Homework 2 Report: Panorama Stitching

CMSC 426 Spring 2021

Hongyu Tu 115323568

For this homework, I split the entire project into 5 parts as instructed: ANMS, Feature Descriptors, Feature Matching, RANSAC (and Homography Estimation), and Image Warping (and Blending). The outputs of all train, test, custom images are placed at the end of this report. For demonstration purposes, I will go over the steps in my pipeline with 2 sets of pictures below.

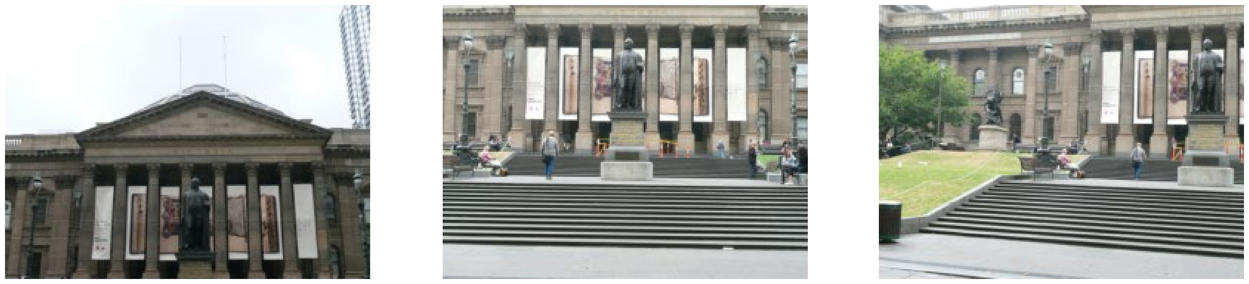


Fig 1. Image set 1

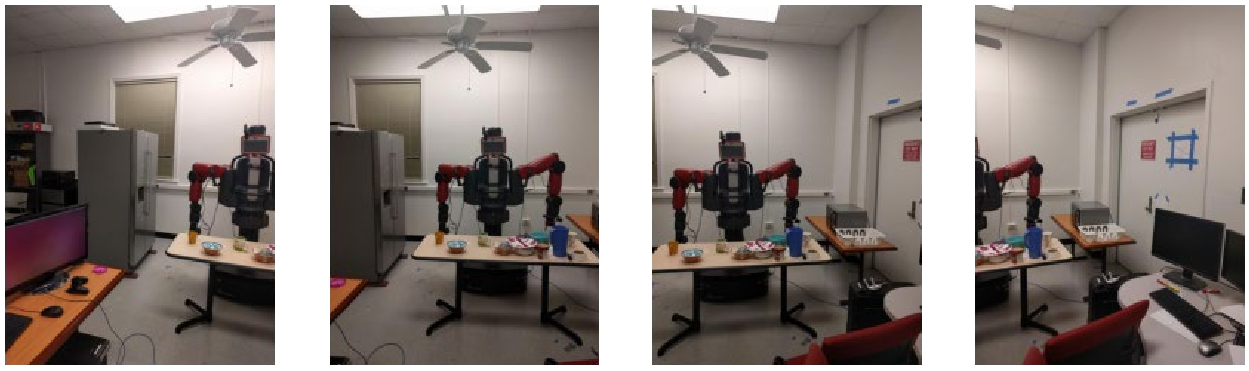


Fig 2. Image set 3

1. ANMS

As instructed, I used the cornerHarris function from the OpenCV library to generate the initial pool of points. To make my code more efficient, I reduced the original ANMS implementation to using just one loop over the list of corner points. In each iteration, Boolean indexing will be used to first compare the current point's C_{img} value to the whole C_{img} value list, marking the index of all suppressing points. Then Broadcasting through x, y list to obtain the distance of all suppressing points, and min will be the new r for the specific point. After trying out a bunch of values for the N_{best}, I decided to go with 500. The output of ANMS looks good as there are enough points and they are evenly spread around the whole image. (Example Figs 3, 4 are on the next page.)

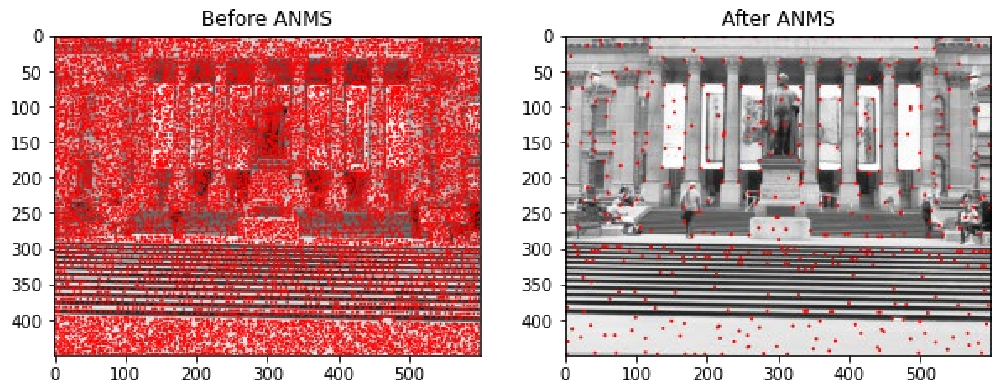


Fig 3. Result of applying ANMS on an image from set 1

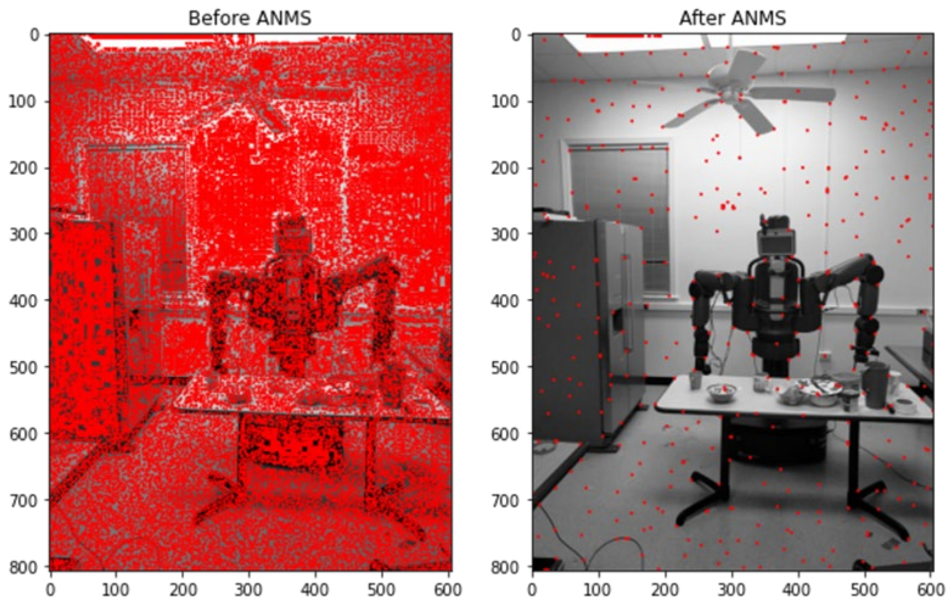
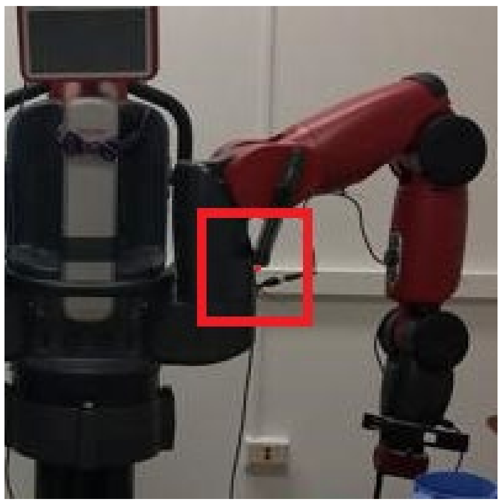


