



Computer Vision - 16720

Homework 2


Andrew id: 

Start date: Feb 7th
Due date: Feb 23rd

Q1.1: Gaussian Pyramid

The output of Gaussian Pyramid are as below.



Figure 1: GaussianPyramid output

Q1.2: The DoG Pyramid

The output of DoG Pyramid are as below.

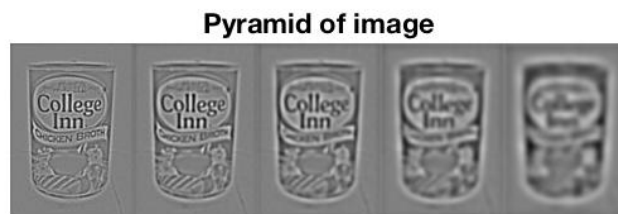


Figure 2: DoGPyramid output

Q1.3: Edge suppression

The output of edge suppression are as below. From the image we can see the edge points has been suppressed and they were not shown on the image.

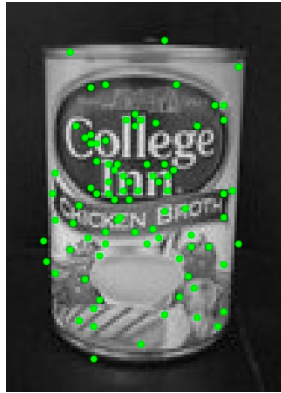


Figure 3: Output with edge suppression

Q1.4: Detecting Extrema

Please see the code submitted on blackboard.

Q1.5: Putting it together



(a) Detected keypoints



(b) Detected keypoints with using imregionalmax

Figure 4: Images with detected keypoints

From the image above we can see keypoints were detected in the image. The total amount of keypoints I got were less than the one on writeup because in finding local extrema I ignored the first and last layer. When comparing the logical outcome matrix of my function `regionalextrema` and the function `imregionalmax` and `imregionalmin` of MATLAB, the three layer in the middle got the same outcome. And since a TA said it's ok to ignore the first and last layer and the edges, I didn't calculate the local extrema of those two layers and it turns out the final outcome was right as well. The image on the right side is the detected keypoints with using `imregionalmax` and `imregionalmin`.

Q2.1: Creating a set of BRIEF tests

Please see the code submitted on blackboard.

Q2.2: Compute the BRIEF Descriptor

Please see the code submitted on blackboard.

Q2.3: Putting it all together

Please see the code submitted on blackboard.

Q2.4: Check point: Descriptor matching

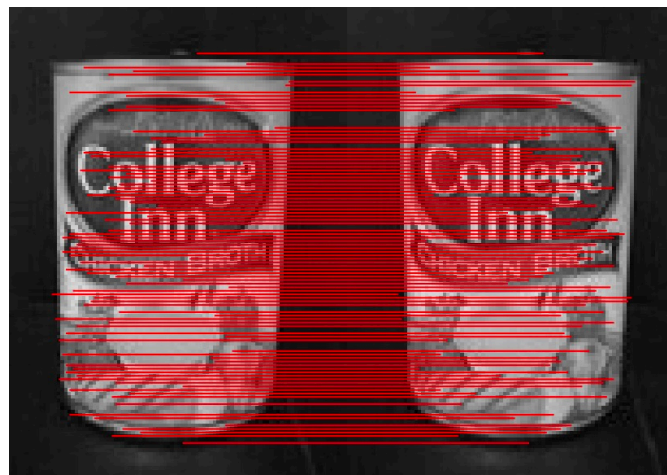


Figure 5: Image match to itself

From the figure above we can see the function performed well in matching image to itself. All matching lines are horizontal means key points are correctly matched. That's because they are exactly two same images without any rotation so they yield same keypoints and same descriptors(with same test pattern).

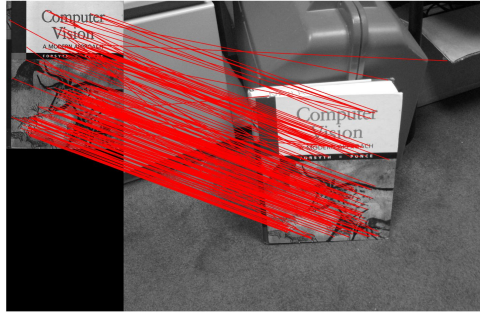


Figure 6: computer vision textbook cover page against pf stand

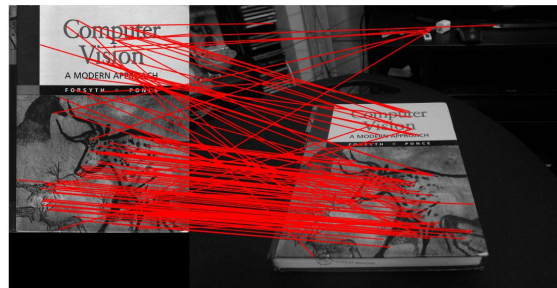


Figure 7: computer vision textbook cover page against pf desk

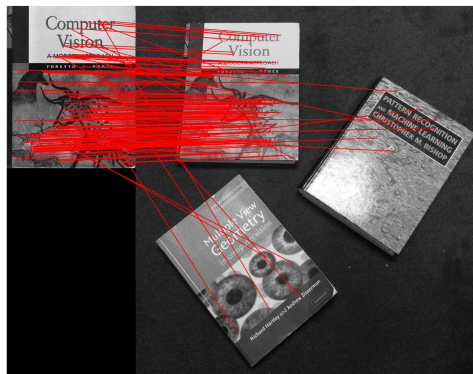


Figure 8: computer vision textbook cover page against pf floor

From the three figure above we can see most of keypoints are correctly matched and few points are wrongly matched when there's no image rotation.



