Computer Vision - Project 1 - MyAutoPano

USING 1 LATE DAY

Muhammad Sultan
Worcester Polytechnic Institute
Worcester, MA, USA
Robotics Engineering
Email: msultan@wpi.edu

Harsh Verma
Worcester Polytechnic Institute
Worcester, MA, USA
Robotics Engineering
Email: hverma@wpi.edu

Jesdin Raphael
Worcester Polytechnic Institute
Worcester, MA, USA
Computer Science
Email: jraphael@wpi.edu

***Abstract*—Project1: MyAutoPano focuses on creating a Panaroma using multiple images. To convert images into the same coordinate system we find the H matrix (Homography matrix) and then stitch and blend the images. Finding the H matrix is done in 3 different ways. In Phase 1 the traditional approach is used to calculate the H matrix by matching corners. This is done by first finding corners using HarrisCOrner Detection, then we use ANMS to reduce unwanted corners and get corners in uniform distance from each other. Once we have our corners we use matching to match corners from one image to the other. Ransac is then used to remove the outlier matches. Finally the H matrix is computed using the corner matches. In Phase 2 we use supervised and unsupervised methods to compute the H4 matrix.**

I. Phase 1: Image Stitching with Traditional Approach

In this section we describe the Traditional approach used to create our Panaroma. The steps for this approach can be summarized into the following points.

1) Corner Detection
2) Adaptive Non Maximal Suppression (ANMS)
3) Feature Descriptor
4) Feature Matching
5) RANSAC
6) Stitching and Blending

A. Corner Detection

The first step in computing the Homography using the traditional approach is to compute Corners. Corners are such pixels where the intensity changes drastically in all directions. To compute the corner points we use CV2's Harris Corner Detection which returns the Corner score Image C_{img} and the corner points. These corners are marked as red dots on the images as shown in Fig 1.

B. Adaptive Non Maximal Suppression (ANMS)

Using the Corner score Image C_{img} and the corner points we get from Harris Corner Detection we apply ANMS. The reason for applying ANMS is to minimize weird artifacts by selecting N_{best} equally distributing corner points across the image. The ANMS algorithm is implemented based on Fig 2. [1]. The corners left after ANMS are shown in Fig 3.

C. Feature Descriptor

After obtaining the N_{best} corners using ANMS we create a Feature Descriptor for each corner point. To do this we follow the following steps [1].

1) Create a Patch of size 41x41 around the corner point
2) Apply Gaussian Blur with a kernel size of 7x7.
3) Subsample the blurred image to 8x8
4) Reshape to obtain 64×1 vector
5) Normalize to remove bias and achieve some amount of illumination invariance.

D. Feature Matching

We want to find the feature correspondence between the two images so that we can stitch them together. For 1 Feature Descriptor in $Image_1$ we calculate the sum of Square Difference (1) with each Feature Descriptor in $Image_2$. Using the lowest and second lowest SSD's ($best\_ssd$ and sec_best_ssd) if the ratio $\frac{best\_ssd}{sec\_best\_ssd} < min\_ratio$ we use it as a match. We have used $min\_ratio = 0.5$. We then repeat this step for all Feature Descriptors in $Image_1$. This gives us a list of strong correspondence (matches) for the features.

$$\sum_{j=1}^{n}(x_{1k} - x_{2j})^2 \tag{1}$$

The features matched are shown in Fig 4.

E. RANSAC (Random Sample Concensus)

We use RANSAC to remove the incorrect matches or outliers found in the Feature Matching Step. The steps for RANSAC are shown below [1].

1) Repeat for n_{max} iterations
   a) Select four feature random pairs p_i I_1 and p'_i from I_2.
   b) Compute H between p_i and p'_i pairs.
   c) Compute inliers where $SSD(p'_i, Hp_i) < \tau$, where τ is threshold (0.5)
2) Keep largest set of inliers.
3) Re-compute least-squares H estimate on all of the inliers.

