RBE549 Project 1 - My AutoPano

Yaşar İdikut
yidikut@wpi.edu

Harshal Bhat
hbhat@wpi.edu

I. Phase 1: Traditional Approach

In this project, we perform panorama stitching using the traditional method. This method involves 6 major steps as follows:

- Corner Detection
- Adaptive Non-maximal Suppression
- Feature Descriptor Generation
- Feature Matching
- Outlier Rejection using Radom Sampling Consensus(RANSAC) and Robust Homography estimation
- Image Warping, stitching, and blending

A. Corner Detection

This section focuses on the Harris and shi-Tomashi corner detection methods using open-cv functions. Firstly we converted the images to grayscale we applied the Harris algorithm using *cv2.cornerHarris* to determine the number of corners meeting a certain threshold and visually represented them overlaid on the original image.

Fig. 1 shows the result of Harris Corner detection on a set of images.

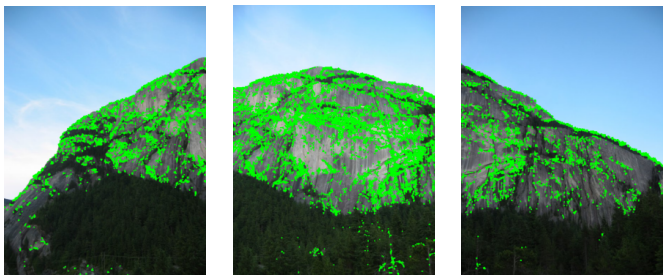

Fig. 1. Harris Corner Detection

B. Adaptive Non-Maximal Suppression (ANMS)

Followed by the corner detection we implemented ANMS for refining the obtained corners so that they are evenly spread out through the image. Employing peak local maxima detection, the algorithm identifies strong corner candidates, and their coordinates are then iteratively compared to determine their suppression based on Euclidean distance. The process results in a set of refined corner points, highlighted in the image using red circles. Figure 3 showcases the effectiveness of ANMS in selecting the 500 most salient corners while discarding redundant ones.

Algorithm 1 Adaptive Non-Maximal Suppression

Require: Corner score Image (C_{img} obtained using corner metric), N_{best} (Number of best corners needed)
Ensure: (x_i, y_i) for $i = 1 : N_{best}$

```
Find all local maxima using C_img
Find (x, y) coordinates of all local maxima
Initialize r_i = ∞ for i = 1 : N_strong
for i = 1 : N_strong do
    for j = 1 : N_strong do
        if C_img[y_j, x_j] > C_img[y_i, x_i] then
            ED = (x_j − x_i)^2 + (y_j − y_i)^2
        end if
        if ED < r_i then
            r_i = ED
        end if
    end for
end for
Sort r_i in descending order and pick top N_best points. =0
```

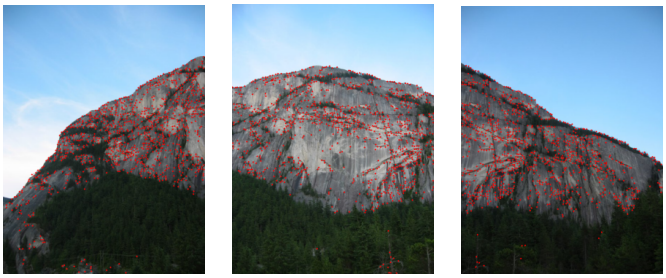

Fig. 2. ANMS with 1000 Best Points

C. Feature Descriptors Generation

After the ANMS step, Nbest corners are obtained. The process involves iterating through each corner, applying a custom feature descriptor that manipulates small image patches of size 41x41. Techniques such as padding, Gaussian blurring, and subsampling are employed to enhance feature representation. The resulting features are normalized and organized into arrays, along with their corresponding positions. This feature extraction mechanism lays the groundwork for subsequent stages in image analysis, aiding robust corner matching.

D. Feature Matching

We have obtained and saved the standardized Feature Descriptors for all images as 64x1 feature vectors. This step includes capturing images (img1 and img2), feature vectors

