LAB 4: EECE 5554 – Robotics Sensing and Navigation

PART 1 – Camera Calibration

This part mainly contained a procedure to ascertain the errors in the camera used for the entire lab and calibrate the images to remove the error caused due to distortion of the images due to the nature of lenses used for capturing the images.

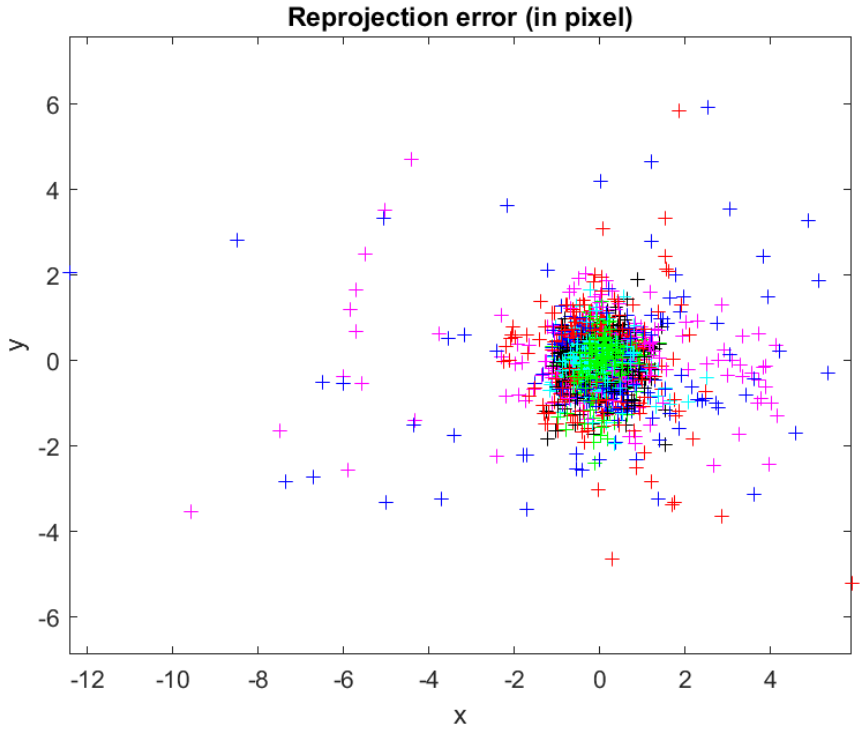
Certain facts about the exercise that need to be noted are:

- The phone used was an iPhone
- There were a total of 27 images of the checkerboard that were clicked for the calibration exercise

On the first round of ascertaining the error, the corners were attempted to be extracted manually by the user. The errors obtained in this case were quite high and as follows:

Pixel error: err = [1.15333 0.82984]

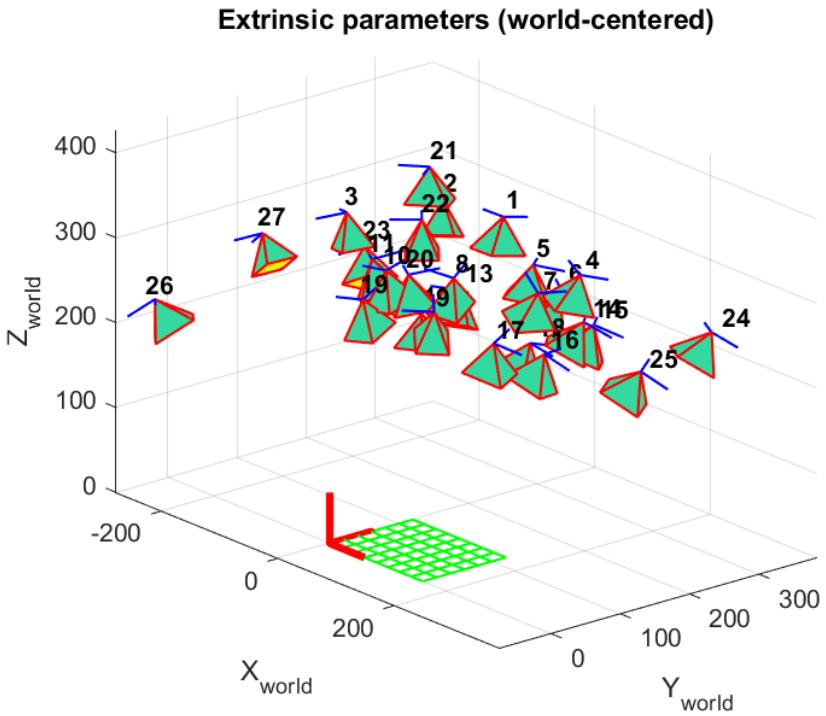
The error plot was as follows:



