# MyAutoPano
# RBE549 Project 1

**Piyush Thapar, Sumukh Porwal**
MS Robotics Engineering
Worcester Polytechnic Institute
Email: pthapar@wpi.edu, sporwal@wpi.edu

***Abstract*—This report presents an analysis of homography estimation techniques, focusing on traditional feature-matching methods for computing homography between image pairs. It explores the mathematical foundations, implementation details, and evaluation of these approaches, providing insights into their effectiveness in different scenarios.**

## I. PHASE 1: TRADITIONAL APPROACH

In this phase, a traditional approach is employed for the panorama stitching process, which follows a series of sequential steps. A comprehensive illustration of this methodology is provided in Figure 1.

- **Corner Detection:** Identifying key points in the images, which serve as the basis for stitching.
- **Adaptive Non-Maximal Suppression (ANMS):** Selecting a set of strong and evenly distributed corners.
- **Feature Descriptor:** Encoding the characteristics of each corner for matching.
- **Feature Matching:** Establishing correspondences between corners across images.
- **RANSAC:** Removing incorrect matches and estimating a robust transformation (Homography).
- **Blending Images:** Aligning and merging images seamlessly into a panorama.

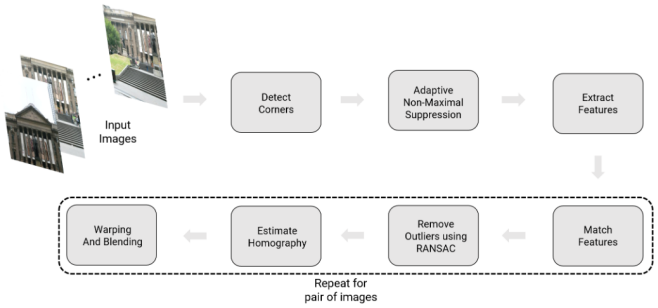


Fig. 1. Overview of Panorama Stitching using the Traditional Method

### A. Corner Detection

The initial step in creating a panorama involves identifying corners, a crucial step in most computer vision tasks. For this project, Harris corners are utilized, implemented using `cv2.cornerHarris`. Corners represent regions in the image with significant variations in intensity in all directions.

The resulting output of corner detection highlights numerous potential corner points. The subsequent step, ANMS, refines these results to retain only the strongest and well-distributed corners.

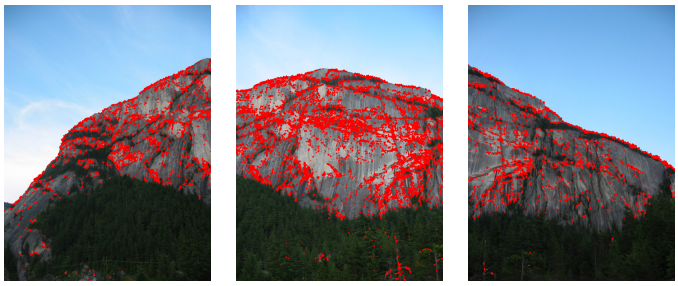

Fig. 2. Corner Detection

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

This step ensures that the selected corners are evenly distributed throughout the image, minimizing distortions during image warping. Since many strong corners may cluster together, ANMS filters them to retain only the most spatially diverse $N_{\text{best}}$ corners. This prevents artifacts that could degrade the panorama quality.

The ANMS algorithm is described below in detail:

**Algorithm 1** Adaptive Non-Maximal Suppression (ANMS)

**Require:** $C_{\text{img}}$: Corner score image, $N_{\text{best}}$: Number of required corners.
**Ensure:** $(x_i, y_i)$ for $i = 1 : N_{\text{best}}$

```
Identify all local maxima in C_img using maximum_filter from scipy.
Retrieve (x, y) coordinates of these local maxima.
Initialize r_i = ∞ for each strong corner i.
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
            Compute Euclidean distance: ED = (x_j − x_i)^2 + (y_j − y_i)^2.
        if ED < r_i then
            Update r_i = ED.
Sort corners by r_i in descending order and select the top N_best.
```

The output of ANMS demonstrates a more uniform and spatially balanced set of corners compared to raw corner detection results.

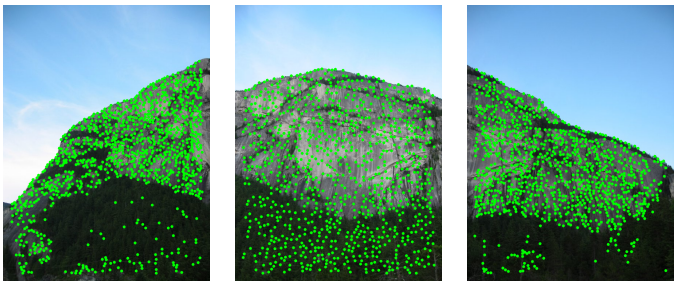

Fig. 3. ANMS with 1000 Best Points

### C. Feature Descriptor

After identifying feature points through ANMS, each point is described by a feature vector that encodes local image characteristics. For this process:

- A patch of size $41 \times 41$ is extracted around each feature point.
- Gaussian blur (`cv2.GaussianBlur`) with kernel size $(5, 5)$ is applied to smooth the patch.
- The smoothed patch is subsampled to $8 \times 8$ and reshaped into a $64 \times 1$ vector.
- This vector is standardized to have zero mean and unit variance, improving robustness to illumination changes.

