16-720B Computer Vision: Homework 2
Feature Descriptors, Homographies and RANSAC

Heethesh Vhavle
Andrew ID: hvhavlen

09 October 2018

1 Keypoint Detector

Figure 1: Keypoint detection without edge suppression (left) and with edge suppression (right)

2 BRIEF Descriptor

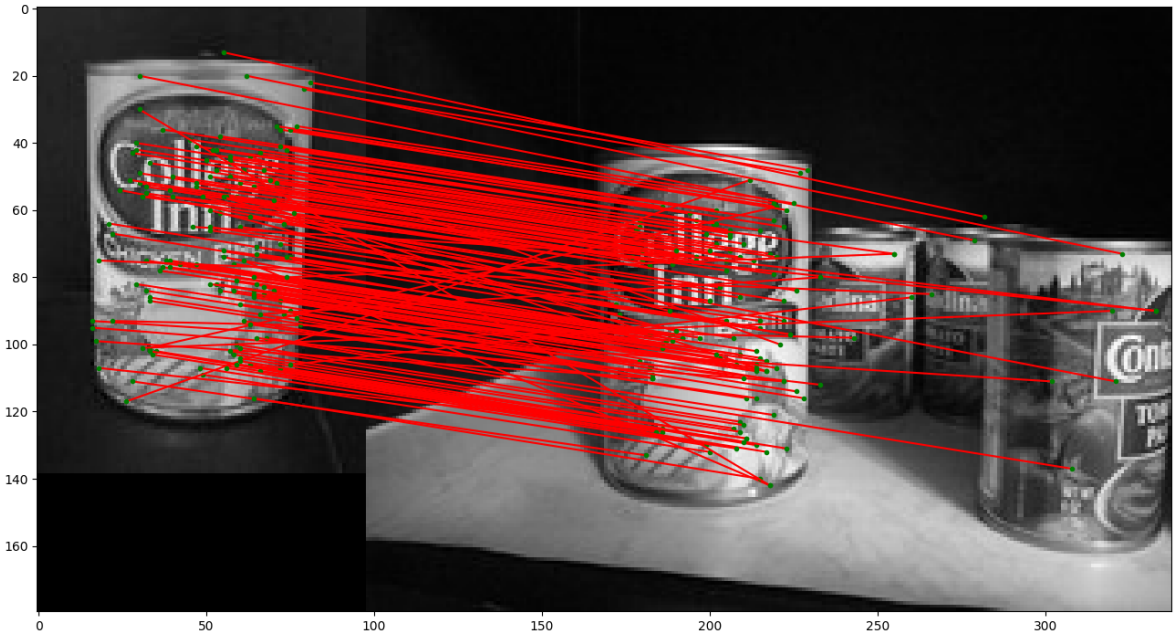

Figure 2: BRIEF matches for *chickenbroth_01.jpg* and *model_chickenbroth.jpg*

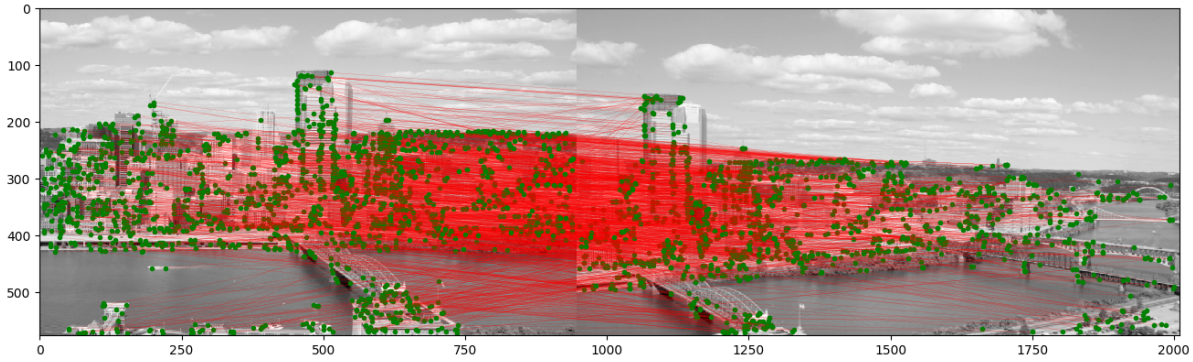

Figure 3: BRIEF matches for *incline_L.png* and *incline_R.png*

It can be seen from the images that there are a good number of matches using the BRIEF descriptor with some outliers. The result is quite good with some images such as Figures 2, 3 and 4. Figure 7 also has good matches, however it has a lot of outliers probably due to similar keypoints detected from the other books. With Figures 6 and 8, where the book is rotated, the BRIEF matches are very bad with very few correct matches or corresponding points. This is obviously due to the reason that BRIEF is not designed for rotation invariance.