Fig 4. Result of applying ANMS on an image from set 3

2. Feature Descriptors

After obtaining the list of 500 corner points from the previous step, we need to generate each point a feature descriptor for later use. To avoid the problem from points near the edge, I duplicated the values in the matrix that is within 20 pixels from all four edges. For each point, I first take a 40 by 40-pixel area surrounding a point, applied a gaussian filter on it so the result is more general. I found setting the K_{size} to be 11 by 11 yields just the right amount of blurring where enough information is maintained. I then subsamples by every 5 pixels to get an 8 by 8 matrix and flattened it to get my 64-entry long descriptor. Each descriptor is then standardized since we only want to keep need the ratio. (Since 1D descriptor can't be plotted, below is an example of a descriptor without being flattened. The actual descriptors obtained from this step are all 1D 64-entry arrays. Fig 5 on the next page is a 2D representation of a feature descriptor.)



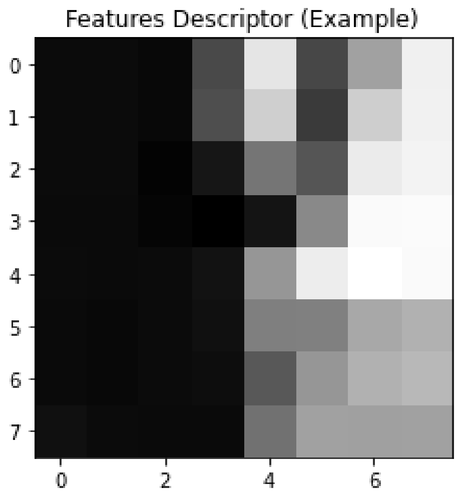


Fig 5. 2D representation of a feature descriptor (Right) from an image (Left) in set 3

3. Feature Matching

In this step, I simply loop through the list of 500 feature points and their corresponding feature descriptors and try to match feature points across different images. Both the value and the index of the smallest one (1NN) and the second smallest (2NN) will be recorded. To reduce the false-positive matches, the ratio $\frac{error_{1NN}}{error_{2NN}}$ will be calculated. The ratio should be small so that we know the only match is much better than all others in the array. I first set the threshold of ratio to be 0.4, which produces the least false-positive matches, but it also has a high chance of eliminating correct matches. Since we are doing RANSAC to eliminate false matches, I end up setting the threshold to 0.8.

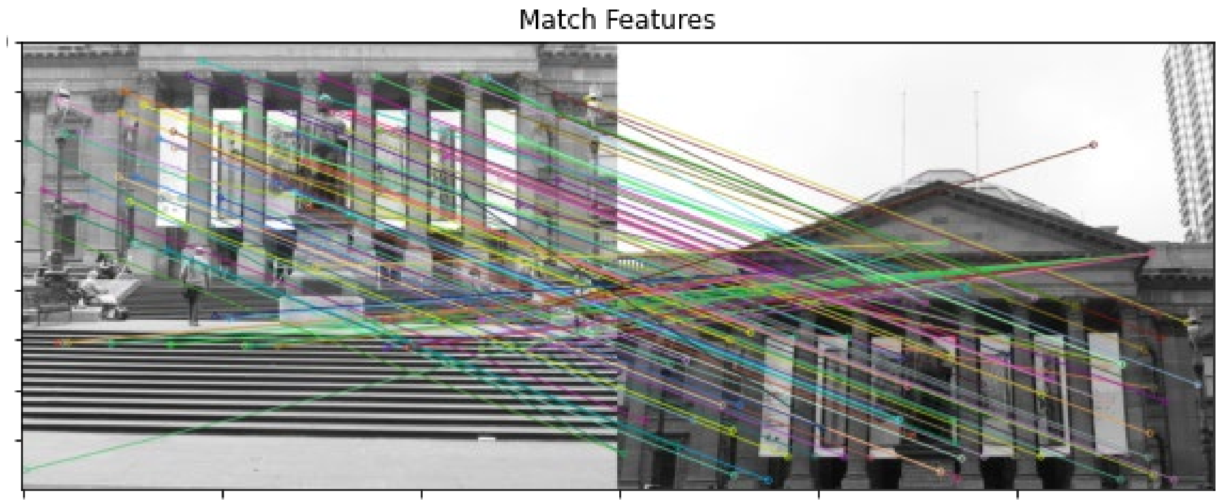


Fig 6. Result of feature match from two images in set 1

4. RANSAC (and Homography Estimation)

To keep only the correct matches for homography estimation, RANSAC was applied to the list of matched points. By looking at my results from the previous part, I think my portion of inliers should be well above 0.3, and I picked my iteration time to be 500, which yields a probability of missing a set of inliers to be $3.5e^{78}$. That should be more than enough for me to find a great homography estimation. Below shown is a comparison between matches without and with RANSAC applied.

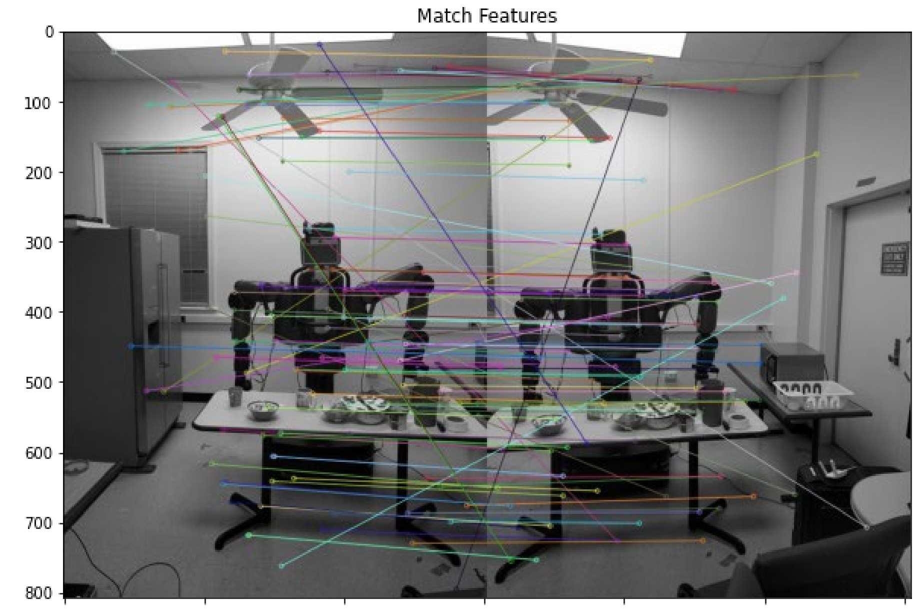


Fig 7. Result of feature match from images in set 3 (**Without RANSAC**)

