**PRAKTIKUM 9**

**Image Halftoning**

**SISTEM PENGOLAHAN CITRA**

**PROGRAM STUDI SISTEM KOMPUTER**

**SCHOOL OF INFORMATION SCIENCE AND TECHNOLOGY**

**UNIVERSITAS PELITA HARAPAN**

**DISUSUN OLEH:**

**Alfa Satya Putra, B.Sc., M.Sc.**

**Intro**

Halftoning is an image processing **to convert grayscale image into binary image** that appear to have a gray scale rendition. Halftoning is required in many present-day electronic applications such as facsimile (FAX), electronic scanning/copying, laser printing, and low bandwidth remote sensing. There are three common methods to create halftone of the image, which will be discussed in three parts below. The image to be halftoned is **house.tif** which can be downloaded from learn.uph.edu.

**Part 1 – Thresholding**

The simplest way to halftone an image is based on thresholding, i.e. two-level (one-bit) quantization. Let f(i, j) be a gray scale image, and b(i, j) be the corresponding binary image based on thresholding. For given threshold T, the binary image is computed as the following:



Tasks:

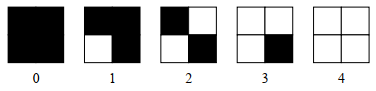
1. Download the house.tif image and read it into Matlab/Octave.
2. Write a Matlab/Octave function [**b,e]=halftoneThreshold(f,T)** that creates a binary image b and error e from original grayscale image f and threshold T based on given equation. Include this function in your lab report.
3. Compute e, the mean squared error (MSE) of the image, given by (Note: NM is number of pixels in the image):



1. Try the function using T = 108. Note the value of e in your lab report. Display the halftoned image b and include the image in your lab report.

**Part 2 - Ordered Dithering**

Human visual system tends to average a region around a pixel instead of treating each pixel individually. It is possible to create illusion of many gray levels, even though there are only two gray levels. With 2 × 2 binary pixel grids, we can represent 5 different “effective” intensity levels. Similarly for 3 × 3 grids, we can represent 10 distinct gray levels. In dithering, we replace blocks of original image with binary grid patterns.



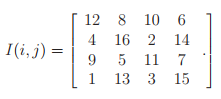
Patterns for 2x2 binary pixel grids

Ordered dithering consists of comparing blocks of the original image to a 2-D grid, known as dither pattern. Each block element is quantized using corresponding value in the dither pattern as threshold. The values in dither matrix are fixed, typically different from each other. Given index matrix I(i,j), dither matrix T(i,j) can be computed by (n2 is total number of elements in the matrix):



Tasks:

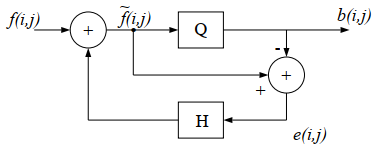
1. Write a Matlab/Octave function [**b,e]=halftoneDither(f,i)** to implement ordered dithering of size 4 to original image f. Index matrix i is given by:



1. Compute dither matrix T(i,j) using equation above.
2. For ordered dithering, it is easier to perform the thresholding of the image all at once, by creating a large threshold matrix by repeating the 4 × 4 dither pattern. For example, the command T = [T T; T T]; will increase the dimensions of T by 2. If this is repeated until T is at least as large as the original image, T can then be trimmed to be the same size as the image. Thresholding can be performed using the command **b = 255\*(f>T).**
3. Compute mean squared error e using the same function from Part 1 Step 3.
4. Note the value of e in your lab report. Display the halftoned image b and include the image in your lab report.
5. Compare output image quality and value of e with part 1’s results. What can you conclude?

**Part 3 (Bonus Point) – Error Diffusion**

In error diffusion, pixels are quantized in a specific order, and residual quantization error for the current pixel is propagated (diffused) forward to unquantized pixels. This keeps overall intensity of output binary image closer to input gray scale intensity. Figure below is a block diagram that illustrates error diffusion. Current input pixel f(i,j) is modified by past quantization errors to **give modified input f˜(i,j).** This pixel is then quantized to **a binary value by Q using threshold T**. Error e(i,j) is defined as **e(i,j) = f˜(i,j) - b(i,j)**   
where b(i, j) is quantized binary image.



e(i, j**) is diffused to “future” pixels by two-dimensional weighting filter h(i, j), known as the diffusion filter.** Modifying input pixel by past errors can be represented by following recursive relationship:

One of the most popular error diffusion method, the Floyd and Steinberg error diffusion, uses filter above.

Tasks:

1. Write a Matlab/Octave function [**b,e]=halftoneErrDiff(f,T)** to implement error diffusion to original image f with T=108 with Floyd and Steinberg
2. Initialize an output image matrix with zeros
3. Quantize the current pixel using threshold T, and place the result in the output matrix
4. Compute quantization error by subtracting binary pixel from gray scale pixel
5. Add scaled versions of this error to “future” pixels of the original image, as depicted by diffusion filter above
6. Move on to the next pixel
7. You do not have to quantize outer border of the image
8. Compute mean squared error e using the same function from Part 1 Step 3.
9. Note the value of e in your lab report. Display the halftoned image b and include the image in your lab report.
10. Compare output image quality and value of e with part 1 & 2 ’s results. What can you conclude?

References:

* <https://www.gnu.org/software/octave/>
* GNU Octave Manual
* Class Materials, Slide Week 12
* Purdue ECE 438 Lab 10b: <https://engineering.purdue.edu/VISE/ee438L/lab10/pdf/lab10b.pdf>