RSN Lab-5 Report

Camera Calibration and Mosaicking

Data collection for Task 3 was done on the Forsyth Street with 50% overlap.

Data collection for Task 6 was done on a brick wall at the campus with 50% overlap.

Data collection for Task 7 was done at Ruggles with 15% overlap.

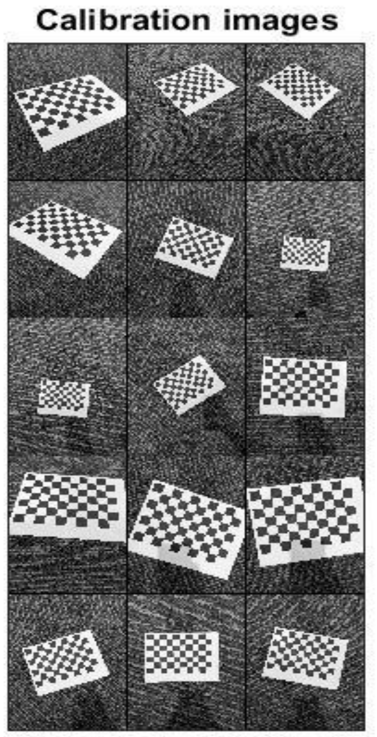
(I am using 1 Late Day for the completion of this Lab)

Part 1:

Done.

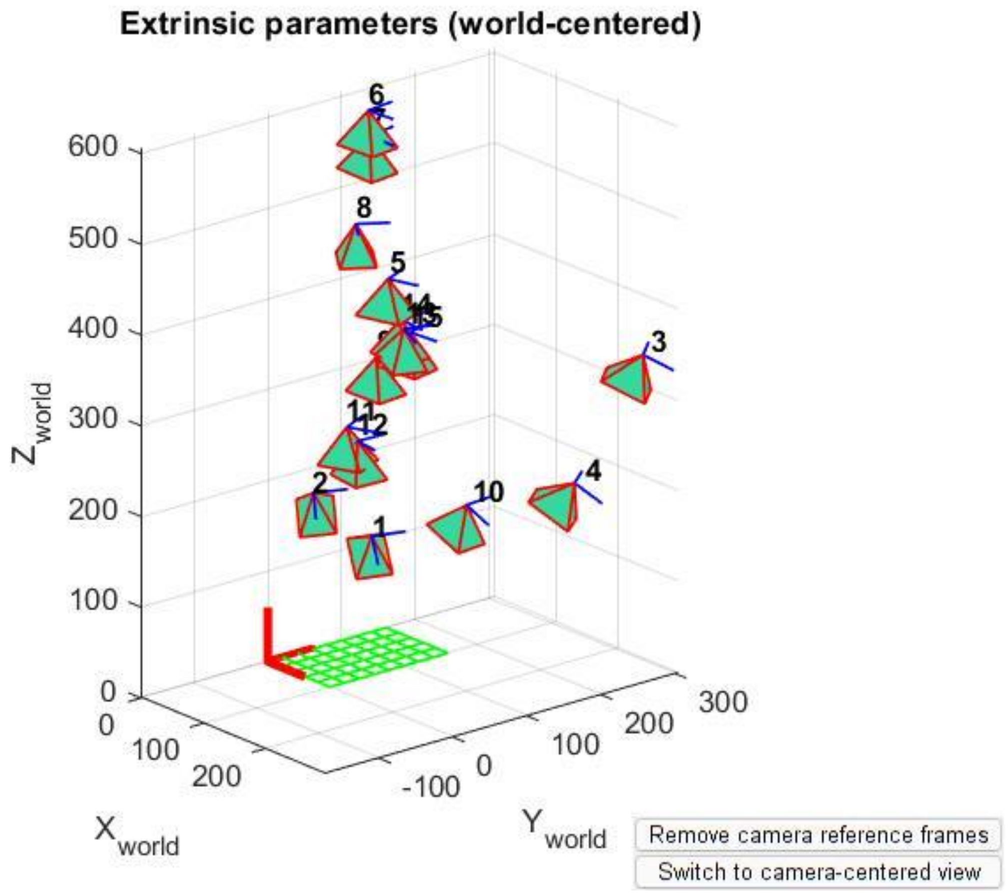
Part 2 – Camera Calibration:

Images taken for Camera Calibration:

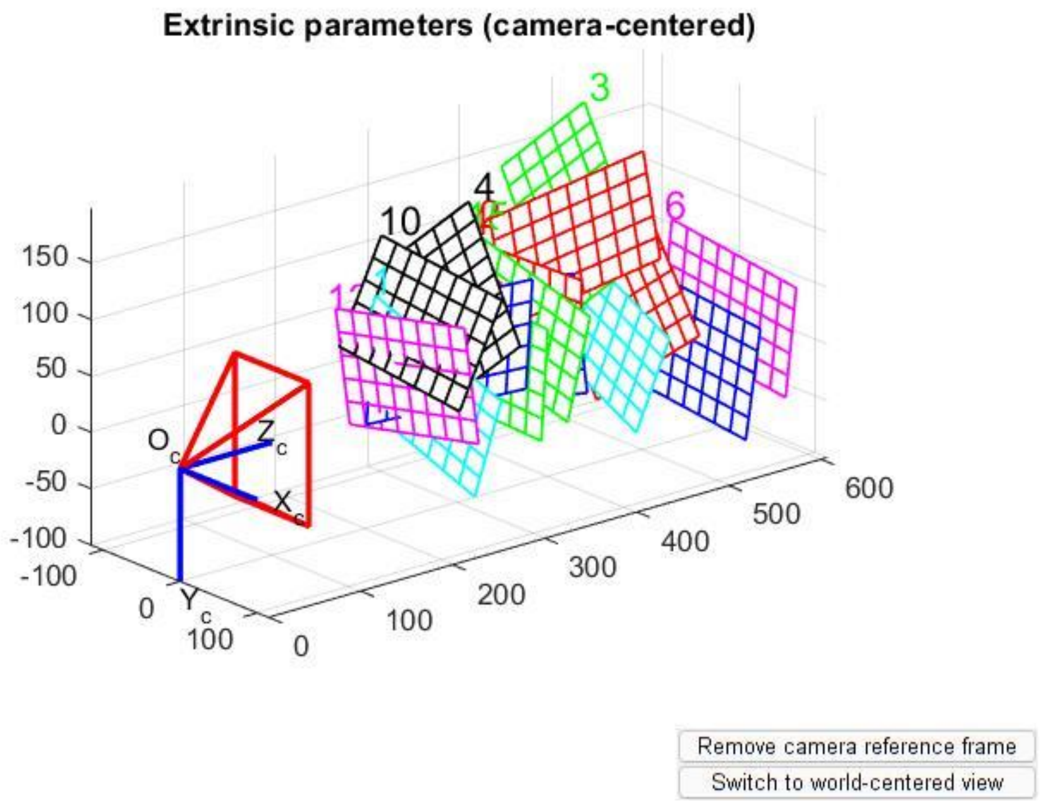


Camera Calibration was done on the set of above 15 Images of a checkerboard of size (6, 9) with the length of a square of 20mm. As we can see, the photos have been captured at different angles in different checkerboard orientations with respect to the camera.