Applying RANSAC would get rid of feature matching thus allowing us to calculate a more optimal H estimate. Fig 5 shows the images with removed outliers using RANSAC.

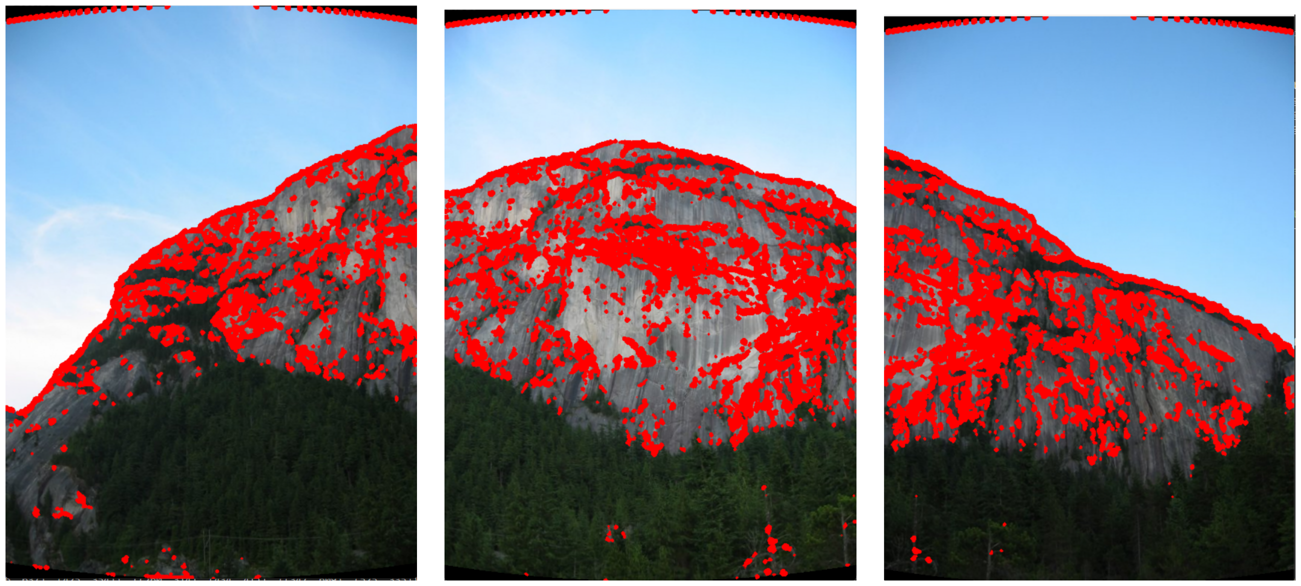
Fig. 1: Corner Points Detected using Harris Corner detection

```
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. 2: ANMS algorithm.

F. Stitching and Blending

Stitching was performed using the cv2 functions *perspectiveTransform()* and *warpPerspective()* In order to align the images and create the final panorama, we tried multiple techniques:

- Aligning all the images from left to right: Accumulated warping with each iteration caused the final image to be too warped to be blended further than 4 images.
- Aligning half the images from left, half from right and then combining them: This technique worked well for 4 images on each side, but when the right and left were to be merged, the common areas were too warped to be blended.
- Applying a *cylindrical projection transform* to each image, left to right: This technique gave the best results for the combined panoramas. For each set of images, the focal length parameter of the cylindrical transformation paradigm needs to be changed. However, a value of 1000-1200 worked well with most sets.

G. ***Results***

A summary of our implementation is:

