# CMSC733: Project 1 - MyAutoPano

Abhilash Mane
University of Maryland
UID : 117402865
Email: amane@umd.edu

Advait Patole
University of Maryland
UID : 118130743
Email: apatole@umd.edu

***Abstract*—The field of computer vision has been a great topic of research for many researchers.The problem of stitching images to form a panorama is a well known topic in this field that can be solved using various image processing algorithms.The paper talks about the technique of stitching images to form a panorama using traditional as well as deep learning approach.The traditional approach deals with extracting and matching the common features in images and blending the images together based on their common features.**

## I. Phase 1: Traditional Approach

In this phase we present our approach to stitch the images together to form a panorama. We extracted features from images and then matched these features.The features of the images are basically the interest points of the image here we have used corners in the images.During matching the features we got a lot of outliers which are removed using RANSAC method.We then computed the homography matrix which shows the mapping between two images.Once the homography matrix is calculated we then warp and blend the images to stitch them together.To make the process of stitching images less computationally intensive we divided the set of images in two part and found the stitched images from those set.To get the final stitched image we merged the stitched image output from the two set.

### *A. Corner Detection*

The first step is to detect the corners in the image.We have implemented two methods to detect the corners : Harris Corner Detection and Shi Tomasi Corner Detection.The following images shows the corners detected in the images.

### *B. Adaptive non-maximal suppression*

After we have used Harris Corner detector we see that there are a lot of redundant corners that are detected in the images.We apply Adaptive Non-Maximal Suppression to suppress the redundant corners and to select the strongest corners in the image.We have selected 500 strong corners in each image and if in a certain image there are less than 500 corners detected then we select all the corners that are present in the image.The Harris Corner detector returns us the corner score map, we have calculated the local maximas in a certain region that gives us the strong corners.The figure shows the suppression of corners detected using ANMS.ANMS will try to find corners which are true local maxima.

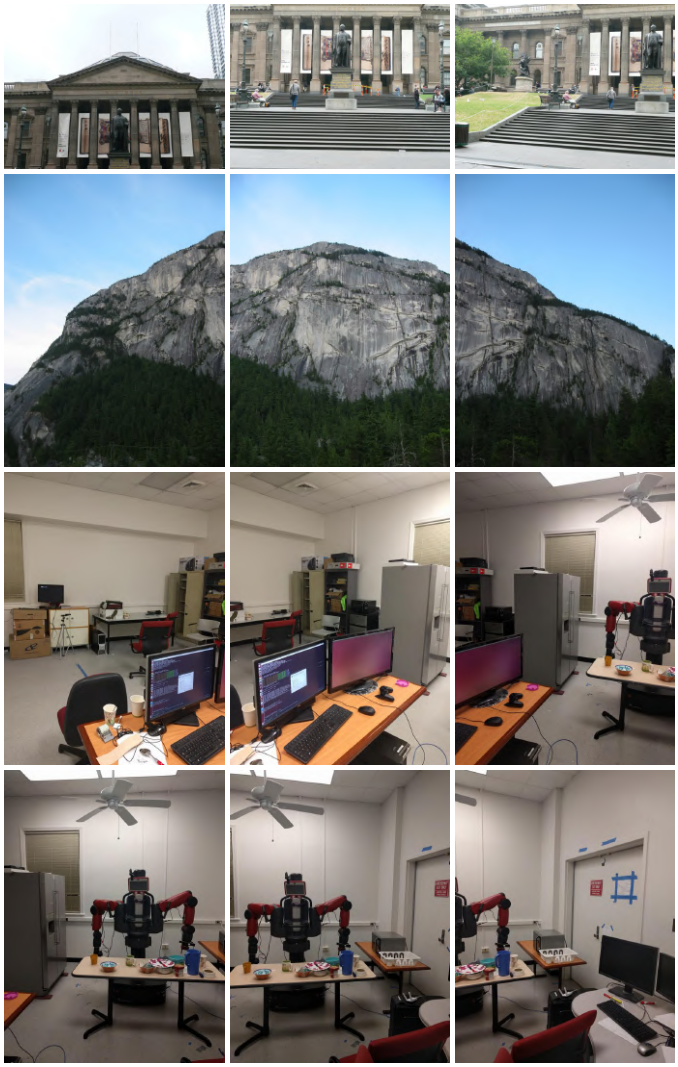


Fig. 1: Input Images

### *C. Feature Extraction*

To match the features and find out the common features in the images we need to first create feature vectors that encodes all the information of the feature.To create feature descriptor we took out a patch of size 40x40 around each key point.We then resize that patch to 8x8 and applied Gaussian blur to it.The resulting 8x8 patch was converted to 64x1 vector to remove bias and illumination which is the feature vectors that

