Project 1: Auto Pano

Aadhya Puttur, Alex Chiluisa
Used on late day

I. Introduction

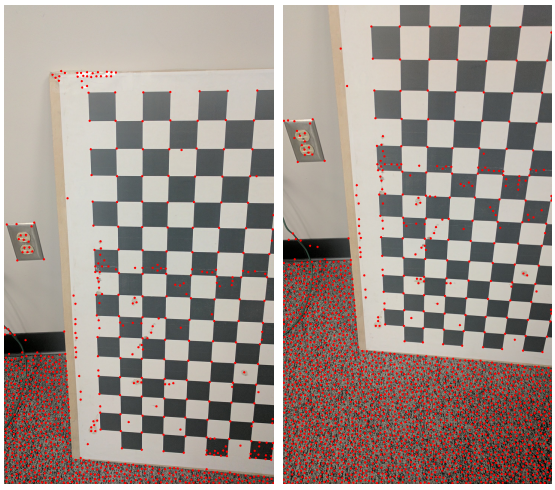
Homography is a powerful tool in computer vision to define the transformation between a set of images. It relies on the plane intersection of regions of interest through matching relevant features with lines. This work describes a traditional approach to blend a set of images that implements several techniques, such as ANMS and RANSAC, as well as, explores supervised and unsupervised approaches to estimate homography between two images.

II. Phase1: Traditional Approach

To implement the traditional approach, we follow the six-step: 1) Corners detection 2) Adaptive non-maximal suppression (ANMS) on the corners 3) Features description 4) Features matching 5) RANSAC to estimate homography. 6) Blending images.

A. Corner Detection

First, we detect corners in the input images using the goodFeatureTrack algorithm for each set of image on the Test folder. The result are shown in Fig. 1 to Fig. 4



(a) (b)

Fig. 1. Corners detected in (a) image 1 (b) image2 of Test 2

A. Chiluisa is with the Department of Robotics Engineering, Worcester Polytechnic Institute, Worcester, MA 01609, USA (e-mail: ajchiluisa@wpi.edu)

A. Puttur is with the Department of Computer Science, Worcester Polytechnic Institute, Worcester, MA 01609, USA (e-mail: aputtur@wpi.edu)

(a) (b)

Fig. 2. Corners detected in (a) image 1 (b) image2 of Test 2

(a) (b)

Fig. 3. Corners detected in (a) image 1 (b) image2 of Test 3

B. Adaptive non-maximal suppression

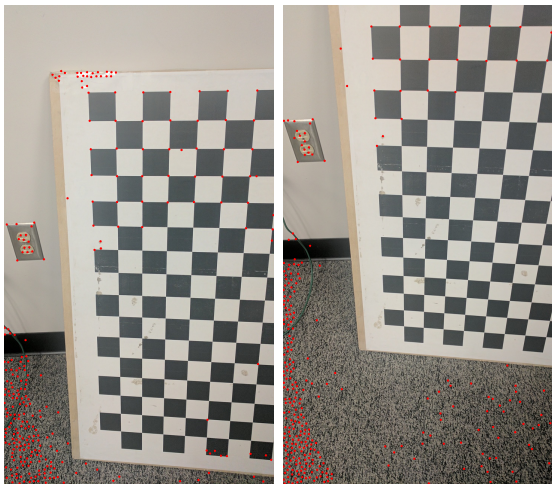
We visualize that for each image in section II-A, they have a large number of detected corners, with several repeated corners. To remove these redundant corners, we applied adaptive non-maximal suppression (ANMS) to keep the corners that are evenly distributed across the whole image as we shown in Fig. 5 to Fig. 8

(a) (b)

Fig. 4. Corners detected in (a) image 1 (b) image2

(a) (b)

Fig. 6. ANMS result for images in Test 2



(a) (b)

Fig. 5. ANMS result for images in Test 1

(a) (b)

