CS512 Assignment 2
Report

Li Xu A20300818

1. Problem Statement

This assignment is to use basic OpenCV functions coded in python to implement image processing operations.

2. Proposed Solution and Implementation Details

To run my program, simply use command python as2.py lenna.jpg, you can change the input file if you want.

In this section, I list all functions in order to show my solution:

- Function 'i' - reload the original image

  Use OpenCV imread function

- Function 'w' - save the current image into the file out.jpg

  Use OpenCV imwrite function
- Function 'g' - convert the image to grayscale using OpenCV conversion function

  Use OpenCV cvtColor function
- Function 'G' - convert the image to grayscale using your implementation of conversion function

  I used the luminosity method to covert the image to grayscale, the formulation of luminosity method is 0.21 R + 0.72 G + 0.07 B
- Function 'c' - cycle through the color counts of the image showing a different count every time the key is pressed

  I set other two color channels to zero to get each single RGB color
- Function 's' - convert the image to grayscale and smooth it using the openCV function, use a track bar to control the amount of smoothing

First use cvtColor to get to grayscale and then use filter2D to smooth the image. For the kernel argument, I created a track bar using createTrackbar function and getTrackbarPos function to pass the value. (Other track bar implementation is similar)

- Function ‘S’ - convert the image to grayscale and smooth it using my own function, use a track bar to control the amount of smoothing

  According to OpenCV documentation, the filter2D function is implemented by the equation below:

$$\texttt{dst}(x,y) = \sum_{\substack{0 \le x' < \texttt{kernel.cols}, \\ 0 \le y' < \texttt{kernel.rows}}} \texttt{kernel}(x',y') * \texttt{src}(x + x' - \texttt{anchor.x}, y + y' - \texttt{anchor.y})$$

  Iterate the image then used self-defined kernel to convolve with the image.

- Function ‘d’ - downsample the image by a factor of 2 without smoothing

  Use OpenCV pyrDown function

- Function ‘D’ - downsample the image by a factor of 2 with smoothing

  First smooth the image with filter2D then use pyrDown to downsample

- Function ‘x’ - convert the image to grayscale and perform convolution with an x derivative filter

  First use cvtColor to covert to grayscale, then iterate the image to compute the derivation of x by

  derivative of x = delta(x+1,y) - delta(x,y)

- Function ‘y’ - convert the image to grayscale and perform convolution with an y derivative filter

  Similarly with function x, to compute derivative of y by

  derivative of y = delta(x,y+1) - delta(x,y)

- Function ‘m’ - show the magnitude of the gradient normalized to the range [0,255]

  The easiest way is to use the result of function x and y, then compute the magnitude by

  Mag = sqrt(dx^2 + dy^2)

  I used the sobel function to compute derivative of x and y then compute the magnitude.

- Function ‘p’ - convert the image to grayscale and plot the gradient vectors of the image every N pixels and let the plotted gradient vectors have a length of K

  Use function x and y to compute the derivative to find the gradient vector, then use OpenCV line function to draw lines. User can define the density of the line (every N pixels) and the length of lines by track bar
- Function ‘r’ - convert the image to grayscale and rotate it using an angle of Q degrees. Use a track bar to control the rotation angle

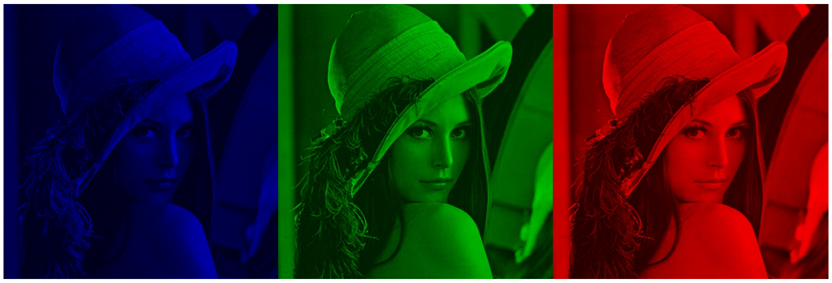
  Use OpenCV getRotationMatrix2D function to rotate the image

