Project 1
Autopano

Using 1 late day

Sakshi Kakde
M. Eng. Robotics
University of Maryland
College Park, MD, 20742
Email: sakshi@umd.edu

Gokul Hari
M. Eng. Robotics
University of Maryland
College Park, MD, 20742
Email: hgokul@umd.edu

***Abstract*—The report describes in brief our solutions for the project 1. The report is divided into two sections. First section explores the traditional approach to find a homography matrix between a set of two images. Second section describes the implementation of a supervised and an unsupervised deep learning approach of estimating homography between synthetically generated data.**

I. Phase 1: Traditional Approach

In this section, we present a traditional approach to create panorama form a given set of images. We first detect the corners form the images and then try match them to compute a homography matrix between them.

A. Corners Detection

The first step in computing a homography matrix between two images is to detect the corner points. We have detected the corners using both Harris and Shi-Tomashi method.

***Harris Corner Detection**:* The *cv2.cornerHarris* function from OpenCV library uses Harris method for corner detection. The output from this function is the corner strength for each pixel, which is used for adaptive non-maximal suppression discussed in section I-B.

(a)

(b)

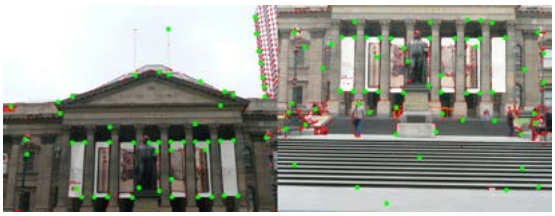
Fig. 1. Corner detection using *cv2.goodFeaturesToTrack*(a) and *cv2.cornerHarris*(b)

***Shi-Tomashi Corners Detection**:* The *cv2.goodFeaturesToTrack* function from OpenCV library uses Shi-Tomashi method along with a non-maximal suppression algorithm to obtain uniform corner points. Hence, the output form the function are corner co-ordinates and not corner strengths. Therefore, we could not use the output from *cv2.goodFeaturesToTrack* for the adaptive non-maximal suppression discussed in section I-B. We have used Shi-Tomashi Corners Detection for all image sets except for Set1 and TestSet1.

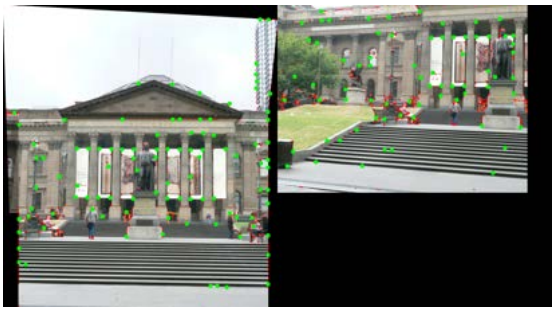
A comparison between corner detection using method I-A and method I-A has been made in figure 1.

B. Adaptive Non-Maximal Suppression (ANMS)

As seen in figure 1b, the detected corners are unevenly distributed. To make them evenly distributed, ANMS will try to find corners which are true local maxima. The output of ANMS is shown in figure 2.



(a)



(b)

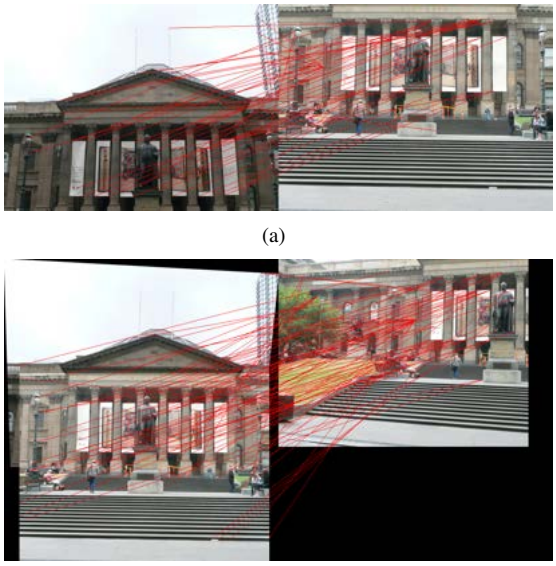
Fig. 2. Output after applying ANMS(green) on corners detected using Harris method(red) for Set1

C. Feature Descriptor

After we get the corner points, we need a descriptor to describe the feature for each point. To obtain that, a patch of size 40×40 centered at each corner point is used. This patch is then blurred and sub-sampled to a dimension of 8×8, which is then flattened to obtain a 64×1 vector.

D. Feature Matching

Now that we have a feature descriptor for each corner point, we can find the point matches from two images. For a corner point from image 1, we computed the sum of square differences between all points in image 2. Then we found a best match as the point with lowest distance and a second best match with second lowest distance. If the ratio of lowest distance and second lowest distance is less than a particular value, we accept the matched pair, else we reject it. The output afer feature matching is shown in figure 3.



(a)

(b)

Fig. 3. Matched pairs after feature matching for Set1

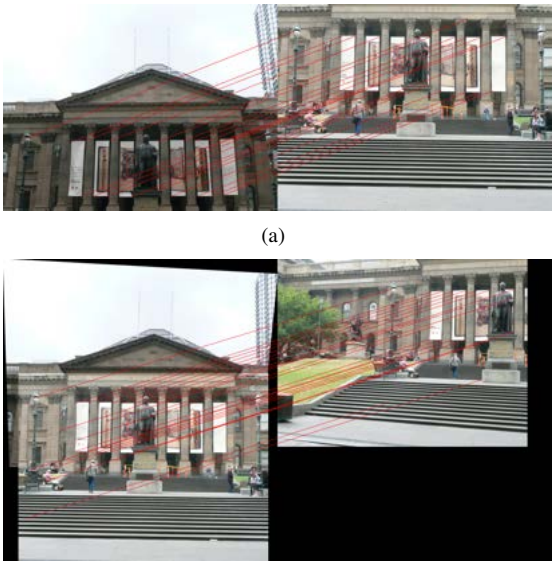
E. RANSAC for outlier rejection and to estimate Robust Homography

We could visually see in figure 3 that there are wrong match pairs present. These outliers could terribly affect our homography calculations. Therefore, to filter out these wrong matches and compute a reliable homography, Random Sample Concensus method has been used. The output after filtering out the wrong matches is shown in figure 4.

