RBE 549 P1 : MyAutoPano

Farhan Seliya
Department of Robotics Engineering
Worcester Polytechnic Institute
Email: faseliya@wpi.edu

Sai Hitesh Viswasam
Department of Robotics Engineering
Worcester Polytechnic Institute
Email: sviswasam@wpi.edu

***Abstract*—This report presents a project on multi-image stitching to generate a seamless panoramic image from different viewpoints. In Phase I, we employed a traditional feature-based approach, detecting key points, extracting and matching features, estimating homographies, and applying perspective transformations for image alignment. In Phase II, we explored deep learning-based homography estimation using both supervised and unsupervised methods. Finally, we compare the performance of these three approaches, analyzing their accuracy, robustness, and effectiveness in panorama generation.**

I. Phase I : Traditional Approach

In this phase, we used classical methods in Computer Vision for detecting boundaries in images. This approach involves multiple steps as shown in the following fig.1. Implementation of each of the individual steps is described in the following sections.

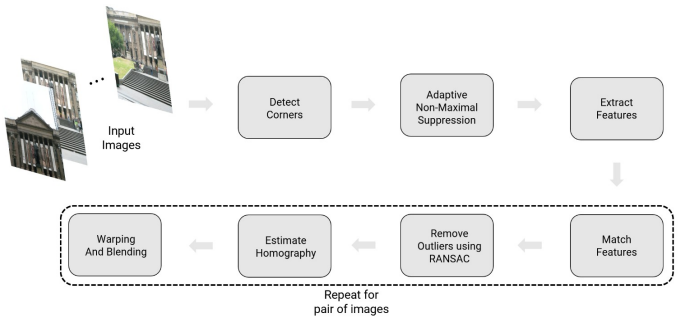


Fig. 1: Panorama stitching using the Traditional Method.

A. Corner Detection

In the first step of the traditional method, we detected all the corners within each component image. This helps us further identify features between each image for matching them eventually. The corners from an image can be obtained easily through the `cv2.cornerHarris()` function with a threshold of 0.001. The result of the corner detection is shown in fig.2

B. Adaptive Non-Maximal Supression (ANMS)

The corners obtained from the Harris corner detection are not spread out evenly across the image as seen above. These corners also include the non-sharp corners from the image as well. Hence, the method of *Adaptive Non-Maximal Suppression (ANMS)* is necessary after corner detection to ensure that the selected points are not only the strongest but also well-distributed across the image, avoiding clustering in regions of high response. The algorithm we followed compared the corner strength of all the corners detected with the local maxima and supresses out the weak corners based on the Euclidian distance from the maxima. This ensures even spread of stronger corners. From the above images, we filtered out the top 500 points based on the corner response. The corners in the images after ANMS is shown in fig.3

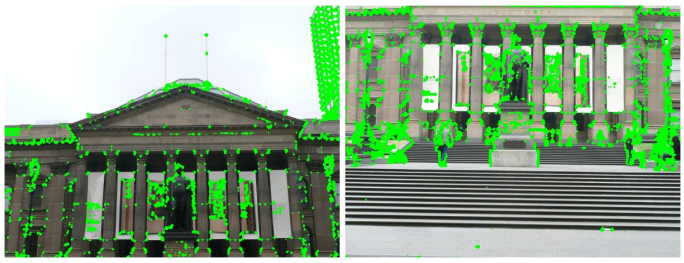


Fig. 2: Corners detected by Harris Corner Detection.

Fig. 3: Corners after ANMS with 500 best corners per image.

C. Feature Extraction

After filtering out the corners, the next step is to define a method to extract the features around each of the corners to compare and match the corners across images. One of the simplest ways to extract a feature encoding from any corner is by applying a simple Gaussian kernel around each corner to blur the image. By further sub-sampling and flattening the output, we would obtain a feature vector that can uniquely describe each corner for matching.

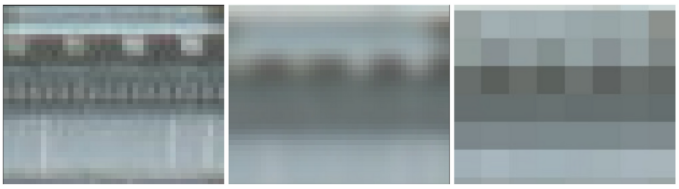


Fig. 4: Feature Extraction process applied for the red corner on the top image. The final patch is then normalised and converted into a 64x1 feature vector.

In our case, we first created a 41x41 sized patch centered around each corner and applied a Gaussian filter of kernel size equal to 3 and the sigma as 1.5. The output obtained form this filter is further subsampled to an 8x8 image, then converted into a feature vector of 64x1,which is standardised to have zero mean and variance of 1.