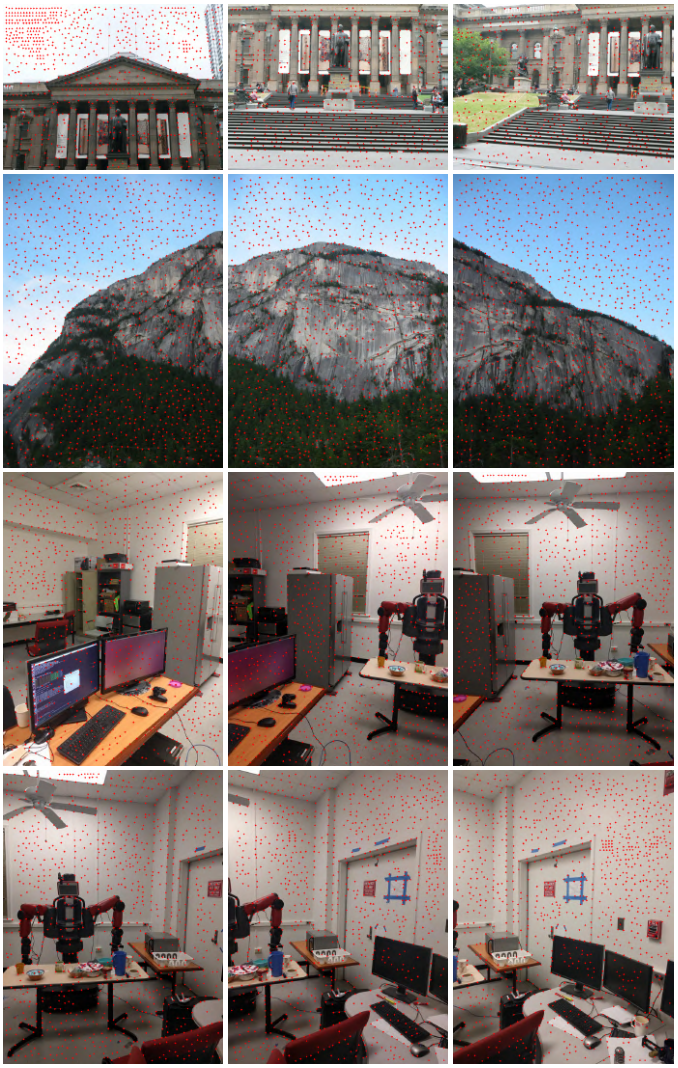


Fig. 2: Corners in Images

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

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

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

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

Fig. 3: ANMS

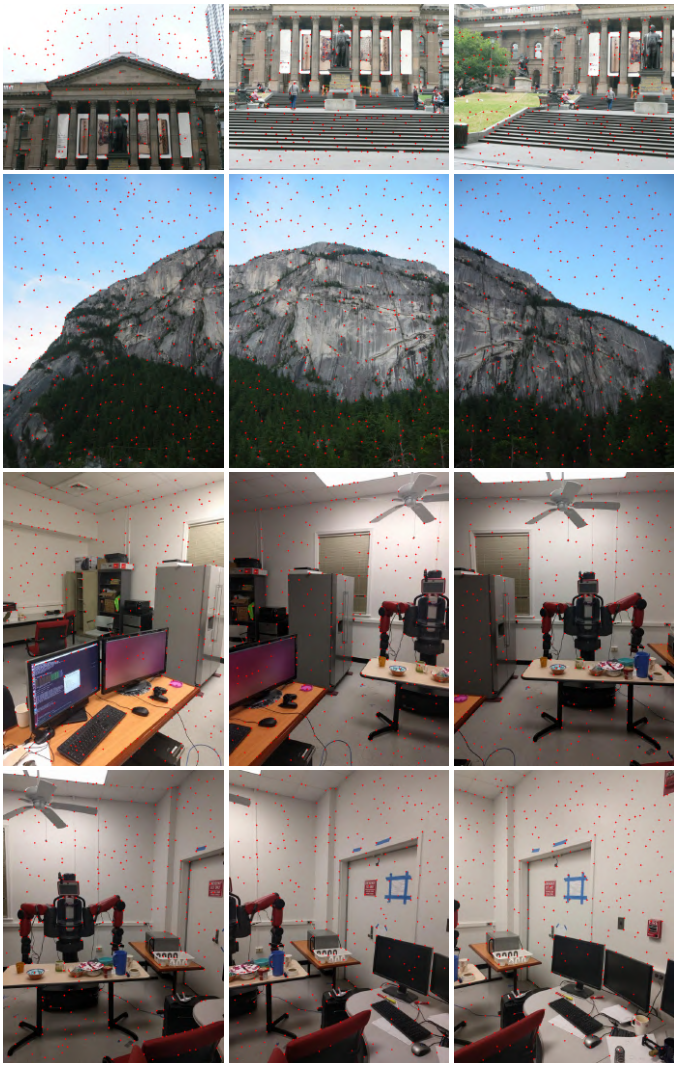


Fig. 4: Corner Detection using Harris Corner Detection

stores the information about the key point.The image shows the feature vectors of the key points.While taking out a patch some key points are lost mostly the corner ones.Once each key point 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.To check if the pair is matching we find out the ratio of smallest SSD and the second smallest SSD of that point with corresponding in the next image. If this ratio is less than 0.8 then the pair is rejected and rest of the pairs are saved. Note, we also set a flag to check if the the number of features is less than 4 so that RANSAC is not applied to reject the outliers.

*1) Feature Matching:* In the previous step, we associated a feature vector with each of our corner points skipping only those which were close to the edge. Now, we move forward with comparing and matching these vectors between two adjacent images. To do so, we loop over every point's feature vector in one of the image and compute the SSD (Sum of squared distances) with all the feature vectors in the other image. Getting the SSD is a computationally heavy task and

Fig. 5: Corner Detection using Shi Tomasi Corner Detector

hence, to improvise the runtime, we simply computed the the sum of the absolute differences. Now, we select the point for which the distance is lowest. Although, we just can't accept the lowest pair since this could be a false match. Hence, we take the ratio of the lowest distance and the second lowest distance and we accept the point pair as matched points if and only if the ratio is below a certain threshold. We have chosen this threshold as 0.85. After feature matching, we return the pairs of matched points. The outputs of matched features for various sets are provided in figures below.

### *D. RANSAC for outlier rejection and to estimate Robust Homography*

The feature matches that we got from the previous step contain few outliers that show that all the matches are not correct and there are incorrect pairs in them which are removed using RANSAC.The input of RANSAC are the coordinates of matched pairs obtained in the previous step.We have randomly selected 4 points from each matched pairs.From these set of

