EE369C: Assignment 3

Due Thursday, Oct 15

This assignment shows how to do off-resonance correction for spiral k-space trajectories. The data set for this assignment is a pretty shot of a GE resolution phantom

`http://www.stanford.edu/class/ee369c/data/resphantom2.mat.`

This includes two full data sets at two different echo times. This will allow you to compute full resolution field maps, that you will use for map based off-resonance correction. Note that the data set is inverted and flipped when directly reconstructed. You can leave it this way, or orient it as you see it here, as you wish.

The data sets are in the matlab variables `d1` and `d2`. Each has 2048 samples per spiral readout, and 16 readouts per scan. The duration of each readout is 8.192 ms. The echo times for each data set are in the matlab variables `te1` and `te2`. The k-space trajectory is stored in `ks` and the density precompensation in `wt`. The data should reconstruct to a 160x160 image.

A reconstruction of the data set is shown in Fig. 1. While this isn't too bad an image, it is clear that it could use some help. Parts of the image are blurred due to local off-resonance frequency shifts. In this assignment we will first make a map of the frequency shifts. Then, generate a multifrequency reconstruction (using your gridding code from the last assignment). Then, you'll do a map based reconstruction, and an autofocus reconstruction.

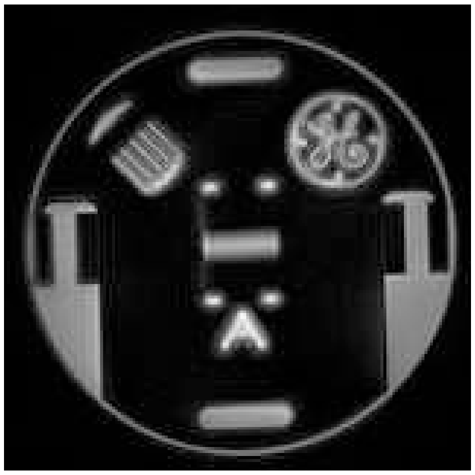

Figure 1: Simple reconstruction of the first data set. While much of the image is in focus, note the blurring around the resolution comb, and the GE logo.

1. Field Map Reconstruct each of the data sets using your gridding reconstruction. If you are unsure of your implementation, use `gridkb.m` from the course web page. Note that `gridkb.m` can return either the image or the k-space data, which may be useful below. Display the two reconstructions.

From the two data sets, compute a field map measured in Hz. Display the field map. If we limit our attention to pixels whose amplitude are greater than 10% of the maximum, what range

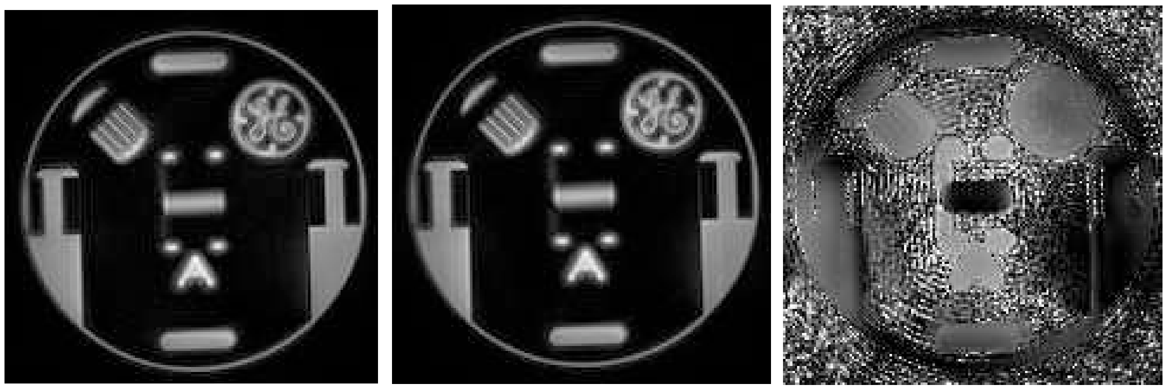

Figure 2: The two data sets look very similar. The first is at at TE of 2.5 ms, and the second at 7.1 ms. The field map shows significant variations in frequency, which correlate well with the off-resonance blurring.

of frequencies do you observe? You may find it useful to implement this as an m-file that takes two images, and returns the field map

```
>> fm = compute_fm(im1,te1,im2,te2)
```

Solution
The two reconstructions, and the field map are shown in Fig. 2. To figure out the range of frequencies we need, we first make a mask that is one for all the pixels greater than 10% of the peak.

```
>> msk = abs(im1) > 0.1*max(max(abs(im1)));
```

Then we find the maximum and minimum frequencies for these pixels,

```
>> max(max(fm.*msk))
ans =
   48.7424
>> min(min(fm.*msk))
ans =
  -76.9491
```

so the range of frequencies goes from -77 Hz to 49 Hz. The lowest frequencies are near the middle of the phantom, and the highest at the edge.

