Homework 3

NAME: Aishwarya Dekhane

Please copy all required images and place into a Word document and upload to Canvas, along with answers to the questions. Please save the Word file as a PDF before submitting.

This assignment illustrates the process of camera calibration. This is an essential part of any modern vision system and also shows how iterative solution methods are used to converge to a "good" solution when the starting point may not be accurate.

1. Download and install the cameral calibration toolbox , written by Jean-Yves Bouguet, from Canvas in the Homework 3 module. The original source can be found here: (find the download in the Getting Started section):

http://www.vision.caltech.edu/bouguetj/calib_doc/

2. Copy the toolbox folder and extract all contents. The default toolbox name is:

TOOLBOX_calib

In MATLAB go to the Home tab and select Set Path. Add the path of the extracted toolbox (on a Mac it is under the /Applications folder, but the exact location is not important)

3. Also copy the folder "calib_example" consisting of five .tif images of a checkerboard pattern taken at different locations, which means the pattern was photographed at different angles to the camera and different distances. This method of camera calibration simplifies the procedure as the precise location of the pattern is not needed, only identifying several points in the image (that correspond to known points in the image. The points are known because the pattern squares have a known size).

Note that the use of square patterns of known dimensions makes location of "corner points" easier, and the user need only identify the outer points of the checkerboard pattern. This assumes the lens does not distort the image substantially. If that is not true, for example the biggest distortion normally found is a "radial distortion" when the image will curve noticeably, particularly at the edge of the pattern. If this is noticed, when the software identifies points that do not match the corner points, it may be necessary to specify an initial value for radial distortion. While this often is represented by a third order polynomial, for the initial value of radial distortion only the first order term, called "kc" is needed. But, often if this estimate of radial

distortion is not input it can be estimated from the data and the correct value obtained, though at the cost of more iterations of this approach.

Change, from within MATLAB, to the calib_example folder. You may use unix-like or DOS like commands, such as cd to change directories, or ls or dir to display the contents of a directory.

4. From the MATLAB command prompt, type "calib_gui". Once the toolbox path is added to MATLAB's path environment variable the command will run.

A small dialog box should appear with title "Camera Calibration Toolbox". Select the option of Standard. This will read all images into memory. The other option is for memory limited systems. For the small number of files used in this assignment the Standard option should be fine.

5. Click on the Read Image button and then type in the base name of the image files used, in this case "Image", then select the type, since these are tif files enter "t"

A set of thumbnail images should be shown, since we're using five images you should only see the five displayed.

Now click on the Extract grid corners button

Save each of the five images with the corners indicated. For the final image alone, an initial value of kc will be beneficial. Save each image that is displayed for the value you select. These images will be part of the document turned in for the assignment.

The toolbox will ask if all desired images are to be used, select the default answer, which means not entry is required.

Select the defaults for wintx and winty, two separate entries. This will show the given Window size, 11x11. This means the corner will be search in a window around the indicated (point clicked on) plus dX and dY, so, in this case, the clicked point plus and minus five points (11 points) in each of the X and Y directions. Also select the default for "automatic square counting". If the initial set of corner points selected is not accurate enough the software will ask for the number of squares in the X and Y direction. If needed, use 13 squares in the X direction and 14 squares in the Y direction.

Now the first image is presented, select the outer four corners, starting with the top left then proceeding clockwise.

NOTE: If using MATLAB online this process may be complicated by a change to the software that makes viewing the crosshairs of the selection tool difficult. What should be shown is two lines where the center point can be placed and selected with

a mouse click. Recent changes to MATLAB resulted in the crosshairs being shown within a dark rectangle, making proper point selection difficult. For the purpose of this assignment it is only a complication and proceed using the best estimate you can select.

Now enter the square sizes uses, dX and dY. Both are 30 (for 30 mm), which is different than the default. The corners for each square will be displayed. They should be close to the actual corners.

Repeat for the remaining images. On the last image you should notice more radial distortion, where it is possible to use an initial guess for kc, the first term in the radial distortion model polynomial. In that image, you can try several different values of an initial guess, starting with no guess then both a small positive and small negative value for the value of kc. Note, if an initial guess is desired the software will need an input of a numerical values, such as 1, than an initial guess, that is normally between -1 and 1. These are usually fairly small, around .1 or -1, but may be larger for wide-angle lenses or, especially, the so-called "fish-eye" lens.

The software sometimes shows an image other than the one where corner points are needed. If so, click on the previous Figure tab to make the needed image appear on screen. You can tell if this happens if the image zooms in when you click on a corner. If that happens, select the previous figure and try again.

6. After you are satisfied with the initial corner extraction, click on the Calibration button on the toolbar. Record the initial error estimates (a screen capture of the MATLAB window will be sufficient)

7. Click on the Reproject on images button and save the display of the error plot. Very likely, some errors will be small while there will be at least a few that are fairly large, 0.5 pixels or larger.

8. Now click on the Recomp.corners button. This will repeat the corner estimation routine with the current estimates as an initial guess. This will redo the numerical corner estimation. Select the option to use all images.

Now redo the calibration step by clicking the Calibration button again. Record these values (another screen capture will be fine)

Click the Save button to save the calibration results.

9. Again, click on Reproject on images button. Save, or copy the plot showing the error distribution.

10. Click on Analyze error and select one of the points near the edge of the plot. Notice how the corner point that caused the error will be displayed in one of the Figure windows.

11. Now click on the Show Extrinsic button. This will show the position of the camera for each image. There are two views available, a camera centered view and a world centered view. Save a copy of each view to include in your report.

12. Now, at the MATLAB command prompt, run the command: "visualize_distortions" and save the plots for your report. There should be three plots produced, a complete model, a tangential model, and a radial model. Notice that the impact of the tangential model is small, and for these images the complete model is very nearly the same as the radial model, which means the tangential error could be ignored for this camera. Tangential error results from the lens and the image plane not being parallel. This is not common for modern cameras, while radial distortion still exists and can be significant for inexpensive lenses, wide angle lenses, or high-magnification lenses.

Note: this is fine for this assignment as it is a means to demonstrate how camera calibration can be done. Note that only a small set of images, at different perspectives, is required, along with the known size of each square. This is in contrast with earlier methods that required precise knowledge of the movement of each image, that is the distance and rotation angle previously needed to be known in precise detail. Instead of that required knowledge known detail about the image, along with iterative numerical methods, are used to arrive at an accurate camera model.

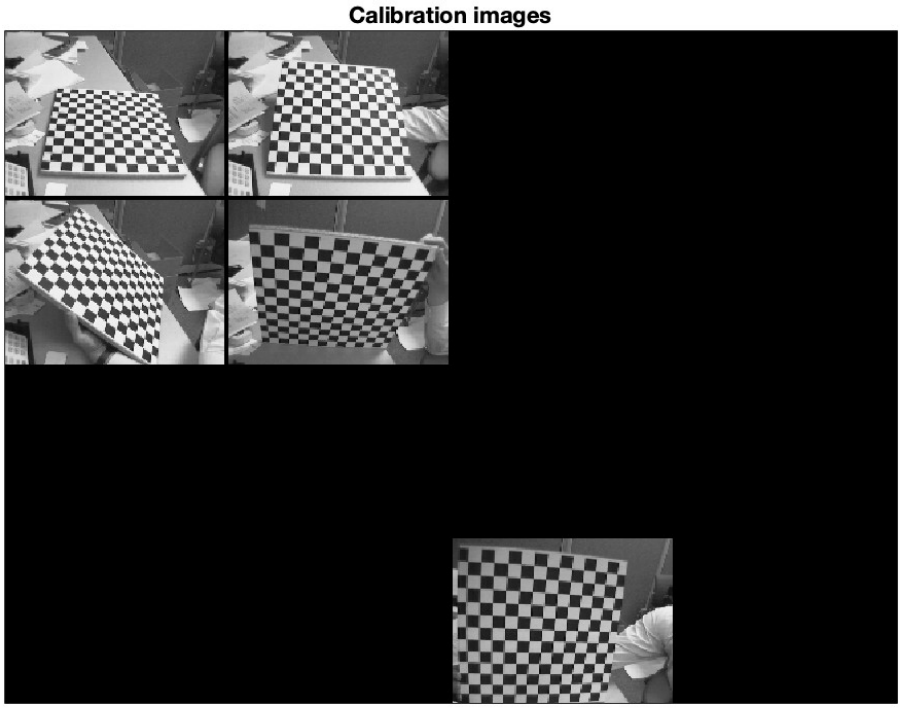
For this assignment, this is sufficient. There is no reason to stop at only one reprojection of the images. That is the purpose of the Analyze error button. If desired, the sources of error could be investigated and those corner points that contribute most to the error, could be re-estimated. While this involves a large amount of clicking on the image and view of errors, this can be done with sufficient time and using only common computer hardware and not specialized equipment. This process could be repeated until observed error has been reduced to an acceptable level.

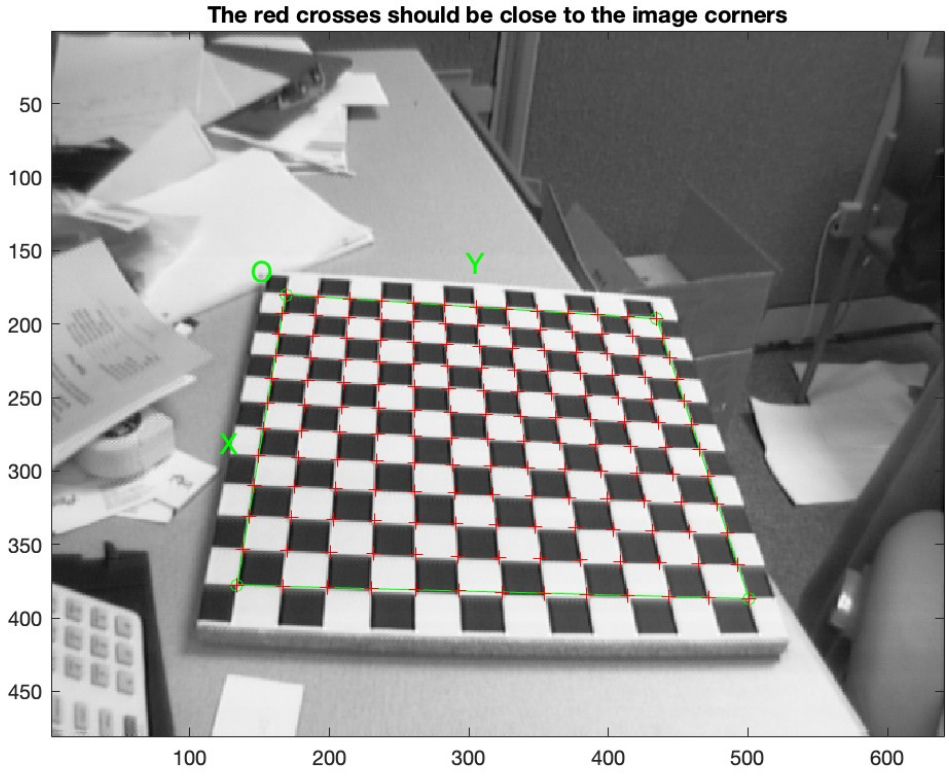
Variations on this method exist, seeking to optimize the error in different ways. But the basic idea is similar: use a closed form solution starting with a linear regression (closed form) estimate, then refine that estimate using numerical methods (such as gradient descent). The error that results can be determined by projecting the estimated corner points back on the original images, as those are available. This can be repeated, using the estimate as an initial guess for the next iteration.

