CS-512-AS2-REPORT

Chen Xu

A20377739

1. Problem statement

1. Use OpenCV read a graph and use webcam to capture a graph and save.
2. When press a key perform the operation corresponding to the key on the original image.
3. Implement reload the original image
4. Save image
5. Convert the image to grayscale using OpenCV conversion function.
6. Convert the image to grayscale using my own implementation.
7. Cycle through the color channels of the image showing
8. Convert the image to grayscale and smooth using the OpenCV function.
9. Convert the image to grayscale and smooth using my function.
10. Down sample the image by a factor of 2 without smoothing.
11. Down sample the image by a factor of 2 with smoothing.
12. Convert the image to grayscale and perform convolution with a x derivative filter.
13. Convert the image to grayscale and perform convolution with a y derivative filter.
14. Show the magnitude of the gradient normalized to the range [0, 255].
15. Convert the image to grayscale and plot the gradient vector of the image every N pixels and let the plotted gradient vector have a length of K.
16. Convert the image to grayscale and rotate it.

2. Proposed solution

1. convert color to grayscale:

   gray = 0.2989 * r + 0.5870 * g + 0.1140 * b

2. Gaussian filter:

   a Gaussian filter modifies the input signal by convolution with a Gaussian function. Gaussian filters have the properties of having no overshoot to a step function input while minimizing the rise and fall time. This behavior is closely connected to the fact that the Gaussian filter has the minimum possible group delay. (reference: https://en.wikipedia.org/wiki/Gaussian_filter)

3. Sobel filter:

   used in image processing and computer vision, particularly within edge detection algorithms where it creates an image emphasizing edges. it is a discrete differentiation operator, computing an approximation of the gradient of the image intensity function. At each point in the image, the result of the Sobel–Feldman operator is either the corresponding gradient vector or the norm of this vector. The Sobel–Feldman operator is based on convolving the image with a small, separable, and integer-valued filter in the horizontal and vertical directions and is therefore relatively inexpensive in terms of computations. (reference: https://en.wikipedia.org/wiki/Sobel_operator)

3. Implementation details

1. First capture of the cam doesn’t get enough light, so I make it record for 15 times and get the last one.

2.When use waitKey (0), do not close the window manually. It will require reopen the CMD and recall the python.

3.Use return to break from multiple loops

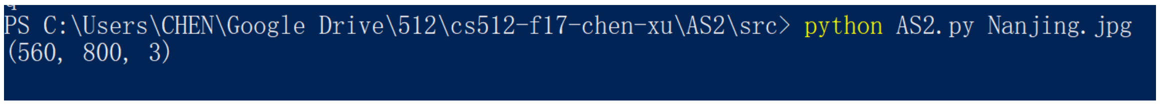
4.When call the Sliderhandler () function, the createTrackbar () passing the value to it and, callback again and again, change the filter kernel and render the picture with new filter.

5. Do not use wrong global variables, which made it hard to debug.

6.first create a window ,then add image to it.

