Project 1 : My AutoPano

Abhijeet Sanjay Rathi
M.S. Robotics
Worcester Polytechnic Institute
Email: asrathi@wpi.edu
Using 1 Late Day

Anuj Jagetia
M.S. Robotics
Worcester Polytechnic Institute
Email: ajagetia@wpi.edu
Using 1 Late Day

***Abstract*—This project presents a comprehensive exploration of panoramic image stitching through the integration of two distinct methodologies: Classical Computer Vision and Deep Learning. The primary objective is to seamlessly merge two or more images, leveraging repeated local features for the creation of cohesive panoramas. In Phase 1, we employ Classical Computer Vision techniques such as RANSAC, ANMS, and AutoPano to achieve corner detection, feature matching, and subsequent warping and blending. Phase 2 delves into the realm of Deep Learning, where advanced algorithms are implemented to enhance the panorama stitching process. The amalgamation of these methods contributes to a robust framework for generating high-quality panoramic images.**

I. Introduction

Computing the homography transformation matrix between two pictures serves as a foundation for various computer vision ideas. This project attempts to construct a homography matrix between two images by applying the ideas of corner detection, feature matching, and feature description. It also attempts to combine and distort these pictures into a smooth panoramic image. Both the classical approach (Phase 1) and deep learning networks (Phase 2) have been used to achieve the panorama effect.

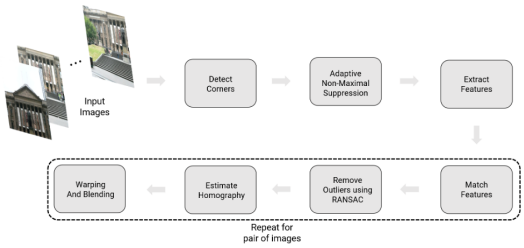

Fig. 1. *Overview of Panorama Stitching*

II. Phase I : Traditional Approach

A. Corner Detection

The first stage in our panoramic stitching method is to identify the corners of the photos we wish to combine.

We decided to use Harris Corners as our corner detection technique. Greyscale images were created from all of the colored input photographs. After that, we used cv2.cornerHarris on every picture to obtain the corners. The outcome of Harris Corner detection on a collection of photos is displayed in Fig. 3.

Fig. 2. Input Images (Set 1)

Fig. 3. Corner Detection

B. Adaptive Non-Maximal Suppression (ANMS)

It is used to refine the corners we got from Harris Corners. As the corners we get are very large in numbers and close to each other.

Refining makes it to equally distributed across the whole image and by doing this we can reduce the artifacts in the warping, application of anms is shown in Fig.4.

C. Feature Descriptors

It is used to describe every feature point to a feature vector of 64x1, basically a numerical representation of the visual feature

