16-720 Computer Vision **Spring 2016**
Azarakhsh Keipour (akeipour@andrew) **Assignment 2**

Q 1.2 The DoG Pyramid

The function is implemented without using any loops to increase speed. The result shown in Fig. 1.

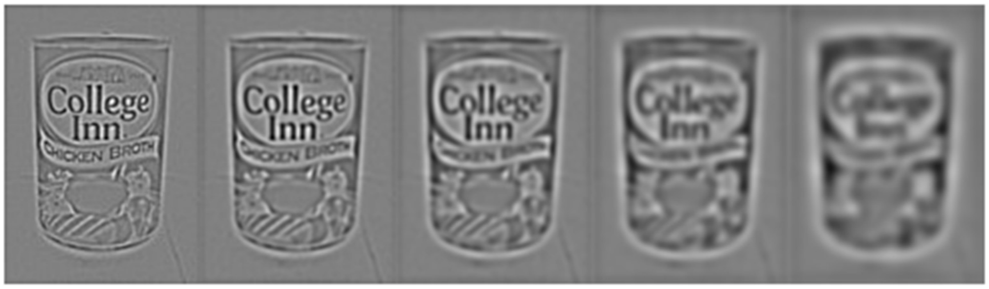

Figure 1. The Difference of Gaussian Pyramid of 'model_chickenbroth.jpg' image

Q 1.3 Principal Curvature calculation

The function is implemented using only one loop on levels (which is unavoidable due to the use of 'gradient' function) to increase speed.

Q 1.4 Detecting Extrema (DoG Keypoint Detection)

The function is implemented without using any loops to increase speed. It finds both types of local extremums (i.e. maximums and minimums).

Q 1.5 Finishing DoG Keypoint Detector

The result of the DoG Keypoint Detection algorithm is shown in Fig. 2 and Fig. 3 with and without edge suppression. In Fig. 3 you can easily see the effect of edge suppression on the edges. The code for plotting the keypoints is in 'DoGdetector.m' file.

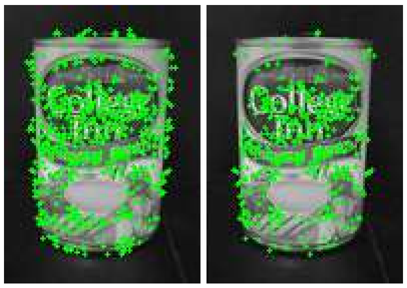

Figure 2. The Difference of Gaussian keypoints of 'model_chickenbroth.jpg' image
(a) Without edge suppression (b) With edge suppression

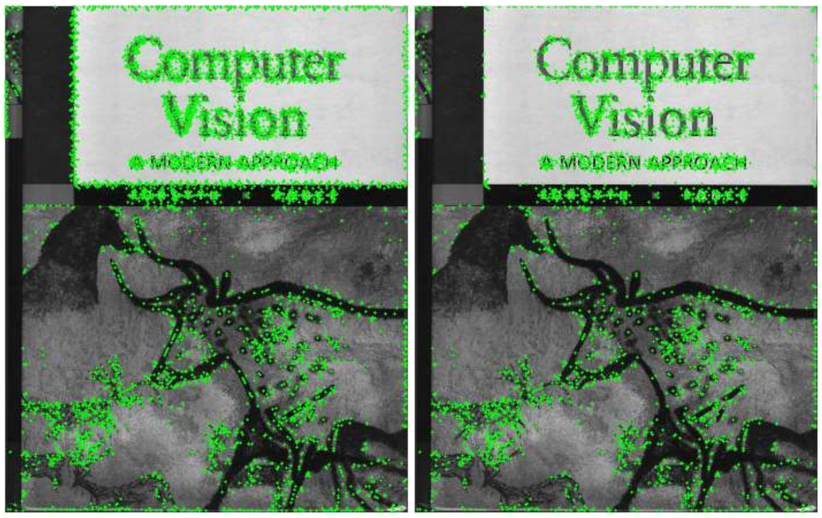

Figure 3. The Difference of Gaussian keypoints of 'pf_scan_scaled.jpg' image
(a) Without edge suppression (b) With edge suppression

Q 2.1 Creating a Set of BRIEF Tests

According to the section 3.2 of the 'BRIEF: Binary Robust Independent Elementary Features' paper by Calonder, Lepetit, Strecha and Fua (referenced in the description of the question), uniform and Gaussian distributions of the generated test set give the best results. I used the type I (uniform distribution for X and Y) to generate the test set. Fig. 4 shows a generated test set for patch size of 9 and 256 tests. The code for plotting the test set is in 'makeTestPattern.m' file.

