Computational Photography

Assignment 2 - Canny Edge Detection
Summer 2023

Introduction

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

In this assignment, as with all of our assignments, you shouldn't be using built-in functions that violate the "spirit" of the assignment. For instance, in part 2, you can't use a function like *conv2* or *imgaussfilt* nor an *edge* function anywhere. However, since you implemented rgb2gray in HW1, you **MAY** use Matlab's rgb2gray function throughout this assignment. In general, use your intuition, and when in doubt ask the instructor or TA.

In this assignment you will demonstrate your ability to:

- Create and apply smoothing and gradient kernels.
- Find edge candidates by thresholding based on gradient strength.
- Apply non-maximum suppression.
- Apply hysteresis.

Grading

Theory Questions	20pts
Gaussian Smoothing	20pts
Computing Gradients	20pts
Non-maximum supression	10pts
Hysterisis	20pts
Apply pipeline to addition image	10pts
TOTAL	100pts

Table 1: Grading Rubric

Dataset

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 addition, use an image of your choosing to apply the Canny Edge Detector pipeline to as well to verify its robustness.

1 (26pts) Theory Questions

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. **Leave your answers as floating point values, with a single digit after the decimal point**.

$$I = \begin{bmatrix} 7 & 7 & 6 & 3 & 3 & 4 & 2 & 2 \\ 3 & 7 & 2 & 6 & 4 & 4 & 5 & 7 \\ 5 & 4 & 7 & 5 & 1 & 1 & 2 & 2 \\ 2 & 1 & 3 & 4 & 1 & 3 & 5 & 6 \\ 6 & 2 & 2 & 7 & 4 & 2 & 5 & 4 \\ 2 & 2 & 2 & 3 & 6 & 6 & 6 & 7 \\ 4 & 6 & 5 & 6 & 7 & 3 & 4 & 1 \\ 5 & 2 & 4 & 6 & 1 & 4 & 1 & 4 \end{bmatrix}$$

Answer: Found through Addition of all 9 available components and dividing by 9.

$$I(Filtered) = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 5.3 & 5.2 & 4.1 & 3.4 & 2.9 & 3.2 & 0 \\ 0 & 3.8 & 4.3 & 3.7 & 3.2 & 2.9 & 3.9 & 0 \\ 0 & 3.6 & 3.9 & 3.8 & 3.1 & 2.7 & 3.3 & 0 \\ 0 & 2.4 & 2.9 & 3.6 & 4.0 & 4.2 & 4.9 & 0 \\ 0 & 3.4 & 3.9 & 4.7 & 4.8 & 4.8 & 4.2 & 0 \\ 0 & 3.6 & 4.0 & 4.4 & 4.7 & 4.2 & 4.0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

2. (5pts) What is the kernel function for a 5×5 Gaussian function with $\sigma = 2$? Normalize the kernel so that its elements sum to one.
Answer:

$$Kernel = \begin{bmatrix} 0.0232 & 0.0338 & 0.0383 & 0.0338 & 0.0232 \\ 0.0338 & 0.0492 & 0.0558 & 0.0492 & 0.0338 \\ 0.0383 & 0.0558 & 0.0632 & 0.0558 & 0.0383 \\ 0.0338 & 0.0492 & 0.0558 & 0.0492 & 0.0338 \\ 0.0232 & 0.0338 & 0.0383 & 0.0338 & 0.0232 \end{bmatrix}$$

3. (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.

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

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

Magnitude= 7.07
Direction= 98.13°

4. Imagine that the matrix below contains is for the magnitude of the *gradients* of an image. Which pixels would we consider edge pixels if we applied *hysteresis* with a low threshold of $T_L = 2$ and a high threshold of $T_H = 4$? Visualize this as a *binary* image/matrix such that a location has a value of one if it is an edge pixel. Do this using:

   (a) 4-way connectivity (2pts)

   (b) 8-way connectivity (3pts)

$$|G| = \begin{bmatrix} 2 & 3 & 4 & 5 & 1 \\ 1 & 0 & 2 & 2 & 1 \\ 4 & 3 & 5 & 1 & 2 \\ 4 & 4 & 4 & 4 & 6 \\ 4 & 5 & 2 & 0 & 2 \\ 2 & 3 & 3 & 0 & 3 \end{bmatrix}$$

2 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 the *circles.gif* images to grayscale, if necessary.

