RBE/CS549: Project 1
My Auto Pano

Using 2 Late days

Prasanna Natu
M.S. Robotics Engineering
Worcester Polytechnic Institute
Worcester, MA
pvnatu@wpi.edu

Peter Dentch
M.S. Robotics Engineering
Worcester Polytechnic Institute
Worcester, MA
pdentch@wpi.edu

***Abstract*—In this Project, we implemented two methods of stitching two or more images to create a panoramic image. The goal is given two images to create a seamless panorama with repeated local features. The first method implemented was Classical Computer Vision Method and the second method was Deep Learning Method.**

***Index Terms*—RANSAC, ANMS, Panorama stitching, Image stitching Mosaicing, AutoPano**

I. Phase I: Classical Approach

In this method, we implemented the classical methods to create a panorama by stitching two or more images. The panorama stitching algorithm was used and is explained in detail along with the methodology with the subsequent output of the images.

The overview of the traditional method which we implemented is represented below:

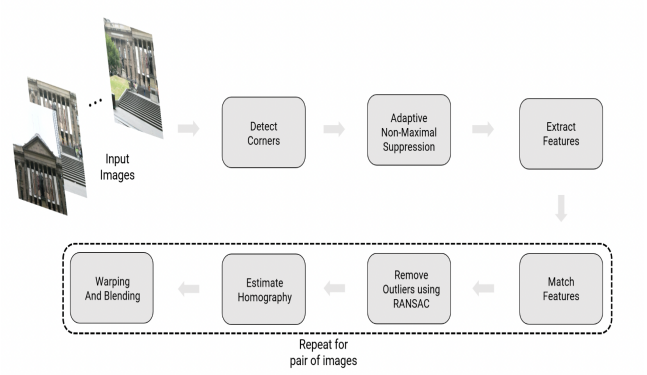

Fig. 1. Overview of the Traditional method

A. Corner Detection

A Corner detection algorithm is used to identify the corners or detection points on the images. Corners are considered to be distinctive and salient features and thus work as the best way to derive a relationship between images. This defines a set of features or descriptors common to the images in the form of visual information of each corner. In this case, for corner detection, we used Harris Corners Detection from Open CV. We found the following parameters optimal: kernel size X, Harris K parameter X, Harris Sobel kernel size X. Following are the images after Edge detection using the Harrison Corner detection method:

Fig. 2. Harris Corners

Fig. 3. Harris Corners

B. Adaptive Non Maximal Suppression

ANMS is used to refine the set of detected corners achieved by using the Harris Corners Detection algorithm.

This refinement step leads to corners such that they are equally distributed across the image to achieve good projective transformation.

1) After the Corner detection algorithm is performed after which the local maxima of the function are identified and the corners are defined as the pixels corresponding to these maxima.
2) Then, to suppress the redundant corners, the distances between each corner and its nearest neighbours are calculated, and the corners that are too close to each other are eliminated. Figure 2 below shows the ANMS output:

Fig. 4. ANMS Output 1

Fig. 5. ANMS Output 2

C. Feature Descriptor

A Feature Descriptor is a numerical representation of a visual feature in an image. It is used to provide a compact and distinctive representation or an identity of a each corner. Feature descriptors of one image are then compared with other descriptors of features in other images to find correspondence and establish relationship between images. Following are the steps that were followed:
1) A patch of size 40X40 is implemented by considering the chosen corners as the centre points of the patch.
2) Gaussian Blur is applied to the image. This results into a blurred output of size 8X8
3) Then the image is resized to a 64X1 feature vector.
4) Finally, the image is standardised to remove bias and achieve illumination invariance.

D. Feature Matching

After the feature descriptor to describe the visual characteristics of features in an image, the next step is to match the features by comparing the descriptors between images and finding correspondences. The goal is to determine which features in one image match with features in another image. This was done by calculating a similarity metric- L2-norm. The steps followed are as given below:

- Pick a point in image 1, compute sum of square differences between all points in image 2.
- Take the ratio of best match (lowest distance) to the second best match (second lowest distance) and if it is below the decided ratio then the matched pair is accepted otherwise it is rejected.
- Repeated this for all the points in image 1.
- This results into only confident feature correspondences and these points will be used to estimate the transformation between the 2 images.
  This is used to determine the percentage of the two images are overlapped with each other.

