Lab 5 – Camera Mosaic

Immanuel Ampomah Mensah
Northeastern University

Camera Calibration

Camera calibration is an essential process that involves estimating the parameters of the camera lens and image sensor. These parameters can be used to correct lens distortion, determine the location of the camera in the scene, and measure the size of objects in world units. These tasks are particularly relevant in machine vision applications, which require accurate detection and measurement of objects.

In this lab, we employed a calibration process using a 7x9 checkerboard with 30mm x 30mm squares. A total of 20 images of the checkerboard were captured and subsequently fed into the Caltech Camera Calibration Toolbox. To estimate the calibration error, the four extreme corners of the rectangular checkerboard pattern were carefully clicked on, and the grid corners were automatically extracted using the MATLAB Camera Calibration toolbox.

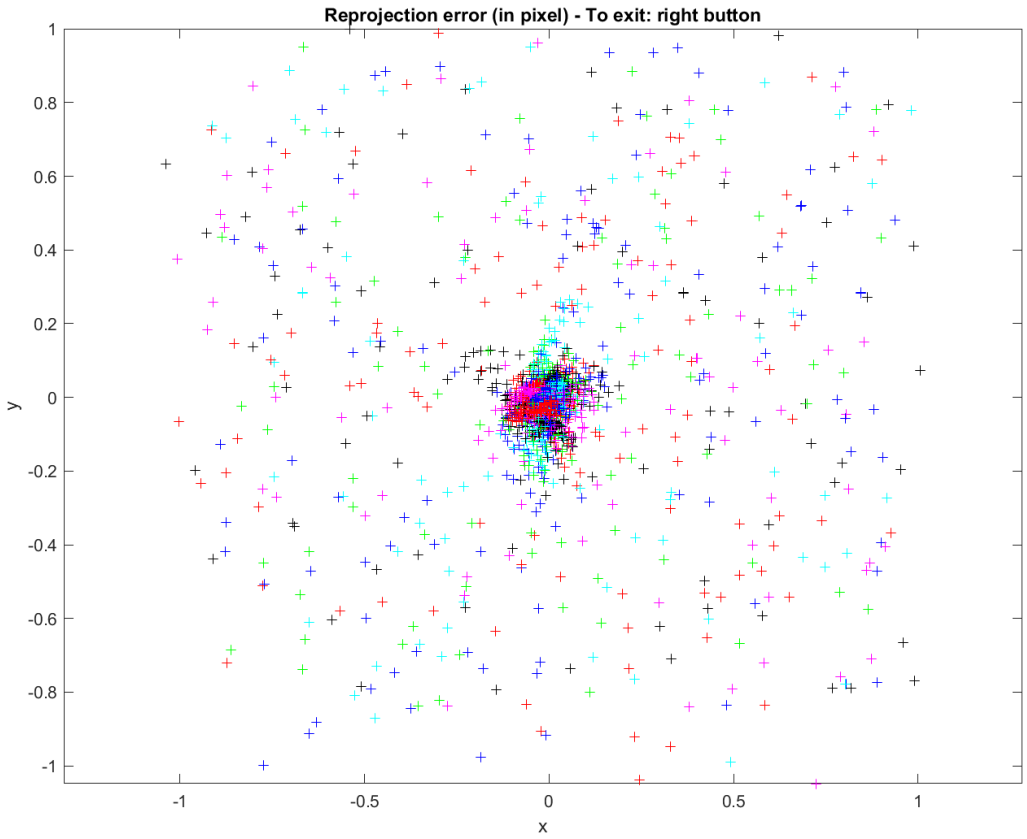

Fig. 1 Reprojection Error

Figure 1 above shows the reprojection error of the calibration images. From the figure, we can see that all the errors lie between -1 and 1 for both the x and y with most of the points clustering around [0,0]. Given that the photos were taken in 4k, this reprojection error is acceptable.

The calibration parameters used are shown below. Firstly, a 3 by 3 window size was used – after some trial and error, it because evident that this was the lowest window size that would give non-zero distortion and pixel error values while still minimizing the pixel error. The square size was also measured after printing the calibration pattern and it came out to be approximately 30mm by 30mm.