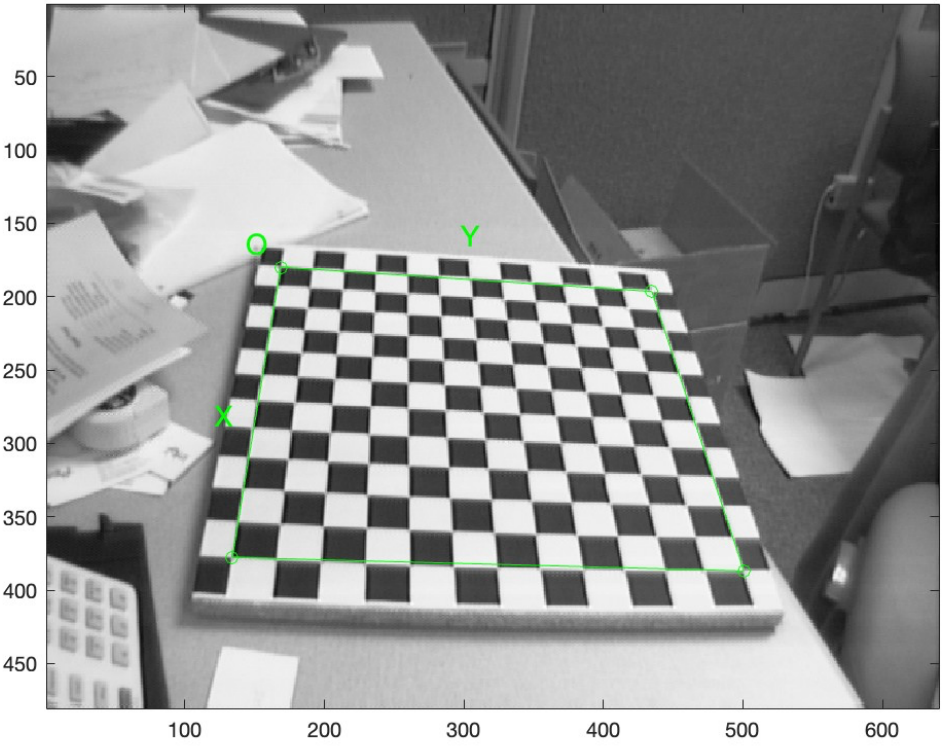
Not part of this assignment: for some images the automatic corner detection may fail, for lenses that have extreme radial distortion in particular. For those cases a manual corner selection option exists in the commands (script files) available, not as

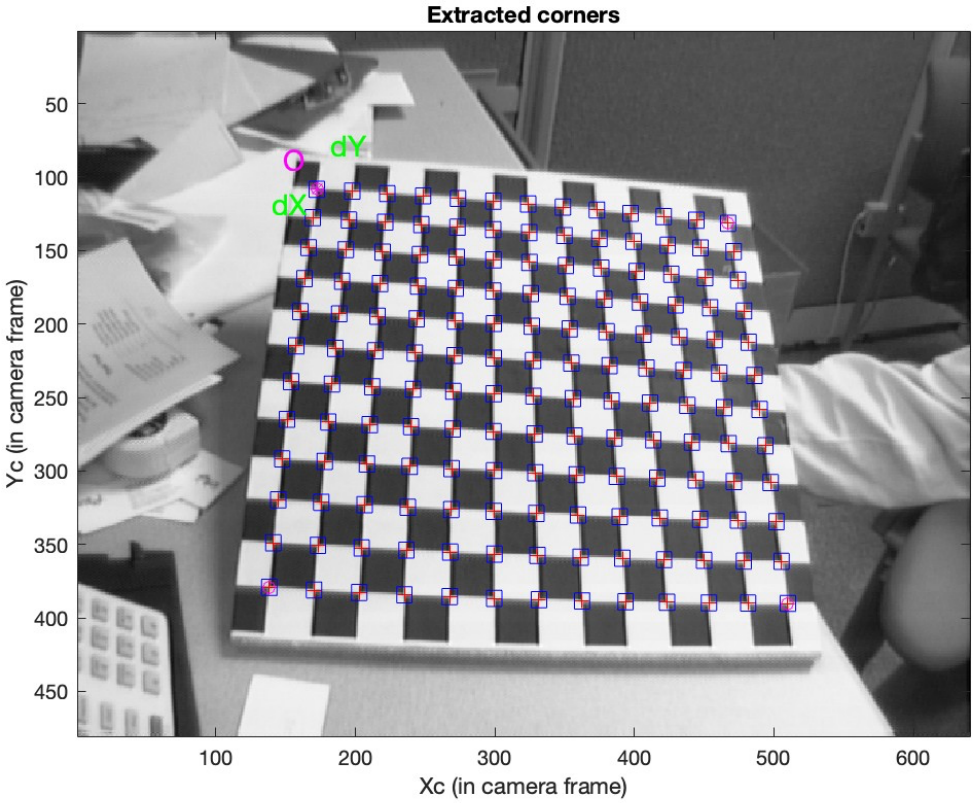
part of the GUI toolbox. This would involve much more selection of each corner in the image, and fortunately is not needed here.

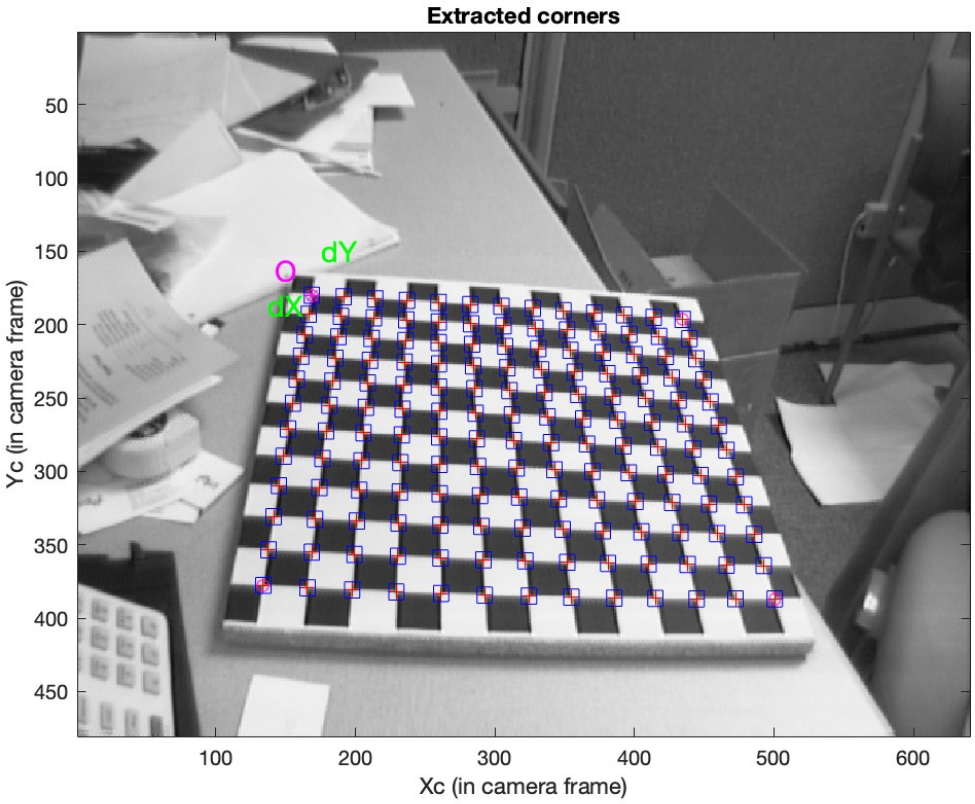
Create a document with the images and plots generated, and turn those in along with answers to the following questions.

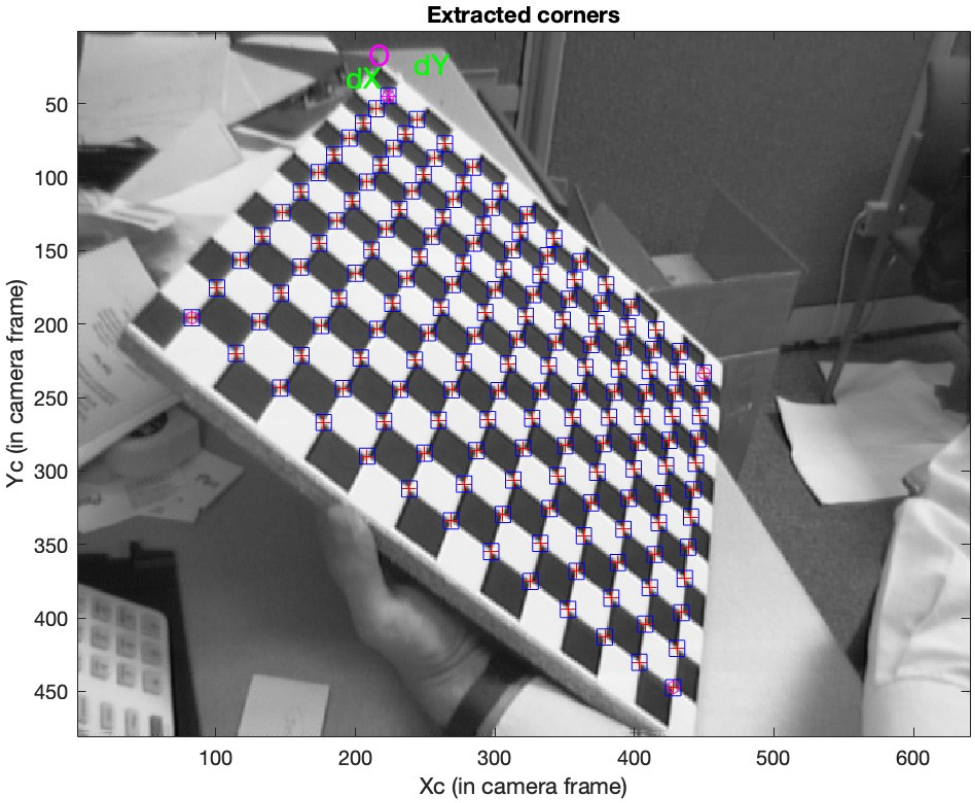

Calibration images

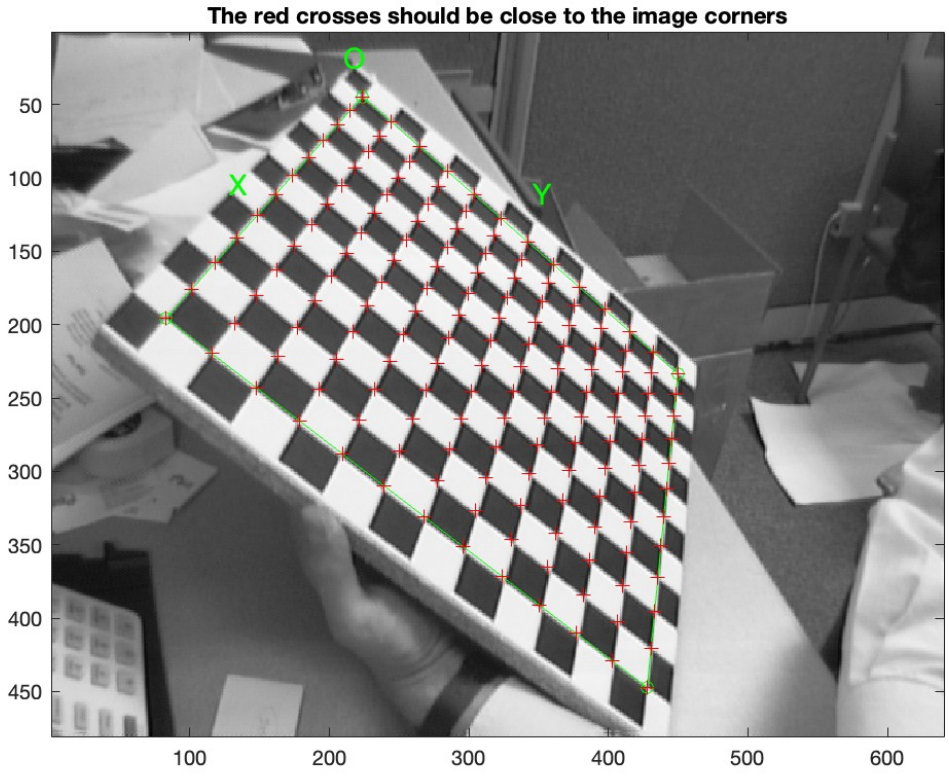

The red crosses should be close to the image corners
O
Y
X
50
100
150
200
250
300
350
400
450
100
200
300
400
500
600

