CAMERA MOSAIC

The following document aims to analysis the process of using Harris Corner Detection in order to stitch images clicked using a phone camera to create a panorama of the subject. The Phone used was a OnePlus 7 Pro. All the pictures were clicked using the main camera sensor of the phone.

Camera Calibration:

Figure 1

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

Focal Length:          fc = [ 3287.56626   3289.80254 ] ± [ 25.90979   26.17140 ]
Principal point:       cc = [ 1999.50000   1499.50000 ] ± [ 0.00000   0.00000 ]
Skew:             alpha_c = [ 0.00000 ] ± [ 0.00000  ]   => angle of pixel axes = 90.00000 ± 0.00000 degrees
Distortion:            kc = [ -0.00813   0.01971   -0.00154   0.00213  0.00000 ] ± [ 0.01008   0.02012   0.00084   0.00083  0.00000
Pixel error:          err = [ 3.66899   4.70691 ]

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

For the Calibration, I used 25 images clicked from different angles and focal lengths. Initially I started to manually run the calibration process by extracting the corners. The images used for the calibration had a dimension of 4000 x 3000 pixels. The error obtained using this process was very high. As seen in figure 1. After that I used the Auto compute feature to automatically detect and compute the corners in the images. After doing that, the Error obtained was **[2.41934 2.72480] px.**

Figure 2

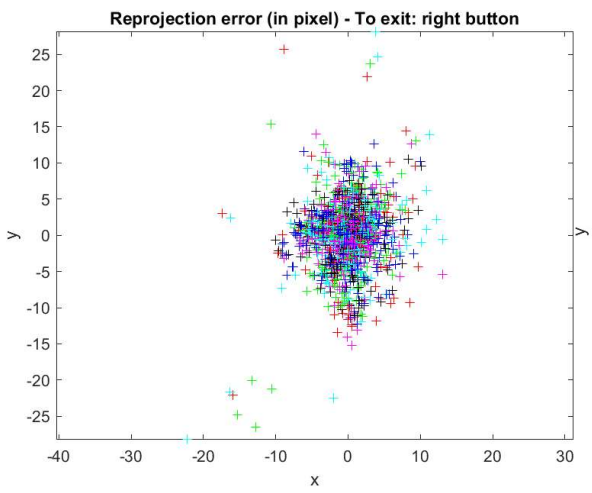
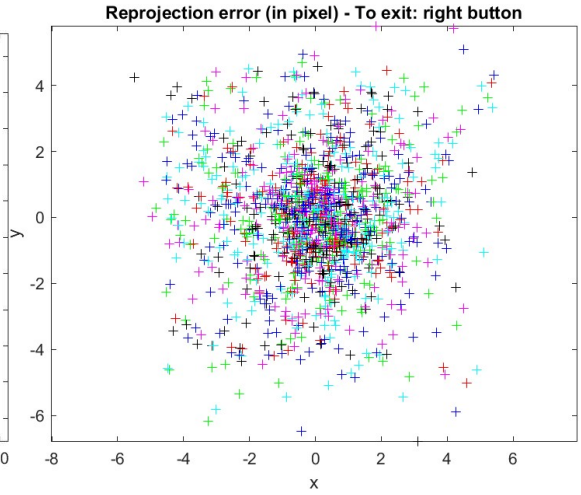


Figure 3



I used the analyse error function to analyse the error present in each image (Figure 2). After analysing the plot and taking a look at the outliers, I found out that the Images 13, 12, 4, 17 and 20 had bad calibration. In order to reduce this error and get in under 2 px in both axis, I recalibrated the images with the bad calibration. After seeing very little improvement in the overall pixel error, I decided to supress Images 17,4 and 12. After recalibration and suppression of images, the error dropped under 2 and the Error analysis is shown in Figure 3. In figure 3 there are still a few outliers, but that is okay as the overall average pixel error is under 2 px.

