EECE 5554 Lab_4 Report

Shaoshu Xu

Hardware / Camera: iPhone7 Plus camera with ios 13

Part 1 – Camera Calibration

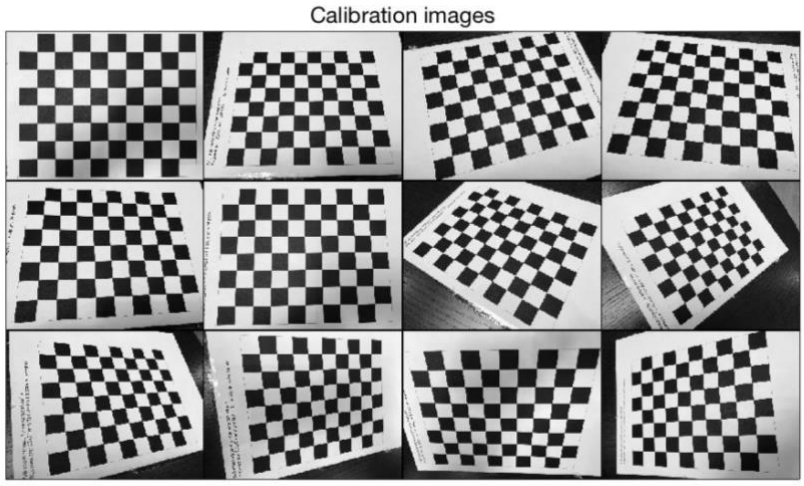


Figure 1: Complete set of calibration images

Figure 1 shows the complete set of calibration images taken by iPhone7 Plus camera. The checkboard pattern paper is firmly attached to a flat table and each square is 20mm x 20mm. **To produce good results, the images should cove the corners of the checkboard pattern enough.**

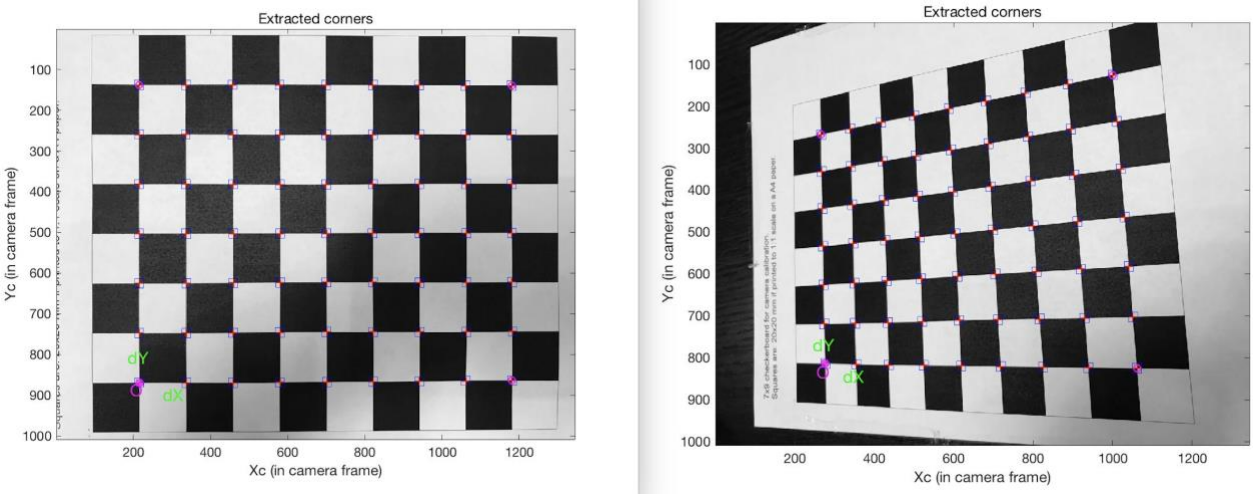


Figure 2: Samples of extracted image corners

After carefully clicking on the four extreme corners on the rectangular checkerboard pattern of each image, the grid corners are automatically extracted by the MATLAB Camera Calibration Toolbox [1], and two samples are displayed on figure 2 (the blue squares around the corner points show the limits of the corner finder window).

```
Calibration parameters after initialization:

Focal Length:          fc = [ 1107.47815   1107.47815 ]
Principal point:       cc = [ 671.50000   503.50000 ]
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: 12
Gradient descent iterations: 1...2...3...4...5...6...7...8...9...10...11...12...13...14...15...16...17...18...19...done
Estimation of uncertainties...done


Calibration results after optimization (with uncertainties):

Focal Length:          fc = [ 1110.89623   1116.71053 ] +/- [ 4.03865   4.11922 ]
Principal point:       cc = [ 670.63682   495.94804 ] +/- [ 2.74556   2.49860 ]
Skew:             alpha_c = [ 0.00000 ] +/- [ 0.00000  ]   => angle of pixel axes = 90.00000 +/- 0.00000 degrees
Distortion:            kc = [ 0.09996   -0.31438   -0.00027   -0.00076  0.00000 ] +/- [ 0.00943   0.03128   0.00075   0.00090  0.00000 ]
Pixel error:          err = [ 0.46287   0.44742 ]

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

Figure 3: Parameters after first calibration

The first calibration parameters are shown in figure 3, we can notice that 19 gradient descent iterations are implemented to reach the minimum. **The standard deviation of the reprojection error is [0.46287, 0.44742] and the mean error is $1.0e^{-11}$*[3.836, 3.087]**. The following figure 4 shows two images with the detected corners and the reprojected grid corners.