Fig. 7. ANMS result for images in Test 3

C. Features description

In order to stitch a set of images, we need to match each feature point (corners) of one image with the other. Therefore, we described each corner by a feature vector. We selected the best 100 points of the ANMS corners. A sample of the results are shown in Fig. 9

D. Features Matching

Using the information of the feature vectors, we matched the corners between two images. We selected every corner of image 1 and computed the sum of square differences (ssd) between all points in image 2. Then, we took the ratio of the best match (lowest distance) to the second best match (second lowest distance) and compare it with a default ratio of 0.99, if the calculated ratio is lower than the default ratio, we kept the matched pair. Fig. 10 to Fig. 13 shows the results.

E. Random Sample Consensus RANSAC

As we can see in Fig. 10 - 13, there are some outliers that we need to reject. Applying a Random Sample Consensus (RANSAC), we remove those outliers. Subsequently, with the remaining valid features, we computed the homography matrix. The results after applying RANSAC are shown in Fig. 14 to Fig. 17 We learned that although these methods help with detect keypoints for stiching, they can be inaccurate at times, failing to detect important corners or incorrect ones as well. This can be due to lighting and viewpoint differences.

F. Blending Images

Having the homography matrix we warp and blend the image set. we go forward to warp and blend them together. We use a built-in function to generate the final result. Due to the time constraint, we did not implement a more elegant solution to blend the images. Nevertheless, we faced problems generating the final panoramic images. We

(a) (b)