The histogram for rotation and number of matches is shown in Figure 9. The histogram plot is as expected. This is because in BRIEF we only select a bunch of random points and compare their intensities within the patch to create a binary descriptor. Although we are selecting keypoints over different scales using Gaussian Pyramids, nowhere are we selecting or taking care of their rotation information. It can be seen from the histogram that after around 10 degrees of rotation, there is a sudden drop in number of matches. Since we are selecting points randomly in the patch, there is a slight increase in matches at 180 degrees. To get better number of matches, our descriptor also should encode some sort of rotation

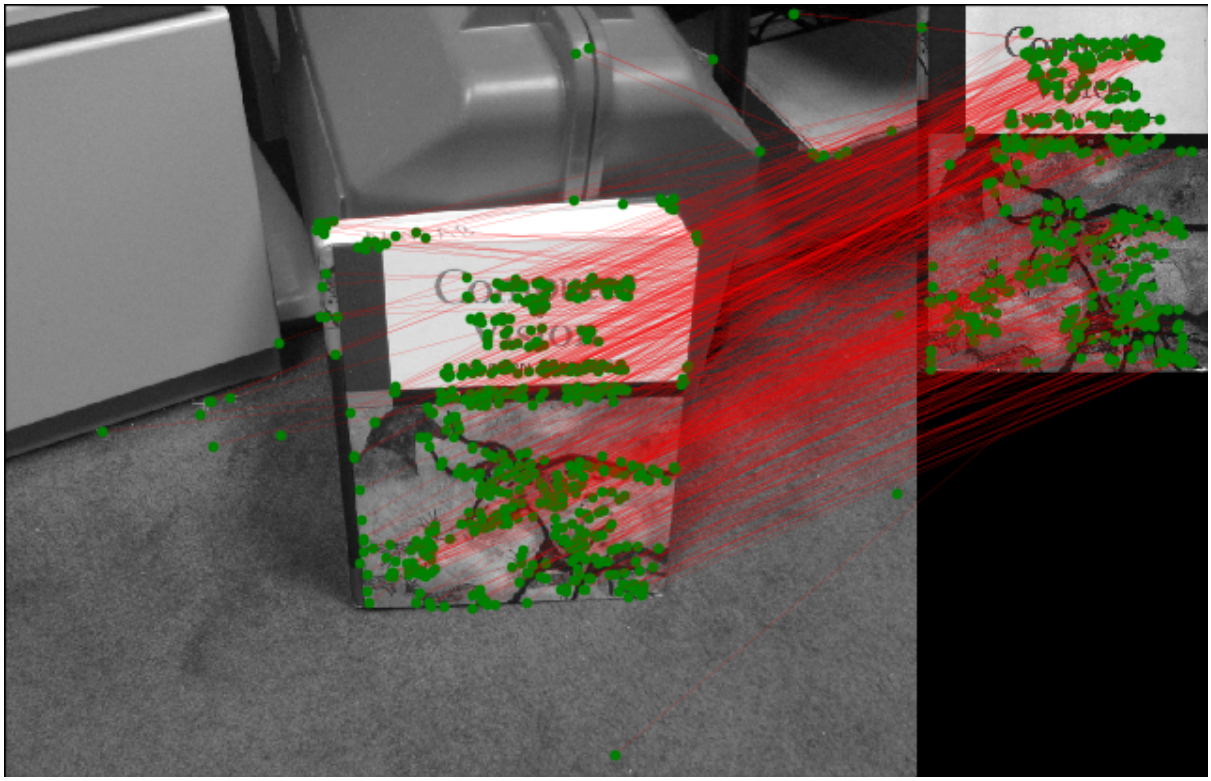

Figure 4: BRIEF matches for *pf_scan_scaled.jpg* and *pf_stand.jpg*

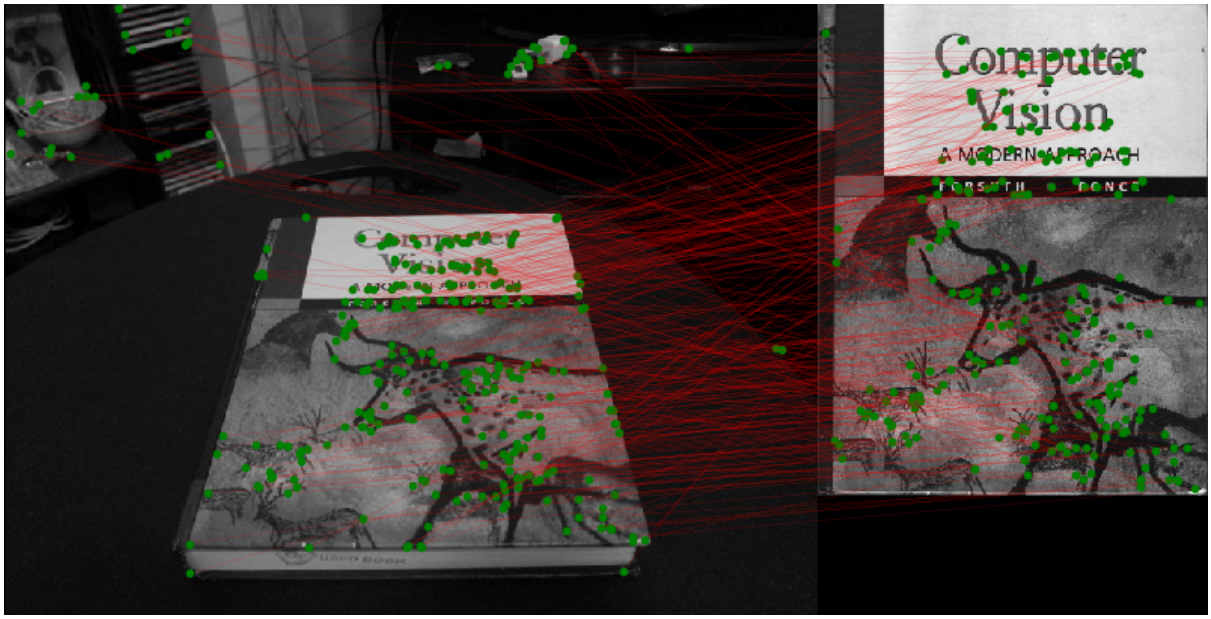

Figure 5: BRIEF matches for *pf_scan_scaled.jpg* and *pf_desk.jpg*

information.

