RBE549 Project1 My AutoPano

Haoying Zhou
Department of Robotics Engineering
Worcester Polytechnic Institute
Worcester, MA, 01609
Email: hzhou6@wpi.edu

Zhentian Qian
Department of Robotics Engineering
Worcester Polytechnic Institute
Worcester, MA, 01609
Email: zqian@wpi.edu

I. Phase 1: Traditional Approach

A. Corner Detection

Let the 2-D grayscale image denoted by I. Consider taking an image patch $(x, y) \in W$(window) and shifting it by (u, v). The sum of squared differences (SSD) between these two patches, denoted $E(u, v)$, is given by:

$$E(u,v) = \sum_{(x,y)\in W} \Big(\underbrace{I(x,y)}_{\text{intensity}} - \underbrace{I(x+u,y+v)}_{\text{shifted intensity}} \Big)^2 \tag{1}$$

$I(x+u, y+v)$ can be approximated by a Taylor expansion. Let I_x and I_y be the partial derivatives of I, such that

$$I(x+u,y+v) \approx I(x,y) + I_x(x,y)u + I_y(x,y)v \tag{2}$$

This produces the approximation

$$E(u,v) \approx \sum_{(x,y)\in W} \left(I_x(x,y)u + I_y(x,y)v\right)^2, \tag{3}$$

which can be written in matrix form:

$$E(u,v) \approx \begin{bmatrix} u & v \end{bmatrix} M \begin{bmatrix} u \\ v \end{bmatrix}, \tag{4}$$

where M is the structure tensor,

$$M = \sum_{(x,y)\in W} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \tag{5}$$

Then the Harris [1] response, which determines if a window can contain a corner or not, is calculated as:

$$R = \lambda_1\lambda_2 - k(\lambda_1+\lambda_2)^2 = \det(M) - k\,\mathrm{tr}(M)^2 \tag{6}$$

where k is an empirically determined constant $k \in [0.04, 0.06]$. Large Harris response R indicate the presence of a corner.

The output for the corner dection is shown in Figure 1. k is set to be 0.04. The threshold for R is set to be 110.

Fig. 1: Output of Harris corner detection

B. Adaptive Non-Maximal Suppression

The algorithm for implementing ANMS is given in algorithm 1. We use the Harris response score R as the corner score Image C_{img}. A custom function is implemented to extract the local maxima. The output of ANMS is visualized in Figure 2. Observe that the output of ANMS is evenly distributed strong corners.

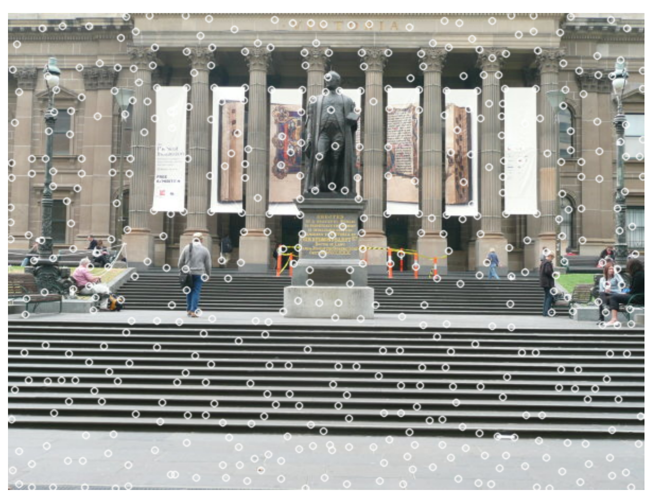


Fig. 2: Output of ANMS algorithm

Algorithm 1: Adaptive Non-Maximal Suppression

Input : Corner score Image (C_{img} obtained using `cornermetric`), N_best (Number of best corners needed)
Output: $(\mathbf{x}_i, y_i)$ for $i = 1 : N_{best}$
Find all local maxima using on C_{img};
Find (x, y) co-ordinates of all local maxima;
Initialize $r_i = \infty$ for $i = [1 : N_{strong}]$;
for $i = [1 : N_{strong}]$ **do**
  for $j = [1 : N_{strong}]$ **do**
    if $C_{img}[y_j, x_j] > C_{img}[y_i, \mathbf{x}_i]$ **then**
      $ED = (x_j - \mathbf{x}_i)^2 + (y_j - y_i)^2$
    end
    if $ED < r_i$ **then**
      $r_i = ED$
    end
  end
end
Sort r_i in descending order and pick top N_{best} points.