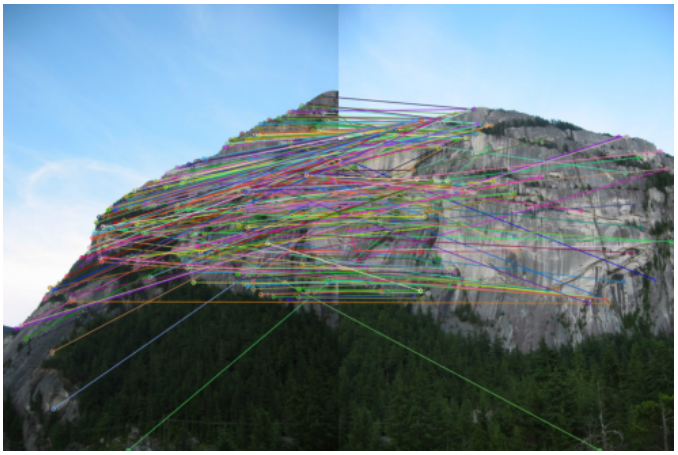

Fig. 3. Image 1 and Image 2 Matched

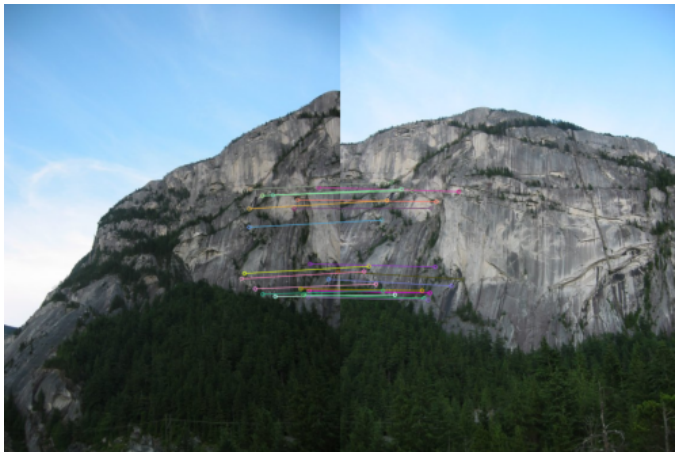

Fig. 4. Image 1 and Image 2 RANSAC Result

(feat1 and feat2), and feature positions(feat pos1, feat pos2). In addition, a threshold parameter is specified to eliminate poor matches. For each feature in the first set, an exhaustive search of the features in the second set is performed to determine the best and second-best matches based on the sum of square distances. Matches are retained if the best match distance to second-best match distance ratio is less than the threshold. Figure 3 shows the result after feature matching.

E. RANSAC for outlier rejection and robust Homography estimation

We calculate Homography, which uses Singular Value Decomposition (SVD) to factor the matrix. A candidate homography matrix is estimated using iterative random sampling of four matches, and inliers are identified by transforming the matched points. This process is repeated multiple times, and the set of inliers with the highest count is deemed the best estimate. The results, which were visualized using draw matches, show that RANSAC is effective in improving the accuracy of feature matching between images.

Algorithm 2 RANSAC

0: **while** $iterations < Nmax$ and $percentageofinliers < 90\%$ **do**
1: Select four feature pairs (at random), p_i from image 1, p_i' from image 2;
2: Compute homography H between the previously picked point pairs;
3: Compute inliers where $SSD(p_i', H \cdot p_i) < \tau$, where τ is some user-chosen threshold and SSD is the sum of square difference function;
4: Increment iterations;
4: **end while**
Repeat the last three steps until exhausted Nmax number of iterations or found more than 90% inliers;
Keep the largest set of inliers;
Re-compute least-squares $\hat{H}$ estimate on all of the inliers.
=0

Fig. 5. Image Stitching and Blending

F. Image Stitching

The image stitching process starts with computing the transformation of key points from the first image to align with the second image using the Homography matrix which was calculated in the previous step. The bounding box is determined and a translation matrix is generated to move the stitched to the positive quadrant. The 2 images are then combined to give a stitched look.

G. Image Poisson Blending

The calculations for transformation and bounding boxes are similar to those for stitching. A translation matrix is used to align 1st and 2nd images, and a mask is created to isolate the region from the first. The isolated region is combined with the corresponding region in the stitched image, and the *cv2.seamlessClone* function is used to achieve smooth blending. Figure 5 shows a composite image that seamlessly blends the contents of both images.

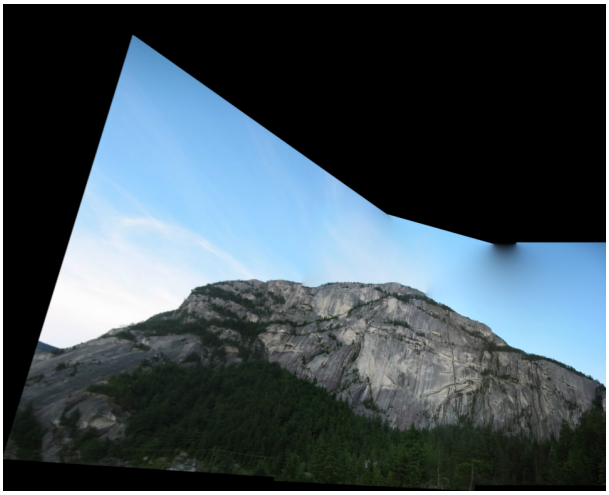

Fig. 6. Final Result of Panorama Stitching

Fig. 7. Panorama Stitching for Museum Set

H. Final Results from Train and Test Sets

I. Multiple Image Stitching

We developed horizontal image stitching for multiple-image stitching, assuming that the images are ordered from left to center; otherwise, we order them brute-force. We begin the algorithm by determining the central image in the directory, then begin panorama stitching from center to left by iterating backward, and save the result as a copy in the left stored image. When it has finished stitching all of the left images, we send this left stored image to the right. We repeat the process of stitching the appropriate image, and the majority of the images are beautifully stitched and blended. This approach works well for six images at most; after six, the perspective transform distorts the result.