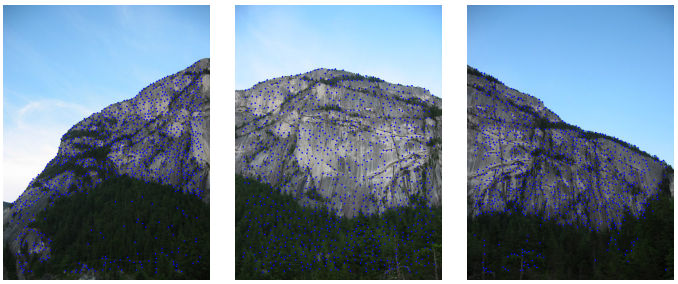

Fig. 4. Feature Descriptors

### D. Feature Matching

With the feature descriptors calculated, corresponding points between two images are matched:

- For each feature in image 1, the sum of squared differences (SSD) is computed with all features in image 2.
- The ratio of the best match to the second-best match is evaluated. Matches with a ratio below a threshold are retained as confident correspondences.
- This process reduces mismatches, ensuring reliable points for Homography estimation.

### E. RANSAC for Outlier Rejection and Robust Homography Estimation

Even with feature matching, some correspondences may still be incorrect. To address this, the RANSAC algorithm is used to robustly estimate the Homography matrix, rejecting outliers:

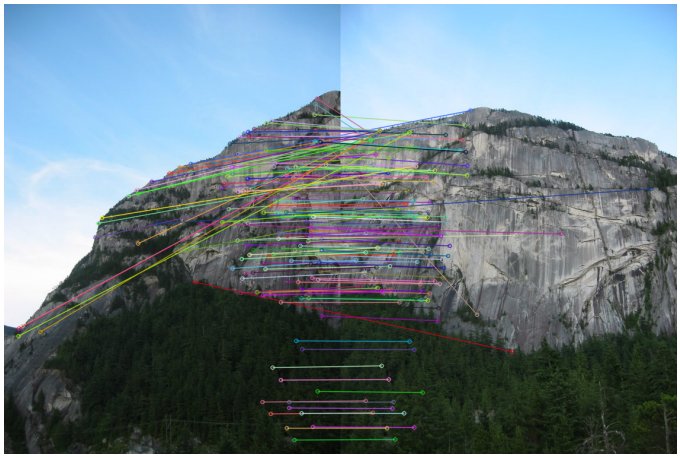

Fig. 5. Feature Matching

**Algorithm 2** RANSAC (RANdom SAmple Consensus)

**Require:** $N_{\text{max}}$: Maximum iterations, $\tau$: Threshold for inliers, $p_i$: Points from Image 1, $p_i'$: Points from Image 2.

**Ensure:** $\hat{H}$: Homography matrix.

1: **while** iterations $< N_{\text{max}}$ **and** percentage of inliers $< 90\%$ **do**
2: Randomly sample four point pairs $(p_i, p_i')$.
3: Compute homography $H$ using the sampled pairs.
4: Determine inliers satisfying $SSD(p_i', H \cdot p_i) < \tau$.
5: Increment iterations.
6: Retain the largest set of inliers and recompute $\hat{H}$ using least-squares.

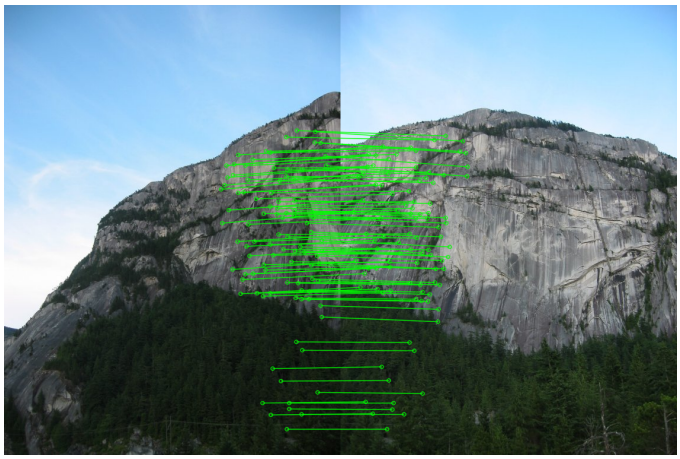

Fig. 6. Inliers after RANSAC

### F. Blending Images

The final panorama is created by aligning and merging the images:

- Compute the bounds of the resulting panorama using the warped image corners.
- Apply a translation matrix to ensure all content fits within the frame.

- Merge the aligned images to produce the seamless panorama.

To Improve blending between the common region of images, we computed weighted masks for each image separately. These masks were normalized to have values between 0 and 1. The final blended image was obtained by taking a weighted sum of the two images, where each pixel's contribution was determined by its corresponding weight in the mask. This approach ensures that overlapping regions blend smoothly, reducing artifacts and visible seams.

Fig. 7. Blended Images

### G. Multiple Image Stitching

For horizontal image stitching involving multiple images, we ensure that the images are ordered optimally based on their matching features. To determine the best stitching order, we use the following algorithm:

1) **Finding Best Matching Pairs**: Each pair of images is analyzed to find the number of good feature matches. Corner detection and feature extraction are performed using methods such as Adaptive Non-Maximal Suppression (ANMS) and feature descriptors. Matches are calculated for all pairs, and the results are sorted in descending order based on the number of matches.
2) **Creating a Panorama Graph**: A graph is constructed where each image represents a node, and edges between nodes represent feature matches. The weight of each edge corresponds to the number of matches between the two images, enabling a structured representation of image relationships.
3) **Finding the Optimal Stitching Order**: Using a modified depth-first search (DFS), the graph is traversed starting from a central or selected image. Neighbors are prioritized based on edge weights, ensuring that images with the strongest matches are stitched first. This approach minimizes errors in alignment and blending.

The algorithm ensures that images are stitched in an order that maximizes feature alignment and reduces distortion. Stitching then proceeds as follows: from the leftmost image to the center (iterating forward) and from the rightmost image to the center (iterating backward). The resulting stitched images are blended seamlessly.

When the number of images is fewer than or equal to four, a straightforward sequential stitching approach from left to right is employed. For five or more images, this optimized ordering algorithm is used to enhance the stitching quality. The technique performs well for up to six images; beyond this, perspective distortions may occur due to the limitations of the perspective transform.