F. Blending Images

To blend two images, we used a simple overlapping method. Using the calculated homography matrix, we transformed image 1 on the projective plane of image 2. Then to avoid negative values, we translated the projected image and simply overlapped it with image 2(Refer figure 5).

The issues we faced are as follows:



(a)

(b)

Fig. 4. Matched pairs after feature matching for Set1

(a) (b)

Fig. 5. Panorama for Set1

Arrangement: If the number of images to be blended are less than or equal to three, then the approach is straightforward. We can go on blending the images sequentially. For example, if we have three images, the blended output of image 1 and image 2 will be merged with image 3. However, when the number of input images increases, this approach is not always effective. For example, if we have eight input images(as in Train/Set3), then by the time we reach the last image, the initial images become very distorted. To solve this problem, we divide the image set into two parts: first half and second half. We go on blending the pair of images from first half image set in forward direction, until we get a single blend for first half image set. Similarly, we blend the pair of images from second half image set in reverse direction, until we get a single blend for second half image set. If the provided images are in sequence, then the first half and second half blends will have a common area, that can be used to obtain a final merge. We will call this method as half split method in this report. Refer figure 6 for the block diagram of the method used for blending.

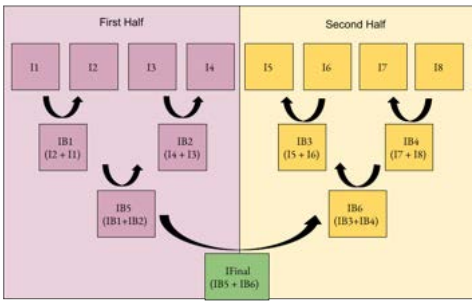
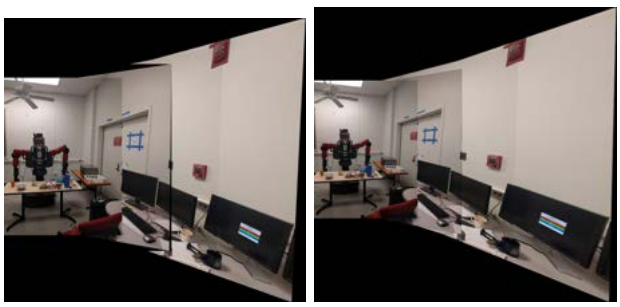


Fig. 6. Half split method for image blending

Overlap*:* When blending two panoramic images, we were getting black gaps as shown in figure ADD. We added a small check while overlapping to avoid these black gaps. The result after the check is shown in figure 7.



(a) (b)

Fig. 7. Black gap present(a) vs. black gap eliminated(b)

Unrelated Images*:* If the two images are unrelated or is having very little overlap, then the output of RANSAC looks like the one shown in figure 8. We used the criteria that for given pairs of matches, if multiple points from image 1 are having the same match from image 2, then ignore the image 2. However, this approach is dependant of the sequence at which the unrelated image appears.

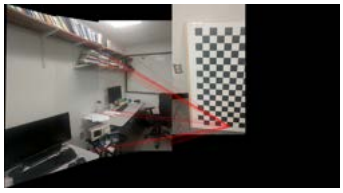


Fig. 8. Image Blending for unrelated images

G. Analysis

The assumption while computing homography matrices from two images is that the points on the images are planar, i.e., $z = 0$. z can be approximated to 0 for the images taken from far. However for closer images(indoor images from dataset), we think this assumption is not true and hence the computed homography matrix cannot be that reliable.

As discussed in section I-F, we tried using the method illustrated in figure 8 to ignore the unrelated images. However, this method is dependant on the sequence in which we stitch the image. In the code, we have provided two options for the user: go sequentially while stitching, or use half split method as in figure 6. This method worked for TestSet4 when the images were stitched sequentially. However, for TestSet2, where the overlap between consecutive images is less(and thus are unrelated), this method failed. Though the algorithm was able to determine that images are unrelated, it was not able to decide which image to keep and which one to discard. We were unable to expand the panorama as wrong image got discarded. An optimal solution would be to automate this choice. However, due to limited time, we were not able to extend our work. The problems with TestSet2 are illustrated in figure 45.

When we have a large number on input images(Train Set 3), the results are not always the same. Ideally this should not happen. We think the reason for this can be, as we move ahead in the stitching algorithm, the size of the image increases. And since we are detecting a limited number of corners, the corners position might be changing resulting in changed results. We intend to solve this problem in future for a reliable stitching method.

The output images after each step for all the test and train set images are attached in the appendix section(IV).

