Computer Vision: Assignment 1

Panorama Stitching

By: Udit Vyas

Roll No: 18110176

Introduction

This assignment is about stitching multiple images of the same scene into one panoramic picture. This process of creating a panorama is describes briefly by the following steps.

Steps

- Key-points and Features Extraction from the images using ORB Algorithms (equivalent to SIFT) and feature matching
- Estimating the Homography matrix using RANSAC Algorithm
- Warping the images by applying the transformation using the estimated Homography matrix
- Blending the images to remove the artifacts

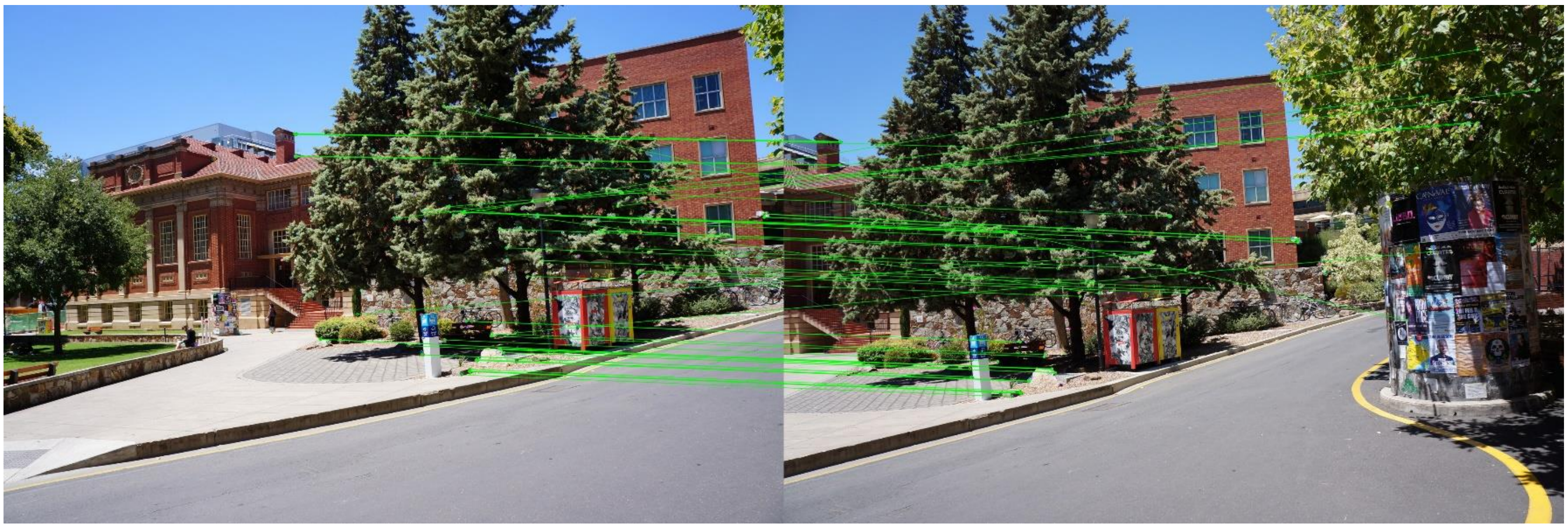
Implementation Details

This section contains the step-by-step procedure for each of the steps mentioned above. For better explanation, I have used one example to show visual results at every step.

1. Key-point and Feature Extraction and Matching:

In this step, according to the original paper, we extract the key-points and their corresponding descriptors from the all the images, using the SIFT algorithm. Since SIFT is not available in the open-source libraries, I have made use of Oriented FAST and Rotated BRIEF (ORB) for extracting the features and descriptors. (keypoints are marked as green dots in the fig. below)

After extracting the keypoints along with their descriptors, we take two images, and match the features using by finding the Euclidean distance between them. The keypoints with the shortest distance between them are matched between the two images. However, in order to make the matching more robust, I have made use of K-Nearest Neighbours (with K=2 by default) matching followed by Lowe's ratio check. Lowe's ratio is taken to be 0.8 by default. This process ensures that only the best matches are retained. After this process, the matching appears as in the figure below. (Notice the green lines showing the matched of keypoints)



The above process still leaves behind some outliers. These are removed in the upcoming step of calculating the Homography matrix between the two images.

2. Estimating the Homography Matrix using RANSAC Algorithm

Based on the matched keypoints that we generated from the previous step, we now try to estimate the Homography matrix. However, the presence of outliers in the matchings can lead to inaccuracies in the calculation of the matrix, which will result into inaccurate stitches. Thus, we use the RANSAC algorithm. Here, we randomly keep *sampling 4 matchings* from all the matches, and try to estimate a Homography matrix using *Singular Value Decomposition*.

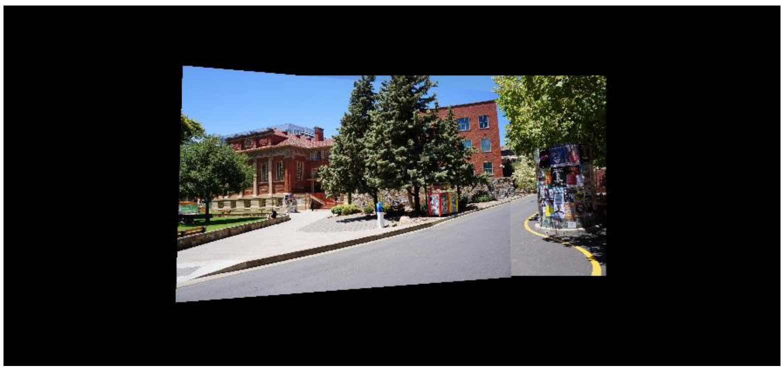
For this calculated H matrix, we find the error for all the matches, after transforming them with H, and add the points as inliers, if the error rate is lesser than some pre-decided threshold (set as 2 in this assignment). We keep repeating this process (for 1500 iterations in this assignment), and simultaneously keep track of the H, with the maximum number of inliers. Thus, at the end of the process, we have the final H matrix for the given two images.

