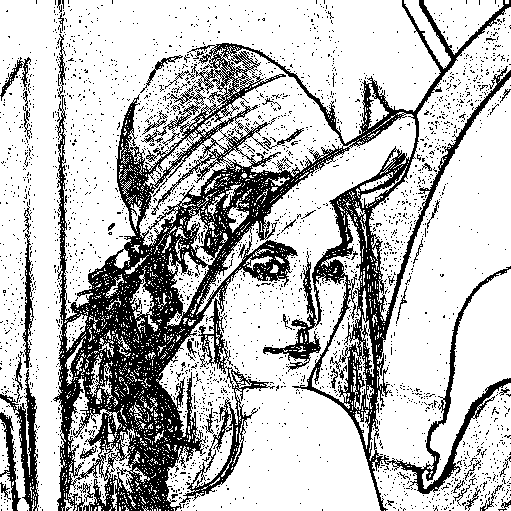
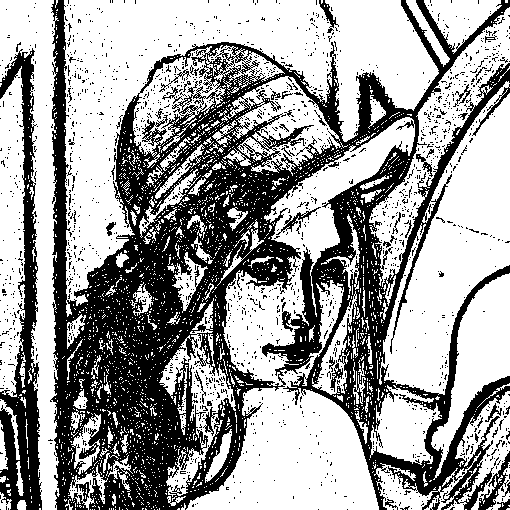
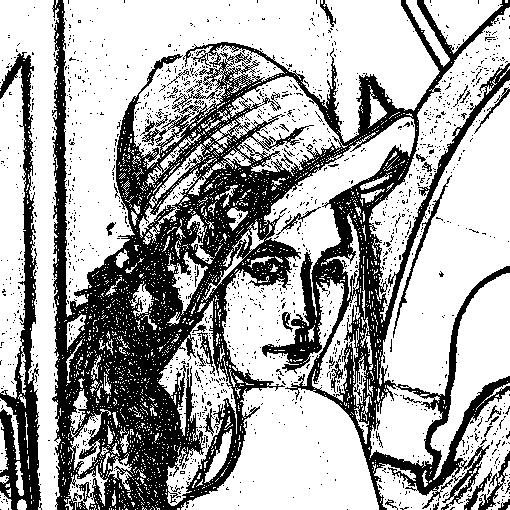
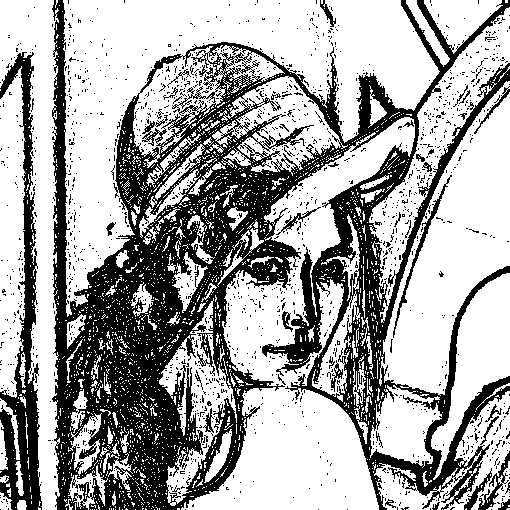
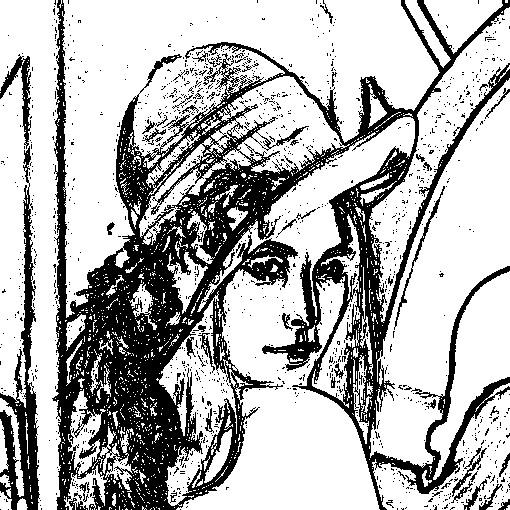
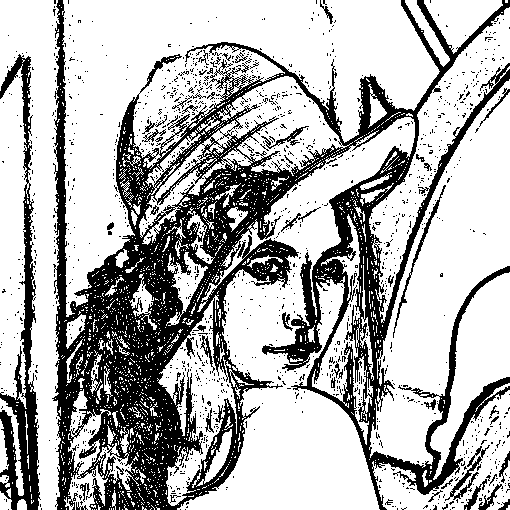
# Computer Vision HW 9: Report