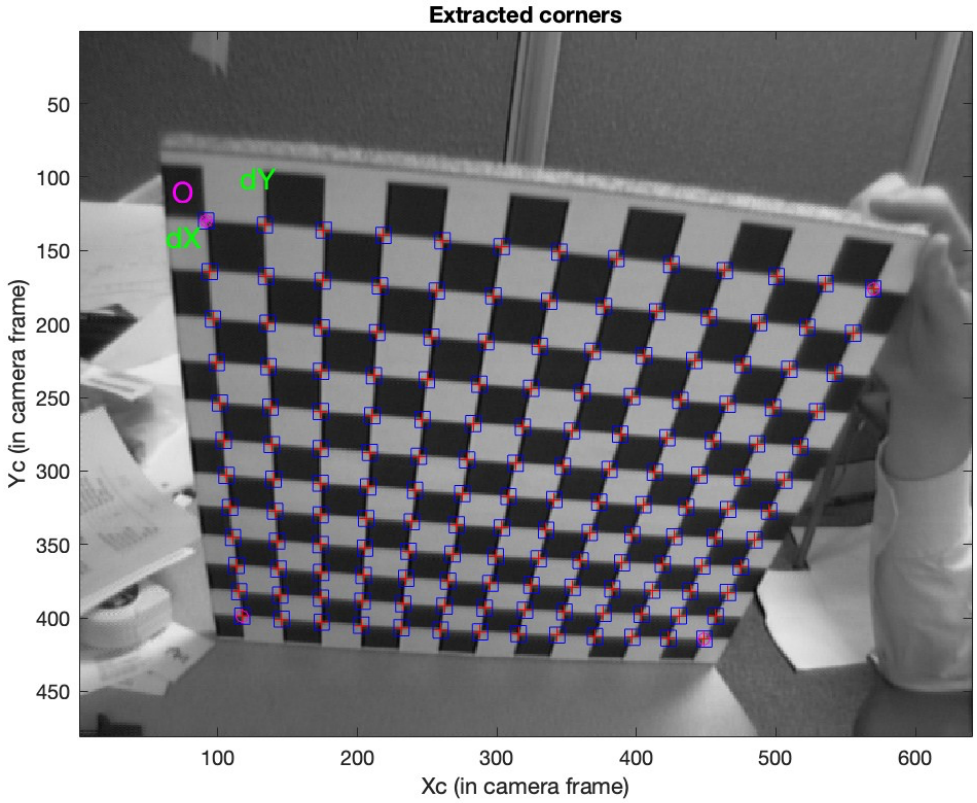

O
Y
X
50
100
150
200
250
300
350
400
450
100
200
300
400
500
600

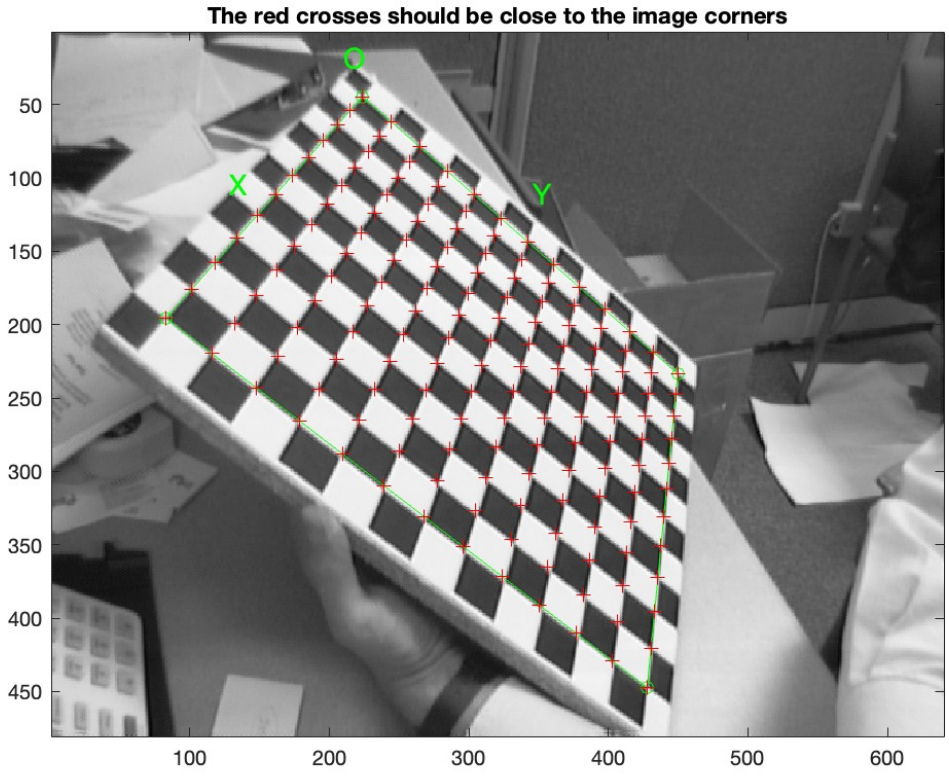

Extracted corners
O
dY
dX
Yc (in camera frame)
Xc (in camera frame)
50
100
150
200
250
300
350
400
450
100
200
300
400
500
600

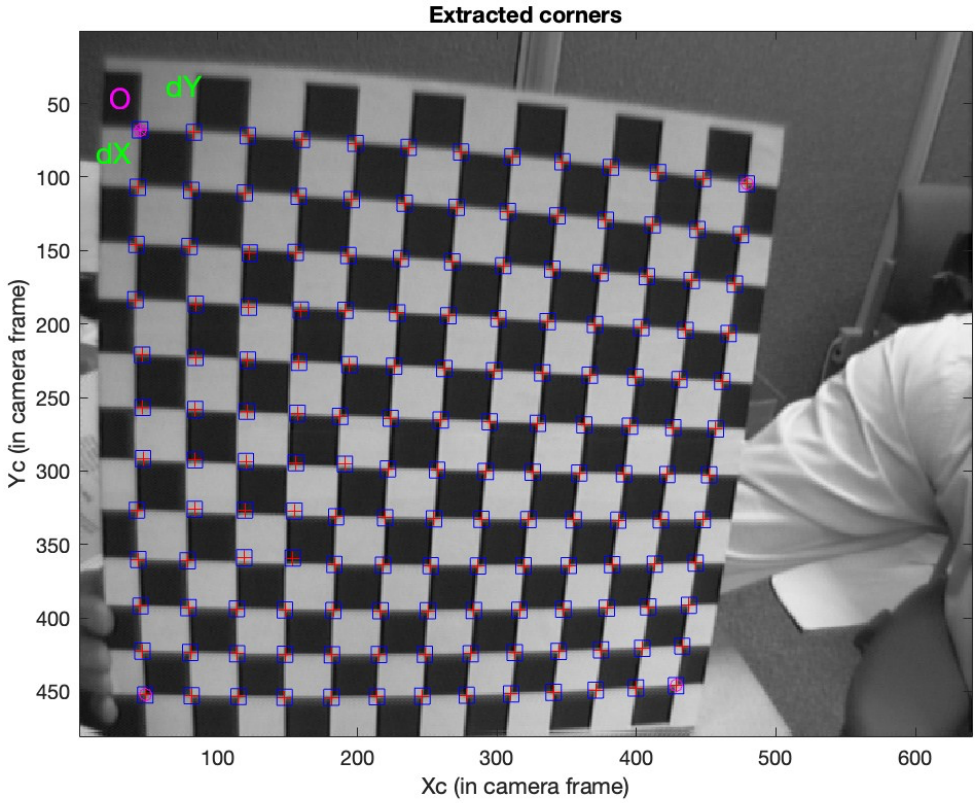

Extracted corners
O
dY
dX
Yc (in camera frame)
Xc (in camera frame)
50
100
150
200
250
300
350
400
450
100
200
300
400
500
600

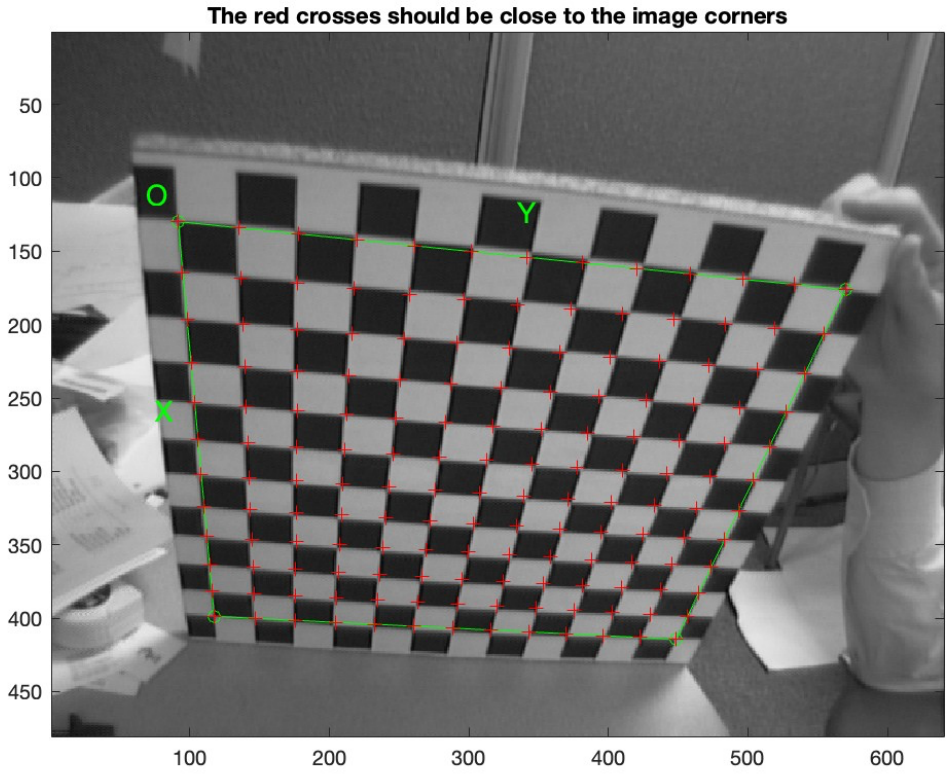

Extracted corners
O
dX
dY
Yc (in camera frame)
Xc (in camera frame)
50
100
150
200
250
300
350
400
450
100
200
300
400
500
600

