EE 605: Digital Image Processing - Assignment 3

Teacher: Prof. Nitin Khanna (nitinkhanna@iitgn.ac.in)

Guidelines:

- You can do this assignment in Matlab and/or Python.
- There should be only one .zip file (containing all your .m files, figures/images corresponding to results and a .pdf and corresponding .tex or .doc(x) file for the summary report). Filename shall be in the format as follows: DIP-A3-rollno.zip (For e.g. DIP-A3-16110145.zip). Submit this zip file on Google classroom.
- There shall be one report for the complete assignment 3, as .tex or .doc(x) file (with corresponding .pdf file) showing the input image, output image, discussion about results etc. corresponding to different questions. Also include the description of parameters chosen and any specific observations made. Filename for the report shall again be in format as follows: DIP-A3-report-rollno.tex or or DIP-A3-report-rollno.doc(x) and DIP-A3-report-rollno.pdf.
- Include a readme.txt file mentioning different steps for running your program and any other comments from your side.
- Make suitable assumptions as needed and clearly mention them at appropriate places in your code, readme file and report.
- In writing code for the following problems, you can use Matlab's built in function for image reading and writing (imread and imwrite), apart from these, please do not use any other function from Matlab's toolboxes related to image processing (except the functions mentioned as allowed function under a particular question). Write the code from scratch, using basic constructs like loops etc. Please confirm with me by posting a query in Google classroom if you think any other in-built function is unavoidable to use. *Note: Allowed in-built Matlab functions/constructs: same as that in Assignment-1.*
- Matlab code should be very well documented. The evaluation will also give consideration to your writing efficient code/algorithms, using dynamic and not-static declarations etc., code having proper indentation and self-explanatory comments etc.
- Unless you state otherwise, it is implicitly assumed that you wrote the code yourself without any help from your friends/online-resources. If you take some help, you must mention that explicitly in your report as well as in your Matlab code comments. Plagiarism related polices as per Institute rules will be applicable and strictly enforced.

Note: Images given in the questions are just shown for reference purpose, they are not upto scale. Feel free to let me know if you need any further clarification.

1. *Halftoning*

An 8-bit monochrome image allows 256 distinct gray levels. Modern computer monitors generally support the display of such images, however some other rendering technologies allow for much fewer gray levels. At the far end of the spectrum are devices, such as printers, that can only display two levels, black or white, for a monochrome image. This assignment will introduce a useful area of image processing called halftoning, which is the conversion of a grayscale image into a binary image. The key in this application is to exploit properties of the human visual system to give the impression of a continuous tone image even though only two levels are present in the rendering. Halftoning is required in several electronic applications such as facsimile (FAX), electronic scanning and copying, and laser and inkjet printing.

This assignment will emphasize two halftoning techniques known as ordered dithering and error diffusion. An important note about rendering: The halftone images you produce in this assignment need to represented exactly pixel-by-pixel to get the proper visual effect. Therefore, when viewing images within Matlab, issue the truesize command just after the image command in order to map 1-to-1 the image pixels to display pixels. Otherwise, the image will likely interpolate on your display and obscure the intended binary result. For the same reason, your halftone results should be written out to TIFF files (using imwrite, NOT by a figure export) and submitted independently along with your report.

To help facilitate comparison between the various halftone images in this lab, we will use an image fidelity metric that incorporates a simple model of the human visual system. This requires some background, so we will begin with a short discussion of image fidelity.

Image Fidelity Metrics

As we explore halftoning methods in this assignment, we will also assess quantitatively how well the halftone images reproduce the original grayscale images. We will see that simply computing the average squared pixel error is a fairly useless metric in this application because it fails to take into account the spatial blurring response of the human visual system. This blurring property is precisely what halftoning methods exploit to make a binary pattern appear as a continuous gray tone.

Let f be the original grayscale image and g be its halftoned version, both having M rows and N columns. Then, you will be comparing output of your systems (g) with original image (f) using the following metrics:

1. $$\alpha_1 = \frac{1}{MN}\sum_{x,y}(f(x,y) - g(x,y))$$

2. $$\alpha_2 = \frac{1}{MN}\sum_{x,y}|f(x,y) - g(x,y)|$$

3. First transform f using $f_1(x,y) = 255\left(\frac{f(x,y)}{255}\right)^{\gamma}$ and similarly obtain g_1, then

   $$\alpha_3 = \frac{1}{MN}\sum_{x,y}|f_1(x,y) - g_1(x,y)|.$$

   For estimating values of α_3, use $\gamma = 2.2$, while use $\gamma = 1/2.2$ for estimating value of α_4.

