Project 1: MyAutoPano

(Using 1 Late Day.)

Rohitkrishna Nambiar (115507944)
University of Maryland
College Park, Maryland 20740
rohit517@umd.edu

Rohith Jayarajan (115458437)
University of Maryland
College Park, Maryland 20740
rohith23@umd.edu

***Abstract*—The estimation of homography between a pair of images is a problem of interest in the field of computer vision. The purpose of this project is to generate a seamless panorama by stitching two or more images. Both the traditional and deep learning approaches are studied and implemented in this project. The concepts of corner detection, feature descriptors, and feature matching, warping and blending are exploited in the traditional approach whereas all these are combined in the deep learning approach.**

I. Introduction

We begin with the traditional approach for creating a panorama and then later cover supervised and unsupervised methods for homography estimation. Section 2 covers the traditional approach and section 3 covers the Deep learning approach. Section 4 covers experiments and results are covered in section 5. The output for additional test sets and training sets are added towards the end.

II. Phase 1: Traditional Approach

The traditional approach to panorama stitching can be summarized into seven steps as shown in Fig. 1. Each of these steps are explained in detail in the following sections. We will be using Set 1 for explaining the algorithm.

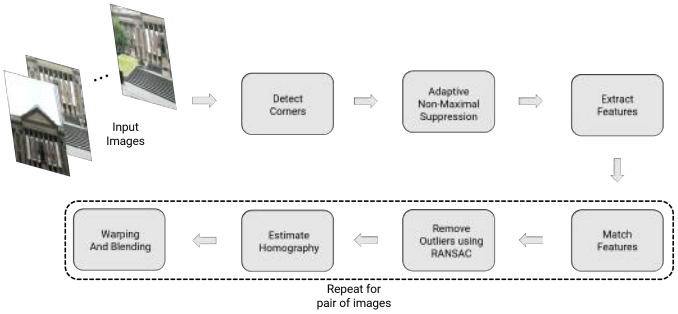
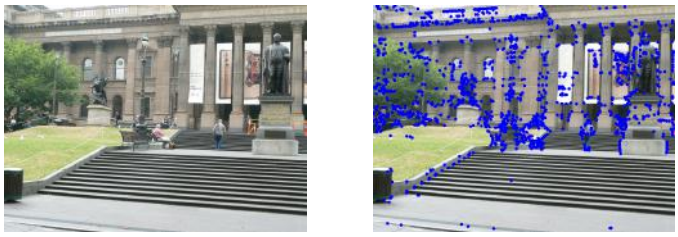


Fig. 1. Pb-lite Algorithm

A. Corner Detection

The first step towards stitching a panorama is to understand how the two images are related geometrically. To do this, we first extract feature points that can be tracked from one image to other. We use corners as feature points and show that these are tracked well across images. Harris corner detector [1] and Shi-Tomasi corners [2] were both tested with the latter one used for final implementation. The input image and the corresponding corners detected by using Shi-Tomasi corner detector is shown in Fig 2.

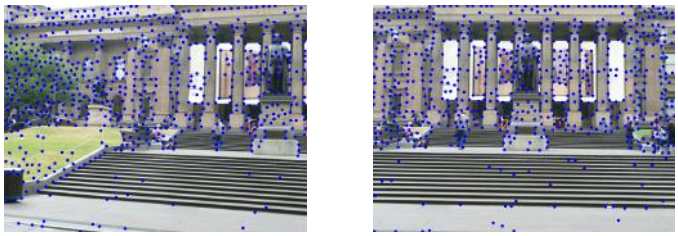


(a) (b)

Fig. 2. (a) Input Image (b) Shi-Tomasi corners.

B. Adaptive Non-Maximal Suppression (ANMS)

From Fig. 2 we observe that without providing any additional parameters to the corner detector, all the corner points are clustered and spaced very close to each other. This can also happen as in an image there can be many strong corner points very close to each other. In these scenario, we want a single strong corner point. We use AMNS to make sure all our corner points are equally spaced so that no undesired artifacts are generated during image warping. For our implementation, we pass additional parameters to Shi-Tomasi corner detectors that specify the number, strength and distance between feature points which acts relatively close to ANMS. The output after ANMS / parameterized Shi-Tomasi detector is shown in Fig. 3. We see that the corners are evenly spaced out.



(a) (b)

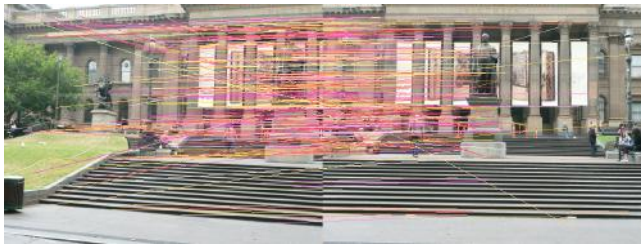
Fig. 3. ANMS output for different images.

C. Feature Descriptor

Once we identify the best features that can be tracked from one image to other, we need to compute a value/descriptor for it such that it can be matched with corresponding feature on the other image. In our implementation to compute the feature descriptor, we first select a 40×40 window around the feature point. Gaussian blur is applied to smooth the patch and remove any noise. The patch is resized to 8×8 and then flattened to obtain a 64 valued feature vector. We also normalize the feature vector which provides some illumination in-variance. We also tested other feature descriptors such as ORB [3] and SIFT [4] for comparison.