C. Feature Descriptor

Gaussian blur is applied to the entire image, as shown in Figure 3. We then take a patch of size 41×41 centered around the feature point. The blurred output is sub-sampled to 8×8 and then reshape to obtain a 64×1 vector. The vector is then standardized by subtracting the mean of the vector and diving by the standard deviation of the vector.

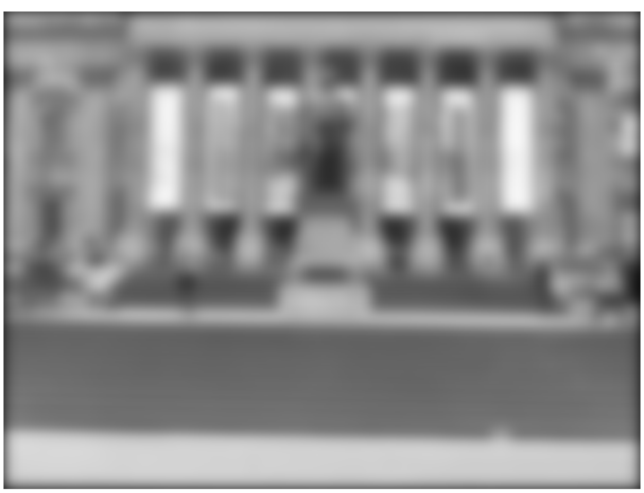


Fig. 3: Gaussian blurred image

D. Feature Matching

Please see Figure 4 for the matched feature between image 2 and 3 in set 1. Observe that there are some wrong matches.

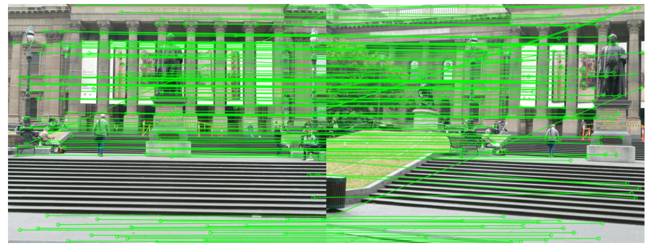


Fig. 4: Output of Feature Matching. Observe the wrong matches.

E. RANSAC for outlier rejection and to estimate Robust Homography

To remove incorrect matches, The homography is computed using Random Sample Concensus algorithm described in algorithm 2.

Algorithm 2: RANSAC

while $iterations < N_{max}$ **do**
  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;
  Compute inliers where $SSD(p'_i, Hp_i) < \tau$, where τ is some user chosen threshold and SSD is sum of square difference function;
  increment $iterations$;
end
Keep largest set of inliers;
Re-compute least-squares $\hat{H}$ estimate on all of the inliers.

The output of feature matches after all outliers have been removed is shown in Figure 5.

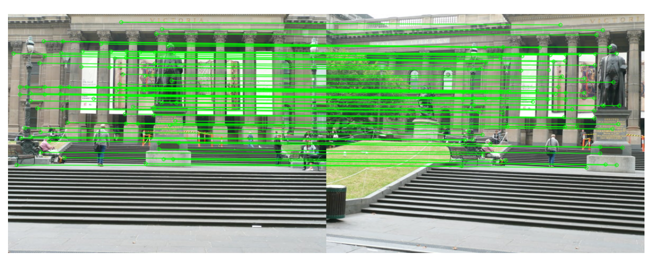


Fig. 5: Feature matches after outliers have been removed using RANSAC.

We implement the Direct Linear Transformation (DLT) [2] algorithm to compute homography H between the picked point pairs, as described in algorithm 3.

Algorithm 3: The basic DLT for H

Objective: Given $n \geq 4$ 2D to 2D point correspondences $\mathbf{x}_i \leftrightarrow \mathbf{x}'_i$, determine the 2D homography matrix H such that $\mathbf{x}'_i = H\mathbf{x}_i$.