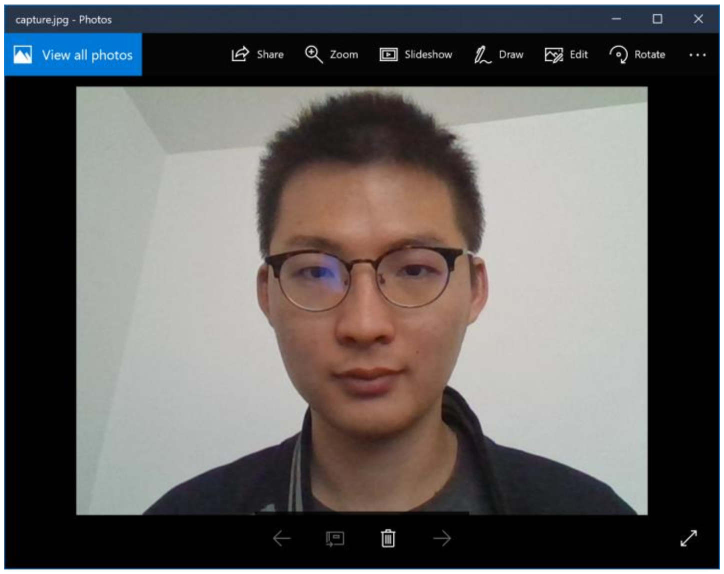
4. Results and discussion

1. Read a image

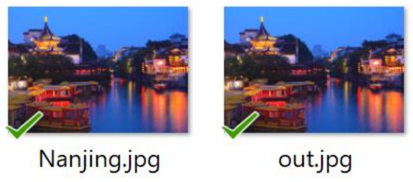


```
PS C:\Users\CHEN\Google Drive\512\cs512-f17-chen-xu\AS2\src> python AS2.py Nanjing.jpg
(560, 800, 3)
```

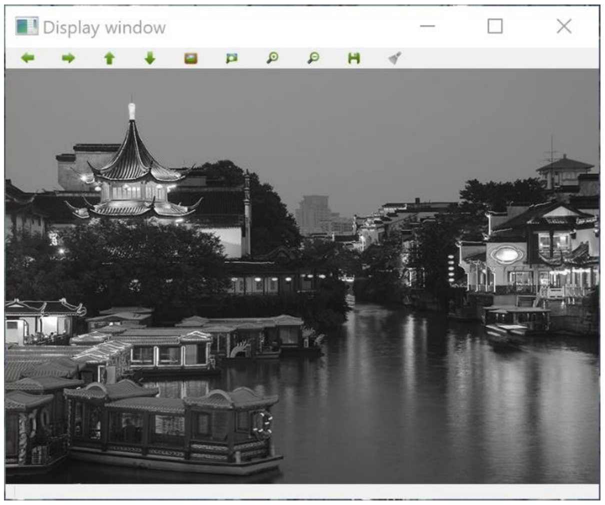
2. Capture a image



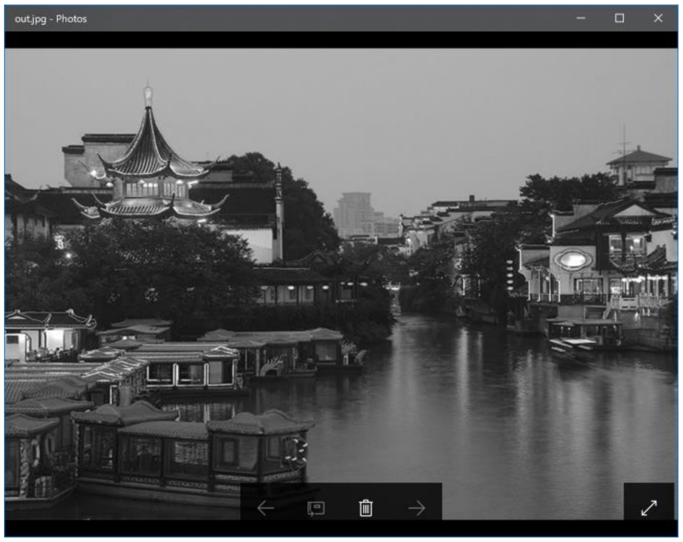
3. Save the current image to the file ‘out.jpg’



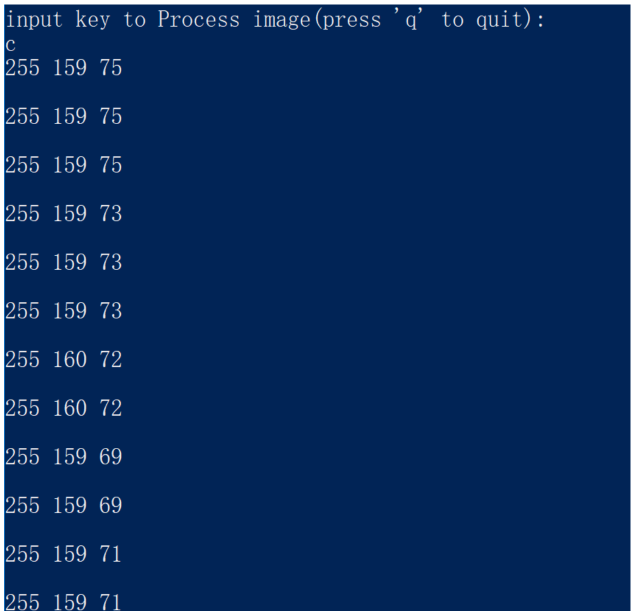
4. Convert the image to the grayscale use OpenCV function