D. Feature Matching

The next step was to match the obtained features between the images. There are many ways to check feature correspondence between a pair of images, but the method we used is by comparing the Nearest Neighbour Distance Ratio (NNDR) as described in [**?**]. The NNDR (Nearest Neighbor Distance Ratio) is defined as:

$$NNDR = d_1/d_2$$

where d_1 and d_2 represent the Euclidean distances in the feature space between a feature and its first and second nearest neighbors, respectively. A value of NNDR closer to 1 indicates lower confidence in the match. We filtered out the matches with the condition that the NNDR of a pair that is less than 0.75 corresponds to a good match. Repeating this for all the points gives us the matching features and points between the images which can be used to calculate homography further. The results of feature matching are shown in fig.5

Fig. 5: Result of the feature matching algorithm.

E. RANSAC for outlier Rejection

As seen in the fig.5, the matches between the corner points within the images is not entirely perfect, as there are a few inaccuracies as well. To get rid of these outlier matches within the image, we use the RANSAC algorithm. RANSAC estimates the best fit homography possible between both the images based on the points that are sampled. The whole algorithm is described as follows:

Algorithm 1 RANSAC for Homogrpahy Estimation

Require: Feature pairs $\{p_i \leftrightarrow p_i'\}$, maximum iterations $N_{\max}$, threshold τ

Ensure: Estimated homography $\hat{H}$

Initialize $S_{\text{best}} \leftarrow \emptyset$ (Best set of inliers)
Initialize $\hat{H} \leftarrow I$ (Identity matrix)
for $k = 1$ to $N_{\max}$ **do**
    Randomly select four feature pairs $\{(p_i, p_i')\}$
    Compute homography H using the selected pairs
    Determine inliers:

$$S = \{(p_i, p_i') \mid \text{SSD}(p_i', Hp_i) < \tau\},$$

    where SSD is the sum of squared differences
    if $|S| > |S_{\text{best}}|$ **then**
        Update $S_{\text{best}} \leftarrow S$
        Update $\hat{H} \leftarrow H$
    end if
end for
Recompute least-squares estimate $\hat{H}$ using all inliers in S_{best}
return $\hat{H}$, inliers in S_{best}

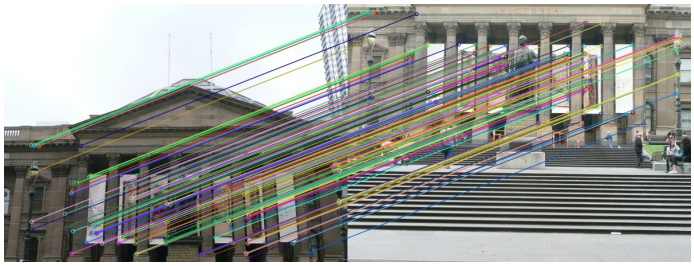


Fig. 6: Feature matches after RANSAC

To implement RANSAC, we considered the threshold value for the maximum Sum of Squares Difference as 5 and we ran the algorithm for 1000 iterations. The better feature matches we obtained after RANSAC are shown in fig.6.

F. Stitching Images for a Panorama

Once the homography matrix is obtained between the images, the stitching can be carried out pairwaise between the images in a scene. For each pair of input images, the homography matrix is estimated and one of the images is warped to align with the perspective of the second image using the homography matrix. This is performed using the `cv2.warpperspective` function on an image and it is combined with the second image so that the common region aligns perfectly.

Fig. 7: Stitching the first pair of Images

Expanding the same approach, we can stitch an entire panorama scene with multiple images by stitching images and the subsequent results pairwise. The stitched output of the first pair is warped with respect to the third image and combined and so on. With this pairwise stitching, the entire scene can be rebuilt as shown in fig. 8,9.

For stitching a panorama from image sets with large number of images, we considered a few assumptions and methods as follows.

- The scene consists of images that are ordered and sequentially taken horizontally from left to right.

Fig. 8: Panorama result of images from Set 1.

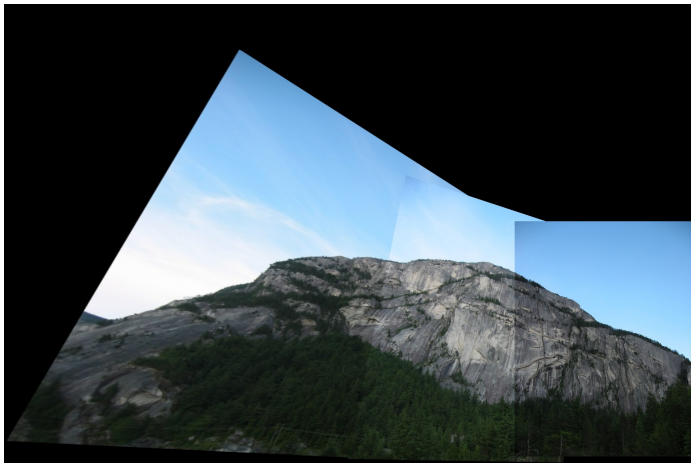
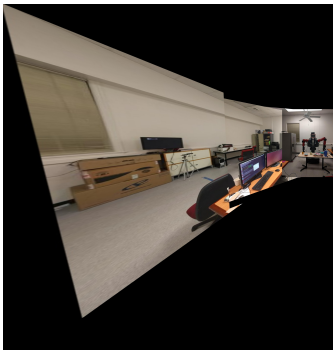


