EE 569: Homework #2

Issued: 01/31/2022 Due: 11:59PM, 02/20/2022

General Instructions:

1. Read *Homework Guidelines* and *MATLAB Function Guidelines* for the information about homework programming, write-up and submission. If you make any assumptions about a problem, please clearly state them in your report.
2. You need to understand the USC policy on academic integrity and penalties for cheating and plagiarism. These rules will be strictly enforced.

Problem 1: Edge Detection (50 %)

(a) Sobel Edge Detector (10%)

Implement the Sobel edge detector and apply to *Tiger.raw* and *Pig.raw* images as shown in Fig. 1 (a) and (b). Note that you need to convert RGB images to grayscale image using the formula below.

$$Y = 0.2989 * R + 0.5870 * G + 0.1140 * B$$

Include the following in your results:

(1) Normalize the x-gradient and the y-gradient values to 0-255 respectively and show the results;
(2) Calculate and show the normalized gradient magnitude map;
(3) Tune the thresholds (in terms of percentage) to obtain an edge map with the best visual performance. The final edge map should be a binary image whose pixel values are either 0 (edge) or 255 (background).

(b) Canny Edge Detector (10%)

The Canny edge detector is an edge detection technique utilizing image's intensity gradients and non-maximum suppression with double thresholding. In this part, apply the Canny edge detector [1] to both *Tiger.jpg* and *Pig.jpg* images from BSDS 500 dataset [2]. You are allowed to use any online source code such as the Canny edge detector in the MATLAB image processing toolbox or OpenCV.

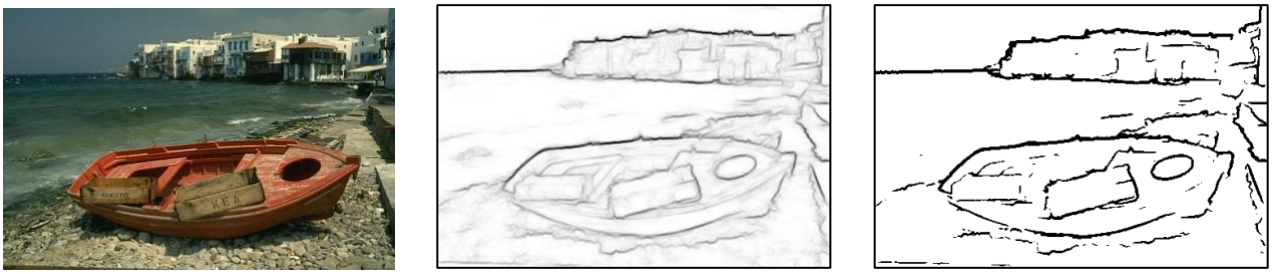
(1) Explain Non-maximum suppression in Canny edge detector in your own words.
(2) How are high and low threshold values used in Canny edge detector?
(3) Generate edge maps by trying different low and high thresholds and discuss your results.

Figure 1: *Tiger* and *Pig* images

(c) Structured Edge (15%)

Apply the Structured Edge (SE) detector [3] to extract edge segments from a color image with online source codes (released toolbox in MATLAB: https://github.com/pdollar/edges). Exemplary edge maps generated by the SE method for the *Boat* image from BSDS 500 are shown in Figure 2. You can apply the SE detector to the RGB image directly without converting it into a grayscale image. Also, the SE detector will generate a probability edge map. To obtain a binary edge map, you need to binarize the probability edge map with a threshold.

(1) Please digest the SE detection algorithm. Summarize it with a flow chart and explain it in your own words (no more than 1 page, including both the flow chart and your explanation).
(2) Random Forest (RF) classifier is used in the SE detector. The RF classifier consists of multiple decision trees and integrate the results of these decision trees into one final probability function. Explain the process of decision tree construction and the principle of the RF classifier.
(3) Apply the SE detector to *Tiger.jpg* and *Pig.jpg* images. State the chosen parameters clearly and justify your selection. Compare and comment on the visual results of the Canny detector and the SE detector.



Boat | Probability edge map | Binary edge map (with p>0.8)

Figure 2: The *Boat* image and its probability and binary edge maps obtained by the Structured Edge detector

(d) Performance Evaluation (15%)

Perform quantitative comparison between different edge maps obtained by different edge detectors. The ultimate goal of edge detection is to enable the machine to generate contours of priority to human being. For this reason, we need the edge map provided by human (called the ground truth) to evaluate the quality of a machine-generated edge map. However, different people may have different opinions about the importance of different edges in an image. To handle the opinion diversity, it is typical to take the mean of a certain performance measure with respect to each ground truth, e.g. the mean precision, the mean recall, etc. Figure 3 shows the 6 ground truth edge maps for the *Boat* image. To evaluate the performance of a generated edge map, we need to identify the error. Every pixel in a generated edge map will be belong to either of the following four classes:

(i) True positive: Edge pixels in the edge map coincide with edge pixels in the ground truth. These are edge pixels the algorithm successfully identifies.