1) Cylindrical transformation improved the results significantly. Fig [2] shows (left to right): 4 images of the train set stitched normally, The same 4 images stitched with cylindrical transform, the panorama of all the images combined in the train set (The largest train set of 8 images)
2) When following the two-halves technique, we found that the centre image becomes too small in size for the algorithm to detect the feature pairs, hence failing the stitching.
3) To get rid of too less matches, we implemented a check in the functions. If the number of feature pairs after RANSAC was less than 4, the particular image was skipped.
4)

II. PHASE2

In this section we describe the Deep Learning Approaches implemented to for finding the Homography estimate. Two models have been trained for this: Supervised Learning (SupervisedNet) and Unsupervised Learning (UnsupervisedNet).

The input to the SupervisedNet are two patches P_A and P_B stacked depthwise along with the H_{4point} matrix which is the label as shown in Eqn (2). The reason we use H_{4point} instead of H is that while converting the H matrix into a single vector we are mixing both the rotational and the translation points [3].

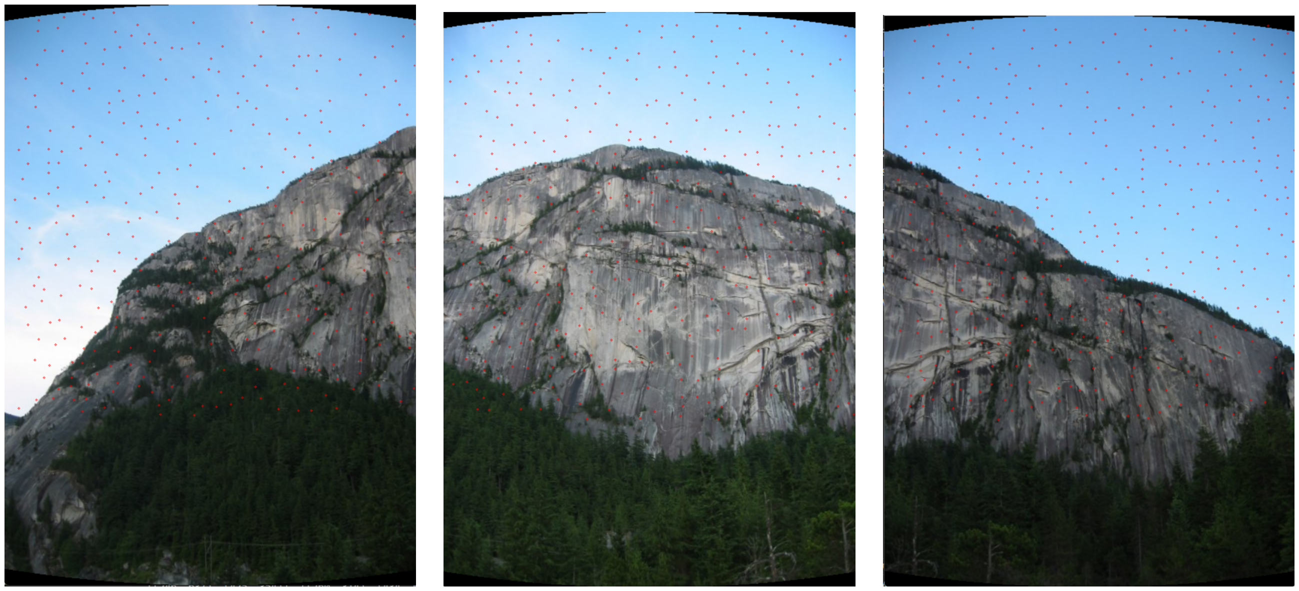

Fig. 3: Corner Points after ANMS

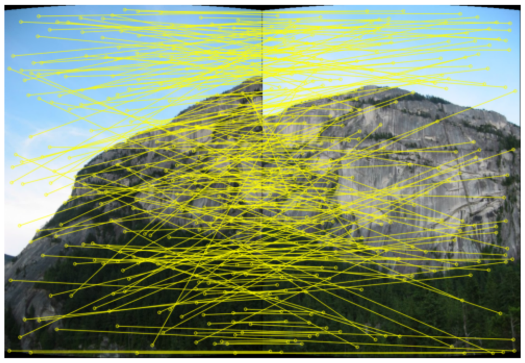

Fig. 4: Matched Features

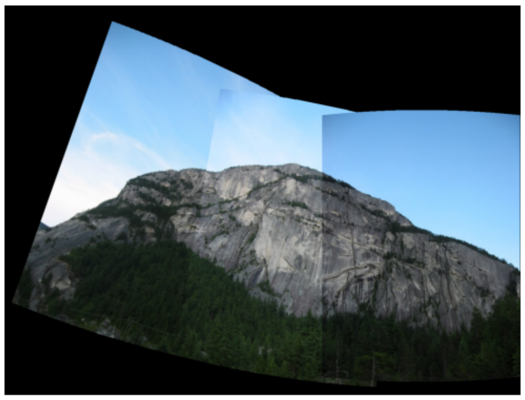

Fig. 6: Stitched Image Result Using Traditional Approach