The resulting image are as below:

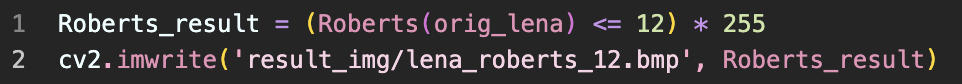
[[1]](#footnote-1) [[2]](#footnote-2)[[3]](#footnote-3) [[4]](#footnote-4)[[5]](#footnote-5) [[6]](#footnote-6)[[7]](#footnote-7)

## Code Explanation:

In the source file, it consists of:

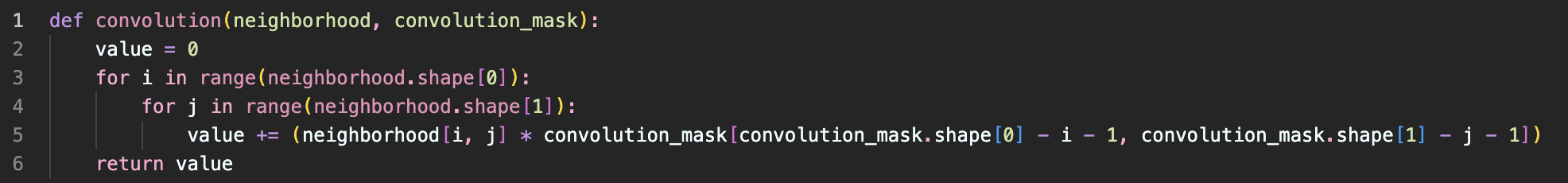
* a convolution function
* functions for each operator / edge detector (correspond to (a)~(g))

Each operator / edge detector function is followed by a code block of this form:

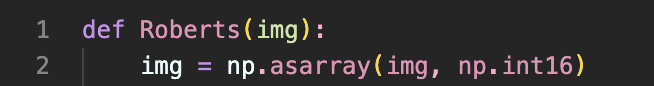


Which executes the function (Roberts() in this example), and performs the required thresholding, then multiply by 255, and saving the result using cv2.imwrite() as usual.

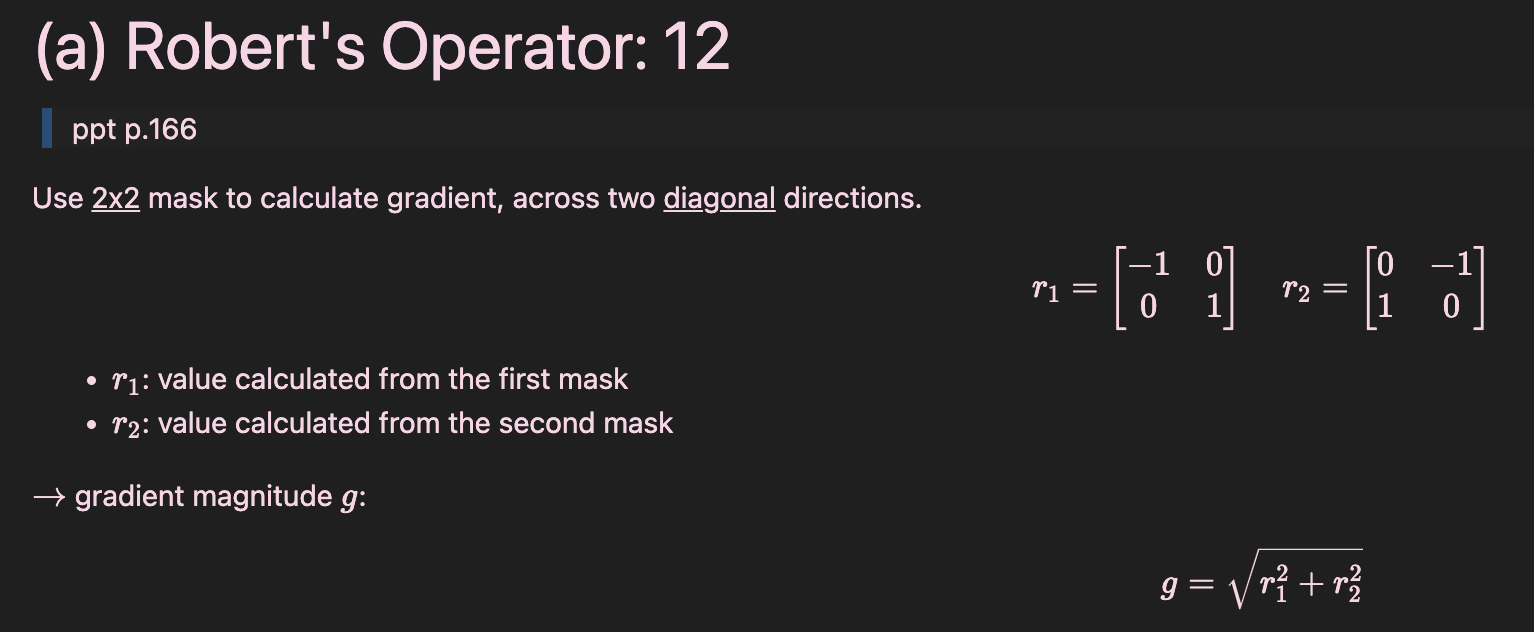
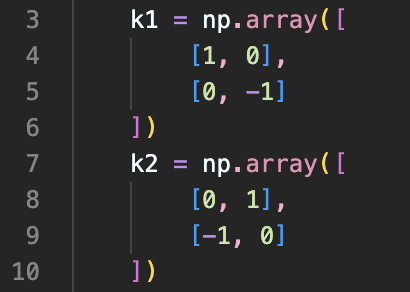
The convolution function is called in each of the functions (a)~(g) (we would explain later):