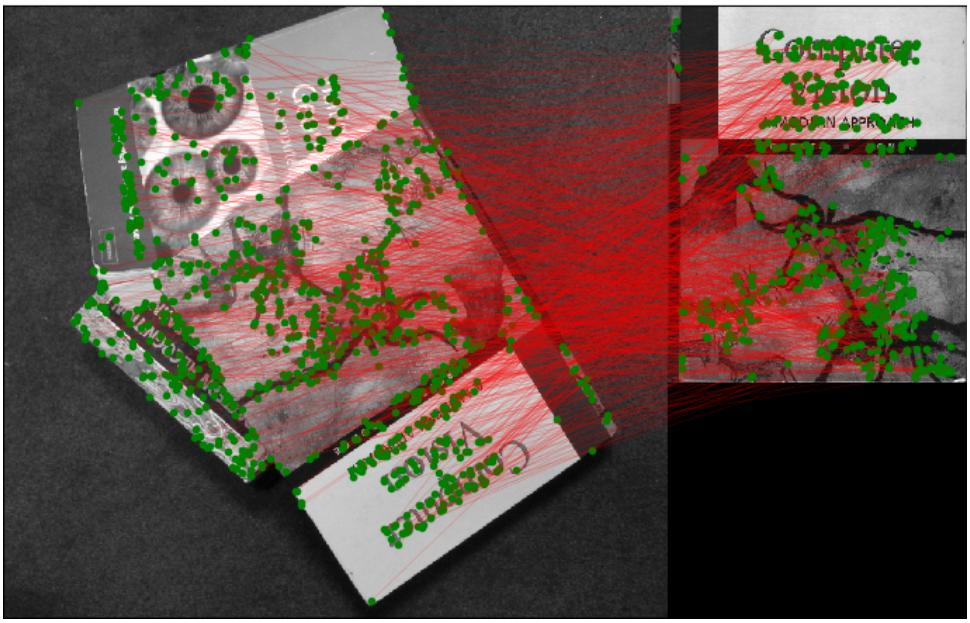

Figure 6: BRIEF matches for *pf_scan_scaled.jpg* and *pf_pile.jpg*

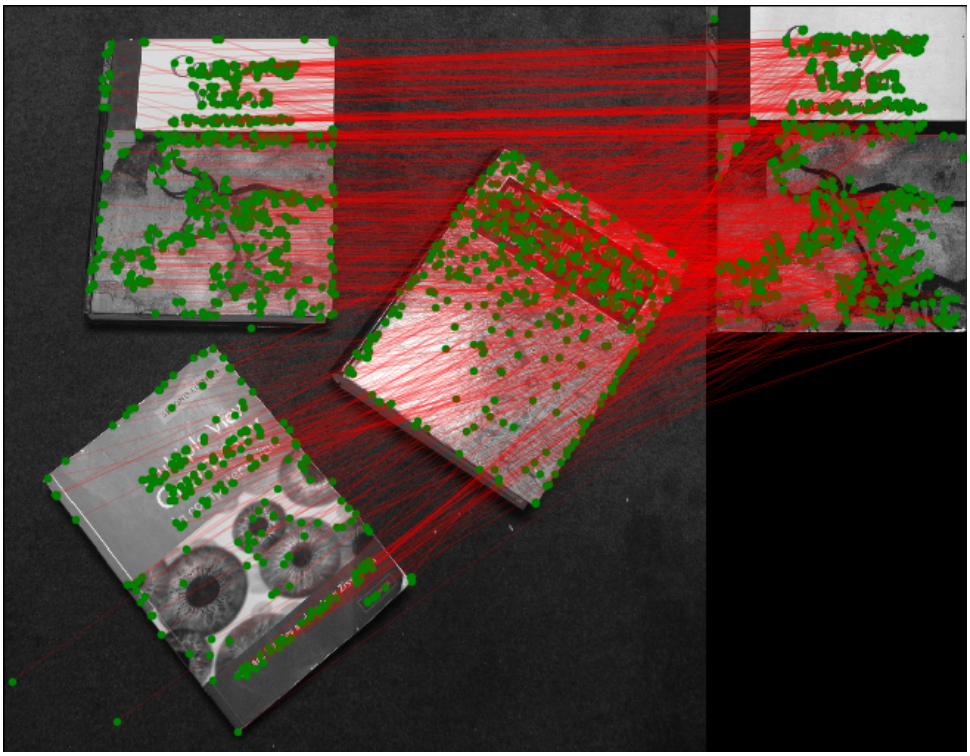

Figure 7: BRIEF matches for *pf_scan_scaled.jpg* and *pf_floor.jpg*

