NYUAD Engineering Computer Vision **Fall 2019**
Project 1: Canny's Edge Detector **Yi Fang**
Due: 11:50PM, Sept. 22nd, 2019

Total number of points = 100. This is a one-person project. No teams allowed.

Project Description: Implement the *Canny's Edge Detector* as described in the lecture notes and in the book. The Canny's Edge Detector consists of four steps: *Gaussian smoothing, gradient operation, non-maxima suppression* and *thresholding.* The input to your program is a grayscale image of size *N* X *M*. Use the 7 x 7 Gaussian mask as shown below (on page 2) for smoothing the input image. Use the center of the mask as the reference center. If part of the Gaussian mask goes outside of the image border, let the output image be undefined at the reference center's location. Note that the sum of the entries in the Gaussian mask do not sum to 1. After performing convolution (or cross-correlation), you need to perform normalization by dividing the result by the sum of the entries (= 140 for the given mask) at each pixel location. Instead of using the Robert's operator, use the Prewitt's operator for gradient operation. If part of the 3 x 3 masks of the Prewitt's operator lies in the undefined region of the image after Gaussian filtering, set the output value to zero (indicates no edge). Use simple thresholding in the fourth step but use the *P-tile* method to set the threshold *T* (described below.)

Design your program in a modular fashion and create separate program functions for the four steps: *Gaussian smoothing, gradient operator, non-maxima suppression* and *thresholding.* Generate the following *normalized* image results with an output range of [0,255]: (1) Normalized image result after Gaussian smoothing. (2) Normalized horizontal and vertical gradient responses (two separate images.) To generate normalized gradient responses, take the absolute value of the results first and then normalize. (3) Normalized edge magnitude image. (4) Normalized edge magnitude image after non-maxima suppression. Lastly, (5) generate a set of binary edge images by thresholding the normalized edge magnitude image after non-maxima suppression (from (4)) using the *P-tile* method for *P = 10%, 30%* and *50%*; i.e., produce three binary edge images for each of *P = 10%, 30%* and *50%.* For example, for *P = 10%, 10%* of the pixels should have normalized magnitude value $\geq T$, where *T* is the corresponding threshold value. Use gray level value 255 for edges (foreground) and 0 for background in your binary edge images. For each *P,* output the corresponding threshold value *T* and the total number of edges detected in the edge image (after thresholding.)

You can use Python, C++/C, Java or Matlab to implement your program. If you plan on using another language, send me an email first. You are not allowed to use any built-in library functions that the language already has for any of the steps that you are required to implement, including the convolution (or cross-correlation) operation. The only library functions you are allowed to use are: *reading* and *writing* an image file, and *matrix* and *vector multiplications*.

Testing your program: Two grayscale test images of size *N* X *M* in bitmap (*.bmp*) format will be provided on NYU Classes for you to test your program. A 15-min time slot will be set up during class time for you to run your program on a test image that will be provided in class. (Details to be announced later.)

Hand in:

(a) A text file that contains the source code, which can be used to compile your program. Put comments in your source code to make it easier for someone else to read your program. Points will be taken off if you do not have comments in your source code.
(b) Executable code of your program.
(c) Output image results as described in (1) to (5) above for the two test images provided on NYU Classes.
(d) An MS Words or PDF file that contains: (i) File names of your source code and executable code. (ii) Instructions on how to compile and run your program. (iii) Output image results as described in (1) to (4) above for each test image provided on NYU Classes. (iv) The binary edge image produced, the threshold value *T* and the total number of edges detected for each of *P = 10%, 30%*

and *50%*. (v) The source code of your program. You can just copy-and-paste over the source code and image results from (a) and (c) above onto a Words file.

7×7 Gaussian mask

1	1	2	2	2	1	1
1	2	2	4	2	2	1
2	2	4	8	4	2	2
2	4	8	16	8	4	2
2	2	4	8	4	2	2
1	2	2	4	2	2	1
1	1	2	2	2	1	1

Note that you need to hand in the source code and image results as separate files and also in the Words or PDF report. Submit files for (a) to (d) above on NYU Classes by the Sept. 22nd, 11:50PM, Abu Dhabi Time.