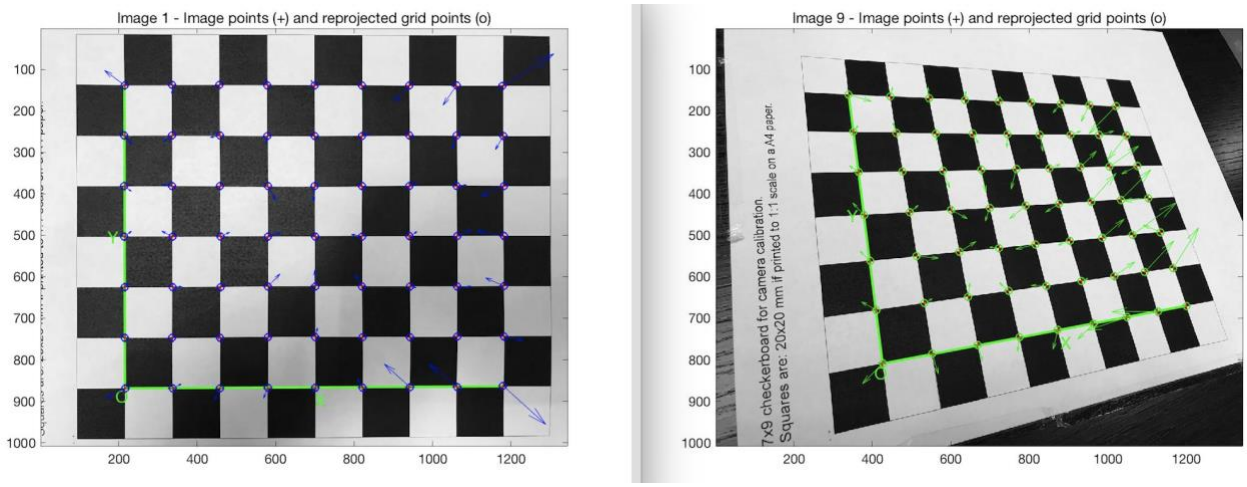
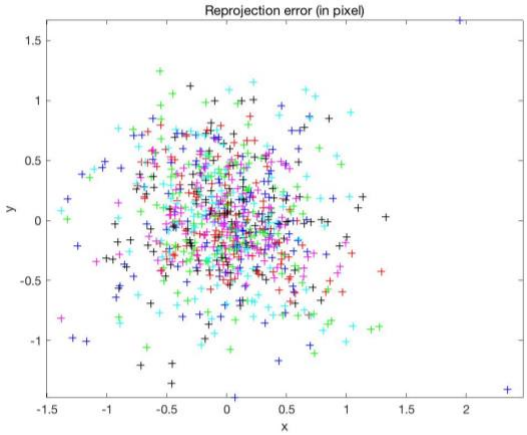


Figure 4: Samples of the reprojection of the grids onto the original images

The reprojection error is plotted in the form of color-coded crosses in figure 5. The spread of the reprojection error are mostly between +/- 0.5 pixels.




```
>> mean(ex')
ans =
   1.0e-10 *
    0.3836    0.3087
```

Figure 5: The reprojection error (in pixel) and mean error

The extrinsic parameters (relative positions of the grids with respect to the camera and world) are shown in a form of a 3D plot in figure 6.

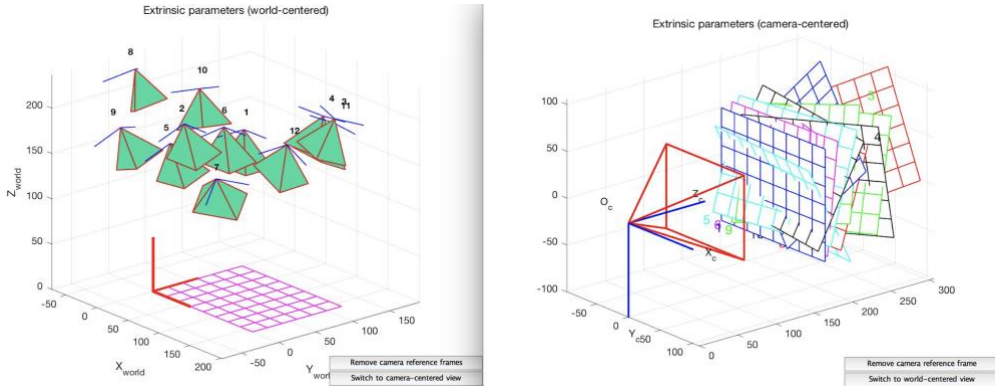


Figure 6: The extrinsic parameters

The second computation and calibration are implemented automatically by using the toolbox. Figure 7 shows the new calibration parameters. We can notice that **the results have no obvious change** and reprojection error is very large across many figures, also **one of the distortion parameter value is always too big.**

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

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


Calibration results after optimization (with uncertainties):

Focal Length:          fc = [ 1110.50569   1116.28693 ] +/- [ 4.03439   4.11481 ]
Principal point:       cc = [ 670.53527   496.04872 ] +/- [ 2.74307   2.49563 ]
Skew:             alpha_c = [ 0.00000 ] +/- [ 0.00000  ]   => angle of pixel axes = 90.00000 +/- 0.00000 degrees
Distortion:            kc = [ 0.09975   -0.31368   -0.00024   -0.00079  0.00000 ] +/- [ 0.00942   0.03122   0.00075   0.00090  0.00000 ]
Pixel error:          err = [ 0.46344   0.44635 ]

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

Figure 7: Parameters after second corner computation and calibration

Two possible reasons are stated below: First, we have not done a very careful job at extracting the corners on some highly distorted images. Second, **due to the depth of field** (DOF, also called *focus range* or *effective focus range*) [2] of the image, **some grid points will become blurred** (example image shown in figure 8), especially when the image is taken at a very oblique angle. Thus, some of the grid corners were not very precisely extracted for many images.

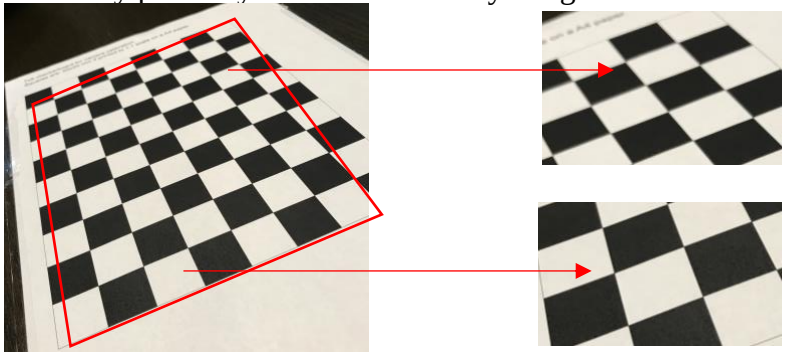


Figure 8: Sharpness of grid corner influenced by DOF

Figure 9 shows the reprojection error after second calibration and the two largest error points.