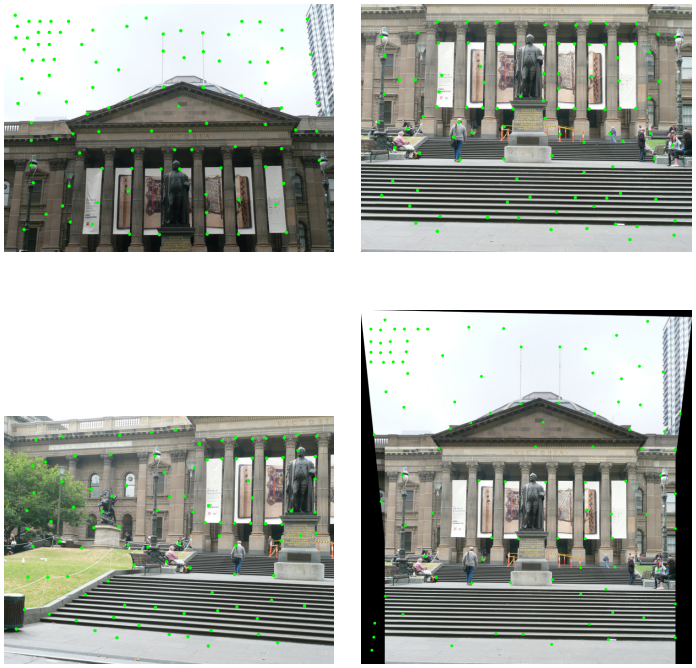

Fig. 4. ANMS

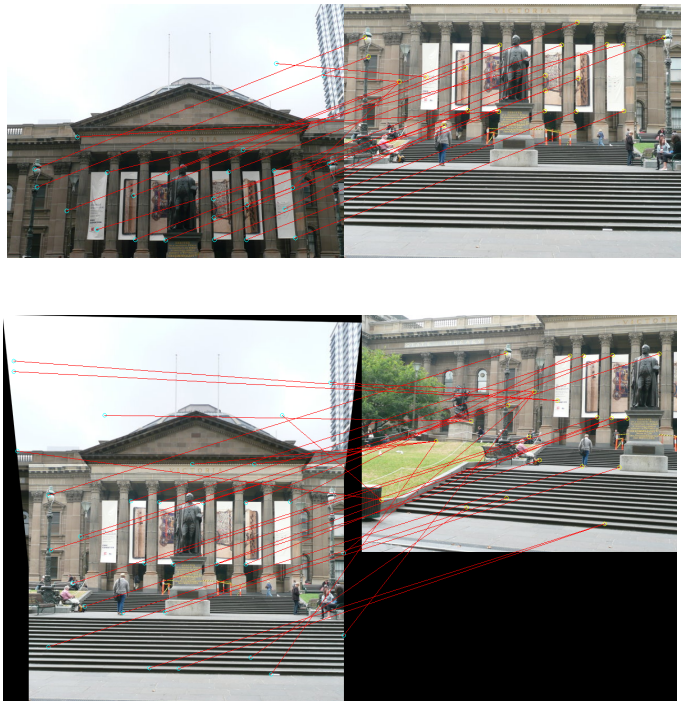

Fig. 5. Feature Matching

in an image.

From the best corners from ANMS. We extracted a 40x40 pixel patches(centered) around the feature point and applied gaussian blur to the patch by using cv2.GaussianBlur, then we sampled the blurred output patches to 8x8 and flatten the result after standardization.

D. Feature Matching

We have successfully acquired and saved the 64x1 feature vectors that represent the standardized feature descriptors for every image. The next step is to align the features of the two pictures we wish to stitch together. We compute the sum of square differences to all the feature vectors in the second image for every feature vector in the first image. The best matching pair is retained if the ratio of that pair to the second best matching pair was less than a predetermined threshold. Next, we carry out this action once more for each feature vector in the first image, output of feature matching is shown in Fig.5.

E. Random Sampling Consesus (RANSAC)

After feature matching we observed that we have some wrong matching which can cause wrong warping and stitching of the image ,to remove this RANSAC is applied. Using cv2.getPerspectiveTransform, we randomly choose a set of four matched couples to compute homography and then calculated SSD (sum squared distance) which cluster the points as inliers or outiers, We repeat this step many times and keep the best inliers and calculate the homography with these inliers using cv2.findHomography to get the best homography shown in Fig.6.

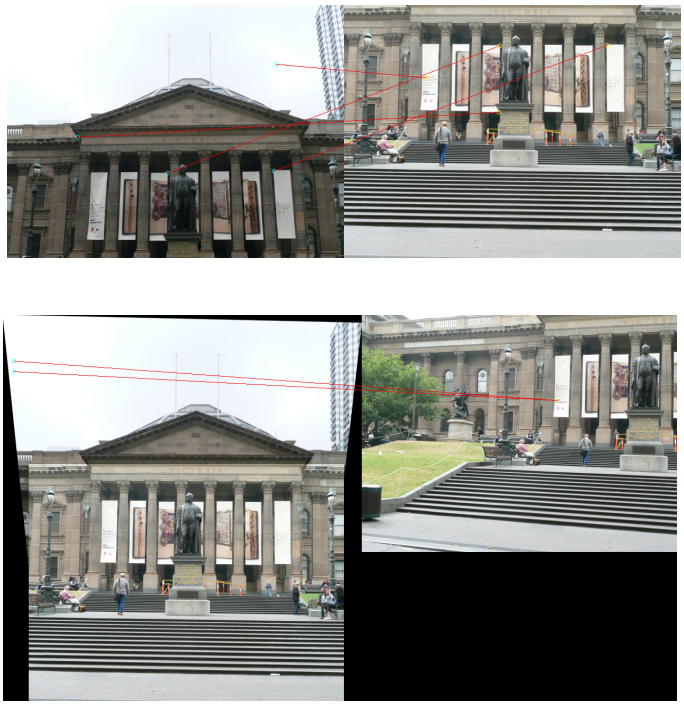

Fig. 6. Feature Matching after RANSAC

F. Warping and Stitching Images

Between the two images under consideration, the feature matching helps in the creation of a relative homography matrix. The (cv2.warpPerspective()) function, which takes the image to be warped and the homography matrix as arguments, is then used to warp image 2 with regard to Image 1.

After the images 1 and 2 are warped with respective to each other (shown in Fig.7), now the stitched image goes as one image with the image 3 (next image) and the same process is done till all the images are stitched together to form a single image on a canvas as displayed in Fig.8.

