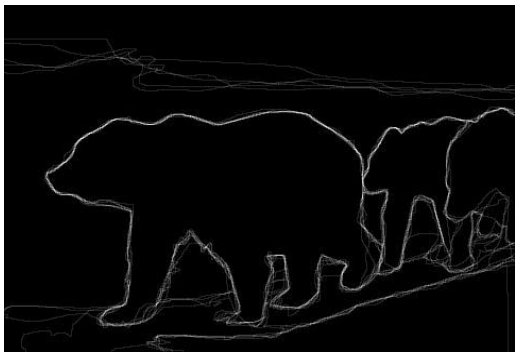
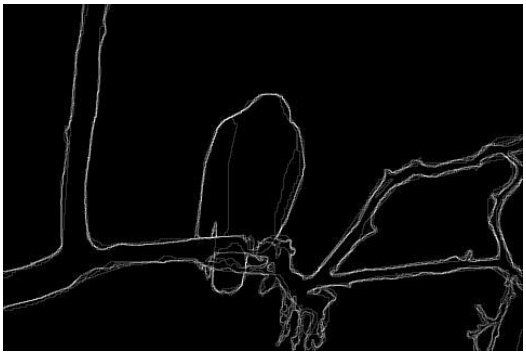



LE/ESSE 4690 Advanced 3D Geospatial Techniques – Fall 2020
LAB 2: Edge Detection

Last updated: Thursday 09/26/20

Brief

- Assigned: October 1st, 2020
- Due: 11:59 PM, October 8th, 2020
- Lab materials are available on 4690 course eClass
- Submission: through the course moodle by the due time
- Required files: report, result images and codes (with comments) in separate files

1. Objectives

The aim of the second lab practical is to enhance your understanding of the lectures on "Edge Detection" through the development of code for an edge detector based on the Canny edge detector (Implement the Canny edge detector – J.F. Canny, "A computational approach to edge detection", IEEE PAMI, 8(6), pp. 679-698), and a quantitative evaluation of the performance of the edge detectors.

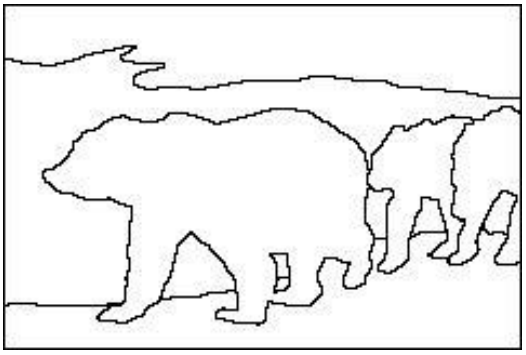
2. Details

2.1. Test and Training Images:

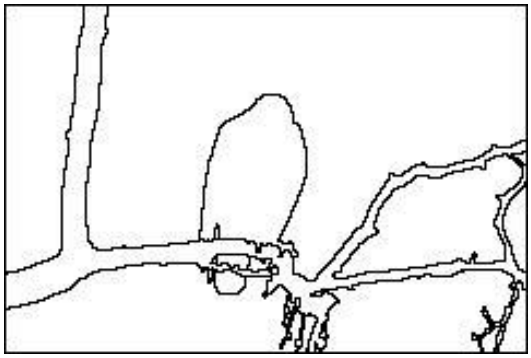
We provide two images from the Berkeley Segmentation Dataset (https://www2.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/segbench/). Each image contains one original image and one reference image which edges were hand labelled by a human subject. You are asked to use one image (Training Images #100075) as a training image, with which you can tune the parameters of your Canny edge detector. Then, you apply your edge detector with the tuned parameters to the test image. You are asked to conduct a quality assessment of the edge images produced using training and test images given.

(a) Training Image #100075 (Original Image)

(b) Test Image #42029 (Original Image)



(c) Training Image #100075 (Reference Image)



(d) Test Image #42029 (Reference Image)

Figure 1. Test and training images from the Berkeley benchmark.