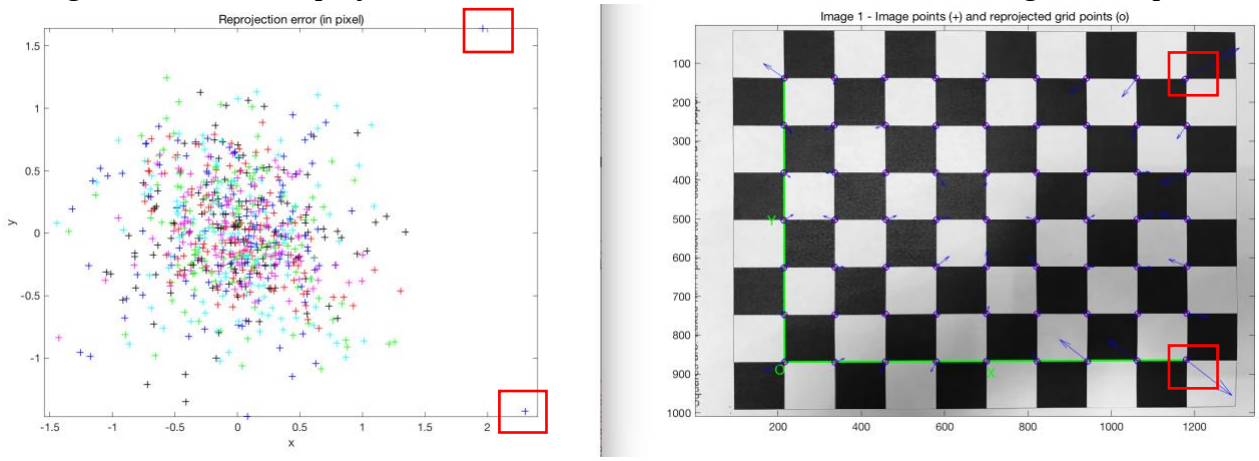


Figure 9: Analyze error to inspect which points correspond to large errors

By inspecting the corresponding largest error points using the Error-Analysis function in the toolbox, we get the detailed information as shown in figure 10.

```
Selected image: 1
Selected point index: 9
Pattern coordinates (in units of (dX,dY)): (X,Y)=(8,6)
Image coordinates (in pixel): (1179.44,138.17)
Pixel error = (2.30364,-1.42576)
Window size: (wintx,winty) = (5,5)
done
```

Figure 10: Analyze error to inspect which points correspond to large errors

This means that the corresponding point is on image 1, at the grid coordinate (8,6) in the calibration grid (at the origin of the pattern). The following image 11 shows a close view of that two points (8,6) and (8,0) on the calibration image.

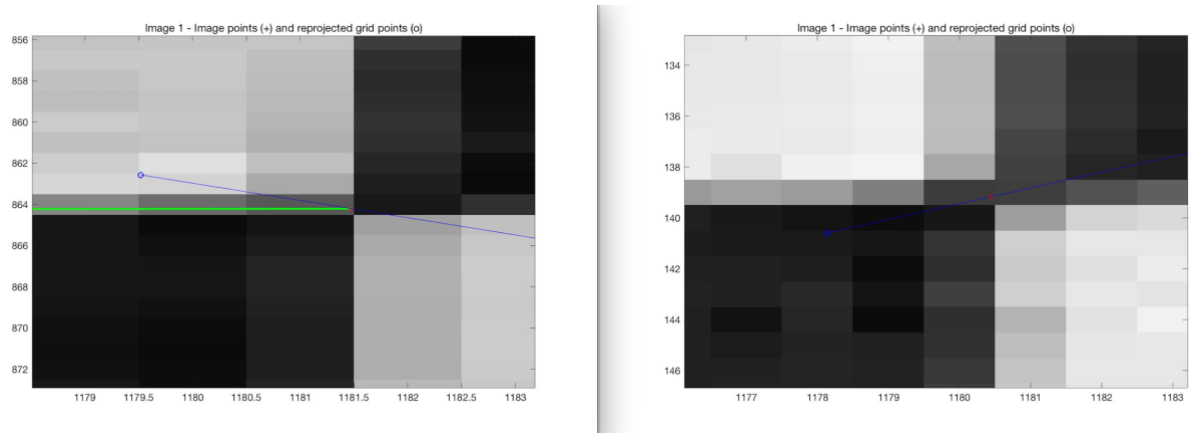


Figure 11: Close view of the largest error points on the calibration image

In figure 12, 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. The second figure shows the impact of the tangential component of distortion. 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.

