EE369C: Assignment 4 Solutions

Due Wednesday, Oct. 25

The problems this week will be concerned with 2DFT SENSE reconstruction. The data is an axial brain scan with an eight channel head coil. It is available here:

`http://ee369c.stanford.edu/data/brain_8ch.mat`

The sensitivities were estimated from the data. The data and coil sensitivities are in the following variables

- `im(160,220,8)` the complex, fully encoded images for each channel
- `map(160,220,8)` the estimated complex sensitivities for each channel

This data and the maps were generated by Miki Lustig.

1. Displaying the Data We have lots of channels, so you will want to display images as an array. One way to do this is use the `montage` function which is in the image toolbox. Unfortunately, `montage` wants a 4D array. You can do this with `reshape` like this:

```
>> montage(reshape(abs(im),160,220,1,8), [0 imax]);
```

where we are displaying the absolute value of `im` over a range of 0 to `imax`. Note that the phase encoding direction of the data is the up/down direction, since it is usually taken to be the shortest dimension. If we were displaying this image for clinical purposes, we would rotate it by 90 degrees, to get it into the standard orientation.

Show the coil data, and the image data using and appropriate range to see the brain (the subcutaneous fat around the edge of the head should be saturated). Note the variation in coil sensitivities.

Solution The image and map data are shown in Figs. **??** and **??**.

2. Multicoil Reconstruction The next task is to compute a reconstruction of the data using the full multicoil reconstruction we described in class, and the square root of the sum of squares reconstruction. Each of these should only require about one line of matlab code! A useful command here is `sum(x,nd)` which sums an array `x` along the `nd` dimension.

Display both reconstructions. Ideally the multicoil reconstruction should have slightly better SNR, and be more uniform, although it is hard to tell here.

Solution The matlab code to generate the full multicoil reconstruction `mcim` and the square root of the sum of squares reconstruction `sqim` are

```
>> mcim = sum(im.*conj(map),3)./sum(map.*conj(map),3);
>> sqim = sqrt(sum(im.*conj(im),3));
```