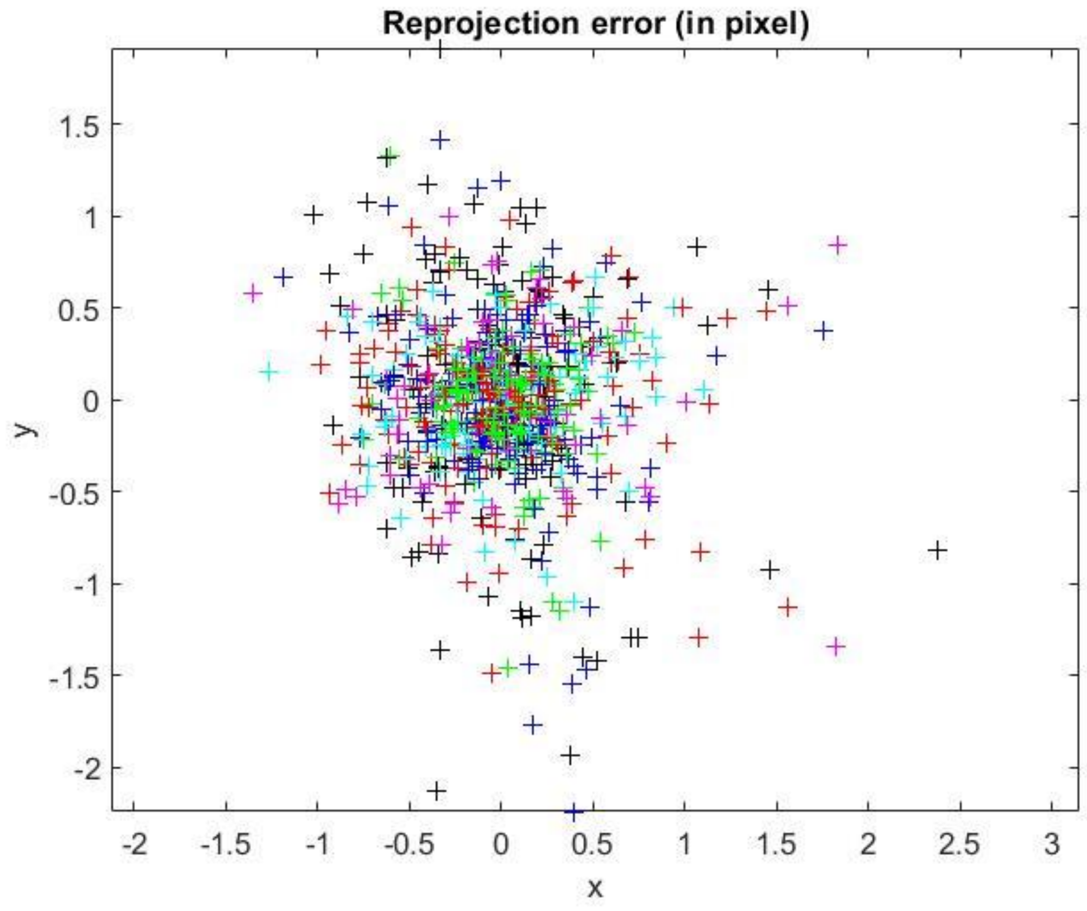
Resultant reprojection pixel error:



The images were taken in the world-centered frame which basically means that the checkerboard was placed on the ground and the camera was moved at various angles to capture wach image. We can see that the cameras are numbered as well to tell us which image was captured by placing camera at what angle approximately. But, it would be the same thing when looked from the mathematical perspective if the camera was kept stationary and the checkerboard was moved to capture the images which can be seen in the below image.



The above image shows the mathematical equivalent perspective of the data if the same images were captured by keeping the camera stationary and moving the checkerboard. Oc represents the origin of the camera frame and Xc, Yc and Zc are the camera axes.



The reprojection error was found to be Pixel error: err = [0.42075 0.45516] which is less than the length of even half of the pixel in X as well as Y axis. The above plot shows a scatter chart of error for every pixel, and we can see its concentrated near 0 which shows our calibration was successful. We can also observe there are a few outliers, but they are extremely sparse unlike the concentration we can see at the center of the image.

The errors could have been a result of not able to carefully pinpoint the location of the corners on a particular image or because of not providing an initial value for distortion coefficient.

Output of Calibration Parameters:

Calibration results (with uncertainties):

Focal Length: fc = [1578.96062 1587.68541] ± [21.93023 21.81168]

Principal point: cc = [762.26178 1042.90600] ± [7.35144 13.99538]

Skew: alpha_c = [0.00000] ± [0.00000] => angle of pixel axes = 90.00000 ± 0.00000 degrees

Distortion: kc = [0.11185 -0.32310 0.00253 0.00150 0.00000] ± [0.01279 0.05791 0.00159 0.00193 0.00000]

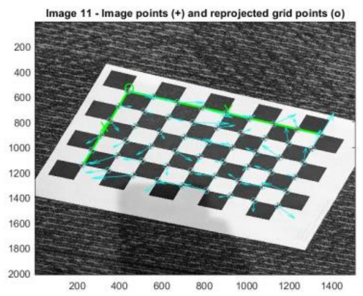
Pixel error: **err = [0.42075   0.45516]**

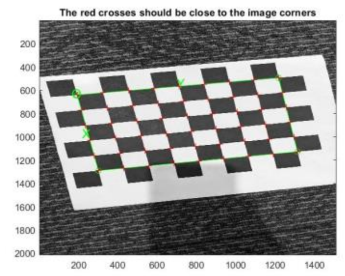
Note: The numerical errors are approximately three times the standard deviations (for reference).

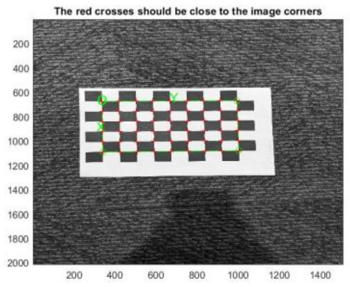
The above data (can also be found in the .mat file uploaded on gitlab) is pasted as is from the terminal of the Matlab which was the output of the camera calibration parameters once the calibration process was over. It gives us the information of the focal length of the camera, the principal point, the skew which is 0 as expected. The distortion coefficients are also negligible as expected because of the high end functionality of modern cameras.