Using this technique, we find the Homography between various pairs of images. However, while transforming all images in the reference pair of the chosen reference image, it is possible that the images towards the extreme of the panorama do not have matching key-points as the images *may not have direct overlapping regions*. In such cases, we have to find intermediate Homography matrices, and *take their products on the way*, to find the required H matrix. For example, H42 = H43 * H32.

Now that we have the H matrix for all the images, which can transform them to the plane of the reference image, we move to the next step.

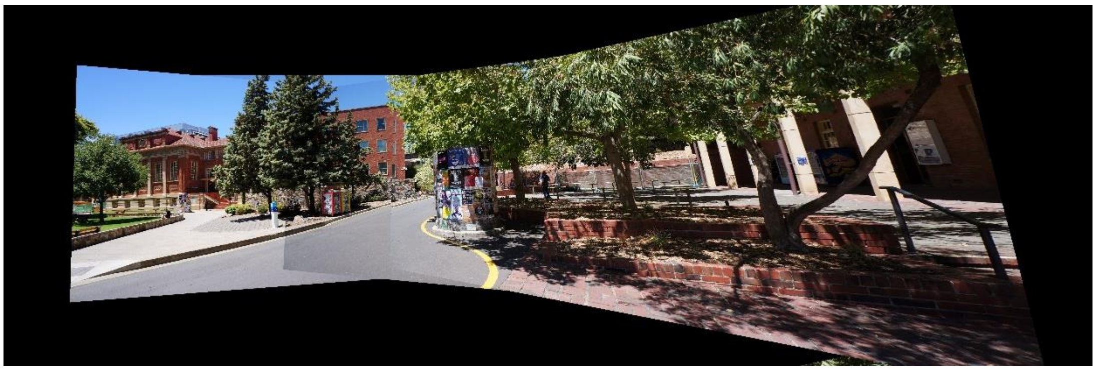
3. Warping the images (No Blending)

In this step, we actually transform the images by multiplying each pixel in the image, with the corresponding Homography matrix. This process will change the perspective of the image, and add the image over the reference image. The transformed pixel value is assigned to a small neighbourhood of the output image, rather than just one pixel, so as to avoid the problems of missing data (This is a naïve approach of interpolation). Warping just one image will appear as in the fig below –



As can be seen in the figure above, there are clearly two images present, the base or the reference image, and the transformed (warped) image.

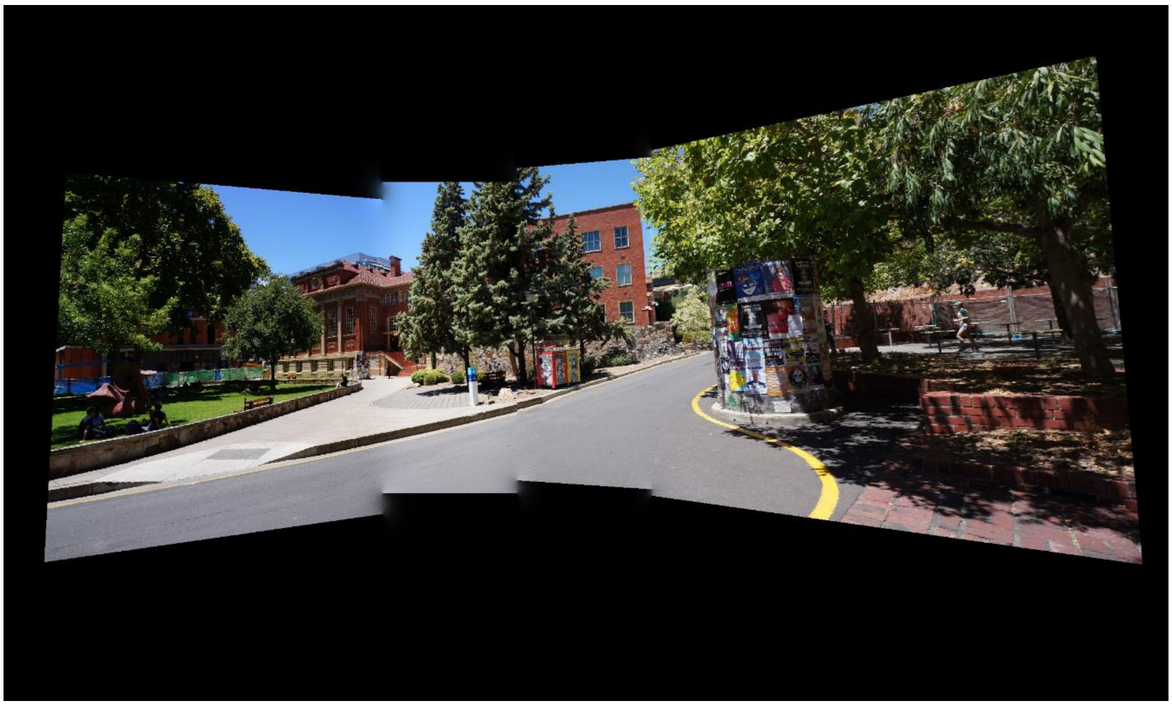
Repeatedly doing this step for all the images (4 in this case) to be stitched yields the following stitched image as a result –



As it can be seen, the panorama in now complete! However, there are thin lines that can be observed on the stitch boundaries due to the difference in the exposures of the two images. This is fixed in the next step.