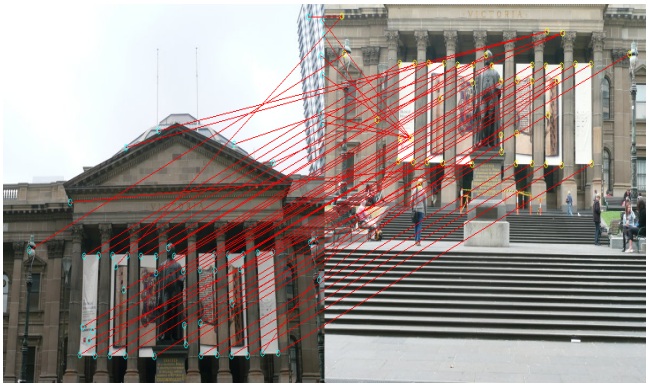

Fig. 6. Feature Matching Output

E. Random Sampling and Consensus (RANSAC)

Despite matching all the features results into having a large amount if outliers that can affect the accuracy of the model. The goal of RANSAC is to identify which part of the data is part of the set and which part is the outlier.

- RANSAC randomly selecting a subset of 4 data points from both the images.
- Then it calculated the error between the model and the remaining data points and identifies which are the closest to the model as inliers.
- The process is repeated multiple times with different random subsets.
- The model with the the largest number of inliers is considered to be the best estimate.

- This method gives us the stochastic demonstration of creating a dataset without outliers.

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

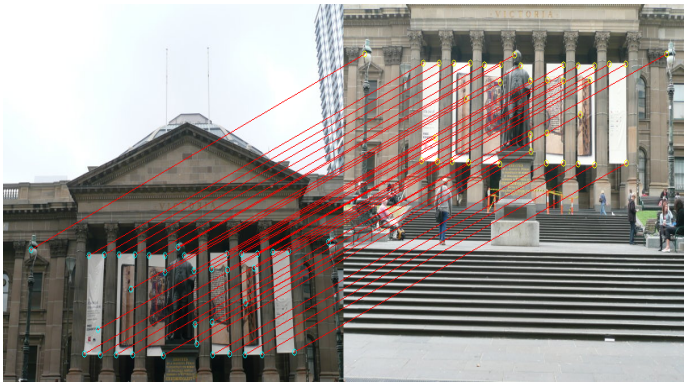

Fig. 7. Feature Matching Output After RANSAC

F. Stitching and Blending

After aligning the images, identifying the inliers and obtaining a relative homography between the two images we need to combine the common parts of both the images into a single, seamless panoramic image.
Below is the description of our steps:

- Compute the estimated homography between the two images from RANSAC
- Warp the second image using the homography information
- Display the wapred second image with the unmodified first image using the homography

Fig. 8. Stitching output 1

G. Results and Conclusion

The resulting panoramas show the accuracy of the homography estimate as they appear to be one single picture. By repeating these steps over multiple images which contain identical parts of the same scenery, a full panorama-style picture can be generated. All images tested were required to be low-resolution in order to keep the program run time under 10 seconds, as identifying and sorting thousands of features takes significant processing power.

Fig. 9. Stitching output 2

II. Deep Learning Approach

The deep learning approach to performing homography involves using a Convolutional Neural Network (CNN) to estimate the homography matrix between two images. The CNN typically consists of several convolutional and pooling layers, followed by a fully connected layer that produces the estimate of the homography matrix. The network is trained using a supervised learning approach as well as unsupervised learning approach.

A. Data Generation

Synthetic or artificially generated data is required to train the model. The MSOCO dataset was used to generate the synthetic pair of images required. We used the MSOCO date set of 5000 images to create the data for unsupervised and supervised approaches.

- An active region is defined and identified in the image. This is done keeping in mind that all the pixels of the patch will lie within the image after warping the random extracted patch.
- Following that a patch of the size of 128 X 128 is selected inside the active region which is defined.
- A random perturbation is done for all four corners in range.
- The perturbed corners along with the original corners are used to extract the data and create homography.
- The original image is then warped using inverse homography.
- A patch is then cropped from the warped image at the exact corners of the original image
- A groundtruth between the homography patches is generated.
- These image patches are stacked and the homography between them is defined.

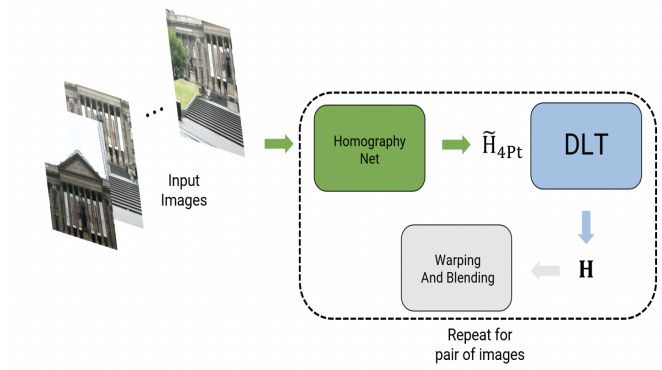

Fig. 10. Overview of panorama stitching using deep learning based method.