II. Phase 2: Deep Learning Approach

In this section, we implement a couple of deep learning approaches, a supervised [1] and an unsupervised learning approach [2], to estimate homography between two images. These deep networks input a 128×128 image patch from both images stacked depth wise resulting in an image pair input shape of $128 \times 128 \times 2$. To implement these models we use a subset of MS Coco data-set to generate synthetic training/testing examples by applying random projective transformations on natural images as suggested in [1]. The data generation procedure is explained in subsection II-A

A. Data Generation

To generate the required data to train and evaluate the networks, we first resize all images in data set to a height and width of 240,320. We randomly crop a patch P_A of size 128×128 in a image I_A at a location p. The corner coordinates of the patch P_A with respect to the image I_A is denoted as C_A. The corners C_A are randomly perturbed by a $[+\rho, -\rho]$ to obtain C_B. The homography H_{AB} is estimated between these four corner correspondences, C_A and C_B. The inverse of H_{AB}, i.e H_{AB}^{-1} is applied to the image I_A to obtain I'_A. We crop patch P_B from I'_A at the same location p. Image I_B can be obtained by warping image I_A with the homography matrix H_{AB}. We also generate 4 point homography $H4_{AB}$,

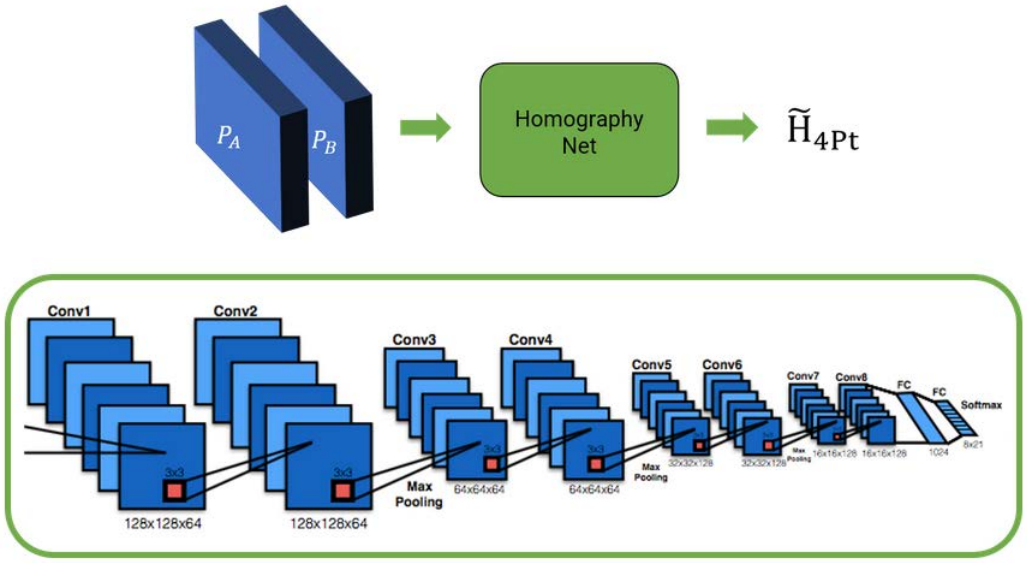


Fig. 9. Block diagram - Supervised model

given in equation 1, between P_A and P_B with the corner correspondences.

$$H4_{AB} = C_A - C_B \tag{1}$$

The patch-pairs P_A and P_B (converted to grayscale, single-channel), are stacked together to form the input of our deep networks. $H4_{AB}$ is taken as the labels for the supervised approach. This approach is repeated for entire image data set.

B. Supervised Model

DeTone et al. [1] proposed HomographyNet, a deep convolutional neural network trained to compute the 4 point homography between two images.

Architecture: This network uses 3x3 convolutional blocks with Batch-Normalization and ReLU activations, and are architecturally similar to Oxford's VGG Net [3]. The network uses eight convolutional layers with a max pooling layer, of pool size = 2×2, stride = 2, after every two convolutions. The first four convolutional layers have 64 filters per layer and the next four convolutional layers use 128 filters. These convolutional layers are followed by two fully-connected-Dense layers. Dropout with a probability of 0.5 is applied after the last convolutional layer and the first fully-connected layer, to avoid overfitting. The first Dense layer has 1024 units and the final Dense layer outputs 8 units. The model block diagram is shown in Figure 9, referred from [1]

Training parameters: We used Adam optimizer with a learning rate of $1e^{-4}$ to train our model for 100 epochs with 25,000 training examples generated from the training set of 5000 images. We calculate euclidean L2 Loss as suggested in the official implementation and use mean absolute error as our training performance metric. We will discuss the model's performance in subsection II-D

C. Unsupervised Model

The scope of practical application of deep homography estimator networks is limited by the requirement of large number of ground truths in the supervised learning approach. Thus, Nguyen et al. [2] proposed an unsupervised learning approach of homography estimation network.

Architecture: This network involves an extension of the 8-CNN layer HomographyNet, where the 4 point homography predictions are used to warp the image I_a to compute the photometric loss function given in equation 2

$$Loss_{Photometric} = ||w(I_A, H4_{AB}), I_B||_1 \tag{2}$$

where, w is a warping function that warps image I_a using a given 4 point homography $H4_{AB}$.