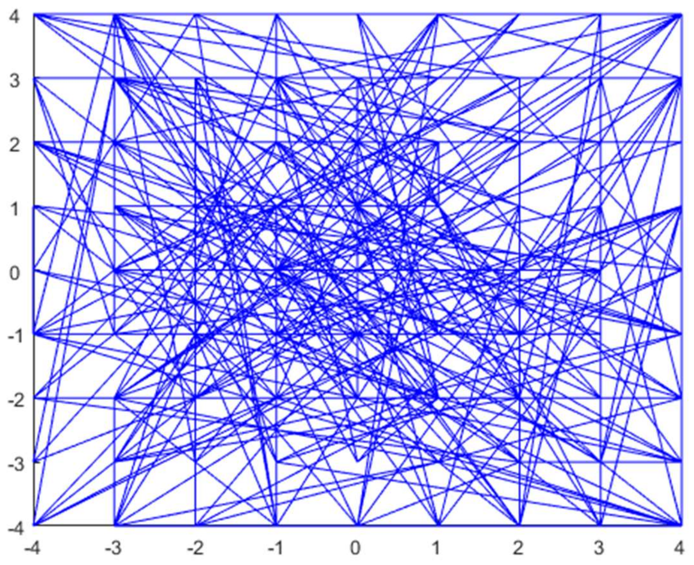

Figure 4. A random set of BRIEF tests generated using uniform distribution

Q 2.2, 2.3, 2.4

All the functions are implemented without using any loops to increase speed. The result of matching BRIEF features on ‘model_chickenbroth.jpg’ and ‘chickenbroth_01.jpg’ images is shown in Fig. 5. The result of matching BRIEF features on ‘incline_L.png’ and ‘incline_R.png’ images is shown in Fig. 6. Finally, the result of matching BRIEF features on ‘pf_scan_scaled.jpg’ against all ‘pf_*.jpg’ images is shown in Fig. 7.

As can be seen in Fig 7, feature matching for ‘pf_pile.jpg’ (top center) and ‘pf_floor_rot.jpg’ (bottom right) images gives poor results. The main reason is that in this two images the vision book is rotated with respect to the original image, while in other images the vision book is not rotated and have slightly different angle of views with respect to the original image. Since our implementation of BRIEF uses location of pixels in the set to define descriptors, it works poorly with rotated images.

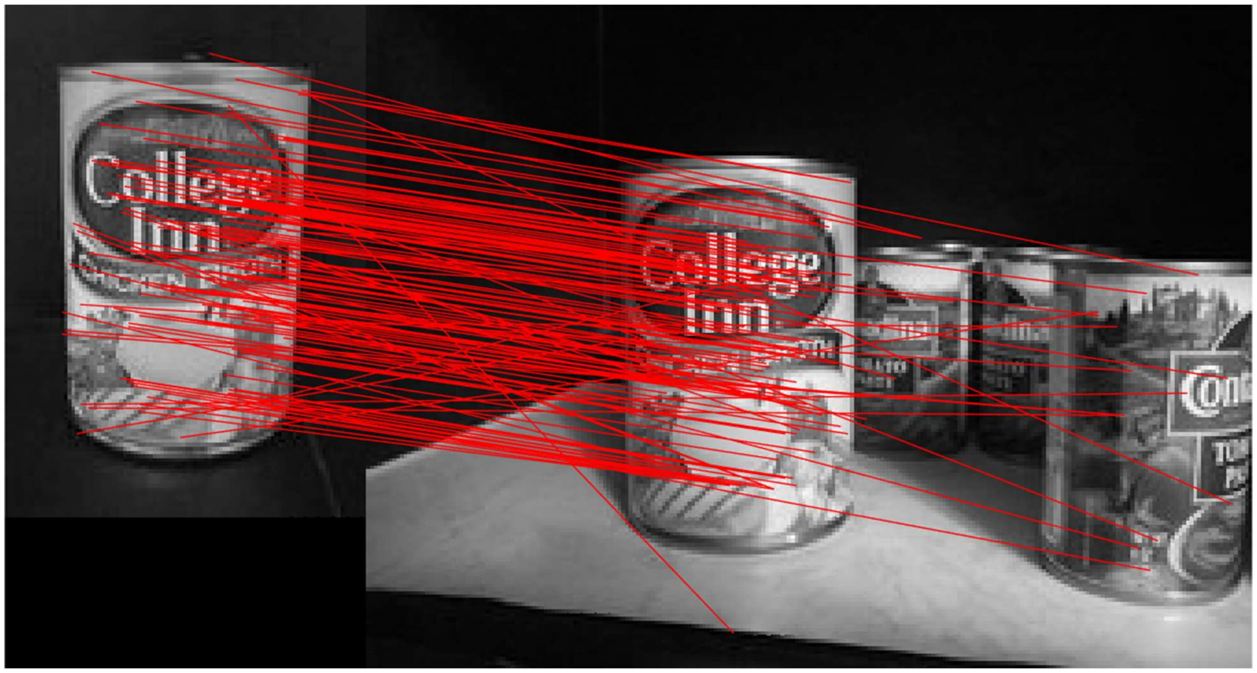

Figure 5. The result of matching BRIEF features on ‘model_chickenbroth.jpg’ and ‘chickenbroth_01.jpg’ images.

