Project 1 – MyAutoPano

Dhruv Agrawal

Robotics Engineering Department
Worcester Polytechnic Institute
Worcester, Massachusetts 01609
Email: dagrawal@wpi.edu

Luis F. Recalde

Robotics Engineering Department
Worcester Polytechnic Institute
Worcester, Massachusetts 01609
Email: lfrecalde@wpi.edu

***Abstract*—Image stitching is a crucial technique in computer vision that enables the creation of wide field-of-view (FOV) images, such as panoramas. The traditional approach involves the following steps: detecting key-points in images, extracting feature descriptors, and using Random Sample Concensus (RANSAC) to robustly estimate the homography matrix between images. Today, deep learning approaches have improved image stitching by employing neural networks for feature detection and matching, often exceeding traditional methods in handling complex transformations. This work implements both traditional and deep learning-based formulations in order to increase our understanding of these techniques.**

I. Phase I: Traditional Approach

This section presents the implementation of panorama stitching considering the traditional formulation. The traditional approach consists of different steps, an overview is presented in Fig. 1.

A. Corner Detection

A corner in an image is a region where the gradient shows significant changes in multiple directions. The Harris Detector [1], a popular algorithm for corner detection, analyzes intensity variations in a pixel's local neighborhood to identify such points. Given a gray scale image denoted as $\mathbf{I} \in \mathbb{R}^{n \times m}$ and an image patch as $(u, v) \in \mathbf{W}$ considering variations in the pixels as $(\Delta u, \Delta v)$. n and m represent the dimensions of the image and $\mathbf{W} \in \mathbf{I}$. The sum of squared differences can be formulated as follows:

$$E(\Delta u, \Delta v) = \sum_{u_k, v_k \in \mathbf{W}} (\mathbf{I}(u_k, v_k) - \mathbf{I}(u_k + \Delta u, v_k + \Delta v))^2 \tag{1}$$

We aim to maximize the function $E(\Delta u, \Delta v)$ to detect the corners in the image. By employing a Taylor series expansion, we approximate (1), resulting in the following formulation:

$$E(\Delta u, \Delta v) = \begin{bmatrix} \Delta u & \Delta v \end{bmatrix} \mathbf{M} \begin{bmatrix} \Delta u \\ \Delta v \end{bmatrix} \tag{2}$$

where the matrix $\mathbf{M}$ is defined as follows:

$$\mathbf{M} = \sum_{u_k, v_k \in \mathbf{W}} \begin{bmatrix} \mathbf{I}_u^2 & \mathbf{I}_u \mathbf{I}_v \\ \mathbf{I}_v \mathbf{I}_u & \mathbf{I}_v^2 \end{bmatrix}$$

The matrices $\mathbf{I}_u$ and $\mathbf{I}_v$ represent partial derivatives, which can be approximated using Sobel operators. We can determine if

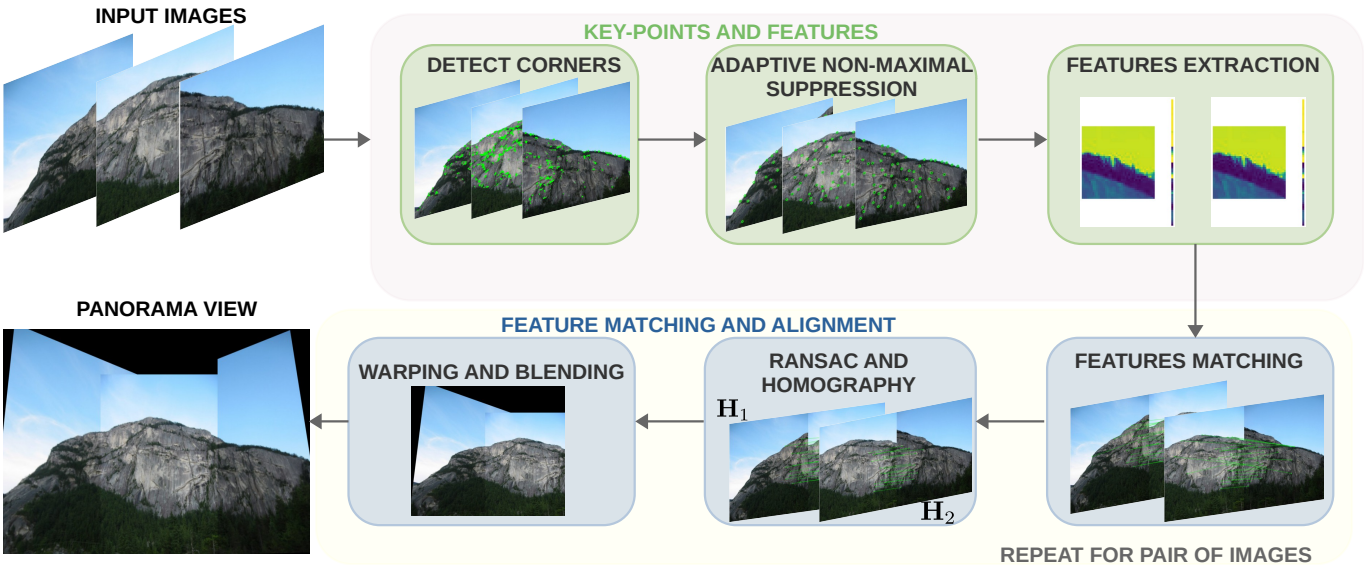

Fig. 1. Overview of panorama stitching using traditional method.

the patch contains a corner by the following

$$r = \det(\mathbf{M}) - k(\text{trace}(\mathbf{M}))^2 \tag{3}$$

when the value of r is significantly large, it indicates the presence of a corner within the region. We implemented the Harris corner detection algorithm, and the results are presented in Fig. 2.

