HW1 Report

蕭文逸 **105062581**

Part 1

A. Gaussian filter

Results:

(σ=5, kernel size = 3)

(σ=5, kernel size = 30)

Discussion:

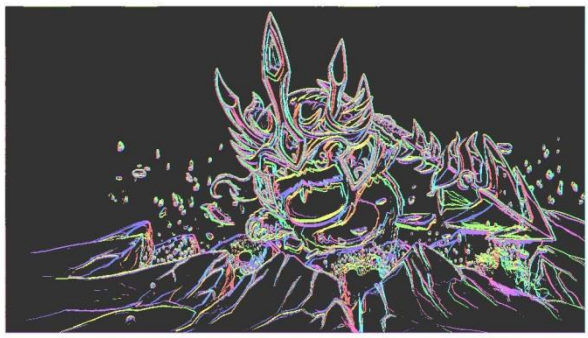
With Increasing of “kernel size” of Gaussian filter, the image will be more blurred.

Here I use building-in function “fspecial.m” and “imfilter.m” for implementation.

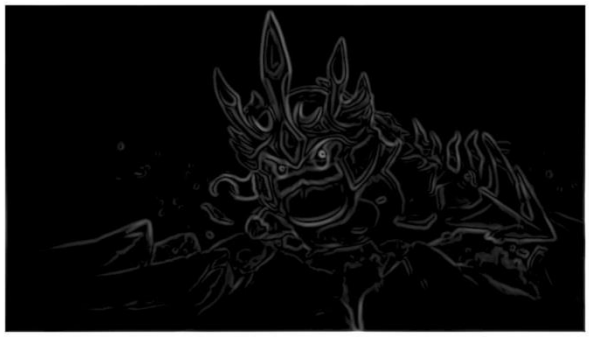
B. Applying Sobel Mask

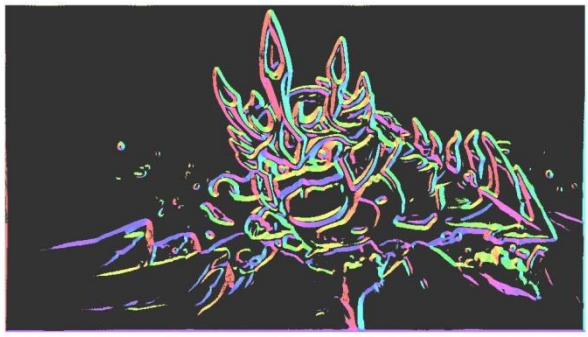
Results:

(Kernel size = 3: Top: magnitude, Bot: direction)



(Kernel size = 30: Top: magnitude, Bot: direction)





Discussion:

1. In figure kernel size = 3, There are more details being reserved. For example:

  (kernel size = 3 & 30)

   Obviously, the lines in left thinner and the angles re more pointed. Additionally, there are more “rocks” in there.
2. The strength of magnitude of “kernel size = 3” figure is bigger. In magnitude figures, the lightness of white line reveals the strength of magnitudes and “kernel size = 3” figure is brighter.
3. The distributions of direction of two different kernel size images are similar except the portion whose details being discarded in “kernel size = 30” figure.

Implementation:

-1	0	+1
-2	0	+2
-1	0	+1

Gx

+1	+2	+1
0	0	0
-1	-2	-1

Gy

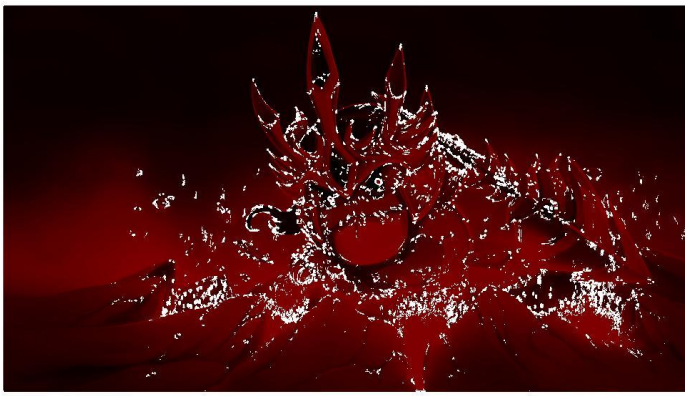
Generating two mask for two coordinates, and then do convolution to obtain results along X and Y. Using there results, the we can get “direction” & “magnitude” from some formula.