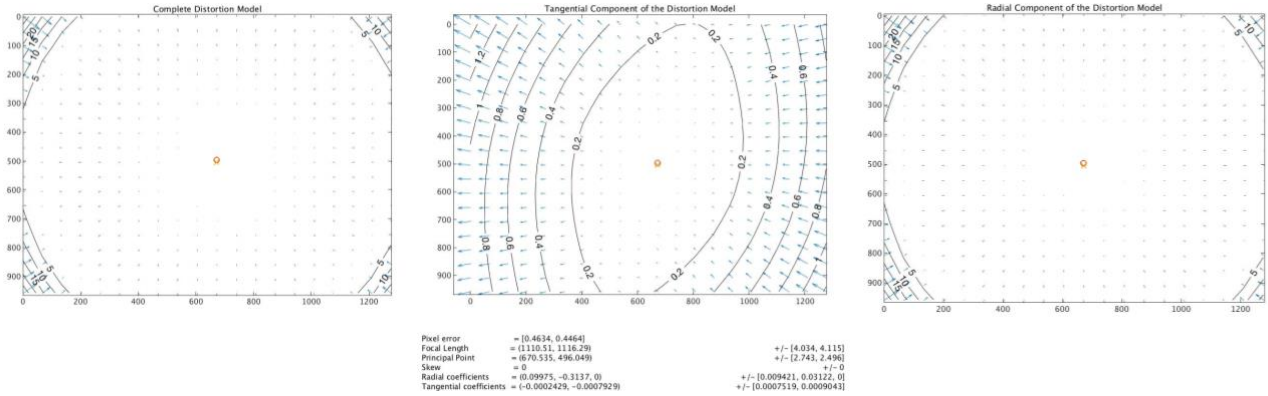


Figure 12: Complete Distortion Model, Tangential Component of Distortion Model and Radial Component of Distortion Model

The following two figures, figure 13 and figure 14 show two samples of original calibration images and undistorted images (after calibration). **We can observe that the undistorted images get more distorted, especially at the four corners.**

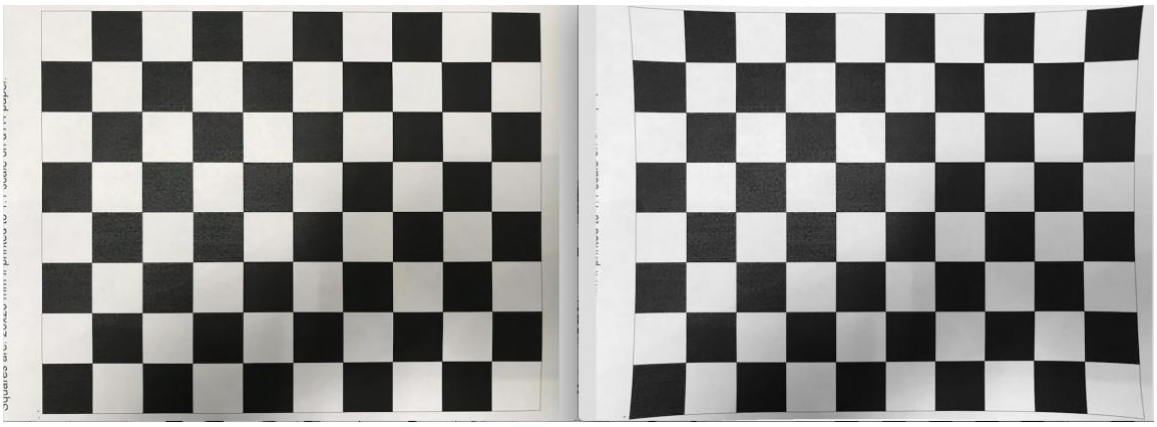


Figure 13: Sample of original calibration image and undistorted image

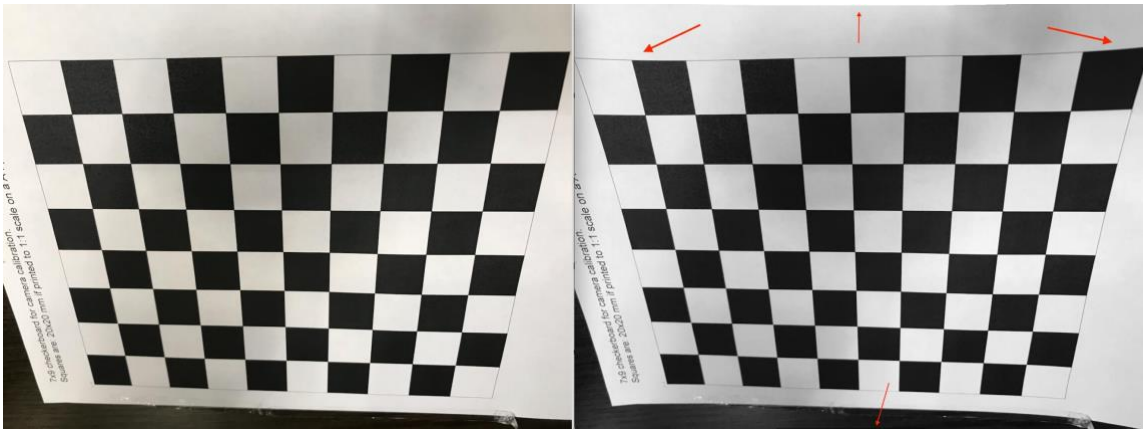


Figure 14: Sample of original calibration image and undistorted image

After further review of some online resources, it turns out that **Apple now calculates the intrinsic parameters of its cameras and undistorts the image [3]**. It also provides support for other apps that want to use the calibration parameters. By observing the first original calibration image (the left image in figure 13), **which is taken right above the grid paper, we can see that there is hardly any distortion on each corner. It is likely that the photo has already been calibrated by the camera. Thus, it is reasonable to see the over-calibration case that happened. As shown in the right image, the four corners become more distorted than the left image.**

For better analysis, both the original images and the over-calibrated images are used to generate panorama in the Photo Mosaicking part.

Part 2 – Photo Mosaicking

1. Mural images on the Latino Students Center building

Figure 15: Multiple overlapping images of mural

Figure 15 shows seven overlapping mural images taken with similar orientation at Latino Student Center building.

