CMSC733: Project 1 - MyAutoPano

Marco Huang and Tin Nguyen
Email: hhuang01@umd.edu and tintn@umd.edu
Use 1 day late

I. Phase 1 - Traditional approach

For every pair of consecutive images (indexed i and i+1), we check and make sure that there is an overlap of 50% or more between them (including those in our custom sets). Our procedure will use the middle-indexed image (e.g., index-2 image in a set of 3 images 1, 2, 3) and try to stitch the remaining images in the same set onto it.

A. Corner Detection

Fig. 1. Corner Detection Output for images in Train Set 1

We choose Harris corners over Shi-Tomasi corners for this first step. The detected corners are shown in Figure 1, Figure 2, Figure 3 (for the three train sets), Figure 4, Figure 5 (for the two custom sets), and Figure 6,Figure 7, Figure 8, Figure 9 (for the four test sets).

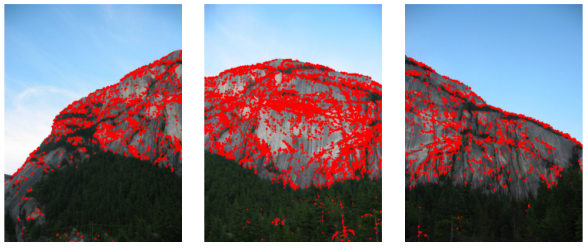

Fig. 2. Corner Detection Output for images in Train Set 2

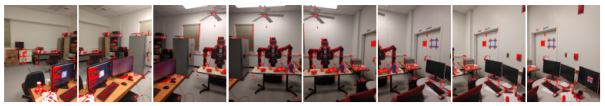

Fig. 3. Corner Detection Output for images in Train Set 3

Fig. 4. Corner Detection Output for images in Custom Set 1

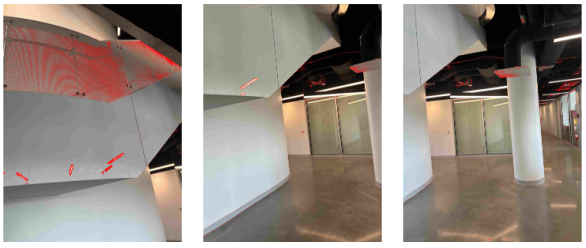

Fig. 5. Corner Detection Output for images in Custom Set 2

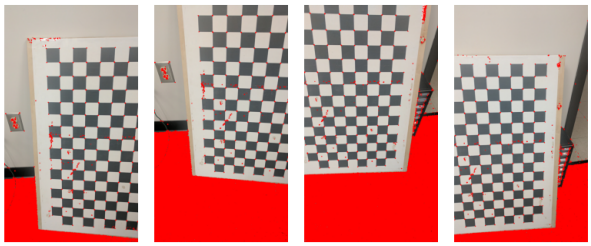

Fig. 6. Corner Detection Output for images in Test Set 1

Fig. 7. Corner Detection Output for images in Test Set 2

Fig. 8. Corner Detection Output for images in Test Set 3

Fig. 9. Corner Detection Output for images in Test Set 4

B. Adaptive Non-Maximal Suppression (ANMS)

The equally distributed corners as detected by the Adaptive Non-Maximal Suppression (ANMS) algorithm are shown in Figure 10, Figure 11, Figure 12 (for the three train sets), Figure 13, Figure 14 (for the two custom sets), and Figure 15,Figure 16, Figure 17, Figure 18 (for the four test sets).

