EE 133 - Digital Image Processing
Department of Electrical and Computer Engineering
Tufts University Spring 2017
Problem Set #1

Distributed: Jan. 19, 2017 Due: Feb. 2, 2017

Problem 1.1

As described in class, the pixels of image $f(r, c)$ of dimensions R by C may be uniquely ordered to form a vector $f(m)$ of dimension $M = RC$ using the lexicographic ordering illustrated in Fig. 1.

(1,1)	(1,2)	(1,3)	(1,4)
(2,1)	(2,2)	(2,3)	(2,4)
(3,1)	(3,2)	(3,3)	(3,4)
(4,1)	(4,2)	(4,3)	(4,4)

1	5	9	13
2	6	10	14
3	7	11	15
4	8	12	16

Figure 1: Pixel ordering options for 2D images

(a) Write a pair of functions that translate between (r, c) and m. The first function should take as input r, c, R, and C, and return the corresponding value of m. The second function should take m, R, and C and return the corresponding r and c.

(b) MATLAB performs this lexicographic operation by the colon function `f1D = f(:)`, where f is a 2D array and f1D is the corresponding vector. Verify that your functions produces the same results by using, as an example, the 2D MATLAB array `f = reshape(1:10,2,5)`.

(c) Verify the lexicographic operation and its inverse by "vectorizing" and "un-vectorizing" the image of the modified Shepp-Logan phantom generated by the MATLAB function `phantom ('Modified Shepp-Logan', 32)`.

Problem 1.2

Write a function that finds the distance transform from a set of pixels, S. Your function should take as input

(a) A binary image $f(m, n) \in \{0, 1\}$ where $S = \{(m, n) | f(m, n) = 1\}$.

(b) A flag indicating which distance metric to use. The options should be D_E, D_4, and D_8.

The function should return an image the same size as f whose pixel values are the distances from the set S. Test your code on a 50×50 images that is zero everywhere except for ones at locations $(10, 10)$, $(40, 10)$, and $(25, 40)$. Verify that the distance functions look like those in Fig. 2

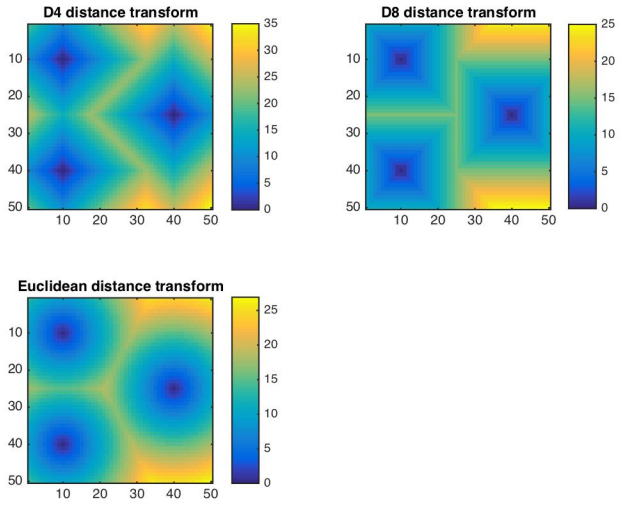

Figure 2: Distance fields for Problem 2

Problem 1.3

One way to create an image of infinite size from a finite size one is though the use of a periodic extension. Suppose we are given an image with M rows and N columns, $f(m,n)$. We define the period extension $\tilde{f}(m,n)$ as

$$\tilde{f}(m,n) = f(m \bmod M, n \bmod N) \tag{1.3.1}$$

where e.g., $m \bmod M$ means m modulo M. See this article for a definition of the modulo operation. This technique for extending a finite image will prove useful when we talk about Fourier transforms and convolution.

(a) Whereas it is not clear what $f(m,n)$ is for m outside of the range 1 to M and n outside of the range 1 to N, $\tilde{f}(m,n)$ is now defined for *every* integer pair (m,n). Explain this by sketching the periodic extension of a 3×5 image

$$f(m,n) = 2m - n \quad \text{with} \quad m = 1,2,3;\ n = 1,2,3,4,5.$$

(b) When considering the period extension of an image, the notion of a distance must be changed a bit. For example, in Fig. 1, the D_4 distance between pixel $(2,1)$ and $(2,4)$ is now 1 and not 3. Repeat Problem 2 for periodic images. How does Fig. 2 change?

Problem 1.4

For this problem you need to write a function to compute the histogram of the gray values in an image. The input for the function should include the image itself as well as N_g, the number of possible gray levels in the image (e.g., 256 for an 8 bit image) and the output should be a vector of length N_g whose i-th entry is the number of times graylevel i occurred in the image. Verify the correctness of your codes using the same 3×5 image as in Problem 3. Test also on one or two of the JPEG images `aircraft`, `beach`, `butterfly`, `city`, and `palermo` included with the problem set.

Problem 1.5

Implement your own histogram equalization method and verify that it works as intended.

Problem 1.6

While Histogram Equalization (HE) is relatively easy to implement, over the years, people have found that it does not always perform as well as they would like and have developed a wide range of alternative methods that are themselves based on HE. In the materials included in this problem set is a `paper3.pdf` a paper published in 2007 describing one such method. The basic idea of these authors is to break the overall histogram into pieces, perform HE on each, and assemble the results together into a single grayscale transformation. Your job here is to implement their method and see how well it works. As described in Section II, parts A and B of the paper, their technique starts by identifying peaks in the histogram of the input image. While they do provide a description of the method they use, peak finding can be a bit tricky. Please feel free to identify by hand the vector of peaks for each image on which you test your code if you do not wish to automated this part of the processing. In addition to implementing the method in the paper, please do the following:

(a) Test the method on the following JPEG images included with this problem set: `aircraft`, `beach`, `butterfly`, `city`, and `palermo`.

(b) Compare the results to regular HE.

(c) Explain what the factor $\log_{10} M$ in equation (3) of the paper really does. How necessary is it at least for the images you are using for this problem?

(d) For many images, it seems as though HE results in too much darkening. The approach in this paper seems to be an improvement. An alternate might be to lighten the result of HE using some other, parametric graylevel transformation. For this part of the problem see if this intuition can be made to work. Can you find some type of transformation (we discussed a few in class) which can improve upon HE? You do not need to worry about automatically setting the parameters, but rather can tune them by hand just to prove the idea works.

Problem 1.7

(a) Read section 3.9.2 of the text and write function that rotates and translates an image. The function should take as input

- An image to be transformed
- An angle for rotation
- A real number indicating the left/right shift
- A real number indicating the up/down shift

and produce as output the "warped" form of the input image. Be sure to document your function making e.g., whether a positive angle corresponds to clockwise ot counterclockwise rotation, whether positive shifts are up/right or down/left, etc.

(b) Read sections 3.9.3 and 3.9.4 as well. Say we have an image with the x, y coordinate system like in Figure 1.5 of the text. Consider the transformation

$$x' = x + a(y - y_c)$$
$$y' = y$$

where y_c is half the number of rows in the image and a is some number. Build a code to apply this transformation to an image. Be sure to use values of a that produce "reasonable" results. You should see that this transformation basically shears the image, but the equations of the transformation above are not those a shear as defined by equations (3.125) and (3.126) in the text. What is going on?