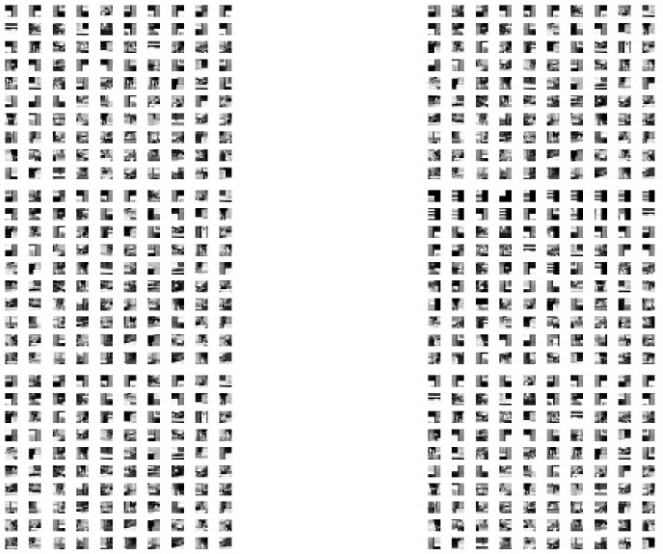


Fig. 6: Examples of random extracted features

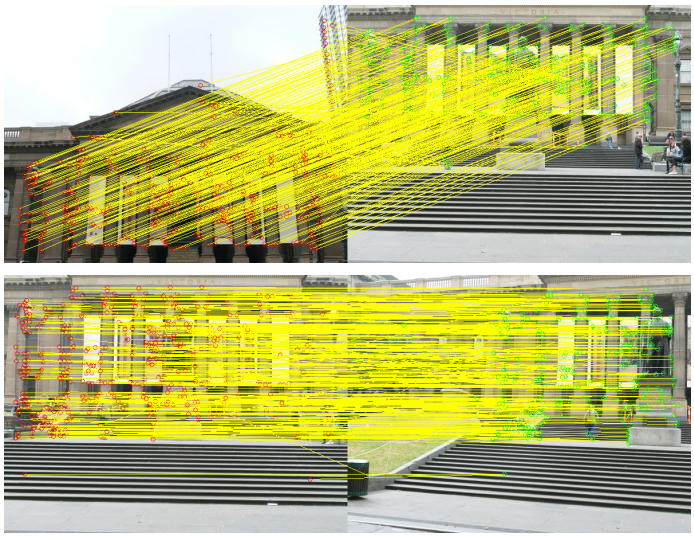


Fig. 7: Matched Features for Set 1

random points we computed the homography matrix H.We used this matrix and did a matrix multiplication between the points in the first image and computed their projection in the second image using H matrix.The error between points was calculated using SSD.We have set the threshold of 3000 if the SSD below it then we select that H matrix.This step is repeated for 4000 iterations to get the best homography matrix that contains the highest number 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.

### *E. Blending Images*

After obtaining homography between the images, we perform warping and stitching.After we computed the homography matrix from the previous step that contains the highest