The case for checkboard blending is the noisy pixels of the carpet take up all corners and due to abrupt changes in

Fig. 8. Custom Set 1 Blended Image

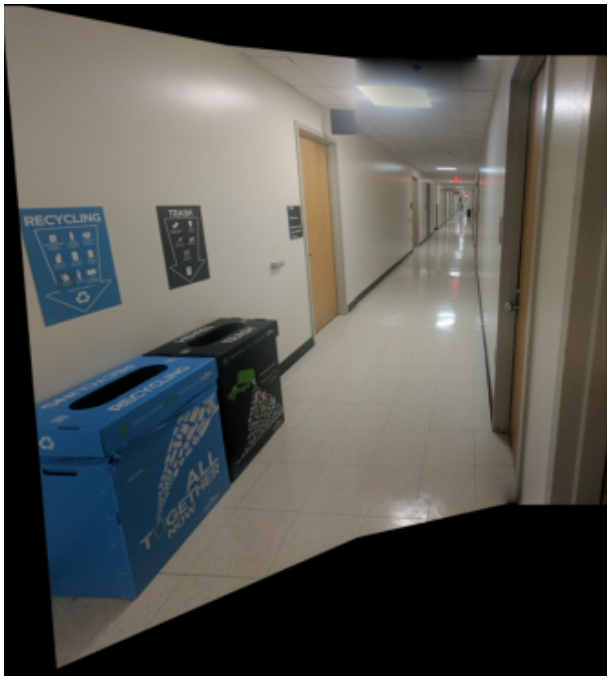

Fig. 9. Test Set 3 Blended Image

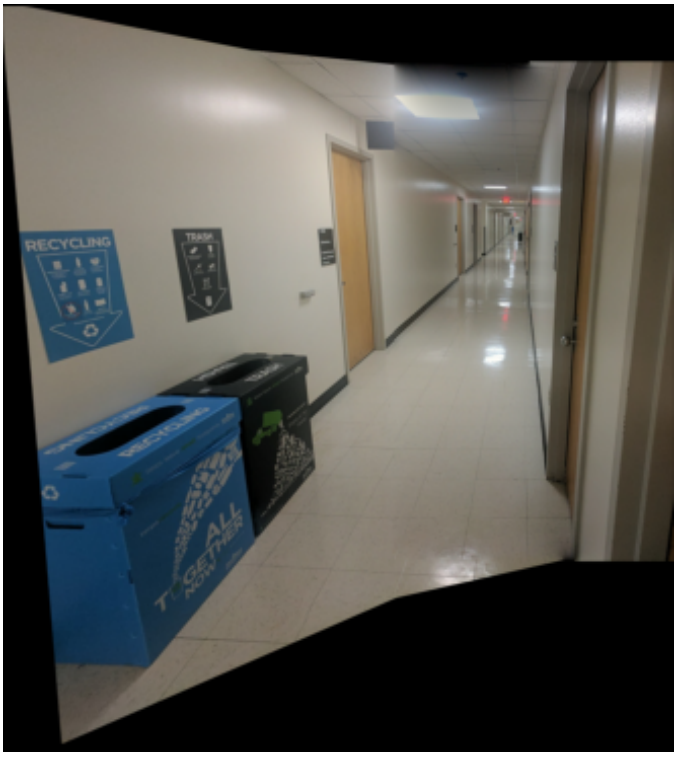

Fig. 10. Caption

pixel intensity for checkerboard the algorithm doesn't blend appropriately.

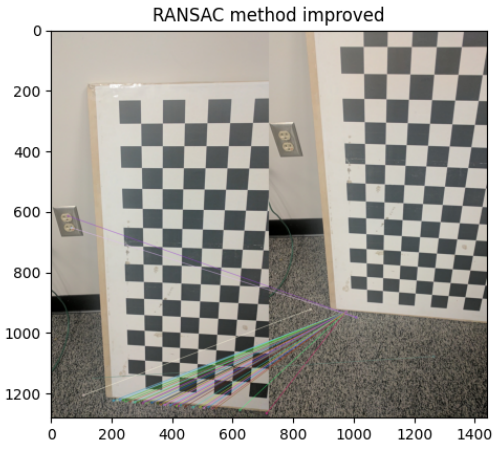

Fig. 11. Failure case of Test Set2 Checkboard RANSAC output

II. Phase 2: Deep Learning Approach

A. Data Generation

A patch generation algorithm was implemented to create the test, validation, and testing dataset for both the supervised and unsupervised models.

