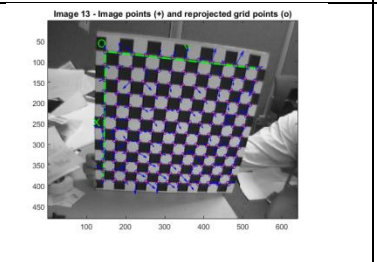


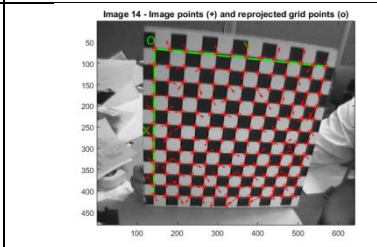


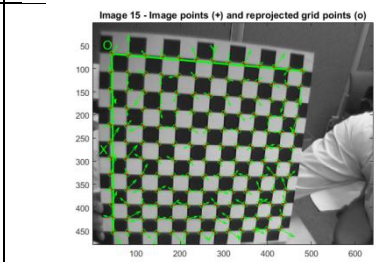


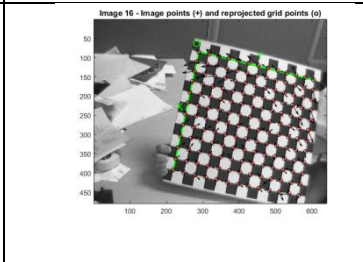


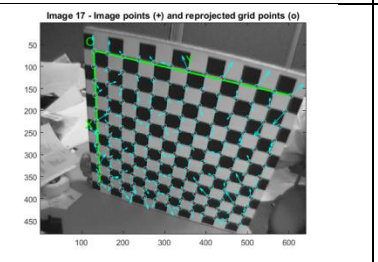


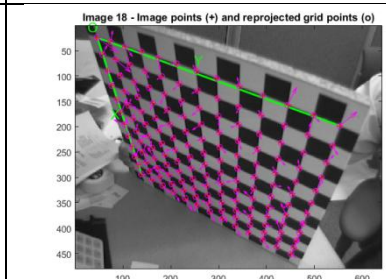


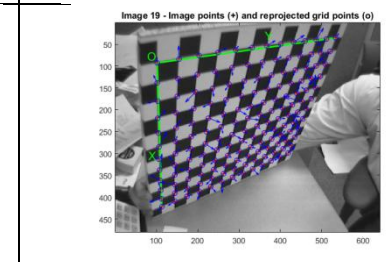


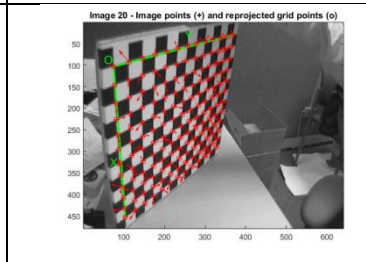




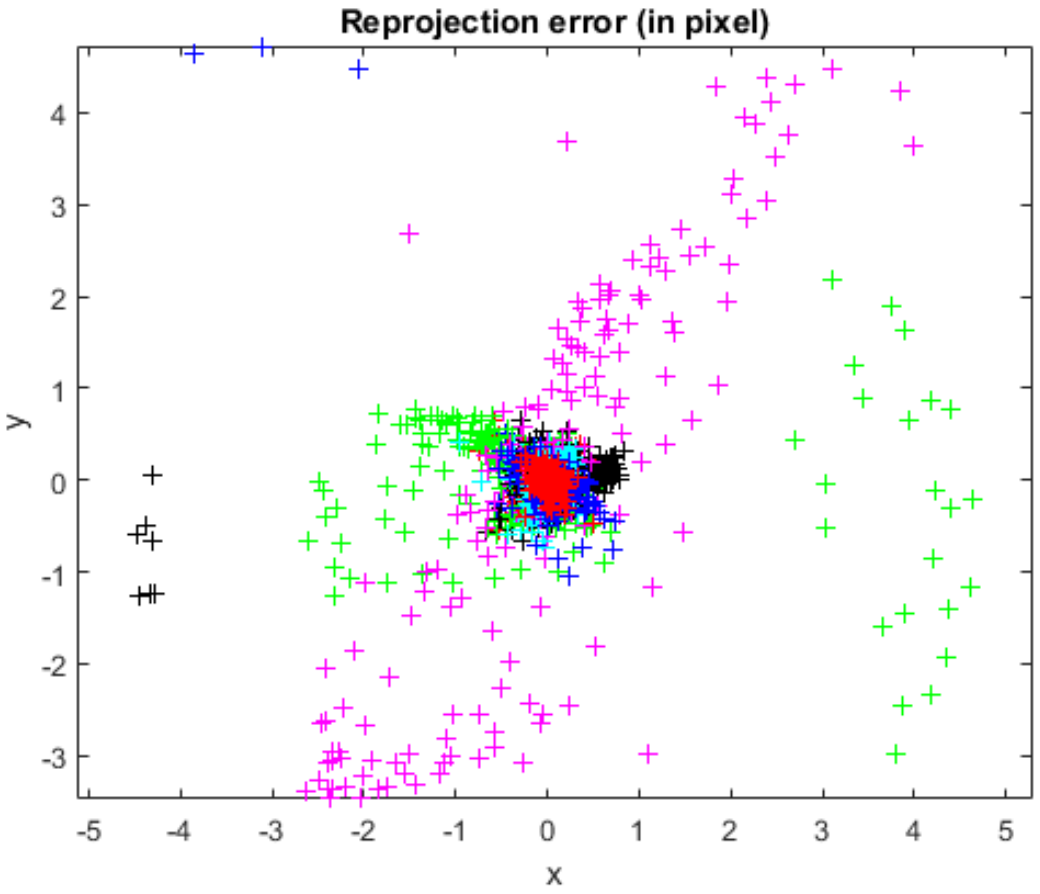




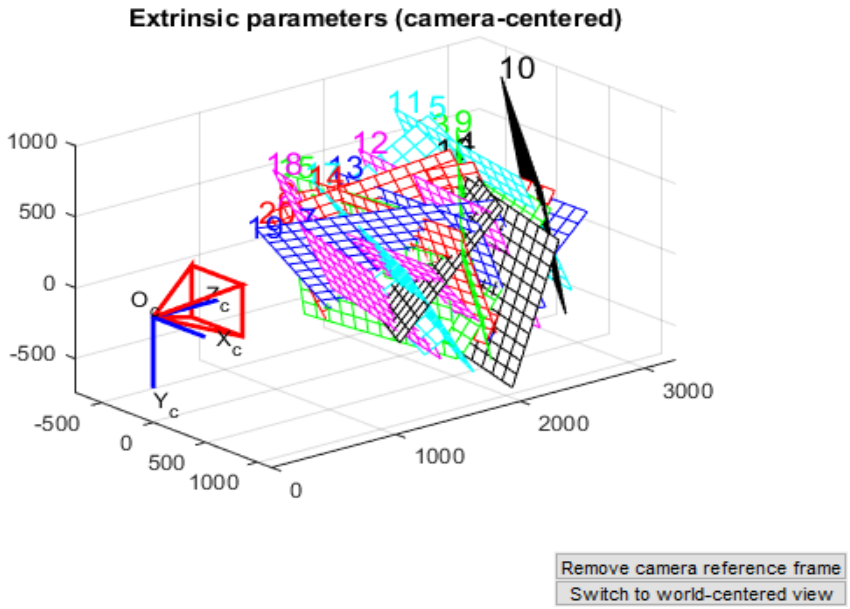