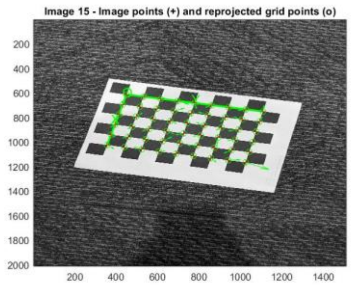
The pixel error can be observed to be lesser than the width and height of even half of the pixel.

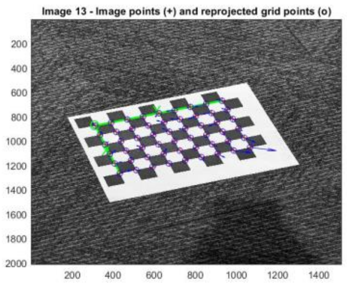
Below are a few images before and after calibration showing the extracted corners:

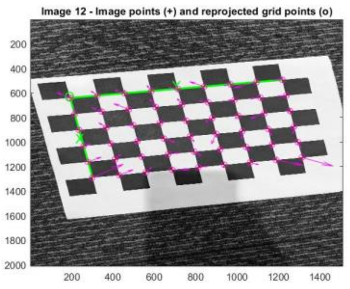












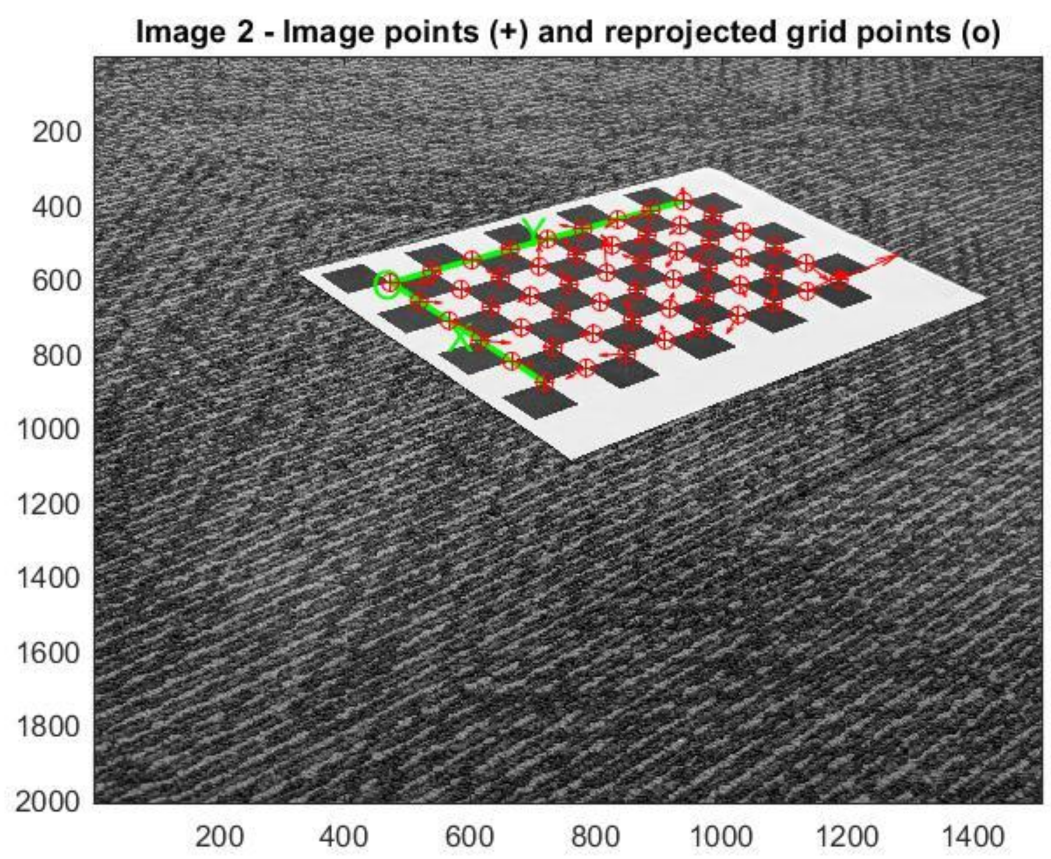
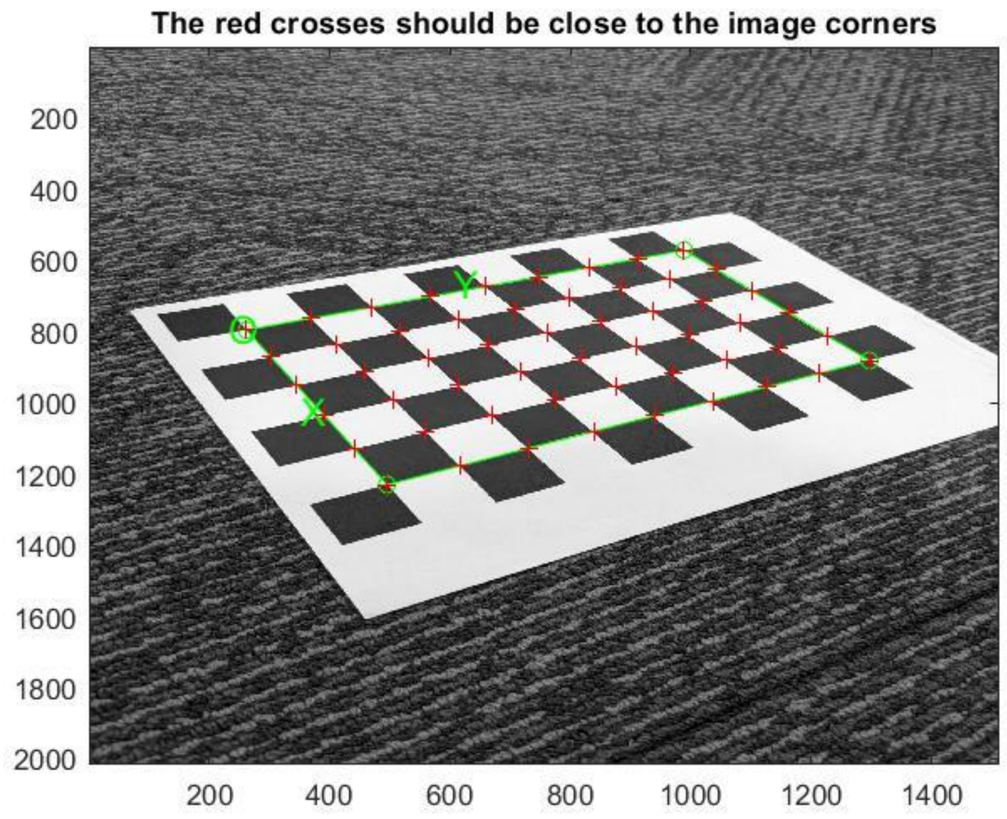
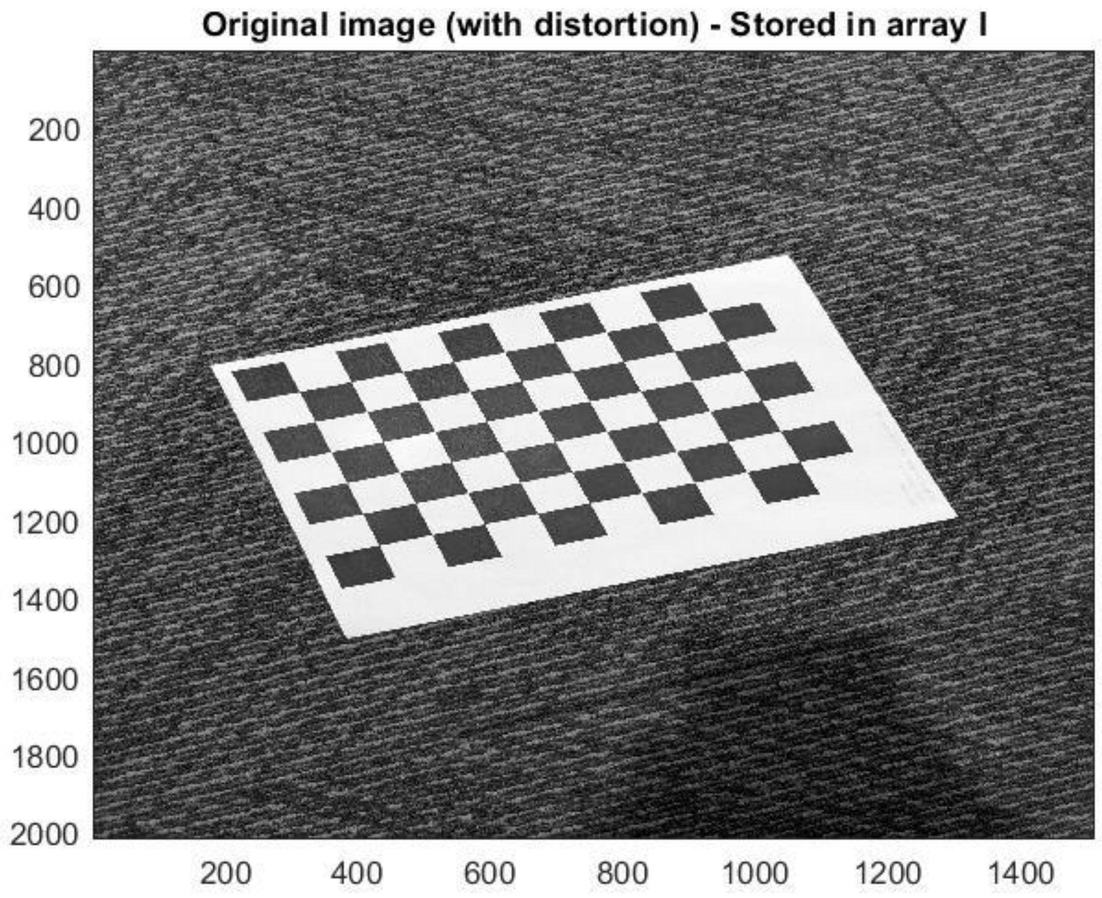