Fig. 8. ANMS result for images in Test 4

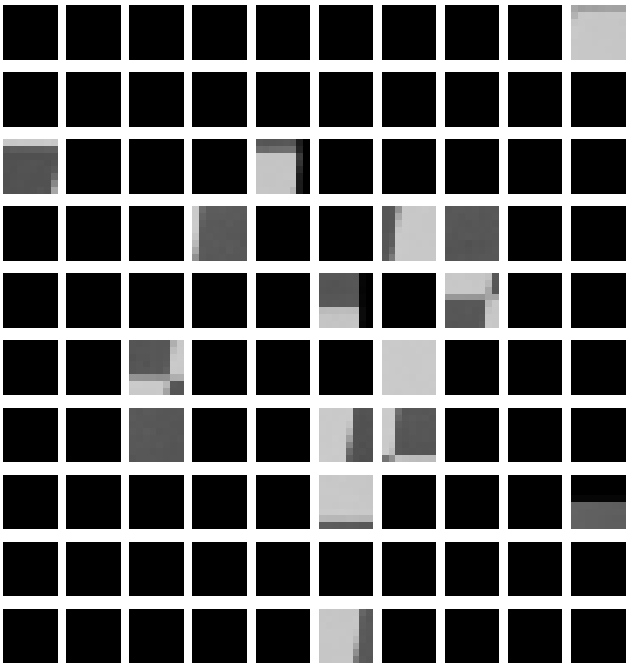


Fig. 9. 8x8 subsample patches in image 1 on Test 1

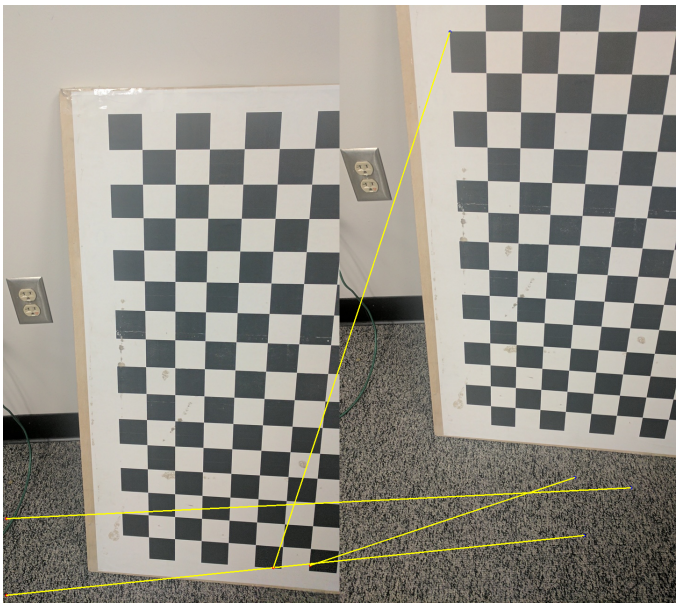


Fig. 10. Feature matching between image 1 and image2 on Test 1

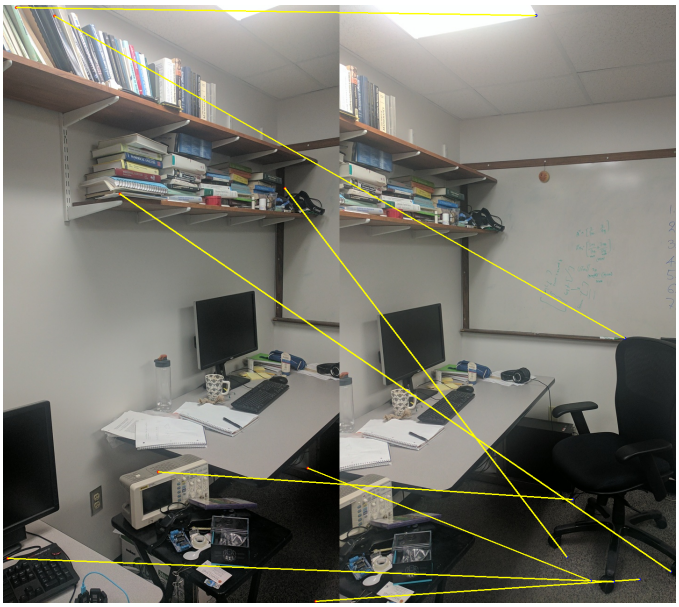


Fig. 11. Feature matching between image 1 and image2 on Test 2

encounter problems in the implementation of RANSAC that jeopardize the outcome, sometimes the homography matrix is not generated which causes the image to not blend properly. Further investigation is needed to iterate over the problem to mitigate it. The result of blending two images is shown in Fig. 18 and Fig. 19

III. Phase2

A deep convolutional neural network (CNN) are mostly commonly used to identify patterns in images and their strength comes from layering. The architecture consists of four types of layers: convolution, pooling, activation, and fully connected. The convulutional neural network that we are using is based on the paper ”Deep Image Homography Estimation”. We use CNN to estimate homography between a pair of images, also known as HomographyNet.

A. *Supervised*

In our supervised approach, we use the ground truth labels. To estimate homography we take random 128 by 128 patch from the image in grayscale and compare it to another patch that is produced using random perturbation. Then we get the homography of the two patches using ”cv2.getPerspectiveTransform” of the two sets of 4 corners of the images.

For the supervised model, we use The Layer takes in an input of size 2, which is the grayscale images of the (128 x 128) patches.