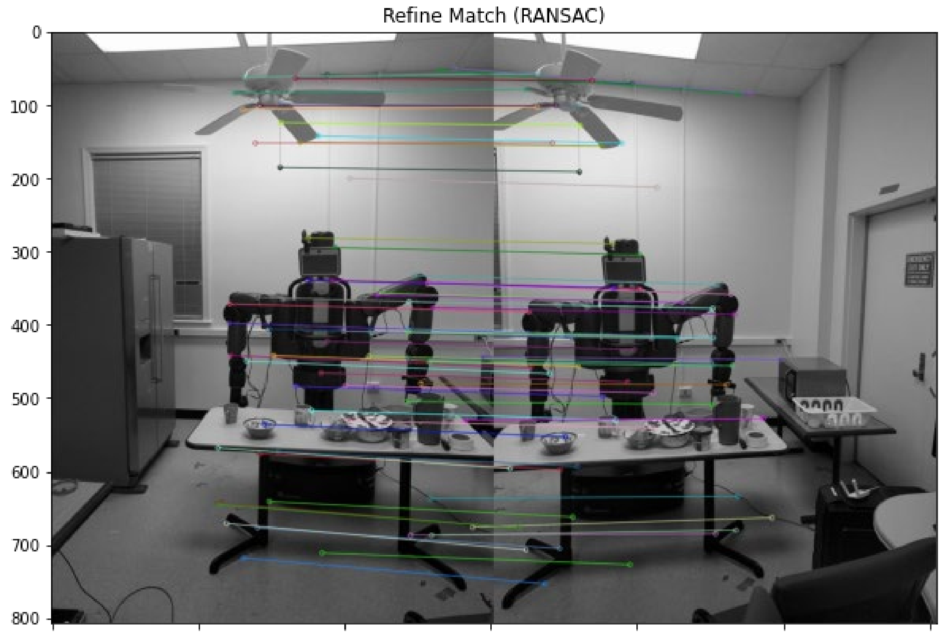


Fig 8. Result of refined match from same match above (**With RANSAC**)

5. Image Warping (and Blending)

The hardest task in this step will be finding the canvas size that will fit all images after transformation by applying the homography matrix. I made a function to find all four corners of the images after projection. We can then find the size of the canvas and the amount of shift that we need so that all images will fit in the positive domain.

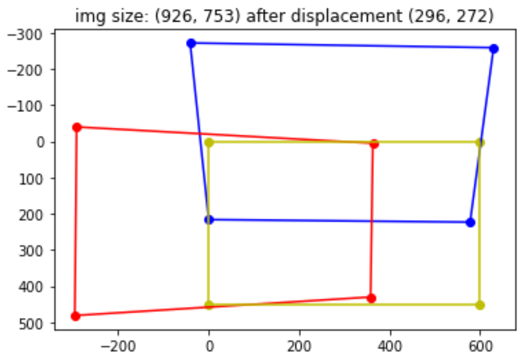


Fig 9. Projection of corners for images in set 1

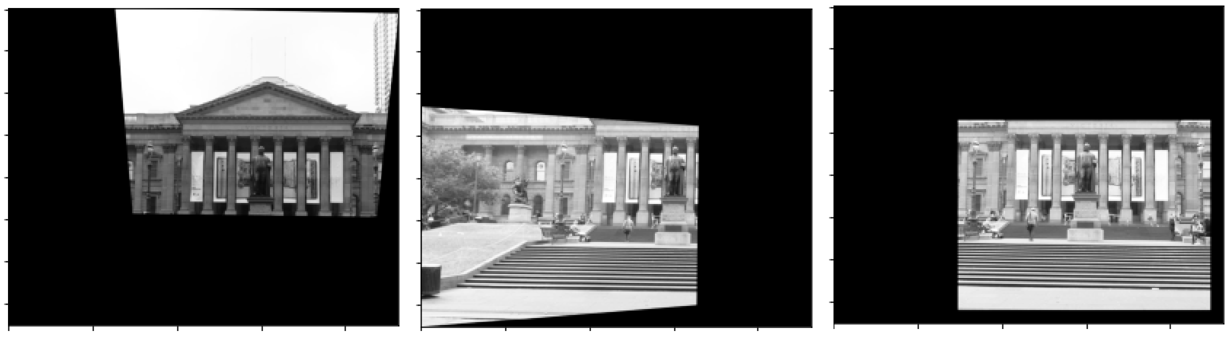


Fig 10. Projection of three images in set 1 with adjusted positioning

(The left, middle, right image each correspond to the Blue, Red, Yellow image in Fig 9)

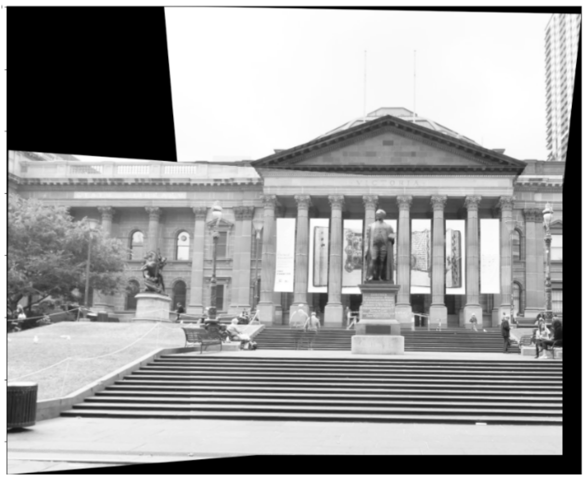
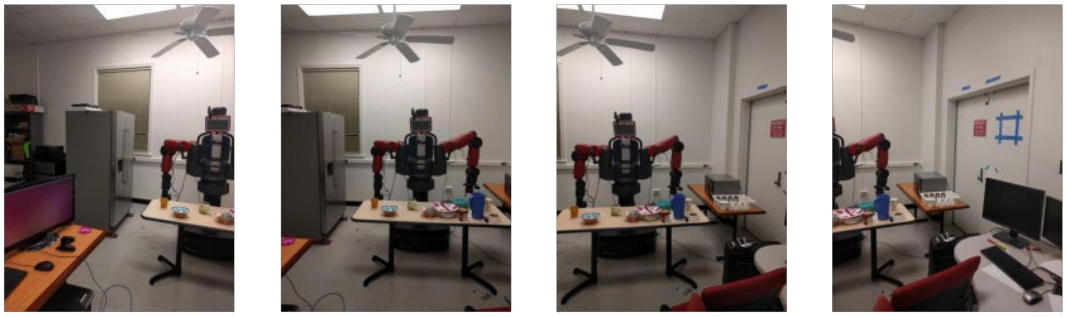


Fig 11. The output Panorama Image (The position matches what is shown in Fig 9)

When dealing with sets with four pictures, I found that not all pictures can be matched onto one single picture like what I did for sets with only three images. My solution is to match the middle two image first, and then treat it the same way as a set of three images. This solves the problem that the edge picture can't be matched on the center image.



Step 1: Match the middle two images.

