EE369C Fall 2020-21
Medical Image Reconstruction

Assignment 7

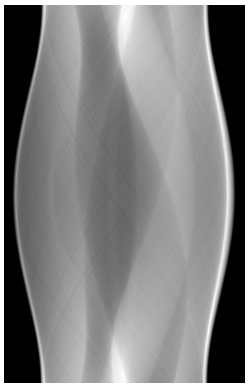
Due Nov. 19

Assignments, and the Class This will be the last assignment for the class. If you do a project, you can skip assignment 6 and 7. Note that the due date is in two weeks, which is the last day of class.

Projection Reconstruction for CT For this assignment we will reconstruct fan beam data acquired from a commercial CT system. The data is from Sarah Patch when she worked at the GE Global Research Center (she is now a Prof. of Physics at Marquette Univ.). It was acquired on a multi-detector system.

We will do this with a rebinning reconstruction. In order to do this we will need a parallel beam back projection algorithm, which we will do first.

1) Parallel Beam Backprojection We will start with parallel beam back projection, and simulated parallel beam data of the famous Shepp-Logan phantom. The phantom and the sinogram data look like this



The data is in the matlab file `sl_phantom.mat`, and has the variables `slp` which is an image of the phantom, and `pd` with is the projection data. The projections are the columns of the matrix (the transpose is shown above). There are 256 samples per projection and 402 projections over 180 degrees.

To reconstruct the projection data you need to first rho filter each column, and then backproject the result. Write a matlab function for rho filtering. Then use the `back_projection.m` function from the web site to do the back projection. The image should look pretty close to the image above.

Include the reconstructed image, and also a reconstruction without doing the rho filter.

2) Display the Fan Beam Sinogram Next we will look at the fan beam data. There are 888 samples (detectors) over an arc of about 55°. A total of 984 projections are acquired over a full 2π rotation. The data is in the matlab file `fan_beam_data.mat` available on the web site via the link. The data is the variable `d`. The fan angle in radians is in the variable `fan_angle`.

The data is quite pretty. Make an image of the sinogram, and include it. You may have to adjust the window to make it show up well.

3) Parallel Beam Reconstruction of the Fan Beam Data Before we get to the fan beam algorithm, it is interesting to try the parallel beam reconstruction directly.

First rho filter the sinogram, and display the data, again windowing down so that you can see most of the data (there are a few bright areas that will be clipped). Note that the SNR is noticeably worse in some areas than others. Compare the low SNR areas to the sinogram you plotted above. Why is the SNR lower in these areas?

Next, backproject the rho filtered data. Since we have 2π of projection angles, choose any interval of π, and perform the backprojection. The data will support a reconstruction of 512 samples. Display the result. You should be able to tell there is something in the data, but it won't be pretty.

4) Rebinning Next is the rebinning operation. What we will do is compute a new parallel projection data set for each θ that we have. In this case we have about six γ samples for each θ. First, define

```
>> [nsamples nprojections] = size(d);
>> dtheta = 2*pi/nprojections;
>> dgamma = fan_angle/nsamples;
```

For each projection angle θ we will go through the fan beam data to produce a parallel projection. The projection angle θ_p for the n^{th} fan beam set and the m^{th} ray is

$$\theta_p = n\Delta\theta + m\Delta\gamma$$

where we assume m goes from `-nsamples/2+1` to `nsamples/2`. Recall from the notes that the rebinning operation essentially resamples the fan beam sinogram along diagonal lines. This is the equation of that diagonal line. The first term is the constant offset for a parallel projection we are creating, and the second term is the linear offset with fan angle.

To calculate the linear offset, consider the projection at angle $\theta_p = 0$. Then

$$\begin{aligned} n\Delta\theta &= -m\Delta\gamma \\ n &= -\frac{\Delta\gamma}{\Delta\theta}m \end{aligned}$$

We can then use this to compute any parallel projection by adding the projection angle constant offset. If we define `ndth` for m, the parallel projection data for a sample `ns` and parallel projection `np` is given by

```
>> dp(ns,np) = d(ns,np+ndth(ns));
```

where `ndth` should be rounded to an integer to keep matlab from complaining. The sinogram extends over 2π, so you need to take care of the end cases, where the index will wrap around (use the modulo function `mod()`). This is essentially a nearest neighbor interpolation.

Show the rebinned sinogram. It should look similar to the sinogram above. The main difference is that the trace of particular bright objects now look like sinusoids, before they were distorted depending on where they were in the field of view.

Make sure you check for discontinuities or gaps. This will help you debug your code.

5) Backprojection There is one more correction we should do, the correction for the non-uniform spacing of the parallel projections, but we'll put that off for a moment.

Use your parallel beam backprojection algorithm to reconstruct the rebinned data. Again, use any set of π projections. Show the reconstruction. This should look pretty nice.

6) Sample Density Correction Finally, we should correct the rebinned parallel beam data for the non-uniform beam spacing. This is a fairly small effect for this data, but it still helps.

The fan-beam rays are at

$$m_f = \sin(m\Delta\gamma)/(\Delta\gamma).$$

measured in beam widths, and `m=[-ns/2:ns/2-1]`. If `mx = max(abs(mf))`, what we want is to resample the data uniformly over the range `[-mx:(mx-1)]`, where the `-1` is to make the number of samples even. Use the matlab routine `interp1()` to do the resampling. For a projection `dp(:,jj)` the corrected data `dc(:,jj)` is

```
>>  dc(:,jj) = interp1(mf,dp(:,jj),[-mx:(mx-1)]);
```

By default, `interp1()` uses linear interpolation, which is probably good enough here. Note that the number of samples in the corrected projections will be smaller that what you started with.

Reconstruct the result, and compare to your previous reconstruction. There should be fewer artifacts, and the resolution and definition of structures towards the edge of the FOV should be better.