To warp a given image using 4 point homography, we need to convert it to it's equivalent homography matrix H_{AB} using direct linear transformation. This H_{AB} matrix is now used to warp the image using a differentiable warping layer, an extension of spatial transformer networks proposed by Jaderberg et al. [4]. The block diagram of the unsupervised model's pipeline, referred from [2], is shown in figure 10c

Training Parameters: We used a similar training set-up to that of the supervised model, with Adam optimizer, learning rate $1e^{-4}$ for 100 epochs and used photometric loss function in equation 2 to monitor the unsupervised model's training performance. We will discus the performance of this model in

D. Performance Results

Supervised model: The L2 loss and mean absolute error (MAE) during training the supervised model is shown in figure 11. We notice that, at the loss quickly converges and results an MAE of 3.414 with the training data at the end of 100 epochs. Our supervised model resulted an average mean corner error of **5.966** on 5000 random test patches generated from

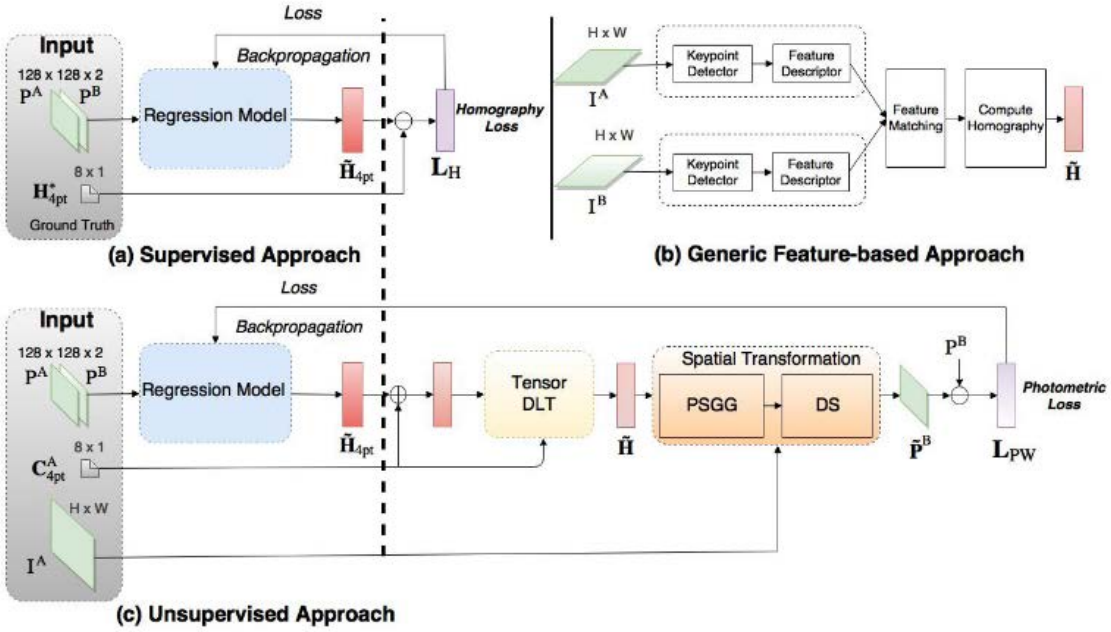


Fig. 10. Block Diagram - Unsupervised Model

TABLE I
SUPERVISED AND UNSUPERVISED MODEL PERFORMANCE OVER ALL 3 DATASETS

	MCE_{sup}	MSE_{sup}	MCE_{unsup}
Train Set	1.91	6.39	9.67
Val Set	5.53	54.52	9.52
Test Set	5.59	55.59	9.42

1000 test images in test data-set that was released 24hrs before submission.

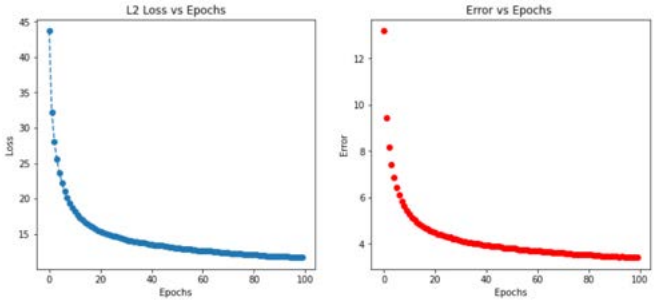


Fig. 11. Supervised model Results (Left:- Training Loss ; Right:- Training Error)

These mean corner error (MCE) and mean squared error (MSE) results are shown in table I under columns MCE_{sup}, MSE_{sup},

Unsupervised model: The Photometric loss during training the unsupervised model over 100 epochs is shown in figure 12. We notice that, the photometric loss is around 20 and is still converging at the 100th epoch. We could not train the model for more epochs due to computational and time constraints but we can expect the model to result better performance if trained longer based on the inference. The mean corner error metric for first 500 patches all three datasets is found in the last column (MCE_{unsup}) of table I

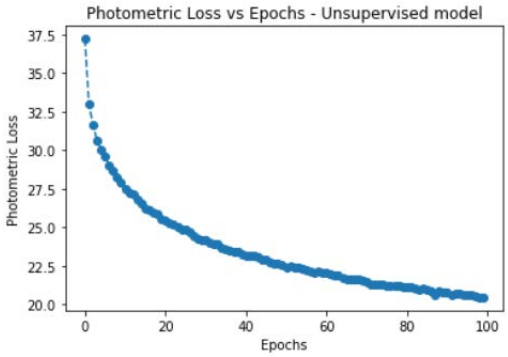


Fig. 12. Unsupervised model - Photometric Training Loss

Inference: In Table I , we notice that the results of supervised model shows some model-overfitting, as the performance with the train-set is much better than that of the test and validation sets. Meanwhile the unsupervised model, though performs poorer due to incomplete training at 100 epochs, shows no sign of overfitting.

In addition to the results presented, during our initial experiments with the supervised model, we initially trained the model for 50 epochs with only 5000 image patches, due to computational constraints, opposed to our final submission trained with 25,000 images. In this experiment, though the model's training loss and train-set error converged at the end of 50 epoch training, the model was unable to generalise well

on the validation data-set due to overfitting. This behavior is in accordance to the findings by Nguyen et al. [2], a notable shortcoming of supervised approach. However, as we increased the number of training samples to 25,000, the model was able to generalise better, relatively.

