Project 1: My AutoPano

Raghav Nandwani
M. Engg. Robotics
University of Maryland
College park, MD
Email: raghav15@terpmail.umd.edu

Yu Shen
M.S. Computer Science
University of Maryland
College park, MD
Email: yushen@cs.umd.edu

***Abstract*—The aim of this project was to stitch multiple images that are having some common overlapping area between them. The homography between these images was computed to transform from their respective projection plane to the another projection plane. This homography estimation was done using 2 approaches - the classical traditional approach and the deep learning approach. In deep learning we used both supervised and unsupervised approaches to estimate the homography.**

I. Phase 1: Traditional Approach

For traditional approach we extract features from the two set of images, match them and estimate homography by rejicting Outliers with the help of RANSAC. Then based on that Homography we stitch one image on top of the another, to create a seamless panaroma. the overview of the pipeline is shown in Fig. 1

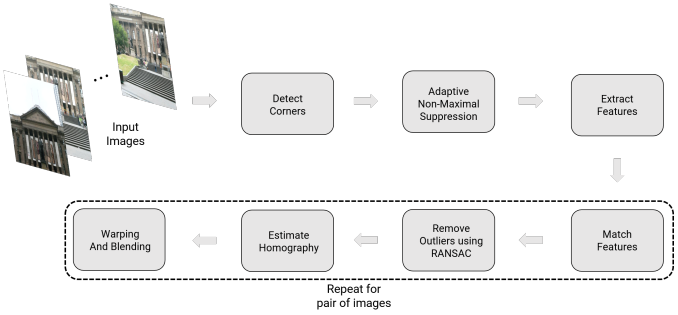
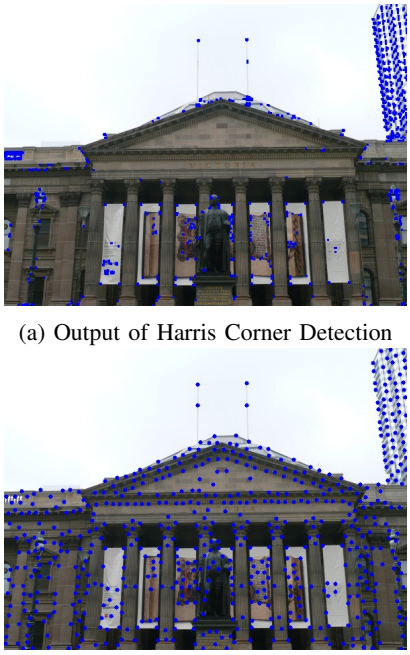


Fig. 1: Overview of Traditional panaroma stitching pipeline

A. Corner Detection

The first step involves the corner detection. Here for feature extraction we have used corner detection because corners are stable and uniquely identifiable when we want to find same corners in the other image.

For corner detection we used the standard in-built Shi Tomasi corner detection. We didn't used Harris because we found that in Harris every pixel in the image is given a score is for how likely this is a corner,instead in Shi-Tomasi we get the cordinates of the best corners that are there in that image.And we also found out that the corners in Harris weren't segregated well and many of the corners were clustered at one position as shown in Fig. 2.



(a) Output of Harris Corner Detection

(b) Output of Shi-Tomasi Corner Detection

Fig. 2: Comparison of corner detection techniques

B. Adaptive Non-Maximal Suppression (ANMS)

ANMS is used so that the corners that will be used as features must be the best corner in the area so, the purpose of this step is to have the best features and make sure the corners are equally distributed in the image as in reality not every corner will be perfectly sharp so with the help of ANMS we want to choose best corners N_{best} which are truly local Maxima in the region.

Here (Fig. 3) Corner score i.e. (C_{img}) is obtained using Harris corner detection technique.

The outputs shown in Fig.4 shows the comparison of the corners before and after ANMS.

C. Feature Descriptor

We defined feature descriptor around each ANMS corners. In order to do that we extracted a 40×40 area from the grayscale picture keeping the detected ANMS corner as center.