Fig. 10. ANMS Output for images in Train Set 1

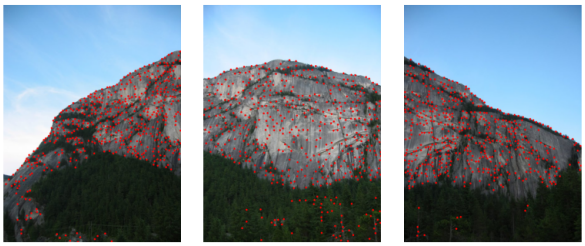

Fig. 11. ANMS Output for images in Train Set 2

Fig. 12. ANMS Output for images in Train Set 3

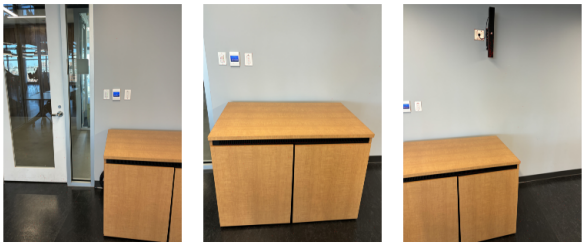

Fig. 13. ANMS Output for images in Custom Set 1

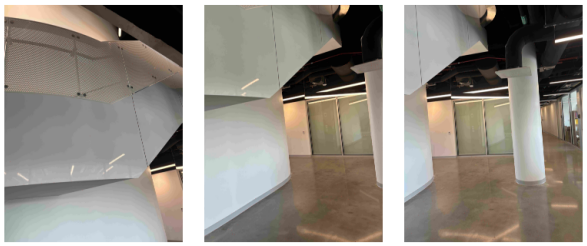

Fig. 14. ANMS Output for images in Custom Set 2

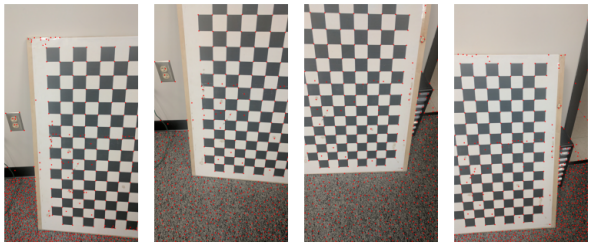

Fig. 15. ANMS Output for images in Test Set 1

Fig. 16. ANMS Output for images in Test Set 2

Fig. 17. ANMS Output for images in Test Set 3

Fig. 18. ANMS Output for images in Test Set 4

C. Feature Descriptor

To improve visibility, we select 100 corners randomly out of the corners returned by ANMS in each set. We show their feature vectors in Figure 19, Figure 20, Figure 21 (for the three train sets), Figure 22, Figure 23 (for the two custom sets), and Figure 24,Figure 25, Figure 26, Figure 27 (for the four test sets).

