COMPUTER VISION ASSIGNMENT 2

Implementation of Canny's Edge Detector

Edge detection is very important to find sudden changes in an image. In this assignment, we are going to learn about the implementation of a very important edge detector called Canny's edge detector.

Guidelines

1) Submit all of your code and results in a single zip file with name FirstName_RollNumber_02.zip

2) Submit single zip file containing
   a) code.ipynb
   A python Notebook file including all the functions in separate cells
   b) Results
   A folder with result images for all sample images given.
   c) Readme.txt
   Should explain how to run your code in the file.
   d) **Report.pdf**
   Explanation of your implementation and highlighting interesting results

3) Submit the hard copy of the mathematical part.

4) Email instructor or TA if there are any questions. You can discuss with each other's But cannot look at or use others code.

5) This document has two parts, first **Implementation** and second is the **mathematical** part. Implementation part has two portions, the first **Steps for implementation** and second **Problems.**

6) Follow the given link for details. This has all the information needed to solve the assignment. Steps for implementation in the assignment are also from the given link. [1]
   http://suraj.lums.edu.pk/~cs436a02/CannyImplementation.htm

7) Before solving the problem section, first, go through the steps which will help you implement Canny's edge detector step by step.

8) You cannot use any built-in function unless specifically mentioned

Deadline for Programming part is Sunday 6th October 2017 before 11:59 pm.
Deadline for Mathematical part is Thursday 3rd October 2019 before 9:00 pm. (Submit a hard copy of mathematical part after the class on Thursday)

Note: You can submit assignment till 1 day after the deadline with 15% deduction.

Implementation

There are some basic steps for the implementation of a Canny edge detector.

1. Generation of Masks
2. Applying Masks to Images
3. Compute gradient magnitude
4. Compute gradient Direction
5. Non-Maxima Suppression
6. Hysteresis Thresholding

Steps for implementation:

1. **Generation of Masks:** This module requires the value of **sigma** as an input and generates x- and y-derivative masks as output.
   - Mask size: To generate the masks, the first step is a reasonable computation of the mask size. Mask size depends on the value of sigma and T. Formula of size of half mask is as follows:

     sHalf = round(sqrt(-log(T) * 2 * sigma^2))

   - Where sigma and T are input parameters.
     i. Sigma can be **0.5, 1, 1.5**… The lower limit of sigma is 0.5.
     ii. Width of the mask is based on parameter T. T can be any value **between 0 and 1**. For example **T=0.3.** The above formula will give us the size of half mask.
   - The total mask size would then be computed as follows:

     N= 2*sHalf +1 (total mask size)

Example:

sigma=0.5; T=0.3

sHalf = round(sqrt(-log(T) * 2 * sigma^2)) => 1

N= 2*sHalf +1 => 3

[Y, X] = np.meshgrid ((-sHalf : sHalf+1), (-sHalf : sHalf+1)) => Y= [[-1,0,1] [-1,0,1] [-1,0,1]]

- In Canny's method, the masks used are the **1st derivative of Gaussian** in x- and y-directions. The gaussian formula is as follows:

  G (x, y) = exp (-(x^2 + y^2)/(2*sigma^2)

- Now take the first derivative of Gaussian w.r.t ‘x’ and call it ‘fx’ and put the valve of X, Y in it. Repeat this procedure w.r.t ‘y’ as well. These (Gx, Gy) are the two masks that will be convolved with the image in the next step.

- Multiply the Gx and Gy with 255 to scale up and then round off these values so that convolution is performed with integer values rather than floats.

  Save the scale factor, because gradient magnitude will be later scaled down (after convolution) by the same factor.

Note: You should write a function to calculate the gradient of gaussian not just save it.
calculate_gradient (filter_size, sigma)
Write a separate function to calculate filter size named calculate_filter_size (sigma, T)

2. **Applying Masks to Images:** The masks are applied to the images using convolution. Store convolution results in fx and fy.