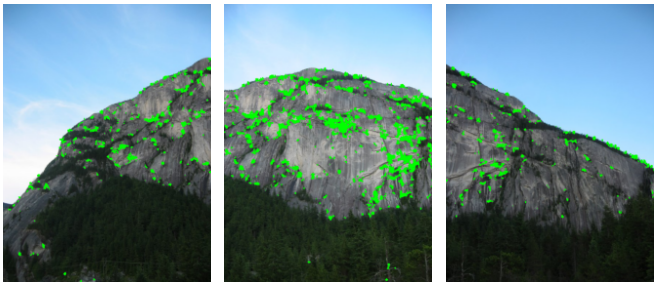

Fig. 2. Harris corner detection

B. Adaptive Non-Maximal Suppression

While the Harris detector successfully identifies corners in the image as illustrated in Fig. 2, these features are not uniformly distributed. The purpose of Adaptive Non-Maximal Suppression (ANMS) [2] is to enhance the spatial distribution of key points. Specifically, it aims to select the N_{best} most optimal corners by identifying those that are true local maxima. This process is detailed in Algorithm 1

Algorithm 1: Adaptive Non-Maximal Suppression

Data: Corner score Image $\mathbf{C}$ and the maximum of best corners N_{best}
Result: (x_i, y_i) for $i = 1 : N_{best}$
Initialized $r_i = \infty$ for $i = 1 : N_{strong}$;
for $i = [1 : N_{strong}]$ **do**
  for $j = [1 : N_{strong}]$ **do**
  end
  if $\mathbf{C}_{[y_j, x_j]} > \mathbf{C}_{[y_i, x_i]}$ **then**
    $ED = (x_j - x_i)^2 + (y_j - y_i)^2$;
  end
  if $ED < r_i$ **then**
    $r_i = ED$;
  end
end
Sort r_i in descending order and pick top N_{best} points

The results illustrate corner points with an enhanced spatial distribution across the image, as shown in Fig. 3, with $N_{best} = 200$.

C. Features Descriptor

We successfully identified the corners or key points in the image with an enhanced spatial distribution. The next step is to generate descriptors that facilitate comparisons between these key points. Each key point is represented by a feature vector or descriptor, derived from a patch centered on the key point that captures its distinguishing features. Various methods can be used to generate feature descriptors, including

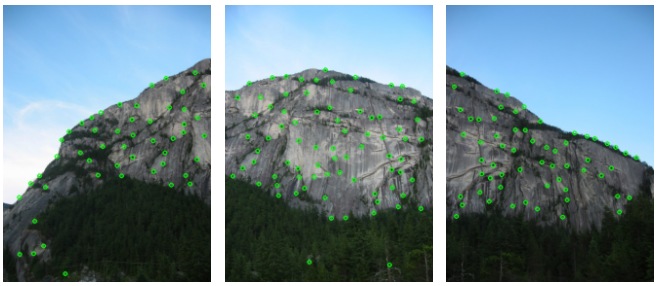

Fig. 3. ANMS considering $N_{best} = 200$

Gaussian functions, gradient orientations, or texture-based approaches. In this work, we adopted a straightforward method by applying a Gaussian function to the patch and subsequently sub-sampling the resulting blurred output.

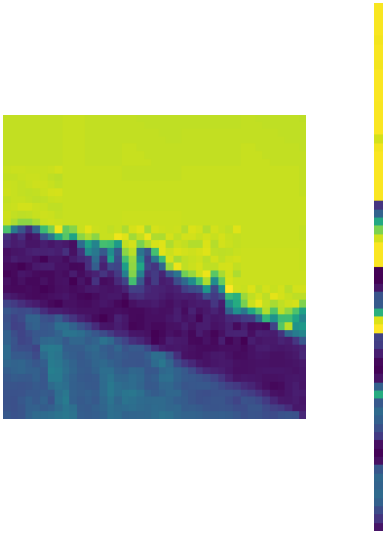

Fig. 4. The feature vector of a selected key point

A 41×41 patch along with its corresponding feature vector is shown in Fig. 4.

D. Features Matching

Previously, we generated a feature vector for each key point. Now, the task is to identify the feature vector correspondences between the two images. To achieve this, for a given point in Image 1, we calculate the differences with all points in Image 2 and compute the squared Euclidean norm for each. Using this metric, we determine the ratio between the best match (lowest distance) and the second-best match (second lowest distance). If this ratio is below a predefined threshold, the match is retained; otherwise, it is discarded.

We repeat this process for every point in Image 1. The results of the feature vector matching are depicted in Fig. 5, illustrating the algorithm's capability in identifying correspondences between feature vectors.

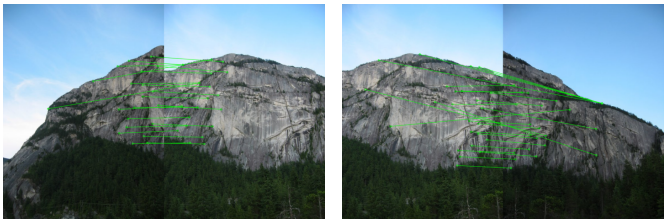

Fig. 5. Features matching

E. Random Sample Concensus (RANSAC) for outlier rejection and estimation of Robust Homography

In the previous section, we demonstrated the ability to match feature vectors between images. However, as illustrated in Fig. 5, the results reveal that we are identifying not only correct correspondences but also incorrect ones. To mitigate these matches, this work employs RANSAC [3] method to compute the homography using only the accurate matches between images.