D. Feature Matching

Once we obtain the feature vectors for the two images that we want to stitch, we now have to match them. To compute the feature matches, we compare each feature vector in one image to feature vectors in the other image using the *Sum of Squared (SSD)* distances metric. To reduce the number of false matches among feature vectors we use the ratio test which is nothing but the ratio of the value of strongest match to the value of second strongest feature match.



(a)

(b)

Fig. 4. Feature Matching output. (a) ratio = 0.8 (b) ratio = 0.2

In Fig. 4 we see the output of the feature matching algorithm where we change the value of the ratio in the ratio matching test. Here, as we lower the value of the ratio, the matches get better (reduced outliers). Thus the optimum value is a trade-off between the quality and number of matches.

E. RANSAC for robust estimation

To remove outliers and estimate homography robustly, we use RANSAC which stands for **RAN**dom **SA**mple **C**onsensus. The algorithms is as follows:

1) Select four feature points randomly p_i from image 1 and $p_i^{'}$ from image 2.
2) Compute the homography $\boldsymbol{H}$ between p_i and $p_i^{'}$.
3) Using all the other matches we compute the inlier where **SSD**$(p_i, Hp_i < \tau)$, where τ is the threshold for inlier matching.
4) Repeat steps 4-6 until all iterations have been completed or we get inliers higher than set value (ex. 90%).
5) Keep largest set of inliers.
6) Re-compute least square estimate $H^{'}$ based on highest inlier points.

The output of the RANSAC algorithm can be seen in Fig. 5

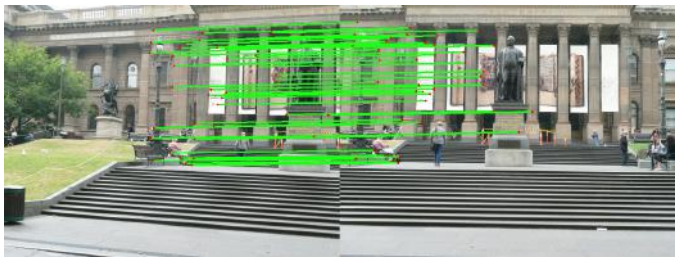


Fig. 5. RANSAC estimation and outlier rejection

F. Blending Images

Once we get the homography between the two images, we warp image 1 with respect to image 2 and blend them. For blending, we have simply used the averaging technique. Further, warping from the left helps to append images in a series and not have undesirable artifacts. Other approaches such as warping from the center to either sides can also be successful in stitching panorama with large number of images. We also assume that the images are in sequence. To automate this, we can compare features among all the images to estimate the sequence.

Fig. 6. Stitching with 2 images and average blending

Fig. 7 shows the input image sequence to the algorithm. Fig. 8 shows the output panorama image. Other images in Train set and Test set are added in the appendix.

III. Phase 2: Deep Learning Approach

In the traditional approach mentioned in Phase 1, which followed a pipeline of combining the process of corner detec-

(a)

(b)

(c)

Fig. 7. Input Images

Fig. 8. Image panorama with 3 images

tion, Adaptive Non Maximal Suppression, feature extraction, feature matching and robust estimation of homography using RANSAC. The deep learning approach to solve this problem combines all the above mentioned steps of the traditional approach to estimate the homography between two images. It can be observed that this approach is a faster one and also robust if the network is generalizable. Below described are the implementation details and specifics of this deep learning approach which were described in the paper [5].

A. Data Generation

To train any convolutional neural network we need data and for homography estimation using deep learning, we need to generate data. Each data point for our model consists of a pair of images and the value of homography between them. For this, we generate images that are synthetic parirs. To ensre that the network is not biased we use the images from the MSCOCO dataset which is a set of large number of images.

For image data generation, we need the two images to be of the same size. The following are the steps in the data generation phase:

1) Crop a random patch P_A of size 128×128 where the image has both the row and column dimensions greater than or equal to 128. This is to ensure that all pixels in the patch lies within the image after performing warping operation on the extracted patch. The original image I_A is shown in 9 and the cropped patch P_A is shown in 10.

Fig. 9. Original image I_A.

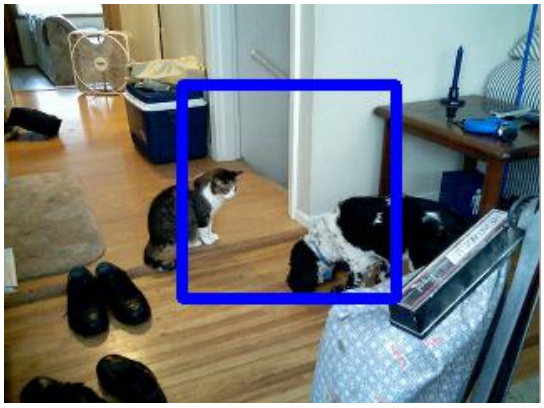


Fig. 10. Corners for patch P_A in blue.

