Project 1: Auto Pano

Abhinav Modi
Masters of Engineering in Robotics
University of Maryland, College Park
Email: abhi1625@umd.edu

Kartik Madhira
Masters of Engineering in Robotics
University of Maryland, College Park
Email: kmadhira@terpmail.umd.edu

Prateek Arora
Masters of Engineering in Robotics
University of Maryland, College Park
Email: pratique@terpmail.umd.edu

I. INTRODUCTION

Homography between 2 image frames is one of the most significant concept in Computer Vision. It holds many applications in the fields of Image Processing and Robotics. It is defined as the transformation between two planes of interest. In this project we use Homography to warp multiple images and stitch a panorama using three different techniques- Traditional approach using feature matching and RANSAC, Supervised approach to predict a 4 point parametrization of Homography between two images and an Unsupervised approach to predict Homography without the presence of a ground truth. These approaches have been briefed in the sections that follow.

II. PHASE 1: PANORAMA STITCHING USING TRADITIONAL APPROACH

The Traditional approach can be separated into 5 steps. Sequentially, these are corner/feature detection, Adaptive Non-maximal Suppression(ANMS), Feature Description and Matching, RANSAC for outlier rejection to estimate robust Homography, and finally blending the images. The underlying assumption in creating a panorama by this approach is that there is about 60-70% overlap in the two consecutive images. We demonstrate this algorithm using the two images from a Custom Set created from an image of Yosemite mountains, shown in figure(62):

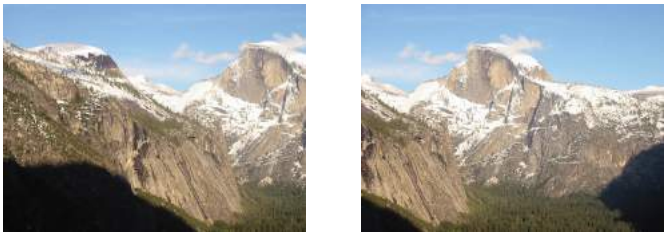


Figure 1: Two sequential images from left to right for creating the panorama

A. Corner/ Feature Detection

The first step in stitching a panorama is detecting features in an image. We detect the Shi-Tomasi features in the image using **cv2.goodFeaturesToTrack()**. We can also use Harris corners as the required features but Shi-Tomasi gives us more of the good quality features to work with. The output of corner detection for above two images can be seen in the figure(2).

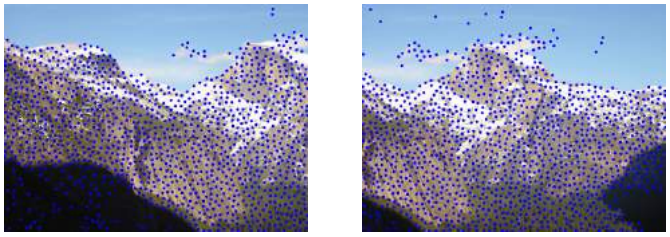


Figure 2: Two sequential images from left to right for creating the panorama

B. Adaptive Non Maximal Suppression(ANMS)

In this step we find the N_{best} corners in the image. This done because in a real image, a corner is never perfectly sharp and it might get a lot of hits in the previous step. The steps involved in ANMS are as follows:

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
```

Figure 3: ANMS algorithm

The final list is then sorted in descending order and N_{best} corners are selected. The output after ANMS is shown in figure(4)

C. Feature Description and Feature Matching

In this part, we describe each feature point by a feature vector. We first take a 40×40 patch around the keypoint. Then we apply a gaussian blur on the patch and down sample blurred patch to a 8×8 matrix. Then we reshape the matrix to a column vector and standardize it to remove bias and

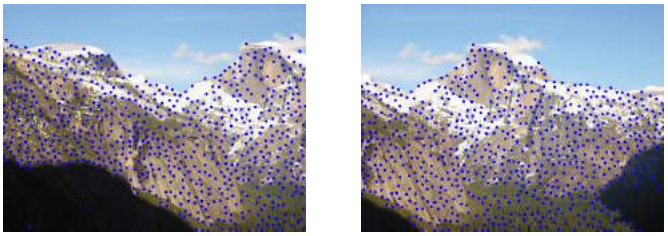


Figure 4: Showing Corners detected in the two images

illumination effect. The first four feature descriptors in the list obtained from ANMS for the first image can be seen in the figure(5)

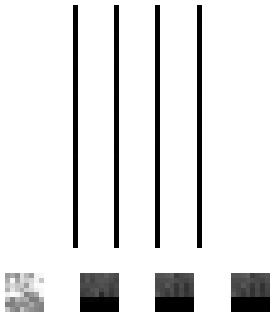


Figure 5: 4 Feature vectors and their corresponding 8×8 patches from left to right

The rest of the feature descriptors and 8×8 patches are not shown in the report as they are very large in number. But they have been stored in their corresponding Set folder in the zip file submitted. Path: **'./Phase1/Code/Results/*'**

Once each keypoint is encoded by a 64×1 vector in every image, we compute match pairs in two different images of 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. A threshold of 0.5 is set on the ratio of lowest and second to lowest SSD for each feature point in the first image to determine if the matched pair should be saved. If this ratio is less than 0.5 then the pair is rejected and rest of the pairs are saved. Note, we also set a flag which returns an error in case there are less than 45 matches between 2 images, because these images cannot be stitched. This number is chosen arbitrarily, feel free to play around with it.

Figure 6: Output of Feature Matching

D. Random Sample Consensus: RANSAC

The feature matching output also contains some incorrect pairs which should be removed to obtain a more accurate Homography. RANSAC provides us a method to do that. The input of RANSAC are the coordinates of matched pairs obtained in the previous step, desired number of robust pairs (we used a 90% probability of inliers), a threshold = 30 representing the sum-squared distance between estimated feature location and actual feature location, and the maximum number of iterations N max = 3000. We follow the steps: randomly selecting 4 feature pairs, computing exact the Homography matrix H, computing the SSD between estimated points after transformation and actual points in image 2. This algorithm is repeated for N_{max} times to find the largest set of inliers. Now, the final Homography to be applied is calculated from this set of inliers. The output obtained from the RANSAC is shown in the

Figure 7: Output after RANSAC

E. Stitching and Blending

After obtaining homography between the images, we perform warping and stitching. The stitching is performed in a sequential manner i.e., the images are selected in order from left to right and/or top to bottom. This is done in the following way:

- Compute homography between $N-1$ and N^{th} image. Here $N = 2, 3..$
- Apply this homography to the 4 corner points of the $N-1^{th}$ image and obtain the minimum X and Y translation of the image. Then, remove this offset from the homography matrix such that top-left corner of the $N-1^{th}$ image is $(0, 0)$ and warp the $N-1$ image using this new homography matrix.
- A bigger frame is then constructed, which will contain the stitched images for the panorama, by adding the sizes of the warped$(N-1)$ image and the N^{th} image and place these images in this frame.
- Use the pano image created in the previous step and the $N+1^{th}$ image to the repeat the steps 1-3 and voila! you have a panorama.

Once the final panorama image is obtained, a final Gaussian filter of size (5,5) is applied to the image and thus .The output panoramas for the images given in the Train Sets are shown in figures(8),(9),(65). A Custom set was also created using the images from the Yosemite mountains(12).

Figure 8: Panorama for images from Train Set1

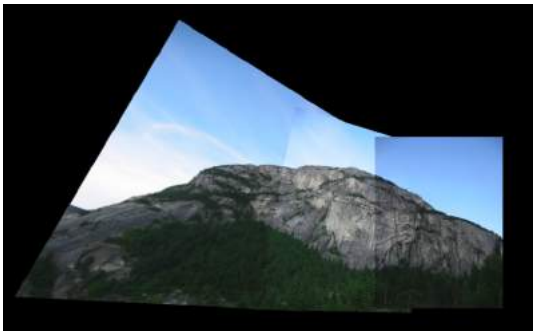


Figure 9: Panorama for images from Train Set2

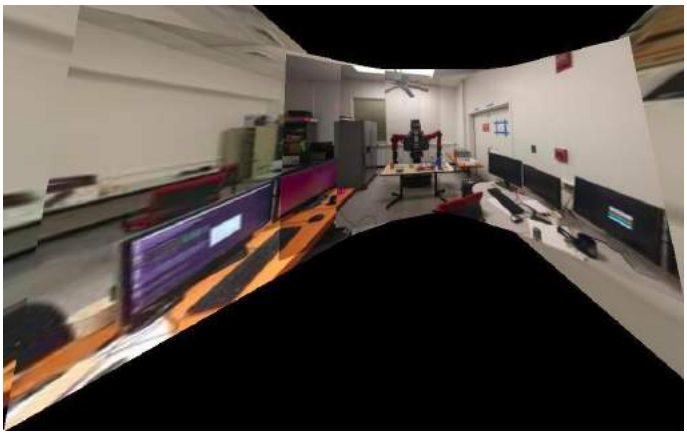


Figure 10: Panorama for images from Train Set3

F. Discussion and Conclusion

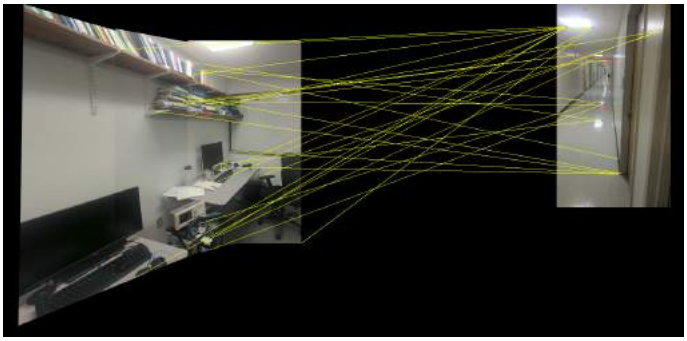
In this section we created a panorama using a traditional approach by matching features between two images. This approach gives satisfactory results which can still be improved using additional techniques like blending or averaging the pixel values of the two images at the same location while stitching. Also, the algorithm produces better results when all the image planes are at infinity. As seen in some cases the estimated homography is not very accurate. This happens because the both image planes are not at the same depth. In such cases better results can be produced by projecting these images on a cylinder and then performing the stitching.

Figure 11: Panorama for images from Custom Set1

Figure 12: Panorama for the images form Test Set 1

The stitching algorithm fails for a cyclic order of images as can be seen in the figure(12). Thus, a more robust algorithm for stitching the images is required. Still, the algorithm is robust for cases when there are less than a required set of matched pairs. For this project we have kept this number 45. This is an arbitrary choice and works well for most of the cases. This can be seen for the Test Set 4. The figure(**??**) shows the number of matches obtained after feature matching. If we apply RANSAC to this case it leaves us with 4 matched pairs which are corresponding pairs of the random points chosen while computing the initial homography during RANSAC. This shows that there is no/very little overlap present between the two images. Thus the algorithm is robust enough to handle such cases and rejects such images during the stitching process.

Some of the intermediate results were saved for all the sets of images available to us. These results are provided in the *Appendix* Section in the end of the document.



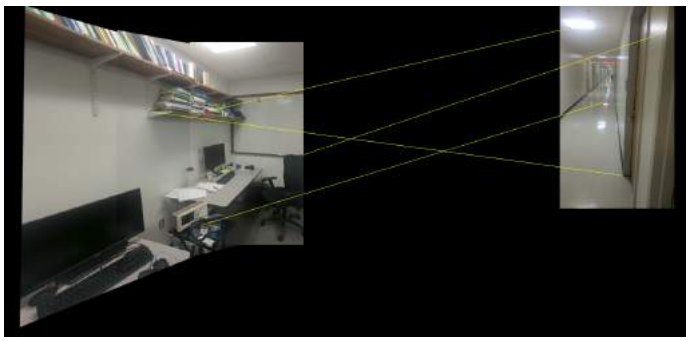