Algorithm 2: RANSAC

Data: Four feature matches from images $\mathbf{P}$ and $\mathbf{P}^{'}$
Result: Homography matrix $\mathbf{H}$ between pair of images
for $i = [1 : N_{max}]$ **do**
  $\mathbf{P} \leftarrow$ *Four random features from image 1*
  $\mathbf{P}^{'} \leftarrow$ *Four random features from image 2*
  $\mathbf{H} \leftarrow (\mathbf{P}_1, \mathbf{P}_1^{'})$ *Compute homography*
  $\left\|\mathbf{p} - \mathbf{H}\mathbf{p}^{'}\right\| < \tau$ *Compute inliers*
  if *inlier* $> 90\%$ **then**
    break;
  end
end
Keep the largest set of inliers
Re-compute least-squares $\hat{\mathbf{H}}$

The results of the RANSAC algorithm are illustrated in Fig. 6, which demonstrates a substantial improvement compared to the results depicted in Fig. 5. It should be noted that RANSAC is not the only method available to address this problem, as one can also employ certain modifications to the L1 and L2 distances.

Fig. 6. Feature matches after outliers have been removed using RANSAC

F. Image Stitching

Once the homography matrix between the pair of images is obtained, it becomes possible to combine multiple views into a single image, resulting in a seamless panoramic view. However, in this technique, it is quite important to blend the common region between images without affecting the regions that are not common.

1) Poisson Blending: This technique is used to seamlessly blend one image into another when stitching multiple images. Poisson blending [4] achieves this by transferring gradient information from the source image to the destination image while maintaining the overall smoothness of the stitched result.

This approach is based on solving a Poisson equation for the image to ensure that the gradients from the second image are maintained while minimizing discontinuities at the boundary of the stitch. The Poisson equation is represented as:

$$\nabla^2 \Phi = div(F)$$

where Φ is the resulting image function, F is the vector field that represents the differences in pixel intensities and div(F) is solved to generate a smooth transition.

2) Alpha Blending: This technique combines multiple images using the value *alpha*, which represents the transparency of each image. By enabling smooth transitions between the foreground and background, it is particularly well-suited for tasks such as image compositing.

The formula for *alpha* blending combines the pixel colors of two images according to the transparency of the foreground image:

$$C_{out} = \alpha C_{foreground} + (1 - \alpha) C_{background}$$

The results associated with the Poisson blending technique are illustrated in Fig. 7, where the results of a pair of images and the final panoramic output are presented.

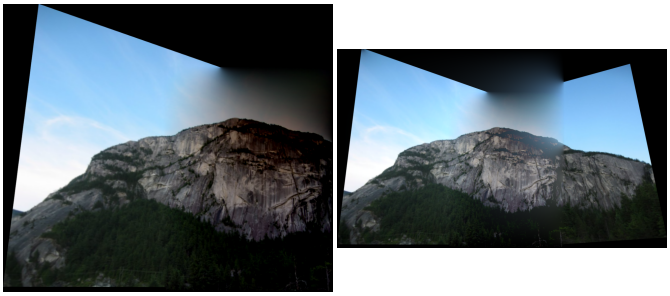

Fig. 7. Stitching and blending of train images Set 2

The results of the Alpha blending technique are inferior to those of Poisson blending, primarily because Alpha blending is a simpler method that combines two images without accounting for brightness gradients. Additionally, the value of α must be carefully chosen for each image set to ensure satisfactory results with Alpha blending.

G. Panorama Views from Train Images

1) Results from Train Set 1 and 2: This section presents the outcomes of employing the traditional techniques for image stitching, with the provided images in the dataset being considered for analysis. In Fig 8 a set of three images are stitched together where the left part shows the stiching of two

images and the right one shows the stiching of the third image to the result of the first stitch. Similarly, any number of images can be stitched together to create a panorama view using our algorithm as seen in Fig. 7. Figures 7 and 8 show the results of the stitching process applied to the training image sets, Set 1 and Set 2, respectively.

2) Results from Train Set 3: The stitching process for the training images in Set 3 was particularly challenging because these images were captured close to objects in the scene, rendering the imaginary plane principle inapplicable. In contrast, for Sets 1 and 2, the views could be approximated to a plane since the images were taken from a greater distance. This difference explains why the stitching and blending in Set 3 are less effective compared to the results presented earlier.

Fig. 8. Stitching and blending of Train Images Set 1

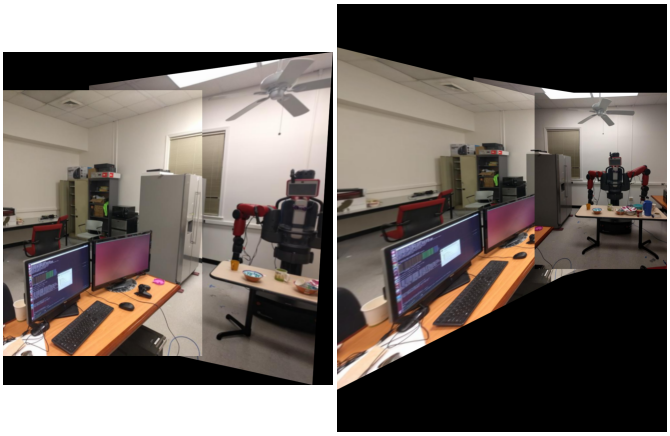

Fig. 9. Stitching and blending of Train Images Set 3 left side

Using an image from Set 3 as the central frame, the stitching process can be executed in relation to this reference frame. The resulting stitched images for the right and left sides are presented in Fig. 10 and Fig. 9, respectively. Finally, the panoramic view of this data set is presented in Fig. 11

