CMSC733 Project1: MyAutoPano!

Srinidhi Sreenath
Robotics Graduate Student
University of Maryland

John Kanu
PhD in Computer Science
University of Maryland

***Abstract*—The aim of this project is to implement an end-to-end pipeline to perform panorama stitching given a set of unordered images using both traditional computer vision techniques and the deep learning approach.**

PHASE 1 - PANAROMA USING TRADITIONAL COMPUTER VISION TECHNIQUES

The first step is to capture a set of images to be stitched into a panorama. A set of images to be stitched for panorama is shown below.

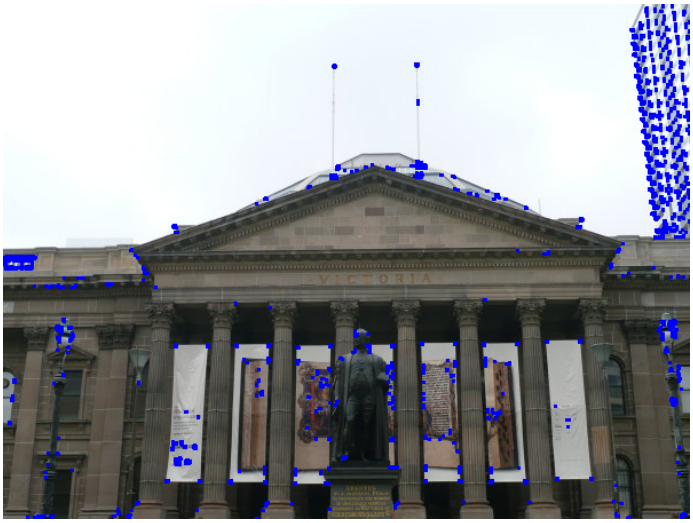
The stitching process is explained using images 1 and 2.

Corner Detection

The aim of this step is to detect corners spread all across the image to avoid weird artifacts in warping.

Harris features: The first method used here is harris corners. The images are converted to grayscale from color for corner detection. A grayscale image is shown below.

Harris features were extracted using cv2.cornerHarris with block size = 2 and aperture parameter for Sobel operator = 3. To get the optimal corners, a threshold of 0.01 times the max value of the harris detections. The corners detected on the images are marked and shown below.



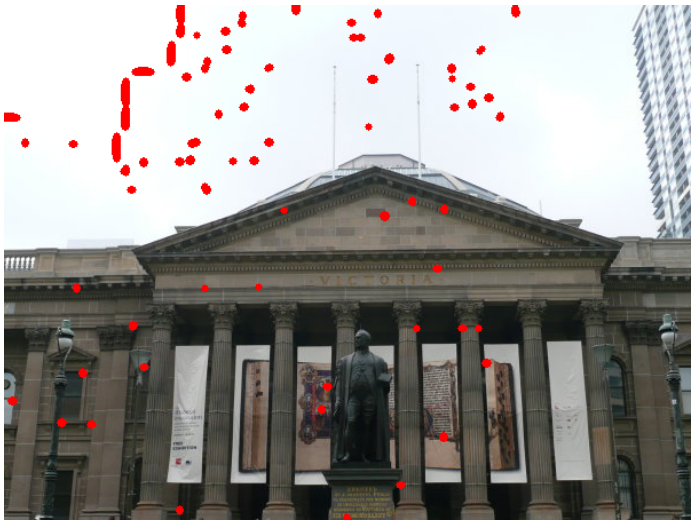
Adaptive Non-Maximal Suppression (ANMS): The objective of this step is to detect corners such that they are equally distributed across the image in order to avoid weird artifacts in warping.

Since the amount of corners detected are more and not equally spaced, there will be weird artifacts in warping. To avoid that, out of the N strong corners which are local maximas, a set of best corners is chosen by implementing the ANMS algorithm.