These are displayed in Fig. **??**

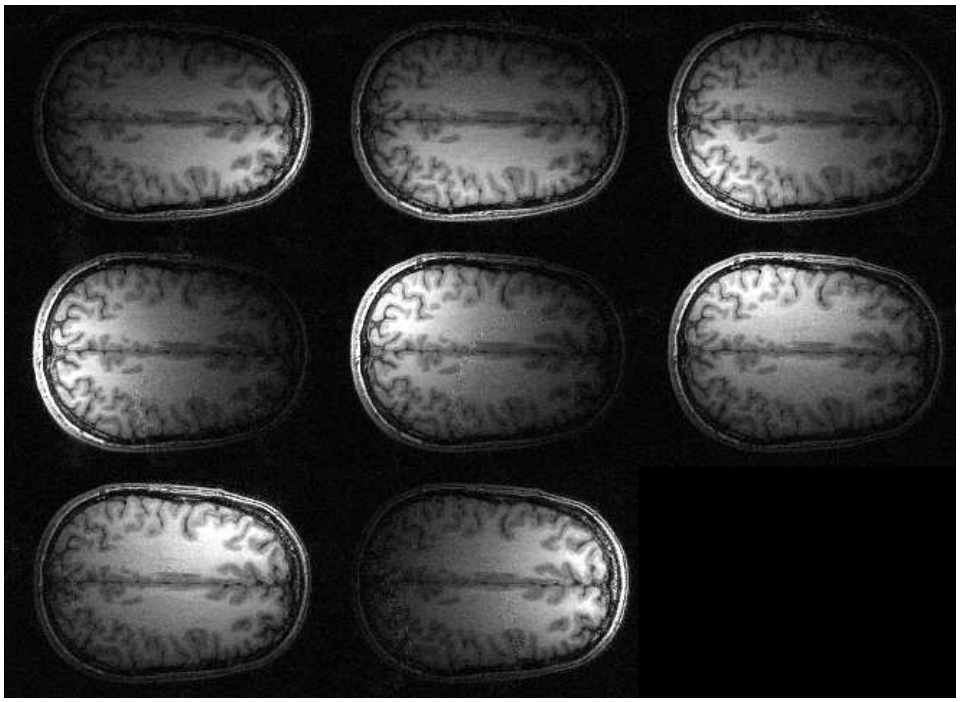

Figure 1: Image data, windowed from 0 to 900

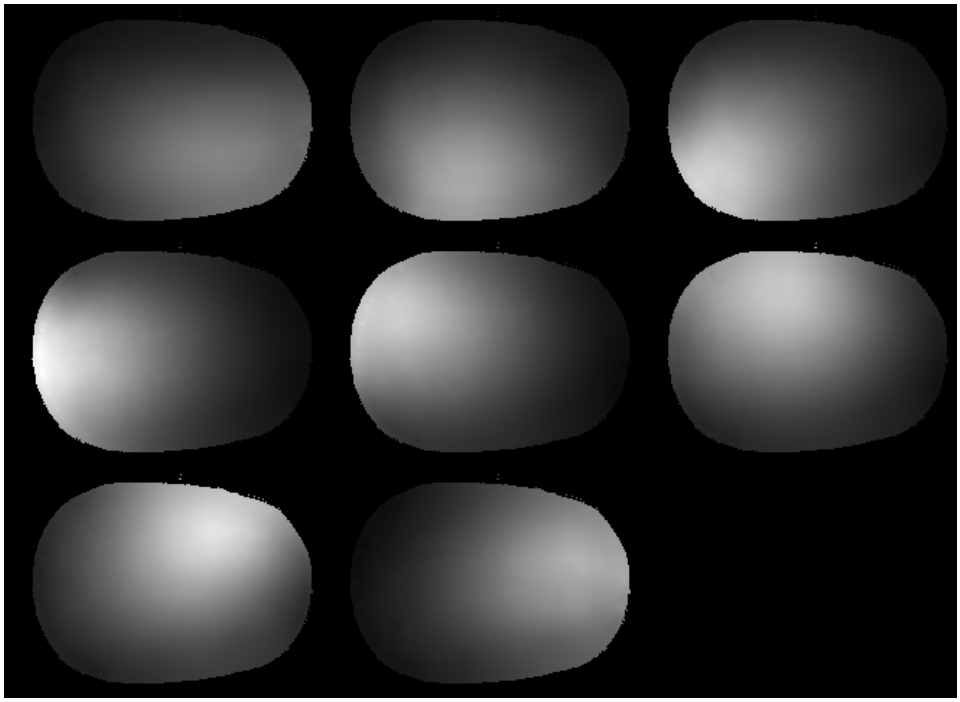

Figure 2: Image data, windowed from 0 to 6

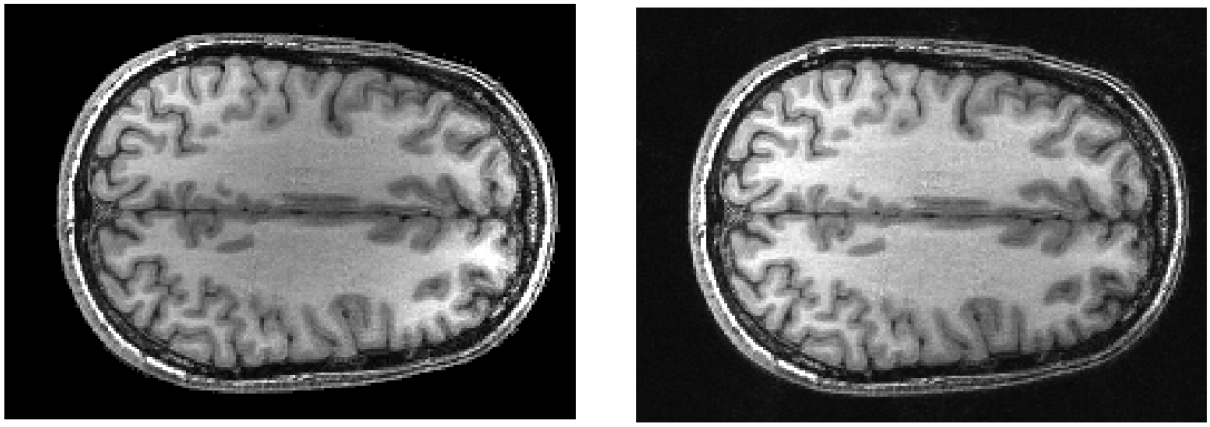

Figure 3: Full multicoil reconstruction (left) and the sum of squares reconstruction (right).

3. g-Factor Maps The most important characteristic of a SENSE acquisition is the geometry factor g. This tells you how well conditioned the reconstruction problem will be.