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);

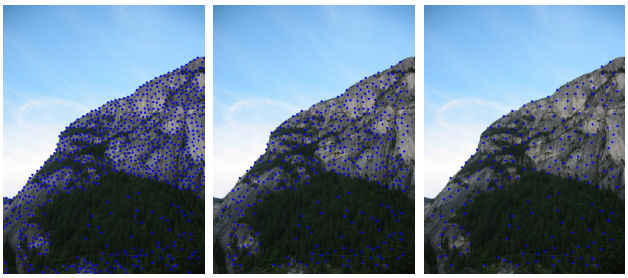
Initialize $r_i = \infty$ for $i = [1 : N_{strong}]$

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

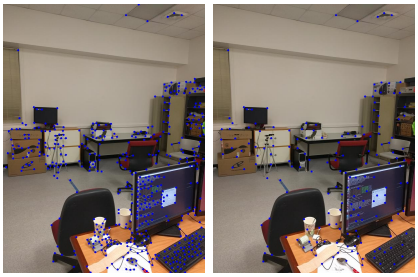
Sort r_i in descending order and pick top N_{best} points

Fig. 3: ANMS algorithm Pseudo Code

(a) Set 1: All corners (616), ANMS outputs of 400 best corners and ANMS outputs of 200 best corners



(b) Set 2: All corners (774), ANMS outputs of 400 best corners and ANMS outputs of 200 best corners



(c) Set 3: All corners (489), ANMS outputs of 180 best corners

Fig. 4: Adaptive Non-Maximal Suppression (ANMS) Output

Applied Gaussian blur to reduce the noise and resized the 40X40 patch to a 8×8. Then we obtain a feature vector of a point which is a reshaped 64×1 vector. Than just standardize the vector in order to remove any biases and make the vector invariant of varying illumination and brightness. We now defined the local information around each corner in form of a vector. This is like encoding the information of each point in the image as a vector.

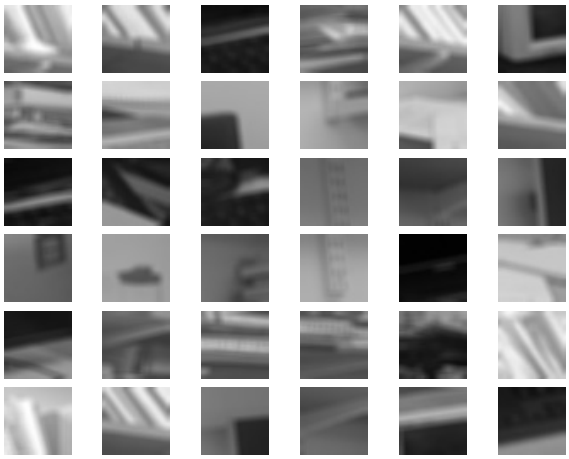


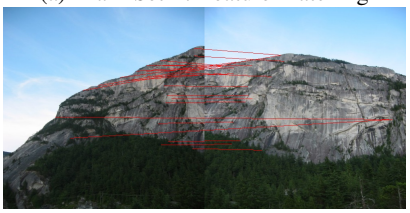
Fig. 5: Feature Descriptor Patches of 40×40

D. Feature Matching

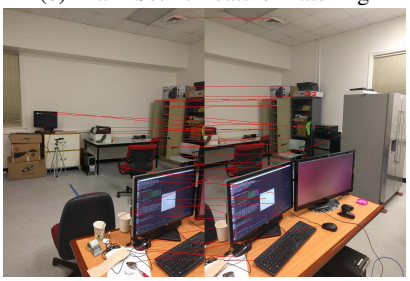
Now we have N_{best} - 64×1 vector that are defined for one image. Now we have to do this for other image as well. Now we have to compare these two sets of feature vectors from these two images and match them in the two set of images. To find these feature correspondences we used the method of calculating the distance between two vectors i.e. calculating the norm between them, we did it with the help of numpy.linalg.norm() function. To find these correspondences we calculated the correspondences , with each point in image 1 we calculate its feature vector's distance with all the feature vectors of the image 2. Then we calculate the ratio of the distance of best match (i.e. lowest distance) with the second best match (i.e. the second lowest distance) and put a threshold to this distance. This ratio is calculated in order to ensure that the match we are obtaining after this process is not only the best match but also the by far the most appropriate match. For ex.

The following is the output of feature matching done as explained in this step.