ANMS algorithm is implemented to choose 100 best corners. The coordinates for strong corners is extracted as follows:

- a maximum filter is applied to the grayscale image to get local maxima values. Input to define the filter size is chosen as 25. When a small filter input size was used, a lot of local maxima is detected (in range of 10000). A large filter input size reduced the number of local maxima coordinates to obtain (reduced to range of 3000).
- The local max values are used to create a binary mask.
- The local maxima coordinates are then extracted from the binary mask.

After the local maximas are obtained, the ANMS algorithm is used to detect 100 strong points. The output of the ANMS detected strong corners are as follows:



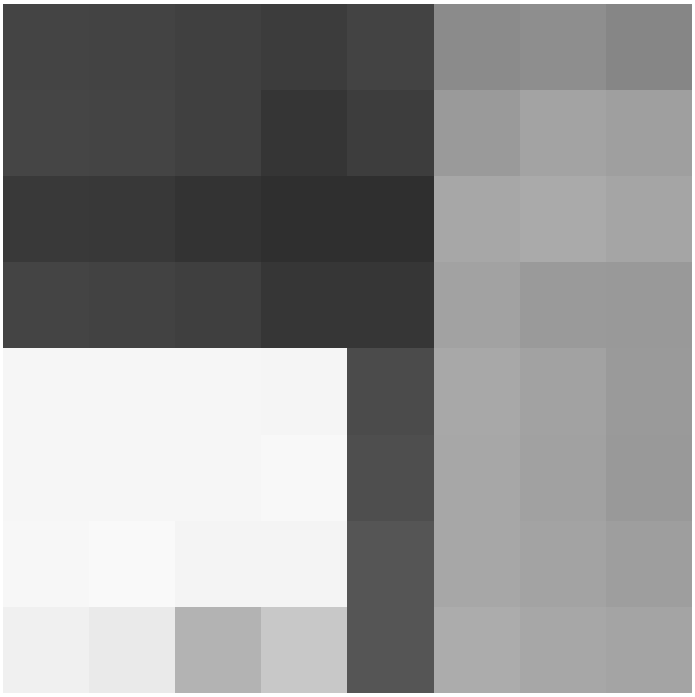
OpenCV's good features to track: OpenCV has inbuilt implementation to detect strong corners using cv2.goodFeaturesToTrack. 500 corners were detected using this, with quality level 0.01 and minimum distance between each corner as 10. The output of the corners detected is shown below:

The above results are better than the ANMS output and hence the strong corners obtained from this for feature description and matching.

Feature Description

Once the strong corners are obtained, each is now a feature point and has to be described by a feature vector to encode the information of each feature by a vector. The following steps are used to describe each feature by a vector:

- a patch of size $40x40$ is extracted around the feature point.
- A Gaussian filter is applied to patch of kernel size $(5, 5)$
- The patch is resized to an $8x8$ image. A sample of such a patch is shown below:



- The patch is reshaped to a $64x1$ vector.
- The vector is standardized to have zero mean and variance of 1. Standardization is used to remove bias and to achieve some amount of illumination invariance. This vector now represents the feature point.

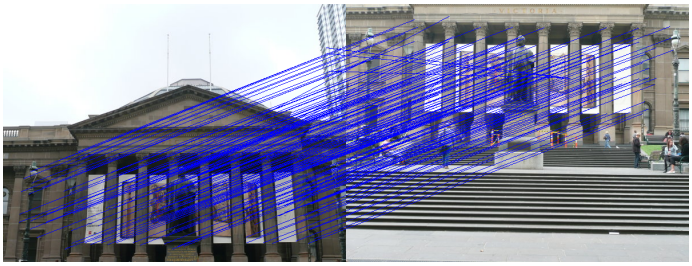
Feature Matching

Now that each feature point is described by a feature vector, a feature point in first image is to be matched with the corresponding feature point in image 2. To do this, for each feature point in image 1, the sum of squared distances (SSD) for the corresponding feature vector is calculated with each and every feature vector in image 2.

Once the SSD's are calculated, the minimum and second minimum SSD values are obtained. A ratio is defined which is basically minimum SSD value over second minimum SSD value. If the ratio is below a threshold, then the feature point is matched with the feature point corresponding to minimum SSD value.

