ECE 6364 Spring 2016 HW 3 Due 2/9

Problem 1. Fundamentals of Digital Image Processing - Jain: Problem 2.14 a

Problem 2. Consider the simple matrix $\underline{\mathbf{A}} = \frac{1}{\sqrt{8}}\begin{bmatrix} 2 & 2 \\ 2 & -2 \end{bmatrix}$. Is the matrix $\underline{\mathbf{A}}$: a) symmetric? b) toeplitz? c) circulant? d) singular or non-singular? e) invertible?

Problem 3. Fundamentals of Digital Image Processing - Jain: Express the 2-D convolution of **problem 2.5a.ii** as a doubly Toeplitz block matrix operating on a 6x1 vector obtained by column ordering of $x(m,n)$. (as in Problem 2.16-Jain, but just do part 2.5a.ii).

Problem 4. Linear convolution of an MxN image with a PxQ mask generates a (M+P-1)x(N+Q-1) output image. However, when **modeling** the input/output of an image sensor, we typically trim the output image to be the same size as the input image. We simply omit the output pixels formed when the center-pixel of the mask is not over-lapping any pixel of the input image. As you know from homework, if we row or column order the input and output images, we can form the simple matrix-vector equation to represent this convolution as $\vec{\mathbf{g}} = \underline{\mathbf{H}}\,\vec{\mathbf{f}}$ where $\vec{\mathbf{g}}$ and $\vec{\mathbf{f}}$ are row or column ordered MxN images denoting, respectively, the known "blurred" image $\vec{\mathbf{g}}$ and the unknown undegraded image $\vec{\mathbf{f}}$. Matrix $\underline{\mathbf{H}}$ is the blur operator whose elements are, each, equal to an element of the convolving mask or equal to zero. The non-zero elements in each row of matrix $\underline{\mathbf{H}}$ will contain the values of the mask as they overlay pixels of the input image in a flip-and-shift view of convolution, where we flip-and-shift the mask not the image.

1) Right-click on the cameraman image below and save it as a *.tif image on your computer.

2) Write a matlab program to open the 256x256 pixel cameraman image on your computer and subsample it to size 128x128. There are more than one way to do implement subsampling.
3) Form a mask by sampling a 2-D Gaussian function whose standard deviation is 0.8 pixel widths. What PxP size mask should you use? Surely, choose odd numbers for P so that you sample the Gaussian at (x,y)=(0.0,0.0). Normalize the mask to sum to 1.0;
4) Convolve the mask with the cameraman image to form a blurred image the same size 128x128 as the reduced size cameraman image. Do you need to re-scale the blurred image 0-255? Should you?
5) Look closely at the edges of the output image in 4) . You should notice that the blurred image is unusually dark near the very outer edges of the image. You could fix this by normalizing the mask to sum to 1.0

whenever part of the flipped-and-shifted mask lies outside the region of the input image. It only takes a bit of programming. Instead of doing that programming, go to step 6).

6) Now row or column order the 128x128 cameraman image. Then form a matrix $\underline{\mathbf{H}}$ of size (MN)x(MN) and fill-in its elements such that $\vec{\mathbf{g}} = \underline{\mathbf{H}}\vec{\mathbf{f}}$ generates the same output image as in step 4) above. Compute the blurred image $\vec{\mathbf{g}} = \underline{\mathbf{H}}\vec{\mathbf{f}}$ using a matrix-vector multiplication and note the amount of time it takes to execute. Most of the compute-time is spent multiplying zeroes in $\underline{\mathbf{H}}$ times non-zero values in $\vec{\mathbf{f}}$. The blurred output image of the cameraman is still unusually dark around the edges and should look exactly the same as in step 4).
7) Take the matrix $\underline{\mathbf{H}}$ from 6) above and normalize each row of the matrix so that it sums to 1.0. Now re-compute $\vec{\mathbf{g}} = \underline{\mathbf{H}}\vec{\mathbf{f}}$ using a matrix-vector multiplication. The outer portions of the output image are no longer unusually dark.

Place the images in a Word document, write up a one paragraph report, and turn the hardcopy in with your HW.

```
% ece 6364
clear
% Import cameraman image
filename = 'cameraman.tif';
[Im,map] = imread(filename);
if ~isempty(map)
    Im_intensity = ind2gray(Im,map);
else
    if size(Im,3)==3
        Im_intensity = rgb2gray(Im);
        Im_intensity = ind2gray(Im_intensity,gray(256));
    else
        Im_intensity = ind2gray(Im,gray(256));
    end
end

figure(1),imshow(Im_intensity)
```