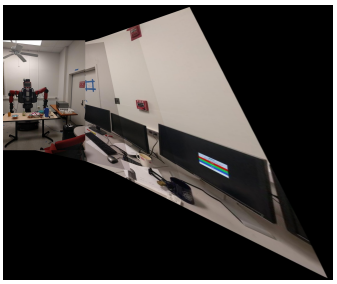
Fig. 9: Panorama result of images from Set 2.

- Images are still taken pairwise only while stitching, but the stitching starts from centre image and expands outward in either the left or right direction and the half panorama is stitched first.
- Similarly, starting from the center again, the other half of the panorama is stitched and finally, both the halves are stitched together to make the whole scene. This is to ensure that the images at the end are warped only to a minimal extent. Sequentially warping images from the start to the end causes the last images to be warped out of proportion, destroying the panorama.
- Some other measures we took were **resizing** the results of each panorama by 0.8 times so as to reduce the size of the entire image, thereby accomodating larger panoramas. Another trick we used was to apply **cylindrical warping** onto the images to cover a larger FOV.

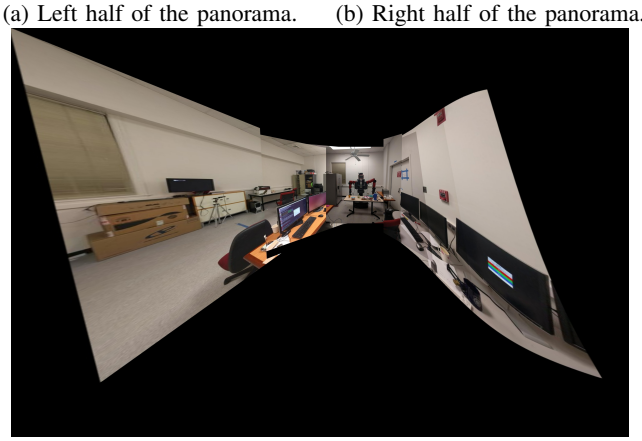
The results of the panoramas by stitching multiple images on the Train Set 3 and two custom sets are shown in the following figures.



(a) Left half of the panorama.



(b) Right half of the panorama.



(c) Final Panorama of Set 3.

Fig. 10: Stitching the images of Set 3 starting from centre to avoid excessive warping of images

Fig. 11: Panorama result of images from Custom Set 1.

Finally, the sets provided can also have images that do not match with the panorama scene. To reject these images from the panorama scene, we check the number of feature matches for these images during the feature matching step itself. If the number of matches are significantly lower than the matches that have been encountered so far, then it is understood that the image does not belong to the panorama scene and the algorithm reports an error for that pair. That image is rejected and then the stitching process continues with the next images in order.

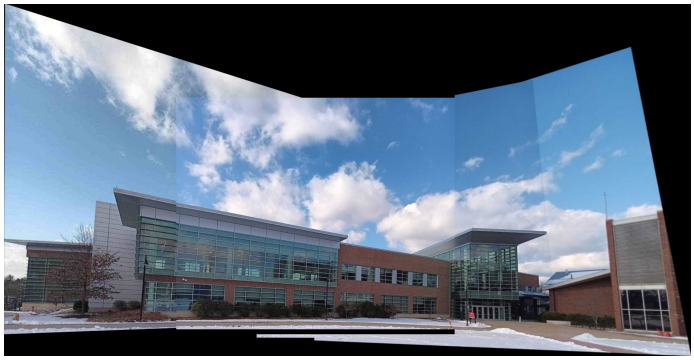


Fig. 12: Panorama result of images from Custom Set 2.

G. Blending the Images

In the image stitching process, blending is essential to ensure a seamless transition in the overlapping regions between the warped image and the second image. This is achieved using OpenCV's `cv2.seamlessClone`, which performs gradient-domain blending to maintain natural transitions in lighting and texture. First, the overlapping region is identified using a binary mask that detects non-zero pixel values in both images. The second image is then positioned in the stitched canvas, and seamless cloning is applied in the overlap area, using the center of this region as the anchor point. This approach effectively eliminates visible seams, ensuring a visually coherent stitched output.

H. Evaluation of Test Images

We used the earlier approach to stitch the panoramas of the images provided in the test sets and the results are shown below. In case of Test Set 1, evaluating corners using ANMS for the checkerboard images took a significant amount of time because of the huge number of corners in the images. To improve the computation time, we reduced the quality of the images and compressed them below 100 KB for the algorithm to work faster. We also reduced the number of corners in the ANMS output from 1000 to 500 and this resulted in faster computation.

