**GAZI UNIVERSITY**

**FACULTY OF ENGINEERING AND ARCHITECTURE**

**COMPUTER ENGINEERING**

**CENG471 INTRODUCTION TO IMAGE PROCESSING**
**ASSIGNMENT II**

# Canny Edge Detector

Prepared by

**Selçuk SOLMAZ**

**161180052**

Instructor

**Dr. DUYGU SARIKAYA**

**ANKARA, DECEMBER 2020**

# 1. Canny Edge Detector

Canny edge detector tries to find edge points with a sudden color change in the image.It uses multiple algorithms. It was developed by John F. Canny in 1986 [1].

The result is reached by using more than algorithm in Canny Edge Detector. These algorithms are:

- Grayscale: The first step of Canny Edge Detector is to convert the image to grayscale [1].
- Gauss Filter: Gaussian Filter smooth the image to remove noise in the image. . Thus, we avoid the problem of false detection [2].
- Gradients: Gradient is used to determine horizontal, vertical and diagonal edges in blurry image. Filters such as Prewitt or Sobel are used. At this stage, edge gradient, direction and angle are found [1,2].
- Non-maximum suppression: It looks at the relationship of a perceived edge to its neighboring edge. According to this relationship, the edge is suppressed or the value is preserved [1,2].
- Threshold: Strong edge pixels and weak edge pixels are determined. Weak edge pixels are suppressed while strong edge pixels are preserved [1,2].
- Hysteresis: Blob analysis is applied to the weak edge pixel and its 8 neighbors. In case of one strong edge, this weak edge point must be preserved [1,2].

## 1.1 Gaussian Filter

Thanks to the Gaussian filter, I cleared the image from the noise. I used Opencv's ready library.


**Figure 1. Gaussian Filter Result**

## 1.2 Gradients

I filtered the blurry image vertical and horizontal with the Sobel Filter. You can see the Sobel filter from Figure 4. Then, I took the hypotenuse of Ix and Iy and reached the Gradient value. The result is given below (Figure 3) [1,3,4].

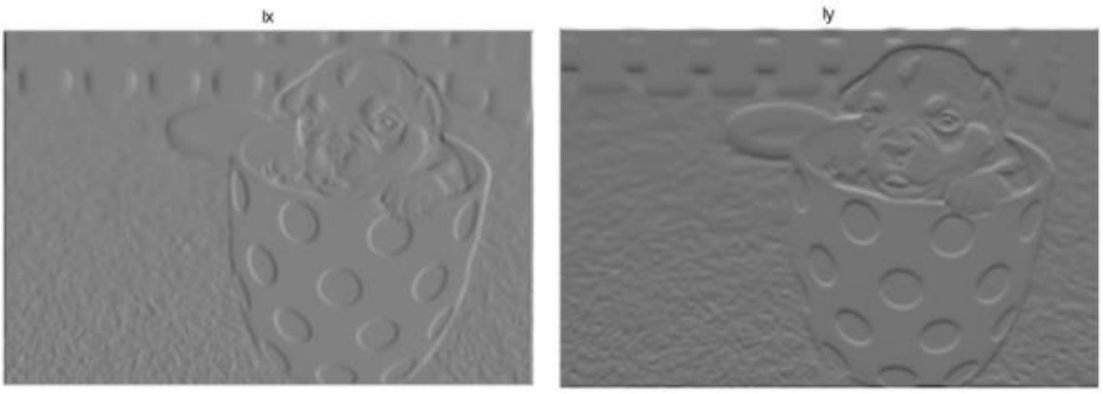


**Figure 2. Gx and Gy Result**

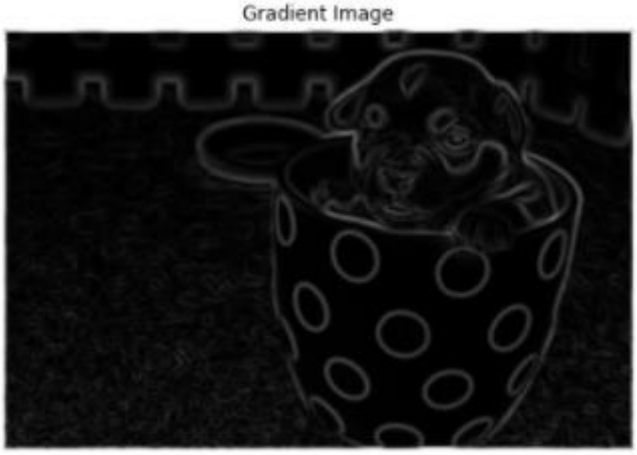


**Figure 3. Gradient Result**

$$\mathbf{G}_x = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} * \mathbf{A} \quad \text{and} \quad \mathbf{G}_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * \mathbf{A}$$

**Figure 4. Sobel Filter [1]**

## 1.3 Non Maximum Suppression

Angle was calculated according to theta value. +180 has been added to negative angles. Then, the relations of each side with its neighbors are examined. Depending on their relationship, the edge is suppressed or preserved [3,4].