Fig. 7. *Stitched Image of Image 1 and 2*

Fig. 8. *Final Stitched Image*

G. Results and Conclusion

The generated panoramas, which appear to be a single image, demonstrate the correctness of the homography estimate.

By following these procedures, we may create a full panorama-style image by stitching together any number of photographs that feature the same landscape elements.

As the Images are not properly blended i.e. from where 2 images are attached we can see some uneven contrast and lines in the stitched image, for this we can use different types of blending to get a smooth panoramic image.

Fig. 9. *Stitched Output for Set 2*

For test set 2 images, all the points after RANSAC feature matching were in the black region; therefore, it couldn't proceed as the matching was less than the minimum required points.

III. Phase II : Deep Learning Approach

A. Data Generation

To train the model we need pair of images with known homography, for this we generated our own synthetic database by using a small subset of MSCOCO dataset. The reason for this is, it is very hard to define homography for such a huge data.

1) Step 1: We obtain a random patch of size 128 x 128 from an image and named it as Pa, the four corners of the patch is stored as Ca.

2) Step 2: The corners are then randomly peturbated in the range (-32,32) and translated and it's corners are stored as Cb Using this, we calculated the homography using (`cv2.getPerspectiveTransform()`) function.

3) Step 3: We warp the image Ia to get image Ib using (`cv2.warpPerspective`) and crop the image with four-corners and we get a patch Pb. Now we generate the ground truth $H_4pt = Ca - Cb$.

B. Supervised Model

The regression network suggested in the paper serves as the foundation for the supervised method applied in this project, known as `HomographyModel`, To obtain the anticipated 4-point homography matrix $\widetilde{H}_{4pt}$, we supply the ground truth labels H_{4Pt} for each picture pair as well as the input (stacked image pairs of PA and PB).

1) Structure of Neural Network: The architecture, implemented in the `SuperNet` class, is a Convolutional Neural Network (CNN) comprising eight convolutional layers (`conv1` to `conv8`). Each convolutional layer is followed by batch normalization and Rectified Linear Unit (ReLU) activation.

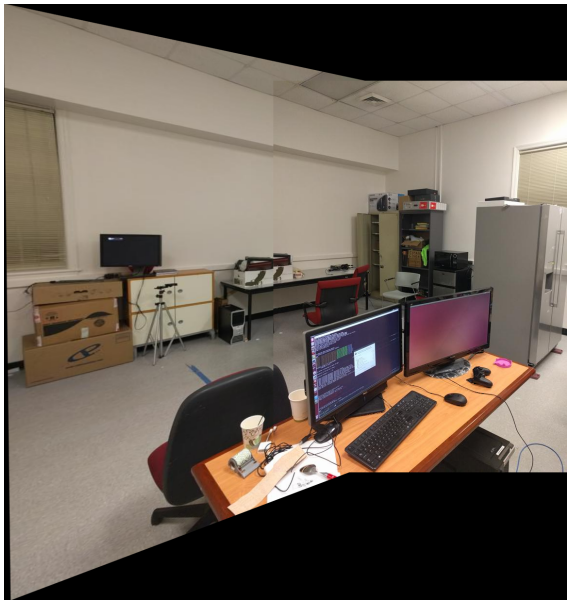

Fig. 10. *Stitched Output for Set 3*

Fig. 11. *Set 3 Output after 3 images*

After image 2 the homography of Stitched image with image 3 is not valid therefore the image is stretched wrongly when combined with image 3.

Solution to this problem is to combine image from both the sides keeping one image as reference image and then join the stitched image from left and right.

Max-pooling layers with a kernel size of 2 and stride of 2 are employed for downsampling.

The number of input channels for the first layer is 2 and is fed with an image of 128X128X2 grayscale image.

After the convolutional layers, the feature maps are flattened and passed through two fully connected layers (`fc1` and `fc2`). The network incorporates dropout (`dropout`) with a rate of 0.5 for regularization.

The final layer outputs an 8-dimensional vector representing the predicted homography parameters. This network is shown in Fig. 19