2) Now we have patch P_A with four known corners. We need to perturb these four corners by a value in the range $[-\rho, \rho]$. In our case, the value of ρ is 32. The four new corners give us a closed patch. The perturbed corners of P_A is in green as shown in figure 11.
3) Now warp the original image I_A with the inverse of the transformation matrix between the corners in patch P_A and the perturbed corners from P_A to get image I_B. Cropped Patch P_A is shown in figure **??**.
4) Crop image I_B with the corners to be the same as those were in P_A to get patch P_B. Cropped Patch P_B is shown in figure 12. Another example of patches is shown in 13

After following the above steps, we have two images and the corresponding homography between them. We do not compute the homography for the label but the difference in the coordinates of the corners in P_A and P_B which is H_{4Pt}. Now we stack these two image patches to get an input of size

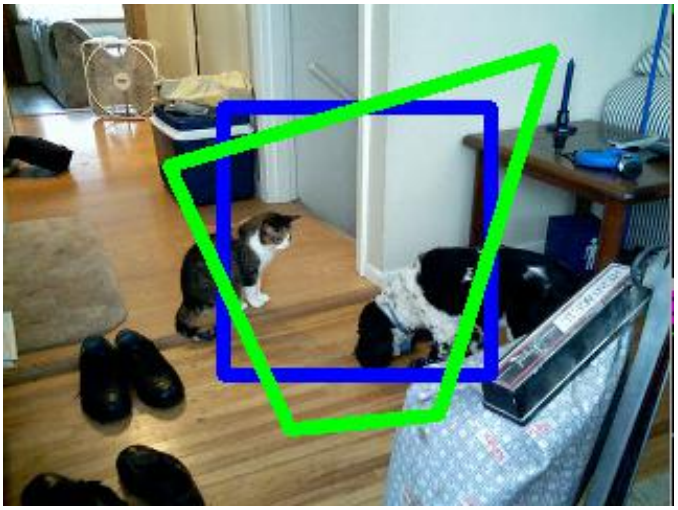


Fig. 11. Perturbed corners for patch P_B in green.

(a) (b)

Fig. 12. (a) Patch P_A (b) Patch P_B.

$128 \times 128 \times 2K$ where K is the number of channels in each patch.

B. Supervised Learning Approach

The overview of the deep learning based method is shown in figure 14. Given two images, the HomographyNet gives the output of H_{4Pt} from which the homography can be extracted using DLT.

(a) (b)

Fig. 13. (a) Patch P_A (b) Patch P_B.

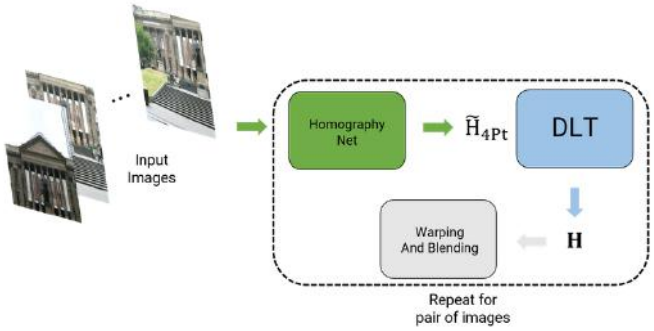


Fig. 14. Overview of HomographyNet.

The architecture of the HomographyNet has 8 convolutional layers of kernel size 64, 64, 64, 64, 128, 128, 128, 128 with batch normalization and ReLU activation after each convolution layer, two fully connected layers with 8 real valued numbers as output. Dropouts with probability 0.5 are applied after the final convolutional layer and the first fully connected layer. The architecture of the HomographyNet can be seen in figure 16 and each convolutional block has a structure as shown in

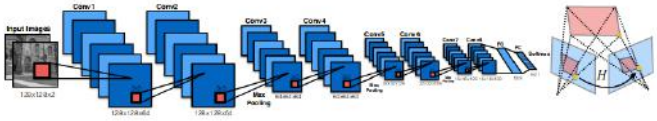


Fig. 15. Architecture of HomographyNet.

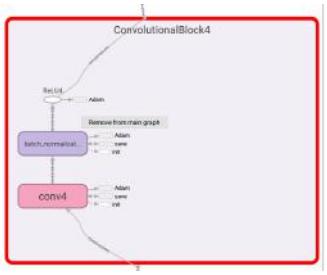


Fig. 16. Convolutional block in HomographyNet.

The loss function used is the Euclidean (L2) loss and the optimizer is ADAM with learning rate of 1e-4. The graph statistics of the network as given in TensorBoard can be visualized in Fig 17. The loss per iteration and loss per epoch in the training phase as well as validation loss are shown in figure 18.

The histrogram plot of the predicted H_{4Pt} by the network can be seen in Fig 19 and 20 and distribution plot in Fig. 21.