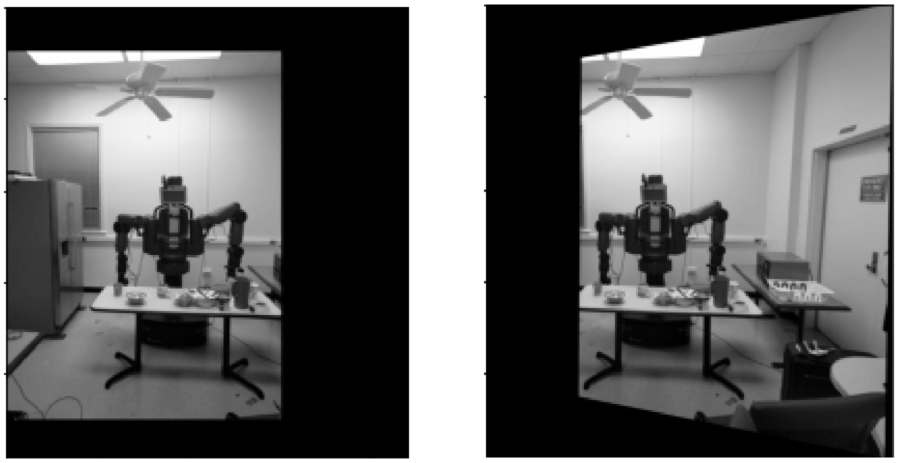


Fig 12. Projection of two middle images in set 3 with adjusted positioning

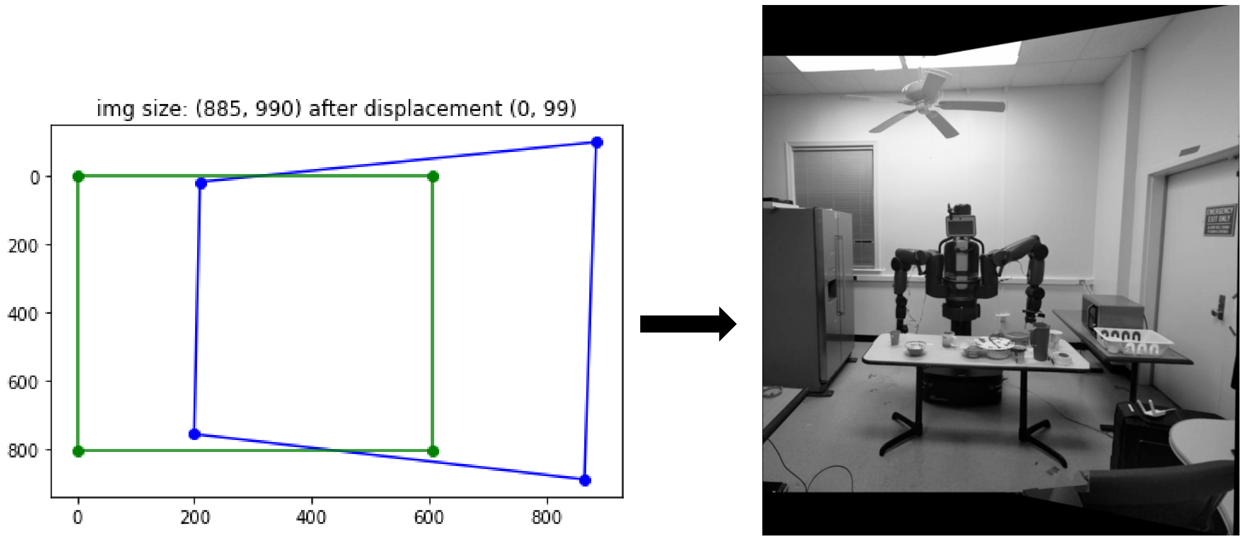


Fig 13. The output Panorama Image for two middle images in set 3

Step 2: Match the two side images with the middle combination output.

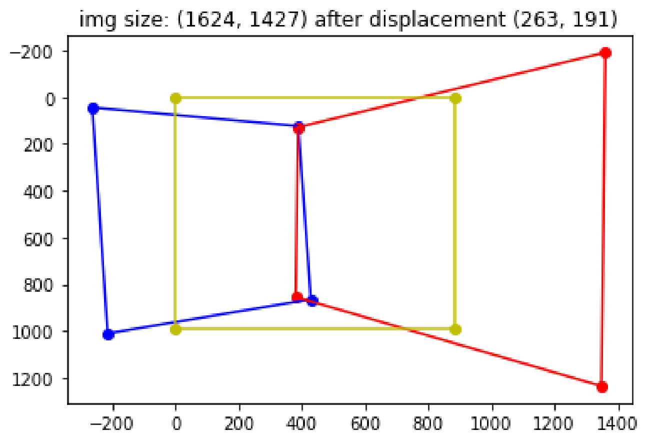


Fig 14. Projection of corners for two side images and the middle combination output image

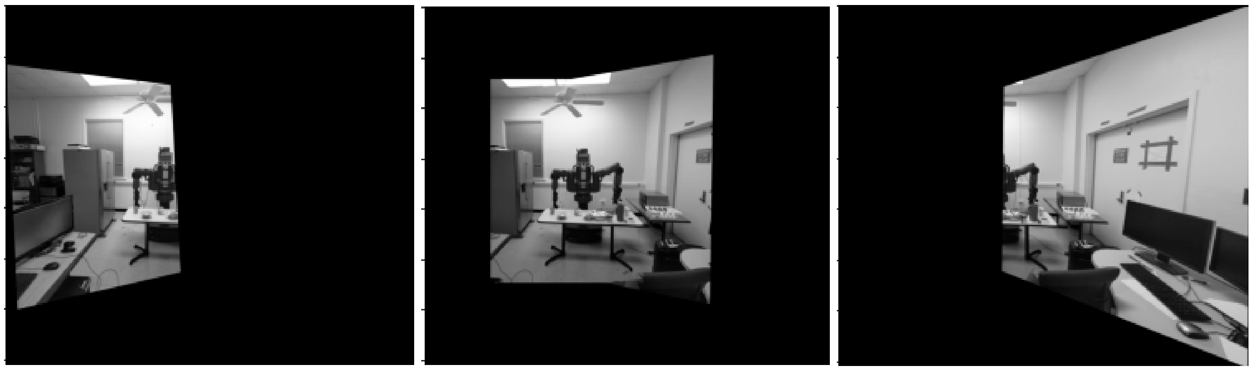


Fig 15. Projection of three images in set 3 with adjusted positioning

(The left, middle, right image each correspond to the Blue, Yellow, Red image in Fig 14)