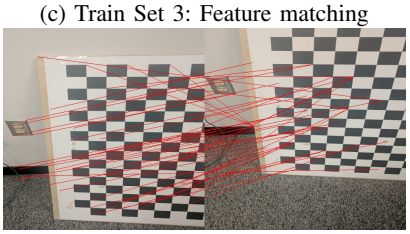
(a) Train Set 1: Feature matching



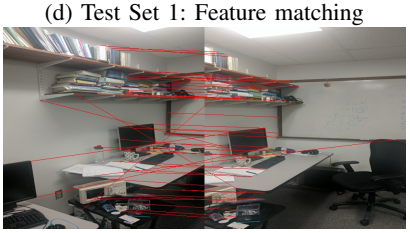
(b) Train Set 2: Feature matching



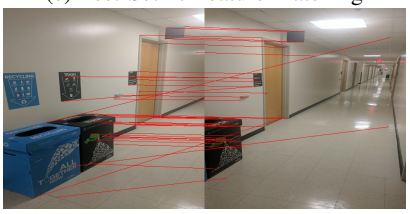
(c) Train Set 3: Feature matching



(d) Test Set 1: Feature matching



(e) Test Set 2: Feature matching



(f) Test Set 3: Feature matching

Fig. 6: Output of Feature Matching

This result is pretty decent, as we can see there are many correct matches. But we also see there are many incorrect matches which are generally outliers in our case. So we used Random sample consensus (RANSAC) as a technique for outlier rejection.

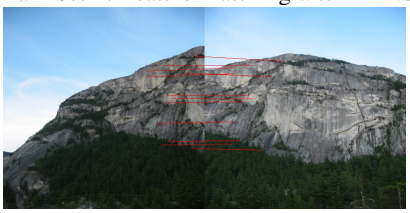
E. RANSAC for outlier rejection and Estimate Homography

As we saw in the results of previus step, not all feature match is correct. Now we will try to remove those outliers using RANSAC technique to compute a Robost Homography.

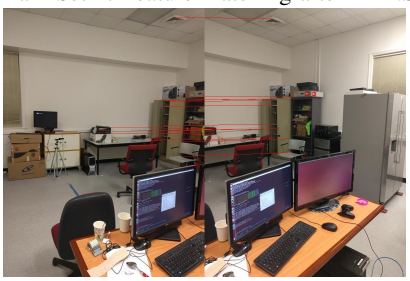
RANSAC steps are as follows:-

1) Select random four points from the ANMS corners and their corresponding feature matched points in the other image, lets say (p_i from image 1 and p_i' from image 2.
2) Compute Homography using these four points. (Note : if you get an error in computing Homography or it returns as Object Array, it maybe because there are 3 or more than 3 points lie on the same line. In that case select those points again)
3) Now we calculate the inliers in such a way that based on this Homography every points in image 1 must project to the image 2 using this Homography matrix. so we assign a threshold τ to the distance within which it will lie in the other image i.e. $SSD(p_i', Hp_i) < \tau$. Here SSD is thesum of squared difference.
4) Keep doing this untill you have done max number of iterations (N_{max}).
5) So we keep the largest set of inliers, Recompute the Homography ($\hat{H}$) again using all the inliers. (Fig. 7)

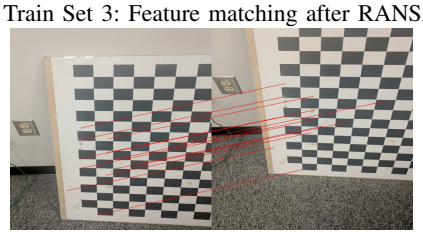
(a) Train Set 1: Feature matching after RANSAC



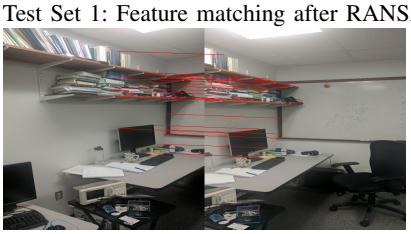
(b) Train Set 2: Feature matching after RANSAC



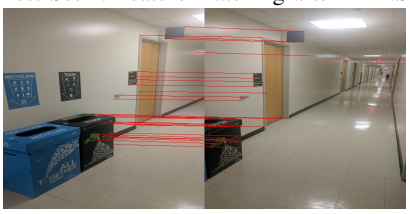
(c) Train Set 3: Feature matching after RANSAC



(d) Test Set 1: Feature matching after RANSAC



(e) Test Set 2: Feature matching after RANSAC



(f) Test Set 3: Feature matching after RANSAC

Fig. 7: Final Output of Feature matching after RANSAC

F. Stitching and Blending

For Stitching we warped all the images on the first image on which we extracted corners from,and simply compared the values of two images as other than the warped region all of the image canvas will be black.