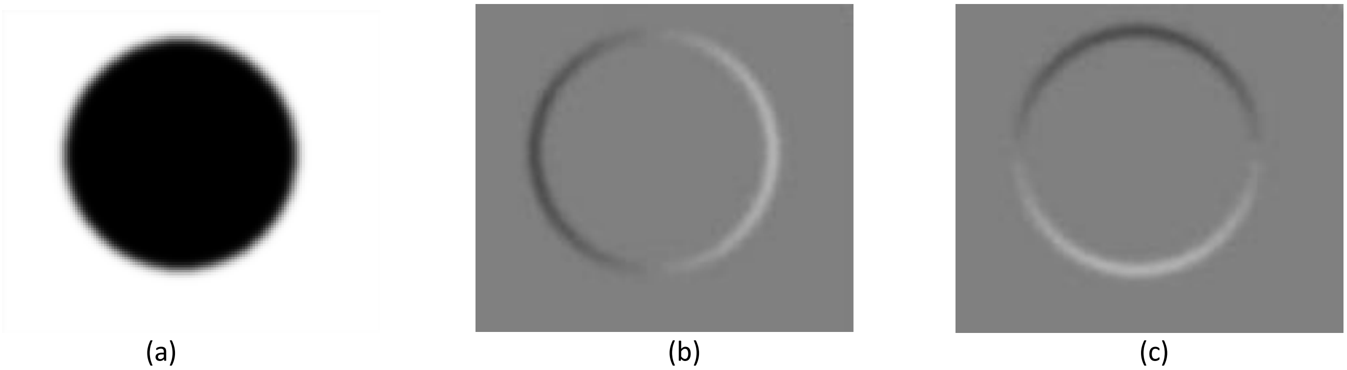


Figure 1: Image (a) is an input image 'circleBlured.png'. Image (b) is generated after convolving the input image with Gy (Gaussian derivative w.r.t 'y'). Image (c) is generated after convolving the input image with Gx (Gaussian derivative w.r.t 'x').

3. **Compute gradient magnitude**: Compute the gradient magnitude of images (fx, fy) at each pixel after convolving with masks (Gx, Gy). M is computed from fx and fy images, using the magnitude formula:

$$M=\sqrt{f_x^2 + f_y^2}$$

Figure 2: Gradient magnitude of input image normalized b/w 0-255

Note: The result is then scaled down by the same factor which was used to scale up the masks. To write output to image files, the min and max values are scaled to 0 and 255 respectively.

4. **Compute gradient Direction**: Compute the gradient direction of images (fx, fy) at each pixel after convolving with masks (Gx, Gy). Phi is computed using atan2 function by the following a formula:

$$\theta = \arctan \frac{f_y}{f_x}$$

Note: Convert the angle returned by math.atan2 function to degrees and add 180 to get an output range of 0-360 degrees.

5. **Non-Maxima Suppression:** Non-Maxima suppression step makes all edges in M one pixel thick.

- The first step is to quantize gradient direction into just four directions.

Value assigned	Angle
0	0 to 22.5 157.5 to 202.5 337.5 to 360
1	22.5 to 67.5 202.5 to 247.5
2	67.5 to 112.5 247.5 to 292.5
3	112.5 to 157.5 292.5 to 337.5