### H. Handling Very Little/No Overlap

To ensure high-quality image alignment, we rejected images with no or very little overlap. Specifically, if multiple inliers were found to have the same destination keypoints, they were skipped as they likely resulted from incorrect feature matching. Additionally, if the number of inliers after filtering was less than 10, the image was discarded. Furthermore, if the homography matrix can't be computed, the image was excluded from the final panorama to prevent misalignment and artifacts.

### I. Final Results from Train and Test Sets

This Section includes final outputs for all the Train and Test Cases. We were able to successfully stitch all the images for all the Train and Test cases.

Fig. 8. Train Set 1 Panorama Image

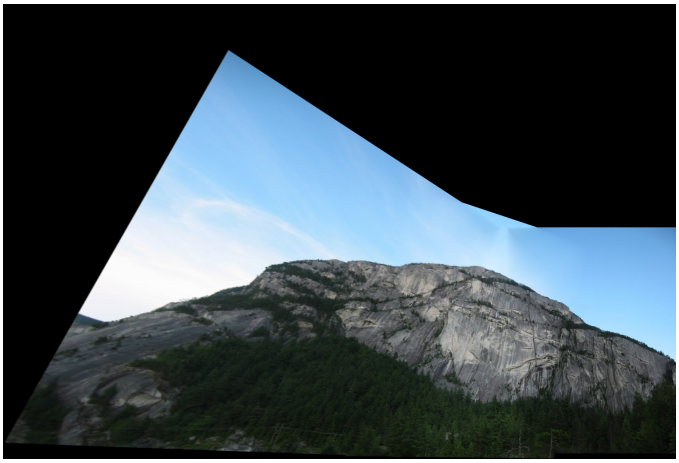

Fig. 9. Train Set 2 Panorama Image

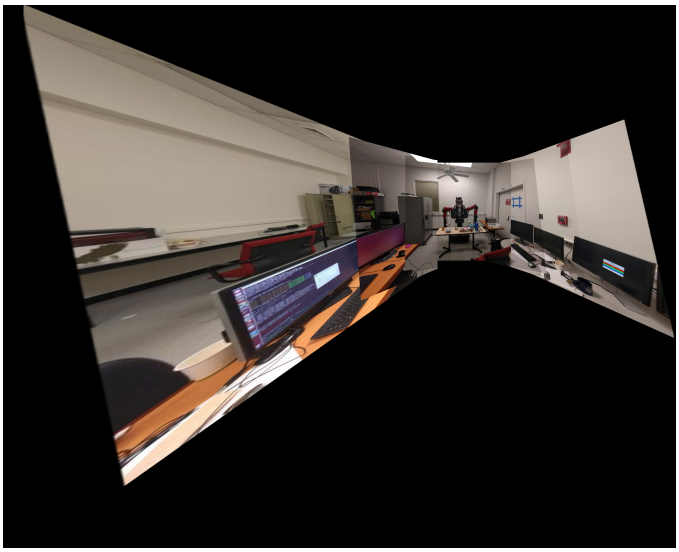

Fig. 10. Train Set 3 Panorama Image

Fig. 11. Train CustomSet 1 Panorama Image

Fig. 12. Train CustomSet 2 Panorama Image

Fig. 13. Test Set 1 Panorama Image

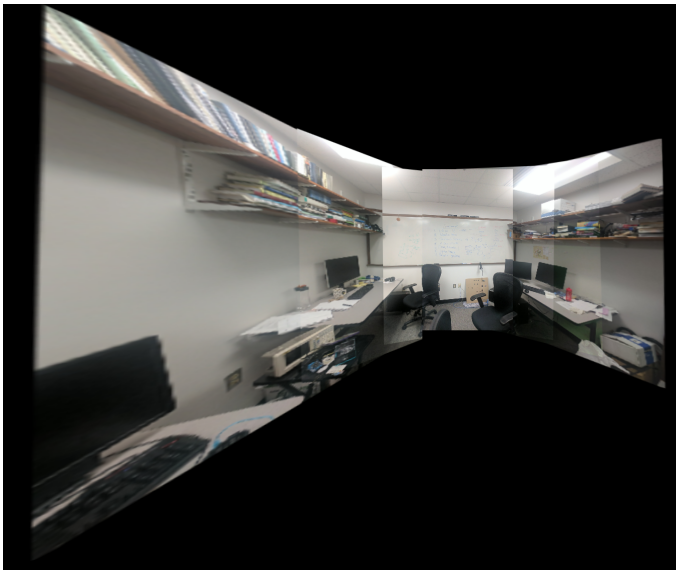

Fig. 14. Test Set 2 Panorama Image

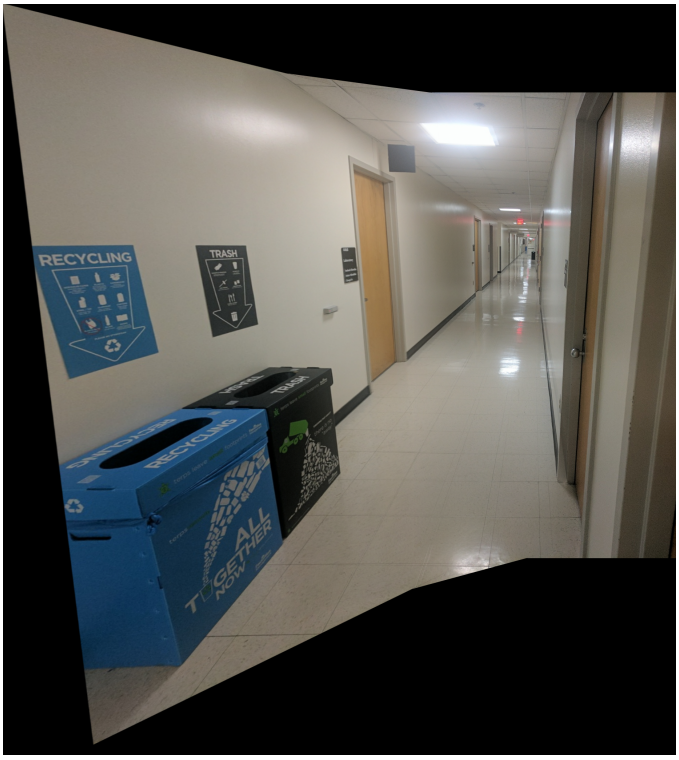


Fig. 15. Test Set 3 Panorama Image

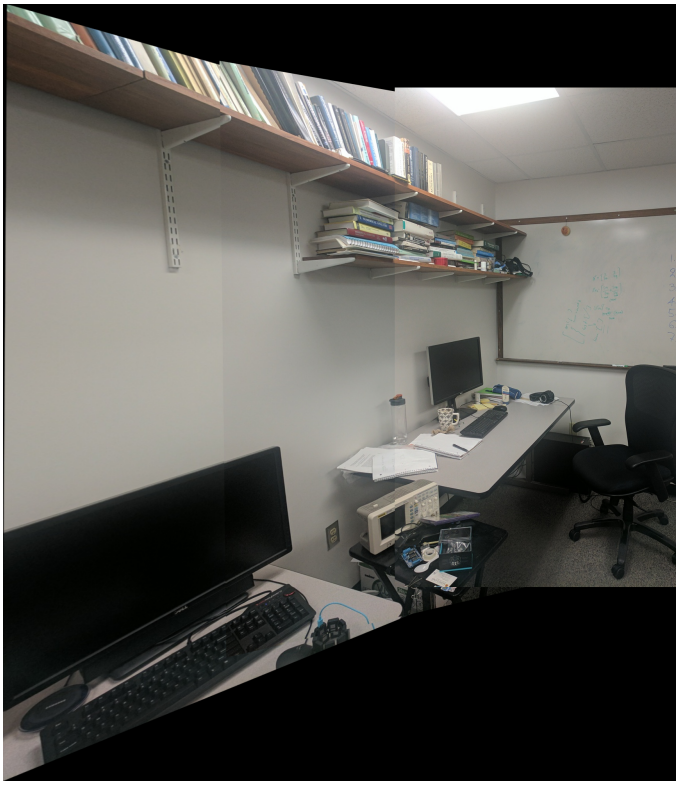

Fig. 16. Test Set 4 Panorama Image

## II. PHASE 2: DEEP LEARNING APPROACH

We implemented two deep learning approaches to estimate the homography between two images. The deep learning model effectively combines corner detection, ANMS, feature extraction, feature matching, RANSAC and estimate homography all into one.

### A. Data Generation

To train a Convolutional Neural Network (CNN) to estimate homography between a pair of images , we need data (pairs of images) with the known homography between them. We generate synthetic pairs of images to train a network. Below are the steps followed to generate data:

1) Obtain a random patch ($P_A$ of size $M_P \times N_P$) from the image ($I_A$ of size $M \times N$) with $M > M_P$ and $N > N_P$ such that all the pixels in the patch will lie within the image after warping the random extracted patch with maximum possible perturbation is in $[-\rho, \rho]$. In our case, $M = 640$, $N = 480$, $M_p = N_p = 128$ and $\rho = 32$
2) Performed a random perturbation in the range $[-\rho, \rho]$ to the corner points (right bottom corner, left bottom corner, top right corner, and top left corner) of $P_A$ in $I_A$.
3) Using the value of $H_A^B$ to warp $I_A$ and obtain $I_B$. Also used `cv2.warpPerspective` to implement this part. After this, extracted the patch $P_B$ using the corners in $C_A$.
4) We generated labels (ground truth homography between the two patches) as $H_A^B$. However regressing the 9 values of the homography matrix directly yielded bad results. Instead, another way was to regress the amount the corners of the patch $P_A$ denoted by $C_A$ need to be moved so that they are aligned with $P_B$. This is denoted by $H4Pt$ and was used as our labels. Remember, $H4Pt$ is given by $H4Pt = C_B - C_A$.
5) Now, we stack the image patches $P_A$ and $P_B$ depthwise to obtain an input of size $M_P \times N_P \times 2$ where 2 is the number of channels in each patch which is a grayscale image.

Example synthetic pairs of images to train and validate our model are shown below.

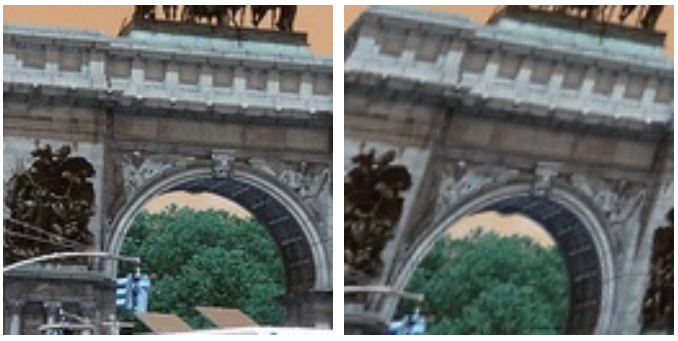

Fig. 17. Synthetic Dataset - Sample 1

Fig. 18. Synthetic Dataset - Sample 2

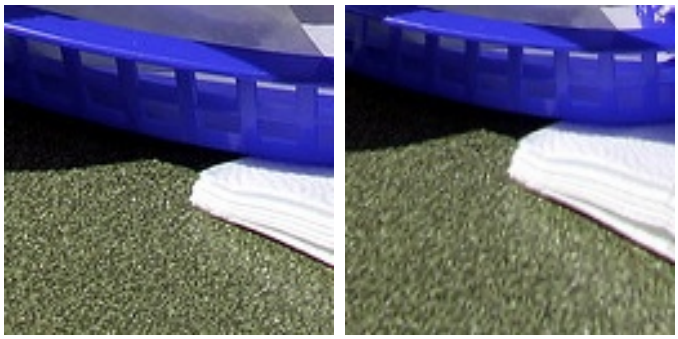

Fig. 19. Synthetic Dataset - Sample 3

Fig. 20. Synthetic Dataset - Sample 4

### B. Supervised Approach

The overview of Supervised Approach for predicting Homography between 2 images is figure 21.

