Computer Vision Project 1 - MyAutoPano

Kaushik Kavuri Subrahmanya
Robotics Engineering
Worcester Polytechnic Institute
Worcester, Massachusetts 01609
Email: ksubrahmanya@wpi.edu
Using 2 Late Days

Butchi Adari Venkatesh
Robotics Engineering
Worcester Polytechnic Institute
Worcester, Massachusetts 01609
Email: badari@wpi.edu
Using 2 Late Days

***Abstract*—In this project, we aim to stitch two or more images with repeated local features to create a single seamless panorama image. We approach this problem from two directions: traditional approach, and deep learning methods. In deep learning methods, we implement both supervised and unsupervised methods to stitch our panorama images.**

I. Phase 1 - Traditional Approach

In this approach, we first detect 'corners' in an image. We then find strong corners distributed uniformly throughout the image using ANMS (Adaptive Non-Maximal Suppression)algorithm. These corners are then reduced to vectors known as feature descriptors. By comparing these feature descriptors for two images, we find matching features for both images. As some of these matches may not be correct, we eliminate false matches and compute homography between the two images using RANSAC algorithm. Using this homography, we then blend the images to obtain our final panorama. An overview of the approach is provided in fig.1. Each step is discussed as follows.

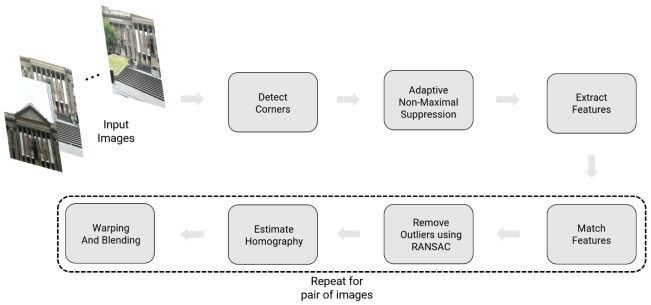

Fig. 1: Traditional Approach Overview

A. Corner Detection

we use the Harris corners method to detect corners in each image. Harris Corner outputs an image with a 'score' for each corner: the brighter the image, the stronger the corner. A sample output is shown in Fig. 2

B. Adaptive Non-Maximal Suppression

We want to reduce weak corners and if our corners are not uniformly spread out throughout the image, it could result in weird artifacts during warping. To address this,

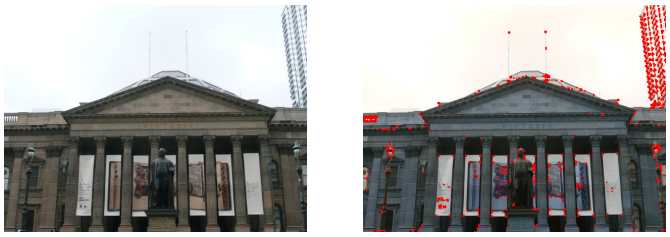

Fig. 2: Input and output figures of Corner Detection

we use ANMS algorithm. As the name suggests, ANMS suppresses corners that are close to each other and only takes the strongest corners. These strong corners are distributed uniformly throughout the image. A sample output is shown in Fig. 3

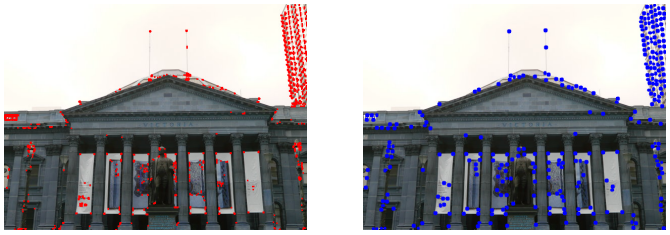

Fig. 3: Input and output figures of ANMS

C. Feature Descriptor

Each corner represents a unique feature in that image. We vectorize these corners to help perform calculations on them. We take a 41*41 pixel patch around the corner and apply a Gaussian Blur on this patch to denoise the patch. This blurred patch is then subsampled to a size of 8*8 and finally reshaped to 64*1.To remove bias and achieve some illumination invariance, we standardize the vector to have zero mean and variance of 1. Feature Descriptors of sample input are shown in Fig. 4