Fig. 19. Feature Descriptor Output for images in Train Set 1

Fig. 20. Feature Descriptor Output for images in Train Set 2

Fig. 22. Feature Descriptor Output for images in Custom Set 1

Fig. 21. Feature Descriptor Output for images in Train Set 3

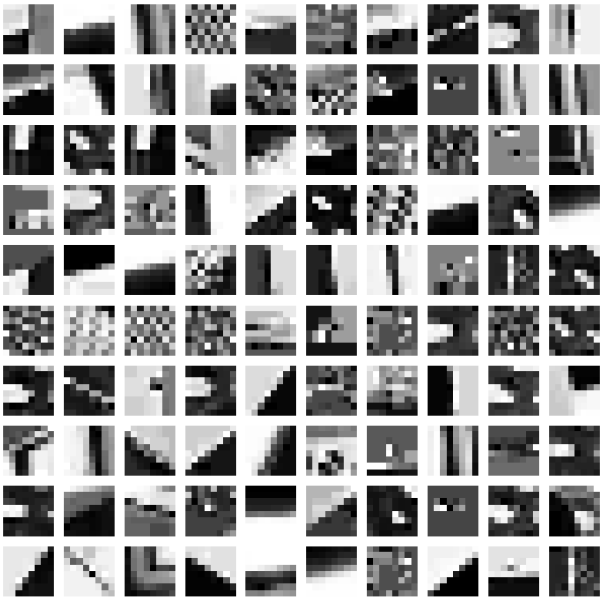

Fig. 23. Feature Descriptor Output for images in Custom Set 2

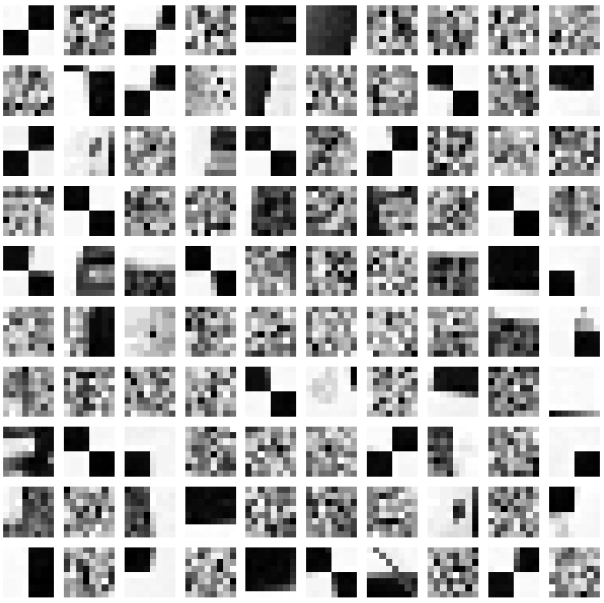

Fig. 24. Feature Descriptor Output for images in Test Set 1

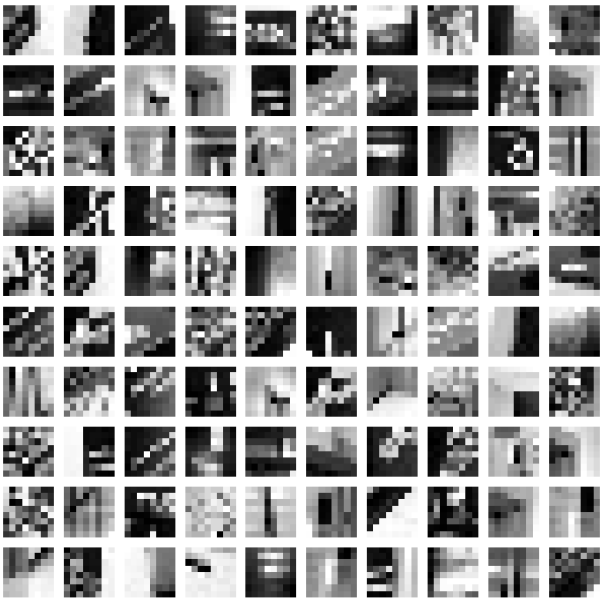

Fig. 26. Feature Descriptor Output for images in Test Set 3

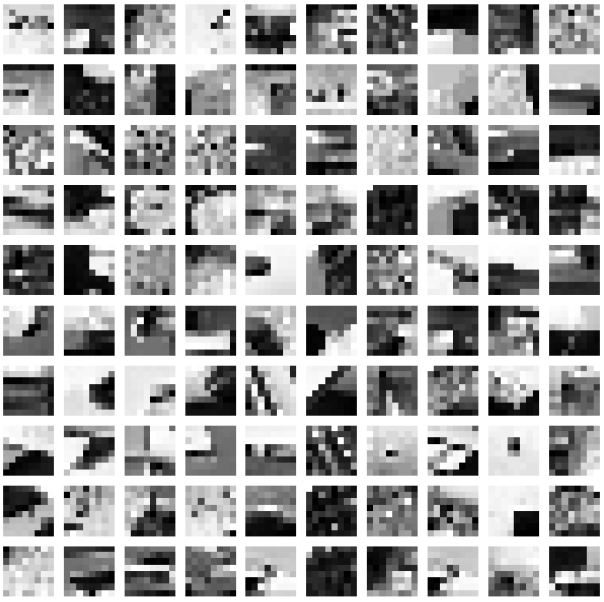

Fig. 25. Feature Descriptor Output for images in Test Set 2

Fig. 27. Feature Descriptor Output for images in Test Set 4

D. *Feature Matching*

For each pair of consecutively-indexed images in the same set, we perform the feature matching procedure with 65% match as the threshold to accept or reject each corner based on its match ratios with its best or second-best matches in the next consecutive image. The matched features for each pair of images are shown in Figure 28, Figure 29, Figure 30 (for the three train sets), Figure 31, Figure 32 (for the two custom sets), and Figure 33,Figure 34, Figure 35, Figure 36 (for the four test sets).

