Semester I, 2019 – 2020 November 23, 2019

CS517: Digital Image Analysis

Final Lab Test

- For the Final Lab Test you will implement Canny's edge detection algorithm as described in class.

- You will write a MATLAB function that takes as input a grayscale image and the edge detection parameters. The parameters should be the standard deviation for the Gaussian convolution and the gradient magnitude thresholds (Low and Hi for implementing hysteresis). The output of your function should be a binary image with the results of edge detection.

    - `function[outImg] = MyCannyEdgeDetector( inImg, sSigma, tLow, tHigh )`

- IMPORTANT: For this assignment you should not use any predefined image processing or filtering routines in MATLAB.
    - You are only allowed to use in-built functions to read and save images and 2D convolution.

- Remember from the lecture that in Canny edge detection, we will first smooth the images, then compute gradients, magnitude, and orientation of the gradient. This procedure is followed by non-max suppression, and finally hysteresis thresholding is applied to finalize the steps.

- **More details on Non-Maxima Suppression (`MyNonMaxSupression.m`)**:

Definition: Non-maximal suppression means that the center pixel, the one under consideration, must have a larger gradient magnitude than its neighbors in the gradient direction. That is: from the center pixel, travel in **the direction of the gradient** until another pixel is encountered; this is the first neighbor. Now, again starting at the center pixel, travel in the direction opposite to that of the gradient until another pixel is encountered; this is the second neighbor. Moving from one of these to the other passes though the edge pixel in a direction that crosses the edge, so the gradient magnitude should be largest at the edge pixel.

Algorithmically, for each pixel p (at location x and y), you need to test whether a value $M(p)$ is maximal in the direction $\theta(p)$. For instance, if $\theta(p) = pi/2$, i.e., the gradient direction at $p = (x, y)$ is downward, then $M(x, y)$ is compared against $M(x, y-1)$ and $M(x, y+1)$, the values above and below of p. If $M(p)$ is not larger than the values at both of those adjacent pixels, then $M(p)$ becomes 0. For estimation of the gradient orientation, $\theta(p)$, you can simply use $atan2(I'_y, I'_x)$.

- **More details on Hysteresis (`MyHysteresis.m`)::**

After application of non-maximum suppression, remaining edge pixels provide a more accurate representation of real edges in an image. However, some edge pixels remain that are caused by noise and color variation. In order to account for these spurious responses, it is essential to filter out edge pixels with a weak gradient value and preserve edge pixels with a high gradient value. This is accomplished by selecting high and low threshold values. If an edge pixel's gradient value is higher than the high threshold value, it is marked as a strong edge pixel. If an edge pixel's gradient value is smaller than the high threshold value and larger than the low threshold value, it is marked as a weak edge pixel. If an edge pixel's value is smaller than the low threshold value, it will be suppressed. The two threshold values are empirically determined and their definition will depend on the content of a given input image.

So far, the strong edge pixels should certainly be involved in the final edge image, as they are extracted from the true edges in the image. However, there will be some debate on the weak edge pixels, as these pixels can either be extracted from the true edge, or the noise/color variations. To achieve an accurate result, the weak edges caused by the latter reasons should be removed. Usually a weak edge pixel caused from true edges will be connected to a strong edge pixel while noise responses are unconnected. To track the edge connection, blob analysis is applied by looking at a weak edge pixel and its 8-connected neighborhood pixels. As long as there is one strong edge pixel that is involved in the blob, that weak edge point can be identified as one that should be preserved.

