CENG471 Introduction to Image Processing
Fall 2020-2021
Lecturer: Dr. Duygu Sarıkaya
Teaching Assistant: Emre Doruk
Gazi University, Department of Computer Engineering
Assignment 2 due on 22nd of December, Tuesday 23:59

The goal of this assignment is to implement a Canny Edge Detector from scratch using Python. Important Note: For this assignment, you are not allowed to use the Canny Edge Detector methods from vision libraries such as OpenCV. You can use other methods and operators from these libraries for other aspects of the homework such as image uploading, writing to a file, grayscale conversion, applying Gaussian filter, matrix operations etc. You can use the image provided in the assignment folder.

Canny edge detector:

The Canny edge detector is an edge detection operator that uses a multi-stage algorithm to detect a wide range of edges in images. It was developed by John F. Canny in 1986. Among the edge detection methods developed so far, Canny edge detection algorithm is one of the most strictly defined methods that provides good and reliable detection, and is therefore among the most popular methods. [1]

The Process of Canny edge detection algorithm can be broken down to 6 different steps:

1. Grayscale Conversion (5 points)
2. Apply Gaussian filter to smooth the image in order to remove the noise (20 points)

   The blur removes some of the noise before further processing the image. You can determine the kernel size and sigma value through trial and error.

3. Find the intensity gradients of the image (30 points)

   The Gradient calculation step detects the edge intensity and direction. The gradients can be determined by using a Sobel filter.

   Then, the magnitude G and the slope θ of the gradient are calculated as follow (the derivatives Ix and Iy with respect to x and y are calculated by convolving Image (I) with Sobel kernels in x and y directions):

$$|G| = \sqrt{I_x^2 + I_y^2},$$
$$\theta(x, y) = arctan\left(\frac{I_y}{I_x}\right)$$

   Gradient intensity and Edge direction

4. Apply non-maximum suppression to get rid of unnecessary edges (10 points)
   We apply non-maximum suppression to remove redundant/duplicate edges identified by Sobel Edge. We want just one line to show the edge rather than having multiple lines for the same edge. This can be done by the Non-Max Suppression Algorithm. Please see [1] for details.

5. Apply double threshold to determine potential edges. (5 points)
   You can set two thresholds, a high and a low threshold to filter out edge pixels with a weak gradient value and preserve edge pixels with a high gradient value.

6. Track edge by hysteresis: (5 points) Finalize the detection of edges by suppressing all the other edges that are weak and not connected to strong edges. (Fort his stage, partial solutions are okay, you don't need to consider all cases). Please see [1] for details.

   For example, you can loop through each pixel in the image, if the value of the pixel is weak and there are not any neighboring pixel with value 255, you can set the value of the pixel to 0.

Report (25 points)

Submission:

You will submit a jupyter notebook (ipynb file) with executable Python script and a short report. Please do not forget to add comments at the top of the related section(comment line). Important Note: For this assignment, you are not allowed to use the Canny Edge Detector methods from vision libraries such as OpenCV. You can use other methods and operators from these libraries for other aspects of the homework such as image uploading, writing to a file, grayscale conversion, applying Gaussian filter, matrix operations etc. Please refer to course slides, tutorials and practicals to set up a running Python environment, Jupyter notebook and to import these libraries. You can check the documentation of each library (available online) to get more information about the functions you will use.

Your report should contain;

- a brief overview of the problem in your own words,
- the details of your approach,
- the results of your algorithm on the image provided with the homework. You should show the result (output) after each step, and explain what happens at each step.
- If your algorithm failed to give a satisfactory result on a particular image, provide a brief explanation of the reason(s).

Please, give all references that you used.

Important Note:

This is an individual assignment, meaning that you will be working on it alone (please check the Class Rules and Expectations below, also available in the syllabus)

Grading

The assignment will be graded out of 100: You will receive full points only when your script 1)executes, 2)gives the correct answer, and when 3) the explainations are provided.

Course Rules and Expectations

All work on programming assignments must be done individually unless stated otherwise. You are encouraged to discuss with your classmates about the given assignments, however, everything that is turned in for each assignment must be your own work. In particular, it is not acceptable to: submit another person’s assignment as your own work (in part or in its entirety), get someone else to do all or a part of the work for you, submit a previous work that was done for another course in its entirety (self- plagiarism), submit material found on the web as is etc. These acts are in violation of academic integrity (plagiarism), and these incidents will not be tolerated. Homeworks, programming assignments, exams and projects are subject to Turnitin (https://www.turnitin.com/) checks.

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

[2] Image Ref : The image used for the assignment is from https://images.pexels.com/photos/39317/chihuahua-dog-puppy-cute-39317.jpeg?auto=compress&cs=tinysrgb&dpr=1&w=500