Report of Problem Set 1

Name: zihaosheng
Student ID: 000000000000 **2020/04/08**

1. Consider the filter f=[1, 2, 1] and the 1D image I=[0,1,2,3,3,3,1,3,6]. What is the results of f*I? Pad the image with zeros at the boundaries if necessary.

In order to make the result have the same length as the original image, I added a zero on each side of the original image respectively, and the result is [1 4 8 11 12 10 8 13 15].

2. Name two specific ways in which one could reduce the amount of fine, detailed edges that are detected with the Canny edge detector.

1. We can use Gaussian kernel with larger size.
2. And another choice is to raise threshold.

3. Hybrid images. In this problem you will create hybrid images as described in [1]. Take two images, A and B, that you' ll want to have blend from one to the other. Try to make the objects in the two images occupy more or less the same region. Construct a hybrid image from A (to be seen close-up) and B (to be seen far away) as follows: out = blur(B) + (A-blur(A)) Where blur is a function that low-pass filters the image. You can write your own blur function, or use the upBlur and blurDn functions supplied in [2] (which go up and down. Gaussian pyramid levels). You will want to blur by more than just one Gaussian pyramid level. How does the blurring level affect your perception of the results?

My code is in '/q3/hybrid.py'. The images that I deal with are cat and tiger. As shown in Fig. 1, the low-passed image is tiger, so we can see it at a further distance, and the high-passed image is cat, so it can be perceived at a closer distance.

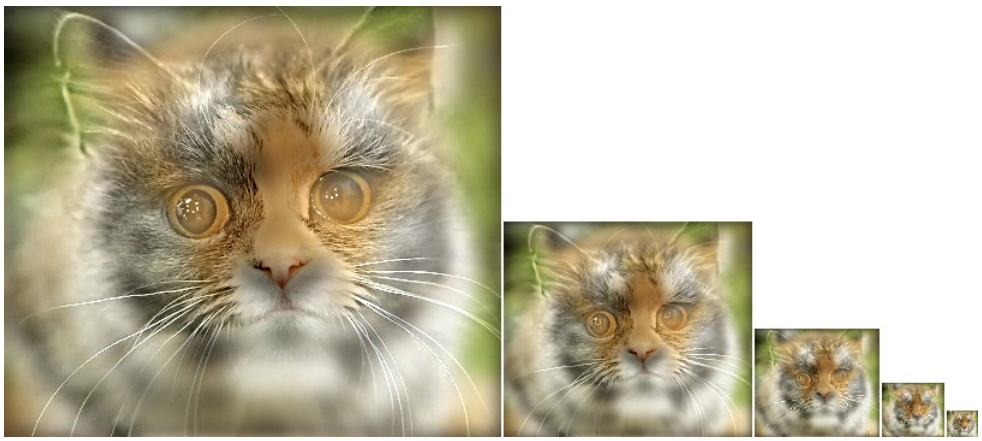

图 1: Hybrid image of cat and tiger and visualization.

After I try different parameters of blur functions, I find when the sigma of low pass filter becomes larger, which means the tiger becomes more blurred, we have to move to a further distance than before

to perceive the tiger. And when the the sigma of high pass filter becomes larger, which makes the cat becomes clearer, so we can see the cat at a closer distance than before and we have to move to a further distance than before to perceive the tiger. More details are in '/q3/hybrid.py'.

4. Read the book chapter chapter-local-image-feature-David.pdf (find it in canvas) and complete the following programing exercises.

(1) Build a Harris corner detector; for each corner, estimate scale and orientation. Now test how well your list of neighborhoods behaves under rotation, translation, and scale of the image. You can do this by a simple exercise in matching. For each test image, prepare a rotated, translated, and scaled version of that image. Now you know where each neighborhood should appear in the new version of the image —check how often something of the right size and orientation appears in the right place. You should find that rotation and translation cause no significant problems, but large scale changes can be an issue.

As shown in Fig. 2, I build a Harris corner detector, and scale and orientation are estimated for each corner. And the code is '/q4/harris.py'.

图 2: Estimation of scale and orientation using Harris corner detector.

For each test image, I prepare a rotated, translated, and scaled version of that image, and achieve a simple matching in Fig. 3, 4 and 5. And the code is '/q4/match.py'.

图 3: Mathces of rotation using Harris corner detector.

图 4: Matches of translation using Harris corner detector.

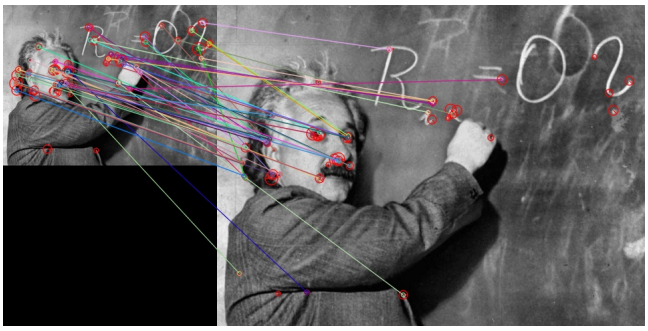

图 5: Matches of scaling using Harris corner detector.

We can see that translation has the least influence on corner recognition. For rotation, we can see that there are still many matches between the eyes and the hands. But maybe it's because of the problem of the descriptors I use, some places don't match. When the image becomes smaller, there are many corners that have not been recognized before. As we can see, some of the corners of the letter R are paired to the ears. And some interest points of the number like 2 on the blackboard are matched to the eyes and the letter R.

(2) Use DoG detector in SIFT to detect the interesting points for the same test image, and compare the results with Harris corner detector on the variances of rotation, translation and scaling.

To facilitate observation, I only drew 150 matching points. It can be seen that the effect of DoG in SIFT is better. No matter rotation, translation or scaling, it gives better recognition results, and matches well. And the code is '/q4/DoG-SIFT.py'.

图 6: Mathces of rotation using DoG in SIFT.

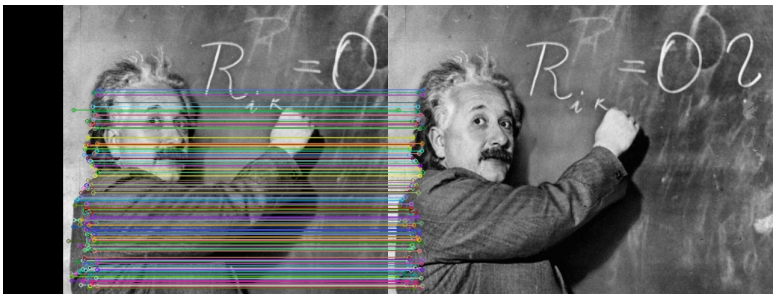

图 7: Matches of translation using DoG in SIFT.

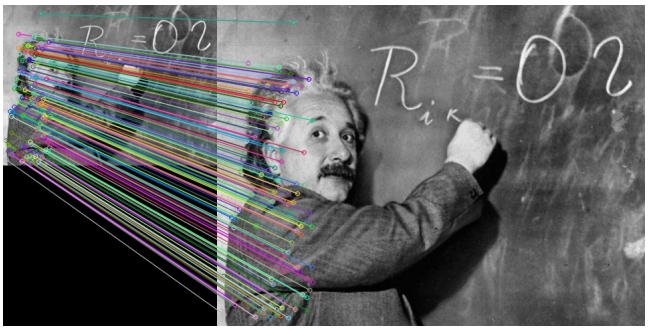

图 8: Matches of scaling using DoG in SIFT.