Figure 13 shows the projective transformations performed on patches from images sampled from test set, with ground truth corners (blue) and predicted corners (green) from the supervised and unsupervised models

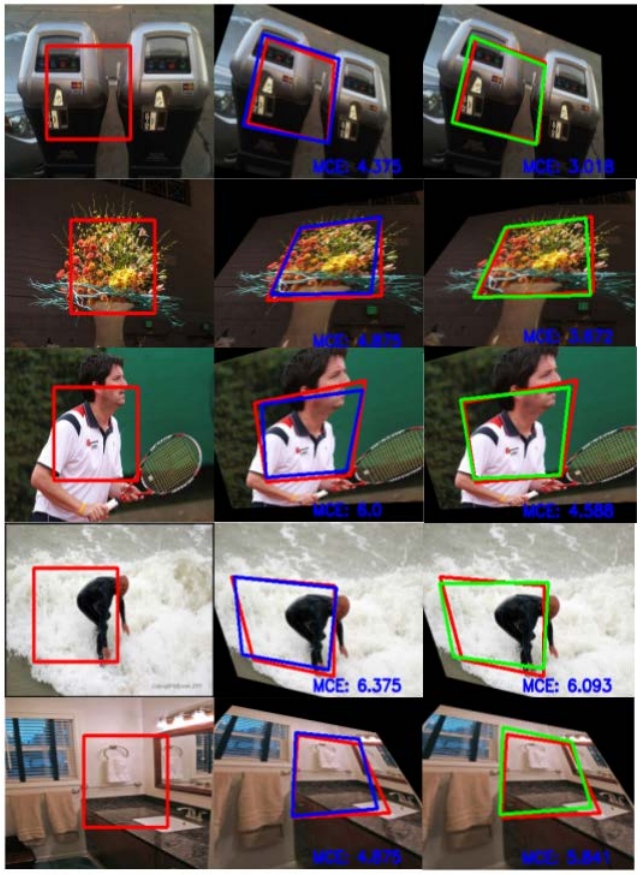


Fig. 13. Visualising model performance - Ground Truth(Red) Unsupervised (Blue) and Supervised (Green)

We tried to integrate the models generated in the deep learning approach to the image blending pipeline built in Phase 1, by utilising the homography predicted by the deep network to form a panoramic image blending pipeline. We cropped five patches from the pair of images to be stitched and computed the average homography predicted between the 5 patches. However, image stitching was unsuccessful with this predicted output of the deep learning approach.

III. Conclusion

In this project, we have implemented three approaches to estimate homography in Phase 1 and Phase 2, which is represented in figure 10. In Phase 1, the generic feature-based approach was followed to estimate homographies and perform the image stitiching successfully for given test sets. In Phase 2, we implemented both the supervised and unsupervised deep learning approaches to estimate homography for the given MS Coco dataset.

References

[1] DeTone, Daniel, Tomasz Malisiewicz and Andrew Rabinovich. "Deep Image Homography Estimation." ArXiv abs/1606.03798 (2016): n. pag.

[2] Nguyen, Ty, S. Chen, S. S. Shivakumar, C. J. Taylor and V. Kumar. "Unsupervised Deep Homography: A Fast and Robust Homography Estimation Model." IEEE Robotics and Automation Letters 3 (2018): 2346-2353.

[3] K. Simonyan and A. Zisserman. Very deep convolu-tional networks for large-scale image recognition.CoRR,abs/1409.1556, 2014.

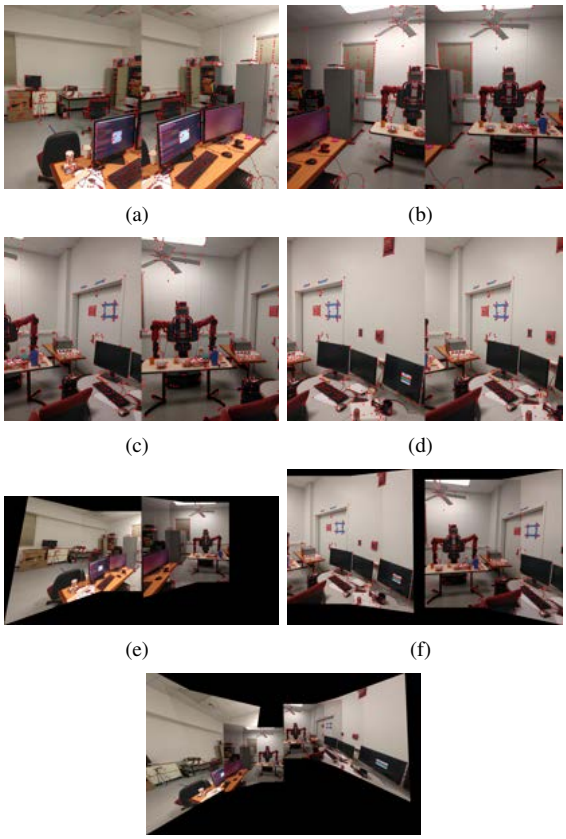
[4] M. Jaderberg, K. Simonyan, A. Zisserman,et al., "Spatial transformernet-works," inAdvances in Neural Information Processing Systems,2015, pp. 2017–2025

IV. Appendix - Phase 1

This section is to print the panorama images and the intermediate steps involved in phase 1, for the Test/custom datasets, from .

