BIM 241 Final Report

Nonrigid Motion Correction in 3D Using Autofocusing with Localized Linear Translations

Ciara Jekel, Sujoy Ghosh, Yufei D Zhu

Introduction

Because of their long duration, MR scans are sensitive to motion. Historically, numerous methods have been proposed to correct for this including models with elements such as translation, rotation, shear, and scaling. However, these techniques are often computationally intensive and difficult to implement. This paper uses an autofocusing algorithm that locally minimizes a given motion metric and is predicated on the assumption that, on a sufficiently small spatial scale, the different types of motion can be well approximated as simple linear translations. Possible motion paths are limited to motion measure from multichannel navigator data to reduce the search space. This novel navigation strategy is based on the so-called “Butterfly” navigators, which is a modification of the spin-warp sequence in which the prewinder gradients for phase encodes are slightly modified to traverse the same trajectory at the beginning of each data acquisition. This makes it possible to obtain translational motion estimates with high temporal resolution. With a 32-channel abdominal coil, sufficient number of motion measurements were found to approximate possible linear motion paths for every voxel. This motion correction scheme was performed on two pediatric patients and results showed an overall reduction in artifacts from complex, nonrigid motion.

Methods

Motion is first measured using the Butterfly sequence during the data acquisition. This effectively confines the search space and limits the computational load the program has to estimate motion with. In our study, a spin-war sequence with multiple repetition time is used. The Butterfly sequence effectively modifies the prewinder portion of the timing diagram and collects motion data during a very short period of time. It is named as such because of the resulting trajectory shape resembling a butterfly wing. After the data is collected in k-space, a series of post processing steps must occur before the information can be adjusted in the image domain.

The first of these steps is the motion correction. As a simplification, nonrigid motion is approximated as local linear translations rather than rotation, contraction/expansion, and other complicated transformations. Linear phase correction to all the acquired k-space data is performed with motion measurements from each coil. After correcting for different motion measurements, the next step is to determine which correction yielded the best result for a particular location. For this, gradient entropy is used as a good metric for motion artifacts. This criterion is minimized when the image consists of uniform brightness separated by sharp edges and has been found to be a good model for normal MR scans.

To obtain local translational data, the gradient entropy evaluation is performed on windowed sections of the scan. The scan is multiplied by a 3D Hanning window before performing the gradient entropy calculations. The size of this window affects the autofocusing selection. Windows sized too small for local anatomy can amplify motion artifacts, while larger windows allow more blurring from motion. Artifacts from fat at the edge of the body were present in the 2cm window and noise outside the body is large. At a 14cm window, the upper liver has an artifact, and the arteries are overly blurred. A window size of 10cm was selected for a sufficient balance between image sharpness and artifacts. A comparison of the effects can be seen in Figure 1.

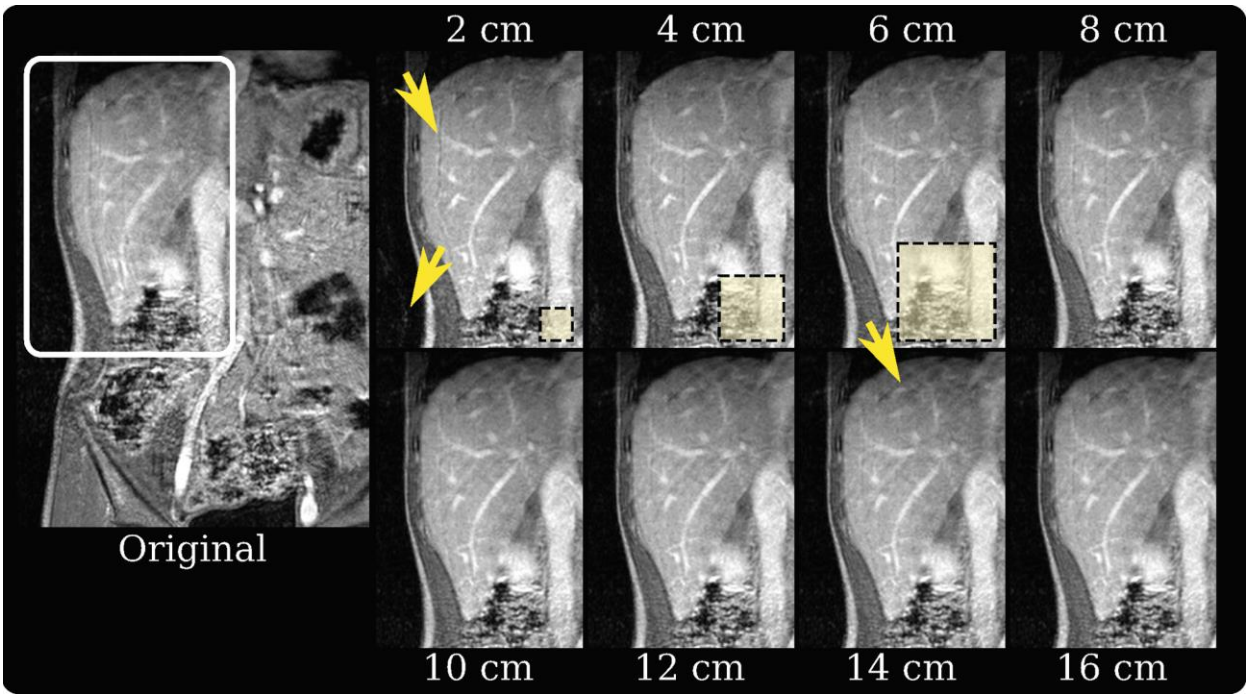


Figure 1: Effect of different window sizes on motion corrected scans. Artifacts from fat can be seen clearly in the 2cm window, as well as the elevated noise outside the body. In the 14cm window, a ghosting artifact appears in the upper section of the liver. The 10cm window was selected for good balance between sharpness of the arteries and a lack of significant artifacts.

The validity of the motion estimates was assessed with multiple phantom scans. On a stationary phantom, the motion estimates were confirmed to be stable. A drift of less than 250 µm was observed over 170s. A phantom moving in a periodically repeating linear pattern was used to verify the accuracy of the motion estimates against rigid motion. The algorithm accurately measured and successfully corrected for the motion of the phantom.

Studies conducted

The autocorrection algorithms explained in the previous sections were tested on an abdominal study performed on a 6-year-old patient using a three-dimensional spoiled gradient-recalled echo acquisition sequence. As shown in Fig. 2, translational maps of the signals recorded from the sample show translation along x, y and z axis. The maps indicated variation of signals from the coils for respiratory motion with some coils recording stationary signals originaying from the lower abdominal area which are usually stationary during respiratory motion. It is intuitive to think that motion in any one part of the body shall influence signal only on the coil in the immediate vicinity of the part. One of the major observations of the experiment was the fact that when motion occurs, the signal change is not only measured by the local coil but also registered in reduced magnitudes for adjacent coils as well. This has major implications in the field of MR signal processing as useful data for the sample can also be extracted from adjacent coils. Similar observations can be made in the last row of Fig. 3 where the number of coils activated during any type of motion result in signal reception for multiple coils. Also, it was found that correction using the nearest coil doesn't yield the best results which indicate that motion artifact correction is a multicoil phenomenon. The results were also free from ghosting artifacts produced due to the movement of tissue/ fluid during the scan. There was some amount of blurring involved from corrupted higher k space frequencies which were later corrected using homodyne reconstruction. Fig. 4 showed a similar trend with reduced artifacts and motion correction. In some areas with low signal to noise ratio, small amount of motion was detected by the algorithm. Whereas some artifacts were uncorrect as the algorithm was unable to detect the motion paths for these artifacts. Increasing the search space can help solve this issue. Therefore the study concluded that the best dataset to implement the

autocorrection on butterfly trajectory is one where there are appropriate number of coils relative to the size of the subject which ensures an appropriate signal to noise ratio.