Write an matlab function that determines the g-factor when undersampling in the x and/or y dimensions:

```
g = gfactor(map,Rx, Ry)
```

where `map` are the coil sensitivities, and `Rx` and `Ry` are the acceleration factors in the x and y dimensions. For convenience, assume that `Rx` and `Ry` divide evenly into the image dimensions.

Since the coil sensitivities were estimated from the data, the sensitivity maps only exist over the head itself. This means that a given voxel can be aliased with a variable number of other voxels, and this can be confusing to keep straight.

One option is doing the calculation for each source voxel independently. This takes longer, but is easier to keep straight. Make the sensitivity of the voxel you are interested in the first column in the `C` matrix. The aliased voxels are separated in the image by multiples of `Nx/Rx` and `Ny/Ry` if the image is `Nx` by `Ny` in size. Append the sensitivities of any aliased voxels that have non-zero sensitivities to the `C` matrix. Then when you compute the g factor, just look at the (1,1) element that corresponds to the source voxel you are interested in. That way you don't have to explicitly keep track of how many other voxels are aliased, and where they are.

Compute and display the g-factor maps for `Rx` for 2, 3, and 4, for `Ry` for 2, 3, and 4, and for `Rx=Ry=2`. For the factor of 3, reduce the size of the coil data by one to make it evenly divide by 3. Display the results on a scale of `[0 5]`. Display 0 if there is no sensitivity map at a voxel.

Solution The code to generate the g-factor maps is given below

```
function [g, nc]  = gfactor(C,RX,RY)
%
% [g, nc]  = gfactor(C,Rx,Ry)
%
% Compute the g-factor map for a set of coil sensitities and accelerations
%
%    C  -- array of coil sensitivities, Nx x Ny X Nc
```

```
%    Ry -- x acceleration
%    Ry -- y acceleration
%
%    g  -- g-factor map
%    nc -- number of coils contributing
%

%
% written by John Pauly
% (c) Board of Trustees, Leland Stanford Jr University, 2011
%


[NX NY L] = size(C);

NRX = NX/RX;
NRY = NY/RY;

g = zeros(NX,NY);
nc = zeros(NX,NY);
for ii=1:NX,
    for jj=1:NY,
        if abs(C(ii,jj,1)) < 1e-6,
            g(ii,jj) = 0;
        else
            for LX=0:RX-1,
                for LY=0:RY-1,
                    ndx = mod((ii-1)+LX*NRX,NX)+1;
                    ndy = mod((jj-1)+LY*NRY,NY)+1;
                    CT = C(ndx, ndy, :);
                    CT = CT(:);
                    if ((LX==0) & (LY==0)),
                            s = CT;
                    else if abs(CT(1)) > 1e-6,
                            s = [s CT];
                        end;
                    end;
                    nc(ii,jj) = nc(ii,jj)+1;
                end;
            end;
            scs = s'*s;
            scsi = inv(scs);
            g(ii,jj) = sqrt(scs(1,1)*scsi(1,1));
        end;
    end;
end
```

It take the full maps, and then for each pixel, computes which pixels interfere, and what the g-