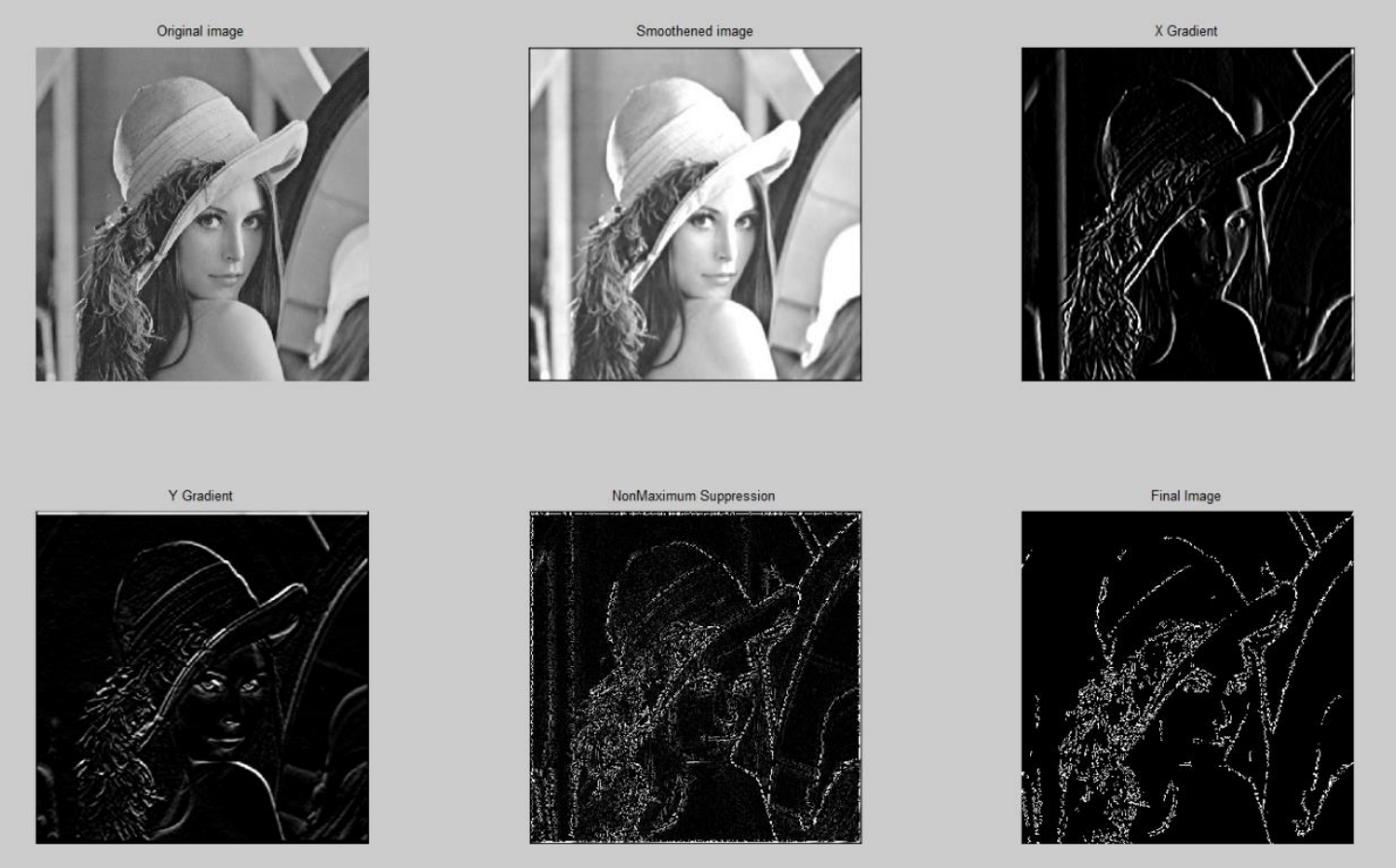
- **Pseudocode for the algorithm is provided in Figure 1 below:**

Algorithm 1 Standalone Canny Edge Detection

Input: Image I, thresholds τ_l, τ_h
Output: Canny edge image E, gradient information M, ϕ

procedure CANNY-STANDALONE
Convolve image with Gaussian filter G
    $I \leftarrow I \circledast G$
Compute gradients
    for each pixel (x, y) **do**
        $g_x(x, y) = \frac{1}{2}I(x-1, y) - I(x, y) + \frac{1}{2}I(x+1, y)$
        $g_y(x, y) = \frac{1}{2}I(x, y-1) - I(x, y) + \frac{1}{2}I(x, y+1)$
        $M(x, y) = \sqrt{g_x(x, y)^2 + g_y(x, y)^2}$ ▷ magnitude
        $\phi(x, y) = \tan^{-1}(g_y(x, y)/g_x(x, y))$ ▷ orientation
    end for
Non-maxima suppression
    for each pixel (x, y) **do**
        Find 2 neighbours in direction of gradient $\phi(x, y)$
        if $M(x, y) \leq M$(any neighbour) **then**
            $M(x, y) \leftarrow 0$
        end if
    end for
Hysteresis thresholding
    for each pixel (x, y) **do**
        if $M(x, y) \geq \tau_h$ **then**
            $E(x, y) \leftarrow 1$ ▷ edge pixel
        else
            $E(x, y) \leftarrow 0$ ▷ non-edge pixel
        end if
    end for
    Recursively find non-edge pixels with $M \geq \tau_l$ and with an edge pixel as a neighbour. Mark them as edge pixels as well.
end procedure

- **Test Script (`FinalLabTest.m`)**
    a. Create a simple test script to show case the performance of your algorithm for the provided three sample images (`Lena512`, `Rice` and `Tire`)
    b. Format of the output should be as follows:



    c. Set the input parameters to generate the best possible results.
    d. Comment on your observations in the README file.

- **Submitting your work:**
    a. All source files and class files as one tar-gzipped archive.
        i. When unzipped, it should create a directory with your ID. Example: **P2008CS1001–F** (NO OTHER FORMAT IS ACCEPTABLE!!! Case sensitive!!!)
        ii. Negative marks if the TA has to manually change this to run his/her scripts!!
    b. Source / class files should include the following: (Case-Sensitive file names!!)
        i. `MyCannyEdgeDetector.m, MyNonMaxSupression.m, MyHysteresis.m, FinalLabTest.m`
        ii. Any other additional functions that you might need to create
        iii. **README** (Should provide all the necessary details to the TA for ease of grading – what works, what doesn't, any assumptions made, etc.)
        iv. *Any other utility functions you might end up creating*

c. ***Negative marks for any problems/errors in running your programs***

d. Submit/Upload it to Moodle or Give a copy to the TA

- **Grading Scheme : 80 Points (Total)**
  - `MyCannyEdgeDetector` (20 Points)
  - `MyNonMaxSupression` (20 Points)
  - `MyHysteresis` (20 Points)
  - `FinalLabTest` (10 Points)
  - README (5 Points)
  - Code + Comments (5 Points)