IV. Experiments

A. Traditional Approach for Homography

The arguments for *cv2.goodFeaturesToTrack* was 700 maximum corners and 0.01 as quality level. The ratio between first

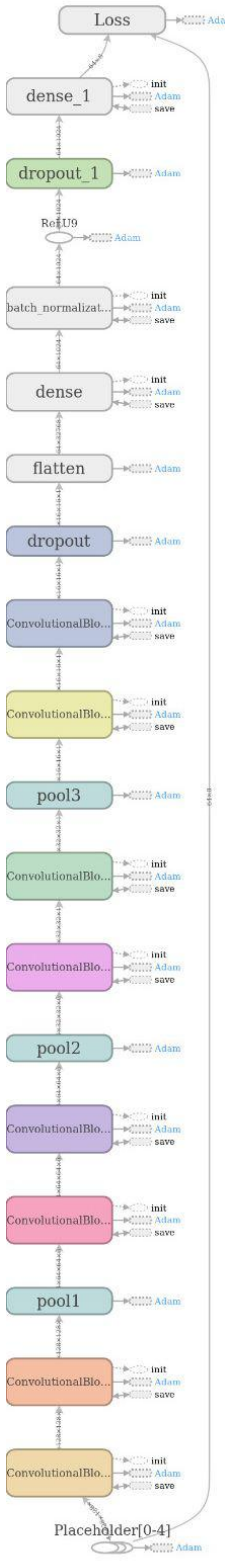


Fig. 17. TensorBoard graph of HomographyNet.

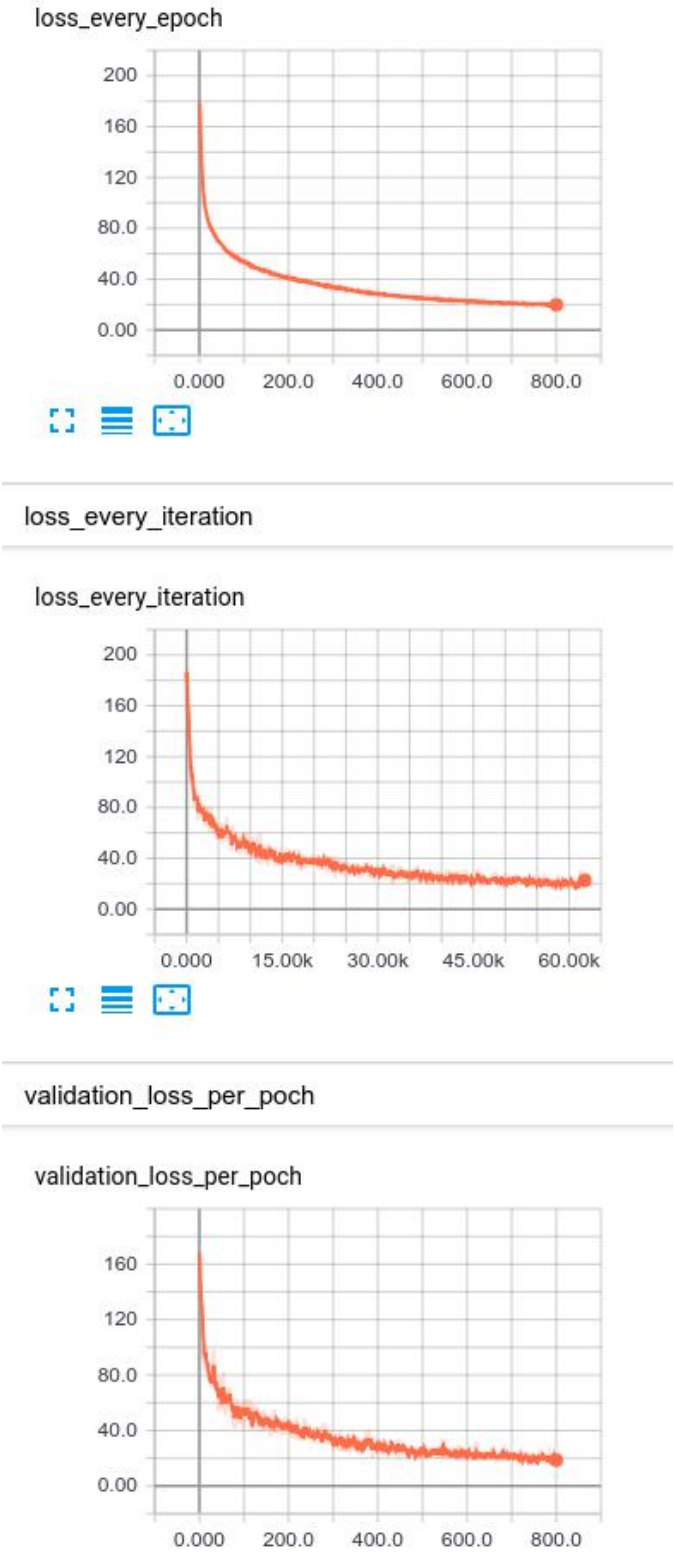


Fig. 18. TensorBoard training loss plot of HomographyNet per iteration.

best and second best matches in feature descriptor was set to 0.5 i.e. if the ratio is less than 0.5, then the feature pair is accepted. RANSAC was run for 1000 iteration or 90 percent

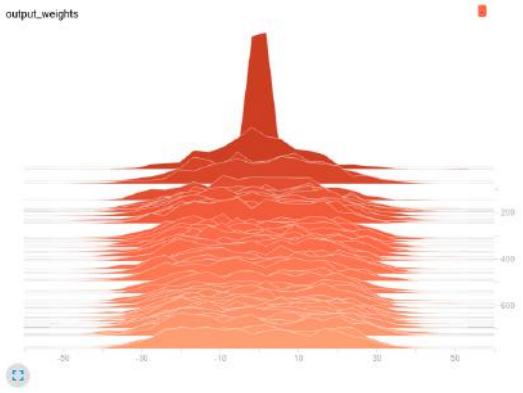


Fig. 19. TensorBoard histogram plot of HomographyNet output.