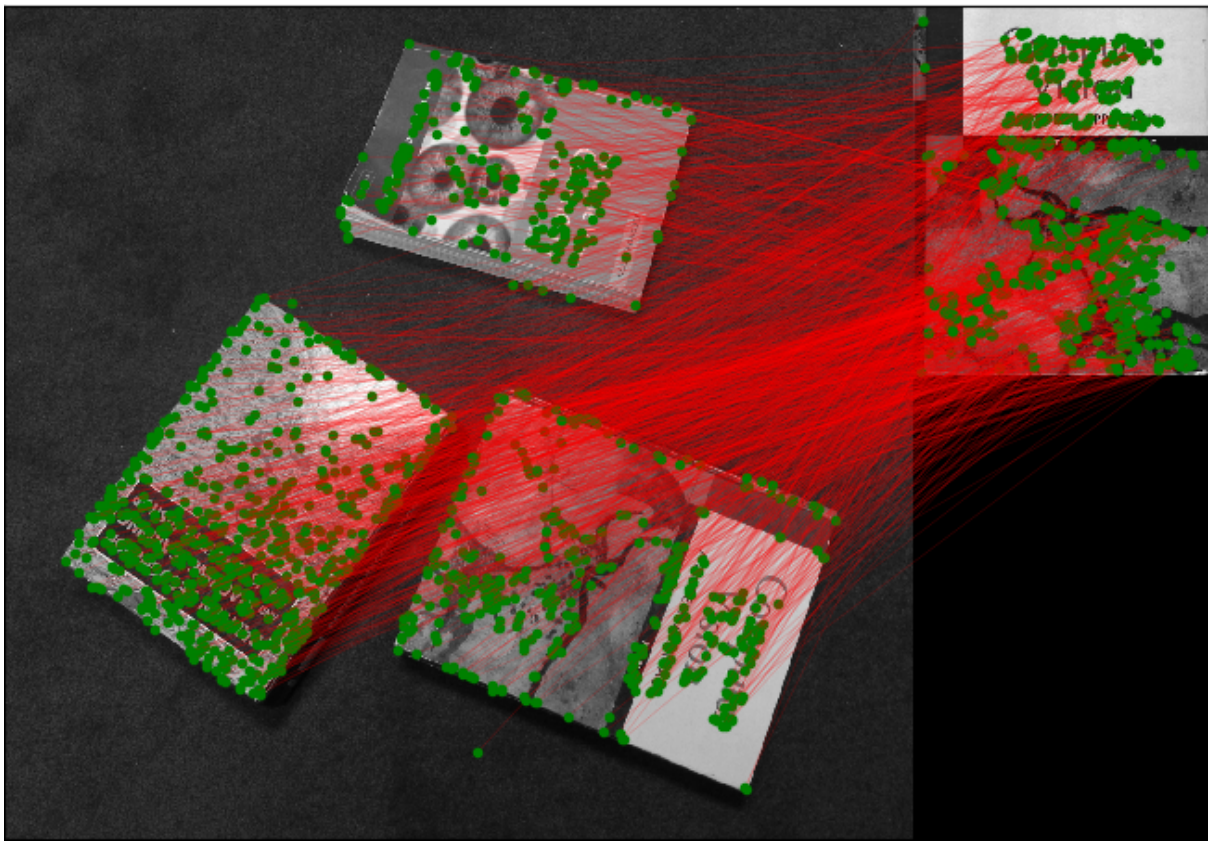

Figure 8: BRIEF matches for *pf_scan_scaled.jpg* and *pf_floor_rot.jpg*

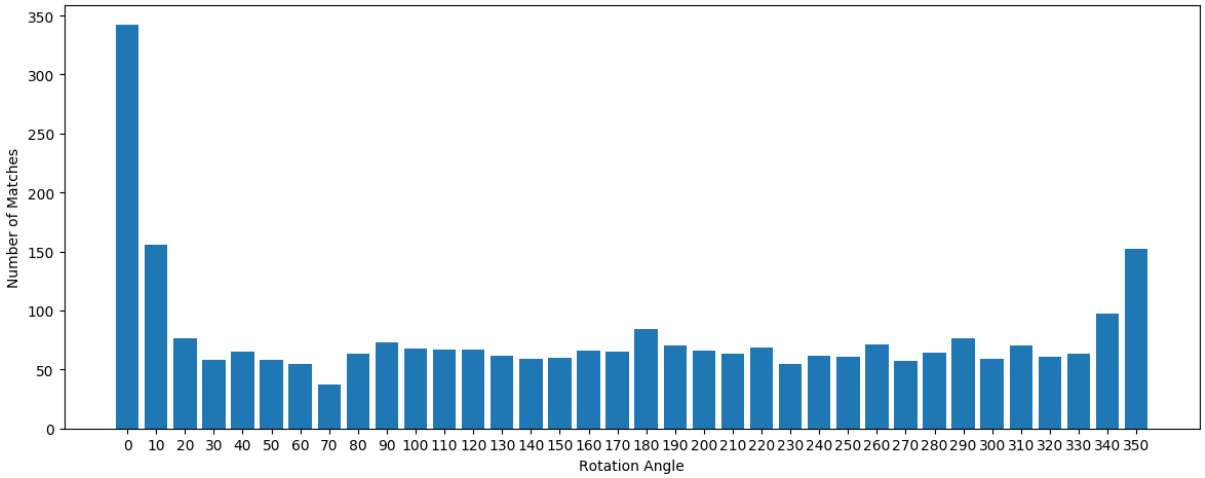

Figure 9: Histogram for number of BRIEF matches and rotation angle for *model_chickenbroth.jpg*