Writing $\mathbf{x}_i = (\mathbf{x}_i, y_i, \omega_i)$ and $\mathbf{x}'_i = (\mathbf{x}'_i, y'_i, \omega'_i)$. For each correspondence $\mathbf{x}_i \leftrightarrow \mathbf{x}'_i$ compute the matrix

$$A_i = \begin{bmatrix} \mathbf{0}^T & -\omega'_i\mathbf{x}_i^T & y'_i\mathbf{x}_i^T \\ \omega'_i\mathbf{x}_i^T & \mathbf{0}^T & -\mathbf{x}'_i\mathbf{x}_i^T \end{bmatrix};$$

Assemble the n 2×9 matrices A_i into a single $2n \times 9$ matrix A;

Obtain the SVD of A. The unit singular vector corresponding to the smallest singular value is the solution $\mathbf{h}$. Specifically, if $A = UDV^T$ with D diagonal with positive diagonal entries, arranged in descending order down the diagonal, then $\mathbf{h}$ is the last column of V;

The matrix H is determined from $\mathbf{h}$ as

$$H = \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix};$$

F. Blending Images

The algorithm we proposed for blending images is described in algorithm 4. The final panoramas for set 1, 2, 3 and two of own custom datasets are shown in Figs 6–10.

Fig. 6: The final panoramas for set 1 using traditional method.

Algorithm 4: Image blending pipeline

Input: Arbitrary number of images

Construct a undirected empty(no edges) graph G with image ids as its vertices;

while *G is not connected and all possible combinations of image are not exhausted* **do**

  Select a new pair of images (i, j);

  Compute the homograph H_{ij} between the pair of images;

  if *H_{ij} exists* **then**

    Add edge (i, j) into graph G;

    Store H_{ij} as well as its inverse $H_{ji} = H_{ij}^{-1}$;

  end

end

Select the node r in graph G with maximum number of edges as the common reference frame;

foreach *node $i \neq r$ in graph G* **do**

  if *Path exist between i and j* **then**

    Compute the homograph transformation H_{ir} from image frame i to image frame r

  end

end

Shift the reference frame by $(\Delta x, \Delta y)$ so that for any image i transfromed into the reference frame r, all image pixels have positive coordinates;

Adjust the reference frame size so that for any image i transfromed into the reference frame r would fit ;

foreach *Homograph transformation H_{ir}* **do**

  `/* account for the shift` $(\Delta x, \Delta y)$ `*/`

$$H_{ir} = \begin{bmatrix} 1 & 0 & \Delta x \\ 0 & 1 & \Delta y \\ 0 & 0 & 1 \end{bmatrix} H_{ir};$$

  Transform image i into reference frame r using H_{ir};

end

Copy the image r into its corresponding location in reference frame r.

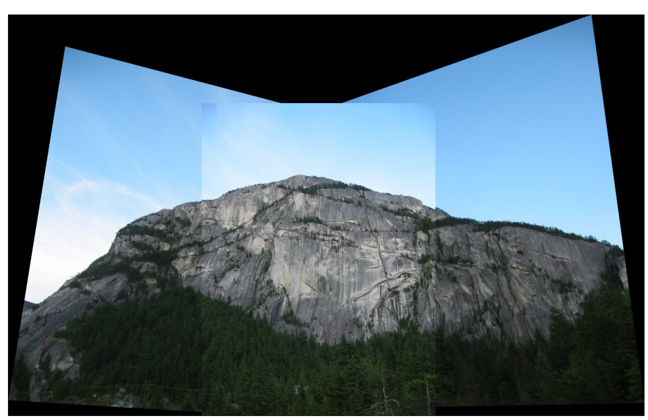


Fig. 7: The final panoramas for set 2 using traditional method.