Algorithm 3 Patch Generation

Input: $patch\_size$, $perturb\_max$, $pixel\_buffer$
Output: Original and Transformed Patches
for each image in the dataset **do**
  for each valid patch coordinate **do**
  Step 1: Define the coordinates of a patch on the original image
  Step 2: Generate random perturbations for each corner of the patch
  Step 3: Create perturbed patch coordinates using the original coordinates and perturbations
  Step 4: Calculate the perspective transform (H) between the original and perturbed coordinates
  Step 5: Apply the inverse transform to the original image to obtain the transformed image
  Step 6: Extract patches from both the original and transformed images
  end for
end for=0

B. Supervised Approach

The architecture used for the supervised approach is VGG-16 structure. The loss function is a simple MSELoss on the model output. Aside from MSELoss, we also use the MACE metric from the original paper. This metric is used in evaluating and comparing the training for both models. After the training is complete, a separate code calculates EPE loss for all datasets (train, validation, and test).

We used the Adam optimizer with a learning rate of 0.0001. Each training epoch evaluates and learns from 5000 training patch pairs and reports loss on 1000 patch pair validation. Batch size is 64. We trained both models for 100 epochs which took 15 hours in total on an RTX3090 GPU. As can be seen from the graphs, 100 epochs were more than enough to get the model to converge.

$$L_{2loss} = \left\| \mathbf{H}_{4points} - \tilde{\mathbf{H}}_{4points} \right\|_2 \tag{1}$$

C. Unsupervised Approach

The model architecture for the Unsupervised model is kept the same as the Supervised model. All the other hyperparameters are also the same (100 epochs, 64 batch size, Adam optimizer with 0.0001 learning rate). The only difference is in the calculation of the loss function. In this model, since the true perturbed corners are not evaluated, a loss function based on photometric loss is used. Again, as can be seen from the training and validation losses and accuracy measures, this model converges.

$$L1_{loss} = \left\| \tilde{P}_A - P_B \right\| \tag{2}$$

	Supervised Model			Unsupervised model		
	Train	Val	Test	Train	Val	Test
EPE	30.535	54.943	55.466	33.308	52.057	52.541
Run time(ms)	26.4	26	25	27	28	28