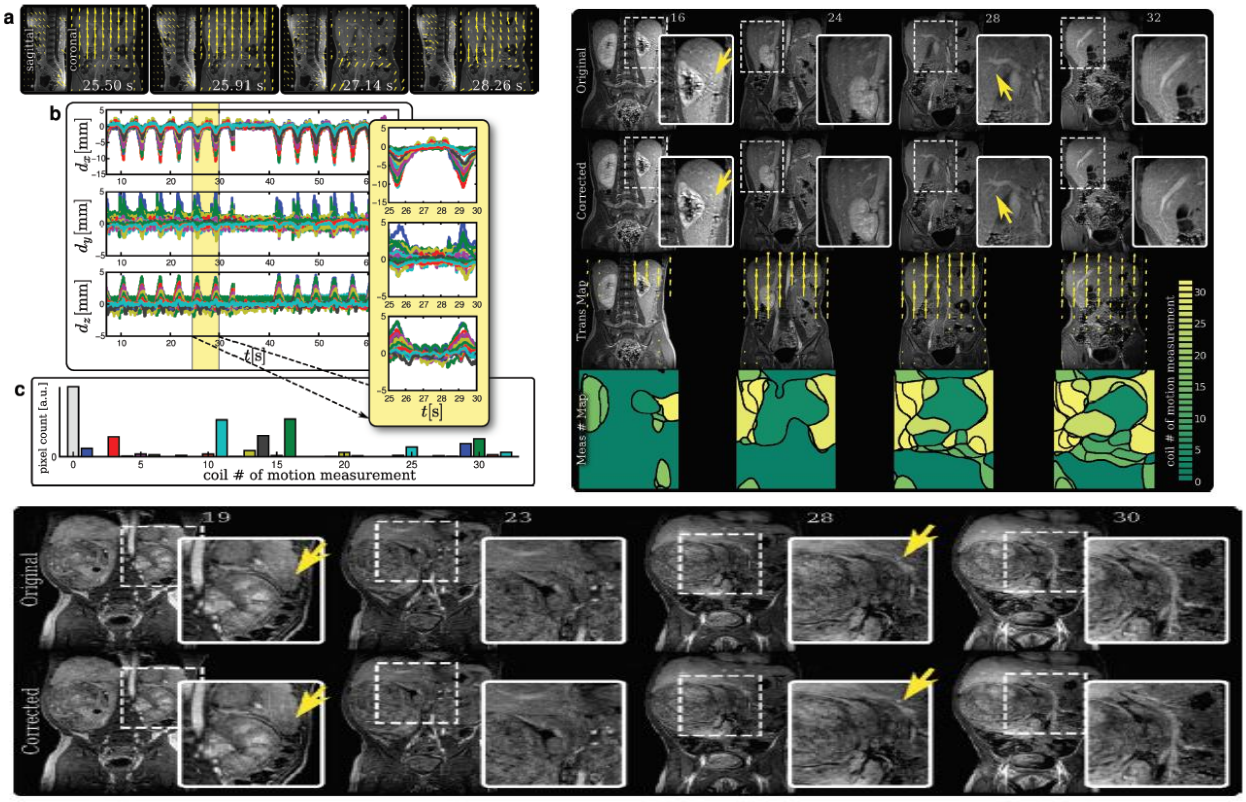


Fig. 2, 3 and 4 (clockwise)

David's Analysis

Introduction

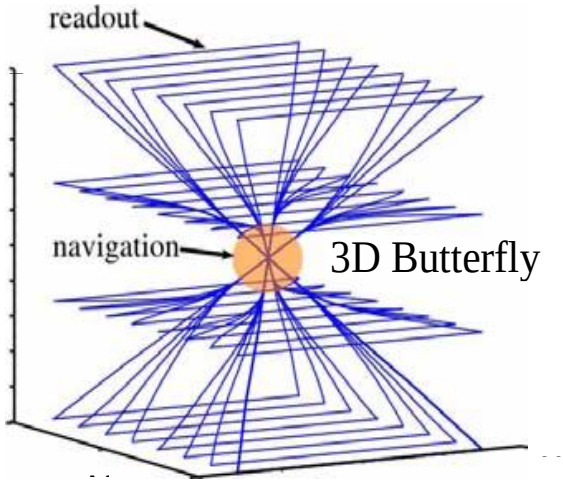
For my individual exploration, I was interested in adjusting the navigator trajectory and analyzing the comparative time penalty. To do this, it was important to identify which part of the code was able to modify the trajectory data as well as which part contained information regarding time. For my update summary, I initially did a cursory read of all the script files searching for the term 'time' and identified a few candidate scripts. However, the code was written with such a complexity that I didn't feel comfortable modifying its content without major errors arising. Therefore, after some guidance, I decided to stick with changing the example1.mat data, the contents of which are shown in the figure below.

```
% The following data structures are stored in the .mat file:
%   DATAA          -- raw k-space data with size nx*ny*nz*nc:
%                        nx -- readout length
%                        ny -- number of phase-encodes
%                        nz -- number of slice-encodes
%                        nc -- number of coils (receivers)
%   NAVA           -- raw navigator data with size nnav*nall:
%                        nnav -- navigator length
%                        nall -- total number of readouts x nc (= ny*nz*nc)
%   TR             -- [us] repitition time
%   k              -- [1/cm] k-space trajectory for navigator data
%   nFRead         -- reconstruction dimension for x
%                     (for partial k-space imaging in x)
%   res            -- [mm] image resolution
%   yorder, zorder -- y-phase encode and z-phase encode ordering
```

My question associated with the above .mat file was: what is the time consequence for each coil if I reduced the number of k trajectories by 1/6? 1/3? Or 1/2? My hypothesis was that the measurement time along with the autofocusing time would decrease because there would be less data for the algorithm to process. However, as we will see in the results section. This was not the case for the autofocusing data.

Methods

Specifically, I targeted the NAVA and k matrices within the example1.mat file as these seem to be most relevant to my goal. Using the size function in MATLAB, I determined that the NAVA matrix has dimension of 18x425984 while the k matrix has a dimension of 1x18. The 425984 dimension of NAVA corresponds to the number of phase encodes (ny) times the number of slice encodes (nz) times the number of coils (nc) as defined for the DATAA matrix. From this, I concluded that the NAVA matrix contains all the motion information captured along a three-dimensional (3D) k space trajectory, as specified by the k matrix. To put it visually, the NAVA matrix contains all the information collected along the blue lines in the following figure (Lustig et al, 2007).



For my code modification, I deleted 1/6, 1/3, and 1/2 of the rows in both NAVA and k. The code for deleting 1/6 of rows is shown below. These fractions were chosen based on their multiplication of the number 18, which is the common dimension for both NAVA and k. I also tried to perform column deletion in a similar manner. However, the algorithms downstream did not like this modification.