Image 2 - Image points (+) and reprojected grid points (o)
X
Y
200
400
600
800
1000
1200
1400
1600
1800
2000
200
400
600
800
1000
1200
1400

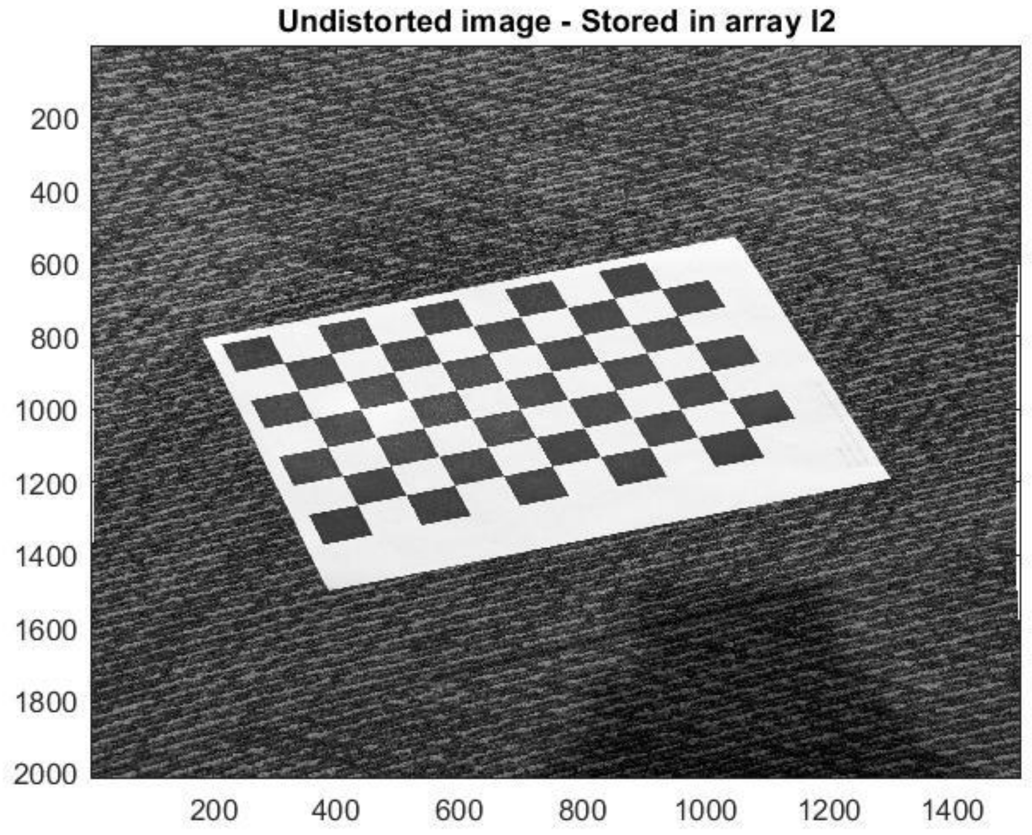


The individual transformation of every image is also stored in the Test folder uploaded to gitlab where the number corresponds to the image number.

