CMSC733 Project 1: MyAutoPano

Abhiskek Kathpal
M.Eng., Robotics
University of Maryland, College Park
Email: akathpal@terpmail.umd.edu
USING 2 LATE DAYS

Jack Rasiel
School of Computer Science
University of Maryland, College Park
Email: jrasiel@cs.umd.edu
USING 2 LATE DAYS

***Abstract*—The task of this project is to stitch two or more images and create a seamless panorama. Implementation is done by two approaches, first approach is using traditional Computer vision techniques and second approach is to use deep learning techniques. For Phase 1 ,task is feature matching and finding robust Homography. For Phase 2, two networks are implemented, supervised and unsupervised for computing robust Homography. The results obtained from these are analysed and compared with the traditional Approach.**

I. Phase 1 - Conventional Panorama Stitching

A. Traditional Approach Overview

The Conventional Panorama Stitching Algorithm can be implemented using the following steps:
1. Corner Detection using Harris or Shi-Tomasi Corners.
2. Implementing Adaptive Non-Maximal Suppression Algorithm to avoid wierd artifacts in warping.
3. Computing Feature Descriptors.
4. Feature Matching using SSD.
5. Outlier Rejection Using RANSAC to estimate Robust Homography.
6. Stitching/Blending various images together.

The pipeline is given by Fig. 1. and is discussed in detail in the next subsections.

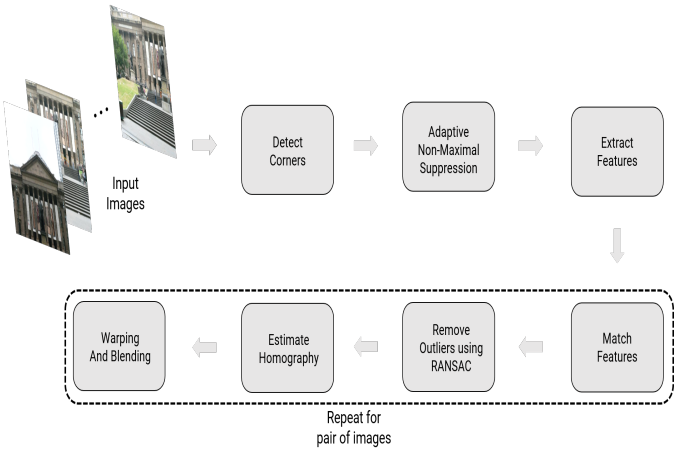


Fig. 1. Panorama Stitching Algorithm pipeline

B. Corner Detection

For the corner detection, 3 techniques have been used, namely, Harris corner detector, Shi-Tomasi Corner Detector and OpenCV's Good Features To Track have been used for this part. The main difference between Harris Corner Detector and Shi-Tomasi Features is the response map equation. Shi-Tomasi computes response map using minimum eigen values.
The third implementation is using goodfeatures to track which has inbuilt non maximal suppression. For other two techniques, we have to implement adaptive non-maximal suppression which is described in next section.

I have found that good Features to track works the best among these. Harris and Shi-Tomasi corners give quite similar results. I have included the output from all the implementations in the figure below:

Fig. 2. Harris Corners

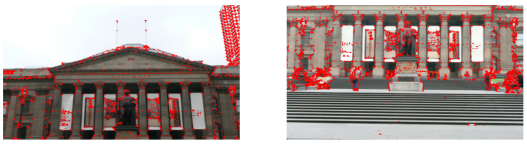


Fig. 3. Shi-Tomasi Corners

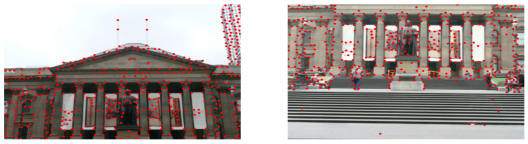


Fig. 4. GoodFeaturestoTrack

C. ANMS Implementation

Adaptive Non-Maximal Suppression algorithm is used to find N strongest corners. For my code, best 100 corners have been selected. The pseudo code for this is shown below:

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**
      $\text{ED} = (x_j - x_i)^2 + (y_j - y_i)^2$
    end
    if $ED < r_i$ **then**
      $r_i = \text{ED}$
    end
  end
end
Sort r_i in descending order and pick top N_{best} points

