CMSC733 Project 1:My AutoPano

Param Dave
Masters in Robotics
University of Maryland, College Park
Email: pdave1@umd.edu
Using 2 late days

Xiangyu Liu
Phd CS
University of Maryland, College Park
Email: xyliu999@umd.edu
Using 2 late days

***Abstract*—The project AutoPano is divided into two phases. First phase is using the classical approach to stiching multiple images using key feature point detection. It involves is to stitch two or more images in order to create one seamless panorama image. Phase two deals with using Supervised and Unsupervised learning approaches to estimate the homography between two images**

I. Phase 1

Phase 1 of the project involves panoramic stitching of N images. It involves the following steps. First computing key features in each image. This can be computed using corners/edges or higher level algorithms like SIFT to detect key feature points in each image. Secondly we use Adaptive Non Maximum Suppression (ANMS) to uniformly distribute the key feature points in the image and suppress multiple repetitive features in local minima. Third we create a feature descriptor process it to create a 64x1 vector and create fature matching where we match all the features of two images. Fourth we perform Homography estimation using RANSAC and finally Wrap and blend the images. The assumption here is that there is about 60-70% overlap in the two consecutive images. The Figure 1 shows overview of Phase 1.

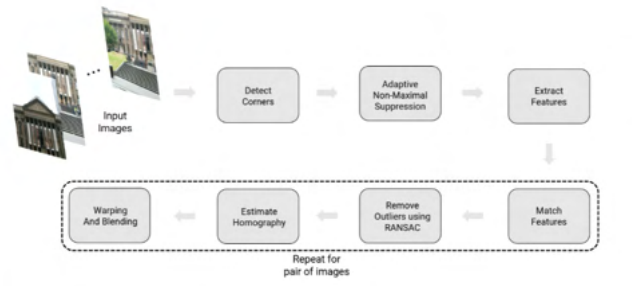

Fig. 1. Overview of Panaroma Stiching

A. Corner Detection

Corner Detection is the first step in this project. Harris corner detection , Shi-Tomasi detection are two methods available to us. Harris corner detection is more popular but in our case Shi-Tomasi gives more uniform results. We use **cv2.goodFeaturesToTrack()** to detect corners and save them. The output Harris corner detection is shown in figure 2 . The output of Shi-Tomasi corner detector is shown in figure 3. The output of ANMS applied on Shi-tomasi corner detector is shown in figure 4

B. Adaptive Non Maximal Suppression(ANMS)

In this step we perform Active Non Maximal Suppression to obtain local maximum feature.In a real image, a corner is never perfectly sharp, each corner might get a lot of hits out of the N strong corners - we want to choose only the N_{best} best corners after ANMS. In essence, we will get a lot more corners than we should. ANMS will try to find corners which are true local maxima. The algorithm is implemented as shown in figure 5.

We used ANMS on Harris Corner detection result as well as on the output of Shi-Tomasi corner detector. Top 300 values were kept and rest were discarded.The output of ANMS is shown in figure 4 .

C. Feature Description

Next step is to describe all the corner points by a feature vector.This is like encoding the information at each feature point by a vector. For this, a patch of 40x40 is taken around that point and is sub-sampled to 8x8 and blurred using Gaussian Bluring. The output is then reshaped to 64x1 vector and standardization is done to get mean 0 and variance 1. 5 random feature descriptors are shown in figure 6

D. Feature Matching

Feature Matching is done by picking one points from first feature vector and computing the SSD with all the other feature descriptors. Distance Threshold is used to find the matched pairs in the images.we compute match pairs in two different images interest by calculating the sum-squared distance of each feature vector in the first image to each such vector in the second image and save them in sorted lists. If the distance threshold is less than 0.7 we keep the pair else discard it. Draw matches inbuilt function of OpenCV is used to display the output of feature matching. second

E. RANSAC for outlier rejection and to estimate Robust Homography

The next step is to use Random Sample Consensus to remove the outliers and find the robust Homography estimate.After those outliers have been removed, we can compute the homography matrix from the inliers. Here, we also implement ourselves the code to compute the homography matrix

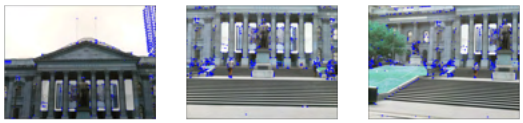

Fig. 2. Harris Corner Output

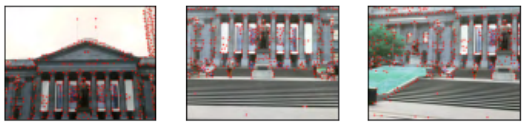

Fig. 3. Shi-Tomasi Corner Output

Fig. 4. ANMS Output

instead of using cv2.getPrespectiveTransform The algorithm has following steps: 1. Select four feature pairs (at random), from image1 and image 2. 2. Compute homography H between the previously picked point pairs. 3. Compute inliers where SSD¡, where is some user chosen threshold and SSD is sum of square difference function. 4. Repeat the previous steps until you have found more than a particular percentage of inliers. 5. Keep largest set of inliers. 6 .Re-compute least-squares H estimate on all of the inliers

Input : Corner score Image (C_{img} obtained using `cornermetric`), N_{best} (Number of best corners needed)
Output: (x_i, y_i) for $i = 1 : N_{best}$

Find all local maxima using `imregionalmax` on C_{img};
Find (x, y) co-ordinates of all local maxima;
((x, y) for a local maxima are inverted row and column indices i.e., If we have local maxima at $[i, j]$ then $x = j$ and $y = i$ for that local maxima);
Initialize $r_i = \infty$ for $i = [1 : N_{strong}]$

```
for i = [1 : N_strong] do
    for j = [1 : N_strong] do
        if (C_img(y_j, x_j) > C_img(y_i, x_i)) then
            ED = (x_j − x_i)^2 + (y_j − y_i)^2
        end
        if ED < r_i then
            r_i = ED
        end
    end
end
```

Sort r_i in descending order and pick top N_{best} points

Fig. 5. ANMS Algorithm

Fig. 6. Randomly selected 5 Feature descriptors

F. Blending Images

For blending we simply overlapped the two images.Here the arrangement plays an important part in stitching After calculating the homography, we transform one image on to the projective plane of other image. To avoid negative values, we translated the projected image and simply overlapped it with other image. We used Half-Split method for image blending. We faced issues like overlapping where there was a black gap in the stitched image. It was corrected by adding a small check while overlapping.Our current algorithm fails to stitch images which are not in order. To stitch the images which are not given in we planned to stitch first image with image having most common feature and repeat it for all images rather than going in a particular order. But due to time constraints we were not able to execute on the said proposal.

The output images after each step for all the test and train set images are attached in the appendix. The appendix contains corners detected using shi-tomasi method for every image. It contains the feature matched after calculating the square error and also contains feature that were filtered using a parameter

alpha. At last the output of the algorithm is show. These steps are repeated for each data set.