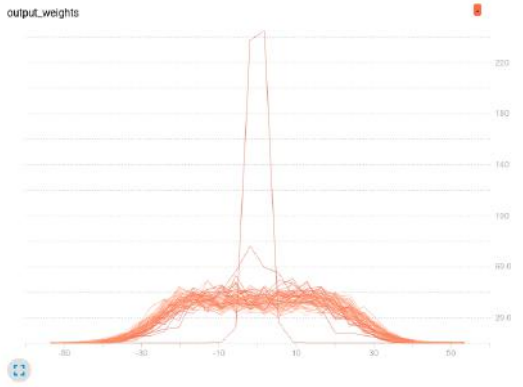


Fig. 20. TensorBoard histogram plot overlay of HomographyNet output.

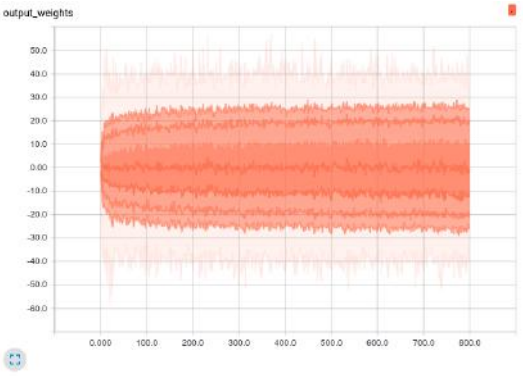


Fig. 21. TensorBoard histogram distribution of HomographyNet output.

of all matches were in inliers, whichever occurred first. The SSD value of less than 0.5 indicated a the presence of an inlier for that particular estimate of a homography.

B. Supervised Learning: HomographyNet

The network mentioned in section III-B was trained for 800 epochs with a minibatch size of 64 on a GTX1060 GPU.

V. Results

A. Traditional Approach for Panorama Stitching

The results for the images 22 in Test Set 3 is shown in figure 23. Results for other test sets and train sets for Phase 1 are added in the Appendix A.

(a) (b) (c)

Fig. 22. Input Images Test Set 3

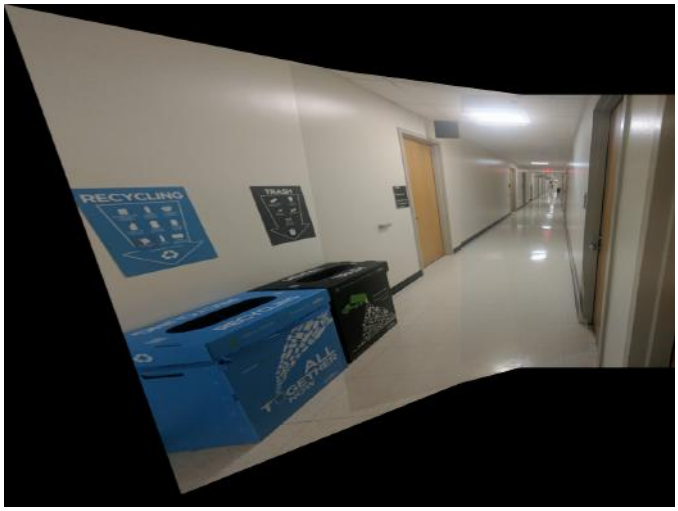


Fig. 23. Test Set 3 Panorama Output

B. Supervised Learning: HomographyNet

The results from HomographyNet are shown in figure 24, 25, 26 and 27. As mentioned in the images, the green bounding box has the perturbed corners as its vertices which is the ground truth. The values predicted by the HomographyNet form the vertices of the red bounding box. Other images tested on the HomographyNet are shown in the Appendix A. Table I shows the statistics for the network.

Data	Num Images	L1 Loss	L2 Loss	Fwd Pass Runtime
Train Set	5000	29.918	211.204	3.145ms (average)
Validation Set	1000	29.409	200.915	3.96ms (average)
Test Set	1000	29.801	211.509	3.942ms (average)

TABLE I
HomographyNet Statistics

Table I shows the L1 loss, L2 loss, number of Images, and the runtime for the forward pass operation for each Train, Validation and Test set. The L1 and L2 loss are in pixels (can be rounded off to nearest integers).

Fig. 24. Results from test set for HomographyNet

Fig. 25. Results from test set for HomographyNet

VI. Conclusion

We have implemented the traditional approach to panorama stitching as well as used deep learning methods to estimate homography. From our results we see that although traditional approach works well, the number of steps and parameters required to tweak are more. Deep Learning approach takes that away as we invest the time to train the network once.

Acknowledgment

We would like to thank Prof. Yiannis and the TA's Nitin and Chahat for their help with the project.