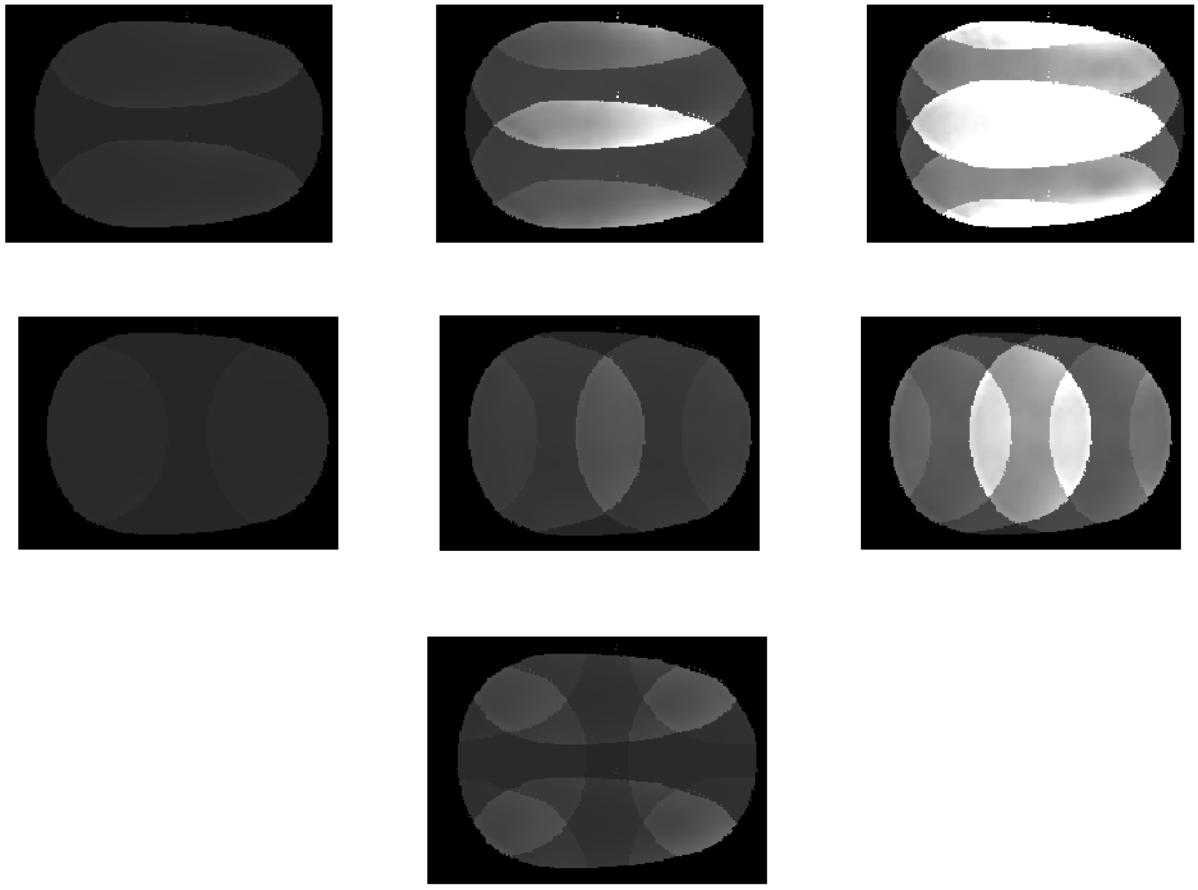

Figure 4: g-factor maps for accelerations of 2, 3, and 4 in the x and y directions, as well as a an acceleration of 2 in both. The scale here is from 0 to 5.

factor is for that set of pixels. It also keeps track of him many pixels are interfering.

The g-factors for the various accelerations are shown in Fig. **??**.

4. Generate Undersampled Images Write a matlab function that takes the fully sampled data, and generates the undersampled aliased images.

```
ima = undersample(im,Rx,Ry);
```

This will take the image data, do a 2D DFT, zero out lines in the frequency domain, and then inverse transform to produce the aliased images. For a factor of 4, you will zero out three out of every four lines. For `Rx=Ry=2` you will zero out every other row and column.

Test your routine on the $Rx = Ry = 2$ case, and display your result.

Solution The matlab code that subsamples the data is shown below:

```
function [ima ma] = undersample(im,Rx,Ry)
%
%[ima ma] = undersample(im,Rx,Ry)
%  generate an accelerated data set
%
%    im -- original images
%    Rx -- acceleration in x
%    Ry -- acceleration in y
%
%    ima -- aliased image data
%    ma  -- undersampled k-space data (optional)
%


%
% written by John Pauly
% (c) Board of Trustees, Leland Stanford Jr University, 2011
%


[Nx Ny L] = size(im);

ima = zeros(Nx,Ny, L);
ma = zeros(Nx,Ny,L);

msk = zeros(Nx,Ny);
for ii=1:Rx:Nx,
    for jj=1:Ry:Ny,
        msk(ii,jj) = 1;
    end
end

% for each coil
for ll=1:L
    ma(:,:,ll) = fft2c(im(:,:,ll)).*msk;
    ima(:,:,ll) = ifft2c(ma(:,:,ll));
end;
```

The aliased data for accelerations of 2 in x and y is shown in Fig. **??**.

5. SENSE Reconstruction Finally, write a matlab function that takes the coil sensitivity maps and the aliased, undersampled images from the previous part, and computes the SENSE reconstruction of the image:

```
im = sense(ima, map, Rx, Ry)
```

The program logic is identical to the gfactor function you wrote above. You just have to add the calculation of the image value for each voxel.