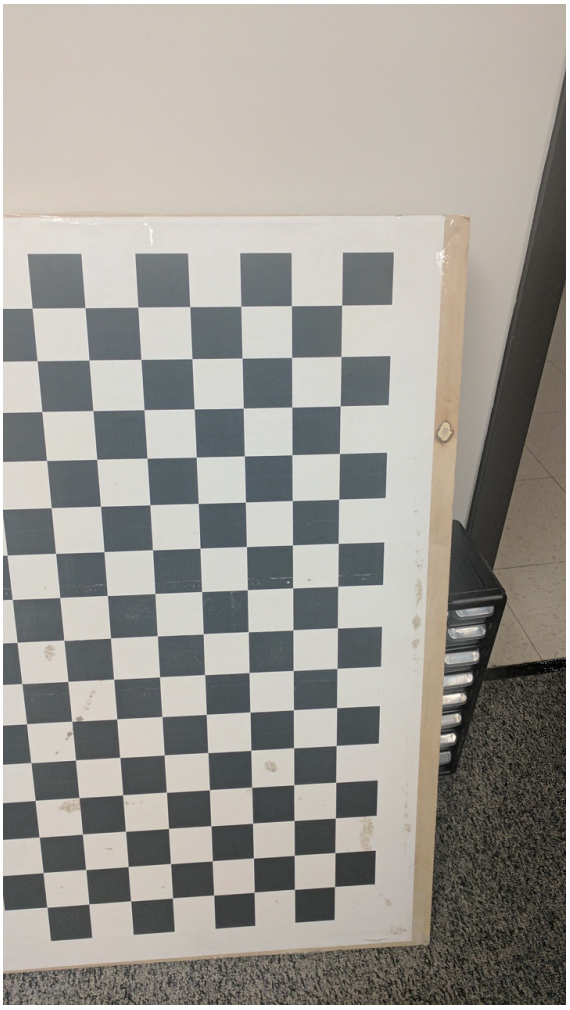

Fig. 12. *Stitched Output for TestSet 2*

In this case the matching was not proper as the checkers of the image are similar to each other which makes the code difficult to find unique corners in the next image and the final output was ssame as the last image provided for stitching.

2) Loss Function: The loss function for tranining utilizes the PyTorch `nn.MSELoss` function. It converts ground truth labels to float and applies the square root to the calculated MSE loss. The loss of the model is plotted over epochs and iterations which is shown in Fig.15 & Fig.16 respectively.

C. Unsupervised Model

The unsupervised neural network in this project is designed to estimate homographies without explicit ground truth labels. The network architecture, named `UnsuperNet`, processes input images and coordinate batches to predict both a warped patch (`PB_pred`) in image B and the 8-dimensional homography parameters (`H4pt_predict`).

1) Structure of Neural Network: The architecture follows a Convolutional Neural Network (CNN) design, featuring eight convolutional layers (`conv1` to `conv8`). Each convolutional

Fig. 13. *Stitched Output for TestSet 3*

The output for TestSet 3 was stitched properly compared to others.

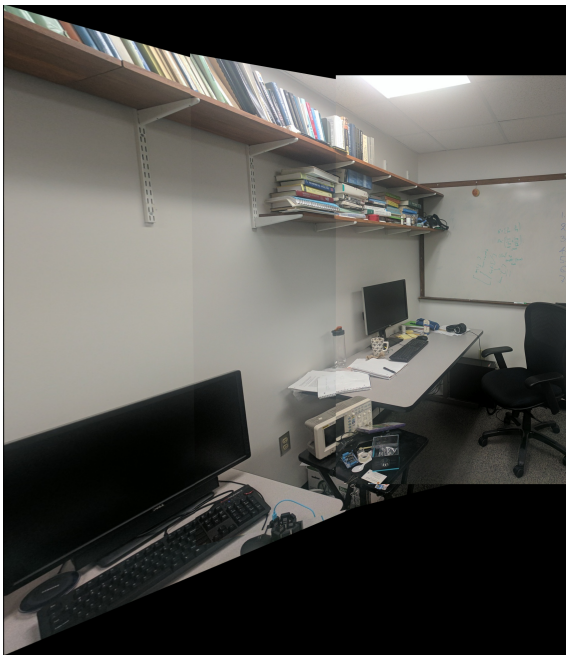

Fig. 14. *Stitched Output for TestSet 4*

The output for TestSet 4 was stitched properly until there was overlapping in the images after stitching of image 3, it couldn't find any similarity in image 4 as the scene was different from previous one.

layer is accompanied by batch normalization and Rectified Linear Unit (ReLU) activation functions. Max-pooling layers with a kernel size of 2 and stride of 2 are employed for downsampling the spatial dimensions.

After the convolutional layers, the feature maps undergo flattening and pass through two fully connected layers (`fc1` and `fc2`). The output of `fc2` is an 8-dimensional vector representing the predicted homography parameters.