Figure 4

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

Focal Length:          fc = [ 3284.59516   3289.62352 ] ± [ 11.24888   11.37326 ]
Principal point:       cc = [ 1999.50000   1499.50000 ] ± [ 0.00000   0.00000 ]
Skew:             alpha_c = [ 0.00000 ] ± [ 0.00000  ]   => angle of pixel axes = 90.00000 ± 0.00000 degrees
Distortion:            kc = [ -0.00432   0.01383   -0.00145   0.00201  0.00000 ] ± [ 0.00439   0.00875   0.00037   0.00036  0.00000
Pixel error:          err = [ 1.80385   1.86850 ]

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

Figure 4 shows the final pixel error of **[1.80385 1.868850] pixels**. As the images were 4000 x 3000 pixels, a pixel error of under 2 pixels on each axis is acceptable.

Figure 5

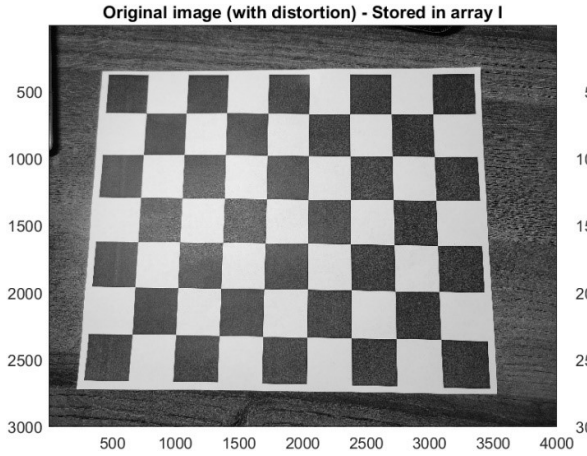


Figure 6

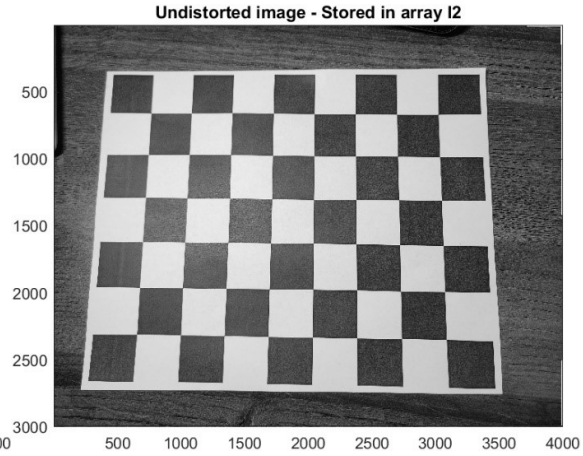


Figure 5 and Figure 6 show the original image "with distortion" and the Undistorted image using the camera parameters from camera calibration respectively. As we can see very evidently that there is very minimal difference in both the images due to the fact that modern cell phone cameras have inbuild undistorting features already present in them. Figure 7 and 8 show the final Extrinsic parameters and final world view of the position of the camera.