For visualization of “direction”, I implement an additionally function: “colorMapGenerator.m”. With this function, we can map the angles to different colors.

C. Structure Tensor

Results:

(Kernel size = 3; Left: 3 x 3, Right: 5 x 5)




(Kernel size = 30; Left: 3 x 3, Right: 5 x 5)


Discussion:

1. With increasing of kernel size of Gaussian filter, there are less “smaller eigenvalues” labeled in the figure.
2. With increasing of window size , there are more “smaller eigenvalues” labeled in the figure.
3. White points agglomerate, which indicate the position of corners.

Implementation:

$$\begin{bmatrix} I_x^2 & I_x I_y \\ I_y I_x & I_y^2 \end{bmatrix}$$

Following the steps:

1. Utilizing former Sobel mask results to compute above matrix.
2. Summing up all values with approprate window size. (Convolution)
3. Eigen decomposition to obtain and select smaller eigenvalues
4. Applying proper threshold (= 1100 here) to eliminate noise

D. Non-Maximal Suppression

Results:

(Kernel size = 3; Left: 3 x 3, Right: 5 x 5)


(Kernel size = 30; Left: 3 x 3, Right: 5 x 5)


Discussion:

1. The results without non-maximal suppression have many “white blocks”. However, here we can see all blocks become points which indicate corners.
2. Again, If the parameters of a figure with higher window size and lower kernel size, it will have more “spots” on it. Because it have more “blocks” in its original version.
3. The position of “spots” will be slightly different from each other, because of the effectiveness of Gaussian filter and the selecting way.

Implementation:

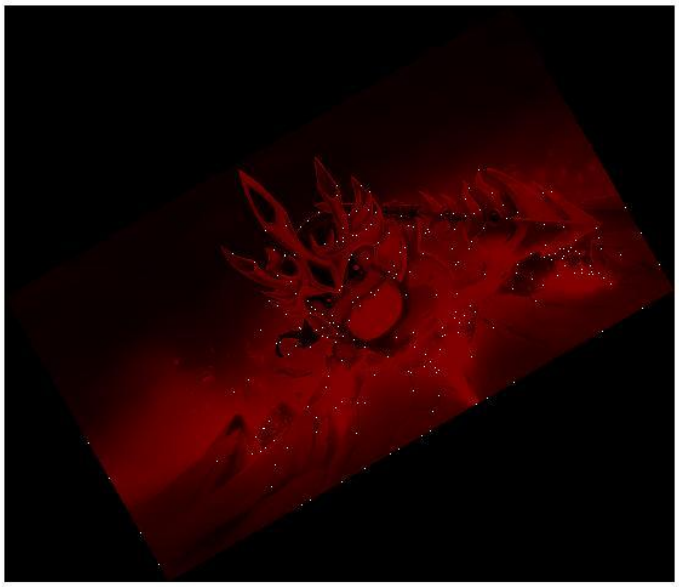
Selecting out the maximum block from a block (size = 7), assign to 1 and 0 for others, instead.

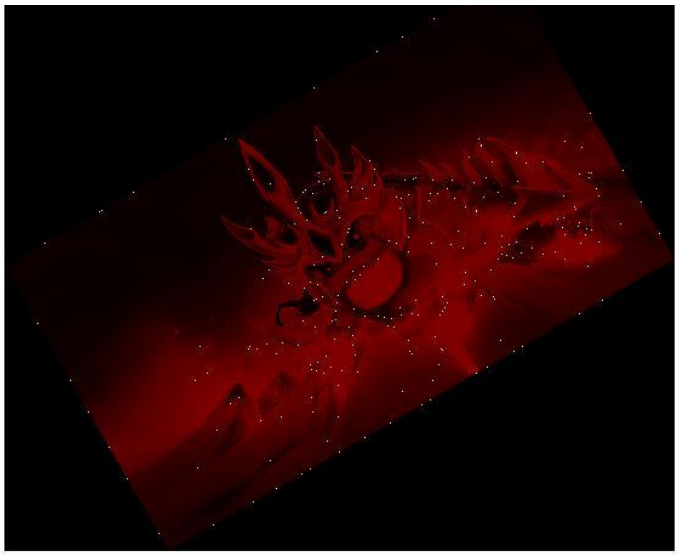
(the maximum pixel is itself) && (this block has not been assigned a representation yet)