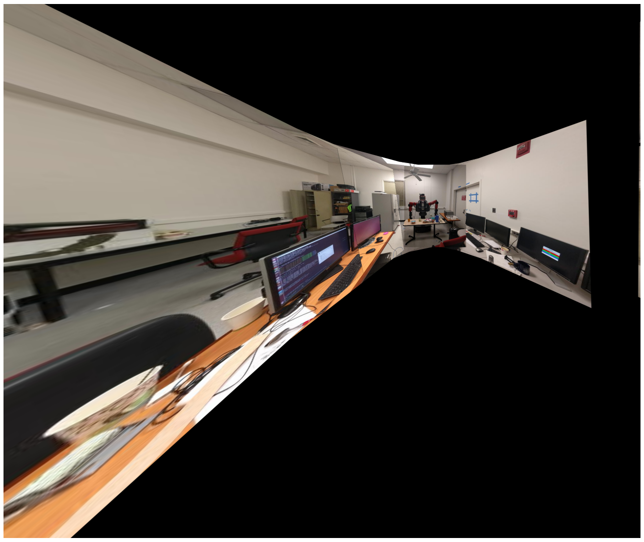


Fig. 8: The final panoramas for set 3 using traditional method.

Fig. 9: The final panoramas for custom set 1 using traditional method.

Fig. 10: The final panoramas for custom set 2 using traditional method.

Fig. 11: The final panoramas for test set 1 using traditional method.

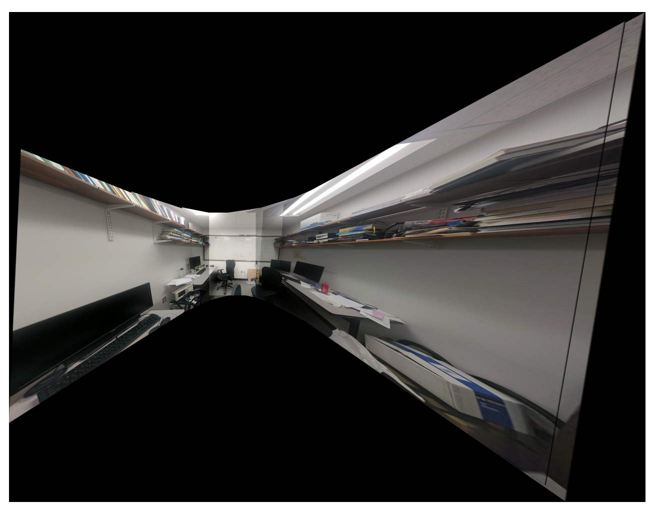


Fig. 12: The final panoramas for test set 2 using traditional method.

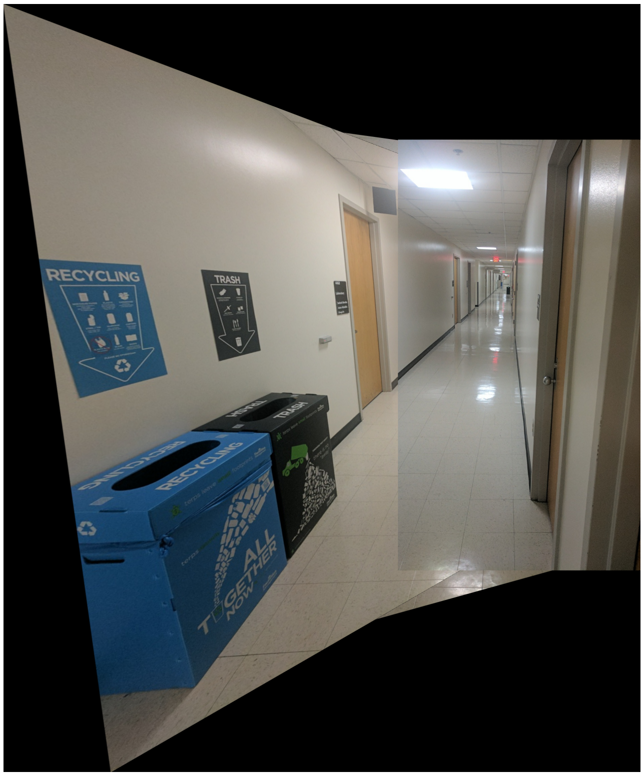


Fig. 13: The final panoramas for test set 3 using traditional method.

