16-720 Homework 2

Chendi Lin

October 11, 2018

Problem 1.1. Gaussian Pyramid

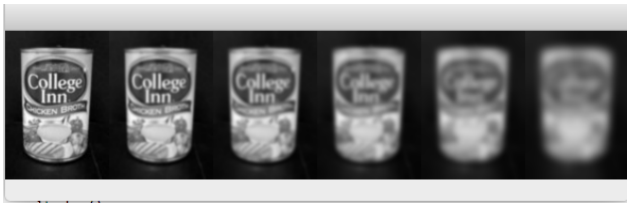

Figure 1: Gaussian Pyramid

Problem 1.2. Difference of Gaussian Pyramid

Figure 2: DoG

Problem 1.5 Keypoint Detector.

Solution . The local extremas in space were found using "*scipy.ndimage.filters.maximum_filter*" and "*scipy.ndimage.filters.minimum_filter*". The local extremas in scale were found using "*scipy.signal.argrelextrema*.

The keypoints detection is shown in the figure below. On the left is the one without edge suppression, while on the right is the one with edge suppression.

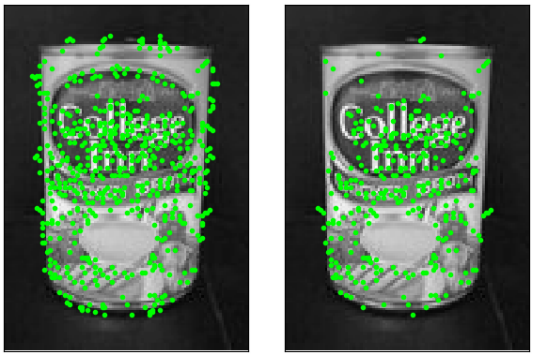

Figure 3: Keypoints Detection without and with edge suppression for "*model_chickenbroth.jpg*"

Problem 2.4. Descriptor Matching

Solution . The matching pictures are shown below.