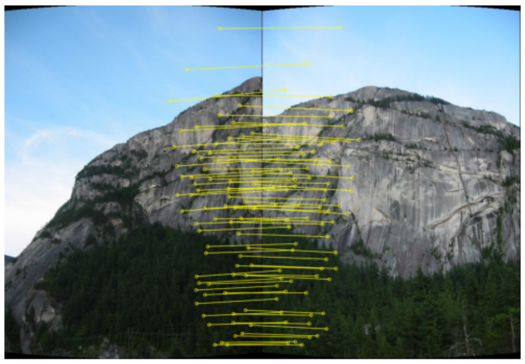

Fig. 5: Outliers removed with RANSAC

$$H_{4point} = \begin{bmatrix} \Delta u_1 & \Delta v_1 \\ \Delta u_2 & \Delta v_2 \\ \Delta u_3 & \Delta v_3 \\ \Delta u_4 & \Delta v_4 \end{bmatrix} \tag{2}$$

The input to the UnsupervisedNet are two patches P_A and P_B stacked depthwise along with the image A I_A and Corners of Patch A C_A.

A. Data Generation

To generate a large dataset with known Homography is difficult. Therefore we use the MSCOCO dataset which contains images of a lot of objects and natural scenery. The following steps were performed for Dataset Generation

1) Randomly crop image to a patch size of 128x128. This is patch A P_A.

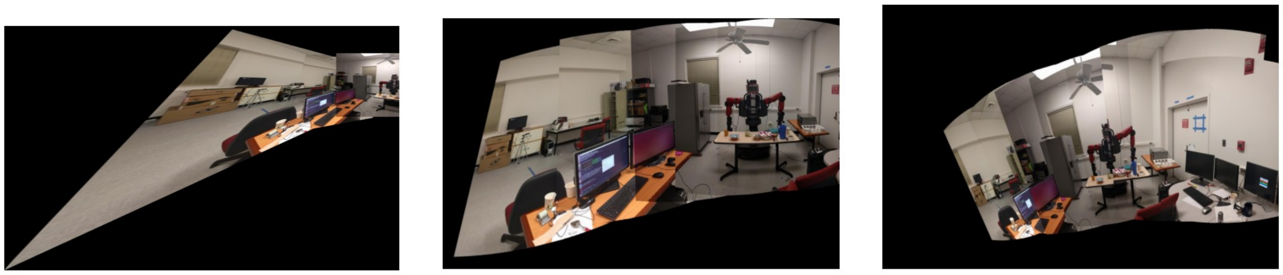
Fig. 7: Non-Cylindrical (Leftmost Image) vs Cylindrical Stitching.

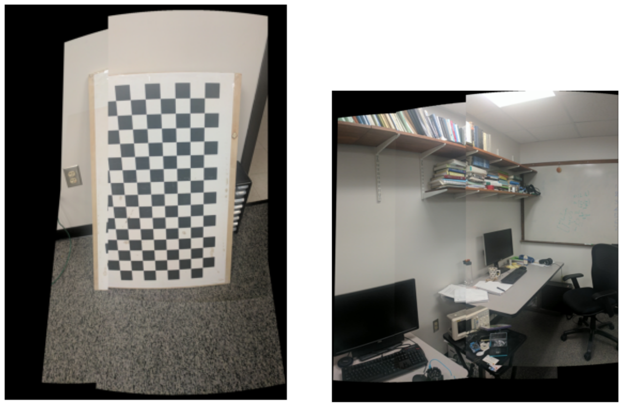
Fig. 8: Phase 1 Test Results

2) Randomly add some perturbations in the range of $[-\rho,\rho]$ to the four corners of patch A. (Chose $\rho = 32$). This gives us corners B C_B of Patch B
3) Compute H_{AB} given the correspondences of P_A and P_B.
4) Get H_{BA} using by getting the inverse of matrix H_{AB}.
5) Apply H_{BA} on image to generate Warp Image.
6) Obtain P_A from image and P_B from warped image.
7) Stack P_A and P_B depth-wise. This will be the input to our model.
8) Calculate Labels H_{4point} using Eqn. (2).