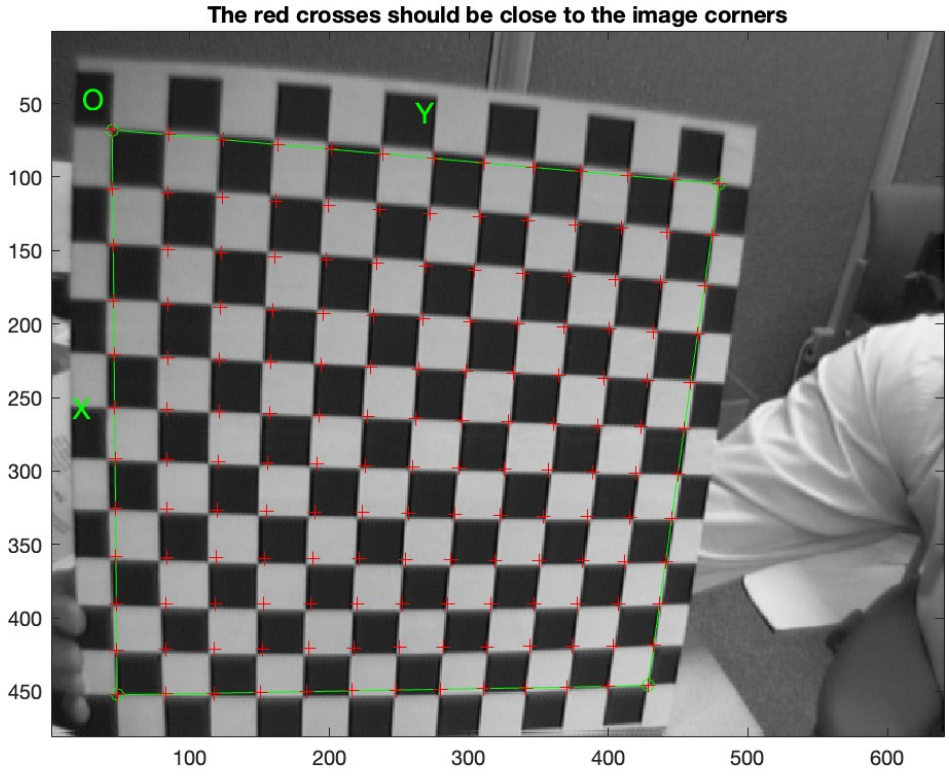

The red crosses should be close to the image corners
O
X
Y
50
100
150
200
250
300
350
400
450
100
200
300
400
500
600


Extracted corners
O
dY
dX
50
100
150
200
250
300
350
400
450
Yc (in camera frame)
100
200
300
400
500
600
Xc (in camera frame)

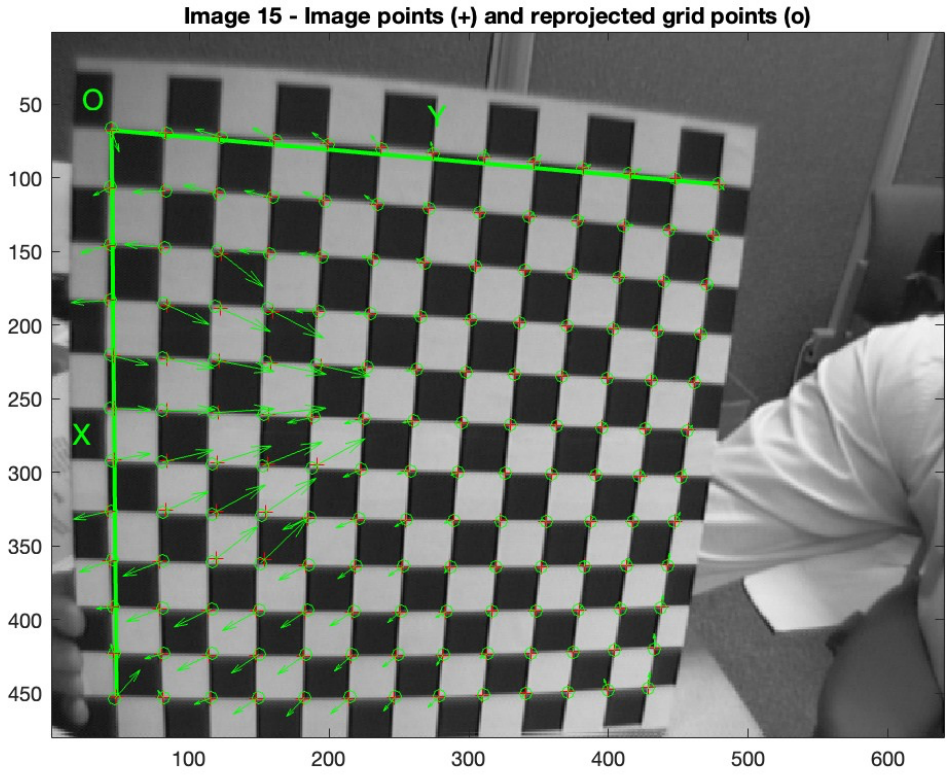

The red crosses should be close to the image corners
O
X
Y
50
100
150
200
250
300
350
400
450
100
200
300
400
500
600


Extracted corners
O
dY
dX
Yc (in camera frame)
50
100
150
200
250
300
350
400
450
100
200
300
400
500
600
Xc (in camera frame)


The red crosses should be close to the image corners
O
Y
X
50
100
150
200
250
300
350
400
450
100
200
300
400
500
600


