ECE 366 Honors Project

Fall 2017

For the following two problems, please turn in your handwritten answers along with your MATLAB code with comments and your MATLAB output files. Make sure that your plots are clearly labeled.

1. [50] In this problem, you will extend the concepts of convolution, filtering and Fourier transform from one-dimensional signals to two-dimensional signals, e.g. images. There are some grayscale sample images (Lenna and Square) on D2L for you to test your code on.
    a. [15] You will implement a 2D-convolution algorithm with different masks to blur a given image. Convolution mask is a matrix usually of size 3x3 or 5 x5. Some examples of convolution masks include Box blur $\frac{1}{9}\begin{bmatrix}1 & 1 & 1\\1 & 1 & 1\\1 & 1 & 1\end{bmatrix}$ and Gaussian blur $\frac{1}{16}\begin{bmatrix}1 & 2 & 1\\2 & 4 & 2\\1 & 2 & 1\end{bmatrix}$. Convolution can be roughly transcribed as producing an output image by adding each element of the image to its local neighbor pixel values, weighted by the mask. Therefore, it is not possible to compute convolution on the image boundary. The size of the boundary is given by the mask size. For mask of size 3×3 the border is 1 pixel, for mask of size 5×5 the border is 2 pixels. Implement a 2D-convolution algorithm in MATLAB using different blurring functions: Box, Gaussian and Laplacian blur. Display the blurred images and comment on the differences between the different blurring mask functions and the effect of the size of the mask.
    b. [20] In this experiment, you will consider the effects of additive noise and the use of Fourier transform to remove noise. The noisy image available from D2L has been generated by adding noise in the form of a cosine function. If we denote the original image as f(x,y), the noisy image can be denoted as f(x,y)+n(x,y) where n(x,y) is a 2-D cosine function. Using frequency domain filtering, devise a procedure for removing the noise.
    c. [15] In this experiment, you are going to examine the importance of magnitude and phase. First, compute the FFT of the Lenna image. Then, set the phase equal to zero, and take the inverse Fourier transform, i.e. set the real part to the magnitude of the image and the imaginary part to zero. The resulting image should look nothing like the original. Explain your results. Let the phase be the original one and set the magnitude equal to one and take the inverse FT, e.g. set the magnitude equal to one, set the real part to cos(theta) and the imaginary part to sin(theta). Rescale the pixel values after the inverse FT has been taken. Explain your results.

2. [50] In this exercise, you will consider the touch-tone system on the telephone. The sound you hear when you push a key is the sum of two sinusoids. The higher frequency sinusoid indicates the column of the key on the touch-pad and the lower frequency sinusoid indicates the row of the key on the touch-pad.

Digit	ω_{row}	ω_{column}
0	0.7217	1.0247
1	0.5346	0.9273
2	0.5346	1.0247
3	0.5346	1.1328
4	0.5906	0.9273
5	0.5906	1.0247
6	0.5906	1.1328
7	0.6535	0.9273
8	0.6535	1.0247
9	0.6535	1.1328

Table 1: Discrete frequencies for touch-tone signals sampled at 8192 Hz.

   a. [5] Create row vectors `d0` through `d9` to represent all 10 digits for the interval $0 \leq n \leq 999$. Listen to each signal using `sound`. For example, `sound(d2,8192)` should sound like the tone you hear when you push a '2' on the telephone.
   b. [6] The function `fft` can be used to compute N samples of the Fourier transform of a finite length signal at frequencies $\omega_k = {}^{2\pi k}/_{N}$. Use `fft` to compute the spectrum of the digits '2' and '9' for 2048 points. Define `omega` to be a vector containing ω_k for $0 \leq k \leq 2047$. Plot the magnitudes of the Fourier transforms for these signals, and confirm that the peaks fall at the frequencies specified in the Table given above. Generate appropriately labeled plots of the Fourier transform magnitudes for these two digits.
   c. [5] Define `space` to be a row vector with 100 samples of silence using `zeros`. Define `phone` to be your own 7 digit phone number by appending the appropriate digits and space. Play your phone number using `sound`.
   d. [8] Load `touch.mat` from D2L. If the file loaded correctly, you should be able to list the names of the variables. The vectors `x1` and `x2` contain sampled versions of the sequence of touch-tones representing two different phone numbers with 7 digits of 1000 samples each, separated by 100 samples of silence. Using `fft`, compute 2048 samples of the Fourier transform for each digit of `x1`. You will need to segment the signal into seven digits and apply `fft` to each digit separately. Apply `fftshift` to the output of `fft` to rearrange the samples of the Fourier transform so that they correspond to the range $[-\pi, \pi]$. Define `X11` through `X17` to contain these samples of the Fourier transform for each digit. To determine the digits of the telephone number represented by `x1`, plot the magnitude of the Fourier transforms and compare the peak frequencies of the signals with those shown in the Table. Repeat this for signal `x2`.
   e. [12] In this part, you will write a function to decode phone numbers automatically from the touch-tones. To help design your decoder, you will look at the energy of the tones at each of the possible frequencies from the Table. The value of

$|X(\omega)|^2$ gives the energy spectral density at each frequency. Write a function `ttdecode` which accepts as its input a touch-tone signal in the format used in part c, and returns as its output a seven element vector containing the phone number. Your function should use `fft` to compute the Fourier transform of each digit in x, and then check the energy at the samples corresponding to the peak frequencies for the touch-tones. Pick which of the row frequencies and column frequencies has the most energy, then use those frequencies to determine what the digit is. Test your function by using `x1` and `x2` as inputs and verify that it returns the correct phone number as in part (d).

f. [14] Most people do not dial their telephone with the precision assumed in this exercise, with each digit exactly the same length and the space between digits always the same length. Modify your function to work with touch-tone signals where the digits and silences can have varying lengths. To simplify things, assume that all the tones and silences are at least 100 samples long. Verify the new version of your function works with the signals `hardx1` and `hardx2`. These signals contain the same tones as `x1` and `x2`, but they are not regularly spaced.