Fig. 26. Results from test set for HomographyNet

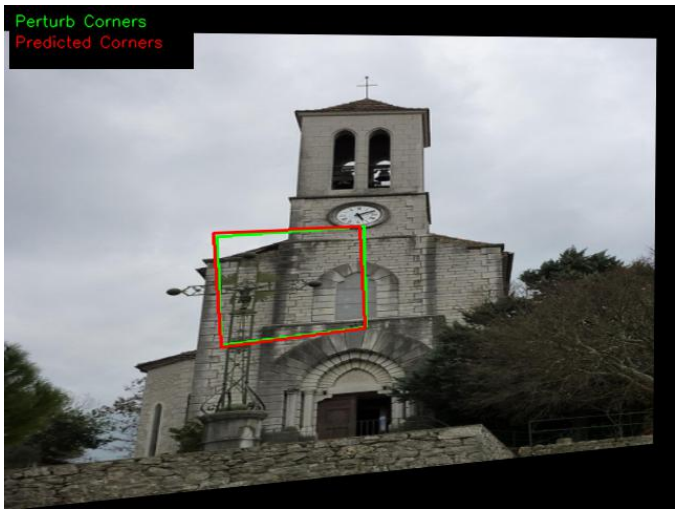


Fig. 27. Results from test set for HomographyNet

References

[1] C. G. Harris, M. Stephens *et al.*, "A combined corner and edge detector." in *Alvey vision conference*, vol. 15, no. 50. Citeseer, 1988, pp. 10–5244.
[2] J. Shi and C. Tomasi, "Good features to track," Cornell University, Tech. Rep., 1993.
[3] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: An efficient alternative to sift or surf," 2011.
[4] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.
[5] D. DeTone, T. Malisiewicz, and A. Rabinovich, "Deep image homography estimation," *arXiv preprint arXiv:1606.03798*, 2016.

APPENDIX A
TRADITIONAL METHOD OUTPUTS

The following image are outputs for the test and train sets for Panorama stitching using the traditional method.

A. Train Set 2

Train set two images are shown in Fig 28. The output panorama is shown in Fig 29.

(a) (b) (c)

Fig. 28. Input Images Train Set 2

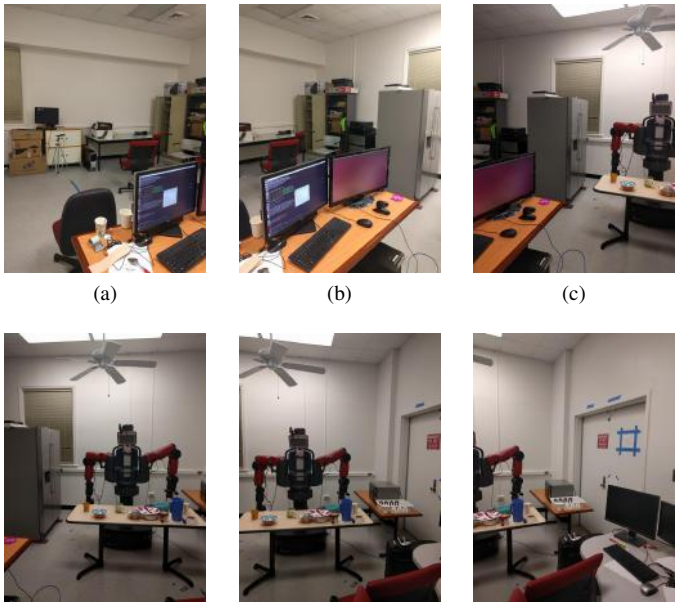
Fig. 29. Train Set 2 Panorama Output

B. Train Set 3

In this set, as the planar assumption fails, the image fails to stitch. Thus we have only created the panorama for 3 images (c), (d) and (e) from Fig 30. The output panorama can be seen in Fig 31.

C. Test Set 2

In this set, as the planar assumption fails, the image fails to stitch. Thus we have created two sets of panoramas as shown in Fig 33 and 34.



(a) (b) (c)

(d) (e) (f)

(g) (h)