5. Convert the image to the grayscale use my implement

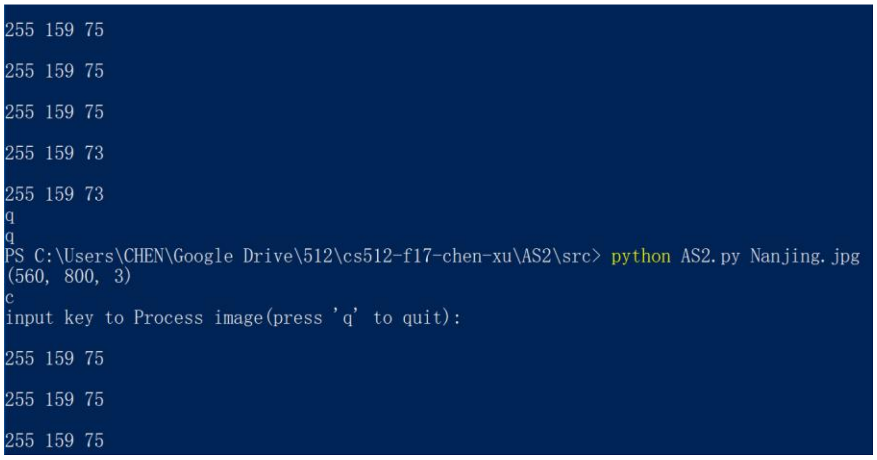


6. Cycle through the color channels, every time the key was pressed.



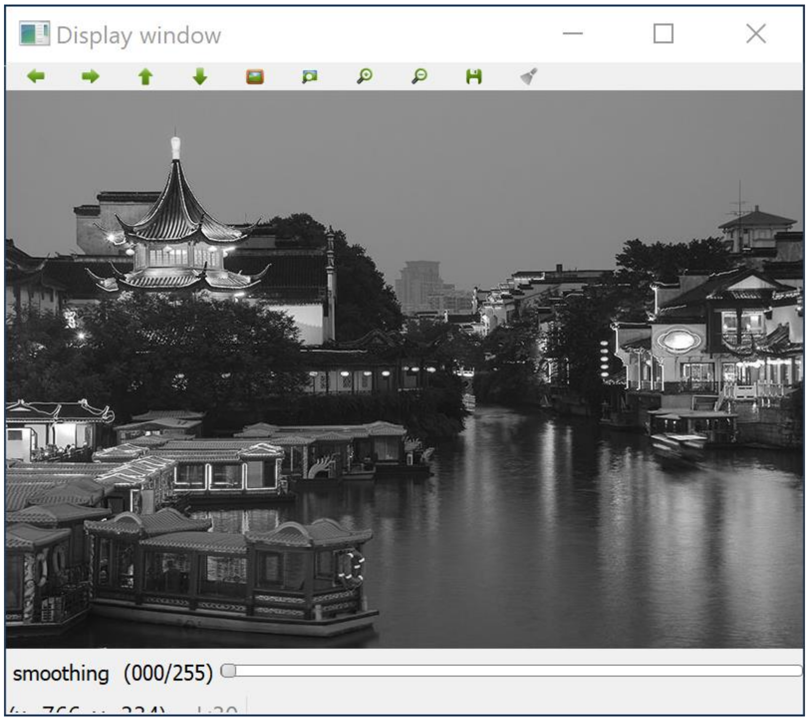
```
input key to Process image(press 'q' to quit):
c
255 159 75

255 159 75

255 159 75

255 159 73

255 159 73

255 159 73

255 160 72

255 160 72

255 159 69

255 159 69

255 159 71

255 159 71
```

Press q to quit.