A threshold value of 0.85 was used initially and a lot of mismatches still occurred. The threshold is now chosen to be 0.5 which gave sufficient matches and very less mismatches.

The matches obtained are shown below:

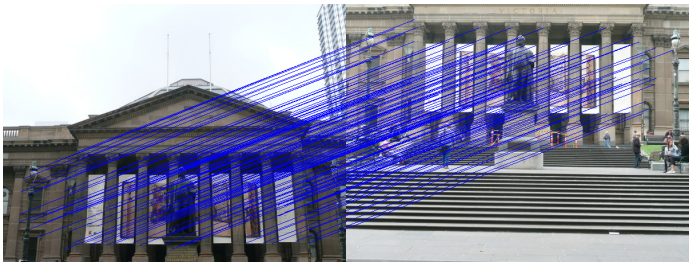


RANSAC for outlier rejection and to estimate Robust Homography

To eliminate some of the mismatches, a robust method called Random Sample Concensus or RANSAC to compute homography. The RANSAC algorithm is as follows:

- Select four feature pairs (at random), p_i from image 1, $p_i^{'}$ from image 2.
- Compute homography H between the previously picked point pairs. This was done using the cv2.getPerspectiveTransform.
- Compute inliers where $SSD(p_i^{'}, H * p_i) < 60.0$. where 60 is a threshold.
- The above steps are repeated for either 10 iterations or when 90% of the points are found to be inliers.
- The set of inliers is then used to show the matching.

The matches obtained after RANSAC are shown below:



The least squares homography matrix is recomputed then using only the inlier points by direct linear transform. Instead of using SVD of the system, a known constraint is used. The last element of the homography matrix is known to be 1 i.e $h_9 = 1$. This is used to set and convert P matrix into $9x9$ matrix and system will become non-homogeneous.

$$PH = \begin{bmatrix} -x_1 & -y_1 & -1 & 0 & 0 & 0 & x_1x_1' & y_1x_1' & x_1' \\ 0 & 0 & 0 & -x_1 & -y_1 & -1 & x_1y_1' & y_1y_1' & y_1' \\ -x_2 & -y_2 & -1 & 0 & 0 & 0 & x_2x_2' & y_2x_2' & x_2' \\ 0 & 0 & 0 & -x_2 & -y_2 & -1 & x_2y_2' & y_2y_2' & y_2' \\ -x_3 & -y_3 & -1 & 0 & 0 & 0 & x_3x_3' & y_3x_3' & x_3' \\ 0 & 0 & 0 & -x_3 & -y_3 & -1 & x_3y_3' & y_3y_3' & y_3' \\ -x_4 & -y_4 & -1 & 0 & 0 & 0 & x_4x_4' & y_4x_4' & x_4' \\ 0 & 0 & 0 & -x_4 & -y_4 & -1 & x_4y_4' & y_4y_4' & y_4' \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} h1 \\ h2 \\ h3 \\ h4 \\ h5 \\ h6 \\ h7 \\ h8 \\ h9 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

This is solved like a $Ax = b$ system and the homography matrix is obtained.

Warping and blending the images

The next step is to warp one image wrt another to transform the first image to the plane of the second image using the homography matrix obtained. The first image is warped to the plane of second image and the warped image is shown below:

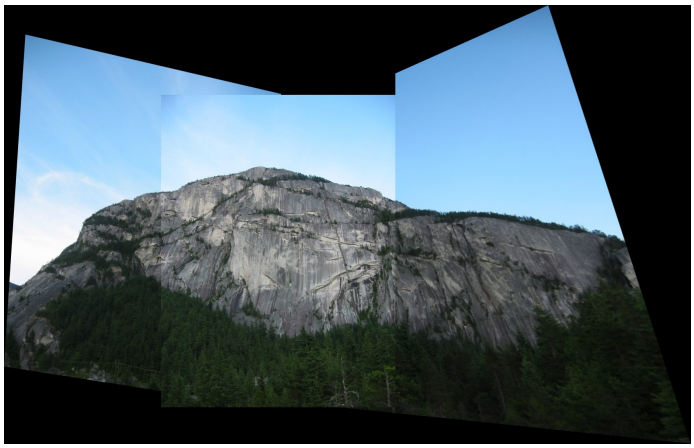
To blend the images, the new translation for the warped image origin is calculated using the Homography matrix i.e $translatedorigin = H * origin$ where $origin = [0, 0, 1]$. The translation coordinates are used to compensate the origin of the second image and then the second image is simply placed at the compensated coordinates as it origin.