Figure 7

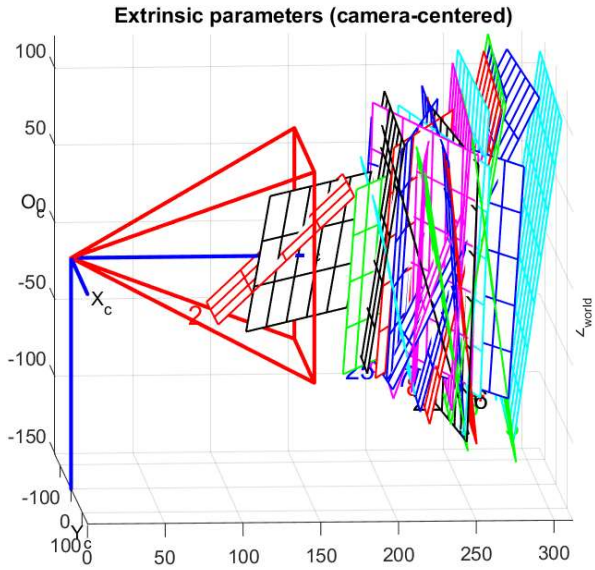


Figure 8

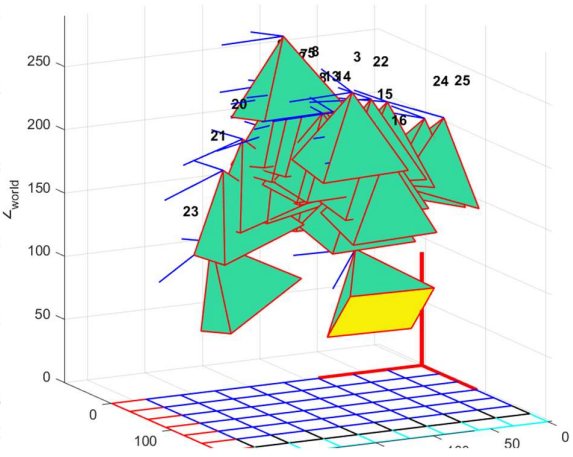


Figure 9

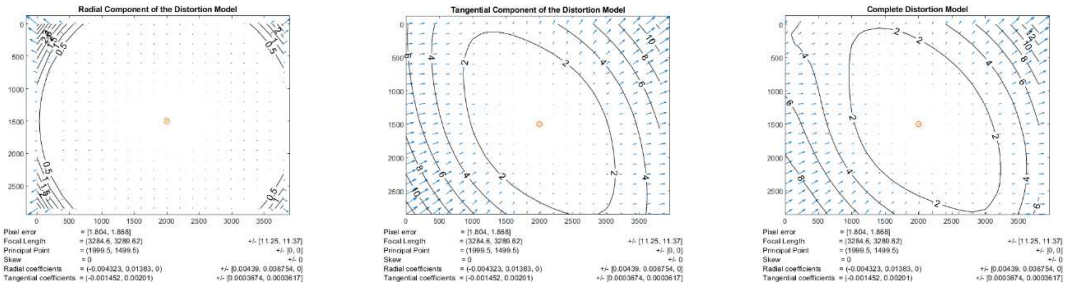


Figure 9 shows the Radial, Tangential and Compete Distortion Model present in the camera module in this cell phone.

LSC Mosaic:

Figure 10

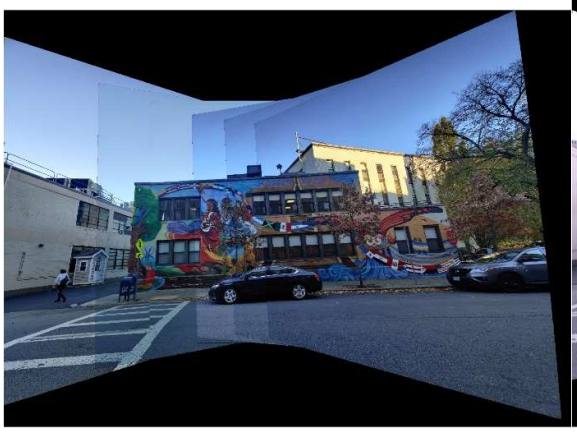


Figure 11

Figure 10 and Figure 11 show the output of using Harris corner detection. They look visually different because the pictures used in Figure 11 were clicked using the Pro mode on the cell phone.

