CMSC426 Project 2: Panorama Stitching

Group Members: Marc Haller, Anthony Mahshigian, Matthew Wong

Project Overview:
The goal for the project from start to finish is to take any number of overlapping images and produce a panorama result via automation.

Each step of the pipeline is outlined below. For a high level overview:

- We begin with detecting corners. The cornermetric function inside of MATLAB gives us estimated corner points
- We then pick a constant amount of the best corners we have with ANMS (Adaptive Non-Maximal Suppression). In this case, we set N_{best} to be 300 (for most images)
- Next we create feature descriptors. We use a surrounding bounding box to get the pattern of the pixels around it
- Then we match the feature descriptors that we created across images. Only keeping the closest matches
- We then use RANSAC to remove outliers (bad matches) and produce a homography. This is a mapping for one image onto the other
- Finally, using the homography, we can stitch together the images. This involves putting them on the same plane and having a result where they overlap

1) Adaptive Non-Maximal Suppression (or ANMS)

The first step is to detect corners. We've made a function ANMS. There weren't many challenges here, as we got corners as expected. We found all the corners in the grayed image by using the matlab cornermetric function and then compared them to get the strongest corner for each region. We wanted to make sure that we can make a meaningful comparison this way and we'll be able to get a good number of matches across the images. There were a lot that didn't represent corners in the images, but overall it looks pretty good for the test data - we're happy with the results and can move forward. Also one thing to note was that during later debugging, we went back and made sure our ANMS was producing consistent results. It was, which is desired and based on the eye test. A lot of them were on areas with high gradients, which is the goal. We played around with our N_{best} value on certain images to produce better results.

2) Feature Descriptor

This was another relatively simple step. Our goal is to make something that we can compare corners/features on. We start this by taking a 40x40 size patch around the feature (using cropping), applying a gaussian blur, then subsampling. We flatten the 8x8 subsample into a 64x1 vector that we can compare to features in other images to determine their similarity. The challenges came when matching these feature descriptors. Even if two feature descriptors are the same exact parts of a picture, they are still very error prone due to blurring, brightness

differences, pixels being in different places, etc.. These feature descriptors are only approximations, but that should work for our purposes.

3) Feature Matching

The next step is to match features based on how similar their comparable 64x1 feature descriptor arrays are. We chose a threshold and filtered it out so for each comparison, only decent matches came out. There are still a lot after this that are obviously not matches, but we at least have what appears to be a good trend / progress in our feature matching. Our next step will make these matches even more accurate. We had a lot of struggles here with finding a good ratio to add matches with, especially when we got the test set. We didn't have many issues for the first set we had, but we noticed that we had to play with our tolerances and tweak them for each pair of images. Usually there wasn't much debugging that had to be done here, but it was a notable challenge.

4) RANSAC

After we get *n* matches, we use the RANSAC robust estimation to remove incorrect matches and compute the 2D homography. For our algorithm, we tried to use an adaptive procedure to determine the number of iterations needed to get a large percentage of inliers. The steps for our algorithm are shown below.

RANSAC Algorithm

1. While N > iterations or iterations < max_iterations
    a. Pick 4 random matches on the first image
    b. Compute the homography for the 4 matches
    c. Get the number of inliers
        i. Apply the estimated homography to all the points on the first image
        ii. Find the difference between applied points and the matched points on the second image using Sum of Square Differences
    d. If the difference is below the inputted tolerance, add point as inliers
    e. Update largest set of inliers (m) if applicable for the current iteration
    f. Update N and *e* by computing $e = 1 - m/n$ and set $N = \frac{log(1-p)}{log(1-(1-e)^4)}$
    g. Increment iterations by 1
2. Recompute the homography using the largest set of inliers obtained in step 1e

