Bulk Motion Correction with Variable Density Spiral Trajectories

By: Farhad Farahani

EE599 Project/Spring 04

Introduction

Motion during MR Imaging causes two main types of artifacts generally referred to as Time-of-Flight and Phase artifacts. The former arises as a result of tissue displacement during the quiescent period between each data sampling period and the following excitation RF, while the latter happens as a result of spin phase induced by motion through magnetic field gradients during readout. Methods have been proposed to mitigate these artifacts. Some methods make use of navigator echoes [1] to compensate for motion, while others like PROPELLER [2] oversample the center of *k*-space and obtain inherent ‘‘navigator’’ information, which permits correction for in-plane displacement and rotation. Variable Density Spiral (VDS) adaptive imaging [3] was proposed to obtain real-time images of the coronary artery based on the rejection of highly distorted real-time low-resolution frames reconstructed from a higher density inner spiral in the k-space trajectory.

The idea behind both VDS-adaptive imaging and PROPELLER is sampling a small region in the center of k-space for each interleave of the acquisition and forming a low-resolution almost motion-free image from that data. In VDS-adaptive imaging this low-resolution image forms a basis for the rejection of the whole acquisition and no compensation scheme is proposed. In PROPELLER, techniques are used to compensate for the bulk motion and rotation estimated from this low-resolution image. Because specific trajectories like PR and Spiral are generally known to reduce flow/motion artifacts in the reconstructed image, this makes them a better choice for use in a motion compensation algorithm. In this work I propose the use of VDS trajectories and then applying the ideas of PROPELLER for bulk motion compensation.

The results shown are primary and based on a simplified simulation of a noisy randomly moving object. In what follows I will describe the algorithm details and the simulation results.

Background

VDS trajectory

Data are collected in *k*-space in 20 interleaves, each consisting of a variable density spiral designed to have higher inner-spiral density and lower outer-spiral density. Figure 1 shows two sample interleaves. The inner/outer spiral coverage ratio determines the resolution of the image reconstructed from the inner spiral and hence affects the accuracy of the compensation algorithm. The inner spiral for each interleave is designed to form an alias-free low-resolution and almost motion-free (beause of the low acquisition time for each inner spiral) image which will serve as a base for the estimation of the translation and rotation of the object for that specific interleave.

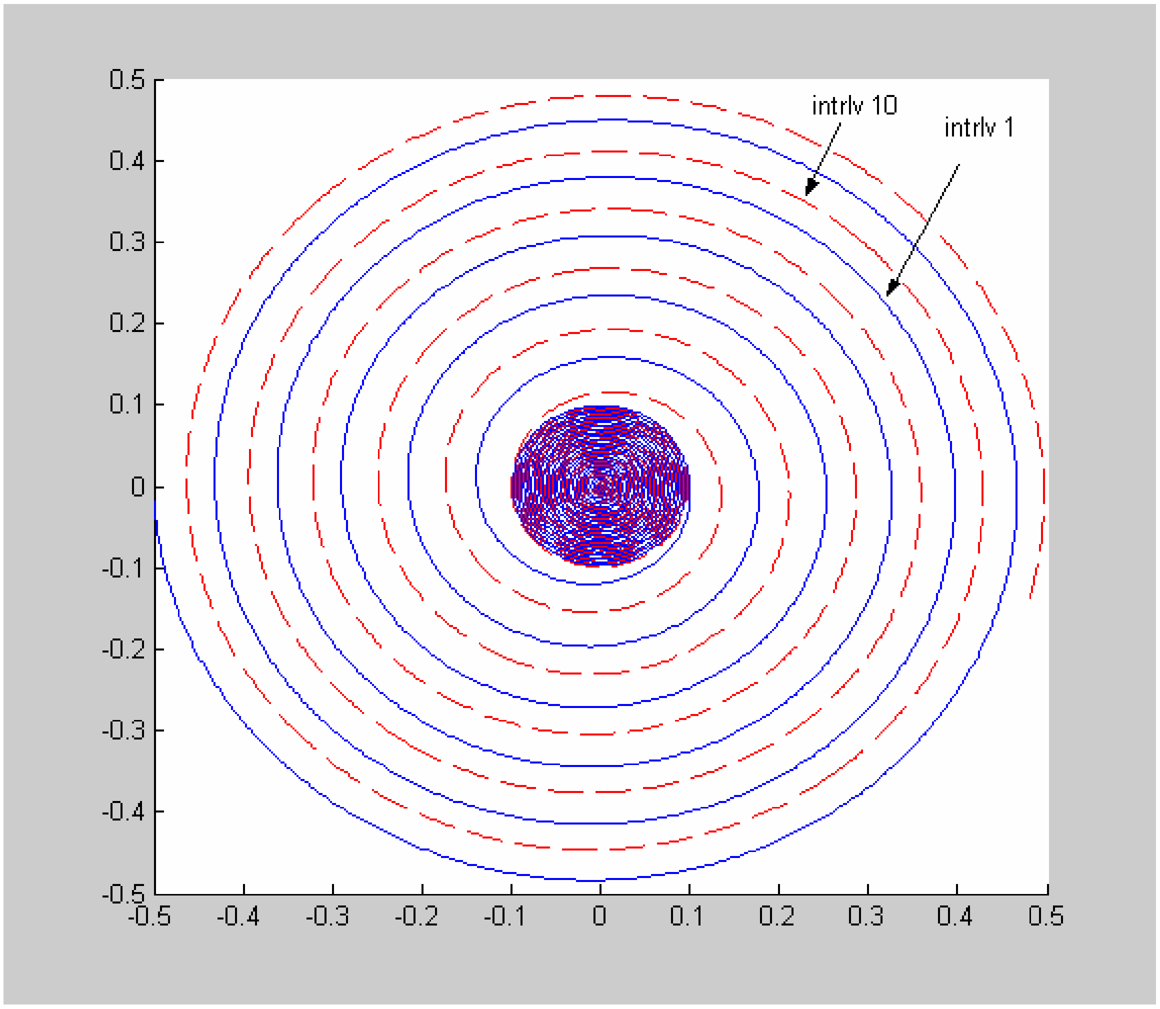


Figure 1. a VDS trajectory.

Bulk Rotation Correction

It is well known that rotation of an object in image space produces identical rotations of its Fourier transform in *k*-space, while shifts of an object in image space produce linear phase shifts in the *k*-space data. Thus, if one looks only at the magnitude of *k*-space data, the bulk rotation of an object can be assessed independent of its bulk translation. As stated above, there is a circle in the center of *k*-space that is densely measured for each interleave. A set of Cartesian coordinates that spans this central circle is defined as *R*. The data magnitude M_n of each interleave inside this circle is gridded onto *R*. Each M_n is then rotated by a series of angles and gridded onto *R* for each angle. The set of angles spans a user-specified range that covers the expected range of rotations. M_1 will serve as a reference and the correlation of M_1 and M_n is measured as a function of rotation angle. This correlation should be largest when M_n is rotated to M_1. It was found empirically that this correlation is more robust if each datum inM_1 and M_n is first multiplied by the square of its distance from the *k*-space origin; this removes the heavy weighting at the *k*-space center due to the concentration of signal energy at low spatial frequencies, and places more emphasis at the edges of *R*, where small rotations produce a larger azimuthal shift. The angle at which the peak of this correlation occurs is an estimate of the angle of rotation for the *n*th interleave. Once the angles are estimated for each interleave, the respective coordinates are rotated to match the estimated rotation of the object at the time of data collection. These rotated coordinates are then used in the translation correction as well as in the final reconstruction.

