RBE/CS549: P1 - MyAutoPano

Shrishailya Chavan
WPI Robotics Engineering
Worcester Polytechnic Institute
schavan@wpi.edu

Using 1 late Day

Sreejani Chatterjee
WPI Robotics Engineering
Worcester Polytechnic Institute
schatterjee@wpi.edu

Using 1 late Day

***Abstract*—This report presents our work on Project 1 - MyAutopano where we have worked on stitching two or more images for creating uninterrupted Panorama. Further, the report following report contains an approach of the Phase 1 and Phase 2 of the Panoramic Task. The Phase 1 is all about using traditional Computer Vision techniques and the Phase - 2 is about Deep-Learning approach. The first method uses the Homography Matrix between two images and the second method uses Supervised and Unsupervised Neural Networks to achieve the stitching.**

***Index Terms*—AutoPano, Homography, Supervised, Unsupervised, Panoramic Task, Panorama**

I. Phase 1: Traditional Approach

In this section we present the Traditional Approach to create a Panoramic Image from a given set of images which is implemented in five steps listed below,

1) Corner Detection using Shi-Tomasi Corner Detector or Harris Corner Detection
2) Adaptive Non-Maximal Suppression(ANMS)
3) Feature Detection and Matching
4) RANSAC Algorithm for Outlier Rejection and Robust Homography
5) Stitching the Results to obtain the Panorama or Blending of Images to get Panorama.
   We will be going through all the above steps mentioned step by step.

A. *Corner Detection*

The first step in stitching a panorama is extracting corners like most computer vision tasks. Here we will use either Harris corners or Shi-Tomasi corners. Further to extract the corners from the image we use (cv2.cornerHarris) or (cv2.goodFeaturesToTrack) to implement this part.

B. *Adaptive Non-Maximal Suppression (ANMS)*

The objective of this step is to detect corners such that they are equally distributed across the image in order to avoid weird artifacts in warping. In a real image, a corner is never perfectly sharp, each corner might get a lot of hits out of the N strong corners - we want to choose only the Nbest best corners after ANMS. In essence, you will get a lot more corners than you should! ANMS will try to find corners which are true local maxima. The algorithm for implementing ANMS is given below. Furthermore, when we apply Shi-Tomasi method (cv2.goodFeaturesToTrack) it automatically comes up with Non-maximal suppression, for this you can skip ANMS if your local kernel size is going to be 3x3 as there would not be much of difference.

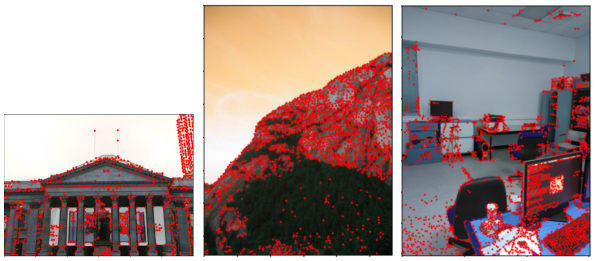

Fig. 1. Corner Detection for Train Set

Fig. 2. Corner Detection for Test Set

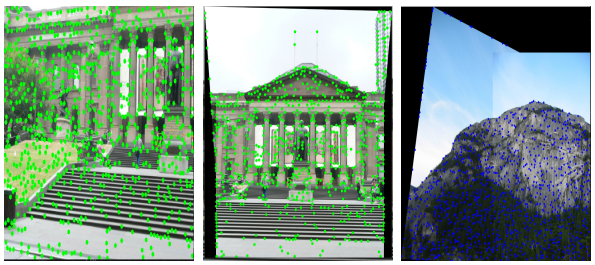

Fig. 4. ANMS for Train Set

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

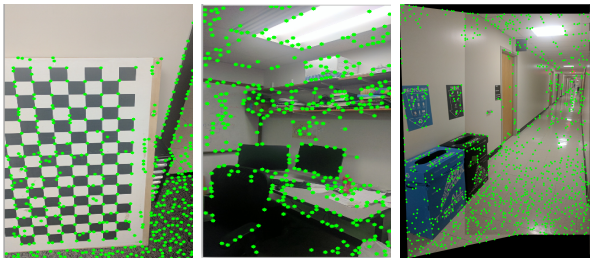

Fig. 5. ANMS for Test Set

C. *Feature Detection and Matching*

In the previous step, you found the feature points (locations of the Nbest best corners after ANMS are called the feature point locations). You need to describe each feature point by a feature vector, this is like encoding the information at each feature point by a vector. One of the easiest feature descriptor is described next. Take a patch of size 41×41 centered (this is very important) around the keypoint/feature point. Now apply gaussian blur (feel free to play around with the parameters, for a start you can use OpenCV's default parameters in cv2.GaussianBlur command. Now, sub-sample the blurred output (this reduces the dimension) to 8×8. Then reshape to obtain a 64×1 vector. Standardize the vector to have zero mean and variance of 1. Standardization is used to remove bias and to achieve some amount of illumination invariance. Furthermore, about Corner Matching we have to match the feature points among the two images that we want to stitch together. In computer vision terms, this step is called as finding feature correspondences between the 2 images.

