ECES 680 - Winter 2016 Drexel University
Instructor: Matthew Stamm

Assignment # 1

This assignment is due by 11:59 p.m. on Friday, January 29. All reports should be submitted as PDFs. Only one report must be submitted per team, but all team member's names must appear on the report.

Part 1

In class, we discussed a method to perform image contrast enhancement known as histogram equalization. There are many other ways to perform contrast enhancement that are commonly used in practice. One important contrast enhancement operation is known as *gamma correction*.

To perform gamma correction, a user modifies each pixel value $u \in \{0, \dots, 255\}$ of a grayscale image using the following mapping

$$v = 255\left(\frac{u}{255}\right)^{\gamma} \tag{1}$$

to obtain the modified pixel value y. The parameter γ is a user specified constant that shapes how the image's pixel value histogram is stretched. The value of gamma is always strictly greater than zero.

- Write a Matlab function that performs gamma correction on a grayscale digital image. This function should accept an image (already read into Matlab) along with the value of γ as inputs and produce a Matlab variable representing the image in unsigned 8-bit integer format as an output. Please comment your code and append it to your report.
- Use your gamma correction function to perform contrast enhancement on the image **pout.tiff**. What happens if $\gamma > 1$? What happens to the image and its pixel value histogram if $\gamma < 1$? What happens if $\gamma = 1$?
- Use your gamma correction function to enhance the low contrast image **moonPhobos.tiff**. Try several different values of γ and report the value that you feel works best. Be sure to include your contrast enhanced image and its pixel value histogram in your report. Additionally, use Matlab's histogram equalization function *histeq* to enhance the image. Compare the results you obtain using gamma correction with the results you obtain using histogram equalization.

Part 2

High-boost filtering is an image enhancement technique that is used to sharpen an image. It works by extracting first extracting the high frequency content $g(x, y)$ from $f(x, y)$. This is then scaled and added back to the original image to obtain the sharpened image $f_s(x, y)$ according to the equation

$$f_s(x, y) = f(x, y) + \alpha g(x, y) \tag{2}$$

where α is a user specified scaling constant. In many cases, the Laplacian filter

$$\begin{bmatrix} 0 & -0.25 & 0 \\ -0.25 & 1 & -0.25 \\ 0 & -0.25 & 0 \end{bmatrix} \tag{3}$$

is used to obtain $g(x, y)$.

- Write a Matlab function that sharpens an image through high-boost filtering using the Laplacian filter. This function should accept an image (already read into Matlab) along with the value of α as inputs and produce a Matlab variable representing the image in unsigned 8-bit integer format as an output. Please comment your code and append it to your report.

- Use your sharpening function to sharpen the image **moon.tiff**. Try several different values of α and report the value that you feel works best. Be sure to include your sharpened image in your report.
- Now use your sharpening function to sharpen the blurry image **outoffocus.tiff**. Try several different values of α and report the value that you feel works best. Be sure to include your sharpened image in your report. Is it possible to completely recover the original in-focus image? Are there any downsides or unintended artifacts introduced by sharpening the image?

Part 3

An noisy image can be denoised using several different filters. In class, we discussed both the averaging and the median filter.

- Write a Matlab script to denoise the images **peppersNoise1.tiff** and **peppersNoise2.tiff** using both the median filter and the averaging filter. Examine the effect of using different filter window sizes including 3×3 pixels and 5×5 pixels. From your results, what are the advantages or disadvantages of each filter? What are the advantages or disadvantages of changing the filter size? Include your denoised images in your report.
- In many cases, edges must be extracted from a noisy image. If this is necessary, it is important to denoise the image first. Write a Matlab script that generates an edgemap from the 3×3 averaging filtered and median filtered versions of **peppersNoise1.tiff**. Create the edgemap by first calculating the image's gradient using the Sobel filters
$$S_X = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad S_Y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \tag{4}$$
to approximate the gradient in the row and column directions, then by comparing the magnitude of the gradient to a threshold. Use the same threshold for both images when generating the edgemap. Please comment your code and append it to your report. What difference do you see between the edgemaps of each filtered image? Which filter has better edge preserving properties?

Part 4

JPEG is one of the most widely used image compression standards.

- Use the Matlab function *imwrite* to save the files **peppers.tiff** and **baboon.tiff** using quality factors of 90, 70, 50, 30, and 10. Compute the PSNR between each original image and its JPEG compressed version at each quality factor. Additionally, record the file size of the image when it is saved at each quality factor. What is the relationship between the image's file size and its quality? What distortions are introduced by JPEG compression? Why do you think they occur? At what quality factor do these distortions become unacceptably strong?
- Write two Matlab function that implement your own JPEG-like encoder and decoder. The encoder should follow the following process:
  1. Segment the image into 8×8 pixel blocks.
  2. Compute the DCT of each block.
  3. Quantize these DCT coefficients using a user specified quantization table Q.
  4. Reorder each block of quantized DCT coefficients into a one-dimensional sequence using zig-zag scanning. You can use *ZigzagMtx2Vector.m* that is provided to you to perform zig-zag scanning and use *Vector2ZigzagMtx.m* for reconstructing the matrix from a zig-zag scanned sequence.

5. Encode the resulting sequence. For Entropy Encoding, use the *JPEG_entropy_encode.m* module provided. This function will read a matrix, in which each row represents a vectorized DCT block, write a bit stream whose filename is always named as **JPEG.jpg**, and return the length of this file. *JPEG_entropy_encode.m* is an interface for generating a text file, *JPEG_DCTQ_ZZ.txt*, and running *jpeg_entropy_encode.exe*. For the entropy decoding, use *JPEG_entropy_decode.m*, which performs the inverse functionality.

The decoder should reconstruct the image by performing each of these steps in reverse. Please comment your code and append it to your report.

- Use your JPEG-like encoder to encode the image **peppers.tif**. First do this using the standard JPEG luminance quantization table

$$\begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix} \tag{5}$$

Record the image's file size and the PSNR between the original and decompressed image. Next, try changing the quantization table and encoding the image. Is it possible to achieve both a lower file size and a higher PSNR? Include the quantization table that you design in your report.