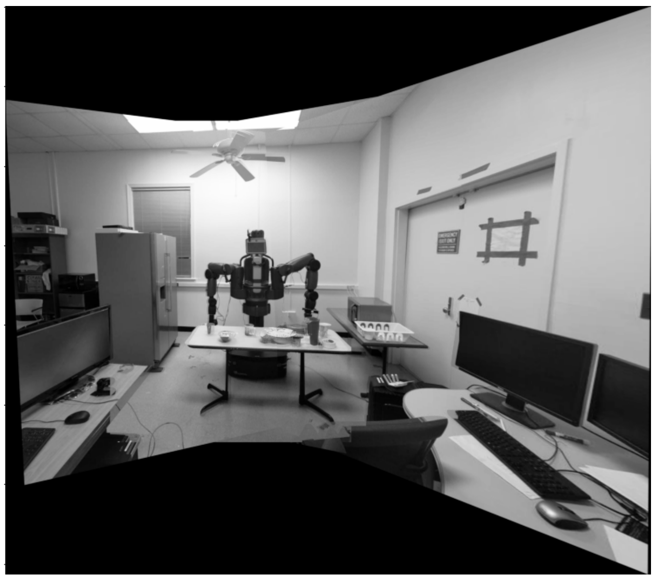


Fig 16. The output Panorama Image of four images in set 3

6. Output panoramas

6.1. set 1

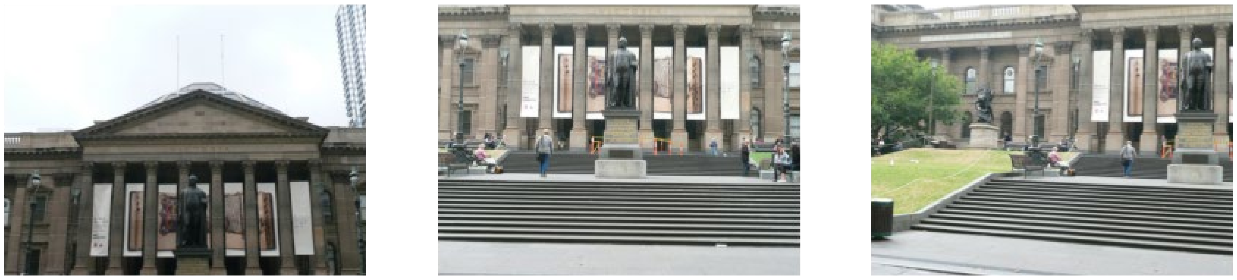


Fig 17. Input images from set 1

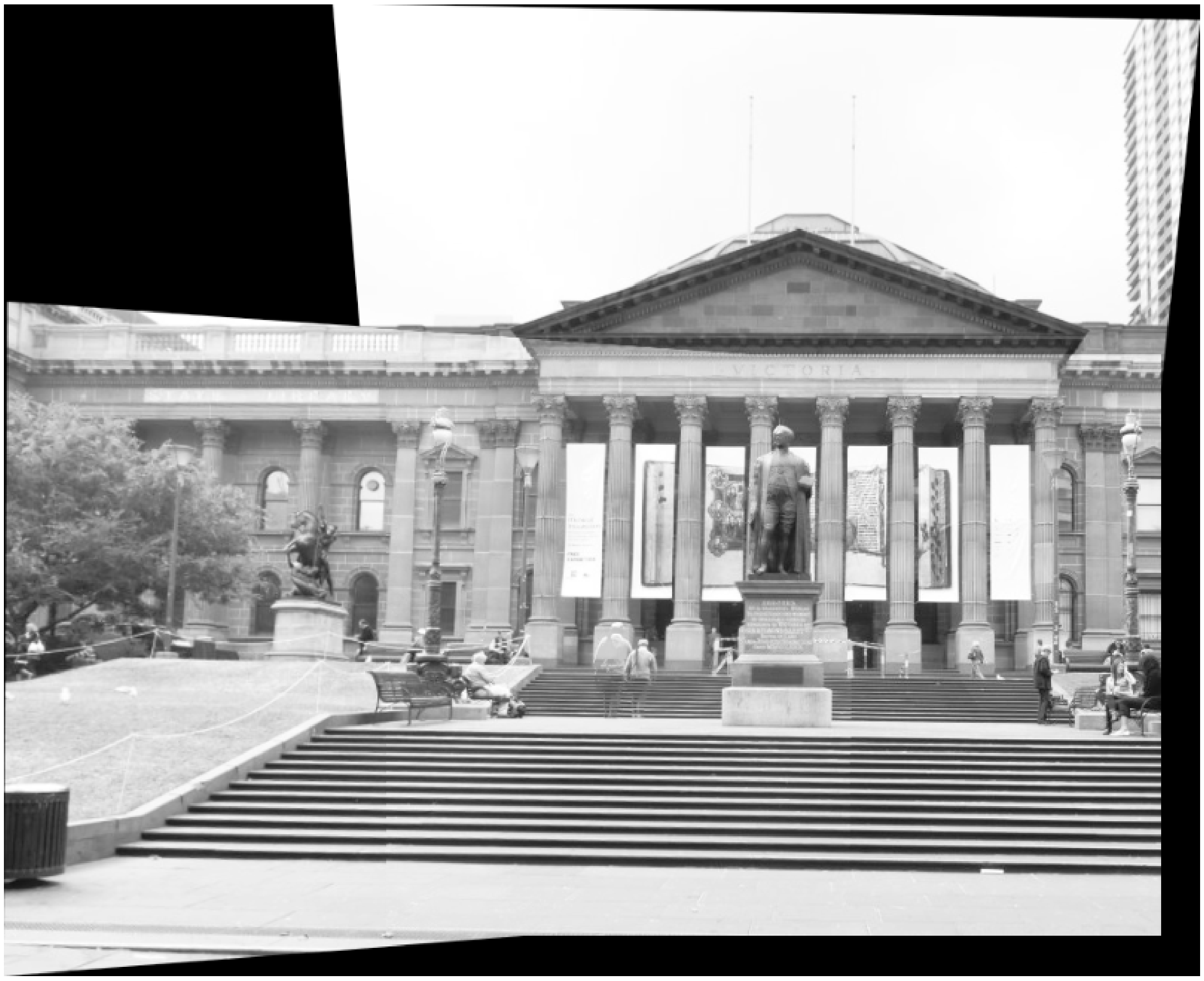


Fig 18. Output panorama image from set 1

6.2. set 2

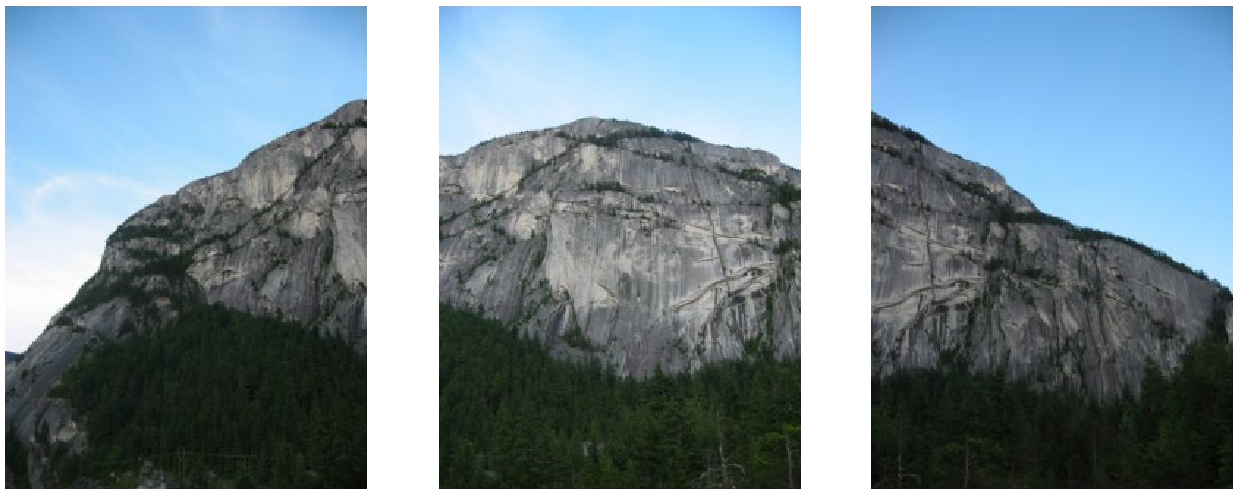