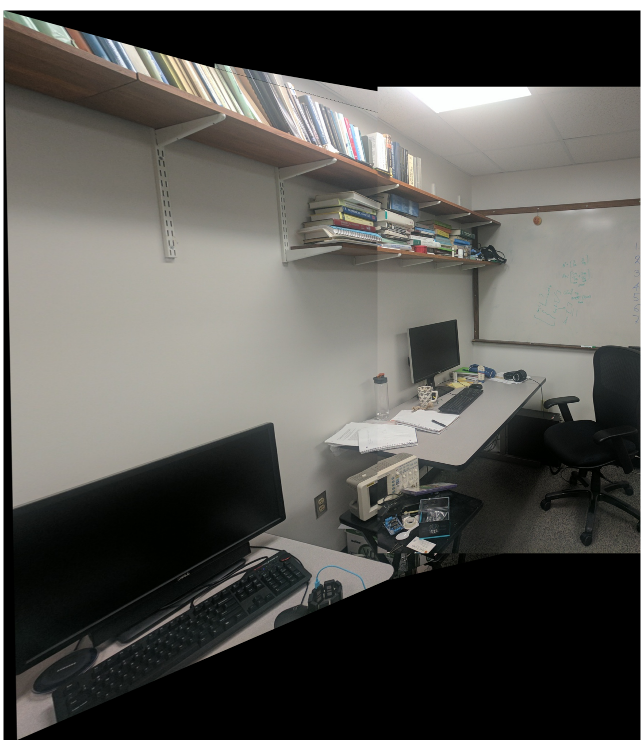


Fig. 14: The final panoramas for test set 4 using traditional method.

II. Phase 2: Deep Learning Approach

A. Data Generation

To generate the data set, we follow the instruction given in the description [3].

We firstly grayscale all the images and resize them to 320×240. Then, we random select a 128×128 patch and implement random noise to the four corners' coordinates within $[-\rho, \rho]$ where $\rho = 32$, the detailed algorithm is shown in algorithm 5

Algorithm 5: Data Generator Algorithm

Input: corners, corners$_{noised}$, a set of images
Output: patch_a, patch_b
  for image in image set **do**
    patch_a $=$ image[corners]
    H = transformation matrix between corners$_{noised}$ and corners
    patch_b $=$ `wrapPerspective`(patch_a, $H^{-1}, *size$)
    H_{4Pt} $=$ corners$_{noised}$ $-$ corners
    save corners and H_{4Pt} to a configuration file
  end for

Then all patch_a will be stored in folder `data_generated/A`, all patch_b will be stored in folder `data_generated/B` and all configuration files will go into `data_generated/config`

B. Supervised Approach

The architecture of the supervised network is visualized in Figure 22 in appendix. The whole supervised learning model is constructed based on the given paper [4]. For the optimizer, we are using using stochastic gradient descent (SGD) with momentum of 0.9 and a base learning rate of 0.005. The batch size selected is 64 and we run the training for 120 epochs.

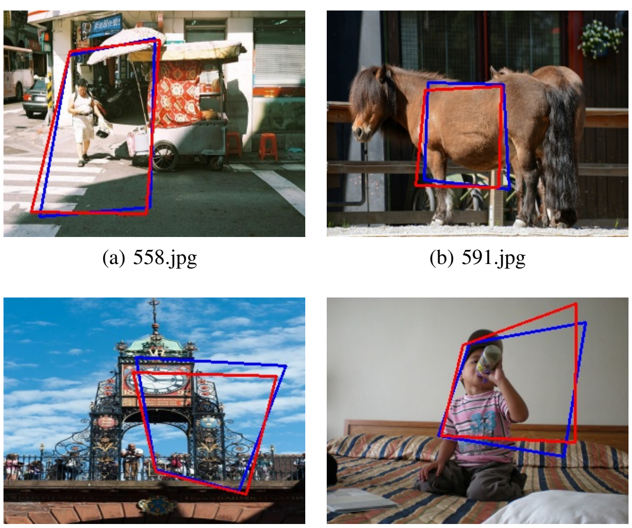
Here are some results from training dataset shown in Figure 15:

(a) 606.jpg (b) 1390.jpg

(c) 1713.jpg (d) 4289.jpg

Fig. 15: Image overlayed with homography estimated by supervised learning model shown in blue and ground truth shown in red for training dataset.