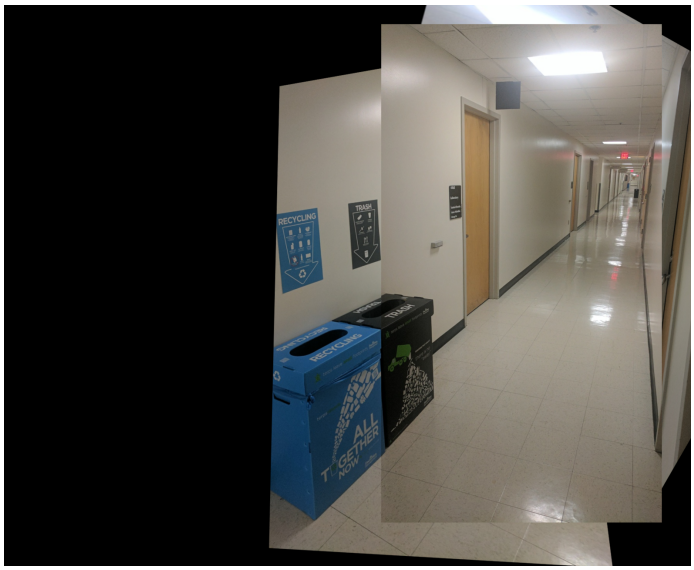
Sometimes, there will be less amount of matched features which results in RANSAC unable to be performed. Therefore when the matched features are very less, the image is ignored for stitching/ blending.

The final blended image is shown below:

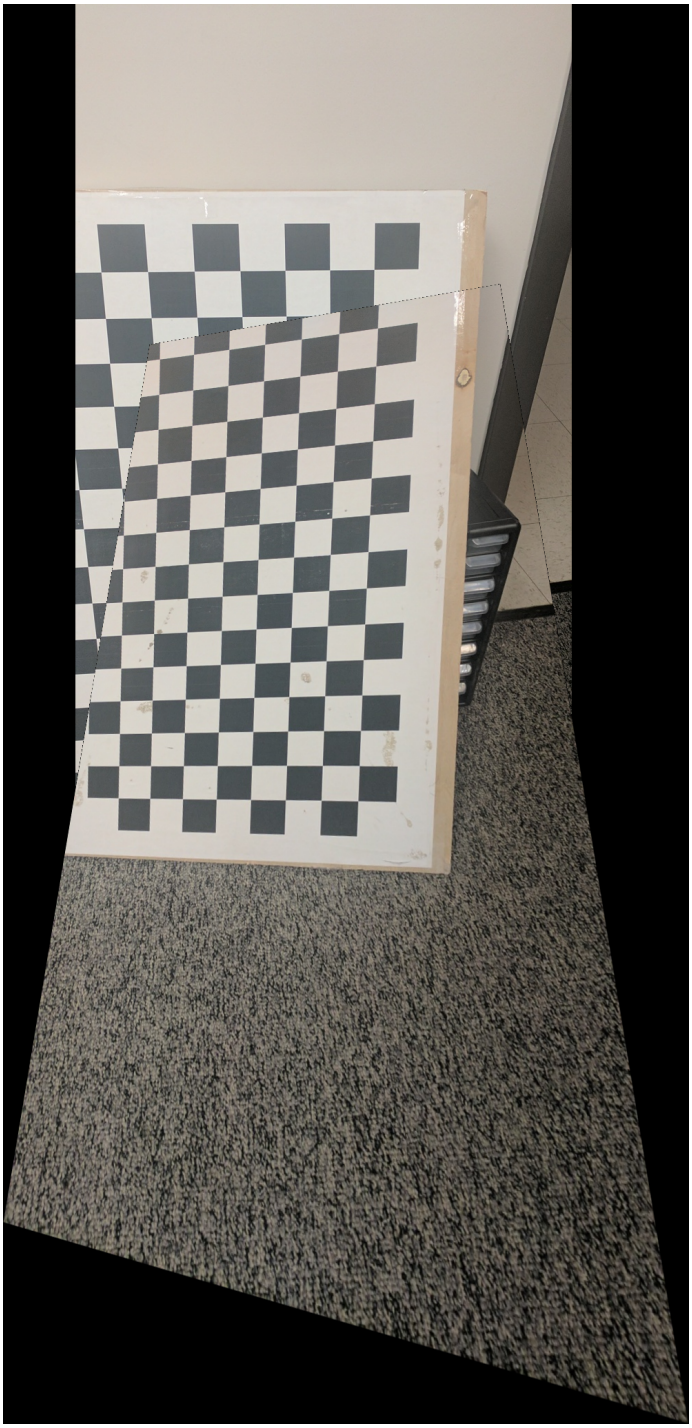
The final panorama constructed is shown below:

Some other panoramas stitched are shown below:

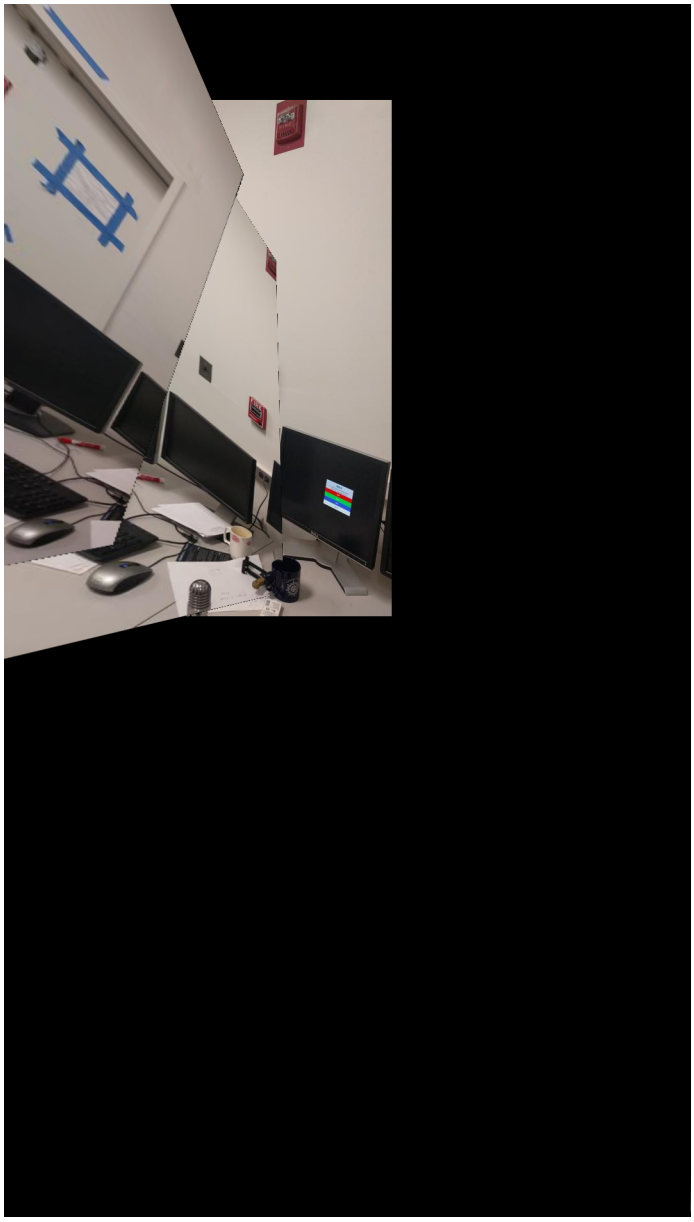




The images are read and the middle image is found and left stitching and right stitching is performed. However, for chess board set, the result was better when the images were read sequentially.