Next, given an odd filter size, K and a Gaussian variance parameter, σ, compute the $K \times K$ Gaussian smoothing kernel and apply it to the 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 (K, σ)

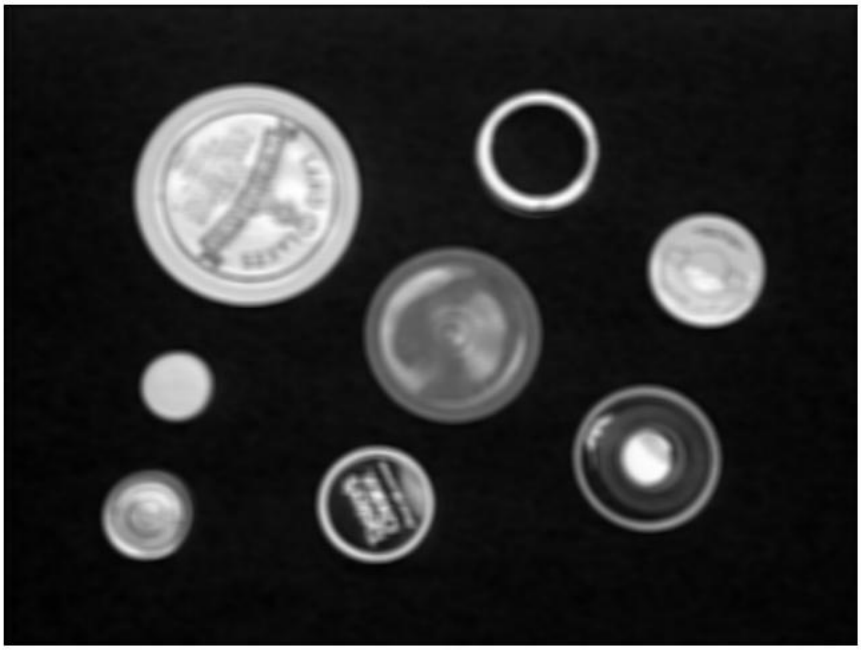
Note: For this part you **may not** use Matlab's *conv2* function for convolution, nor can you use Matlab's *imgaussfilt* (or similar) function for computing the Gaussian kernel. We would like you to implement these at least once yourself.

Original Image is shown above

K=5, σ = 2

K=3, σ = 3

K=7, σ = 4



K=9, σ = 1.5

3 Gradients

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

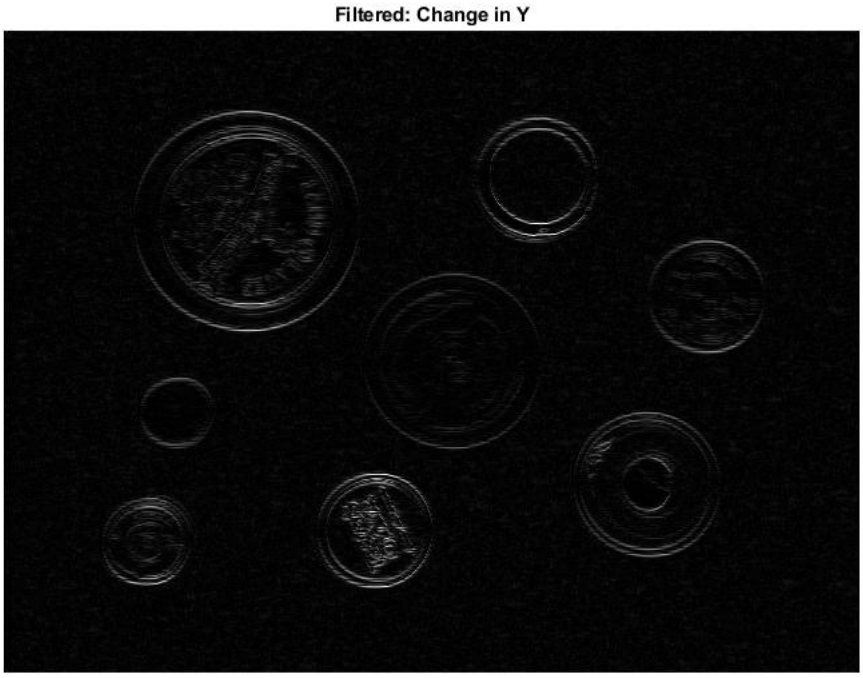
Generate three images using the original image (not smoothed):