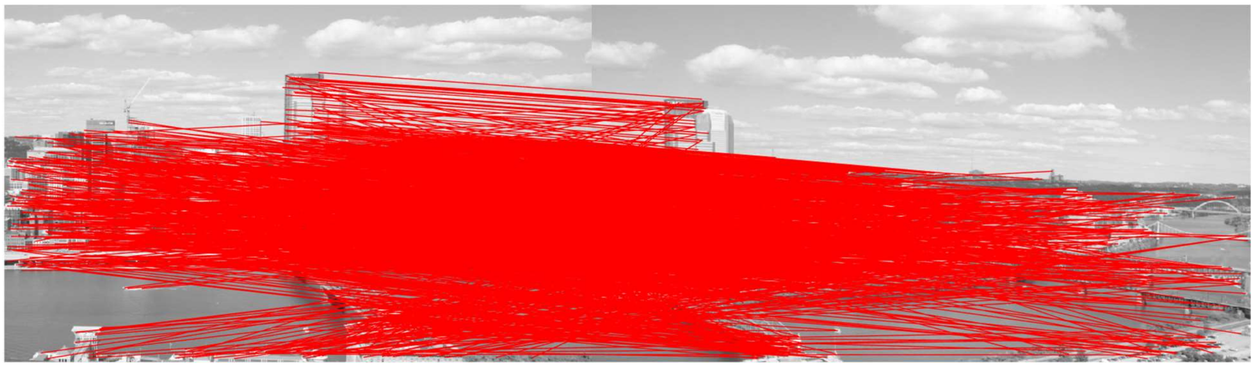

Figure 6. The result of matching BRIEF features on ‘incline_L.png’ and ‘incline_R.png’ images.

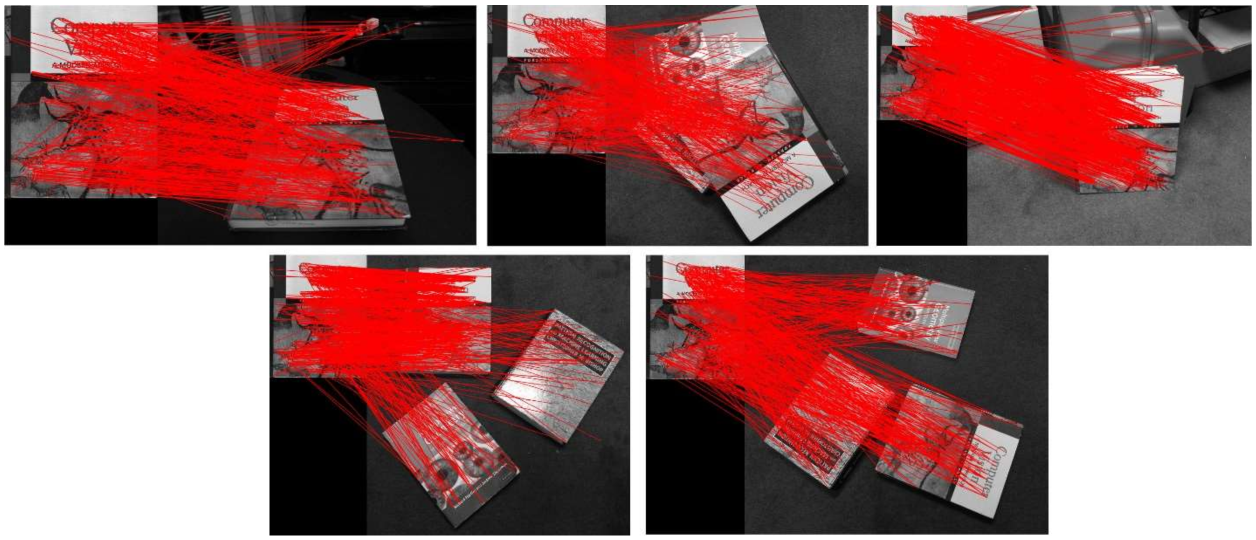
Figure 7. The result of matching BRIEF features on ‘pf_scan_scaled.jpg’ against ‘pf_desk.jpg’ (top-left), ‘pf_pile.jpg’ (top center), ‘pf_stand.jpg’ (top right), ‘pf_floor.jpg’ (bottom left) and ‘pf_floor_rot.jpg’ (bottom right) images.

Q2.5 BRIEF and Rotations

The bar graph of the rotation angle vs. the number of correct matches for ‘model_chickenbroth.jpg’ is shown in Fig. 8.