H. Panorama Views from Test Images

1) Results from Test Images Set 1: Test Image Set 1 features a chessboard pattern, which is considered repetitive and can lead to ambiguous matches during feature detection. This

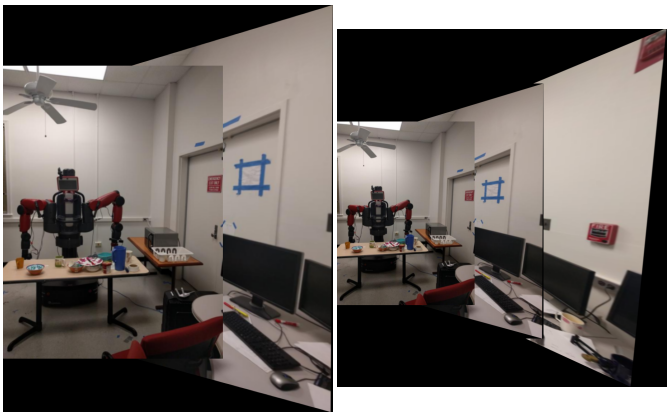

Fig. 10. Stitching and blending of Train Images Set 3 right side

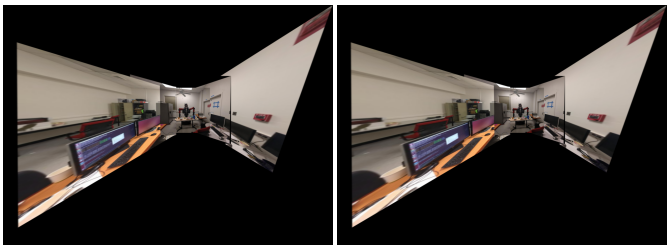

Fig. 11. Stitching and blending of Train Images Set 3

challenge can be addressed using the known dimensions of the chessboard. However, the specifications of the chessboard were not provided for this set, resulting in a sub-optimal performance of our formulation. The outcomes produced by the RANSAC algorithm when applied to Test Image Set 1 are shown in Fig. 12. The results of the stitching and blending process are shown in Fig. 13.

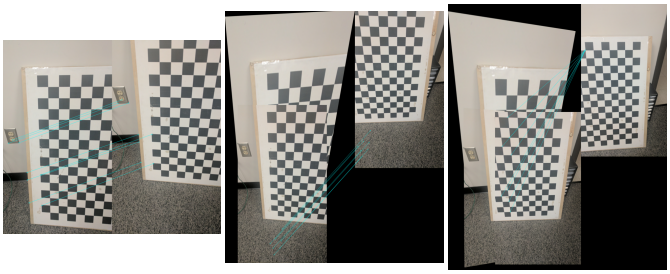

Fig. 12. Feature matches after outliers have been removed using RANSAC form Test Images Set 1

2) Results from Test Images Set 2: Not having the images in the right order makes the stitching process much harder. Our formulation relies on the images being sequential, so it cannot overlap and align them properly. With set 2, it has been tough to find consistent matches between the images, so we have not been able to create a proper panoramic view. The results obtained in this set are illustrated in Fig. 14.

3) Results from Test Images Set 3: This section presents the results for Test Image Set 3, where a panoramic image was

Fig. 13. Stitching and blending of Test Images Set 1

Fig. 14. Stitching and blending of Test Images Set 2

successfully generated, as shown in Fig. 14..

The panorama view generated from Test Image Set 4 is presented in Fig. 14.. This set includes images that lack common features with others in the set, posing a challenge for the stitching process. To enhance our algorithm's robustness, we implemented a stepwise evaluation of feature correspondences by identifying the maximum number of inliers between each pair of images. If no inliers are detected, the algorithm proceeds to the next image in the sequence until the entire set is evaluated. Despite its simplicity, this approach effectively excludes unrelated images from the set and ensures successful image stitching and blending. The resultant panorama is seamless and accurately aligned, demonstrating the efficacy of the proposed method.

I. Panorama Views from Train Images Custom Set

This section presents the results of the stitching algorithm using the traditional approach. For this evaluation, we used images of the Worcester Polytechnic Institute campus. The resulting panorama is shown in Fig. 14..

II. Phase II: Deep Learning Approach

This section presents the implementation of panorama stitching considering the deep learning network approaches. The deep learning approach can be performed in two different ways:

1) Supervised Learning
2) Unsupervised Learning

The following subsections explain both methods in detail.

A. Dataset Generation

To train both supervised and unsupervised approaches for estimating homography between pairs of images, we need datasets comprising pairs with known homographies. The procedure of obtaining these datasets is generally challenging, as determining the homography requires knowledge of the transformations between image pairs. Consequently, synthetic pairs of images are generated to train the networks, using a small subset of images from the **MSCOCO** [5] dataset, which encompasses a diverse array of objects in natural scenes.

The steps to generate the images are as follows:

- Before generating image pairs, all the pairs should be of the same size and the first step is to resize the images to a standard dimension. In our implementation, we **resize** the images to **320x320**.
- We then apply a random crop on the images in feature-rich regions using corner detection method. This is our original patch $\mathbf{P}_A$ having corners $\mathbf{C}_A$.
- On this patch $\mathbf{P}_A$ we apply **random transformation** $\mathbf{H}_B^A$ to get a perturbed patch with corners $\mathbf{C}_B$.
- We calculate the inverse homography $\mathbf{H}_A^B$ that maps from the perturbed patch to $\mathbf{P}_A$ and call this the patch $\mathbf{P}_B$ that has the same coordinates in the transformed image as that of $\mathbf{P}_A$ in the original image.