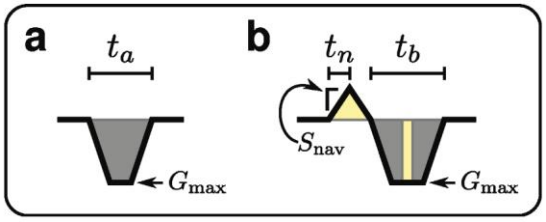
```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Manipulating the Data (YDZ Modification)

% Deleting all the even rows in both k and NAVA
 k(:,(1:6:18))=[];
 NAVA((1:6:18),:)=[];
```

Results

Imaging Time Consequences

First, I will analyze the theoretical time penalty of deleting the alternating proportions of the butterfly sequence. The following figure shows the timing diagram of the original prewinder sequence (a) and the modified prewinder sequence with the Butterfly modification in yellow (b).



The figure above is used to derive the time difference, t_p, between the modified (b) and original (a) prewinder sequence. The overall imaging time consequence of deleting n rows of k and NAVA would just be $n*t_p$.

To derive t_p, we are first going to set up the equations based on the areas of the figure above. This gives us the following:

$$t_n^2 S_{\text{nav}} = (t_b - t_a) G_{\text{max}} \qquad \text{[A1a]}$$

$$t_b = t_n^2 \frac{S_{\text{nav}}}{G_{\text{max}}} + t_a \qquad \text{[A1b]}$$

Next, we are going to set up the total time, t_{total}, variable for (b) and combine that with equation A1b. Notice that there is an extra t_n term. The reason for this is that the navigator data are acquired along one gradient axis. This gives us:

$$t_{\text{total}} = t_n + 2t_n + t_b \qquad \text{[A2a]}$$

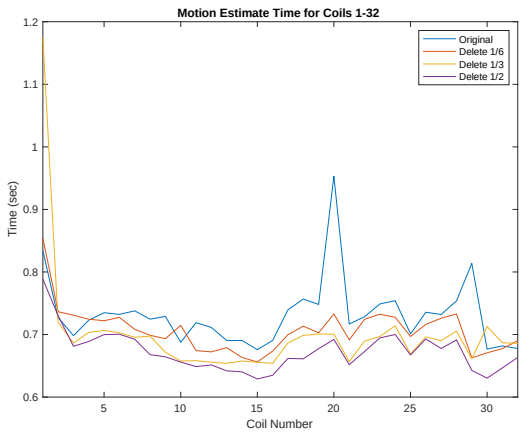
$$= 3t_n + t_n^2 \frac{S_{\text{nav}}}{G_{\text{max}}} + t_a. \qquad \text{[A2b]}$$

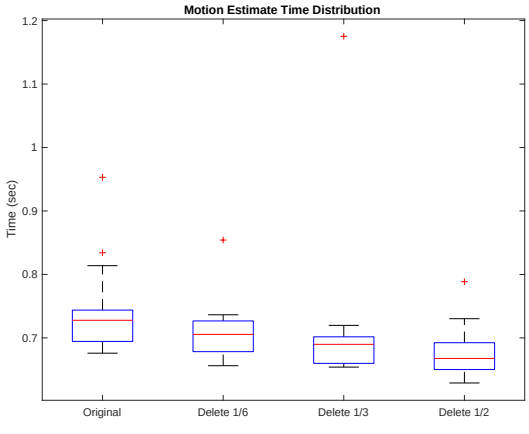
Finally, after modifying A2b, we get t_p (t_{total}-t_a) to be the following:

$$t_p = 3t_n + t_n^2 \frac{S_{\text{nav}}}{G_{\text{max}}}. \qquad \text{[A3]}$$

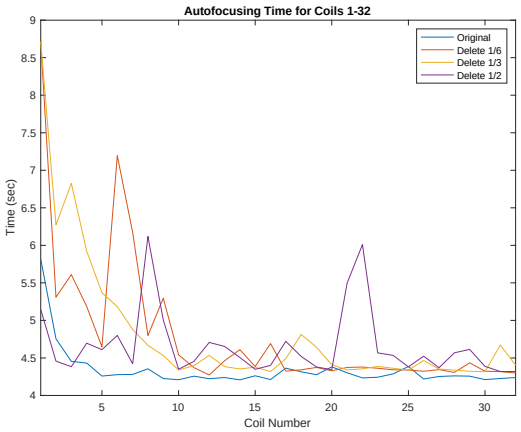
Post-Processing Time Consequences

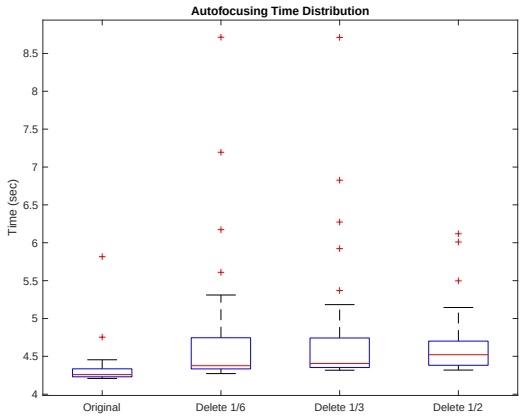
To analyze the consequence of deleting alternating numbers of k and NAVA on post-processing time, I plotted the time outputs of the demo.m script. The values for time of motion estimate for coils 1-32 are plotted in the following two figures.



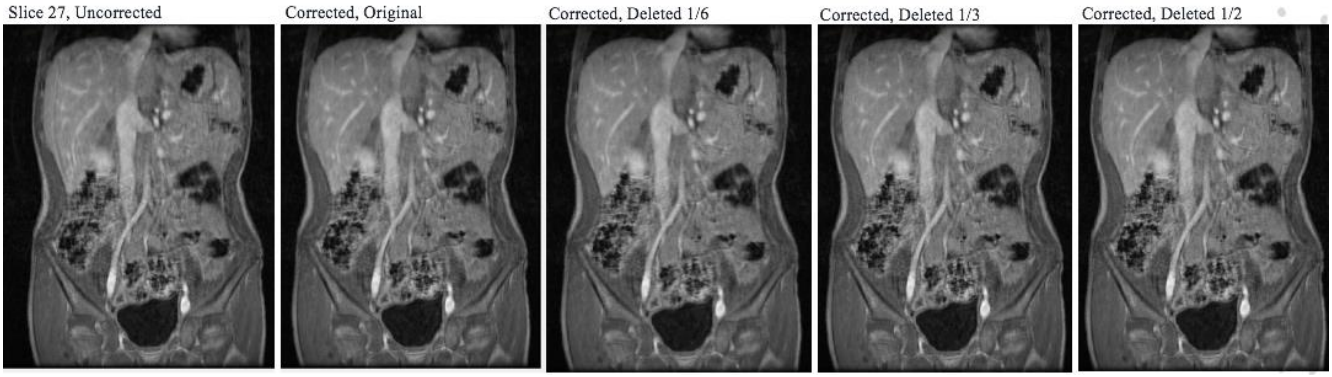


The time consequences for autofocusing, or the gradient entropy calculation, are shown in the following plots:





Additionally, the uncorrected and corrected image of a single slice of the image space is also shown in the following figure:



Discussion

Perhaps the most interesting finding in the results section is the gradually increasing trend in the autofocusing time as more and more data rows are deleted. I surmise that the reason for this is that since the gradient entropy step takes in information from redundant translational motion estimates, deleting data will result in a harder time for the entropy algorithm to localize the translation data in image space. Without additional analysis and testing of the code, however, it seems we may not know the exact reason behind this increase in time. On the other hand, the median time for each coil's measurement of translation shows a decreasing trend. This is expected because as data is deleted, the algorithm has less to process.