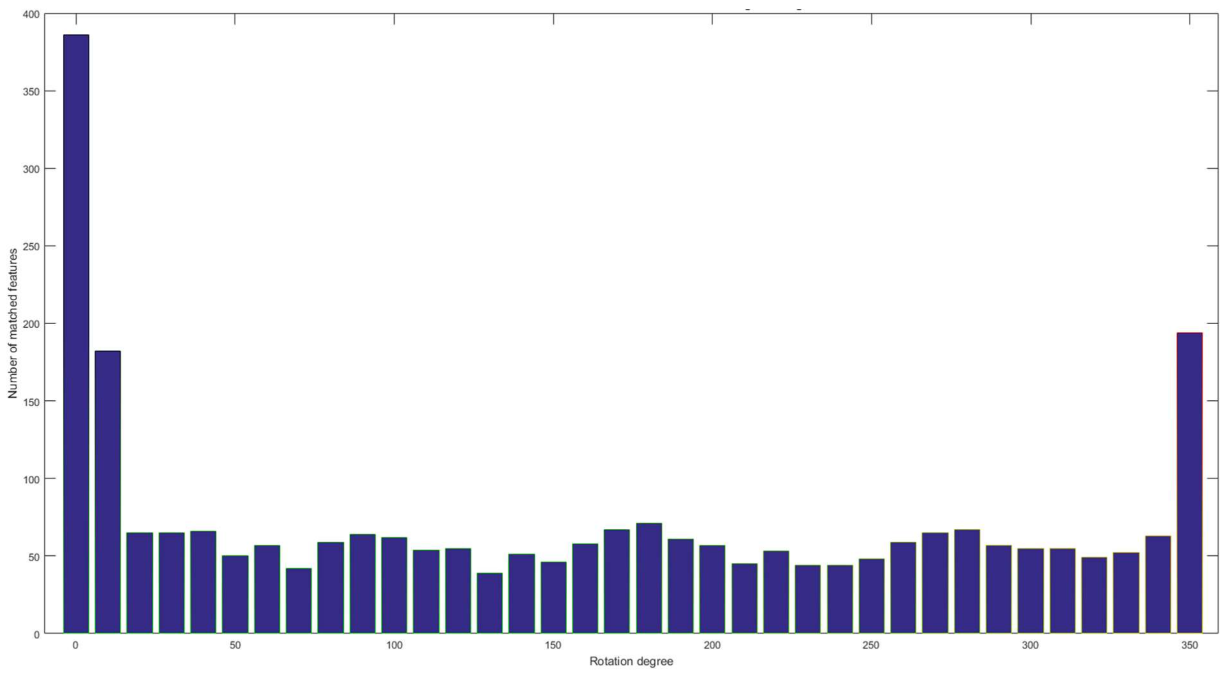

Figure 8. Rotation angle vs. the number of correct matches for ‘model_chickenbroth.jpg’

The number of correct matches heavily drops when the image rotates. The main reason is that our implementation of BRIEF uses intensity for pixels at certain locations (calculated using DoG keypoints and the set of BRIEF tests). When the image rotates, intensities of the fixed locations in the patches around the keypoints change; therefore, the patch can't be matched with the corresponding patch in the original (unrotated) version.

Q 3.1 (a)

First, we rewrite the Equation 8 with the homogeneous coordinates for p and q:

$$p^i \equiv \mathbf{H}q^i \Rightarrow \begin{bmatrix} u_1 \\ v_1 \\ 1 \end{bmatrix} \equiv \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} u_2 \\ v_2 \\ 1 \end{bmatrix} \Rightarrow \begin{bmatrix} cu_1 \\ cv_1 \\ c \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} u_2 \\ v_2 \\ 1 \end{bmatrix}$$

In order to eliminate c we divide all the rows of the both sides of the equation by the third row and eliminate the third row. Thus:

$$\begin{bmatrix} cu_1 \\ cv_1 \\ c \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} u_2 \\ v_2 \\ 1 \end{bmatrix} \Rightarrow \begin{bmatrix} u_1 \\ v_1 \end{bmatrix} = \begin{bmatrix} \frac{h_{11}}{h_{31}u_2} & \frac{h_{12}}{h_{32}v_2} & \frac{h_{13}}{h_{33}} \\ \frac{h_{21}}{h_{31}u_2} & \frac{h_{22}}{h_{32}v_2} & \frac{h_{23}}{h_{33}} \end{bmatrix} \begin{bmatrix} u_2 \\ v_2 \end{bmatrix}$$

By expanding the equations, we will have:

$$u_1 = \left(\frac{h_{11}}{h_{31}u_2} + \frac{h_{12}}{h_{32}v_2} + \frac{h_{13}}{h_{33}} \right) u_2 \Rightarrow \left(h_{31}u_2 + h_{32}v_2 + h_{33} \right) u_1 - \left(h_{11}u_2 + h_{12}v_2 + h_{13} \right) = 0$$

$$v_1 = \left(\frac{h_{21}}{h_{31}u_2} + \frac{h_{22}}{h_{32}v_2} + \frac{h_{23}}{h_{33}} \right) v_2 \Rightarrow \left(h_{31}u_2 + h_{32}v_2 + h_{33} \right) v_1 - \left(h_{21}u_2 + h_{22}v_2 + h_{23} \right) = 0$$