Fig. 12. Feature matching between image 1 and image2 on Test 3

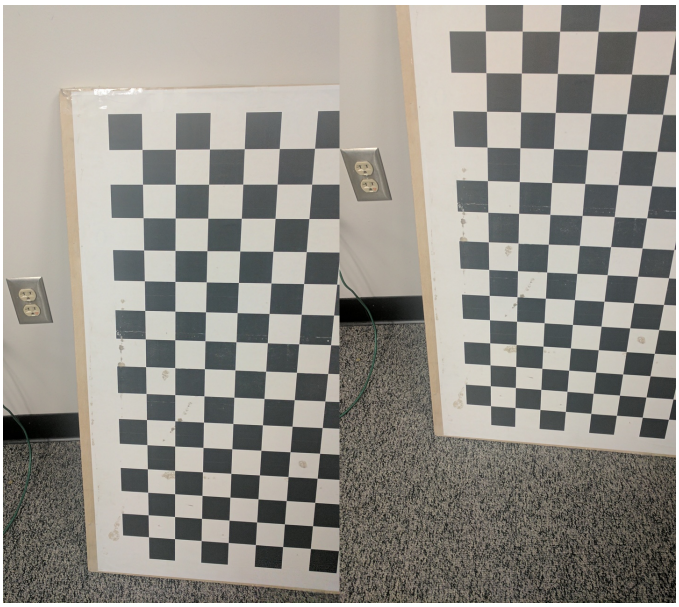


Fig. 14. RANSAC result for Test 1

Fig. 13. Feature matching between image 1 and image2 on Test 4

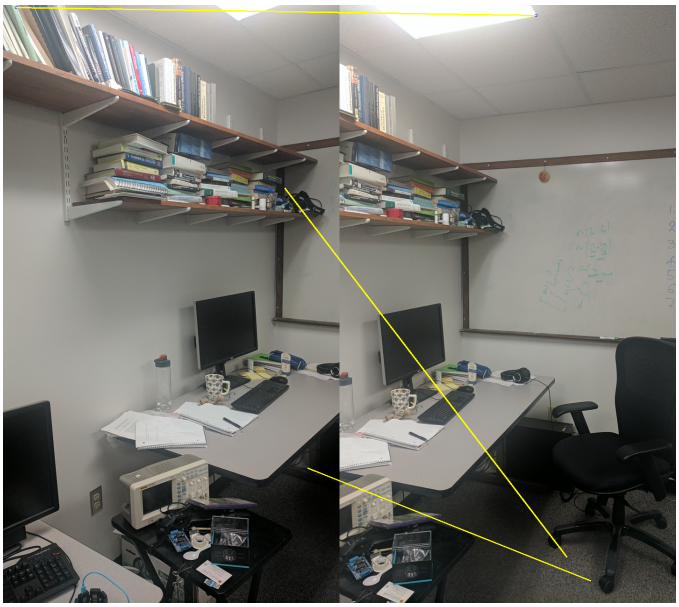


Fig. 15. RANSAC result for Test 2

The homography is calculated in eq. 1, where the [u,v] are mapped to $[u^{'}, v^{'}]$ which is our scaling factor. This was a similar process as phase 1, only this time we will want to calculate these offsets for each of the 4 corners of the patches with the warped patch. This can be calculated by using $\Delta u_1 = u_1^{'} - u_1$. This is also known as 4-point parameterization H_4point.

$$\begin{pmatrix} u' \\ v' \\ 1 \end{pmatrix} \sim \begin{pmatrix} H_{11} & H_{12} & H_{13} \\ H_{21} & H_{22} & H_{23} \\ H_{31} & H_{32} & H_{33} \end{pmatrix} \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} \quad (1)$$

Our network architecture is based on the architecture from "Deep Image Homography Estimation" where it uses 3x3 convulutional blocks, BatchNorm2D, and ReLU. At the end of theses convulutional layers we get a linear output of size 8. This architecture is shown in Fig. 22

With the supervised approach we compared the homography from the training images with ground truth H_4pt. This comparison allowed the model to train by using the loss calculated between the predicted 4-point homography and the ground-truth H_4pt homography. The model produced and used for testing had 32 parameters. H mixes in the rotation, translation, and scale of the homography transformation of an image. The rotation tends to have a small effect on the loss error because of it's small magnitude. In this method, we use pixel coordinate matching by warping the images and