2. Sample before/after distortion correction

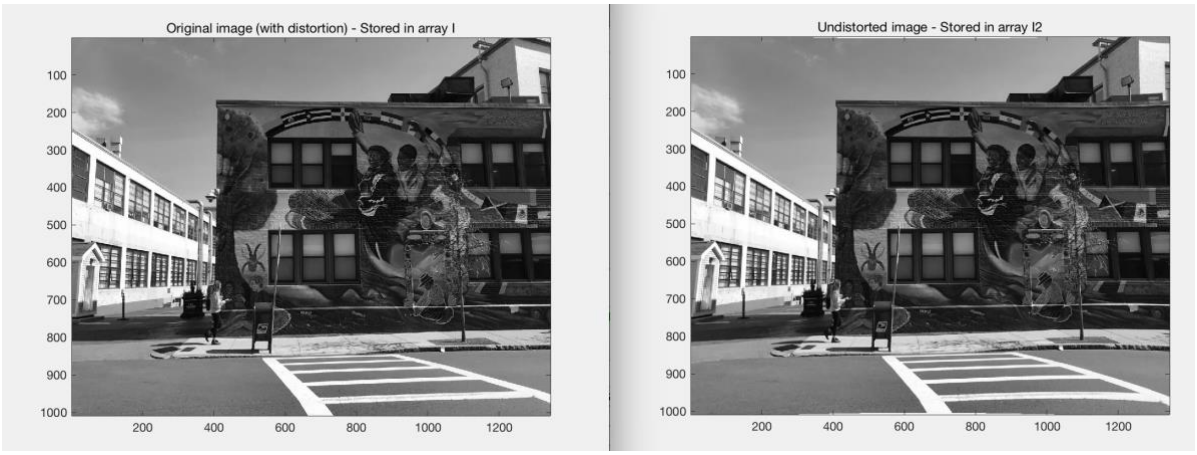


Figure 16: Sample image before/after distortion correction

A sample of original image and undistorted image (after calibration) shown in figure 16.

3/4. Harris feature detector

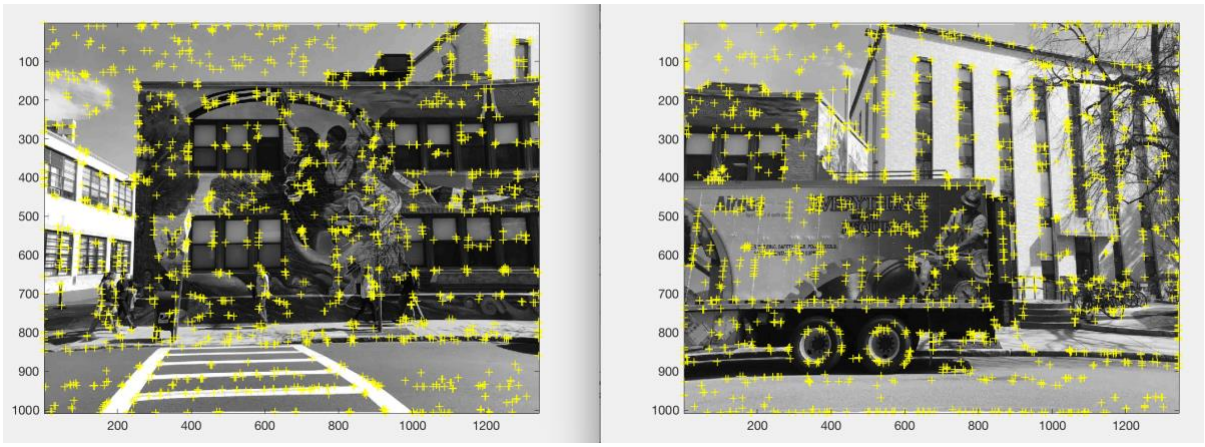


Figure 17: Sample of Harris feature detector result

To make features well distributed across the whole image, the parameter *maximum number of interest points* is set to 1500 and *tile* is [15, 15]. Figure 17 shows two samples of Harris feature detector results.

5. Estimate the position

The estimated position of each camera for each image is shown as the red circles in figure 18. The x coordinate is defined as the 1/2 value of each image's horizontal coordinate; y coordinate is defined as the 1/3 value of each image's vertical coordinate from bottom to top.

6. Blend the images

As mentioned before, both the original images and the over-calibrated images are used to generate panorama. Due to the reason that our images are taken with the similar camera orientation, so the **2D affine geometric transform method** is used here to initialize all the transforms to the identity matrix.

In figure 18 (or figure 19), it is obvious that **the building part is blended accurately**. However, **mismatch happens at the road and crosswalk part.** The main reason could because **the algorithm we used to create panorama is a 2D spatial method**, and it is hard to blend the 3D area in the images accurately. For example, the building part and truck part are 2D surface area, but the crosswalk isn't.

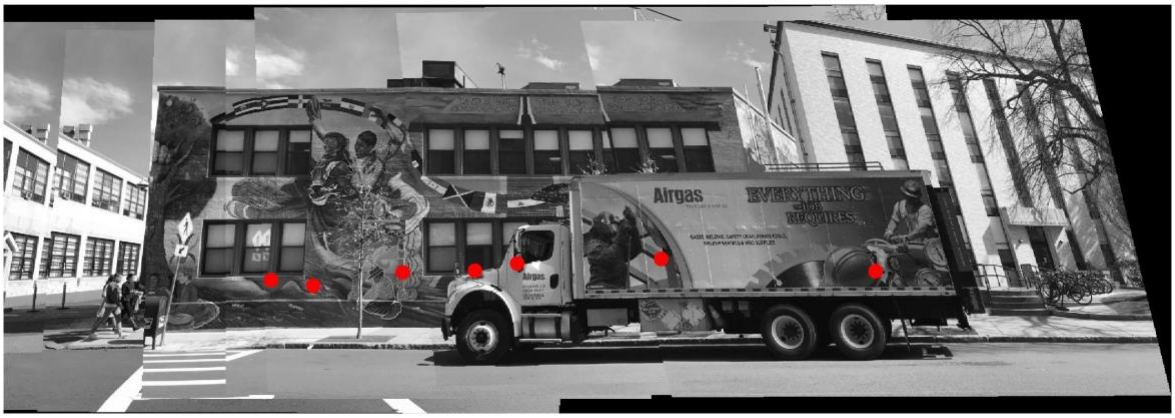


Figure 18: Panoramic mosaic using original images (before calibration)