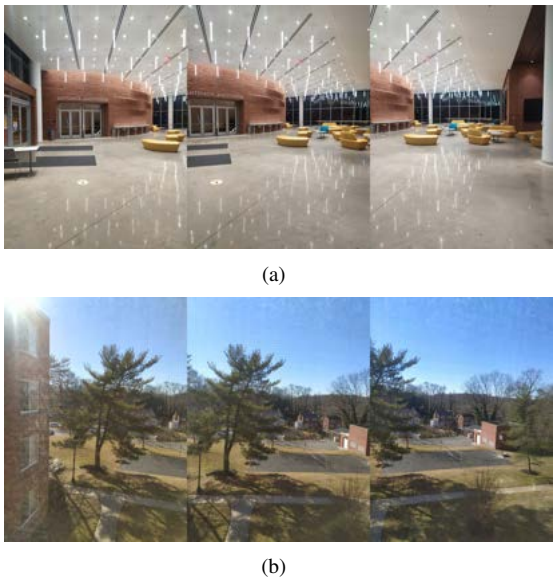
(a) (b)

Fig. 14. Shi-Tomashi corners for Set2



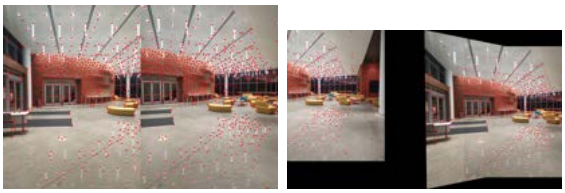
(a) (b) (c) (d) (e) (f) (g)

Fig. 15. Shi-Tomashi corners for Set3



(a) (b)

Fig. 16. Input images for CustomSet1 and CustomSet2



(a) (b)

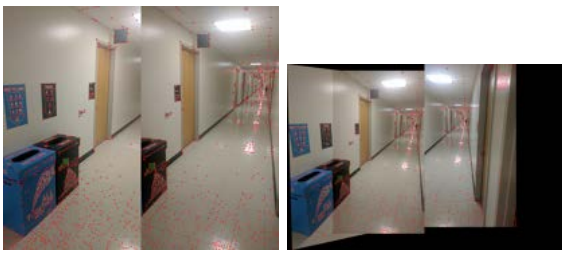
Fig. 17. Shi-Tomashi corners for CustomSet1

(a) (b)

Fig. 18. Shi-Tomashi corners for CustomSet2

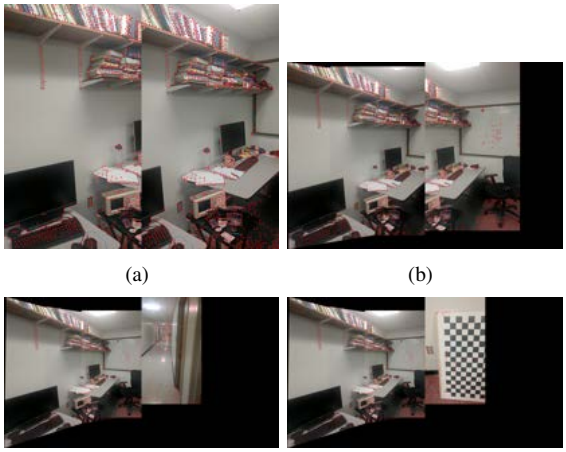
(a) (b)

Fig. 19. Harris corners and ANMS output for TestSet1



(a) (b)

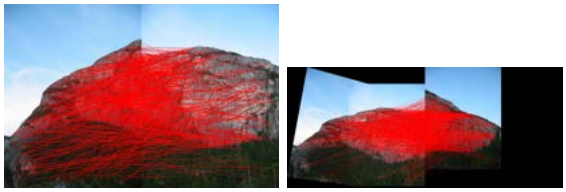
Fig. 20. Shi-Tomashi corners for TestSet3



(a) (b)

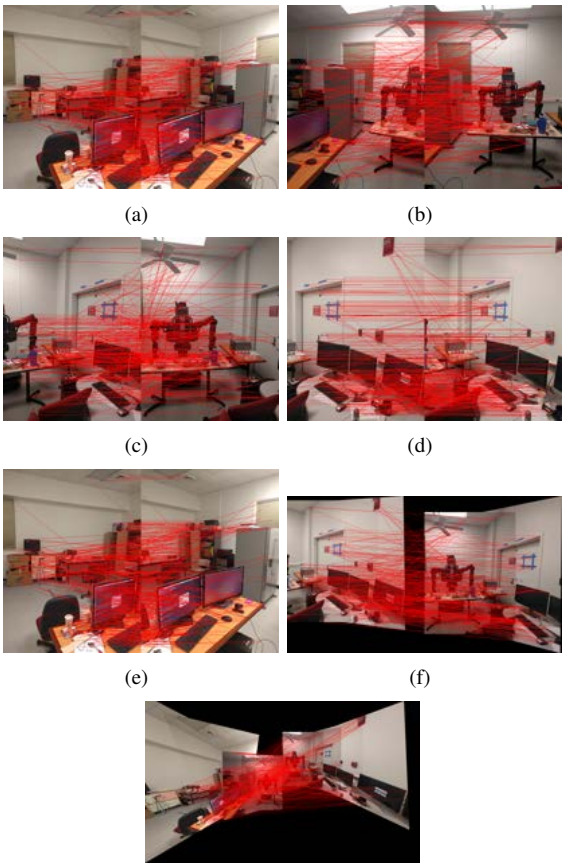
(c) (d)

Fig. 21. Shi-Tomashi corners for TestSet4



(a) (b)

Fig. 22. Matched Pairs for Set2



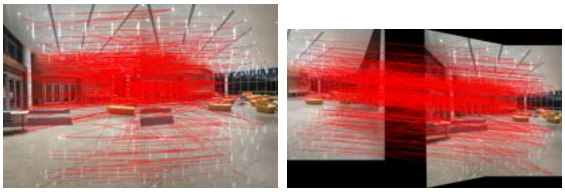
(a) (b)