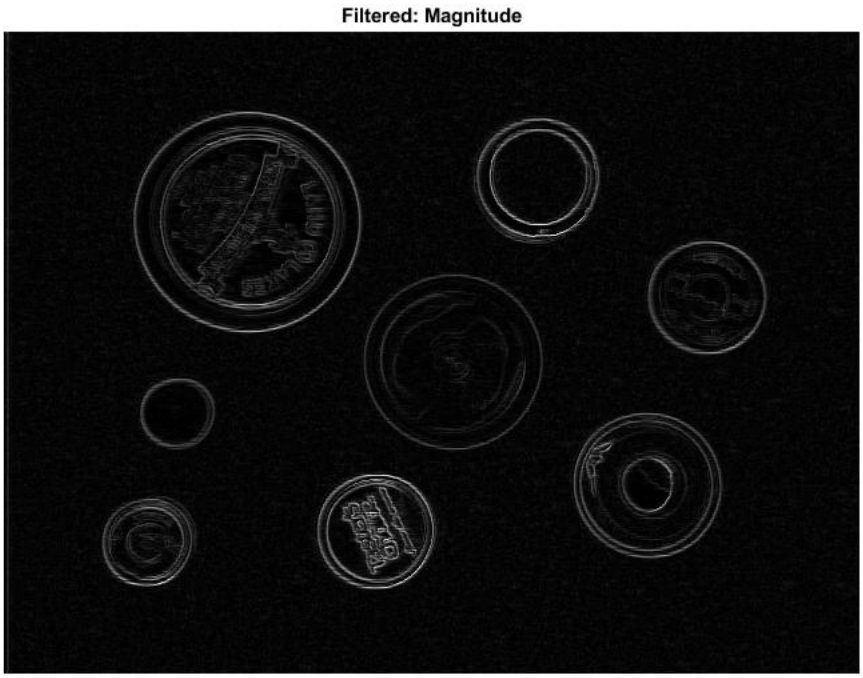
1. One which has the absolute value of the change with respect to the change in x
2. One that has the absolute value of the change with respect to the change in y
3. One that has the overall magnitude of the combined gradients.

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. You can choose the parameters of the kernel as you see fit. Show these images as well (6 total images).

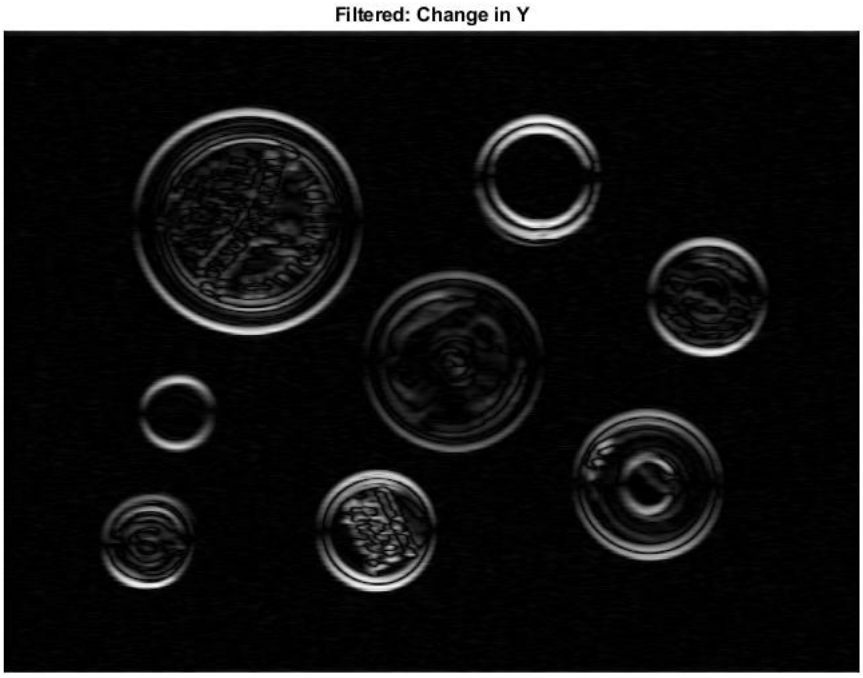
Note: Since you already demonstrated in the previous part your ability to perform convolution, for the remaining parts of the assignment you **may** use Matlab's *conv2* and/or *imgaussfilt* function.

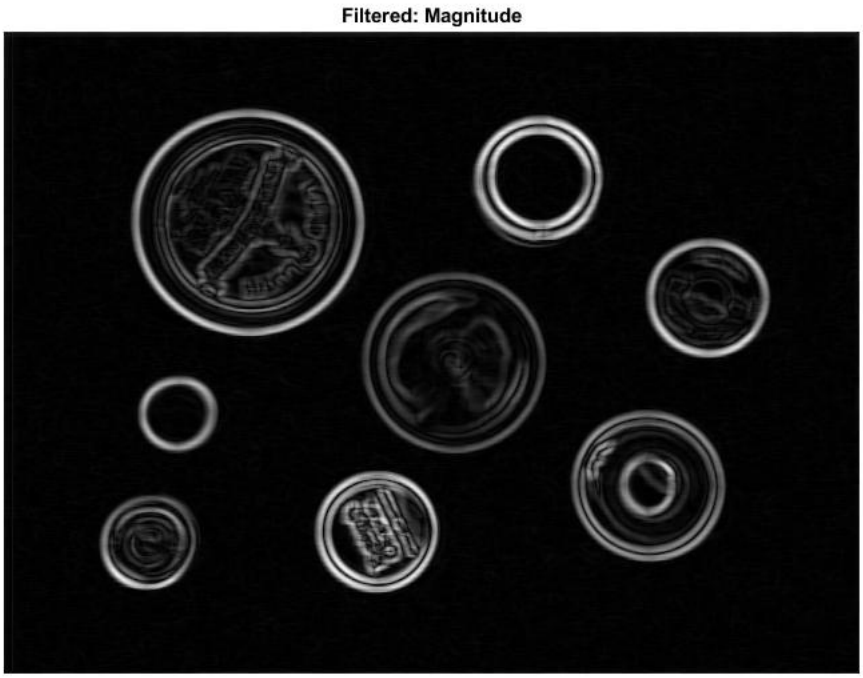
3.1 Pre-Smoothened




Filtered: Magnitude

3.2 Smoothened with $K = 7$ & $\sigma = 3$




Filtered: Magnitude

4 Non-maximum suppression

Our next step is to apply *non-maximum supression* to the gradient information.

Using the gradient information, compute the magnitude and angle of the gradient for each pixel. Then, for all pixels, compare the magnitude of the gradient of this pixel to two nearest neighbors, in the direction to and opposite the gradient direction, as defined in the table below. If the magnitude of its gradient is the largest of those neighbors, flag it to continue through the pipeline.

A few notes about doing this in Matlab:

- Use the *atan*2 function to get the angles. This returns the *four-quadrant atan* with values in the range of $[-\pi, \pi]$. See the Matlab documentation for more details.

Pixels to check	**Angle**
Left and right	$\theta < -\frac{7\pi}{8}$ or $\theta \geq \frac{7\pi}{8}$ or $-\frac{\pi}{8} \leq \theta < \frac{\pi}{8}$
Up and down	$-\frac{5\pi}{8} \leq \theta < -\frac{3\pi}{8}$ or $\frac{3\pi}{8} \leq \theta < \frac{5\pi}{8}$
Up-right and down-left*	$-\frac{3\pi}{8} \leq \theta < -\frac{\pi}{8}$ or $\frac{5\pi}{8} \leq \theta < \frac{7\pi}{8}$
Up-left and down-right*	$-\frac{7\pi}{8} \leq \theta < -\frac{5\pi}{8}$ or $\frac{1\pi}{8} \leq \theta < \frac{3\pi}{8}$

* These may be counter-intuitive. But since the y-axis is inverted for the image coordinate plane, the angles are reflected about the x-axis.

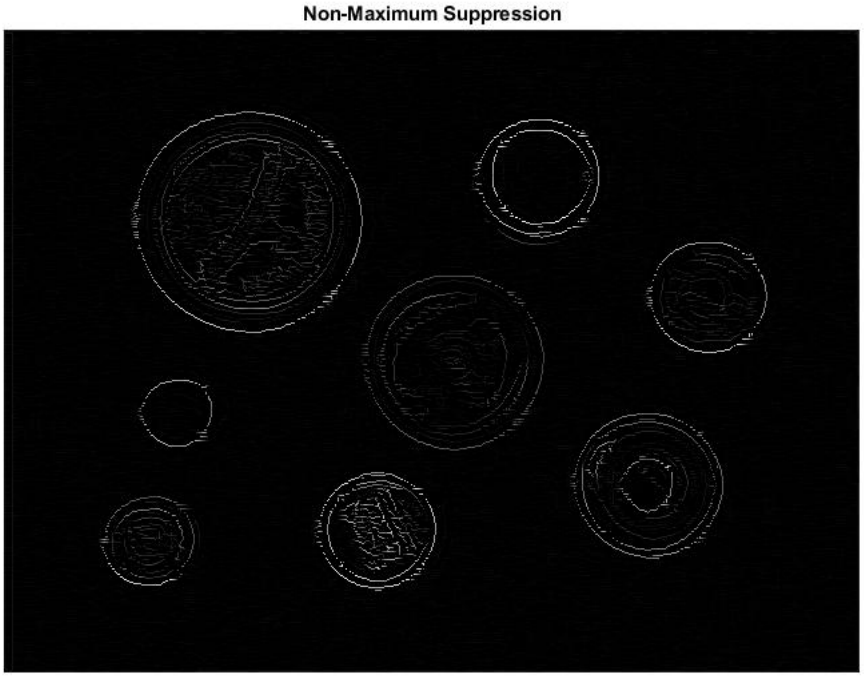


Figure 1: Non-Maximum Suppression.

5 Hysterisis

For our final step, set 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. The result should be a *binary* image. Here you just need to provide one example output, but also report the parameter choices (smoothing kernel and thresholds).