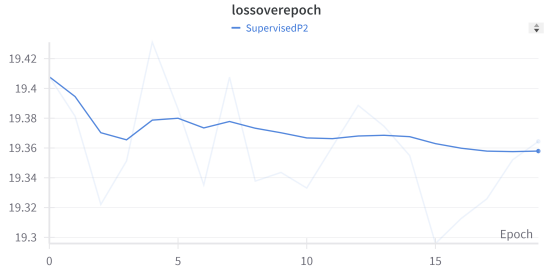

Fig. 15. *Supervised : Loss over Epochs*

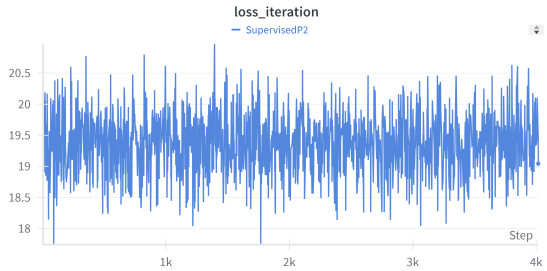

Fig. 16. *Supervised : Loss over Iterations*

A dropout layer (`dropout`) with a dropout rate of 0.5 is included for regularization, followed by a warp using Kornia (`warp_perspective`) to transform patches from image A (`Pa`) based on the predicted homography parameters.

The `Tensor_DLT` function is utilized for Direct Linear Transform (DLT) homography estimation, incorporating the predicted homography parameters (`H4pt`) and points in image A (`C4pt_A`).

2) Loss Function: The loss function for training is L1 loss (`F.l1_loss`), measuring the mean absolute error between the predicted patch in image B (`PB_pred`) and the ground truth patch in image B (`PB`). The loss of the model is plotted over epochs and iterations which is shown in Fig.17 & Fig.18 respectively.

Model	Supervised	Unsupervised
Epochs	20	20
Batch Size	50	128
Learning Rate	0.005	0.0001
Optimizer	SGD	AdamW
EPE	53.383	53.509

D. Deep Learning Stitching

In order to create a panoramic image from the models we trained, we gave the whole image of size 128X128 into the model as a single patch, resized it according to the requirement of the model after giving the input to the model, we did the same process for consecutive images as we did in Phase 1. which was stitching image 1 and 2 and then stitch that the output of image

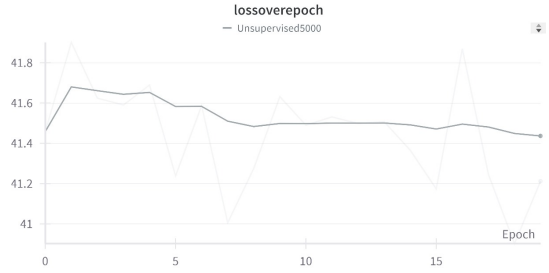

Fig. 17. *Unsupervised : Loss over Epochs*

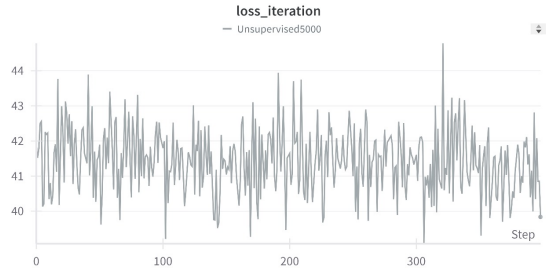

Fig. 18. *Unsupervised : Loss over Iterations*

2 to image 3 and so on. We stacked all the images according to the CNN network. The output image of the panorama is then again resized to size 256X256 as the output be the network was very small because of the stitching of multiple images.The input and output of the Supervised Model are shown in Fig.20 & Fig. 21 respectively and the input and output of the Unsupervised Model are shown in Fig.22, 23 & Fig. 24 respectively

E. Results and Outcomes

A comparison between the two methods of homography estimate performances was conducted and is displayed in Fig. 25 and 26 for Supervised and Unsupervised Model respectively. The Image overlayed with homography estimated by deep learning model shown in Green and ground truth is shown in Blue. In this case the Supervised and Unsupervised approach yielded almost same estimates.

REFERENCES

[1] D. DeTone , T. Malisiewicz and A. Rabinovich, *Deep Image Homography Estimation* 2016.

[2] T. Nguyen , S. W. Chen, S. S. Shivakumar, C J. Taylor and V. Kumar, *Unsupervised Deep Homography: A Fast and Robust Homography Estimation Model* 2018.