3. Results and Discussion

- Function ‘g’ and ‘G’

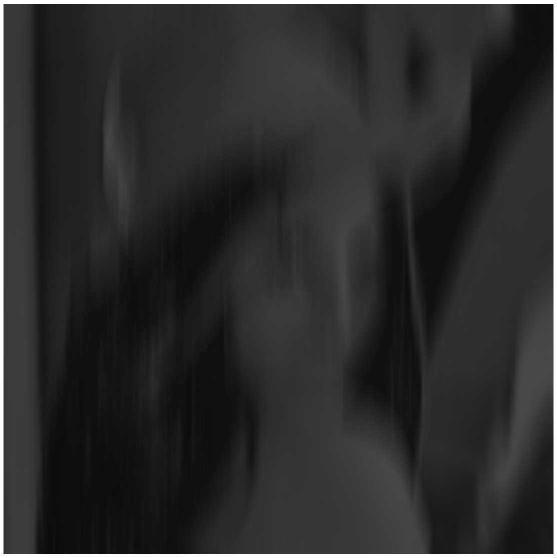
My implementation is slower than the OpenCV build-in cvtColor function.

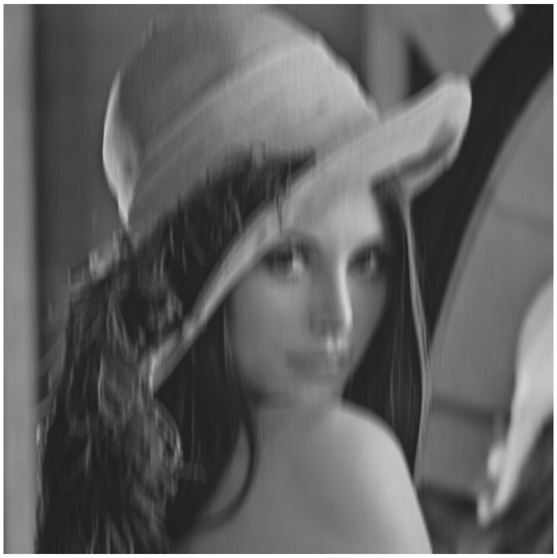
- Function ‘c’



Firstly I failed to changed image color every time c button is pressed, it turned out I forgot to using the global variable to store original image.

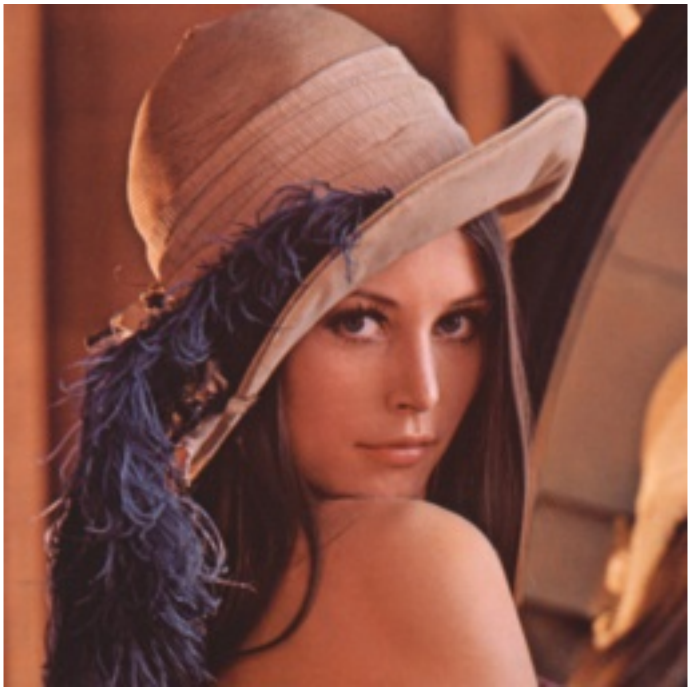
- Function ‘s’ and ‘S’ (function s used 10X10 filter and function S used 5X5 filter)

