HOMEWORK #2

Edge Detection, Morphological Processing and Digital Halftoning

- The size of every image in this assignment is 256 x 256. Each pixel is represented by 1 byte.
- Pixel values in a binary image are either 0 or 255.

Problem 1: Edge Detection (30%)

An edge map is a binary image, where 0 implies the edge, and 255 implies the background. Please perform edge detection on the two images in Figure 1 (i.e., *building.raw* and *building_noise.raw*) and show the edge map results in your report.

(a) Basic edge detection algorithms
Please implement two basic edge detection algorithms: the 1st-order derivative gradient method and the 2nd-order derivative plus zero crossing. Please discuss how you choose the threshold values which are required in each of the two algorithms.

(b) Advanced edge detection algorithms
Please implement some pre-processing and post-processing tools to improve the results from (a). For example, you might want to enhance the contrast of Figure 1(a) and perform noise removal on Figure 1(b). State your pre-processing and post-processing steps clearly in your report and explain how they help in your obtained edge detection results.


(a) building.raw

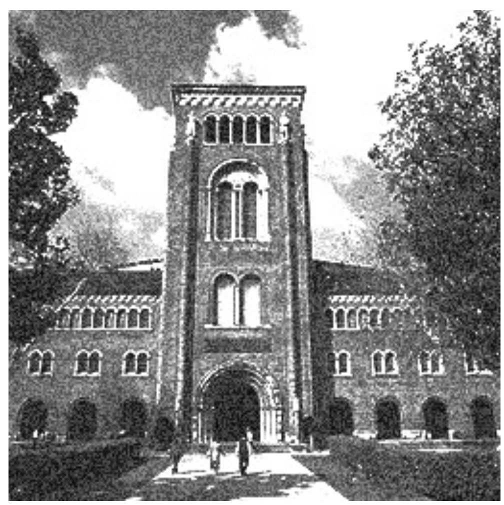
(b) building_noise.raw

Figure 1: Two images for edge detection.

Problem 2: Morphological Processing (30%)

For this problem, objects are of pixel value 0, and the background is of pixel value 255.
Please implement three morphological processing operations: shrinking, thinning, and skeletonizing. Show your converged results by applying the three operations to Figure 2 (*patterns.raw*) and Figure3 *(pcb.raw)*. A pattern table sheet (patterntables.pdf) is attached for your reference.

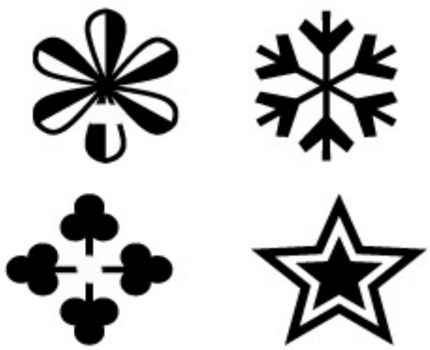

Figure 2: patterns.raw

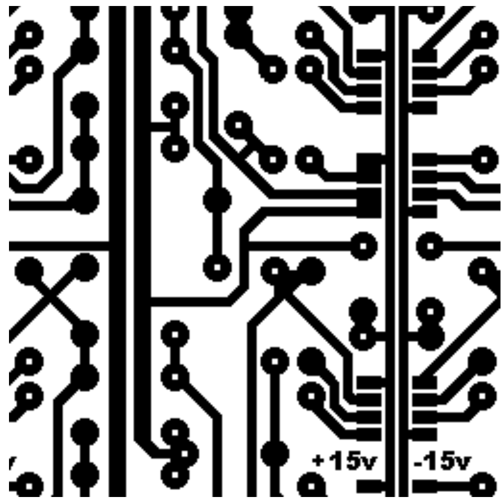

Figure 3: pcb.raw

Problem 3: Digital Halftoning (40%)

There are 256 gray levels for pixels in Figure 4 (*barbara.raw*). Please implement the following four procedures to convert *barbara.raw* to a binary image. In the following discussion, *F(i,j)* and *G(i,j)* denote the pixel of the input and the output images at position *(i,j)*, respectively. Compare the results obtained from these four algorithms in your report.

Figure 4: barbara.raw.

(a) Fixed Threshold Dithering

1. Choose one value, T, as the threshold to divide the 256 levels into two ranges. An intuitive choice of T would be 127.
2. For each pixel, map it to 0 if it is smaller than T, otherwise, map it to 255.

i.e.,

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

(b) Random Dithering

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

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

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

The build-in rand function in C/Matlab/Java/Python generate numbers in uniform distribution. Please choose two random generators (based on different distribution function), specify what are your choices, compare and discuss the difference in the dither results to justify your choice.

(c) Dithering Matrix (Pattern)

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 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 first introduced by Bayer [1]. The Bayer index matrices are defined recursively using the formula:

$$I_{2n}(x,y)=\begin{bmatrix}4*I_n(x,y)+1 & 4*I_n(x,y)+2\\ 4*I_n(x,y)+3 & 4*I_n(x,y)\end{bmatrix}$$

The index matrix can then be transformed into a threshold matrix T for an input gray-level image with normalized pixel values (*i.e.* with its dynamic range between 0 and 1) by the following formula:

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

where N^2 denotes the number of pixels 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:

$$G(i,j)=\begin{cases} 1 & if\, F(i,j)>T(i \bmod N, j \bmod N) \\ 0 & else \end{cases}$$

where G(i,j) is the normalized output binary image. Please create 2x2, 4x4 thresholding matrices and apply them to halftone the Barbara image.

If the printer is able to print 4 different intensity levels instead of 2. Design a method to generate a printer-ready image of the Barbara image. Show your best result and discuss the method you design.

(d) Implement Floyd-Steinberg's error diffusion with serpentine scanning
(details of this technique will be explained in the lecture.)

Reference
[1] B. E. Bayer, "An optimum method for two-level rendition of continuous-tone pictures," *IEEE International Conference on Communications,* vol. 1, June 11-13 1973, pp. 11-15.

Your report should include, but not limited to the following:

1. Description of your motivation
2. Description of your approach and procedures
3. Results from the provided testing images
4. Discussion of your approach and results
5. Your answer to the non-programming questions, if any
6. Your findings from your own created testing images. You should use AIGC (AI Generated Content) tools to create your own testing images for each exercise and document what you find after applying your code for processing. Please include your created image and the processed results in your report with detailed explanation and discussion for each question.

Please compress your submission into ONE .zip file.
Your submission should include the following items ONLY

Your report (.pdf format)
Compiling and executing instructions
Your source code in a subfolder

Please do NOT include any image data files or executables.