5. Blending the images to remove artifacts

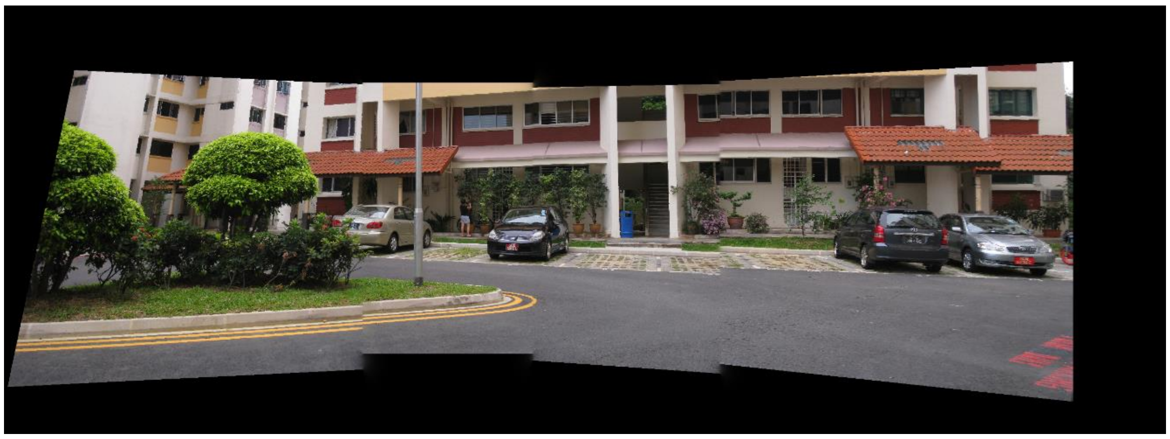
In order to remove the edges between the images, we use the technique of image pyramids. As we cumulatively keep stitching the images into the final images, with each new warped image, we find the overlapping region with the stitched image of the previous step. Then, we find the Gaussian and Laplacian pyramids of both these images. These images are then reconstructed with a weighted Mask addition. The mask is a binary mask, which decided the contribution of the image pyramid components while reconstructing the images. The result of the blended image is as below.



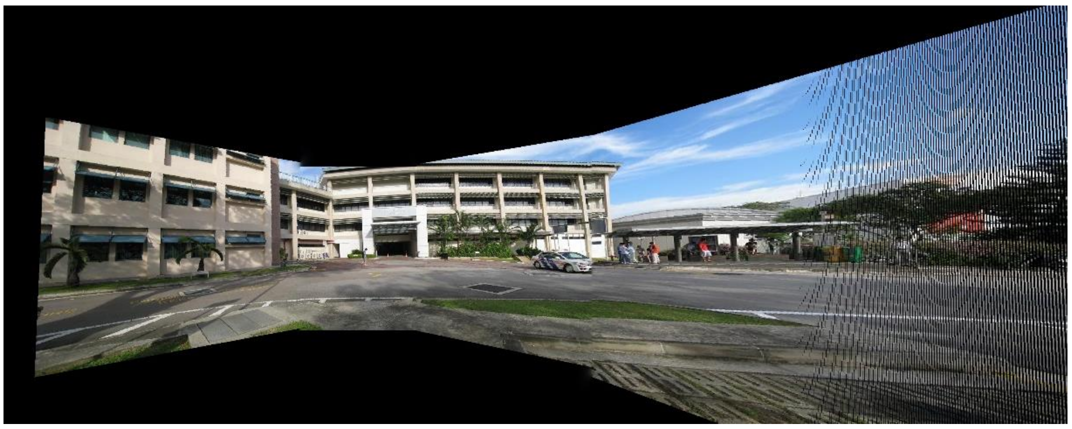
Implementation Results

Please note that for the general code, the underlying assumption is that the pictures given shall be in an order describing the scene from left to right, and are taken at regular intervals such that they have common features that can be matched with the adjacent picture. In some of the set of images provided in the assignment, the pictures are not following the same sequence as mentioned above. In this case, in order to get the best results, it is necessary to have re-ordering in the Homography matrix calculations, and warping functions. All such changes from the default setting are mentioned along with every case.

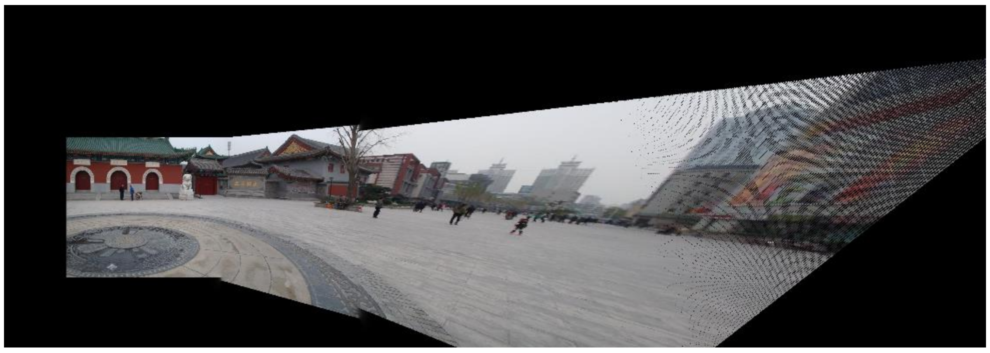
1. Set 1 (4 Images) [STC_0033.JPG, STD_0034.JPG, STE_0035.JPG, STF_0036.JPG]



