161180067

Müzeyyennur YILGIN

22.12.2020

GAZI UNIVERSITY Faculty of Engineering

Computer Engineering

CENG471 - INTRODUCTION TO IMAGE PROCESSING

Assignment 1

Overview

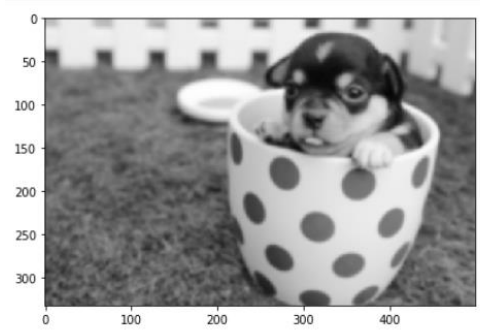
The goal of the project is to better understand the canny edge detector. We were asked to create a canny edge detector from scratch without using a ready-made library. It has been observed that the canny edge marker occurs in 5 steps. These are as follows:

- *Filtered with gaussian or median filter*
- *Find the intensity gradients using sobel filter*
- *Apply non-maximum suppression to get rid of unnecessary edges*
- *Apply double threshold to determine potential edges*
- *Track edge by hysteresis [1]*

Method

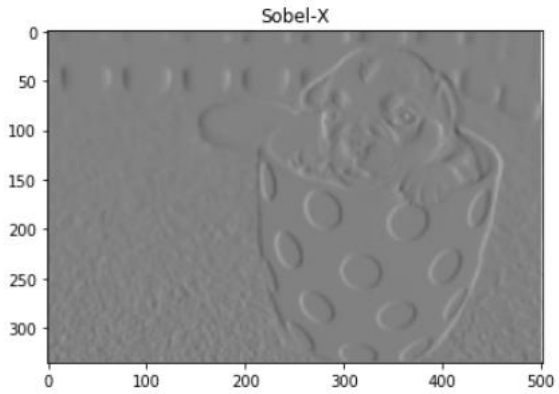
I first uploaded the image in the project. Then I converted the image to gray using COLOR_RGB2GRAY in cv2 library.

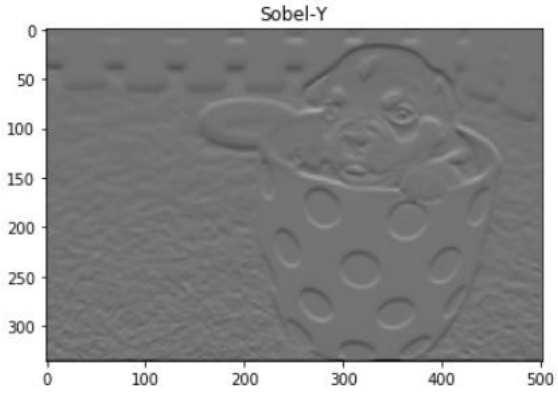
Then, I used gaussian filter. Gaussian filter reduces noise. It smoothed the image. Here, I assign the kernel and sigma value to a random value by trial and error. I assigned the sigma value 1.2 and the kernel value 5. I also evaluated my own filter by comparing it with the library value.

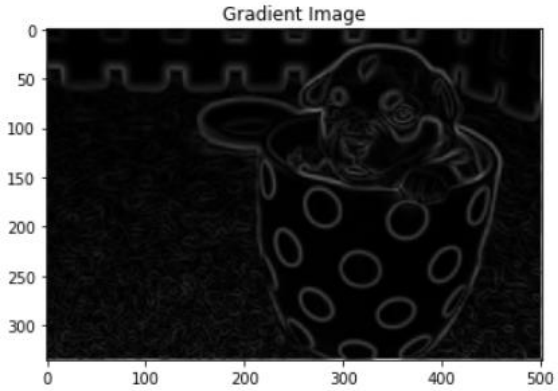


22.12.2020

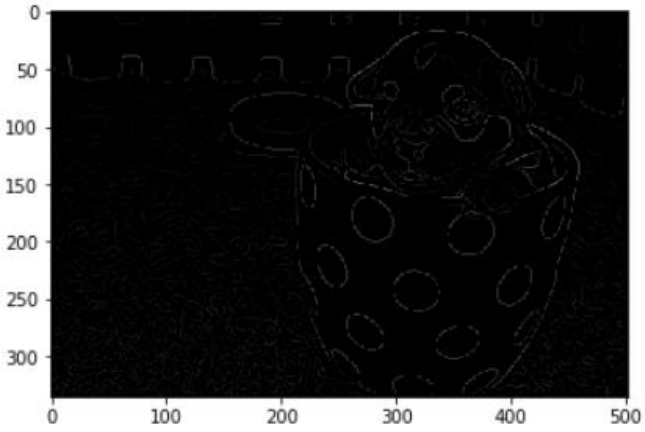
I used a sobel filter to calculate the gradient. The Sobel filter reveals sharp edges. It has two matrices. These matrices are separate for edges that appear horizontally and vertically (Gx, Gy). First, I created two functions for the x and y matrices of the sobel filter. Here, I have filtered the image for x and y separately. Then I convolve it using the library (signal.convolve). Then I calculated the gradient size and theta value using the formulas given in the homework. I got help from numpy library for this.