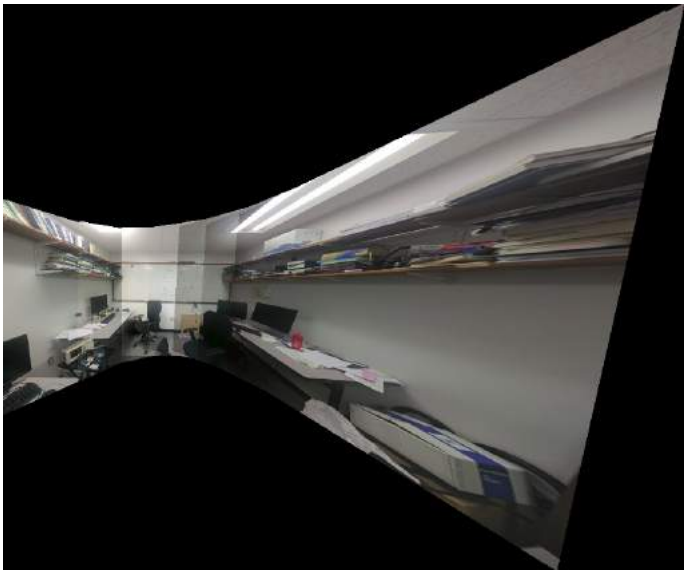


Figure 13: Panorama for the images form Test Set 2 and Test Set 3



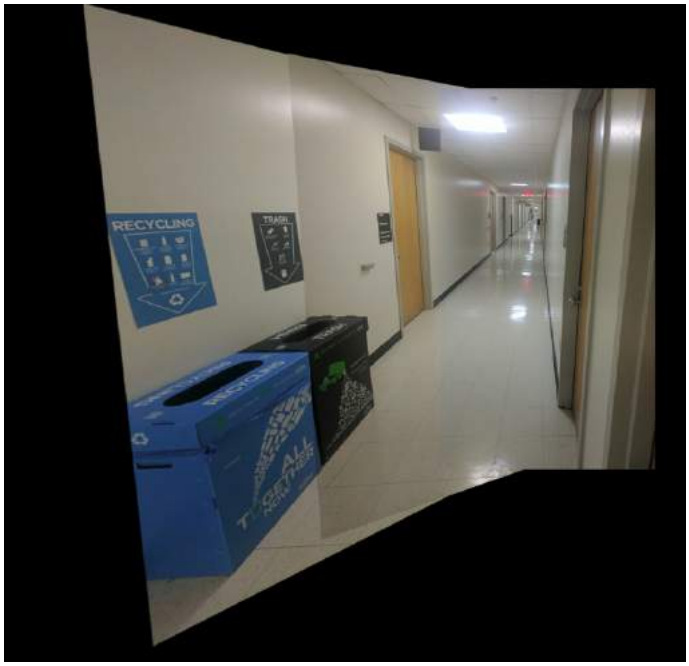


Figure 14: Panorama for the images form Test Set 2 and Test Set 3

III. PHASE 2:Deep Learning Approach

In this method we implement the entire pipeline of traditional Panorama stitching using deep learning. Deep learning here is used to compute the homography matrix between several images. Robust and fast Homography estimation is required applications, especially in robotics, to estimate pose between multiple aerial images for collaborative autonomous exploration and monitoring. Both supervised and unsupervised approach have been implemented to compute the homography. The empirical results presented in the paper demonstrate that compared to traditional approaches, the unsupervised algorithm achieves faster inference speed, while maintaining comparable or better accuracy and robustness to illumination variation. Furthermore, the unsupervised method has superior adaptability and performance compared to the corresponding supervised deep learning method.

A. Data Generation

In order to train the networks we create synthetic dataset. For the supervised approach we generate synthetic image data set with corresponding labels. Here labels are our ground truth which is only used while training the supervised network. We generate the synthetic data set from MSCOCO dataset which contains a lot of objects in natural images. Since the convolution doesn't accept image sizes of arbitrary shape we crop two patches from the same image namely patch A and patch B. In order to to get a complete patch of dimensions 128x128, the left top corner of a patch can only be placed in a limited area which is shown in figure 15.

The first patch, patch A is a square patch of the dimension 128x128 as shown in figure 16. Then, in order to create a different patch(or in a sense image) the corners of the first patch are perturbed i.e each corner point's location $\boldsymbol{x_i}$=(x_i,y_i) is changed by a small amount in the range [-ρ,ρ] where ρ is an arbitrarily chosen perturbation. In our case the value of ρ

Figure 15: Active Region indicated by highlighted area [1]

Figure 16: Patch A marked in red and Patch B(perturbed) marked in blue [1]

selected is 16. The second patch, namely Patch B is shown in figure 16. This perturbation is saved as ground truth for our supervised model. The perturbed patch is not a perfect square of dimension 128x128(dimensions accepted by our network) and thus, to get a perfect square patch, we inverse warp the original image as shown in figure 17 it using the homography matrix. This homography matrix is computed using cv2.getPerspectiveTransform between corner point of patch A and corner points of patch B(which is equal to sum of each corner point of patch A and its respective perturbation)

The following data fields have been saved for supervised approach :

- Patch1 and Patch2 stack
- $\tilde{\mathbf{H}}_{\mathbf{4pt}}$ ground truth

For the Unsupervised part data fields saved are as followed:

- Patch1 and Patch2 stack
- Corner points of patch in the first image (C4_A)
- Reference to original image from which the patch was cropped

Figure 17: Patch A marked in red and Patch B marked in blue after the image has been warped image

B. Supervised Approach

In the paper the task of computing homography is formulated as a classification problem and regression problem and here we have implemented the regression network. The only difference is that in the classification we discretize the output into 8 points and 21 bins (8x21 output) while in the regression one the output is 4 point parameterization of homography .

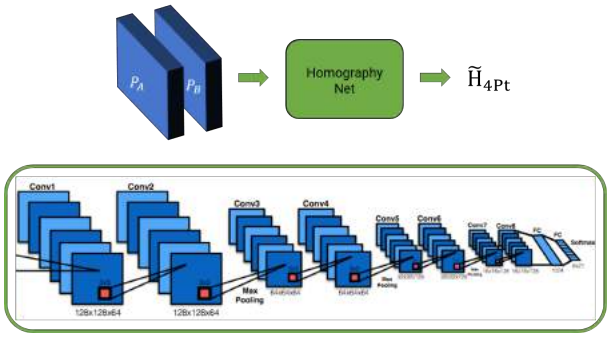


Figure 18: Architecture of Supervised HomographyNet [1]

The networks use 3x3 convolutional blocks with Batch normalization and ReLUs, this is very similar to the VGG architecture. Both networks take as input a two-channel gray scale image sized 128x128x2. In other words, the two input images, which are related by a homography, are stacked channel-wise and fed into the network. We use 8 convolutional layers with a max pooling layer (2x2, stride 2) after every two convolutions. The 8 convolutional layers have the following number of filters per layer: 64, 64, 64, 64, 128, 128, 128, 128. The convolutional layers are followed by two fully connected layers. The first fully connected layer has 1024 units.The regression network directly produces 8 real-valued numbers as shown in figure 21 and uses the Euclidean (L2) loss as the final layer during training. [2]

Approximately 40000 images were generated for the training of the network from train set of 5000 images of the MSCOCO dataset. Owing to the nature of HomographyNet infinite number of images can be generated from a finite set of train images. The perturbation values were set to random values between -16 and +16.

C. Result-Supervised

In our network we use Adam optimizer as the optimizer with a learning rate of 0.0001 with a batch size of 128. For training we initially intended to run for 400 epochs but owing to plateauing of the loss of the network we stopped the training at 200 epochs.Also, training the network on 40000 images for 200 epochs took approximately 8 hours.

The final graph obtained with the updated weights takes an average of 0.69 seconds to produce a 4-point parametrized output between two images that are related by a homography matrix. To perform an initial sanity check on the output of the network we input two equal patches(patchA=patchB). The resultant warping of the first image should be same image as the first image itself and is also what we get from the network(See fig. 16).

For getting the homography matrix between the images from the 4-point output of the network we take the corner locations of the original image and then add the 4-point output to this to get the corner locations in the second image. Now that we have corresponding corner locations in the images respectively, we calculate the homography between them. Outputs for train, validation and test datasets can be seen in figs. 21, 22 and 23 respectively.

