

**ECEN 498**

**Fundamentals of Computer Vision**  
**Prof. Benjamin Riggan (Instructor)**

**Python 1**

*Python 1 - Canny Edge Detection*

*Written by*  
**David Perez**

**Department of Electrical and  
Computer Engineering (ECE)  
University of Nebraska  
Lincoln/Omaha**

**Due: 03/01/2024**

# Introduction

The purpose of this assignment is to implement the Canny Edge detection process in python. The Canny Edge detection process as implemented in this report consists of 4 distinct steps. The 4 steps are noise suppression, computation of gradient magnitude and direction, non-maximum suppression, and dual thresholding with hysteresis. Using this architecture a sample image can have these operations applied to it resulting in a 1 pixel wide layer determining the edges in an image.

## Technical Methodology

As stated in the introduction the first step in the Canny Edge detection process is noise reduction. Before this filter is applied it is worth noting that the input image remains in its original pixel value state where each pixel has a value of 0 to 255. This report's implementation starts by creating a gaussian kernel where sigma (the standard deviation) is initially 1 and the kernel is of a 3x3 shape. The input image is first surrounded by 1 pixel zeros ("same padding" is applied) so the image output shape is the same before running the gaussian kernel across it.

After the Gaussian kernel has been applied to the input image the *Soble* was applied to the image where the kernel also had a leading  $\frac{1}{8}$  coefficient. The correlation of this filter and the previous Gaussian filter was applied to the image by using two for loops, multiplying the overlaid kernel value with the corresponding pixel value and summing up those values to produce an output image pixel value. Since the *Soble* kernel is a derivative kernel two kernels needed to be applied (one in the x direction and one in the y direction). These output images could then be pixel-wise squared and summed together, the square root of this new image would produce the gradient magnitude image. The gradient direction image is produced by using numpy's *arctan2* function in which the gradient with the first argument being the image the gradient y image and the second being the gradient x image. This is then converted to degrees to create the gradient direction image.

With the gradient magnitude and direction of the input image non-maximum suppression can be used to limit the number of pixel edges that appear in the output image. The implementation first looks at each pixel in the gradient magnitude image. Using the gradient direction image at that pixel value, the pixel in the direction of the gradient and the pixel in the

opposite direction of the gradient are considered. The two chosen pixels are determined by breaking up the cartesian system into positive and negative degrees. Each of the 8 neighboring pixels was assigned a set of 45 degree ranges that would determine if the gradient direction is pointing at one of these pixels. Once the two pixels were located, if the gradient magnitude was greater than the magnitude of the pixel in the gradient direction and opposite the gradient direction than that pixel would be considered a local maximum, otherwise it would be assigned a value of zero. This process was repeated for every pixel in the gradient magnitude image resulting in the non-maximum suppression image.

After non-maximum suppression is applied, dual thresholding with hysteresis was applied to the resulting image. First two binary images were created by declaring pixel values to be zero or 255 based on two different thresholds. That is, if the pixel had a value greater than the high threshold it would have a value of 255 otherwise it would be assigned a value of 0. The second threshold has the same binary output but the pixel values have to be greater than the lower threshold and also lower than the higher threshold. Using these new two images the hysteresis approach can be used to combine into a more accurate final image. Hysteresis in this approach was implemented by first checking if the current weak edges (or lower threshold image) had a pixel value of 255 at that index. If there was a pixel of 255 and there was a neighboring pixel in the strong edge image in the 8 surrounding pixels then the output image would receive a pixel value of 255 at that location. If there wasn't any neighboring strong pixel surrounding that 255 weak pixel then the output image would receive a value of 0 at that location. Furthermore, if the pixel value at that weak edge location was zero, that location at the output image would receive the value of the strong edge pixel value at that location. Lastly, this process was repeated 25 times to allow for strong and weak edges to combine.



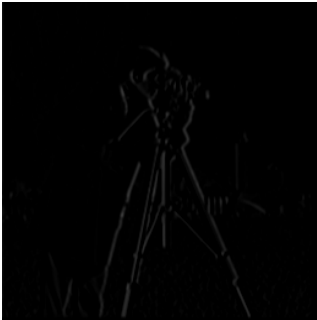

## Experimental Details

When setting up experiments for this assignment there were a few parameters that could be adjusted to elicit different responses from our Canny Edge detector. The parameters of choice are the gaussian standard deviation for noise creation as well as the thresholds for the hysteresis. In an attempt to fine tune the parameters a standard deviation of 1, 2, and 5 will be used to gauge how the detector responds to changes in noise. These values will be changed with a fixed threshold value to determine the direct impact of the noise reduction at the input. From there different sets of threshold values

will be tested on the best set of noise reductions to isolate effects of parameter changes. Furthermore, there will be three images examined to observe the detector's response to various image inputs.