(c) (d)

(e) (f)

(g)

Fig. 23. Matched Pairs for Set3

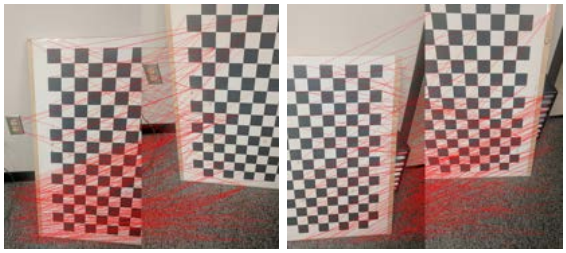


(a) (b)

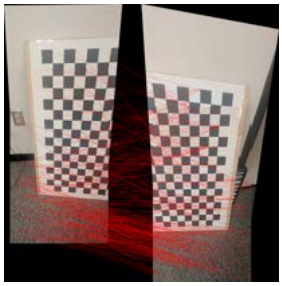
Fig. 24. Matched Pairs for CustomSet1

(a) (b)

Fig. 25. Matched Pairs for CustomSet2



(a) (b)



(c)

Fig. 26. Matched Pairs for TestSet1

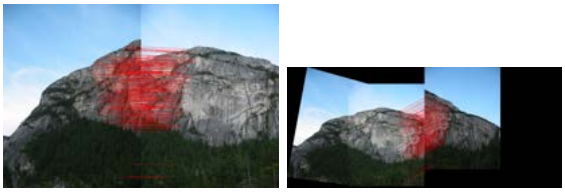
(a) (b)

Fig. 27. Matched pairs for TestSet3

(a) (b)

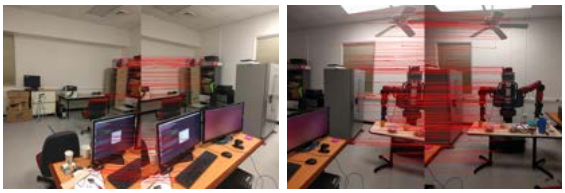
(c) (d)

Fig. 28. Matched pairs for TestSet4



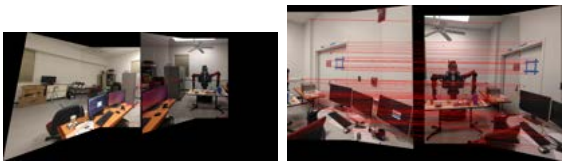
(a) (b)

Fig. 29. Filtered Matched Pairs for Set2



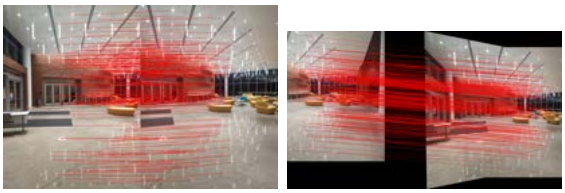
(a) (b)

(c) (d)



(e) (f)

Fig. 30. Filtered Matched Pairs for Set3

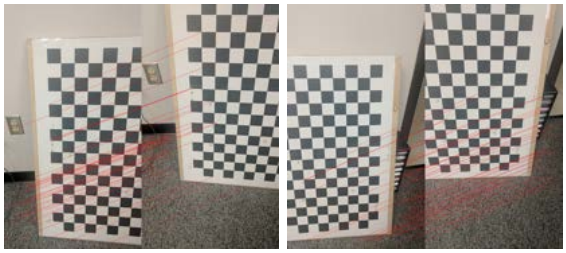


(a) (b)

Fig. 31. Filtered Matched Pairs for CustomSet1

(a) (b)

Fig. 32. Filtered Matched Pairs for CustomSet2



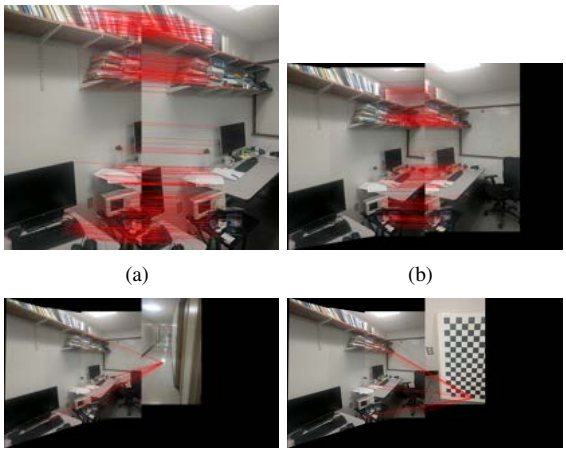
(a) (b)

(c)

Fig. 33. Filtered Matched Pairs for TestSet1

(a) (b)

Fig. 34. Filtered Matched pairs for TestSet3



(a) (b) (c) (d)

Fig. 35. Filtered Matched pairs for TestSet4

(a) (b)

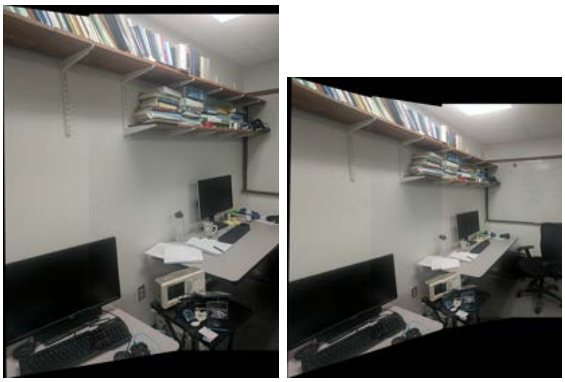
Fig. 36. Panorama for Set2

(a) (b)