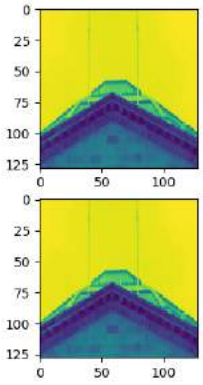


Figure 19: Two equal patches being input into the network for sanity check

1) *Train set Homography Accuracy:*
2) *Validation set Homography Accuracy:*
3) *Test set Homography Accuracy:*

D. Unsupervised Approach

While the supervised deep learning method has promising results, it is limited in real world applications since it requires ground truth labels. Unsupervised approach, on the other hand, achieves faster inference speed, while maintaining comparable or better accuracy and robustness to illumination variation. In addition, the unsupervised method has superior adaptability and performance compared to the corresponding supervised deep learning method.

The architecture of the Unsupervised network is shown in figure 28. First part of the Unsupervised network is exactly the same as supervised network which is followed by TensorDLT and Spatial Transformation Layer. The main contribution of the paper is the part after getting $\tilde{\mathbf{H}}_{\mathbf{4pt}}$, i.e. TensorDLT and Spatial tranformer, which are both differentiable functions.

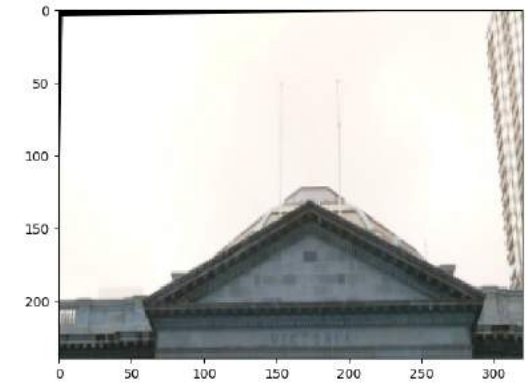


Figure 20: Output of the sanity check warped image with near zero error

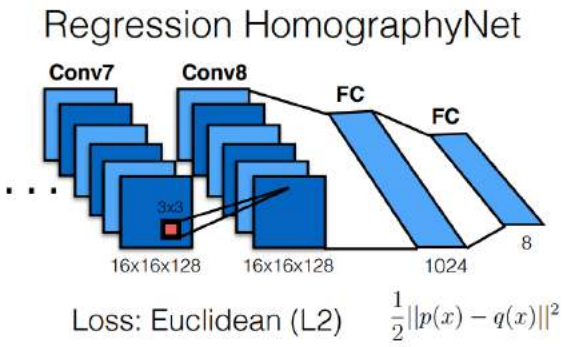


Figure 21: Regression HomographyNet

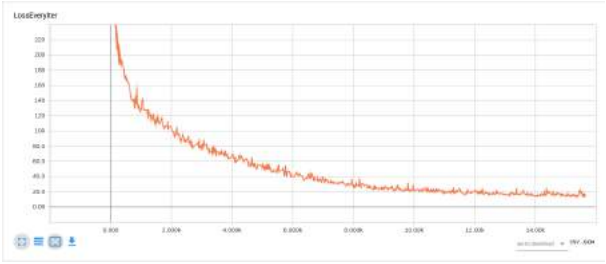


Figure 22: Loss per Iteration

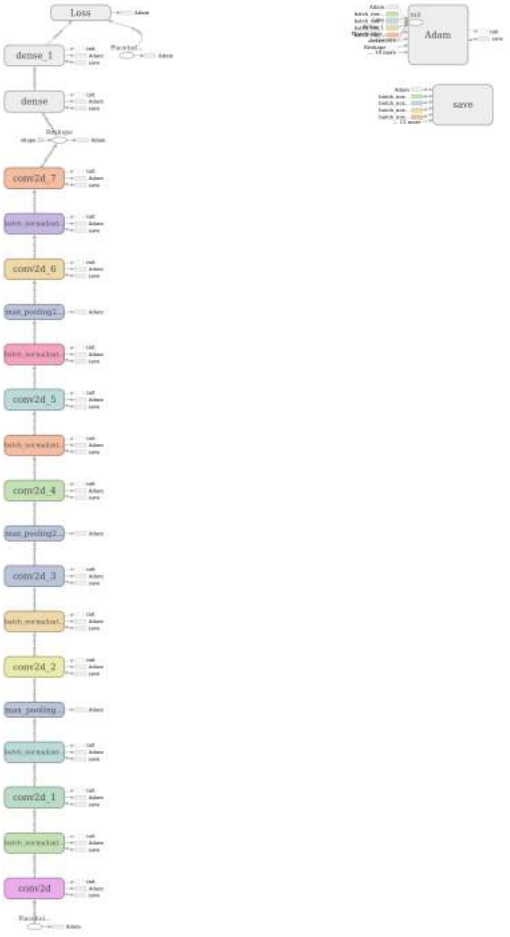


Figure 23: HomographyNet architecture

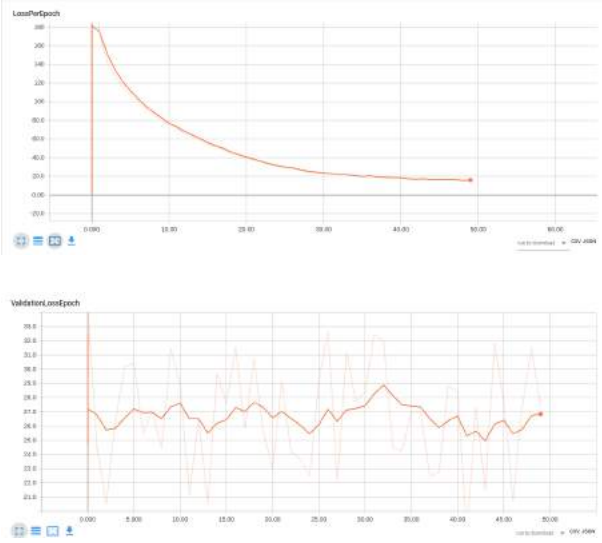


Figure 24: From left to right (a) Loss per epoch (b) Validation loss per epoch

1) TensorDLT: TensorDlt converts the $\tilde{\mathbf{H}}_{\mathbf{4pt}}$ to a 3x3 homography matrix. The main contribution of the paper is

Figure 25: Image overlayed with homography estimated by deep learning model shown in red and ground truth shown in blue on Train set

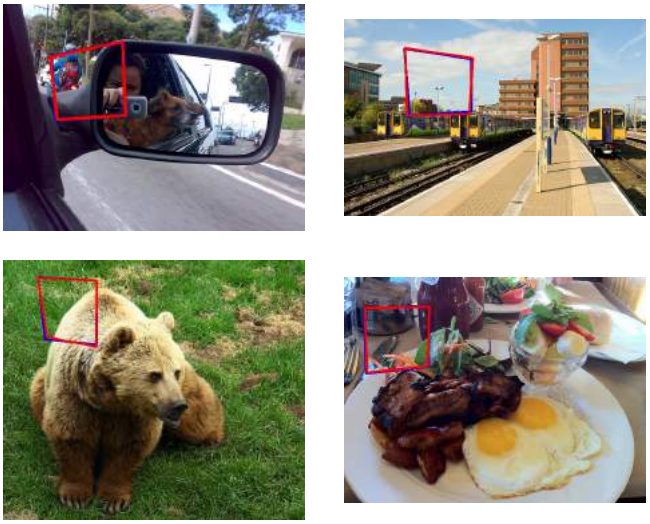


Figure 26: Image overlayed with homography estimated by deep learning model shown in red and ground truth shown in blue on Validation set

to make this function differntiable. A function is similar to opencv function getPerspectiveTransform() but is differentiable. To convert 4 point parameterization,$\tilde{\mathbf{H}}_{\mathbf{4pt}}$, to 3x3 matrix the problem is formulated $Ax = b$ and we solve the system of linear equations. Corner points of first patch i.e. patch A, are needed to form the system of equation which are fed directly into TensorDLT layer. After we get the homography matrix, we warp the original image (Ia) using 3x3 homography matrix. A matrix is defined as follows:

$$\hat{A}_i = \begin{bmatrix} 0 & 0 & 0 & -u_i & -v_i & -1 & v_i^{'}u_i & v_i^{'}v_i \\ u_i & v_i & 1 & 0 & 0 & 0 & -u_i^{'}u_i & -u_i^{'}v_i \end{bmatrix} \quad (1)$$

where u_i,v_i are corner points of patch A and $u_i^{'}$,$v_i^{'}$ are corner points of patch B.

Figure 27: Image overlayed with homography estimated by deep learning model shown in red and ground truth shown in blue on Test set

B martix is defined as

$$\hat{b}_i = \left[-v_i^{'}, -u_i^{'}\right]^T \tag{2}$$