II. Phase 2

In this section, we present the deep learning based end-to-end homography estimation pipeline. Firstly, since the CNN we are using is not fully convolutional, it cannot accept inputs with arbitary size. Therefore, we need to generate synthetic patches by ourselves. Now we will present some details on the data preparation:

- Resize: Yes, we will resize the input images to 320*240, before extracting patches.
- Patch size: 128*128
- Perturbation ρ: 32

Furthermore, to avoid potential overfitting, we do not use a fixed training dataset. Instead, during training, the patches are always generated randomly.

A. Supervised Learning Approach

In this subsection, we will present our supervised learning approach. Firstly, we outline our hyperparameters and details of our supervised learning training:

- Number of parameters: 34192264
- Forward pass time: 0.012 seconds per image
- Learning rate: 5e-3
- Optimizer: Adam
- Batch size: 100
- Number of Batches in one epoch: 50
- Number of Epochs: 200
- Loss function: photometric loss
- Device: 1 RTX 2080Ti

We provide a figure of the model architecture we are using in Figure 7. We also present our training process. Figure 8 is the EPE error over epochs on the training set and the validation set.

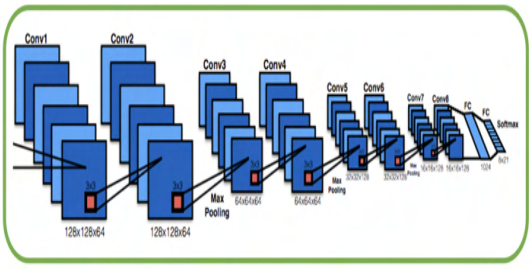

Fig. 7. Model architecture of supervised learning

In table II-A, we compare the final EPE error on the training/validation/test set. It shows that our model generalizes very well.

Now we present four images in Figure 9, 10, 11, 12, showing homographies against a synthetic ground truth.

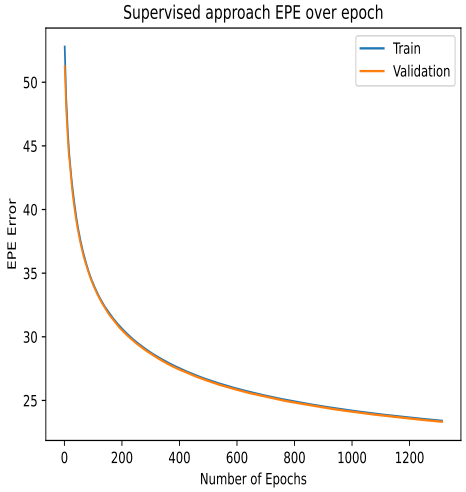

Fig. 8. EPE Error on the training set and test set. It is interesting that the training set error and test set error are quite close (good generalization). The reason is that we are using a dynamic training dataset, which means that training batches generated are always randomized. However, one drawback of this training style is the inputs distribution is varying all the time. Therefore, the training is steady but hard to achieve small errors. To address this, one may need a larger model.

EPE Error	Training Set	Validation Set	Test Set
Supervised	23.4	23.33	21.49
Unsupervised	37.26	36.04	36.82

TABLE I
EPE error across training/validation/test set

Fig. 9. Image overlayed with homography estimated by supervised learning shown in red and ground truth shown in blue

B. Unsupervised Learning Approach

In this subsection, we will present our unsupervised learning approach. Firstly, we outline our hyperparameters and details of our unsupervised learning training:

- Number of parameters: 34192264

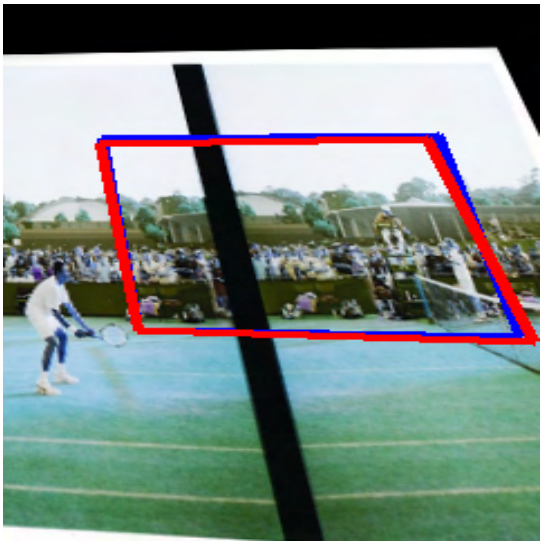

Fig. 10. Image overlayed with homography estimated by supervised learning shown in red and ground truth shown in blue

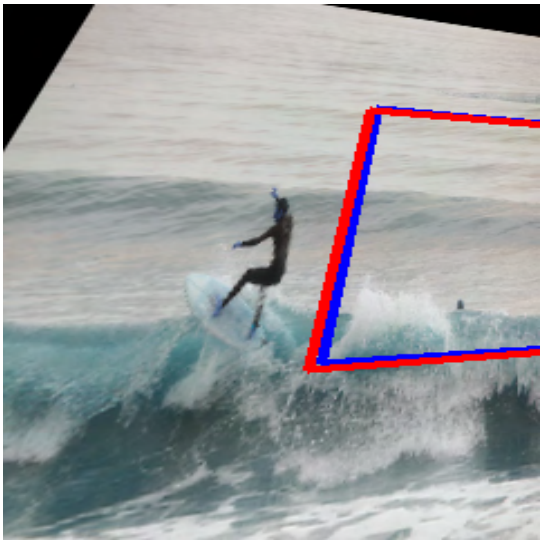

Fig. 11. Image overlayed with homography estimated by supervised learning shown in red and ground truth shown in blue

- Forward pass time: 0.014 seconds per image
- Learning rate: 1e-4
- Optimizer: Adam
- Batch size: 100
- Number of Batches in one epoch: 50
- Number of Epochs: 200
- Loss function: MSE loss
- Device: 1 RTX 2080Ti

We provide a figure of the model architecture we are using in Figure 13.

Now, we firstly present our training process. Figure 14 is the EPE error over epochs on the training set and the validation set.

In table II-A, we compare the final EPE error across the training/validation/test set. It shows that our model generalizes very well.

Fig. 12. Image overlayed with homography estimated by supervised learning shown in red and ground truth shown in blue

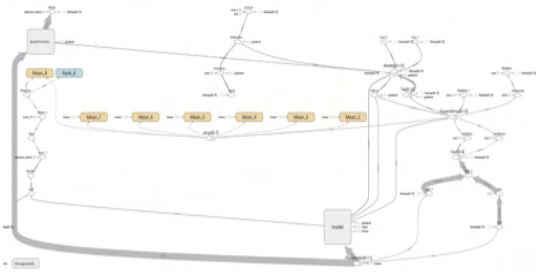

Fig. 13. Model architecture of unsupervised learning

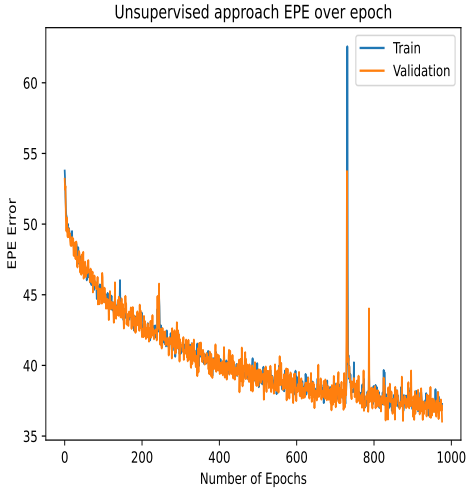

Fig. 14. EPE Error on the training set and test set

Now we present four images in Figure 15, 16, 17, 18, showing homographies against a synthetic ground truth.