The settings for the images were as follows:

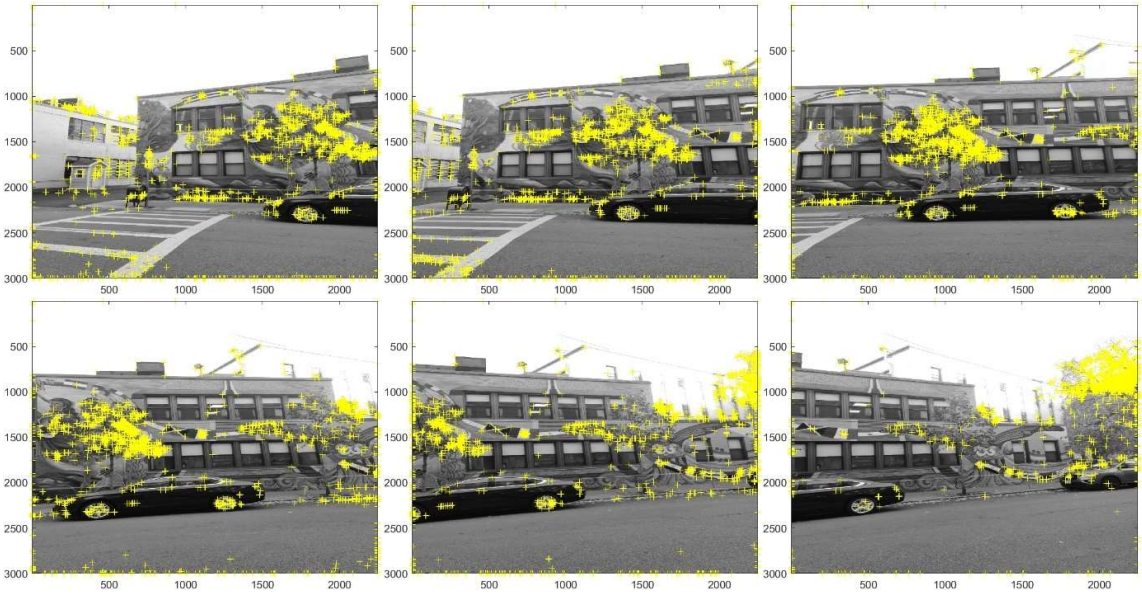
Camera Parameters	Values
Focal Length	4.75 mm
Aperture	f/1.65
ISO	100
Exposure time	1/60 s

Keeping these settings the same for all the images in the dataset helped capture images which share a similar histogram. The Images used to obtain the output shown in Figure 10 were clicked using the default camera application of the phone. Due to this the camera keeps changing the camera parameters before every photo based on various environmental parameters. Due to this we can observe a clear separation in colours in Figure 10, where as Figure 11 shows consistent colours throughout.

Figure 12

Figure 12 shows the different images used to obtain the output shown in Figure 11. Figure 13 shows the output of the Harris corner detection implemented on the Images shown in Figure 12. In order to obtain these results, I had to play around with the input parameters of the Harris detector.

Figure 13



Before implementing the Harris Detector, the images were Undistorted using the data obtained from the camera calibration step. In order to do this, Matlab has a function called 'undistortImage' which takes the Image and camera parameters and input. To form the Intrinsic matrix from the calibration step I used the definitions shown in figure 14.

Figure 14

Intrinsic Camera Parameters:

`IntrinsicMatrix` — Projection matrix
3-by-3 identity matrix

Projection matrix, specified as a 3-by-3 identity matrix. The object uses the following format for the matrix format:

$$\begin{bmatrix} f_x & 0 & 0 \\ s & f_y & 0 \\ c_x & c_y & 1 \end{bmatrix}$$

The coordinates $[c_x, c_y]$ represent the optical center (the principal point), in pixels. When the x and y axis are exactly perpendicular, the skew parameter, s, equals 0.

$f_x = F*s_x$

$f_y = F*s_y$

F, is the focal length in world units, typically expressed in millimeters.

$[s_x, s_y]$ are the number of pixels per world unit in the x and y direction respectively.

fx and fy are expressed in pixels.

The Intrinsic Matrix formed was:

$$\begin{bmatrix} 3284.59516 & 0 & 0 \\ 0 & 3289.624 & 0 \\ 1999.5 & 1499.5 & 1 \end{bmatrix}$$

The Radial and Tangential distortion values were obtained from the distortion model shown in Figure 9.

