Project 1 : Autopano
CMSC733

Abhishek Nalawade
Master of Engineering in Robotics
University of Maryland, College Park, 20740
abhi1793@umd.edu
Using 1 Late Day

Aditya Jadhav
Master of Engineering in Robotics
University of Maryland, College Park, 20740
amjadhav@umd.edu
Using 1 Late Day

***Abstract*—This document presents our submission of Project 1 - Autopano aimed at stitching two or more images for creating uninterrupted Panorama. The report contains a detailed solution to Phase 1 and Phase 2 of the Panorama Problem. Phase -1 uses traditional Computer Vision techniques and the Phase - 2 makes use of a 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. The Homograhpy Net method used is illustrated in [1].**

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,

- Corner Detection using Shi-Tomasi Corner Detector
- Adaptive Non-Maximal Suppression
- Feature Detection and Matching
- RANSAC Algorithm for Outlier Rejection and Robust Homography
- Stitching the Results to obtain the Panorama

A. Corner Detection

Corners just like edges are characterized by sudden changes in Intensity in more than one direction. This step is for detecting such points of sudden change in the given images using any corner detector technique. For our purposes, we have used the Shi-Tomasi Good Features to track function provided by OpenCV (**cv2.goodFeaturesToTrack()**). The Shi-Tomasi Corner Detector produces better quality results than the Harris Corner Detector because of its use of Response Map equation and minimum eigenvalues.

B. Adaptive Non-Maximal Suppression

The Corners detection in the previous method include regions around them as well. Hence, this step is employed to detect the true local maxima which are evenly distributed. The ANMS algorithm finds the N_{best} corner in the image. The ANMS algorithm pseudo code is show ahead:

Fig. 1. Corner Detection Custom Set

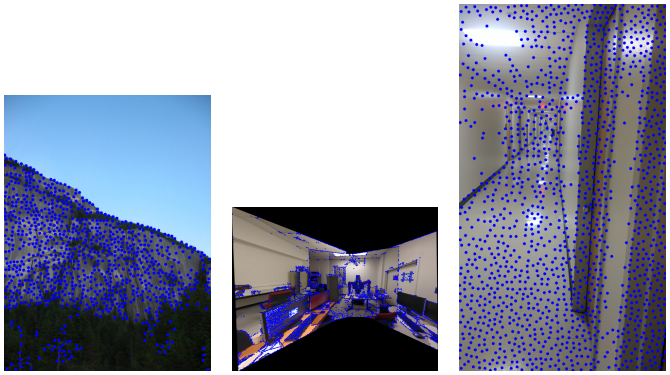

Fig. 2. Corner Detection Custom Set

C. Feature Detection and Matching

This step involves extracting the features around all the N_{best} corners found in the previous step. Each corner is characterized by its feature which is useful for matching corners. Firstly, a patch of 40 X 40 is used centered at each corner point in the image and is then Gaussian Blurred. This patch is then sub-sampled to 8 X 8 and reshaped to a 64 column vector.

Features detected for each corner until this point now need to be matched to points in another image. For a corner point in an image, we compute its Sum of Squared Differences with all the other feature descriptors from the image. After this we use a threshold value to accept the corner points with the lowest Sum of Squared Differences. These matched features are then drawn using a Draw Matches function after concatenating the

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

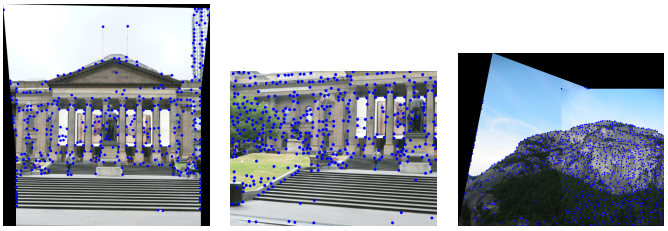

Fig. 4. ANMS Set

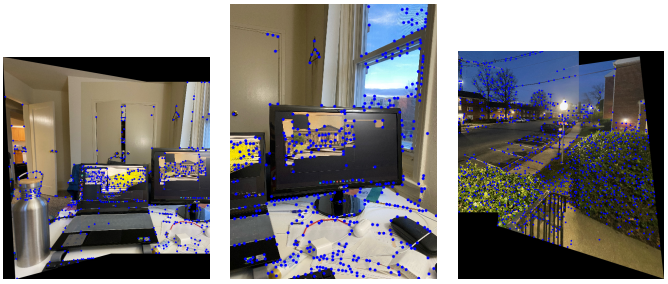

Fig. 5. ANMS Custom Set

two images in consideration alongside each other.