The dataset comprises 5000 images, from which 30 patches are generated for each image using the aforementioned procedures, resulting in a total of **150,000 images**; a subset of these images is illustrated in Figure 15.

B. Supervised Approach

A supervised learning approach is a type of machine learning in which a model is trained on labeled data. This means that the algorithm learns from a dataset that already contains input-output pairs, where each input is associated with a correct output. In our implementation, the input of the model is the pair a pair of gray-scale image patches, and the ground truth labels are the homography between them.

The objective of this optimization problem is to minimize the error between the estimated output of the neural network and the ground-truth homography. In this work we use the structure presented in **HomographyNet** [6].

Given the complexity of using homography labels as the target output for neural networks—primarily due to challenges posed by rotational and translational dimensions; this work

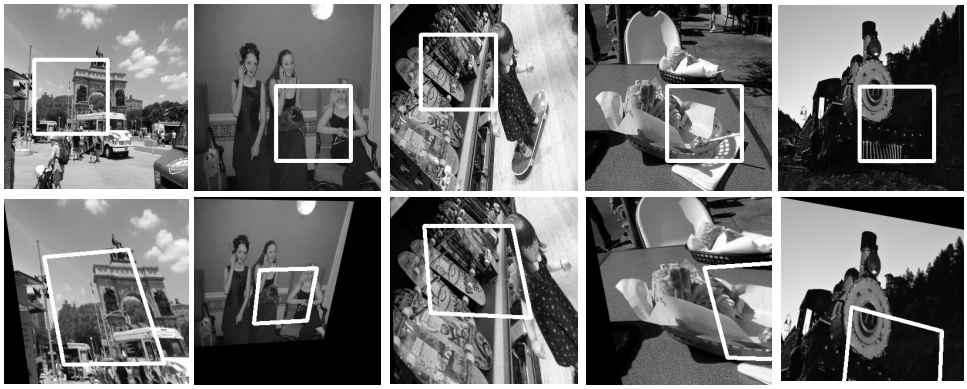

Fig. 15. Images from the dataset showing the original image with the patch P_A and the image after the transformation with the patch C_A.

uses the element H_{4pt}, which involves calculating the difference of the corner coordinates, as outlined below:

$$\mathbf{H}_{4pt} = \mathbf{C}_A - \mathbf{C}_B$$

with a one to one mapping from $\mathbf{H}_{4pt}$ to $\mathbf{H}_A^B$. The $\mathbf{H}_{4pt}$ is **normalized** and taken as our ground truth.

The patch size for each image is **128x128**, and the patches $\mathbf{P}_A$ and $\mathbf{P}_B$ are **normalized** and stacked on top of each other to obtain a single input. Fig. 17 shows the representation of the supervised formulation.

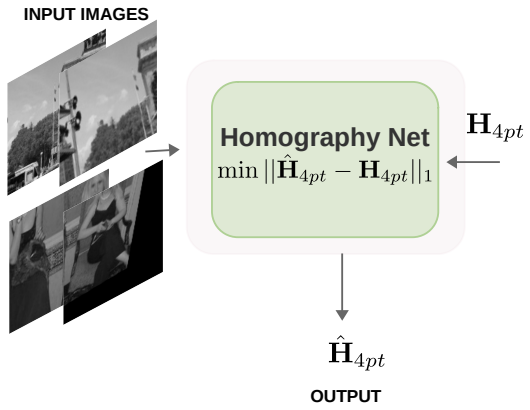

Fig. 17. Structure of the Supervised approach

1) HomographyNet: With the input and output of the supervised architecture delineated, the subsequent step involves a formal definition of the model as follows:

We use the HomographyNet, a deep CNN having **8 convolutional** layers and **2 fully connected** layers. Each layer has **3x3 convolutional** blocks with **BatchNorm** and **ReLUs**. The network takes as input a two-channel gray-scale image i.e the image stack P_A and P_B sized **128x128x2**. We use 8 convolutional layers with a **max pooling** layer (2x2, stride 2) after every two convolutions. The 8 convolutional layers have the following number of filters per layer: 64, 64, 64, 64, 128, 128, 128, 128.

The convolutional layers are followed by two fully connected layers. The first **fully connected** layer has **1024** units. **Dropout** with a probability of **0.5** is applied after the final convolutional layer and the first fully-connected layer. The output of the network directly produces **8 real-valued numbers** which are the predictions of the difference of perturbation from $\mathbf{C}_A$ to $\mathbf{C}_B$. More details are presented in Fig. 16.

2) Hyper parameters: The hyperparameters used for the model are as follows:

- An SGD optimizer is used with a learning rate of 0.006 and decreased by a factor of 5/6 after every 30 epochs.
- The loss metric used is L_1 comparing the outputs and labels.
- The model is trained for 100 epochs with a batch size of 64.