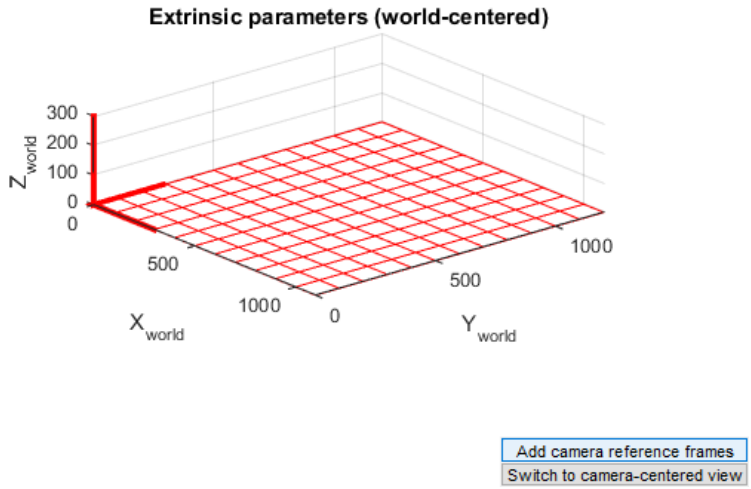


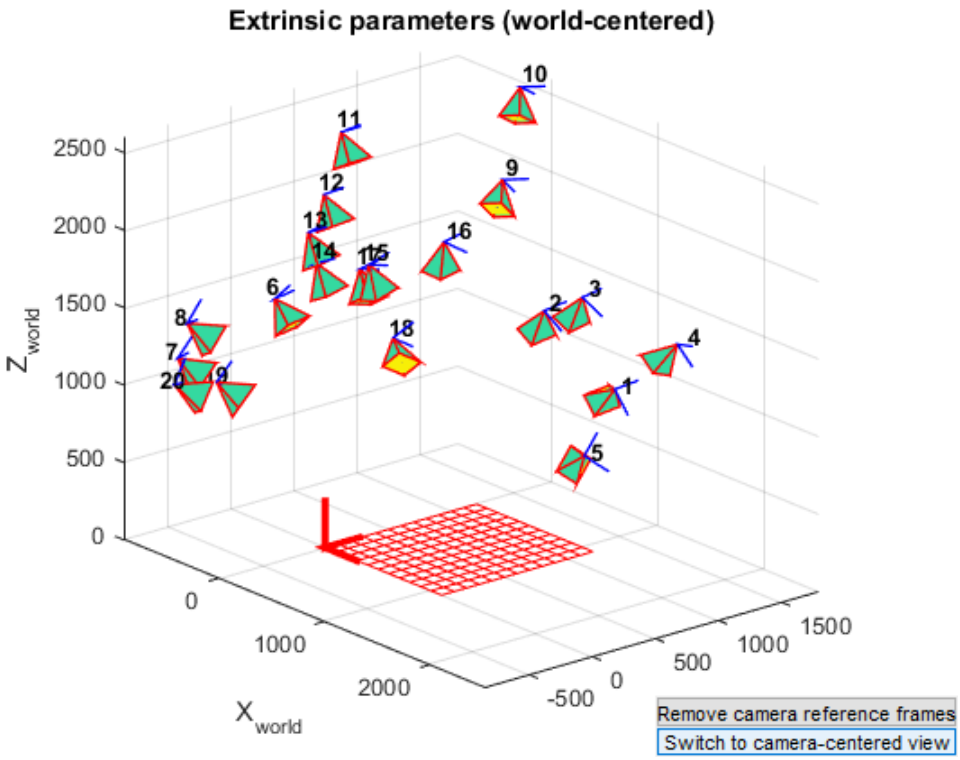
Parameters ***alpha_c*** and the distortion coefficient ***kc*** were not been estimated because we chose the default parameter at the first step above. Therefore, the angle between the x and y pixel axes is 90 degrees. Also we have a pixel error estimated as [*0.48330 to 0.32917*], and re-projection error is also shown below with crosses are color coded.