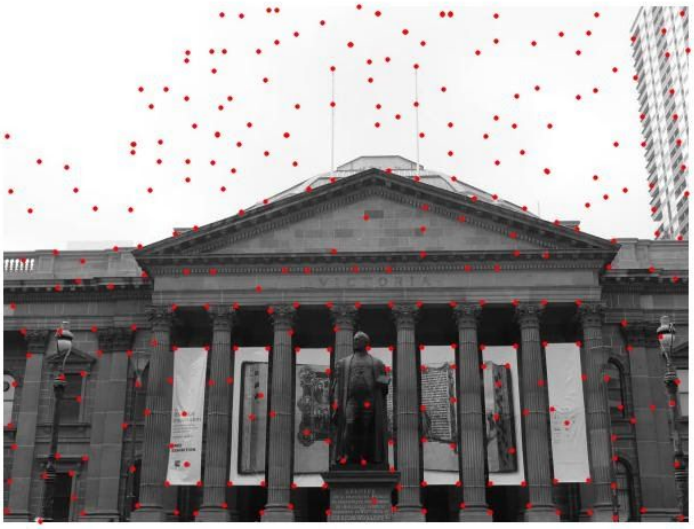
5) Stitching and Blending

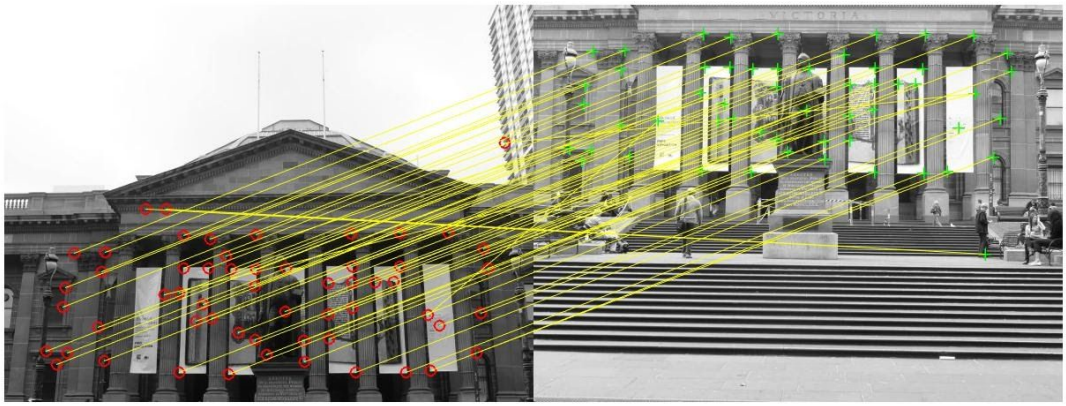
Our final step is the stitch the images together. The strategy we initially implemented was to first stitch the first two images and then the third one to that composite, and so on, until all the images form a single composite. We realized we had issues with the black space bordering the intermediate panoramas and we switched our strategy. Our new strategy was to compute the homography between any two images in the set and then have a final round of stitching using those computed homographies. This way, our feature matching and RANSAC worked much

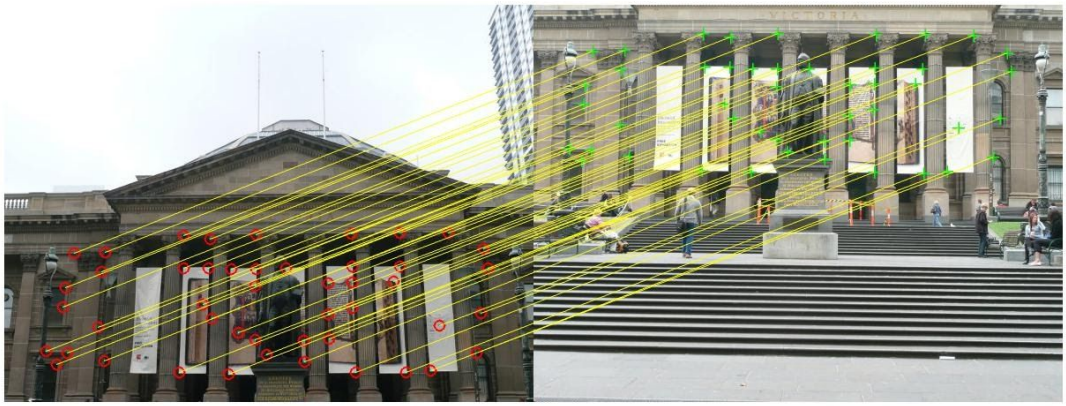
better, although we had issues with the homographies with it. However, we found it to be a better result.