## Results

Before beginning tests a basis for each phase in the Canny Edge detection will be shown to emphasize how each stage affects the input image. First the image has noise reduction with a standard deviation of 1. The thresholding for this image was 10 for the high threshold and 0.01 for the lower threshold (See **Table 1**).

			
Original Image	Noise Reduced Image	Gradient_x	Gradient_y

**Table 1:** Noise Reduction and Gradient Correlation on Input Image

With noise reduction and the x and y gradient kernels applied to the image, the magnitude and direction can be seen in **Table 2**.

 <p><b>Gradient Magnitude</b></p>	 <p><b>Gradient Direction</b></p>
--	---




**Table 2:** Gradient Magnitude and Direction

With the magnitude and direction calculated and after non-maximum suppression is applied the results are shown in **Figure 1**.






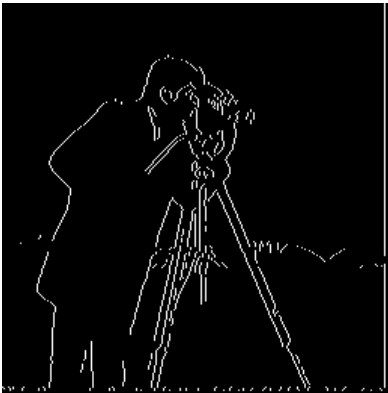

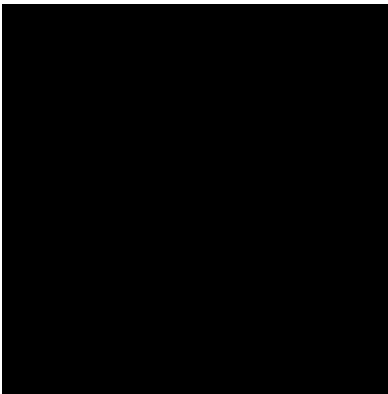
**Figure 1:** Non-Maximum Suppression

Establishing the thresholds to create two binary images to produce a final thresholded image can be seen in **Table 3**.

		
High Threshold	Low Threshold	Dual-Threshold Hysteresis

**Table 3:** Dual-Thresholding Hysteresis Stages

To emphasize the effects of the noise reduction gaussian kernel values, a sigma value (standard deviation) of values 1,2, and 5 have been experimented with. For the sake of redundancy, only the noise reduced image, and the output of the Canny Edge detector will be shown in **Table 4**.

Gaussian Standard Deviation	Noise Reduced Input Image	Canny Edge Detection Output
1		
2		
5		

**Table 4:** Various Noise Reduction Standard Deviations

Observing **Table 4** it can be seen that there is a cutoff until the noise reduction starts to impact the accuracy of the edge detector. When the noise reduction has a standard deviation of 5 the output is almost entirely blank. With a threshold of 2 the

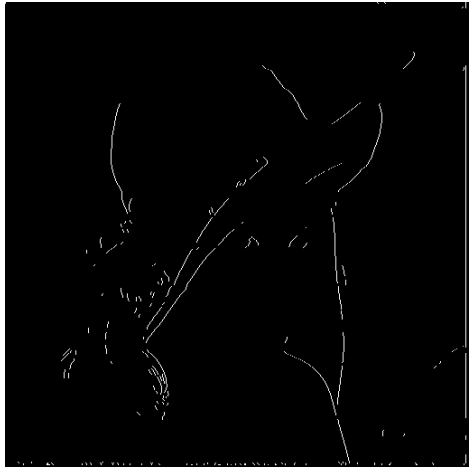
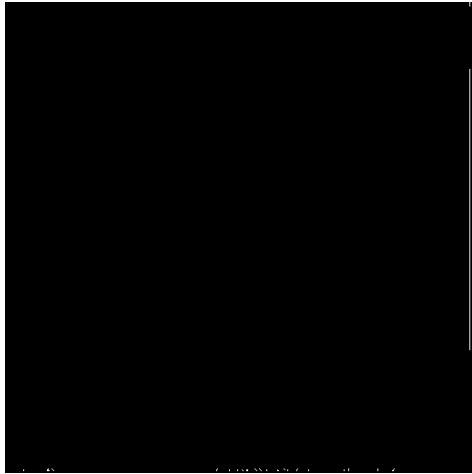
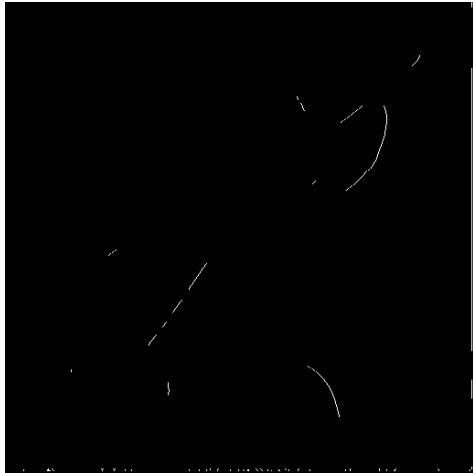
detector seems to produce more meaningful edges (i.e the focus of the picture edges are exemplified while the background is mostly ignored).