*1) Architecture:* The Supervised Homography Network simultaneously performs homography estimation and feature extraction through a fully convolutional design, utilizing an architecture similar to VGG Net. While effective, this approach may struggle in scenarios involving significant occlusions or repetitive textures. To address these challenges and enhance accuracy, supervised learning techniques leverage ground truth data to provide crucial guidance.

A common training strategy involves using labeled datasets containing image pairs with precomputed $H4Pt$ values from the data generation phase. The network is optimized to directly predict the known $H4Pt$ and loss function is L2 loss between predicted and ground truth $H4Pt$, thereby ensuring a more focused learning process. The architecture is illustrated in

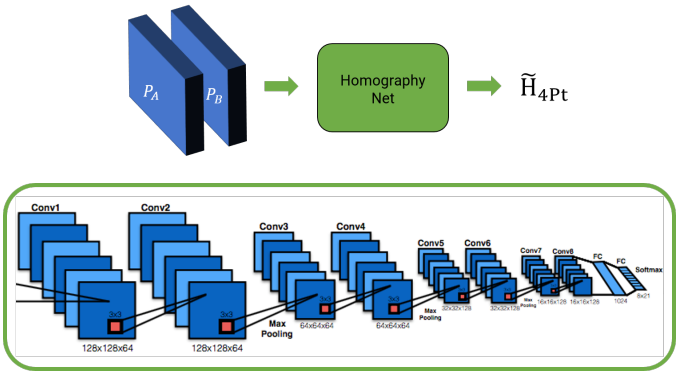


Fig. 21. Overview of the Supervised Approach

Figure 22 and 37.

```
==========================================================================
Layer (type:depth-idx)                   Output Shape              Param #
==========================================================================
SupNet                                   [128, 8]                  5,520
├─Sequential: 1-1                        [128, 128, 8, 8]          --
│    └─Conv2d: 2-1                       [128, 64, 128, 128]       1,216
│    └─BatchNorm2d: 2-2                  [128, 64, 128, 128]       128
│    └─ReLU: 2-3                         [128, 64, 128, 128]       --
│    └─Conv2d: 2-4                       [128, 64, 128, 128]       36,928
│    └─BatchNorm2d: 2-5                  [128, 64, 128, 128]       128
│    └─ReLU: 2-6                         [128, 64, 128, 128]       --
│    └─MaxPool2d: 2-7                    [128, 64, 64, 64]         --
│    └─Conv2d: 2-8                       [128, 64, 64, 64]         36,928
│    └─BatchNorm2d: 2-9                  [128, 64, 64, 64]         128
│    └─ReLU: 2-10                        [128, 64, 64, 64]         --
│    └─Conv2d: 2-11                      [128, 64, 64, 64]         36,928
│    └─BatchNorm2d: 2-12                 [128, 64, 64, 64]         128
│    └─ReLU: 2-13                        [128, 64, 64, 64]         --
│    └─MaxPool2d: 2-14                   [128, 64, 32, 32]         --
│    └─Conv2d: 2-15                      [128, 128, 32, 32]        73,856
│    └─BatchNorm2d: 2-16                 [128, 128, 32, 32]        256
│    └─ReLU: 2-17                        [128, 128, 32, 32]        --
│    └─Conv2d: 2-18                      [128, 128, 32, 32]        147,584
│    └─BatchNorm2d: 2-19                 [128, 128, 32, 32]        256
│    └─ReLU: 2-20                        [128, 128, 32, 32]        --
│    └─MaxPool2d: 2-21                   [128, 128, 16, 16]        --
│    └─Conv2d: 2-22                      [128, 128, 16, 16]        147,584
│    └─BatchNorm2d: 2-23                 [128, 128, 16, 16]        256
│    └─ReLU: 2-24                        [128, 128, 16, 16]        --
│    └─Conv2d: 2-25                      [128, 128, 16, 16]        147,584
│    └─BatchNorm2d: 2-26                 [128, 128, 16, 16]        256
│    └─ReLU: 2-27                        [128, 128, 16, 16]        --
│    └─MaxPool2d: 2-28                   [128, 128, 8, 8]          --
├─Dropout: 1-2                           [128, 128, 8, 8]          --
├─Sequential: 1-3                        [128, 8]                  --
│    └─Linear: 2-29                      [128, 1024]               8,389,632
│    └─ReLU: 2-30                        [128, 1024]               --
│    └─Dropout: 2-31                     [128, 1024]               --
│    └─Linear: 2-32                      [128, 8]                  8,200
==========================================================================
Total params: 9,033,496
Trainable params: 9,033,496
Non-trainable params: 0
```

Fig. 22. Summary of the Supervised Network

*2) Training Parameters:* For training the homography model, we employ the Stochastic Gradient Descent (SGD) optimizer with a learning rate of 0.005 and a momentum of 0.9. The model is trained with a minibatch size of 64 for a total of 20 epochs. Additionally, a learning rate scheduler, `torch.optim.lr_scheduler.StepLR`, is utilized with a step size of 4 and a decay factor of 0.1 to adjust the learning rate dynamically during training.