The homography is used to map one image onto the second one. Then what we did was create a new image and put both data onto the new one. The new image has a size big enough to fit them both. It also has a fair amount of black space where neither image is. We tried a few different strategies for the overlapping regions, and struggled with many blending algorithms. If we averaged pixels we would have a very blurry image due to slight misalignments from our homographies. We then tried to alternate whose pixels we used, but this also resulted in a very poor result due to the misalignment. We ultimately decided to keep the base image's pixels, and have the 2nd image only have pixels added on that don't overlap. This produces a much more visibly pleasing result, and one that is the same as the examples. We were satisfied with this because it wasn't blurry and the overlap regions weren't very obvious at first glance all the time. Now we'll talk about some of the results.

Set 1

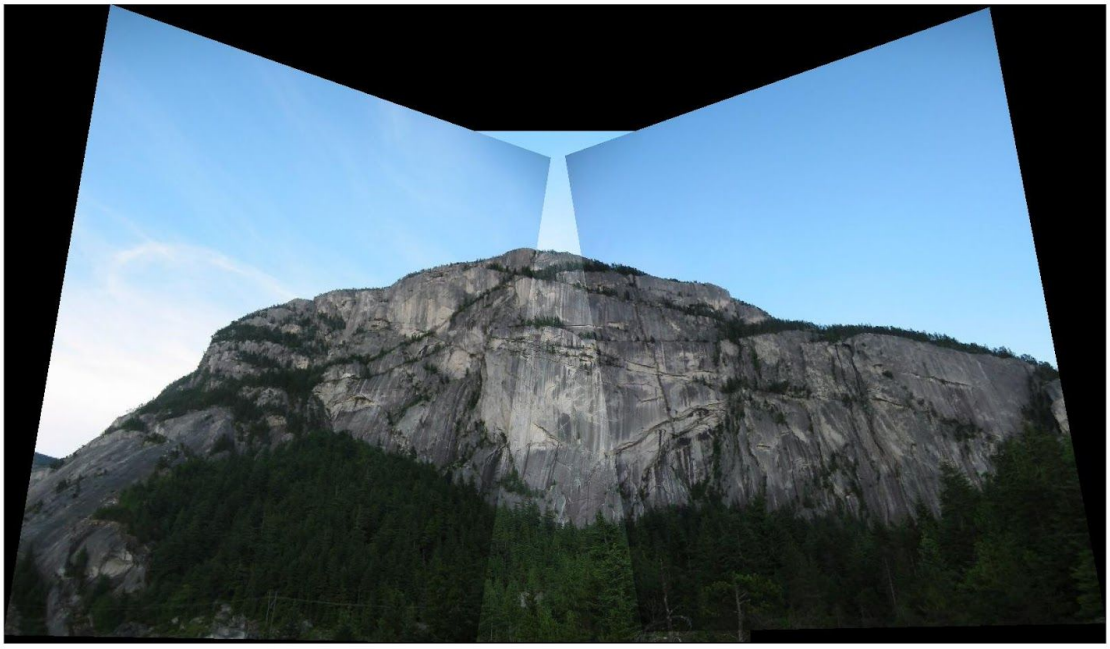




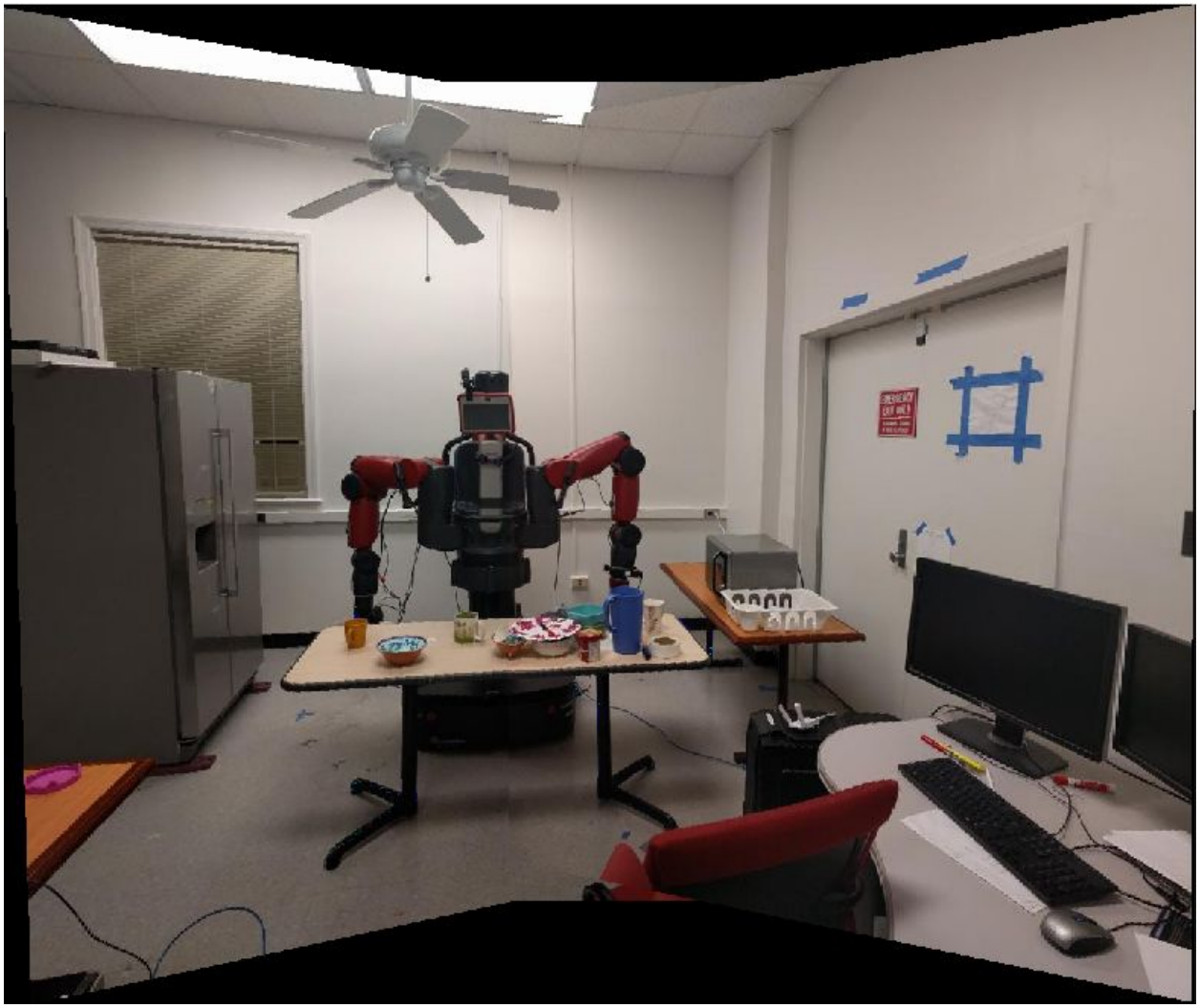



VICTORIA

Set 2



Set 3