Table 1 Quantize gradient directions

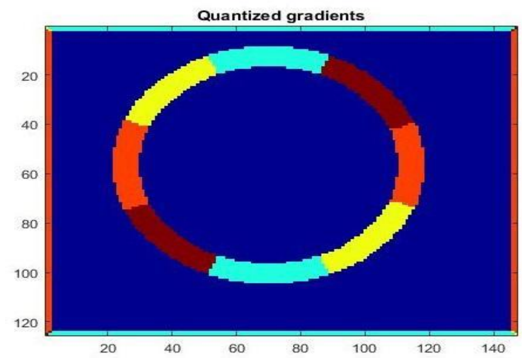
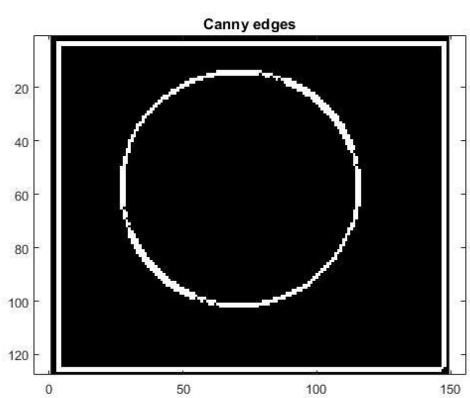
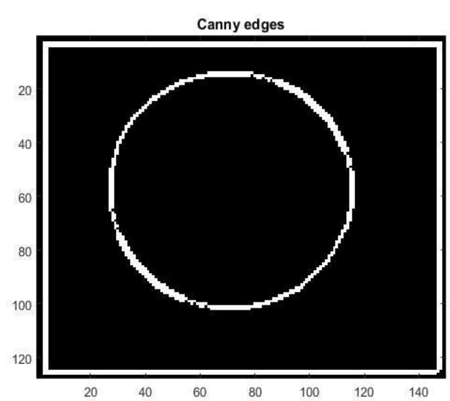


Figure 3: Image after quantizing gradient directions

- The next step is to pick two neighbors of each edge point **along the gradient direction**. This is because gradient direction is perpendicular to the edge, and therefore, this is the direction in which we are searching for edge points.
- Therefore, the two neighbors that need to be picked for comparison are the north and south neighbors. If the edge point (r, c) is greater than both these neighbors, then it is maintained in M otherwise it is made zero.

6. **Hysteresis Thresholding:** The final step in Canny's edge detection algorithm is to apply two thresholds to follow edges.
    - First made the border pixels zero, so that finding neighbors does not go out of bounds of the image.
    - Next, the image is scanned from left to right, top to bottom. The first pixel in non-maxima suppressed magnitude image which is above a certain threshold, Th, is declared an edge.
    - Then all its neighbors are recursively followed, and that above threshold, Tl, is marked as an edge.
    - Thus, there are really two stopping conditions:
        i. if a neighbor is below Tl, we won’t recurs on it.
        ii. if a neighbor has already been visited, then we won't recurs on it.





Note: See [1] for more details

Problems:

For Assignment 2, you have to implement the Canny Edge Detector. Test Images are given in the assignment folder. Create separate functions for all 4 steps.

For each image, you have to submit

a) images showing gradients in X and Y direction **(10)**
b) image showing the Magnitude of gradients at each pixel **(10)**
c) Image of quantized orientations **(15)**
d) Final Result for sigma = 1, sigma = 0.5 and sigma = 2 **(20)**
e) For sigma =1, submit 2 different results for 2 different pairs of Th, Tl in Hysteresis thresholds. **(25)**

Note: Perform each experiment for all images provided and save resultant images in the Results folder and name them as
[original Name] _fx_[sigma value].jpg
[original Name] _fy_[sigma value].jpg
[original Name] _magnitude_[sigma value].jpg
[original Name] _ quantized_[sigma value].jpg

Mathematical Part

Problem 1: Calculate and plot (using matplotlib.pyplot's functions e.g. plot_surface) derivative and double derivative of Gaussian filter. **(10)**

Problem 2: Prove that Convolutions are commutative a*b=b*a (* => convolution) **(05)**

Problem 3: Calculate the rank of given matrices. **(05)**

A= [[1 2 3] [3 6 9]]

B= [[1 2 3] [4 5 6] [7 8 9]]

Note: add page number on top of every page in the hard copy

References:

[1] http://suraj.lums.edu.pk/~cs436a02/CannyImplementation.htm