4. Apply Gaussian low-pass filter of size 7×7 to both f_1 and g_1 to obtain f_2 and g_2. The, obtain f_3 (and g_3) using $f_3(x,y) = 255\left(\frac{f_2(x,y)}{255}\right)^{(1/3)}$. Finally estimate

   $$\alpha_5 = \frac{1}{MN}\sum_{x,y}|f_3(x,y) - g_3(x,y)|.$$

   For estimating values of α_5, use $\gamma = 2.2$, while use $\gamma = 1/2.2$ for estimating value of α_6.

To perform above tasks, please write a function fidelity_rollno(f,g) which will return above six measures of fidelity between images f and g. Note that in calculating any of the above six metrics, the final difference should be estimated between two matrices of data type double and with values in the range $[0, 255]$.

Thresholding and Random Noise Binarization

The simplest method of converting a grayscale image to a binary image is by thresholding. Let f be a grayscale image, and g_1 be the corresponding binary image based on simple thresholding using $T = 127$ (values less than T go to 0, otherwise 255).

Read the image provided with this assignment campus.png using imread and display it, alongwith the output g_1 corresponding to it.

Add uniform additive noise (uniform over $[-128, 128]$) to f to obtain f_1. Let g_2 be the corresponding binary image based on simple thresholding using $T = 127$ (values less than T go to 0, otherwise 255). Display g_2 using imshow. Notice that even though the resulting binary image is somewhat noisy, the false contouring has been dramatically reduced, and with enough blurring (e.g. looking at it from a distance) it gives the impression of having several gray levels.

Ordered Dithering

The goal in halftoning is to give the impression of grayscale tones while using only black and white pixels. Although the random thresholding technique described above can produce this effect, it is not often used in real applications since it yields very noisy results. Here, we will describe a better class of halftoning techniques known as ordered dithering.

Because the human visual system tends to average a region around a pixel instead of sensing each pixel individually, we can create the illusion of many gray levels in a binary image that in actuality only contains two gray levels. Using 2×2 binary pixel grids, we can represent 5 different "effective" intensity levels, as illustrated in Figure 1. Similarly for 3×3 grids, we can represent 10 distinct gray levels. In dithering, we replace blocks of the original image with these types of binary grid patterns.

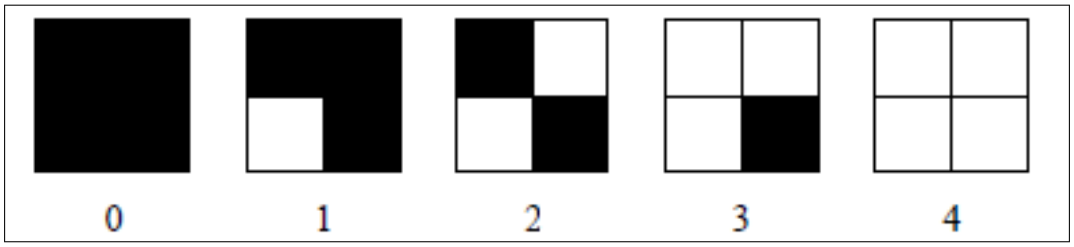

Figure 1: Five different patterns of 2×2 binary pixel grids

Ordered dithering consists of comparing blocks of the original image to a 2-D grid of thresholds called a dither pattern. Each element of the original block is quantized according to the corresponding threshold value in the dither pattern. The values in the dither matrix are fixed, but are typically different from each other.

A dither matrix can also be defined by a so called index matrix. The index matrix determines the order in which dots are "turned on" (change from black to white) as the image becomes darker (greater absorptance). For the example a index matrix is given by $I_2 = [1, 2; 3, 0]$, where 0 indicates the first pixel to turn on, and 3 indicates the last pixel to turn on. This index matrix is a special case of a family of dither matrices first defined by Bayer [1], which are defined recursively by Figure 2, where I_{2n} is the new $2N \times 2N$ matrix and I_n is the old $N \times N$ matrix.

$$I_{2n} = \begin{bmatrix} 4 * I_n + 1 & 4 * I_n + 2 \\ 4 * I_n + 3 & 4 * I_n \end{bmatrix}$$

Figure 2: Bayer dither

For each index matrix, there is a corresponding threshold matrix which is used to halftone the image. The threshold matrix can be determined from the index matrix I by the relationship, $T(i, j) = 255 \frac{I(i,j)+0.5}{N^2}$, where N^2 is the total number of elements in the matrix. This produces thresholds evenly spaced between 0 and 255. The halftoning is performed by thresholding each pixel in the original image according to the values in T(i, j).