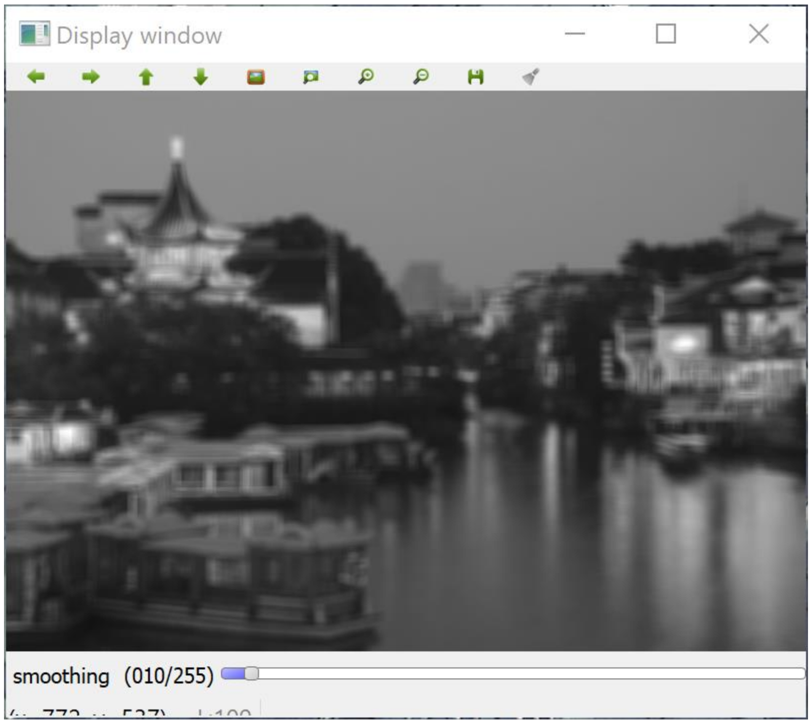


```
255 159 75

255 159 75

255 159 75

255 159 73

255 159 73
q
q
PS C:\Users\CHEN\Google Drive\512\cs512-f17-chen-xu\AS2\src> python AS2.py Nanjing.jpg
(560, 800, 3)
c
input key to Process image(press 'q' to quit):

255 159 75

255 159 75

255 159 75
```

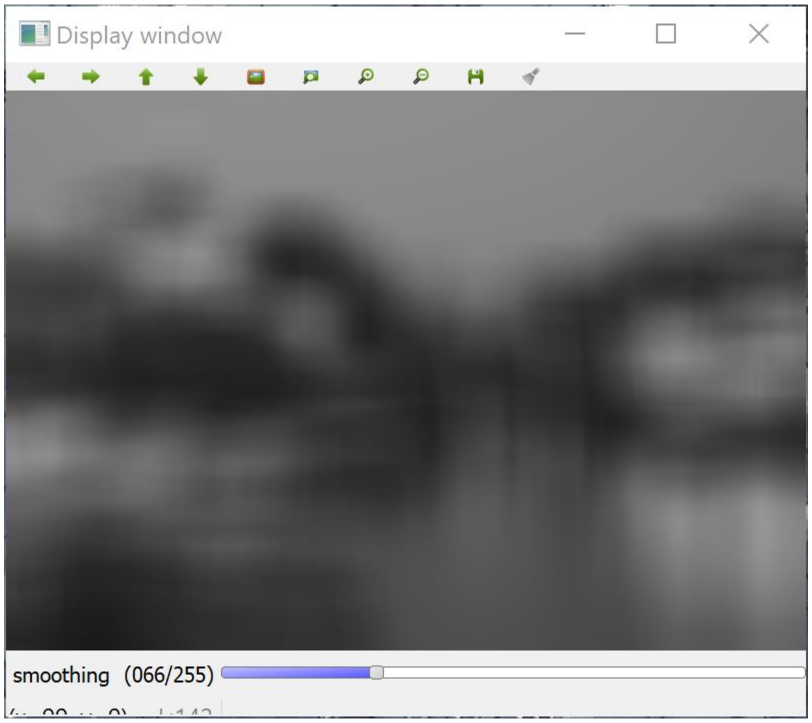
7. Convert the image to grayscale and smooth it using OpenCV function.