Next we display ***extrinsic parameter*** in the ***Camera calibration tool***. Using a 3D plot: the first image below show the camera center, second image shows the world view in one projection and the last also shows the world view in another projection.

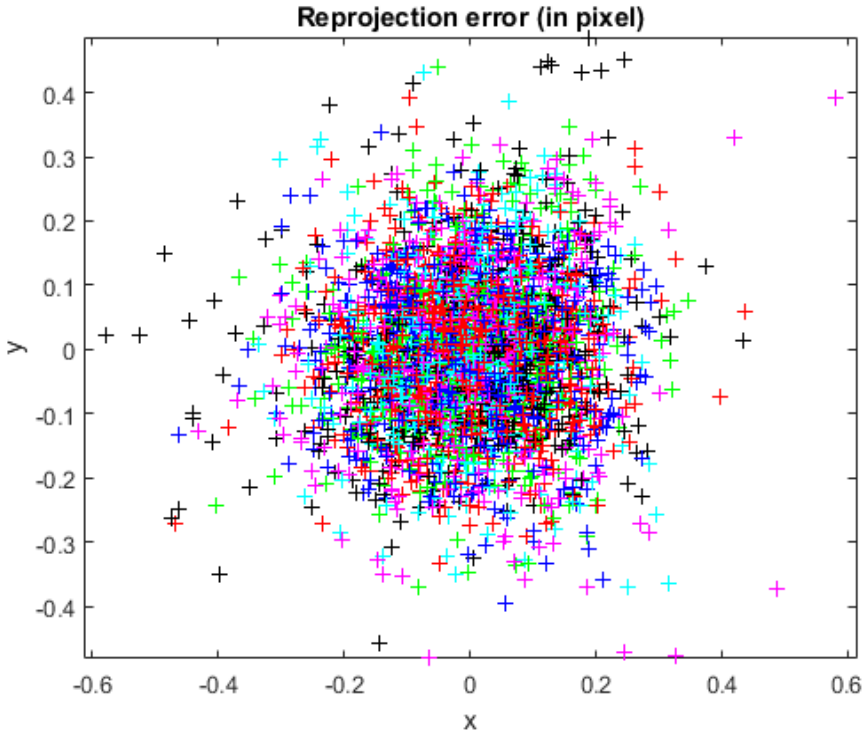






camera position and orientation is represented by a green pyramid in the world view