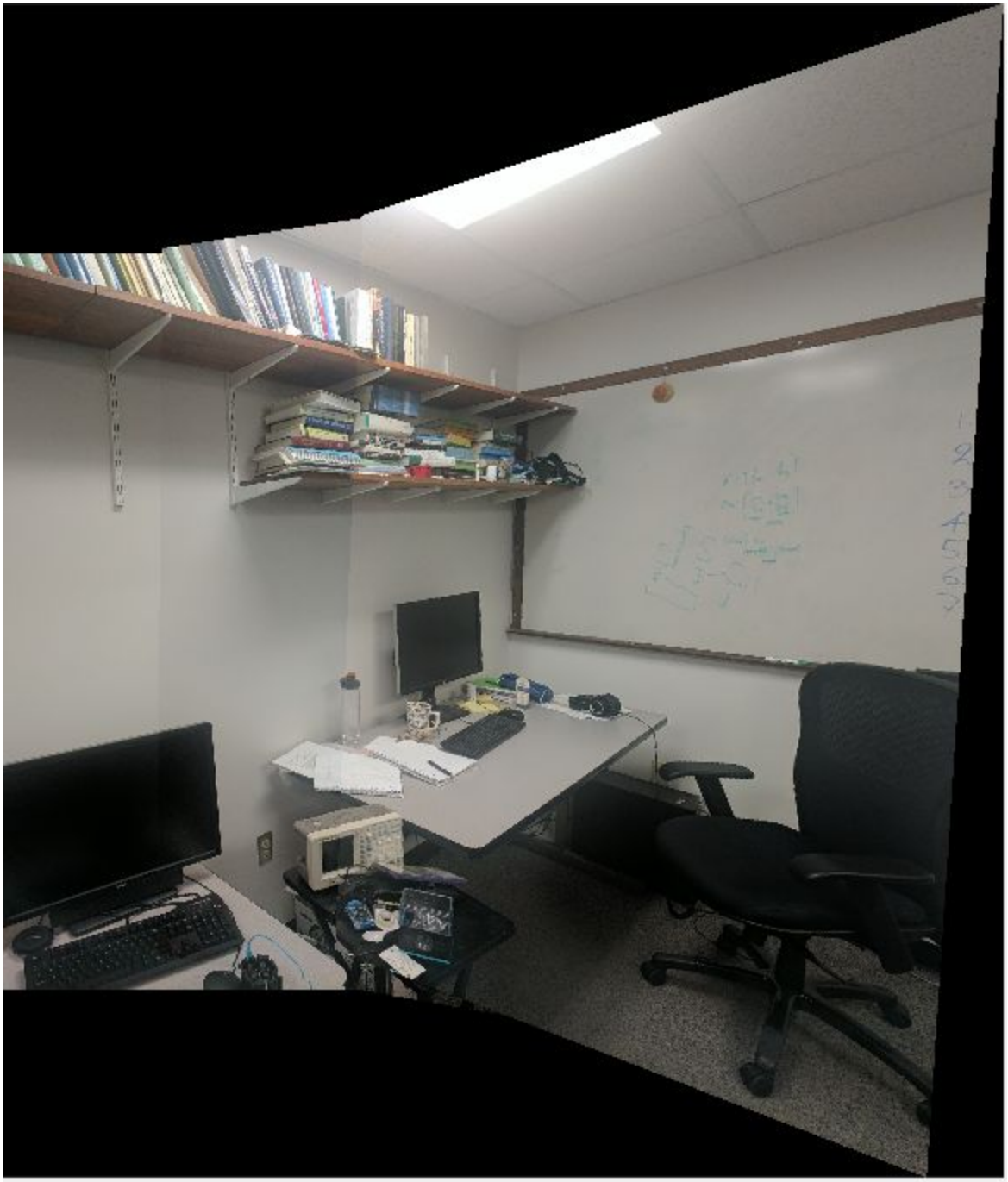
This version starts from center and gives a better image, but it matches less well beyond this point.

Test Set 1

This test set is of a checkerboard. This set presented plenty of challenges for us. To start, many of the matches were very bad since there are a lot of parts on these pictures that look the same. This forced us to have to tweak parameters/tolerances a lot to get an image that works. We still got the correct order of images, which we are very proud of, but the alignment isn't too great because of plenty of false matches.