Image before and after correction of Distortion:



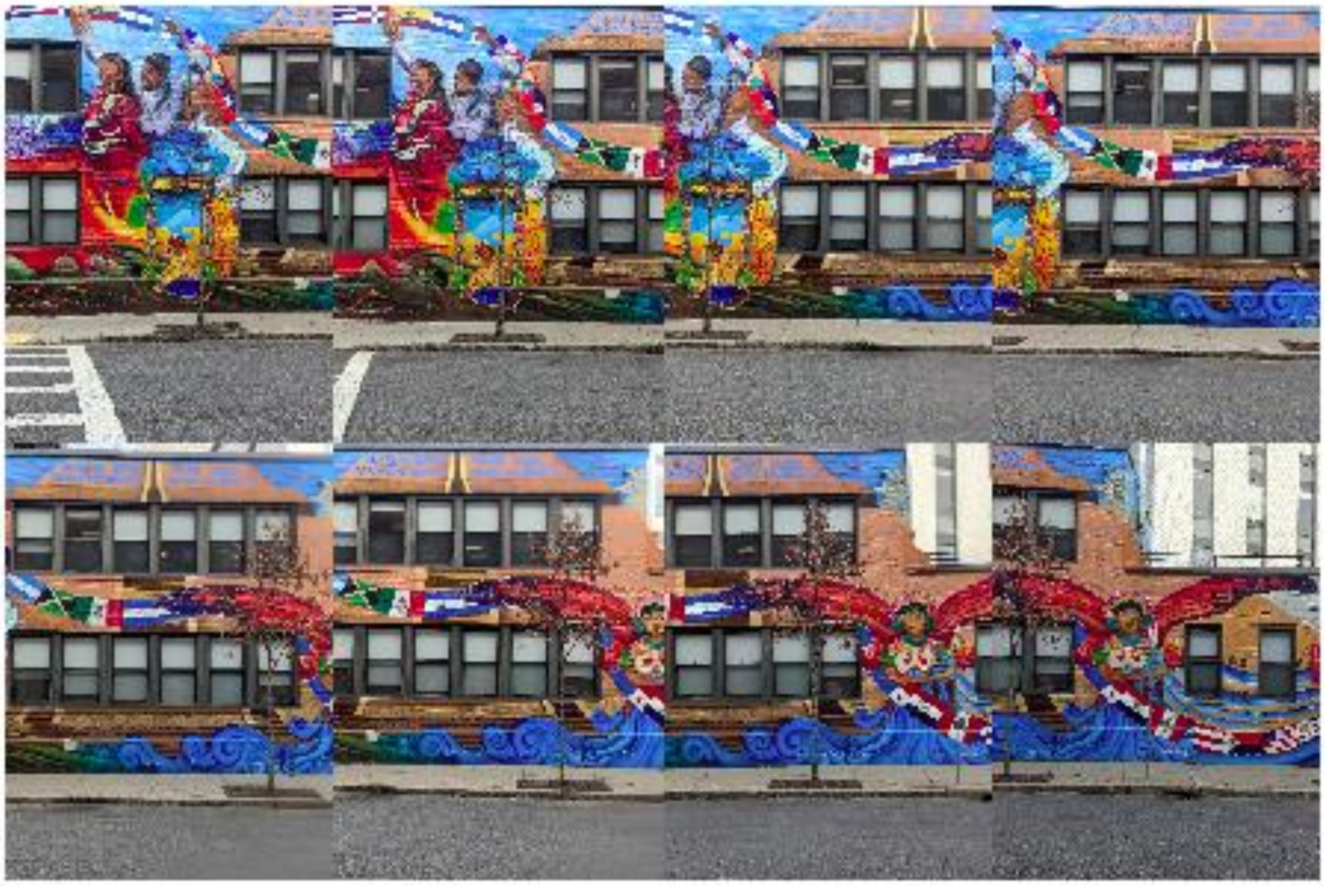
The above image is distorted as captured from the camera.



The above image is the image corrected after getting data from the calibration parameters and the distortion matrix. While it is a little difficult to notice the difference because of the cameras of today, we can see that in the corrected or undistorted image, it is compressed a little on both the center of left and right edges to correct for distortion. The left and right edges of the corrected image are curved inwards toward the center to correct for the distortion.

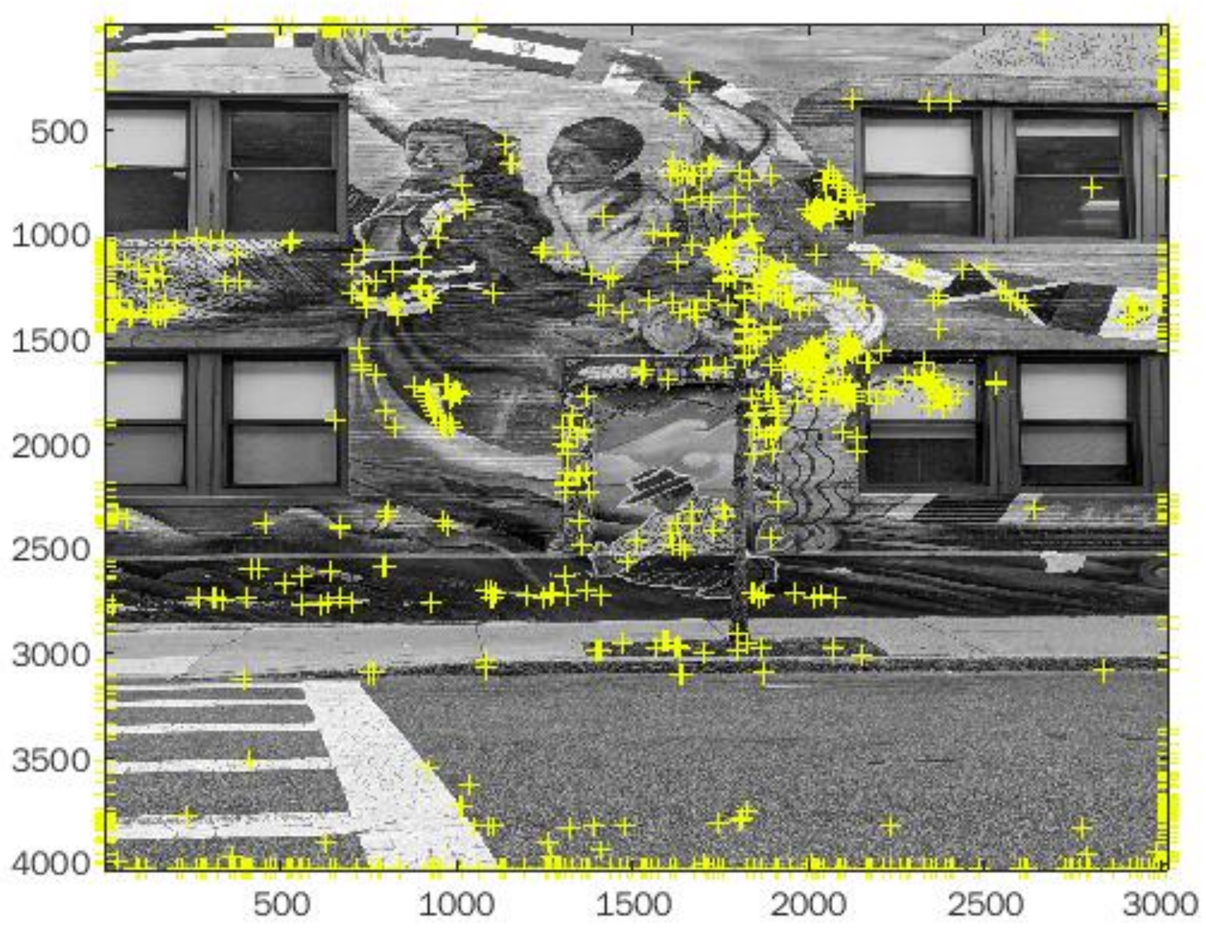
Part 3 – Data Collection:

Images to be used for Mosaic:



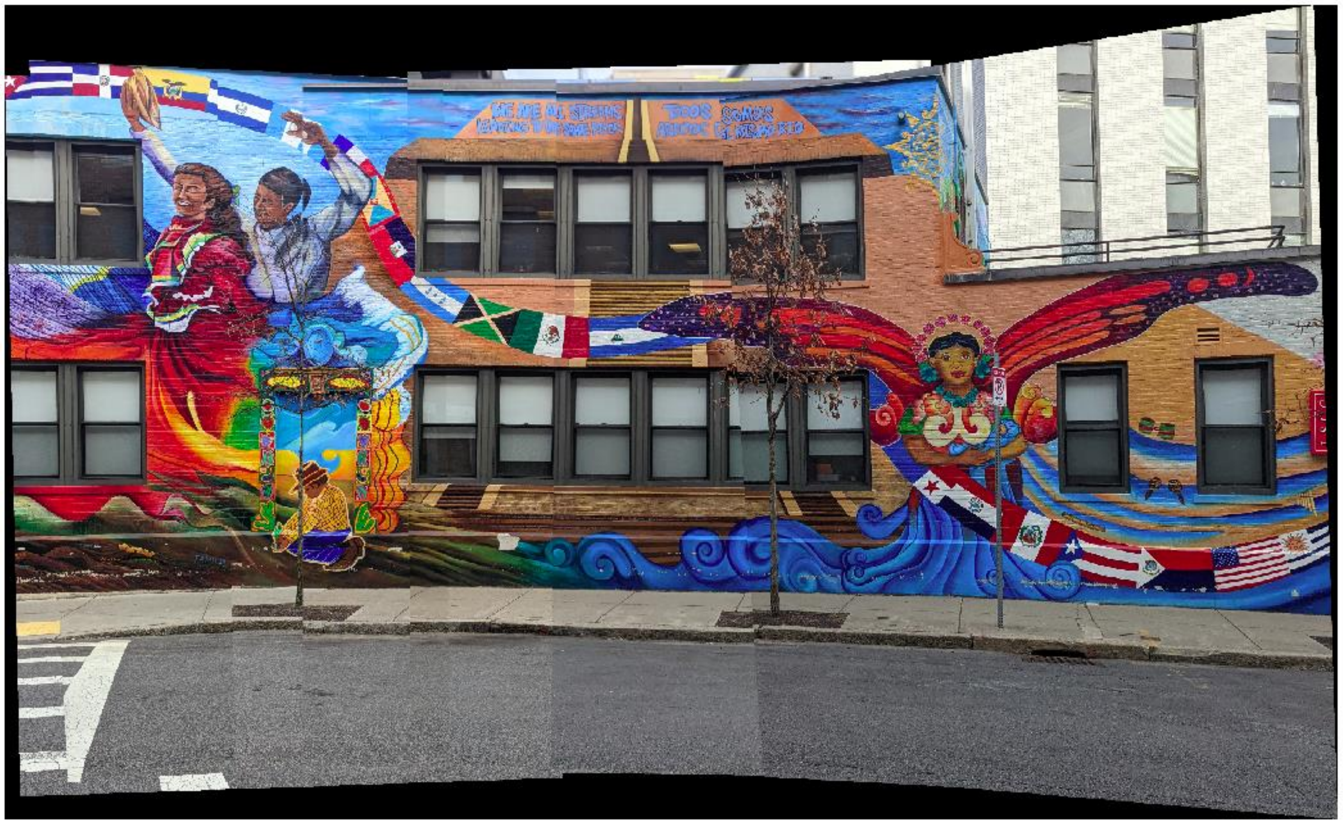
The above set of images is used to create a mosaic by stitching with the help of Harris Corners.

Part 4 – Distribution of Harris Corners:



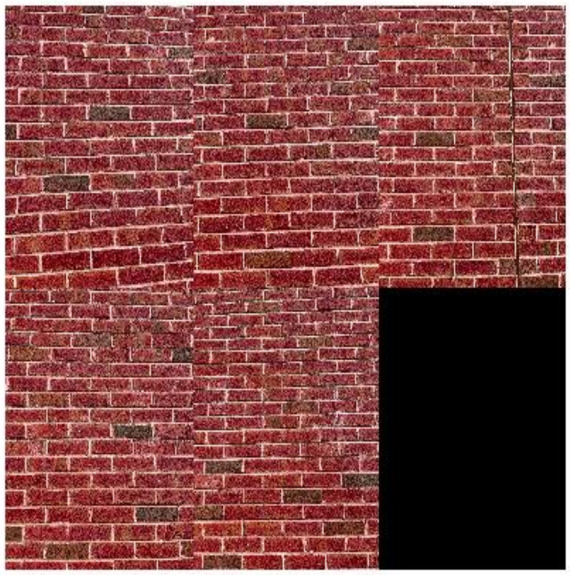
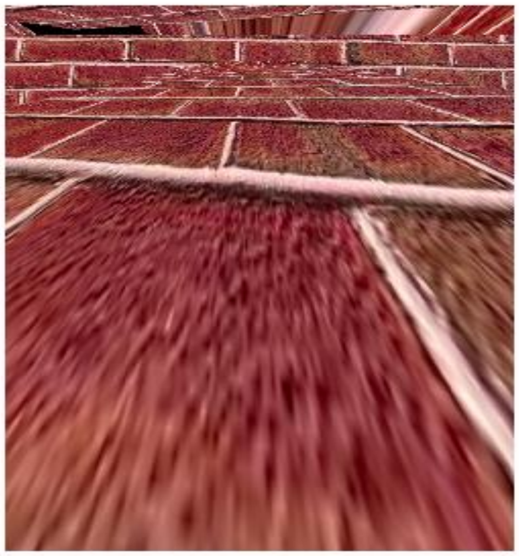
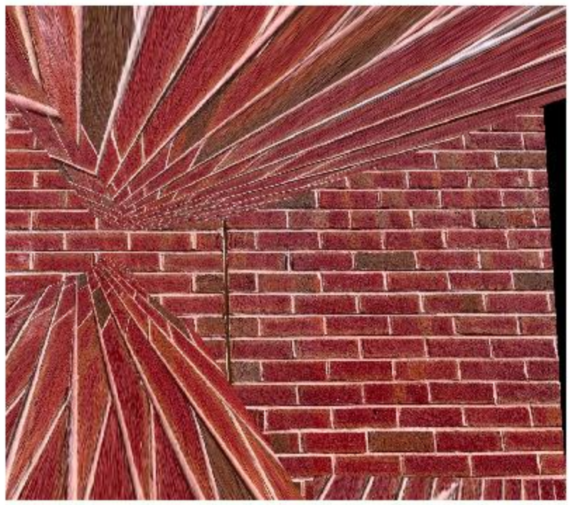
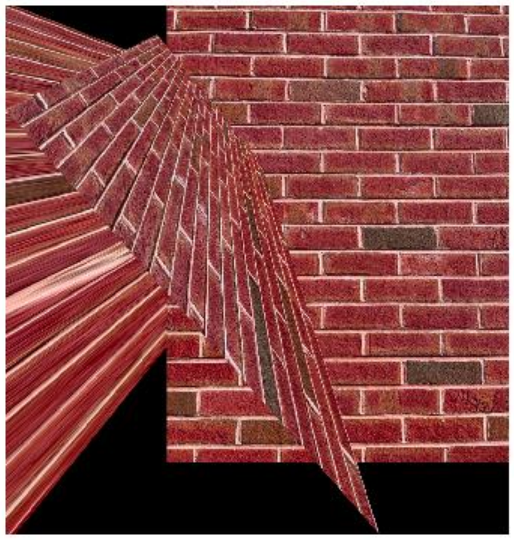
The above image shows the harris corners detected by the harris corner detection function. We can choose corners as a distinct feature as it shows variations in 2 dimensions giving us more accuracy than say if it were distinct in just one dimension. It can be easily used to stitch the images.

Part 5 – Final Mosaic:



This is a panorama generated by using Harris corners as features in the individual images and then stitching them together.

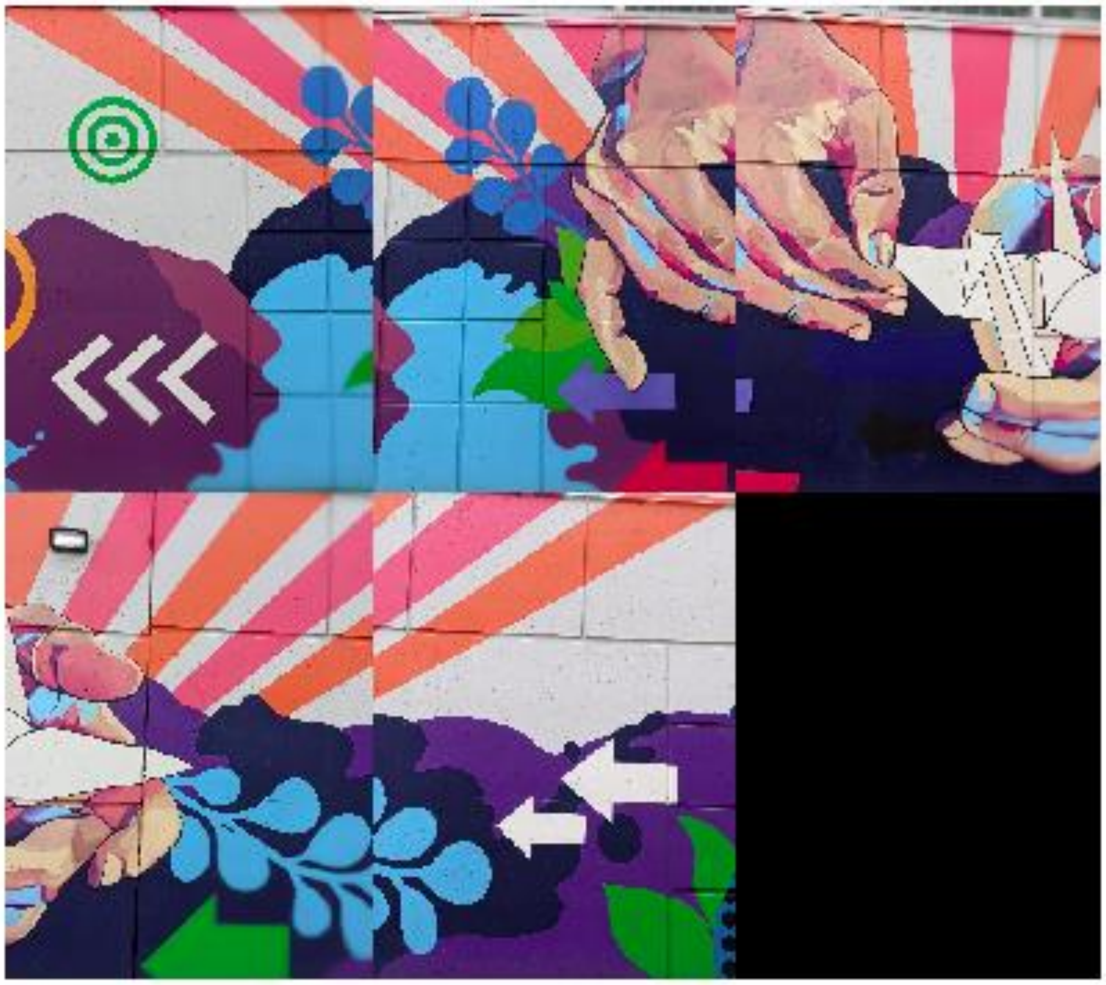
Part 6 – Cinder block imagery, the code fails!!:

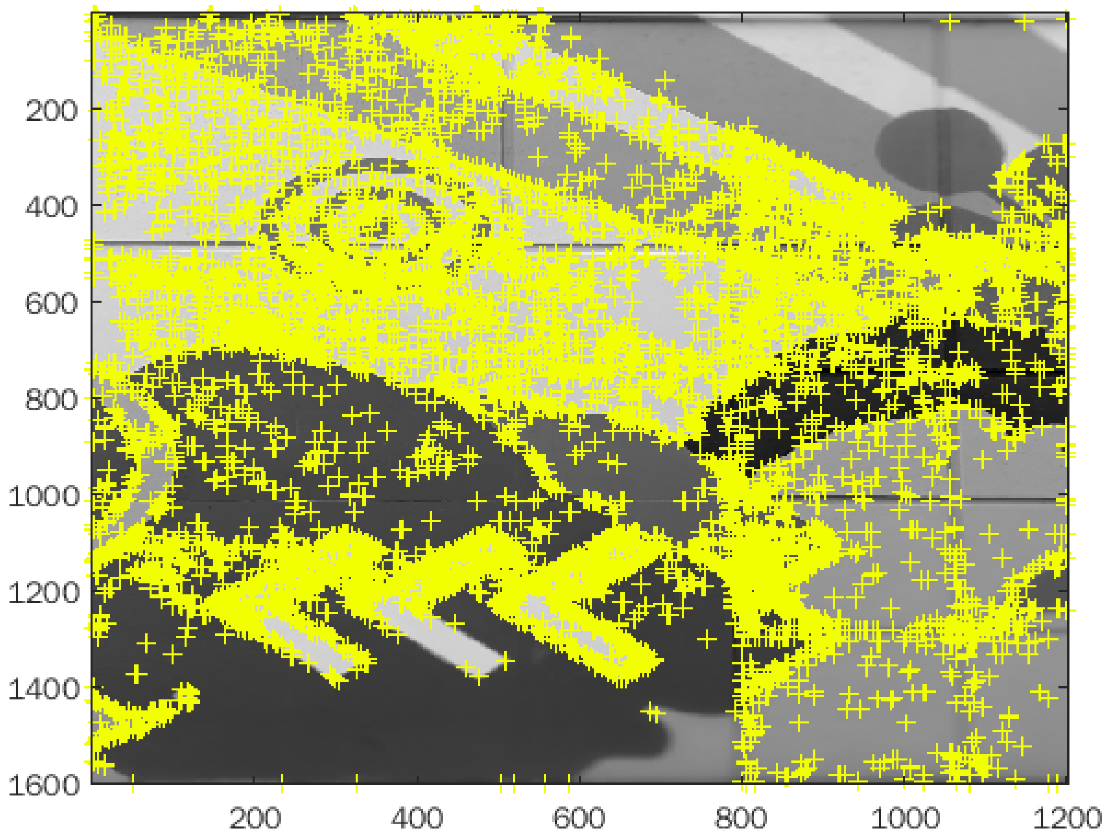
The first image in the set of four images shown above is the montage of five images that I have used for the cinder block mosaicking task. It is clear that the code fails to deliver in this case, and I have shown three resultant images obtained with three different configurations of the number of samples and tile size used.

We can observe that the blocks are in a repetitive pattern and cannot be distinguished as much in comparison to the graffiti that we used earlier. There is no clear way of distinguishing one small brick from another as they might nearly have the exact same features (at least when Harris corners are considered). So, when I run them with a different number of samples and different tile sizes, a lot of similar-looking Harris corners are detected which creates a problem of singularity while stitching the images together as observed from the results of the code. Hence, because of the lack of distinguishable robust features, some of the images blow up and disturb the complete panoramic view of the image.

Part 7 – Initial and Final Mosaic:

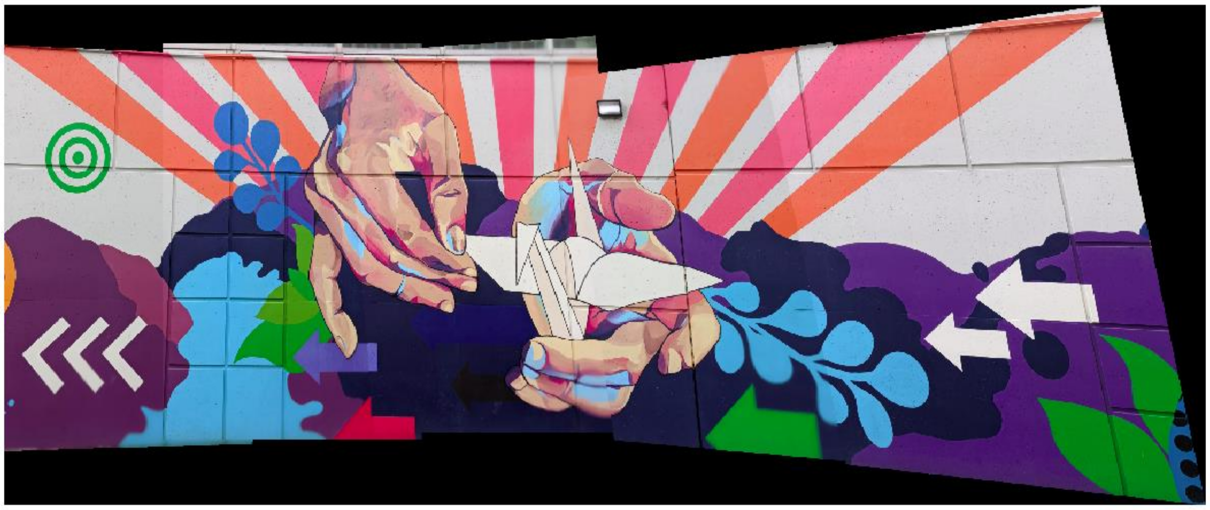


The above shown montage is the set of five images with 15% overlap used for creating a panoramic mosaic.



It can be observed that the number of samples used here are considerably higher than that used for the first graffiti with 50% overlap. This is because as there is significantly less overlap, with the same number of features as the first one, there is a lesser concentration of features in the areas that overlap which causes trouble in matching the images and stitching as the confidence level would be lower.

When the number is increased, we get enough features (Harris Corners) to be able to accurately match and stitch the images with high confidence to create the panorama shown below.



The script files for all are uploaded on Gitlab. The script name ending in 15 is for the 15% overlap and the script name ending in NW is for the cinder block. Along with the scripts and the report, some other things that are uploaded are the images used, the output data from calibration and the output data of individual images used in calibration.