The red crosses should be close to the image corners
O
Y
X
50
100
150
200
250
300
350
400
450
100
200
300
400
500
600

```
Initialization of the intrinsic parameters - Number of images: 5


Calibration parameters after initialization:

Focal Length:          fc = [ 668.74193   668.74193 ]
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: 5
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 = [ 666.19143   674.04044 ] +/- [ 3.90052   5.01169 ]
Principal point:       cc = [ 300.22570   241.83625 ] +/- [ 6.18357   5.35080 ]
Skew:             alpha_c = [ 0.00000 ] +/- [ 0.00000  ]   => angle of pixel axes = 90.00000 +/- 0.00000 degrees
Distortion:            kc = [ -0.32208   0.61858   -0.00057   -0.00224  0.00000 ] +/- [ 0.02811   0.12881   0.00162   0.00170  0.00000
Pixel error:          err = [ 0.72083   0.34691 ]

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



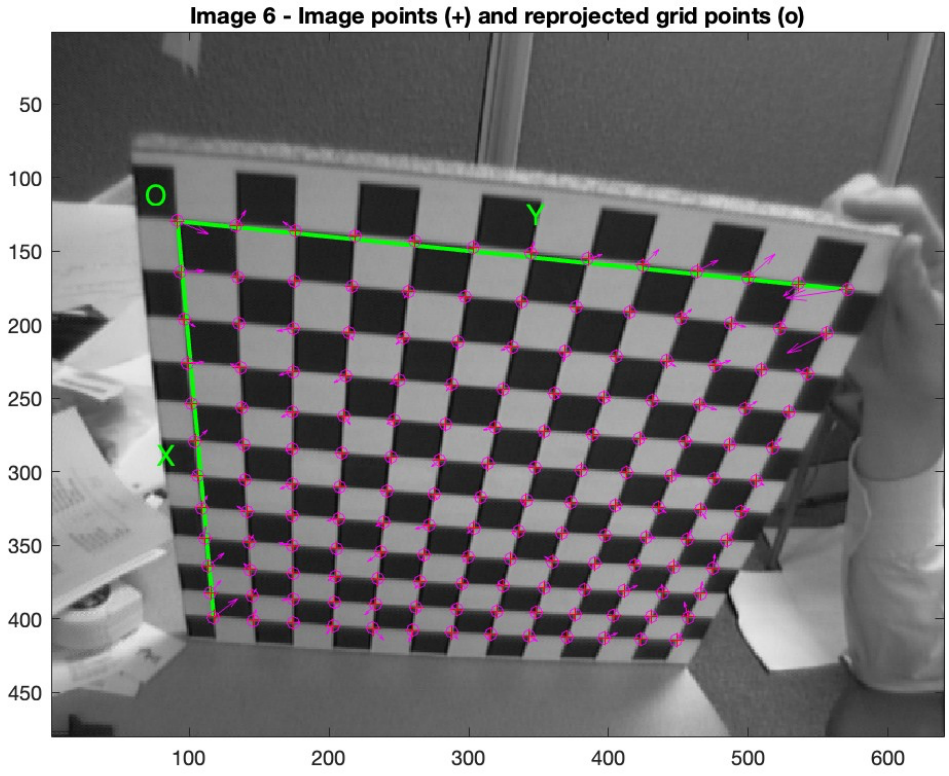


Image 6 - Image points (+) and reprojected grid points (o)
O
Y
X
50
100
150
200
250
300
350
400
450
100
200
300
400
500
600

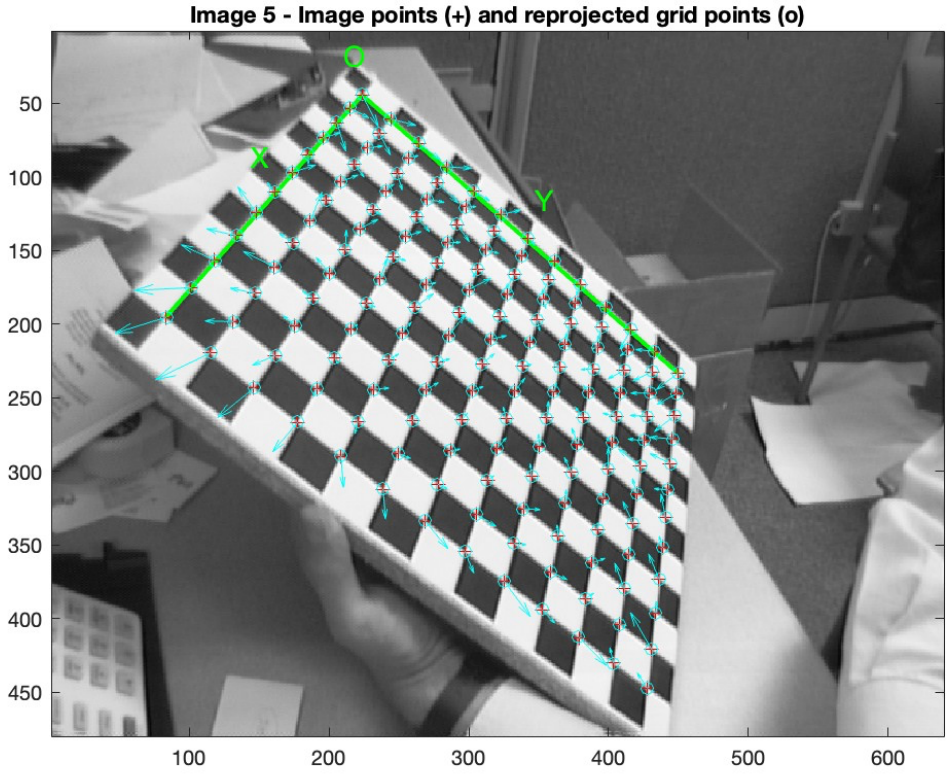


Image 5 - Image points (+) and reprojected grid points (o)
O
X
Y
50
100
150
200
250
300
350
400
450
100
200
300
400
500
600

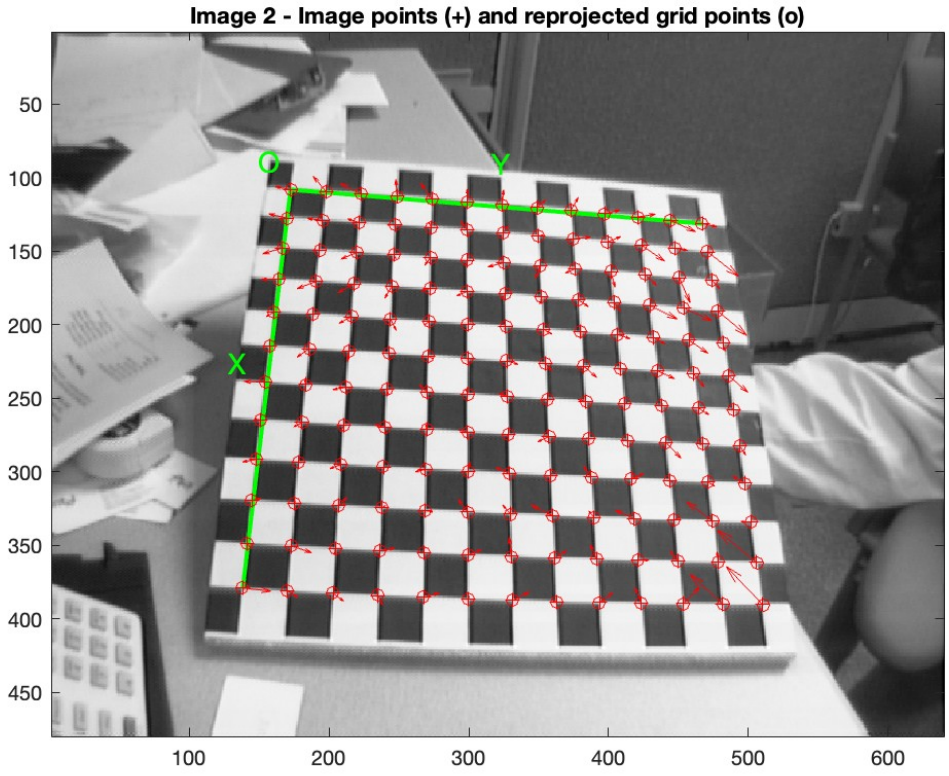


Image 2 - Image points (+) and reprojected grid points (o)
O
Y
X
50
100
150
200
250
300
350
400
450
100
200
300
400
500
600

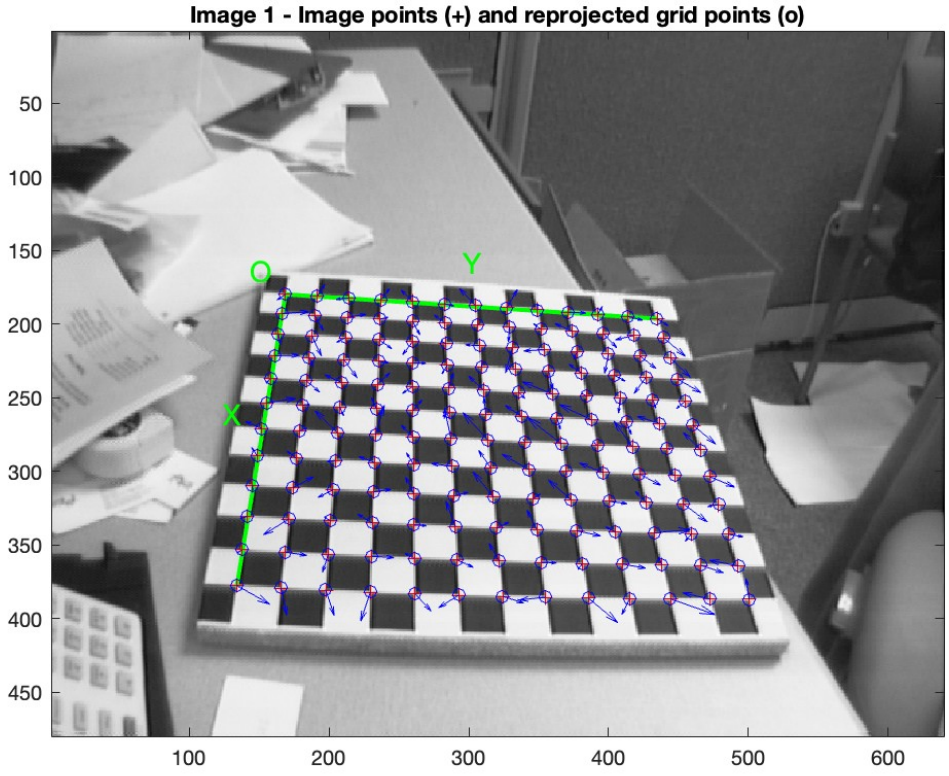
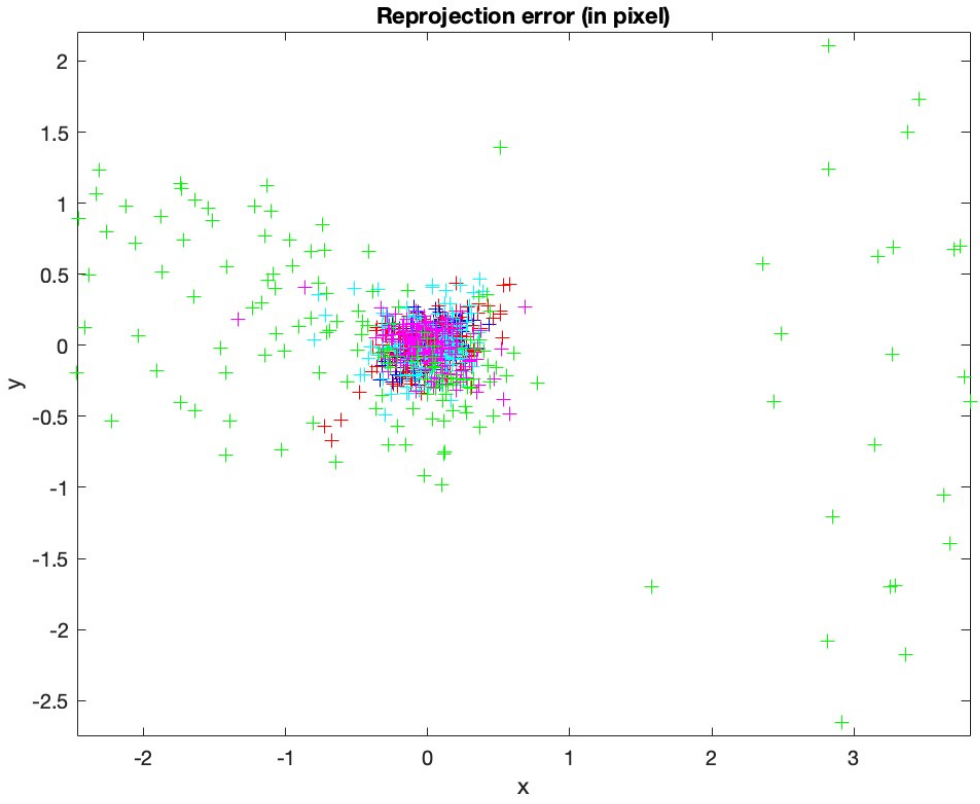


Image 1 - Image points (+) and reprojected grid points (o)
O
Y
X
50
100
150
200
250
300
350
400
450
100
200
300
400
500
600


Reprojection error (in pixel)
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2.5
y
-2
-1
0
1
2
3
x

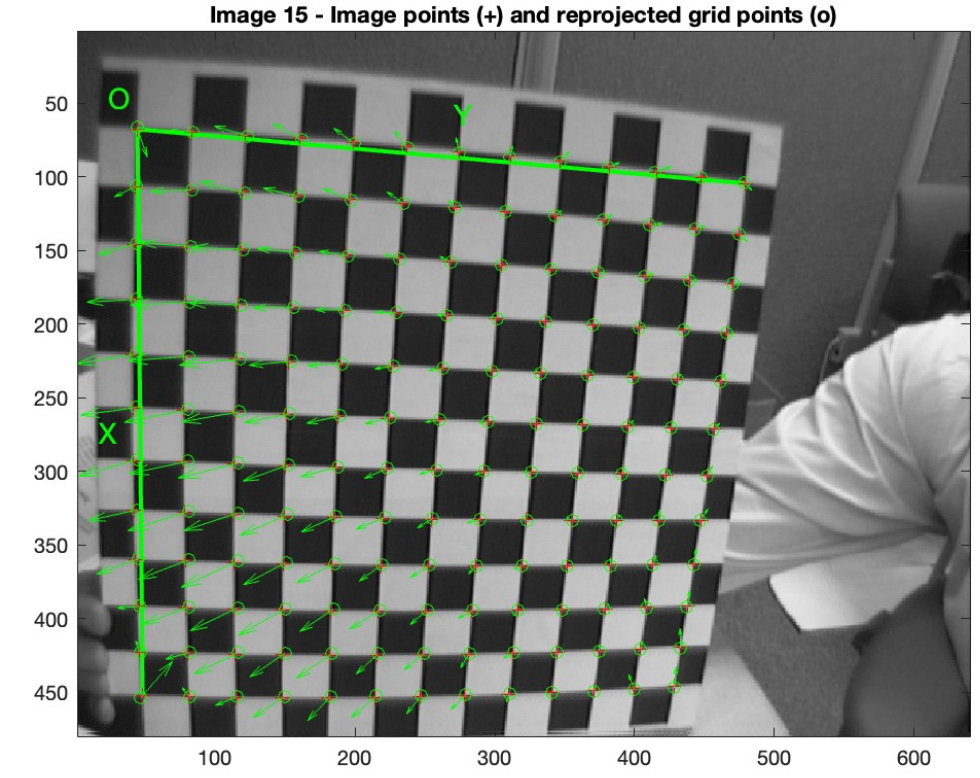


Image 15 - Image points (+) and reprojected grid points (o)
O
Y
X
50
100
150
200
250
300
350
400
450
100
200
300
400
500
600

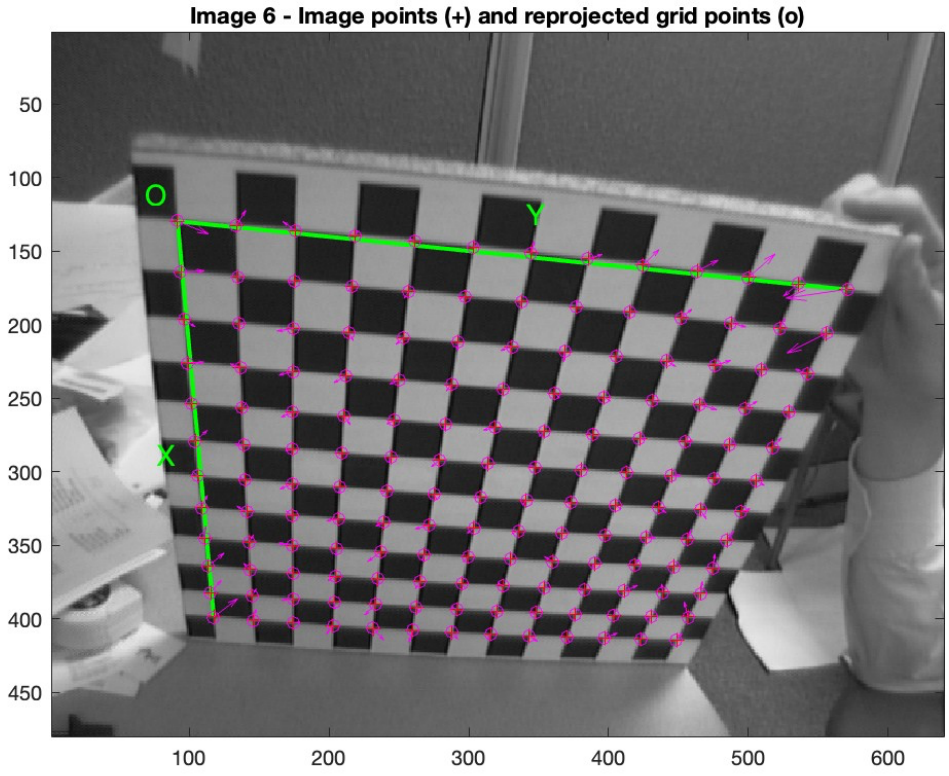


Image 6 - Image points (+) and reprojected grid points (o)
O
Y
X
50
100
150
200
250
300
350
400
450
100
200
300
400
500
600

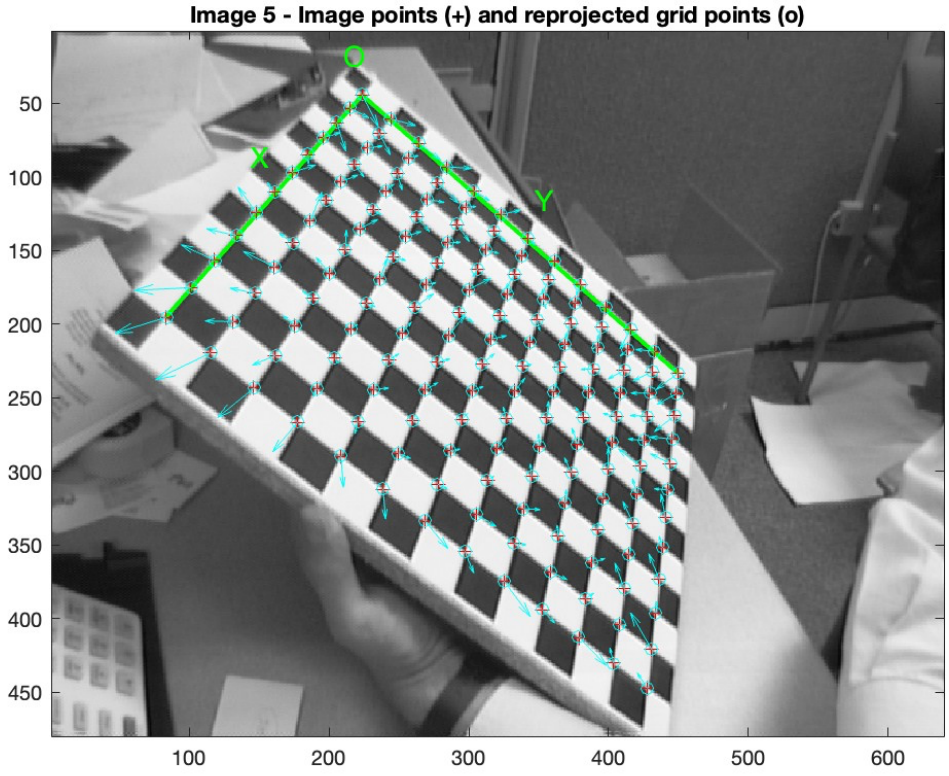


Image 5 - Image points (+) and reprojected grid points (o)
O
X
Y
50
100
150
200
250
300
350
400
450
100
200
300
400
500
600

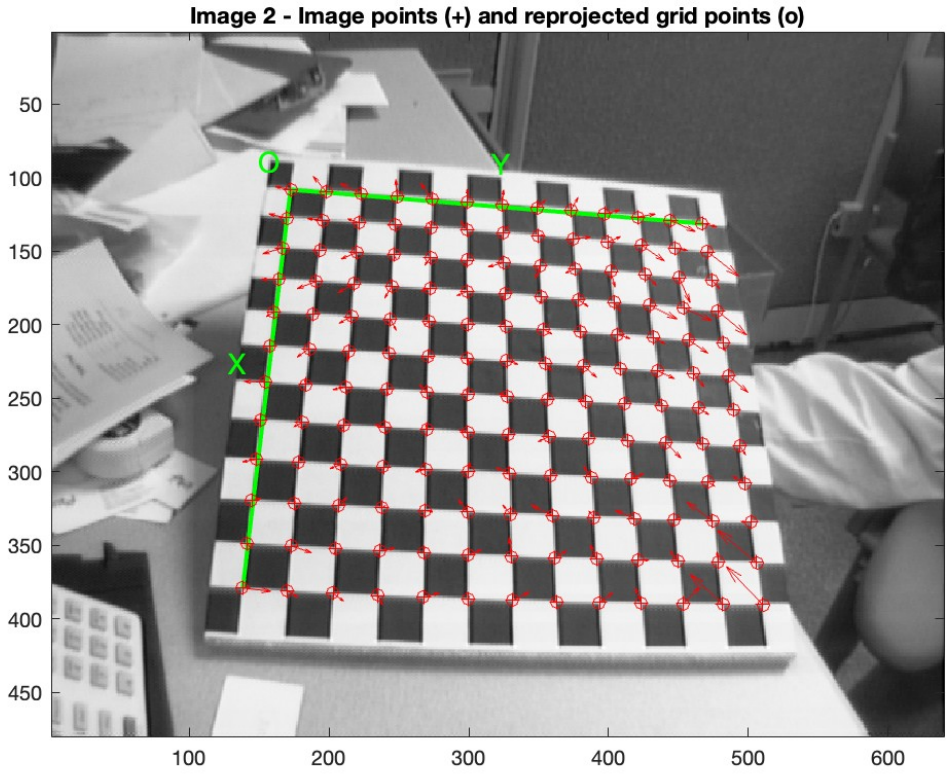


Image 2 - Image points (+) and reprojected grid points (o)
O
Y
X
50
100
150
200
250
300
350
400
450
100
200
300
400
500
600

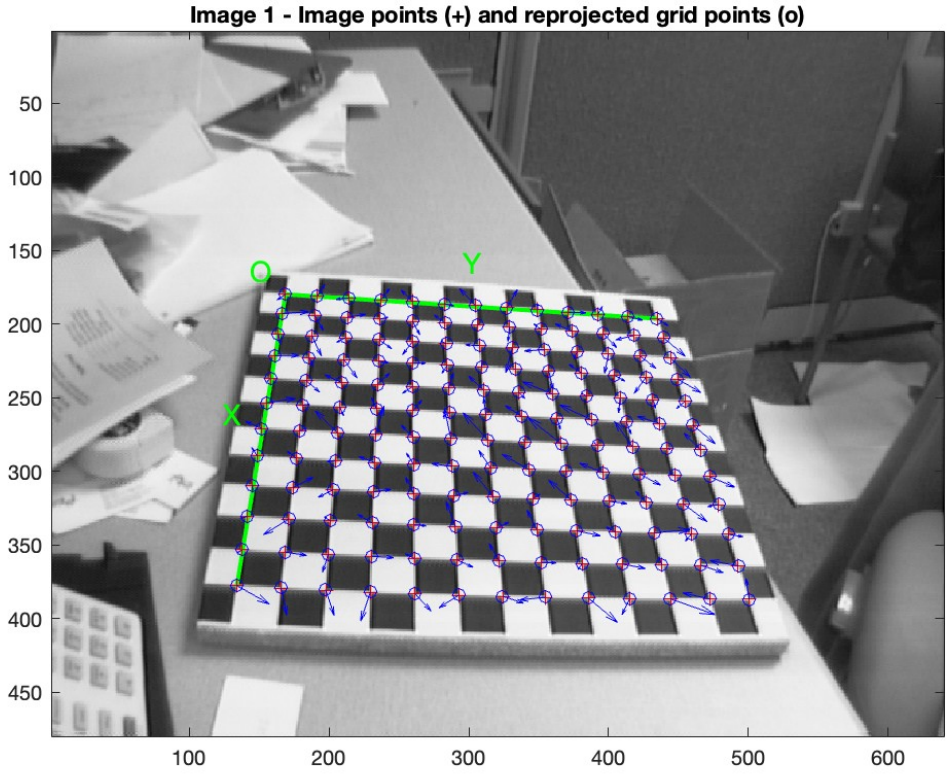
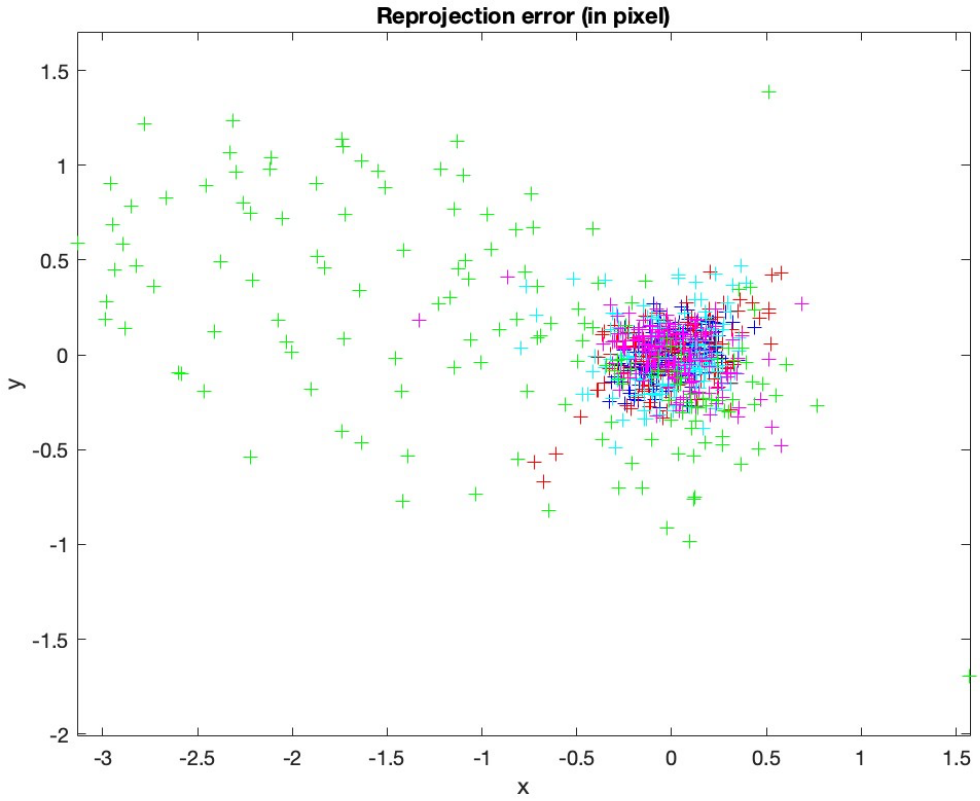
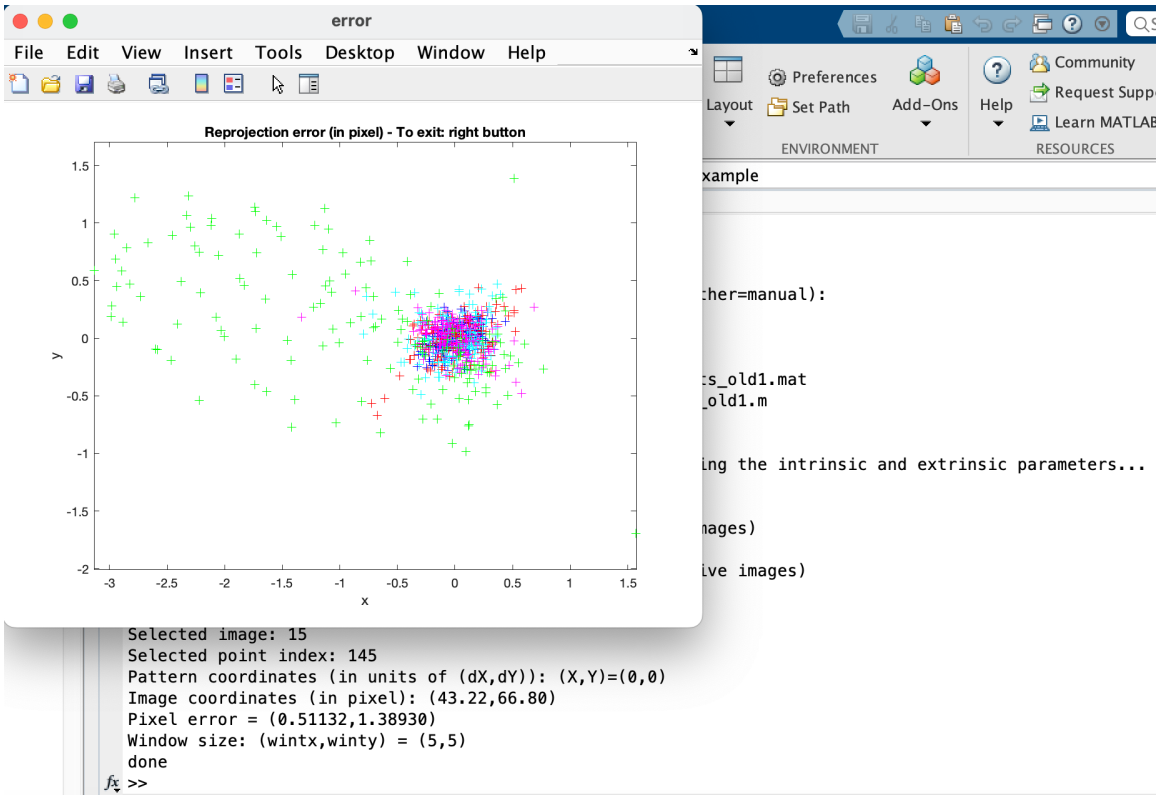
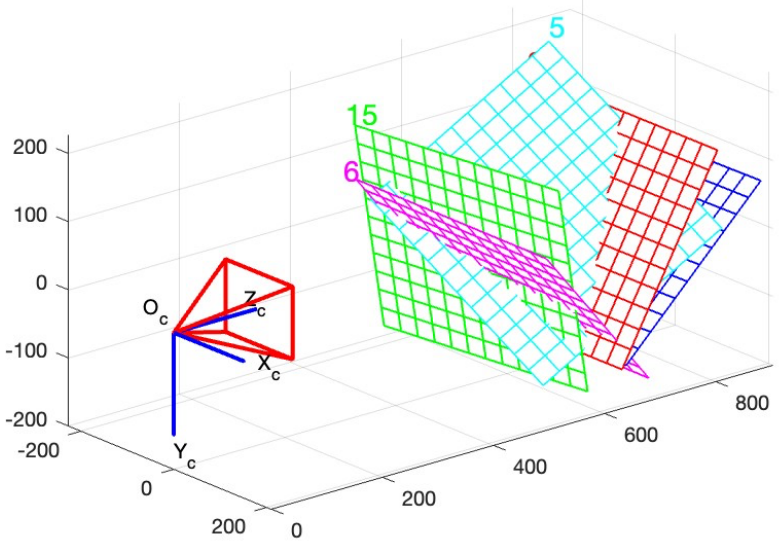


Image 1 - Image points (+) and reprojected grid points (o)
O
Y
X
50
100
150
200
250
300
350
400
450
100
200
300
400
500
600


Reprojection error (in pixel)
y
x
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-3
-2.5
-2
-1.5
-1
-0.5
0
0.5
1
1.5


error
File Edit View Insert Tools Desktop Window Help
Reprojection error (in pixel) - To exit: right button
y
x
Layout
Preferences
Set Path
Add-Ons
Help
Community
Request Supp
Learn MATLAB
ENVIRONMENT
RESOURCES
xample
her=manual):
s_old1.mat
old1.m
ng the intrinsic and extrinsic parameters...
ages)
ve images)
Selected image: 15
Selected point index: 145
Pattern coordinates (in units of (dX,dY)): (X,Y)=(0,0)
Image coordinates (in pixel): (43.22,66.80)
Pixel error = (0.51132,1.38930)
Window size: (wintx,winty) = (5,5)
done
>>

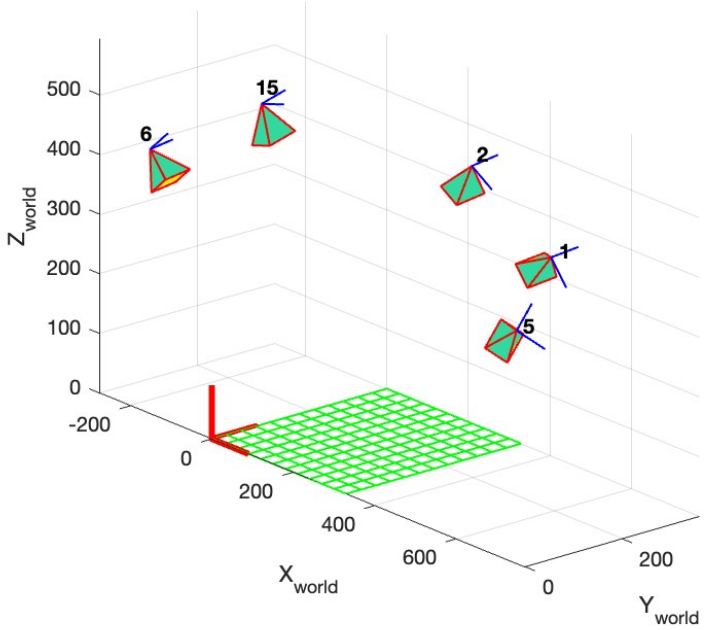
Extrinsic parameters (camera-centered)



Remove camera reference frame

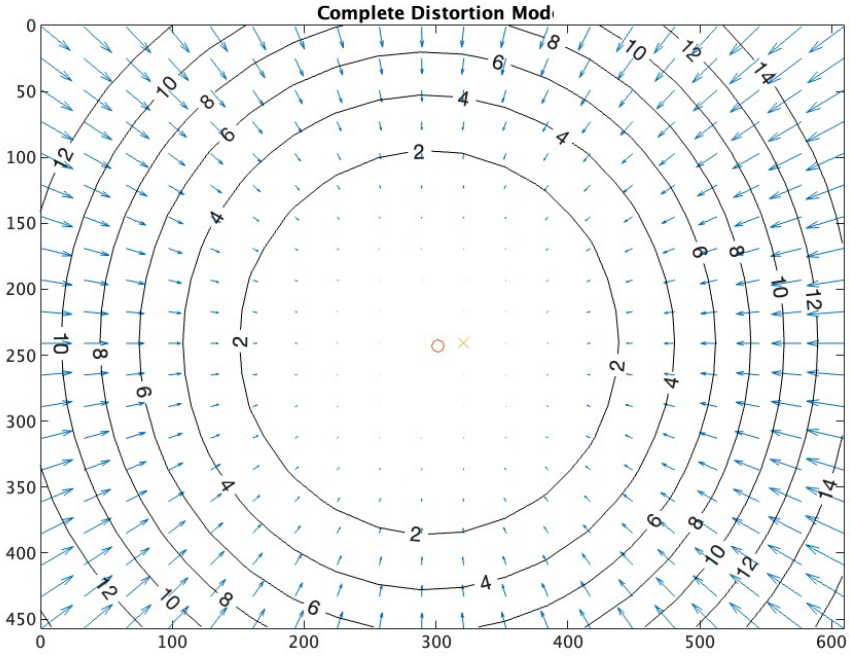
Switch to world-centered view

Extrinsic parameters (world-centered)

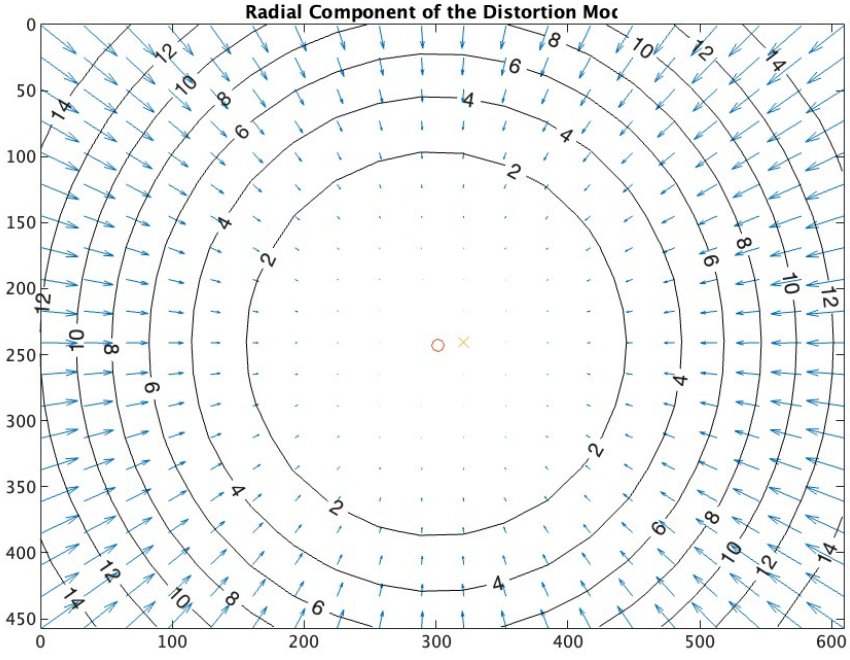


Remove camera reference frames

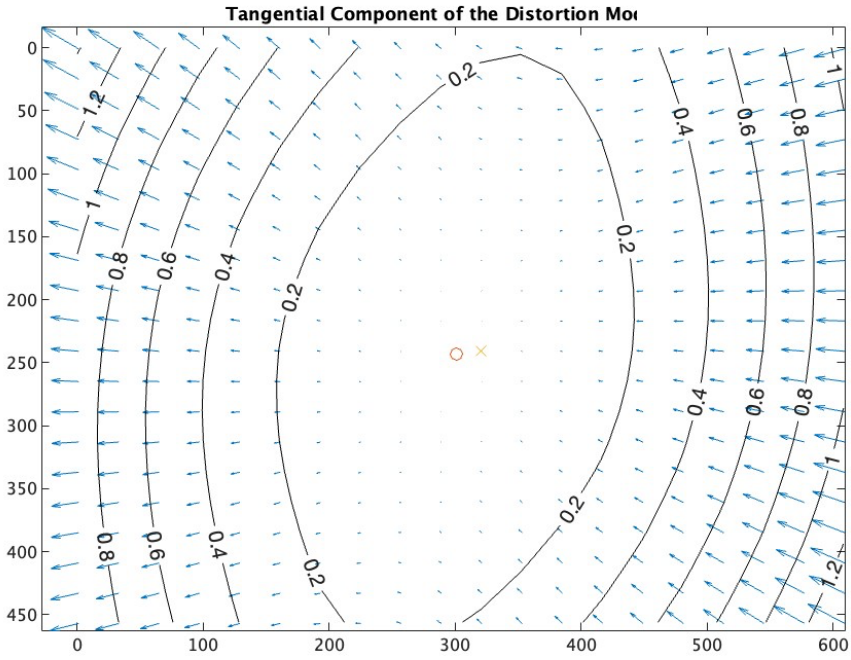
Switch to camera-centered view



Pixel error	= [0.6203, 0.2802]	
Focal Length	= (666.191, 674.04)	+/- [3.901, 5.012]
Principal Point	= (300.226, 241.836)	+/- [6.184, 5.351]
Skew	= 0	+/- 0
Radial coefficients	= (-0.3221, 0.6186, 0)	+/- [0.02811, 0.1288, 0]
Tangential coefficients	= (-0.0005741, -0.002236)	+/- [0.001615, 0.001699]



Pixel error	= [0.6203, 0.2802]	
Focal Length	= (666.191, 674.04)	+/- [3.901, 5.012]
Principal Point	= (300.226, 241.836)	+/- [6.184, 5.351]
Skew	= 0	+/- 0
Radial coefficients	= (-0.3221, 0.6186, 0)	+/- [0.02811, 0.1288, 0]
Tangential coefficients	= (-0.0005741, -0.002236)	+/- [0.001615, 0.001699]



Pixel error	= [0.6203, 0.2802]	
Focal Length	= (666.191, 674.04)	+/- [3.901, 5.012]
Principal Point	= (300.226, 241.836)	+/- [6.184, 5.351]
Skew	= 0	+/- 0
Radial coefficients	= (-0.3221, 0.6186, 0)	+/- [0.02811, 0.1288, 0]