Next, I tried to recompute the corners automatically using the auto-compute feature of the calibration tool provided to us. The error obtained after calibrating the results by recomputing the corners automatically were as follows:

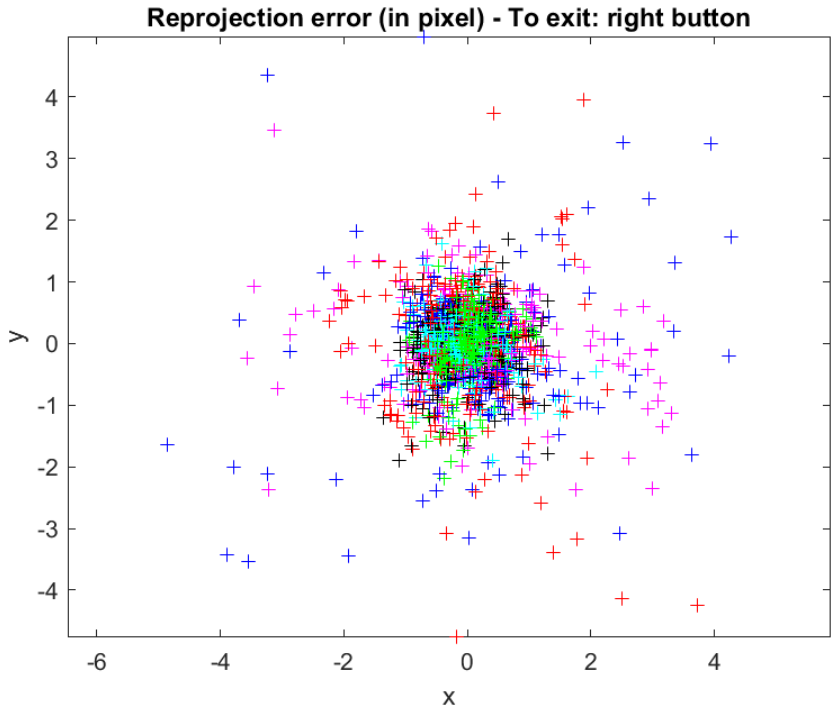
Pixel error: err = [0.82822 0.76294]

The 3D world centered view of the positions of the camera was as follows:

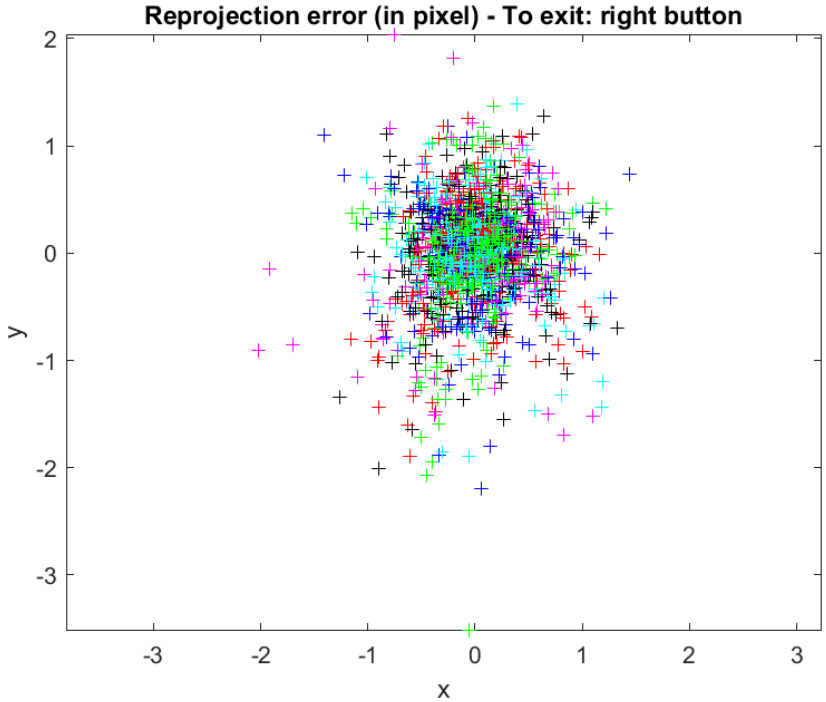


But I felt this error was still quite high and kept a target of reducing this to around 0.4 across both the axis. I began with trying different window sizes from the default value of 5. This was not very helpful and changed the error by a small value for different values. Hence, I decided to keep it as default (=5).