Figure 9: Match of two incline images

From the image above we can see keypoints of two incline images were detected and mostly correct matched.

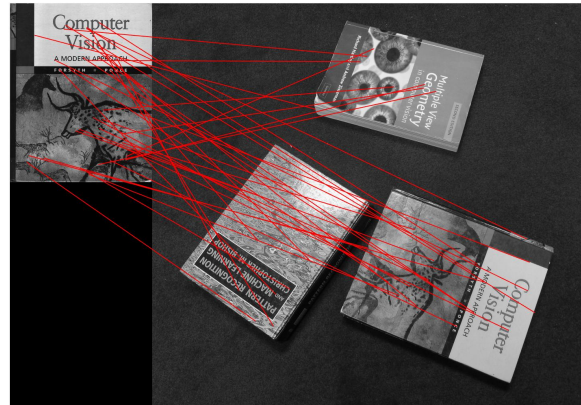


Figure 10: computer vision textbook cover page against pf floor rot



Figure 11: computer vision textbook cover page against pf pile

From the two figure above we can see most of keypoints are wrongly matched with image rotation.

From the outcome above we can see that cases without any rotation performed better than those rotated images. It's because the BRIEF descriptor encodes information from a 9×9 patch p centered around the interest point and when a image was rotated, interested point was rotated to another coordinate and the points around it were changed as well. But the patch was still extract in the same way even though the points around were changed, i.e, the patch didn't rotate along with image rotation. That's the reason rotated image cannot be matched to the origin image, or wrongly matched to another point.

Q2.5: BRIEF and rotations

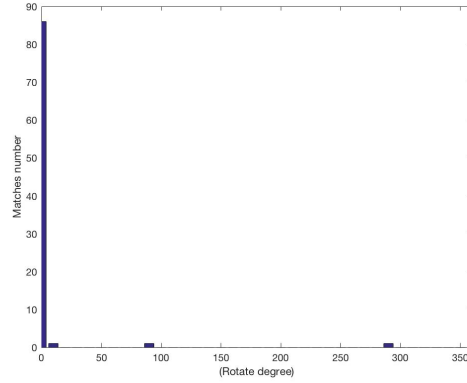


Figure 12: Number of correct matches at each rotation

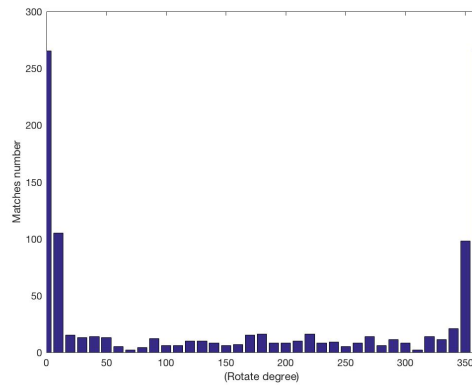


Figure 13: Number of matches at each rotation

In plotting figure 13, I simply calculated the number of total matches and plot them out. With figure 12, I firstly calculated the rotated coordinate of keypoints with a homography matrix calculated by the degree of rotation. And then compared the correct locs (rotated coordinate extracted by the indices in matches) with the real outcome (locs of rotated image calculated by function `brifflite` and extracted by the indices in matches). Then I calculated the `ssd` between them. If the `ssd` was less than the tolerance value (for figure 12: 25), I count it as a correct match. It turned out that most of them were wrong match in the total amount of matches. It's because with image rotation the coordinate of interest point itself changed and the intensity value around it changed as well. But the patch didn't rotate. So the value that unchanged index `compareX` and `compareY` lead to are changed and thus change the value (0 or 1) in descriptor. And we use this 'incorrect' descriptor to match them, so the number of correct match decrease drastically with even 10 degree rotation.