G. Overview of the Pipeline and the Results

1) Pipeline Overview: This is the overview of the implemented pipeline in the code.

1) Resize all images to a fixed size So that during processing, there is no mismatch of size between images.
2) Corner detection and ANMS to extract good widespread corners from the image.
3) Define feature vectors of the image corners and match with the feature vectors of the other image
4) Perform outlier rejection and estimate the robust homography considering all the inliers.
5) Project every image on one reference image using warp perspective. We had a translation vector that translates all the images as there might be a scenario where the subsequent images are projected either the left or above the first image plane. Then perform stitching on these images and repeat this for all the subsequent images.

2) Results: The outputs for the Train set images Fig. 8 .

Fig. 8: Stitched Panorama for Set 1

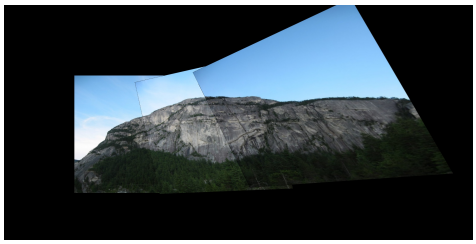


Fig. 9: Stitched Panorama for Set 2

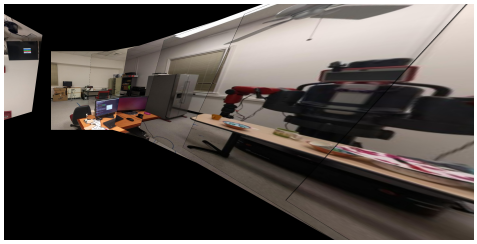


Fig. 10: Stitched Panorama for Set 3

The outputs for the Test set images

Fig. 11: Stitched Panorama for Set 1

Fig. 12: Stitched Panorama for Set 2

If we manually re-order images of Test Set 2 we obtain the following result.

Fig. 13: Stitched Panorama for Set 2

Fig. 14: Stitched Panorama for Set 3

Fig. 15: Stitched Panorama for Set 4

H. Phase 1: Conclusion and Future Steps

There is one limitation that we observed in our algorithm, our algorithm works with finding the feature correspondences in the consecutive images and warp the image on the first image plane of the directory. So if the images in the directory are not in order (like in given Test Case2) our algorithm will stop there itself and will generate images untill the correspondences it was able to find more than 4 inliers. Because of this very same reason, the algorithm failed in Test Set4.

To correct this we will have to make some changes to our algorithm. To generate the panoramas of a set of images (size n), we will generate a graph $G = (V, E)$ by checking each pair of the images, where V is the set of image and E is the set of the image pairs that have common area (i.e., enough inlier matches). Then we can lable the connected-component by Connected-component labeling algorithm, and get the largetest connected-component. After that, we can choose a base image in the largetest connected-component, and warp other images in this connected-component to the base space according to the path to the base image (multiply the homographies together), and throw away other images outside of this connected-component. In this way, we should be able to generate the panoramas without outliers images.

Another future step for this algorithm could be the Poison Blending [1], so that one image stitches with another image, there isn't any visual boundaries between them.

II. Phase 2: Deep Learning Approach

In this section, we will introduce how we generate the data, how we implement a supervised approach and a unsupervised approach, and show the results of them.

A. Data Generation

We use the MSCOCO dataset [2] to generate our data. We generate 8 pairs of related patches in each image, with the following steps:

- Randomly choose a 128×128 patch P_A near the center area of the image.
- Perform a random pertubation in the range $[-32, 32]$ to the corner points in the previous step
- Calculate H and warp the original image, then get a warped patch P_B.

There are 5000 images in the training set, so we can generate 40000 pairs of patches as the training data. A sample pair of patches is shown in Fig..

Fig. 16: Patch pair generation.

B. Supervised Approach

The supervised network is designed to take a pair of patches $[P_A, P_B]$ as input ($128 \times 128 \times 2$), and output the offset of 4 corners H_{4pt} (1×8). Here we use nearly the same network as the original paper [3] (shown in Fig.17), but use Regression HomographyNet and normalize the output $\tilde{H}_{4pt}$ with maximum offset 32. The loss function is $loss = \|\tilde{H}_{4pt} - H_{4pt}\|_2$. The optimizer we used is Adam, and the learning rate is 10^{-4}. Our architecture is shown in Fig.18.