The next step to reduce the pixel error was to analyze the error to see if there are any particular images that contribute to the total error more the others, or if certain images behave like outliers and hence contribute unevenly to the error and then suppress them. I used the analyze error feature of the calibration tool for this. The error plot was as follows before any analysis:



In this plot, we can see that the images that are represented by the dark blue, pink, and red crosses have a large amount of points that lie outside the major concentration of points near the center i.e. (0,0). On clicking on these points, it was observed that these points were from images 24, 25, and 26 from my image set. Hence, these three images then subsequently suppressed, and another calibration was conducted on the remaining set of images. The calibration at this point gave the following error and plot of error for the images:

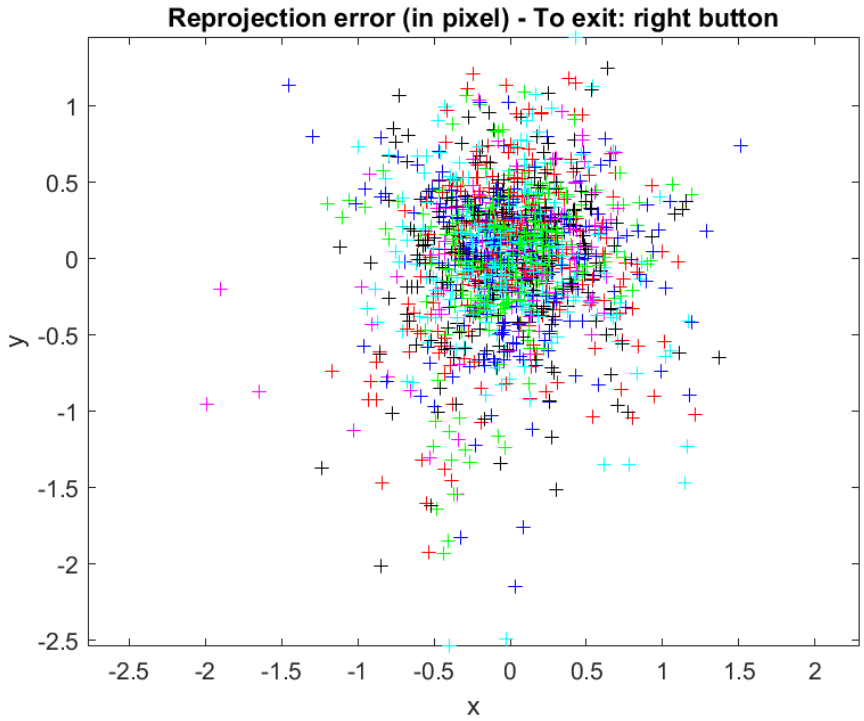


Pixel error: err = [0.41463 0.50584]

In this plot, the x-error had almost reached the target 0.4, but the y-error was quite high. With an aim to reduce this error, I next decided to suppress images represented by the light green and pink crosses in the above error plot. These were images 12 and 15. The results of calibration after suppressing these images was as follow:

Pixel error: err = [0.41364 0.47979]