Temporal resolution refers to the frequency in which the data can be sampled. In this case, the data refers to the 3D k space matrices. During our presentation, Dr. Cheng asked whether when deleting our k space data if we would have enough information to estimate linear translation. This is a good point because even though it may be quicker for the MRI scanner to acquire the data, the overall TR time and scan time may not be decreased because these scans are respiratory gated, meaning that alternating TRs start with each respiratory cycle rather than being one continuous scan. Therefore, we can make the argument that deleting these k space rows and trajectory may not have an impact on the overall scan time itself. Rather, it is the post-processing times that are most impacted.

Reference

1. Lustig et al. Butterfly: A Self Navigating Cartesian Trajectory. Proc. Intl. Soc. Mag. Reson. Med. 15 (2007)

Ciara's Analysis

Effect of Field Inhomogeneities on Motion Estimate

To explore the effect of field inhomogeneities on computed motion estimates, I simulated growing inhomogeneity over time due to field drift, a constant localized inhomogeneity such as the effect of a metallic implant in a patient, and an increasing localized field inhomogeneity. Since the motion estimate is sensitive to changes in the navigator data, these field inhomogeneities may affect the accuracy of the method, and this exploration will help determine the robustness of the motion estimate algorithm in scans with inhomogeneities.

Field Drift

Typical B_o fields experience drift on the order of 0.1ppm per hour. In the short time period of a single scan acquisition, this drift can be approximated to be zero. However, accumulated field drift over the course of a full scan, especially when total acquisition time is increased by using gated scans and the modified butterfly trajectory, may not be negligible in the motion estimate algorithm. This is similar to the effect of patient motion, where in each acquisition the patient is assumed stationary, but movement is possible between a series of acquisitions.

To simulate linear field drift, I use the following equations to modify previously collected data:

$$W_n = e^{2\pi i n v 5e-7}, \qquad K'_n = W_n K_n$$

Where n is the acquisition number, K_n is the raw k-space data from the original code, v is a scaling factor to adjust the rate of field drift, and K'_n is the field drift adjusted k-space data. This method assumes that in a single acquisition, the field drift is constant, and that between each TR frame there is a linear increase in field strength from the previous acquisition. To encompass a wide range of field drift speeds, values for v were chosen as 1, 1000, 1e8, and 1e17.

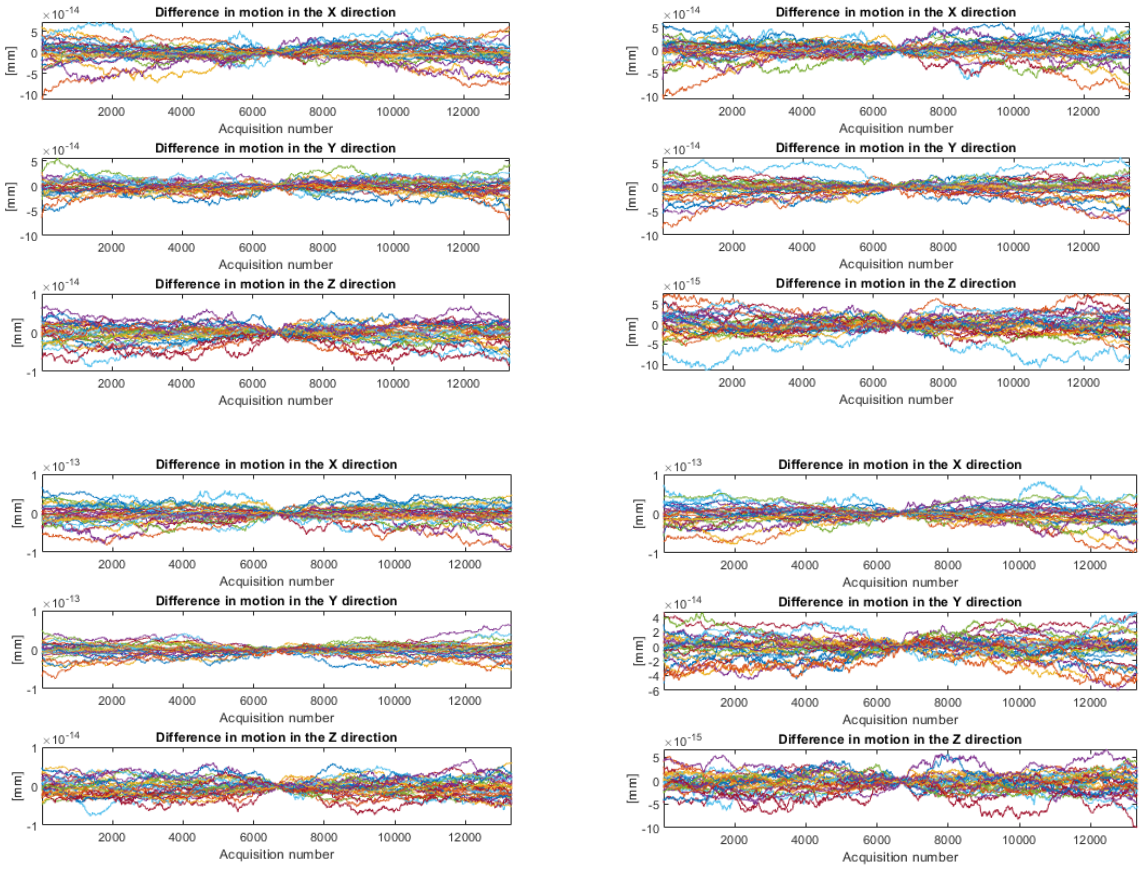


Figure A: Top left v=1, top right v=1000, bottom left v = 1e8, bottom right v=1e17. Each motion estimate from a different coil is shown in a different color. Differences in the motion estimate are negligible, though each scaling factor has different behaviors.

Though each scaling factor behaves differently qualitatively, as seen in Figure A, the motion estimate is largely unaffected by linear field drift. The average difference of the magnitude of the motion estimates for each direction is summarized in Table A below. These amounts are all negligible, and the motion estimate is approximately unchanged.

v	X [mm]	Y [mm]	Z [mm]
1	1.3366e-14	8.5005e-15	1.6051e-15
1000	1.2559e-14	1.0827e-14	1.6865e-15
1e8	1.4309e-14	1.0069e-14	1.6275e-15
1e17	1.3936e-14	1.0210e-14	1.4504e-15

Table A: The average difference in magnitude between the original motion estimate and modified motion estimate across all acquisitions in the X, Y, and Z directions for each scaling factor, v.