Q3.1(a): Write out an expression for A

Since \mathbf{H} is a 3*3 matrix I could write \mathbf{H} as: $\mathbf{H} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}$

If \mathbf{p} is proportional to $\mathbf{H}^* \mathbf{q}$, then I could define $\mathbf{np} = \mathbf{H}^* \mathbf{q}$. Where $\mathbf{p} = \begin{bmatrix} u_1 \\ v_1 \\ 1 \end{bmatrix}$ and $\mathbf{q} = \begin{bmatrix} u_2 \\ v_2 \\ 1 \end{bmatrix}$

$$\text{So } \mathbf{n}^* \mathbf{p} = \begin{bmatrix} nu_1 \\ nv_1 \\ n \end{bmatrix} = \mathbf{H}^* \mathbf{q} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} * \begin{bmatrix} u_2 \\ v_2 \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11}u_2 + h_{12}v_2 + h_{13} \\ h_{21}u_2 + h_{22}v_2 + h_{23} \\ h_{31}u_2 + h_{32}v_2 + h_{33} \end{bmatrix}$$

From the third row above we could derive that $n = h_{31}u_2 + h_{32}v_2 + h_{33}$

Then we could derive that $u_1 = \frac{h_{11}u_2 + h_{12}v_2 + h_{13}}{h_{31}u_2 + h_{32}v_2 + h_{33}}$ and $v_1 = \frac{h_{21}u_2 + h_{22}v_2 + h_{23}}{h_{31}u_2 + h_{32}v_2 + h_{33}}$

Then we have two equation:

$$u_1(h_{31}u_2 + h_{32}v_2 + h_{33}) - (h_{11}u_2 + h_{12}v_2 + h_{13}) = 0$$

$$v_1(h_{31}u_2 + h_{32}v_2 + h_{33}) - (h_{21}u_2 + h_{22}v_2 + h_{23}) = 0.$$

We could write this two equations by vector multiply like:

$$\begin{bmatrix} -u_2 & -v_2 & -1 & 0 & 0 & 0 & u_1u_2 & u_1v_2 & u_1 \\ 0 & 0 & 0 & -u_2 & -v_2 & -1 & v_1u_2 & v_1v_2 & v_1 \end{bmatrix} * \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \\ h_{33} \end{bmatrix} = 0$$

That's the outcome of one pair of points, i.e, $\mathbf{a}_i * \mathbf{h} = 0$ and the left matrix is \mathbf{a}_i .

$$\text{With more pairs of points, } \mathbf{A} = \begin{bmatrix} \mathbf{a}_1 \\ \mathbf{a}_2 \\ \dots \\ \mathbf{a}_N \end{bmatrix} \text{ and } \mathbf{h} \text{ is same as above, } \mathbf{A}\mathbf{h} = 0$$

(b): How many elements are there in h?

Ans: Since elements in h is the elements in H sorted into a vector, there are 9 elements in h.

(c): How many point pairs (correspondences) are required to solve this system?

Ans: As the answer of question (a) shown, each pair could provide two equations, and since the degree of freedom of \mathbf{H} is 8, we'll need at least 8 equations to solve it out. Which means we need 4 pairs to solve this system.

(d): Step through the procedure of estimating the elements in h to find a solution to minimize this homogeneous linear least squares system

Ans: Matrix A can be computed in the way in Q(a). To minimize the homogeneous linear least square system we want to find $h = \arg \min_h \|\mathbf{A}h\|$. First we take a derivation and come up with:

$$d(h^T A^T A h + \lambda(1 - h^T h))/dh = 0$$

Solve this we can get:

$$2A^T A h - 2\lambda h = 0$$

We could derive:

$$(A^T A - \lambda)h = 0$$

So h is an eigenvector of $A^T A$. To find the h minimize this system, we can compute svd of A:

$$A = U \Sigma V^T = \sum_{i=1}^9 \sigma_i u_i v_i^T$$

In MATLAB the singular values σ will be sorted in a descending order, so σ_9 is the smallest singular value. To correspond with this smallest singular value σ_9 , we take the last column of singular vector (right singular vector) from svd as h. That's the solution h to minimize the homogeneous linear least square system.

STEP:

- (1): A can be computed by the way derived in Q(a).
- (2): Apply svd to A and get the eigenvector matrix V.
- (3): h is the last column of V.

Q4.1: Planar Homographies: Implementation

At first I did the DLT normalization for the input matrix because my output of warped image was totally twisted at first and I read some paper said a normalization might help. Well the normalization did help but the main problem about the code was in the RANSAC because I didn't find the 'most inlier' at first. So I comment the lines of calling DLT normalization function but also kept this function in case I might use it in future.

Q5.1: RANSAC

Please see the code submitted on blackboard.

Q6.1: Warp img2 and blend them together

The warped image is as below. And the outcome of panorama was clipped at the edge. It'll be fixed in Q6.2.

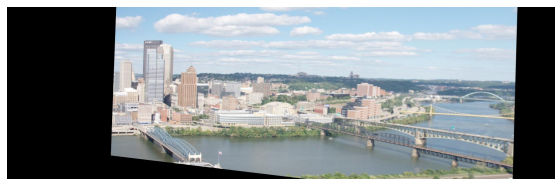


Figure 14: Img2 warped into img1's frame



Figure 15: Panorama image with edge clipped

Q6.2: Stitching image without clip

Please see the code submitted on blackboard.

Q6.3: Output panorama

The panorama is as below.



Figure 16: Final panorama view.

7: Extra credits

To fix this, we could use SIFT (scale invariant feature transform) descriptor instead of BRIEF to match key point. In the assignment, the matrix contains the coordinate and their level of the key point. Instead we could add their direction information and scale. The new matrix locs should be $p \times 4$ where p is the number of detected keypoints. And the descriptor should be $p \times 128$ with each row is the descriptor of a certain keypoint.