```
Re-extraction of the grid corners on the images (after first calibration)
Window size for corner finder (wintx and winty):
wintx ([] = 5) = .5
winty ([] = 5) = .5
Window size = 3x3
Number(s) of image(s) to process ([] = all images) =
Use the projection of 3D grid or manual click ([]=auto, other=manual):
Processing image 1...2...3...4...5...6...7...8...9...10...11...12...13...14...15...16...17...18...19...20...
done

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

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


Calibration results after optimization (with uncertainties):

Focal Length:          fc = [ 3124.76936   3124.18502 ] +/- [ 2.55072   2.52139 ]
Principal point:       cc = [ 1491.40158   1987.11037 ] +/- [ 1.10969   1.42641 ]
Skew:             alpha_c = [ 0.00000 ] +/- [ 0.00000  ]   => angle of pixel axes = 90.00000 +/- 0.00000 degrees
Distortion:            kc = [ 0.06521   -0.12352   -0.00187   -0.00103  0.00000 ] +/- [ 0.00136   0.00426   0.00015   0.00013  0.00000 ]
Pixel error:          err = [ 0.30081   0.29296 ]

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

Fig 2. Calibration Parameters and Results

As shown in Figure 2, the calibration parameters and results are presented. Of particular interest are the distortion coefficients (kc) and pixel errors (err), as they serve as the primary indicators of calibration quality. The obtained pixel errors were [0.30081, 0.29296], which are considered reasonable values for 4k images. These results suggest that the calibration process was successful and can be used to accurately correct lens distortion in future image processing.

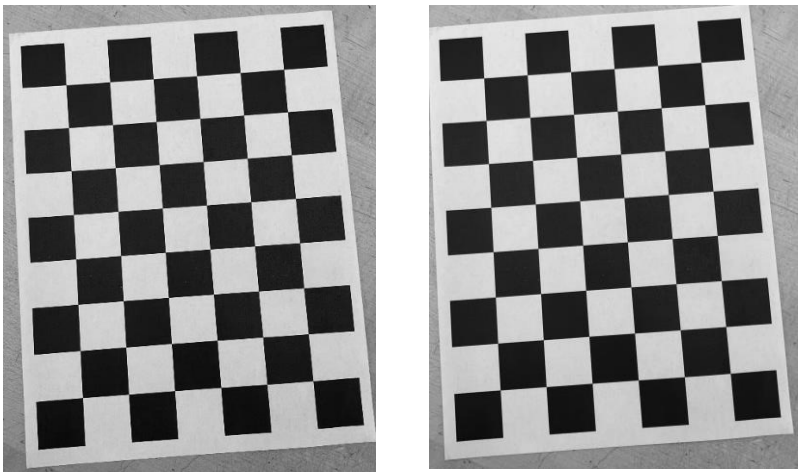

Fig. 3 Original Image (Left) and Undistorted Image (Right)

The results of undistorting an image are presented in Figure 3. The visual inspection of the images suggests that undistorting the image leads to a slight increase in quality. However, the difference between the images is minimal in our case, likely due to the decent calibration already provided by modern phones.

LSC Mosaic

Mosaic algorithms detect distinctive features in multiple images, match them together, and align them using geometric transformations to create a seamless panorama. We utilized the Harris Feature detection algorithm which is a method for identifying corners and edges in an image by looking for areas with high variation in intensity in all directions. This works by calculating the change in intensity for small shifts in all directions and then selecting points with high variation as features.