Tangential coefficients	= (-0.0005741, -0.002236)	+/- [0.001615, 0.001699]

---

1. If the initial selection of corners is not accurate, how would that impact the final result. Consider the fact that the initial corner estimate is used only in the closed form, or linear regression, portion of the calibration routine. You should consider two distinct cases: the initial corner selection is only slightly incorrect and if the initial selection is significantly in error.

If the initial selection of corners is inaccurate, it can significantly impact the final calibration result. In cases where the initial corner selection is only slightly incorrect, the effect on the final outcome might be minimal. However, if the initial selection is significantly erroneous, it could lead to substantial errors in the calibration process.

In the closed-form or linear regression phase of the calibration routine, the initial corner estimates serve as the starting point for refining the camera model using numerical methods such as gradient descent. If these initial estimates are close to the actual corner locations, the iterative refinement process might converge to a satisfactory solution, albeit with some additional iterations to correct for the initial error.

However, if the initial corner selection is significantly flawed, it could lead to the optimization process converging to a local minimum rather than the global minimum. This can result in a calibration with larger errors or even failure to converge to a valid solution. Additionally, inaccuracies in the initial corner selection may propagate throughout the calibration process, leading to distortions in the final camera model.

In summary, the accuracy of the initial corner selection is crucial for the success of the calibration process. Even slight inaccuracies can necessitate additional iterations to achieve convergence, while significant errors can lead to unreliable calibration results.

---

2. How could the "significant error" case be identified as part of the sequence of steps followed above?

In the sequence of steps outlined in the assignment, identifying the "significant error" case involves several observations and actions during the calibration process:

1. Initial Corner Selection: During the initial stage of corner selection, if the chosen corners appear noticeably misaligned with the actual corners of the checkerboard pattern in the images, it could indicate a significant error. This can be visually assessed by comparing the selected corners to the actual corners of the checkerboard.

2. Corner Extraction: After selecting the corners, the corner extraction process involves automatically detecting the checkerboard corners in the images. If the extracted corners deviate substantially from the initially selected corners, it suggests that the initial selection was erroneous.

3. Reprojection Errors: After performing the calibration and reprojecting the points onto the images, examining the reprojection errors provides further insight. If certain points exhibit large reprojection errors (e.g., exceeding 0.5 pixels), it indicates that those corners may have been incorrectly estimated during the initial selection.

4. Analysis of Errors: Utilizing the "Analyze error" function allows for the examination of individual points contributing to the overall error. If points near the edges of the error plot consistently appear, it suggests that corners in those regions may have been inaccurately selected initially.

5. Re-evaluation of Calibration: If the reprojection errors are high or the analysis of errors highlights specific problematic corners, it indicates the need for re-evaluation. This involves repeating the corner estimation routine with the current estimates as initial guesses, followed by recalibration.

By closely monitoring these steps and observing any deviations or inconsistencies, the "significant error" case can be identified within the calibration process. This recognition prompts corrective actions, such as re-evaluating corner selections and recalibrating, to improve the accuracy of the camera calibration.

---

3. How do you expect the error in the calibration model to change for the following conditions:

a. Only two images are used, with only a slight rotation between them

- **Limited Accuracy:** Using just two images, especially with minimal rotation, provides limited information for the calibration process. This can lead to a less accurate camera model with potentially higher error compared to scenarios with more diverse image sets.