3) Experiments and Results: The model is trained on the dataset of size 150,000 image pairs for 100 epochs for about **8.0** hours on a single NVIDIA RTX GPU. Each image is resized to **320x320** dimensions and subsequently converted to

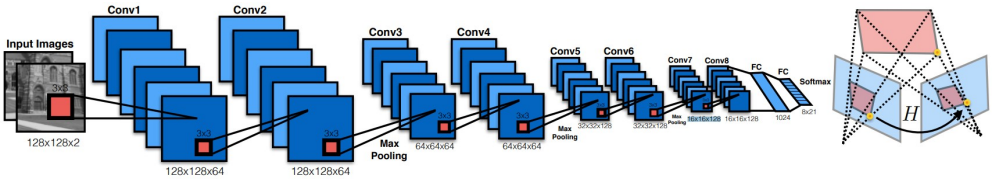

Fig. 16. Internal structure of the Supervised neural network

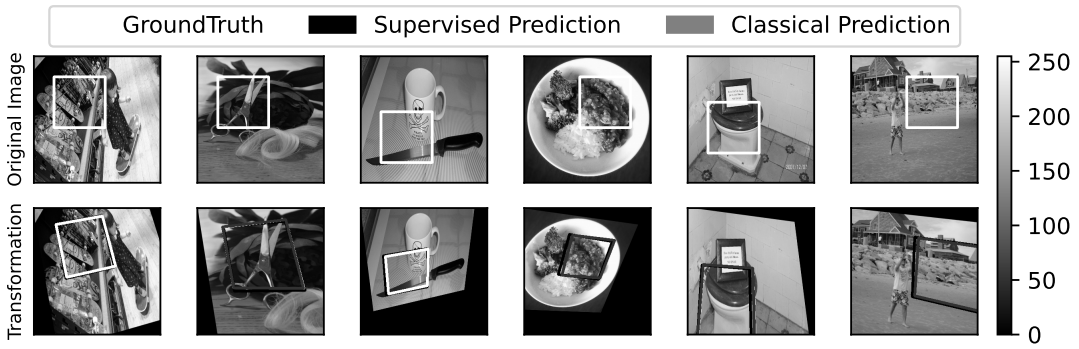

Fig. 18. Comparison results of the supervised and traditional approache considering validation images

gray-scale. Patch pairs are produced utilizing the methodology detailed in II-A, and are fed into the neural network with a batch size set at **64**. The outcomes of the training and validation processes are elaborated in Fig. 20.

The obtained supervised structure appears to generalize results effectively, which may be attributed to the utilization of input-output normalization and the application of the L1 norm distance metric during the learning process.

Figure 18 presents a comparison between the outcomes of the supervised structure and the traditional approach as proposed in I. The analysis reveals that the supervised method demonstrates an ability to estimate $\mathbf{H}_{4pts}$ and subsequently compute homography $\mathbf{H}_B^A$ with precision. This is evidenced by the precision of this formulation in mapping the corner $\mathbf{C}_A$ to $\mathbf{C}_B$. The traditional approach fails to accurately map the corner in two out of the five images, thus underscoring the advantages of the supervised method over the traditional technique.

4) Image Stitching Supervised: This section presents the results of image stitching using the supervised approach. To establish a ground truth, the SIFT algorithm is applied to compute features among the various images, thereby facilitating the computation of the ground truth homography. Fig. 19 presents the results of both the SIFT algorithm and the supervised approach. In this case, the supervised method proves to be insufficient in accurately estimating the homography matrices. We attribute these limitations to the location and dimensions of the patches, given that the supervised method relies only on a small patch for homography estimation, whereas the SIFT algorithm processes the entire image. Although our proposed method lacks sufficient accuracy, the computational time required to process each image is significantly less than that required by SIFT, more details in Table I

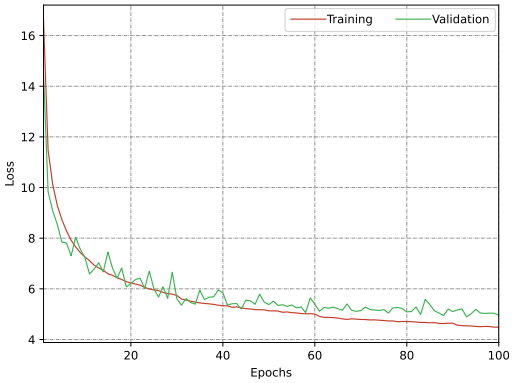

Fig. 20. Cost values during training and validation supervised structure

The results associated with the stitching of different images are presented in Fig. 21, we can see how the supervised approach is not able to accurately stitch pairs of images.

5) Supervised approach Test Images: Ultimately, we present the performance evaluation of the supervised method

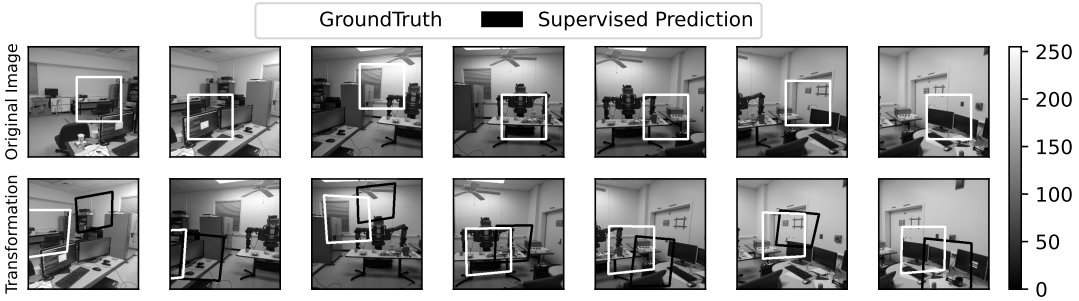

Fig. 19. Comparison results of the supervised approach and Sift Algorithm

TABLE I
COMPUTATIONAL TIME AND ACCURACY

Supervised Approach	Inference Time
	0.6 ms
SIFT	**Inference Time**
	100 ms

TABLE II
COMPUTATIONAL TIME AND ACCURACY SUPERVISED

Supervised	Training	Validation	Test
Accuracy	3.0	4.8	4.9
Inference Time	10 ms	0.6 ms	0.6 ms

Fig. 21. Stitching process using supervised approach

applied to training, validation, and test datasets. Table II details the results for each specific metric.