a) Pick a point in image 1, compute sum of square differences between all points in image 2.
b) Take the ratio of best match (lowest distance) to the second best match (second lowest distance) and if this is below some ratio keep the matched pair or reject it.
c) Repeat this for all points in image 1.
d) You will be left with only the confident feature correspondences and these points will be used to estimate the transformation between the 2 images, also called as Homography.
e) Use the function cv2.drawMatches to visualize feature correspondences. Below is an image showing matched features.

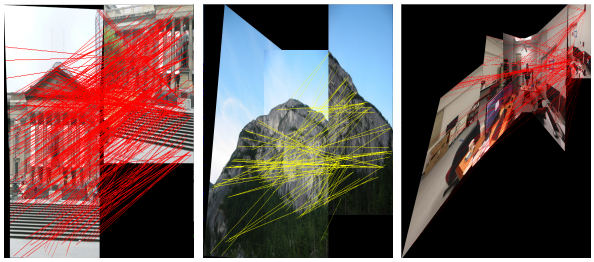

Fig. 6. Feature Detection and Matching for Train Set

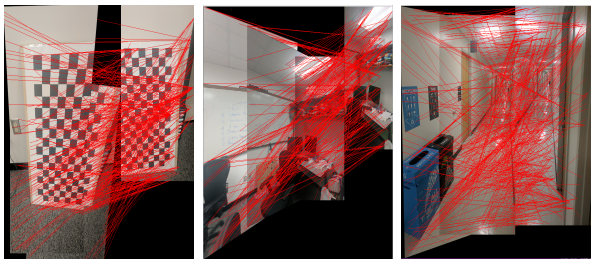

Fig. 7. Feature Detection and Matching for Test Set

D. *RANSAC for outlier rejection and to estimate Robust Homography*

We now have matched all the features correspondences but not all matches will be right. To remove incorrect matches, we will use a robust method called Random Sample Concensus or RANSAC to compute homography.

f) Select four feature pairs (at random), pi from image 1, pi from image 2.
g) Compute homography H between the previously picked point pairs.
h) Compute inliers where SSD(pi,Hpi)¡, where is some user chosen threshold and SSD is sum of square difference function.
i) Repeat the last three steps until you have exhausted Nmax number of iterations (specified by user) or you found more than some percentage of inliers (Say 90for example).
j) Keep largest set of inliers.
k) Re-compute least-squares H(cap) estimate on all of the inliers.

The output of feature matches after all outliers have been removed is shown below.

Fig. 8. RANSAC for outlier rejection on Train Set

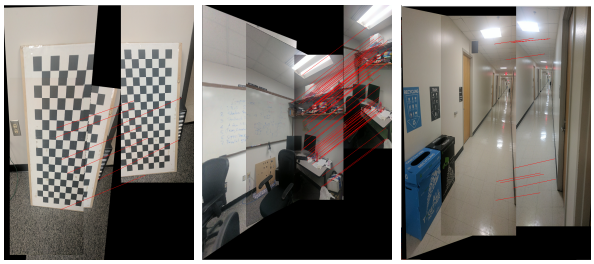

Fig. 9. RANSAC for outlier rejection on Test Set

E. Stitching the Results to obtain the Panorama or Blending of Images to get Panorama

Here, the Homography matrix that is obtained from the LSM(Least Square method), is used to warp the image that will be stitched to the second image. Furthermore, we get the translation which is obtained using all the feature points from the respective corresponding images. The important point here is that if we get translation as negative on either axis, then instead of moving the warped image, the original image is moved in the other direction. After the alignment is set, we get a much larger matrix that carries the dimensions of the two images involved.

F. Results

The corner detection function parameter corner quality needed to be tweaked many times after change of images. We also encountered problems stitching an already stitched image to another image. Our algorithm works well for stitching upto 4 images but falters a bit after the number of images to be stitched increases. In short we can say that, Panorama can be produced by overlaying the pairwise aligned images to create the final output image. Below are the Panoramic Images which were obtained from the Three TrainSets and First Three TestSetsout of Four TestSets Provided.