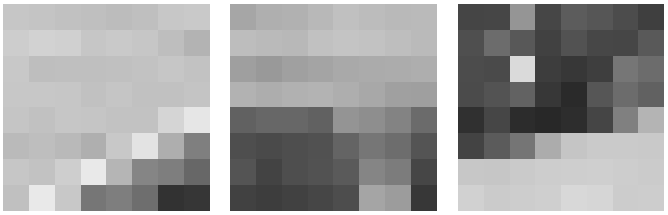

Fig. 6. Feature Detection Set

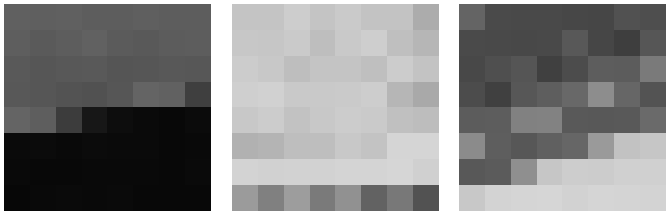

Fig. 7. Feature Detection Custom Set

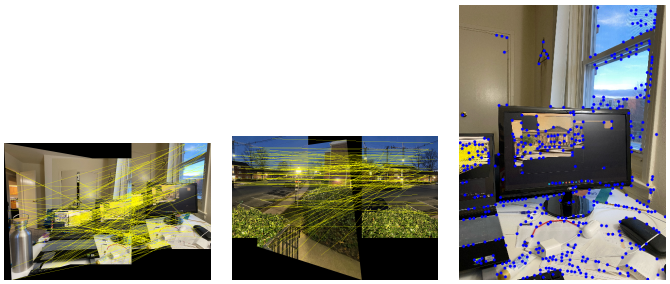

Fig. 8. Feature Matching Custom Set

Fig. 9. Feature Matching Test Set

D. RANSAC for Robust Homography

In order for the pipeline to avoid noisy and incorrect feature pairs and obtain better Homography, we incorporate the RANSAC outlier rejection algorithm. RANSAC takes coordinate pairs produced in the previous step, desired probability, a SSD threshold and max number of iterations N. Once the desired probablity percentage is achieved or a certain number of iterations are exhausted, then the inliers obtained are given as the points which are used for the Homography application. Least Squares method is used to obtain the robust Homography matrix.

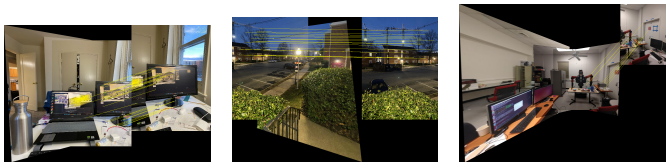

Fig. 10. RANSAC Matching Custom Set

Fig. 11. RANSAC Matching Test Set

E. Stitching the Images

The Homography matrix obtained from the Least Square method, is used to warp the image that will be stitched to the second image. The translation is then obtained using all the feature points in the corresponding images. If the translation is 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, a much larger matrix carries the dimensions of the two images involved.

F. Results

The goodFeaturesToTrack() 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.

Fig. 12. Panoramas Custom Set

Fig. 13. Panoramas Set

II. Phase - 2 : Deep Learning Approach

We use deep learning to imitate the entire pipeline of traditional Panorama stitching in this method. Deep learning is used in this case to compute the Homography matrix between multiple images. To estimate pose between multiple aerial images for collaborative autonomous exploration and monitoring, robust and fast Homography estimation is required. To compute the Homography, both supervised and unsupervised approaches were used. The results show that, when compared to traditional approaches, the unsupervised algorithm achieves faster inference speed while maintaining comparable or better accuracy and robustness to illumination variation. Furthermore, the unsupervised method outperforms the corresponding supervised deep learning method in terms of adaptability and performance.