The above equations are 2 independent equations for corresponding points p^i and q^i. By rewriting them again in matrix form we have:

$$\left.\begin{matrix} h_{31}u_2u_1 + h_{32}v_2u_1 + h_{33}u_1 - h_{11}u_2 - h_{12}v_2 - h_{13} = 0 \\ h_{31}u_2v_1 + h_{32}v_2v_1 + h_{33}v_1 - h_{21}u_2 - h_{22}v_2 - h_{23} = 0 \end{matrix}\right\} \Rightarrow \begin{bmatrix} -u_2 & -v_2 & -1 & 0 & 0 & 0 & u_2u_1 & v_2u_1 & u_1 \\ 0 & 0 & 0 & -u_2 & -v_2 & -1 & u_2v_1 & v_2v_1 & v_1 \end{bmatrix} \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \\ h_{33} \end{bmatrix} = \mathbf{0}$$

Thus, for corresponding points p^i and q^i we have:

$$\mathbf{a}^i\mathbf{h} = \mathbf{0}$$

$$\mathbf{a}^i = \begin{bmatrix} -u_2 & -v_2 & -1 & 0 & 0 & 0 & u_2u_1 & v_2u_1 & u_1 \\ 0 & 0 & 0 & -u_2 & -v_2 & -1 & u_2v_1 & v_2v_1 & v_1 \end{bmatrix}, \qquad \mathbf{h} = \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \\ h_{33} \end{bmatrix} = \mathbf{H}^s$$

For *N* corresponding points, we will have *2N* independent equations and can compose the matrix A as follows:

$$\mathbf{A} = \begin{bmatrix} \mathbf{a}^i \\ \mathbf{a}^2 \\ \vdots \\ \mathbf{a}^N \end{bmatrix} \Rightarrow \mathbf{Ah} = 0$$

Q 3.1 (b)

Currently there are 9 elements in **h** (i.e. all elements of **H** are available in **h**). However, since the degree of freedom of **h** is only 8 (the elements of matrix **H** are independent up to a scale factor, resulting in 9-1=8 degrees of freedom), we can force one of the elements to 1, thus reducing the number of elements in **h** to 8.

Q 3.1 (c)

As reasoned in part (b), the elements of matrix **H** are independent up to a scale factor, therefore there are only 9-1=8 degrees of freedom in elements of **h**. On the other side, each new correspondence (point pair) gives us 2 independent equations (information). Thus, we will need at least 4 pairs of points (correspondences) to find all the elements of **h**. It is important that no 3 of the 4 pairs should be on the same line, otherwise we will not have 8 independent equations (information) to solve the **h**.

Q 3.1 (d)

The sum squared error for $\mathbf{Ah} = \mathbf{0}$ can be written as:

$$S = \left\|\mathbf{Ah} - \mathbf{0}\right\|^2 = \left\|\mathbf{Ah}\right\|^2 = \left(\mathbf{Ah}\right)^T\left(\mathbf{Ah}\right) = \mathbf{h}^T\mathbf{A}^T\mathbf{Ah}$$

In order to minimize *S* we can use derivation w.r.t. *h*:

$$S' = 0 \Rightarrow \left(\mathbf{h}^T \mathbf{A}^T \mathbf{A}\mathbf{h}\right)' = \mathbf{0} \Rightarrow 2\mathbf{A}^T \mathbf{A}\mathbf{h} = \mathbf{0} \Rightarrow \mathbf{A}^T \mathbf{A}\mathbf{h} = 0$$

This means that **h** is the eigenvector corresponding to the 0 eigenvalue of $\mathbf{A}^T\mathbf{A}$ matrix. For an overdetermined system of equations in the presence of noise, we should find the eigenvector of the smallest (closest to 0) eigenvalue of $\mathbf{A}^T\mathbf{A}$ matrix.

We know that the SVD decomposition of matrix **A** gives the eigenvectors of $\mathbf{A}^T\mathbf{A}$ matrix. Therefore, we can directly use this to obtain the eigenvector corresponding to the smallest (closest to 0) eigenvector; this eigenvector is **h**.

The step-by-step procedure will be as follows:

1. Construct matrix **A** from stacking $\mathbf{a}^i$ equation matrices as explained in part (a).
2. Perform SVD decomposition and decompose matrix **A** to $\mathbf{U\Sigma V}^T$.
3. Vector **h** is the last column of **V** (supposing that the last column corresponds to the smallest eigenvector).

Or

1. Construct matrix **A** from stacking $\mathbf{a}^i$ equation matrices as explained in part (a).
2. Find eigenvectors of $\mathbf{A}^T\mathbf{A}$.
3. Vector **h** is the first column of **V** (supposing that the first column corresponds to the smallest eigenvector).

I used the latter in my implementation.

Q 4.1 Planar Homography Implementation

Computing the homography matrix using steps in the question 3 to the raw data gives very bad results. In order to obtain acceptable results, I normalized data using the normalization algorithm explained in part 4.4.4 (page 109) of (Hartley, Zisserman). Here are the steps for computing homography matrix **H** in $p_1 = \mathbf{H}p_2$:

1. Normalization of points p_1:
    a. Transform the points such that their centroid will be the origin (simply subtract the mean of all point coordinates from the points).
    b. Scale all the points such that the average distance of points to the origin is $\sqrt{2}$ to obtain new points $\tilde{p}_1$ (simply divide the point coordinates by the mean of their distances and multiply by $\sqrt{2}$).
    c. Compute a similarity transformation $\mathbf{T_1}$, consisting of a translation and scaling, that takes points to the new normalized coordinates.
2. Normalization of points p_2: Repeat the steps a, b, c above for the points p_2 in order to obtain normalized point $\tilde{p}_2$ and similarity transformation $\mathbf{T_2}$.