E. Rotation and scaling

Results 1:

(Rotation = 30, Kernel size = 3; Top: 3 x 3, Bot: 5 x 5)





Discussion:

After rotation, the distribution of spots is similar to original. However, there is more noise in the margin of these figures. Because this approach of corner detection considers only along X or Y coordinates, if we rotate the figure, it may see rotated margin as “corner” along original coordinates.

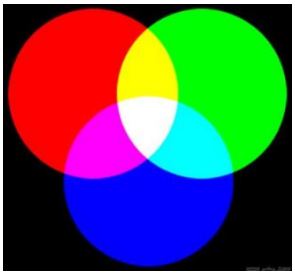
Results 2:

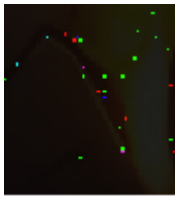
(Scaling = 0.5, Kernel size = 3; Top: 3 x 3, Bot: 5 x 5)

Discussion:

Obviously, the spots are getting more. This is because after resizing image into a smaller one, some details would be discarded such as lines for connecting different portion, etc. Aside from that, the distribution of spots would appear in some non-reasonable places.

F. Comparison





Red: Rotation
Green: Scaling
Blue: Normal

I use three basic colors for representing different kinds of processing approach. The result reveals that the spots usually group in adjacent places. However, there are few white points, which mean there are few three kinds of result have the same position at the same time.
The reason would be the way we restore the images. Using build-in functions such as “imresize” (for scaling) or “imrotate” (for rotation), both of them would discard some information due to interpolation. Additionally, these steps may also cause some point number being disappeared. So when we restore and try to compare them, they usually appears in near spaces.

The distribution of these three kinds of approach has been discussed in part E. And the bottom-right one of above three images shows these properties: there are most green spots on it, RGB points appear in adjacent spaces but rarely at the same places.

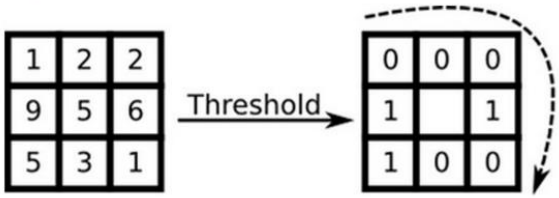
Part 2

A. LBP

Results:

(Left: Kobe, Right: Gasol)

Implementation:

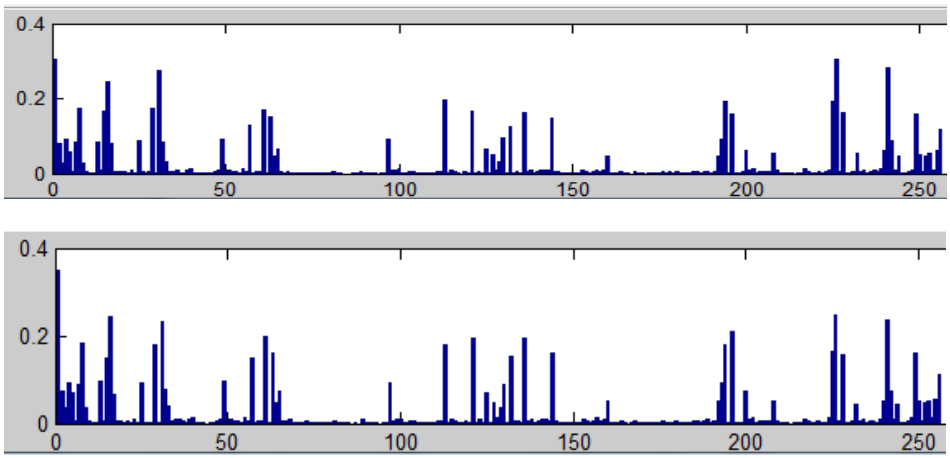


1. Threshold a block with the central pixel
2. Concat all bits in clockwise direction to form a steam of binary bits
3. Compute this stream of binary bits in decimal and Assign to the pixel