Fig. 22 presents the outcomes of the supervised approach in comparison to the traditional formulation. The traditional formulation failed in two scenarios, whereas the supervised approach successfully estimated the location of the patch in both cases. The stitching process considering images of the test dataset is presented in Fig. 25

C. Unsupervised Approach

The unsupervised learning approach is a type of machine learning in which the model is trained using data that are not labeled, meaning that the input data do not have ground truth for comparison. In unsupervised learning, the goal is for the model to find patterns, relationships, or structures in the data without the need for explicit supervision.

1) Dataset Generation: The data generation steps are exactly the same for the unsupervised network compared to the supervised network. In fact, the same data set generated for the supervised method is used for the unsupervised method, which consists of **150,000** image pairs of size **128x128** each. The only difference is that we do not use the ground truth homography for the error computation and learning process.

2) HomographyNet: The model pipeline is an extension of the previous model, where the same HomographyNet model is used and two more blocks are added, namely the **TensorDLT** and **Spatial transformer** blocks [7] as shown in Fig. 23.

The HomographyNet model works the same way in the forward iteration of the model giving the same H_{4pt} output. The next processes are as follows:

- The TensorDLT layer takes this output and the corners C_A and calculates the estimated homography $\tilde{\mathbf{H}}_{\mathbf{A}}^{\mathbf{B}}$.
- The spatial transformer takes the original image I_A and warps it using estimated homography to give an estimated image $\tilde{I}_B$.
- Using the coordinates C_A we extract the estimated patch $\tilde{\mathbf{P}}_{\mathbf{B}}$ from the estimated image $\tilde{I}_B$.
- The patches P_B and $\tilde{P}_B$ are compared using L_1 loss function and the weights of the model are updated to reduce this loss.

More details of the unsupervised approach are presented in Fig. 23

3) Hyperparameters: The hyperparameters used for the model are as follows:

- An SGD optimizer is used with a learning rate of **0.0001** and is decreased by a factor of **10** after every **10** epoch.

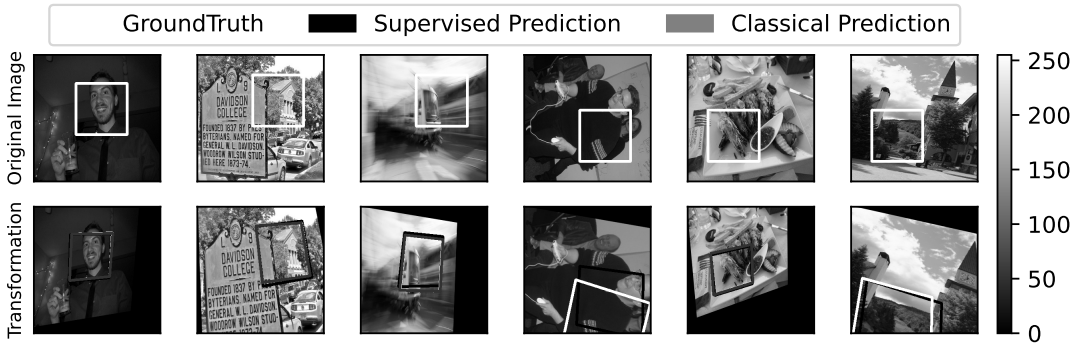

Fig. 22. Comparison results of the supervised and traditional approach considering test images

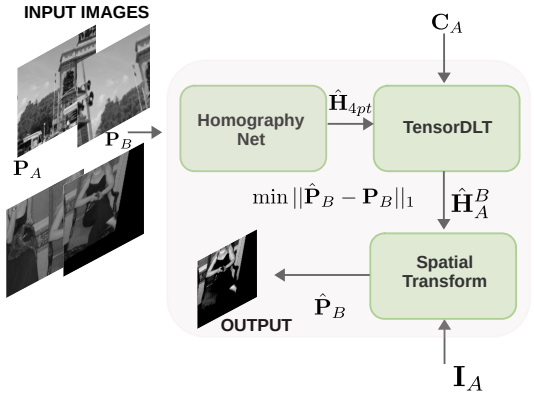

Fig. 23. Structure of the Unsupervised approach

- The loss metric used is L_1 comparing the patch P_B to the estimated patch $\tilde{P}_B$.
- The model is trained for **50** epochs with a batch size of **64**.