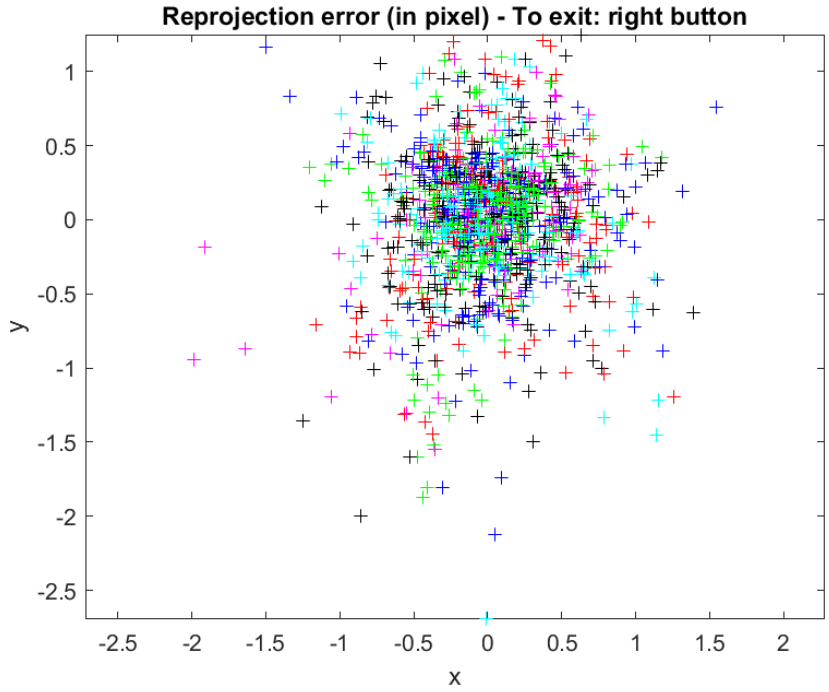
The pixel error plot generated after this cycle if image suppression is shown below:



Here, I decided to next suppress the images with the cyan and red crosses (images 14 and 17). The error after this was:

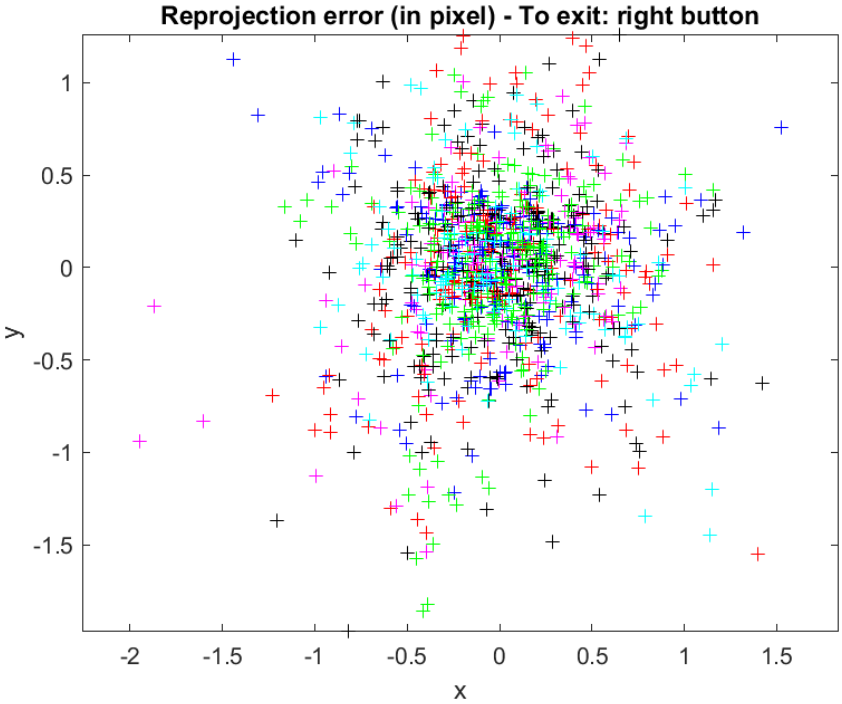
Pixel error: err = [0.42198 0.47067]

The error plot at this point was as shown below:



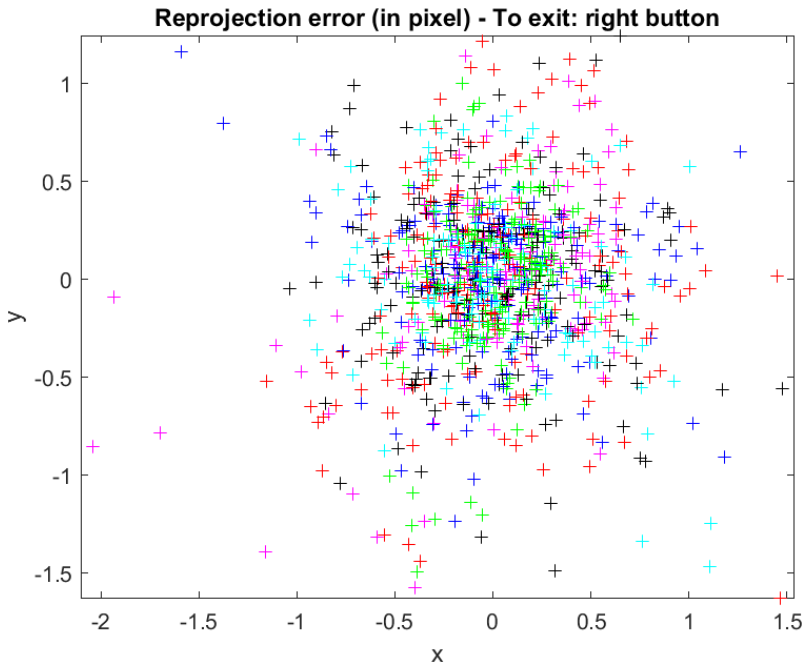
By the time I reached here, I had already suppressed 7 images in total and reached a total of 20 images which I had decided to keep as a possible number of minimum images for calibrating the error. But since the error was a little bit far from the desired value, I decided to continue a little more. This time I suppressed the cyan and dark blue crosses (images 5 and 13) and got the following results:

err = [0.42963 0.45680]

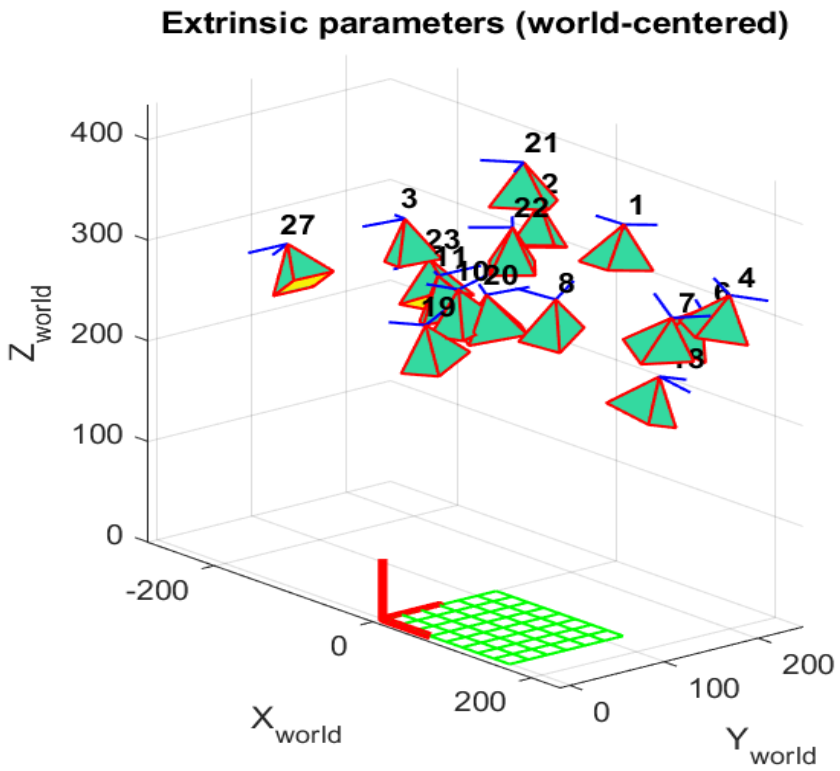


Here, I decided to suppress the images for the black and light green crosses at the bottom of the graph to reduce the y-error further. The results of this suppression are as follows:

Pixel error: err = [0.41705 0.43258]



The error at this stage was still higher than the targeted but the total images that I was working on had not reduced to only 16. Also, looking at the plot above, I realized that suppressing any more images would probably not reduce the error by a substantial amount anymore. But I did give it a try by attempting to suppress a few more images, but as predicted, it did not help much. Hence, considering all these factors, I decided to stop my analysis part here to try and reduce the error in the calibration of the camera.



The final world view of the position of the camera while capturing images that were finally left was as shown.

It can be noticed here that the error was eventually reduced to almost half of what it was at the beginning just by suppressing some images from the set.

PART 2 – Photo Mosaicing

2.3The photos were captured as explained at Forsyth street and the images were compensated for the camera distortions using the 'Undistort image' feature of the calibration tool. The resultant images produced by the toolbox were in black-and-white. The original images and the calibrated ones have been shown below. The colored images are the original images while the black and white images are the undistorted images produced by the calibration toolbox.

