**Lab/Homework #7**

Problem 1 - Some 1-D ffts of simple functions.

Create a 1-D sequence of 1024 zeroes. Assuming that the 513th point in the sequence is the origin of the axis, set the central several values to 1, forming a rect() function with the differing widths as described below. Make sure that the number of points above 513 is the same as the number below 513, so that the function is symmetrical.

1. Begin with a width of 7 points, and plot the rect() function and its transform. Since the function is real and symmetric, the transform will be real and symmetric, so that you can plot the real part of the transform only (you can use the matlab real() function). Don't forget the proper fftshift calls.

1. Double the width of the rect to 15 points, and repeat.

1. Double again to 31 points, and repeat.

1. Now repeat the same steps but using a triangle function. Let the triangle function peak at 0 with height 1, and scale down to 0 linearly at the width values in the previous problem.

Make sure the above spectra are plotted with the zero frequency value in the center (by using fftshift).

Problem 2 - 2-D transforms.

Create a 1024 by 1024 array of zeroes. Assuming that zero frequency is at (513,513), create a square array 7 points on a side of magnitude 1 centered in the array.

1. Calculate the 2-D transform using fft2(). Plot the absolute value of the output array (that is, abs(a) ) as an image. Scale and display. Also plot a cut along one of the coordinate axes to view the transform magnitude as a function of frequency.

1. Repeat for a 31 by 31 central square of 1's.

1. Now, create a circular region of 1's of diameter 16 centered in the array. Plot and submit its transform.

As before, be sure to put zero frequency in the middle.

Problem 3 - A low-pass filter.

Copy the file lab7prob3data from the web data area to your directory. This file is 512 by 512 points in size, and is a noisy photograph. Note: you will want to familiarize yourself with the ‘fftshift’ command in Matlab to view the image spectra. This command puts the origin of a frequency array in the center rather than at the edge.

1. Low-pass filter this function by applying a mask saving only the central Fourier coefficients. Set the mask width to 255 and create the filtered image. Display and submit.

1. Now low-pass filter the image with the mask width only 127. Display and submit. Do either of these filtering operations reduce the noise in the image without causing too much degradation in signal ?

1. Try a few other sizes of the mask and pick the size you think gives the best trade-off here. Display and submit your optimal image.

Problem 4 – Sharpening a blurred image.

Copy the file lab7prob4data from the web data area to your directory. This file is also 512 by 512 points in size.

1. Display the image and note its appearance. The image should look like it was taken from a defocused camera.

1. We know that a blurred image is lacking in high-frequency Fourier components, so we will try to improve this image by boosting the level of frequencies in the higher part of the spectrum. Create a frequency domain filter function of the following form:

g(q) = exp(-q^2/(2\*sigma\*sigma))/sigma/sqrt(pi)\*0.99+0.01

where q is the Fourier coefficient that is the transform variable of the spacedomain variable r. Along the u and v axes, q will range from from –256 to

+255. Plot g(q) over this range for several values of sigma between 5 and 50.

1. Note that the above function is greatest near zero frequency, and decreases to 0.01 near the ends of the array. Using this function as a two-dimensional quantity, scale each value in the Fourier transform of the image by the reciprocal of g(q) at that point, thus increasing the contribution from the highfrequency components. Try a few values of sigma from the range above and pick the value that gives the sharpest image without too much other distortion.

Display and submit your optimal image and one from another value of sigma. Note: If you code this problem using the ‘meshgrid’ command in matlab to generate the 2-d functions, the code will be faster. Otherwise use nested loops.

Problem 5 – Image Manipulation in the Frequency Domain.

*Note: Throughout this question, you are free to use the function imagesc.*

1. Download, read in, convert to grayscale, and display lab7prob5data from the class website. This is a jpg so you can use the function *imread*.

1. Take the 2D FFT and display the log magnitude (log(abs(img))) of the output. Comment on the features you see in the frequency domain and how they might correspond to features of the image in the spatial domain.

We can see that patterns in an image are often clearly visible in the frequency domain. Can we manipulate the frequency domain in a simple way to add interesting pattern effects to an image?

1. Read in an image of your choice, convert it to grayscale, crop it to a size that is a power of 2 (FFT works better when the input data has dimensions that are powers of 2), and display it. Take the 2D FFT (remember the proper shifts) and display the log magnitude of the output.

1. By manipulating only a few pixels in the frequency domain, add a 2D cosine pattern to your original image. Display the absolute value of the new patterned image when you’re done, and explain which pixels you modified and why.

1. Make another small change to the frequency domain image to alter the previous pattern or create a new one. For example, increase the frequency of the cosine, rotate the cosine, add another cosine orthogonal to the first one, add a high pass filter, etc. Again display the patterned image and explain the design choice you made.