*3) Results:* Figure 23 presents the training and validation loss across epochs, while Table I summarizes the average End-Point Error (EPE) for the supervised model. The algorithm run-time for forward pass of the network after the graph has been initialized is **1.1 ms**.

| Dataset | Avg EPE |
|---|---|
| Train | 33.8836 |
| Validation | 31.4324 |
| Test | 28.2562 |

TABLE I
AVERAGE EPE FOR THE SUPERVISED MODEL

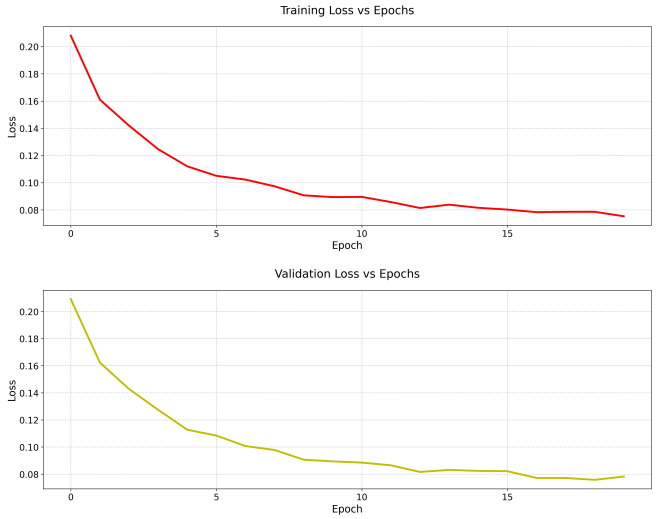


Fig. 23. Training and Validation Loss vs Epochs for Supervised Approach

## C. Unsupervised Approach

The Unsupervised Approach for predicting homography between two images is summarized in Figure 24.

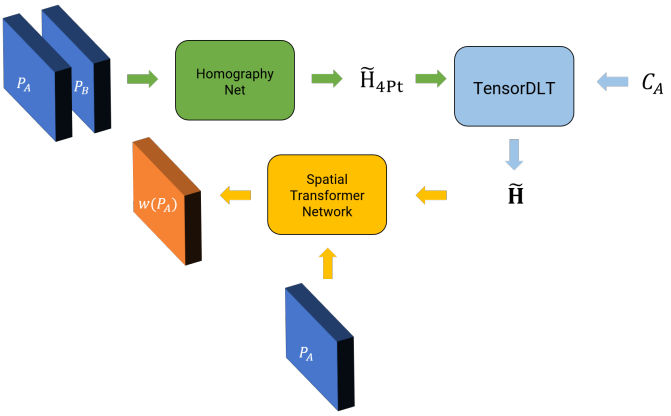


Fig. 24. Overview of the Unsupervised Approach

*1) Architecture:* The Unsupervised Homography Network integrates homography estimation and feature extraction within a fully convolutional design, leveraging an architecture akin to VGG16. The training process involves estimating $H4Pt$, which is subsequently converted into a homography matrix using Tensor Direct Linear Transform (DLT). This computed homography matrix is then utilized to warp image $I_A$, from which a predicted patch $B$ is extracted. The model is trained by computing an L1 loss between the predicted and actual patch $B$, eliminating the need for labeled data or ground truth values.

The architecture is depicted in Figure 25 amd 38.

```
==========================================================================================
Layer (type:depth-idx)                   Output Shape              Param #
==========================================================================================
UnSupNet                                 [128, 4, 2]               --
├─Sequential: 1-1                        [128, 128, 16, 16]        --
│    └─Conv2d: 2-1                       [128, 64, 128, 128]       1,216
│    └─ReLU: 2-2                         [128, 64, 128, 128]       --
│    └─Conv2d: 2-3                       [128, 64, 128, 128]       36,928
│    └─ReLU: 2-4                         [128, 64, 128, 128]       --
│    └─MaxPool2d: 2-5                    [128, 64, 64, 64]         --
│    └─Conv2d: 2-6                       [128, 64, 64, 64]         36,928
│    └─ReLU: 2-7                         [128, 64, 64, 64]         --
│    └─Conv2d: 2-8                       [128, 64, 64, 64]         36,928
│    └─ReLU: 2-9                         [128, 64, 64, 64]         --
│    └─MaxPool2d: 2-10                   [128, 64, 32, 32]         --
│    └─Conv2d: 2-11                      [128, 128, 32, 32]        73,856
│    └─ReLU: 2-12                        [128, 128, 32, 32]        --
│    └─Conv2d: 2-13                      [128, 128, 32, 32]        147,584
│    └─ReLU: 2-14                        [128, 128, 32, 32]        --
│    └─MaxPool2d: 2-15                   [128, 128, 16, 16]        --
│    └─Conv2d: 2-16                      [128, 128, 16, 16]        147,584
│    └─ReLU: 2-17                        [128, 128, 16, 16]        --
│    └─Conv2d: 2-18                      [128, 128, 16, 16]        147,584
│    └─ReLU: 2-19                        [128, 128, 16, 16]        --
├─Sequential: 1-2                        [128, 8]                  --
│    └─Flatten: 2-20                     [128, 32768]              --
│    └─Dropout: 2-21                     [128, 32768]              --
│    └─Linear: 2-22                      [128, 1024]               33,555,456
│    └─ReLU: 2-23                        [128, 1024]               --
│    └─Dropout: 2-24                     [128, 1024]               --
│    └─Linear: 2-25                      [128, 8]                  8,200
==========================================================================================
Total params: 34,192,264
Trainable params: 34,192,264
Non-trainable params: 0
```

Fig. 25. Summary of the Unsupervised Network

*2) Training Parameters:* For training, the homography model employs the Adam optimizer with a learning rate of 0.0001 and $\beta_1 = 0.9$. The model is trained using a minibatch size of 128 over 50 epochs.

*3) Tensor Direct Linear Transform (Tensor DLT):* A fundamental component of the architecture is the Tensor DLT layer, which converts the four-point parameterization output into a full $3 \times 3$ homography matrix. This layer plays a crucial role in the network's learning process, enabling backpropagation during transformation.

*4) Spatial Transformation Layer:* The Spatial Transformer Layer enhances the network's capability by enabling inverse warping of images. It computes the normalized inverse of the homography matrix to generate a pixel coordinate grid for the warping process. Bilinear interpolation is employed for image sampling, ensuring differentiability for loss function backpropagation. This functionality is implemented using the `kornia.geometry.transform.warp_perspective` function, which supports gradient backpropagation.