The final parameters used for Harris Detector were:

Disp = 2000 points, tile = [1 1] window size.

Brick Wall Mosaic:

Figure 15

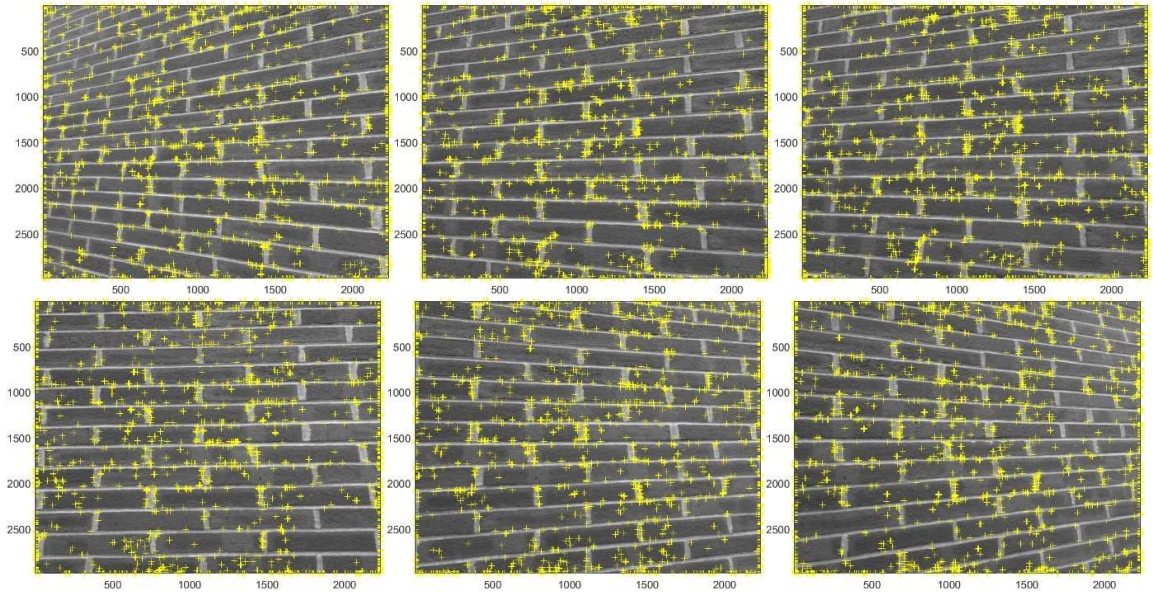


Figure 15 shows the output of the Harris corner detector implemented on the different images of a brick wall.

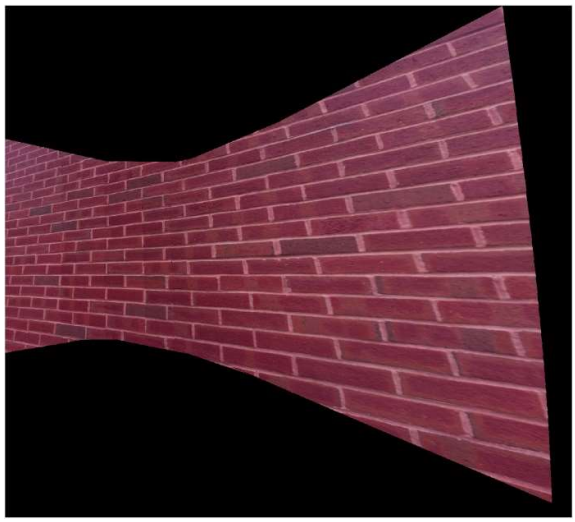
The Brick wall has a repetitive pattern and the corners detected are placed at similar distances in each of the image. How the stitching of the images works, is the software will try to match the points found in different images and stitch them together. As the points are placed at similar distances, it is difficult to find a perfect match.