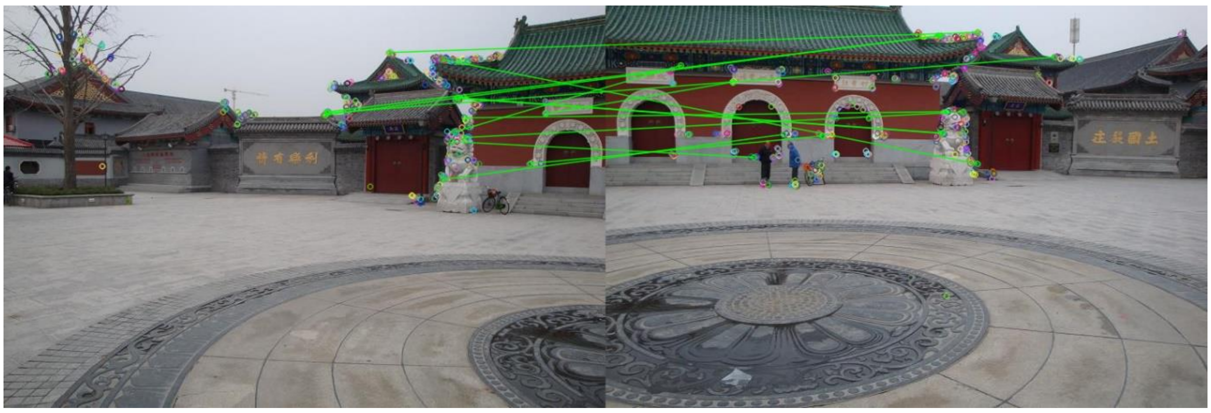
2. Set 2 (4 Images) [2_1.JPG, 2_2.JPG, 2_3.JPG, 2_4.JPG]



3. Set 3 (3 Images) [3_3.JPG, 3_4.JPG, 3_5.JPG]

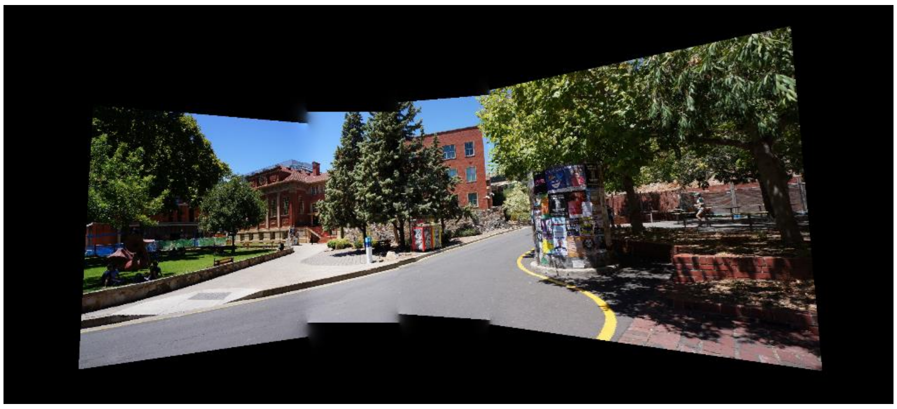


In set 3, the images 3_2.JPG and 3_3.JPG are posing a problem for the algorithm. Below attached is the feature matching for these images.

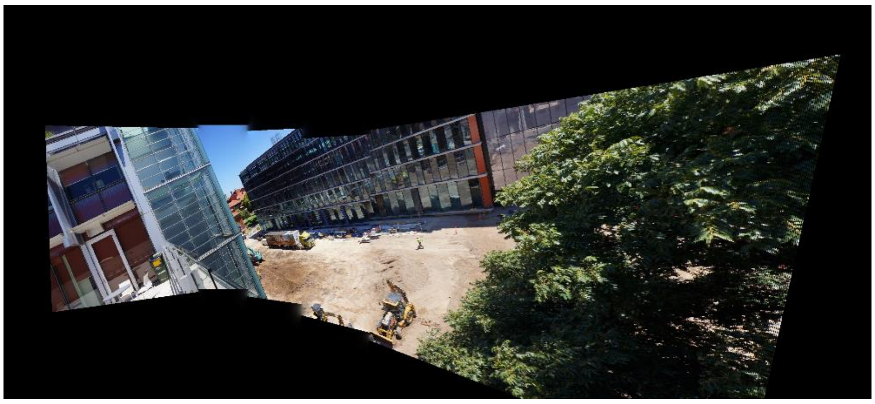


As it can be seen from the image, the features off the building are symmetrical, and the pictures are not very close to each other. Therefore, the algorithm connects incorrect features, that seemingly are similar to each other due to symmetry. This problem can only be solved if there more continuous shots of this building. Since there are only 5 images provided, and two middle images are problematic, therefore, only 3 images could be stitched in this case. Therefore, please comment the code appropriately to avoid errors, while running SET 3.