Fig. 5. Adaptive Non-Maximal Suppression

The output after implementing this algorithm after corner detection is shown below:

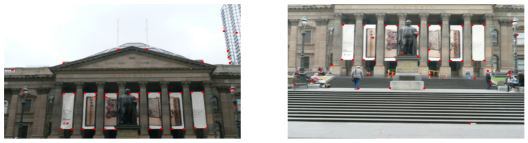


Fig. 6. Harris after ANMS

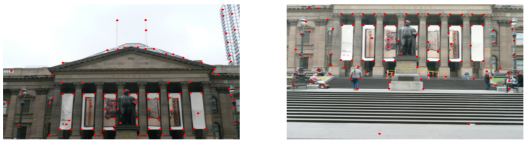


Fig. 7. Shi-Tomasi after ANMS

D. Feature Descriptors

Next step is to describe all the corner points by a feature vector. For this, a patch of 40x40 is taken around that point and is sub-sampled to 8x8 and blurred using Gaussian Bluring. The output is then reshaped to 64x1 vector and standardization is done to get mean 0 and variance 1. The output of one of patch is shown below:

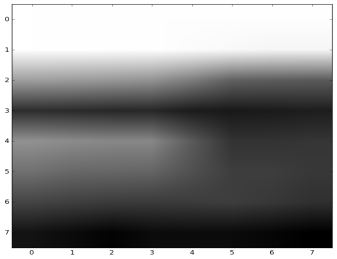


Fig. 8. 8x8 Feature Descriptor Patch1

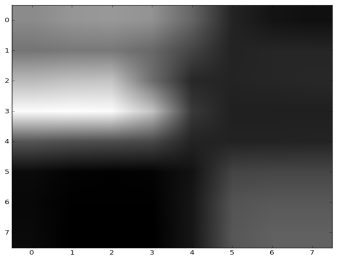


Fig. 9. 8x8 Feature Descriptor Patch2

E. Feature Matching

Feature Matching is done by picking one points from first feature vector and computing the SSD with all the other feature descriptors. Distance Threshold is used to find the matched pairs in the images. Draw matches inbuilt function of OpenCV is used to display the output of feature matching.

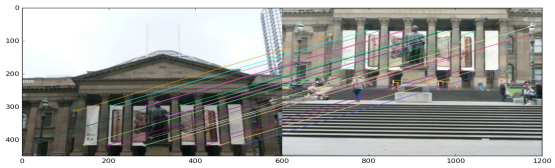


Fig. 10. Shi Tomasi Initial Match

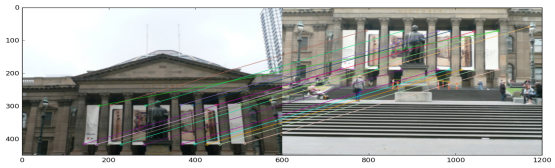


Fig. 11. Harris Initial Match

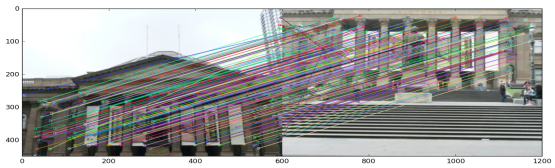


Fig. 12. Good Features to track Initial Match

F. RANSAC

The next step is to use Random Sample Consensus to remove the outliers and find the robust Homography estimate. The algorithm has following steps:
1. Select four feature pairs (at random), from image1 and image 2.
2. Compute homography H between the previously picked point pairs.
3. Compute inliers where SSD¡, where is some user chosen threshold and SSD is sum of square difference function.
4. Repeat the previous steps until you have found more than a particular percentage of inliers.
5. Keep largest set of inliers.
6 .Re-compute least-squares H estimate on all of the inliers.