Fig. 15. Image overlayed with homography estimated by unsupervised learning shown in red and ground truth shown in blue

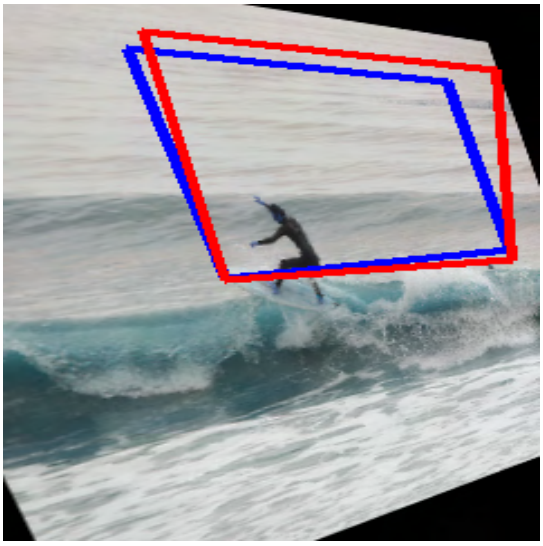

Fig. 17. Image overlayed with homography estimated by unsupervised learning shown in red and ground truth shown in blue

Fig. 16. Image overlayed with homography estimated by unsupervised learning shown in red and ground truth shown in blue

Fig. 18. Image overlayed with homography estimated by unsupervised learning shown in red and ground truth shown in blue

C. Panoramas Using Deep Learning Approaches

Although, both our supervised learning approach and unsupervised learning approach performs well as shown in Figure . However, when we try to apply it to the Test Sets, it is not performing very well. Here, we aim to study the potential reasons of failures. We use the Image1 and Image2 from Test Set 3 and compares the stitched images across the traditional approach, supervised learning approach, and the unsupervised learning approach in Figure . The traditional approach performs very well as expected. We can actually find that images from deep learning approach is correctly rotated. The reason of failure comes from the the inaccurate shift components (the corresponding $H[1;3]$ and $H[2;3]$) components in the homography matrix). Therefore, we think there should be two possible improvement directions:

- Training more accurate deep learning model.
- Search the shift component so that the warped images match the originial images since the rotational component is already correctly learned.

This also complies with the intuition that the rotational features between two patches can be learned more easily.

III. Conclusion

A. Phase 1

- The algorithm uses corners to detect feature which can be replaced by a scale and illumination and rotation invariant method like sift.

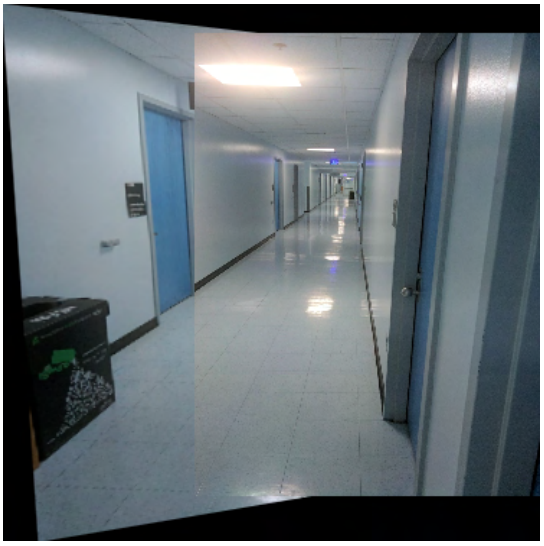

Fig. 19. Stitching with traditional approach

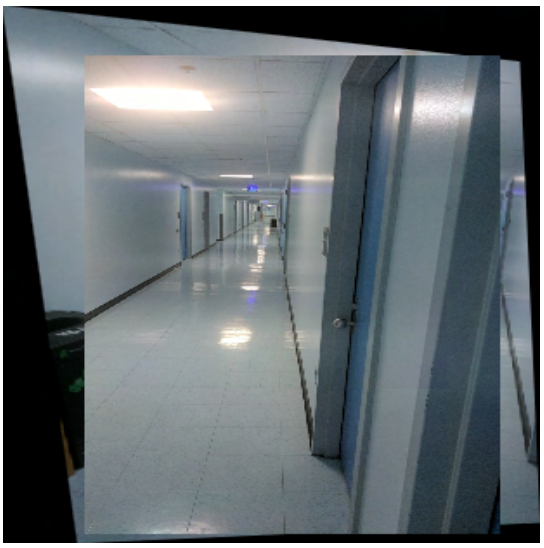

Fig. 20. Stitching with supervised approach

- The algorithm depends on sequence of image data given which can be improved upon where we take image 1 as base image and calculate the relevant image to stitch and repeat it.
- Final image can be further processed to obtain seamless Panorama.

B. Phase 2

Here we tend to compare the traditional approaches and deep learning approaches.

- Model Complexity: Deep learing models are complex to construct but easier to use since it is end-to-end. But it also requires more memory for parameters.
- Accuracy: Traditional approaches are more accurate than our deep learning approach since the features can be perfectly matched.

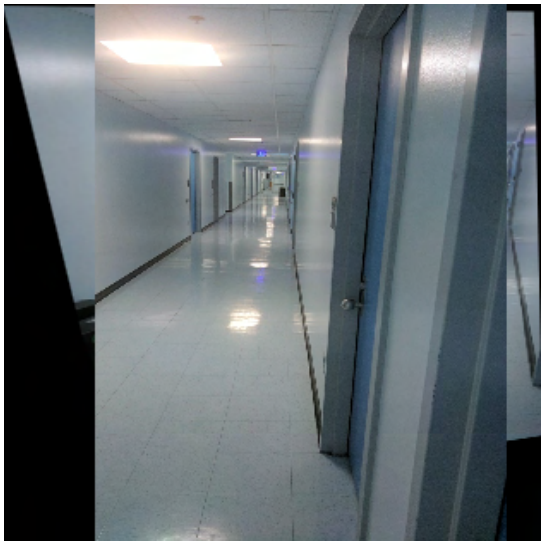

Fig. 21. Stitching with unsupervised approach

- Inference speed: The deep learning approach is more faster, requiring only about 0.01 second to run, since it is end-to-end. The traditional methods take several seconds to process one pair of images.

ACKNOWLEDGMENT

The authors would like to thank the professor and TAs for preparing the comprehensive study material and homework!