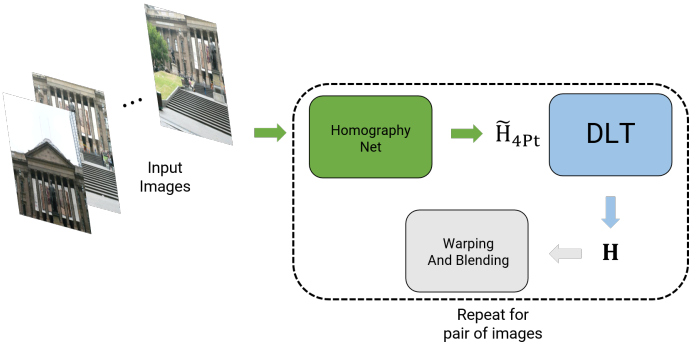

Fig. 14. Overview of the Deep Learning Approach

A. Data Generation

We chose to use the [1]'s example with an image size of 128 X 128 and a ρ value range of -32 $< \rho <$ 32, as suggested by the [1]. To begin, we randomly crop a patch P_A and save it in a suitable directory. The corners C_A and C_B are then randomly perturbed using the perturbation ρ in a positive to negative limit. The H4pt Homography between corner correspondences is given by $C_A - C_B$. The patch pairs are then stacked together to form the deep network inputs.

B. Supervised Approach

To be effective and learn the function of viewpoint matching, the supervised technique requires a large amount of data. We aimed to design a network that would perform well in determining a pixel shift of patches and then determining the homography matrix between the two patches using the data generation for infinite training data.

1. Network: The network contains eight convolution layers, one of which is fully connected, and eight layer outputs.. Our network was built in the same way as the one described, but with shortcuts. In the network, there are three shortcuts that lead from the input layer to the input of every other convolution layer. Following that, we employed the ReLU activation function and batch normalization.

2. Results: Using batch size 64 and a learning rate of 0.0001, we trained a model for 100 iterations. For all predictions, the loss is simple subtraction between the predicted homography and ground truth homography. On a GTX 1660 Ti, a forward pass of the supervised network took 0.002 seconds after the

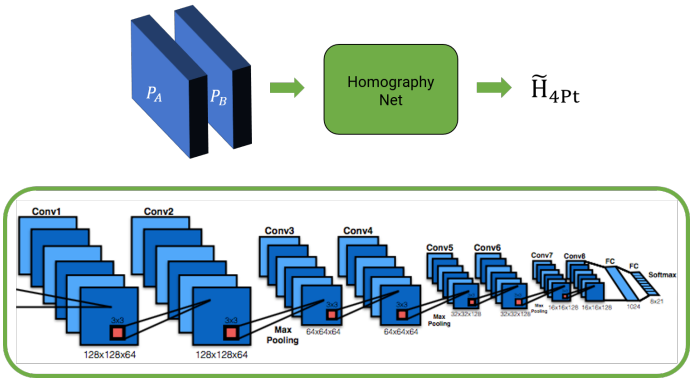

Fig. 15. Supervised Network using HomographyNet

network was loaded and the graph was initialized. Because of the inefficacy of the architecture, the outcomes of the supervised model were highly unsatisfactory. Our Supervised Model was producing a loss in hundreds after training for 50 iterations.

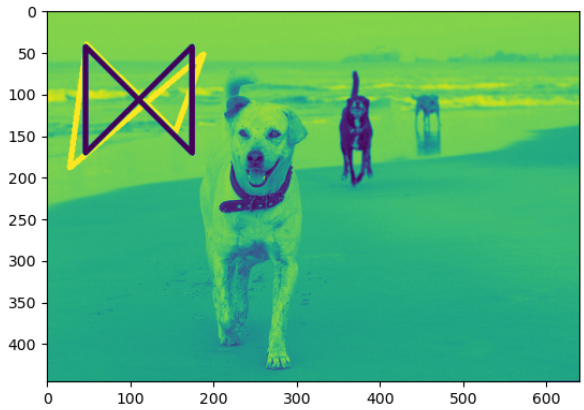

Fig. 16. Supervised Network Results - 1

C. Unsupervised Approach

In contrast to the supervised method, the unsupervised method computes the homography between two images without utilizing explicit labels for the 4-point homography description. This method's network is made up of four parts, as follows.

1. Regression Homography Model: This is the exact same network used in the supervised homography detection method. This network is fed two patches from the same image, one of which is warped via homography. In the four-point formulation, this model learns and predicts the homography that connects the two patches. This four-point estimate is then