Fig. 19. *Network Structure*

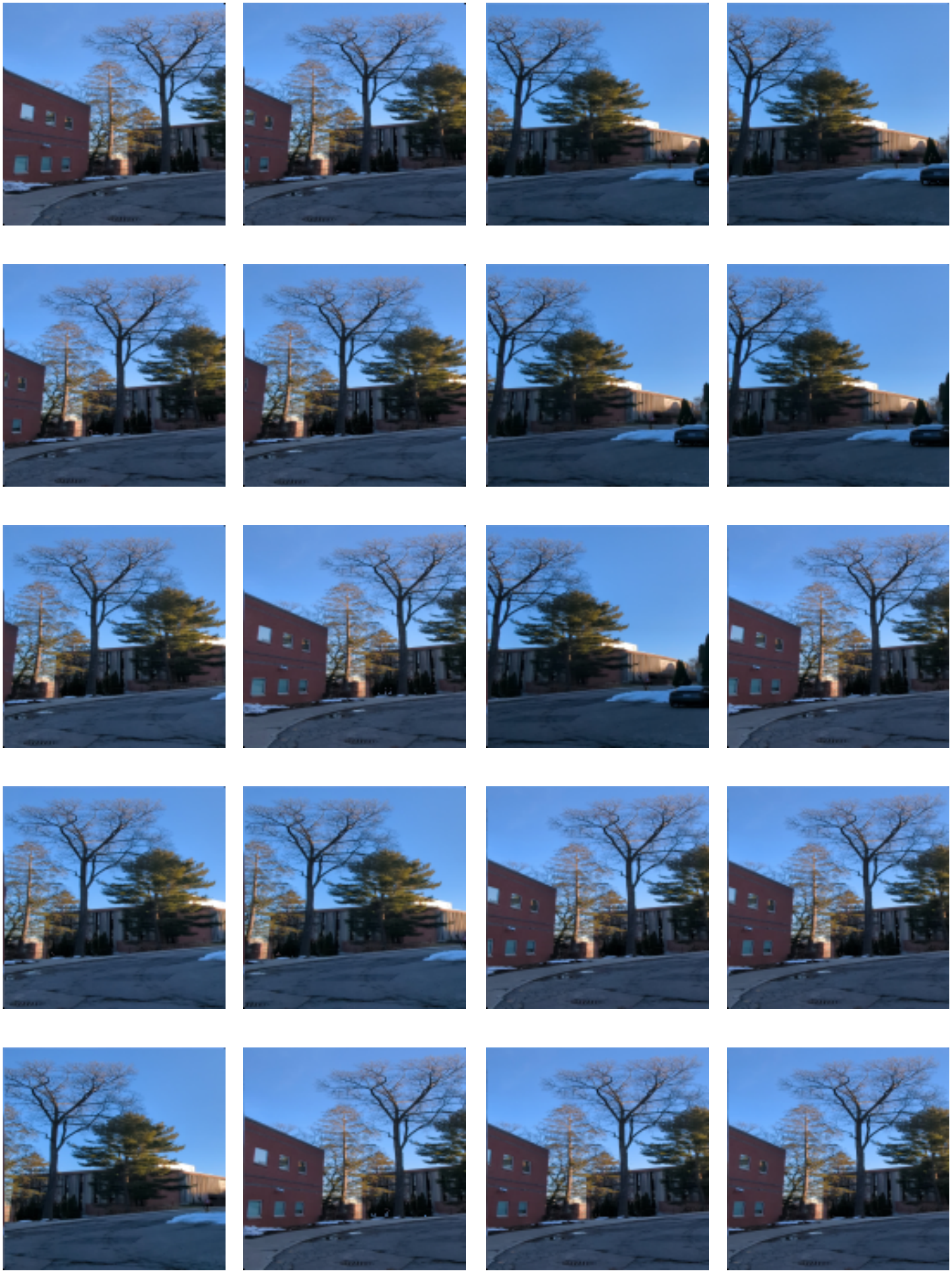

Fig. 20. Input images for Supervised model

Fig. 21. *Final Stitched Image Output from Supervised Model*

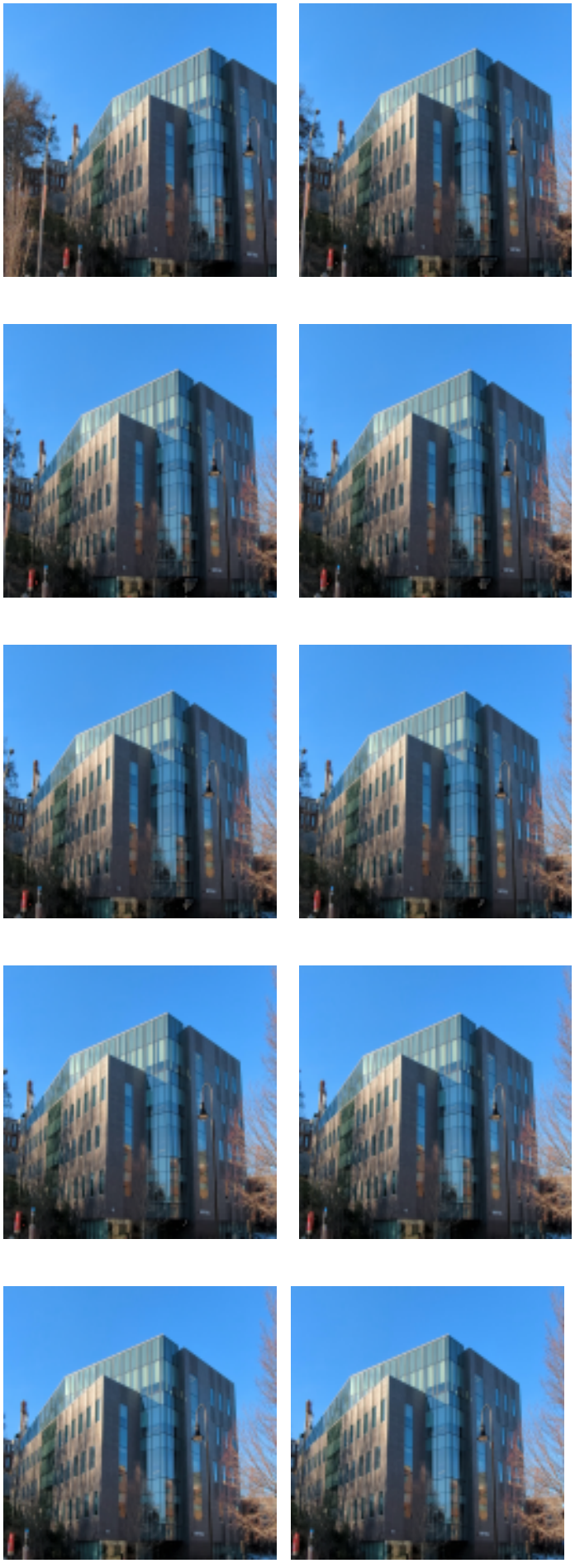

Fig. 22. 10 input images for unsupervised model