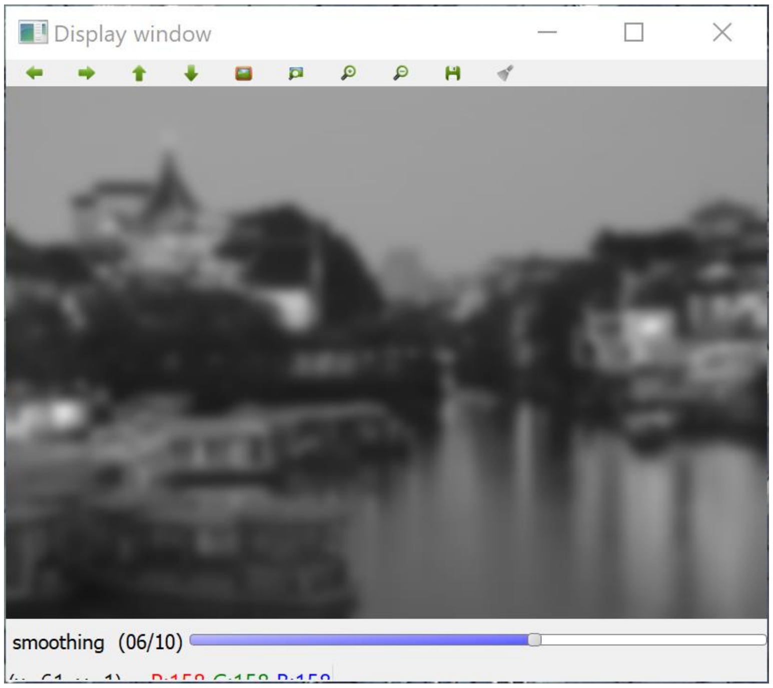
smoothing (000/255)



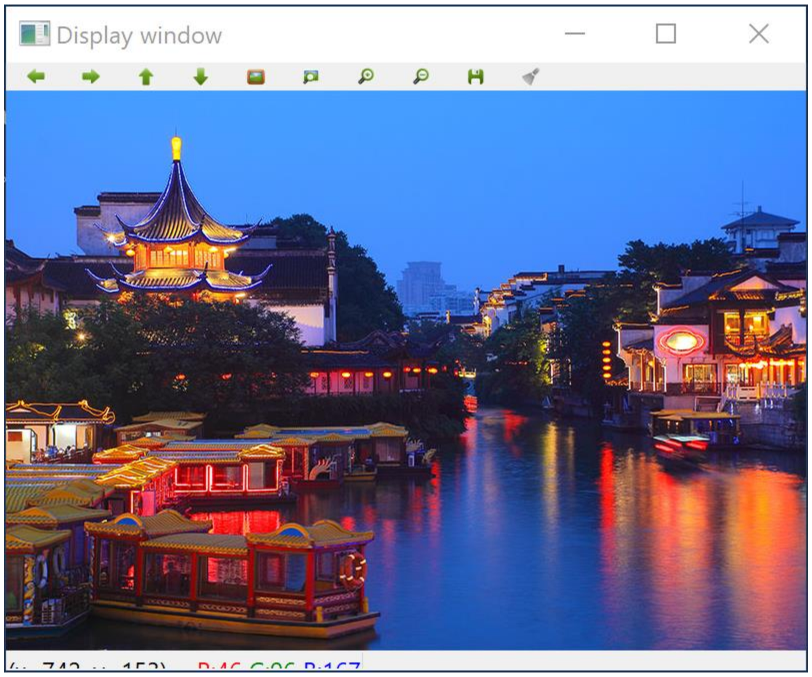
smoothing (010/255)