In Test Set 4, outlier images unrelated to the panorama scene were successfully identified and excluded by our algorithm. These outliers exhibited significantly fewer feature matches compared to the other images in the set, indicating their incompatibility with the panorama. Conversely, the remaining images demonstrated a comparable number of feature matches, enabling the algorithm to include them in the panorama.

Fig. 13: Panorama result of images from Test Set 1.

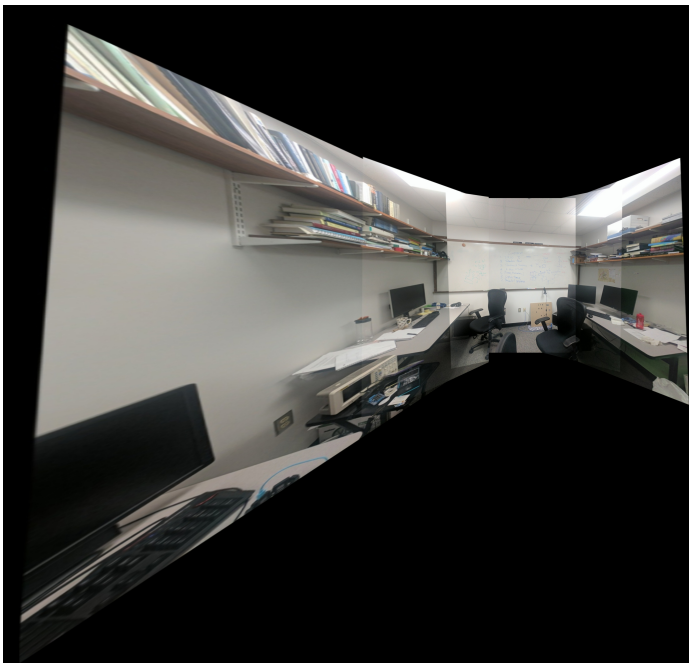


Fig. 14: Panorama result of images from Test Set 2.

Fig. 15: Panorama result of images from Test Set 3.

Fig. 16: Panorama result of images from Test Set 4.

I. Phase II - Deep Learning Approach

To train a Convolutional Neural Network (CNN) for estimating homography between image pairs , we require datasets consisting of image pairs with known homography transformations. However, acquiring such data is challenging, as it necessitates precise knowledge of the 3D motion between image pairs to compute the homography. A more practical alternative is to generate synthetic image pairs for training. To ensure the network remains unbiased, we utilize images from the MSCOCO dataset, which features diverse objects in natural scenes.

A. Data Generation

To construct the test, validation, and training datasets for both supervised and unsupervised models, a patch generation algorithm was implemented. This process involves extracting image patches, applying transformations, and generating ground truth homography labels to facilitate network training.

⇒ **Input:**

- Image I_A of size $M \times N$
- Patch size $M_P \times N_P$
- Perturbation range $[-\rho, \rho]$

⇒ **Output:**

- Stacked patch pairs (P_A, P_B)
- Homography labels H_{4Pt}

⇒ **Algorithm:**

1) **Patch Extraction:** A random patch P_A of size $M_P \times N_P$ is extracted from the original image I_A, ensuring that all pixels remain within the image boundaries after transformation.
2) **Perturbation:** Random perturbations within the range $[-\rho, \rho]$ are applied to the four corners of P_A, along with a uniform random translation to introduce spatial variation.
3) **Homography Computation:** The inverse homography H_{BA} is computed using the perturbed coordinates, and it is applied to I_A to generate the transformed image I_B.
4) **Patch Extraction from Warped Image:** The corresponding patch P_B is extracted from I_B using the original patch coordinates.
5) **Label Generation:** The displacement between the perturbed and original patch corners is computed as $H_{4Pt} = C_B - C_A$, serving as the ground truth for training.
6) **Data Preparation:** The patches P_A and P_B are stacked depthwise, forming an input of size $M_P \times N_P \times 2K$, where K represents the number of image channels (3 for RGB, 1 for grayscale).

⇒ **Dataset Configuration:**

The training images were initially resized to the standard dimensions specified in the Dataset Configuration I. Following this, the patch extraction process was performed on the resized images. This patch generation method ensures that the training data is diverse and well-structured, facilitating the model's ability to learn accurate homography estimation. Below is a sample 1 of the data after the patch generation algorithm was applied. We extract five patch pairs from each training image, resulting in a training set consisting of 25,000 samples.