Bulk Translation Correction

In this case the data are kept complex complex. D_n is the complex data for the *n*th interleave contained within the central circle and gridded into *R*. If D_1 and D_n form low-resolution images d_1 and d_n respectively, then the translation for each interleave can be found by finding the maximum value of the convolution of d_1* and d_n. This convolution is best performed by first multiplying D_1* by D_n, and taking the inverse Fourier transform of the product. After this inverse Fourier transform is taken, the peak magnitude around a sinc interplolated version is found. The location of the peak is the estimated translation in the *x* and *y* directions, and a corresponding linear phase is removed from the collected data in that interleave.

Final reconstruction

Final reconstruction uses all corrected data comprising the corrected k-space values and rotated trajectories. A sinc-kernel gridding on a 2X Cartesian grid with preweighting is performed for the final reconstruction. The preweighting function is found from a voronoi diagram of the stack of the corrected interleaved spirals.

Methods

VDS generation

A variable density spiral with N=20 interleaves is used so that the motion artifacts be ignorable during each interleave acquisition. The inner spiral for each interleave is designed to give an FOV=24cm and the outer one is designed to give an FOV=2.4cm. This way each inner-spiral interleave gives a low-resolution alias-free image while the outer spirals give an alias-free image when stacked together. Resolution is assumed 1mm so maximum k-space radius to be covered is set to rmax=5 cm^{-1}.Maximum gradient amplitude and maximum slew rate are set to 3.9 G/cm and 14500 G/cm/s respectively. Sampling rate is set to T=4 *usec*. The inner spiral is set to be 1/5 of the max k-space coverage so images generated from the inner spirals will have a 5mm resolution. If this ratio is lowered we will have less inner-spiral resolution , and hence less accuracy in the rotation and translation estimation but less motion artifacts during the inner-spiral acquisition. Increasing this ratio will cause more motion artifacts during acquisition but more accuracy in the estimation algorithm. Since I am simulating discrete shifts per each interleave and unfortunately could not simulate the motion artifact **during** the readout I won't test for the effects of varying the inner/outer spiral coverage ratio and heuristically chose it to be 1/5, But it is interesting to test the effect of this parameter on the performance with real data. Figure 1 shows two generated interleaves. With these constraints each interleave will have an acquisition time of 16.1ms (4026 samples). The portion of acquisition time spent for the inner spiral turns out to be 7.2ms (1812 samples).

I use a modified version of Brian Hargreaves' code [4] to generate the VDS with the specified parameters. The codes are included in the appendix :

genvds.m : uses vds.m to generate the trajectory
vds.m : Brian's code
q2r21.m : Modified version of Brian's code. Instead of a polynomial FOV I modified the code to generate a stepwise FOV. FOV=24 when r<1 and FOV=2.4 for 1<r<5.

Simulating the object

For each interleave I produce the object consisting of ovals,circles,squares,stripes and then apply a uniform random shift of -2cm to 2 cm and a random rotation of –pi/4 to pi/4 .Figure 2 is the 256x256 true static image when built on a Cartesian grid. White Gaussian noise is then added to the k-space data generating SNRs of 10,20,30,40 dB.

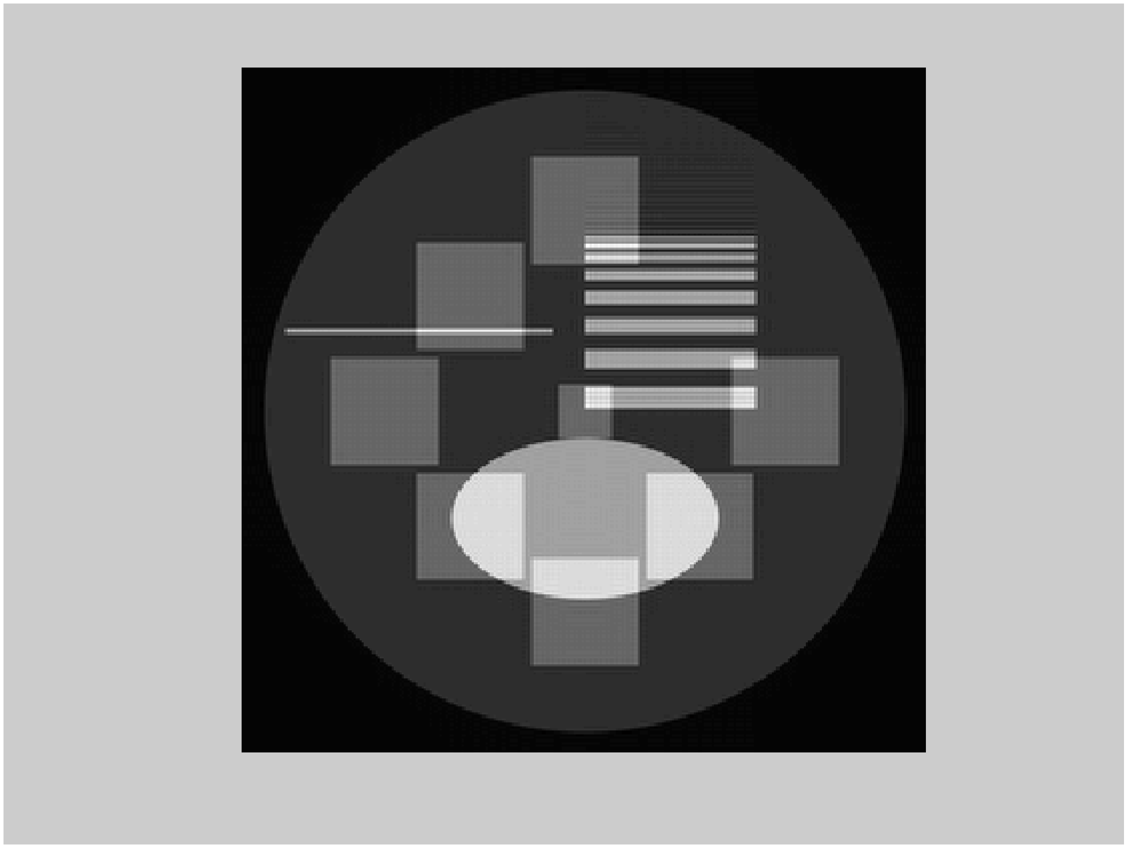


Figure 2.Original static image.The outer circle is the FOV and is not included in the original image(just added for the picture)

The code for this part is found in the appendix :

Buildmovingobj.m : uses buildobj.m to produce the moving object for interleaves. Computes k-space values on the trajectories.
Buildobj.m : code to generate the static object.
Main.m : the part for adding noise.