2. Multifrequency Reconstruction Write an m-file that takes a data set, and returns reconstructions over a set of frequencies,

```
>> im_mf = mf_recon(d, k, w, n, te, tad, fmin, fmax, fstep)
```

where `d` is the acquired data, `k` is the k-space trajectory, `w` is the preweighting, and `n` is the image size, as usual. The reconstruction is performed starting at `fmin`, and goes to `fmax` in steps of `fstep` Hz. Note that the echo time is required, `te`, as is the A/D time `tad`. The time of each sample during the acquisition is then

```
>>  [ns ni] = size(d);
>>  t = te + [0:ns-1]*tad/ns;
```

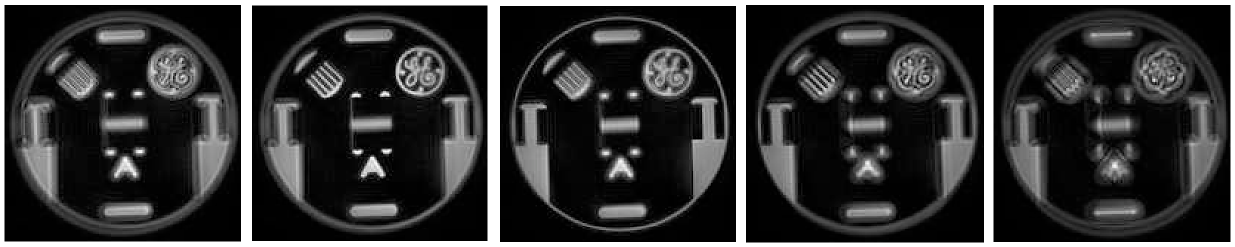

Figure 3: Reconsructions at -128 Hz, -64 Hz, 0 Hz, 64 Hz, and 128 Hz. Note that the structures in the center of the phantom look best at -64 Hz, while the edges look better at 0 Hz, or even 64 Hz.

This includes the phase evolution during the time between the RF pulse and the start of the acquisition.

Your reconstruction will return a 3D array `im_fm(n,n,nf)` where `nf` is the number of frequency steps from `fmin` to `fmax`.

Your reconstruction can either use the simple approach of doing the full gridding reconstruction of

```
>> d.*exp(-i*2*pi*f*t)
```

for each frequency `f`, or it can be more efficient, and grid once, and multiply by a phase function in k-space as we talked about in class (the discrete frequency reconstruction).

Reconstruct the first data set for `f` equal to -128 Hz to 128 Hz in steps of 16 Hz. 128 Hz is one cycle over the readout duration. The step size is smaller than we need by a factor of two. Show reconstructions at -128 Hz, -64 Hz, 64 Hz, and 128 Hz. The 0 Hz reconstruction should look like Fig. 1. Go through the set interactively, and note what frequencies different parts of the image come into focus. This will help you debugging the next part.

Solution

Reconstructions at different frequencies are shown in Fig. 3. The resolution comb and the GE logo look better at -64 Hz.

3. Field Map Based Reconstruction The field map tells you the frequency (in Hz) of each voxel. Use that to choose a reconstruction from the multifrequency reconstruction on a pixel-by-pixel basis. Display your result. Where do you see artifacts?

Hint: This should look much better! If not, check to see if you have the sign of the frequency shift switched.

Solution

All we have to do is convert the field map into an index, and use that to choose which image to show. If `fm` is the field map in Hz, and `fms` is the field map scaled to an index,

```
>> fms = round((fm-mnf)/stf)+1;
>> fms = max(fms,1);
>> fms = min(fms,nf);
```

where `mnf` is the minimum frequency, `stf` is the frequency increment, and `nf` is the total number of frequencies. Then we simply extract the right pixels from the stack of reconstructed images

```
>> im_mp = zeros(n,n);
```