3 Planar Homographies: Theory

a) Let $\tilde{\mathbf{x}}_\mathbf{i} = \begin{bmatrix} x_i & y_i & 1 \end{bmatrix}^\top$ be the set of projected 2D points on the first camera and $\tilde{\mathbf{u}}_\mathbf{i} = \begin{bmatrix} u_i & v_i & 1 \end{bmatrix}^\top$ be the set of projected 2D points on the second camera.

For **N** correspondence points, the homography between these two views is given as follows.

$$\lambda_i \tilde{\mathbf{x}}_\mathbf{i} = \mathbf{H}\tilde{\mathbf{u}}_\mathbf{i} \quad for \quad i = 1 : N \tag{1}$$

For any two given points, we can write the above in a matrix form as follows.

$$\lambda \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix} \tag{2}$$

From the matrix above, we can derive the following relations between $\tilde{\mathbf{x}}$ and $\tilde{\mathbf{u}}$. For inhomogeneous coordinates, $x' = x/\lambda, \quad y' = y/\lambda$.

$$x' = \frac{h_{11}u + h_{12}v + h_{13}w}{h_{31}u + h_{32}v + h_{33}w} \tag{3}$$

$$y' = \frac{h_{21}u + h_{22}v + h_{23}w}{h_{31}u + h_{32}v + h_{33}w} \tag{4}$$

With loss of generality, setting $w = 1$ and simplifying the terms above,

$$x'(h_{31}u + h_{32}v + h_{33}) = h_{11}u + h_{12}v + h_{13} \tag{5}$$
$$y'(h_{31}u + h_{32}v + h_{33}) = h_{21}u + h_{22}v + h_{23} \tag{6}$$

Rearranging the above equations,

$$x'h_{31}u + x'h_{32}v + x'h_{33} - h_{11}u - h_{12}v - h_{13} = 0 \tag{7}$$
$$y'h_{31}u + y'h_{32}v + y'h_{33} - h_{21}u - h_{22}v - h_{23} = 0 \tag{8}$$

In the above equation, we have 9 unknowns and a single pair of points just gives us 2 equations. However, H_{33} is a scaling factor and we can normalize the matrix by dividing every other element in **H**. Now we still have 8 unknowns to find. Therefore, in order to solve this, we will require a minimum of 4 points which will give us 8 equations. In general, we can formulate 2N equations using N points. The following matrix can be formed by stacking each of the 2 equations from all the points,

$$\begin{bmatrix} -u_1 & -v_1 & -1 & 0 & 0 & 0 & u_1x'_1 & v_1x'_1 & x'_1 \\ 0 & 0 & 0 & -u_1 & -v_1 & -1 & u_1y'_1 & v_1y'_1 & y'_1 \\ -u_2 & -v_2 & -1 & 0 & 0 & 0 & u_2x'_2 & v_2x'_2 & x'_2 \\ 0 & 0 & 0 & -u_2 & -v_2 & -1 & u_2y'_2 & v_2y'_2 & y'_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ -u_N & -v_N & -1 & 0 & 0 & 0 & u_Nx'_N & v_Nx'_N & x'_N \\ 0 & 0 & 0 & -u_N & -v_N & -1 & u_Ny'_N & v_Ny'_N & y'_N \end{bmatrix} \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \\ h_{33} \end{bmatrix} = 0 \tag{9}$$

We can write this in the form below, where h is a vector of the elements of **H** and **A** is a matrix composed of elements derived from the point coordinates.

$$\mathbf{Ah} = 0 \tag{10}$$

b) As it can be seen in equation 10, **h** has 9 elements.

c) As explained in the previous section, we need a minimum of 4 points to solve this system.