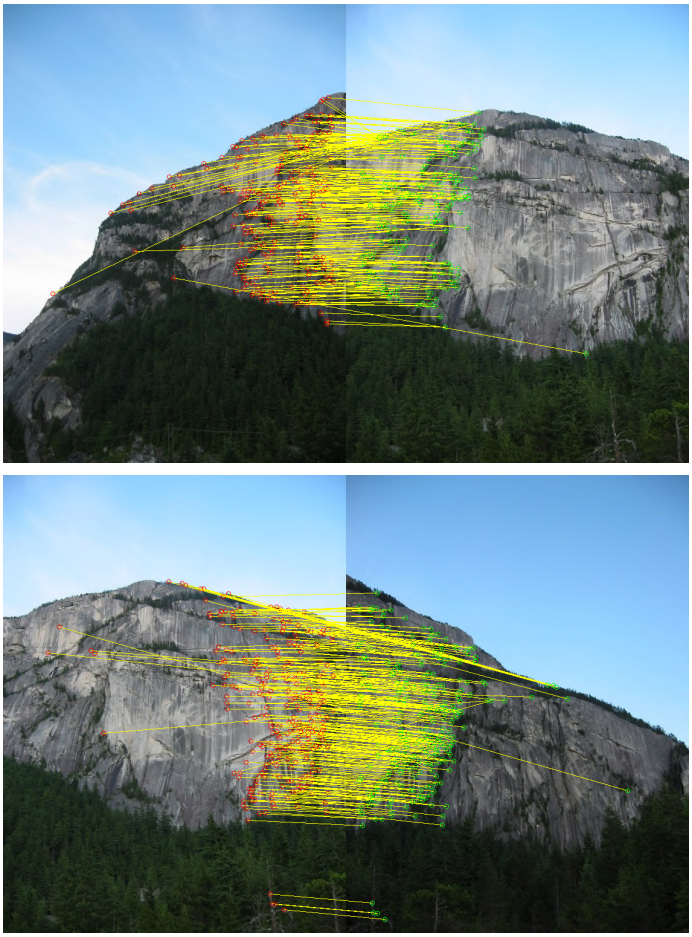


Fig. 8: Matched Features for Set 2

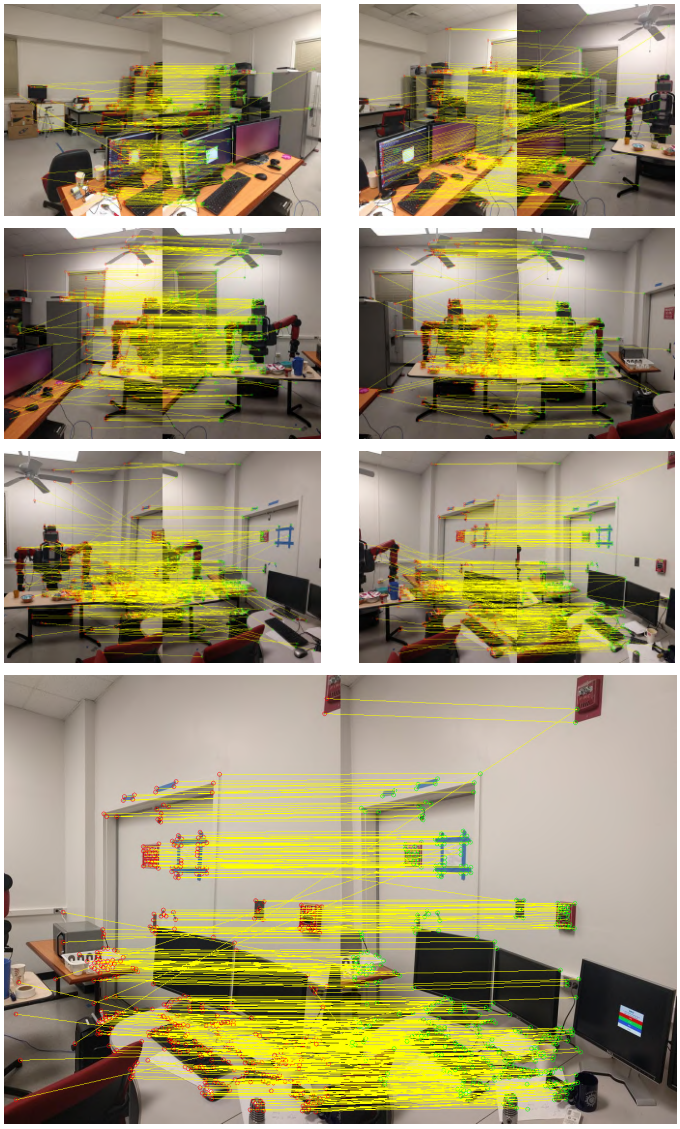


Fig. 9: Matched Features for Set 3

number of inliers.To stitch the images together we took 4 corner points of both the images.The four corner points of the first image were projected in the frame of reference of the second image using perspective transformation.But on doing so we found out that the homography matrix is negative due to which the projected coordinates of the corners become negative and those points will not be saved.To solve this issue we found the transformation of the left corner point of image and found the translation such that the left corner point of the image should have started from (0,0) in ideal condition.We used this translation and generated a translation matrix.We applied same translation on the second image as well so that both images are left in same frame before stitching them together.After applying these transformations we get warped images of both the images.Before stitching we erode the borders of the warp images so that when they are stitched there are no black lines visible at the borders of the images.We stitch these warp images.We then used this stitched image and combine with the next image, if there no matches found or very less matches found we ignore that image otherwise we follow the same procedure as above.This procedure is repeated till the last image in the image list.We have tried two approaches to get the panorama in the first approach we have merged the images in the order that they are given in and in second approach we have divided the image set in two halves and stitched the images in left half and right half and in the end we combined the resultant image from the left half and right half to get the final pano image.

### *F. Conclusion*

In this section we have used traditional approach to generate the panorama by matching the common features in the images.This approach works fine if there are adequate common features between images otherwise we have to reject that image from the panorama.In some cases homography matrix comes out to be negative hence we had to add translation as well so that we don't lose the pixels.When there are more than 4 images the stitching fails in cyclic order hence we tried dividing the images in two parts and applied blending and stitching on them.However we observed that if the number

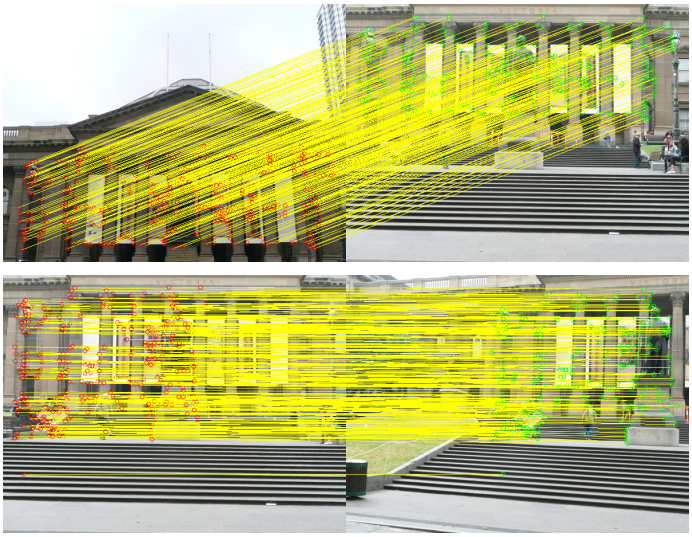


Fig. 10: Matched Features after RANSAC for Set 1

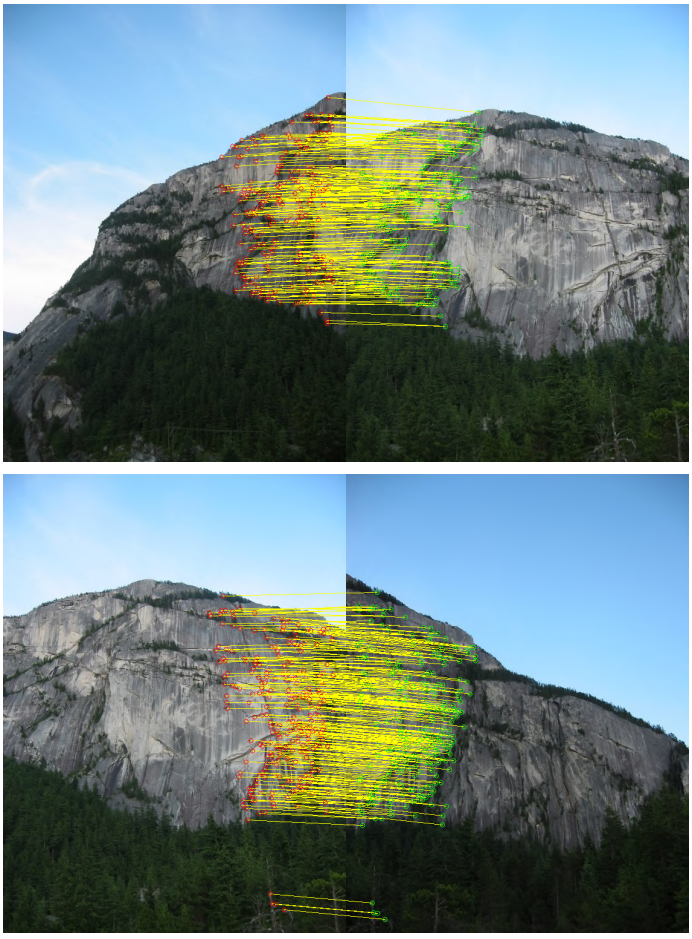
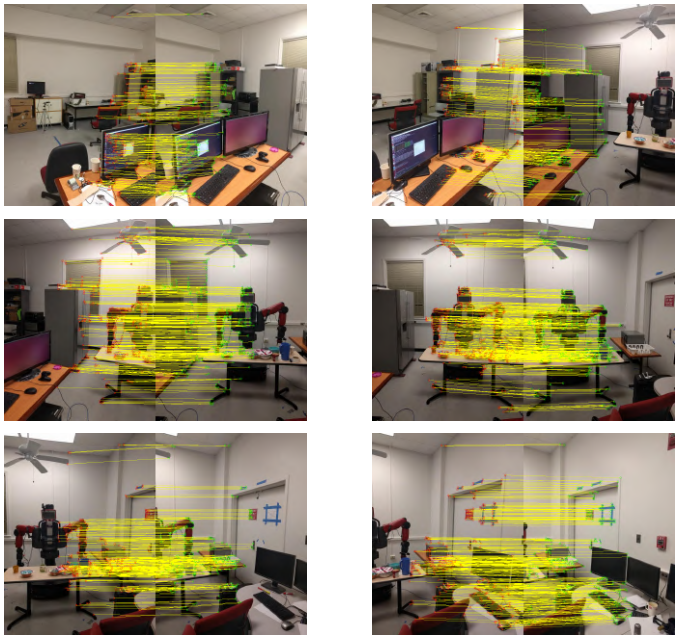