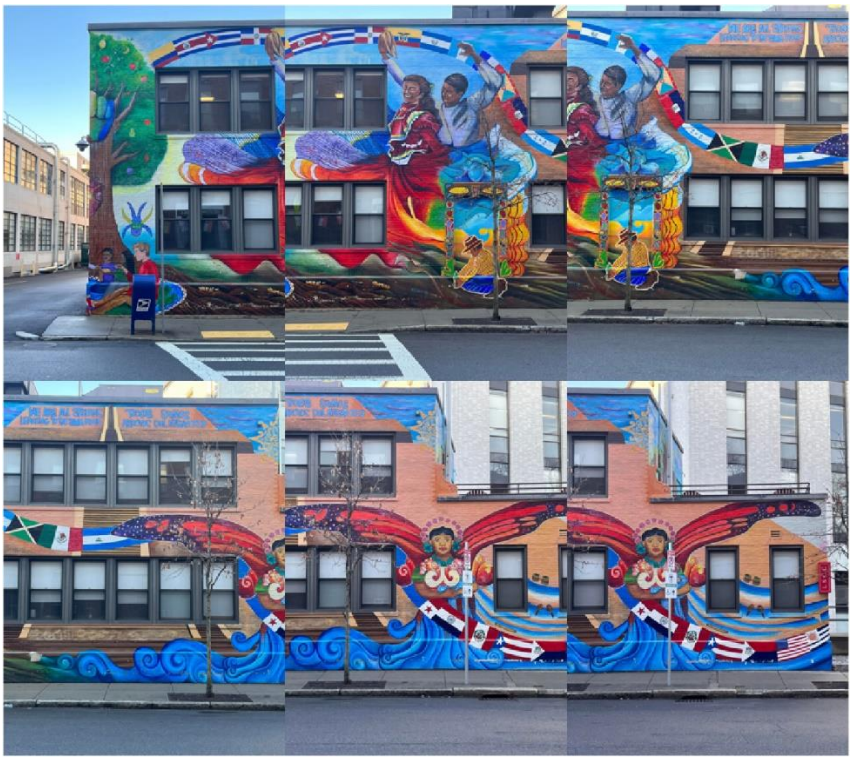

Fig 4. LSC Image Set

Figure 4 shows the image set used for this section of the lab. The images were undistorted using the calibration from the camera calibration process described earlier. The original images were also used for comparison and there was minimal difference in the result.

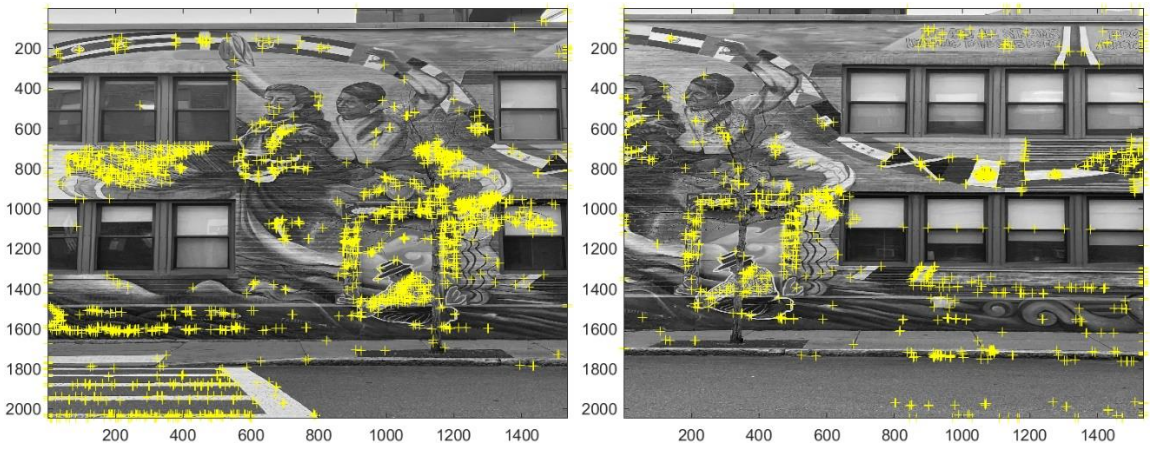

Fig 5. Harris Feature Detection samples of LSC Building Art

The distribution of Harris corners is shown above for the Latin Student Center building. In both images, we can see a somewhat even distribution of feature detection across the image. The number of interest points was set to 4000 for the Harris detection. For the transformation estimation, 99.9% confidence and 2000 'MaxNumTrials' were chosen because this combination of values resulted in the best results. We would ideally want a high number of trials to improve the results but doing so hurts the performance of the algorithm due to the limited memory and computing power. Confidence sets the confidence level of the estimation, and 'MaxNumTrials' sets the maximum number of iterations used in the estimation process. By setting these parameters appropriately, the estimation process can be made more robust to outliers and noise in the feature point matching.

Fig. 6 LSC Panorama

The stitching results obtained from the modified code are presented in Figure 6. The code was adapted from the original MATLAB example by replacing the ‘buildingDir’ with the folder directory of the images and altering the feature detection and extraction process. Specifically, the original code used the SURF feature detection method, while the modified code utilized the Harris feature detector. The Harris detector was applied using a tile size of [2 2] and a maximum number of detected features of 4000, and feature extraction was performed using ‘[features, points] = extractFeatures(grayImage,[x,y]);’.