Gridding

Gridding is an essential part of the algorithm. A 2X Cartesian grid (512x512) is used to reduce the aliasing artifact in gridding. The kernel used is a sinc covering 11 grid samples (L=-5:5) windowed by a hamming function. The preweighting density compensation function is computed using the areas of the voronoi diagram cells. Because the specific code used for area computation generates some problems at the spiral edges I use a simple code for fixing that, assuming the area related to the edge of the spiral is a continuation if the trend in the previous turn. Here are the attached associated code :

Voronoidens.m	*: computes the area of a trajectory*
Vorodensmulti.m	*: uses the above to compute for all interleaves*
Fixaa.m	*: fixed the edge problem for one interleave*
Fixa.m	*: fixes the problem for all interleaves stacked together.*

Bulk rotation correction

As stated in the background section the inner spiral data for each interleave is gridded into a Cartesian grid R. The grid is chosen to be a 2X grid on the inner circle with radius 0.1. So the grid should be 512/5*512/5. The nearest power of two to 512/5 is chosen for the grid size hence the grid will be 128x128. Each inner spiral will be gridded onto R as well as its rotations for a set of angles. The angle giving the highest correlation should be chosen. This algorithm is not that much computationally efficient but is chosen for sake of simplicity. It takes around 15 minutes to estimate the rotation angle for all the interleaves on a p4 machine to a 0.001 *rad* precision. To reach a 0.001 *rad* precision the search for the rotation angle is done in three iterations. First a search for *theta*=[-pi/4:0.04:(0.04+pi/4)] is done and the angle *theta* is found. Then a finer resolution search for [*theta*-0.1:0.01:*theta*+0.1] is performed and *theta* is refined. The third iteration will be for [*theta*-0.02:0.001:*theta*+0.02]. After *theta* is estimated the whole interleave spiral will be rotated by that angle. The code for this part is attached:

Fixrot.m : estimates the rotation angles and fixes the trajectories

Bulk translation correction

The fixed trajectories from the previous stage will be used afterwards. As explained in the background section each inner-spiral interleave data will be again gridded onto a Cartesian grid and the convolution with the first image will be computed via Fourier transform. The peak of this convolution gives the estimated shift in the x and y direction. Since the inner-spiral images will be of 5mm resolution the estimated shifts will be limited to this quantization. In a try to overcome this issue the 2D correlation function is sinc interpolated around the peak by a factor of 20 and the peak is refined. The code is attached :

Fixtrans.m : fixes the shifts in the x and y directions.

Final reconstruction

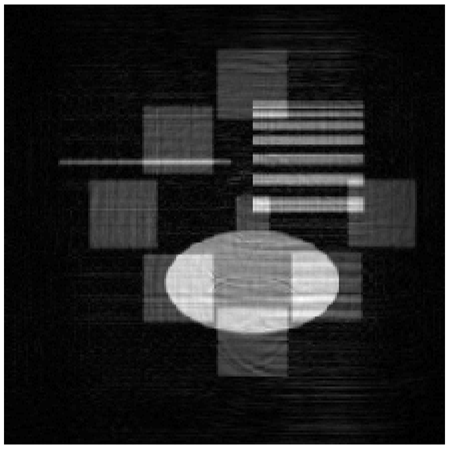
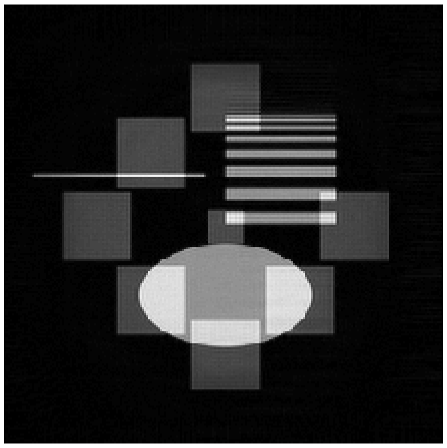
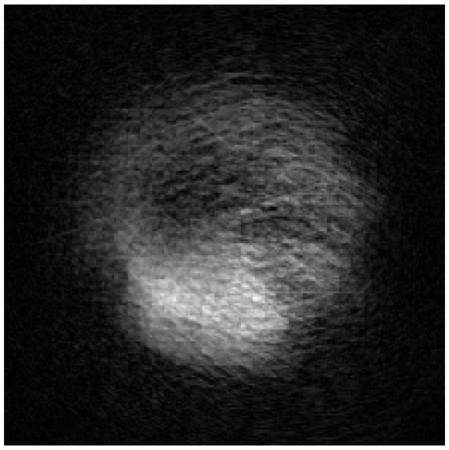
The fixed data for all interleaves is stacked upon each other and then gridded and reconstructed. The density estimation should be done for the stacked interleaves.

Main.m : The main code wrapping everything together.

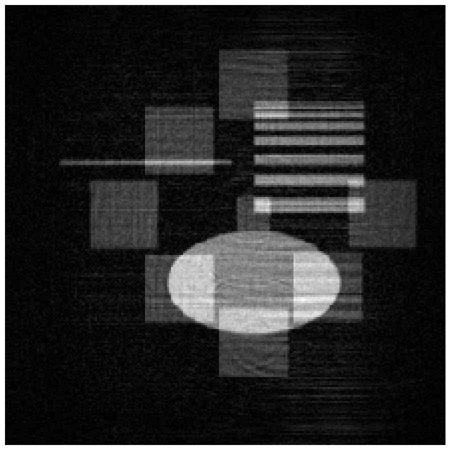
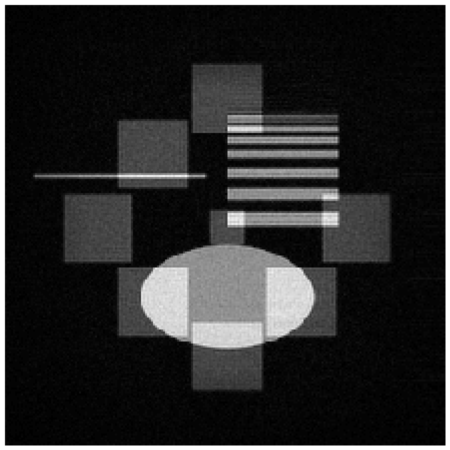
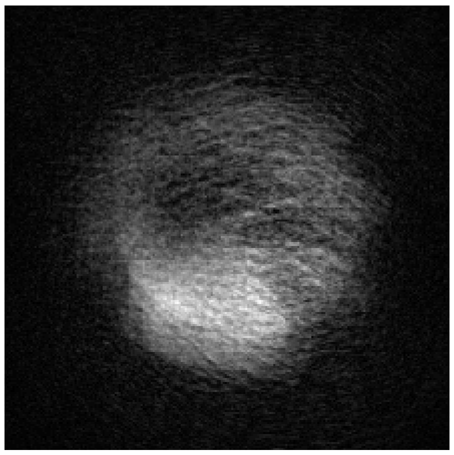
Results

On the next pages you can find the resulting reconstructed images. They are 512x512 because of the 2X grid but just the inner 256x256 pixels containing the FOV is shown. Compensated images are the ones with the above algorithm applied. Static images are the static images reconstructed by the same gridding algorithm and is the ideal to be achieved by the compensation algorithm and so will serve as a baseline for comparison. The uncompensated image is the moving image with no motion compensation applied.