B. Histogram and Similarity

Results:

(Top: Kobe, Bot: Gasol)



Similarity: 0.9908

Implementation:

Put values of each pixel into buckets accordingly. Then do “vector normalization” and “inner product”.

C. Division of Images

Results:

```
 2x 2 = 0.962291
 3x 3 = 0.929781
 4x 4 = 0.937501
 9x 9 = 0.840593
20x20 = 0.770329
```

Implementation:

Concat new vector in the end of old vectors. So the dimension of a compact vector would be 256 * n^2.

Discussion:

The reason why the similarity would decrease with increase of number of divided blocks is the dimension. If we divide one picture and compute their histogram respectively and finally concat all of them, the difference between two original pictures will be amplified. And the most direct reflection of this phenomenon would be the dimension.

D. Uniform LBP

Results:

(Left: Kobe, Right: Gasol)

Implementation:

Create a lookup table that indicates how to map original LBP values to new uniformed ones. The construction of a table is as the following steps:

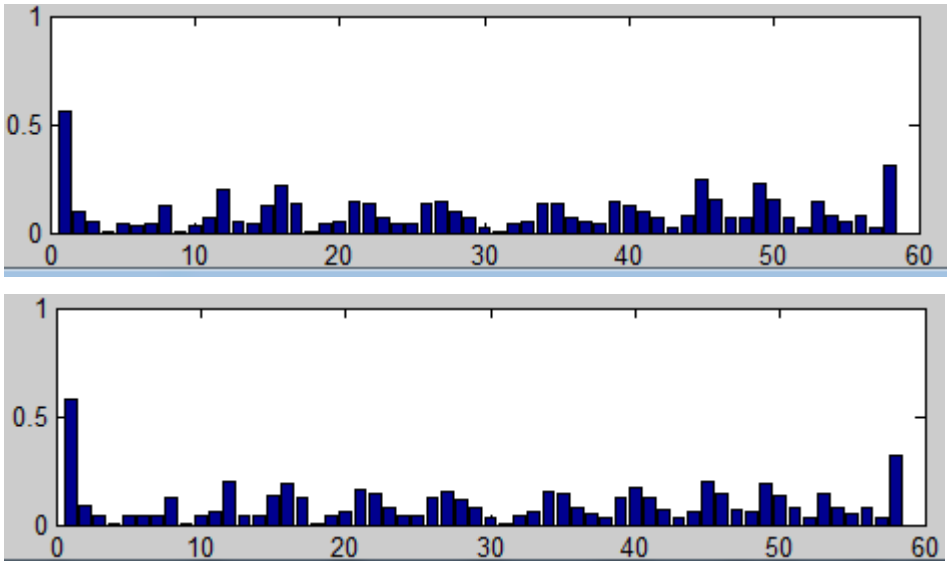
1. From 0:255 (Original 256 dimension of LBP)
2. To examine the number of transition
3. If it's two or zero, give it a "label"
4. Otherwise, label it with 0.

Finally, we can get a table that the biggest label is 58 (0 ~ 59).

E. Histogram and Similarity

Results:

(Top: Kobe, Bot: Gasol)



Similarity: 0.9944

Implementation:

See LBP (Part B).

F. Division of Images

Results:

```
 2x 2 = 0.976039
 3x 3 = 0.955893
 4x 4 = 0.958307
 9x 9 = 0.904792
20x20 = 0.833632
```

Implementation:

See LBP (Part C).

Discussion:

Overall, the similarity is decreasing, but note that the ability to distinguish two images is worse than LBP. The reason is there are less dimensions in uniformed LBP (only 59). Less dimension means it’s harder to descript the difference.