Since the image is usually much larger than the threshold matrix, the dither pattern is repeated periodically, or tiled, across the full image.

Create Bayer threshold matrices of sizes $2 \times 2, 4 \times 4$, and 8×8. Generate three different halftone images for campus.png by applying these three dither patterns to the original image, f (call them g_3, g_4, g_5). When displaying the halftone images in Matlab, remember to use the truesize command to prevent interpolation of your binary image. Note that it is sometimes better to view the halftone results from a slight distance.

Error Diffusion

Another class of halftoning techniques are called error diffusion. In this method, the pixels are quantized in a specific order (raster ordering is commonly used), and the residual quantization error for the current pixel is propagated (diffused) forward to local unquantized pixels. This keeps the local average intensity of the binary image close to the original grayscale image.

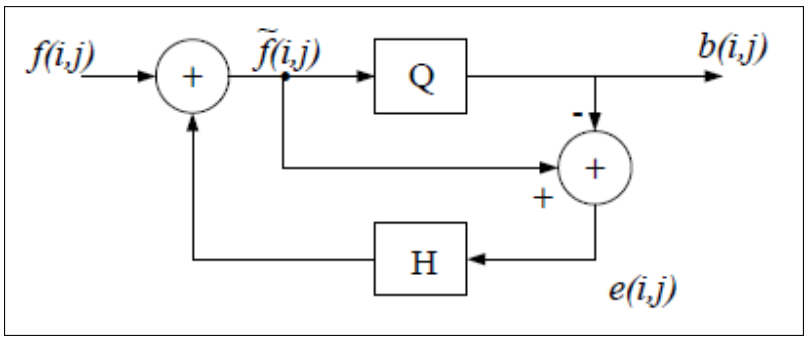

Figure 3: Block diagram of the error diffusion method

Figure 3 is a block diagram that illustrates the error diffusion algorithm. The current input pixel f(i, j) is modified by adding certain past quantization errors, producing a modified input. This pixel is then converted to a binary value by the quantizer Q, using some threshold T . The error e(i, j) of quantizing the current pixel is diffused to "future" pixels by means of a two-dimensional weighting filter h(i, j), known as the diffusion filter.

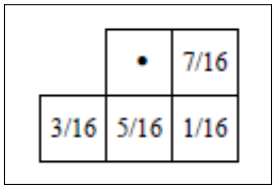

Figure 4: Point Spread Function of the error diffusion filter proposed by Floyd and Steinberg

A very popular error diffusion method, proposed by Floyd and Steinberg [2], uses the diffusion filter shown in Figure 4. Since the filter coefficients sum to one, the local average value of the quantized image will be equal to the local average grayscale value.

Apply the error diffusion technique to campus.png using a threshold $T = 127$.

A straight forward implementation of the error diffusion algorithm is detailed in the following steps, performed on each pixel in raster order:

1. Initialize an output image matrix with zeros.
2. Quantize the current pixel to 0 or 255 using using the threshold T, and place the result in the output matrix.
3. Compute the quantization error by subtracting the binary pixel from the grayscale pixel.
4. Add scaled versions of this error to "future" pixels of the original image, according to the diffusion filter of Figure 4.
5. Proceed to the next pixel.

Display the result in Matlab using image (remember to use truesize), and compare this to the original image. Again, it is best to view the results from a slight distance.

Export all the halftone results (g_1 to g_6) to TIFF files using imwrite and create a Table listing all the six fidelity measures for all these halftone results.

Re-run the complete code for this question on two more images captured by you and submit all the above results on those two images, in addition to the image provided, campus.png. You should carefully capture these two images of such scenes where you expect the least and most differences between g_1 to g_6.

References

[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.

[2] R. W. Floyd and L. Steinberg, "An adaptive algorithm for spatial greyscale," Journal of the Society for Information Display, vol. 17, no. 2, pp. 75–77, 1976.

[3] campus.png `https://drive.google.com/file/d/1wxWWt73cpnxVyp2BEyaZLhao4T8g2ODB/view?usp=sharing`