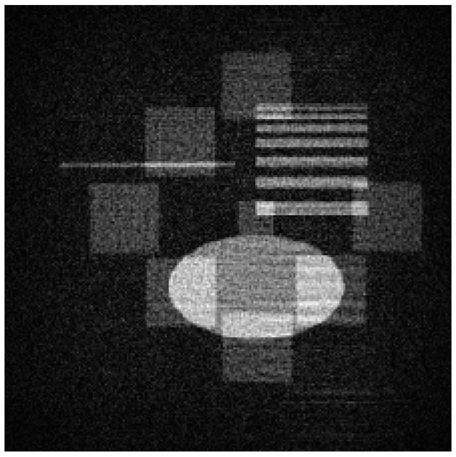
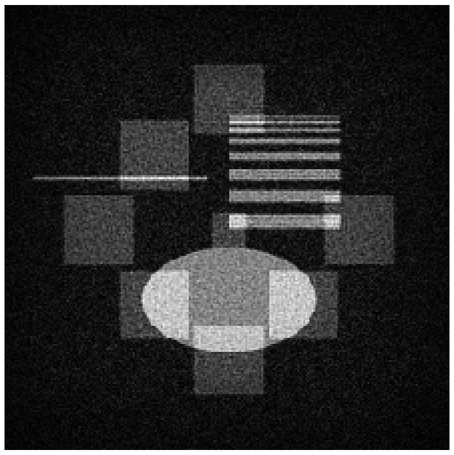
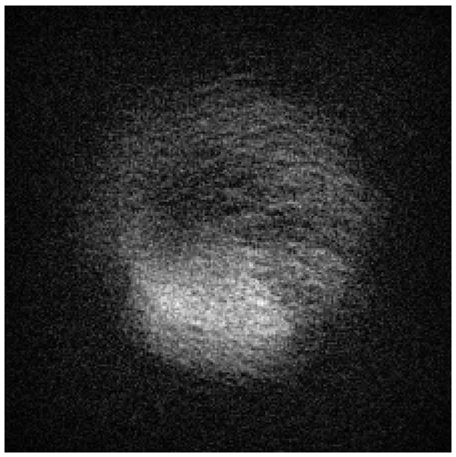
SNR=40dB

Compensated image	
Static image	
Uncompensated image	

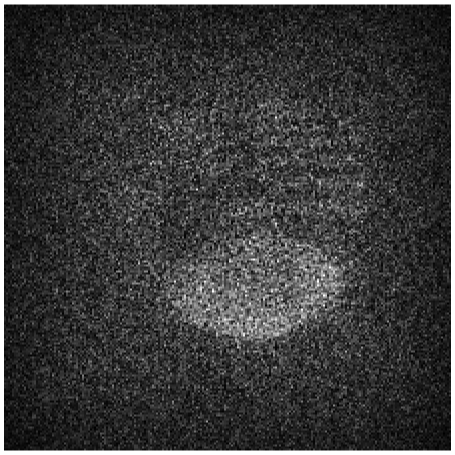
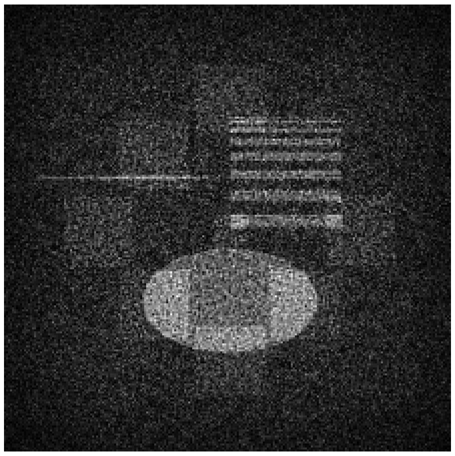
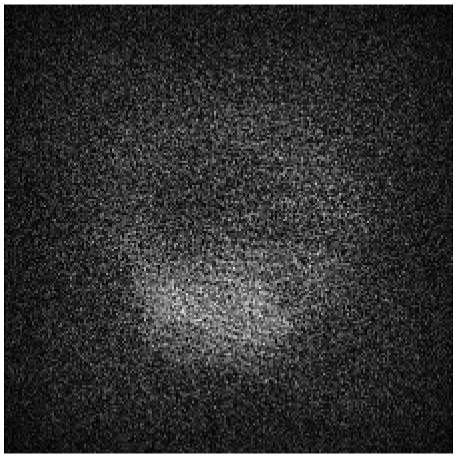
SNR=30dB

Compensated image	
Static image	
Uncompensated image	

SNR=20dB

Compensated image	
Static image	
Uncompensated image	

SNR=10dB

Compensated image	
Static image	
Uncompensated image	

Discussion

The compensation algorithm seems to work well when SNR is as low as 20bB. At SNR=10dB the image is so noisy that the compensation can not perform a good job although the image is still better than the uncompensated image.

In all the images (immediately seen in the SNR=40dB case) the compensated image shows some strange artifact which is like having some ‘texture’ in the background ! I think this may be due to the abnormal density variation in the stacked interleave of spirals after the rotation compensation. Figure 3,4 shows the stacked original interleaves before and after rotation compensation. Because the interleaves make random rotations after the motion compensation the final stacked trajectory is not uniform on the outer (and inner) spiral. Since the **outer** spiral was designed to give an FOV=24 when stacked together like in figure 3 this nonuniformity (figure 4) causes some aliasing in the spatial frequencies that the density is lower than expected. This is further justified by closely looking at the compensated image and seeing some vague edges of other parts in that background artifact.