4) TensorDLT: This paragraph explains the working of TensorDLT layer. The inputs to this layer are $\tilde{H}_{4pt}$ and C_A. Let the coordinates of each point in C_A be (u_i, v_i). We obtain the estimate coordinates of $\tilde{P}_B$ as

$$\tilde{C_B} = H_{4pt} + C_A$$

Let the coordinate of each point in $_B$ be (u_i', v_i') The homogeneous tranfomation between the points is written as:

$$x' = Hx$$

$$\begin{bmatrix} u' \\ v' \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$$

By expanding the above equation we obtain:

$$u_i' = \frac{h_{11}u_i + h_{12}v_i + h_{13}}{h_{31}u_i + h_{32}v_i + 1}$$

$$v_i' = \frac{h_{21}u_i + h_{22}v_i + h_{23}}{h_{31}u_i + h_{32}v_i + 1}$$

We can rearrange the terms in the form **Ah=0**, where

$$h = [h_{11}\ h_{12}\ h_{13}\ h_{21}\ h_{22}\ h_{23}\ h_{31}\ h_{32}\ 1]^T$$

$$A_i = \begin{bmatrix} u_i & v_i & 1 & 0 & 0 & 0 & -u_i'u_i & -u_i'v_i & -u_i' \\ 0 & 0 & 0 & u_i & v_i & 1 & -v_i'u_i & -v_i'v_i & -v_i' \end{bmatrix}$$

$$A = [A_1 A_2 A_3 A_4]^T$$

for each corner point coordinate in the patch.
The solution to this is found by calculating the eigenvalues of matrix A and taking the eigenvector of the **minimum eigenvalue** as the h vector. The H matrix is then calculated by reshaping the h vector into a 3x3 matrix.

5) Experiments and Results: The model is trained on the dataset of size **150,000** image pairs for **50** epochs for about **12** hours on an NVIDIA **A30** GPU. All images are resized to **320x320** and converted to grayscale as done for the supervised training. The inputs are provided to the model and the H_{4pt} values are obtained which are given to the tesnorDLT layer along with C_A to obtain the estimated homography matrix. We use **kornia transform** to warp the image I_A to obtain the etimated image $\tilde{I}_B$. Applying the coordinates C_A to $\tilde{I}_B$ gives us the estimate patch $\tilde{P}_B$. The loss L_1 is calculated for P_B and $\tilde{P}_B$ and the model weights are updated based on the loss value.
The results of the training and validation processes are elaborated in Fig. 24

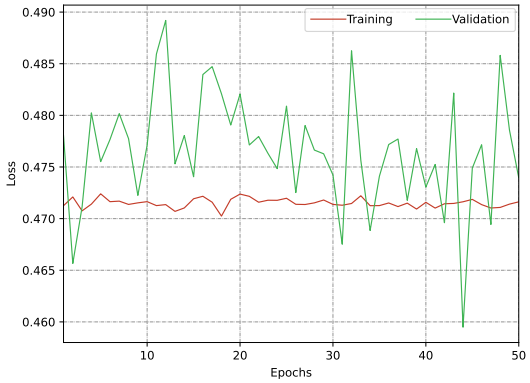

Fig. 24. Loss values during training and validation of the unsupervised structure

III. Checkpoints

Since the checkpoints of our model are >20MB, we have uploaded them on OneDrive.
Link for Supervised Network weight checkpoints
Link for Unsupervised Network weight checkpoints

References

[1] C. Harris, M. Stephens *et al.*, "A combined corner and edge detector," in *Alvey vision conference*, vol. 15, no. 50. Citeseer, 1988, pp. 10–5244.
[2] M. Brown, R. Szeliski, and S. Winder, "Multi-image matching using multi-scale oriented patches," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1. IEEE, 2005, pp. 510–517.
[3] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.
[4] P. Pérez, M. Gangnet, and A. Blake, "Poisson image editing," in *Seminal Graphics Papers: Pushing the Boundaries, Volume 2*, 2023, pp. 577–582.
[5] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*. Springer, 2014, pp. 740–755.
[6] D. DeTone, T. Malisiewicz, and A. Rabinovich, "Deep image homography estimation," *arXiv preprint arXiv:1606.03798*, 2016.
[7] T. Nguyen, S. W. Chen, S. S. Shivakumar, C. J. Taylor, and V. Kumar, "Unsupervised deep homography: A fast and robust homography estimation model," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 2346–2353, 2018.

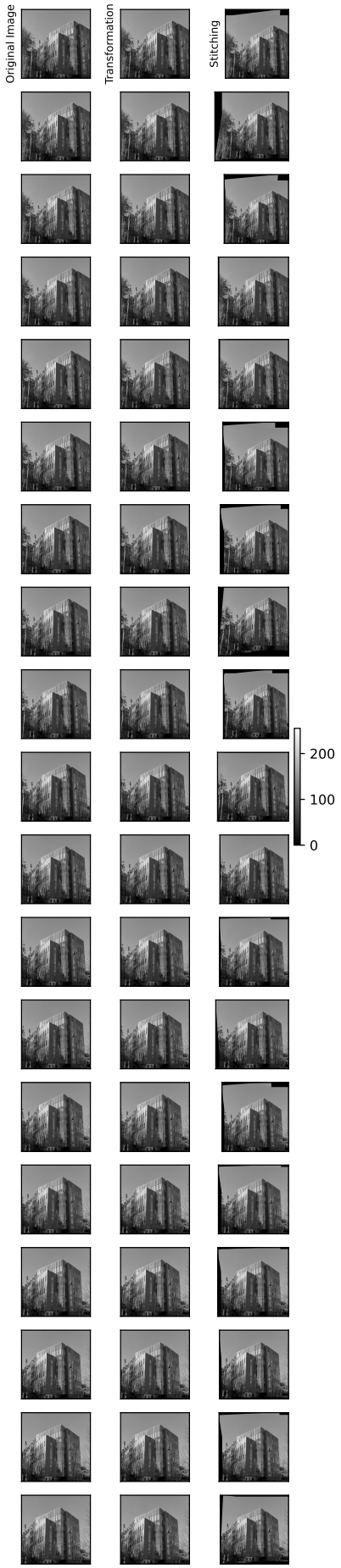

Fig. 25. Stitching pair of images of the test dataset