IV. APPENDIX

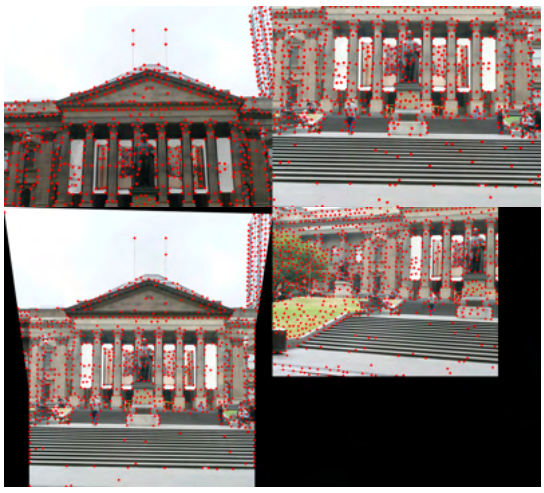

Fig. 22. Corners detected For Train Set1

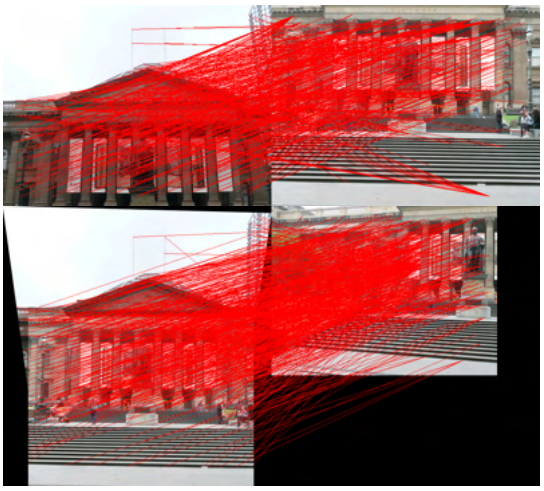

Fig. 23. Feature Mapping for Train Set1

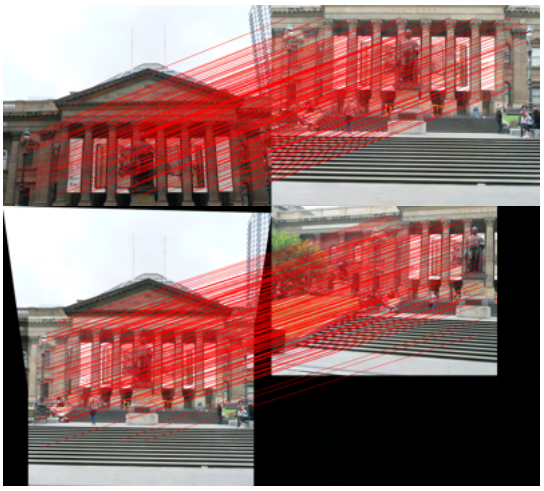

Fig. 24. Filter Matched Pairs for Train Set 1

Fig. 25. Panoroma Stiching of two images of Train Set 1

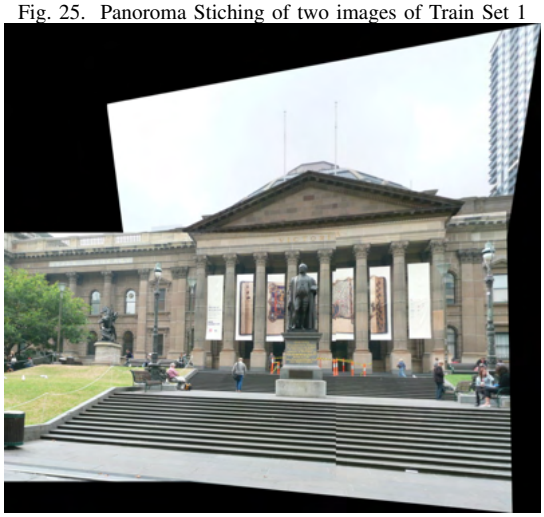

Fig. 26. Panoroma Outputs for Train Set 1