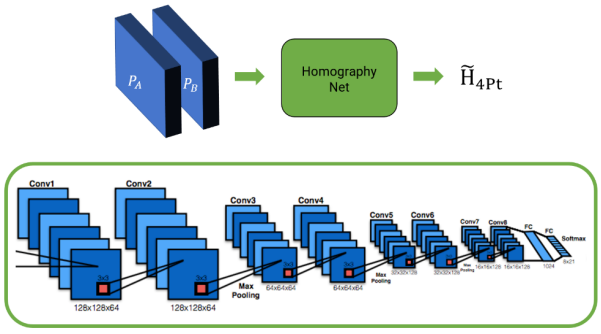


Fig. 17: Architecture of supervised model in the original paper.

C. Unsupervised Approach

The unsupervised network is designed to take a pair of patches $[P_A, P_B]$ as input ($128 \times 128 \times 2$), and output the homography matrix H directly. We concact the supervised network with a TensorDLT layer and a Spatial Transformer layer, as introduced in the original paper [4]. The loss function is the $L1$ norm of the original patch P_B and the warped patch $w(P_A, H)$, which is $l = \|w(P_A, H) - P_B\|_1$. The overall process is shown in Fig.. The architecture is shown in Fig.20.

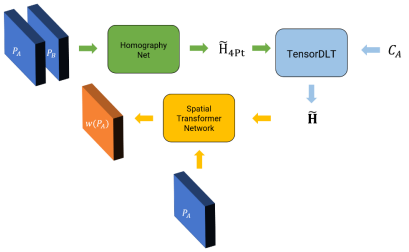


Fig. 19: Overall process of unsupervised model.

D. Panoramas of learning approach

Since the learning based approaches we implemented need input images with fixed size, they are not that flexible as the classical methods. We simply resize the two images to the fixed size of the network to get the homography of two images, and then generate the panoramas as we introduced in phase 1. However, the results is bad compare with the classical feature based method.

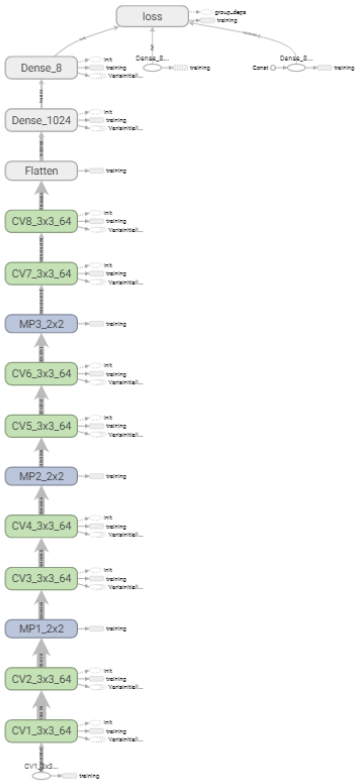


Fig. 18: Architecture of supervised model.

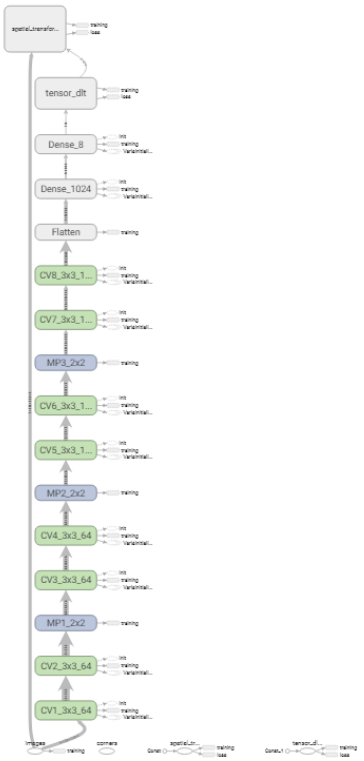


Fig. 20: Architecture of unsupervised model.

E. Experiment Results

In this section we will show our experiment results from different perspectives.

1) Visualization: We visualize the classical feature based, supervised, unsupervised estimated homographies against a synthetic ground truth for 1 image from Train set, 1 image from Val set, and 2 images from Test set. The results of classical feature based approach is shown in Fig.23. The results of supervised approach is shown in Fig.24. The results of unsupervised approach is shown in Fig.25.