Figure 19: Panoramic mosaic using undistorted images (after calibration)

[**for reference only**]

Part 3 – Mosaicking with varying overlap

1. Cinder block wall images with 50% overlap

Figure 20: Multiple overlapping images of cinder block wall

Figure 20 shows six overlapping cinder block wall images taken with a similar orientation. **There are some regular and repetitive patterns on the walls.**

Initially, the parameter *maximum number of interest points is set to 1000* and *tile is [15, 15]*. Figure 21 shows two samples of Harris feature detector result. **Most of the features are located at the seam and corners of the block. However, due to the similarity and insufficiency of feature points, the algorithm doesn't work well as before (shown in figure 22).**

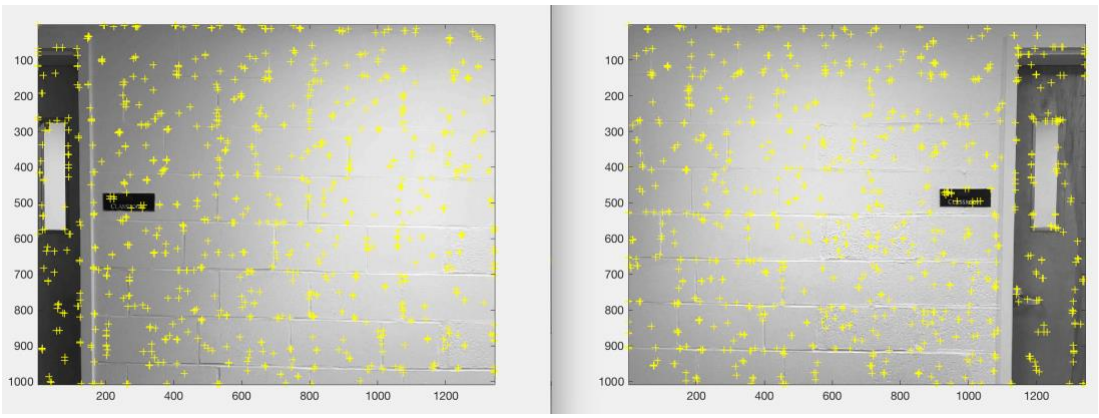


Figure 21: Sample of Harris feature detector result

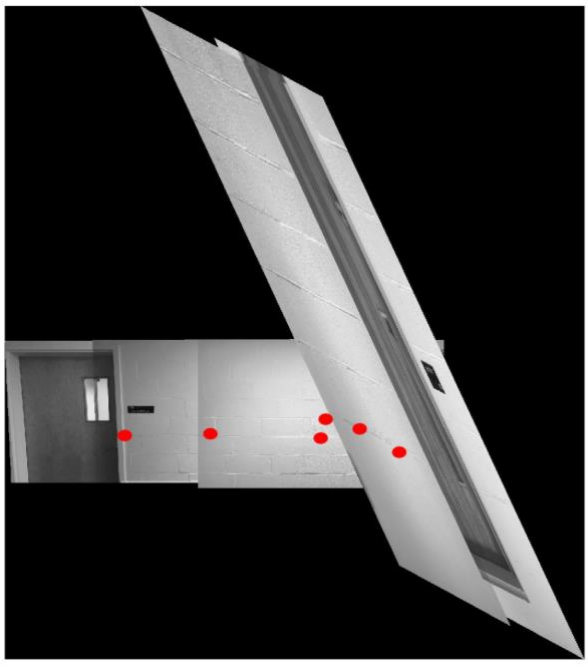


Figure 22: Panoramic mosaic

To get more feature points, the parameter *maximum number of interest points is set to 1500* and *tile is [15, 15]*. Figure 23 shows two samples of Harris feature detector result. **It's worth noting that most of the features are located at the seam and corners of the block.**

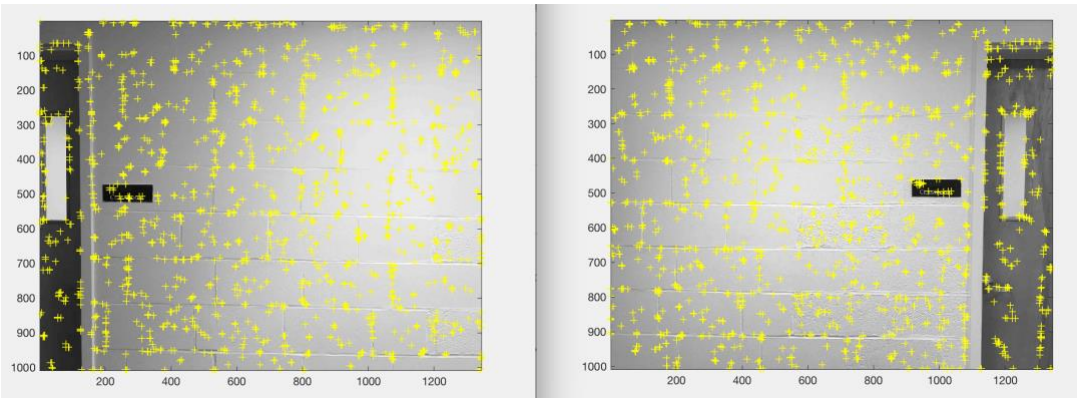


Figure 23: Sample of Harris feature detector result

Figure 24 shows the panoramic image. **We can see the algorithm works well after increasing the number of feature points. Figure 24 also shows the comparison between the panoramic mosaic relative to the first image and relative to the center image.** It is easy to see that **the lower panorama image is of better aesthetically pleasing** than the upper one. Because the upper one tends to distort most of the images that form the panorama.