The second set of tests seeks to gain some insight into what thresholds will produce the most accurate edges. Various sets of threshold values are to be investigated with two separate images. The results of these tests can be seen in **Table 5**. Note that for these tests the best noise reduction gaussian standard deviation of 2 was chosen. Additionally, since the first threshold has been used by the previous input image, a new image was experimented on. The input image for **Table 5** tests is shown in **Figure 2** below.



**Figure 2:** Lena Test Image



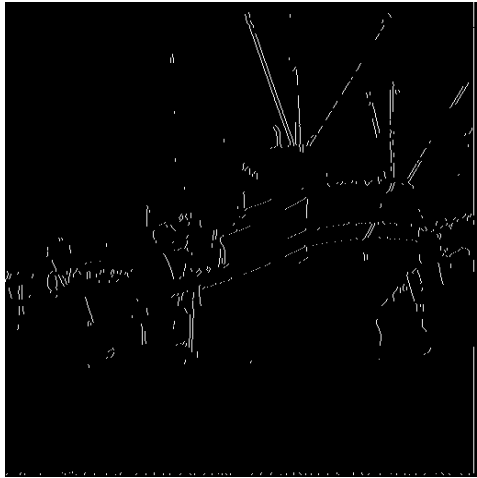
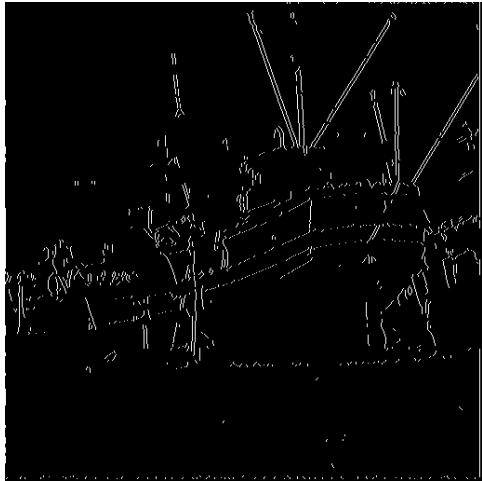
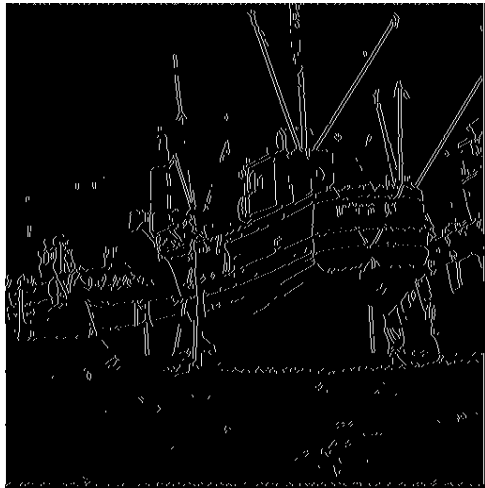
Thresholds	Detector Output
High: 10, Low: 0.01	 The detector output for High: 10, Low: 0.01 shows a black image with white, thin, curved lines that form a stylized, abstract shape, possibly resembling a letter 'X' or a similar character. The lines are somewhat jagged and noisy.
High: 20, Low: 1	 The detector output for High: 20, Low: 1 is a completely black image, indicating no features were detected at these thresholds.
High: 15, Low: .5	 The detector output for High: 15, Low: .5 shows a black image with white, thin, curved lines that form a stylized, abstract shape, similar to the one in the first row but with slightly different line thickness and noise.

**Table 5:** Varied Thresholds

From the tests shown in Table 5 it appears the initial set of thresholds tends to work best for this specific image. To further test these parameters, another image was experiments shown in **Table 6** with the original image shown in **Figure 3**.



**Figure 3:** Fishing Boat Input Image

Thresholds	Detector Output
High: 10, Low: 0.01	
High: 8, Low: 0.02	
High: 6, Low: .03	

**Table 6:** Fishing Boat Varied Thresholds

When conducting these tests it appeared that the smaller the high threshold was the better the performance of the detector. The optimal values that were seen for the fishing boat input image was when the high threshold was 8 and the low threshold was 0.02.

## Conclusion

This report reflected on a manual implementation of the Canny Edge detector. As stated this detector consisted of 4 distinct stages, namely, gaussian noise reduction, *Sobel* kernel filtering, non-maximum suppression, and dual threshold hysteresis. Using this detection configuration different values of noise reduction and thresholds were experimented with to finetune the detection system. The result of this experimentation showed that the optimal values for the gaussian noise reduction was a standard deviation of 2. The optimal threshold values found were determined to be a high threshold of 2 and low threshold of 0.2. In conclusion, this assignment showed how derivative kernels and clever neighborhood comparison techniques could be used to create an edge detector.