B. *Supervised Net*

In the supervised approach, we use the network architecture proposed in the paper by DeTone Et al. [3]. This architecture is shown in the in Figure 9. The input to the Architecture is two patches P_A and P_B stacked depth-wise. The network has eight convolutional layers. Each Convolutional layer has a ReLU Layer and a Batchnorm2D layer following it. Max pooling is done after every two convolutional layers. The output of the last convolutional layer is passed through two fully connected layers. The network output is an 8 vector which is H_{4point} matrices. The model is trained using a mini-batch size of 128 over 50 epochs using an SGD optimizer with a learning rate of 0.05 and a momentum of 0.9. The loss function used is the mean squared error function.

C. *Unsupervised Net*

For the Unsupervised portion, the first part of the neural network is the Supervised Net. This gives us the H_{4point} prediction from P_A to p_B The Unsupervised Net has two extra steps as compared to the Supervised Net. This is because here we do not have the true labels for $H_{4points}$. Thus we need to reconstruct P_B using P_A and the predicted $H_{4points}$. The first step is the Tensor Linear Direct Transformation (DLT). It takes H_{4point} matrix and the corner points of (P_A) and computes the 3x3 Homography matrix which transforms (P_A) to (P_B). This Homography matrix is then used with the $kornia.geometry.transform.warp\_perspective$ function to warp (P_A) to get a prediction of (P_B), denoted as $(\tilde{P_B})$. L1 loss is then calculated between this predicted between $(\tilde{P_B})$ and the actual patch (P_B).

However, in our implementation, the model starts with low L1 loss values and stays roughly around the same value over as many epochs as we run, indicating that the model does not learn anything. This model is also trained over 50 epochs with a batch size of 128. The AdamW optimizer is used with a learning rate of $1e^{-4}$, β_1 value of 0.9, β_2 value of 0.99, and ϵ value of $1e^{-8}$.

The issue is that the very first batch of $H_{4points}$ matrices we get, has these matrices with very small values. The values are so small that the DLT calculates the 3 x 3 Homography matrices to be almost *Identity* (which is logical). This means that the $(\tilde{P_B})$ is almost always going to be identical to (P_A),

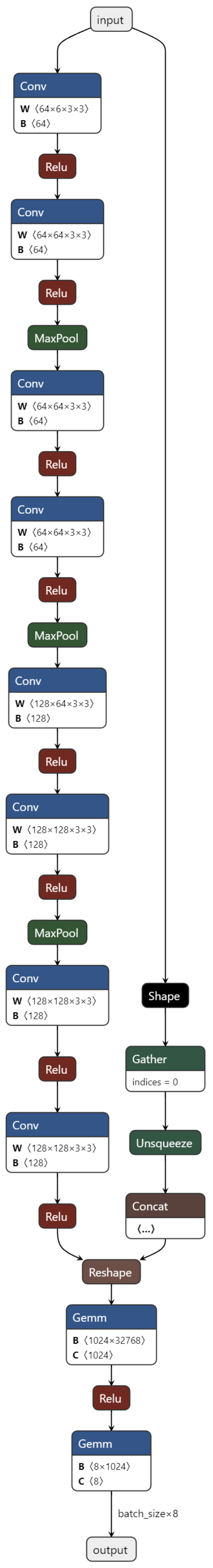

Fig. 9: Supervised Network Architecture

since applying an 'almost' identity matrix to (P_A) is not going to change it much.

We have tried to train the model with different parameter value combinations for the Learning Rate, Batch Size, and number of patches used. We also tried calculating the math for the DLT manually by taking some arbitrary values