Figure 24: Panoramic mosaic relative to the first image (upper) vs. for the center image (lower)

2. Graffiti art images with 15% overlap

Figure 25: Multiple overlapping images of RUGGLES graffiti art

Figure 25 shows six overlapping mural images taken with similar orientation at Ruggles Station. **The overlap of these six image is considerably small than before.**

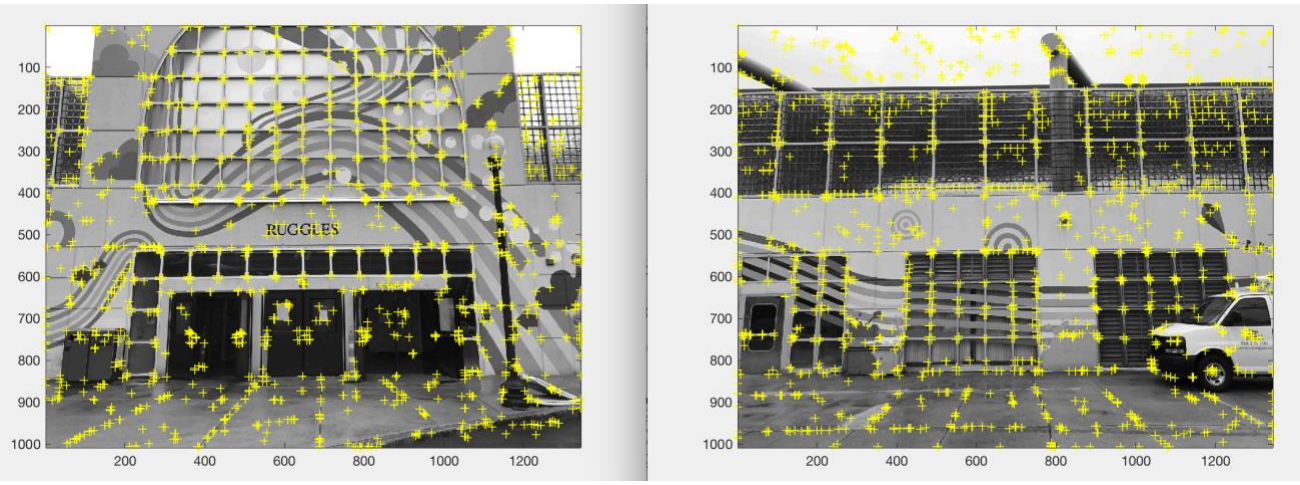


Figure 26: Sample of Harris feature detector result

As shown in figure 26, **most of the detected features are located at the right-angle corners of the square pattern.**

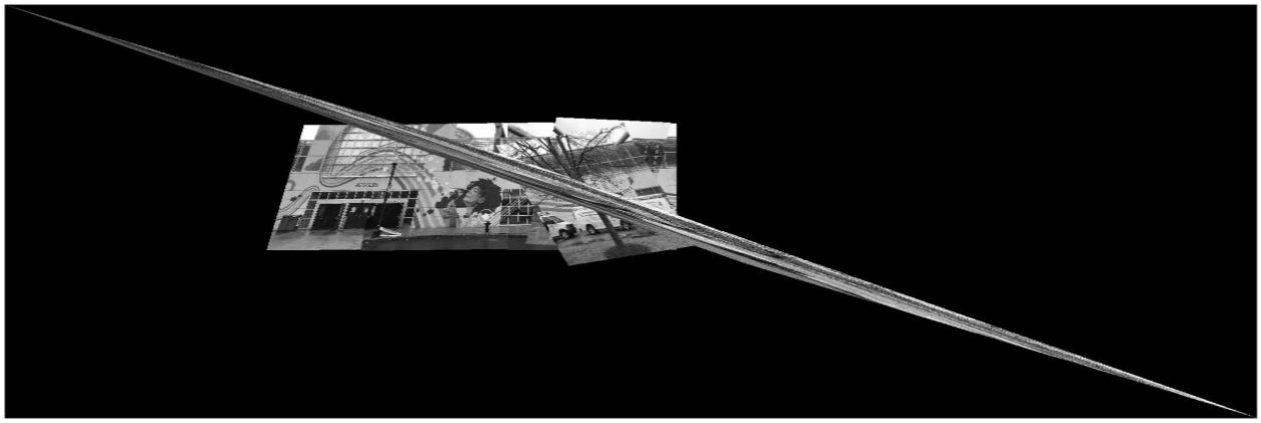


Figure 27: Panoramic mosaic using original images (before calibration)

Due to the reason that the images have only 15% overlap, it shows a bad result of the mosaicking algorithm as shown in figure 27. Because **many feature points are in the upper part of the image, which is all the repeated square pattern, it is hard to blend the images with proper order.**

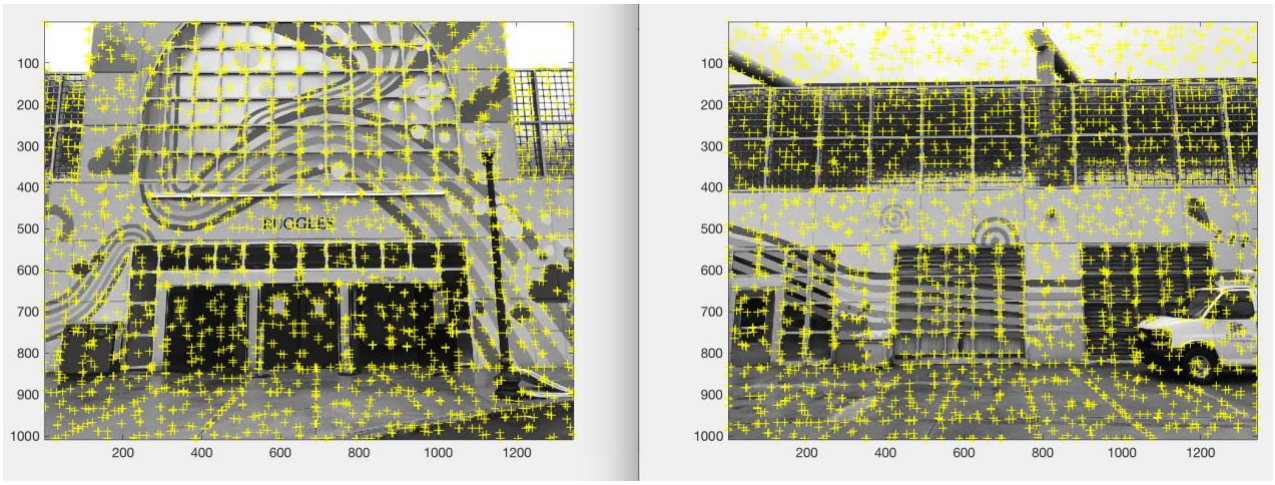


Figure 28: Sample of Harris feature detector result

To increase the number of feature point and make a more even distribution, the parameter *maximum number of interest points* is set to 3000 and *tile* is set to [35, 35]. The new Harris features detected are shown in figure 28.