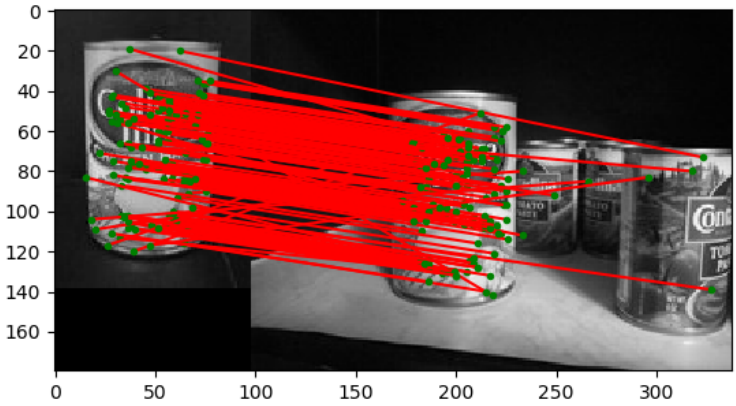

Figure 4: Descriptor Matching of Chicken Broth

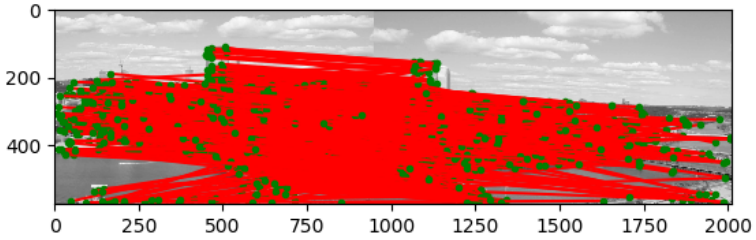

Figure 5: Descriptor Matching of Incline Pictures

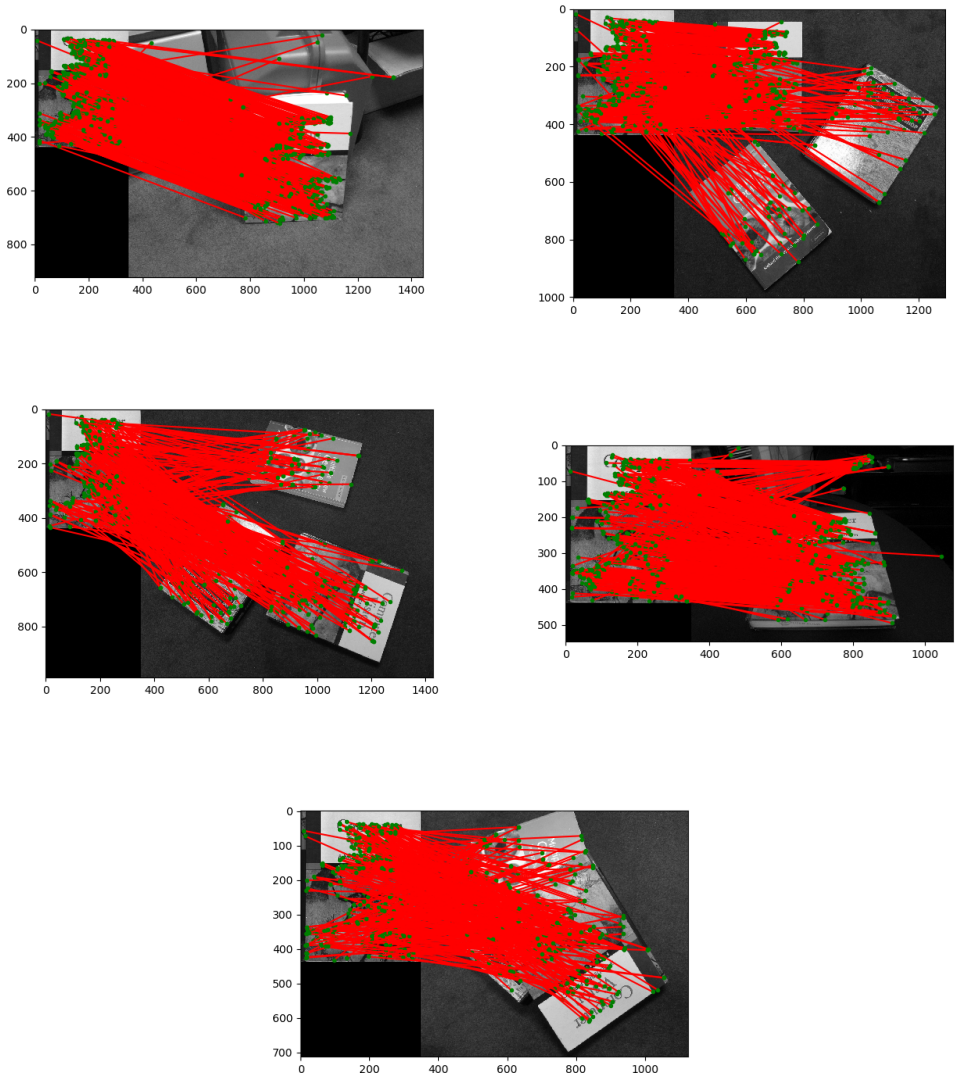

Figure 6: Descriptor Matching of Computer Vision Textbook Cover Page. (a) Stand book. (b) Books on the floor. (c) Books on the floor with rotation. (d) Book on the desk. (e) A pile of textbooks.

We can see from the figure above that, generally, BRIEF matching performed well in the pictures of chicken broth and *incline* images. The matching lines were mostly parallel, indicating that those matching points were correct. That was because the features of the scenes were distinctive and no rotation was involved. Matching for textbook images were more complicated. When the scanned picture was matched with the "standing" book and the picture of book on the floor, the matching worked nicely. However, when matching with multiple books, the performance was much worse, because the interest points of books were similar, and the BRIEF matching found it hard to differentiate the interest points of different books. When the rotation was applied, the matching result was even worse, because the relationship of the pixels was altered. More details will be discussed in the next problem.

