BENG280A, Principles of Biomedical Imaging 10/14/15
Fall Quarter 2015

HOMEWORK #3
Due at 5 pm on Friday 10/23/15

Homework policy: Homeworks can be turned in during class prior to the due date or to the TA's mailbox in the Graduate Student Lounge. Late homeworks will be marked down by 20% per day. If you know that you need to turn in a homework late because of an emergency or academic travel, please let the TA know ahead of time. Collaboration is encouraged on homework assignments, however, the homework that you submit should reflect your own understanding of the material. It is recommended that you make a copy of the homework for yourself (e.g. scan it in) before you turn it in.

Required Readings: Review the notes from class and prior readings as needed.

Problems:

1. Derive the general expression for the sinogram corresponding to a point target at location (x_0, y_0). Evaluate your expression for the following three specific locations for the point target: (a) (40, 0); (b) (0, -30); and (c) (40, -30). Compare your MATLAB results from HW2 to your expressions. Do your expressions predict the MATLAB results?
   HINT: Refer to the slide in the lower righthand corner one page 6 of the ctf1 lecture. If the point of interest (x_0, y_0) lies along the line of projection (shown in the slide), then what must be the value of r? You can simply cite this observation for your derivation.

2. Consider a square object of width 20 mm centered at (0,0) with a uniform attenuation coefficient of 1 inside the square and zero everywhere else. When plotting the projections, assume an FOV of 40 mm and use an initial gridsize of 0.5 mm (e.g. use -20:0.5:20 to define the domain for your plots in MATLAB; you can also experiment with other gridsizes)
   a) Derive an analytical expression for the projections as a function of angle. Use your expression and MATLAB to plot the projections to generate a sinogram of your object (use increments of 1 degree from 0 to 179 degrees). **NOTE:** While you may use the Radon function in MATLAB to check your work, you must show the derivation of the projections to receive credit for this part.
   b) Use MATLAB to generate the backprojection of the object using the projections from part a. An easy way to do this is to backproject at a projection angle of 0 degrees and then use the MATLAB function **imrotate** to rotate each backprojection. Use the bilinear and crop options in imrotate. Comment on the features of the backprojected image. Does decreasing the increments (e.g. using 0.5 degree increments) improve the image?

   c) Derive the 2D Fourier transform of the object. Look at the value of the expression of the transform along the line $k_x = k_y$. Compute the inverse transform of this last expression and compare it to your expression for the projection at 45 degrees. **HINT:** Take into account the convolution/multiplication property.

3. A parallel beam CT imaging system is used to image an object defined as:

$$f(x,y) = rect(x,y) ** \begin{bmatrix} \delta(x-1,y) + \delta(x+1,y) + \delta(x,y-1) + \delta(x,y+1) + \\ \delta(x-1,y-1) + \delta(x+1,y+1) + \delta(x+1,y-1) + \delta(x-1,y+1) \end{bmatrix}$$

   (a) Sketch the object.

(b) Come up with a simpler expression for the object (this will make the rest of the problem **much easier**).
(c) Sketch the projection at 0 degrees.
(d) Sketch the projection at 45 degrees.
(e) Derive the Fourier transform of the object
(f) Show that the projection-slice theorem holds for the projection at 0 degrees.
(g) Show that the projection-slice theorem holds for the projection at 45 degrees.
(h) Consider forming a crude back-projection image estimate using the projections at 0 and 90 degrees. Sketch what this estimate would look like.
(i) **BONUS:** Derive and sketch the projection at 30 degrees.

Matlab Exercise: The purpose of this exercise is to familiarize you with the MATLAB functions used for performing 2D Fourier transforms and manipulating and displaying images. **Boldfaced** items should be turned in with the homework. You can always get more information about a command by typing *help <name of command>*, e.g., *help fft2*.
Steps:

1. First download the file BE280Ahw1im.mat from the course website.
2. Load the image into MATLAB with the command: *load BENG280Ahw1im*.
3. Type *whos* to see the variables in your MATLAB workspace. You should see a variable named *Mimage*. Type *size(Mimage)* to see how large the image is.
4. Use the command *imagesc(Mimage)* to display the image – you should see a sagittal image of a head. To change the colormap display to gray-scale, type *colormap(gray(256))* which will result in a display with 256 shades of gray. **Print out a copy of the image.** Try experimenting with different numbers for the colormap, e.g. *colormap(gray(20));*
5. Compute the 2D Fourier transform of the image with the command *Mf = fft2(Mimage);* where the 2D transform will now be stored in the variable *Mf*. Remember to add the semi-colon at the end of the command, otherwise MATLAB will display all the numbers in the matrix! The command *fft2* puts the zero-frequency value of the transform at the first indices of the matrix. For display it's convenient to put the zero-frequency value in the center of the matrix. To do this, type *Mf = fftshift(Mf);*
6. Type *imagesc(abs(Mf))* to display the magnitude of the transform. It will be hard to see anything, because the dynamic range of the Fourier transform is so large. To get a sense of the dynamic range, find the minimum value of the Fourier Transform with the command *min(min(abs(Mf)))* and the maximum value with the command *max(max(abs(Mf)))*. **Record these minimum and maximum values. What is the ratio of the maximum to minimum value?**
7. To scale the image so that you can get a better sense of what the Fourier transform looks like, you can use the *imagesc* command with the syntax: *imagesc(abs(Mf),[cmin cmax])* where *cmin* will be displayed as the darkest value on the image and *cmax* will be displayed as the lightest value on the image. For example, typing in *imagesc(abs(Mf),[200 1e6])* should give you a nice looking result. Experiment with different values of *cmin* and *cmax*.
8. You can also look at the real part, the imaginary part and the phase of the transform with the commands *imagesc(real(Mf))*, *imagesc(imag(Mf))*, and *imagesc(angle(Mf))*, respectively. If necessary you can use the *[cmin cmax]* option to scale the image properly. **Print out images of the magnitude and phase and real and imaginary parts of the transform. To plot more than one image on the same Figure, you can make use of the *subplot* command in MATLAB.**
9. Another way to look at the transform is to use the *mesh* command. Try *mesh(abs(Mf))*. It will be a little hard to see what is going on, so do the following: Define *span = 128 + (-20:20);* then type *mesh(abs(Mf(span,span)));*
10. ***Resolution***. What happens when we zero out the outer regions of the Fourier transform?
    (a) Resolution reduction in the x-direction.

```
>> res_span = 129+(-16:16);
>> Mf2 = zeros(256,256);
>> Mf2(:,res_span) = Mf(:,res_span);
>> Mf2 = fftshift(Mf2);
>> M_resx = ifft2(Mf2);
>> imagesc(abs(M_resx)); % This will show reduction of resolution in the x-direction.
```

    (b) Demonstrate resolution reduction in the y-direction**. Hand in code and image.**
    (c) Demonstrate resolution reduction in the x and y directions. **Hand in code and image**

11. ***Missing data in k-space***. We can also zero out the inner regions of the Fourier Transform.

```
>>Mfzero = Mf;
>>Mfzero(ky,kx) = 0;
>>iMFzero= ifft2(fftshift(Mfzero)); % look at resulting image.
```

Zero out the following:

(a) kx = 129; ky = 129
(b) kx= 1:256; ky = 129+(-16:16);
(c) kx = 129+(-16:16); ky = 1:256;
(d) kx = 129+(-16:16); ky = 129 + (-16:16);
(e) kx = 1:2:256;ky = 1:256;

For each set of parameters, plot out the Fourier transform and the resulting image. Give a qualitative explanation of why the image looks the way it does.

12. ***Spikes in the data***. You can put a spike at location (kx,ky) in Fourier space with the following commands

```
>>Mfspike = Mf;
>>spike = 100e6;
>>Mfspike(ky,kx) = Mf(ky,kx) + spike; % add spike
>>iMFspike = ifft2(fftshift(Mfspike)); % look at resulting image.
```

Put spikes at:

(a) kx = 129; ky = 129 – note this point corresponds to the center of k-space (i.e. kx = 0, ky = 0 in terms of the coordinates we used in class).
(b) kx= 161; ky = 129
(c) kx = 161; ky = 161

For each spike location, plot out the Fourier transform and the resulting image. Give a qualitative explanation for why the image looks the way it does. For example, what accounts for the direction of the artifacts?