Fig. 11: Matched Features after RANSAC for Set 2



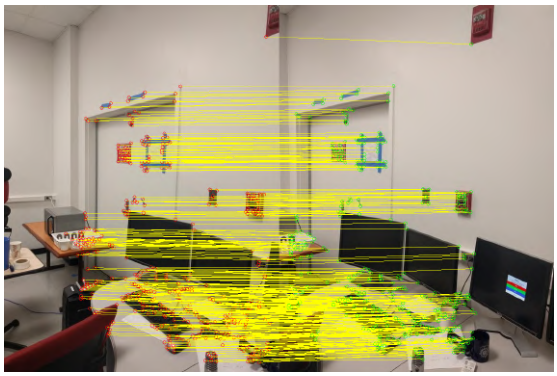


Fig. 12: Matched Features after RANSAC for Set 3

of images to be stitched is very large then it also causes a problem because the size of stitched image becomes very big and it requires a lot of computing power and takes a lot of time as well to get the panorama.We also found out that w had to set the parameters like iterations and threshold value in RANSAC to remove the outliers and in this procedure sometimes we got very few inliers, we need to figure out this issue so that it will work for all images without having to tune it for different images.We ave tried to reject the images if there are not enough matches between them but we need to figure out we need to break the chain of images that are found before that particular image through stitching if there is no sufficient matches or just skip the image.We observed that when we were stitching the images there were black lines that were formed at the boundaries of the images due to warping of the images we created a mask of image and used that mask to erode the boundaries, this helped to reduce the black line and blend the images together.We have also kept a flag that checks whether there are sufficient matches between images and if there are no sufficient matches we ignore that image.