3. Compute homography matrix $\tilde{\mathbf{H}}$ as described in question 3 on the normalized points so that $\tilde{p}_1 = \tilde{\mathbf{H}}\tilde{p}_2$.
4. Denormalize $\tilde{\mathbf{H}}$ (convert from normalized coordinates to image coordinates) to obtain homography matrix $\mathbf{H}$ using $\mathbf{H} \equiv T_1^{-1}\tilde{\mathbf{H}}T_2$ equation.

I also added an additional step:

5. Set the determinant of the homography matrix **H** to 1 by dividing the matrix elements by $\left(\det\left(\mathbf{H}\right)\right)^{\frac{1}{3}}$.

The results surprisingly got better after normalization. Fig. 9 shows the result of the warping of ‘incline_R.jpg’ image to ‘incline_L.jpg’ image using homography matrix. I used $ratio = 0.25$ in matching features to reduce the number of incorrect matches in all .

Q 5.1 Image Stitching Without RANSAC (With Clipping)

In Fig. 9 the result of warping ‘incline_R.jpg’ image to ‘incline_L.jpg’ image coordinates with clipping and without performing RANSAC for calculation of homography matrix H is shown. Also, the result of blending (stitching) this warped image with ‘incline_L.jpg’ is shown in Fig. 10. We can see that the homography matrix is very sensitive to incorrect matches and without performing RANSAC the resulted homography matrix is not very useful.

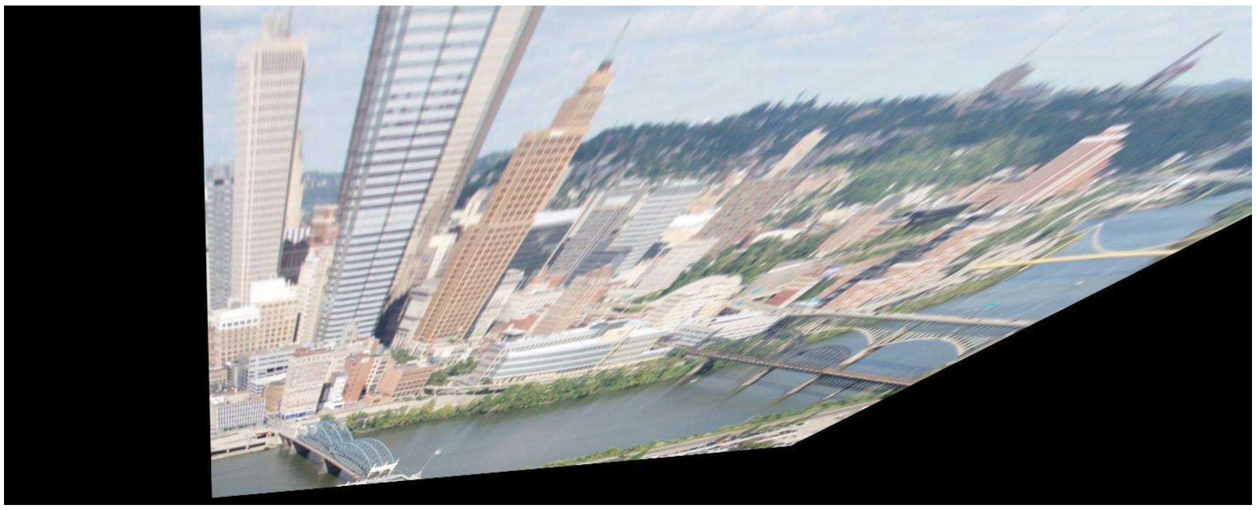

Figure 9. Result of warping ‘incline_R.jpg’ image to ‘incline_L.jpg’ image coordinates with clipping and without performing RANSAC for calculation of homography matrix H.