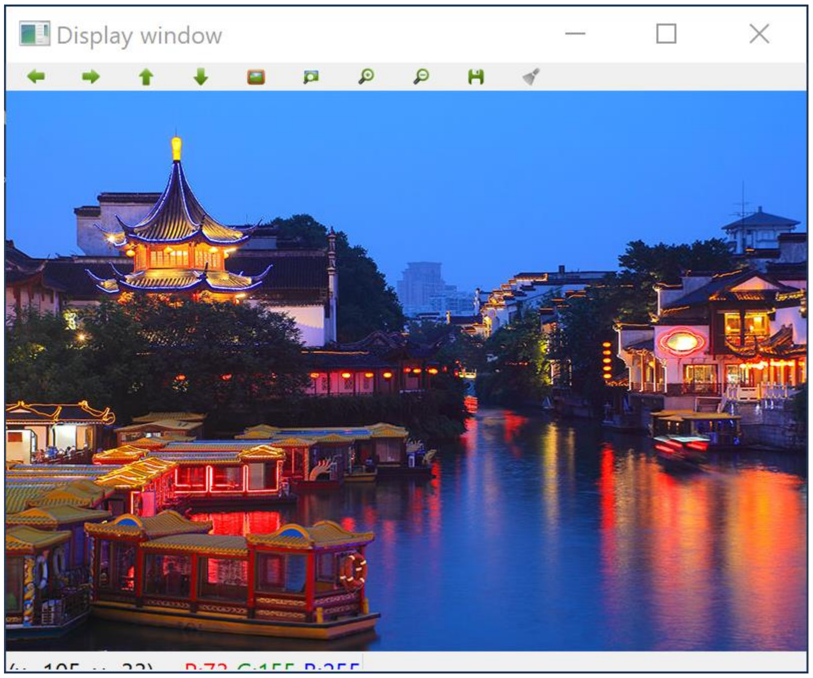
8. Convert the image to grayscale and smooth it using my implement.



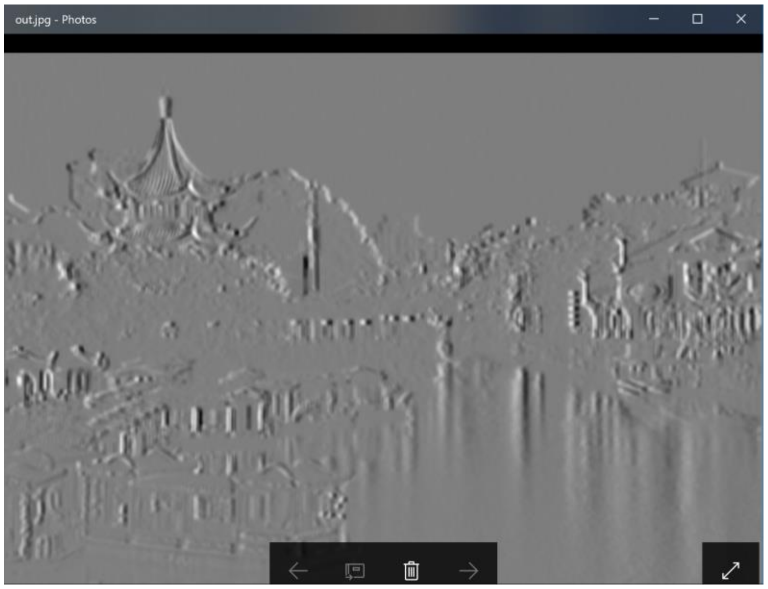
9. Down sample the image by factor of 2 without smooth



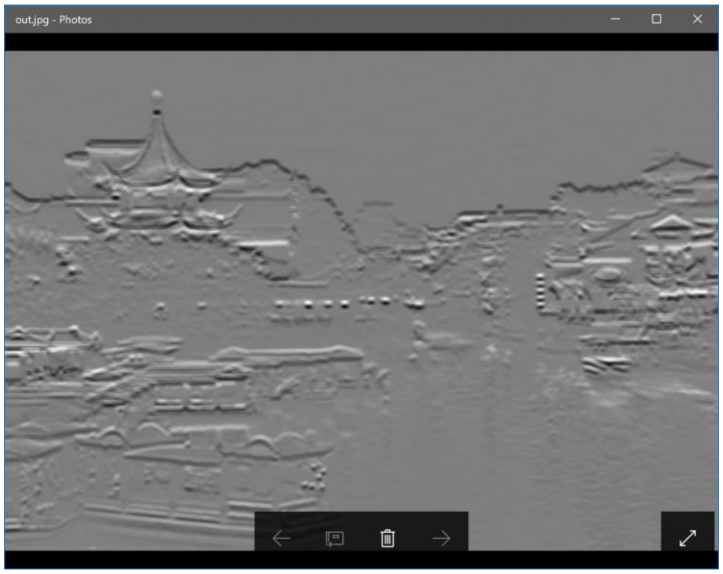
10. Down sample the image by factor of 2 with smooth