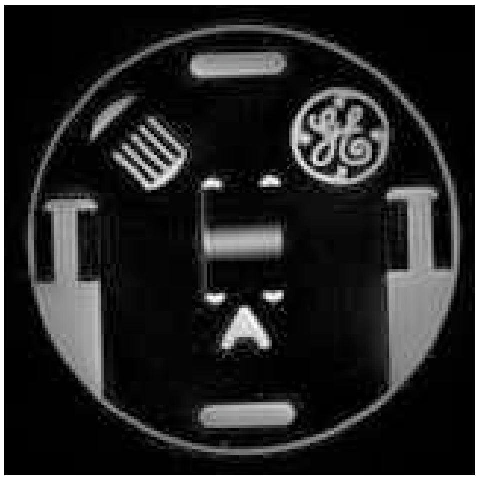

Figure 4: Map based multifrequency reconstruction.

```
>> for ii= 1:n,
>>     for jj=1:n,
>>        im_mp(ii,jj) = im_mf(ii,jj,fms(ii,jj));
>>    end;
>> end;
```

where `im_mf` is the multifrequency stack of reconstructions, and `im_mp` is the map based reconstruction (there has to be a better way to do this without looping!). The result is shown in Fig. 4. Note that most of the image is well focused. If you look closely, you can see some structured artifacts parallel to edges.

4. Autofocus Reconstruction Finally, we'll look at an autofocus reconstruction. Essentially, what we want to do is automatically choose one of the multi-frequency reconstructions based on a focus metric.

a) Phase Correction Since we are counting on the off-resonance phase to produce an imaginary image component, we need to correct for the constant image phase. We'll do this by computing a low resolution reconstruction (instantaneous in time) for each frequency. We use the multifrequency reconstruction you wrote above, but just using the first 1 ms of data (256 samples, 1/8 of the total data)

```
>> im_r=mf_recon(d1(1:256,:),k(1:256,:),w(1:256,:),n,te,tad/8,fmin,fmax,fstep);
```

The phase correction is then

```
>> pc = exp(-i*angle(im_r));
```

We are going to use the absolute value of the imaginary part of the phase corrected reconstruction as a focus metric. If `im_mf` is the multifrequency reconstruction of data set 1, display

```
>> imshow(abs(imag(im_mf(:,: n).*pc(:,:,n))),[])
```

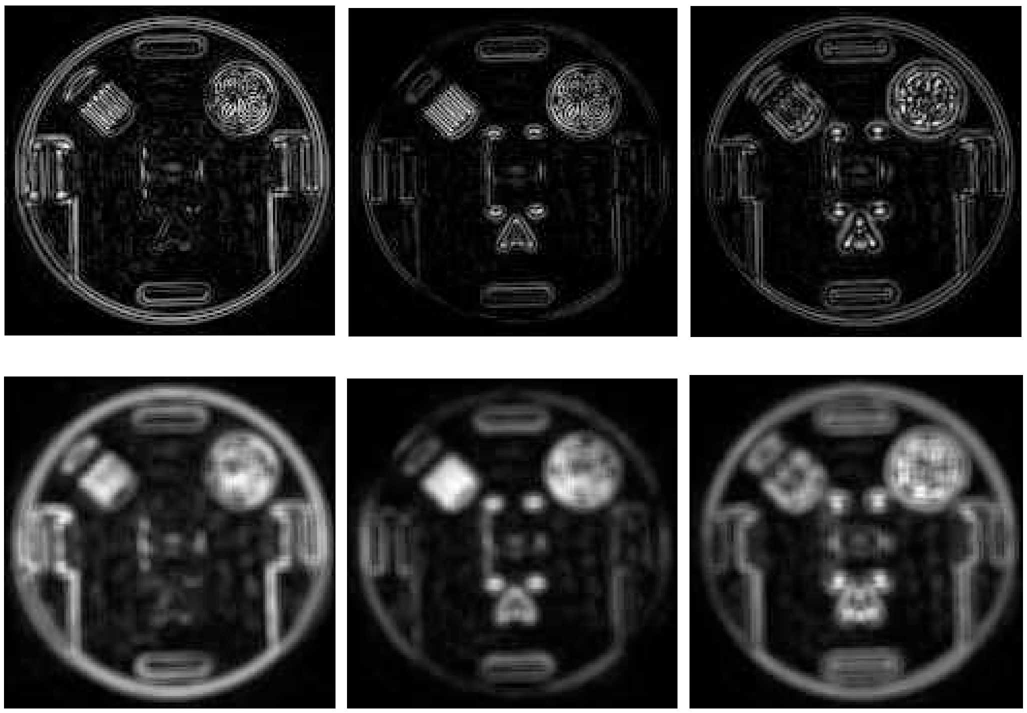

Figure 5: Focus metric for -64 Hz, 0 Hz, and 64 Hz. Top row is unfiltered, and the bottom row is integrated over a 5x5 window.

for n corresponding to f = -64 Hz, 0 Hz, and 64 Hz. Does the focus metric decrease when the frequency is correct?