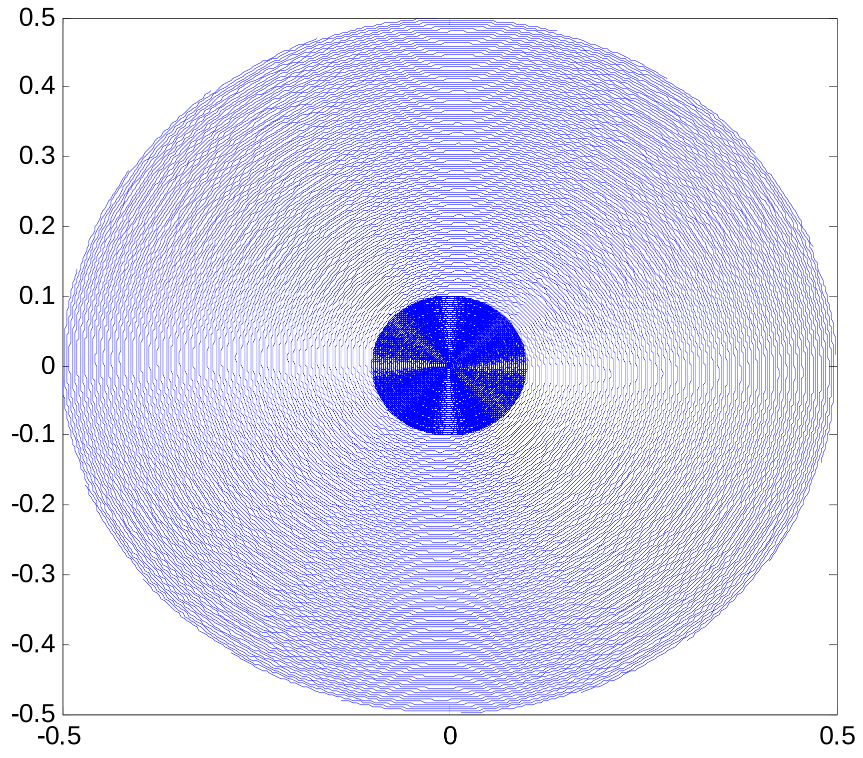


Figure 3.ALL interleaves before rotation compensation

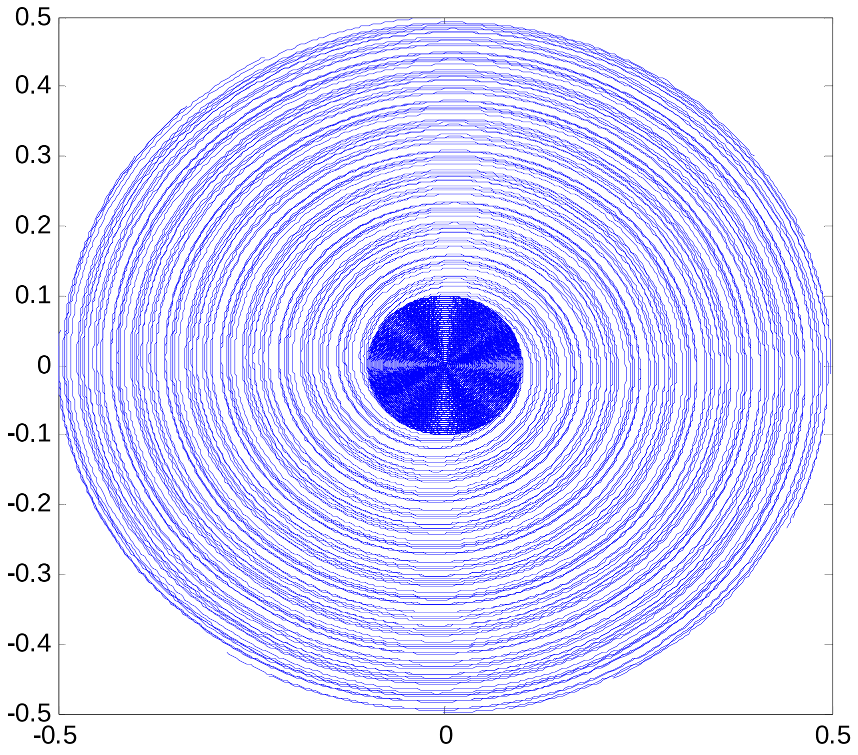


Figure 4. ALL interleaves after rotation compensation.

To wrap up, the bulk motion compensation scheme proposed works almost well for the simulated discrete movements of the object. However, the algorithm performance should be tested on real data to see the effects of non-rigid motion,flow,off-resonance and most importantly the spin phase effects during each readout. In the case of spin-phase artifacts lowering the ratio of inner/outer spiral coverage and increasing the number of interleaves to lower the acquisition time maybe useful. Also if the noise variance or distortion was not the same for each interleave we can use a weighting to vary the contribution of the interleaves' data to the final reconstruction.

References

[1] Ehman R, Felmlee J. Adaptive technique for high-definition MR imaging of moving structures. Radiology 198;173:255–263.

[2] J Pipe, "Motion Correction with PROPELLER-MRI: Application to Head Motion and Free-Breathing Cardiac Imaging" Magn. Reson. Med., 42:963-969, 1999.

[3] MS Sussman, JA Stainsby, N Robert, N Merchant, GA Wright, "Variable-Density Adaptive Imaging for High-Resolution Coronary Artery MRI" Magn. Reson. Med. 48:753-764, 2002.

[4] http://www-mrsrl.stanford.edu/~brian/vdspiral/

Appendix : MATLAB codes

MAIN.m

```
% generate the variable density spiral ;
k=genvds;
%generate the k-space data for the moving object
movingtx=buildmovingobj(k,pi/4,20);
%generate a static object for comparison
statictx=buildmovingobj(k,0,0);
%add noise

%10db
movingtx10db=movingtx+sqrt(0.1*mean(mean(abs(movingtx).^2)))*randn(4026,20);
statictx10db=statictx+sqrt(0.1*mean(mean(abs(statictx).^2)))*randn(4026,20);
%20db
movingtx20db=movingtx+sqrt(0.01*mean(mean(abs(movingtx).^2)))*randn(4026,20);
statictx20db=statictx+sqrt(0.01*mean(mean(abs(statictx).^2)))*randn(4026,20);
%30db
movingtx30db=movingtx+sqrt(0.001*mean(mean(abs(movingtx).^2)))*randn(4026,20);
statictx30db=statictx+sqrt(0.001*mean(mean(abs(statictx).^2)))*randn(4026,20);
%40db
movingtx40db=movingtx+sqrt(0.0001*mean(mean(abs(movingtx).^2)))*randn(4026,20);
statictx40db=statictx+sqrt(0.0001*mean(mean(abs(statictx).^2)))*randn(4026,20);
fix the rotation

%compute the voronoi areas for all interleaves
area=vorodensmulti(k);
%fix the rotation
k_fixed10db=fixrot(movingtx10db,k,area);
k_fixed20db=fixrot(movingtx20db,k,area);
k_fixed30db=fixrot(movingtx30db,k,area);
k_fixed40db=fixrot(movingtx40db,k,area);

% fix the translation
tx_fixed10db=fixtrans(movingtx10db,k_fixed10db,area);
tx_fixed20db=fixtrans(movingtx20db,k_fixed20db,area);
tx_fixed30db=fixtrans(movingtx30db,k_fixed30db,area);
tx_fixed40db=fixtrans(movingtx40db,k_fixed40db,area);

%reconstruct the final images

%compute the stacked interleaves weighting function
a=voronoidens(k(:));
a=fixa(a);
a_10db=voronoidens(k_fixed10db(:));
a_10db=fixa(a_10db);
a_20db=voronoidens(k_fixed20db(:));
a_20db=fixa(a_20db);
a_30db=voronoidens(k_fixed30db(:));
a_30db=fixa(a_30db);
a_40db=voronoidens(k_fixed40db(:));
a_40db=fixa(a_40db);

%motion corrected image
correctedim_10db=ift(grid(tx_fixed10db(:).*a_10db,k_fixed10db,512,5));
correctedim_20db=ift(grid(tx_fixed20db(:).*a_20db,k_fixed20db,512,5));
correctedim_30db=ift(grid(tx_fixed30db(:).*a_30db,k_fixed30db,512,5));
correctedim_40db=ift(grid(tx_fixed40db(:).*a_40db,k_fixed40db,512,5));
%moving image if no compensation
uncorrectedim_10db=ift(grid(movingtx10db(:).*a,k,512,5));
uncorrectedim_20db=ift(grid(movingtx20db(:).*a,k,512,5));
uncorrectedim_30db=ift(grid(movingtx30db(:).*a,k,512,5));

uncorrectedim_40db=ift(grid(movingtx40db(:).*a,k,512,5));
%reconstructed static image for baseline comparison
staticim_10db=ift(grid(statictx10db(:).*a,k,512,5));
staticim_20db=ift(grid(statictx20db(:).*a,k,512,5));
```

```
staticim_30db=ift(grid(statictx30db(:).*a,k,512,5));
staticim_40db=ift(grid(statictx40db(:).*a,k,512,5));

%display the images
figure(1);
dispimage(abs(correctedim_40db(128:383,128:383)));
title('correctedim_40db');

figure(2);
dispimage(abs(staticim_40db(128:383,128:383)));
title('staticim_40db');

figure(3);
dispimage(abs(uncorrectedim_40db(128:383,128:383)));
title('uncorrectedim_40db');

figure(4);
dispimage(abs(correctedim_30db(128:383,128:383)));
title('correctedim_30db');

figure(5);
dispimage(abs(staticim_30db(128:383,128:383)));
title('staticim_30db');

figure(6);
dispimage(abs(uncorrectedim_30db(128:383,128:383)));
title('uncorrectedim_30db');

figure(7);
dispimage(abs(correctedim_20db(128:383,128:383)));
title('correctedim_20db');

figure(8);
dispimage(abs(staticim_20db(128:383,128:383)));
title('staticim_20db');

figure(9);
dispimage(abs(uncorrectedim_20db(128:383,128:383)));
title('uncorrectedim_20db');

figure(10);
dispimage(abs(correctedim_10db(128:383,128:383)));
title('correctedim_10db');

figure(11);
dispimage(abs(staticim_10db(128:383,128:383)));
title('staticim_10db');

figure(12);
dispimage(abs(uncorrectedim_10db(128:383,128:383)));
title('uncorrectedim_10db');
```