Brick Wall

To evaluate the performance of the stitching algorithm further, the same approach was applied to photos of a brick wall with an overlap of approximately 50%. After testing several parameter values, the same set of parameters as the LSC Mosaic example was chosen for the best performance. However, compared to the LSC Mosaic, producing a clear and complete stitch with all the images reflected in the result proved challenging. Attempts were made to address this challenge by increasing the number of detected features of the Harris detector and the 'MaxNumTrials' parameter, but this resulted in either memory issues or considerable lag time. In one instance, the process took up to 20 minutes before being halted. The difficulties encountered in generating a successful stitch for the brick wall images highlight the importance of selecting appropriate parameter values and the limitations of the stitching algorithm when dealing with certain types of images.

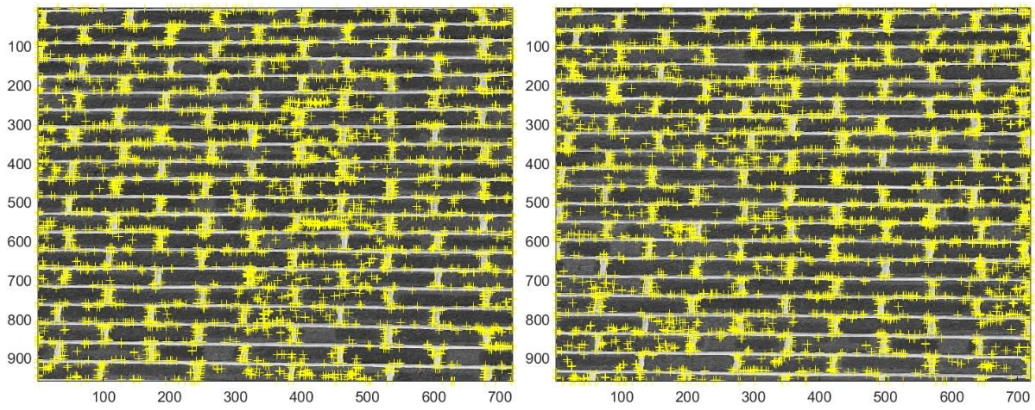

Fig 7. Harris Feature Detection samples of Brick Wall

Figure 7 above shows sample feature detection for 4000 features. At a glance, it is clear that there is very little distinction between the features detected in both images hence reducing the performance.

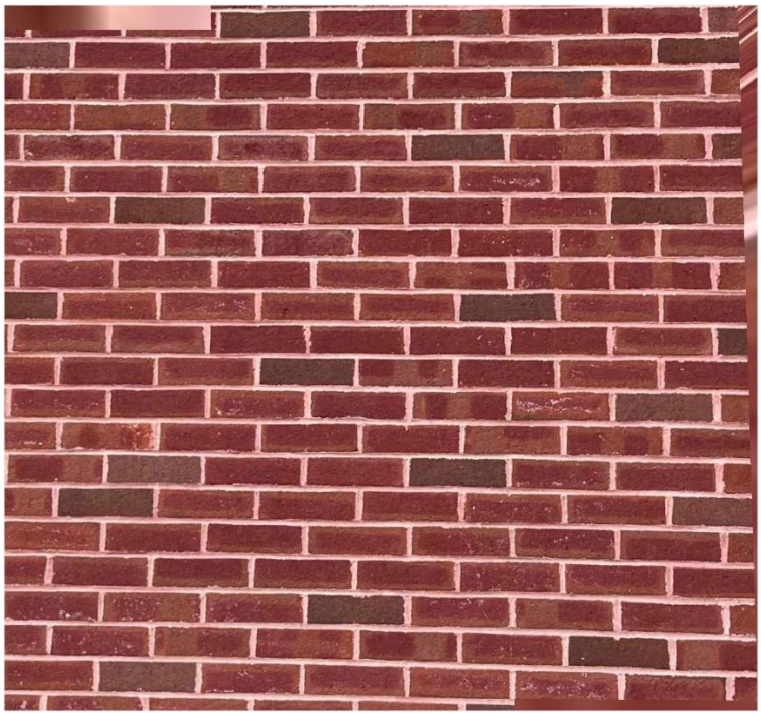

Fig 8. Best Brick Panorama

The algorithm's ability to detect and match distinctive features across images is critical for generating seamless composites. However, a brick wall may pose a challenge due to its repetitive pattern, and limited color variance. This limits the number of distinctive features available for the algorithm to work with, resulting in inferior mosaic quality. On the other hand, a colorful wall mural would provide more distinct features, such as unique patterns, shapes, and colors, that the algorithm can leverage to create more accurate feature matches. Therefore, the success of a photo mosaic algorithm in generating high-quality mosaics is significantly influenced by the distinctiveness and number of features in the images.