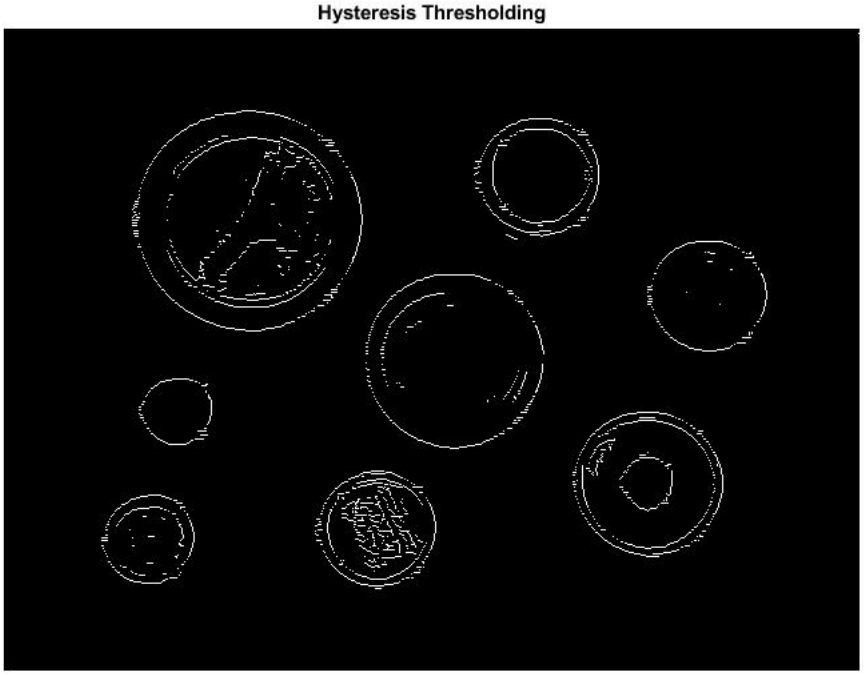


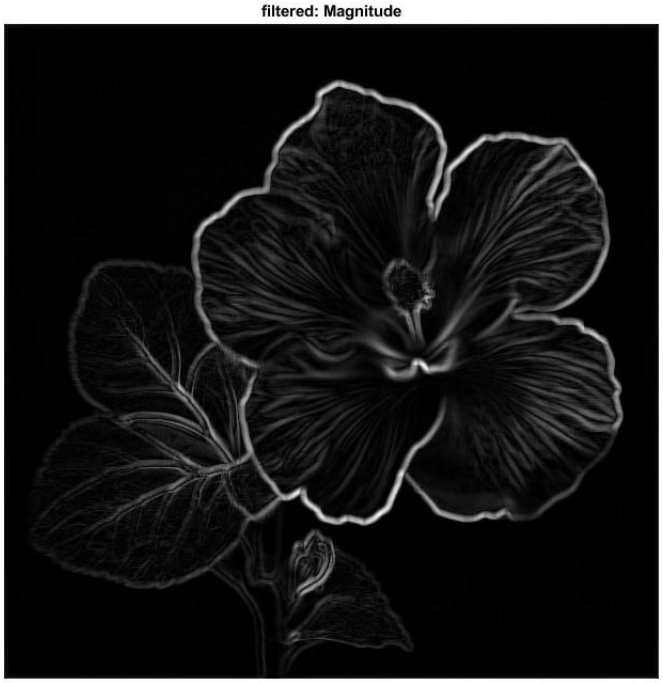
Figure 2: Hysteresis with threshold 0.2 to 0.5 of maximum

6 Test on another image

Now that you have all the stages of your Canny Edge Detector implemented, somehow obtain another image and apply your Canny Edge Detector to it. Show the result as it goes through each part of the pipeline.

Figure 3: Original Flower Image

filtered: Magnitude



Non-Maximum Suppression

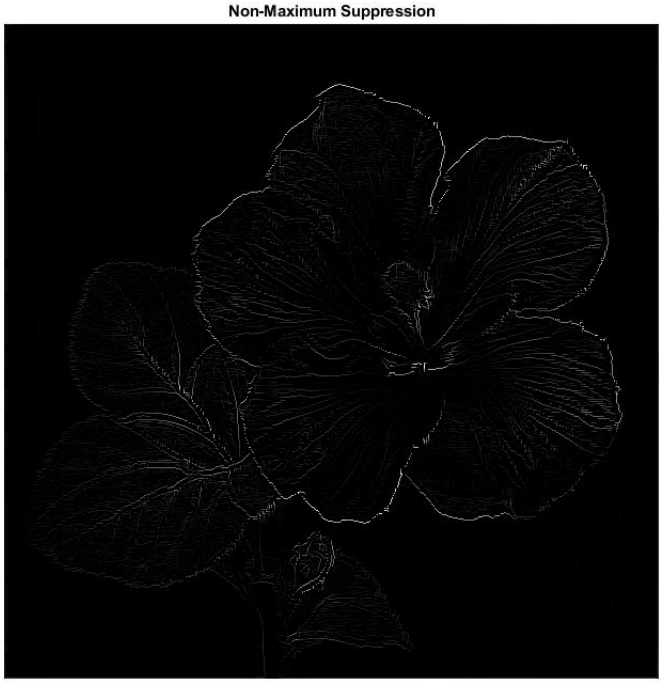


Figure 4: Hysterisis with Thresholds 0.05 to 0.1 of maximum

Submission

For your submission, upload to Blackboard a single zip file containing:

1. PDF writeup that includes:
   (a) Your answer to the theory question(s).
   (b) For Part 2, 5 images. The original grayscale, then 4 combinations of the parameters (N, σ).
   (c) For Part 3, 6 total images. Three showing the partial gradients and magnitude of the gradient without pre-smoothing, and three after. In addition, report the parameters of the kernel you used for smoothing.
   (d) For Part 4, your non-maximum suppressed image.
   (e) For Part 5, your binary edge image usng hysteresis. In your report provide the parameters of the smoothing kernel and the low and high threshold values for the hysterisis process.
   (f) For Part 6, 5 total image:
      i. Original image (either color or gray)
      ii. Smoothed image (along with choice of smoothing filter parameters).
      iii. Gradient magnititude image
      iv. Hysterisis edge image (along with choice of thresholds)
      v. Final edge image with non-maximum suppression applied.
2. A README text file (**not** Word or PDF) that explains:
   (a) Any unique features of your program (if applicable).
   (b) Any instructions on how to run your script to reproduce your results.
3. Your source file(s).
4. The chosen image(s) that you processed.