b. A modest number of images are used, with many variations in image orientations, such as rotations and translations.

- **Improved Calibration:** Employing a moderate number of images that capture various orientations (rotations and translations) should yield a more accurate calibration with lower error. The increased variety of viewpoints allows the calibration process to better capture the camera's response under different viewing conditions.

c. A very large number images are used, with many different variations in image orientations, rotations and translations. Keep in mind that several of these images may be very similar to other images and that numerical methods are used as part of the error reduction approach (that is, to find the solutions of the camera model equations). Will the error always goes to zero as the number of images used increases ? Why or why not?

- **Reduced Error, Not Zero:** While using a very large number of images with significant variations in orientation generally reduces calibration error, it might not always reach zero. Here's why:
    - **Measurement Imperfections:** Even with many images, slight errors in capturing the checkerboard (imperfect image acquisition or minor square size variations) can introduce noise into the calibration.

- **Numerical Methods:** The numerical optimization techniques used for error reduction might not achieve a perfect minimum error due to inherent limitations or tolerances.
- **Model Limitations:** The chosen camera model itself might not be able to perfectly capture all the complexities of real-world lens distortion.

d. If the entered square size is incorrect, the values of dX or dY, how could this be identified during the calibration sequence used?

- **Reprojection Errors and Visual Inspection:** An incorrect square size or dX/dY values can be identified through reprojection error analysis and visual inspection:
  - Reprojection Errors: High overall reprojection error or outliers with significant deviations on the reprojection error plot might indicate an issue.
  - Visual Inspection: After reprojection, a mismatch between the reprojected corners and the actual checkerboard corners in the image could be a sign of an incorrect square size or dX/dY values.

---

4. a. How important is even lighting for this calibration approach?

Even lighting is quite important for this camera calibration approach for a few reasons:

Accurate Corner Detection: Uneven illumination across the image can make it difficult to accurately detect the checkerboard corners. Areas with significant shadows or highlights might lead to missed corner points or imprecise localization. Consistent lighting ensures all corners are clearly visible and facilitates reliable detection.
Consistent Distortion: Variations in lighting can introduce inconsistencies in how the lens distorts the image across different regions. Even lighting helps to minimize these variations and allows the calibration process to capture a more consistent model of the lens distortion.

b. Why is a regular pattern of squares of known size used, why not just use the outer corner points of each image ?

A regular pattern of squares of known size is used instead of just using the outer corner points of each image due to several reasons. Firstly, the regular pattern provides more comprehensive information about the camera's geometry and perspective distortion. By capturing the entire checkerboard pattern, the calibration algorithm can analyze distortions across the entire image area, leading to a more precise estimation of camera parameters. Additionally, the known size of the

squares facilitates accurate measurement of distances in the image, which is crucial for calibration. Using only the outer corner points may not provide sufficient data to fully characterize the camera's intrinsic and extrinsic parameters, resulting in less accurate calibration outcomes.

---

5. Increasing wintx and winty should make it easier to find corner points, and therefore reduce error. Why is this statement not true for large values of wintx and winty ? What could increase the error for large wintx and winty values depending on the indicated (clicked) point ?

While increasing wintx and winty might seem like a straightforward way to improve corner point detection and reduce error, it's not always beneficial for large values. Here's why:

- **Increased Ambiguity:** Larger window sizes encompass a bigger image area around the clicked point. This can introduce ambiguity, especially in areas with repetitive patterns or textures. The corner finder might struggle to identify the true corner point within the larger window, potentially leading to selecting a suboptimal location.
- **Noise Amplification:** Image noise can become more prominent with larger window sizes. The increased area incorporates more noise pixels, which can influence the corner finder and lead to inaccurate corner localization, potentially increasing error.

Here's how large wintx and winty values can increase error depending on the clicked point:

- **Off-Center Clicks:** If you click slightly off-center from the actual corner, a larger window might include a significant portion of the adjacent square. This can mislead the corner finder and cause it to select a point on the edge of the adjacent square instead of the true corner, resulting in significant error.

---

6. Read this blog post, from Cleve Molar of Mathworks, on one definition of a matrix condition number, particularly the part on matrix inverses: https://blogs.mathworks.com/cleve/2017/07/17/what-is-the-condition-number-of-a-matrix/

How would the condition number of the matrix used in determining camera calibration potentially be significant in finding the numerical solution?

1. Impact on Numerical Stability: The condition number of the matrix used in determining camera calibration is crucial for numerical stability during the solution process. A high condition number implies that the matrix is ill-conditioned, indicating that small changes in input data or roundoff errors during computation can significantly affect the accuracy of the solution.

2. Sensitivity to Perturbations: A high condition number indicates that the matrix is sensitive to perturbations in the input data. In the context of camera calibration, this sensitivity means that even minor variations in the corner points detected in images or inaccuracies in measurements can lead to substantial errors in the calibration model.

3. Effect on Error Magnification: The condition number acts as a magnification factor for errors. In camera calibration, a high condition number amplifies the impact of errors in the input data, such as inaccuracies in corner detection or noise in images. This can result in larger discrepancies between the estimated camera parameters and the true values.

4. Influence on Calibration Accuracy: The condition number indirectly affects the accuracy of the camera calibration process. A higher condition number indicates a more challenging numerical problem, making it harder to obtain accurate estimates of camera parameters. Therefore, understanding and managing the condition number is essential for achieving reliable and precise camera calibration results.

---

7. Floating point machine arithmetic is not always exact. In particular, the minimum difference between floating point numbers that a computer can resolve is often called the "machine epsilon". In MATLAB this is assigned the variable: eps. Type "eps" at the command prompt in MATLAB and report the result. How is this significant in determining the stopping condition of any numerical algorithm, such as gradient descent or Newton's method used in camera calibration?

The machine epsilon (eps), typically around 2.2204e-16 in MATLAB, represents the smallest difference between floating-point numbers. It's crucial in determining the stopping condition of numerical algorithms like gradient descent or Newton's method in camera calibration.

Here's why eps matters:

1. Convergence Criterion: eps sets a threshold for stopping iterations. When the change in solution becomes smaller than eps, it suggests further iterations won't significantly enhance accuracy, aiding in convergence.

2. Error Margin: Monitoring changes relative to eps helps define acceptable error margins. The algorithm aims for decreasing errors close to eps, indicating convergence to desired accuracy.

3. Overfitting Prevention: Stopping iterations based on eps prevents overfitting. Continuing beyond eps may not substantially improve accuracy and can lead to unnecessary computational burden.

4. Numerical Stability: Considering eps ensures numerical stability. Stopping when changes are comparable to eps helps avoid issues with round-off errors and instability.

In essence, eps guides when to halt iterations, balancing computational efficiency with solution accuracy in camera calibration and other numerical algorithms.