Localized Field Inhomogeneity

Patients undergoing scans with metallic implants such as IUDs, hip implants, etc. will have artifacts in final scans due to local field inhomogeneities caused by the object. Should this field effect lie in the path of the butterfly navigators, there may be effects on the motion estimate.
To simulate this effect, I use the following equation to modify previously collected data.

$$W_m = e^{2\pi i v 5\mathrm{e}-7}, \qquad K'_n = W_m K_n$$

Where m is the index of a single element of the navigator data, K_n is the raw k-space data from the original code, v is a scaling factor to adjust the strength of field inhomogeneity, and K'_n is the inhomogeneous field adjusted k-space data. Only one element of the k-space data is modified. This equation assumes that the effect of field drift is negligible. To encompass a wide range of field inhomogeneities, values for v were chosen as 1, 1000, 1e8, and 1e17.
The motion estimates behave similarly to the case with linear field drift. Effects of the localized inhomogeneity are negligible. Values for the average difference in the original and modified motion estimate are listed in Table B below.

v	X [mm]	Y [mm]	Z [mm]
1	1.5029e-14	9.6746e-15	1.7342e-15
1000	1.5349e-14	1.1488e-14	1.5759e-15
1e8	1.4867e-14	1.1571e-14	1.6629e-15
1e17	1.5374e-14	1.0978e-14	1.6815e-15

Table B: The average difference in magnitude between the original motion estimate and modified motion estimate across all acquisitions in the X, Y, and Z directions for each scaling factor, v.

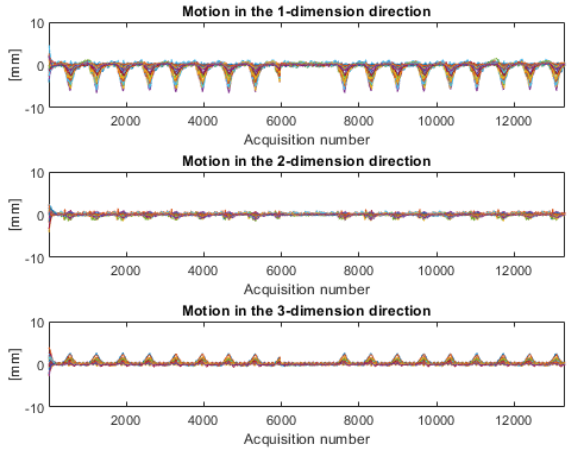
Increasing Localized Field Inhomogeneity

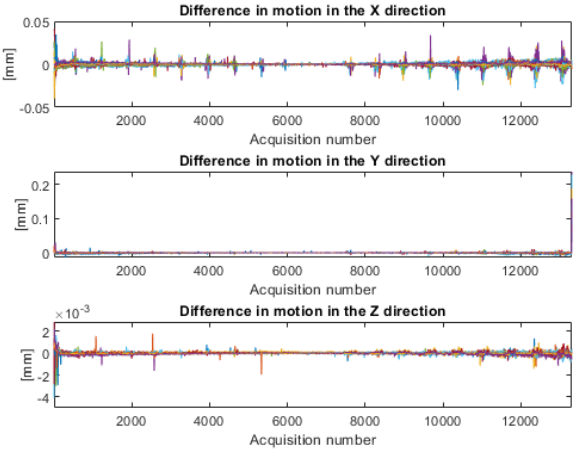
To simulate a growing localized field inhomogeneity, I use the following equation to modify previously collected data.

$$W_m = e^{2\pi i n v 5\mathrm{e}-7}, \qquad K'_n = W_m K_n$$

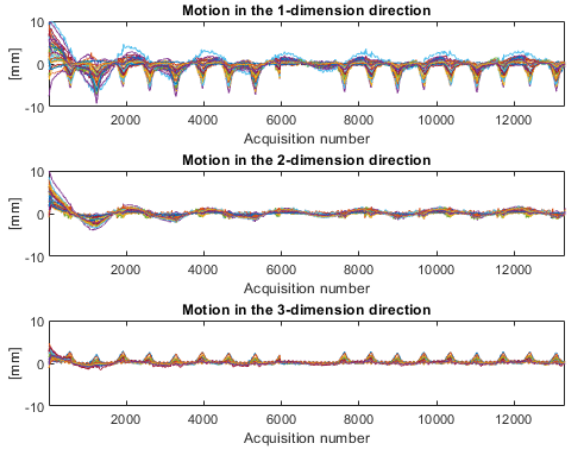
Where m is the index of a single element of the navigator data, n is the acquisition number, K_n is the raw k-space data from the original code, v is a scaling factor to adjust the strength of field inhomogeneity, and K'_n is the inhomogeneous field adjusted k-space data. Again, only one element of the k-space data is modified for each acquisition. This equation assumes that the local field inhomogeneity is growing linearly between each acquisition. To encompass a wide range of field inhomogeneity growth rates values for v were chosen as 1, 1000, 1e8, and 1e17.

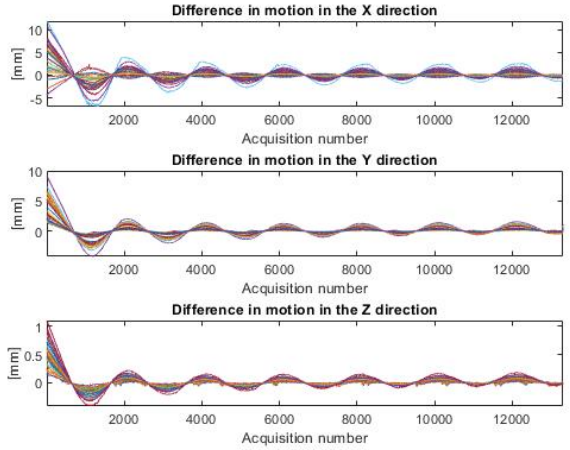
1


Motion in the 1-dimension direction
Motion in the 2-dimension direction
Motion in the 3-dimension direction
[mm]
Acquisition number

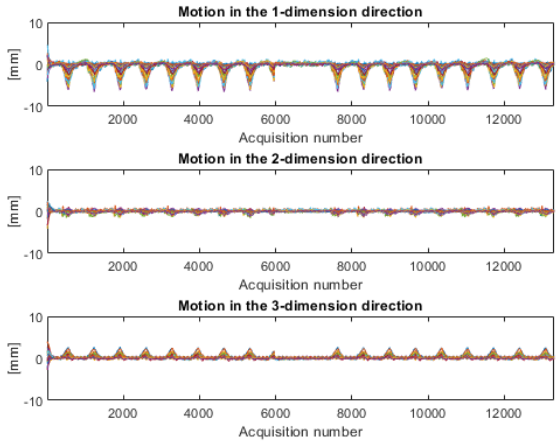

Difference in motion in the X direction
Difference in motion in the Y direction
Difference in motion in the Z direction
[mm]
Acquisition number

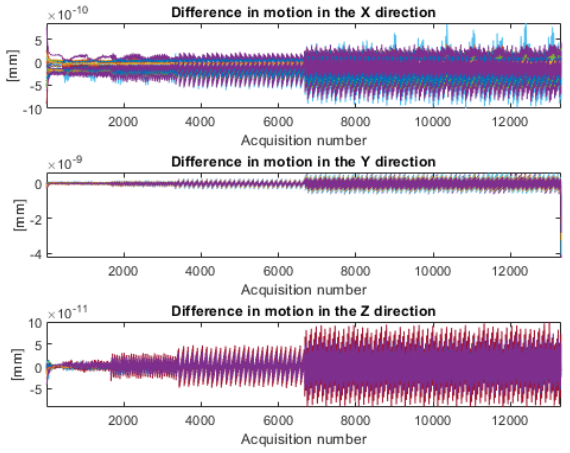
1e3


Motion in the 1-dimension direction
Motion in the 2-dimension direction
Motion in the 3-dimension direction
[mm]
Acquisition number