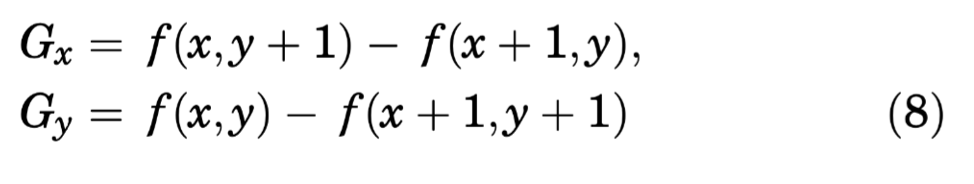
For the functions corresponding to (a)~(g), they actually have similar structure. Take Robert’s for example, the first step is to convert our image (the original lena.bmp) into a ndarray:

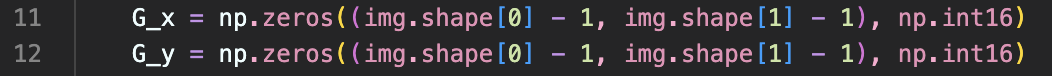


Then we define the masks as in the textbook, I wrote brief descriptions in the corresponding markdown cell:

To calculate the as defined in the ppt, we first initialize an empty ndarray for each :





If we look at the dimension settings, we can see that both the height and width are subtracted by 1 since our masks are 2x2. If our mask is 3x3, then we subtract both dimensions by 2 like below:



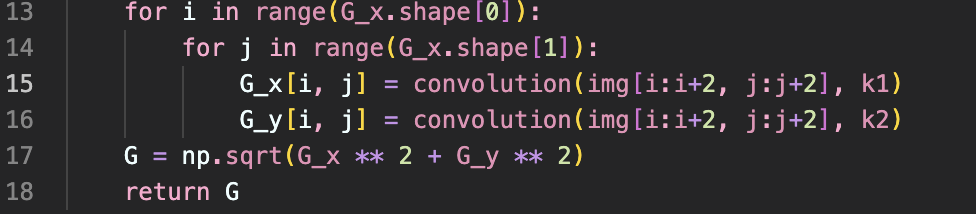
Also, for (a)~(d), we define , but for some edge detectors, we calculate the result by choosing the maximum, under such cases, we only define one (problem (e)~(g)).



The last thing to mention that we need to set np.int16 instead of np.int8 (which will cause wrong results for some problems.)

Going back to our example Roberts function, we loop through each pixel (as the same size of G\_x). For each pixel in the resulting array, in order to calculate in the gradient magnitude formula, we calculate by calling the convolution function respectively. In the convolution function, we deal with a 2x2 neighborhood each time, thus we use the index img[i:i+2, j:j+2] as the first input. The second input is the corresponding mask in order to calculate .

After these calculations are all done for the resulting picture, we then take the squares for each and the square root for the whole result to get our result G.



1. (a) Roberts, threshold = 12 [↑](#footnote-ref-1)
2. (b) Prewitt, threshold = 24 [↑](#footnote-ref-2)
3. (c) Sobel, threshold = 38 [↑](#footnote-ref-3)
4. (d) Frei and Chen, threshold = 30 [↑](#footnote-ref-4)
5. (e) Kirsch, threshold = 135 [↑](#footnote-ref-5)
6. (f) Robinson, threshold = 43 [↑](#footnote-ref-6)
7. (g) Nevatia Babu, threshold = 12500 [↑](#footnote-ref-7)