EE369C: Assignment 2

Due Thursday, Oct 17

Introduction This assignment introduces the basic operations in a gridding reconstruction algorithm. Starting from a very simple routine, you will add pre-weighting density correction, k-space oversampling, and deapodization.

The data set for this assignment is a simulated phantom using a small spiral acquisition with 6 interleaves of 1536 samples. This is a typical trajectory for real-time cardiac imaging. With a repetition time T_R of 24 ms, this produces a new image every 144 ms, at a rate of 7 images/s. The k-space trajectory, pre-weighting function, and simulated data are in the file:

`http://www.stanford.edu/class/ee369c/data/rt_spiral_03.mat`.

The complex k-space data is in the matlab variable `d`, and the pre-weighting function is in the matlab variable `w`. The k-space trajectory (k_x, k_y) is stored as $k = k_x + i * ky$ in the complex matlab variable `k`. `k` is scaled relative to a $\pm k_{max}$ of ± 0.5. The k-space trajectory doesn't actually reach $\pm k_{max}$, but is scaled to produce the right field-of-view when reconstructed as a 128x128 image.

We will start with a very basic gridding algorithm that doesn't do density correction, uses a very simple separable triangular kernel, uses a 1X grid, and doesn't do deapodization. This m-files is available at

`http://www.stanford.edu/class/ee369c/mfiles/grid1.m`.

Note that this only does the gridding operation. You still need to do an inverse 2DFFT to produce an image.

```
function m = grid1(d,k,n)

% function m = grid1(d,k,n)
%     d -- k-space data
%     k -- k-trajectory, scaled -0.5 to 0.5
%     n -- image size

% convert to single column
d = d(:);
k = k(:);

% convert k-space samples to matrix indices
nx = (n/2+1) + n*real(k);
ny = (n/2+1) + n*imag(k);

% zero out output array
m = zeros(n,n);

% loop over samples in kernel
for lx = -1:1,
  for ly = -1:1,

    % find nearest samples
    nxt = round(nx+lx);
```

```
    nyt = round(ny+ly);

    % compute weighting for triangular kernel
    kwx = max(1-abs(nx-nxt),0);
    kwy = max(1-abs(ny-nyt),0);

    % map samples outside the matrix to the edges
    nxt = max(nxt,1); nxt = min(nxt,n);
    nyt = max(nyt,1); nyt = min(nyt,n);

    % use sparse matrix to turn k-space trajectory into 2D matrix
    m = m+sparse(nxt,nyt,d.*kwx.*kwy,n,n);
  end;
end;

% zero out edge samples, since these may be due to samples outside
% the matrix
m(:,1) = 0; m(:,n) = 0;
m(1,:) = 0; m(n,:) = 0;
```

This loops over the area covered by gridding kernel, which is small. For each sample of the kernel, the gridding operation for the entire data vector is done with the sparse() matrix call. This sets up an n×n sparse matrix with values `d.*kwx.*kwy` at matrix locations `(nxt,nyt)`. This automatically gets converted to a full 2D matrix when it is added to `m`.

If you are unsure what this does, you can rewrite this routine to use a loop over the data samples. This can be outside the kernel loop (simplest option, since most of the vector operations can be eliminated), or in place of the line with the `sparse` command (not as simple, but faster). If you just replace the `sparse` function with a loop you can add the data to the grid with a simple addition:

```
for nd = 1:length(d),
   m(nxt(nd),nyt(nd)) =  m(nxt(nd),nyt(nd)) + d(nd)*kwx(nd)*kwy(nd);
end
```

This does the same thing as the `sparse` command, and is clearer but slower.

1. Simple Gridding Reconstruction Reconstruct an 128×128 image of the simulated phantom data with this algorithm. There is a dominant low frequency artifact. What is it due to? Display your reconstruction.

Extend the algorithm to use the preweighting function `w` that has been provided. Display your reconstruction.

2. Oversampled Gridding Reconstruction Extend the algorithm to reconstruct on a 2X grid. This is a grid that is sampled twice as finely in k-space, and has twice the FOV in image space. The kernel should extend for ±2 samples on the 2X grid. Display your 2X reconstruction. What artifacts have been reduced or eliminated?