(ii) True negative: Non-edge pixels in the edge map coincide with non-edge pixels in the ground truth. These are non-edge pixels the algorithm successfully identifies.

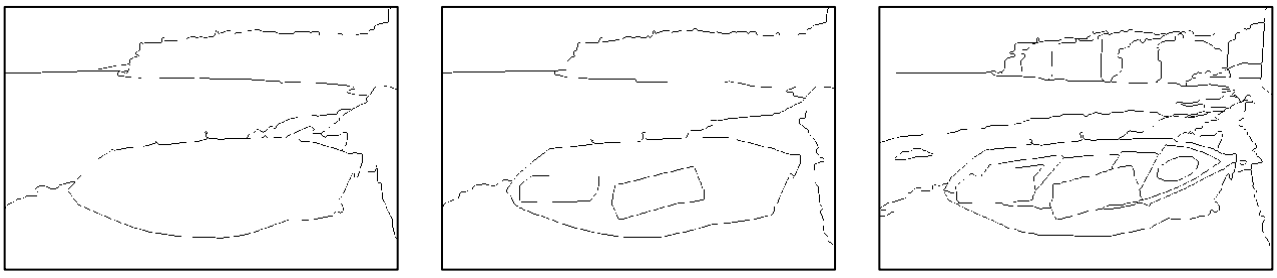
(iii) False positive: Edge pixels in the edge map correspond to the non-edge pixels in the ground truth. These are fake edge pixels the algorithm wrongly identifies.

(iv) False negative: Non-edge pixels in the edge map correspond to the true edge pixels in the ground truth. These are edge pixels the algorithm misses.

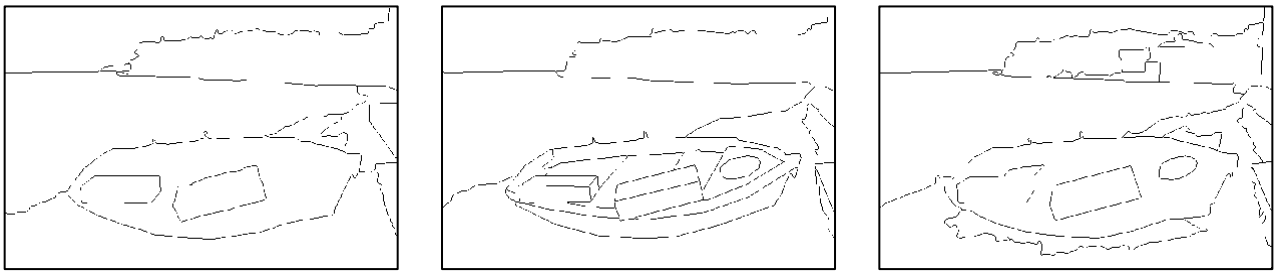
Clearly, pixels in (i) and (ii) are correct ones while those in (iii) and (iv) are error pixels of two different types. The performance of an edge detection algorithm can be measured using the F measure, which is a function of the precision and the recall.

$$\text{Precision}: P = \frac{\#\text{True Positive}}{\#\text{True Positive} + \#\text{False Positive}}$$
$$\text{Recall}: R = \frac{\#\text{True Positive}}{\#\text{True Positive} + \#\text{False Negative}} \quad (1)$$
$$F = 2 \times \frac{P \times R}{P + R}$$

One can make the precision higher by **increasing** the threshold in deriving the binary edge map. However, this will result in a lower recall. Generally, we need to consider both precision and recall at the same time and a metric called the F measure is developed for this purpose. A higher F measure implies a better edge detector.



Ground Truth 1 Ground Truth 2 Ground Truth 3



Ground Truth 4 Ground Truth 5 Ground Truth 6

Figure 3: Five ground truth edge maps for the *Boat* image

For the ground truth edge maps of *Tiger.jpg* and *Pig.jpg* images, evaluate the quality of edge maps obtained in Parts (a)-(c) with the following steps:

(1) Calculate the precision (P) and recall (R) for each GT (provided in *.mat* format) separately using the function provided by SE software package. Compute mean P and mean R **for each GT**. Average all *mean P* and *mean R* to get *overall mean precision* and *overall mean recall* for all GT. Finally, calculate the F measure (scalar) from *overall mean precision* and *overall mean recall*. Please use a table to show the precision and recall for each ground truth, their means and the final F measure (scalar). Comment on the performance of different edge detectors (i.e. pros and cons.)