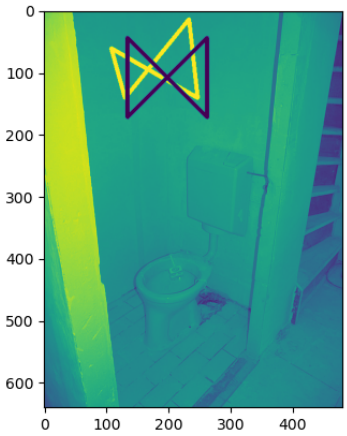

Fig. 17. Supervised Network Results - 2

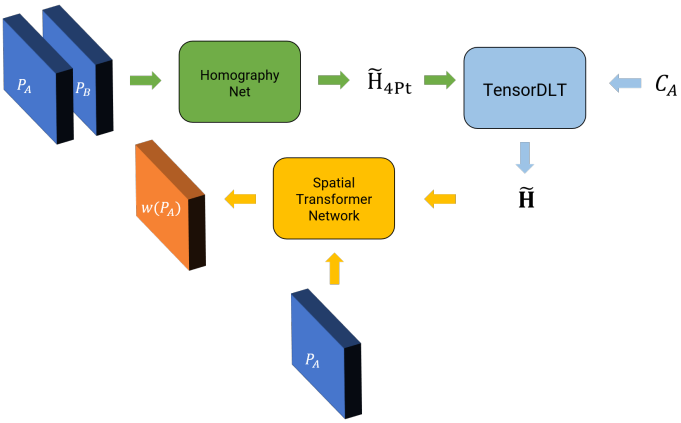

Fig. 18. Unsupervised Network

multiplied by the original patch's four corners to obtain an estimate of the warped patch's four corners.

2.TensorDLT Network: The four corners of the original and distorted patches are fed into this network as input. This then creates a system of linear equations and solves it in a differentiable method with singular value decomposition, allowing errors to propagate back through the network. The (3x3) homography matrix is obtained as a result of this procedure. In a differentiable approach, the network learns the functionality of OpenCV's cv2.getPerspectiveTransform() function.

3. Spatial Transformer Network: The homography matrix predicted by the TensorDLT network, as well as the original image from which the original patch was extracted, are fed into this network. It then learns how to apply homography to the original image to create a distorted patch. In a sense, the network learns how to use the openCV cv2.warpPerspective() method in a differentiable fashion.

4. Photometric loss: The L1 photometric loss is used to compare the warped patch produced from the Spatial Transformer Network to the ground truth warped patch. This loss is propagated backwards in the network. The Loss of Unsupervised Network fluctuates around 15 to 20 after training for 100 epochs.

5. Results: We would've been able to achieve significantly better results if we had more time with the Unsupervised network. But the implemented network also works pretty well considering the results obtained.

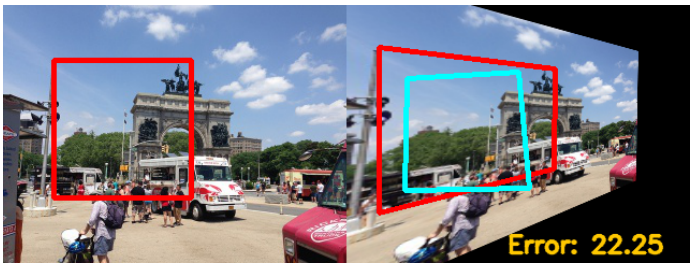

Fig. 19. Unsupervised Network Results - 1

Fig. 20. Unsupervised Network Results - 2

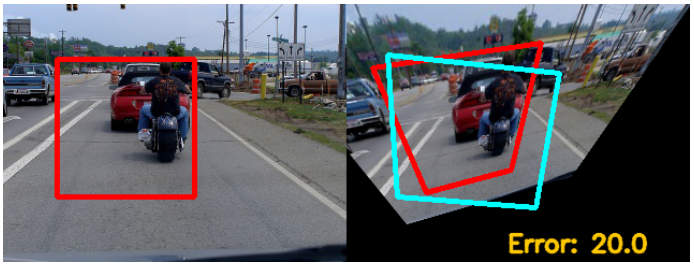

Fig. 21. Unsupervised Network Results - 3

III. Conclusion

For this project we implemented two approaches for stitching of two given images into a panorama, namely the traditional computer vision approach and the deep learning approach. Phase 1 was concerned with a feature based approach in finding the best Homography Matrix for the blending operation. And the deep learning approach made use of the Supervised and Unsupervised Neural Networks to estimate better Homography matrices.

References

[1] DeTone, Daniel, Tomasz Malisiewicz and Andrew Rabinovich. "Deep Image Homography Estimation." doi: 10.1109/TPAMI.2010.161. arXiv:1606.03798

[2] TensorDLT

[3] Nguyen, Ty, et al. "Unsupervised deep homography: A fast and robust homography estimation model." IEEE Robotics and Automation Letters 3.3 (2018): 2346-2353.

Fig. 22. Panoramas Results

Fig. 23. Panoramas Results