Homework 2

Implement the Canny edge detector for a grayscale image.

Step 1. Write a function, called my_Normalize(img), to prepare the image:

(1) If img is a grayscale image, do nothing; if it is a color image, convert it to a grayscale image;

(2) Normalize this image so that it becomes a matrix whose elements are float-point numbers and within range [0,1]

Step 2. Write a function, called my_DerivativesOfGaussian(img, sigma), to compute the derivatives of smoothed images:

(3) Construct two 3×3 Sobel kernels, S_x and S_y as the derivative operators on both x and y directions

(4) Construct a Gaussian kernel g_σ where σ is the variance specified as the input parameter sigma; set its mask size to $6\sigma + 1$ (Note: Check example program 9)

(5) Construct the derivative of Gaussian kernels, g_x and g_y by convolving the above two kernels: $g_x = S_x * g_\sigma$; $g_y = S_y * g_\sigma$

(6) Apply both kernels g_x and g_y on the input image, and get two resultant images I_x and I_y

(7) Normalize both I_x and I_y and render them

(8) Return the **un-normalized** derivative images I_x and I_y

Step 3. Write a function, called my_MagAndOrientation(Ix, Iy, t_low),to compute the magnitude and orientation of the gradient image:

(9) Compute the magnitude image: $m(x,y) = \sqrt{(Ix(x,y)^2 + Iy(x,y)^2)}$;

(10) Compute the orientation of gradient, store it in an image: $O(x,y) = \tan^{-1}(Iy(x,y)/Ix(x,y))$ (Note: you can use numpy.arctan2. See https://docs.scipy.org/doc/numpy/reference/generated/numpy.arctan2.html)

(11) Round each element in O into one of the four values: $[0, \frac{\pi}{4}, \frac{\pi}{2}, \frac{3\pi}{4}]$ (check Lecture 10 slides page 7). Here save it as one of the four integers: $[0,1,2,3]$, respectively.

Note: In the slides, the y^+ axis is upwards, while in a matrix representation, the y^+ direction is downwards. An easy modification to tackle this is to switch case 1 and case 3 in your implementation. Check your computation using 'TestImg1.jpg' example.

(12) Normalize the magnitude image.

(13) Plot the normalized magnitude image; and the rounded integer orientation image; return them.

Step 4. Write a function, called my_NMS(mag, orient, t_low), to perform non-maximal suppression along the gradient direction:

(14) Create a zero matrix called mag_thin

(15) For each pixel mag[i][j] whose value is smaller than t_low, ignore it

(16) For each pixel whose value value is bigger than t_low, if it is bigger than its two neighbors in the gradient direction, let mag_thin[i][j] = mag[i][j].

(17) Plot and return mag_thin.

Step 5. Write a function, called my_linking(mag_thin, orient, tLow, tHigh), to perform linking using hysteresis thresholding.

(18) Create a zero matrix called result_binary

(19) (Forward Scan) Perform a forward scan on rows (i=0 to maxRow-1) and columns (j=0 to maxCol-1): for each pixel mag_thin[i][j] whose value is ≥ tHigh, check if its Right (or BottomRight, or Bottom, or BottomLeft, according to the edge direction) neighboring pixel has a value bigger than tLow. If so, mark that neighboring pixel's value to tHigh.

(20) (Backward Scan) Perform a backward scan on rows (I = maxRow-1 to 0) and columns (j = maxCol-1 to 0): for each pixel whose value ≥ tHigh, check if its Left (or TopLeft, or Top, or TopRight) neighboring pixel has a value bigger than tLow, if so, mark that neighboring pixel's value to tHigh.

(21) Fill the result_binary image: result_binary[i][j] = 1 if mag[i][j] ≥ tHigh.

(22) Return the result_binary image.

Step 6. Compose all the above steps 1-5, to get a function called my_Canny(img, sigma, tLow, tHigh).

Hints:

a. First test the correctness of your Step 1 ~ 3 using 'TestImg1.jpg' to see whether the gradient orientations are computed correctly.
b. Adjust tLow and tHigh to see different detection results. The values should be between 0 and 1.
c. In this homework, together with your codes, please submit your resultant binary images. Provide a readme.txt file , indicating what parameters were you used to generate this resultant images.

Your codes and resultant images are due: 11:59pm, Mar. 3rd.

Submit your homework on Moodle.

If you are late for k $(k < 7)$ days, your score will be calculated as:

Final homework score = (The score based on your codes) $\times\, 0.95^k$

If you are late for more than 7 days, you get no score.