3. Deapodization Correction The kernel we are using is a separable triangle function in k_x and k_y. Compute the apodization produced by this kernel for the 2X oversampled reconstruction, and divide it out of the reconstructed image. Plot a cross-section through the phantom before and after correction. Display your corrected reconstruction.

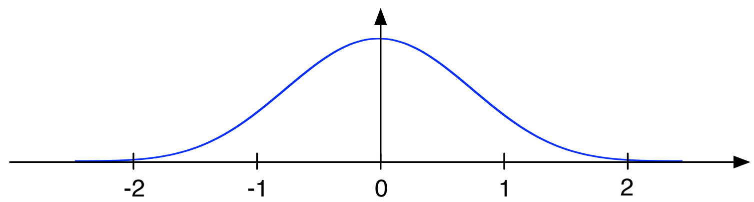

Figure 1: Optimum Kaiser-Bessel Kernel for a 2X grid, with a kernel width of 5 samples on the 2X grid.

4. Improved Kernel Many kernels work well on a 2X grid. One of the most widely used kernels is the Kaiser-Bessel window. It's shape is dependent on a single parameter β, which makes it easy to describe. Also, there is an analytic expression for the inverse transform that is required for the deapodization function. Modify your gridding routine to use the Kaiser-Bessel kernel, given a value of β, and a kernel width. Use the expressions from the Beatty paper.

Reconstruct the data set `rt_spiral_03.mat`. Fully correct for apodization, based on the analytic expression in the paper. Assume a width of 5 grid samples on the 2X grid, and choose the optimal value of β from Eq. 5 of the paper (which is listed in Table III). As a reality check, the kernel should look like Fig. 1. The function $I_0(x)$ used in computing the kernel is the zero order modified Bessel function of the first kind, and is implemented in matlab as `besseli(0,x)`.

You will want to pre-compute the kernel. Assume that we will allow the maximum error due to kernel sampling to be 10^{-3}, choose either nearest neighbor or linear interpolation for the kernel, and choose an appropriate kernel sampling density.

Display images of the full 2X FOV, and the central 1X FOV that would normally be displayed. Also, plot a cross section through the middle of the phantom to demonstrate that the deapodization is correct.

5. Non-Integer Oversampling Modify your gridding routine to reconstruct at non-integer oversampling factors. Use an oversampling ratio of 1.25, and choose a kernel size to achieve a reconstruction error of 10^{-3}. Find the optimum β for the Kaiser-Bessel kernel. Use an interpolation method for the kernel to give you a 10^-3 maximum error.

Recontruct the data set `rt_spiral_03.mat`. Fully compensate for apodization using the analytic expression. Show images of the 1.25X FOV, the central FOV that would normally be displayed, and plot a cross section to show that the deapodization is working correctly.

6. Density Calculation For this problem, we will use a variable density spiral data set of a simulated phantom. This has twice the sampling density near the origin. The data is in `var_dens.mat`, also in the `data` subdirectory. This contains the k-space trajectory `k`, and the complex data `d`. The acquisition has 2048 points per spiral interleave, and 6 interleaves. The image matrix size is 128 by 128. Use a 2X grid for the reconstruction, and the kernel from question 4. Fully correct for the apodization.

a) Post-Compensation with Gridding Implement post-density compensation by gridding unity to estimate and correct the density on the grid points. Display the resulting post-density compensated image.

b) Precompensation with Voronoi Use the `voronoidens.m` routine from the `mfiles` subdirectory to estimate the density on the data points using the Voronoi diagram. This routine returns the area associated with each sample, which is the weighting function (*i.e.* 1/density). You also need to devise a method for assigning reasonable values to the k-space samples at the edge of the sampled region, since the `voronoidens.m` routine returns areas of `NaN` for these samples.

Display the resulting pre-density compensated image. Plot a cross-section through the middle of each image to compare the two methods. Which produces a more accurate reconstruction? Note that the backround inside the circular ring of the phantom has a DC level of 20% of the peak, and is not zero (i.e., this is not an artifact!).

Note: There are a few streaks in the reconstruction of this data set that may not go away. These are subtle, but still noticeable.