Fig. 13: Panorama of Set 1

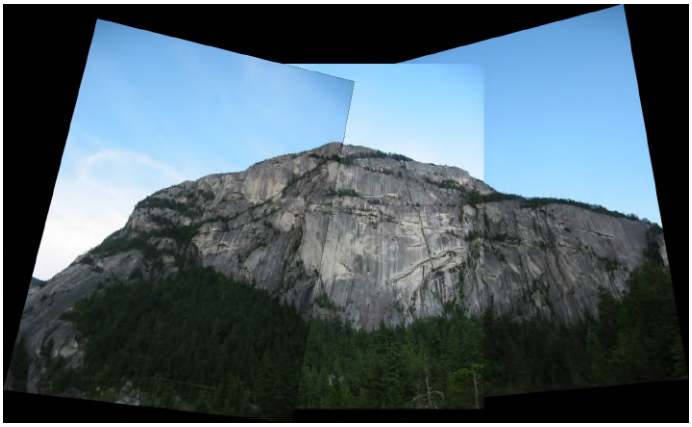


Fig. 14: Panorama of Set 2

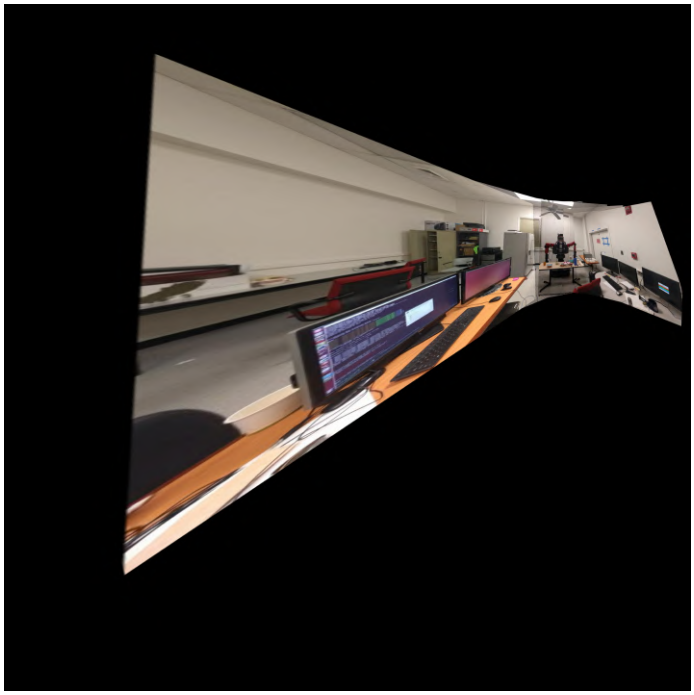


Fig. 15: Panorama of Set 3

Fig. 16: Panorama of Custom Set 1

Fig. 17: Panorama of Custom Set 2

Fig. 18: Panorama of Custom Set 3

Fig. 19: Corner Detection and ANMS outputs for Test Set 2

Fig. 20: Corner Detection and ANMS outputs for Test Set 1

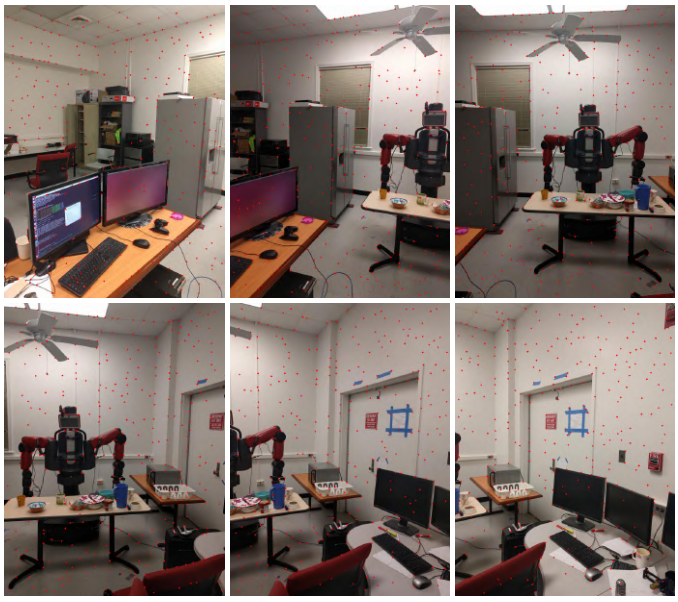


Fig. 21: Corner Detection and ANMS outputs for Set 3

Fig. 22: Corner Detection and ANMS outputs for Test Set 4

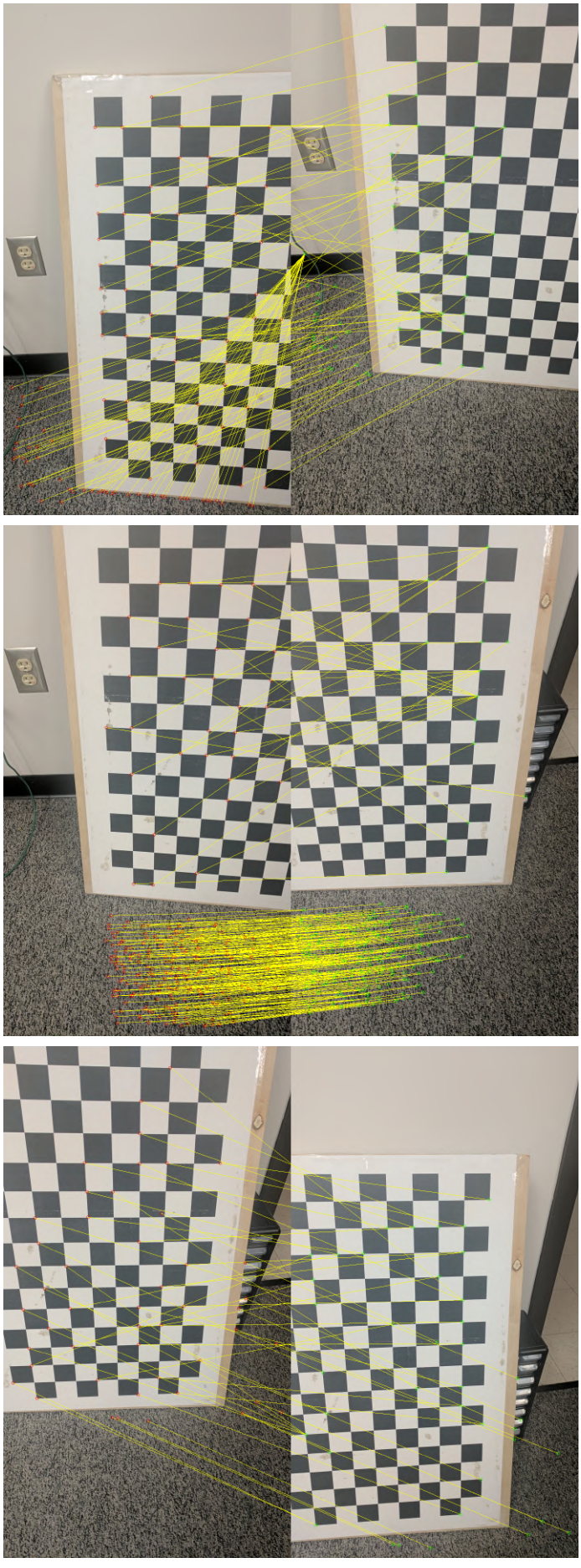


Fig. 23: Matched Features for Test Set 1

Fig. 24: Matched Features for Test Set 2




Fig. 25: Matched Features for Test Set 3

Fig. 26: Matched Features for Test Set 4

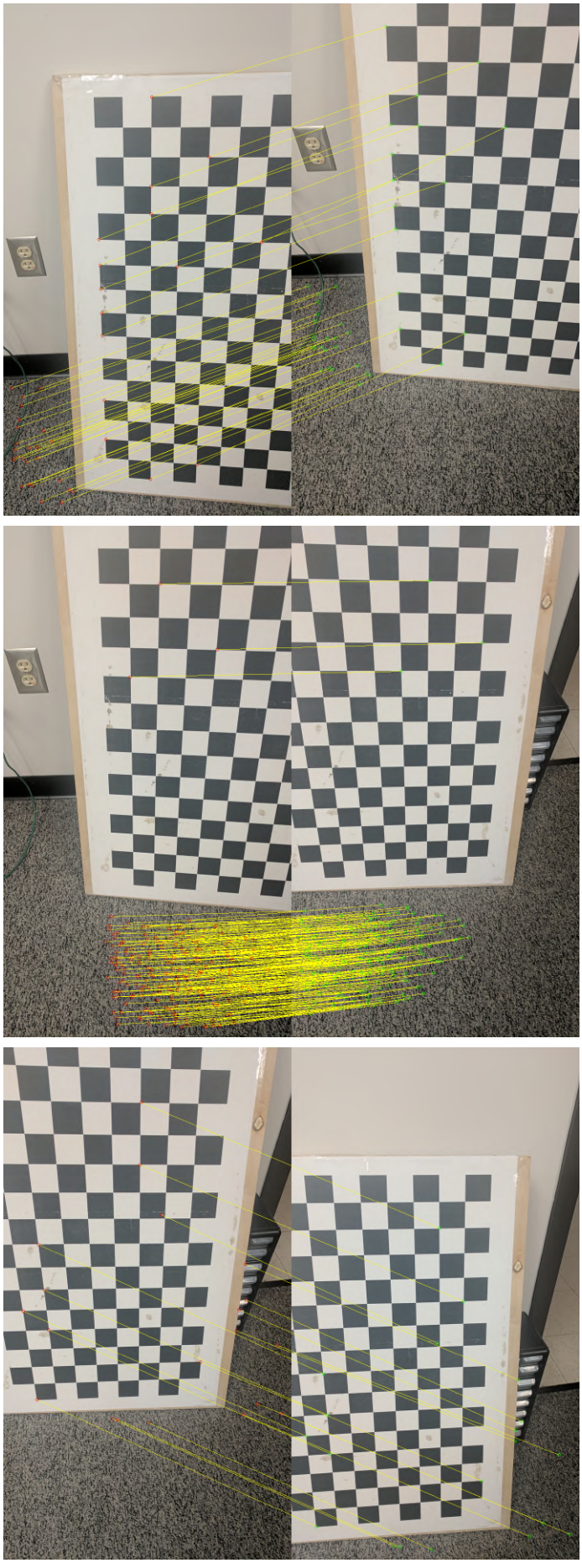


Fig. 27: Matched Features after RANSAC for Test Set 1

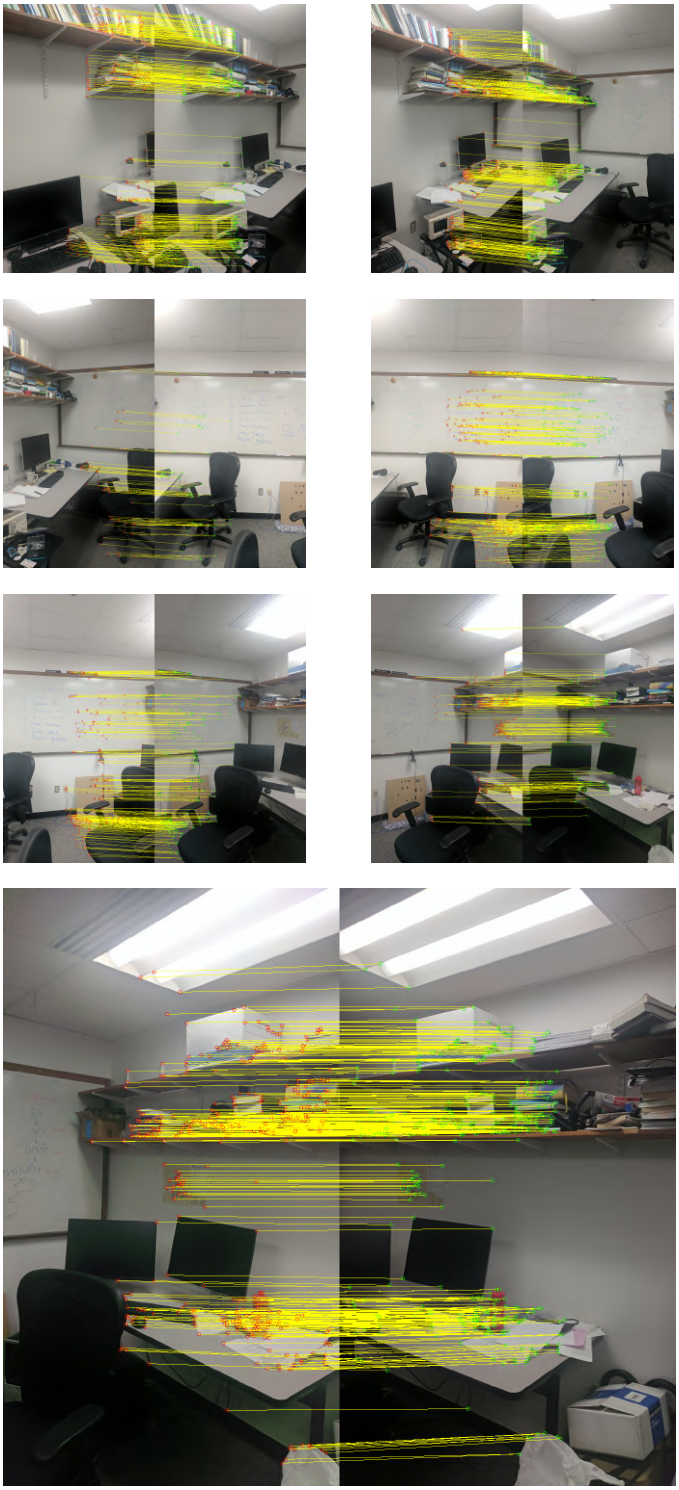


Fig. 28: Matched Features after RANSAC for Test Set 2


Fig. 29: Matched Features after RANSAC for Test Set 3

Fig. 30: Matched Features after RANSAC for Test Set 4

Fig. 31: Panorama of Set 1

Fig. 32: Panorama of Set 2


Fig. 33: Panorama of Set 3

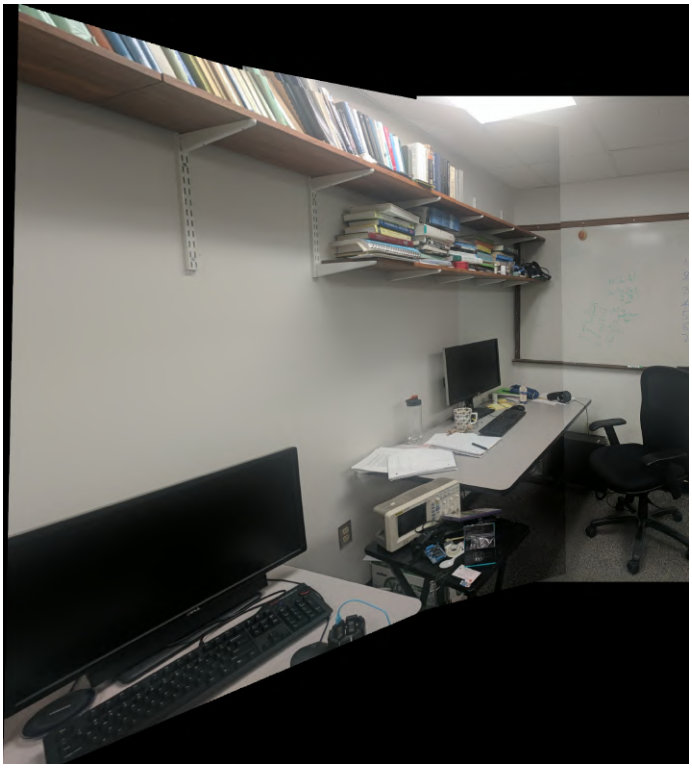


Fig. 34: Panorama of Custom Set 4

## II. PHASE 2 : DEEP LEARNING APPROACH

In this section we discuss how the state of the art deep learning techniques have influence on the homography and stitching of the images. The deep learning model tries the estimation of Homography. The main advantage of such an approach is that the algorithm can be made robust if the base network is genearlizable. Supervised as well as Unsupervised training is implemented to estimate the homography between image pairs using CNN.

### A. Data Generation for model

Instructions from [1] and webpage are used to generate the data for training of the model. the image is resized to 320 x 240, converted to grayscale, the patch size is chosen to be (128 x 128), and the amount of perturbation ($\rho$ = 32). We use a Statically generated dataset to train our models. 10 synthetic train/val instances are randomly generated for each image and are saved to local files Each of these instance consist of an image pair(patch A and patch B) and the $H_{4Pt}$ values. There are 5000 images in the train set and 1000 images in the validation set, thus, our generated dataset consists of 50000 examples for training and 10000 examples foo validation. for training purposes we need to read from these saved files and feed them to the network.