4. Set 4 (4 Images) [DSC02931.JPG, DSC02932.JPG, DSC02933.JPG, DSC02934.JPG]

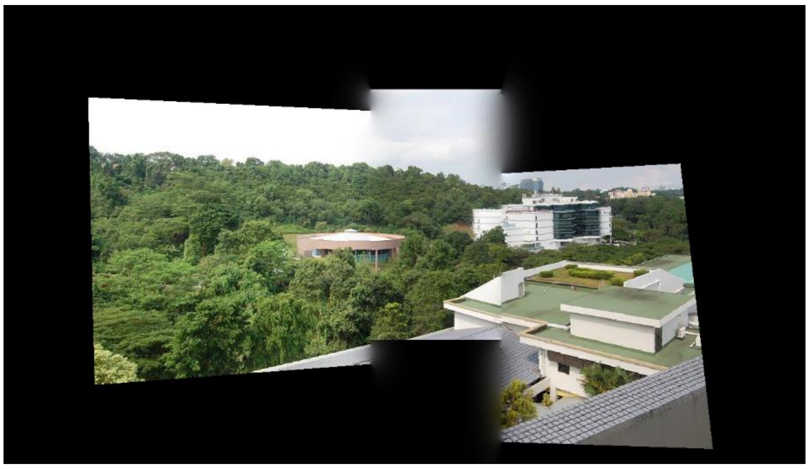


5. Set 5 (5 Images) [All images]



6. Set 6 (5 Images) [1_1.JPG, 1_2.JPG, 1_3.JPG, 1_4.JPG, 1_5.JPG]

The images in this case are provided in a different order than the usual, therefore, while blending them, we get some unusual borders. The sequence of the images may be altered, to create relatively better images. However, with complex arrangements, the blending becomes tricky, as can be observed from the output.



There are two underlying assumptions in the code –

- There are 4 images provided as input, and they are all adjacent to each other (unlike the SET6 case).
- The images contain some overlapping region so that there can be matching SIFT features.

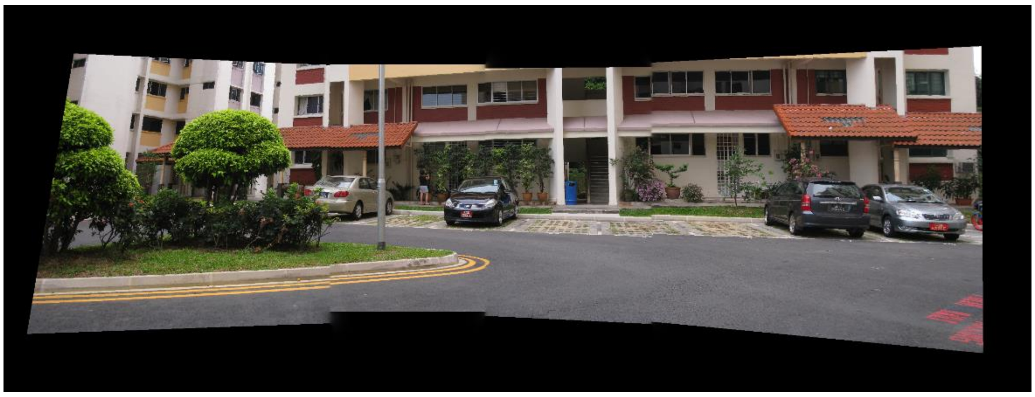
Exceptions:

- In SET3, since I have stitched only 3 images, one must comment the stitching and blending of image1, to avoid errors.
- In SET6, since the arrangement of the images in the actual scene is different than the assumption mentioned above, the last warp function (mywarp_far) must be replaced with its variant (mywarp). The code is currently present commented. It must be appropriately switched to get correct result.

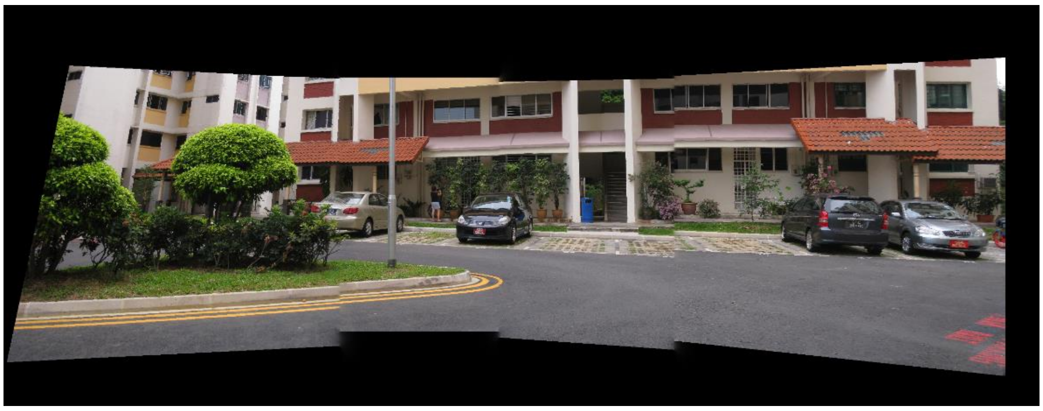
Results of Inbuilt Functions and Comparison

Set 1:

Using Inbuilt Functions:

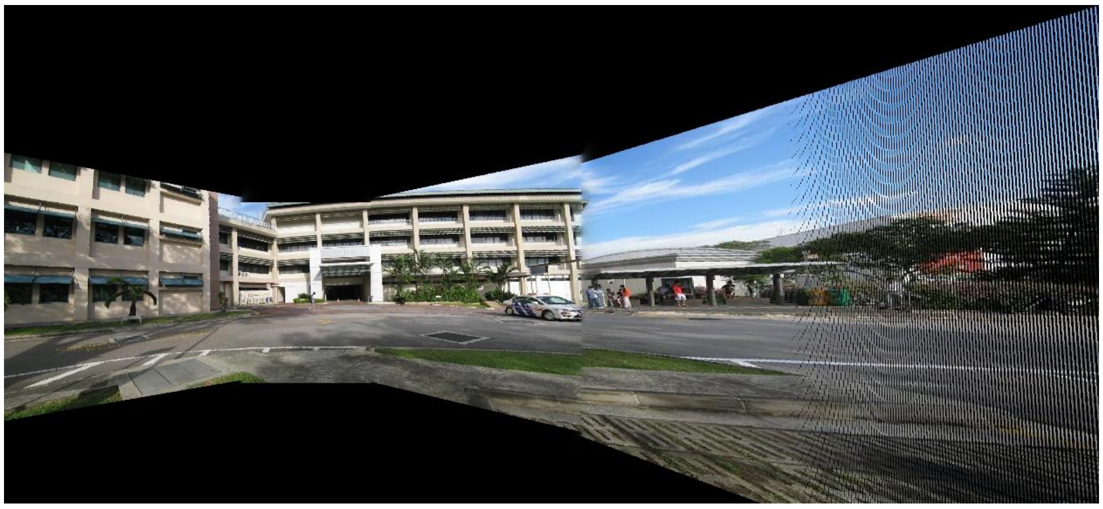


Using self-made function:

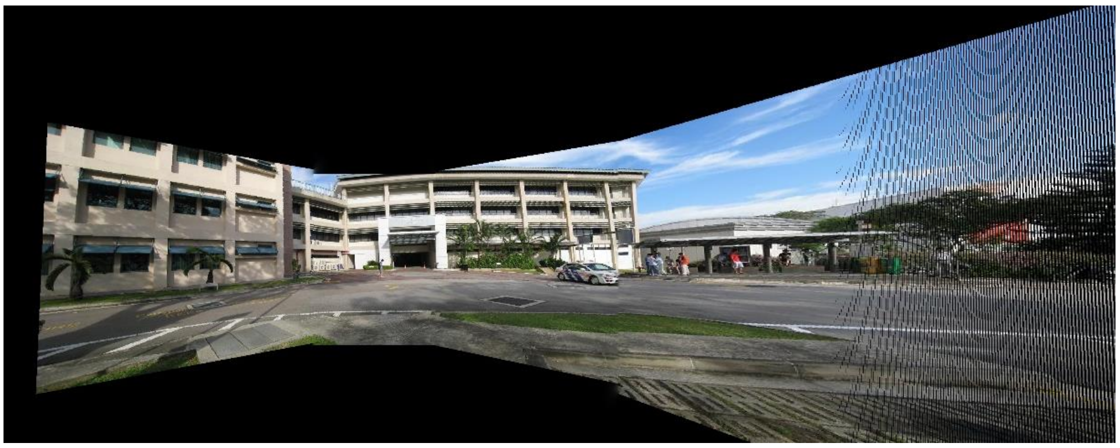


Set 2:

Using inbuilt functions:



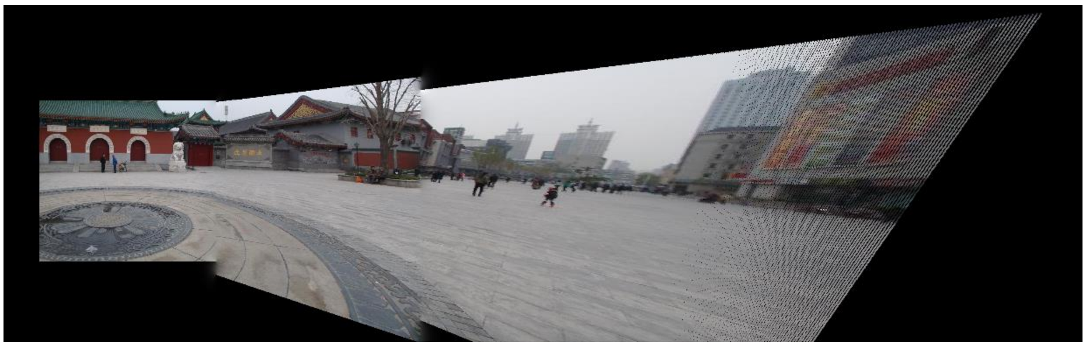
Using self-made functions:



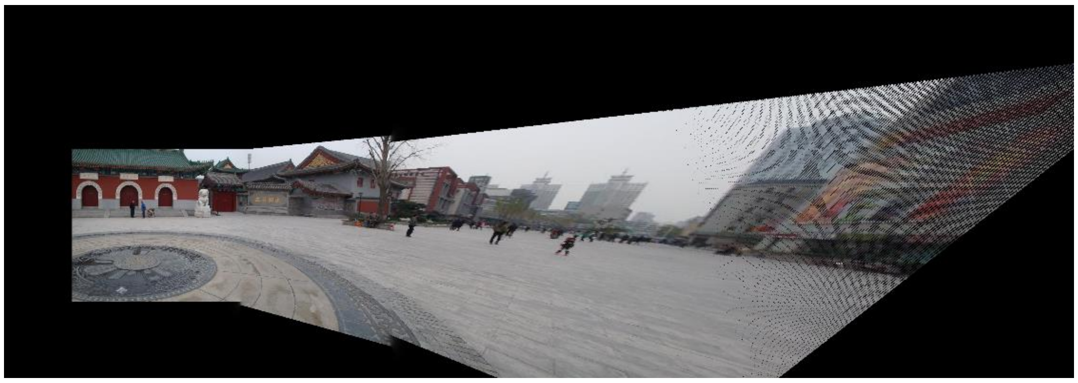
Set 3:

Note: Even with inbuilt functions, it is not able to stitch Image 2 and Image 3, as mentioned previously due to wrong features (symmetrical buildings)

Using in-built Functions:

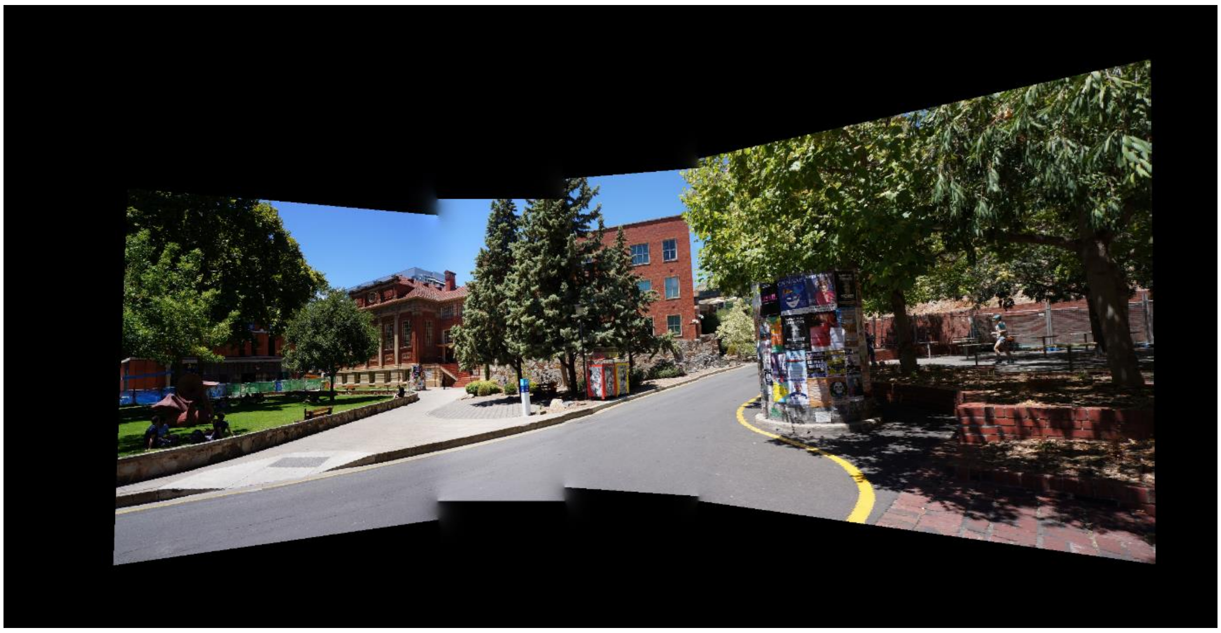


Using self-made functions:

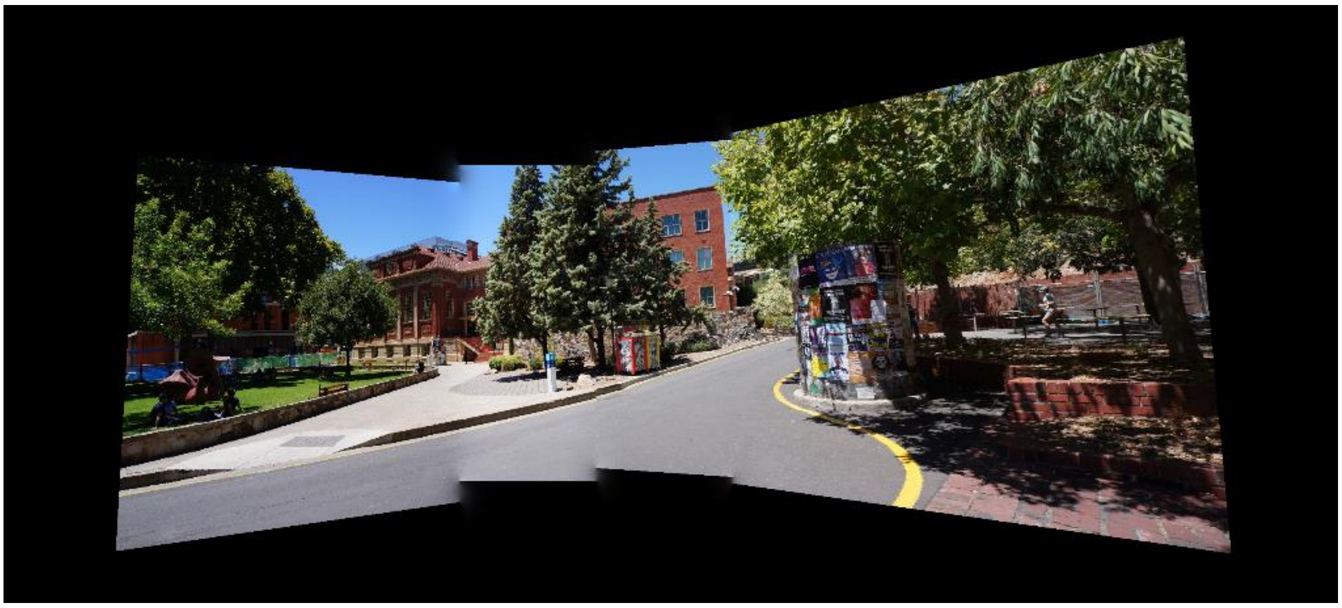


Set 4:

Using inbuilt functions:

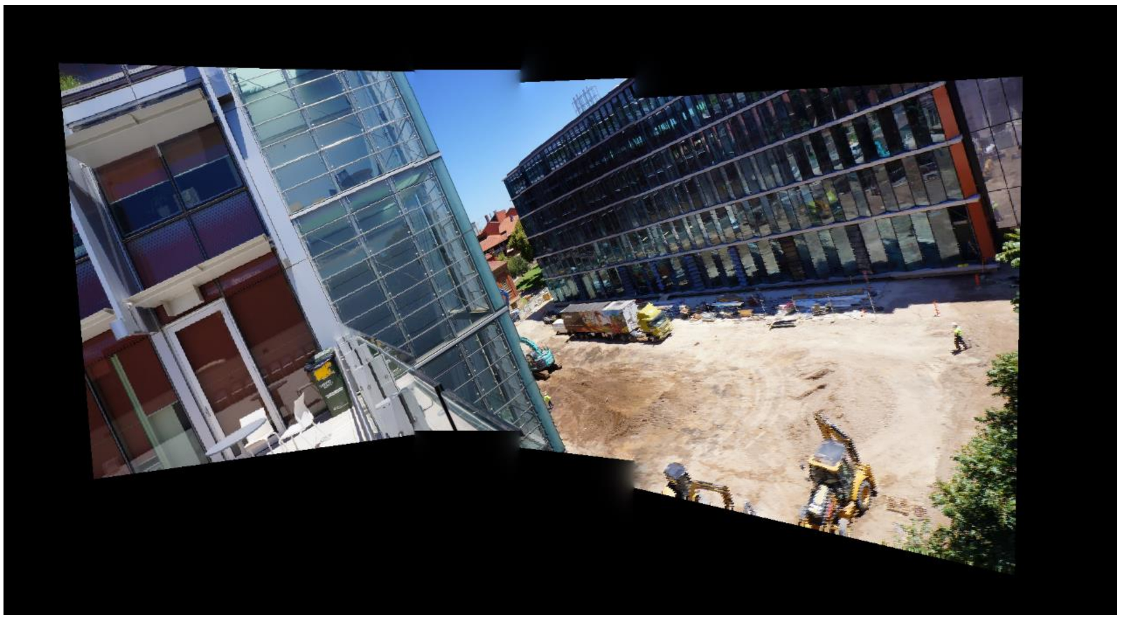


Using self-made functions:

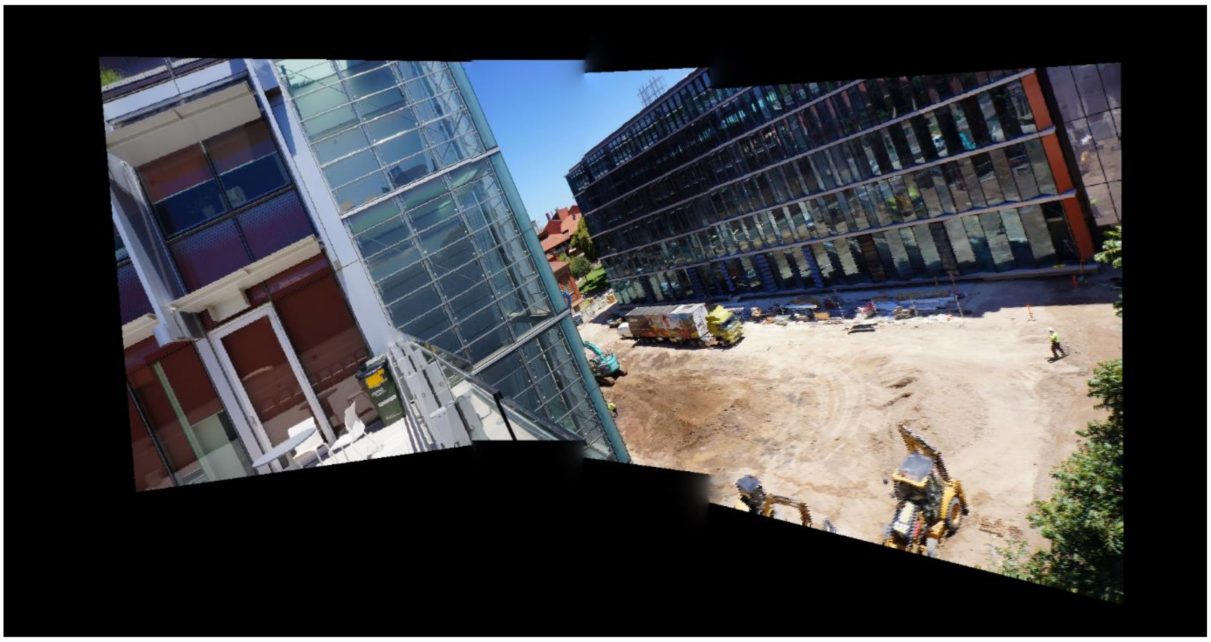


Set 5

Using inbuilt Functions:

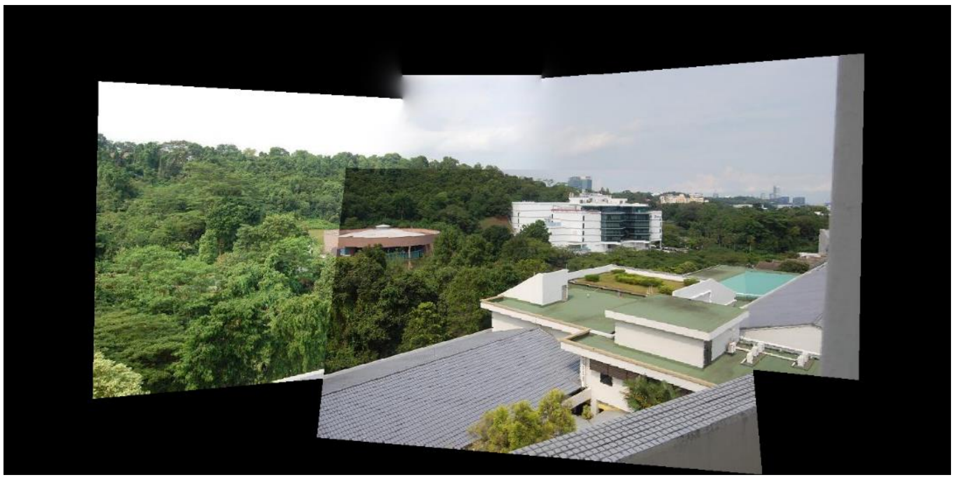


Using self-made functions:



Set 6

Using inbuilt Functions:



Using self-made functions: