CS 435 – Computational Photography
Assignment 2

In this assignment you will demonstrate your ability to implement the individual components of a Canny Edge Detector

Grading Scheme

1. Theory Questions (20pts)
2. Gaussian Smoothing (20pts)
3. Gradients (20pts)
4. Threshold (10pts)
5. Hysteresis (20pts)
6. Applied Pipeline to Paper Photo (10pts)

Theory Question(s)

1. (5pts) Apply a 3×3 mean filter to the following 2D matrix. You may assume that the filter is only applied to areas of the data that have a full 9 samples to process. *Feel free to use Matlab to help you compute this, however, realize that you may be asked to do this without a calculator on an exam.*

7	7	6	3	3	4	2	2
3	7	2	6	4	4	5	7
5	4	7	5	1	1	2	2
2	1	3	4	1	3	5	6
6	2	2	7	4	2	5	4
2	2	2	3	6	6	6	7
4	6	5	6	7	3	4	1
5	2	4	6	1	4	1	4

2. (5pts) What is the kernel function for a 5×5 Gaussian function with $\sigma = 1$? Show the filter as-is, and then discretize it so that all values are integers, that the minimum value is 1, then *normalize* it so that all its elements sum to 1. *Feel free to use Matlab to help you compute this, however, realize that you may be asked to do this without a calculator on an exam.*

3. (5pts) What is the derivative of a 3×3 mean kernel with respect to x and y

4. (5pts) Given the following 2D kernels, what is the magnitude and direction of the gradient at the center pixel in I? *Feel free to use Matlab to help you compute this, however, realize that you may be asked to do this without a calculator on an exam.*

$$\frac{\partial}{\partial x} = \begin{bmatrix} -\frac{1}{3} & 0 & \frac{1}{3} \\ -\frac{1}{3} & 0 & \frac{1}{3} \\ -\frac{1}{3} & 0 & -\frac{1}{3} \end{bmatrix}, \frac{\partial}{\partial y} = \begin{bmatrix} -\frac{1}{3} & -\frac{1}{3} & -\frac{1}{3} \\ 0 & 0 & 0 \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \end{bmatrix}$$

$$I = \begin{bmatrix} 7 & 7 & 6 \\ 3 & 7 & 2 \\ 5 & 4 & 7 \end{bmatrix}$$

Programming Introduction

For this assignment we are going to implement the various stages of a Canny Edge Detector and apply them to a few images, observing the results along the way.

I have provided a single sample image, *circles1.gif*, for us to observe the effect of stages of the edge detector. In the end you will also be asked to take a photo of a piece of paper, and to apply your edge detector to it (we'll be using this image for a later program!).

Part 1: Gaussian Smoothing

The first step in the Canny Edge Detector is to apply a Gaussian smoothing kernel to your image. Our edge detection will be done in grayscale, so first convert your image to grayscale, if necessary.

Next, given an odd filter size, N and a Gaussian variance parameter, σ, compute the $N \times N$ Gaussian smoothing kernel and apply it to your image to generate a smoothed new image. Just apply the kernel to the "inside" of the image, that is, areas where there is a large enough neighborhood to apply your kernel.

Show the original grayscale image and then the results for at least 4 different combinations of (N, σ)

*Note: For this point you **may not** use Matlab's `conv2` function. I would like you to at least once implement convolution yourself.*

Smoothed image with $N = 5, \sigma = 3$

Part 2: Gradients

Next, we'll compute the gradients on our **original** image.

Generate three images

1. One which has the intensity with respect to the change in x
2. One that has the intensity with respect to the change in y
3. One that has the overall magnitude of the combined gradients.

Since the `imshow` function expects integers in the range of [0,255], you should first cast your image to `uint8` prior to displaying it.

Doing this on your original image you'll likely see "noise", so next try first applying a smoothing filter (like you developed in the previous part) to remove noise prior to extracting gradients. Show these images as well.

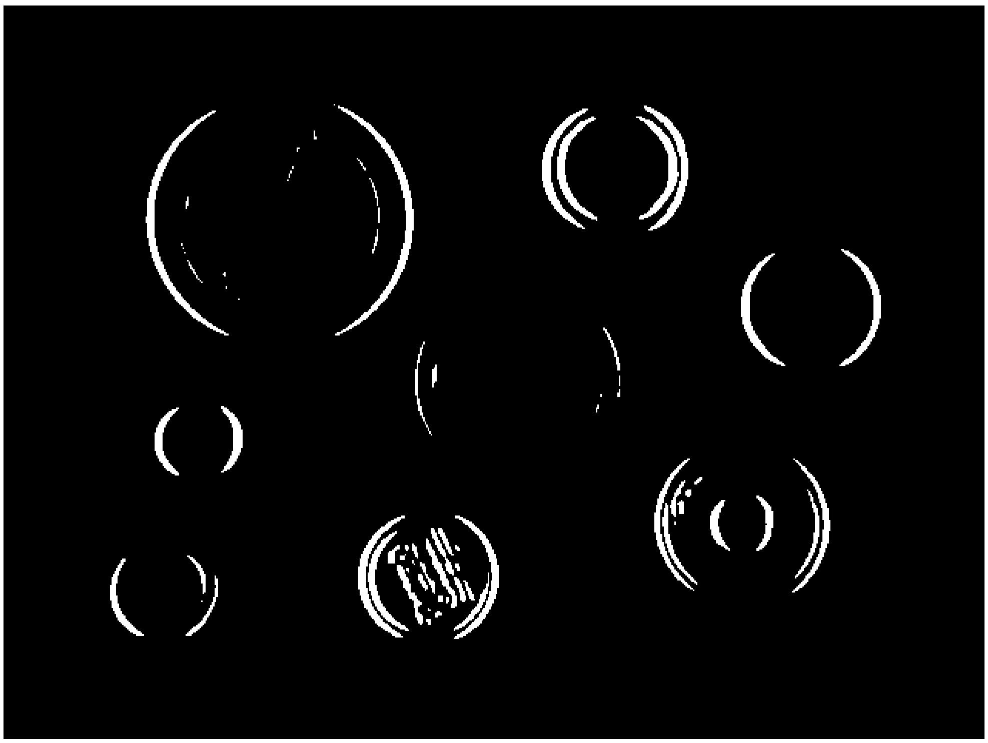
*Note: Since you already demonstrated in the previous part your ability to perform convolution, for the remaining parts of the assignment you **may** use Matlab's `conv2` function.*

Magnitude of gradient on smoothed image.

Part 3: Threshold

On your gradient magnitude image, apply a threshold to obtain binary images. Experiment with a few different thresholds.

Show your output for a few different choices of your threshold.



Binary threshold edges

Part 4: Hysteresis

Now allow a user to specify a low and high threshold such that a pixel is an edge pixel if its gradient is greater than the high threshold, **or** if it is greater than the low threshold **and** borders (8-way) a pixel that is above the high threshold.

Part 5: Test on Another Image

Now that you have all the stages of your Canny Edge Detector implemented, take a photo of a piece of paper on a dark background and apply your Canny Edge Detector to it. Show the result as it goes through each part of the pipeline.

Submission

1. Assignments must be submitted via Bd Learn
2. Submit a single compressed file (zip, tar, etc..) containing:
    a. A PDF file containing:
        i. Your answer to the theory question(s).
        ii. Your original grayscale image and at least four smoothed images for Part 1
        iii. Your six (2*three) images demonstrating the gradients for Part 2. In addition, let us know what smoothing parameters you used to generate your second set of gradient images.
        iv. At least two threshed binary images along with their thresholds, for Part 3
        v. Your binary image for Part 4
        vi. Your original photo, and images demonstrating the edge detector pipeline for Part 5
    b. A README text file (**not** Word or PDF) that explains
        i. Features of your program
        ii. Name of your entry-point script
        iii. Any additional information necessary for us to run your program(s).
    c. Your source files
    d. The chosen image that you are processing.