We can indentified points with large amount errors by using the **Analyze error, the results is display below this shows us the selected image** the grid coordinate in the calibration grid (at the origin of the pattern)

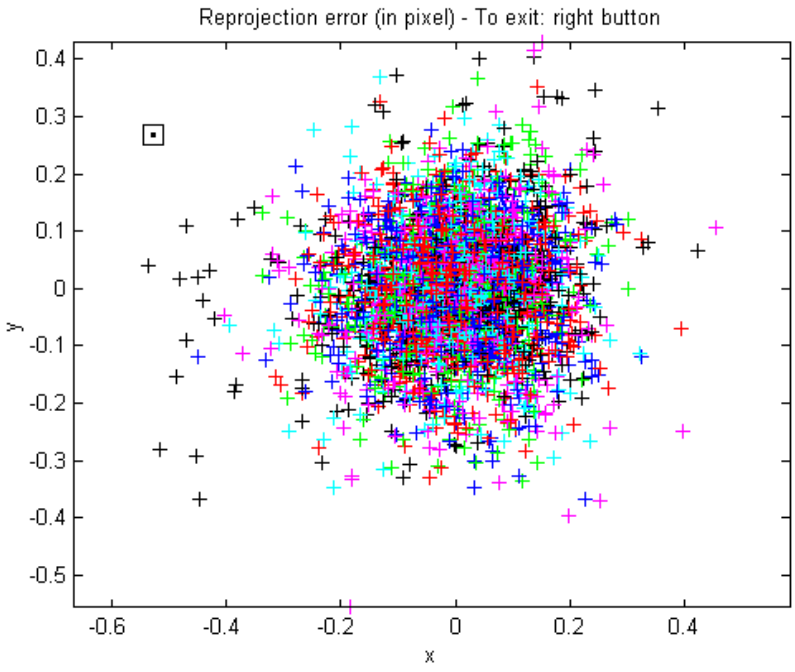


Error Analysis at different point.

Selected image: 16 *Selected point index: 73* *Pattern coordinates (in units of (dX,dY)): (X,Y)=(0,6)* *Image coordinates (in pixel): (450.78,119.45)* *Pixel error = (-0.39837,-0.35038)* *Window size: (wintx,winty) = (5,5)*	***Selected image: 6*** *Selected point index: 25* *Pattern coordinates (in units of (dX,dY)): (X,Y)=(0,10)* *Image coordinates (in pixel): (499.46,166.65)* *Pixel error = (0.32532,-0.47587)* *Window size: (wintx,winty) = (5,5)*	***Selected image: 19*** *Selected point index: 5* *Pattern coordinates (in units of (dX,dY)): (X,Y)=(4,12)* *Image coordinates (in pixel): (483.70,148.91)* *Pixel error = (-0.00316,-0.31882)* *Window size: (wintx,winty) = (5,5)*
Selected image: 2 *Selected point index: 22* *Pattern coordinates (in units of (dX,dY)): (X,Y)=(9,11)* *Image coordinates (in pixel): (473.27,332.43)* *Pixel error = (-0.46811,-0.27055)* *Window size: (wintx,winty) = (5,5)*	***Selected image: 5*** *Selected point index: 119* *Pattern coordinates (in units of (dX,dY)): (X,Y)=(10,3)* *Image coordinates (in pixel): (190.18,242.96)* *Pixel error = (0.16103,-0.02894)* *Window size: (wintx,winty) = (5,5)*	***Selected image: 20*** *Selected point index: 115* *Pattern coordinates (in units of (dX,dY)): (X,Y)=(6,3)* *Image coordinates (in pixel): (172.26,279.62)* *Pixel error = (0.39590,-0.07410)* *Window size: (wintx,winty) = (5,5)*

Exploring the error inspection tool further by re-computing the corners of the specific images using a different window size (larger or smaller). This is useful in cases where the corners have been badly extracted on one or several images. I re-extracted the corner of images *[1 2 3 4 5 6 10 18]* using windows size of 9, *[7 8 9]* using size 8 and *[11 12 13 14 15 16 17]* using windows 11