Homography has 8 degrees of freedom, despite having 9 elements. This is because the last element or the scaling factor does not contribute to a new degree of freedom. Scaling all the nine elements will cancel out the common factor in equations 3 and 4.

Each point correspondence gives us 2 equations to help solve this system.

d) We need to solve the problem of minimizing the error while estimating **H**. We can add and extra constraint $|H| = 1$ to avoid the obvious solution of H being all zeros. We can solve this using Singular Value Decomposition (SVD).

$$A = USV^{\top} \tag{11}$$

In Python, we can use *np.linalg.svd*, where the singular values will be in descending order. In order to find the **h** that best fits all the points, we can take the last singular vector from **V** corresponding to the lowest singular value from **S**. Finally we can reshape it to get **h**. This gives us a DLT (direct linear transform) homography that minimizes algebraic error and the estimated homography may not be as good as the error does not have a geometric meaning.

4 Planar Homographies: Implementation

Implemented in *planarH.py*.

5 RANSAC

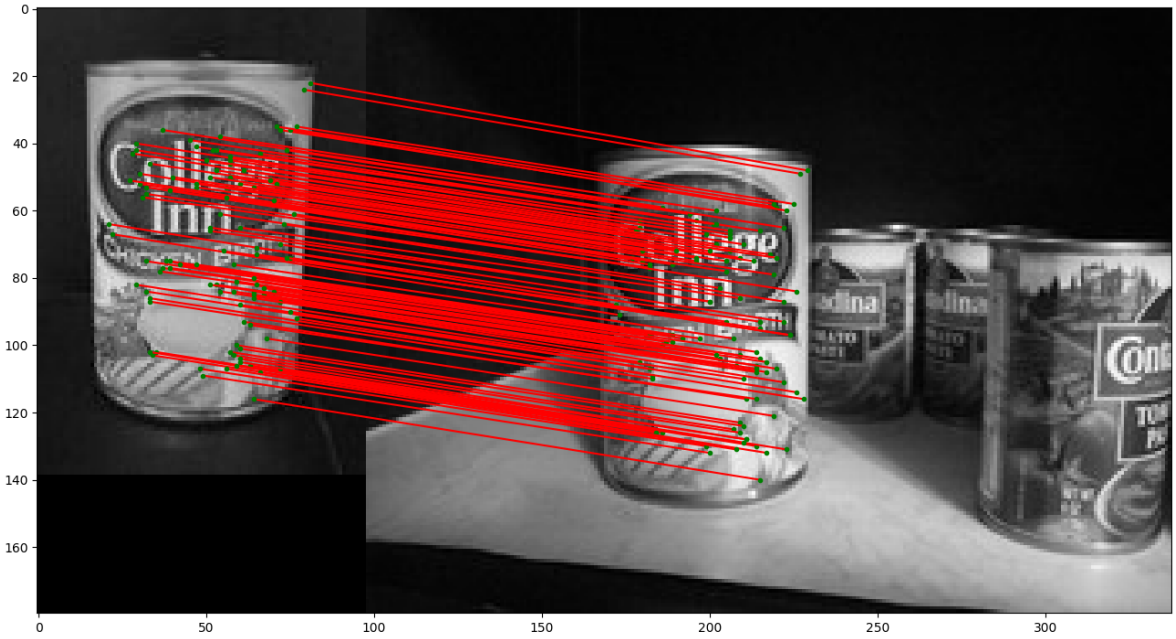

Figure 10: Matches after using RANSAC to remove the outliers for *model_chickenbroth.jpg* and *chicken-broth_01.jpg*