Solution

The focus metric is shown if Fig. 5 for the unfiltered case (top row) and then averaged over a 5x5 window (bottom row).

The main problem people had was with the phase correction. If this isn't right, the low frequency phase dominates that from the high frequencies that we are looking for. The thing to look for is that the focus metric should show mostly edges.

b) Focus Metric The focus metric is the absolute value of the phase corrected imaginary component, integrated over a local window. The absolute value of the phase corrected imaginary component is

```
>> fc = abs(imag(im_mf.*pc)))
```

For each frequency, we can integrate over a 5x5 window by

```
>> fcf = 0*fc;
>> for n = 1:nf,
>>   fcf(:,:,n) = conv2(fc(:,:,n), ones(5,5)/25, 'same');
>> end;
```

This integrates over a 5x5 widow, registered with the original data. Display the filtered `fcf` for f = -64 Hz, 0 Hz, and 64 Hz.

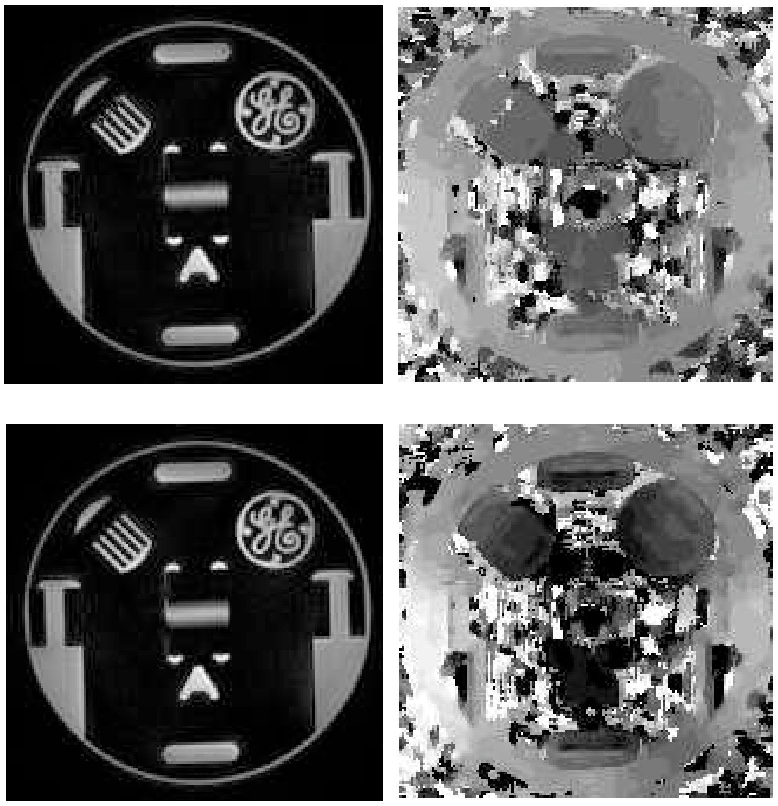

Figure 6: Autofocus reconstructions over +/- 128 Hz (top row) and +/- 64 Hz (bottom row).

Solution
This is shown in Fig. 5.

c) Autofocus Reconstruction For each pixel, choose the reconstruction with the smallest focus metric. Keep track of both the reconstruction, and the index of the frequency that was chosen. This is a field map derived from the data itself. Display the autofocus reconstruction and the field map.

This should look pretty good, but not perfect. What simple thing can you do to fix it? Describe your fix, and show the improved reconstruction and field map.

Hint: Do you need +/- 128 Hz frequency range? What does too big a range do?
Solution
The autofocus reconstruction and the map are shown in the top row of Fig. 6 for +/- 128 Hz, which is +/- 1 cycle of phase over the readout. Most of the image looks very good. The exception is the resolution comb.

We will converge to the solution if we within +/- one cycle. However, if the frequency is far enough off, we can end up taking a value on the "tail" of the focus metric, and being completely wrong. This is what is happening with the resolution comb.

We can improve things by narrowing down the range of frequencies we search. If we restrict ourselves to +/- 64 Hz (which is most of the range we found with the field map) things look much better. The autofocus reconstruction and the field maps are shown in the bottom row of Fig. 6.