TABLE I
MODEL PERFORMANCE COMPARISON

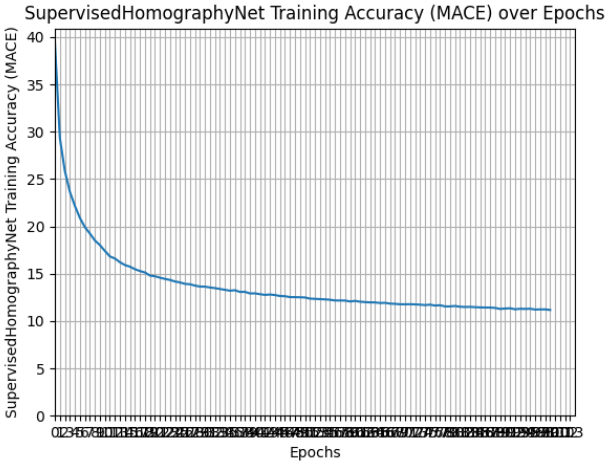

Fig. 12. Training Dataset: Mean Average Corner Error vs Epochs

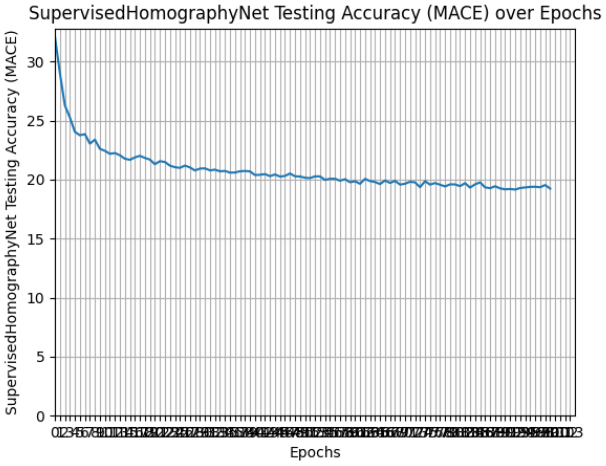

Fig. 14. Validation Dataset: Mean Average Corner Error vs Epochs

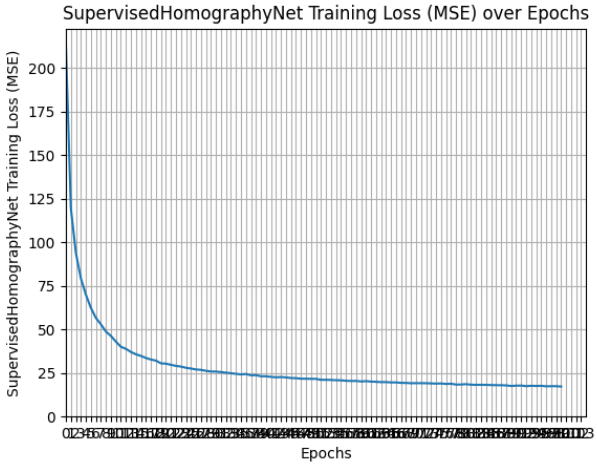

Fig. 13. Training Dataset: Mean Squared Error (Loss) vs Epochs

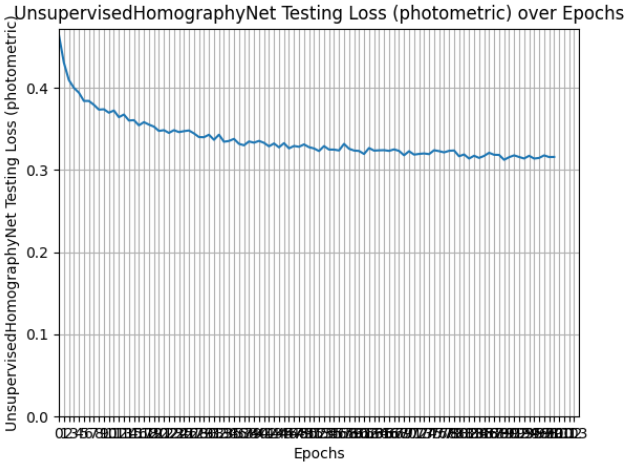

Fig. 15. Validation Dataset: Mean Squared Error (Loss) vs Epochs

D. Video output Results

In this section, we share the results of our deep learning based image stitching algorithm. Overall, it didn't perform as well as the traditional one, probably due to low accuracy estimation of homography.

E. Discussion and Conclusion

In this study, we conducted a comprehensive performance evaluation of homography estimation methods. Figure 12 depicts a comparison between selected/perturbed patch (selected in red, perturbed in green) and four-corner sets fCB computed using supervised (blue), unsupervised (yellow), and feature-based traditional (light blue) methods. The supervised method consistently provides the most accurate estimates, whereas the feature-based approach is unstable depending on the number of features and their matching status. The unsupervised method, while more stable than the feature-based approach, consistently produces estimates that are inferior to the supervised method. Table I shows the average Endpoint Error (EPE) and runtime results, demonstrating that the supervised model outperforms the unsupervised model without overfitting. Traditional methods perform well when features are accurately identified. The unsupervised method shows superior generalization and video stitching results indicate

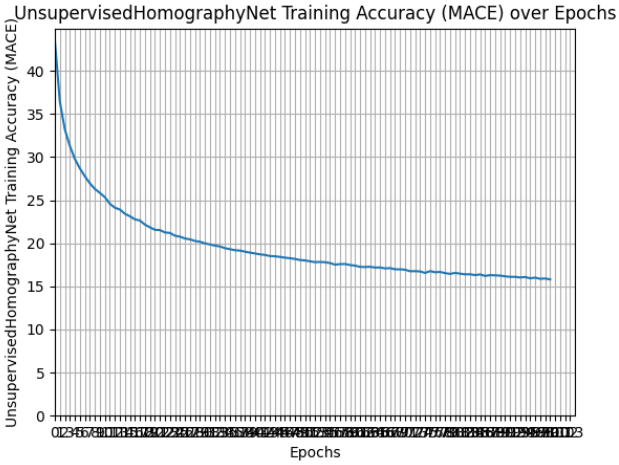

Fig. 16. Mean Average Corner Error(Accuracy) vs Epochs

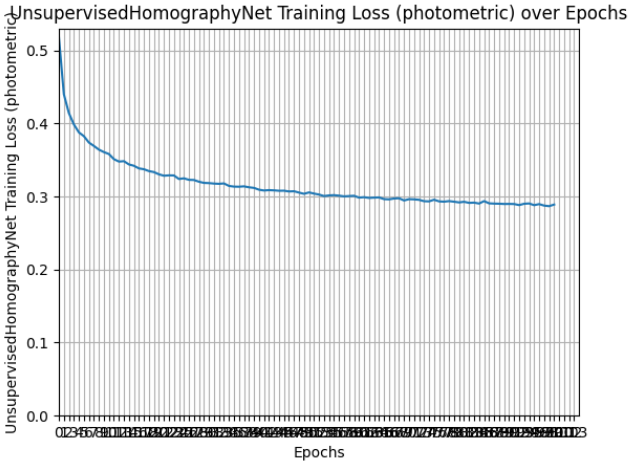

Fig. 17. Mean Squared Error(Loss) vs Epochs

potential improvements with a better-tuned model.