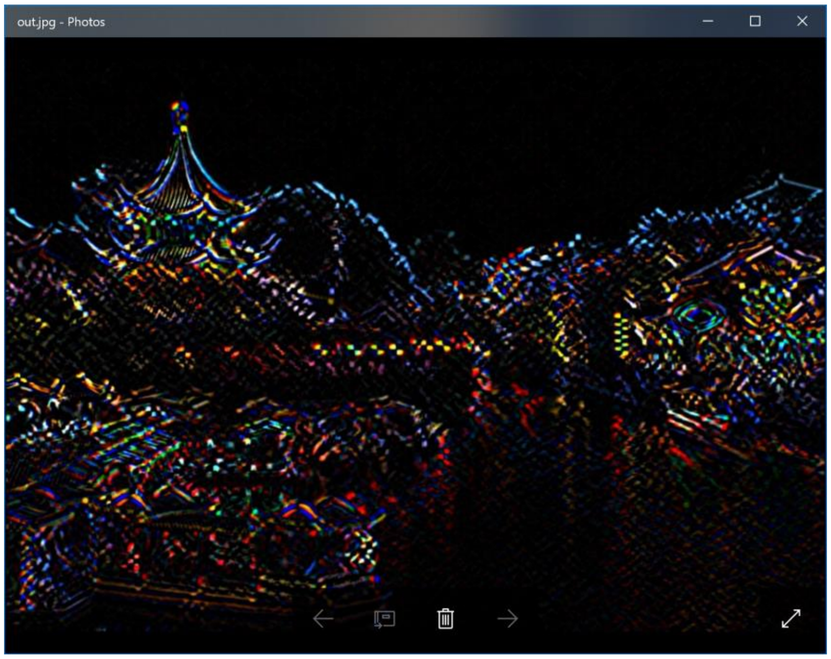
11. Convert the image to grayscale and convolution with X derivative filter. Normalize the image.



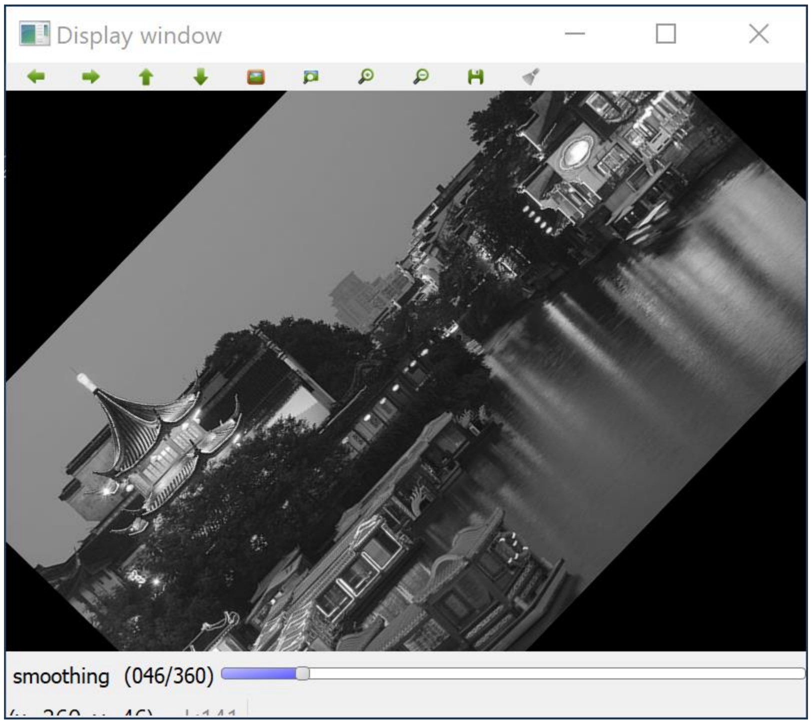
12. Convert the image to grayscale and convolution with Y derivative filter. Normalize the image.

