Camera Calibration and Image Stitching

Luke Davidson - Professor Singh - EECE 5554 - Lab 5

Note: Using 1 late day

Abstract

Determining whether a picture is an image of a cat or a dog, a human or a car, or a green light or a red light is rather simple to a human being. Our eyes and brains are innately able to process and describe an image, or set of images, at any given moment to analyze and draw conclusions about what is happening in the scene. Applying these steps of processing, describing, and analyzing images to robotic applications is a constantly evolving challenge that is addressed in computer vision applications. Through this study, we applied the method of Harris Corner Detection to obtain interest points and their feature descriptors across images. Using those points and descriptors, we were able to calculate position and orientation of an object, and match image features across a set of images to align them and create a panorama. We were successfully able to replicate a human's ability to understand the position and orientation of an object and the alignment of multiple images with overlapping features to create a larger perception of the environment.

Test Setup

We completed two individual experiments in this study relating to feature detection and alignment. First, we calibrated our phone cameras by taking pictures of a calibration image (shown in *Figure 1*) and calculating the relative orientation of the camera based on the detection of the corners of the calibration image. The image was placed as flat as possible on a flat surface and 20 images were taken at various different angles.

Our second experiment was aligning overlapping images of murals to create a panorama by identifying interest points and matching feature descriptors of those interest points. We took three different sets of images to create three panoramas: a mural where the images overlap by roughly 50%, a featureless brick/concrete wall, and another mural where the images overlap by roughly 20%. Images were taken with the camera face as parallel to the wall as possible as the assumption that it was a 2D surface was made. The images are displayed in the following section.

Data Collection and Analysis

Part 1: Camera Calibration

As stated above, the test setup involved placing a calibration image on a flat surface as securely as possible and taking images at various angles. The dataset of images is displayed below in *Figure 1*:

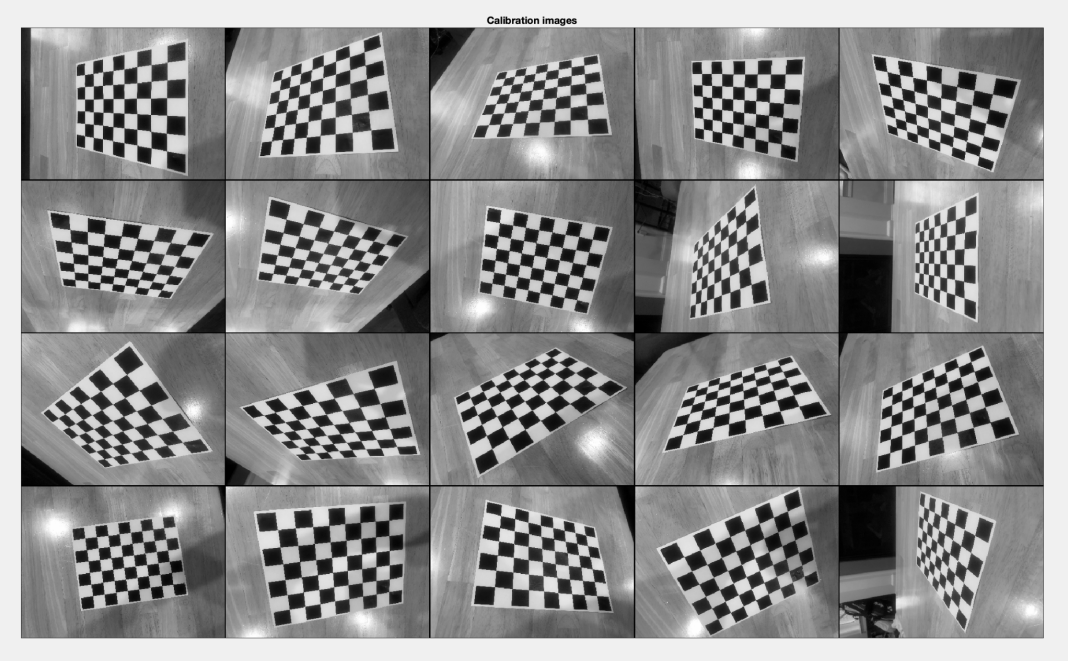


Figure 1: Set of images taken for camera calibration

By manually telling MATLAB where the four outside corners of each image of the above grid are, plus the size of each square, it was able to calibrate the camera with respect to both the position and orientation of the camera lens. To get a better idea of how the position and orientation calculations contribute to the calibration process, below are the visual extrinsic representations of the dataset. *Figure 2* represents the orientation of the grid with respect to a stationary camera, and *Figure 3* represents the orientation of the camera with respect to a stationary grid.

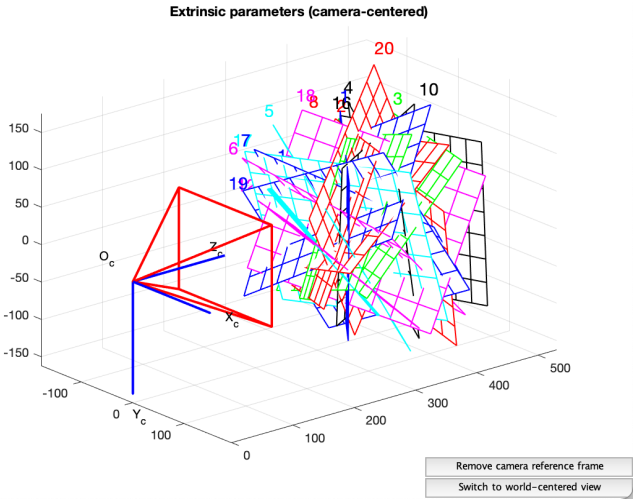


Figure 2: Grids w.r.t. Camera Position

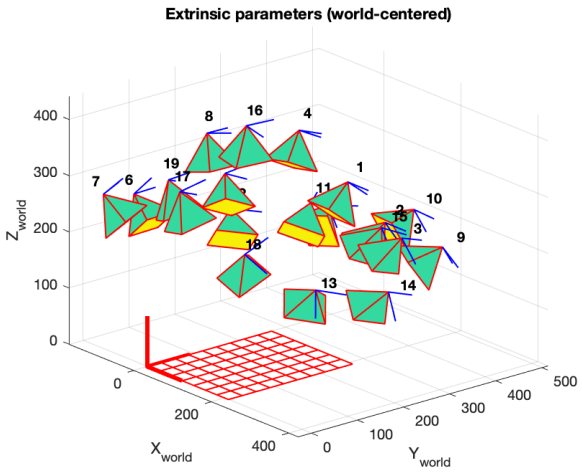


Figure 3: Camera Positions w.r.t. the Grid

Each individual corner was initially calculated based on the given locations of the four main corners of the grid and the length of each grid line. Since the selection of the corner is not expected to be the exact pixel of the corner due to human error, an initial buffer window of size 11x11 pixels was applied to the corner detection. The initial corner estimation and reprojected corner estimations are shown below in *Figures 4 and 5*:

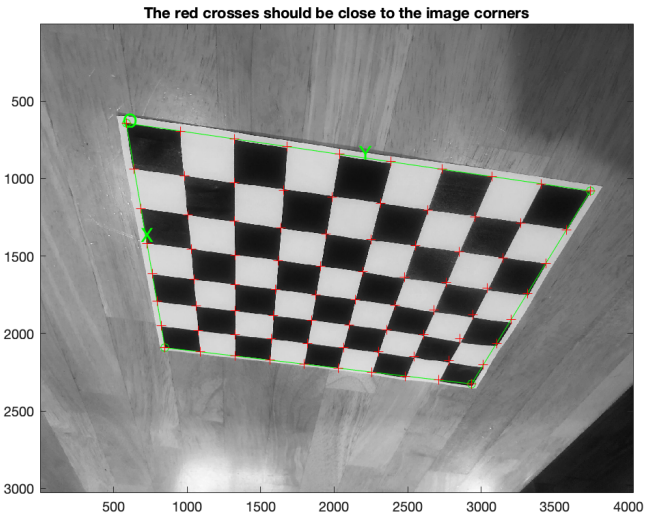


Figure 4: Initial corner estimations

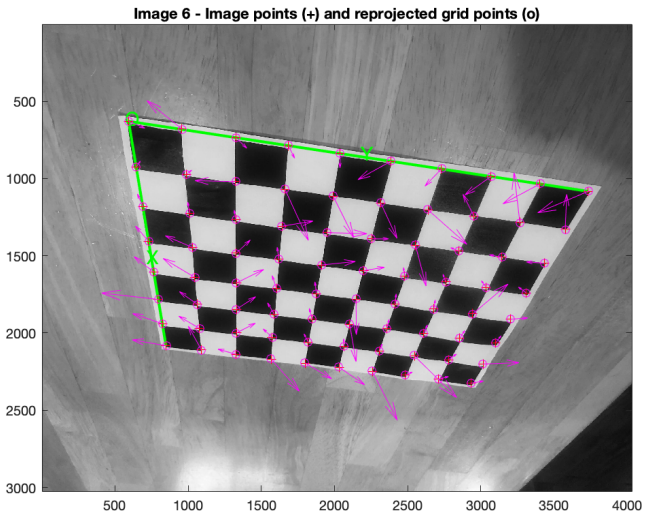


Figure 5: Reprojected corner estimations

Based on the initial window size of 11x11, the following calibration results were obtained:

Calibration results after optimization (with uncertainties):

Focal Length: fc = [3342.39856 3326.37989] +/- [4.97954 5.09573]
Principal point: cc = [2000.87248 1457.67162] +/- [9.24275 7.18494]
Skew: alpha_c = [0.00000] +/- [0.00000] => angle of pixel axes = 90.00000 +/- 0.00000 degrees
Distortion: kc = [0.00982 -0.02023 -0.00251 0.00081 0.00000] +/- [0.00796 0.02552 0.00080 0.00101 0.00000]
Pixel error: err = [1.82486 1.82047]

The correlating error graph can be seen below in *Figure 6*:

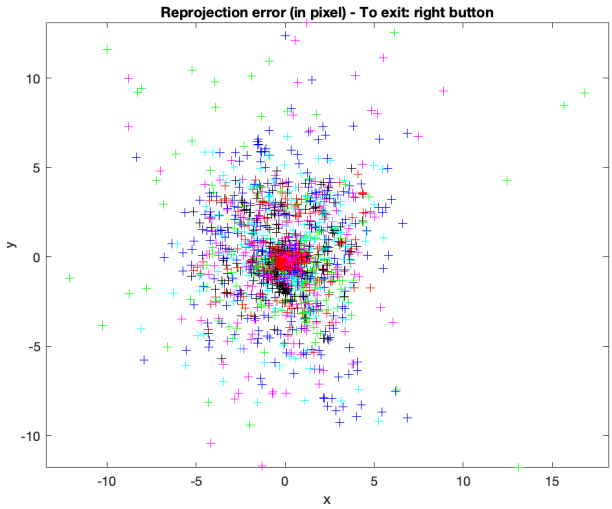


Figure 6: Initial Reprojection Pixel Error

As one can see in *Figure 6*, there were certain images that resulted in more error than others. The main three images that resulted in more error were Image 7 (green crosses), 12 (pink) and 15 (blue). This larger error results from the corners of those images not being detected as exactly as many of the other images. To further calibrate the camera and eliminate some sources of this error, I chose a larger window size of 19x19 to extract the corners out of images 7, 12 and 15. The recalibrated results, along with the new reprojection pixel error plot are shown below:

Calibration results (with uncertainties):