D. Feature Matching

To find matching features between two images, we take the feature descriptors of each corner point and calculate the sum of square differences (SSD) to all corner points in the second image. We then only keep those associations where the ratio of

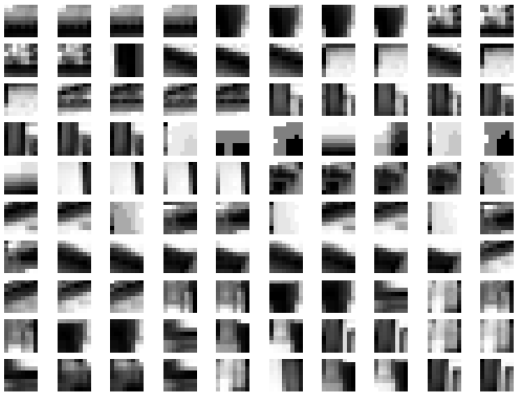

Fig. 4: Feature Descriptors of the sample image

lowest match to the second lowest match is below a threshold ratio. A sample output of this step is shown in Fig. 5 where the matches are visualized using the cv2.drawMatches function.

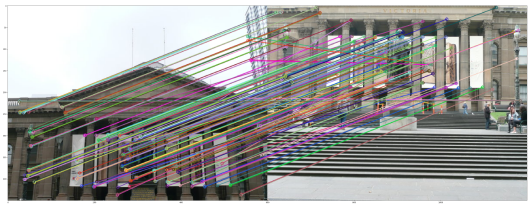

Fig. 5: Feature Matching Output

E. RANSAC

Even though we matched features between the given images, the match might not be correct. We use Random Sample Consensus (RANSAC) algorithm to reduce these incorrect matches and to compute Homography between the images. In this step, we first select 4 random feature pairs from both images and compute their homography. Then we apply this homography on the first feature pair. If the sum of square difference between the feature pair of second image and the feature pair of the first pair (after applying the previously computed homography) is below a threshold, the pair is saved. The largest set of such inliers is then selected and the homography between the two images is then calculated.

F. Blending

The images are warped using the homography calculated previously and then stitched together. This stitched image and the next image in the set of images are then sent as a pair to be stitched. If the RANSAC output of any two images is 0, or very low, the image is dropped and the function passes the previous stitched image to the next image to try to stitch them. The outputs are as shown in fig. 7 - 13.

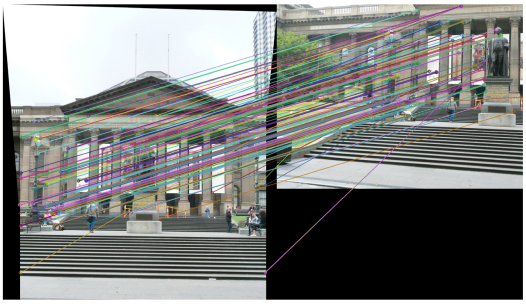

Fig. 6: Feature Matching Output of the first stitched pair and the final image

Fig. 7: Final Output of Set1 Images

G. Final Outputs

Below are the results of the panorama stitcher on train sets and test sets.

II. Phase 2

In the second phase of the homework, we approach this problem from a different approach. Here, we train a deep learning model to estimate the homography between two images. This way, we can combine corner detection, ANMS, feature extraction, feature matching, RANSAC and estimate homography all into one.