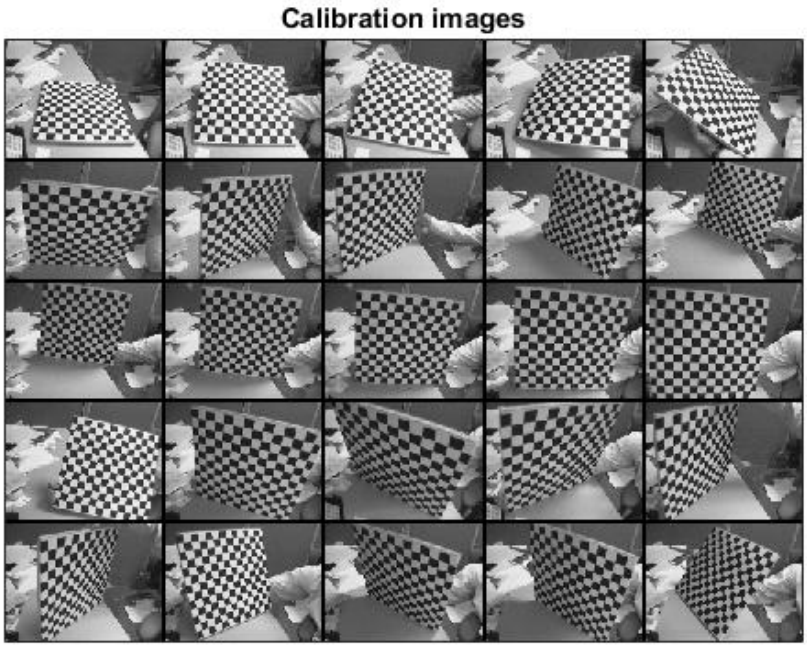
Re-extraction of the grid corners on the images (after first calibration) *Window size for corner finder (wintx and winty):* *wintx ([] = 5) = 9* *winty ([] = 5) = 9* *Window size = 19x19* *Number(s) of image(s) to process ([] = all images) = [1:5 6 10 18]* *Use the projection of 3D grid or manual click ([]=auto, other=manual):* *Processing image 1...2...3...4...5...6...10...18...* *Done*	***Re-extraction of the grid corners on the images (after first calibration)*** *Window size for corner finder (wintx and winty):* *wintx ([] = 5) = 8* *winty ([] = 5) = 8* *Window size = 17x17* *Number(s) of image(s) to process ([] = all images) = [7 8 9]* *Use the projection of 3D grid or manual click ([]=auto, other=manual):* *Processing image 7...8...9...* *Done*
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) = [11 12 13 14 15 16 17]* *Use the projection of 3D grid or manual click ([]=auto, other=manual):* *Processing image 11...12...13...14...15...16...17...* *Done*	***Re-extraction of the grid corners on the images (after first calibration)*** *Window size for corner finder (wintx and winty):* *wintx ([] = 5) = 6* *winty ([] = 5) = 6* *Window size = 13x13* *Number(s) of image(s) to process ([] = all images) = [19 20]* *Use the projection of 3D grid or manual click ([]=auto, other=manual):* *Processing image 19...20...* *Done*



Re projection error after changing the windows size: This result in a reduced error size, the results is save as Calib_Results_old0.mat.

Selected image: 16 *Selected point index: 128* *Pattern coordinates (in units of (dX,dY)): (X,Y)=(7,2)* *Image coordinates (in pixel): (159.35,241.18)* *Pixel error = (-1.00491,-1.26406)* *Window size: (wintx,winty) = (9,9)*	***Selected image: 6*** *Selected point index: 3* *Pattern coordinates (in units of (dX,dY)): (X,Y)=(2,12)* *Image coordinates (in pixel): (507.02,94.66)* *Pixel error = (-1.40624,-3.92616)* *Window size: (wintx,winty) = (6,6)*
Selected image: 2 *Selected point index: 12* *Pattern coordinates (in units of (dX,dY)): (X,Y)=(11,12)* *Image coordinates (in pixel): (484.30,450.56)* *Pixel error = (-1.47341,-0.25131)* *Window size: (wintx,winty) = (8,8)*	***Selected image: 19*** *Selected point index: 153* *Pattern coordinates (in units of (dX,dY)): (X,Y)=(8,0)* *Image coordinates (in pixel): (107.22,337.28)* *Pixel error = (-2.87554,-4.97096)* *Window size: (wintx,winty) = (6,6)*

Adding additional 25 images: Re-calibrate the camera using the complete set of 25 images without re-computing everything from scratch, the ***mosaic*** is shown below.



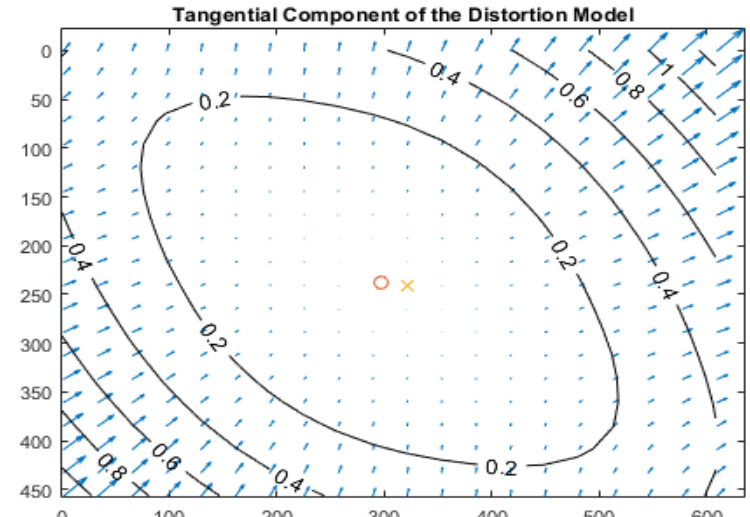
Extract grid corners to extract the corners on the five new images, with default window sizes **wintx=winty=5**: with corner extraction on the five images and optimize by running the **Calibration** again.