Focal Length: fc = [3345.61693 3329.96363] ± [4.40058 4.49969]
Principal point: cc = [1999.87299 1463.80382] ± [8.20269 6.35813]
Skew: alpha_c = [0.00000] ± [0.00000] => angle of pixel axes = 90.00000 ± 0.00000 degrees
Distortion: kc = [0.00368 -0.00559 -0.00137 0.00054 0.00000] ± [0.00702 0.02259 0.00070 0.00089 0.00000]
Pixel error: err = [1.65344 1.55885]

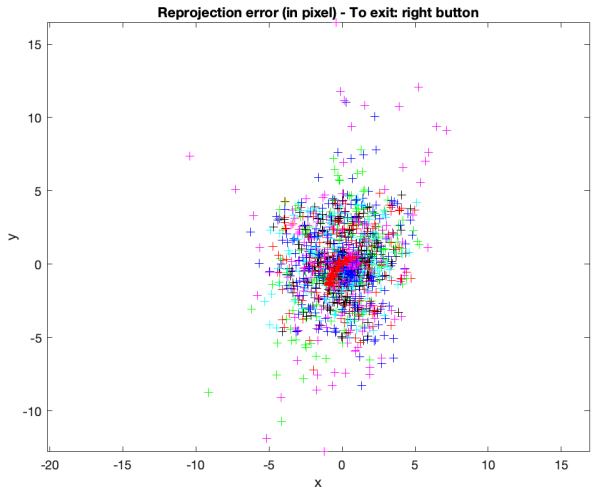


Figure 7: New reprojection pixel error plot

Increasing the window size to search for the exact corner proved to be successful in eliminating some of the pixel error and improving the overall calibration of the camera. As one can see in *Figure 7*, the pixel errors are more concentrated around the ideal (0,0) location compared to *Figure 6*. This is also seen in the overall pixel error calculation, dropping from ~1.82 to ~1.60. However errors in images 7, 12 and 15 are still seen in *Figure 7*. This is likely due to the quality of those images specifically. Multiple factors can contribute to this error including image distortion, picture quality, and the orientation of the grid at the time of the image.

Example images showing the effects of distortion of both before and after camera calibration are shown below in *Figures 8 and 9*, respectively:

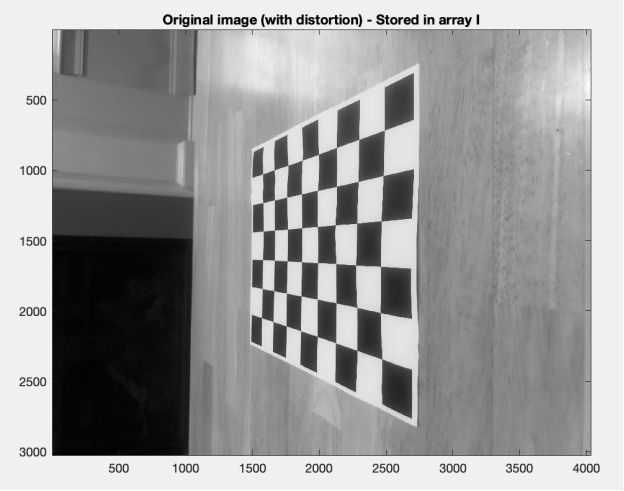


Figure 8: Example image before calibration, showing distortion

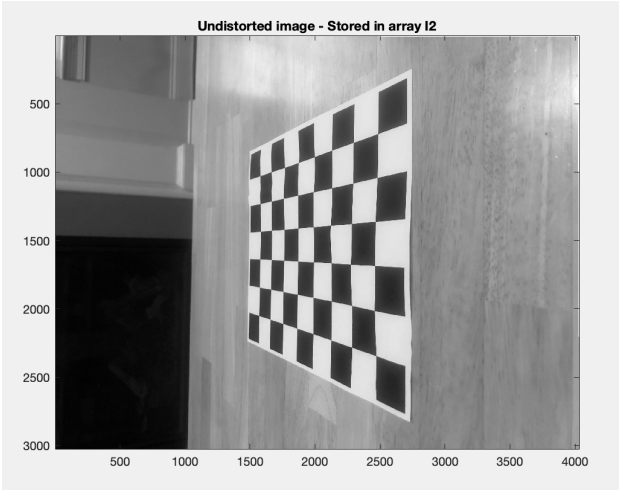


Figure 9: Image after calibration

Part 2: Harris Corner Detection and Image Stitching

For the second part of this study, we focused on the detection and alignment of interest points between overlapping images. In order to align two images with overlapping sections, two main things need to be found: interest points and their corresponding feature descriptors. Interest points are defined as the locations in an image whose position and orientation can be easily identified based on the surrounding environment. In our application, we used the Harris Corner Detection algorithm to extract feature points. Harris Corner Detection works by convolving certain filters, typically the Sobel Filter, over images to extract corners. Corners are of high interest because they are matched with one specific location in an image. For example, in an image of a solid color, it is hard to extract interest points because each pixel window will look similar to every other pixel window, so a given point can be anywhere in the 2D plane. Similarly with an edge, each pixel window up and down that edge will look similar, so it is hard to tell where a feature may lie on that 1D edge. Corners will look different no matter which way you move them, so their precise location is easy to calculate. The

Harris Corner Detection algorithm also calculates the strength of a corner, known as an interest point's cornerness response function. The higher this value is compared to its surrounding interest points, the better and more unique that point is.

In order to align interest points between two images, a method must be used to identify which points are in fact the same between the two images. In order to describe the relative position of a point in an image, one must know its relative position compared to its surrounding features in the image. This relative position is known as the interest point's feature descriptor. A very simple example of a feature descriptor is the pixel window surrounding an interest point. In a non-scaled, non-rotated set of images, comparing each pixel window surrounding each interest point in one image to those in the other image, one can determine which windows match, and thus, which points are matches. More complex forms of feature descriptors are typically needed due to scaling, rotation, and orientation changes of images.

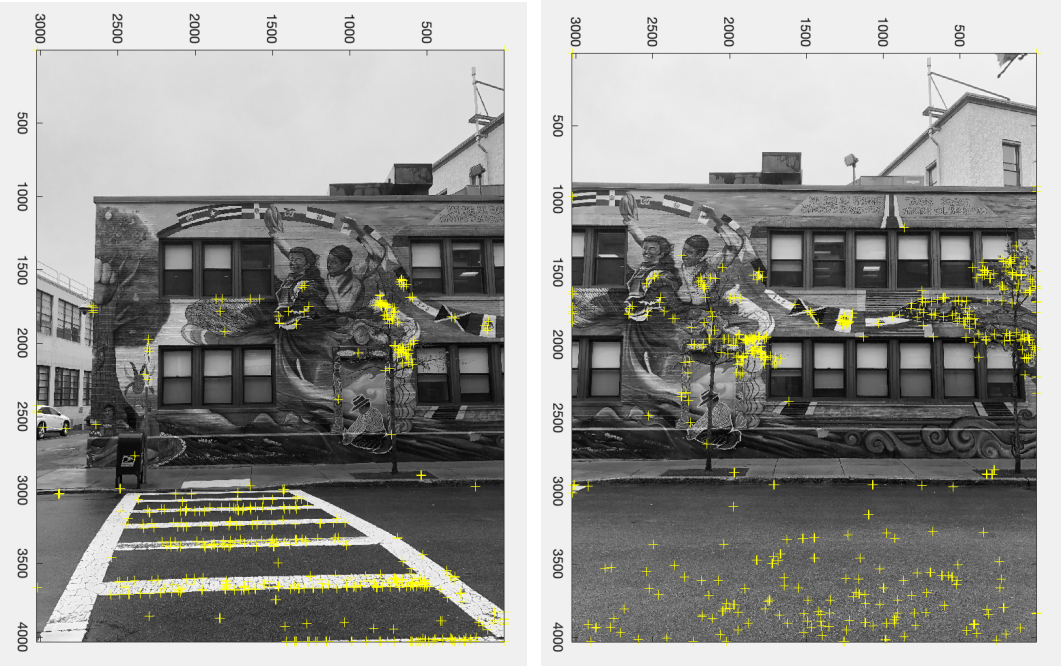
In this study, we used the Harris Corner Detection method to detect and align interest points in sets of five overlapping images. The images, results and analysis are shown.