B. Supervised Approach

The two images which have their homography defined previously and have common ground truth between them are used here. The input image size we considered was a patch of 128X128.

The network was fed with an image of 128X128X2 grayscale image first. The two grayscale image patches were stacked behind each other and were fed into the network as Tensor [BCHW] and stored as numpy arrays.

Then the input was changed to an image of 128X128X6 RGB. The two RGB image patches were stacked behind each other and again were fed into the network as Tensor [BCHW] and stored as numpy arrays.

The overview of the supervised deep learning system for homography estimation is shown below:

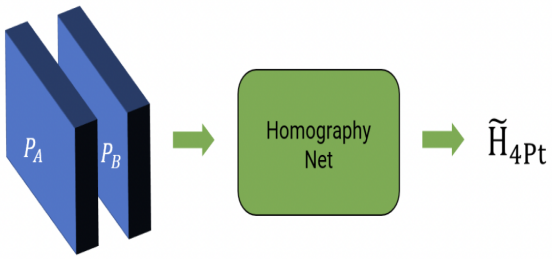

Fig. 11. Overview of the supervised deep learning system for homography estimation.

The architecture of the supervised approach is as below :

The output architecture of the supervised homography is shown:

1x8 matrix which represents the perturbation (H4Pt) in each corner of patch A corresponding to patch B.

The model trains to learn the Homography between Patch A and Patch B and displays the H4Pt values as the output

For this network, we saw a training loss of 13.80 on average. The training loss for the supervised approach is shown in the graph below:

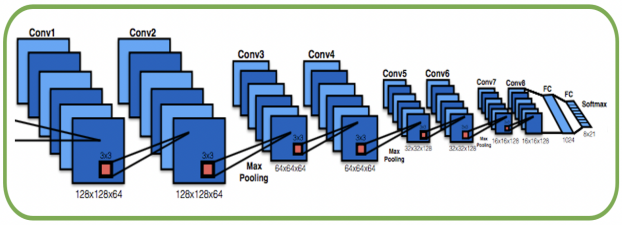

Fig. 12. Architecture of the network

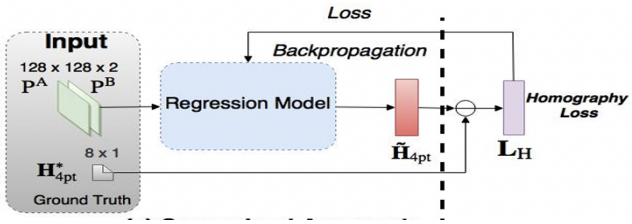

Fig. 13. Supervised Homography Architecture Diagram

Test images for supervised approach are given below:

C. Unsupervised Approach

The architecture of the unsupervised approach is as below:

In the unsupervised learning the images are stacked and passed to the network very similarly to the supervised. The difference it the unsupervised inputs are given without labels. At the end of the network we attached a TensorDLT module which takes in the value of the output and calculates the Homography matrix for the given value of the input image coordinates.

This Homography matrix is then transferred to a spatial transformer network or in this case we have used Kornia.geometry.get.perspective to get the warped image of the

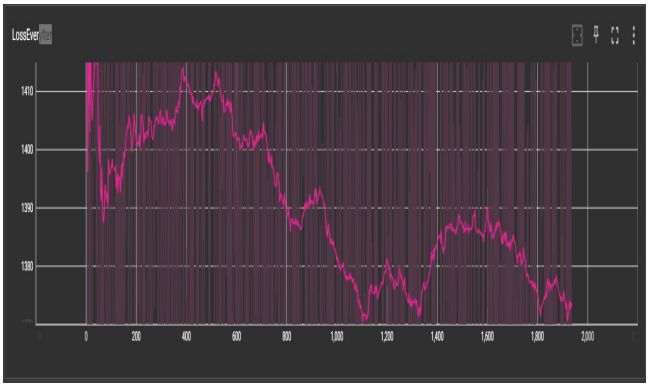

Fig. 14. Supervised Network: Training Loss

Fig. 15. Supervised Test 1

Fig. 16. Supervised Test 2

original image.
The TensorDLt output is kept differentiable for back propagation of the network to update the weights of the network as a result we would reduce losses and approach better results.
The loss for the unsupervised network was 0.081 on average. As we were skeptical of the input of the Kornia.geome/ STN we tried with both the original images and warped image as input to Kornia.