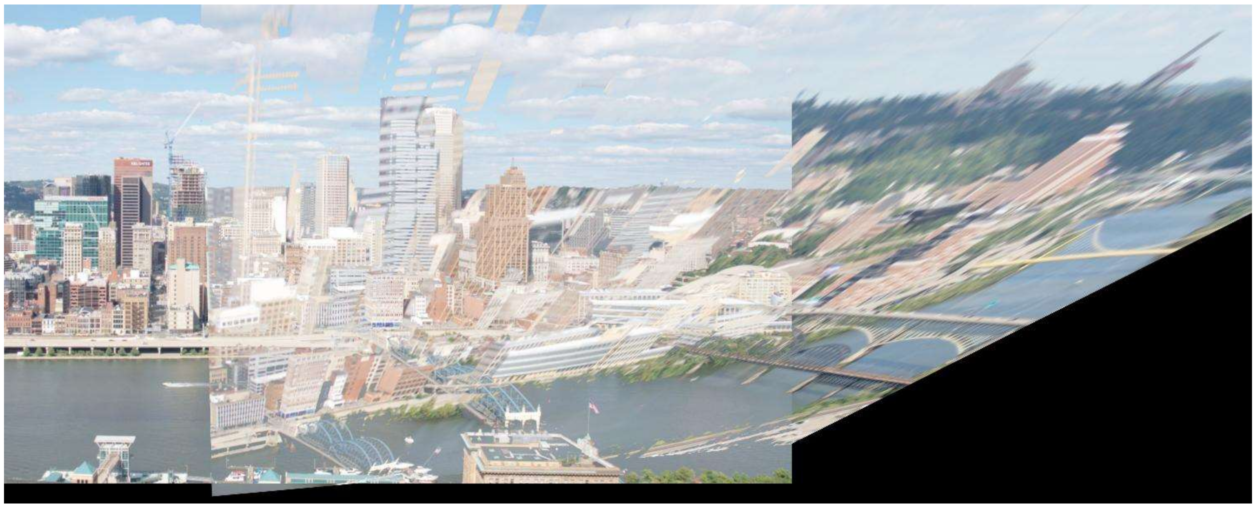

Figure 10. Result of stitching 'incline_L.jpg' and 'incline_R.jpg' images with clipping and without performing RANSAC for calculation of homography matrix H.

The calculated homography matrix **H** used above is:

$$\mathbf{H} = \begin{bmatrix} 0.8525 & 0.7020 & -37.8411 \\ -0.1453 & 2.3887 & -508.6307 \\ -0.0001 & 0.0027 & -0.1070 \end{bmatrix}$$

I used $ratio = 0.25$ in matching features to reduce the number of incorrect matches in this and all subsequent parts.

I also wrote a function `blendImages` that blends the two input images together by taking the maximum intensity of the image pixels for each pixel. This function is used in this and all subsequent parts.

I chose $\text{outsize} = [600\ 1500]$ for the output size. This can be changed on the first line of 'imageStitching' function.

Q 5.2 Image Stitching Without RANSAC (Without Clipping)

In Fig. 11 the result of generating panorama (stitching) with 'incline_R.jpg' and 'incline_L.jpg' images without clipping and without performing RANSAC for calculation of homography matrix **H** is shown. Without performing RANSAC the resulted homography matrix is not very useful.

By examining the result in Fig. 11, we can find the scaled version of Fig. 10 at the top right corner of the figure.

I chose $width = 1000$ for the output size. This can be changed on the first line of 'imageStitching_noClip' function.