2) Spatial Transformation Layer: This layer applies the 3x3 homography estimate output by the Tensor DLT to the pixel coordinates x_i of image I_A in order to get warped coordinates $H(x_i)$. These warped coordinates are necessary in computing the photometric loss function in Eqn. 3 that will train the neural network. In addition to warping the coordinates, this layer must also be differentiable so that the error gradients can flow through via backpropagation. 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). Spatial Tranformation layer is similar in working to opencv function warpPerspective(). This warped image is used to calculate the photometric loss defined in the following section.¡

3) Photometric Loss function: Drawing inspiration from traditional direct methods for homography estimation, an analogous loss function is defined. For an image pair $I^A(\mathbf{x_i})$ and $I^B(\mathbf{x_i})$ we want the network to estimate the 4 point parametrization of the homography $\tilde{\mathbf{H}}_{\mathbf{4pt}}$. This loss minimizes the average L1 pixel-wise photometric loss. This loss is given by the function :

$$L_{PW} = \frac{1}{|\mathbf{x_i}|}\sum_{\mathbf{x_i}}|I^A(\mathcal{H}(\mathbf{x_i})) - I^B| \tag{3}$$

Here $\mathcal{H}(\mathbf{x_i})$ is obtained from the $\tilde{\mathbf{H}}_{\mathbf{4pt}}$ which is estimated by the Supervised Network in the previous section. For better results L_1 error is chosen instead of L_2 error because it has been observed to be more suitable for image alignment problems [4].

E. Result-UnSupervised

The results can be seen in the figures(29) and (30)

F. Conclusion and Discussion: Phase 2

Although the supervised approach performs well in both validation and test set, some results in supervised section suffered from minor error due to changes in illumination of the patches. Also when the test was performed on patches that were perturbed more than the perturbation range on which the network was trained on, the error in ground truth and predicted homography was large(which is quite obvious).

For the unsupervised approach we were able to implement both the TensorDLT and Spatial Transformer Layer, but even running the code for more than 50 epochs didn't provide good results. In order to augment our understanding of Unsupervised process we referenced the code provided officially by the authors of the paper and generated results to observe the differences between ground truth and the patch warped with the homography estimated by the unsupervised model.

REFERENCES

[1] Cmsc733 computer vision. https://cmsc733.github.io/2019/proj/p1/ph2.

[2] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Deep image homography estimation. *arXiv preprint arXiv:1606.03798*, 2016.

[3] 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.

[4] Iuri Frosio Hang Zhao, Orazio Gallo and Jan Kautz. Loss functions for neural networks for image processing. *CoRR*, abs/1511.08861, 2015.

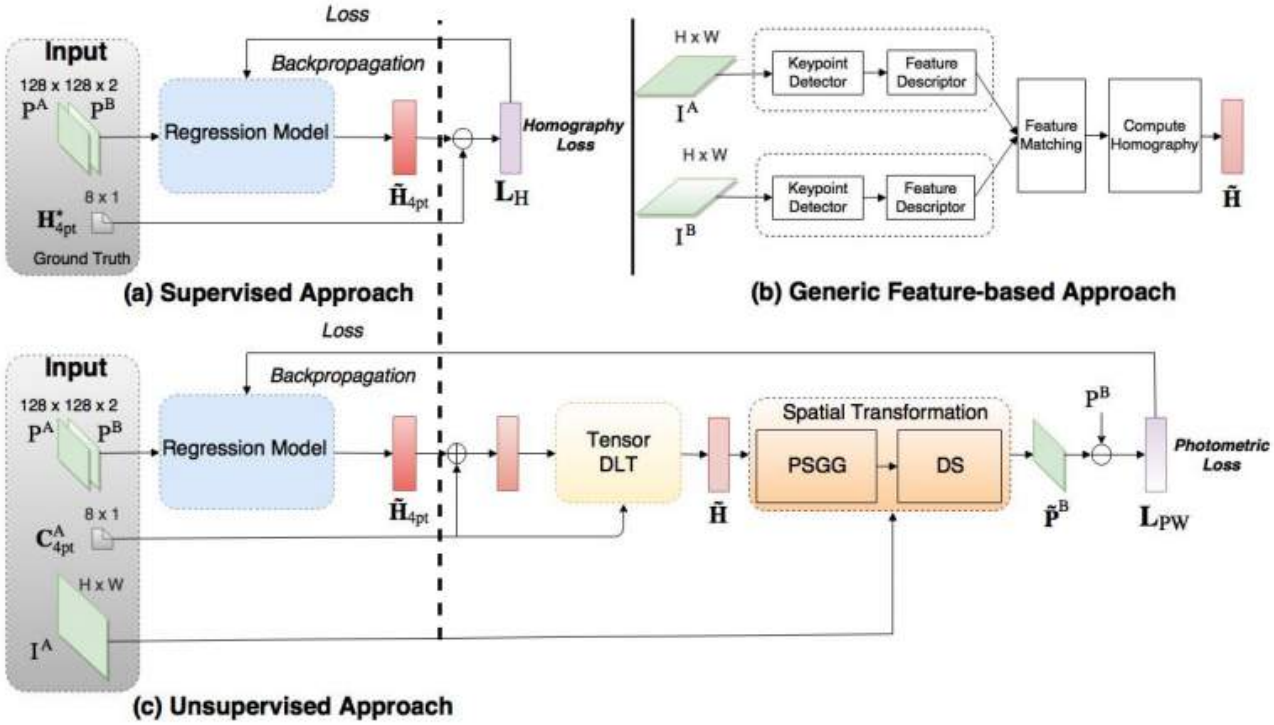


Figure 28: Overview of homography estimation methods; (a) supervised deep learning approach; (b) Feature-based methods; and (c) unsupervised method [3]