Fig. 37. Panorama for CustomSet1

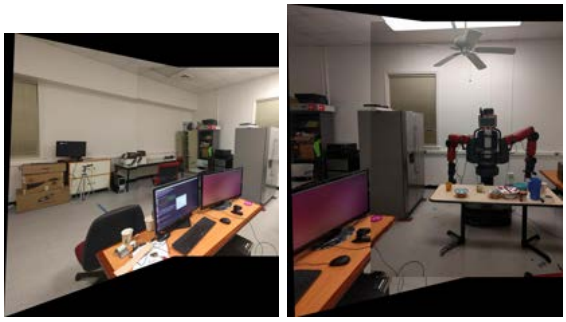
(a) (b)

Fig. 38. Panorama for CustomSet2

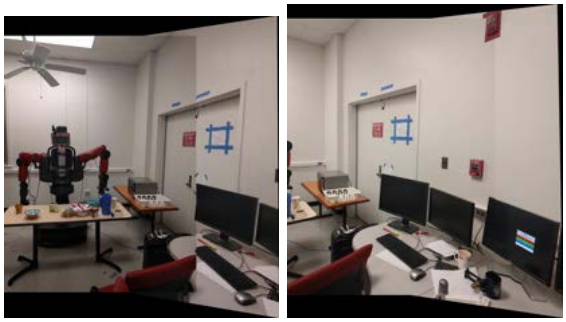


(a) (b)

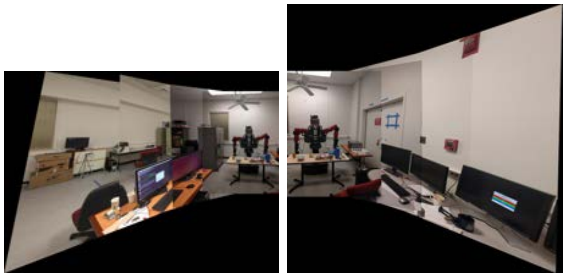
Fig. 39. Panorama for TestSet4



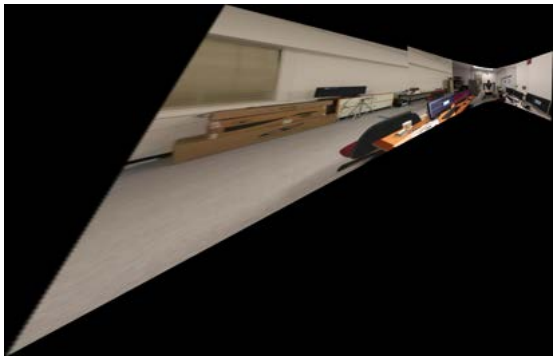
(a) (b)



(c) (d)



(e) (f)



(g)

Fig. 40. Panorama for Set3

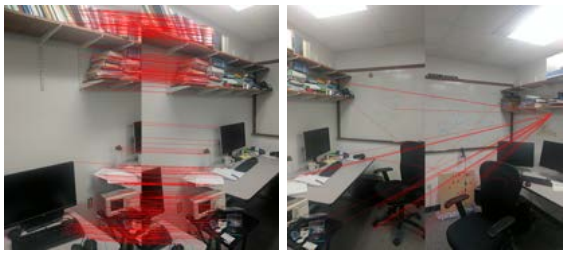
(a) (b)

(c)

Fig. 41. Panorama for TestSet1

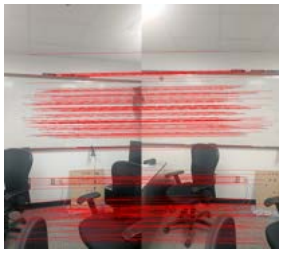
(a) (b)

Fig. 42. Panorama for TestSet3

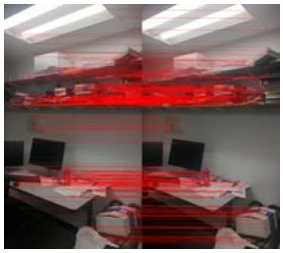


(a)

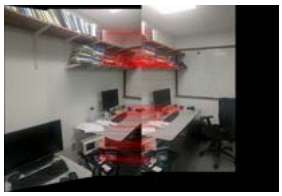
(b)



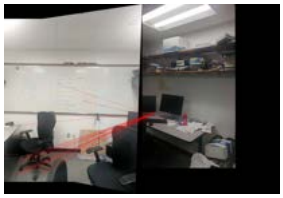
(c)



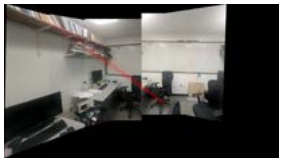
(d)



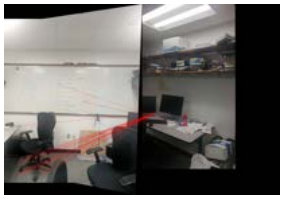
(e)



(f)



(g)



(h)

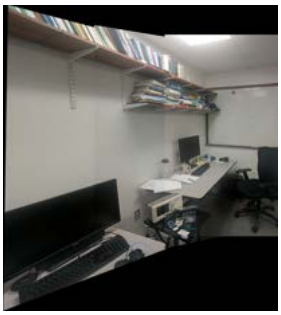
(i)

Fig. 43. Unrelated image detection for TestSet2.

(a)

(b)

(c)



(d) (e)

Fig. 44. Problems with TestSet2

(a)

Fig. 45. Problems with TestSet2. Since the image 4 in dataset is not overlapping with image 3, the panorama of first half till image 3. The panorama of first half and second half is not related by a common area. Hence, the final panorama(h) is just the first half.