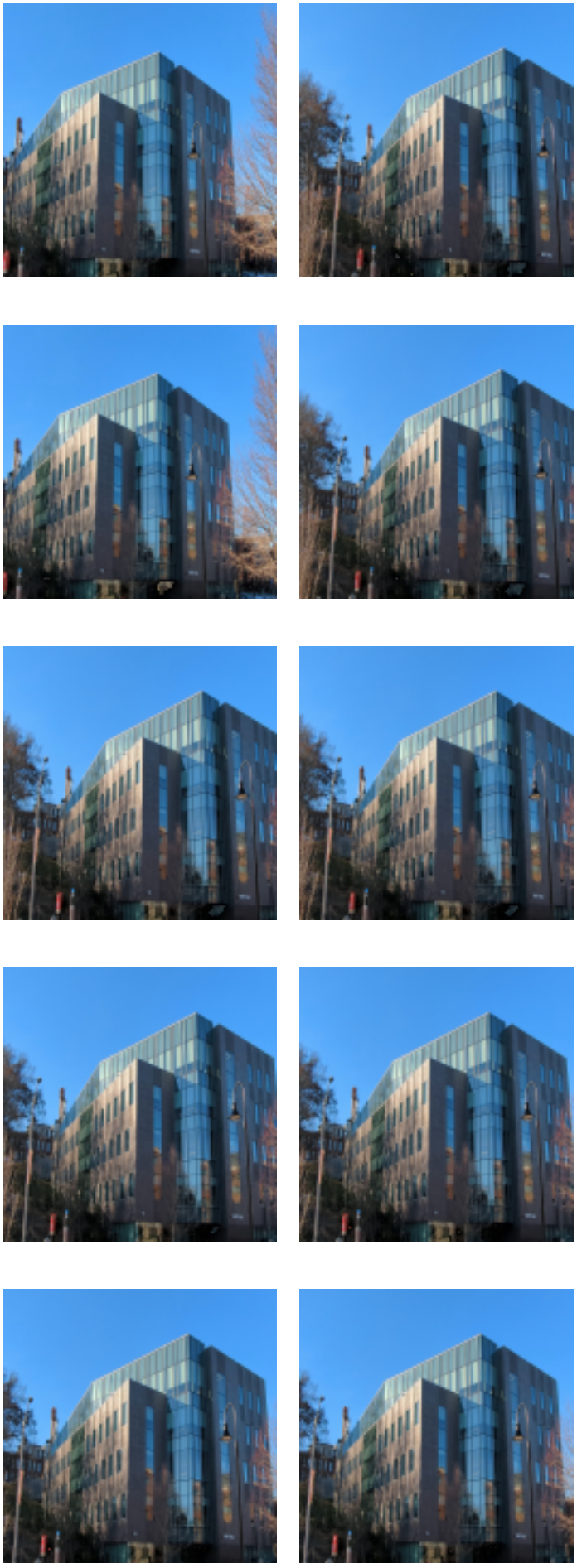

Fig. 23. Next 10 input image for Unsupervised model

Fig. 24. *Final Stitched Image Output from Unsupervised Model*

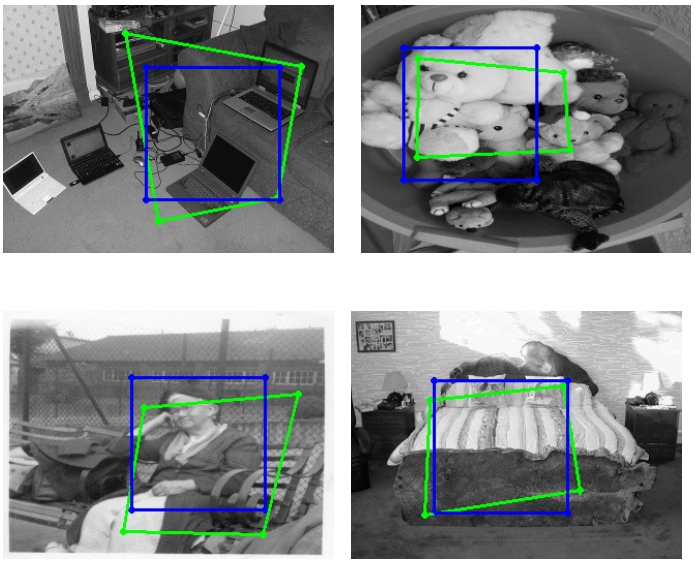

Fig. 25. *Supervised Output for 4 images*