GENVDS.m

```
function [k]=genvds;
clear kk;
fov=24;
N=20;           % # of interleaves
smax=14500;     % max slew rate
gmax=3.9;       % max gradient amplitude
T=0.000004;     % sampling time
rmax=5;         % max k-space raduis to be covered ( resolution = 1mm )

%generate one interleave
[k,g,s,time,r,theta] = vds(smax,gmax,T,N,fov,0,0,rmax);
r=abs(k);
theta=angle(k);
```

```
%generate all interleaves
for i=0:(N-1),
     kk(:,i+1)=r'.*exp(j*(theta-i*2*pi/N))';
end

%downsample the inner spiral ( will not violate G,S limitations )
k=[kk(1:20:36200,:); kk(36201:end,:)];

%scale to -0.5<k<0.5
k=k/10;
```

VDS.m

```
%
%       function [k,g,s,time,r,theta] = vds(smax,gmax,T,N,F0,F1,F2,rmax)
%
%
%       VARIABLE DENSITY SPIRAL GENERATION:
%       ----------------------------------
%
%       Function generates variable density spiral which traces
%       out the trajectory
%
%                       k(t) = r(t) exp(i*q(t)),             [1]
%
%       Where q is the same as theta...
%               r and q are chosen to satisfy:
%
%               1) Maximum gradient amplitudes and slew rates.
%               2) Maximum gradient due to FOV, where FOV can
%                  vary with k-space radius r, as
%
%                       FOV(r) = F0 + F1*r + F2*r*r          [2]
%
%
%       INPUTS:
%       -------
%       smax = maximum slew rate G/cm/s
%       gmax = maximum gradient G/cm (limited by Gmax or FOV)
%       T = sampling period (s) for gradient AND acquisition.
%       N = number of interleaves.
%       F0,F1,F2 = FOV coefficients with respect to r - see above.
%       rmax= value of k-space radius at which to stop (cm^-1).
%               rmax = 1/(2*resolution);
%
%
%       OUTPUTS:
%       --------
%       k = k-space trajectory (kx+iky) in cm-1.
%       g = gradient waveform (Gx+iGy) in G/cm.
%       s = derivative of g (Sx+iSy) in G/cm/s.
%       time = time points corresponding to above (s).
%       r = k-space radius vs time (used to design spiral)
%       theta = atan2(ky,kx) = k-space angle vs time.
%
%
%       METHODS:
%       --------
%       Let r1 and r2 be the first derivatives of r in [1].
%       Let q1 and q2 be the first derivatives of theta in [1].
%       Also, r0 = r, and q0 = theta - sometimes both are used.
%       F = F(r) defined by F0,F1,F2.
%
%       Differentiating [1], we can get G = a(r0,r1,q0,q1,F)
%       and differentiating again, we get S = b(r0,r1,r2,q0,q1,q2,F)
%
%       (functions a() and b() are reasonably easy to obtain.)
%
%       FOV limits put a constraint between r and
```

```
%
%               dr/dq = N/(2*pi*F)                          [3]
%
%       We can use [3] and the chain rule to give
%
%               q1 = 2*pi*F/N * r1                          [4]
%
%       and
%
%               q2 = 2*pi/N*dF/dr*r1^2 + 2*pi*F/N*r2        [5]
%
%
%
%       Now using [4] and [5], we can substitute for q1 and q2
%       in functions a() and b(), giving
%
%               G = c(r0,r1,F)
%       and     S = d(r0,r1,r2,F,dF/dr)
%
%
%       Using the fact that the spiral should be either limited
%       by amplitude (Gradient or FOV limit) or slew rate, we can
%       solve
%               |c(r0,r1,F)| = |Gmax|                               [6]
%
%       analytically for r1, or
%
%               |d(r0,r1,r2,F,dF/dr)| = |Smax|                      [7]
%
%       analytically for r2.
%
%       [7] is a quadratic equation in r2.  The smaller of the
%       roots is taken, and the real part of the root is used to
%       avoid possible numeric errors - the roots should be real
%       always.
%
%       The choice of whether or not to use [6] or [7], and the
%       solving for r2 or r1 is done by q2r21.m.
%
%       Once the second derivative of theta(q) or r is obtained,
%       it can be integrated to give q1 and r1, and then integrated
%       again to give q and r.  The gradient waveforms follow from
%       q and r.
%
%       Brian Hargreaves -- Sept 2000.
%
%       See Brian's journal, Vol 6, P.24.
%
%
%       See also:  vds2.m
%

% =============== CVS Log Messages ==========================
%       $Log: vds.m,v $
%       Revision 1.3  2002/11/18 05:36:02  brian
%       Rounds lengths to a multiple of 4 to avoid
%       frame size issues later on.
%
%       Revision 1.2  2002/11/18 05:32:19  brian
%       minor edits
%
%       Revision 1.1  2002/03/28 01:03:20  bah
%       Added to CVS
%
%
% ===========================================================

function [k,g,s,time,r,theta] = vds(smax,gmax,T,N,F0,F1,F2,rmax)

disp('vds.m');
gamma = 4258;
```

```
oversamp = 8;          % Keep this even.
To = T/oversamp;       % To is the period with oversampling.



q0 = 0;
q1 = 0;
theta = zeros(1,10000);
r = zeros(1,10000);
r0 = 0;
r1 = 0;

time = zeros(1,10000);
t = 0;
count = 1;

theta = zeros(1,1000000);
r = zeros(1,1000000);
time = zeros(1,1000000);

while r0 < rmax
	[q2,r2] = q2r21(smax,gmax,r0,r1,To,T,N,[F0,F1,F2]);

	% Integrate for r, r', theta and theta'
	q1 = q1 + q2*To;

	q0 = q0 + q1*To;
	t = t + To;

	r1 = r1 + r2*To;
	r0 = r0 + r1*To;

	% Store.
	count = count+1;
	theta(count) = q0;
	r(count) = r0;
	time(count) = t;

	if (rem(count,100)==0)
		tt = sprintf('%d points, |k|=%f',count,r0);
		disp(tt);
	end;
end;

r = r(oversamp/2:oversamp:count);
theta = theta(oversamp/2:oversamp:count);
time = time(oversamp/2:oversamp:count);

%	Keep the length a multiple of 4, to save pain...!
%
ltheta = 4*floor(length(theta)/4);
r=r(1:ltheta);
theta=theta(1:ltheta);
time=time(1:ltheta);

k = r.*exp(i*theta);

g = 1/gamma*([k 0]-[0 k])/T;
g = g(1:length(k));

s = ([g 0]-[0 g])/T;
s = s(1:length(k));
```