and comparing them with the formulas shown in the paper for the DLT network. We further verified our DLT implementation by comparing the 3 x 3 matrices we got from it with the ones we got from $kornia.geometry.transform.get\_perspective\_transform$ from function, and they were identical. We then proceeded to verify to change the loss function to check if it made a difference, but unfortunately, it did not.

D. Results and Discussion

The table below shows the results of the training, validation, and testing of the Supervised and Un-supervised approaches.

	Supervised			**Unsupervised**		
	Train	**Val**	**Test**	**Train**	**Val**	**Test**
EPE	5.10	15.58	15.70	0.223	0.222	0.224
time (min)	7.46			17.55		

TABLE I: Results of Supervised and Unsupervised Learning for Homography Estimation

The Fig 10 shows the patches P_A, P_B, P'_B displayed on the test image for both Supervisde and Un-Supervised Learning. P_A is shown in Blue, P_B is shown in Red and P'_B is shown in green. We can see from Fig. 10 that the predicted P'_B for the Supervised Model is similar to the actual P_B. On the other hand, the Un-Supervised Model has a large error where P'_B is more similar to P_A.

Fig 11 and Fig 12 show the stitching using the H estimate by the supervised and unsupervised Models. The supervised model works better when stitching a lower number of images as shown in Fig 13. After that, the error in the H estimate accumulates leading to incorrect stitching.