*5) Results:* Figure 26 presents the training and validation loss over epochs, while Table II summarizes the average End-Point Error (EPE) for the unsupervised model. The algorithm run-time for forward pass of the network after the graph has been initialized is **1.9 ms**.

| Dataset | Avg EPE |
|---|---|
| Train | 52.8069 |
| Validation | 51.8856 |
| Test | 47.4981 |

TABLE II
AVERAGE EPE FOR THE UNSUPERVISED MODEL

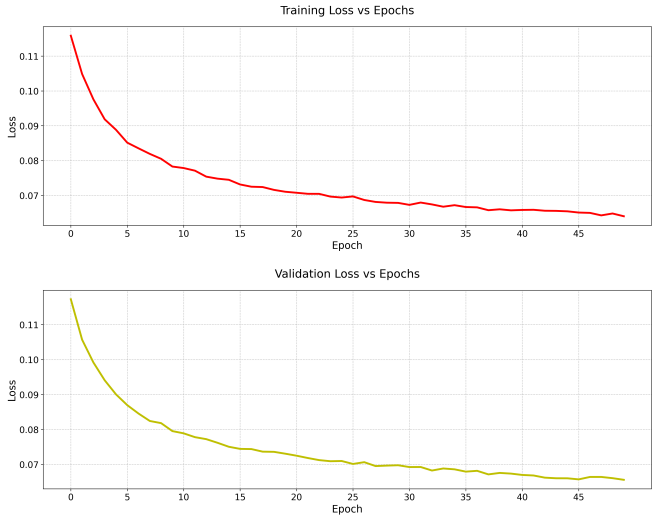


Fig. 26. Training and Validation Loss vs Epochs for Unsupervised Approach

### *D. Stitching Images using Supervised or Unsupervised Approach*

Multiple approaches were explored for stitching images, leveraging both supervised and unsupervised models. Initially, images were resized to 128x128 before being passed through the model for homography prediction. However, since translation was not accurately predicted in either model, feature matching and RANSAC from Phase 1 were used to estimate translation, while the trained model predicted the remaining homography parameters. Despite these refinements, the stitched images lacked visual coherence.

A second approach assumed minimal movement between consecutive frames in a slow-moving video. A 128x128 patch was extracted from the middle-right section of the first image, and the same corner coordinates were used to extract a corresponding patch from the second image. The homography matrix was then predicted between these two patches and applied to the full images. However, this method also failed to produce satisfactory results.

A third and most effective approach in all the strategies we tried was this one, even though results were not good, they are reported below in the results section. involved obtaining the homography matrix from the Phase 1 (Traditional) approach to estimate translation. The images were then resized to 128x128 pixels and provided as input to the trained model for homography prediction. The predicted homography matrix was scaled back to the original image dimensions using the following transformation:

$$S_1 = \begin{bmatrix} s_{x1} & 0 & 0 \\ 0 & s_{y1} & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{1}$$

$$S_2^{-1} = \begin{bmatrix} \frac{1}{s_{x2}} & 0 & 0 \\ 0 & \frac{1}{s_{y2}} & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2}$$

$$H_{original} = S_2^{-1} \cdot H_{predicted} \cdot S_1 \tag{3}$$

Finally, the translation values in $H_{original}$ were replaced with those obtained from the traditional homography matrix before stitching the images. This iterative process continued with each stitched image being merged with the next frame, resulting in significantly improved panorama quality. The final images were saved for further evaluation.

### *E. Results*

The figures 27, 28, 29 and 30 presents classical feature based (from Phase 1), supervised, unsupervised estimated homographies against a synthetic ground truth for various images for Train/Val/Test set.

The figures 31, 32 and 33 presents stitching of Test Set using Supervised Approach.

The figures 34, 35 and 36 presents stitching of Test Set using Unsupervised Approach.

## III. Conclusion

This project consisted of two phases. In phase 1, we used the traditional approaches to detect features, match them, calculate transformation between image frames using the homography matrix and finally stitch/blend them. In phase 2, we used the deep learning approaches to achieve similar results. Overall, deep learning approach proved to be fast and more robust.

## IV. Appendix

Download the Checkpoints from the link below.

- Supervised
- Unsupervised

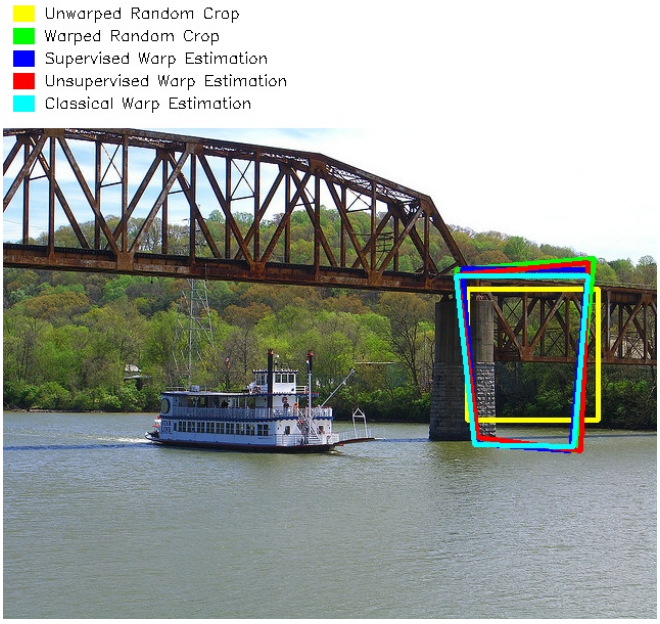


Fig. 27. Train Set Image output of Traditional, Supervised and Unsupervised approaches on Homography estimation comparison


Fig. 28. Val Set Image output of Traditional, Supervised and Unsupervised approaches on Homography estimation comparison