q2r21.m

```
%
%	function [q2,r2] = q2r2(smax,gmax,r,r1,T,Ts,N,F)
```

```
%
%       VARIABLE DENSITY SPIRAL DESIGN ITERATION
%       ----------------------------------------
%       Calculates the second derivative of r and q (theta),
%       the slew-limited or FOV-limited
%       r(t) and q(t) waveforms such that
%
%               k(t) = r(t) exp(i*q(t))
%
%       Where the FOV is a function of k-space radius (r)
%
%       FOV = F(1) + F(2)*r + F(3)*r*r  + ... ;
%
%       F(1) in cm.
%       F(2) in cm^2.
%       F(3) in cm^3.
%       .
%       .
%       .
%
%       The method used is described in vds.m
%
%       INPUT:
%       -----
%       smax    = Maximum slew rate in G/cm/s.
%       gmax    = Maximum gradient amplitdue in G.
%       r       = Current value of r.
%       r1      = Current value of r', first derivative of r wrt time.
%       T       = Gradient sample rate.
%       Ts      = Data sampling rate.
%       N       = Number of spiral interleaves.
%       F is described above.
%


% =============== CVS Log Messages ==========================
%       This file is maintained in CVS version control.
%
%       $Log: q2r21.m,v $
%       Revision 1.1  2002/03/28 01:03:20  bah
%       Added to CVS
%
%
% ===========================================================



function [q2,r2] = q2r2(smax,gmax,r,r1,T,Ts,N,Fcoeff)

gamma = 4258;                 % Hz/G

F = 0;          % FOV function value for this r.
dFdr = 0;               % dFOV/dr for this value of r.



% for rind = 1:length(Fcoeff)
%       F = F+Fcoeff(rind)*r^(rind-1);
%       if (rind>1)
%               dFdr = dFdr + (rind-1)*Fcoeff(rind) * r^(rind-2);
%       end;
% end;


% this line modifies the desired trajectory FOV to be simply a step-wise function rather
than a polynomial
if r>1, F=24; else, F=20*24; end
```

```
GmaxFOV = 1/gamma /F/Ts;              % FOV limit on G
Gmax = min(GmaxFOV,gmax);     %

maxr1 = sqrt((gamma*Gmax)^2 / (1+(2*pi*F*r/N)^2));


if (r1 > maxr1)
                      % Grad amplitude limited.  Here we

                      % just run r upward as much as we can without
                      % going over the max gradient.
       r2 = (maxr1-r1)/T;
       %tt = sprintf('Grad-limited r=%5.2f, r1=%f',r,r1);
       %disp(tt);

else

       twopiFoN = 2*pi*F/N;
       twopiFoN2 = twopiFoN^2;

       %      A,B,C are coefficents of the equation which equates
       %      the slew rate calculated from r,r1,r2 with the
       %      maximum gradient slew rate.
       %
       %      A*r2*r2 + B*r2 + C  =  0
       %
       %      A,B,C are in terms of F,dF/dr,r,r1, N and smax.
       %


       A = 1+twopiFoN2*r*r;
       B = 2*twopiFoN2*r*r1*r1 + 2*twopiFoN2/F*dFdr*r*r*r1*r1;
       C = twopiFoN2^2*r*r*r1^4 + 4*twopiFoN2*r1^4 + (2*pi/N*dFdr)^2*r*r*r1^4 +
4*twopiFoN2/F*dFdr*r*r1^4 - (gamma)^2*smax^2;


       [rts] = qdf(A,B,C);    % qdf = Quadratic Formula Solution.
       r2 = real(rts(1));     % Use bigger root.  The justification
                              % for this is not entirely clear, but
                              % in practice it seems to work, and
                              % does NOT work with the other root.




       % Calculate resulting slew rate and print an error
       % message if it is too large.

       slew = 1/gamma*(r2-twopiFoN2*r*r1^2 + i*twopiFoN*(2*r1^2 + r*r2 + dFdr/F*r*r1^2));
       %tt = sprintf('Slew-limited r=%5.2d  SR=%f G/cm/s',r,abs(slew));
       %disp(tt);
       sr = abs(slew)/smax;

       if (abs(slew)/smax > 1.01)
              tt = sprintf('Slew violation, slew = %d, smax = %d, sr=%f, r=%f,
r1=%f',round(abs(slew)),round(smax),sr,r,r1);
              disp(tt);
       end;

end;


%      Calculate q2 from other pararmeters.

q2 = 2*pi/N*dFdr*r1^2 + 2*pi*F/N*r2;
```

BUILDMOVINGOBJ.M

```
function [tx]=buildmovingobj(k,thetarange,shiftrange);
```

```
N=20; % # of interleaves
for i=1:N,
    i
    j=sqrt(-1);
    kx=real(k(:,i));
    ky=imag(k(:,i));
    kr=abs(k(:,i));
    phi=angle(k(:,i));

    %random rotation angle -pi/4 > pi/4
    rot=(rand(1)-0.5).*2*thetarange;
    if i==1,rot=0; end
    %generate the rotation
    t=buildobj(kr.*exp(j*(phi-rot)));
    % random shifts of -20>20 pixels -2>2 cm
    shiftx=2*shiftrange*(rand(1)-0.5);
    shifty=2*shiftrange*(rand(1)-0.5);
    %generate the shifts
    t=t.*exp(-j*2*pi*kx*shiftx-j*2*pi*ky*shifty);
    tx(:,i)=t;
end
```

BUILDOBJ.M

```
function [tx]=buildobj(k);
%krange = (-128:127)/256;
%[ky kx] = meshgrid(krange,krange);
kx=real(k);
ky=imag(k);
kr = sqrt(kx.^2 + ky.^2);
tx_circ = 240*240*jinc(240*kr); % FTx of circle with diameter 24cm(all FOV)

% Make some Squares on the outside
 tx_grid = 0;

 tx_grid = tx_grid + rectangle_fourier(kx,ky,40,40,0,75);
 tx_grid = tx_grid + rectangle_fourier(kx,ky,40,40,-75,0);
 tx_grid = tx_grid + rectangle_fourier(kx,ky,20,20,0,0);
 tx_grid = tx_grid + rectangle_fourier(kx,ky,40,40,0,-75);
 tx_grid = tx_grid + rectangle_fourier(kx,ky,40,40,75,0);
 tx_grid = tx_grid + rectangle_fourier(kx,ky,40,40,43,43);
 tx_grid = tx_grid + rectangle_fourier(kx,ky,40,40,43,-43);
 tx_grid = tx_grid + rectangle_fourier(kx,ky,40,40,-43,-43);

 tx_grid = tx_grid/2;

% Make some Stripes that have high spatial resolution
tx_stripes = 0;
start = -65;
for a=1:8;
start = start + a/2;
tx_stripes = tx_stripes + rectangle_fourier(kx,ky,a,64,start,32);
start = start + 3*a/2;
end
% Make a big oval (for no real reason)
tx_oval = ellipse_fourier(kx,ky,60,100,40,0);
% Make a thin rectangle
tx_pen = rectangle_fourier(kx,ky,2,100,-30,-62);
% put it all together
tx = (tx_pen+tx_grid + tx_stripes + tx_oval);
%tx = (tx_pen+tx_circ/3+tx_grid + tx_stripes + tx_oval);
%im = ift(tx);
%dispimage(abs(im));
```