2.2. Canny Edge Detector:

Write code for a Canny edge detector using MATLAB as following steps:

- Smooth input image by $G = \exp(-\frac{x^2 + y^2}{\sigma^2}),\ J = I \bullet G$
- Compute image derivatives $J_x,\ J_y$
- Compute edge strength $E_s = \sqrt{J_x^2 + J_y^2}$ and direction $E_o = \arctan(\frac{J_x}{J_y})$
- Non-maximum suppression to derive single pixel width edge
- Implement hysteresis thresholding: given high threshold t_h and low threshold $t_l\,(t_h \geq t_l)$, mark as edges all points with either:
  1. E_s is larger than t_h
  2. E_o is larger than t_l and connected to an edge point $\hat{e}$ with $Es(\hat{e}) > t_h$ by other edge points with strength $Es(\hat{e}) > t_l$, in the direction of the edge at $\hat{e}$

Helpful MATLAB functions (aside from **edge (.., 'canny')**) include filter2, gradient, and **fspecial ('gaussian',..)**. Experiment with running your edge detector on a couple of your favorite images, with different values for σ, t_l, and t_h. The output of your program should be a binary image. Note that you should implement Gaussian convolution as a sequence of horizontal and vertical convolutions. Your

implementation of hystheresis should be efficient. For an image with n pixels it should take O(n) time for the hystheresis step.

2.3. Edge detector quality assessment

- Construct a binary image mask *G*, that is "true" (equal to one) at every pixel using the training reference image (Figure 1(c)) and "false" (equal to zero) otherwise;
- Construct similar binary image mask *D*, for the edges detected by your Canny edge detector.
- Use this binary mask as a function of the decision threshold to assess the performance of the edge detectors by constructing a receiver operating characteristic curve (ROC) as follows: First compute:
  1. The number of edge pixels correctly detected as "true positives"
     - $N(TP) = \sum pixel(D \cap G)$
  2. The number of edge pixels incorrectly detected as "false positives"
     - $N(FP) = \sum pixel(D \cap G')$,where G' is the complement of G (i.e., NOT G)
  3. The total number of edge pixels, $N(E) = \sum pixels(G)$, and the total number of background pixels, $N(B) = \sum pixels(G')$
- Calculate $N(E) + N(B) = N$, the total number of pixels that could be marked as edges or not, and comment of whether this is equal to the number of pixels in your original images. Hence, compute the probabilities of:
  1. Edge detection: $P(D) = \{N(TP) + N(TP)\}/N$
  2. The detected edge pixels being correct: $P(TP) = N(TP)/N(E)$
  3. Pixels detected as belonging to an edge being incorrect: $P(FP) = N(FP)/N(B)$

2.4. Optimization of edge detector

You experiment with the values of these parameters and notice the difference in the final result. A few of parameters that govern the performance of Canny edge detector need to be optimized as follows:

- Use a mask width $w = 5\sigma$ and $t_h = 2t_l$; The parameter optimization is conducted only over σ and tl; the other thresholds can be arbitrary or heuristically determined in your own way;
- With various σ (3 different values), calculate the edge strength Es and direction Eo;
- With each σ, plot the histogram of edge strength Es using the test image given and determine a threshold which you think is optimal (cf. reference image). Explain how and why the edge strength threshold is selected;
- With determined thresholds and parameters, generate the edge maps D using your version of Canny edge detector and conduct the quality assessment (2.3);
- Determine the most optimal parameter values which produce the best performance you think using the performance measure addressed in 2.3. Explain how the parameters for Canny edge detector are optimized;
- Apply your edge detector with the optimized parameters (σ, t_l) and generate the edge map using the test image given;
- Discuss the quality of the final edge map using the quality measure in relation to the performance shown through the training steps (optimization steps).

3. Bells & Whistles (Extra Points)

It is possible to get extra credit for this project, as well, if you come up with some clever extensions which impress the TA.

4. Writeup

For this project, and all other projects, you must write a project report (no min. page length; max. 12 pages with 11 fonts and single space including figures and tables). In the report, you will describe each of the steps listed below and write a brief report containing:

- A description of the techniques used, where appropriate in concise mathematical terms;
- A description of what was done, what data was used, how it was obtained, what MATLAB library functions and procedures were used, the results obtained;
- An analysis and critique of the results;
- Any conclusions you can draw from the exercise;
- Also, discuss anything extra you did and feel free to add any other information you free is relevant.

5. Rubric

+50 pts: Working implementation of Canny edge detector
+30 pts: Working implementation of quality assessment and parameter optimization
+20 pts: Writeup with several examples of hybrid images
+10 pts: Extra credit (up to ten points)
-5*n pts: Lose 5 points for every time you do not follow the instructions for the hand in format
-5*n pts: Lose 5 points for every day you delay the submission of your report and codes beyond the due date.