Third Mosaic

Next, we take a look at the effect of overlapping on the performance of the stitching algorithm. Figures 8 and 9 show the feature detection for both sets of images. In order to obtain acceptable end results, the feature detection limit was set to 50000 and 8000 for the 15% and 50% overlap respectively.

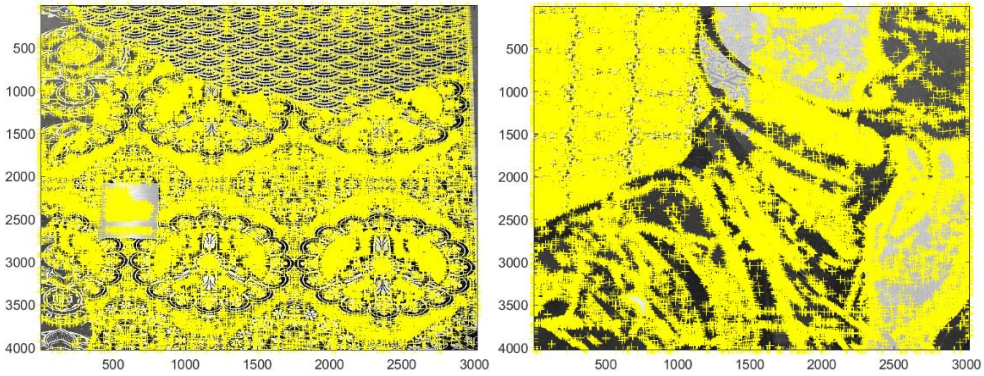

Fig 8. 15% Feature Detection

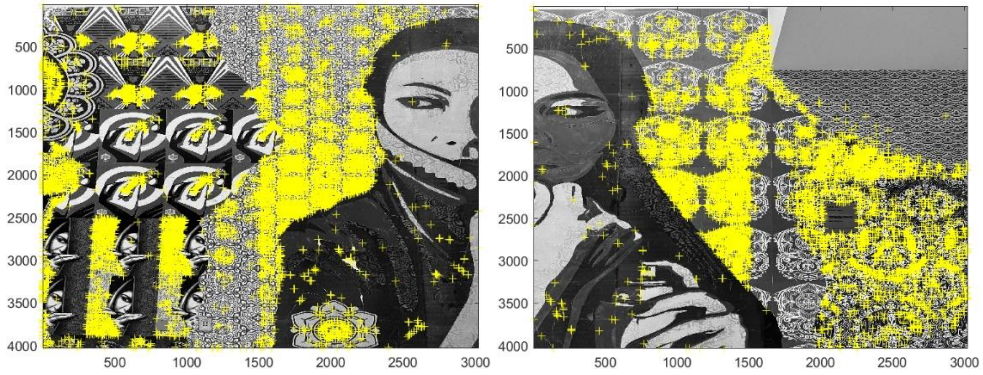

Fig 9. 50% Feature Detection

From the results shown in Figures 10 and 11, the overlap percentage is a significant parameter in image stitching. A higher overlap percentage results in an improved and complete stitching result. This is because a larger overlap provides more common features between adjacent images, which makes it easier for the stitching algorithm to match and align them.

Fig 10. 15% Panorama

Fig 11. 50% Panorama

However, an increase in the overlap percentage also increases the computational time and memory required to perform the stitching process. A higher overlap percentage means more images to stitch and more feature points to match, which can slow down the algorithm and require more memory.

Conclusion

The Harris feature detection algorithm is a widely used technique for detecting and matching features in images that is commonly applied in camera mosaicking. It works by detecting corners in an image based on their intensity changes, and then using these corners as reference points for matching features between images. Camera mosaicking is a challenging task that involves aligning and stitching together multiple images to create a panoramic or composite view. The success of camera mosaicking depends on the accuracy and efficiency of the feature detection and matching algorithms, as well as the appropriate selection of parameters such as the overlap percentage between images. While the Harris feature detection algorithm has shown to be effective in detecting and matching features in images, the quality and completeness of the resulting mosaic can vary based on the distinctiveness and number of features in the images being stitched together, and the appropriateness of the algorithm's parameter selection.