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.

References

[1] D. DeTone, T. Malisiewicz, and A. Rabinovich, "Deep Image Homography Estimation." arXiv, 2016. doi: 10.48550/ARXIV.1606.03798. [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, 2017. doi: 10.48550/ARXIV.1709.03966.

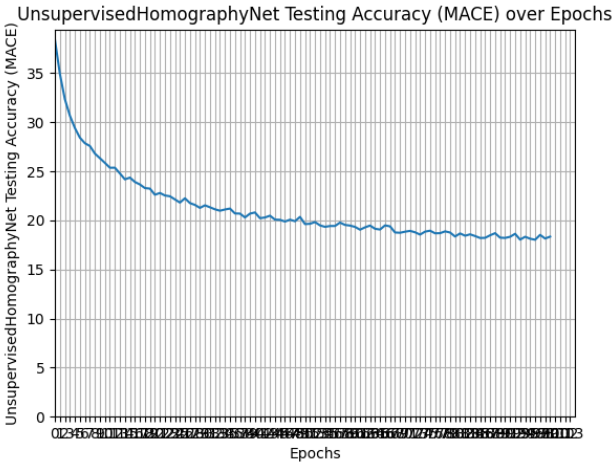

Fig. 18. Mean Average Corner Error(Accuracy) vs Epochs

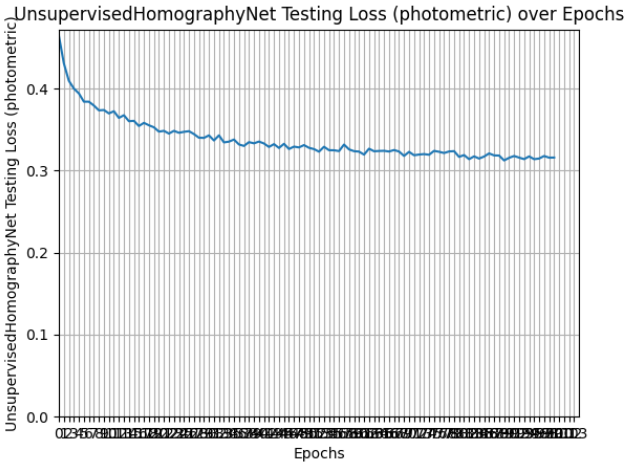

Fig. 19. Mean Squared Error(Loss) vs Epochs

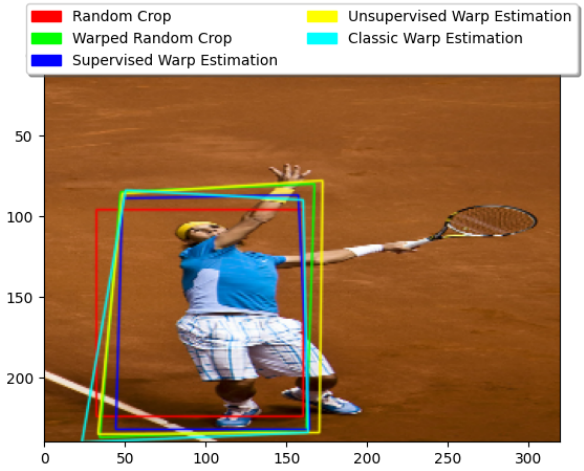

Fig. 20. Train Set Image output of Traditional, Supervised and Unsupervised approaches on Homography estimation comparison