Dataset #1: Latin Mural

The first set of images we took was of the Latin History Mural near Ruggles. We took five images spanning across the whole mural, overlapping by about 75%. The images are displayed below:

Figures 10 - 14: Dataset 1 Images

As explained above, the first step in aligning the images is locating the interest points in the image. Using the Harris Corner Detection algorithm, examples of the top “N” detected corners in images *15* and *16* are shown below:



Figures 15 and 16: Detected Harris Corners

By extracting the feature descriptors for each of these points, we were able to match individual features, estimate the transformation between each neighboring image pair, and align them all together to create a final mural, shown below in *Figure 17*:

Figure 17: Latin Mural Panorama

In order to get the images to align well and fully display, I had to increase the number of interest points the Harris method outputted. Increasing this value allowed there to be a sufficient amount of matches between the images, and an increase in the quality of image alignment.

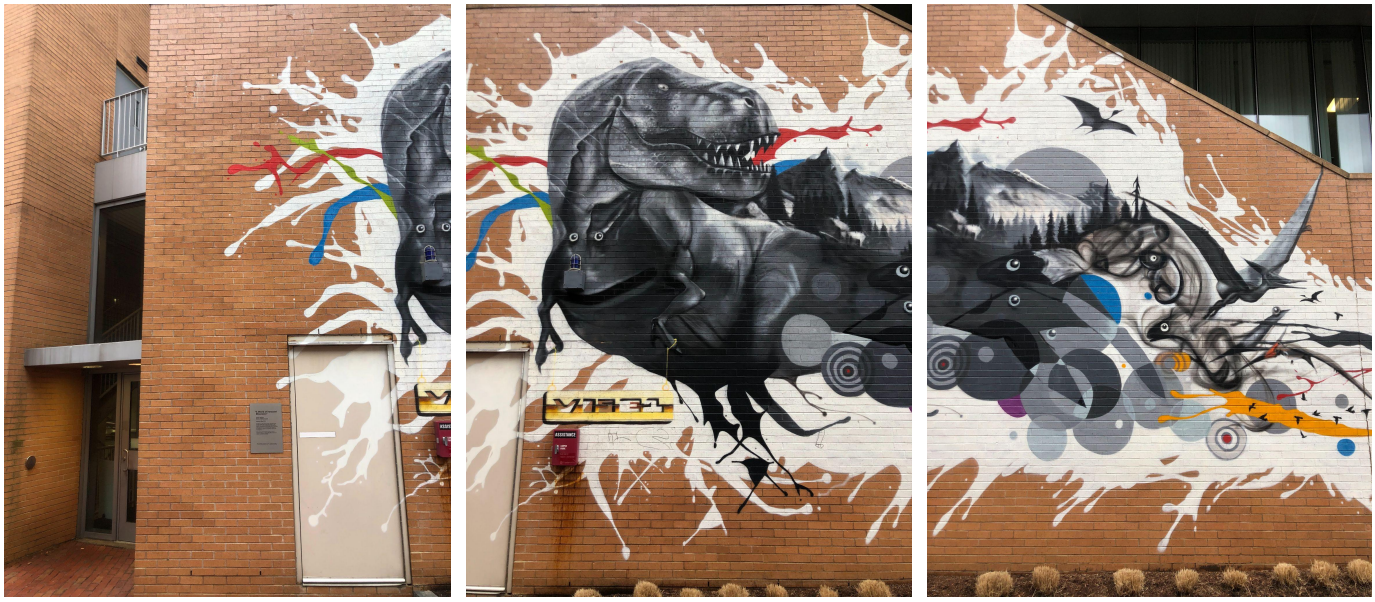
We completed this process for two more datasets: a featureless concrete wall and a mural with image overlap of only roughly 20%. The images for each data set, as well as the final mural, are shown below:

Dataset #2 Mosaic: Concrete wall

Figures 18-22: Concrete wall dataset

Figure 23: Concrete wall panorama

Dataset #3 Mosaic: Dinosaur Mural



Figures 24-28: Dataset 3 pictures

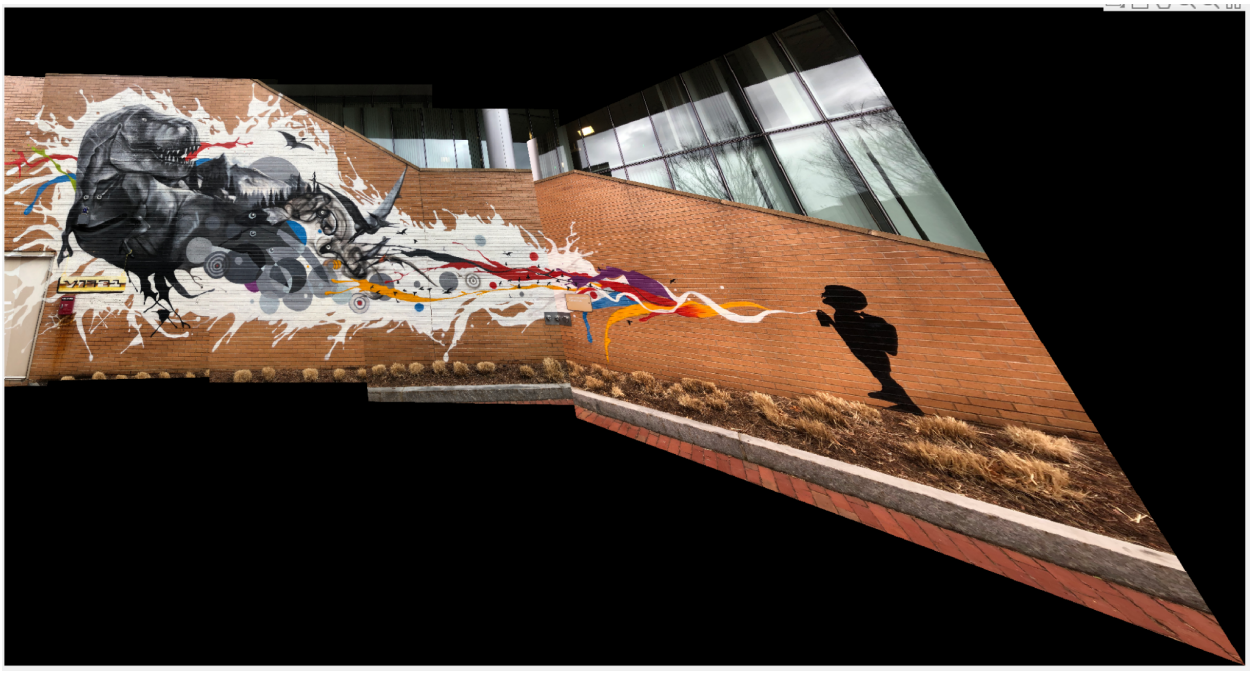


Figure 29: Dataset 3 mural with N = 5000

As one can see, the panoramas in *Figures 23 and 29* did not come out as well as the first dataset. For the second dataset, this is due to the lack of defining features in the image. Since all the images are of blank, concrete, many of the feature descriptors of the interest points are very similar. This causes there to not be many good matches between the images, and therefore, a worse alignment. Similarly for dataset 3, there was less overlap of the images, and therefore less interest points to match. To solve this, I increased the number of interest points outputted by the Harris function. The resulting panorama with more matching features is shown below:

Figure 30: Dataset 3 panorama with N = 10000

Conclusion

Being able to define interest points in images is extremely valuable to many different robotic applications involving camera imaging. Through this study, we were successfully able to use the Harris corner detection algorithm to identify the orientation and position of an object, calibrate a camera, identify key interest points, define feature vectors, and align images based off of interest point matches.