HW2 practical

==========

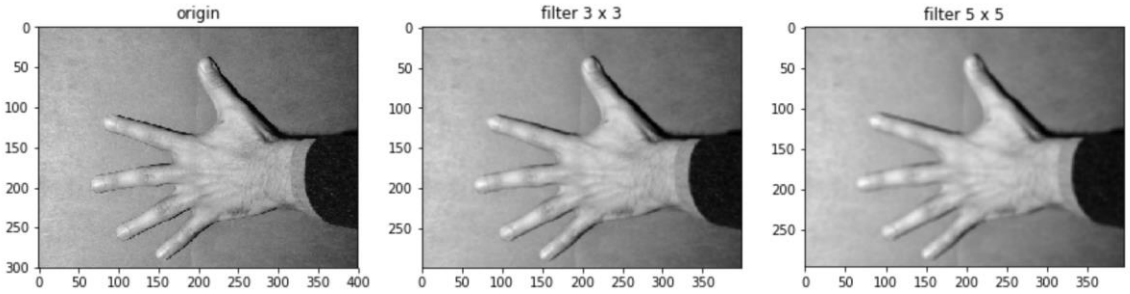
Lifan Wang lw2435 N14854019

==========

Q1:

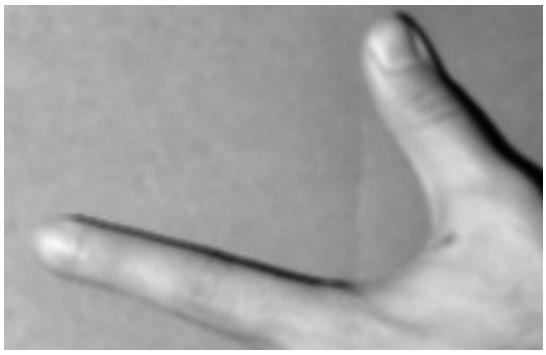
Algorithm: prepare a 3 x 3 and a 5 x 5 filter, write a conv_2d function to move the filter throughout the image. Just pay attention the filters need to be normalized.

Result:



Zoom in:

Filter 3 x 3



Filter 5 x 5

Analysis:

The larger the kernel is ,the more blurring effect it would perform!

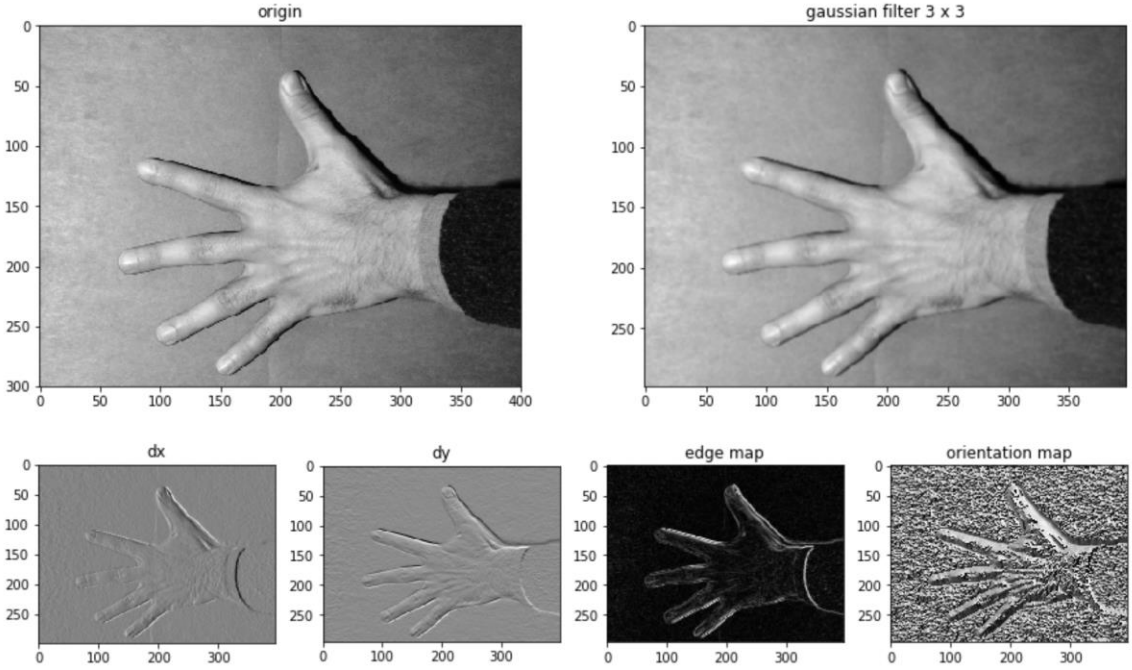
Q2:

Algorithm:

Step1:Use Gaussian filter to smooth the original image.

Step2:Use derivative filter to output the x-edges, the y-edges, the edge map, the orientation map of the original images.

Result:



Analysis:

1. Python “image.show” function would not auto-adjust the intensity value from range[-a,b] to range[0,255], so we need to write a function “rgb_kernel” to adjust that, meanwhile changing from single grey channel to RGB channels
2. Edge map has shown more details than dx-map and dy-map
3. Orientation map sort of shown the depth of the image
4. While calculating the orientation, we need to use np.arctan2(x,y) function to avoid unwanted division by zero.