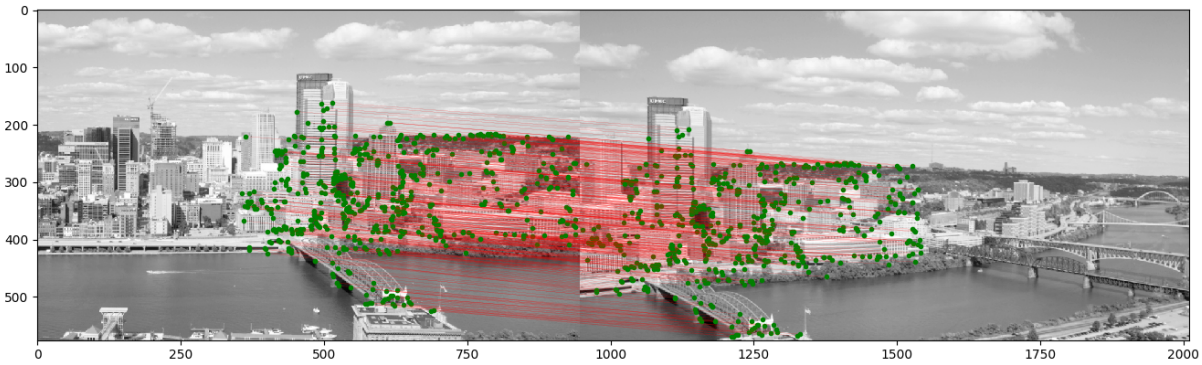

Figure 11: Matches after using RANSAC to remove the outliers for *incline_L.png* and *incline_R.png*

6 Stitching it Together: Panoramas

Figure 12: Stitched panorama of Dusquesne Incline (with clipping)

Figure 13: Stitched panorama of Dusquesne Incline (without clipping)

7 Augmented Reality

Implemented in *augmented_reality.py*. Just run the script to display the projected sphere on the image.

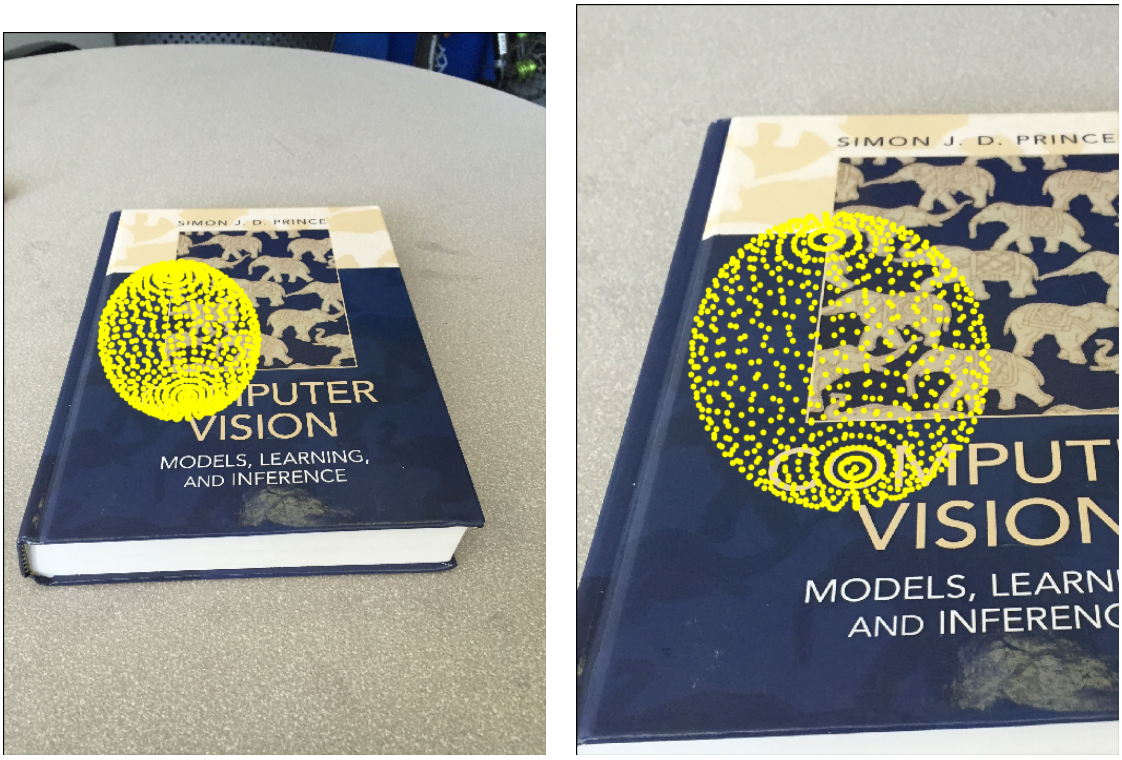

Figure 14: The 3D sphere projected on the 2D plane, centered at 'O' of 'Computer'