Fig. 30. Input Images Train Set 3

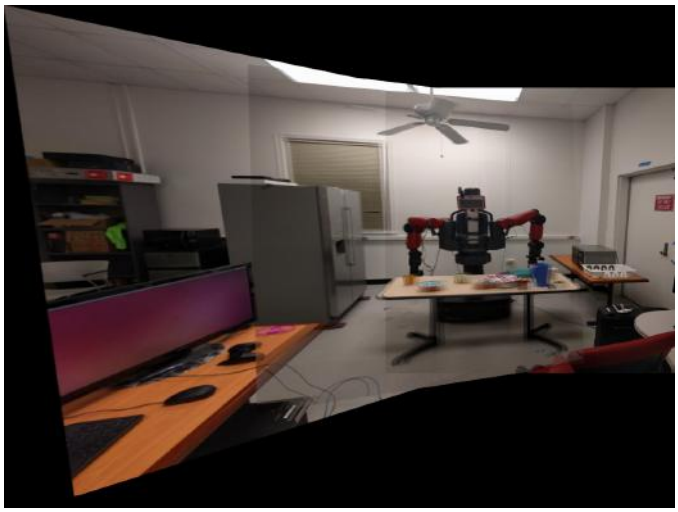
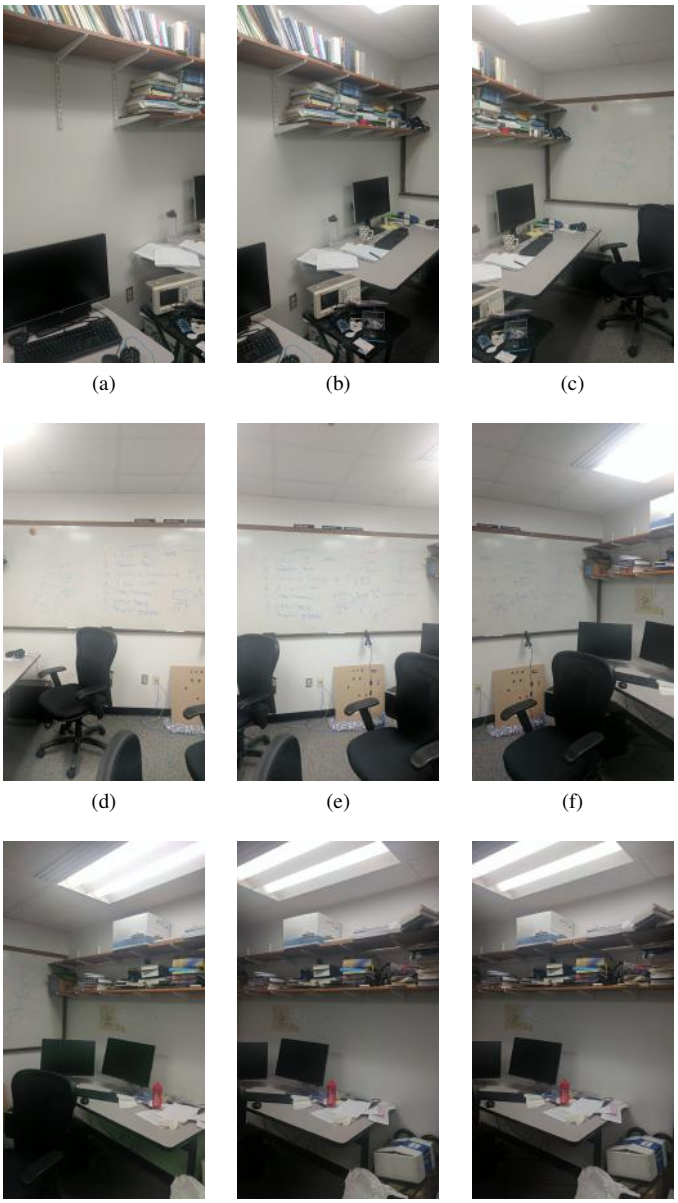


Fig. 31. Train Set 3 Panorama Output



(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

Fig. 32. Input Images Train Set 2

D. Test Set 1 and Test Set 4

Test Set 1 fails to work as the checkerboard images cannot be matched accurately. This maybe caused as all the corners might have the same scores. For Test Set 4, the images are same as Test Set 2.

E. HomographyNet Results

Here we additionally tested 4 images on our supervised network. As mentioned in the images, the green bounding box has the perturbed corners as its vertices which is the ground truth. The values predicted by the HomographyNet form the vertices of the red bounding box.