Q3:

Algorithms:

Step1:Rewrite the conv_2d(A,B) function to preprocess the two elements “A,B” by normalizing them

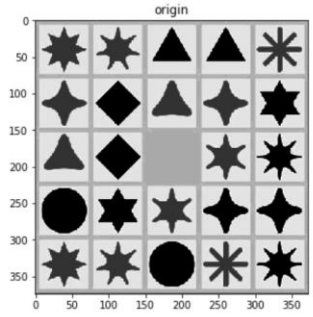
Step2:carve out a template from the original image and feed into parameter A, and the B stands for the original image, then the output of the function conv_2d would be the correlation map

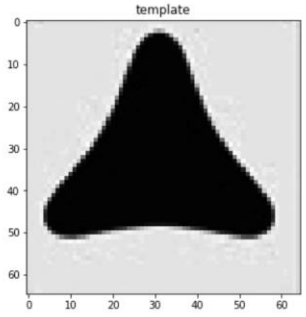
Step3:use a threshold to suppress the non-local maximum value to zero

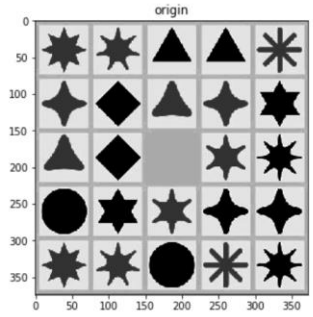
Step4:use a Laplace kernel to enhance the local maximum value

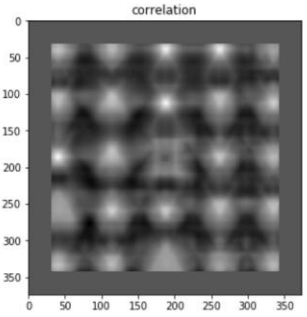
Step5:attach the template onto the local maximum location.

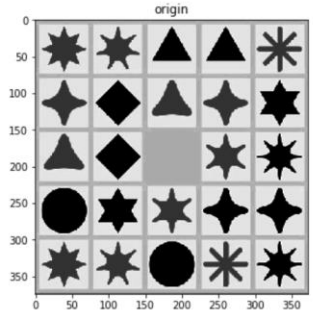
Result:

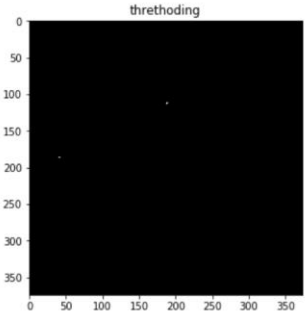

origin

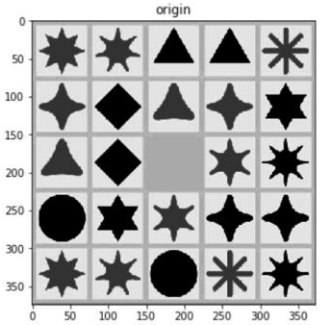

template

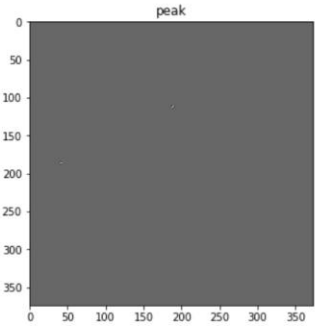

origin

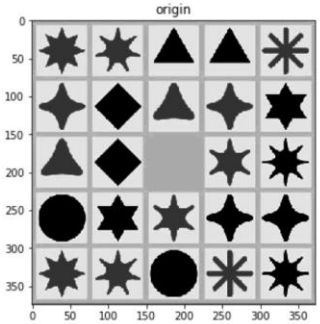

correlation

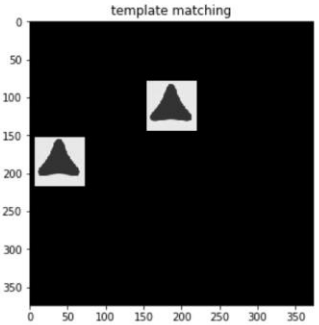

origin


threthoding


origin


peak


origin


template matching

Analysis:

1. If we do not perform normalization onto the window of the filter, the correlation value would no longer be limited by [-1,1], Sometimes would exceed the maximum value limitation of a certain type(eg. ‘int’). So Normalization would be important
2. We would use peak detection via laplace kernel by following Canny's idea of "non-maximum suppression"
3. Since we could threshold out the locations of peaks in the thresholding step, why bother use Laplace peak detection to achieve the same result in the following step?

Documentation:~~~~~~

Grey_kernel(): transform the “[pixels x 3]” 2D-array into [pixels] 1D-array

Squ_image(): transform the [pixels] 1D-array into [height x width] 2D-array

RGB_kernel(): scale the [height x width] 2D-array, transform it into [height x width x 3] 3D-array

Conv_2d: mode1: directly convolution; mode2 :normalized convolution(correlation)

Flow chart~~~~~~

Grey_kernel()
Squ_image()
Conv_2d()
RGB_kernel()
Showpic()