Figure 29: New panoramic mosaic using original images (before calibration)

From figure 29, **we can see a better panoramic mosaic than before**. However, there is still mismatch of the last three images. **Possible reasons could be that there are three similar vans parking in front of the wall and several trees obscured the camera.**

3. ISEC Pedestrian Bridge images with 50% overlap (interesting extension)

Figure 30: Multiple overlapping images of ISEC Pedestrian Bridge

Figure 30 shows six overlapping images taken with a similar orientation at ISEC pedestrian bridge. **We can notice that all the images have almost the same pattern.**

The Harris feature detector result shows in figure 31. The parameter *maximum number of interest points is set to 3000* and *tile is [35,35]*. **Most of the features are located in the metal fence part.**

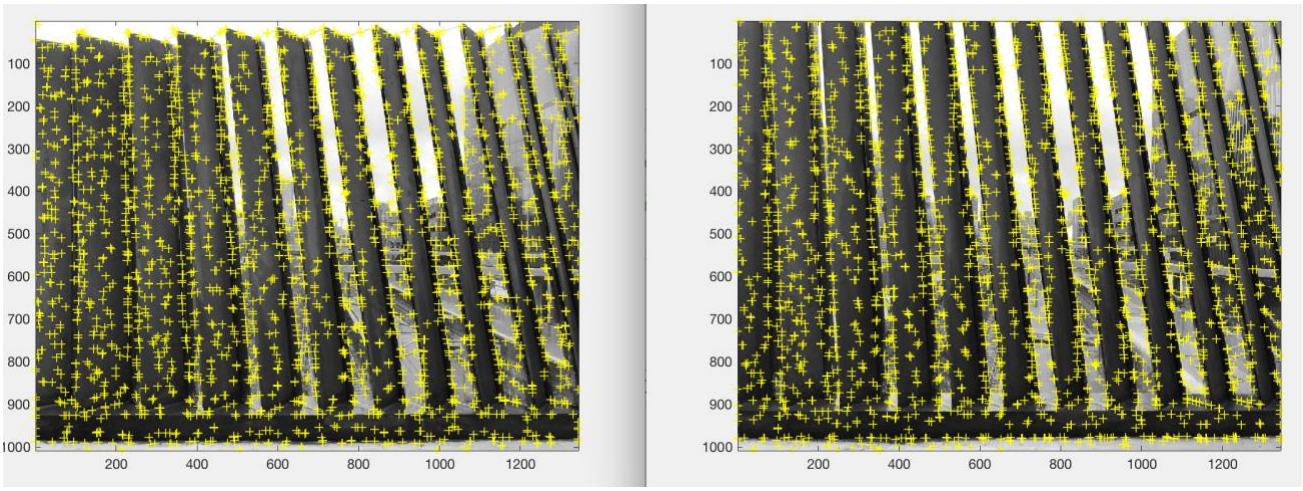


Figure 31: Sample of Harris feature detector result

Figure 32 shows the panoramic mosaic image. **The first two images are blended correctly, but the rest images are stacked up together.** Even though the *maximum number of interest points* and *tile* is set to a very big value **it still cannot work well.**

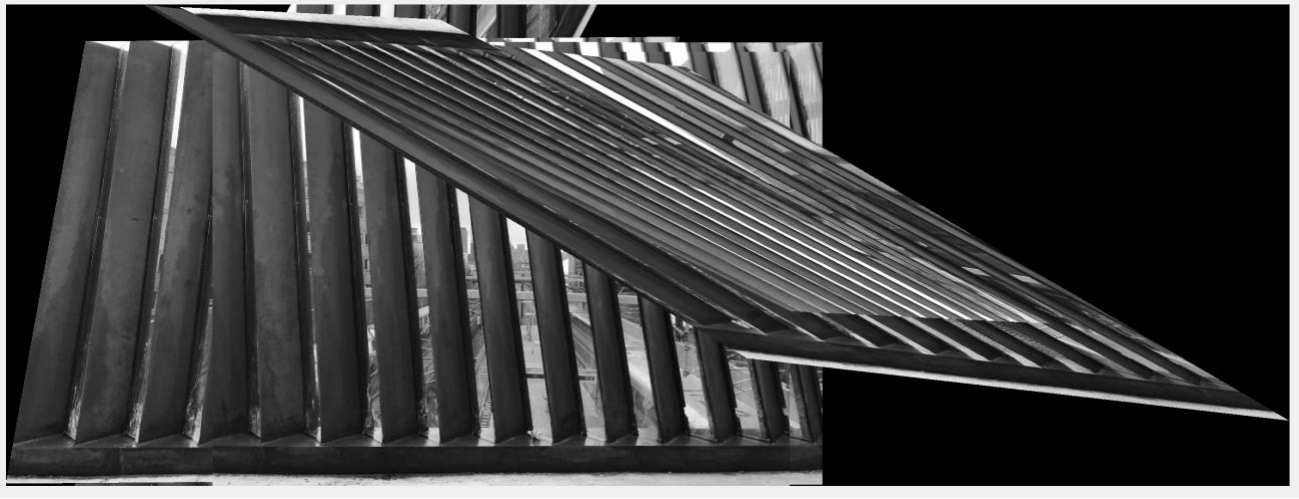


Figure 32: Panoramic mosaic using original images (before calibration)

Reference:

[1] Camera Calibration Toolbox for Matlab®, Jean-Yves Bouguet

[2] https://en.wikipedia.org/wiki/Depth_of_field

[3] https://developer.apple.com/documentation/avfoundation/avcameracalibrationdata