Project 1 - My AutoPano

Ashwin Kuruttukulam
UID: 115906518
Email: ashwinvk@umd.edu

Aruna Baijal
UID: 116130110
Email: abaijal@umd.edu

***Abstract*—The purpose of this project is to stitch two or more images in order to create one seamless panorama image. Each image should have few repeated local features (30-50% or more, emperically chosen). In this project, you need to capture multiple such images. We do this using geometric Computer Vision approach and Deep Learning approach.**

I. Phase 1

In the traditional method, we first detect corners in the images. These corners are filtered using Adaptive Non-Maximal Suppression or ANMS algorithm. The final corners in the images are used to match features between two images. The matched features undergo RANSAC to remove outliers. The final matching features are used to stitch the images together.

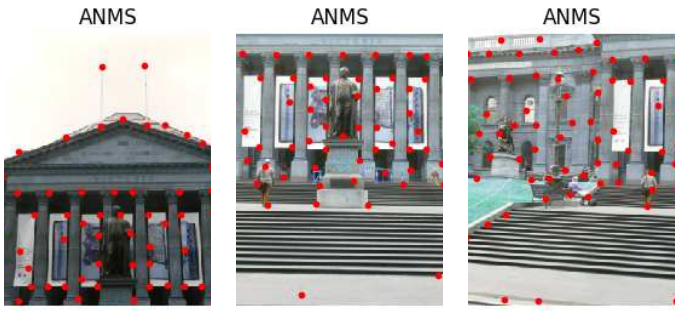

Fig. 1: Corners detected after ANMS for train set 1

A. Corner Detection

We made use of cv2 function goodFeaturesToTrack to detect corners in our images. The function returns the strongest 'N' corners in an image. The function takes a radius and a threshold value for corner strength. Within a given radius, the function returns a single corner of highest strength. The corner strength should be greater than the threshold value * strength of strongest corner to be accepted as a corner. For our project we considered 600 corners with a corner strength threshold ratio of 0.05, and radius as 20.

B. Adaptive Non-Maximal Suppression (ANMS)

The ANMS algorithm returns equally distributed corners across the image. This makes sure we do not focus on strong corners in only a part of the image resulting in losing out features from less strong parts of the image. As we observed, feature mapping would work well without equally distributed corners for train set 1 as there are multiple strong features all along the image. But for the train set 2, since the strong corners exist along the edge of the hill, we will not be able to obtain feature matching without corners all along the surface of the hill too. Hence, ANMS is applied to make sure we maintain features everywhere in the image.

C. Feature Descriptor

We create patches of size 40X40 centered at the features detected. These patches are then passed through a Gaussian filter. We sub-sample this patch to a 8X8 patch and then finally reshape it into a 64X1 vector. We make sure this vector is normalized - has zero mean and variance of 1. This process takes care of the difference in brightness and illumination within an image.

D. Feature Matching

To match features between 2 images, we calculate the least square distance between 2 pairs. We conclude the pair is a match if the distance between these two points is significantly lower than the distance of the first point with any other point.We have taken the threshold ratio of distance between 2 points and next closest point to be 0.8 for our implementation.

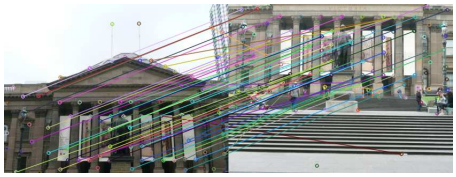

Fig. 2: Matches before RANSAC train set 1

E. RANSAC for outlier rejection

The matches that we get from feature matching are not all good matches, as we can see from the image. We use RANSAC to remove the bad matches. We randomly select any 4 matched pairs between the 2 images. Corresponding to these 4 matches, we generate a Homography using cv2 function getPerspectiveTransform. We apply this generated homography to our image 1 and calculate the distance between the warped points from resultant image and our second image. If the number of matches that agree with this transform are

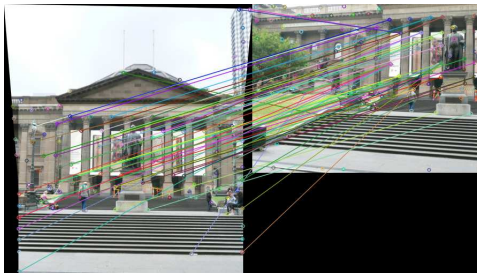

Fig. 3: Matches before RANSAC train set 1

greater than a threshold value (45% in our case), we accept the inliers for this homography as our good matches.

If we do not find enough good matches, we ignore the second image from the set of images to create the panoramic image.

Fig. 4: Matches after RANSAC train set 1

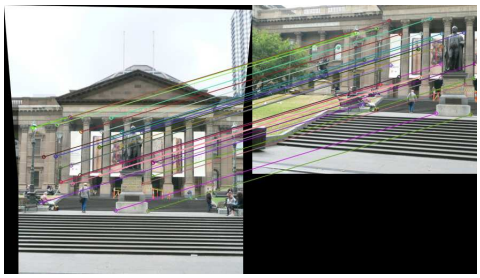

Fig. 5: Matches after RANSAC train set 1

F. Blending Images

The image pairs that have sufficient matches are stitched together to form a panoramic image. We recalculate the homography for all the inlier matches and apply this homography to the image 1. This warped image after transformation may generate points with negative coordinates. We need to translate the original image 1 further in the positive direction to ensure after warping all coordinates in the warped image are positive so we don't lose data. We include the translation in the homography matrix, then apply the homography to the image using cv2 function warpPerspective. We calculate the new warped coordinates of the matches by applying the homography to these coordinates to generate the warped coordinates.
To stitch the two images we shift the second image with respect to first image till the matched points coincide. We overwrite the warped and padded first image with values of second image to stitch them together. We were unable to generate a common contrast/brightness level for the combined image.

Fig. 6: Intermediate Panoramic image train set 1

Fig. 7: Final Panoramic image train set 1

G. Train Sets

For train set 3, we dropped image 1 from the set as it was resulting in extreme warping and hence was not a good fit for a panoramic image. Due to the large number of images in the set, if we continued warping and combining images from one end the resultant warping kept getting worse and ultimately warped the first end so much that the lower corners would end up above the top corners. To combat that we warped and combined half of the images from one end and generated a second combination from the other end. We then combined these two intermediate panoramas to generate a final panorama.

Below are results for training and custom data sets.

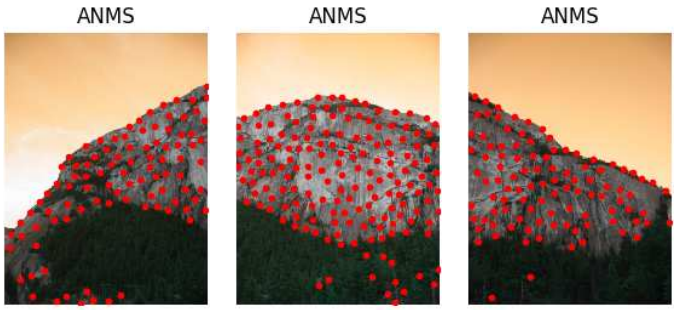

Fig. 8: Corners detected after ANMS for train set 2

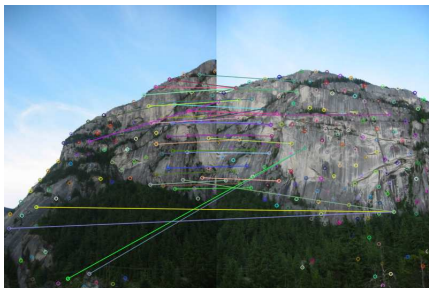

Fig. 9: Matches before RANSAC train set 2

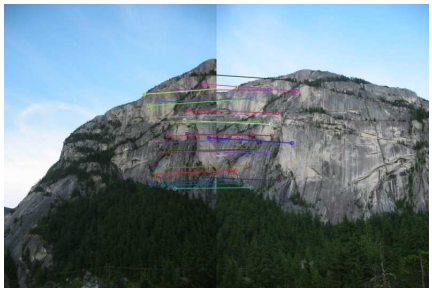

Fig. 10: Matches after RANSAC train set 2

Fig. 11: Intermediate Panoramic image train set 2

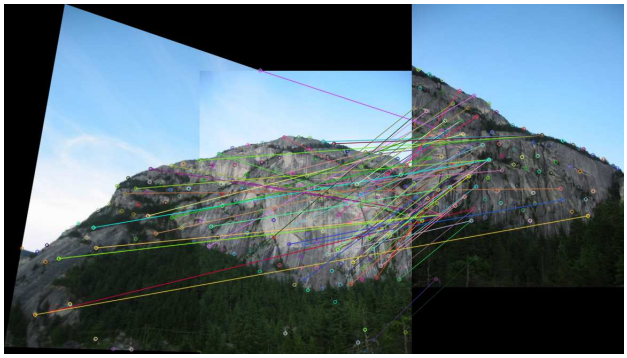

Fig. 12: Matches before RANSAC train set 2

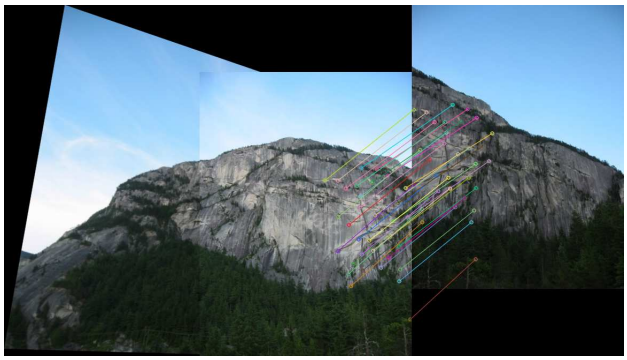

Fig. 13: Matches after RANSAC train set 2

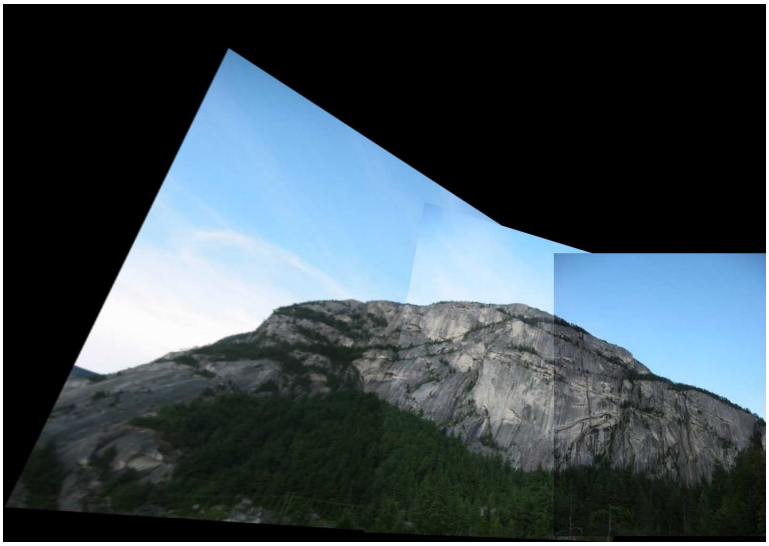

Fig. 14: Final Panoramic image train set 2

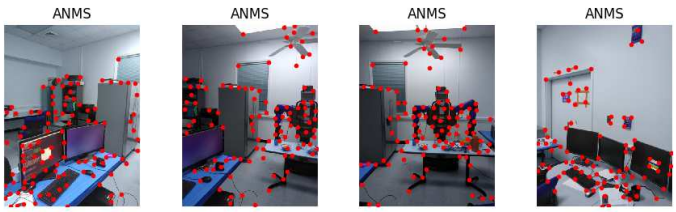

Fig. 15: Corners detected after ANMS for train set 3

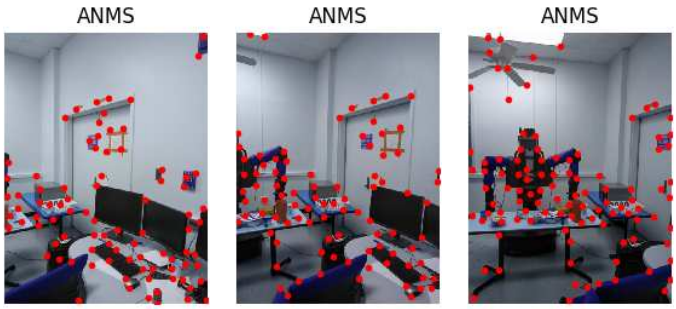

Fig. 16: Corners detected after ANMS for train set 3

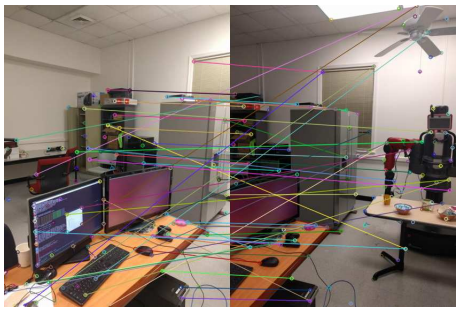

Fig. 17: Matches before RANSAC train set 3

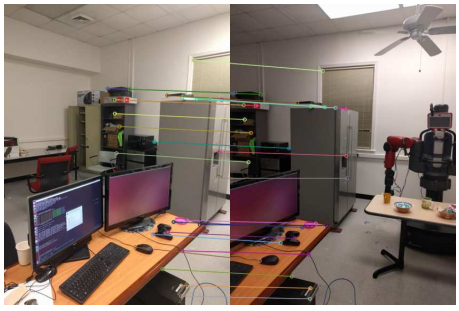

Fig. 18: Matches after RANSAC train set 3

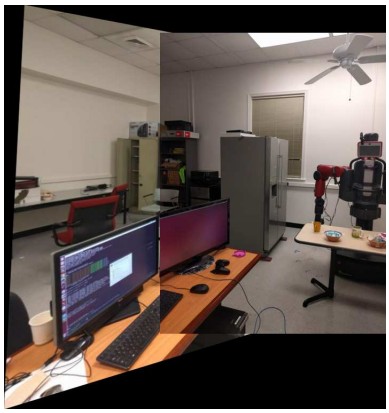

Fig. 19: Intermediate Panoramic image train set 3

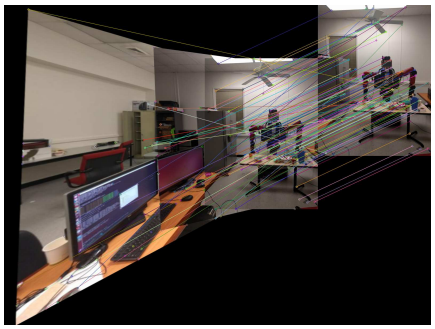

Fig. 20: Matches before RANSAC train set 3

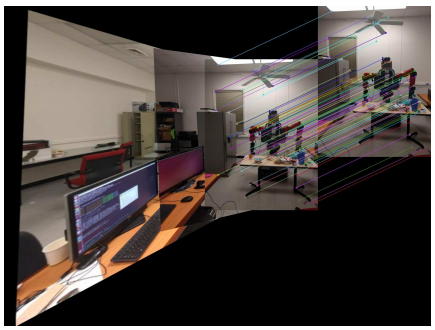

Fig. 21: Matches after RANSAC train set 3

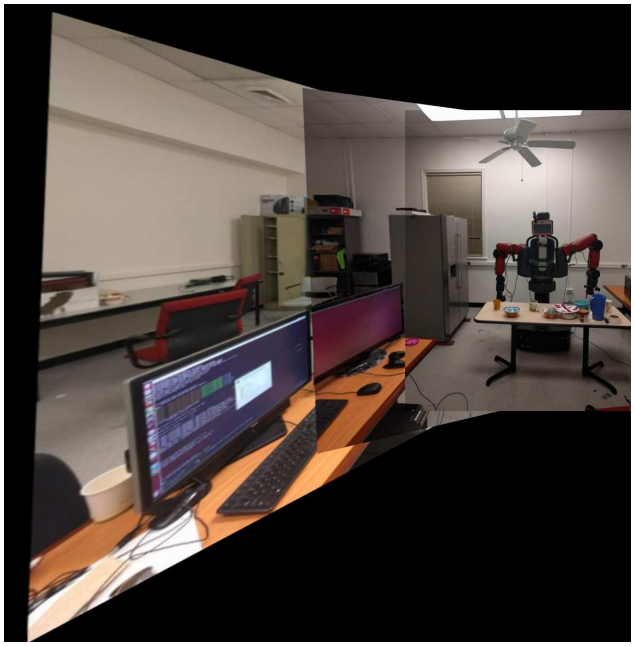

Fig. 22: Intermediate Panoramic image train set 3

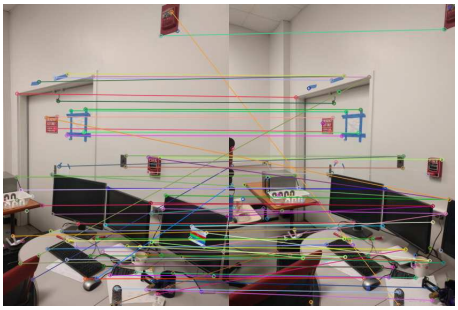

Fig. 23: Matches before RANSAC train set 3

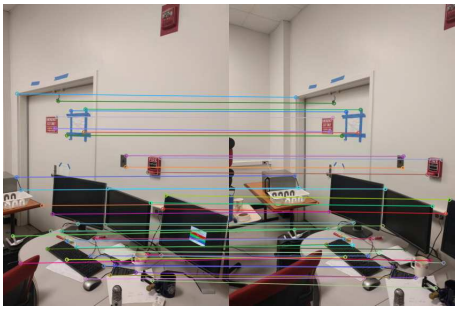

Fig. 24: Matches after RANSAC train set 3

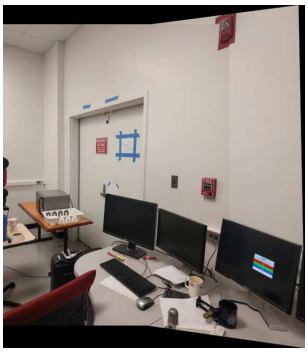

Fig. 25: Intermediate Panoramic image train set 3

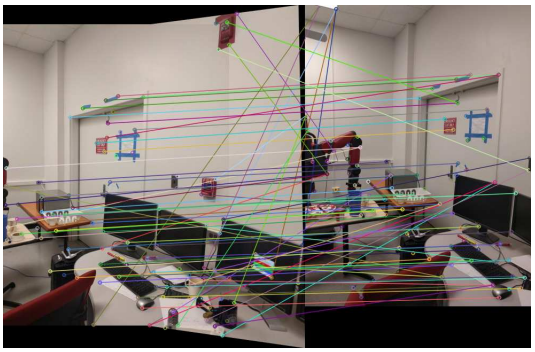

Fig. 26: Matches before RANSAC train set 3

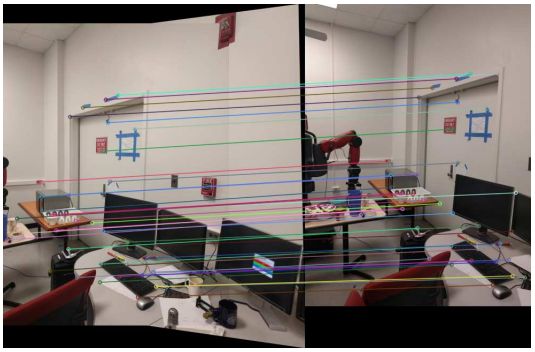

Fig. 27: Matches after RANSAC train set 3

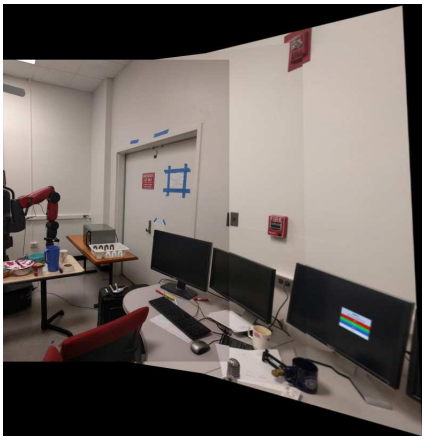

Fig. 28: Intermediate Panoramic image train set 3

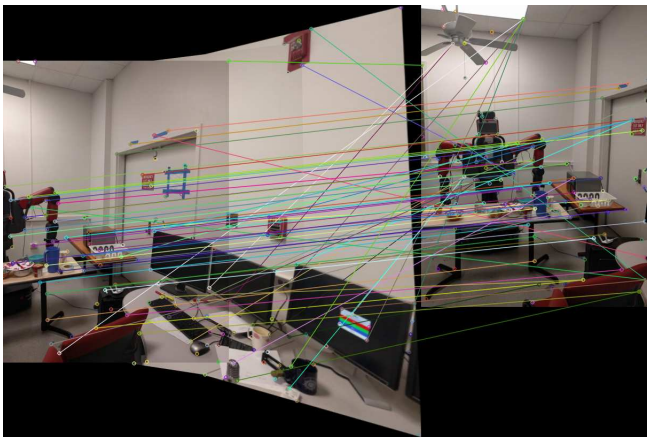

Fig. 29: Matches before RANSAC train set 3

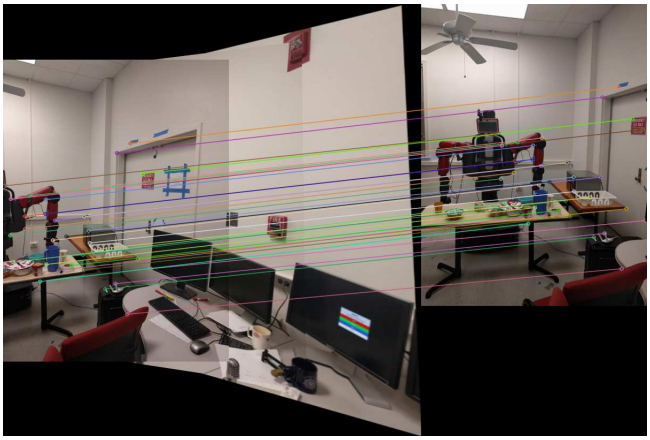

Fig. 30: Matches after RANSAC train set 3

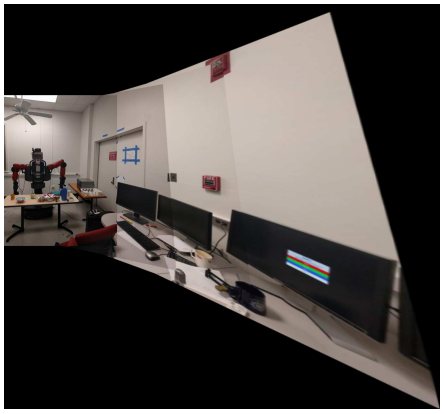

Fig. 31: Intermediate Panoramic image train set 3

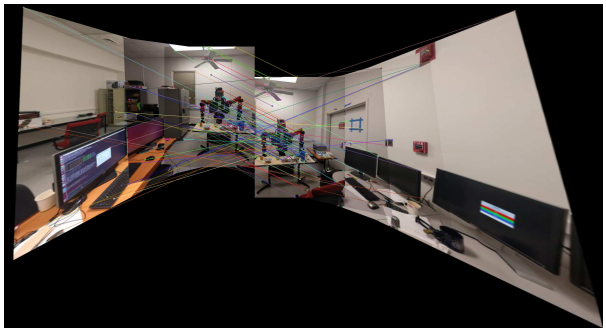

Fig. 32: Matches before RANSAC train set 3

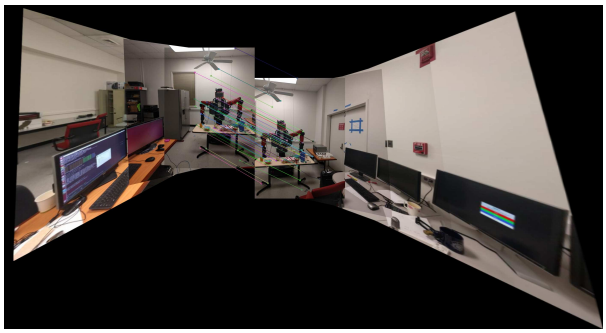

Fig. 33: Matches after RANSAC train set 3

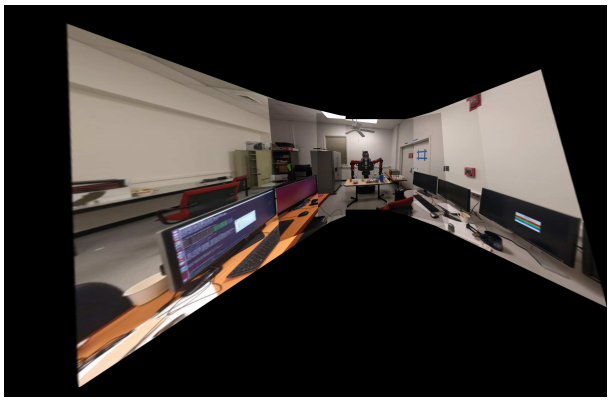

Fig. 34: Final Panoramic image train set 3

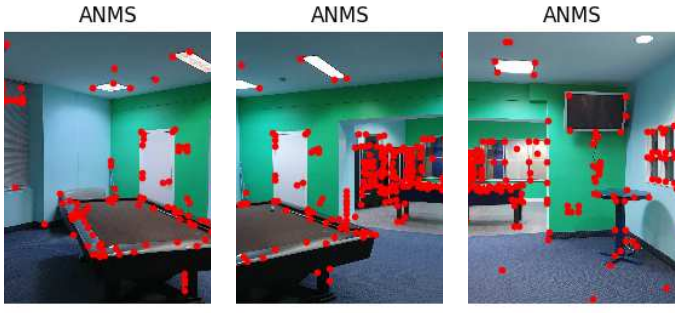

Fig. 35: Corners detected after ANMS for custom train set 1

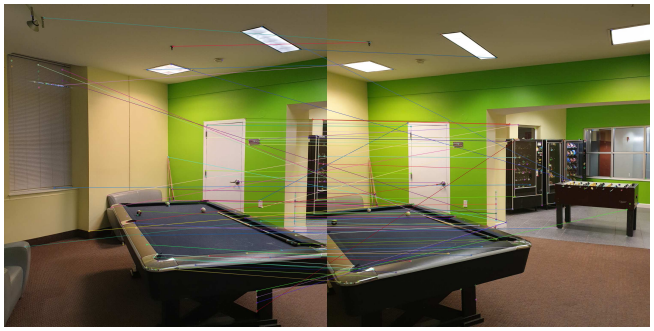

Fig. 36: Matches before RANSAC for custom train set 1

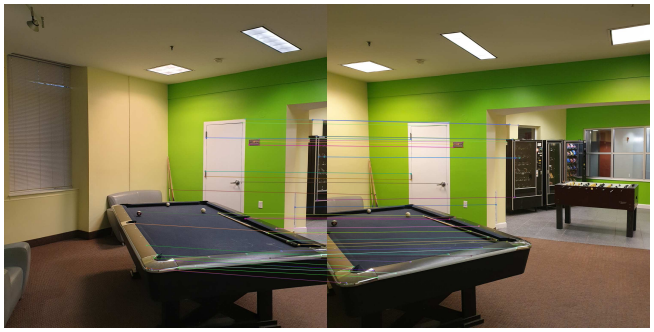

Fig. 37: Matches after RANSAC for custom train set 1

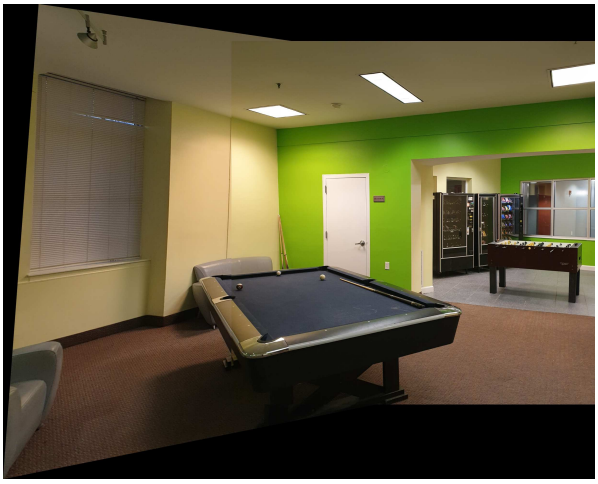

Fig. 38: Intermediate Panoramic image for custom train set 1

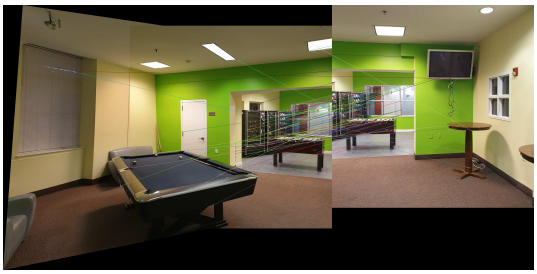

Fig. 39: Matches before RANSAC for custom train set 1

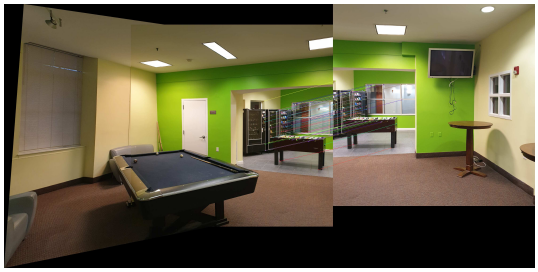

Fig. 40: Matches after RANSAC for custom train set 1

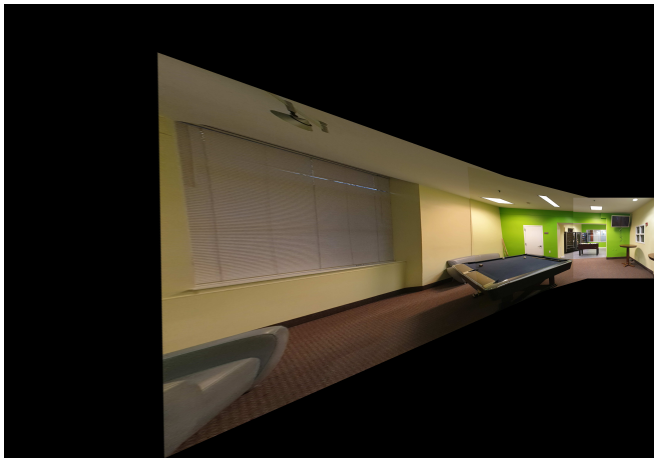

Fig. 41: Final Panoramic image for custom train set 1

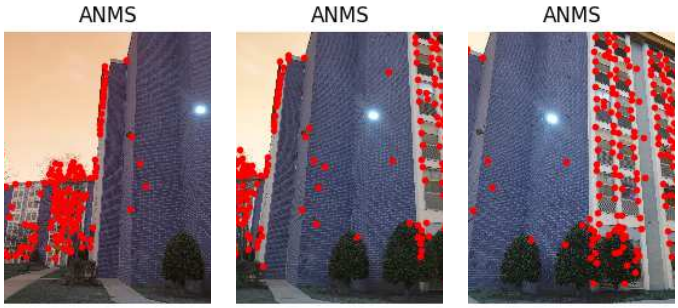

Fig. 42: Corners detected after ANMS for custom train set 2

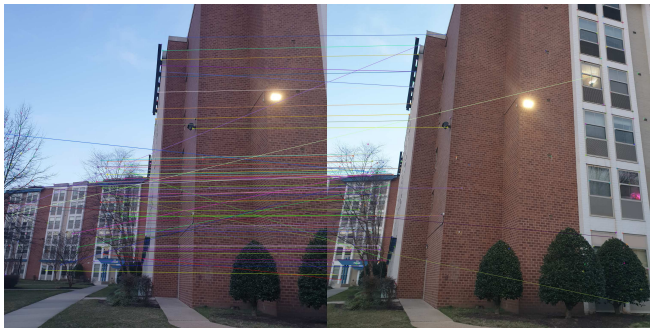

Fig. 43: Matches before RANSAC for custom train set 2

Fig. 44: Matches after RANSAC for custom train set 2

Fig. 45: Intermediate Panoramic image for custom train set 2

Fig. 48: Final Panoramic image for custom train set 2

Fig. 46: Matches before RANSAC for custom train set 2

Fig. 47: Matches after RANSAC for custom train set 2

H. Test Sets

For test set 1, the algorithm found good matches between image 1 and 2 but fails to find good matches between the combined images 1,2 and image 3 or 4. For test set 2, the algorithm does not find matching features between combined images 1,2,3 and image 4, so it drops image 4 from the sequence. Similarly it drops images 5 and 6. Similar to train set 3, it next starts matching from the opposite end and creates a panoramic image of images 7, 8, 9. Next it matches the two panoramic images to build a final image. Results for test sets.

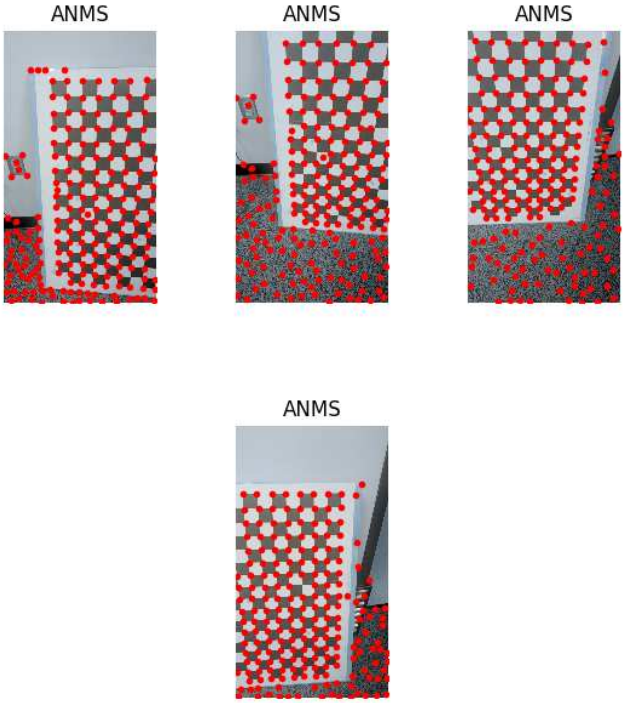

Fig. 49: Corners detected after ANMS for test set 1

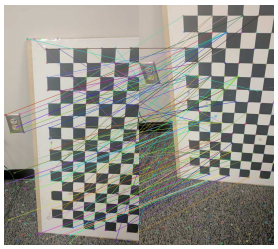

Fig. 50: Matches before RANSAC for test set 1

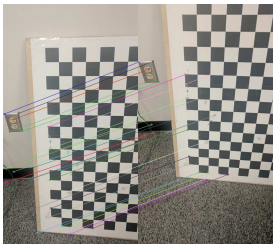

Fig. 51: Matches after RANSAC for test set 1

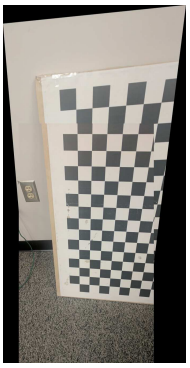

Fig. 52: Intermediate Panoramic image for test set 1

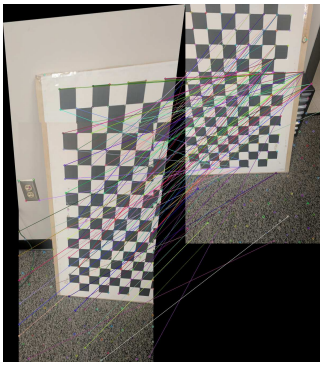

Fig. 53: Matches before RANSAC for test set 1

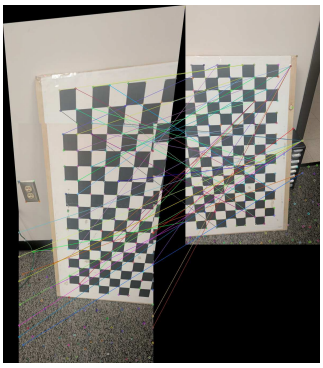

Fig. 54: Matches after RANSAC for test set 1

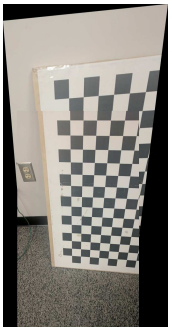

Fig. 55: Final Panoramic image for test set 1

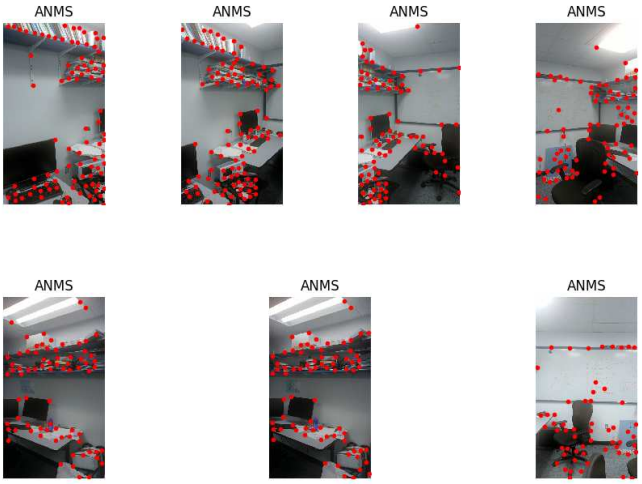

Fig. 56: Corners detected after ANMS for test set 2

Fig. 57: Matches before RANSAC for test set 2

Fig. 58: Matches after RANSAC for test set 2

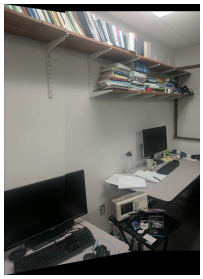

Fig. 59: Intermediate Panoramic image for test set 2

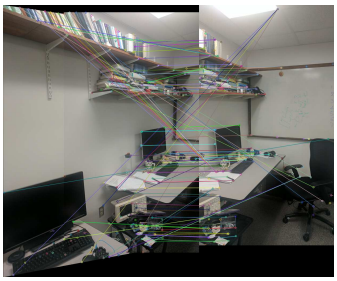

Fig. 60: Matches before RANSAC for test set 2

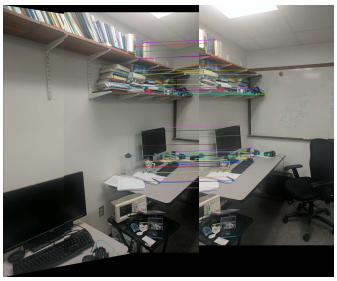

Fig. 61: Matches after RANSAC for test set 2

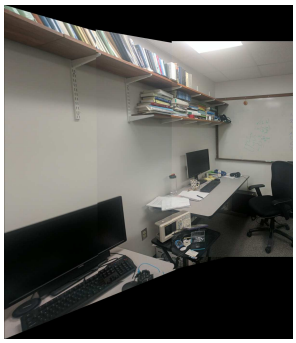

Fig. 62: Intermediate Panoramic image for test set 2

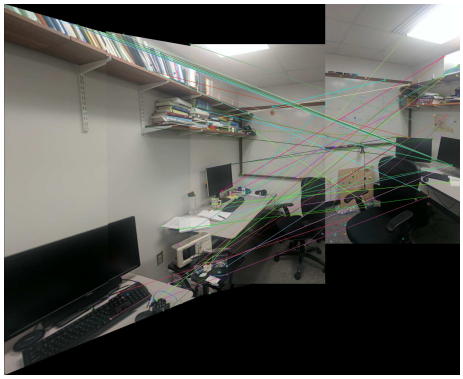

Fig. 63: Matches before RANSAC for test set 2

Fig. 64: Matches before RANSAC for test set 2

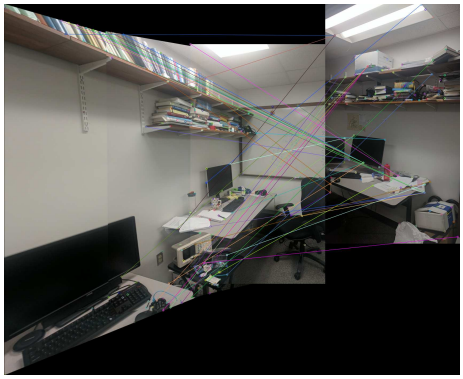

Fig. 65: Matches before RANSAC for test set 2

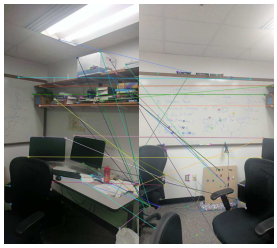

Fig. 66: Matches before RANSAC for test set 2

Fig. 67: Matches after RANSAC for test set 2

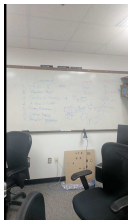

Fig. 68: Intermediate Panoramic image for test set 2

Fig. 69: Matches before RANSAC for test set 2

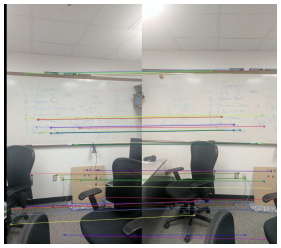

Fig. 70: Matches after RANSAC for test set 2

Fig. 71: Intermediate Panoramic image for test set 2

Fig. 72: Matches before RANSAC for test set 2

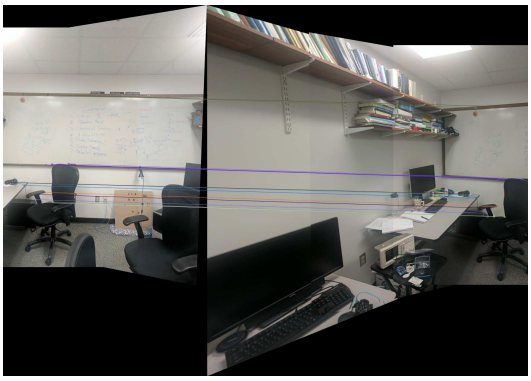

Fig. 73: Matches after RANSAC for test set 2

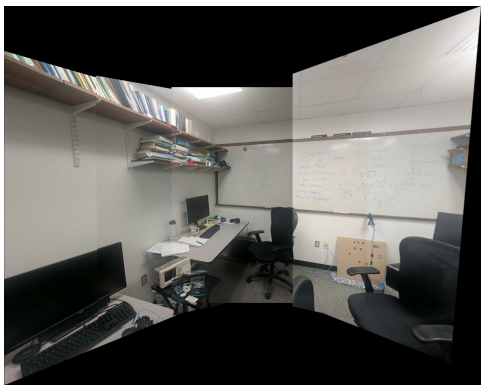

Fig. 74: Final Panoramic image for test set 2

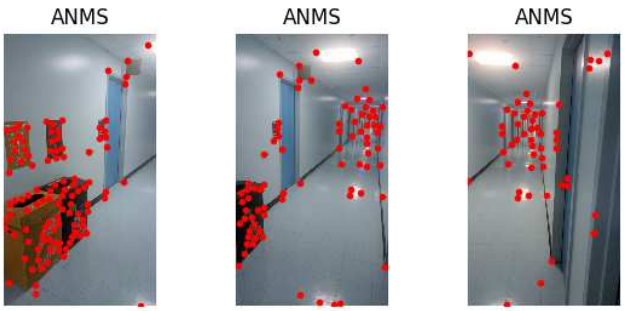

Fig. 75: Corners detected after ANMS for test set 3

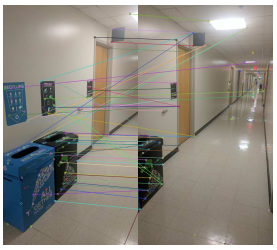

Fig. 76: Matches before RANSAC for test set 3

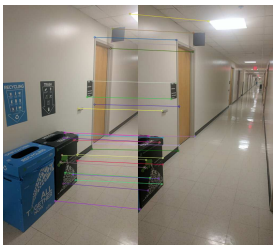

Fig. 77: Matches after RANSAC for test set 3

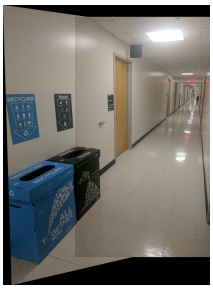

Fig. 78: Intermediate Panoramic image for test set 3

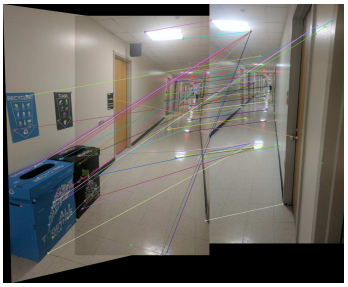

Fig. 79: Matches before RANSAC for test set 3

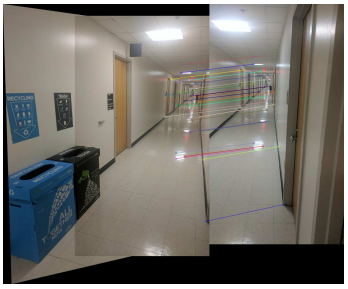

Fig. 80: Matches after RANSAC for test set 3

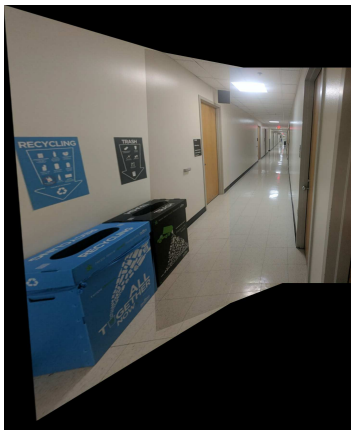

Fig. 81: Final Panoramic image for test set 3

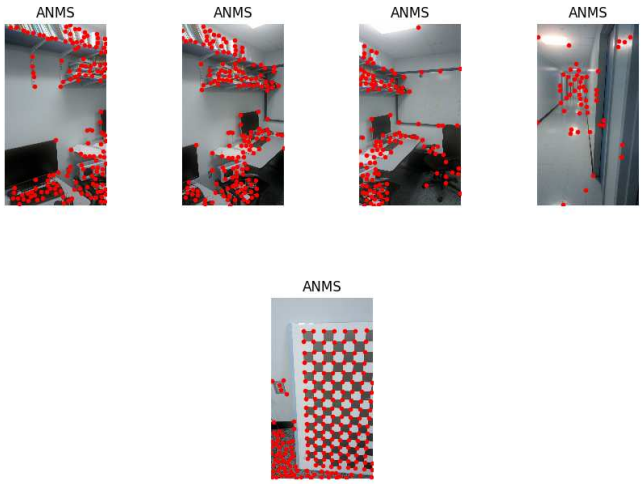

Fig. 82: Corners detected after ANMS for test set 4

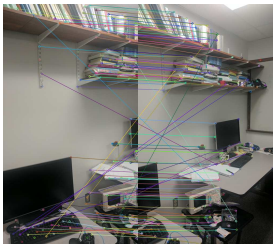

Fig. 83: Matches before RANSAC for test set 4

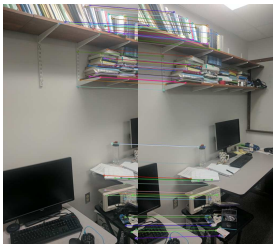

Fig. 84: Matches after RANSAC for test set 4

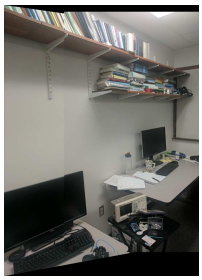

Fig. 85: Intermediate Panoramic image for test set 4

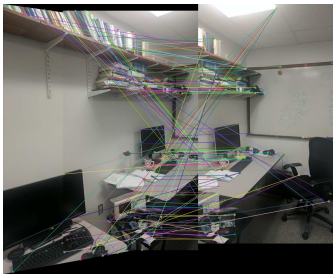

Fig. 86: Matches before RANSAC for test set 4

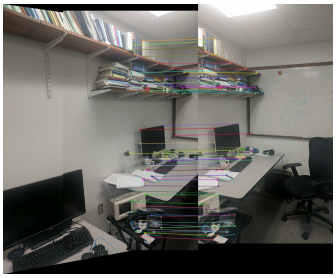

Fig. 87: Matches after RANSAC for test set 4

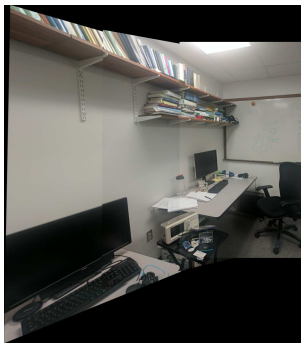

Fig. 88: Intermediate Panoramic image for test set 4

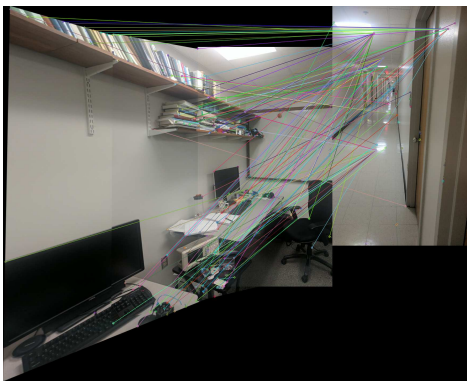

Fig. 89: Matches before RANSAC for test set 4

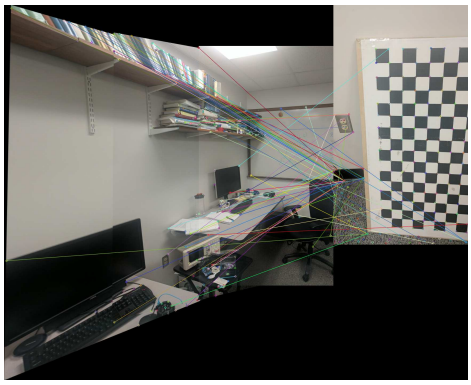

Fig. 90: Matches before RANSAC for test set 4

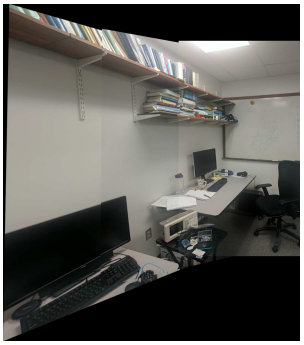

Fig. 91: Final Panoramic image for test set 4

II. Phase 2

In this phase we implemented the deep homography network described in [1]. A convolution neural network is used to estimate the homography between two images. This network has been shown to perform better than the traditional approach in [1]. The homography net can further be used as the homography estimator in the previously mentioned pipeline to stitch different images with overlapping view areas. Both unsupervised and supervised approaches have been implemented in this project.

A. Homography representation

A homography transformation is usually represented by a 3x3 matrix. But the matrix only has 8 degrees of freedom. A homography can also be represented by the corner location changes when it is applied to an image. This method allows for a unique representation of homography which can be trained using a neural network. The homography matrix is computed using cv2.getPerspectiveTransform between corner point of the two patches.

B. Data Generation

First a square from the central region of the training images is cropped. Then the corners are randomly offset by values ranging between 0 and ρ (32 in our case) pixels. Then the homography is calculated from the corner displacement using SVD decomposition of the homography equation.

This homography is then applied to the image. The same area of the new warped image is cropped to get the resultant warped patch of the original image.

We make sure that we pick a patch from the image such that after warping, the resultant patch remains inside the coordinates of the image. We used a margin of 40 pixels on all sides.

We then stack these 2 patches together to create input for our network.

The distance (simple difference) between the corner points of patch A and patch B denotes the 4 point homography between the two images and will be sent as input in the network to calculate loss. The 4 point homography is our label, it shows us the ground truth.

C. Supervised Learning

We used the same architecture as the one in the paper.

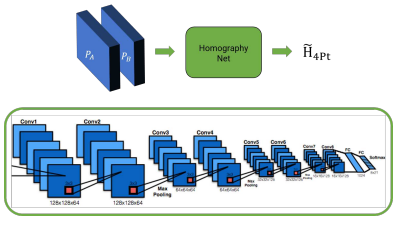

Fig. 92: Network architecture for supervised learning

[1] suggests two approaches for estimating homography using a neural net. We have implemented the regression approach. The neural network takes the input of 2 concatenated arrays and estimates the shift in each corner position during the homography transformation. From this a homography matrix can be calculated. A 128x128 sized patch is cropped for data generation. The regression network directly produces 8 real-valued numbers and uses the Euclidean (L2) loss as the final layer during training. We generated 40000 synthetic data for training from the 5000 images provided. The random perturbations had a maximum value of 32 pixel distance.

The network is similar to the VGGArchitecture where we use 3x3 convolution blocks. Each block has batch normalization and a ReLU added to it. We have 4 blocks with each block having 2 convolution layers and a max pooling layer. This is followed by two fully connected layers. As mentioned in [1], we use the L2 loss for our network with the labels passed as input providing ground truth.

The adam optimizer was used with a constant learning rate of 0.0001 over 50 epochs. We observed that increasing the learning rate resulted in higher testing loss. As mentioned in the above section the output of the network is the corner perturbations, from which the homography was calculated. The output images shown the ground truth perturbation along with the estimated perturbations.

D. Unsupervised Learning

In this approach we initially use the same approach as supervised learning. After getting the 4 point Homography, we generate our 3x3 Homography matrix from it using TensorDLT. We calculate the matrix using corner points of the patches. Once we have the Homography matrix, we apply it to the entire image.

After TensorDLT we use the Spatial Transformation Network. It takes the inverse Homography and applies it to the coordinates of the second image. We calculate the loss of the network and use in back propagation.

E. TensorDLT

TensorDLT converts the corner perturbations to a 3x3 homography matrix. The corner perturbations are formulated as Ax = b and we solve the system of linear equations. Corner points of first patch are used to form the system of equation which are fed into TensorDLT layer. The original image is then warped using this homography.

F. Spatial Transformation

This layer uses the homography estimated by TensorDLT to get the warped pixel coordinates. The photometric loss function is then calculated from the warped pixel coordinates as mentioned below. This layer has three components (1) Normalized inverse computation inverse of the homography estimate; (2) Parameterized Sampling Grid Generator (PSGG); and (3) Differentiable Sampling (DS).

G. Photometric Loss function

A function similar to the traditional homography estimation equation is used as the loss function. L1 error is used because

it has been shown to perform better for image alignment problems.

Both TensorDLT and Spatial Transformer Layer were implemented and the training and testing loss plots are shown below. The testing loss can be improved by including various data augmentation techniques. For this project we have referenced the code provided by the authors of [2]

$$L_{PW} = \frac{1}{|\mathbf{xi}|} \sum \mathbf{xi} \left| I^A \left(\mathcal{H} \left(\mathbf{xi} \right) \right) - I^B \right|$$

H. Network Results

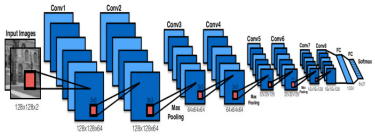

Fig. 93: Network architecture for supervised learning

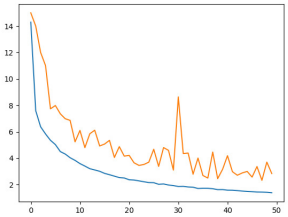

Fig. 94: Training and testing Loss for Supervised network

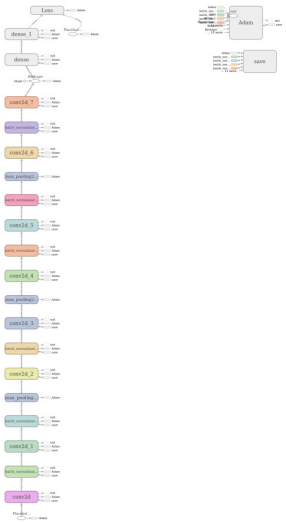

Fig. 95: Network architecture for unsupervised learning

III. Conclusion

Since the deep learning approach is trained on synthetic data, larger data sets can improve performance further. The supervised approach has been shown to be give better results

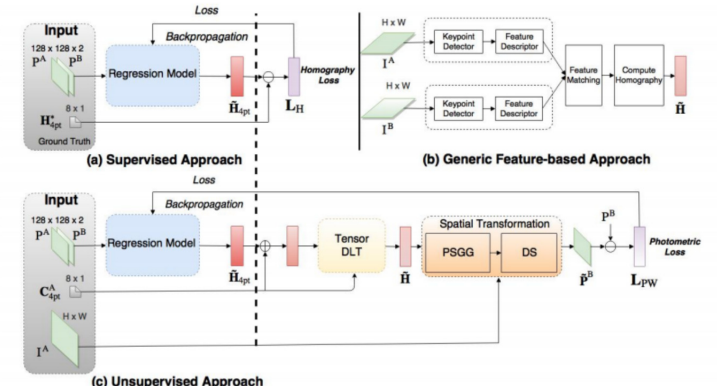

Fig. 96: Approach for supervised and unsupervised learning

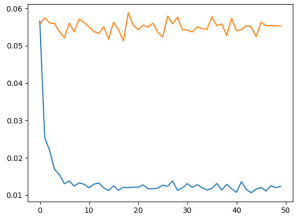

Fig. 97: Training and testing Loss for Unsupervised network

can traditional matching as mentioned in [1]. And through data augmentation the unsupervised approach is also expected to give similar results.

References

[1] DeTone, D., T. Malisiewicz, and A. Rabinovich. "Deep Image Homography Estimation. arXiv 2016." arXiv preprint arXiv:1606.03798.

[2] Ty Nguyen, Steven W Chen, Shreyas S Shivakumar, Camillo Jose Taylor, and Vijay Kumar. Unsupervised deep homography: A fast and robust homography estimation model. IEEE Robotics and Automation Letters, 3(3):2346–2353, 2018.

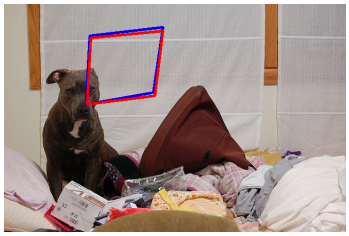

Fig. 98: Sample Output

Fig. 99: Sample Output

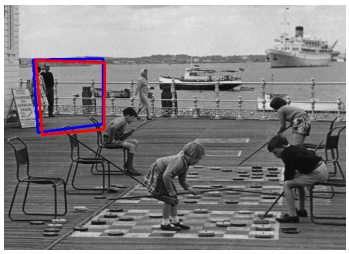

Fig. 100: Sample Output

Fig. 101: Sample Output

Fig. 102: Sample Output

Fig. 103: Sample output