### B. Network Architecture

Similar network architecture is used for both, supervised ans unsupervised model. The architecture is shown in the figure. we use the same architecture as that presented in [1]. Here the architecture accepts pair of (patch A and patch B), and the output is a 4-point homography $H_{4Pt}$.

| Model Name | Train EPE | Val EPE | Test EPE |
|---|---|---|---|
| **Supervised** | 12.43 | 12.64 | 13.11 |
| **Unsupervised** | 12.19 | 12.91 | 13.47 |

### C. Supervised Model

Adam optimizer with learning rate 5e-3 and batch size of 32 to train the supervised network model. When the model is trained on the static dataset, it easily experiences overfitting as expected, hence a dropout layer with a probability of 0.5 is added afte the FC-1024 layer. Is is observed that scaling of image values into range [0,1](by dividing the image pixel value by 255), and $H_{4Pt}$ values into the range [-1,1](by dividing corner perturbation by 32) improves the network coverage. The plot of training vs. validation loss is shown in the figure. Here, the loss is mean squared error between the groundtruth and the predicted $H_{4Pt}$ values after they have been scaled into the range [-1,1]. Therefore, to scale them back into pixel value, we must apply the following formula: let $x$ be the MSE loss, the MSE loss on the pixel scale is $32\sqrt{x}$.

### D. Unsupervised Model

Adam optimizer with learning rate 1e-4 and batch size of 32 to train the supervised network model. Scaling of image values into range [0,1](by dividing the image pixel value by 255), and $H_{4Pt}$ values into the range [-1,1](by dividing corner perturbation by 32) is done. The plot of training vs. validation loss and $H_{4Pt}$ loss i shown in figure. Over time, the model learns to minimise the photometric loss. We only show the $H_{4Pt}$ to demonstrate the capability of the model while estimating the Homography during the training.

### E. Performance on the Train/Test set

The performance of our model is demonstrated in terms of mean corner pixel error of our models on the train, validation, and secret test set in table 1. The calculation of this mean corner error is same as in [1]. In the table *Static* means the model is trained on statically created dataset.The recorded run time of the Network Architecture is 5ms on the GeForce RTX 3070 GPU. From the 'Performance Table', we can observe that the models generalize very well on the test set. The mean corner error on the train and test set do not differ by much, but there is some room for improvement. This can be due to time and computational constraints.

### F. Comparision between Classical and Deep Learning Method for Homography Estimation

In this section, we show some examples to compare between traditional and deep learning method for estimating homography between 2 images. For deep-learning method, in order to estimate the homography between image A and image B, we do as follows:

- Resize A and B to (320, 240) and convert them to grayscale

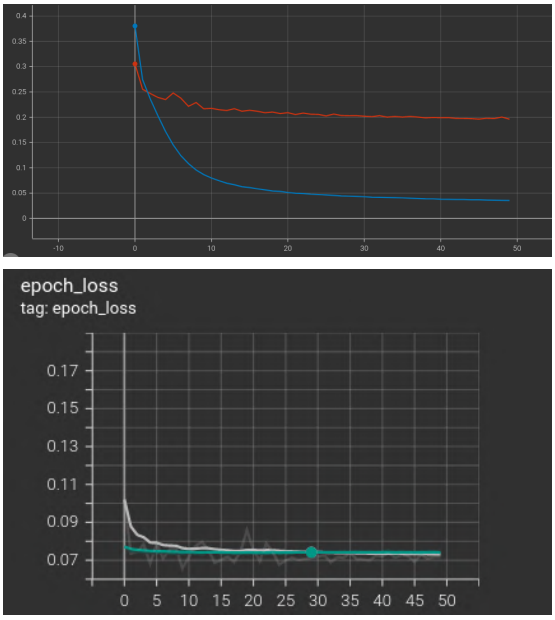



Fig. 35: test output for supervised and unsupervised model

Fig. 36: test output for supervised model

- Obtain 5 random crops of size (128, 128) in A, and obtain 5 random crops at those same locations of size (128, 128) in B (always make sure that in those 5 crops,there is a crop at the center of the image since this crop is likely to give more accurate estimates).
- For each pair of random crops at the same location, forward them through our supervised/unsupervised model to get the homography. The final homography is the average between all computed homography matrices.

We created a homography estimation model using the classical machine learning methods and deep learning methods. The necessity of high quality dataset generation is a major dependency for deep learning methods. Hence the deep learning models perform highly on synthetic datasets. We also observe that a Traditional approach works well in any real life scenarios while the deep -learning model is well suited for controlled, synthetic datasets. Hence the estimation and stitching were not as good as the classic learning methods.

### *G. Conclusion*

We create a homography estimation model using classic machine learning methods and the deep learning methods but deep learning methods are more dependent on high quality dataset generation process. While the performance of the deep learning model is high on synthetic dataset it seems very limited on real world images. Hence, the homography estimation and stitching were not as good as the classic machine learning methods.

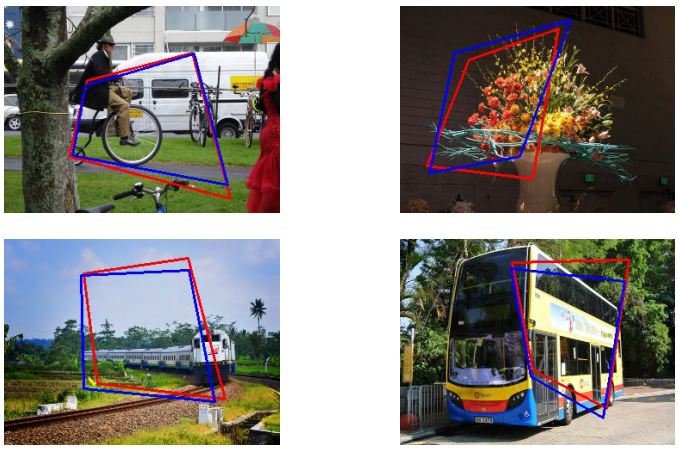


Fig. 37: test output for unsupervised model