2) Quantitative result: We use average EPE as evaluating metrics, which is defined as $average\|\tilde{H}_{4pt} - H_{4pt}\|_2$ for all the 4 points and all the images. The results are shown in Tab.. The EPE of supervised approach on Train set is much smaller than that of unsupervised approach, which is reasonable because there are direct supervised information in during the supervised training process. But we only train 100 epoches in supervised approach, while 200 epoches in unsupervised approach, so the EPE's of supervised approach on Val set and Test set is larger than those of unsupervised approach.

TABLE I: Average EPE

Approach	Train set	Val set	Test set
Supervised	1.03	12.67	12.62
Unsupervised	4.82	9.67	9.62

We also show the training loss and validation loss during the training process of supervised and unsupervised approach in Fig.21 and Fig.22, respectively. In the unsupervised approach, we only monitor the H_{4pt} loss without backpropagation. As we can see from the figures, the supervised model will converge faster than the unsupervised model, and is more accurate when train the same epochs. The training loss of H_{4pt} in unsupervised model decreases as the epoch number increases, which means the unsupervised model is actually learning since the H_{4pt} loss is just under monitoring without backpropagation. The validation losses of both model increase slowly after several epochs, which means the model may overfit the training data. A possible solution is augmenting the training data.

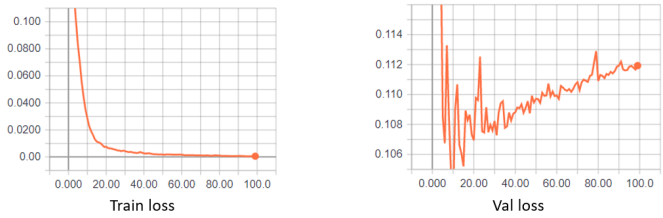


Fig. 21: Loss of supervised approach.

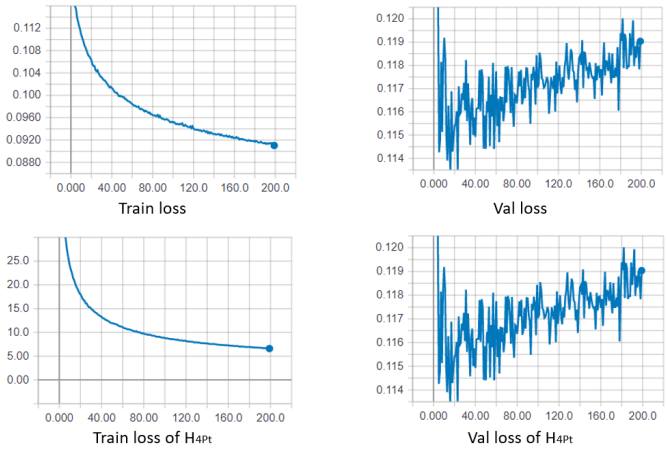


Fig. 22: Loss of unsupervised approach. We only monitor the H_{4pt} loss (no backpropagation).

F. Phase 2: Conclusion

The supervised model can learn faster than the unsupervised model, and is more accurate with the same training epochs. In patch level, the supervised model and unsupervised model can get nearly same performance as the classical feature based method, but because of the fixed size of input images, they can not get the similar performance as classical feature based method in image level easily. Simply resizing the input image doesn't make the performance so good, and the error will be enlarged after resizing back. DSAC will be a good method to try.

Fig. 23: Results of classical approach on Train/Val/Test/Test images. The blue rectangle is P_A, the green polygon is the ground truth warped patch $w(P_B)$, and the red polygon is our results.

Fig. 24: Results of supervised approach on Train/Val/Test/Test images. The blue rectangle is P_A, the green polygon is the ground truth warped patch $w(P_B)$, and the red polygon is our results.

Fig. 25: Results of unsupervised approach on Train/Val/Test/Test images. The blue rectangle is P_A, the green polygon is the ground truth warped $w(P_B)$, and the red polygon is our results.

REFERENCES

[1] Patrick Perez, Michel Gangnet† and Andrew Blake, *Poisson Image Editing*,Microsoft Research UK

[2] Tsung-Yi Lin, Genevieve Patterson, Matteo R. Ronchi, *COCO-Common Object in Context*,http://cocodataset.org/home

[3] Daniel DeTone, Tomasz Malisiewicz, Andrew Rabinovich, *Deep Image Homography Estimation*

[4] Ty Nguyen, Steven W. Chen, Shreyas S *Unsupervised Deep Homography: A Fast and Robust Homography Estimation Model*