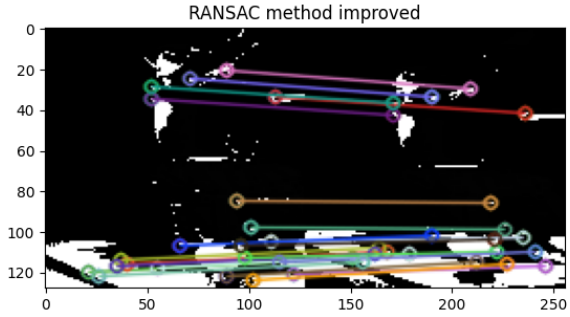

Fig. 21. Classical Approach RANSAC output

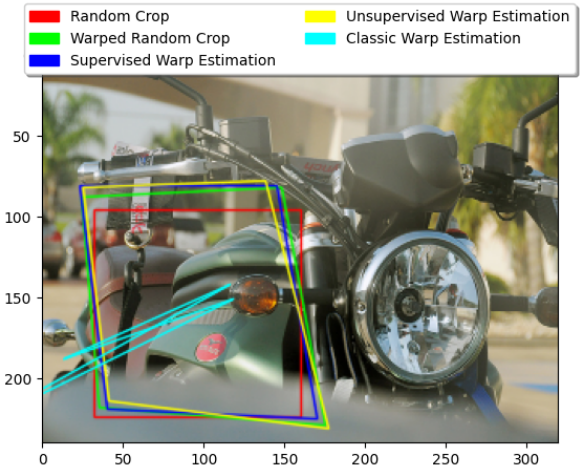

Fig. 22. Test of Traditional, Supervised and Unsupervised approaches on Homography estimation where Traditional approach is outperformed by Deep Learning methods because the feature matched corners are not good

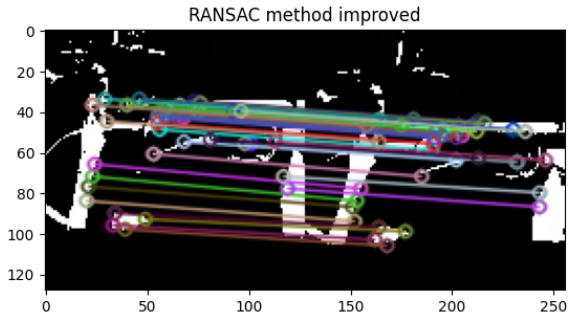

Fig. 23. Corresponding RANSAC output

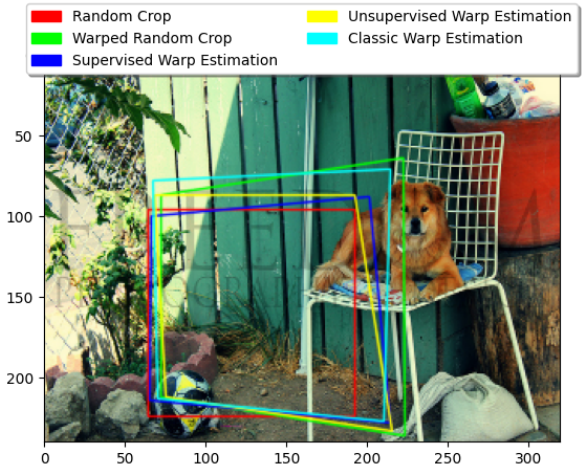

Fig. 24. Test Set Image output of Traditional, Supervised and Unsupervised approaches on Homography estimation comparison

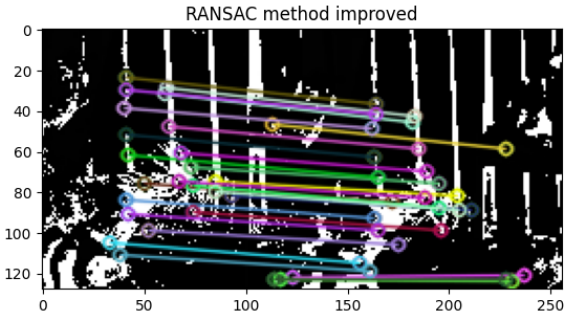

Fig. 25. Corresponding RANSAC output for Comparison

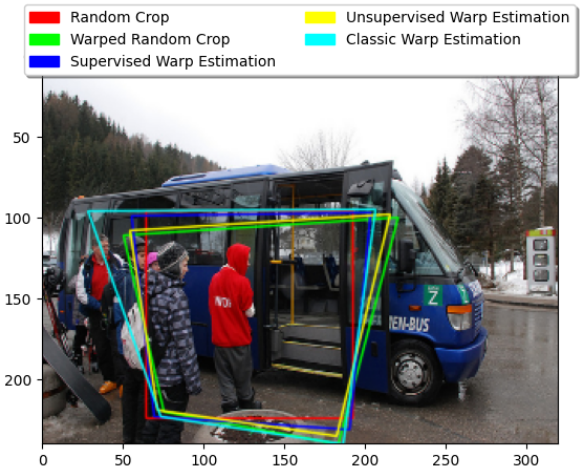

Fig. 26. Validation Set Image output of Traditional, Supervised and Unsupervised approaches on Homography estimation comparison