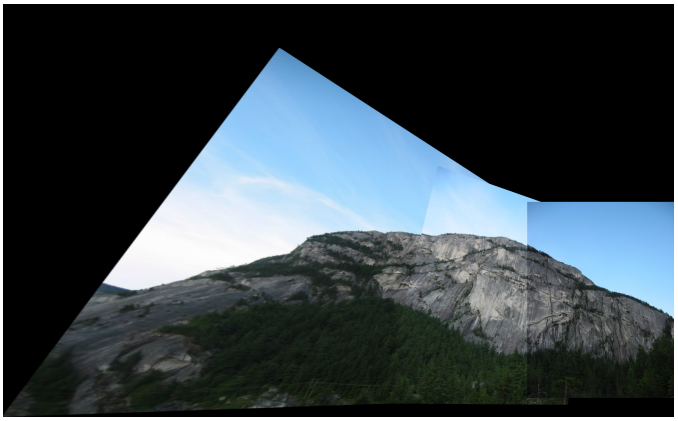

Fig. 8: Final Output of Set2 Images

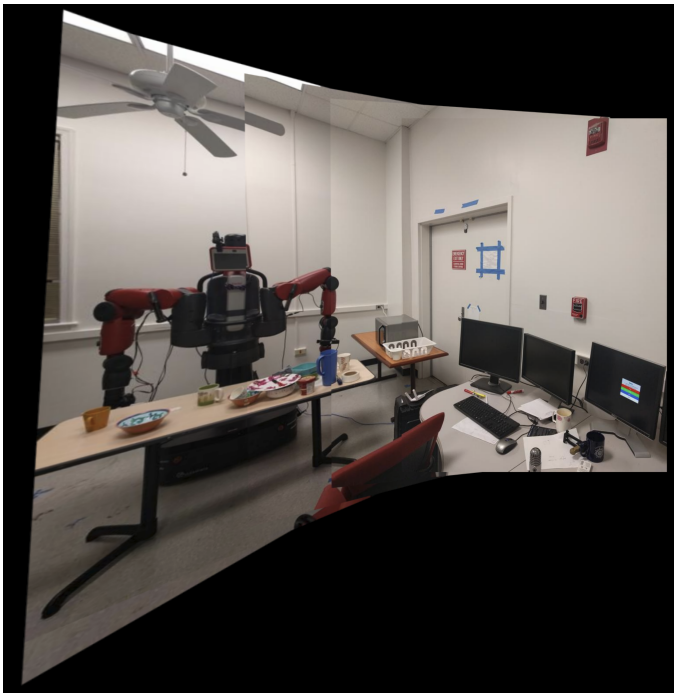

Fig. 9: Final Output of Set3 Images

A. Data Generation

To train a deep learning network model to estimate homography between two images, we need a pair of images with known homography between them. To generate these datasets, we use MSCOCO dataset. We take a patch of size 128*128 from an image in the dataset and perturb the 4 corners of this patch by a random amount from [-32,32]. We then calculate the homography between the original corners and the perturbed corners of the patch. After this, we warp the image by the inverse of the homography calculated earlier. Then we extract the warped patch from the warped image using the coordinates of the original patch. We now have both patches and the homography between them. A sample is shown in Fig. 14.

We then train the Homography Net to estimate the perturbation

Fig. 10: Final Output of Test Set1 Images

of each corner. This estimate is fed into a Direct Linear Transformer to get the final 3*3 homography matrix. Using this homography, we warp and blend images as mentioned earlier in Phase 1. An overview of this process is shown in Fig. 15.

B. Supervised Approach

In this approach, we input the image patch, the warped image patch, and the calculated homography between them. The model is trained to estimate the perturbation of each corner (H4pt). The loss function for this approach is given by the L2 loss between the predicted H4pt and the actual H4pt. The network architecture is given below in Fig. 16

The loss per Epoch of the model is shown below in Fig.17 The actual perturbed patch and the estimated perturbation predicted by this model are as shown in Fig. 18

Network parameters of supervised network are as follows in Fig.19

Loss values are in table 1.