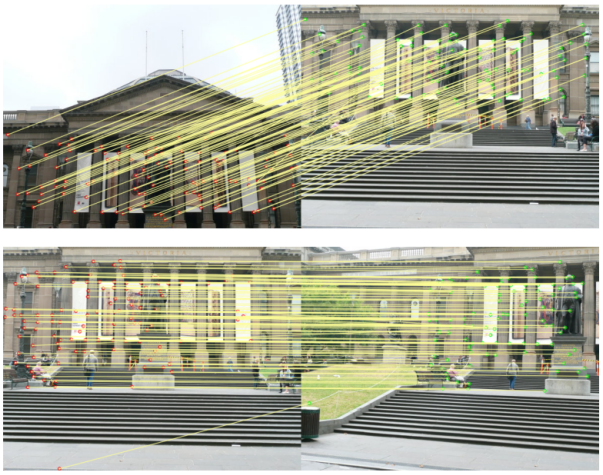

Fig. 28. Feature Matching Output for images in Train Set 1

Fig. 29. Feature Matching Output for images in Train Set 2

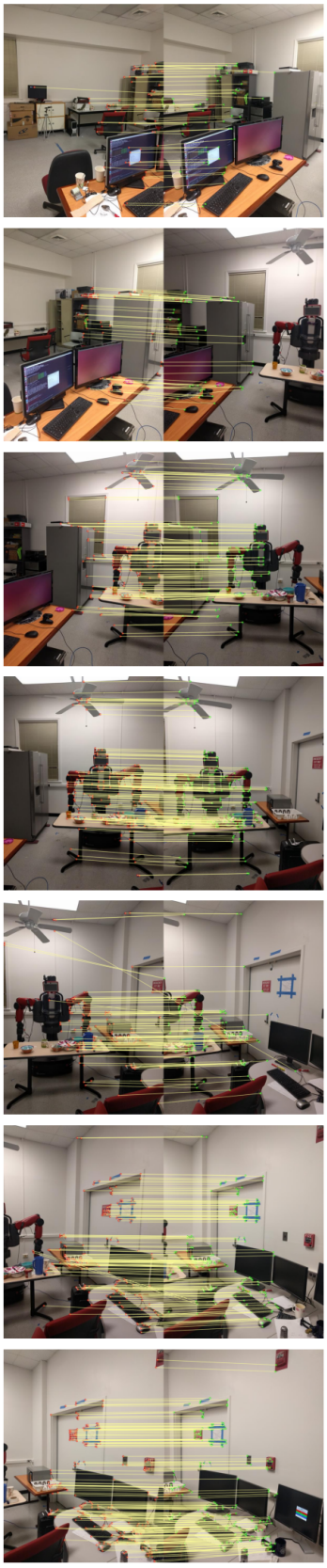

Fig. 31. Feature Matching Output for images in Custom Set 1

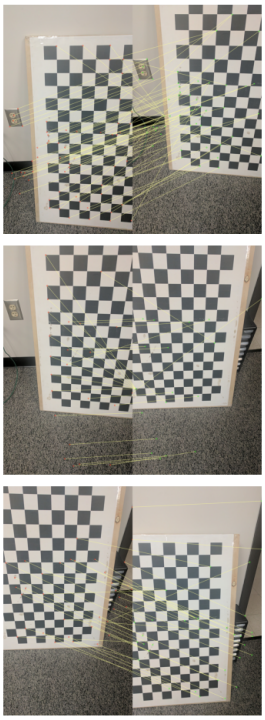

Fig. 33. Feature Matching Output for images in Test Set 1

Fig. 32. Feature Matching Output for images in Custom Set 2

Fig. 35. Feature Matching Output for images in Test Set 3

Fig. 36. Feature Matching Output for images in Test Set 4

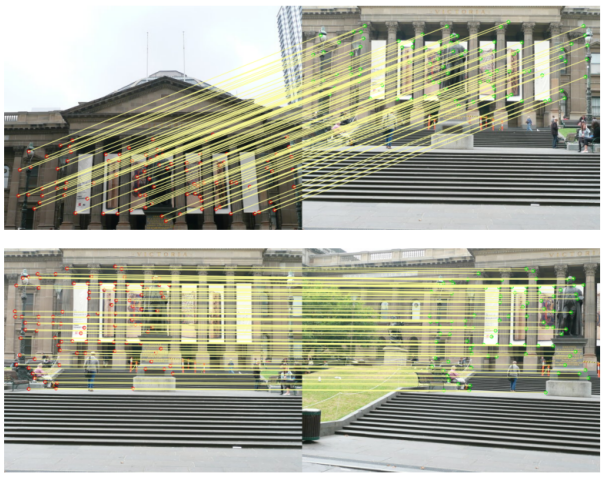

Fig. 37. RANSAC and Robust Homography Output for images in Train Set 1

E. RANSAC and Robust Homography

After the Random Sample Consensus (RANSAC) method removes outliers (wrong matches) with hyperparameters $\tau = 5$ and $N_{max} = 1000$, we obtain robust homography matrices. The remaining feature matches (after RANSAC removes the outliers) are shown in Figure 37, Figure 38, Figure 39 (for the three train sets), Figure 40, Figure 41 (for the two custom sets), and Figure 42,Figure 43, Figure 44, Figure 45 (for the four test sets).

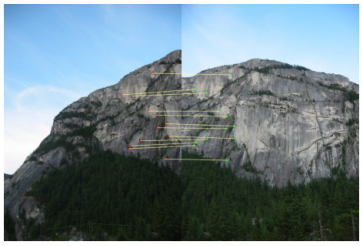

Fig. 38. RANSAC and Robust Homography Output for images in Train Set 2

Fig. 40. RANSAC and Robust Homography Output for images in Custom Set 1

Fig. 41. RANSAC and Robust Homography Output for images in Custom Set 2

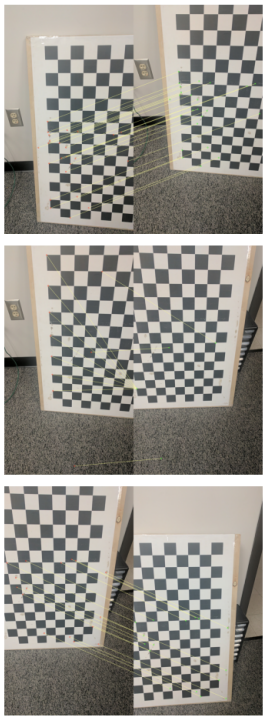

Fig. 42. RANSAC and Robust Homography Output for images in Test Set 1

Fig. 43. RANSAC and Robust Homography Output for images in Test Set 2