When the number of images are too high, some images aren't stitched because when matched features are less, the image is ignored for blending.



II. Phase 2: Deep Learning Approach

In this phase, we implemented two neural networks for homography estimation. One is supervised, and the other is unsupervised.

A. Data generation

Image patches were selected uniformly randomly within the patchable region of the image. Each patch is 64 pixels in height and widght.

Color channels are averaged to convert each image to grayscale. Converting each color to grayscale reduces the number of channels, and therefore reduces training time. We assume that collapsing the color channels does not present a significant loss of information.

Patches are warped as described in the project specifications, with a maximum perturbation amount of 10 pixels.

B. Supervised Approach

In this section we implemented the same network architecture as described in [1]. The network takes a 64x64x2 tensor as input representing the two warped and unwarped patches. The network consists of 8 convolutional layers and 2 fully-connected layers. The final layer predicts the 8-point vector representing the difference between the corner points of the patches.

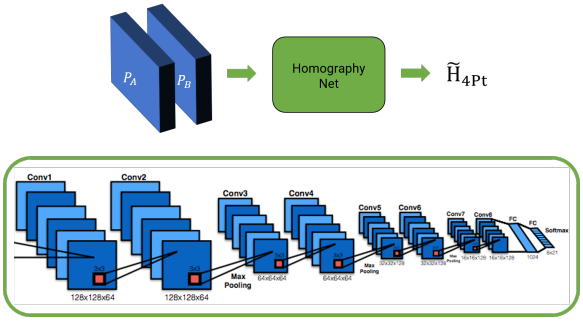


Fig. 1. Supervised network architecture

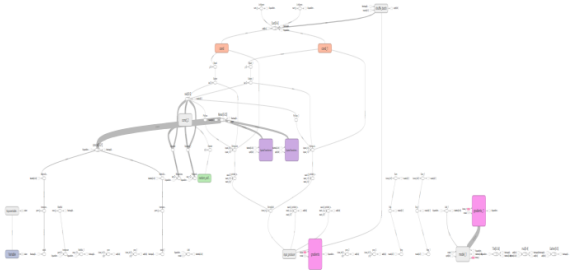


Fig. 2. Tensorboard graph

1) Training: We trained the network using 2 Nvidia Tesla P100 GPUs on the Google Cloud Platform. We minimized the L2-norm of the difference between the predicted corner point translations $\widetilde{H_{4Pt}}$ and the actual translations H_{4Pt}. The loss function is given as

$$||\widetilde{H_{4Pt}} - H_{4Pt}||_2$$

.

Figure 3 shows the loss curve over 3000 iterations.

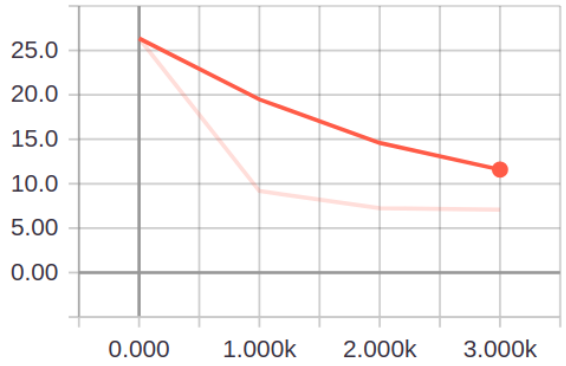


Fig. 3. Supervised network loss

C. Unsupervised Approach

In this section we also implemented the same network architecture as described in [1]. The network takes the same concatenation of two patches and outputs the warped patch.