Difference in motion in the X direction
Difference in motion in the Y direction
Difference in motion in the Z direction
[mm]
Acquisition number

1e8


Motion in the 1-dimension direction
Motion in the 2-dimension direction
Motion in the 3-dimension direction
[mm]
Acquisition number


Difference in motion in the X direction
Difference in motion in the Y direction
Difference in motion in the Z direction
[mm]
Acquisition number

1e17

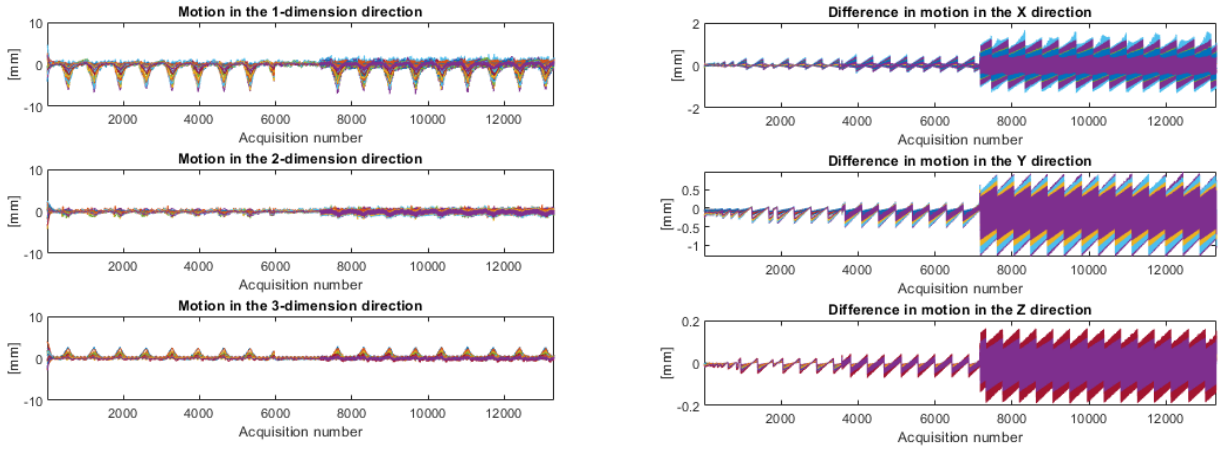


Figure C: Left: Total motion estimates, Right: Difference in motion estimates. The scaling factor is labeled at the left of each pair, and motion estimate for each coil is shown in a different color.

V	X [mm]	Y [mm]	Z [mm]
1	9.5693e-04	5.9160e-04	5.3510e-05
1000	0.5921	0.4929	0.0583
1e8	1.3958e-10	8.9889e-11	1.3694e-11
1e17	0.1743	0.1939	0.0273

Table C: The average difference in magnitude between the original motion estimate and modified motion estimate across all acquisitions in the X, Y, and Z directions for each scaling factor, v.

As seen in Table C, there is a significant effect on the motion estimates for scaling factors of 1000 and 1e17, but the effects for scaling factors 1 and 1e8 are negligible. This can be seen in Figure C where the shape of the motion estimates for v=1 and v=1e8 are nearly identical to the original, but the motion estimates for v=1000 and v=1e17 are affected. For v=1000, there is a sinusoidal effect super-positioned over the original motion estimates. For v=1e17, the difference is in high frequency noise in the motion estimates of the late scans. The differences in magnitude across different scaling factors, and the periodic behavior of the two highest magnitude scaling factors suggest effects of resonance may influence the behavior of the motion estimate in cases where a localized field inhomogeneity increases over time.

Field Inhomogeneities Discussion

The method of calculating motion in the scan is robust against linear field drift and localized field inhomogeneities, but susceptible to influence when an increasing localized field inhomogeneity is in the path of the butterfly navigators as simulated above. However, these simulations are limited and simple. Other field drift patterns may cause more significant effects, such as a periodically repeating drift pattern. Additionally, localized field inhomogeneities are much more complex and would affect a larger region of the navigator. Though this would not affect the resultant negligible motion estimate differences in the temporally static case, it may affect the magnitude, resonance, or shape of the effects in the increasing case.

References

Tal, A., & Gonen, O. (2013). Localization errors in MR spectroscopic imaging due to the drift of the main magnetic field and their correction. *Magnetic resonance in medicine*, *70*(4), 895–904. https://doi.org/10.1002/mrm.24536

Sujoy's Analysis

Investigation on the use of Projection Onto Convex Sets (POCS) for image reconstruction

The paper has implemented the use of homodyne reconstruction to reconstruct the image from partial k space. It allows the reduction of computation time for fully symmetric data. MRI images have a real (symmetric) and imaginary (antisymmetric) component. The key idea in the homodyne algorithm is to preweight the k-space data so that when we take the real part of the image data, it corresponds to a uniform weighting in k-space. The symmetric component can be reconstructed using conjugate symmetry property:

$$f(-x) = f*(x) \quad (1)$$

The weighting function can be also divided into ramp and step functions that become relevant for reducing ghosting artifact produced by materials like layers of fat. In this paper, ramp type conjugate symmetry preweighting function has been used. The results of homodyne algorithm have been shown in Fig. 1. The resultant image clearly showed organ boundaries for the 26^{th} z layer indicating the effectiveness of the technique. The darker regions on the difference image indicate a higher negative phase difference between the original and homodyne image. These regions indicate the amount of phase correction that has been implemented on the image as the symmetric components cancel out. One of the major issues faced by homodyne reconstruction is problems associated with the interaction between phase correction and conjugate synthesis. This can be resolved by iterative techniques like POCS which basically constrain the low-resolution image space data and in the k space the data is matched to the estimate whenever it is available. The final estimate after multiple iterations satisfies both the criterion and leads to much sharper and more detailed image. In the spatial frequency domain, the phase encodes that were acquired are replace the present phase estimate and inverse Fourier transform of the new image is then adjusted to match that of the symmetric component. The image converges in very few iterations after which the noise floor is reached. In the present example $5/8^{th}$ of k space was used for the symmetric data and different iterations of 5, 10, 15 and 100 were tested on the autocorrected data set. As shown in Fig. 2, the blurring visible in the original image is observed to decrease with every iteration. The maximum possible sharpness in edges was observed for 15 iterations, beyond which no noticeable change in image quality was recognized. This indicates that the noise floor for POCS technique is reached by 15 iterations. The computational time achieved was nearly 15 seconds. On a similar note, homodyne implementation was achieved in 6 seconds. A test performed at 100 iterations did not indicate any major change in image properties and hence the limit for phase correction was reached. The results indicate that phase correction in POCS techniques is quite comparable to that observed in Homodyne.