Fig. 44. RANSAC and Robust Homography Output for images in Test Set 3

Fig. 45. RANSAC and Robust Homography Output for images in Test Set 4

F. Blending Images

The final, blended images (panoramas) are produced by stitching together images from the previous step. We use the blending algorithm proposed by Pham and Samudrala (2019, gold submission as published on an earlier version of the course's website).

To blend the common regions between images, we leverage the information of those shared regions as encoded in the homography matrix. Let $H_{m,n}$ be the homography matrix which warps image m onto image n ($m < n$). We can decompose this homography matrix into several homography matrices between each pair of consecutively-indexed images.

$$H_{m,n} = \prod_{i=m}^{n-1} H_{i,i+1} = H_{m,m+1}H_{m+1,m+2}...H_{n-1,n} \quad (1)$$

This equation should cover warping all the lower-indexed images onto the middle-indexed image. To warp all the higher-indexed images onto the middle images, we use the inverse formula $H_{n,m} = H_{m,n}^{-1}$.

To reject a pair of images with too little overlap, we either drop the set of images before the lower-indexed images or drop the set of images after the higher-indexed images, depending on which of those two sets has fewer images.

The resulting panoramas are shown in Figure 46, Figure 47, Figure 48 (for the three train sets), Figure 49, Figure 50 (for the two custom sets), and Figure 51,Figure 52, Figure 53, Figure 54 (for the four test sets).