(2) Similar to (1), compute the mean P and mean R **for each threshold across all GT**. For each threshold, compute mean F measure. Plot the curve of how F measure changes by threshold. Find the best F measure and discuss the plot for different edge detectors.

(3) The F measure is image dependent. Which image is easier to a get high F measure – *Tiger* or *Pig*? Please provide an intuitive explanation to your answer.

(4) Discuss the rationale behind the F measure definition. Is it possible to get a high F measure if precision is significantly higher than recall, or vice versa? If the sum of precision and recall is a constant, show that the F measure reaches the maximum when precision is equal to recall.

Problem 2: Digital Half-toning (30%)

(a) Dithering (15%)

Figure 4 *Bridge* is a grayscale image. Implement the following methods to convert it to half-toned images. In the following discussion, $F(i,j)$ and $G(i,j)$ denote the pixel at position (i,j) in the input and output images respectively. Show and compare the results obtained by the following algorithms in your report.

Figure 4: *Bridge* [4]

1. Fixed thresholding

Choose one value, *T*, as the threshold to divide the 256 intensity levels into two ranges. An intuitive choice of *T* would be 128. For each pixel, map it to 0 if it is smaller than *T*, otherwise, map it to 255, i.e.

$$G(i,j) = \begin{cases} 0 & \text{if } 0 \le F(i,j) < T \\ 255 & \text{if } T \le F(i,j) < 256 \end{cases}$$

2. Random thresholding

In order to break the monotones in the result from fixed thresholding, we may use a random threshold. The algorithm can be described as:

- For each pixel (i,j), generate a random number in the range 0 ~ 255, so called $rand(i,j)$
- Compare the pixel value with $rand(i,j)$. If the pixel value is greater, then map it to 255; otherwise, map it to 0, i.e.

$$G(i,j) = \begin{cases} 0 & if\ 0 \le F(i,j) < rand(i,j) \\ 255 & if\ rand(i,j) \le F(i,j) < 256 \end{cases}$$

A choice of random threshold could be from uniformly distributed random variables. Check your coding language documentation for proper random variable generator.

3. Dithering Matrix

Dithering parameters are specified by an index matrix. The values in an index matrix indicate how likely a dot will be turned on. For example, an index matrix is given by

$$I_2(i,j) = \begin{bmatrix} 1 & 2 \\ 3 & 0 \end{bmatrix}$$

where 0 indicates the pixel that is the most likely to be turned on, and 3 is the least likely one. This index matrix is a special case of a family of dithering matrices firstly introduced by Bayer [5]. The Bayer index matrices are defined recursively using the formula:

$$I_{2n}(i,j) = \begin{bmatrix} 4 \times I_n(i,j) + 1 & 4 \times I_n(i,j) + 2 \\ 4 \times I_n(i,j) + 3 & 4 \times I_n(i,j) \end{bmatrix}$$

The index matrix can then be transformed into a threshold matrix *T* for an input grayscale image with normalized pixel values (i.e. with its dynamic range between 0 and 255) by the following formula:

$$T(x,y) = \frac{I_N(x,y) + 0.5}{N^2} \times 255$$

where N^2 denotes the total number of pixels in the threshold matrix, and (x,y) is the location in the matrix. Since the image is usually much larger than the threshold matrix, the matrix is repeated periodically across the full image. This is done by using the following formula:

$$G(i,j) = \begin{cases} 0 & if\ F(i,j) \le T(i\ mod\ N, j\ mod\ N) \\ 255 & otherwise \end{cases}$$

Please create I_2, I_8, I_{32} threshold matrices and apply them to halftone *Bridge* image (Fig. 4). Show each of your threshold matrix and your corresponding dithered image. Compare your results and discuss the difference.

(b) Error Diffusion (15%)

Convert the 8-bit *Bridge* image to a half-toned one using the error diffusion method. Show the outputs of the following three variations and discuss these obtained results. Compare these results with Dithering Matrix method in part (a). Which method do you prefer? Why?