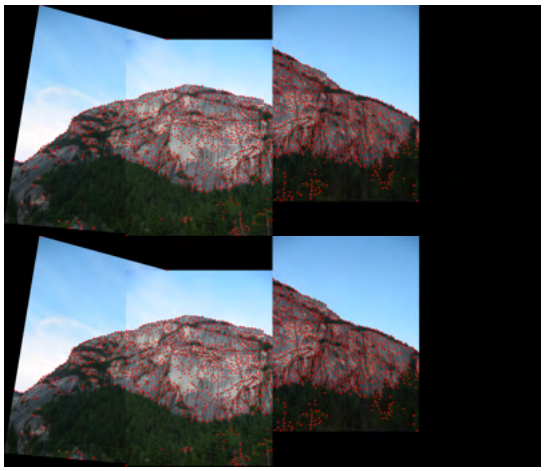

Fig. 27. Corner Detected in Train Set 2

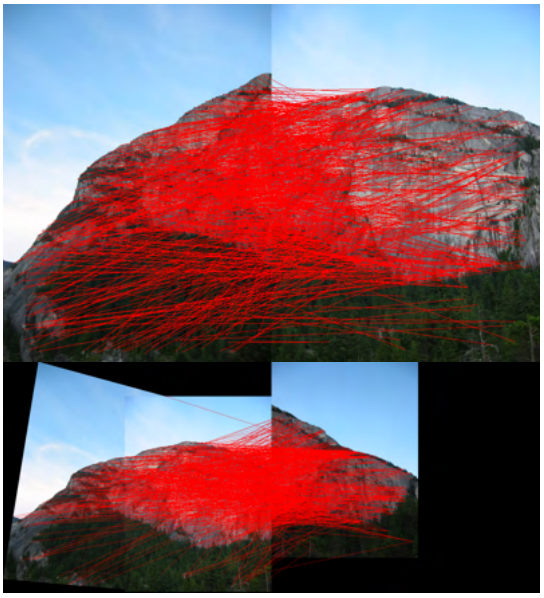

Fig. 28. Feature Matching in Train Set 2

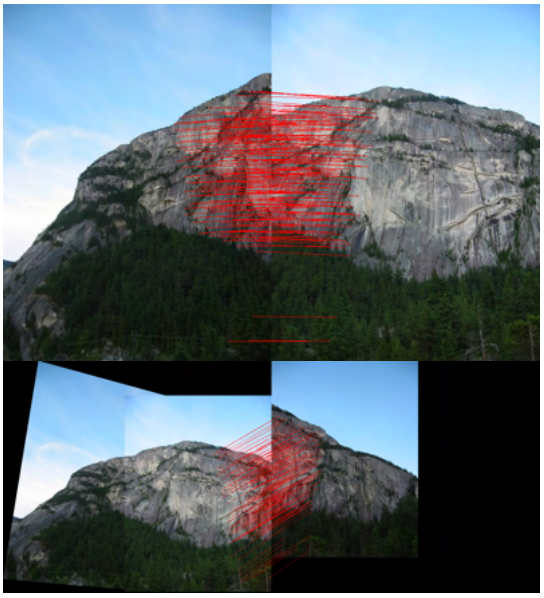

Fig. 29. Filtered Feature Matching in Train Set 2

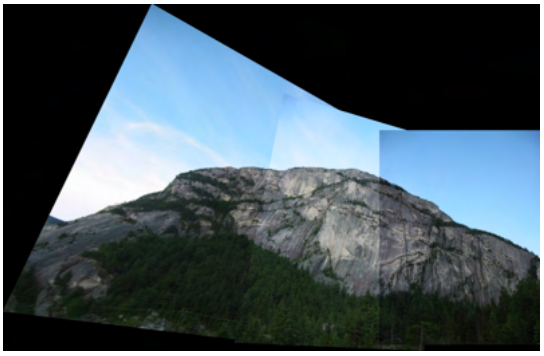

Fig. 30. Panorama Output Train Set 2

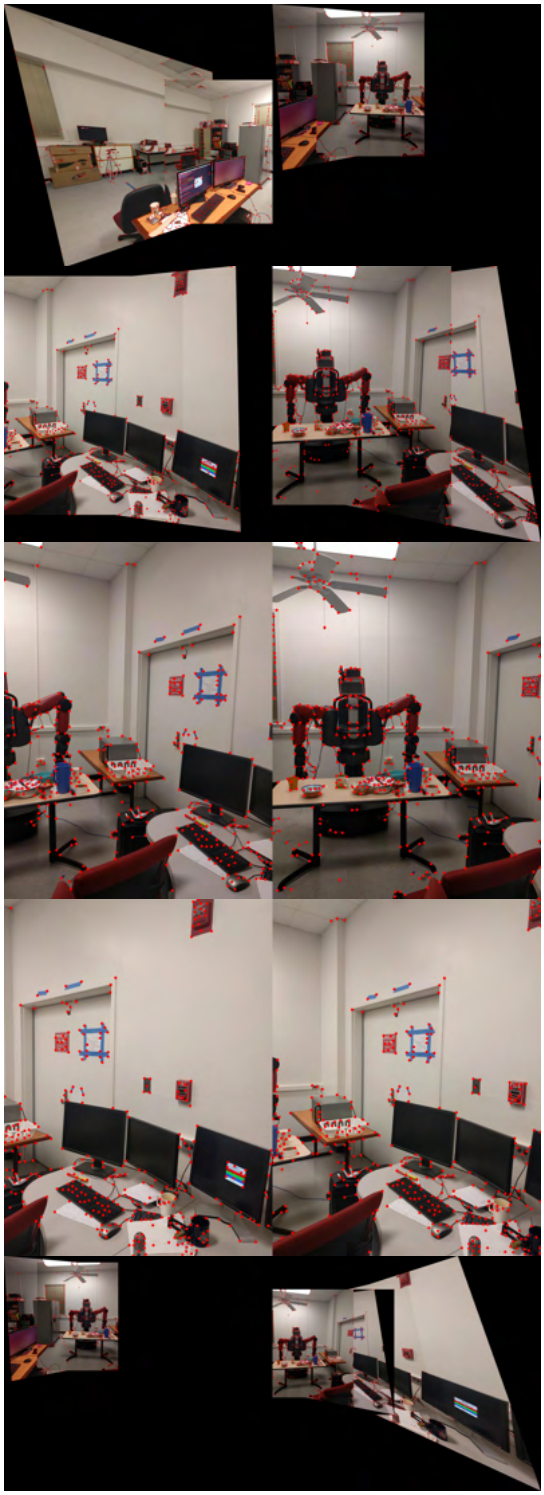

Fig. 31. Corner Detected in Train Set 3