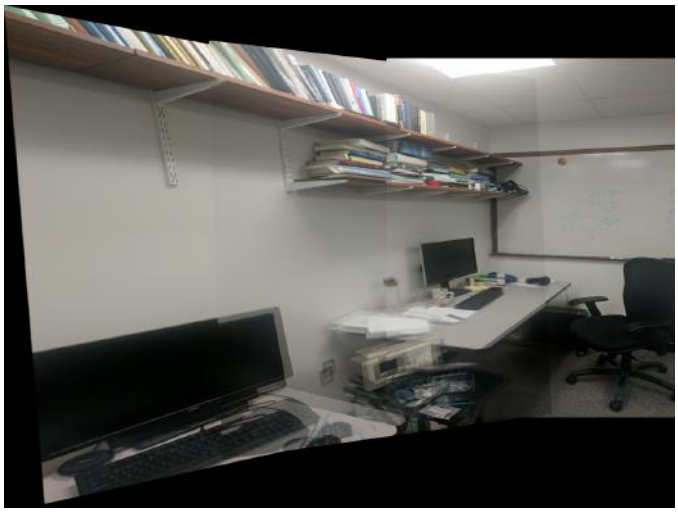


Fig. 33. Test Set 2 Panorama Output 1

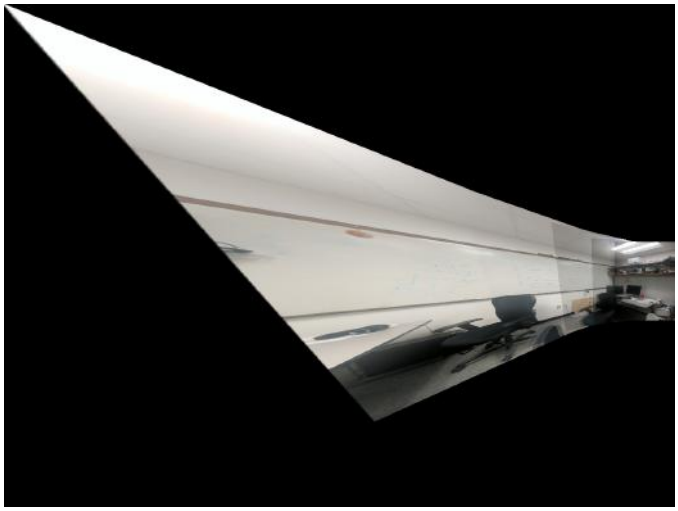


Fig. 34. Test Set 2 Panorama Output 2

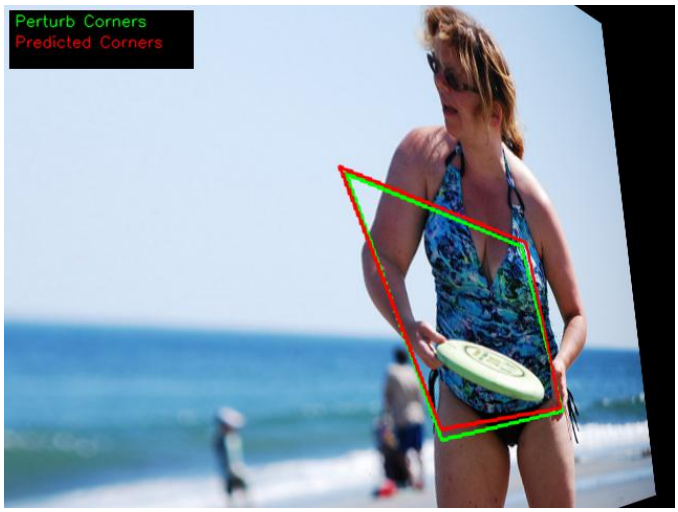


Fig. 35. Results from test set for HomographyNet

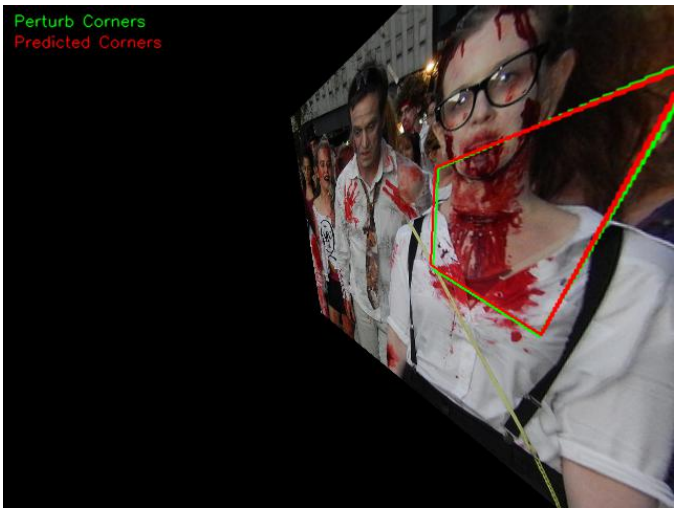


Fig. 36. Results from test set for HomographyNet

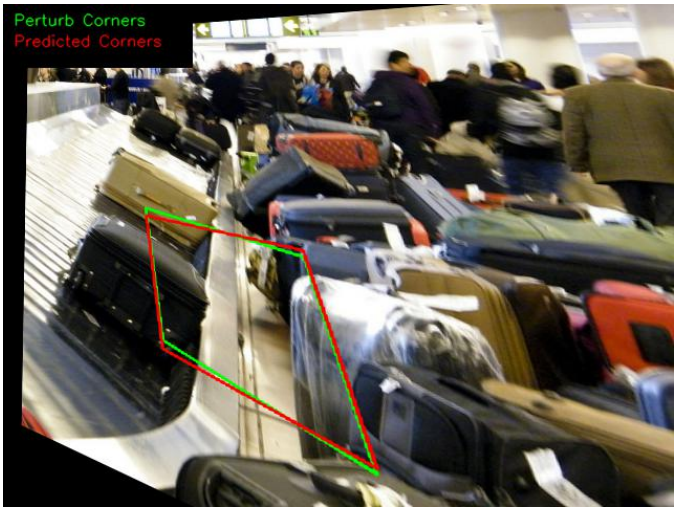


Fig. 37. Results from test set for HomographyNet

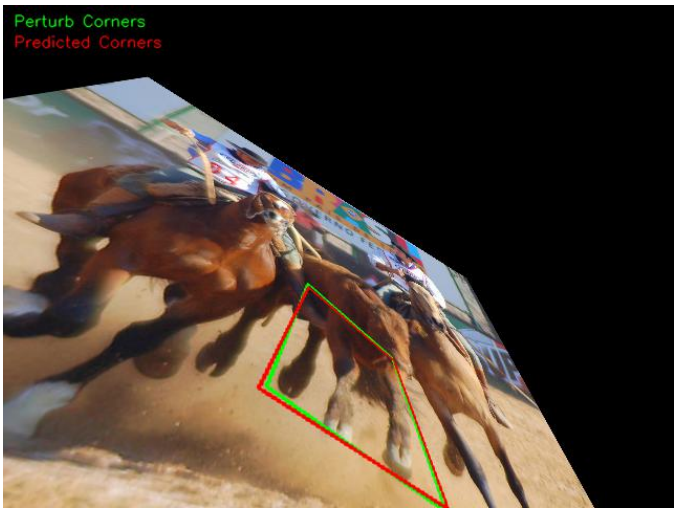


Fig. 38. Results from test set for HomographyNet