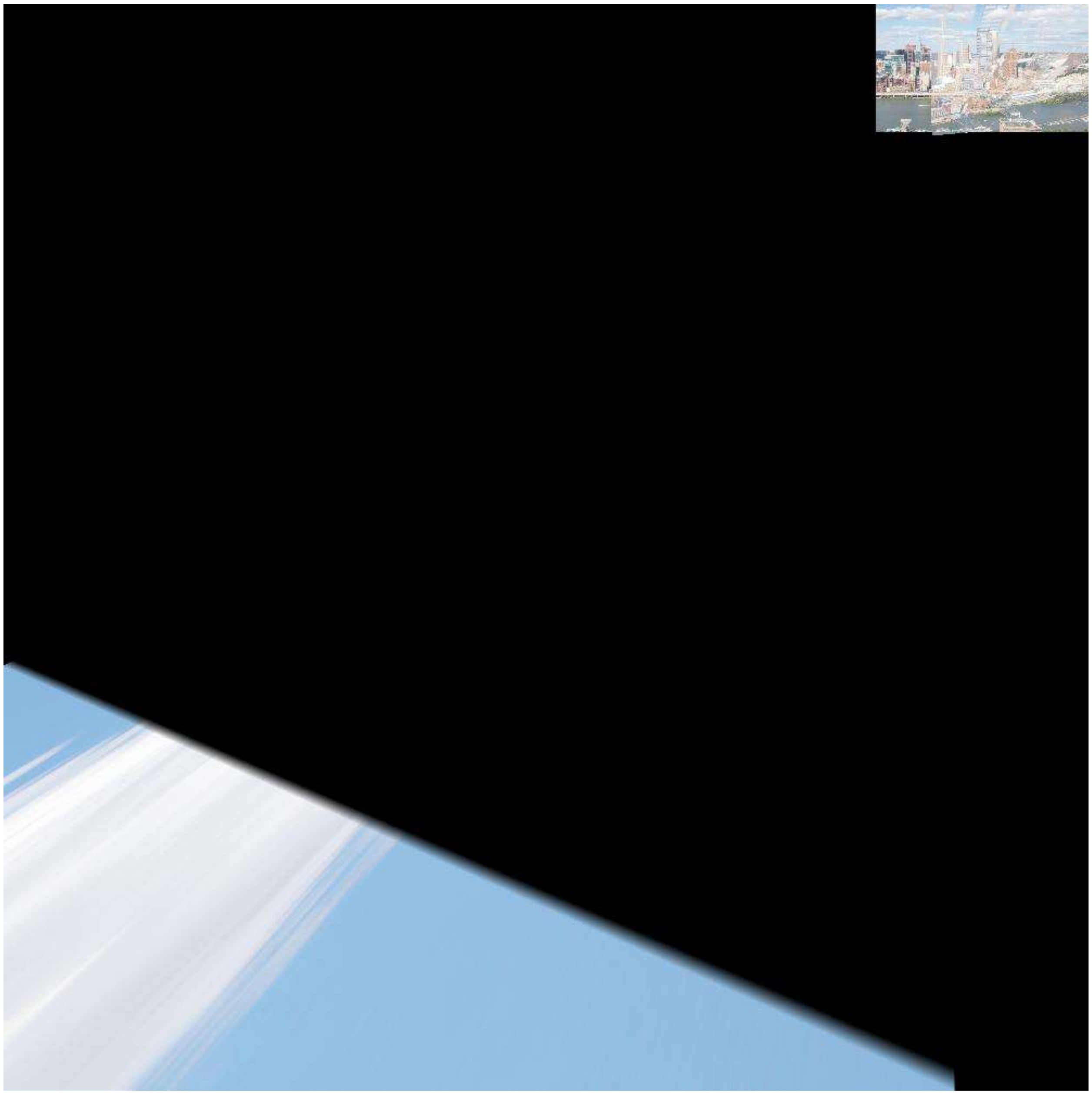

Figure 11. Result of stitching 'incline_L.jpg' and 'incline_R.jpg' images without clipping and without performing RANSAC for calculation of homography matrix H.

Q 6.1 RANSAC Implementation

I used the Euclidian distance of the projected (warped) points from second image (using **H**) from the points in first image for finding inliers. The default values for the number of iterations and tolerance that work both fast and efficient were chosen 500 and 5, respectively.

Q 6.2 RANSAC Results

The final panorama view with ‘incline_L.jpg’ and ‘incline_R.jpg’ images without clipping and using homography estimated with RANSAC is shown in Fig. 12.

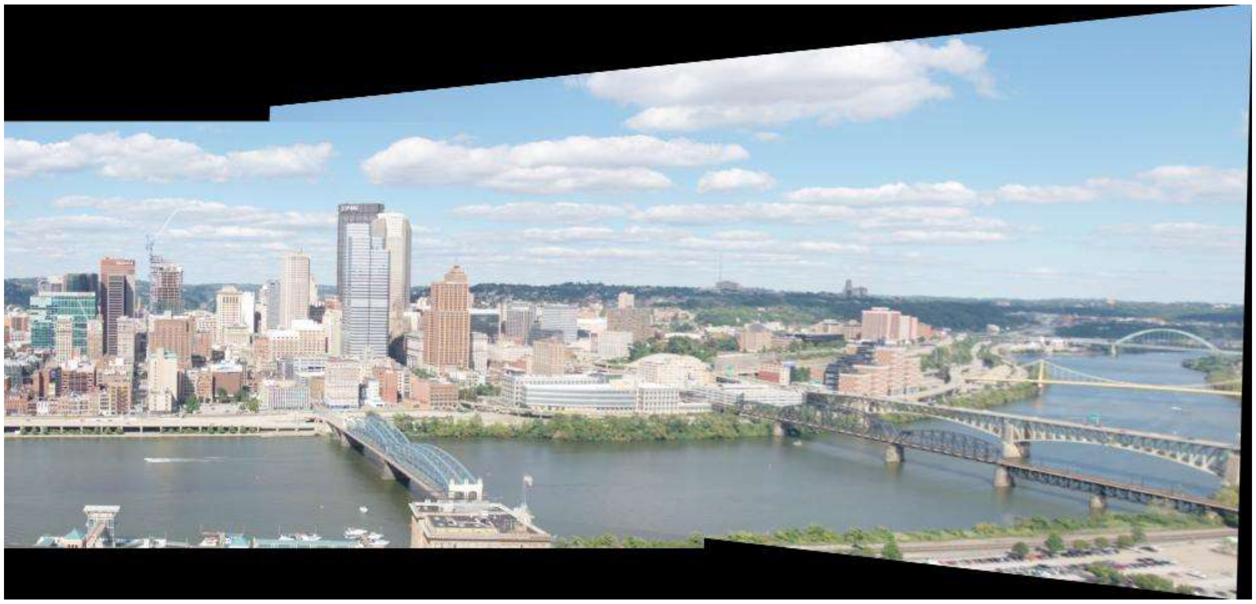

Figure 12. Result of stitching ‘incline_L.jpg’ and ‘incline_R.jpg’ images without clipping and with estimating homography matrix H using RANSAC.

References

[1] R. Hartley and A. Zisserman. Multiple View Geometry in Computer Vision (2nd Edition). Cambridge University Press, ISBN: 0521540518, 2004.