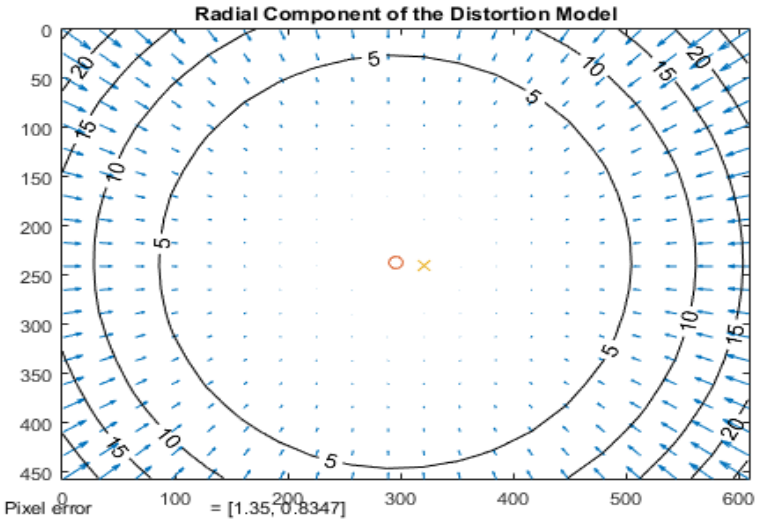
Next we recomputed the image corners of the four last images using different window sizes. Use ***wintx=winty=9*** for images 22 and 24, use ***wintx=winty=8*** for image 23, and use ***wintx=winty=6*** for image 25. Follow the same procedure as previously presented (three calls of **Recomp. corners** should be enough). After re-computation, Calibration was repeated once again and **Save** in **Calib_Results.mat.**

Add/Suppress images. : The setup is now back to what it was before suppressing 5 images 16, 18, 19, 24 and 25. Now running 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.

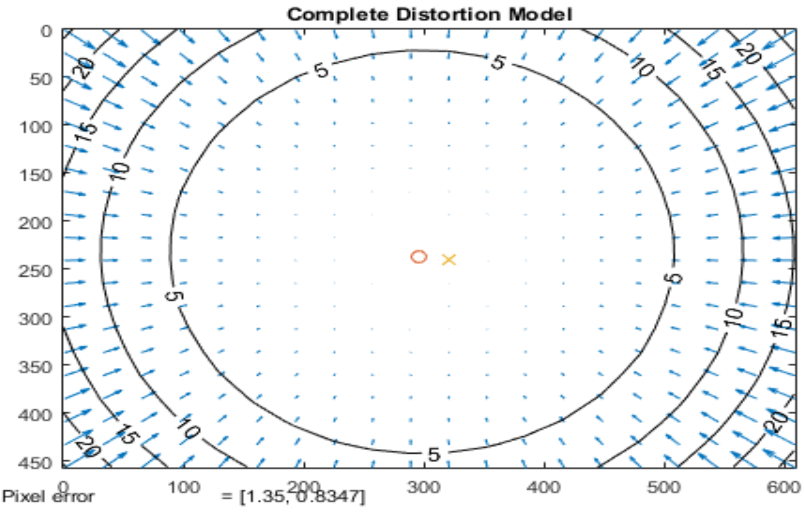
Visualize distortions: it's helpful to visualize the effect of distortions on the pixel image, and the importance of the radial component versus the tangential component of distortion. *The three following images are then produced:*

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. Points at the corners of the image are displaced by as much as 25 pixels. The second figure shows the impact of the tangential component of distortion. On this plot, the maximum induced displacement is 0.14 pixels (at the upper left corner of the image). 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.,





Pixel error	= [1.35, 0.8347]	
Focal Length	= (654.96, 654.646)	+/- [2.754, 2.789]
Principal Point	= (295.18, 236.56)	+/- [5.488, 5.287]
Skew	= 0	+/- 0
Radial coefficients	= (-0.2412, 0.1034, 0)	+/- [0.02177, 0.08706, 0]
Tangential coefficients	= (-0.001475, 0.001134)	+/- [0.001209, 0.001224]