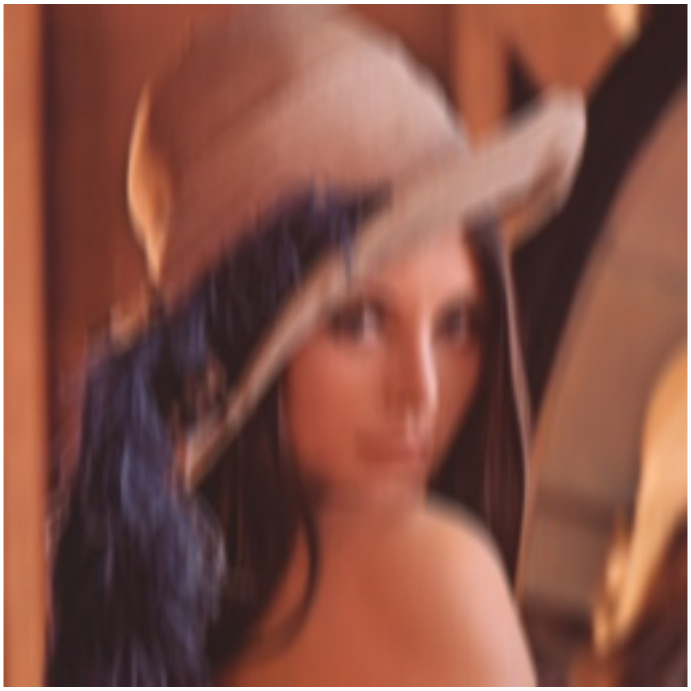




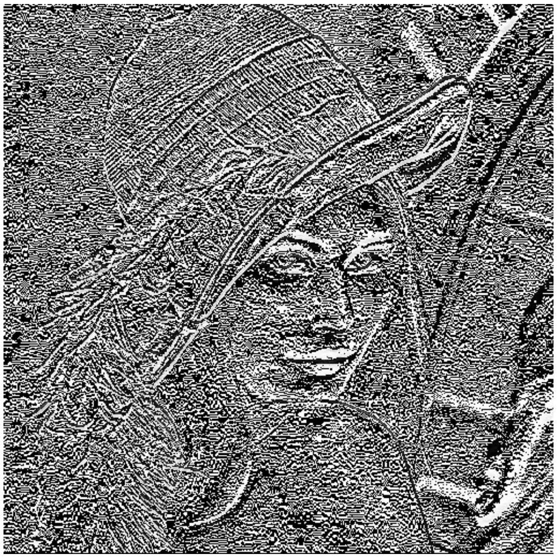
We can tell the more smoothing amount, the more blur the image is

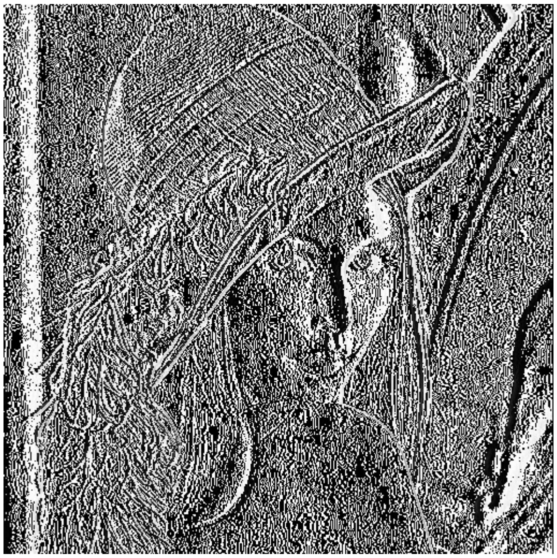
- Function ‘d’ and ‘D’ (original size, function D smooth with 5X5 filter)