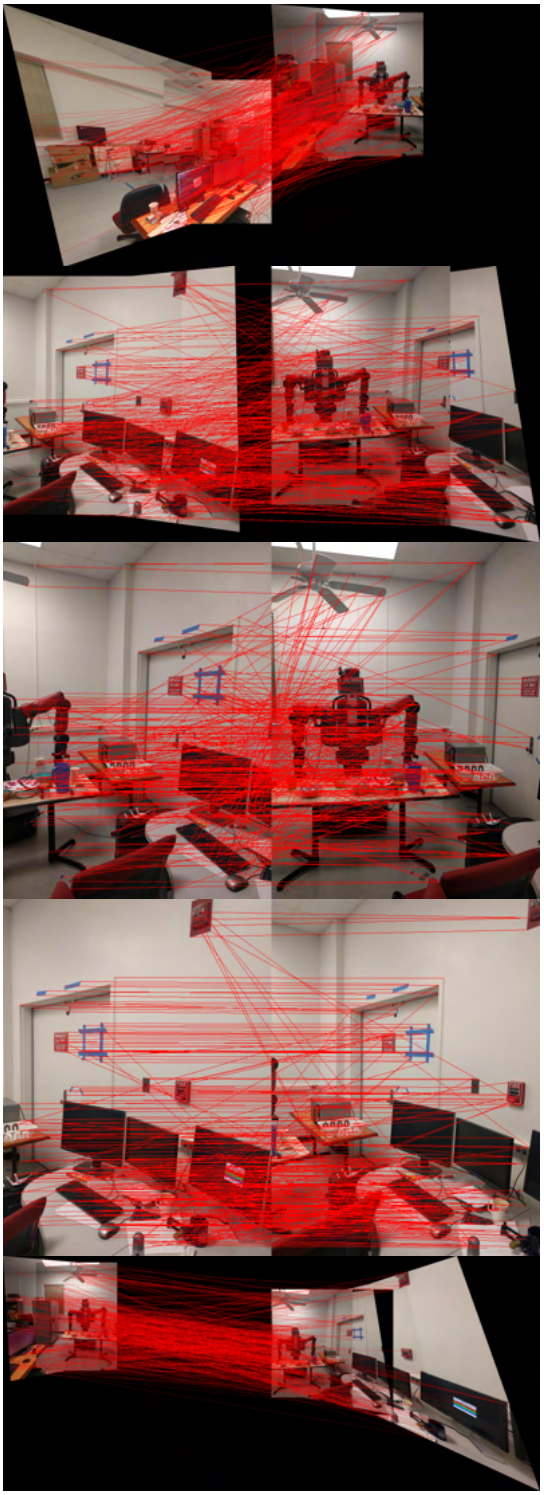

Fig. 32. Feature Matching in Train Set 3

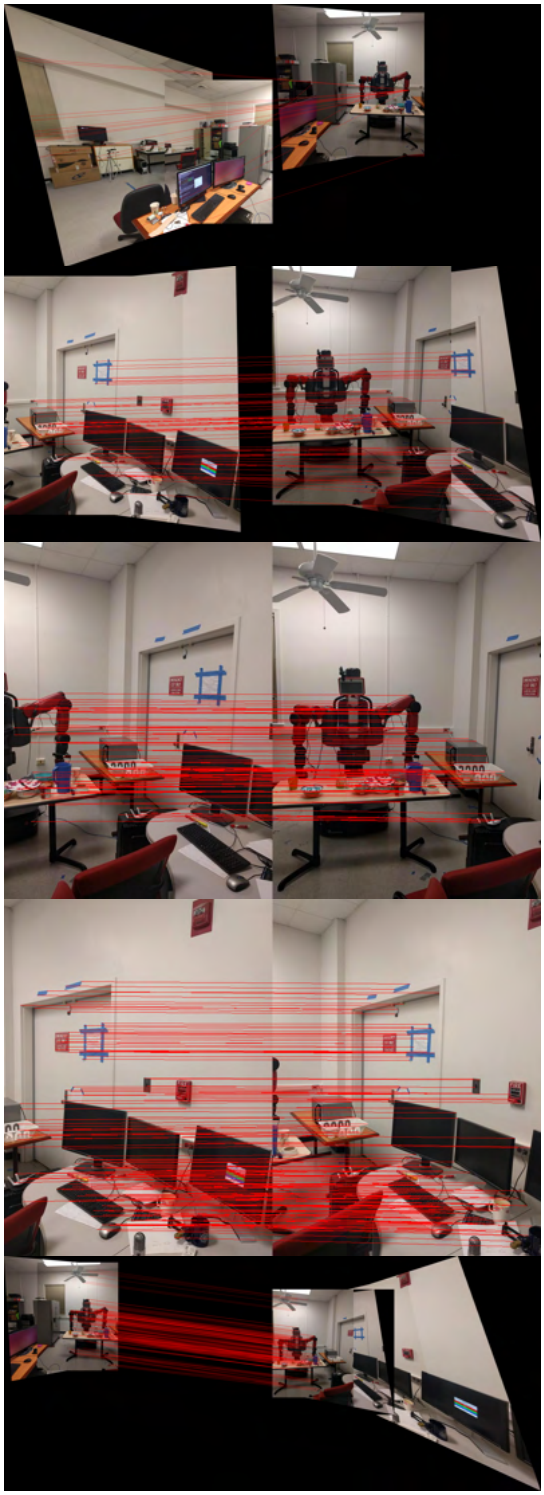

Fig. 33. Filtered Feature Matching in Train Set 3

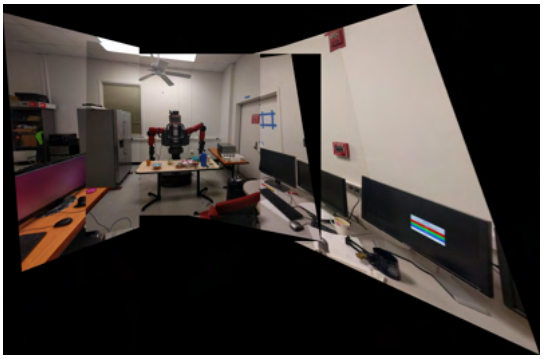

Fig. 34. Panorama Output Train Set 3

Fig. 35. Corner Detected in Test Set 1

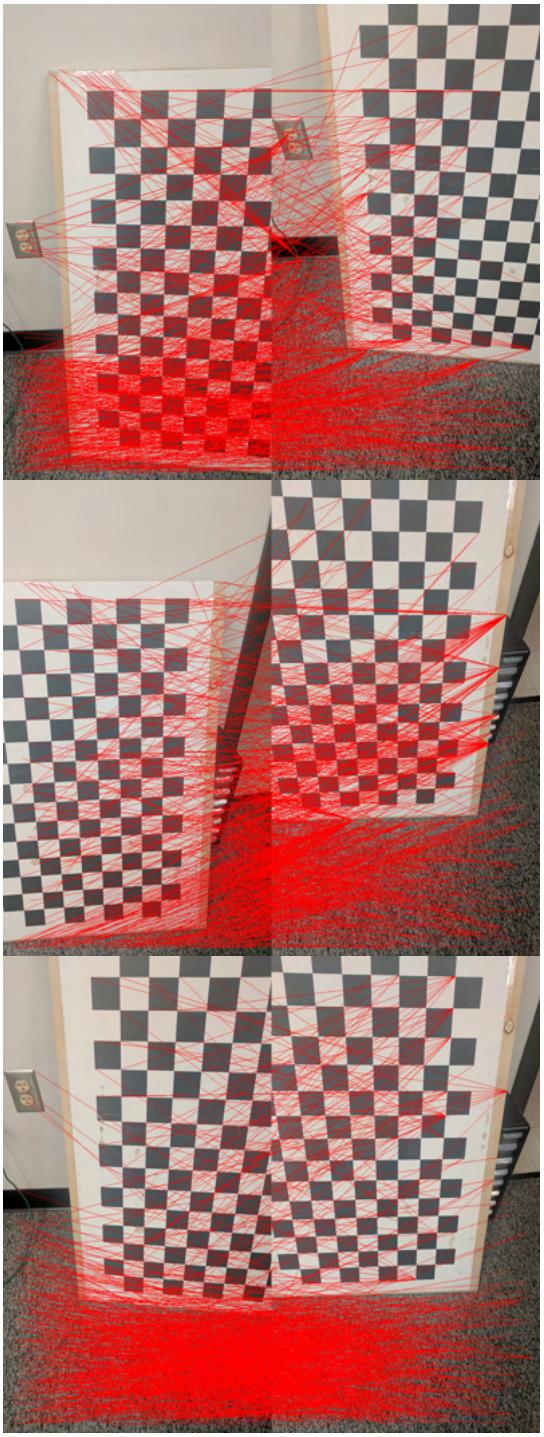

Fig. 36. Feature Matching in Test Set 1

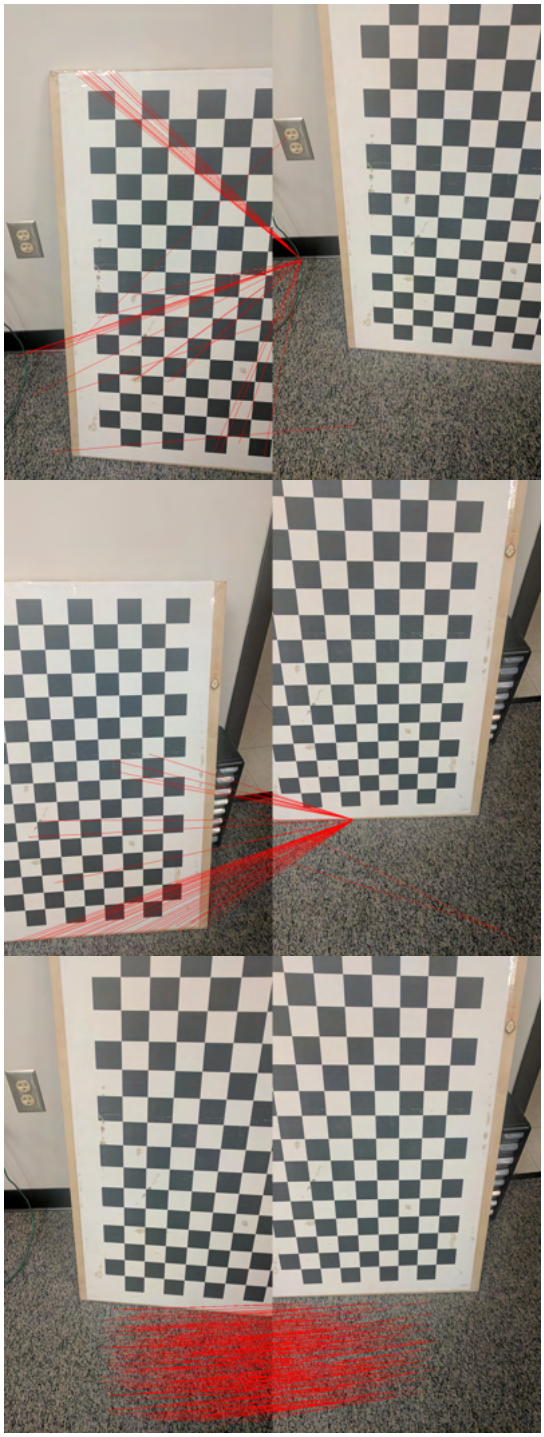