Pixel error	= [1.35, 0.8347]	
Focal Length	= (654.96, 654.646)	+/- [2.754, 2.789]
Principal Point	= (295.18, 236.56)	+/- [5.488, 5.287]
Skew	= 0	+/- 0
Radial coefficients	= (-0.2412, 0.1034, 0)	+/- [0.02177, 0.08706, 0]
Tangential coefficients	= (-0.001475, 0.001134)	+/- [0.001209, 0.001224]

Computation of extrinsic parameters:

Compute the extrinsic parameters attached to this image given the intrinsic camera parameters previously computed. Initialization of the intrinsic parameters - Number of images: 25.

Calibration parameters after initialization:

Focal Length: *fc = [660.11430 660.96109] +/- [1.23329 1.25361]*

Principal point: *cc = [295.96027 238.24920] +/- [2.47164 2.36911]*

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

Distortion: *kc = [-0.25184 0.13934 -0.00163 -0.00021 0.00000] +/- [0.00979 0.03989 0.00053 0.00054 0.00000]*

Pixel error: *err = [0.46716 0.52306]*

Calibration results after optimization (with uncertainties):

Focal Length: *fc = [654.96014 654.64649] +/- [2.75397 2.78884]*

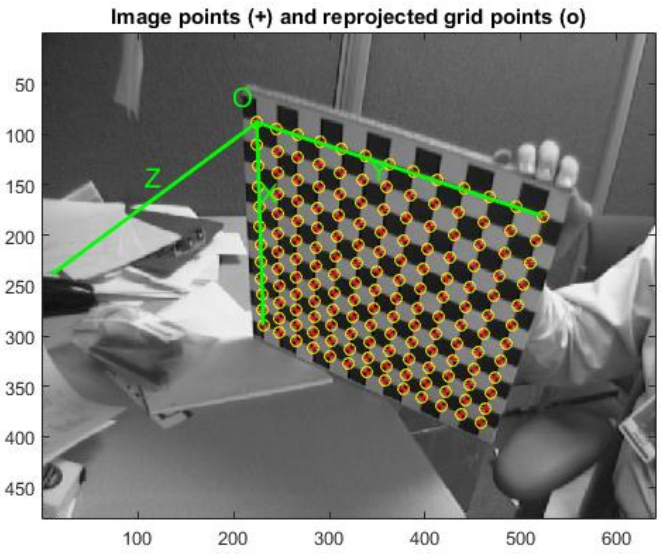
Principal point: *cc = [295.17959 236.55964] +/- [5.48755 5.28675]*

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

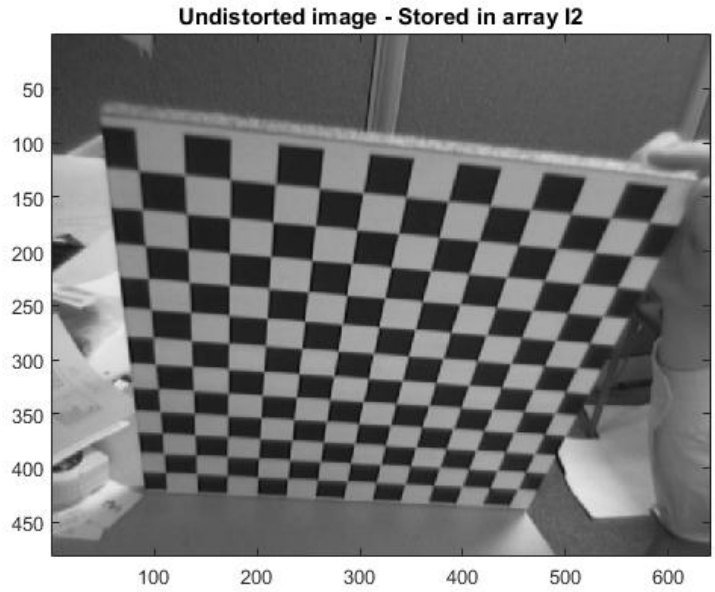
Distortion: *kc = [-0.24118 0.10335 -0.00148 0.00113 0.00000] +/- [0.02177 0.08706 0.00121 0.00122 0.00000]*