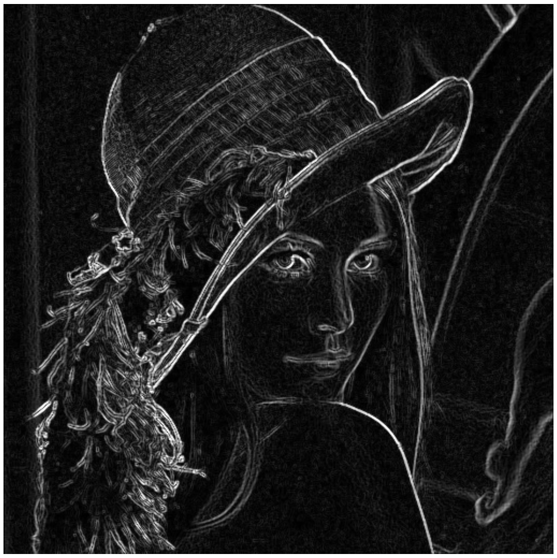


- Function ‘x’ and ‘y’



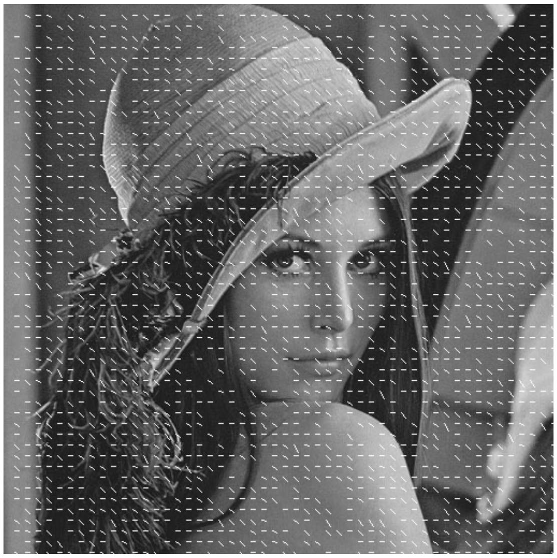


- Function ‘m’



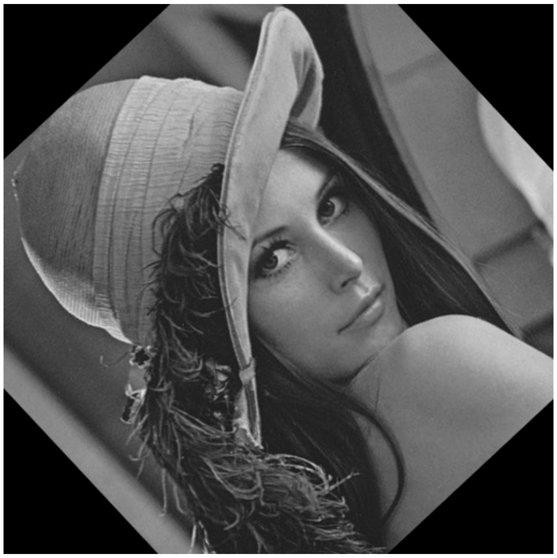
Compared to the output image of function x and y, we can tell the sobel filter can only get the approximate derivative. Since function x and y is computed by the definition of derivative.

- Function ‘p’ (every 10 pixels and the length of the line is 5 pixels)



Spent a lot of time to deal with boundary and corner condition of iterating the image.

- Function ‘r’ (rotation angle is about 45 degree)



4. References

- Using OpenCV with Python http://www.cs.iit.edu/~agam/cs512/share/using-opencv.pdf
- Three algorithms for converting color to grayscale https://www.johndcook.com/blog/2009/08/24/algorithms-convert-color-grayscale/
- Convolutions with OpenCV and Python https://www.pyimagesearch.com/2016/07/25/convolutions-with-opencv-and-python/
- Image Filtering https://docs.opencv.org/2.4/modules/imgproc/doc/filtering.html#image-filtering
- OpenCV2-Python https://github.com/abidrahmank/OpenCV2-Python
- Trackbar as the Color Palette http://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_gui/py_trackbar/py_trackbar.html#trackbar
- Drawing Functions in OpenCV https://docs.opencv.org/3.1.0/dc/da5/tutorial_py_drawing_functions.html
- Sobel Derivatives https://docs.opencv.org/2.4/doc/tutorials/imgproc/imgtrans/sobel_derivatives/sobel_derivatives.html