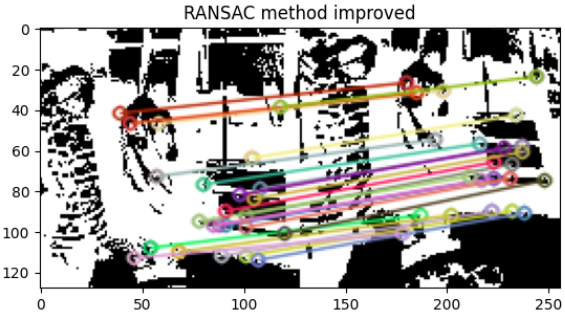

Fig. 27. Corresponding RANSAC output for Comparison

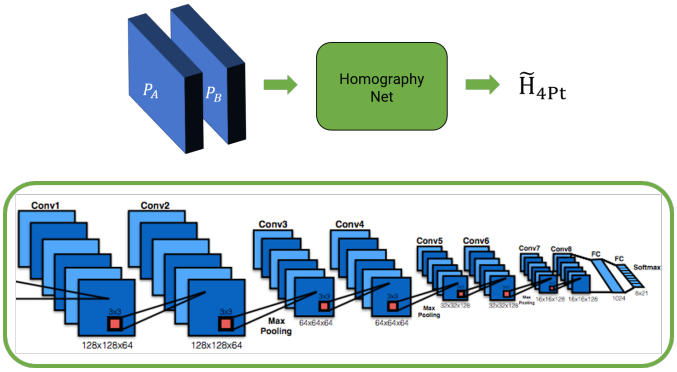

Fig. 28. Supervised Model

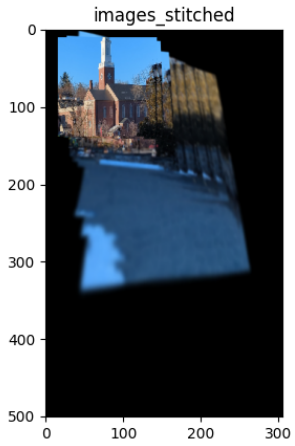

Fig. 29. Tower Image Stitched

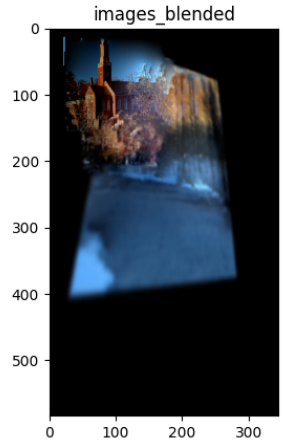

Fig. 30. Tower Image Blended

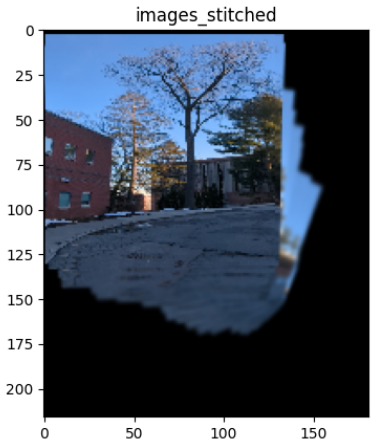

Fig. 31. Trees Image Stitched

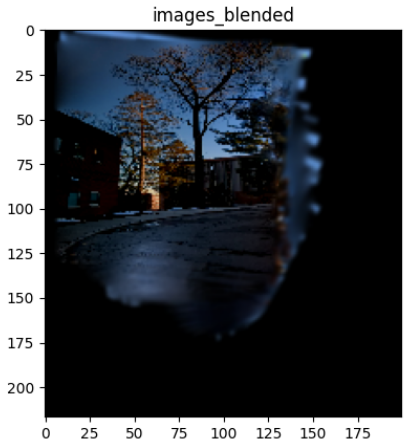

Fig. 32. Trees Image Blended

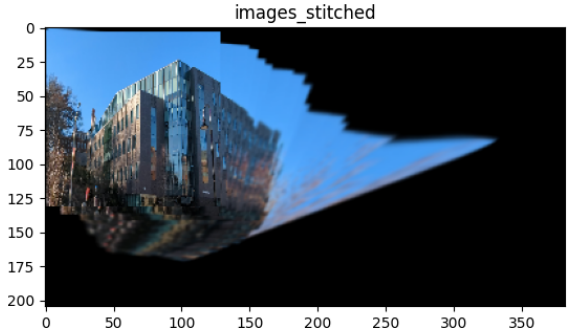

Fig. 33. Unity Hall Building Stitched

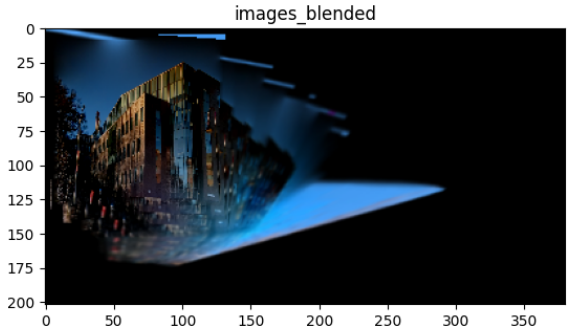

Fig. 34. Unity Hall Blended