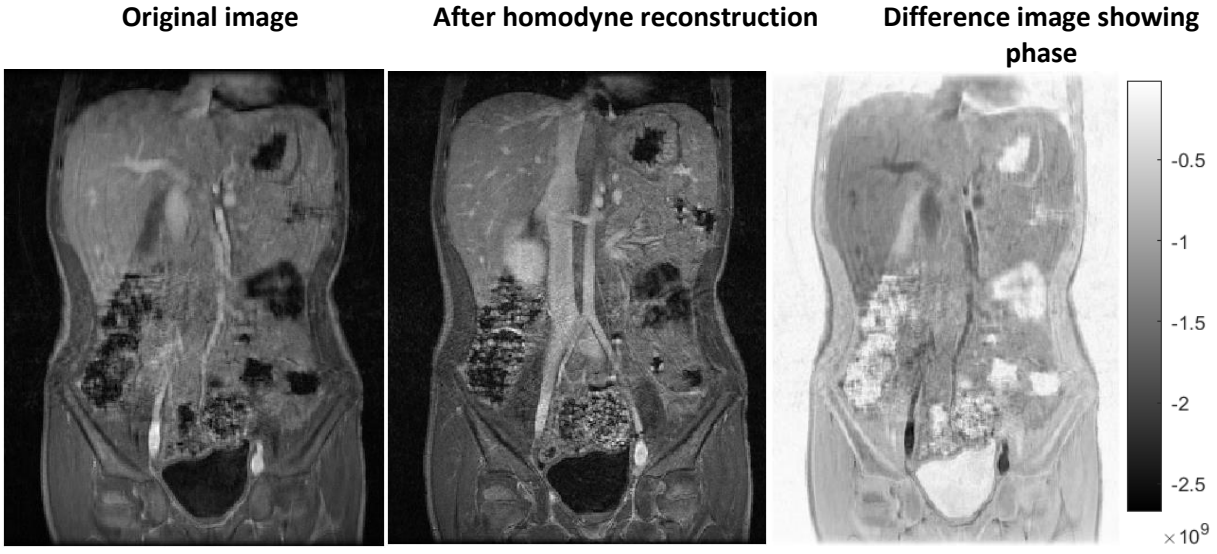


Figure 1: Image reconstruction by homodyne technique and corresponding phase change.

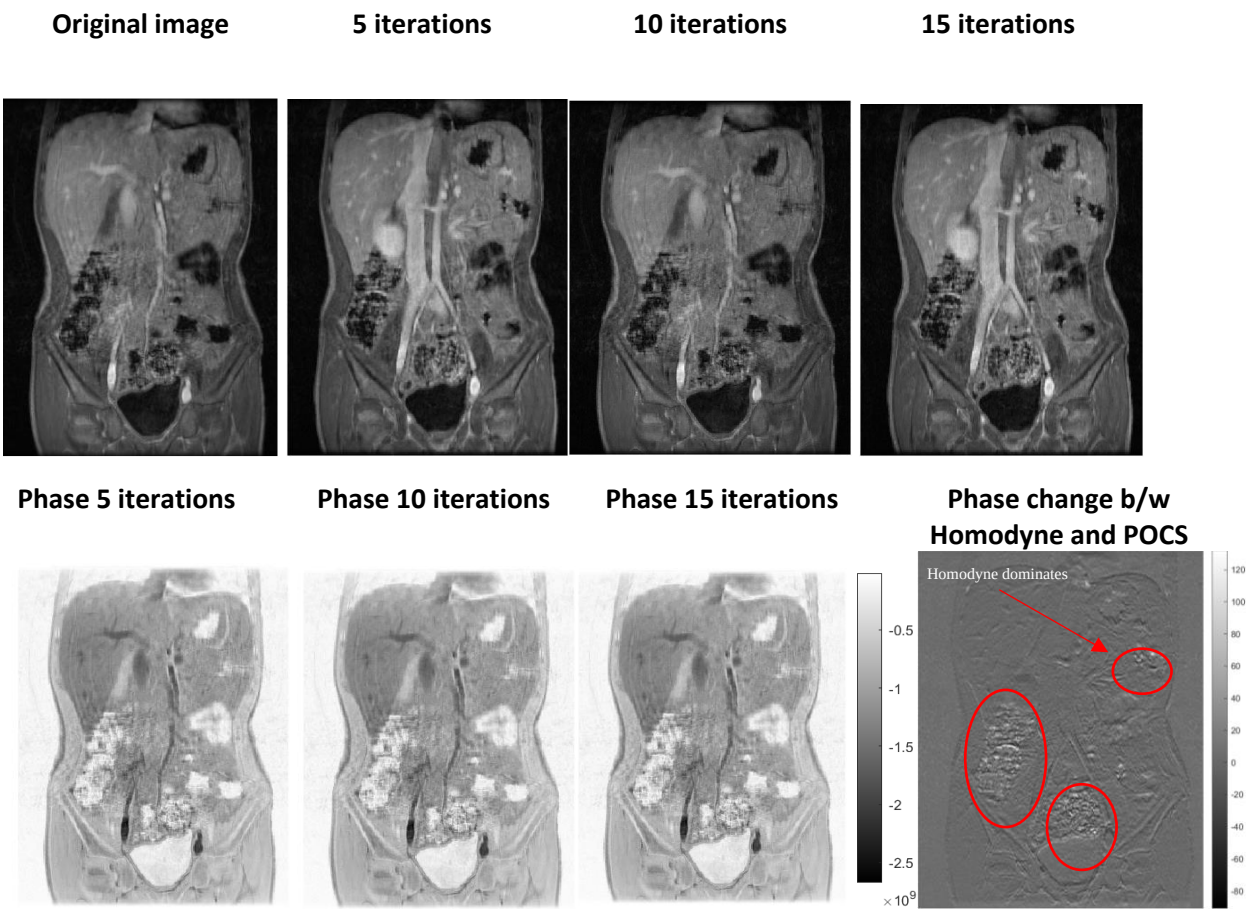


Figure 2: Image reconstruction by POCS technique and corresponding phase change.

Comparing the phase change between homodyne and POCS in Fig. 2 shows the white regions where homodyne has performed more phase correction than POCS and the black regions are areas where POCS correction dominates. This variation in phase correction can be explained by the feature of homodyne algorithm to utilize high frequency components in preweighting function, which are more abundant in the center of the k space. The symmetric data obtained for partial image reconstruction lies at this center, which makes the features of the final image sharper than POCS technique. Also, the motion artifacts in the image possibly contribute to wrongful estimation of vessel boundaries with actual motion parameters. Apart from minor phase inconsistencies, both the techniques provide similar phase correction and hence, can be used interchangeably. It can be concluded from the experiments performed in the data set that Homodyne technique used in this paper has been more effective in reducing ghosting artifacts and obtaining greater image sharpness than POCS technique. The computational time advantage is also another consideration that is relevant for cases involving autocorrection for images where dynamic motion is observed. The POCS technique can further be improved in the future by increasing the number of coils and consequent increase of pixels generates that can provide a large sample space for greater signal to noise ratio.

References

[1] Partial k space reconstruction, J. Pauly, Stanford University notes, 2005.

[2] Cheng, J. Y.; Alley, M. T.; Cunningham, C. H.; Vasanawala, S. S.; Pauly, J. M.; Lustig, M. Nonrigid Motion Correction in 3D Using Autofocusing With localized Linear Translations. Magn Reson. Med. 2012.