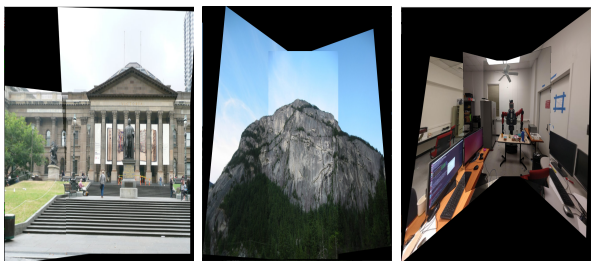

Fig. 10. Final Panoramic Image of Train Set

Fig. 12. Corner Detection of the TestSet 4

Fig. 13. ANMS of the TestSet 4

Fig. 11. Final Panoramic Image of Test Set

The TestSet4 was provided with mixture of TestSet1, TestSet2 and TestSet3, where we had totally 5 images from which first three images are from TestSet2, the fourth image is from TestSet3 and last image is from TestSet1. The results were quite interesting for the TestSet4 as they were not going to be same as that of other TestSets because of different images present.

The main problem with this Dataset was that it had three different backgrounds and for the Panorama stitching you are supposed to have only one background with every image resembling the previous image by atleast 30-50

The Results for the TestSet4 were not as expected or we can say that they were not similar to the previous three TestSet results.

For the TestSet4, I was able to get a partial Panoramic of first three images out of 5 images from the set, this is because the first three images had 30-50overlap and the other two images are totally different so it is not possible to make a seamless panorama with all the five images.

From this we can conclude that we need to capture

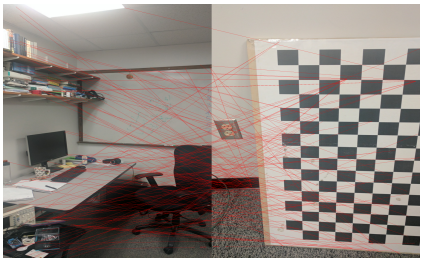

Fig. 14. Feature Detection and Matching of the TestSet 4

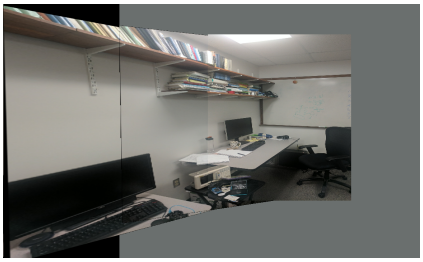

Fig. 15. Final Panorama of the TestSet 4

two sets of images in order to stitch a seamless panorama. Each sequence should have at-least 3 images with 30-50 image overlap between them.

II. Phase 2: Deep Learning Approach

In this section we will repeat the same task as Phase1, but we will generate our homography matrix using 2 Deep Learning models, Supervised and Unsupervised. The dataset used for training both the models are a small subset of MSCOCO dataset containing 5000 images for TRaining set and 1000 images for Validation set. We received another set of Data in the later part of the cycle on which we tested our model performance. The process of Phase 2 had the following steps

i) Synthetic Data Pair Generation
ii) Supervised Model
iii) Unsupervised Model

A. *Synthetic Data Generation*

We preprocess the training and validation set to be used as input in the training model that we create later. We take a random original image from the training set. We then select a random patch of size (128X128)with perturbation range of [-32, 32]. We use cv2.warpPerspective to achieve the same. We use cv2.getPerspectiveTransform to obtain a (3X3) Homography Matrix. We use this Homography(**H**) Matrix and the corner points of Patch A(original patch) to cretae the corner points of Patch B(Warped image). We stack the original and the warped image to form the input for the Training models.

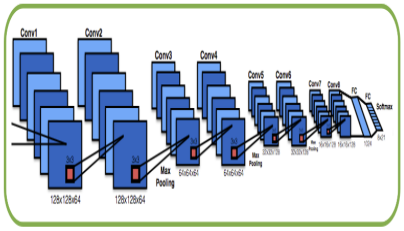

Fig. 16. Homography Model Architecture

B. *Supervised Model*

This model is implemented following the same architecture as mentioned in the paper [1]. The model consists of 8 convolution layeres followed by 2 fully connected layers, ending with a softmax. The input to the model as mentioned earlier is a 6 channel stacked image of Patch A and Patch B of size (128X128) as generated during the Data Generation phase. The output is a vector of size(1X8), which is the Homography representing the difference of 4 corners between two patches. The architecture of this model is shown in Figure 16. The loss function used here is a modified L2 Loss, where the root of mean squared loss between predicted Homography and Ground Truth Homography is compared. The plot for training loss over each epoch is shown in Figure 17. We have used Adam optimizer for this model with a learning rate of 0.005. We have also used a Dropout of 0.1 in order to achieve lower loss.

C. *Unsupervised Model*

The same model architecture is used as the supervised model. The output Homography is then processed through a TensorDLT class behaves as the cv2.getPerspectiveTransform where it takes the corners of the original patch as an input and the Homography to generate a 3X3 Homography matrix. This 3X3 matrix is then paseed though a Spatial Transformation Network to receive the warped patch of the original Patch. We have used warpPerspective fnuction of the kornial library to achive the Spatial transform. The loss in the unsupervised Model is not a traditional prediction vs groudtruth comparison. Here we are calculating the Photometric loss which is warping the original Patch using model output **H** matrix and compare the warped patch with ground truth Patch B from the dataset. The optimizer used here is ADAM and learning rate 0.005. The Epoch to Loss plot is shown in Figure 18.