The feature matching output after ransac is shown below:

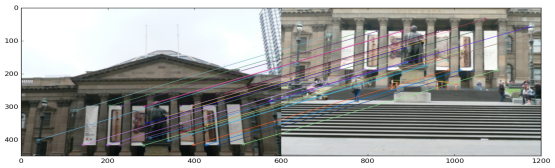


Fig. 13. Harris After RANSAC

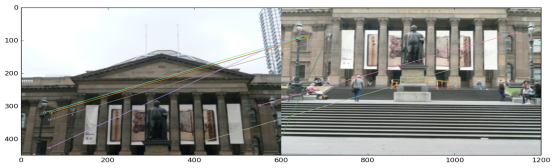


Fig. 14. Good Features to track After RANSAC

G. Image Stitching

For the Image Stitching, I have used 3rd party code which is able to stitch two images. I was not able to solve the issue of stitching if the images are not in ordered. If they are in ordered, the algorithm works perfectly. There are some memory issues if there are more than 4-5 images, the pano size got very large. The final outputs of some of data are shown below:

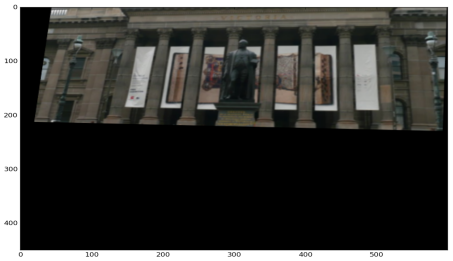


Fig. 15. Warped Output

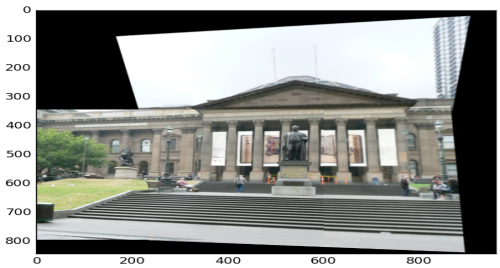


Fig. 16. Output1

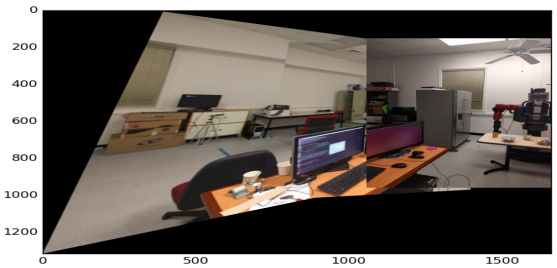


Fig. 19. Output4

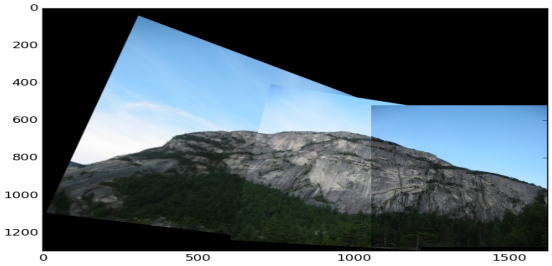


Fig. 17. Output2

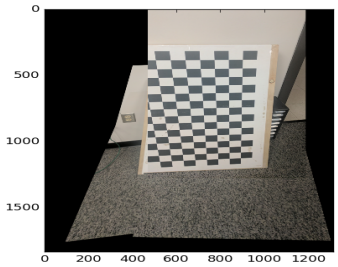
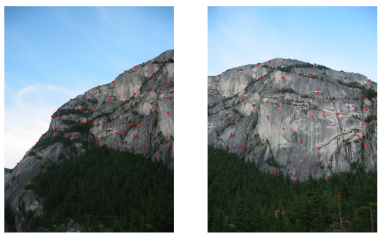
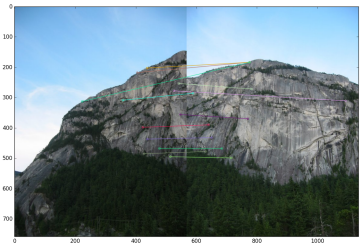