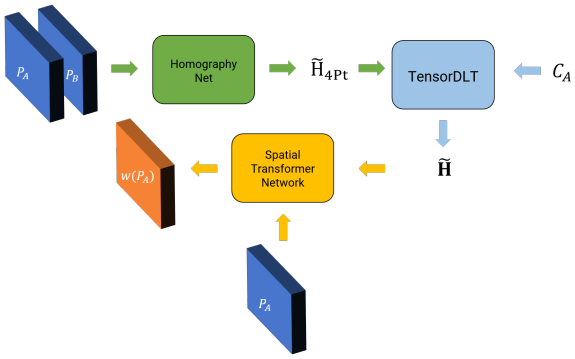


Fig. 4. Supervised network loss

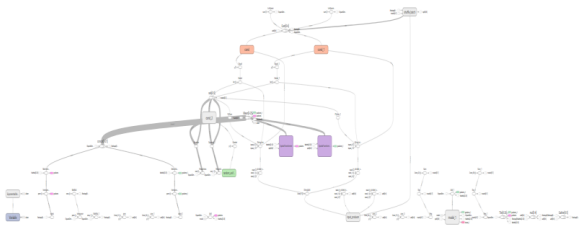


Fig. 5. Tensorboard graph

1) Training: We trained the network using 2 Nvidia Tesla P100 GPUs on the Google Cloud Platform. We minimized a photometric loss function, the L1-norm of the difference between the predicted image $w(P_A)$ and the original patch P_A. The loss function is given as

$$||w(P_A, H_{4Pt}) - P_B)||_1$$

Figure 10 shows the loss over 3000 iterations.

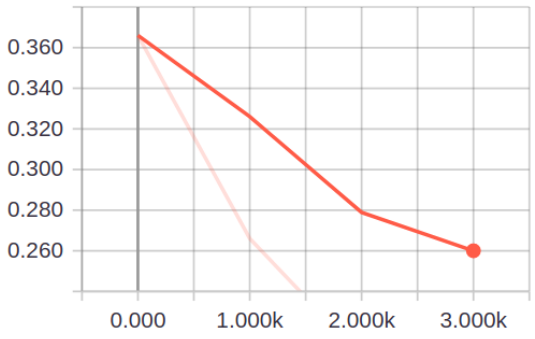


Fig. 6. Unsupervised network loss

We finally present the loss scores after 3000 iterations.

Network	Train	Validation	Test
Unsupervised	10.9343	10.9743	10.8850
Supervised	11.3791	11.4471	11.4067

2) Estimated homographies: Below are images from the test dataset, overlayed with the homography estimated by our supervised learning model and unsupervised learning model.

Fig. 7. Supervised

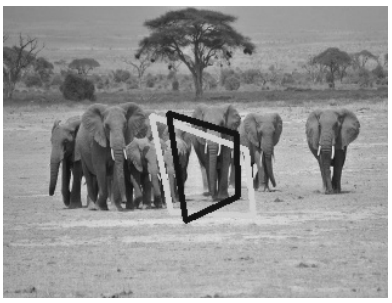


Fig. 8. Supervised

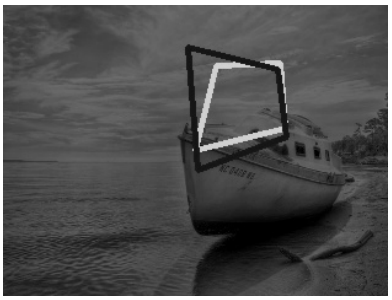


Fig. 9. Supervised

Fig. 10. Supervised

Fig. 11. Unsupervised

Fig. 12. Unsupervised

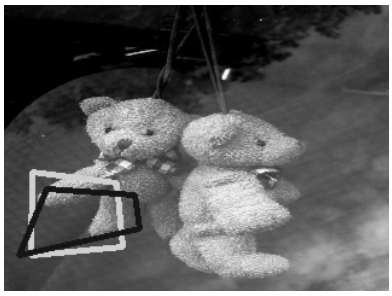


Fig. 13. Unsupervised

Fig. 14. Unsupervised