These seven sets of images are the pictures that were used for this part of the lab. As can be seen quite clearly, the uncalibrated (colored) pictures look undistorted and very much perfect to the real site when observed from our eyes. The calibrated images (black-and-white) are actually distorted. This can be noticed mainly at the corners of the pictures when we can see the straight lines bending. Therefore, we realize that the images produced by the camera used for clicking the original pictures had already been calibrated internally by the phone to remove the distortions. Hence, for this part of the lab, the images used for stitching the images together are the original ones since they are actually the undistorted ones. But the calibrated pictures were also stitched together out of curiosity to see what kind of output we get. Both these panoramas have been shown below. The parameters used for this part were the default ones. It was found that the features were distributed sufficiently across the image.

In the below pictures which are the stitched panoramas of the mosaic pictures, we see how the pictures are stitched when they are distorted and undistorted.

The panorama of the undistorted (original) images has been stitched pretty well. We can make out that it is not perfect, but it is decent enough. The final image is best aligned near the buildings. This finds reasoning with the fact that the Harris feature detector tries to detect corners. And this helps it detect a lot of the corners of different features of the buildings such as the building borders, windows, railings, etc. Hence, these features are the most prominent ones in the picture and hence result in the code matching these prominent features more easily than things like

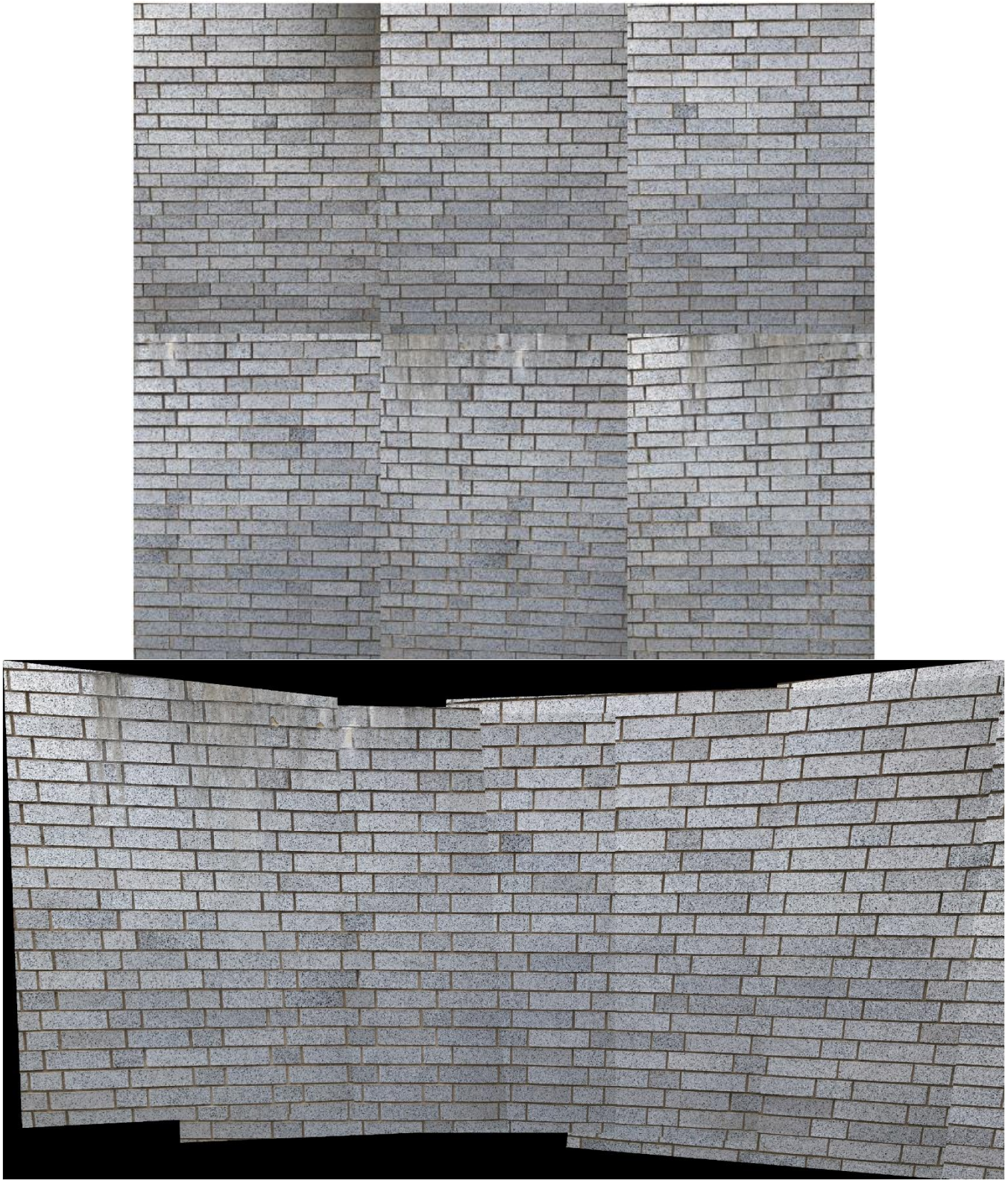
the road and the car which has less features with corners. Also, the detector used by us is a 2D detector and hence is not able to detect the 3D structure of the road which is in the axis towards the camera.