Parameter	Value
Resized Image Size	(320, 240)
Patch Size	(128, 128)
Perturbation Range	(-32, 32)

TABLE I: Model Parameters and Values

Fig. 1: Left to Right: Original Image, Patch A, Patch B

B. Supervised Approach

Each of the data sample generated contains a pair of 128×128 image patches and the corresponding ground truth 4-point homography. $\mathbf{H}_{4Pt}$. We input these data samples in batches of size 50 into a VGG-16 network architecture, as illustrated in Figure 17. The network outputs the predicted 4-point homography $\tilde{\mathbf{H}}_{4Pt}$ as an 8×1 flattened tensor.
The loss function is a simple L2 loss between the predicted and the ground truth 4-point homography . The loss function l is given by:

$$l = \left\| \tilde{\mathbf{H}}_{4Pt} - \mathbf{H}_{4Pt} \right\|_2$$

We used the Adam optimizer with a learning rate of 1×10^{-6}. Each training epoch evaluates and learns from 25,000 training patch pairs and reports the loss on 5,000 validation patch pairs. We trained the model for 20 epochs, which took a total of 3 hours on the Slurm Turing Cluster.

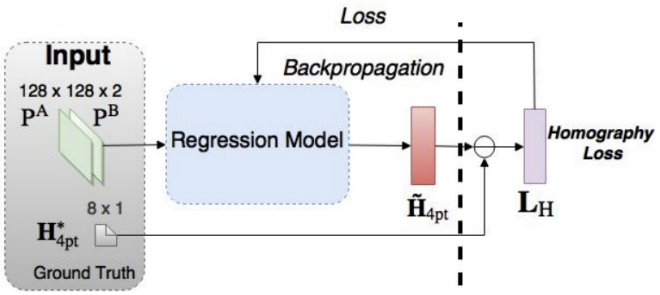


Fig. 2: Supervised Homography Estimation Approach

The loss graphs plotting the EPE losses for the Supervised Network for the training and validation are shown in fig.3 and 4.From the graphs it can be observed that both the loss curves showed gradual decrease and converged to a value only after 20 epochs. In the case of training, the loss converged to 2.04 whereas in the case of validation, it converged to a value of 35.86, clearly showing the case of overfitting to the train dataset. This could be rectified with augmenting the data with even more pathces with different warping. However, despite the difference in loss values, the model could predict homographies of unseen patch pairs very decently as explained further.

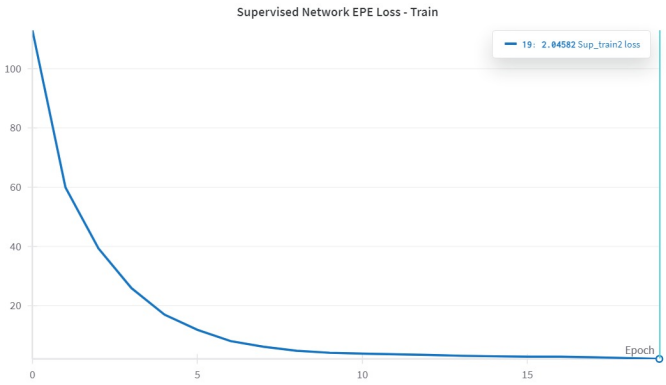


Fig. 3: Training EPE Loss vs Epochs for Supervised Network

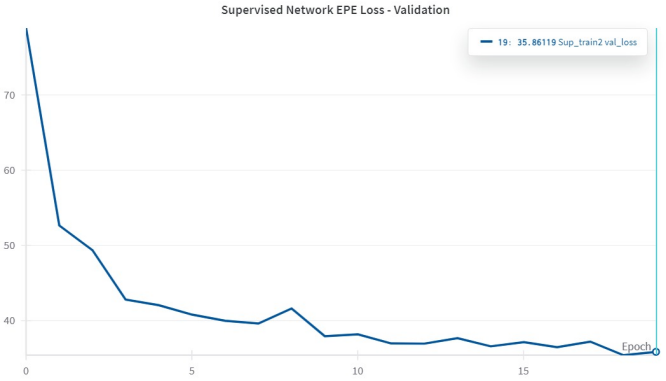


Fig. 4: Validation EPE Loss vs Epochs for Supervised Network

Once the model is trained, we evaluate its performance using previously unseen image pairs. The process begins by feeding two images into the model, which then predicts the 4-point homography between them. To apply this transformation, we first define the original corner points of one image. The model's predicted homography values are then used to slightly adjust these corner points, generating a set of new destination points. Using these original and modified points, we compute the homography matrix, which represents the transformation needed to align the two images. Finally, we apply this transformation to warp and blend the images together, effectively stitching them into a single, aligned output. A sample output can be seen in figure 5 and 6

Fig. 5: Left to Right: Test Image A, Test Image B, Stitched Image

Fig. 6: Left to Right: Test Image A, Test Image B, Stitched Image

C. Unsupervised Approach

The architecture of the unsupervised model remains identical to that of the supervised model. Additionally, all hyperparameters, including the number of epochs (50), batch size (50), and the Adam optimizer with a learning rate of 0.0001, are kept consistent across both models. Since ground truth 4-point homography is not available for direct evaluation, the loss function in this approach differs from the supervised method.