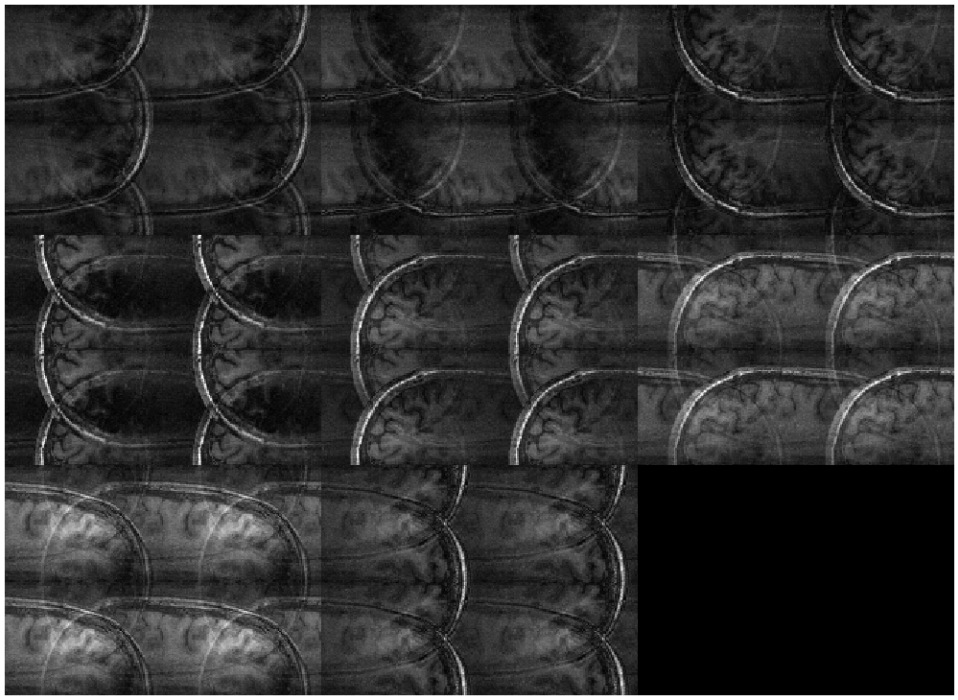

Figure 5: Image data, windowed from 0 to 900

Reconstruct and display images for `Rx` of 2 and 4, `Ry` of 2 and 4, and `Rx=Ry=2`. As a test, try reconstructing the `Rx=Ry=1` case. This should give you exactly the same thing as the multicoil reconstruction for part 2. The full multicoil reconstruction is often called the $R = 1$ SENSE reconstruction. At low SNR's this does much better than the square root of sum of squares reconstruction.

Solution The code for the SENSE reconstruction is shown below. This assumes that the data is tiled to the whole FOV so that it lines up with the maps. It recomputes the decomposition for every pixel, which is inefficient, but simplifies the logic. This looks almost exactly like the g-factor code above.

```
function im = sense( ima, map, RX, RY )
%
% im = sense( ima, map, RX, RY )
%
%  ima -- aliased images, undersampled at Rx and Ry, at full FOV
%  map -- maps of the coil sensitivities
%  Rx  -- x acceleration factor
%  Ry  -- y acceleration factor
%
%  im  -- SENSE reconstruction of ima
%
%  this assumes that the noise correlation matrix is I
%
```

```
%
% written by John Pauly
% (c) Board of Trustees, Leland Stanford Jr University, 2011
%

[NX NY L] = size(map );

NRX = NX/RX;
NRY = NY/RY;

im = zeros(NX,NY);
for ii=1:NX,
    for jj=1:NY,
        if abs(map(ii,jj,1)) < 1e-6,
            im(ii,jj) = 0;
        else
            for LX=0:RX-1,
                for LY=0:RY-1,
                    ndx = mod((ii-1)+LX*NRX,NX)+1;
                    ndy = mod((jj-1)+LY*NRY,NY)+1;
                    CT = map(ndx, ndy, :);
                    CT = CT(:);
                    if ((LX==0) & (LY==0)),
                            s = CT;
                    else if abs(CT(1)) > 1e-6,
                            s = [s CT];
                        end;
                    end;
                end;
            end;
            scs = s'*s;
            scsi = inv(scs);
            m = ima(ii,jj,:);
            m = m(:);
            mr = scsi*s'*m;
            im(ii,jj) = mr(1);

        end;
    end;
end

end
```

The reconstructions are shown in Fig. **??**. An acceleration of 4 in either dimension causes significant artifacts. A total acceleration of 4 by accelerating by 2 in x and 2 in y does better, but still has noticeable artifacts.