Fig 19. Input images from set 2

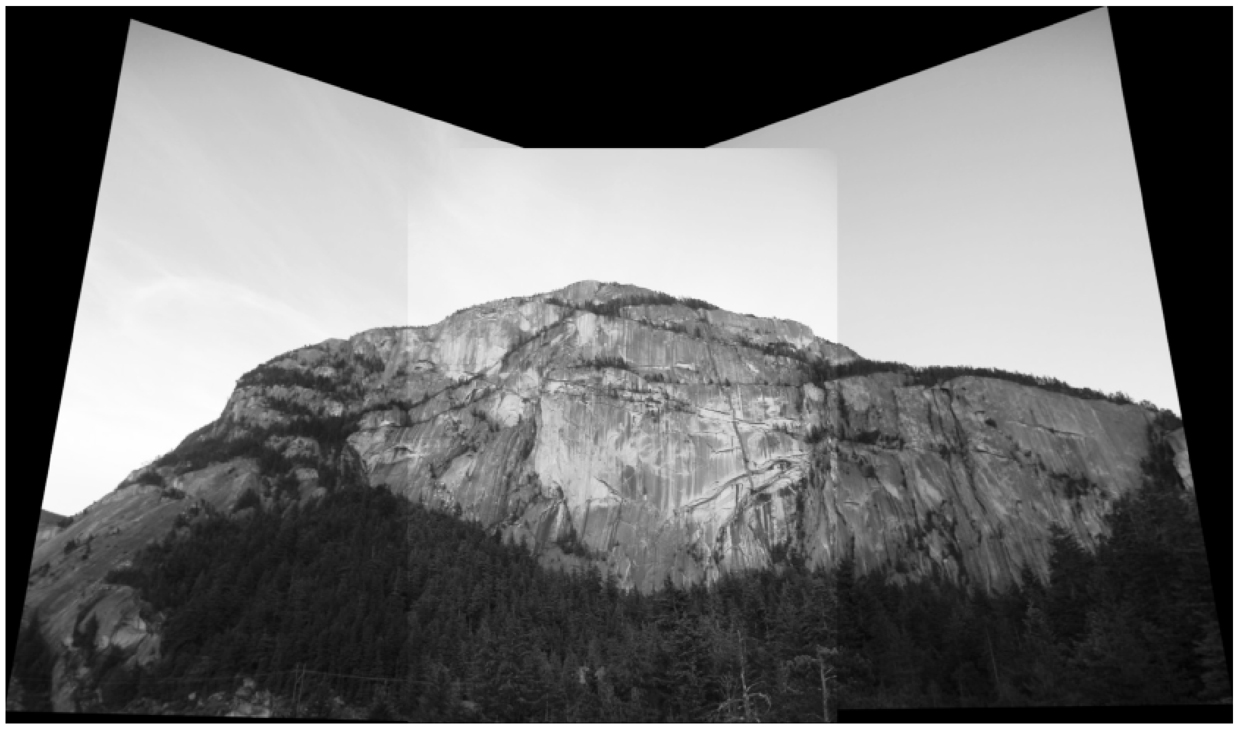


Fig 20. Output panorama image from set 2

6.3. set 3

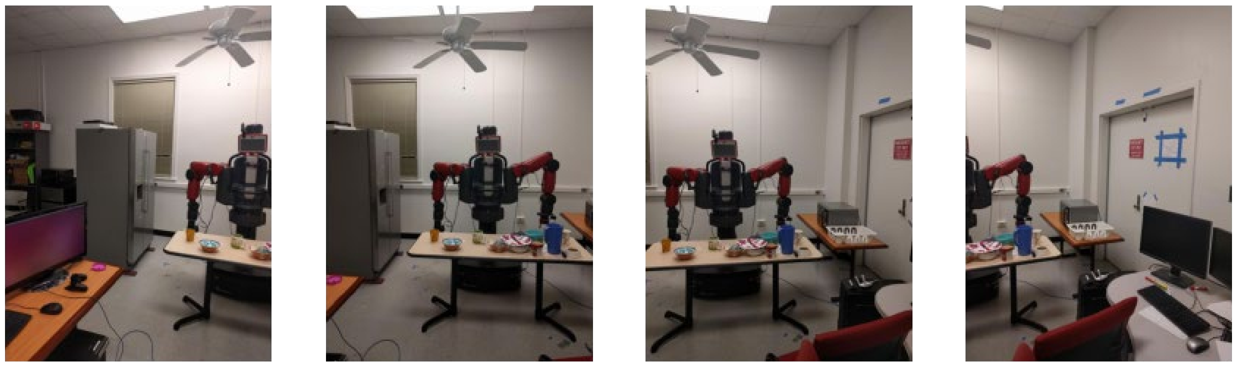


Fig 21. Input images from set 3

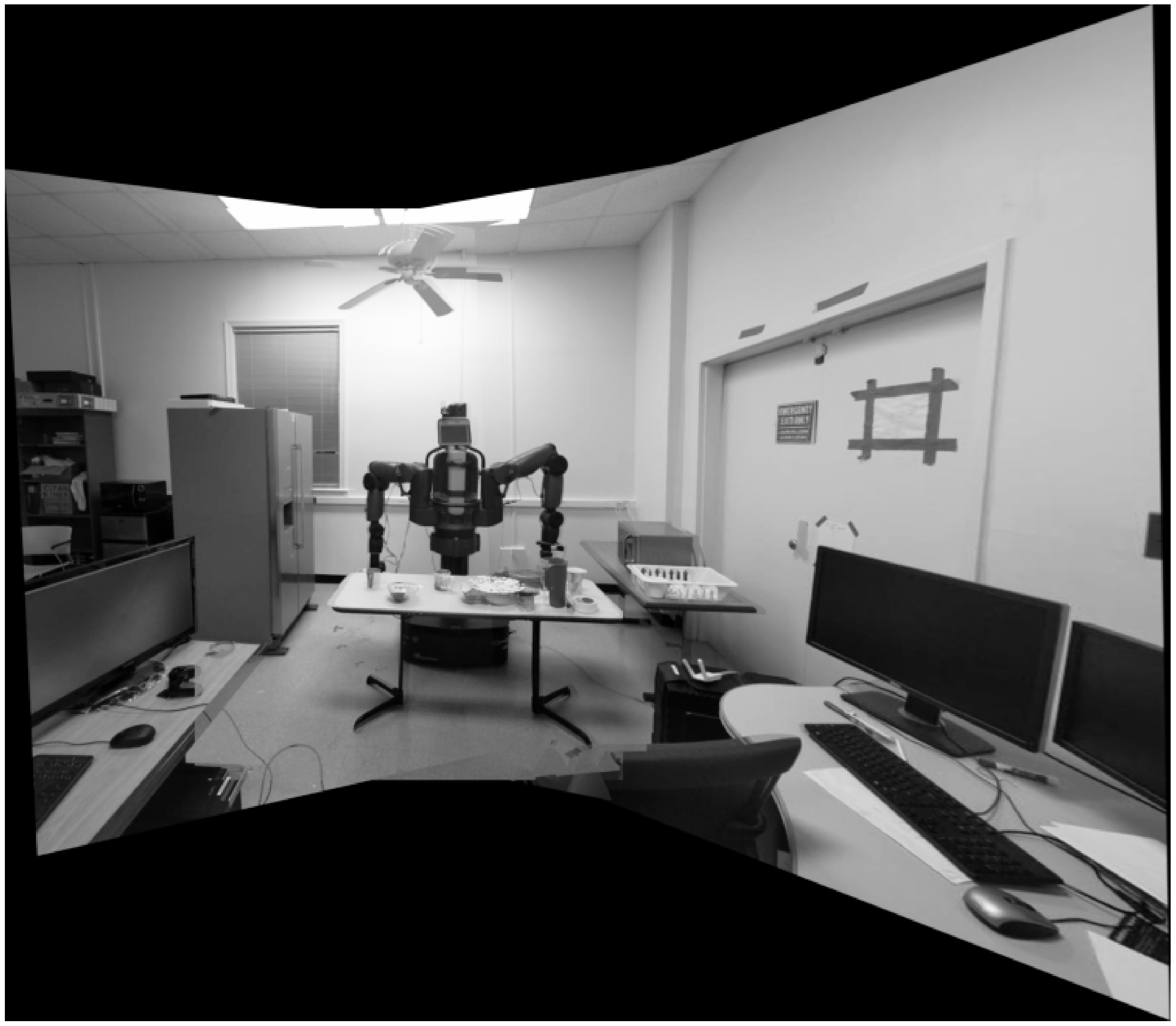


Fig 22. Output panorama image from set 3

6.4. test 1

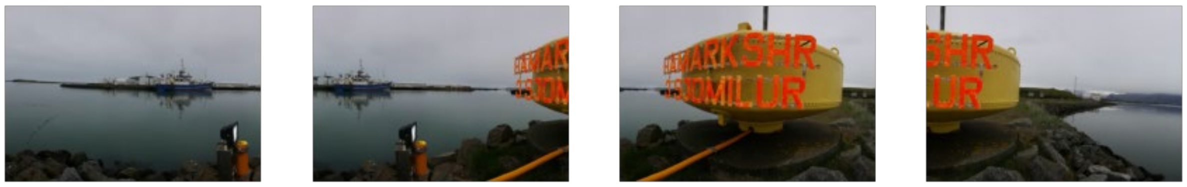