Fig 15 shows the Supervised (Right) and Unsupervised Test results for the Trees and Towers Datasets.

References

[1] R. . R. Perception and M. Learning. (2024) Rbe 549 spring 2024 - project 1. [Online]. Available: https://rbe549.github.io/spring2024/proj/p1/
[2] R. Kedia. (2024) Panorama stitching. [Online]. Available: https://www.scribd.com/document/510892500/Panorama-Stitching
[3] D. DeTone, T. Malisiewicz, and A. Rabinovich, "Deep image homography estimation," *arXiv*, Jun. 2016.

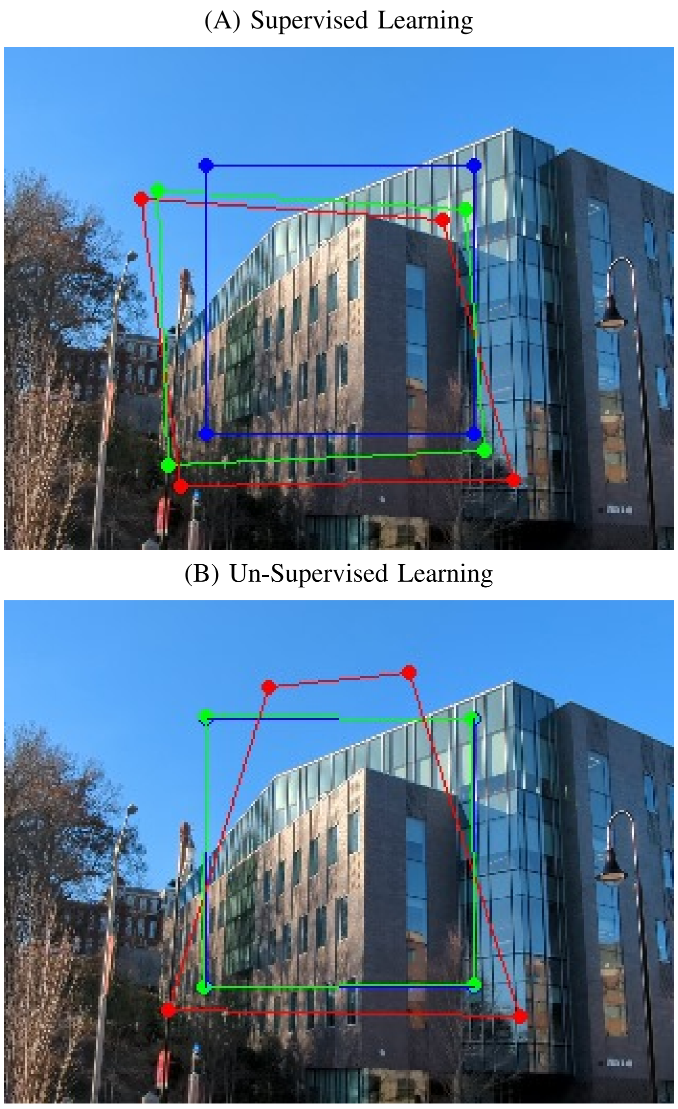

Fig. 10: Image shows the computed points using H estimated in green. Patch A is shown in Blue and Patch B is shown in Red.

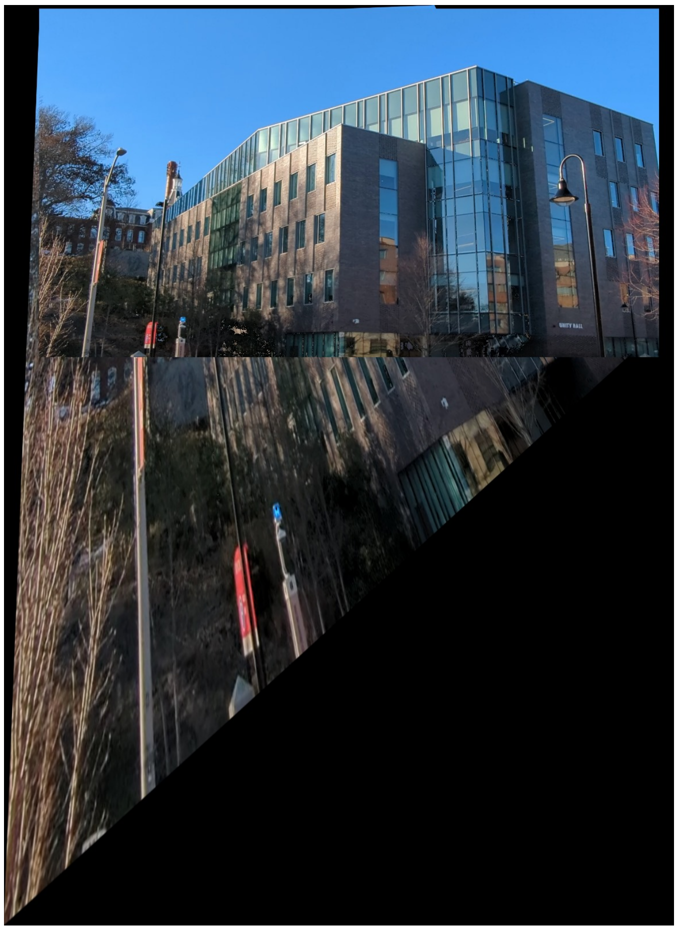

Fig. 11: Stitching performed using H estimate of Supervised Model (Unity Hall)

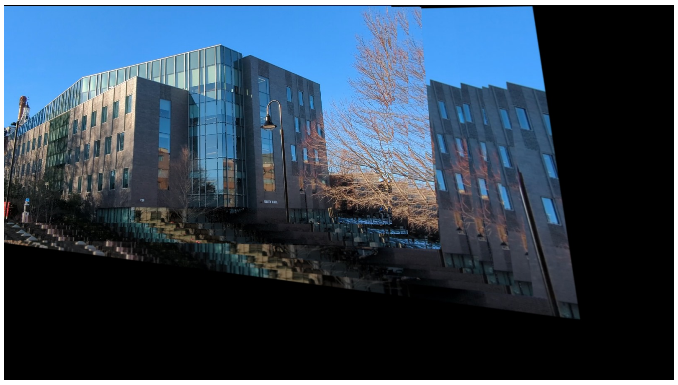

Fig. 12: Stitching performed using H estimate of UnSupervised Model (Unity Hall)

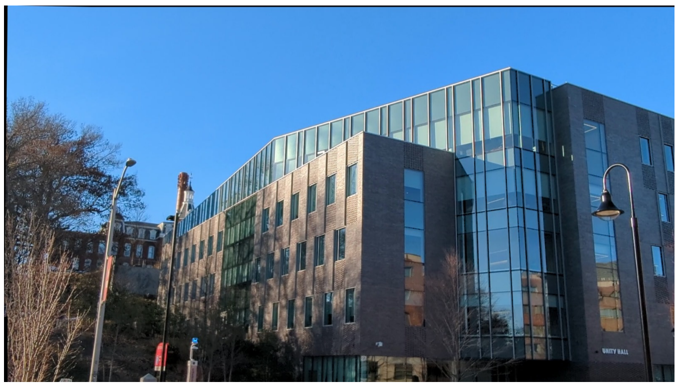
Fig. 13: Stitching performed using H estimate of Supervised Model between 2 images

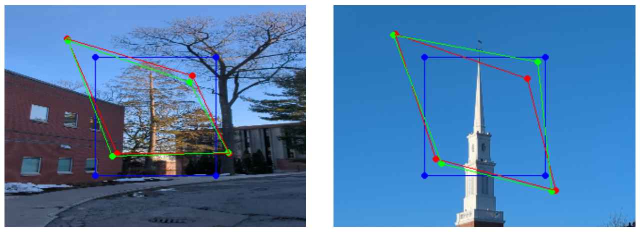
Fig. 14: Visualization of H estimate on Test Set)

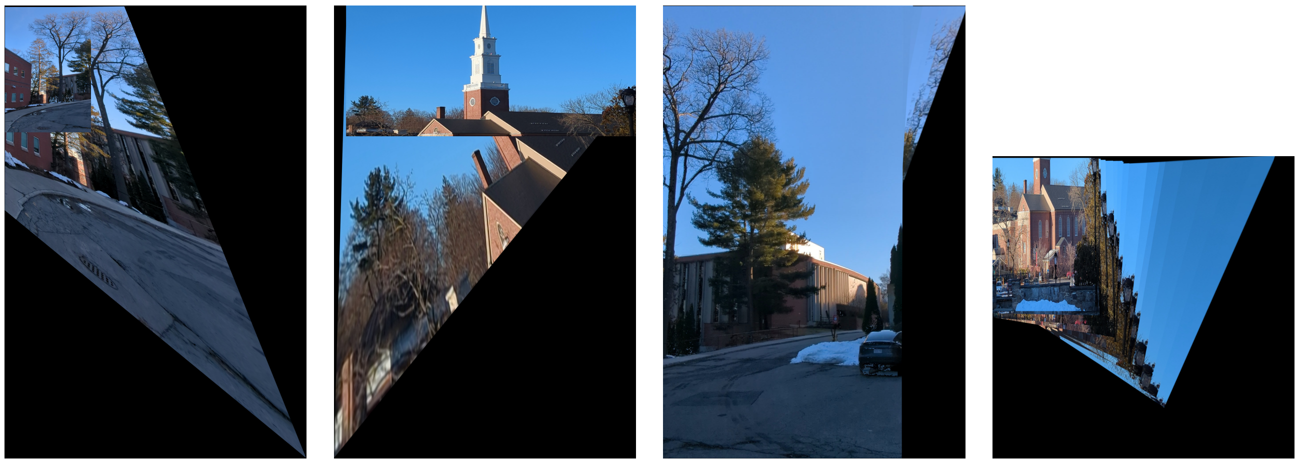
Fig. 15: Phase 2 Test Results for Trees and Tower Dataset. (Left two images are supervised and the right two images are unsupervised)