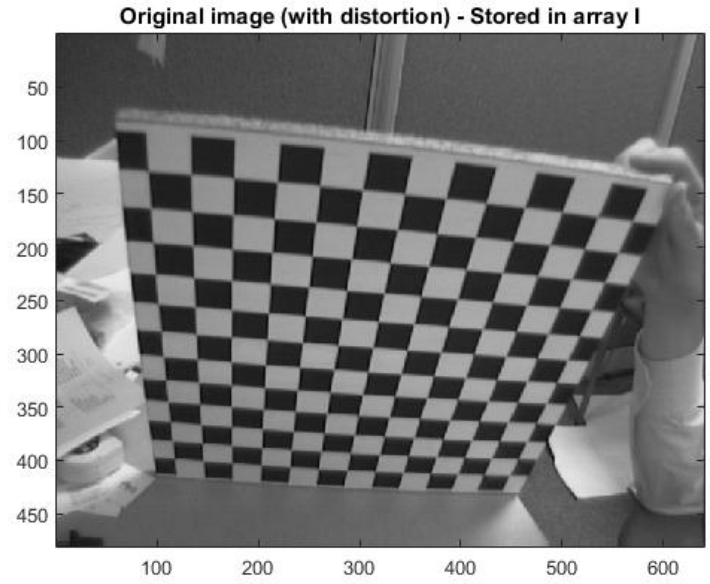
Pixel error: *err = [1.35026 0.83466]*

The extrinsic parameters are encoded in the form of a rotation matrix (**Rc_ext**) and a translation vector (**Tc_ext**). The rotation vector **omc_ext** is related to the rotation matrix (**Rc_ext**) through the Rodrigues formula: **Rc_ext = rodrigues(omc_ext)**.



Undistorted images: This function helps generate the undistorted version of one or multiple images given pre-computed intrinsic camera parameters. ***Here am using image 6 as an example.***





Export calibration data to other formats (Willson-Heikkil and Zhang):

The calibrated data (extracted image corners + associated 3D world coordinates) was exported to Willson-Heikkil and Zhang formats. Useful for comparison purposes

Part two

Calibrating a stereo system, stereo image rectification and 3D stereo triangulation.

Load left and right calibration results: these shows initial values for the intrinsic camera parameters are shown in addition to an estimate for the extrinsic parameters **om** and **T** characterizing the relative location of the right camera with respect to the left camera.

Run stereo calibration: In the stereo toolbox all intrinsic and extrinsic parameters have been recomputed, together with all the uncertainties so as to minimize the re-projection errors on both camera for all calibration grid locations. the uncertainties on the intrinsic parameters for both cameras are smaller after stereo calibration. This is due to the fact that the global stereo optimization is 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.

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]*

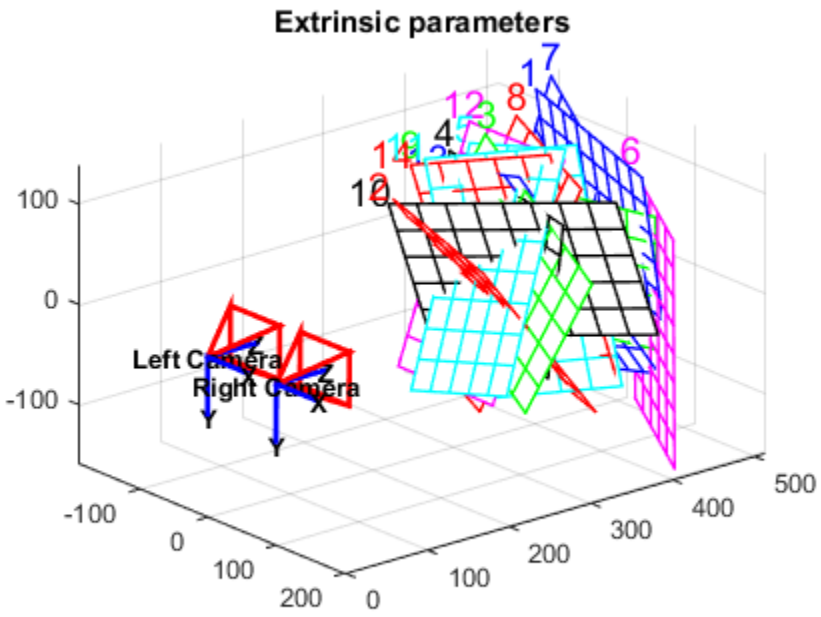
Extrinsic parameters (position of right camera wrt left camera):

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).

Show Extrinsic of the stereo rig: spatial configuration of the two cameras and the calibration planes may be displayed in a form of a 3D plot



Rectify the calibration images: All 14 pairs of images are then stereo rectified (with epipolar lines matching with the horizontal scanned lines) under **left_rectified01.bmp**, **right_rectified01.bmp**,...,**left_rectified14.bmp**,**right_rectified14.bmp**. In addition to generating the rectified images, the script also saves the new set of calibration parameters under **Calib_Results_stereo_rectified.mat** (valid only for the rectified images).