Fig 23. Input images from test 1

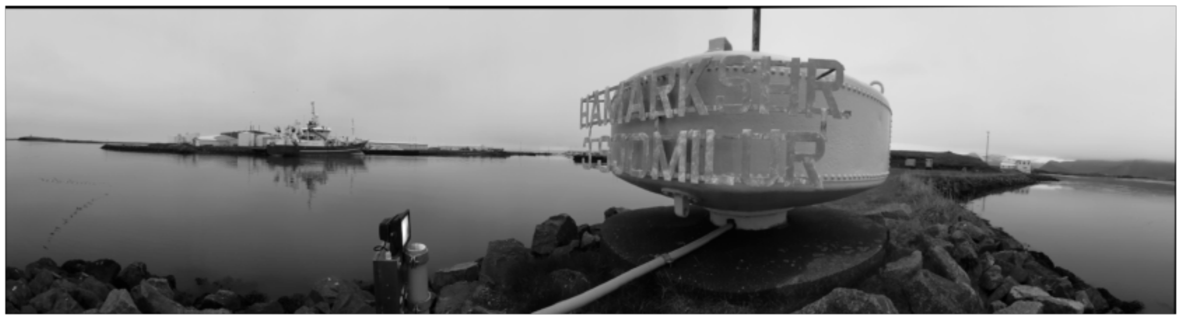


Fig24. Output panorama image from test 1

6.5. test 2

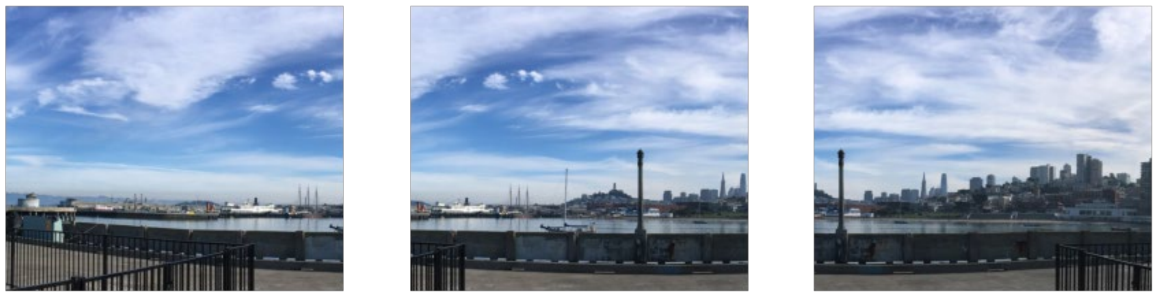


Fig 25. Input images from test 2

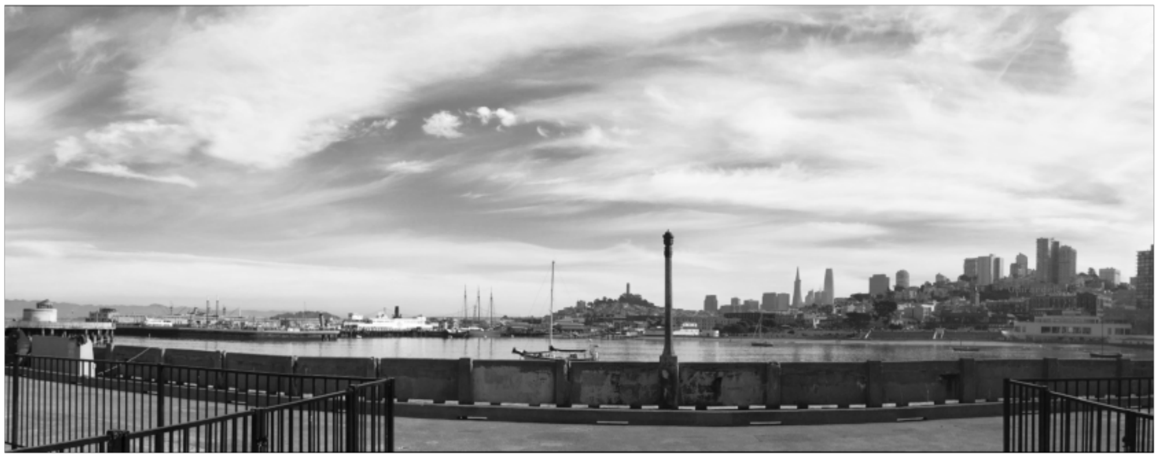


Fig 26. Output panorama image from test 2

6.6. Custom 1