Fig. 46. Panorama Output for images in Train Set 1

Fig. 47. Panorama Output for images in Train Set 2

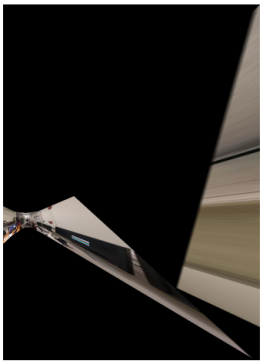

Fig. 48. Panorama Output for images in Train Set 3

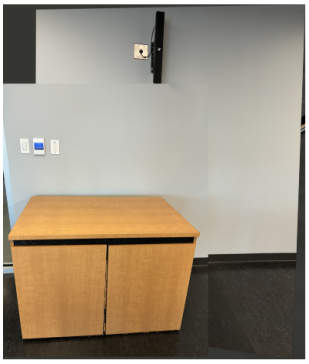

Fig. 49. Panorama Output for images in Custom Set 1

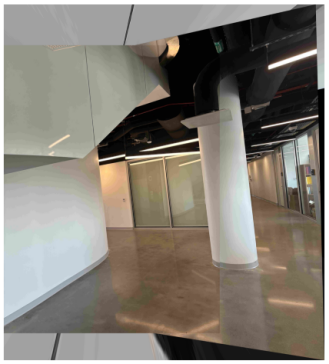

Fig. 50. Panorama Output for images in Custom Set 2

Fig. 51. Panorama Output for images in Test Set 1

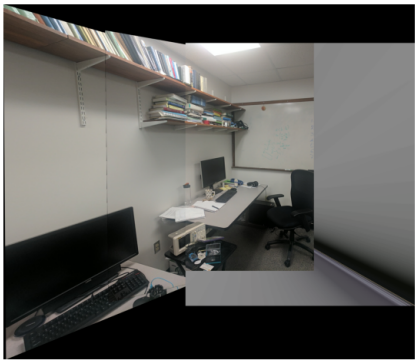

Fig. 52. Panorama Output for images in Test Set 2

Fig. 53. Panorama Output for images in Test Set 3

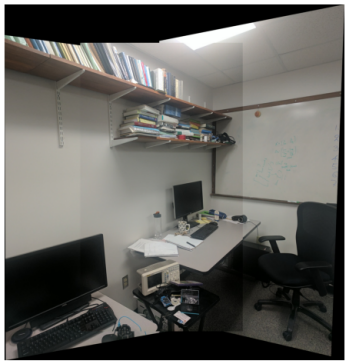

Fig. 54. Panorama Output for images in Test Set 4

II. Phase 2: Deep Learning Approach

In this section, we present our implemented deep learning approach to perform homography estimation and stitch images. Here we train our network both in a supervised and unsupervised way.

A. Data Generation

We follow the instruction on the web page and from [1] to generate synthetic data to train our model. Choose the same image patch size (128×128), and amount of perturbation ($\rho = 16$).

B. Supervised Model

The network architecture for the supervised model receives a pair of image patches, denoted as Patch A and Patch B, as its input. It is designed to output a 4-point homography, represented as H_{4Pt}, which delineates the transformation between these two patches. This architecture was illustrated as a figure in [2].

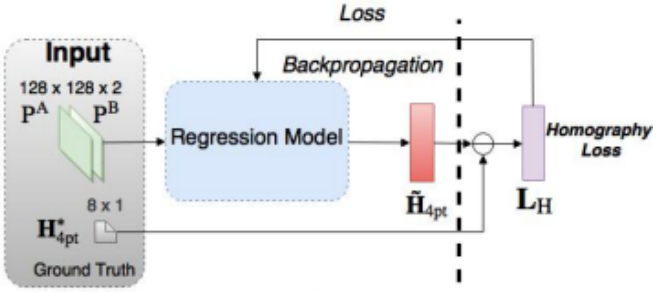

Fig. 55. network architecture for the supervised model

We use Adam optimizer with learning rate $1e-4$ and batch size 32 to train the network. We normalize image by dividing image pixel value by 255, and the H_{4Pt} values into the range $[-1, 1]$ help the network converge better.

We present a plot of the training loss vs. Epochs. Here, the loss is the mean squared error between the predicted and the groundtruth H_{4Pt} values after we have scaled them into the range $[-1, 1]$.