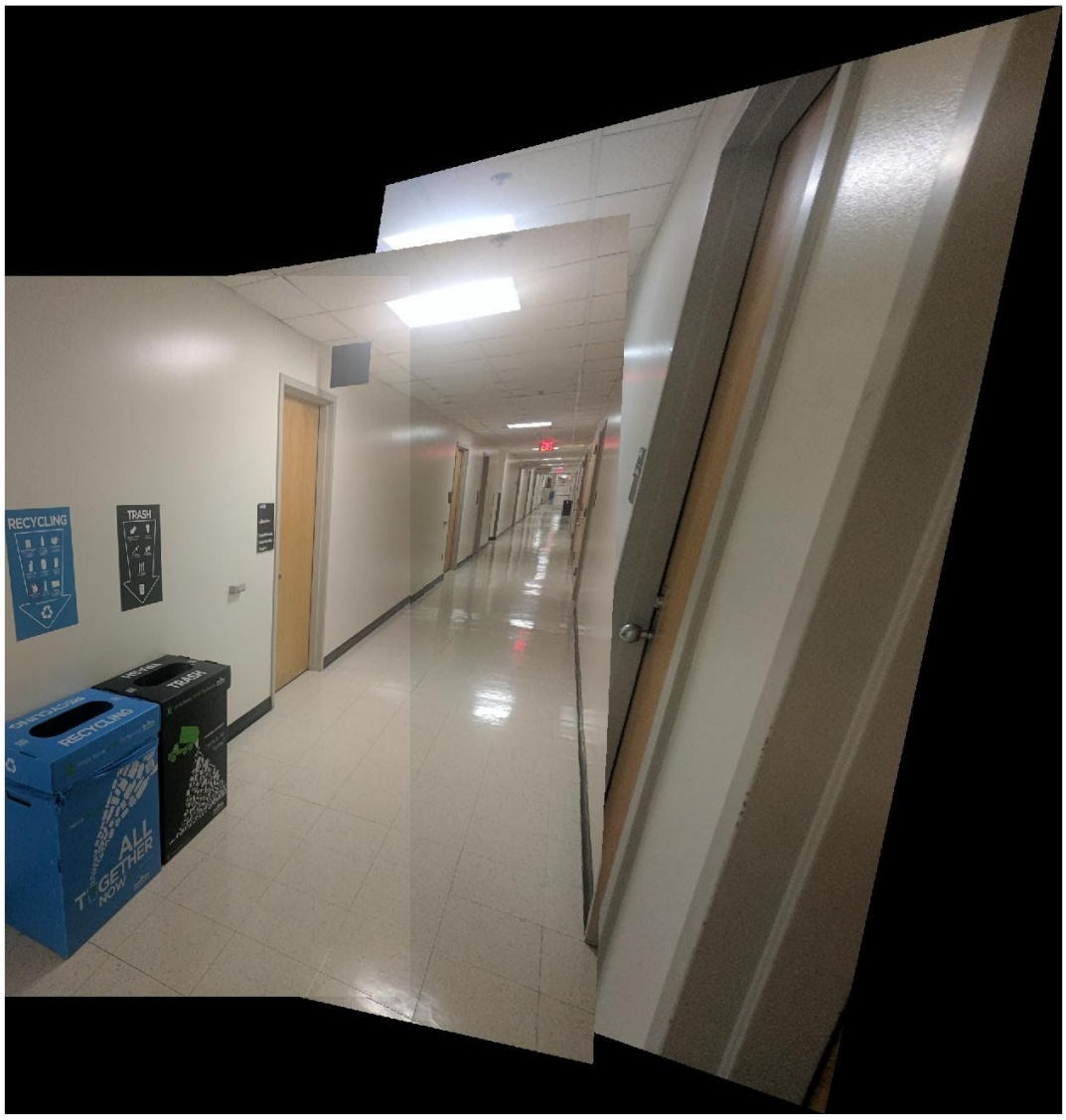
Test Set 2

The first image uses the process of computing homographies first and then stitches after, which produces an image containing more of the images, but has messed up homographies. The second image is stitching one at a time, which has them line up much better, but then it struggles to match thereafter.