PART 3 – Mosaicing with varying overlap

Subsection 1:

The first part of this particular part of the lab was done on pictures clicked from the outer wall of the Cabot center that faces the Huntington Avenue. A total of six pictures were used for this purpose each with an approximately 50% overlap. These six images along with the stitched image are shown below.

In the stitched image, we can see that the images are not actually aligned perfectly at the levels they were supposed to be aligned. Some pictures are aligned with bricks that were actually lower than them in the original image. Also, it can be seen that the first five images have aligned with each other at approximately the mid-section areas signifying that the stitching has taken place near the 50% mark. Only the last (sixth) image seems to have not been aligned near its middle section. All of this can be



observed using the top and bottom areas of the stitched image. It shows us the design in which the images have been stitched with one another.

The parameters used in this case too were the default ones. The features detected were found to be distributed well across the images. This is mainly because of the numerous distinct corners that can be observed in the image due to the brick designs. This can be seen again in the result in which the misalignments are not easily visible. They are more clear towards the borders, but are still quite well-aligned.

Subsection 2:

For this part, the graffiti images taken were from boundary wall of the Ruggles station that faces the campus. There were initially five images that were used to generate the panorama image with an approximate overlap of 15-20%. In this part though, the default values were not producing good images. Infact, the default values would produce images that did not satisfy even the minimum number of matching features between consecutive images. Hence, the parameters were changed experimentally initially to see and understand their effects more deeply. On turning on the display parameter that showed us the features extracted for each pictures, a few things were discovered and steps were taken accordingly to help the system overcome them.

One major thing that was noticed was that features were at times not distributed well over an image. At times, there would be less points near the overlapping area between each image for the program to match them. To overcome this particular issue, the value of N was increased from the default value of 500 to 1000 at first and then to 1500. One major constraint in this was that a very high value of N would result in the code running out of memory. Hence, to keep the limits within the memory available, we had to match all the parameters in such as manner so as to get the best out of the entire package of parameters.

Another issue was that sometimes there would be sufficient points in the overlapping area, but since the tile size was [1,1] initially, it would miss a few points that might've been assigned a few pixels away from the pixels of the first image. This would result in the features not matching even though they were very close to each other and should have matched ideally. To overcome this, the tile value was slowly increased from [1,1] to higher numbers. It was noticed that we would reach a point where we would get a decently good stitched image that would then again get badly stitched as we went higher up with the tile values. This was ascertained to be mainly because the large tile value then accommodated too many features and matched features that were actually not matching. Thus, the tile value that gave a good stitched image was between [2,2] and [3,3].