Here are some results from validation dataset shown in Figure 16:

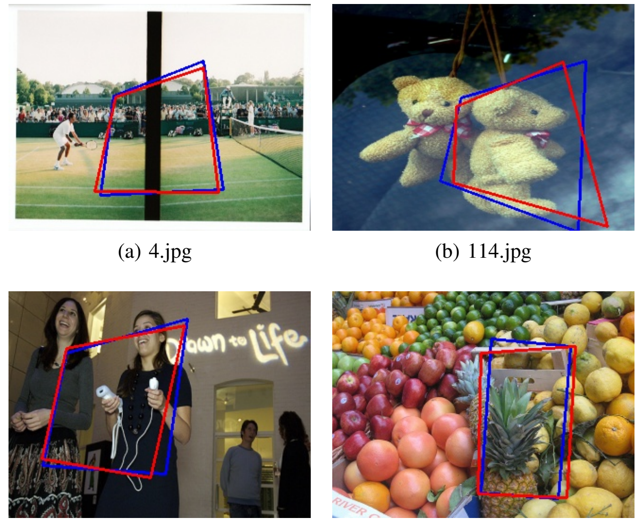


(a) 558.jpg (b) 591.jpg

(c) 616.jpg (d) 845.jpg

Fig. 16: Image overlayed with homography estimated by supervised learning model shown in blue and ground truth shown in red for validation dataset.

Here are some results from test dataset shown in Figure 17:



(a) 4.jpg (b) 114.jpg

(c) 292.jpg (d) 658.jpg

Fig. 17: Image overlayed with homography estimated by supervised learning model shown in blue and ground truth shown in red for test dataset.

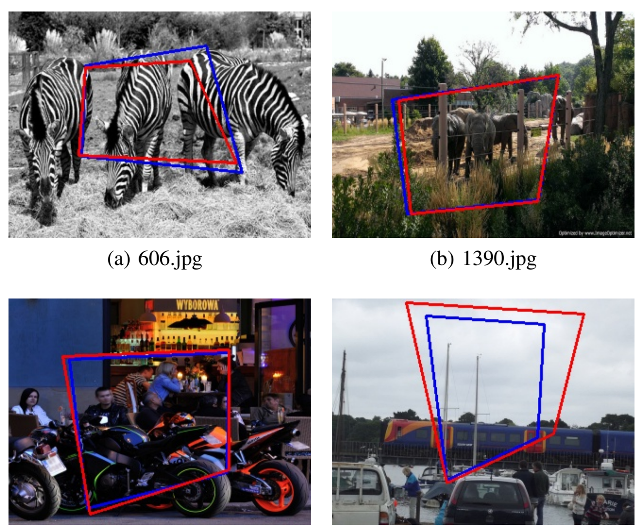
C. Unsupervised Approach

The architecture of the supervised network is visualized in Figure 23 in appendix.

The whole unsupervised learning model is constructed based on the given paper [5]. For the optimizer, we are using Adam optimizer [6] with learning rate as 0.0001.

The batch size selected is 128 and we run the training for 50 epochs.

Here are some results from training dataset shown in Figure 18:



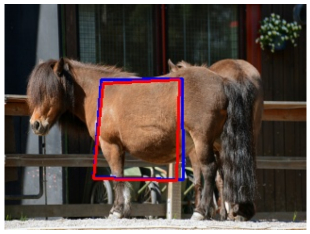
(a) 606.jpg (b) 1390.jpg

(c) 1713.jpg (d) 4289.jpg

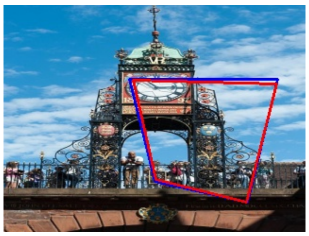
Fig. 18: Image overlayed with homography estimated by unsupervised learning model shown in blue and ground truth shown in red for training dataset.