Fig. 37. Filtered Feature Matching in Test Set 1

Fig. 38. Panorama Output Test Set 1

Fig. 39. Corner Detected in Test Set 2

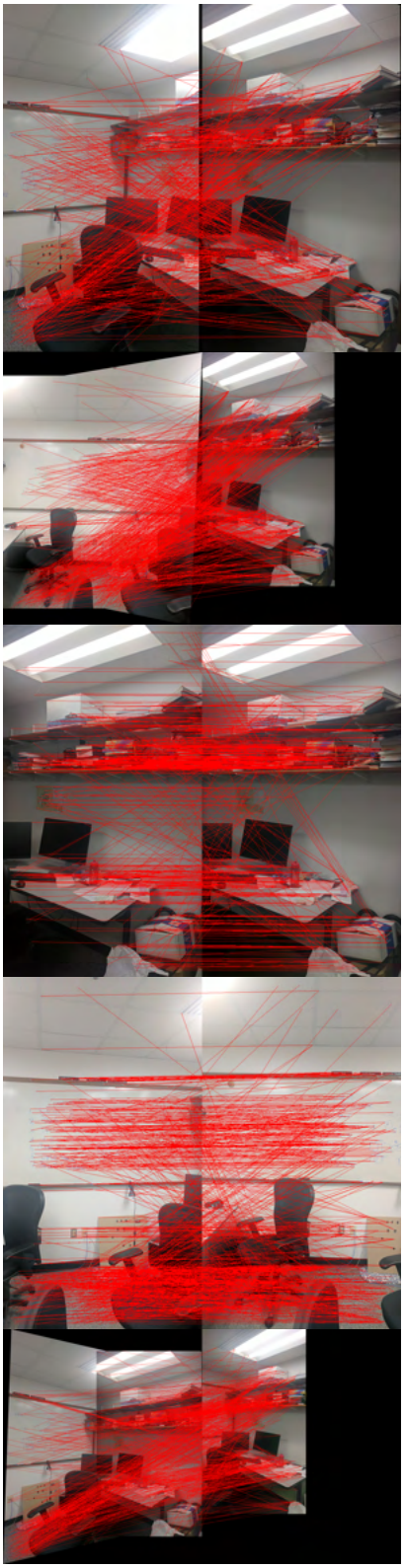

Fig. 40. Feature Matching in Test Set 2

Fig. 41. Filtered Feature Matching in Test Set 2

Fig. 42. Panorama Output 1 Test Set 2

Fig. 44. Corner Detected in Test Set 3

Fig. 43. Panorama Output 2 Test Set 2
Our Algorithm produces two panorama due to dependence on sequence

Fig. 45. Feature Matching in Test Set 3

Fig. 46. Filtered Feature Matching in Test Set 3

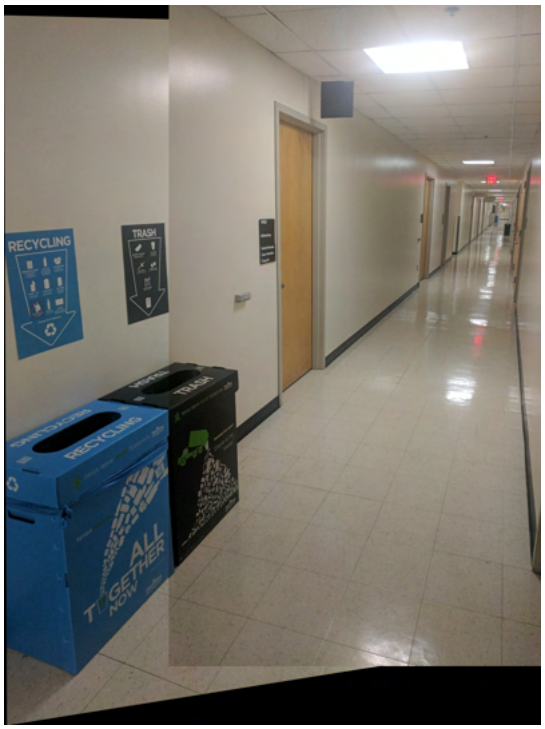

Fig. 47. Panorama Output Test Set 3

Fig. 48. Corner Detected in Test Set 4

Fig. 49. Feature Matching in Test Set 4

Fig. 50. Filtered Feature Matching in Test Set 4

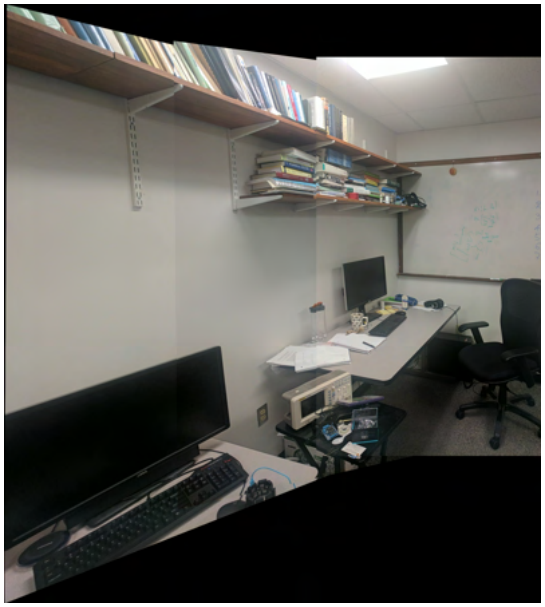

Fig. 51. Panorama Output Test Set 4

Fig. 52. Input Images Custom Set 1

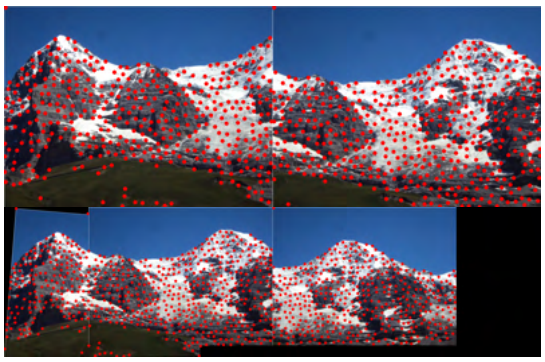

Fig. 53. Corner Detected in Custom Set 1

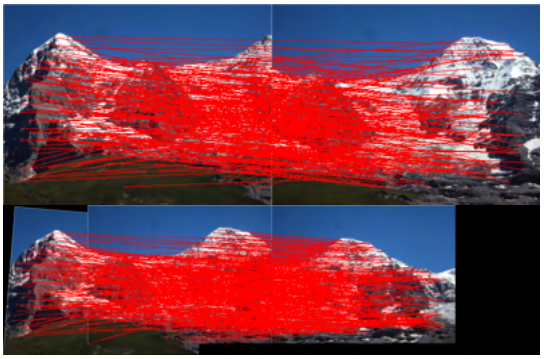

Fig. 54. Feature Matching in Custom Set 1

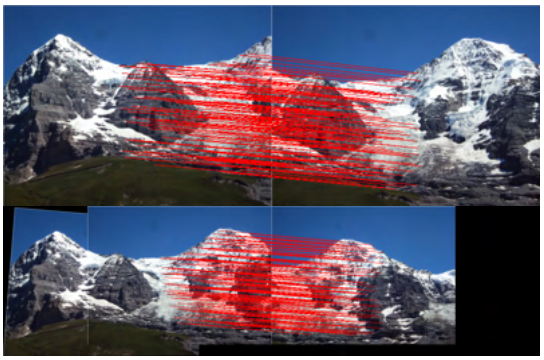

Fig. 55. Filtered Feature Matching in Custom Set 1

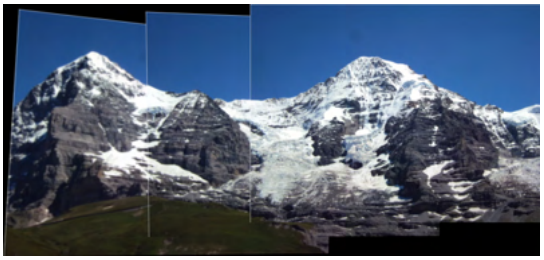

Fig. 56. Panorama Output custom Set 1

Fig. 57. Input Images Custom Set 2

Fig. 58. Corner Detected in Custom Set 2

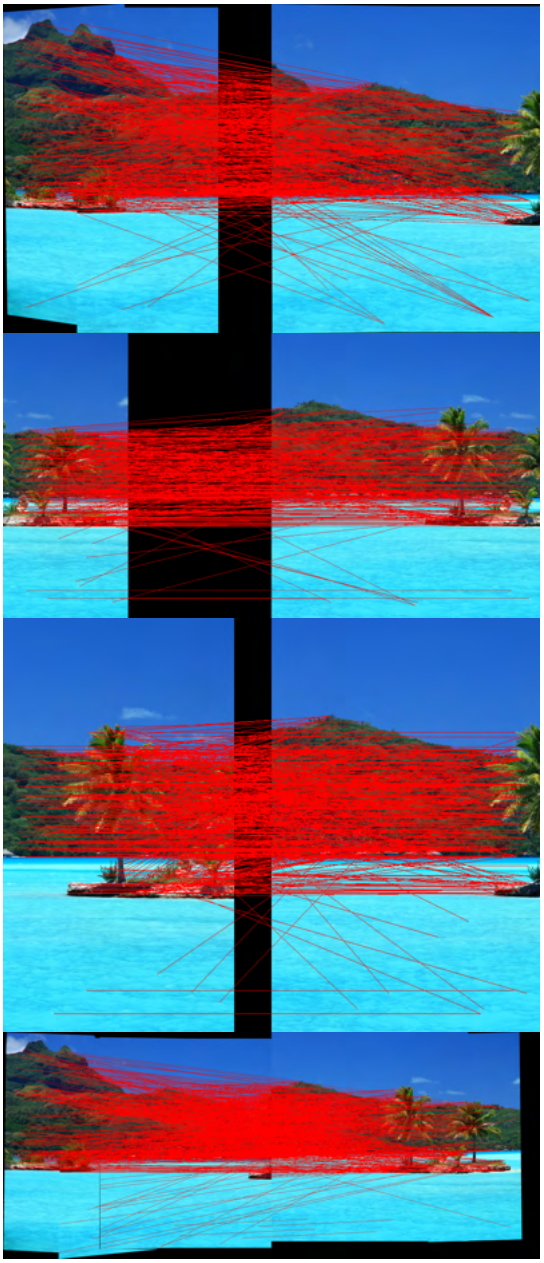

Fig. 59. Feature Matching in Custom Set 2

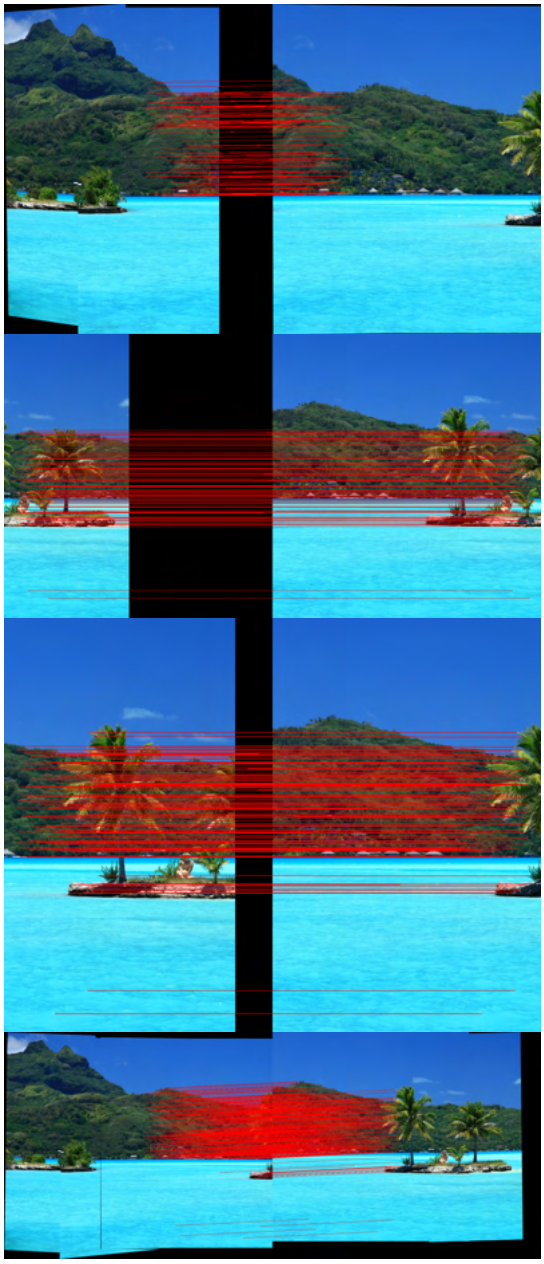

Fig. 60. Filtered Feature Matching in Custom Set 2

Fig. 61. Panorama Output Custom Set 2