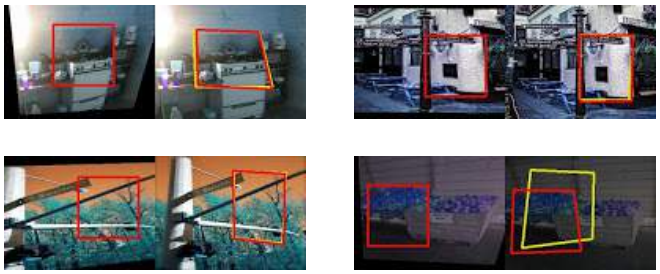


Figure 29: Image overlayed with homography estimated by deep learning model shown in red and ground truth shown in blue on Test set

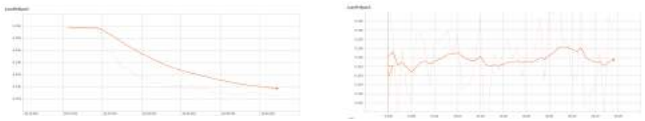


Figure 30: from left to right (a) Loss per iteration on Train set (b) Loss per iteration on Validation set

IV. Appendix

The intermediate results for all the processes- corner detection, ANMS, feature matching, RANSAC and stitching are shown for two arbitrarily selected images of each Train and Test Set for reference purposes.

A. Set 1

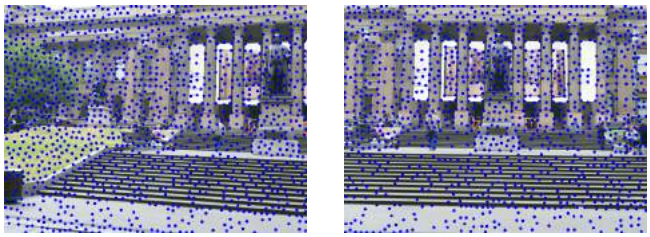


Figure 31: Output of Corner Detection

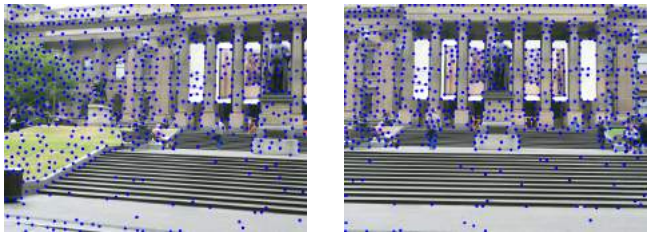


Figure 32: Output of ANMS

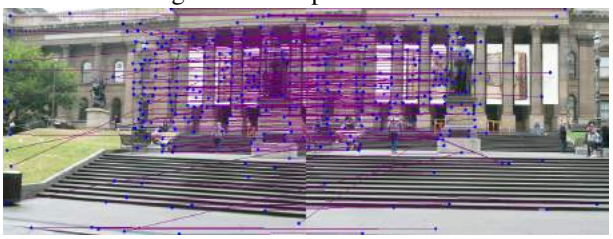


Figure 33: Output of Feature Matching

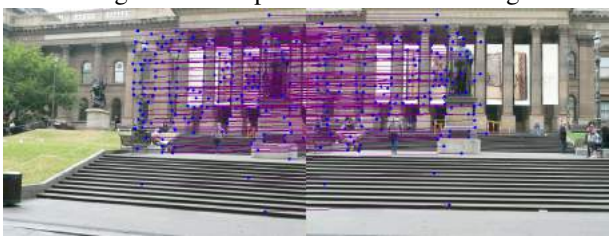


Figure 34: Output after RANSAC

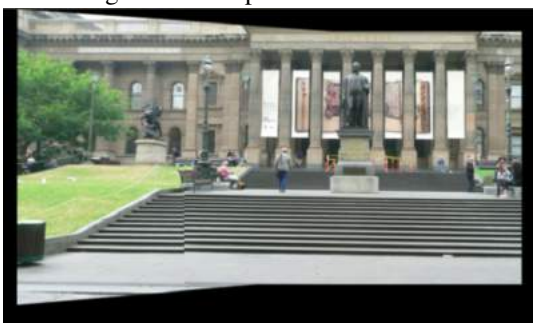


Figure 35: Warping and stitching done for the two images

B. Set 2

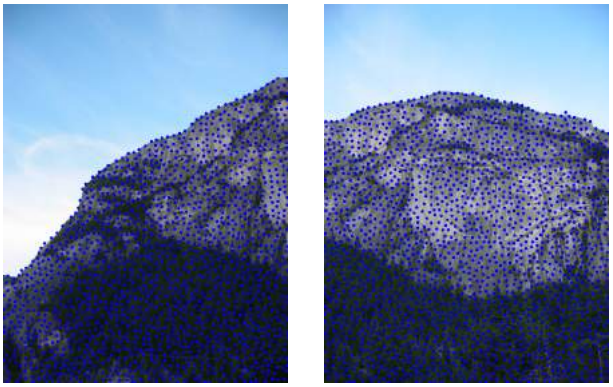


Figure 36: Output of Corner Detection

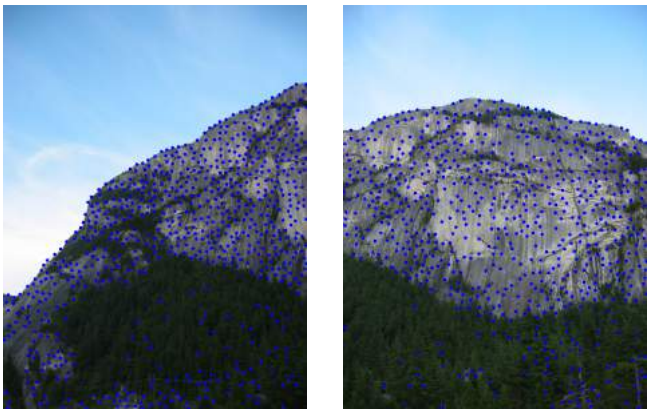


Figure 37: Output of ANMS

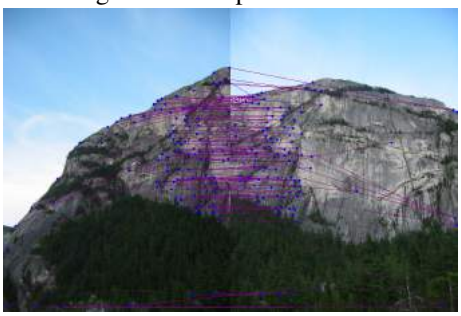


Figure 38: Output of Feature Matching

C. Set 3

D. Set 4

E. Set 5

F. Set 6

G. Set 7

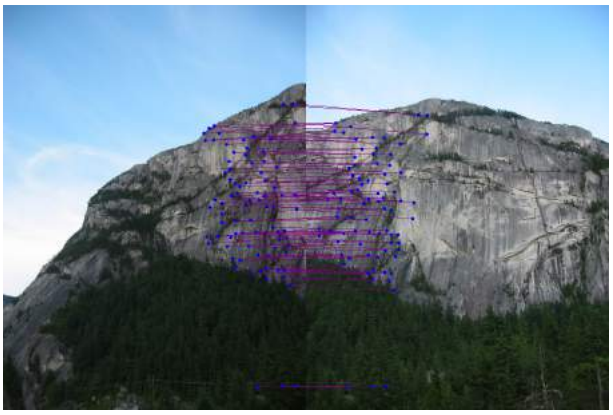


Figure 39: Output after RANSAC

Figure 40: Warping and stitching done for the two images

Figure 41: Output of Corner Detection

Figure 42: Output of ANMS

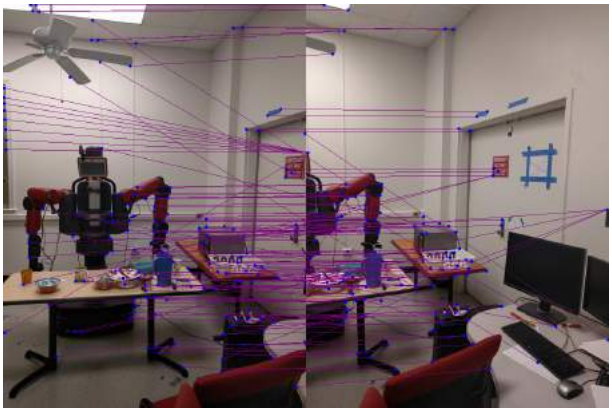


Figure 43: Output of Feature Matching

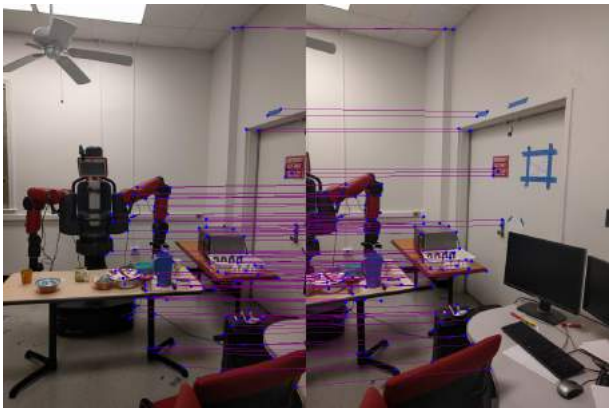


Figure 44: Output after RANSAC

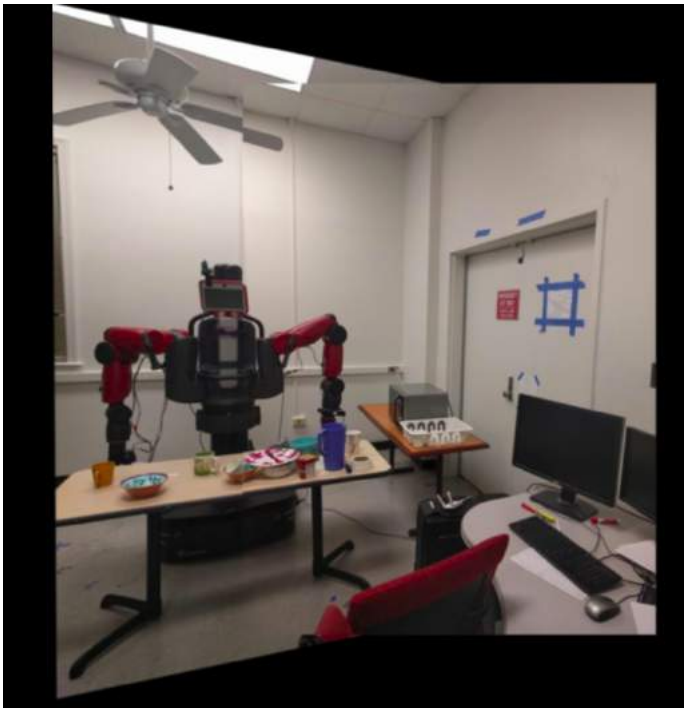


Figure 45: Warping and stitching done for the two images

Figure 47: Output of ANMS

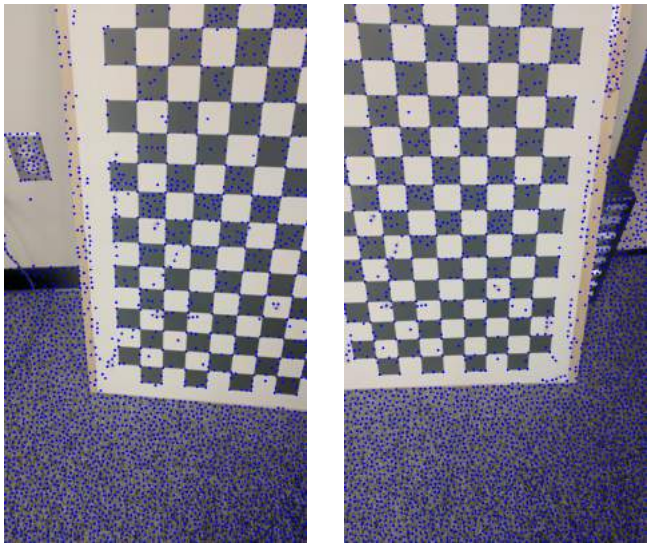


Figure 46: Output of Corner Detection

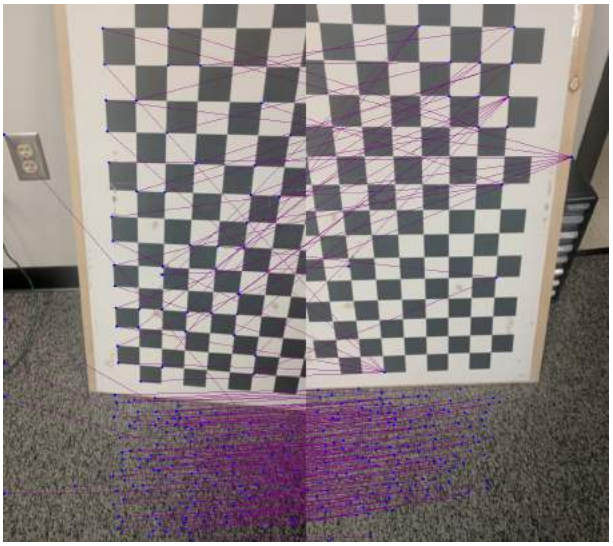


Figure 48: Output of Feature Matching

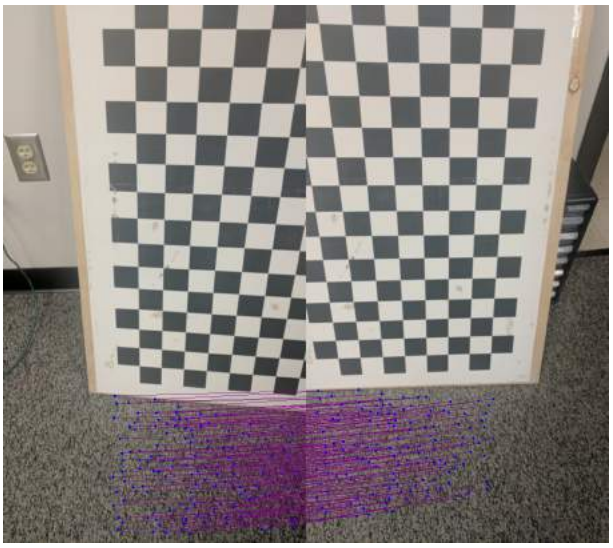


Figure 49: Output after RANSAC

Figure 51: Output of Corner Detection

Figure 50: Warping and stitching done for the two images

Figure 52: Output of ANMS

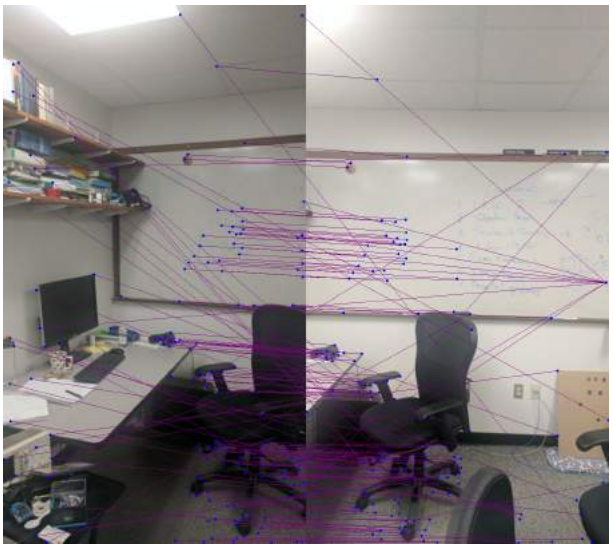


Figure 53: Output of Feature Matching

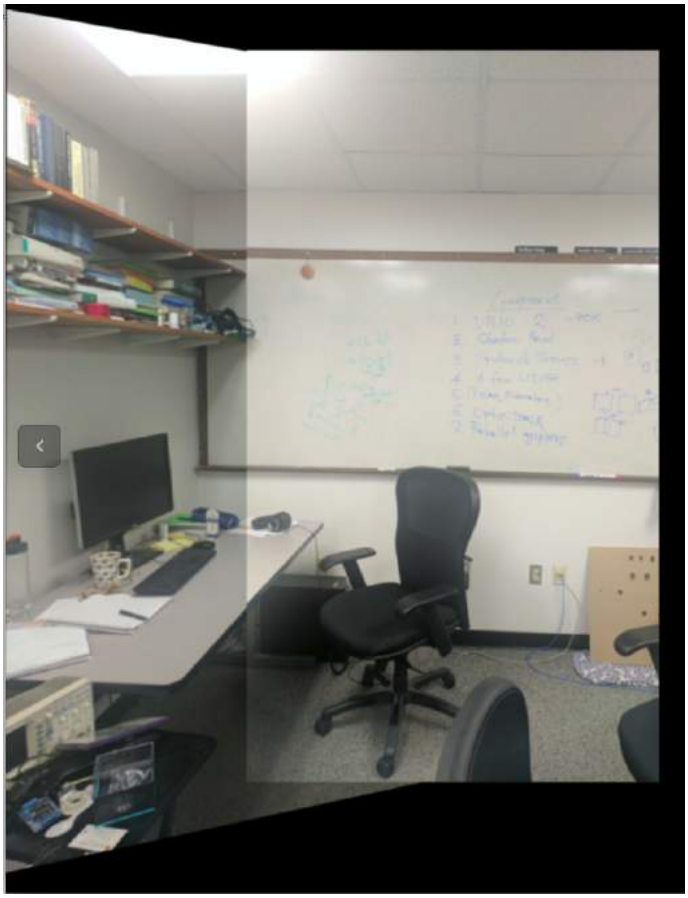


Figure 55: Warping and stitching done for the two images

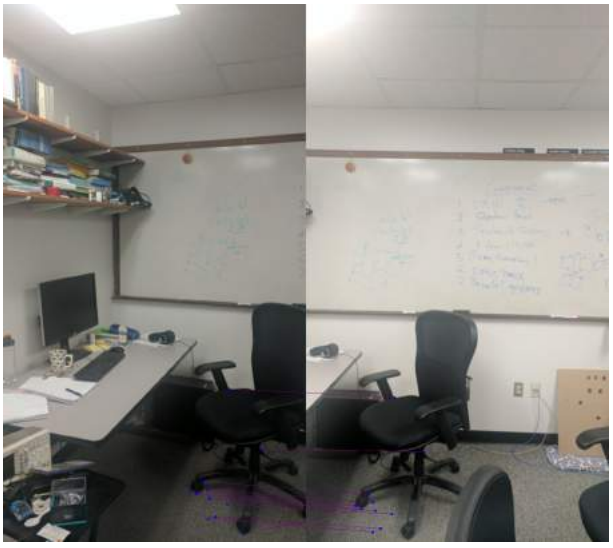


Figure 54: Output after RANSAC

Figure 56: Output of Corner Detection

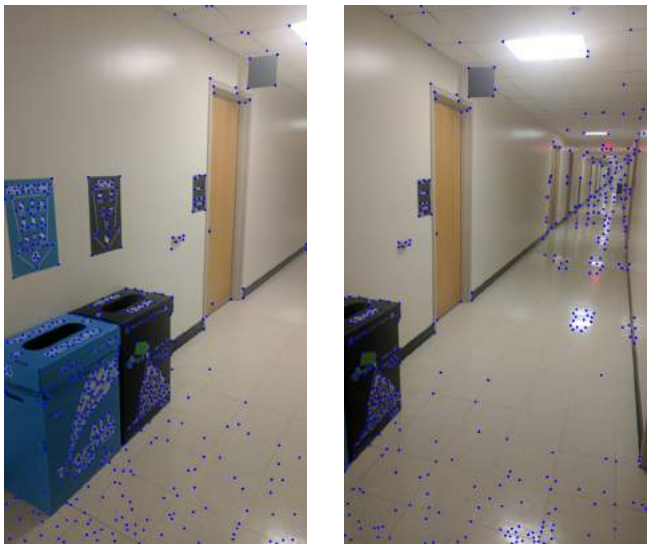


Figure 57: Output of ANMS

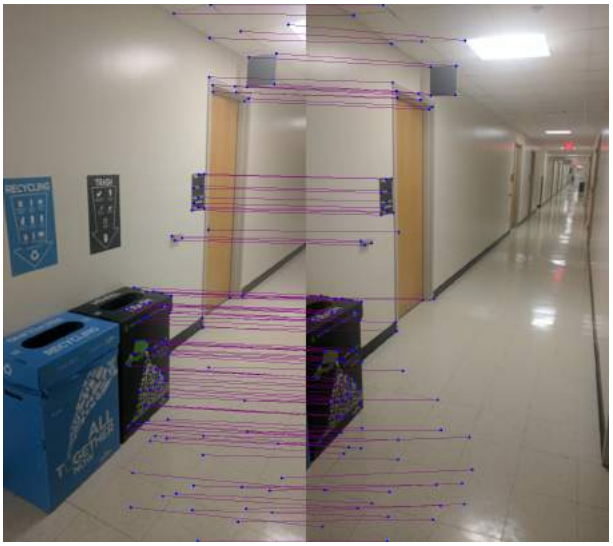


Figure 59: Output after RANSAC

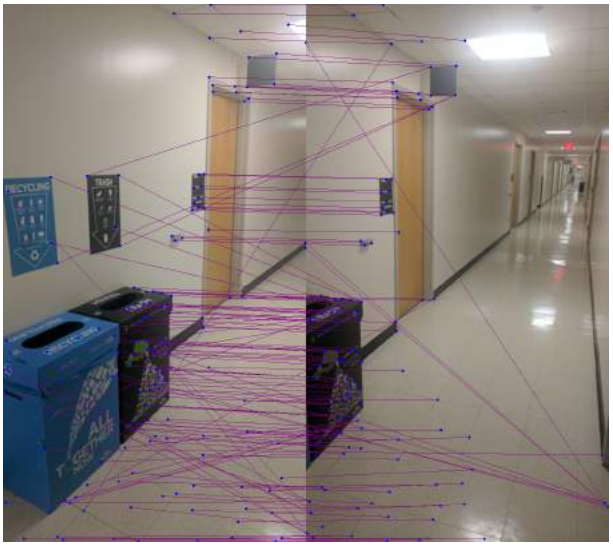


Figure 58: Output of Feature Matching

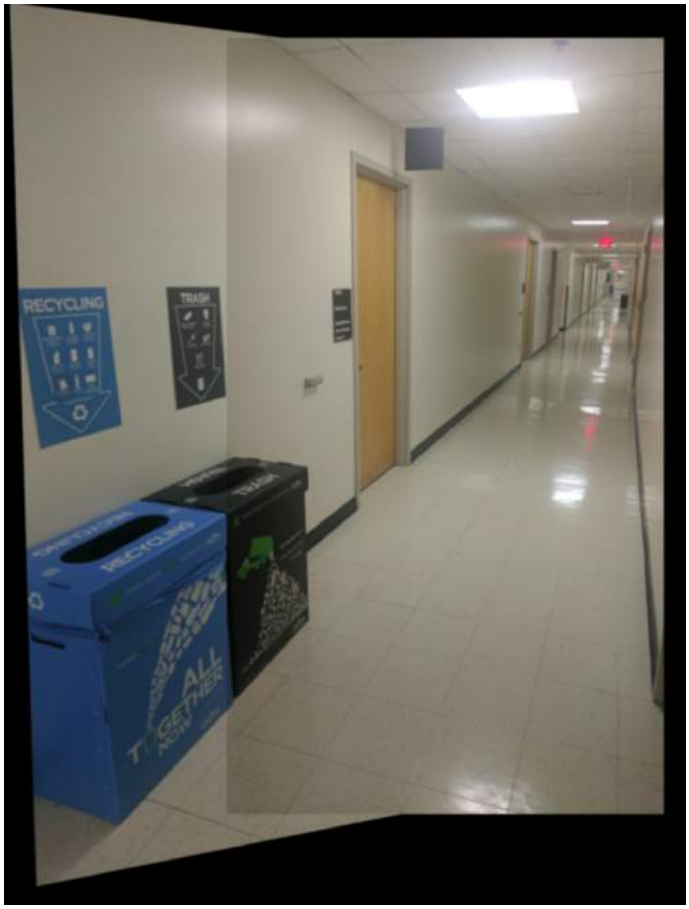


Figure 60: Warping and stitching done for the two images

Figure 61: Output of Corner Detection

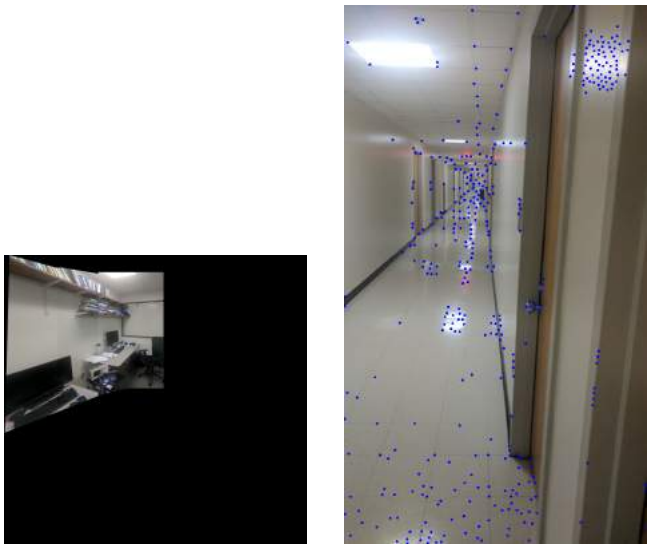


Figure 62: Output of ANMS

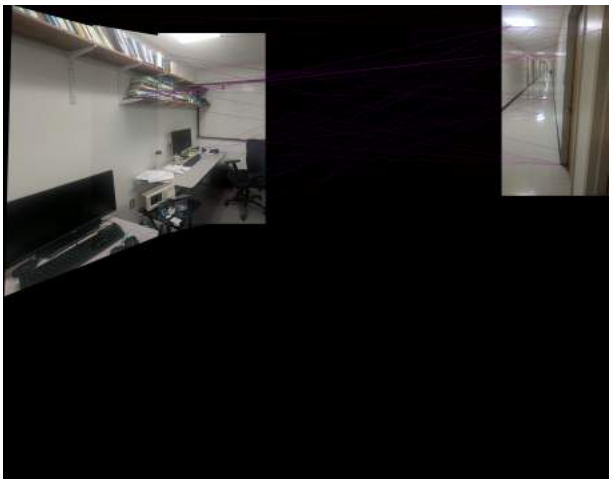


Figure 63: Output of Feature Matching

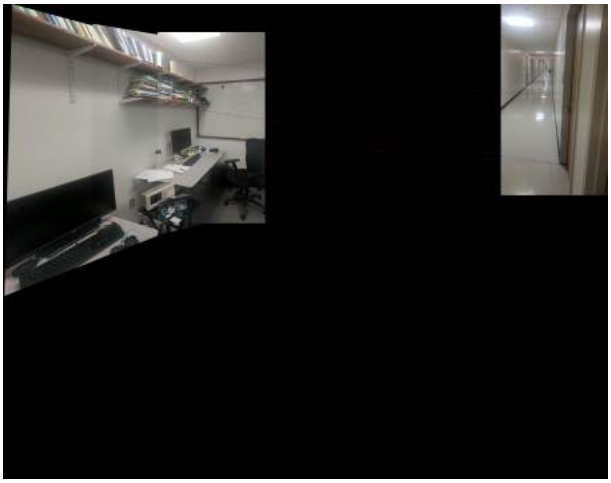


Figure 64: Output after RANSAC

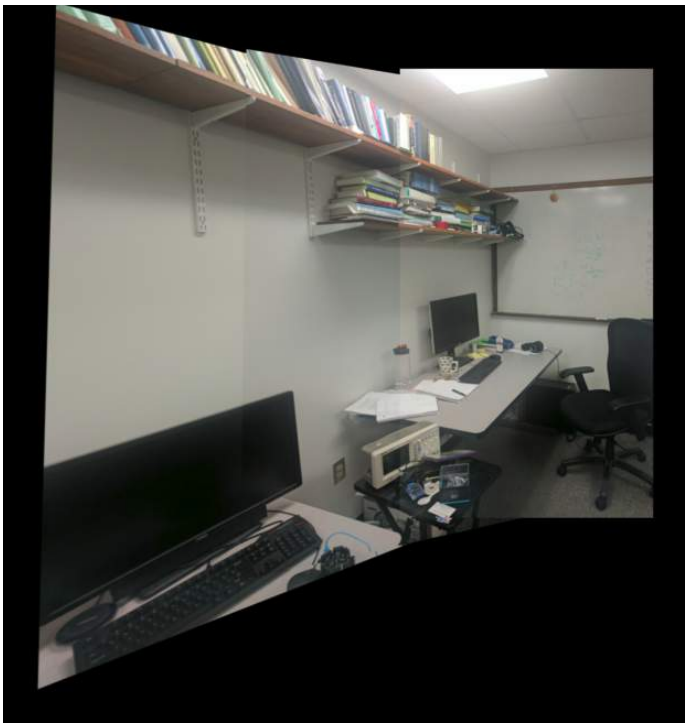


Figure 65: Warping and stitching done for the two images