Fig. 16. RANSAC result for Test 3

Fig. 17. RANSAC result for Test 4

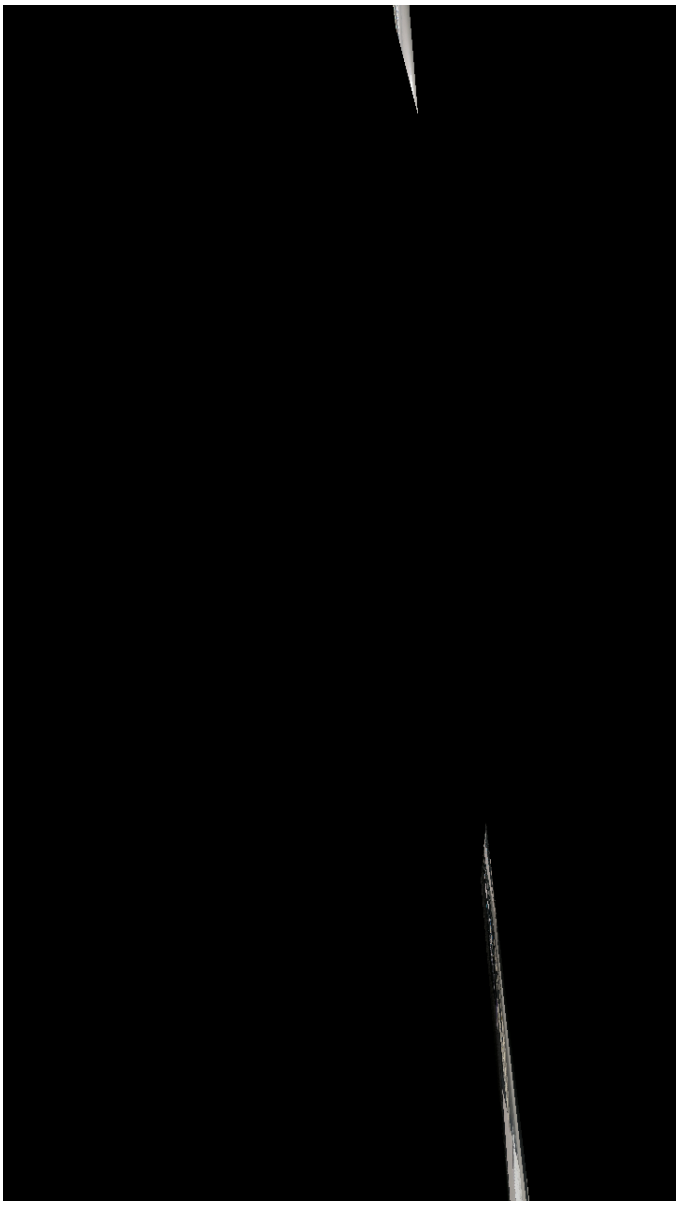


Fig. 18. Panoramic image for image set in Test 2

compare the shift of the pixels using an error metric such as the Euclidean L2 norm of the estimated 4-point homography versus the ground truth. Although we learned through this approach, our results are limited to synthetic datasets and costly labels, where in the unsupervised approach would work better in Fig. 21

B. Unsupervised

In the unsupervised approach, we do not use the ground truth. This method depends on features to compute the homography predictions through learning about the features. In this approach, we want to minimize the loss that is calculated using the L1 error vs using the L2 error that used for supervised.

$$L_{PW} = \frac{1}{|x_i|} \sum_{x_i} \left| I^A(\mathcal{H}(x_i) - I^B(x_i)) \right| \tag{2}$$

$$L_H = \frac{1}{2} \left\| \widetilde{\mathrm{H}}_{4\mathrm{pt}} - \mathrm{H}^*_{4\mathrm{pt}} \right\|_2^2 \tag{3}$$

Fig. 19. Panoramic image for image set in Test 2

(a) (b)

Fig. 20. Corners removed using ANMS in (a) image 1 (b) image2

(a) (b)

Fig. 21. patch A shown in image(a) is shown next to patch B in image(b) with a warped image

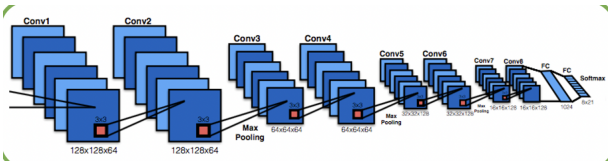


Fig. 22. Deep Image Homography estimation Network Architecture