After a lot of trial and error, the Parameters for Harris Detector used for this were:

Disp = 2000 points, tile = [4 4] window size.

Figure 16 shows the output of the creating a panorama of a brick wall using these parameters.

Figure 16



Third mosaic:

Figure 17

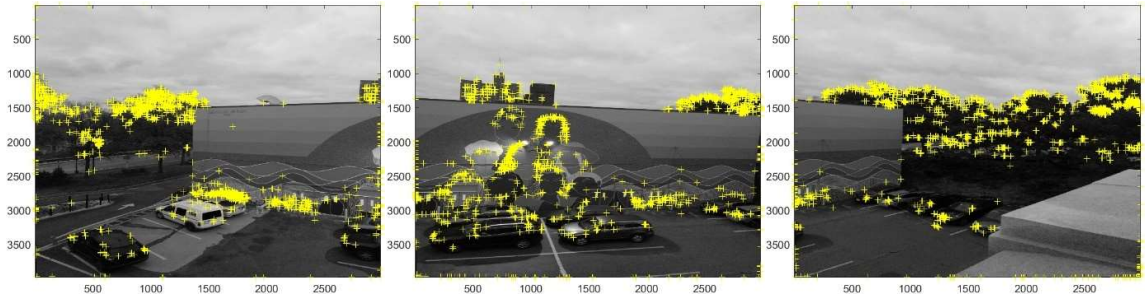


Figure 17 shows the output of the Harris Dectector implemented on images of a mural with roughly 15% overlap.

Figure 18

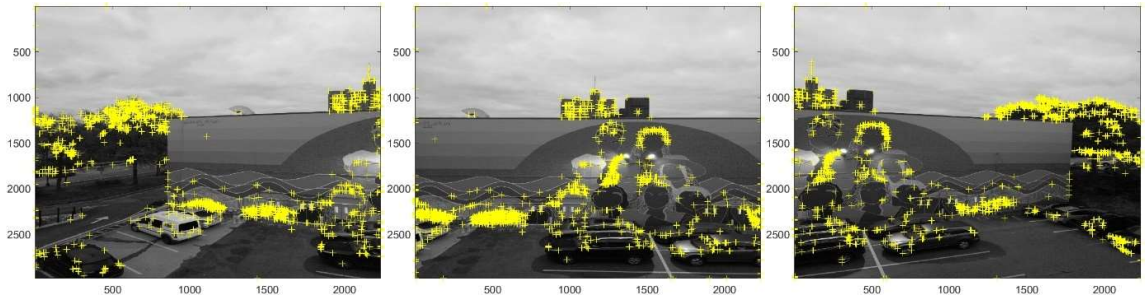


Figure 18 shows the output of the Harris Dectector implemented on images of a mural with roughly 50% overlap.

Figure 19

Figure 20

Figures 19 and 20 show the output of the creation of panorama using the Harris corner detector for 15% and 50% respectively.

We can see in Figure 19, the main mural has been detected and stitched properly, but the edges of this image have been stretched considerably. This is due to lack of overlapping points for comparison in that area. In Figure 20, we can see that the over all image is more uniform including the main mural. As there is 50% overlap in these images, there are more points that overlap and match each other's location, which helps in increasing the performance of creating accurate panoramas.

The Parameters for Harris Detector for 15% overlap used for this were:

Disp = 2000 points, tile = [1 1] window size.

Additionally, Matlab was importing these images with a 90 degree rotation in the anti-clockwise direction. To correct this I used the 'imrotate' function in Matlab to rotate the image by -90 degrees.

The Parameters for Harris Detector for 50% overlap used for this were:

Disp = 2000 points, tile = [2 2] window size.

Figure 21 shows the panorama clicked using the phone's camera application. We can see that the colors are more consistent and the stitching is also good. The middle portion of the image is blurred, due to human error in moving the phone perfectly along the required path, which is a limitation of this approach.

Figure 21