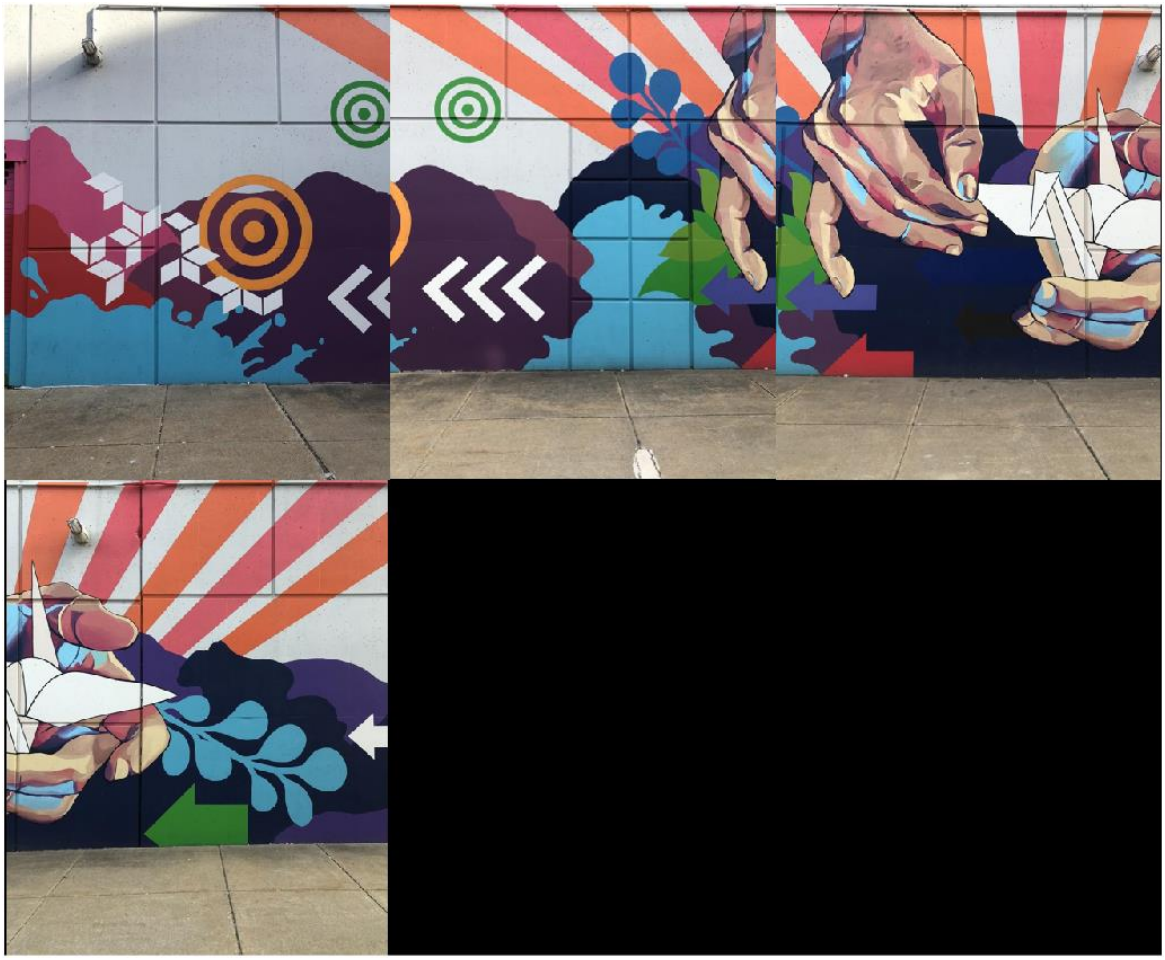
The parameters that we thus used for producing a decent enough panorama were N=1500, and tile=[3,3] though the tile=[2,2] value also gave a good enough result. The original images used along with the results obtained are shown in the images below.

It can be seen that the first four images have been stitched well enough to make sense when we look at them. The last picture seems to look very weird and out of place. This is believed to be due to very less corners that were sharp enough for the detector to detect and use for the stitching. Since there were less corners, the extracted features I this part of the image (which was to be used for

overlapping) were far less. The already less features might have further not matched entirely and hence with very less features matching, the program might have not bee able to stitch the pictures well.

To overcome this, I tried removing to fifth image to be able to better observe the stitching result of the first four images. The images and the results have again been shown below. In this image, we can see that the images have aligned really well with each other and the panorama image formed is very good.

The major features between the images seem to match as the images are switched. Only the first picture is slightly misaligned but is not badly aligned either. This again can be attributed to less corner features near the overlap area.



Conclusion:

This lab gave a very good insight into processing pictures with robotic applications in mind. We first learnt how to calculate the errors in the camera due to the inherent nature of lenses and then accommodating for these errors by calibrating the images to counter these errors. Further, we learnt the concept of feature extraction and subsequently using that feature extraction for an application such as stitching images together using these features.

References:

1. http://www.vision.caltech.edu/bouguetj/calib_doc/
2. https://www.mathworks.com/help/vision/examples/feature-based-panoramic-image-stitching.html