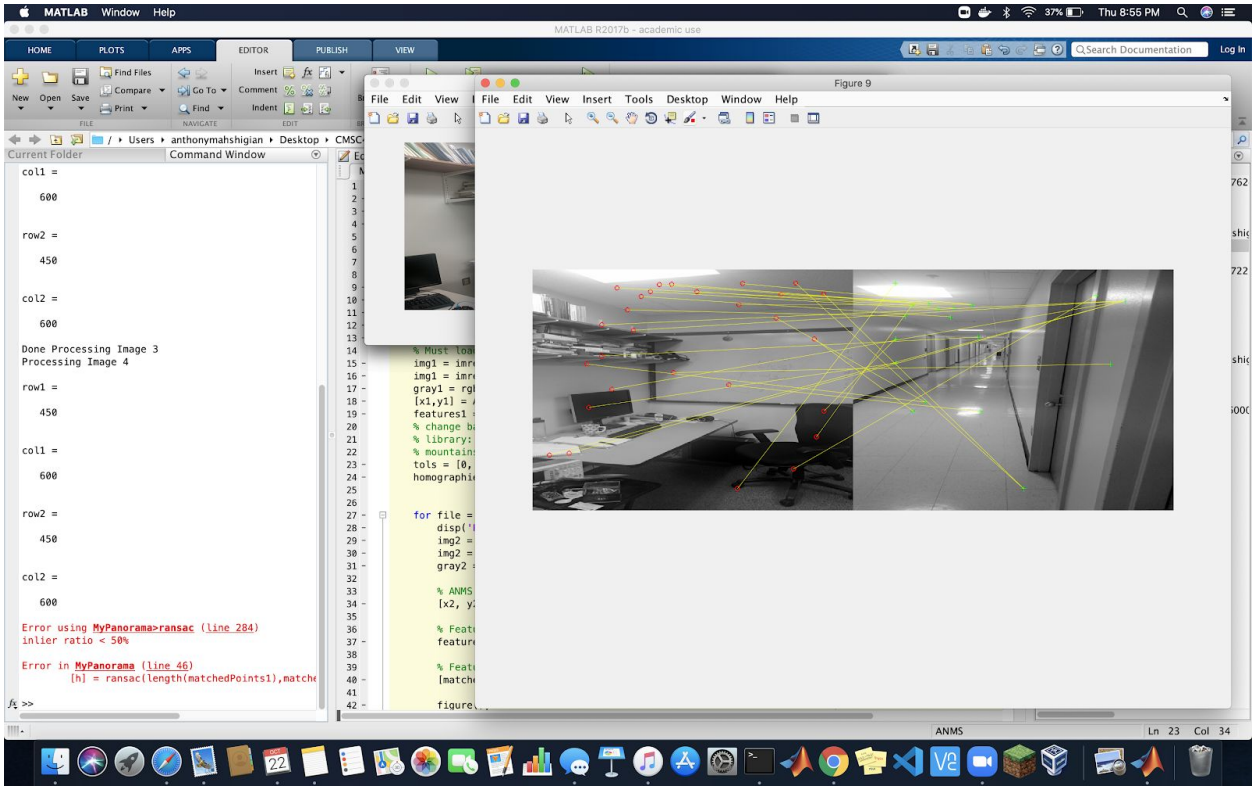
Test Set 3

This test set turned out very good. It was 3 images of a hallway and our code was able to pick up on the common features very well. There are a couple inaccuracies as the homography isn't perfect, but we can see the shape of the hallway very well.



Test Set 4

This is the test set with images that are not from the same pictures. As we can see, there are plenty of matches here that appear to be outliers. The expected behavior should be for the ransac to error by saying there are too many outliers. This is exactly what happens. We included a screenshot of the command window. It states the error that happened and stops the panorama at that moment.



Custom Set 1

Our first custom set is a kitchen with 3 images. There are a lot of parts in this image like the drawers that present challenges because many of the pixels look the same. The mapping still turned out very well, and the images are aligned pretty well.

Custom Set 2

This is another custom set we did with 3 images. We are also satisfied with how it came out, as we can get a good idea of what the room looks like. We can see misalignment on the windowsills as this is not perfect.