To address this, two additional components are introduced. The TensorDLT module takes as input the predicted 4-point homography $\tilde{\mathbf{H}}_{4Pt}$ along with the corner points of the first image patch to estimate the full 3×3 homography matrix $\tilde{\mathbf{H}}$. This estimated matrix is then utilized to warp the first patch using a differentiable warping layer, specifically the Spatial Transformer Network, which employs bilinear interpolation. Finally, the photometric loss is computed between the warped patch and the second patch, and this loss is backpropagated to update the model parameters. We trained the model for 50 epochs, which took a total of 16 hours on the Slurm Turing Cluster.

$$L_{1\text{loss}} = \|\tilde{P}_A - P_B\|_1$$

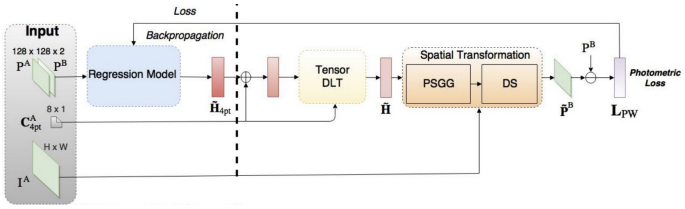


Fig. 7: Unsupervised Homography Estimation Approach

The loss graphs plotting the EPE losses for the Unsupervised Network for the training and validation are shown in fig.8 and 9.Similar to the supervised approach, it can be observed that both the loss curves showed gradual decrease and converged to a value after 50 epochs. In the case of training, the loss converged to 31.77 whereas in the case of validation, it converged to a value of 32.65. The loss curves gradually plateaued around these values and the loss was decreasing slowly with respect to epochs. This can be attributed to the fact that the unsupervised network has many deep layers causing the gradients to vanish or explode, which results in the loss curves to plateau.

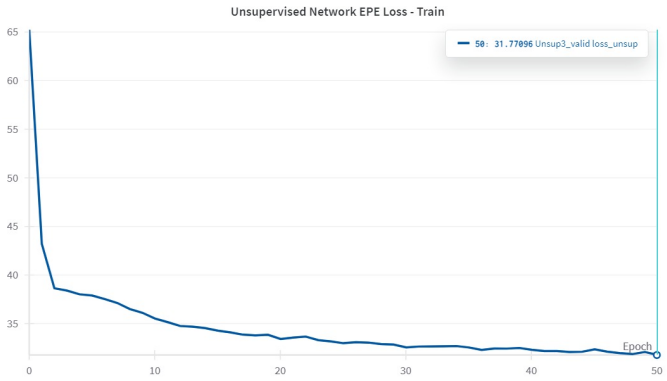


Fig. 8: Training EPE Loss vs Epochs for Unsupervised Network

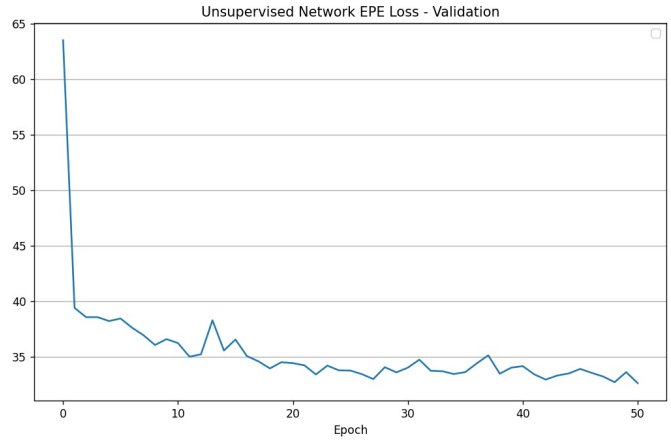


Fig. 9: Validation EPE Loss vs Epochs for Unsupervised Network

D. Discussion & Evaluation

This study presents a thorough performance evaluation of homography estimation techniques. Figure 10 11 12 13 illustrates a comparison between the selected/perturbed patch (marked in green and red, respectively) and the four-corner sets fCB, computed using supervised (yellow), unsupervised (blue), and feature-based traditional (light blue) methods. The supervised approach consistently delivers the most accurate estimates, comparable to those obtained with the feature-based method. In contrast, the unsupervised method consistently yields estimates that are less accurate than those of the supervised approach.