C. Unsupervised Approach

In the unsupervised approach, the homography net used in supervised approach to estimate the h4pt. This is then input to a TensorDLT to obtain the 3*3 homography matrix. This matrix, and the unwarped patch is input to a Spatial Transform

Fig. 11: Final Output of Test Set2 Images

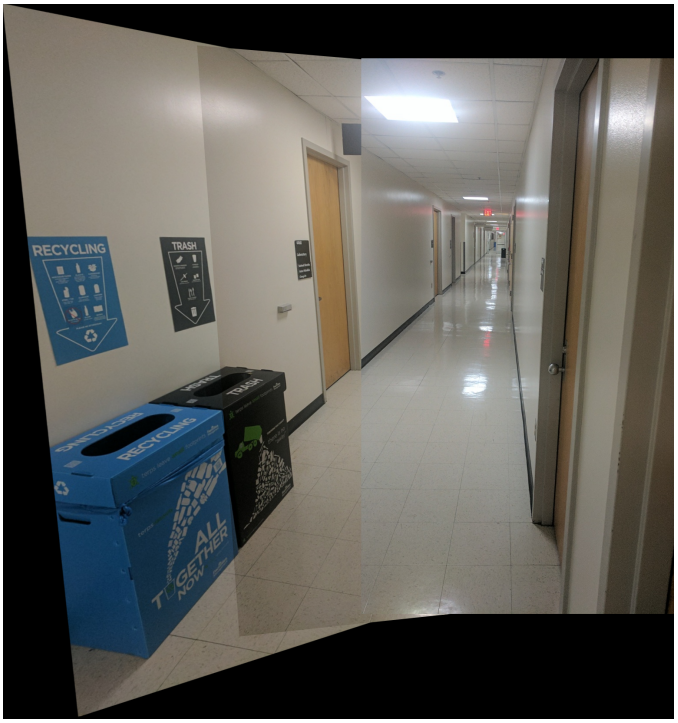

Fig. 12: Final Output of Test Set3 Images

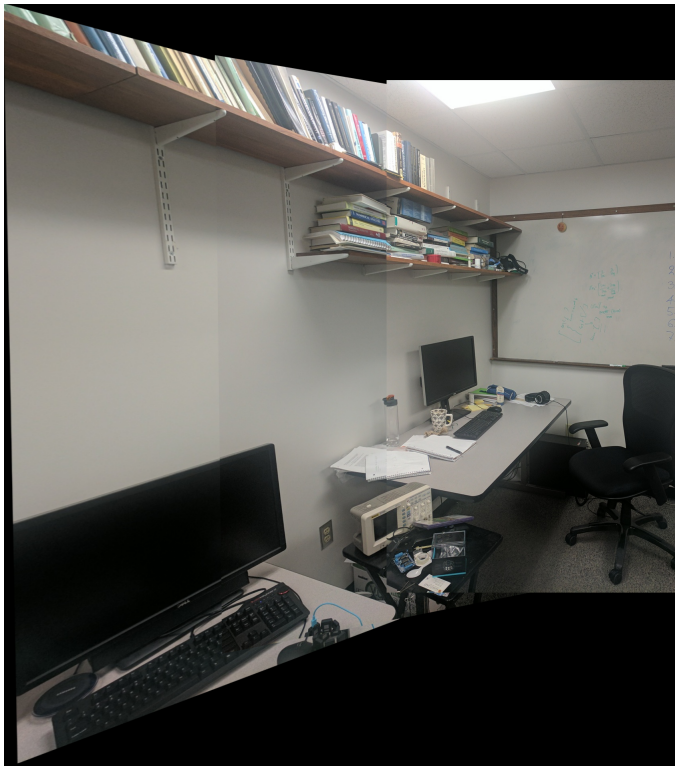

Fig. 13: Final Output of Test Set4 Images