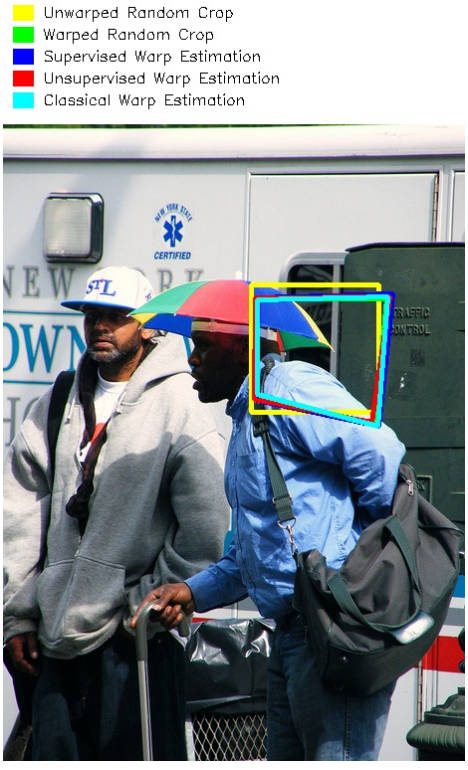


Fig. 29. Test Set Image output of Traditional, Supervised and Unsupervised approaches on Homography estimation comparison


Fig. 30. Test Set Image output of Traditional, Supervised and Unsupervised approaches on Homography estimation comparison

Fig. 31. Stitched Image for Unity Hall Test Set using Supervised Approach

Fig. 32. Stitched Image for Trees Test Set using Supervised Approach

Fig. 33. Stitched Image for Tower Test Set using Supervised Approach

Fig. 34. Stitched Image for Unity Hall Test Set using Unsupervised Approach

Fig. 35. Stitched Image for Trees Test Set using Unsupervised Approach

Fig. 36. Stitched Image for Tower Test Set using Unsupervised Approach

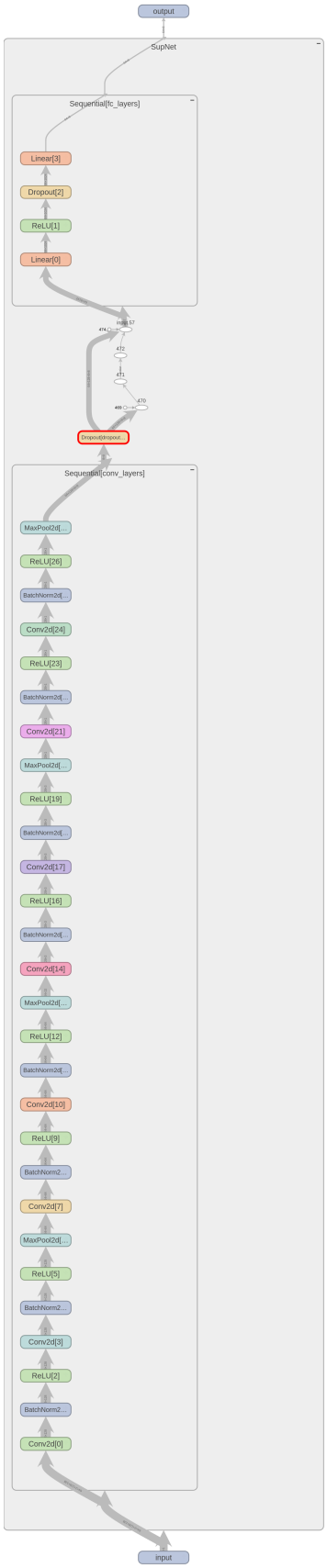


Fig. 37. Supervised Model Architecture

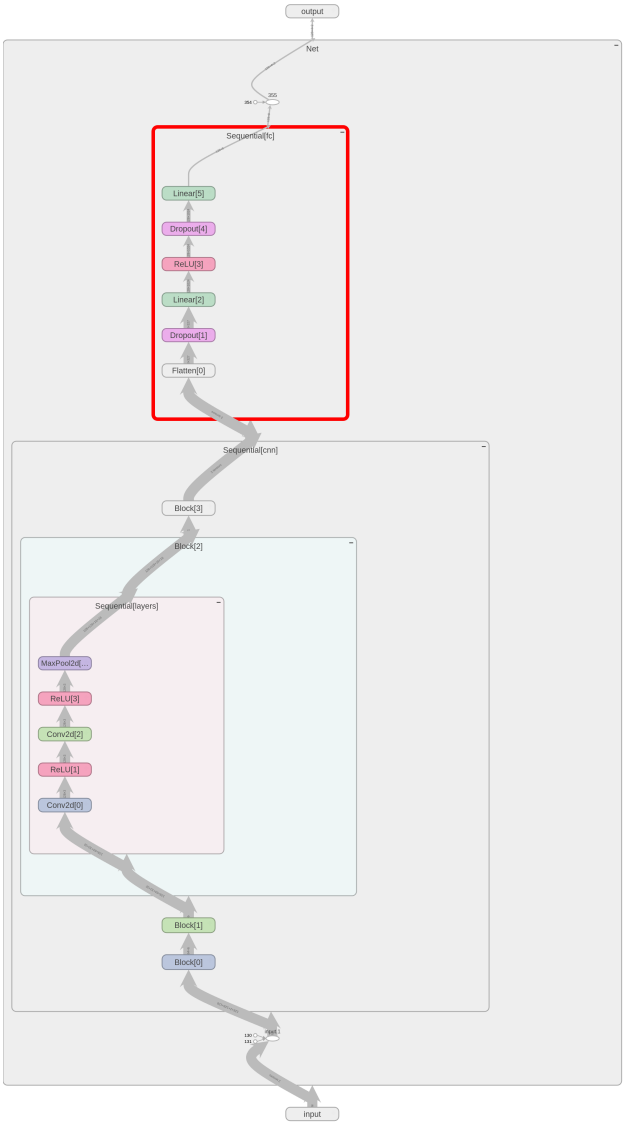


Fig. 38. Unsupervised Model Architecture