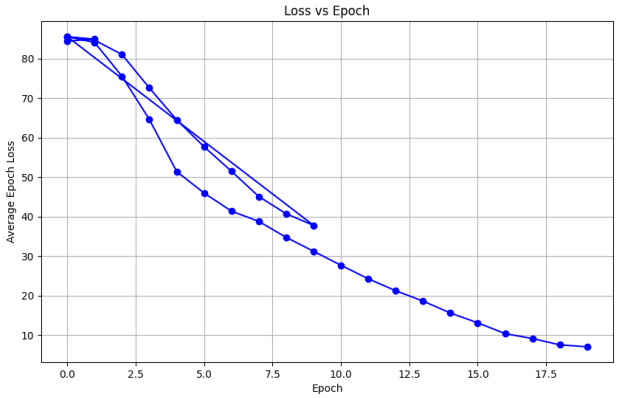

Fig. 56. Supervised loss vs epoch

We train 20 epochs, and the loss is very close to 0. And we provide 4 random images to show our work. Blue for original PB and Red for Predict PB.

Fig. 57. Pre PB vs Ori PB for image 305 in test set

Fig. 58. Pre PB vs Ori PB for image 671 in train set

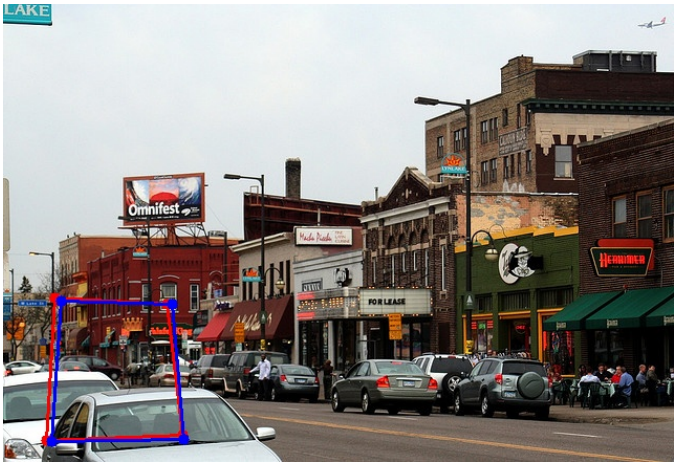

Fig. 59. Pre PB vs Ori PB for image 359 in train set

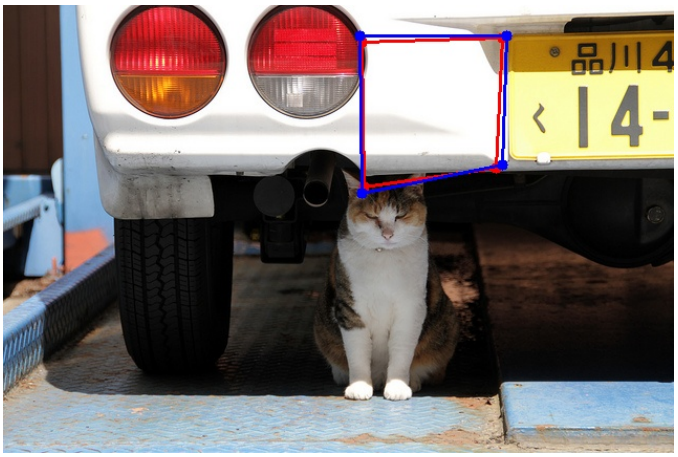

Fig. 60. Pre PB vs Ori PB for image 211 in val set

C. Unsupervised Model

The network architecture for the unsupervised model also receives a pair of image patches, denoted as Patch A and Patch B, as its input. It is designed to output a 4-point homography, represented as H_{4Pt}, which delineates the transformation between these two patches. This model architecture comes from [2].

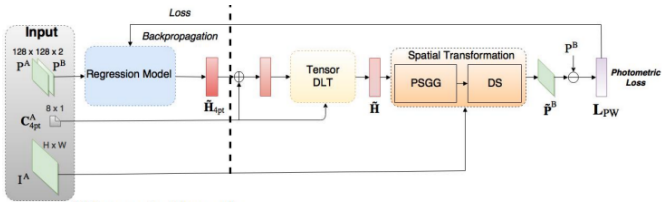

Fig. 61. network architecture for the unsupervised model

We also use Adam optimizer with learning rate $1e-4$ and batch size 32 to train our unsupervised model. We also normalize image by dividing image pixel value by 255, and the H_{4Pt} values into the range $[-1, 1]$ help the network converge better.