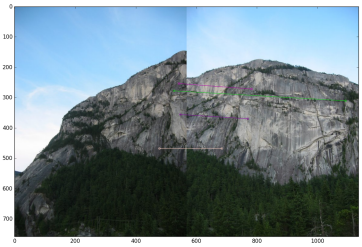
Fig. 18. Output3



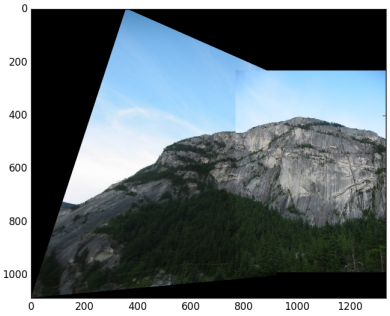
(a) Corners



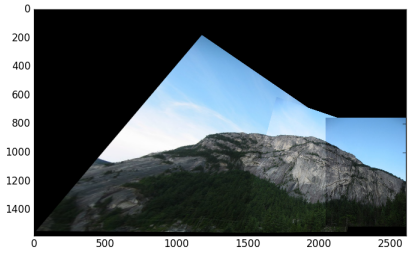
(b) Initial Feature Match



(c) After RANSAC



(d) Blending



(e) Final Result

Fig. 20. Step-by-step visualization of out conventional pipeline.

II. Phase 2, Part 1: Supervised Model

A. Data Generation

To implement the supervised Homography Network, I have generated dataset from MSCOCO Dataset. The input to this supervised network is composed of 2 images and labels are 4 point homography.
Initially, I have saved all the data in pickle file but it was taking a lot of memory and training was very slow when I used that file. Then I decided to go with patch generation on the go, so the memory used reduce to very low because only image patches equal to MiniBatchsize is stored at one time. The training time also reduced.

B. Architecture

The architecture implemented for Homography Net is described in Detone et al. It was a similar to simple VGG Network. It has 8 convolution layers with max pooling and batch normalization. In the end, there are 2 fully connected networks with dropout. The output for this network is 4 point Homography. The architecture is shown in Figure below:

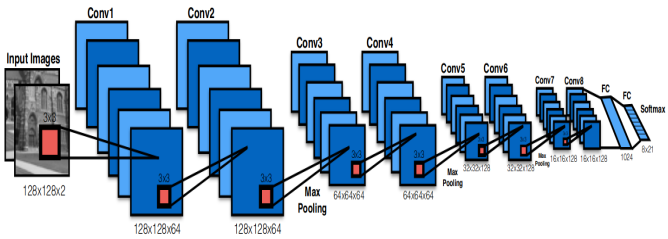


Fig. 21. Supervised Homography Network Architecture

The Tensorflow graph for this network is shown below:

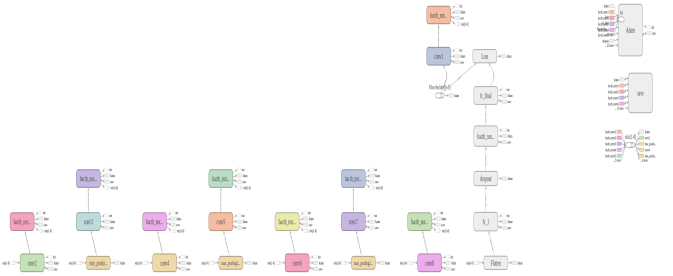


Fig. 22. Tensorflow Supervised Graph

C. Training and Performance

Loss used in this network was L2 norm i.e.

$$||H_4PtPred - H_4Pt||^2$$

During training, I initially used only 5000 images for training but the output was not that good. The loss was not decreasing properly. The output is shown below:

Fig. 23. Training Loss with 5000 images

Then I increased the Number of samples to 20000, the loss started decreasing and it was shown in Fig. below. I was able to run the code for only 50 epochs. The learning rate I have used is 1e-4 with Adam Optimizer. The final training loss is shown below:

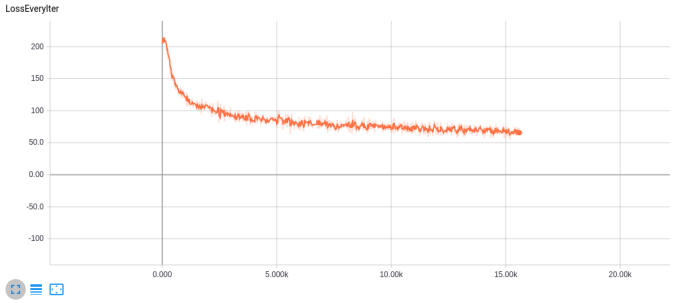


Fig. 24. Loss per iteration

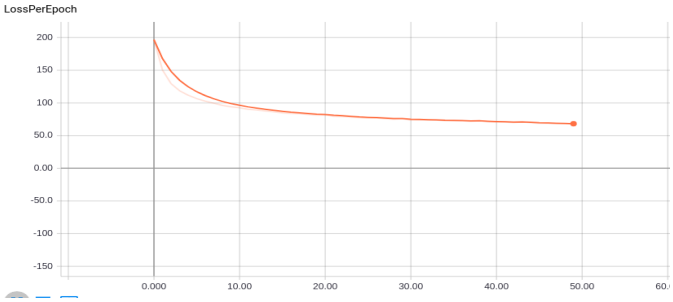


Fig. 25. Loss per Epoch

D. Results

Few outputs were generated by overlapping the new points with the target points which are computed by doing random perturbation. Green is the source. Blue is target and red is predicted using supervised trained network.

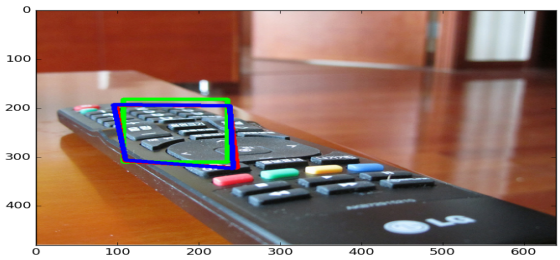


Fig. 26. Output1

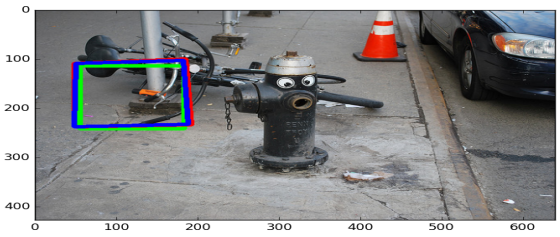


Fig. 27. Output2

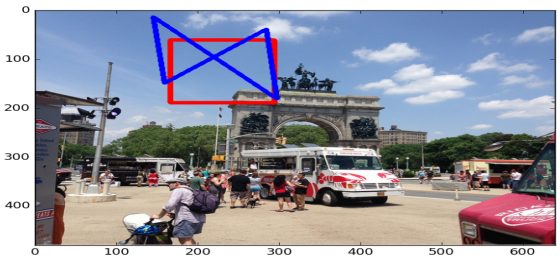


Fig. 28. Output3

III. Phase 2, Part 2: Unsupervised Model

A. Data Generation

To implement the unsupervised Homography Network, I have decided to go with patch generation on the go to keep the memory usage low. For this network , I have stored both patches concatenated along 4 corners of image1.

B. Architecture

Architecture of the unsupervised network consists of mainly three parts:
1.HomographyNet - This network is same as Supervised network and the output of this network is 4-point Homography.
2. TensorDLT - This layer of network is used to implement DLT to get the 3x3 Homography Matrix in a differentiable way. For implementing this part, I have referred official github repository for Unsupervised Network.
3. Spatial Transformer Layer - The next layer applies the 3x3 homography estimate H output by the Tensor DLT to get warped images. These warped images are necessary in computing the photometric loss function i.e. L1 loss subtracting the predicted warped image with the patch of 2nd image. The function for this layer is given in utils.

The architecture of this unsupervised network is shown below:

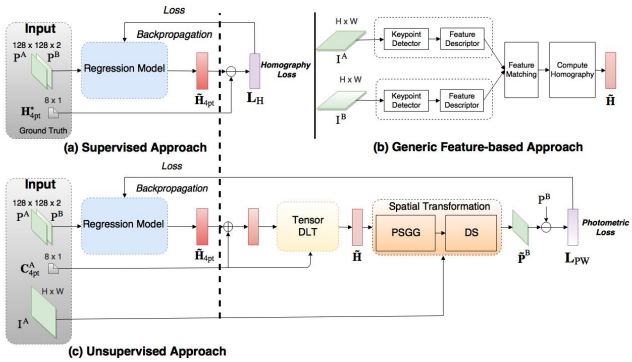


Fig. 29. Unsupervised Network Architecture

The Tensorflow graph for this Unsupervised Network is shown below:

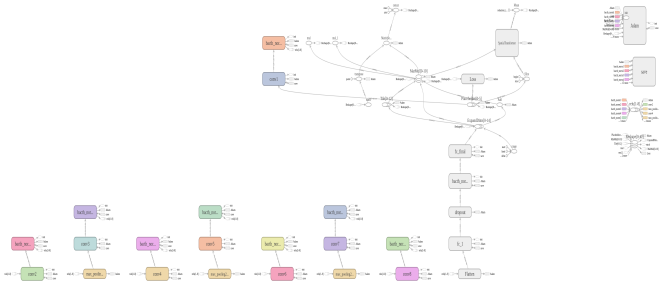


Fig. 30. Tensorflow Unsupervised Graph

C. Training and Performance

Because of these TensorDLT and Spatial Transformer Layer, to back propagate properly learning rate is kept at low i.e. 1e-5 in comparison to higher learning rate for supervised network.

The Unsupervised required less data and is more accurate than Supervised network. The training loss reached to saturation in less number of epochs. The training loss for different hyper parameters is shown below.

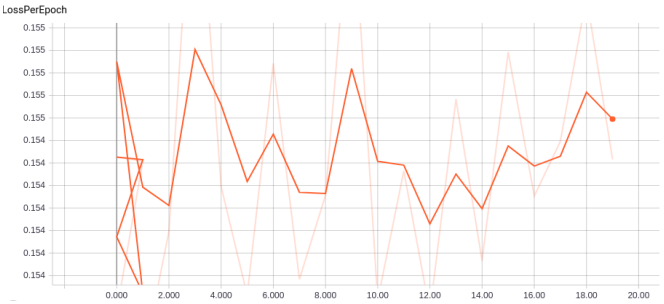


Fig. 31. Loss per epoch learning rate = 1e-3

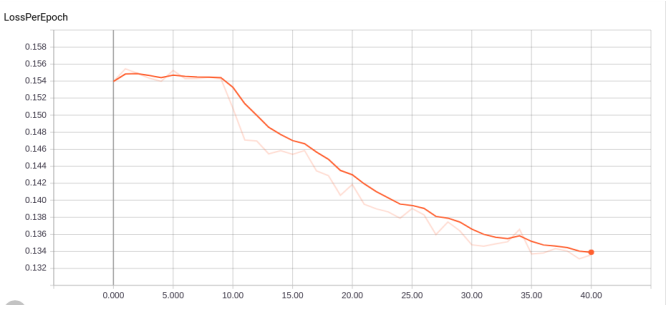


Fig. 32. Loss per Epoch learning rate = 1e-5

IV. Conclusion

Three techniques are implemented to get the robust Homography and create a panorama by warping and blending. The advantage of tradition approach is all the steps of obtaining the final output is visualized properly. But the deep learning approaches are more accurate. The traditional image approach depends on tweaking the parameters of anms and Ransac where as deep learning parameters are independent of particular image. For example, the checker board image is very difficult to stitch with traditional approach.

The deep learning approach also runs much faster when testing the data with given model in comparison to traditional approaches. Unsupervised is much more complicated than supervised but it is faster during training and gives better results.

One disadvantage of deep learning approach, it requires you to manually generate such a large dataset to work properly. With small training data, the deep learning approach will fail miserably.

Image stitching is common for both the traditional and deep learning implementations. With better image stitching algorithm, I could have improved my final outputs for this project. I was not able to generate final panorama using deep learning method, only found homography estimate.

References

[1] Implementation: https://cmsc733.github.io/2019/proj/p1/
[2] DeTone, Daniel, Tomasz Malisiewicz, and Andrew Rabinovich. "Deep image homography estimation." arXiv preprint arXiv:1606.03798 (2016).
[3] https://github.com/tynguyen/unsupervisedDeepHomographyRAL2018
[4] Nguyen, Ty, et al. "Unsupervised deep homography: A fast and robust homography estimation model." IEEE Robotics and Automation Letters 3.3 (2018): 2346-2353.
[5] WarpImage: https://stackoverflow.com/