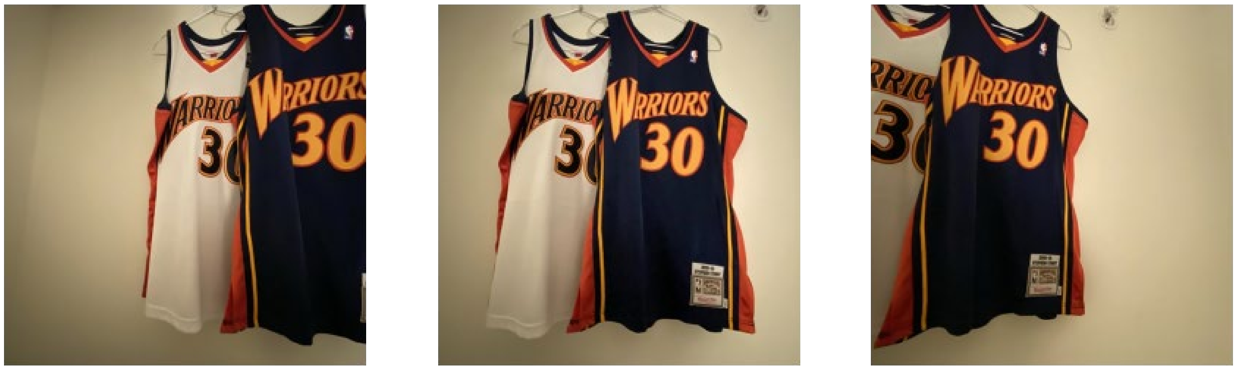


Fig 27. Input images from custom 1

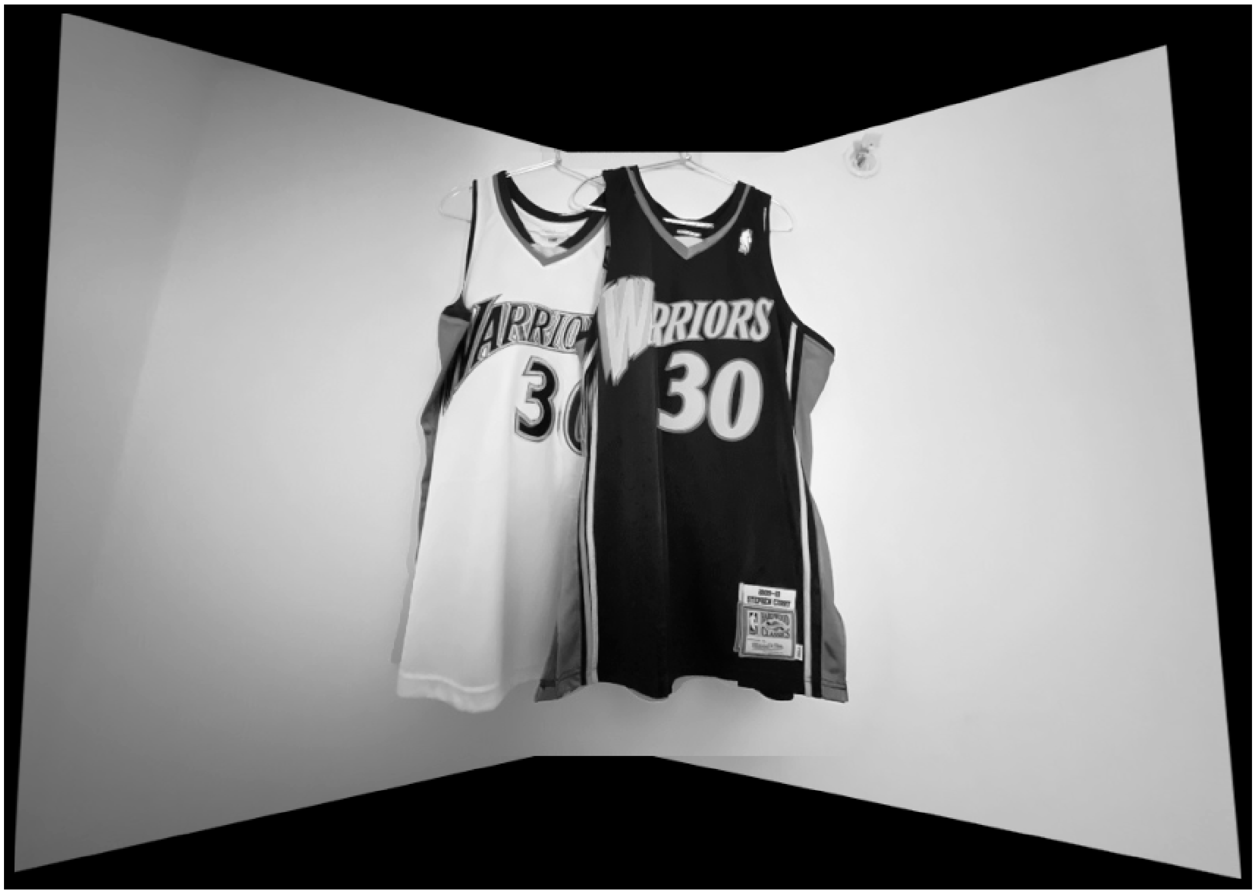


Fig 28. Output panorama image from custom 1

6.7. Custom 2

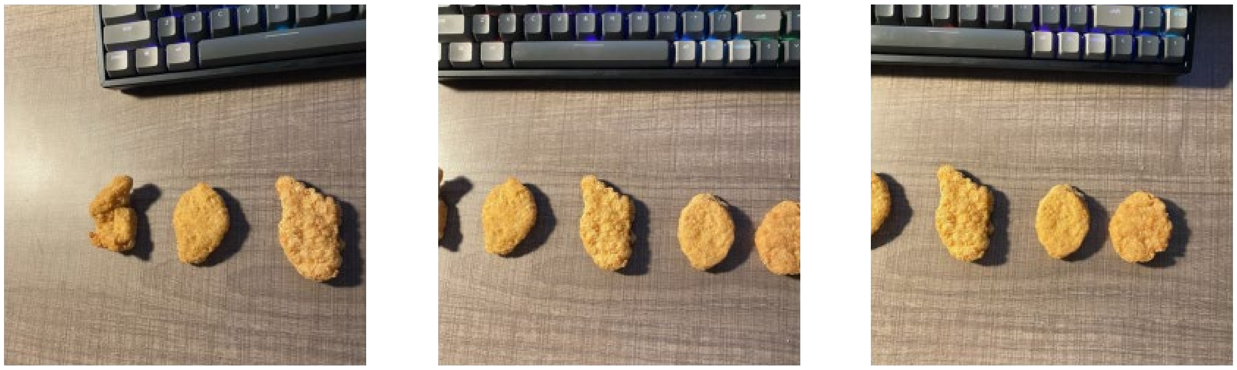


Fig 29. Input images from custom 2

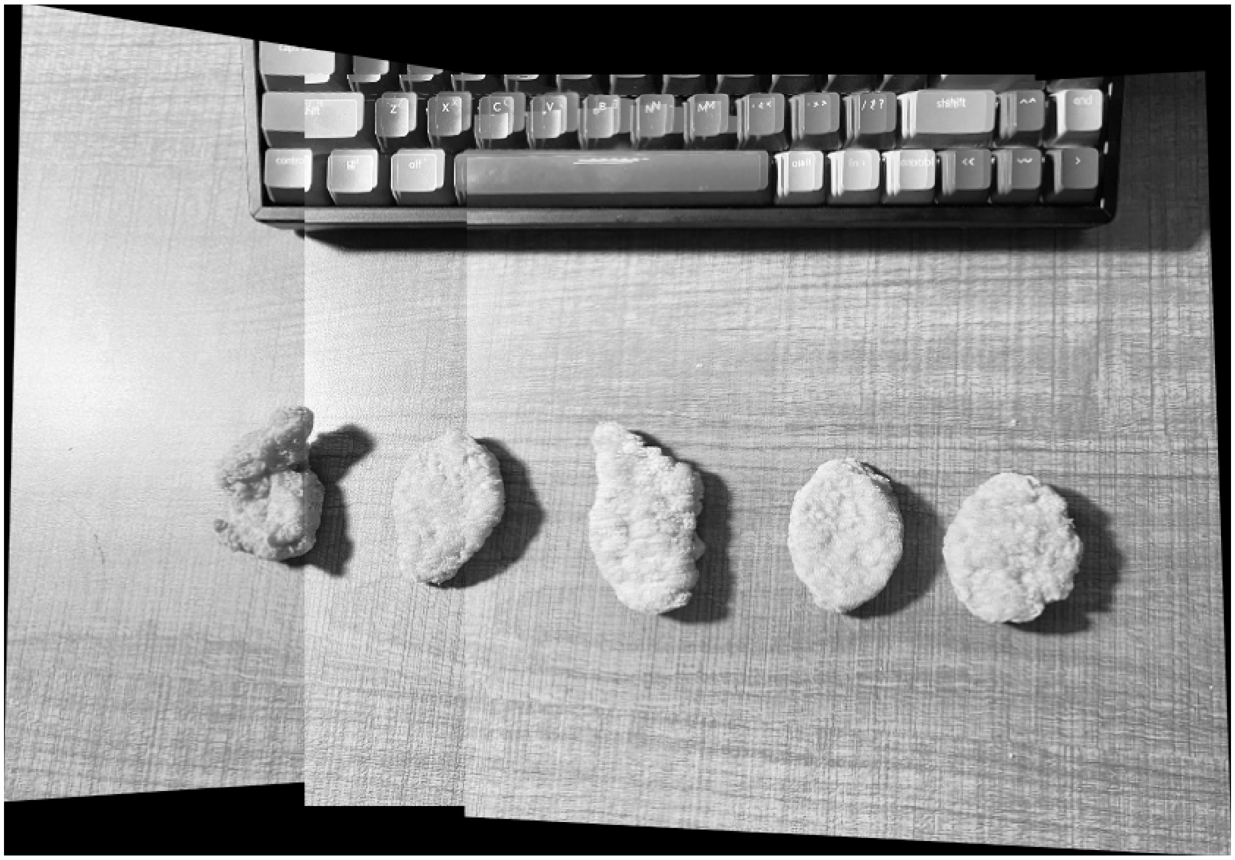


Fig 30. Output panorama image from custom 2