We also train 20 epochs, and the loss is not very close to 0. And we also provide 4 random images to show our work.

Fig. 62. Pre PB vs Ori PB for image 584 in test set

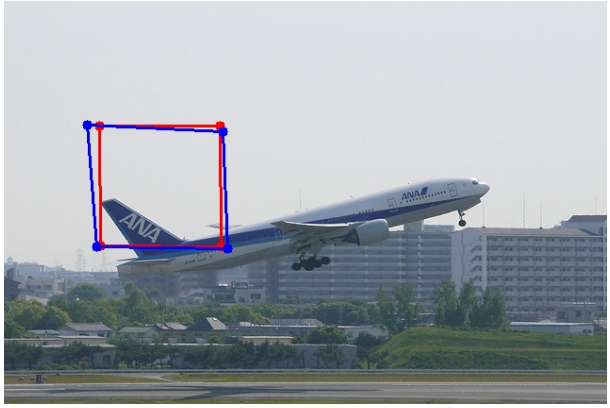

Fig. 63. Pre PB vs Ori PB for image 876 in test set

Fig. 64. Pre PB vs Ori PB for image 386 in train set

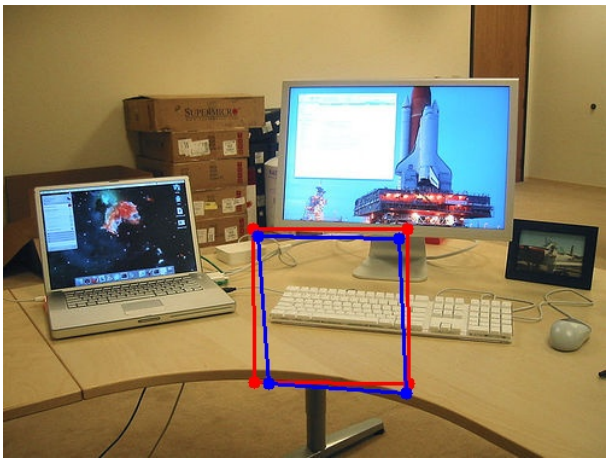

Fig. 65. Pre PB vs Ori PB for image 610 in val set

D. Performance on Train/Val/Test set

We record the mean corner pixel error of our models on the train, validation, and test set. The way we calculate the mean corner error is shown below.

$$||\widetilde{H_{4Pt}} - H_{4Pt}||_2$$

Fig. 66. EPE error

We also record the average forward pass run-time of all three set on GeForce RTX 2060 GPU.

Model name	Train EPE	Val EPE	Test EPE	Run-time
Supervised	12.74px	10.58px	12.59px	0.023s
Unsupervised	15.44px	17.61px	17.53px	0.027s

TABLE I
AVERAGE EPE AND AVERAGE FORWARD PASS RUN-TIME RESULTS OF OUR MODELS ON THE TRAIN/VALIDATION/TEST SET.)

Our models exhibit robust generalization capabilities on the test dataset, with the supervised model demonstrating commendable performance and rapid execution.

E. Comparision Images

First, we identify high-correlation regions based on Phase 1 and then extract a 128*128 patch. Utilizing the model constructed from the phase, we warp the perspective of the second image to align with that of the first image. Similarly, the fourth image is warped to match the viewpoint of the third image. Subsequently, we blend the first and second images, and likewise blend the third and fourth images, thereby generating two medium-sized images. This process is repeated three times, we choose first 8 images.

The operation is then replicated, culminating in the blending of these images to create a singular, comprehensive large-scale image.

Fig. 67. Comparision Images 1

REFERENCES

[1] D. DeTone, T. Malisiewicz, and A. Rabinovich, *Deep Image Homography Estimation*.

[2] T. Nguyen, S. W. Chen, S. S. Shivakumar, C. J. Taylor, V. Kumar, *Unsupervised Deep Homography: A Fast and Robust Homography Estimation Model*.