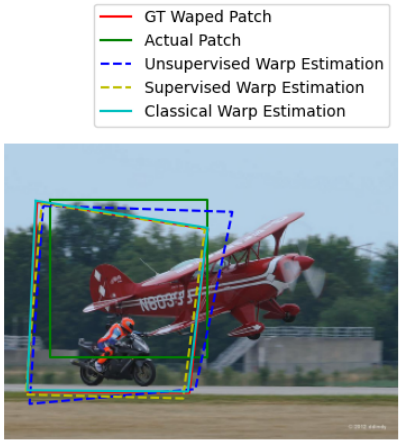


Fig. 10: Train Set Image output of Traditional, Supervised and Unsupervised approaches on Homography estimation comparison

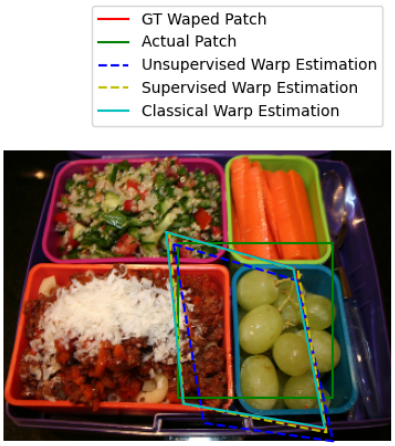


Fig. 11: Train Set Image output of Traditional, Supervised and Unsupervised approaches on Homography estimation comparison

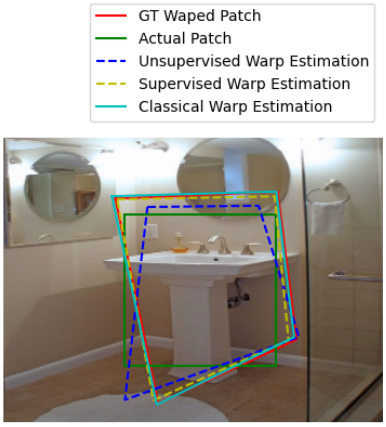


Fig. 12: Train Set Image output of Traditional, Supervised and Unsupervised approaches on Homography estimation comparison

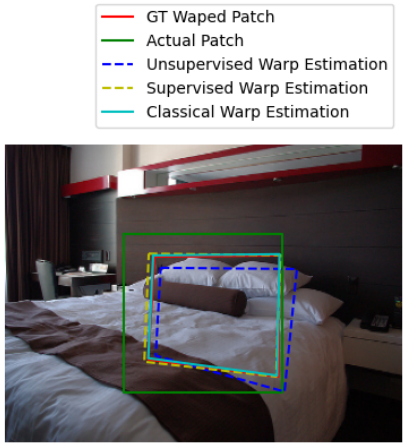


Fig. 13: Train Set Image output of Traditional, Supervised and Unsupervised approaches on Homography estimation comparison

Table I II presents the average Endpoint Error (EPE) and runtime results. While the supervised model showed some degree of overfitting, the validation outcomes were reasonable, as indicated by the convergence observed in the loss curves. Nevertheless, the supervised method outperforms the unsupervised model, whereas traditional methods yield good performance when the features are accurately identified.

	Supervised Model			Unsupervised model		
	Train	Val	Test	Train	Val	Test
EPE	2.04	35.86	25.68	31.77	32.65	40.18
Run time(ms)	25.7	25	23.2	51.8	44.6	42.4

TABLE II: Comparison of model performance metrics

E. Stitching the Images

Similar to the appraoch from Phase I, we stitched the images pairwaise using the Homography matrices obtained from the networks. For this the method we followed was:

- We resized the test image into the dimensions of the patch (128x128) that the networks were trained upon.
- To obtain the Homography matrix from the output of the networks which is the H_{4Pt}, we used the `cv2.warpPerspective()` function.
- Now, with this Homography Matrix, we used the method mentioned in Phase I to overlay both the patches on top of each other, with the Homography matrix coming from the Networks. The full code for this method is mentioned in `Wrapper.py`.

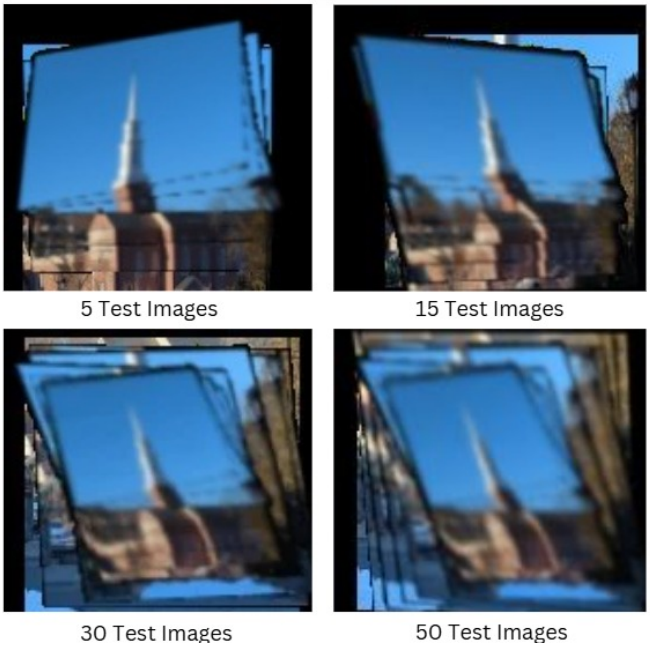


Fig. 14: Results of Stitching images from the Tower Test set using Homography obtained from the Supervised Network for different number of images stitched