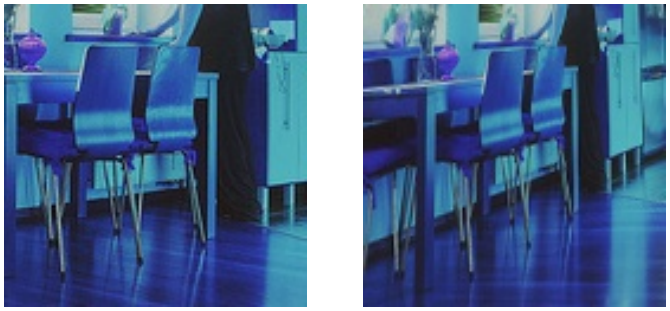

Fig. 14: Original Patch and Warped patch

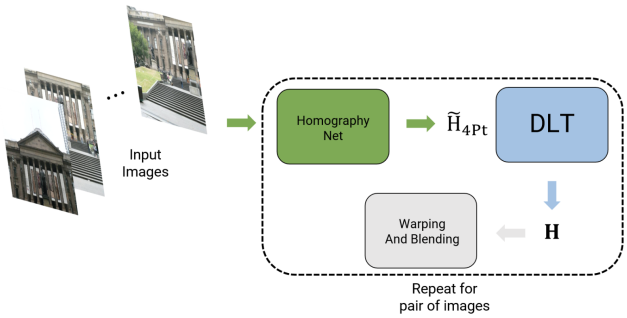

Fig. 15: Deep learning stitching overview

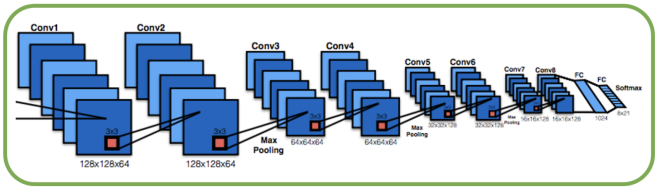

Fig. 16: Supervised Approach Network Model

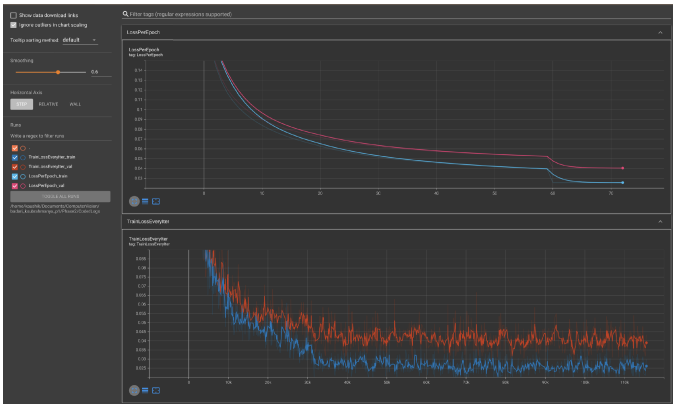

Fig. 17: Supervised Approach Loss per Epoch

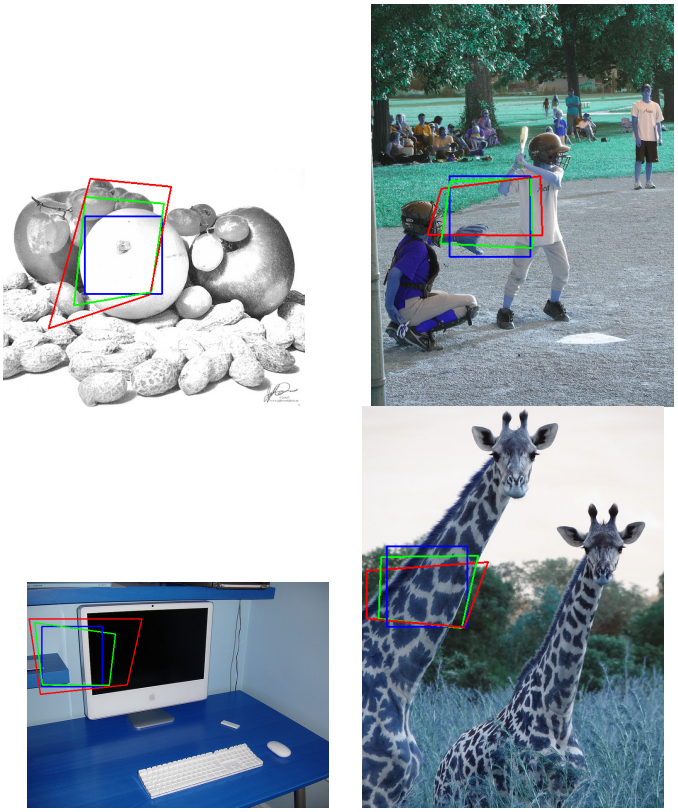

Fig. 18: Red Square - Predicted Perturbation; Green Square - Actual Perturbation; Blue Square - Original Patch