Supplementary Document

Homodyne implementation

```
%% runMotionAutofocus.m script
%%  This is an example of perfoming autofocusing motion
%%  correction using the localized gradient entropy criterion.
%%
%%  Run this script after executing runBflyMotionEstimate.m.
%%  This example uses motionAutofocus.m which is a completely
%%  matlab-based implementation of the algorithm. A compilable
%%  mex version (with omp/acml/cuda) is available that runs
%%  significantly faster.
%%
%% (c) Joseph Y Cheng (jycheng@mrsrl.stanford.edu) 2012

%% SVN info:
%%   Date:     $Date: 2012-02-17 17:04:40 -0800 (Fri, 17 Feb 2012) $
%%   Revision: $Revision: 1084 $
%%   Author:   $Author: jycheng $
%%   Id:       $Id: runMotionAutofocus.m 1084 2012-02-18 01:04:40Z jycheng $


%% example1.mat was acquired with partial k-space in the readout
%% direction. Fortunately, the homodyne reconstruction can be
%% performed post-autofocusing because the autofocusing algorithm
%% depends heavily on the high-frequency contents.

DEBUG_MOTION = 220;
KERNRMM      = 40;

KERNR = round(KERNRMM./res);

% Zero-fill partial k-space
%DATAAh = [zeros(nFRead-nx,ny,nz,nc); DATAA];

% Perform autofocusing!
dbdisp('starting autofocusing...');
[IMC,MOUT] = motionAutofocus3(DATAAc,DDX,DDY,DDZ,...
                              yorder,zorder,KERNR,...
                              DEBUG_MOTION);

% Perform homodyne recon
dbdisp('starting homodyne recon...');
tic;
IMH  = DATAAc;
IMCH = IMC;
for C=1:size(IMC,4)
    tempc = fftnc(IMC(:,:,:,C));
    IMCH(:,:,:,C) = homodyne3D(tempc(end-nx+1:end,:,:),nFRead,1);
    tempo = (IMH(:,:,:,C));
    IMH(:,:,:,C) = homodyne3D(tempo(end-nx+1:end,:,:),nFRead,1);
end
clear tempc tempo;
```

```
toc

%figure(DEBUG_MOTION+10),imshow3s(flipdim([sumofsq(IMH) sumofsq(IMCH)],1));
figure(m),imshow3s(sumofsq(fftnc(DATAAc)));
figure(m+1),imshow3s(flipdim([sumofsq(IMCH)],1));
figure(m+2),imshow3s(flipdim([sumofsq(IMCH)],1)-sumofsq(fftnc(DATAAc)));
colorbar;
```

POCS implementation

```
%% example1.mat was acquired with partial k-space in the readout
%% direction. Fortunately, the homodyne reconstruction can be
%% performed post-autofocusing because the autofocusing algorithm
%% depends heavily on the high-frequency contents.

DEBUG_MOTION = 220;
KERNRMM      = 40;

KERNR = round(KERNRMM./res);

% Zero-fill partial k-space
%DATAAh = [zeros(nFRead-nx,ny,nz,nc); DATAA];

% % Perform autofocusing!
dbdisp('starting autofocusing...');
[IMC,MOUT] = motionAutofocus3(DATAAc,DDX,DDY,DDZ,...
                              yorder,zorder,KERNR,...
                              DEBUG_MOTION);

%Perform POCS recon
dbdisp('starting POCS recon...');
tic;
IMH  = DATAAc; %DATAAC is K space
IMCH = IMC;
%st=size(IMC,4);
for C=1:size(IMC,4)
     tempc = fftnc(IMC(:,:,:,C));
     m=0;

%      IMCH(:,:,:,C) = pocs3d(tempc(end-nx+1:end,:,:),nFRead,1);
    tempo = (IMH(:,:,:,C));
%      IMH(:,:,:,C) = pocs3d(tempo(end-nx+1:end,:,:),nFRead,1);


   % for sli=1:size(tempc,3)

        hnover=0.625*nx;

        data_pk=tempc(:,:,:);
        %noise limit
        threshold_pocs=1;
        %zero padding first guess
        im_init=fftshift(ifftn(fftshift(data_pk)));
```

```
        %using phase term on magnitude

        data_pk(1+nx-hnover:end,:,:)=0;
        %center the symmetric data
        data_center=data_pk;
        data_center(1:hnover,:,:)=0;
        im_ph=fftshift(ifftn(fftshift(data_center)));

        im_init=abs(im_init).*exp(1i*angle(im_ph));
        tmp_k=fftshift(fftn(fftshift(im_init)));
        diff_im=threshold_pocs+1;
        %iterate till er ror difference greater than thresh old
        while(m<15)
            m=m+1;
%         while (abs(diff_im)>threshold_pocs)
            tmp_k(1:nx-hnover,:,:)=data_pk(1:nx-hnover,:,:);
            tmp_im=fftshift(ifftn(fftshift(tmp_k)));
            %applying phase term to magnitude
            tmp_im=abs(tmp_im).*exp(1i*angle(im_ph));
            tmp_k=fftshift(fftn(fftshift(tmp_im)));
            %comparing the i m a ges
            %diff_im=abs(tmp_im-im_init);
            %e=immse(tmp_im,im_init);
            %diff_im=sum(diff_im(:).^2);
            %fprintf('Differen %f\n',diff_im);
            %fprintf('\n The mean-squared error is %0.10f\n', e);
            fprintf('\n The coil %0.10f\n', C);
            fprintf('\n The z %0.10f\n', m);
            %fprintf('\n The si %0.10f\n', st);
            im_init=tmp_im;
        IMCH(:,:,:,C) =im_init ;
        end

    %end

       % for sli=1:size(tempo,3)

%         hnover=0.625*nx;
%
%         data_pk=tempo(:,:,:);
%         %noise limit
%         threshold_pocs=1;
%         %zero padding first guess
%         im_init=fftshift(ifftn(fftshift(data_pk)));
%         %using phase term on magnitude
%
%         data_pk(1+nx-hnover:end,:,:)=0;
%         %center the symmetric data
%         data_center=data_pk;
%         data_center(1:hnover,:,:)=0;
%         im_ph=fftshift(ifftn(fftshift(data_center)));
%
%         im_init=abs(im_init).*exp(1i*angle(im_ph));
%         tmp_k=fftshift(fftn(fftshift(im_init)));
%         diff_im=threshold_pocs+1;
%         %iterate till er ror difference greater than thresh old
```

```
%          m=0;
%          while(m<5)
%              m=m+1;
%          %while (abs(diff_im)>threshold_pocs)
%              tmp_k(1:nx-hnover,:,:)=data_pk(1:nx-hnover,:,:);
%              tmp_im=fftshift(ifftn(fftshift(tmp_k)));
%              %applying phase term to magnitude
%              tmp_im=abs(tmp_im).*exp(1i*angle(im_ph));
%              tmp_k=fftshift(fftn(fftshift(tmp_im)));
%              %comparing the i m a ges
%              %diff_im=abs(tmp_im-im_init);
%              %e=immse(tmp_im,im_init);
%              %diff_im=sum(diff_im(:).^2);
%              %fprintf('Differen %f\n',diff_im);
%              %fprintf('\n The mean-squared error is %0.10f\n', e);
%              fprintf('\n The coil %0.10f\n', C);
%              im_init=tmp_im;
%          IMH(:,:,:,C) =im_init ;
%          end

   % end

clear tempc tempo;
toc
end
figure(m),imshow3s(sumofsq(fftnc(DATAAc)));
figure(m+1),imshow3s(flipdim([sumofsq(IMCH)],1));
figure(m+2),imshow3s(flipdim([sumofsq(IMCH)],1)-sumofsq(fftnc(DATAAc)));
colorbar;
poc=flipdim([sumofsq(IMCH)],1)-sumofsq(fftnc(DATAAc));
```

Images at 100 iterations

Original image ***After homodyne reconstruction*** ***Difference image showing phase***