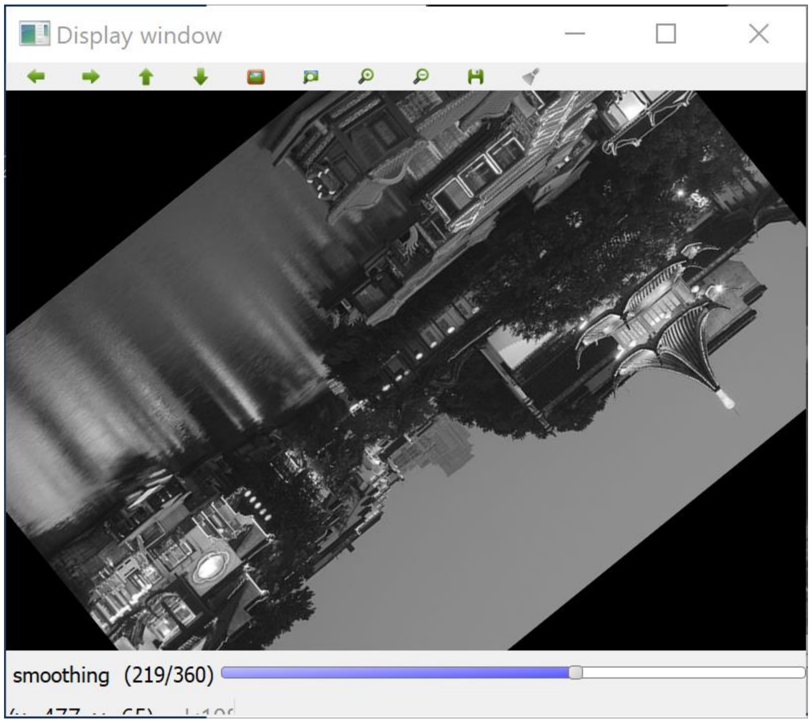


13. Show magnitude of the gradient, compute based on x and y derivative.

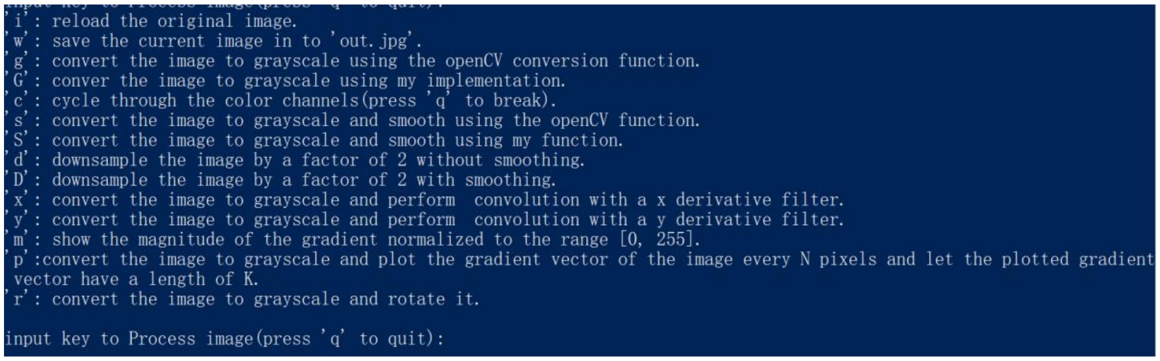


14. Convert the image to grayscale and rotate it.





15. Display the help of this program



```
'i': reload the original image.
'w': save the current image in to 'out.jpg'.
'g': convert the image to grayscale using the openCV conversion function.
'G': conver the image to grayscale using my implementation.
'c': cycle through the color channels(press 'q' to break).
's': convert the image to grayscale and smooth using the openCV function.
'S': convert the image to grayscale and smooth using my function.
'd': downsample the image by a factor of 2 without smoothing.
'D': downsample the image by a factor of 2 with smoothing.
'x': convert the image to grayscale and perform  convolution with a x derivative filter.
'y': convert the image to grayscale and perform  convolution with a y derivative filter.
'm': show the magnitude of the gradient normalized to the range [0, 255].
'p':convert the image to grayscale and plot the gradient vector of the image every N pixels and let the plotted gradient
vector have a length of K.
'r': convert the image to grayscale and rotate it.

input key to Process image(press 'q' to quit):
```

5. References

https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_tutorials.html

https://en.wikipedia.org/wiki/Sobel_operator

https://en.wikipedia.org/wiki/Gaussian_filter

https://en.wikipedia.org/wiki/Grayscale