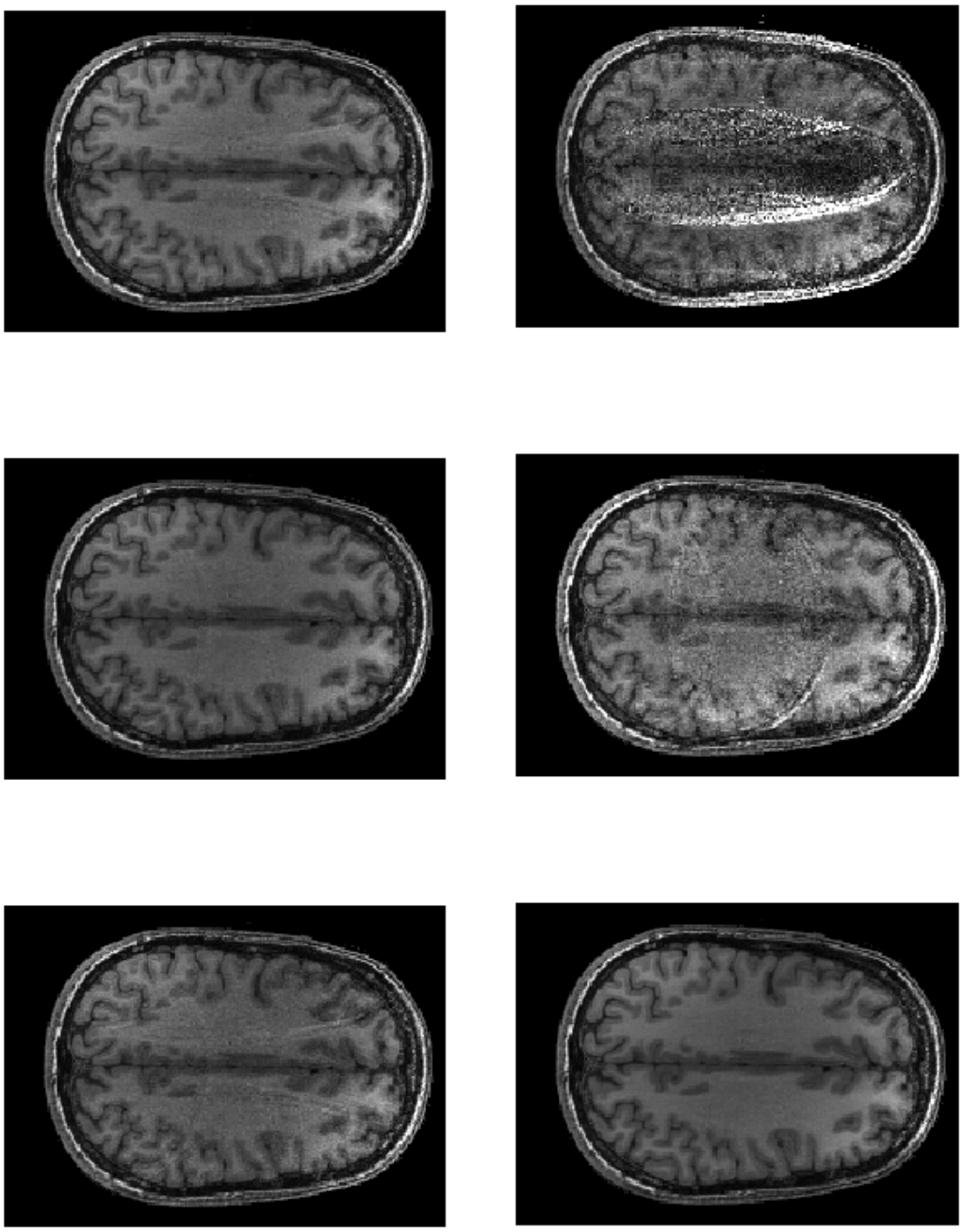

Figure 6: The top two rows are SENSE reconstructions for accelerations of 2 and 4 in x and y. The bottom row shows the acceleration by 2 in both x and y (left), and the $R = 1$ SENSE reconstruction.