Here are some results from validation dataset shown in Figure 19:

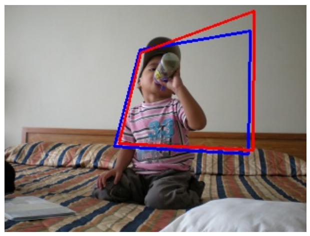
(a) 558.jpg



(b) 591.jpg



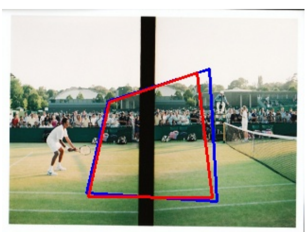
(c) 616.jpg



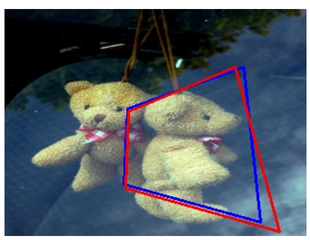
(d) 845.jpg

Fig. 19: Image overlayed with homography estimated by unsupervised learning model shown in blue and ground truth shown in red for validation dataset.

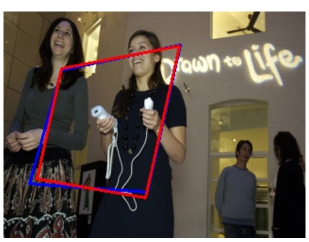
Here are some results from test dataset shown in Figure 20:



(a) 4.jpg



(b) 114.jpg



(c) 292.jpg

(d) 658.jpg

Fig. 20: Image overlayed with homography estimated by unsupervised learning model shown in blue and ground truth shown in red for test dataset.

D. Results and Comparison

III. Extra Credit

A. Collinearity Check in RANSAC Algorithm

For the selected four features pairs in the RANSAC algorithm, we would check if any three points lie on the same

TABLE I: Model Performance

Methods	Average EPE			Run-time (ms)
	Train	Val	Test	
Supervised	26.82	50.14	55.63	2.9
Unsupervised	47.44	63.65	62.68	1.7

line. Let the three points be $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3$. Let $\mathbf{n}_1 = x2 - x1$, $\mathbf{n}_2 = x2 - x3$. We require:

$$(\mathbf{n}_1 \cdot \mathbf{n}_2)^2 \leq (\eta |\mathbf{n}_1||\mathbf{n}_2|)^2 \tag{7}$$

This is equivalent to:

$$|\cos\theta| \leq \eta \tag{8}$$

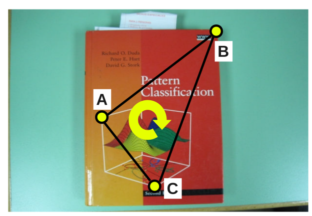
where θ is the angle between $\mathbf{n}_1$ and $\mathbf{n}_2$.

B. Geometric Constraint in RANSAC Algorithm

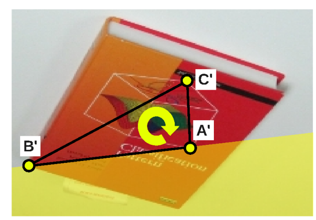
Choose three pairs from the selected four feature pairs. Let the three pairs be $(\mathbf{x}_1 \leftrightarrow \mathbf{x}'_1, \mathbf{x}_2 \leftrightarrow \mathbf{x}'_2, \mathbf{x}_3 \leftrightarrow \mathbf{x}'_3)$. The relative order of points $\mathbf{x}_1$, $\mathbf{x}_2$, $\mathbf{x}_3$ and that of points $\mathbf{x}'_1$, $\mathbf{x}'_2$, $\mathbf{x}'_3$ is the same. Figure 21 depicts it graphically. To put it formally, every subset of three correspondences in the selected four feature pairs must verify the following equation [7]:

$$((x_2 \times x_3)^T x_1) \cdot ((x'_2 \times x'_3)^T x'_1)) > 0. \tag{9}$$

Otherwise, the selected four feature pairs should be discarded, since it leads to an invalid homography.



(a) Points A, B, C in image I_1



(b) Corresponding points A', B', C' in image I_0. Point C' must not be located in the marked region, since A', B', C' must have the same relative order than A, B, C. If this constraint does not hold, this set of correspondences should be discarded. [7]

Fig. 21: Geometric constraint that must be satisfied in each random sample.