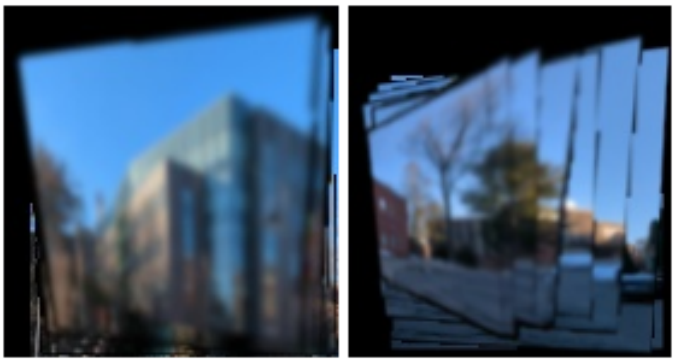


Fig. 15: Results of Stitching images from the Unity Hall Test set (Left) and from the Trees Test set (Right)

F. Evaluation of Test Images

After successfully training and validating the deep learning-based homography estimation models, we conducted testing on

Fig. 16: Results of Stitching & Blending images from the Unity Hall set using the classical approach

images extracted from video sequences. The performance of the deep learning approach was compared against a traditional homography estimation method. Our observations indicate that while the model was able to stitch images with reasonable accuracy, its overall performance was inferior to the conventional approach, likely due to inaccuracies in homography estimation.

The model demonstrated the capability to effectively align and stitch up to five consecutive images with acceptable accuracy. However, as the number of images increased, the quality of the resulting panorama degraded significantly. This decline in performance can be attributed to the accumulation of small errors in homography estimation, which compounded over successive frames, leading to visible misalignments and distortions. A clear example of this limitation is evident in the Tower test dataset, as illustrated in Figure 14, where the panoramic output exhibits noticeable artifacts. Figure 15 depicts the outcome of the Trees and Unity Test dataset.

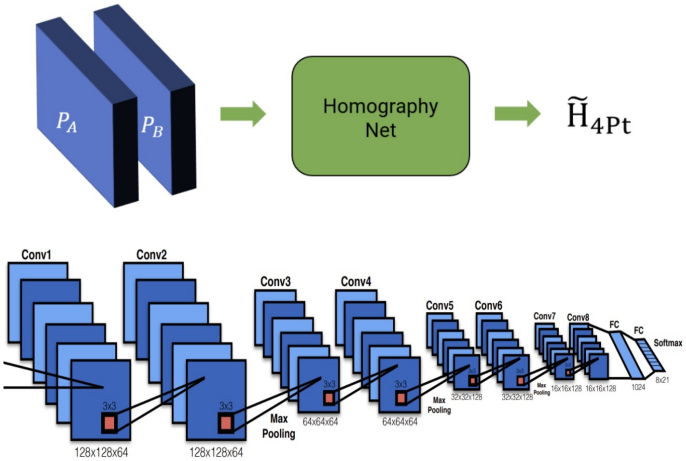


Fig. 17: Supervised Model Architecture

G. Conclusion and Future Improvements

The classical approach using feature-based panorama stitching, as implemented during Phase I of the project, provided significantly more accurate and visually consistent results compared to the deep learning-based homography estimation. This is evident in Figure 16, where the Unity Hall test dataset demonstrates superior stitching quality using the Phase I approach.

To enhance the performance of the deep learning model, improvements such as a more diverse training dataset, refined network architectures, and optimized loss functions are necessary to minimize cumulative errors. Incorporating transformer-based models, robust homography constraints, and hybrid methods that integrate deep learning with traditional techniques could further improve accuracy. Additionally, multi-stage training, fine-tuning, and data augmentation could enhance generalization and robustness, particularly for stitching larger image sets.

REFERENCES

[1] K. Mikolajczyk and C. Schmid, "A performance evaluation of local descriptors," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 10, pp. 1615–1630, 2005. doi: 10.1109/TPAMI.2005.188.

[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," *arXiv preprint arXiv:1709.03966*, vol. 3, Feb. 2018. doi: 10.48550/arXiv.1709.03966.

[3] D. DeTone, T. Malisiewicz, and A. Rabinovich, "Deep Image Homography Estimation," *arXiv preprint arXiv:1606.03798*, Jun. 2016. doi: 10.48550/arXiv.1606.03798.

[4] E. Brachmann, A. Krull, S. Nowozin, J. Shotton, F. Michel, S. Gumhold, and C. Rother, "DSAC - Differentiable RANSAC for Camera Localization," *arXiv preprint arXiv:1611.05705*, Mar. 2018. doi: 10.48550/arXiv.1611.05705.