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1         [-1, 64, 128, 128]           1,216
       BatchNorm2d-2         [-1, 64, 128, 128]             128
              ReLU-3         [-1, 64, 128, 128]               0
            Conv2d-4         [-1, 64, 128, 128]          36,928
       BatchNorm2d-5         [-1, 64, 128, 128]             128
              ReLU-6         [-1, 64, 128, 128]               0
         MaxPool2d-7           [-1, 64, 64, 64]               0
            Conv2d-8           [-1, 64, 64, 64]          36,928
       BatchNorm2d-9           [-1, 64, 64, 64]             128
             ReLU-10           [-1, 64, 64, 64]               0
           Conv2d-11           [-1, 64, 64, 64]          36,928
      BatchNorm2d-12           [-1, 64, 64, 64]             128
             ReLU-13           [-1, 64, 64, 64]               0
        MaxPool2d-14           [-1, 64, 32, 32]               0
           Conv2d-15          [-1, 128, 32, 32]          73,856
      BatchNorm2d-16          [-1, 128, 32, 32]             256
             ReLU-17          [-1, 128, 32, 32]               0
           Conv2d-18          [-1, 128, 32, 32]         147,584
      BatchNorm2d-19          [-1, 128, 32, 32]             256
             ReLU-20          [-1, 128, 32, 32]               0
        MaxPool2d-21          [-1, 128, 16, 16]               0
           Conv2d-22          [-1, 128, 16, 16]         147,584
      BatchNorm2d-23          [-1, 128, 16, 16]             256
             ReLU-24          [-1, 128, 16, 16]               0
           Conv2d-25          [-1, 128, 16, 16]         147,584
      BatchNorm2d-26          [-1, 128, 16, 16]             256
             ReLU-27          [-1, 128, 16, 16]               0
          Dropout-28          [-1, 128, 16, 16]               0
          Flatten-29               [-1, 32768]               0
           Linear-30                [-1, 1024]      33,555,456
             ReLU-31                [-1, 1024]               0
           Linear-32                   [-1, 8]           8,200
================================================================
Total params: 34,193,800
Trainable params: 34,193,800
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.12
Forward/backward pass size (MB): 70.77
Params size (MB): 130.44
Estimated Total Size (MB): 201.33
----------------------------------------------------------------
```

Fig. 19: Supervised Network Parameters

Performance	Train	Val	Test
Supervised	34.4	33.6	33.3

TABLE I: Size of the Models

Network to warp the image(estimated warped image). The loss function is then the l1 difference between the estimated warped image and the actual warped image. An overview of this process is shown below in Fig. 20

We tried approaching the template matching but the results were not good.

III. Conclusion

Here, we tried two approaches to stitch images. The first is the classical approach, and the second is the deep learning approach. The outcomes of this exercise are as shown above.

Acknowledgment

The authors would like to thank the professor for the instructions and knowledge imparted via the course.

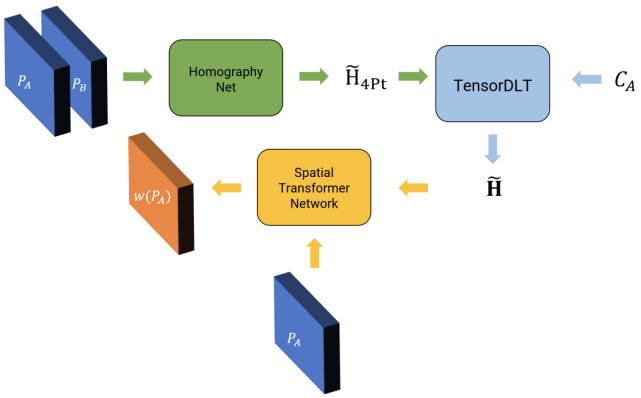

Fig. 20: Unsupervised Approach Network Model