VORODENSMULTI.M

```
%compute voronoi areas for multiple interleaves
```

```
function area=vorodensmulti(k),
N=20;
for i=1:N,
    i
    a=voronoidens(k(:,i));
    a=fixaa(a);
    area(:,i)=a;
end
```

VORONOIDENS.M

```
function area = voronoidens(k);
%
% function area = voronoidens(k);
%
% input:  k = kx + i ky is the  k-space trajectory
% output: area of cells for each point
%           (if point doesn't have neighbors the area is NaN)

kx = real(k);
ky = imag(k);

[row,column] = size(kx);
kxy = [kx(:),ky(:)];
% returns vertices and cells of voronoi diagram
[V,C] = voronoin(kxy);
area = [];
for j = 1:length(kxy)
  j
  x = V(C{j},1); y = V(C{j},2); lxy = length(x);
  A = abs(sum( 0.5*(x([2:lxy 1]) - x(:)).*(y([2:lxy 1]) + y(:))));
  area = [area A];
end
area = reshape(area,row,column);
```

```
% this is to fix the area output from the voronoi diagram and remove
the
% problem in the edge of the spiral. This is to fix all interleaves
stacked upon
% each other.
```

FIXA.M

```
function b=fixa(a);
x=3850:4026;
y=2*3850-4026-1:3850-1;
a(x)=a(y)+a(x(1))-a(y(1));
for i=1:19,
x=x+4026;
y=y+4026;
a(x)=a(y)+a(x(1))-a(y(1));
end
b=a;
```

FIXAA.M

```
% this is to fix the area output from the voronoi diagram and remove the
% problem in the edge of the spiral.
function b=fixaa(a)
a(3451:4026)=a(3451-(4026-3450):3450)+a(3450)-a(2*3450-4026);
b=a;
```

GRID.M

```
function m = grid(d,k,n,L)

%     d -- k-space data
%     k -- k-trajectory, scaled -0.5 to 0.5
%     n -- image size
%     L -- number of cartesian samples falling in the kernel

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

% cconvert 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 = -L:L,
  for ly = -L:L,

    % find nearest samples
    nxt = round(nx+lx);
    nyt = round(ny+ly);

    % compute weighting for triangular kernel
    kwx = sinc(((nx-nxt)/2)).*(0.54-0.46*cos(2*pi*(nx-nxt-L)/(2*L)));
    kwy = sinc(((ny-nyt)/2)).*(0.54-0.46*cos(2*pi*(ny-nyt-L)/(2*L)));
    kwx(find(abs(nx-nxt)>L))=0;
    kwy(find(abs(ny-nyt)>L))=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;
```

FIXROT.M

```
% fixes bulk rotation from inner spiral data
function [kout]=fixrot(tx,kk,area);
N=20;
krange = (-64:63)/128/5;
[ky kx] = meshgrid(krange,krange);
kr_grid = kx.^2 + ky.^2;
bigr=find(kr_grid>0.01);
for i=1:N,
    i
    %select inner spiral data
    k=kk(1:1812,i);
    t=tx(1:1812,i);
    a=area(1:1812,i);
    %grid i'th inner spiral interleave to a 64x64 cartesian grid
    tmp=grid(abs(t).*a,5*k,128,5);
    %set outside circle data to zero
    tmp(bigr)=0;
    %m is the gridded k-space image from ith interleave of the inner spiral
    m(:,:,i)=tmp;
end
% now search for angle
for i=1:N,
    i
    k=kk(1:1812,i);
    t=tx(1:1812,i);
    a=area(1:1812,i);
    kr=abs(k);
    phi=angle(k);
    j=sqrt(-1);
    sigma=[];
    %first interation search
    for th=-pi/4:0.04:(0.04+pi/4),
        k_rotated=kr.*exp(j*(phi-th));
        tmp=kr_grid.*grid(abs(t).*a,5*k_rotated,128,5);
        tmp(bigr)=0;
        m_rot=tmp;
        %sigma is the correlation function
        sigma(end+1)=sum(sum(m_rot.*m(:,:,1).*kr_grid));
    end
    [tmp,ind]=max(sigma);
    th=[-pi/4:0.04:(0.04+pi/4)];
    theta=th(ind)
    sigma=[];
    %second iteration finer search
    for th=theta-0.1:0.01:theta+0.1,
        k_rotated=kr.*exp(j*(phi-th));

        tmp=kr_grid.*grid(abs(t).*a,5*k_rotated,128,5);
        tmp(bigr)=0;
        m_rot=tmp;
        sigma(end+1)=sum(sum(m_rot.*m(:,:,1).*kr_grid));
    end
    [tmp,ind]=max(sigma);
    th=[theta-0.1:0.01:theta+0.1];
    theta=th(ind)
    sigma=[];
    % third iteration finest search with 0.001 precision
    for th=theta-0.02:0.001:theta+0.02,
        k_rotated=kr.*exp(j*(phi-th));
        tmp=kr_grid.*grid(abs(t).*a,5*k_rotated,128,5);
        tmp(bigr)=0;
        m_rot=tmp;
        sigma(end+1)=sum(sum(m_rot.*m(:,:,1).*kr_grid));
    end
    [tmp,ind]=max(sigma);
    th=[theta-0.02:0.001:theta+0.02];
    theta=th(ind)
    %update the k-space trajectory for this interleave using estimated rotation
    kout(:,i)=abs(kk(:,i)).*exp(j*(angle(kk(:,i))-theta));
end
```

FIXTRANS.M

```
% fixes the translation artifact
function [txout]=fixtrans(tx,kk,area);
N=20;
%range for inner spiral k-space data
krange = (-64:63)/128/5;
[ky kx] = meshgrid(krange,krange);
kr_grid = kx.^2 + ky.^2;
bigr=find(kr_grid>0.01);
% generate gridded data
```

```
for i=1:N,
    i
    k=kk(1:1812,i);
    t=tx(1:1812,i);
    a=area(1:1812,i);
    tmp=grid(t.*a,5*k,128,5);
    tmp(bigr)=0;
    m(:,:,i)=tmp;
end
j=sqrt(-1);
% find and correct the shifts
for i=1:N,
    % find the convolution
    d=ift(m(:,:,i).*conj(m(:,:,1)));
    d=abs(d);
    % find the peak
    [a,b]=ind2sub(size(d),find(d==max(max(d))));
    % refine the peak by sinc interpolation around the peak
    dx=d(a-5:a+5,b);
    dy=d(a,b-5:b+5);
    tmp=interp(dx,20);
    [aa,ind]=max(tmp);
    shiftx=a-65+((ind-1)/20)-5 % this is the shift in x direction
    tmp=interp(dy,20);
    [bb,ind]=max(tmp);
    shifty=b-65+((ind-1)/20)-5; % this is the shift in y direction
    % subtract the liear phase from the k-space data
    kx=real(kk(:,i));ky=imag(kk(:,i));
    txout(:,i)=tx(:,i).*exp(5*j*2*pi*shiftx*kx).*exp(5*j*2*pi*shifty*ky);
end
```