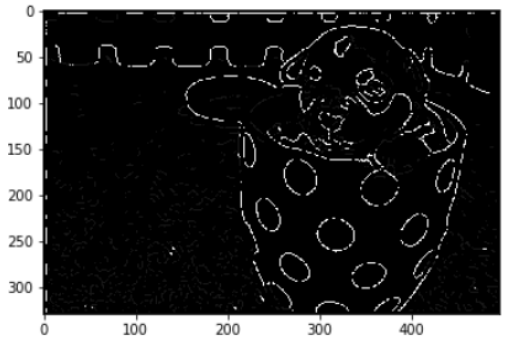


After that, I apply non-maximum suppression. In this way, we get rid of unnecessary edges. For this, I took the gradient degree as a parameter. I converted the radian to degrees. Then I checked all the states with "if condition" for 0, 45, 90 and 135 degrees. I determined which ones should remain as edges. So, I eliminated unnecessary edges.

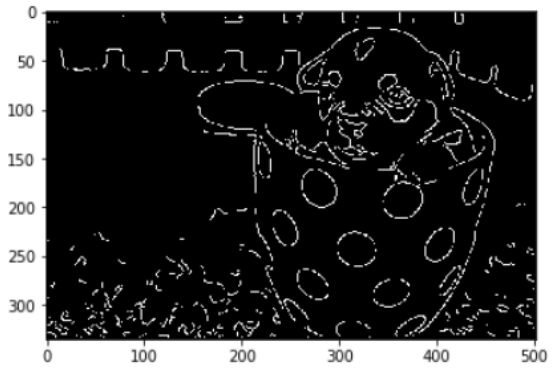


22.12.2020

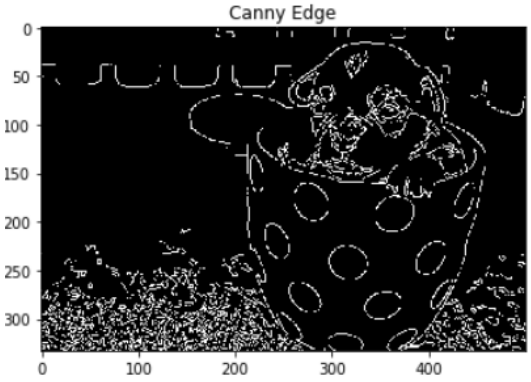
After eliminating unnecessary edges, I applied double thresholds. In this way, I determined possible edges. I assigned the lowest and highest threshold value for separation. I assigned 90 to the highest threshold and 50 to the lowest. Then, I applied thresholding according to these threshold values. I strongly assigned pixels higher than the highest threshold. I assigned pixels that are larger than the lowest threshold value but smaller than the highest threshold value as weak.

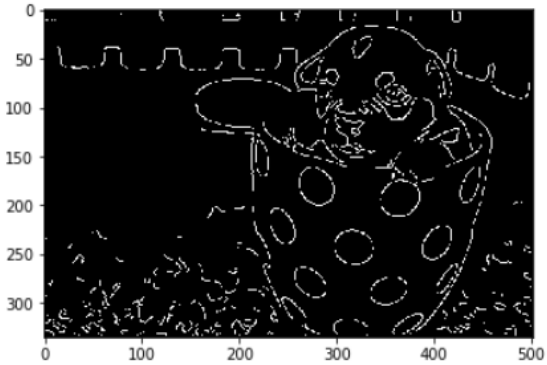


The last step back is remove. Here I track the edges with hysteresis. If the pixel is of weak value and any neighbors are strong, I assign it strong. However, I assign it weak if no neighbors are strong. I did this using "for loop".



After all the steps were finished, I ran a canny edge detector with the ready-made library. Then I compared the results with the detector I prepared.





Result

As can be seen as a result of the comparison, the number of edges found is less than the ready function. The kernel and sigma values determined in the gaussian filter have a share in this failure rate. Better results can be obtained with better values. At the same time, we must determine the highest threshold value and the lowest value well determined in the thresholding. Failure is more likely to occur when thresholding values are not well defined. Given these, better rates should be assigned to make the edge marker better.

Reference

[1] https://en.wikipedia.org/wiki/Canny_edge_detector