Problem 2.5. BRIEF and Rotations

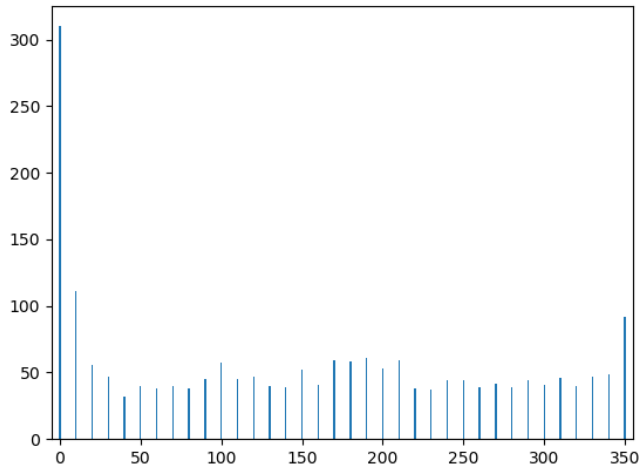

Figure 7: Number of Matches with Rotations

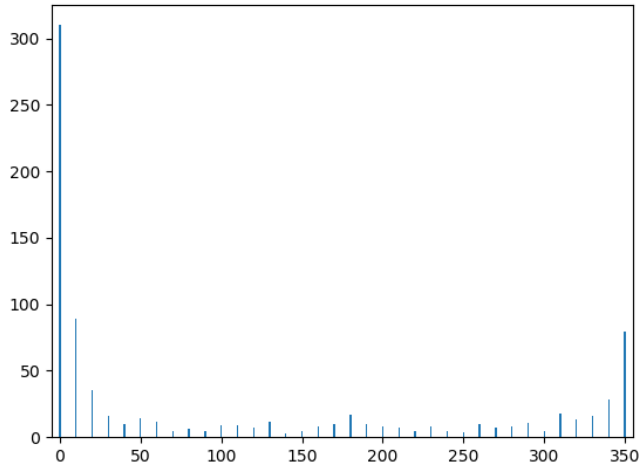

Figure 8: Number of Correct Matches with Rotations

The first figure displays the number of matches vs. the rotation angles, and the second figure shows the number of correct matches vs. the rotation angles. We can see from the figures that, when rotations were applied to the figure, the number correct matches dropped significantly. When the rotation angle was larger than 30 degrees, the number of correct matches was almost 0. That was resulted from the fact that, when we used BRIEF to match the descriptor, we chose a patch around the interest point, and the relationship between the pixels in that pitch determined whether these two interest points matched. When the picture

was rotated, the pixels in the patch were re-arranged and thus, the matching performance got noticeably worse.

Problem 3. Planar Homographies: Theory

Solution (a). the relationship between two views can be expressed as

$$\lambda_n \tilde{\mathbf{x_n}} = \mathbf{H}\tilde{\mathbf{u_n}} \qquad \text{for } \mathbf{n} = \mathbf{1} : \mathbf{N} \tag{1}$$

For an arbitrary $n = i$, the equation can be expressed explicitly as

$$\lambda_i \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} \tag{2}$$

From the third row we get

$$\lambda_i = h_{31}u_i + h_{32}v_i + h_{33} \tag{3}$$

Thus, we can write x_i and y_i in Cartesian as

$$x_i = \frac{h_{11}u_i + h_{12}v_i + h_{13}}{h_{31}u_i + h_{32}v_i + h_{33}} \tag{4}$$

$$y_i = \frac{h_{21}u_i + h_{22}v_i + h_{23}}{h_{31}u_i + h_{32}v_i + h_{33}} \tag{5}$$

Rearrange the equations above we have

$$x_i(h_{31}u_i + h_{32}v_i + h_{33}) - (h_{11}u_i + h_{12}v_i + h_{13}) = 0 \tag{6}$$

$$y_i(h_{31}u_i + h_{32}v_i + h_{33}) - (h_{21}u_i + h_{22}v_i + h_{23}) = 0 \tag{7}$$

which can be written in matrix form as

$$\begin{bmatrix} -u_i & -v_i & -1 & 0 & 0 & 0 & x_iu_i & x_iv_i & x_i \\ 0 & 0 & 0 & -u_i & -v_i & -1 & y_iu_i & y_iv_i & y_i \end{bmatrix} \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \\ h_{33} \end{bmatrix} \tag{8}$$

Thus, $\mathbf{A}$ is the combination of N equations above.

$$\mathbf{A} = \begin{bmatrix} -u_1 & -v_1 & -1 & 0 & 0 & 0 & x_1u_1 & x_1v_1 & x_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ -u_N & -v_N & -1 & 0 & 0 & 0 & x_Nu_N & x_Nv_N & x_N \\ 0 & 0 & 0 & -u_1 & -v_1 & -1 & y_1u_1 & y_1v_1 & y_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & -u_N & -v_N & -1 & y_Nu_N & y_Nv_N & y_N \end{bmatrix} \tag{9}$$

Solution b. There are 9 entries in $\mathbf{h}$.