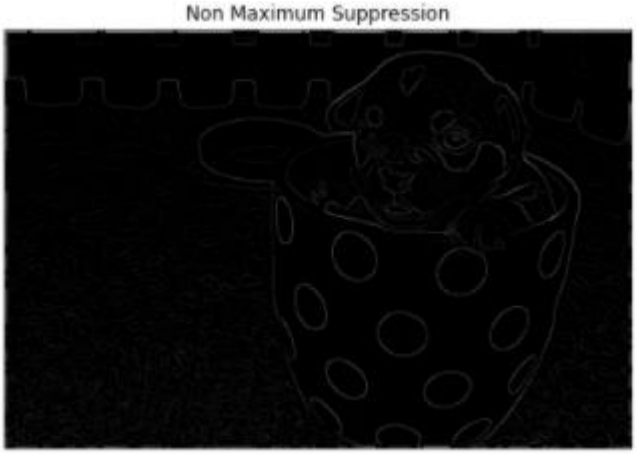


**Figure 5. Non Maximum Suppression Result Image**

## 1.4 Threshold

Thresh is set at low 0.05 and threshold high at 0.09. Low constant is set 0.05 and high constant is set 0.09. According to these values, low thresh and high thresh were calculated. Weak edge and strong edge are defined and strong edges are preserved. Weak edges are suppressed. The weak edge limit is set at 20 [3,4].

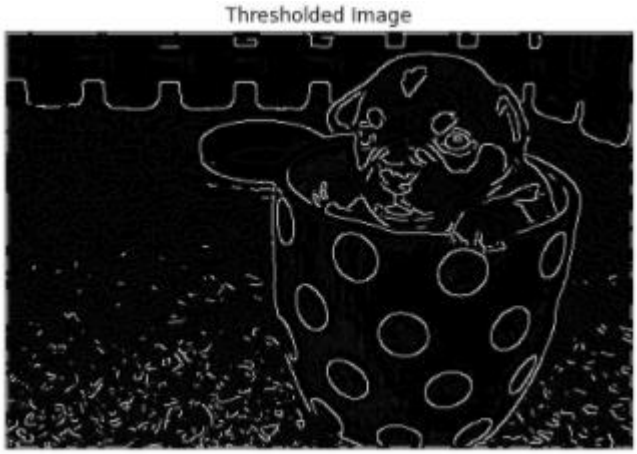


**Figure 6. Thresholded Image**

## 1.5 Hysteresis

Edges in all ranges are checked. Each edge is considered a weak edge. The 8 neighbor edges around this edge is checked for a weak edge or a strong edge. If one of these neighbors is a strong edge, this edge becomes a strong edge. Otherwise, this edge becomes the weak edge and will be suppressed [3,4].

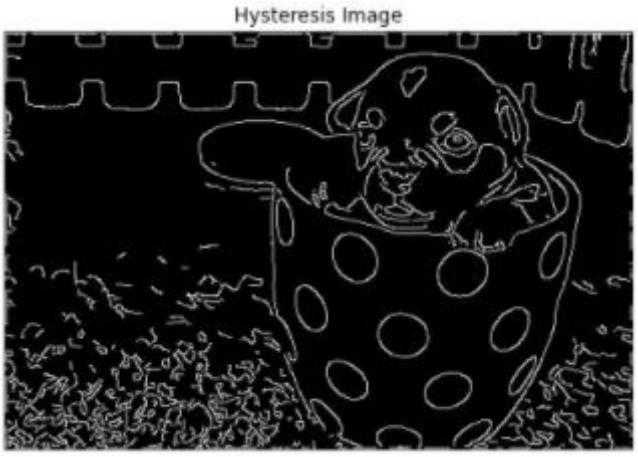


**Figure 7. Hysteresis Image and Final Image**

## 2. Conclusion

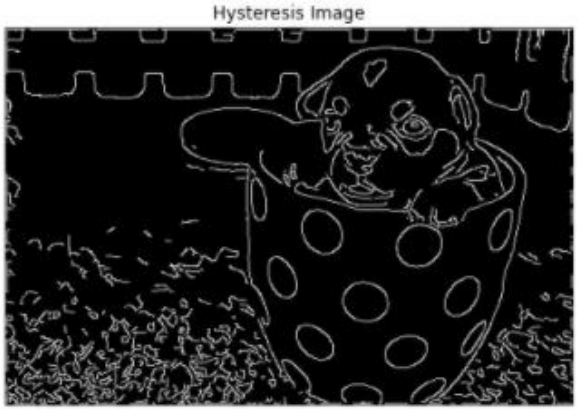


Figure 8. My Final Image


Figure 9. Canny Function Result

Firstly, Figure 8 is the output of the assignment that I prepared, and Figure 9 is the output of the Canny function. Both outputs are very similar. There are small details in between. As an example, the eye area can be given. The reason for this is that I think the Canny function has been optimized more than once. There is little detail between the final result and the Canny function result.

# REFERENCES

1. Canny edge detector. (n.d.). Retrieved from https://en.wikipedia.org/wiki/Canny_edge_detector#:~:text=The%20Canny%20edge%20detector%20is,explaining%20why%20the%20technique%20works.

2. Canny Edge Detection. (n.d). Retrieved from https://docs.opencv.org/master/da/d22/tutorial_py_canny.html

3. Implement Canny Edge Detector in Python using OpenCV. (July,2020). Retrieved from https://www.geeksforgeeks.org/implement-canny-edge-detector-in-python-using-opencv/

4. Sahir, S. (January,2019). Canny Edge Detection Step by Step in Python — Computer Vision. Retrieved from https://towardsdatascience.com/canny-edge-detection-step-by-step-in-python-computer-vision-b49c3a2d8123