C. Update the Number of Iterations

The number of maximum iterations k would be continously udpated in our RANSAC algorithm. Let p be the desired probability that the RANSAC algorithm provides at least one useful result after running. RANSAC returns a successful result if in some iteration it selects only inliers from the input data set when it chooses the 4 points from which the model parameters are estimated. Let w be the probability of choosing an inlier each time a single point is selected, that is,

$$w = \text{number of inliers in data } / \text{number of points in data} \tag{10}$$

w is not well known beforehand, but we can estimate it based on the size of the current largest inlier set:

$$w = \frac{|\text{largest set of inliers}|}{\text{Number of matched features}} \tag{11}$$

Since 4 points are needed for estimating the homography, w^4 is the probability that all 4 points are inliers and $1-w^4$ is the probability that at least one of the 4 points is an outlier, a case which implies that a bad model will be estimated from this point set. That probability to the power of k is the probability that the algorithm never selects a set of 4 points which all are inliers and this must be the same as $1-p$. Consequently,

$$1 - p = (1 - w^4)^k \tag{12}$$

which, after taking the logarithm of both sides, leads to

$$k = \frac{\log(1-p)}{\log(1-w^4)} \tag{13}$$

D. Data Normalization

As stated in [2] (4.4.4, p. 107: Why is normalization essential?), data normalization is an essential step in the DLT algorithm. It must not be considered optional. Thus algorithm 6 is used instead of algorithm 3:

Algorithm 6: The basic DLT for H

Objective: Given $n \geq 4$ 2D to 2D point correspondences $\mathbf{x}_i \leftrightarrow \mathbf{x}'_i$, determine the 2D homography matrix H such that $\mathbf{x}'_i = H\mathbf{x}_i$.

Normalization of $\mathbf{x}$: Compute a similarity transformation T, consisting of a translation and scaling, that takes points $\mathbf{x}_i$ to a new set of points $\tilde{\mathbf{x}}_i$ such that the centroid of the points $\tilde{\mathbf{x}}_i$ is the coordinate origin $(0,0)^T$, and their average distance from the origin is $\sqrt{2}$;

Normalization of $\mathbf{x}'$: Compute a similarity transformation T' for the points in the second image, transforming points $\mathbf{x}'_i$ to $\tilde{\mathbf{x}}'_i$;

DLT: Apply algorithm 3 to the correspondences $\tilde{\mathbf{x}}_i \leftrightarrow \tilde{\mathbf{x}}'_i$ to obtain a homography $\tilde{H}$;

Denormalization: Set $H = T'^{-1}\tilde{H}T$.

Appendix A
Supervised Model Graph

The supervised model graph is shown in Figure 22.

Appendix B
Unsupervised Model Graph

The unsupervised model graph is shown in Figure 23.

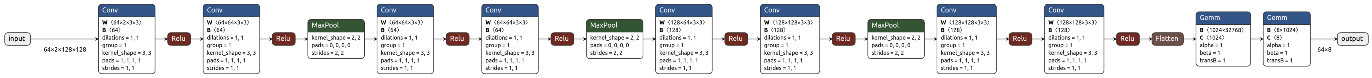


Fig. 22: Architecture of the supervised network.

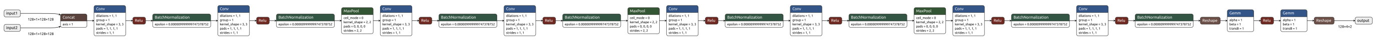


Fig. 23: Architecture of the unsupervised network.

References

[1] Chris Harris, Mike Stephens, et al. "A combined corner and edge detector". In: *Alvey vision conference*. Vol. 15. 50. Citeseer. 1988, pp. 10–5244.

[2] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[3] Nitin J. Sanket, Lening Li, and Gejji Vaishnavi Vivek. *HW0 Guidence*. URL: https://rbe549.github.io/fall2022/proj/p1/.

[4] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. "Deep image homography estimation". In: *arXiv preprint arXiv:1606.03798* (2016).

[5] Ty Nguyen et al. "Unsupervised deep homography: A fast and robust homography estimation model". In: *IEEE Robotics and Automation Letters* 3.3 (2018), pp. 2346–2353.

[6] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[7] Pablo Márquez-Neila et al. "Speeding-up homography estimation in mobile devices". In: *Journal of Real-Time Image Processing* 11.1 (2016), pp. 141–154.