Solution c. Since for homography transformation, there are 8 degrees of freedom in $\mathbf{H}$, and two equations are given for every point, 4 point pairs are required to solve this system.

Solution d. To solve the equation system, we can try to solve the following least square system

$$arg\min_{\mathbf{h}} ||\mathbf{Ah}||_{\mathbf{2}}^{\mathbf{2}}, \qquad \text{such that } ||\mathbf{h}||_{\mathbf{2}}^{\mathbf{2}} = \mathbf{1} \tag{10}$$

It can also be expressed as

$$||\mathbf{Ah}||_{\mathbf{2}}^{\mathbf{2}} = (\mathbf{Ah})^{\mathbf{T}}(\mathbf{Ah}) \tag{11}$$

$$= \mathbf{h}^{\mathbf{T}}\mathbf{A}^{\mathbf{T}}\mathbf{Ah} \tag{12}$$

To find its minimum, we can take the derivative of the system with respect to $\mathbf{h}$ and make it zero

$$\frac{d}{d\mathbf{h}}(\mathbf{h}^{\mathbf{T}}\mathbf{A}^{\mathbf{T}}\mathbf{Ah}) = \mathbf{2A}^{\mathbf{T}}\mathbf{Ah} = \mathbf{0} \tag{13}$$

Therefore, $\mathbf{h}$ is the eigenvector of $(\mathbf{A}^{\mathbf{T}}\mathbf{A})$ corresponding to the zero eigenvalue. To obtain that, we can do *svd* to $\mathbf{A}$ that

$$\mathbf{A} = \mathbf{U\Sigma V}^{\mathbf{T}}$$

Because in Python, *svd* sorts the singularities from large to small, and $\mathbf{U}$ and $\mathbf{V}$ are orthogonal matrices, $\mathbf{h}$ is the last column of $\mathbf{V}$.

Overall, the steps are:
1) Construct $\mathbf{A}$ as described in (c);
2) Do *svd* to $\mathbf{A} = \mathbf{U\Sigma V}^{\mathbf{T}}$
3) extract $\mathbf{h}$ as the last column of $\mathbf{V}$

Problem 6.1. Image Stitching.

Figure 9: Warped Image 2

Figure 10: Panorama Image with Clipping

Since the warped image 2 was translated to the right, we need a wider target image. We extend the whole image by 600 in width.

Problem 6.2. Image Stitching with No Clipping

Figure 11: Panorama Image with No Clipping

Problem 6.3. Generate Final Panorama View

Figure 12: Final Panorama View

Problem 7. Augmented Reality

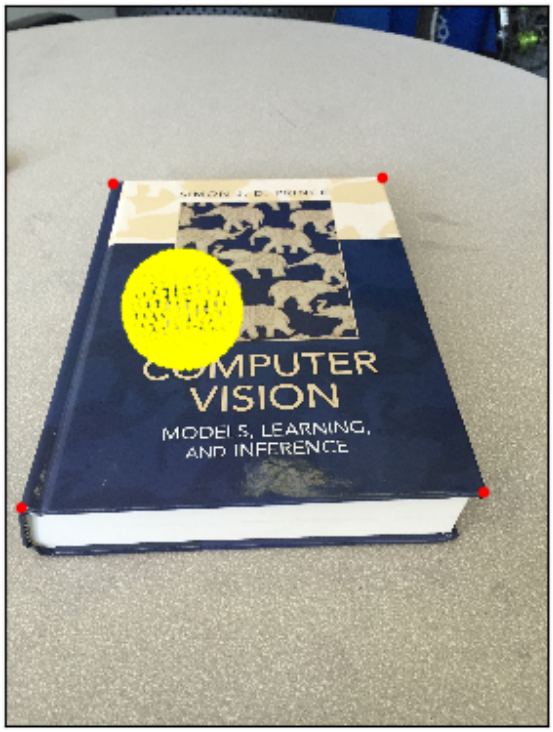

Figure 13: Augmented Reality of Tennis Ball on Textbook

To place the tennis above the middle of the "o" in "Computer" text of the book title, the offset $[5, 10, 3]$ was added to the data in "sphere.txt".