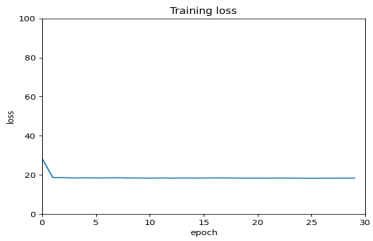

Fig. 17. Supervised Training Loss Plot

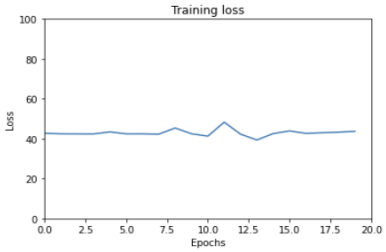

Fig. 18. Unsupervised Training Loss Plot

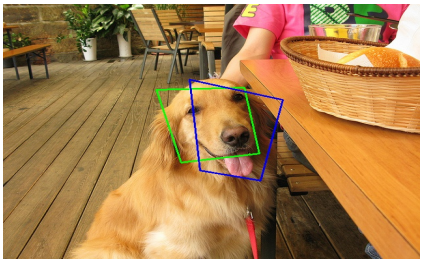

Fig. 19. Groundtruth vs Predicted Patch - Supervised

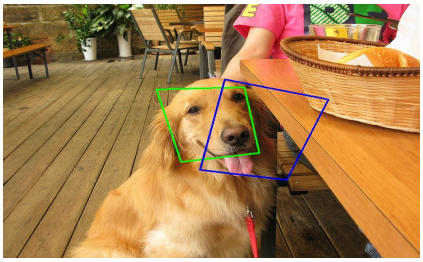

Fig. 20. Groundtruth vs Predicted Patch - Unsupervised

Fig. 21. Stitched Image TestSet1

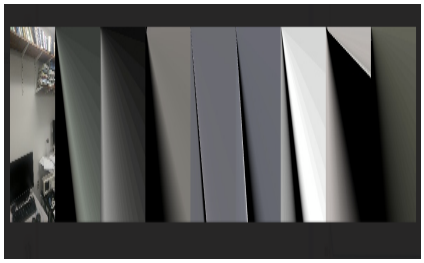

Fig. 22. Stitched Image TestSet2

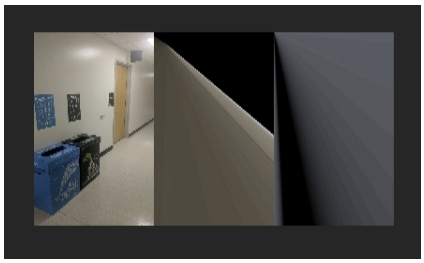

Fig. 23. Stitched Image TestSet3

D. Results

1) Supervised: The model loss does not go significantly lower. We have started with 50 Epochs 128 batches, but finally settled for 30 Epochs and 32 Batches since the loss plot was not going significantly under. One reason that we speculated was the generated dataset, we did not diversify the data by using different color channels or gathering more patches from single images. This may have played a part in the poor performance of the model. As shown in Figure 19, the predicted patch in Blue is quite different from the ground truth patch. This image was taken from the test set provided. *2) Unsupervised:* The model loss is quite high and remains constant. This could be partly because of the poor performance of the Supervised model which is used as a backbone architecture for the unsupervised model. Also the lack of diversity in the data played a part. Here we have used 20 Epochs and 32 batches. As shown

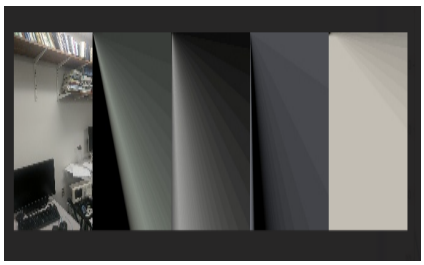

Fig. 24. Stitched Image TestSet4

in Figure 20, here also, the predicted patch in Blue is quite different from the ground truth patch. *3) Image Stitching:* We needed to stitch the images provided in the Test set, using our predicted homography from the model. Fiure 20, 21, 22, 23 are the stitched image but of very poor quality, given both the models are performing poorly.

REFERENCES

[1] https://omyllymaki.medium.com/algorithms-from-scratch-ransac-f5a03bed2fde

[2] https://learnopencv.com/non-maximum-suppression-theory-and-implementation-in-pytorch/

[3] https://www.geeksforgeeks.org/feature-detection-and-matching-with-opencv-python/

[4] https://arxiv.org/pdf/1606.03798.pdf