(1) Floyd-Steinberg's error diffusion with the serpentine scanning, where the error diffusion matrix is:

$$\frac{1}{16}\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 7 \\ 3 & 5 & 1 \end{bmatrix}$$

(2) Error diffusion proposed by Jarvis, Judice, and Ninke (JJN), where the error diffusion matrix is:

$$\frac{1}{48}\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 7 & 5 \\ 3 & 5 & 7 & 5 & 3 \\ 1 & 3 & 5 & 3 & 1 \end{bmatrix}$$

(3) Error diffusion proposed by Stucki, where the error diffusion matrix is:

$$\frac{1}{42}\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 8 & 4 \\ 2 & 4 & 8 & 4 & 2 \\ 1 & 2 & 4 & 2 & 1 \end{bmatrix}$$

Describe your own idea to get better results. You don't need to implement your idea if the time is limited. Please explain why your proposed method will lead to better results.

Problem 3: Color Half-toning with Error Diffusion (20%)

Figure 5: The *Bird* image [6]

(a) Separable Error Diffusion (4%)

One simple idea to achieve color halftoning is to separate an image into CMY three channels and apply the Floyd-Steinberg error diffusion algorithm to quantize each channel separately. Then, you will have one of the following 8 colors, which correspond to the 8 vertices of the CMY cube at each pixel:

$$W = (0,0,0),\ Y = (0,0,1),\ C = (1,0,0),\ M = (0,1,0),$$

$$G = (1,0,1),\ R = (0,1,1),\ B = (1,1,0),\ K = (1,1,1)$$

Note that (W, K), (Y, B), (C, R), (M, G) are complementary color pairs. Please show and discuss the result of the half-toned color *Bird* image. What is the main shortcoming of this approach?

(b) MBVQ-based Error diffusion (16%)

Shaked et al. [7] proposed a new error diffusion method, which can overcome the shortcoming of the separable error diffusion method. They partition the RGB color space into six Minimum Brightness Variation Quadrants (MBVQ) as shown in Fig. 2 of [7]. Then, the MBVQ-based error diffusion can be conducted as follows. Given the RGB value and the **accumulated** quantization error at a pixel located in (x, y). They are denoted by RGB(x, y) and e(x, y), respectively.

- Determine its MBVQ quadrant based on RGB (x, y);
- Find the vertex V of the MBVQ tetrahedron to closest RGB (x, y) + e (x, y) and output the value of V at location (x, y);
- Compute the new quantization error using RGB (x, y) + e (x, y) - V
- Distribute the quantized error to the future pixel error buckets e using a standard error diffusion process (e.g. the FS error diffusion).

Please read [7] carefully, and answer the following questions:

1) Describe the key ideas on which the MBVQ-based Error diffusion method is established and give reasons why this method can overcome the shortcoming of the separable error diffusion method.
2) Implement the algorithm using a standard error diffusion process (e.g. the Floyd-Steinberg error diffusion) and apply it to Fig. 5. Compare the output with that obtained by the separable error diffusion method.

Appendix:

Problem 1: Edge detection

Tiger.raw	481x321x3 (Width x Height x Band)	24-bit	color (RGB)
Pig.raw	481x321x3	24-bit	color (RGB)
Tiger.jpg	481x321x3	24-bit	color (RGB)
Pig.jpg	481x321x3	24-bit	color (RGB)
Tiger_GT.mat	(containing 5 ground truth annotations)		
Pig_GT.mat	(containing 5 ground truth annotations)		
(The following three are examples)			
Boat.jpg	481x321x3	24-bit	color (RGB)
Boat_prob.png	481x321	8-bit	gray
Boat_binary.png	481x321	8-bit	gray

Problem 2: Digital Half-toning

Bridge.raw	600x400	8-bit	gray

Problem 3: Color Half-toning with error diffusion

Bird.raw	500x375x3	24-bit	color (RGB)

Reference Images

Images in this homework are from the BSDS 500 [2] and the Online images [4][6].

References

[1] J. Canny, "A computational approach to edge detection," *IEEE Transactions on pattern analysis and machine intelligence*, no. 6, pp. 679–698, 1986.
[2] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik, "Contour detection and hierarchical image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 5, pp. 898–916, May 2011. [Online]. Available: http://dx.doi.org/10.1109/TPAMI.2010.161
[3] P. Dollár and C. L. Zitnick, "Structured forests for fast edge detection," in *Proceedings of the IEEE International Conference on Computer Vision*, 2013, pp. 1841–1848.
[4] [Online] https://unsplash.com/photos/f05TIl5AOJc
[5] B. E. Bayer, "An optimum method for two-level rendition of continuous-tone pictures," *SPIE MILE-STONE SERIES MS*, vol. 154, pp. 139–143, 1999
[6] [Online] https://pixabay.com/photos/discus-fish-fish-fauna-1943755/
[7] D. Shaked, N. Arad, A. Fitzhugh, I. Sobel, "Color Diffusion: Error-Diffusion for Color Halftones", *HP Labs Technical Report*, HPL-96-128R1, 1996.