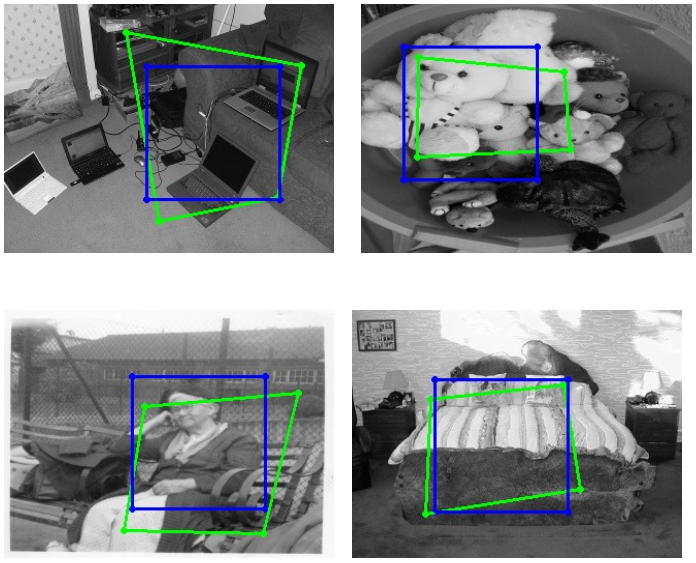

Fig. 26. *Unsupervised Output for same 4 images as supervised*

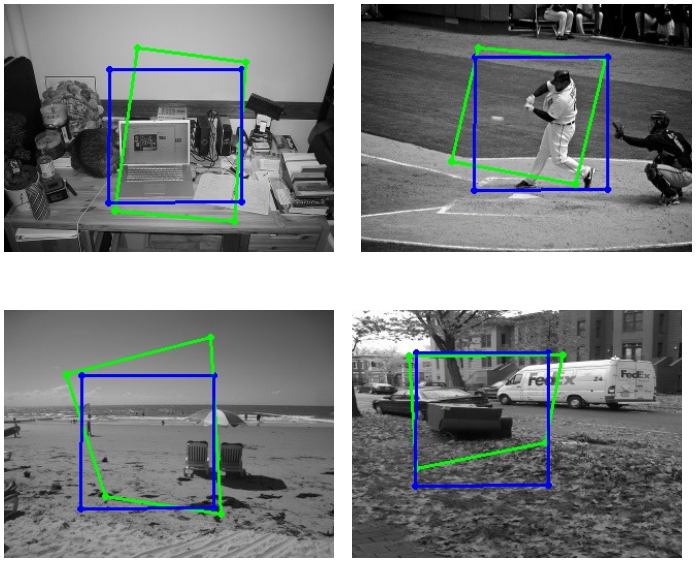

Fig. 27. *Unsupervised Output images*

Fig. 28. Input Images (CustomSet 1)

Fig. 29. *Output Panorama for CustomSet1*