The input image size we considered was a patch of 128X128.

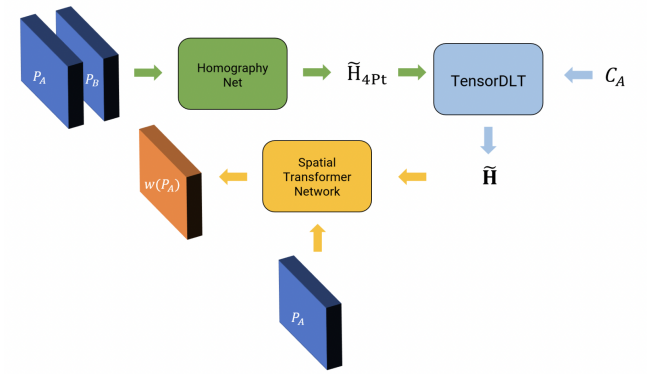

Fig. 17. Overview of the unsupervised deep learning system for homography estimation.

The network was fed with an image of 128X128X1 grayscale image. The two grayscale image patches were stacked behind each other and were fed into the network as Tensor [BCHW] and stored as numpy arrays.

The output architecture of the unsupervised approach is as below:

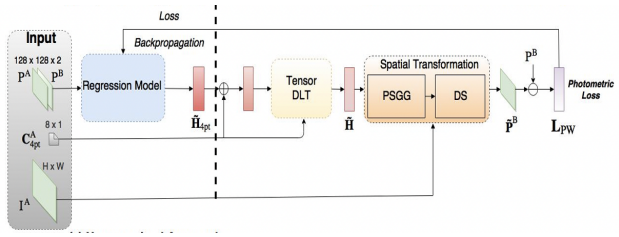

Fig. 18. Unsupervised Homography Architecture Diagram

The training loss for the unsupervised approach is shown in the graph below:

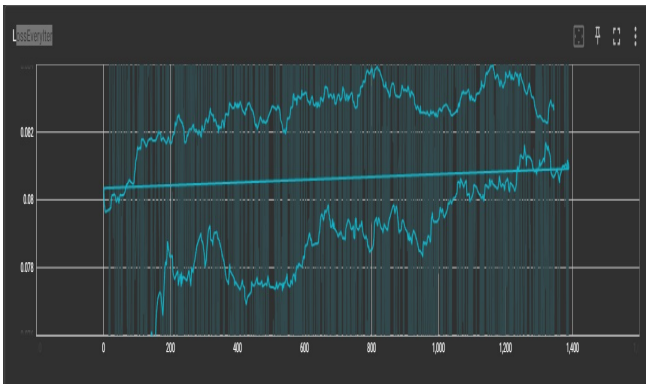

Fig. 19. Unsupervised Network: Training Loss

III. Result and Conclusion

In phase 1 the Corner Harris worked well for all most all images better than SHi tomashi corner detector.Studied different types of Blending such as Poison Blending etc. In Phase 2 for Homography network supervised approach both grayscale and RGB input worked similarly, with RGB being just edge better than grayscale. In unsupervised the TensorDLT must be made differentiable in order to compute the backward propogation of the gradient.instead of given STN network kornia.geometry.transform.warp_perspective() was used to find warped of original image.

References

[1] Nguyen, T., Chen, S., Shivakumar, S., Taylor, C., Kumar, V. (2018, February 21). Unsupervised deep homography: A fast and robust homography estimation model. Retrieved February 4, 2023, from https://arxiv.org/abs/1709.03966

[2] UK, P., Pérez, P., UK, M., Profile, M., UK, M., Gangnet, M., . . . Metrics, O. (2003, July 01). Poisson image editing: ACM SIGGRAPH 2003 papers. Retrieved February 4, 2023, from https://dl.acm.org/doi/10.1145/1201775.882269

[3] DeTone, D., Malisiewicz, T., Rabinovich, A. (2016, June 13). Deep image homography estimation. Retrieved February 4, 2023, from https://arxiv.org/abs/1606.03798

[4] Brachmann, E., Krull, A., Nowozin, S., Shotton, J., Michel, F., Gumhold, S., Rother, C. (2018, March 21). DSAC - differentiable RANSAC for camera localization. Retrieved February 4, 2023, from https://arxiv.org/abs/1611.05705