Published in final edited form as:
Neuroinformatics. 2013 July ; 11(3): 291–300. doi:10.1007/s12021-013-9176-3.

A Graphics Processing Unit Accelerated Motion Correction Algorithm and Modular System for Real-time fMRI

Dustin Scheinost[a], **Michelle Hampson**[b], **Maolin Qiu**[b], **Jitendra Bhawnani**[b], **R. Todd Constable**[a,b,c], and **Xenophon Papademetris**[a,b]

[a]Department of Biomedical Engineering, Yale University, New Haven, CT 06510, USA

[b]Department of Diagnostic Radiology, Yale University, New Haven, CT 06510, USA

[c]Department of Neurosurgery, Yale University, New Haven, CT 06510, USA

Abstract

Real-time functional magnetic resonance imaging (rt-fMRI) has recently gained interest as a possible means to facilitate the learning of certain behaviors. However, rt-fMRI is limited by processing speed and available software, and continued development is needed for rt-fMRI to progress further and become feasible for clinical use. In this work, we present an open-source rt-fMRI system for biofeedback powered by a novel Graphics Processing Unit (GPU) accelerated motion correction strategy as part of the BioImage Suite project (www.bioimagesuite.org). Our system contributes to the development of rt-fMRI by presenting a motion correction algorithm that provides an estimate of motion with essentially no processing delay as well as a modular rt-fMRI system design. Using empirical data from rt-fMRI scans, we assessed the quality of motion correction in this new system. The present algorithm performed comparably to standard (non real-time) offline methods and outperformed other real-time methods based on zero order interpolation of motion parameters. The modular approach to the rt-fMRI system allows the system to be flexible to the experiment and feedback design, a valuable feature for many applications. We illustrate the flexibility of the system by describing several of our ongoing studies. Our hope is that continuing development of open-source rt-fMRI algorithms and software will make this new technology more accessible and adaptable, and will thereby accelerate its application in the clinical and cognitive neurosciences.

Keywords

real-time fMRI; motion correction; graphics processing unit; open-source software

1) INTRODUCTION

Real-time functional magnetic resonance imaging (rt-fMRI) is an emerging technology that gives individual subjects moment-to-moment visual feedback of regional brain activity or patterns (Christopher deCharms, 2008; deCharms, 2007; Weiskopf et al., 2004; Weiskopf et

Corresponding author: Dustin Scheinost Magnetic Resonance Research Center 300 Cedar St PO Box 208043 New Haven, CT 06520-8043 Tel: (203) 785-6148 Fax: (203) 785-6534 dustin.scheinost@yale.edu.

Information Sharing Statement

The software described in this paper can be freely downloaded at http://www.bioimagesuite.org. The relevant source code for the motion correction algorithm can be found in Registration/rtmotioncorrection.cu, Registration/vtkbisRTMotionCorrection.cpp, and bioimagesuite/main/bis_realtimefmri.tcl. The minimum CUDA version need to compile that motion correction algorithm is CUDA 2 and the minimum GPU is a GTX 8800 or new card with at least 256mb of ram. We would greatly appreciate any suggestions, comments or feedback to improve the present rt-fMRI system.

al., 2007). As first demonstrated by Yoo et al. (Yoo and Jolesz, 2002), this approach allows individuals to develop an enhanced ability to exert control over activity in specific regions of the brain that have previously been implicated in neural processes or disorders. Learning to control the response of the brain may go a long way toward learning to effectively mediate behavior. Rt-fMRI has been shown to be successful in providing non-invasive real-time biofeedback for a wide range of brain processes, including motor processing (deCharms et al., 2004; Posse et al., 2001), auditory perception (Yoo et al., 2006), pain control (deCharms et al., 2005), emotion modulation (Phan et al., 2004), and linguistic processing (Rota et al., 2009). Additionally, changes in functional and effective connectivity have been demonstrated in subjects during rt-fMRI training sessions (Lee et al., 2011; Rota et al., 2011), and changes in resting-state functional connectivity have been demonstrated after subjects received rt-fMRI biofeedback training (Hampson et al., 2011a).

Recently, rt-fMRI has received increased interest with several published studies in the past two years (Caria et al., 2011; Caria et al., 2010; Cusack et al., 2011; Goebel et al., 2011; Hamilton et al., 2011; Hampson et al., 2011a; Hampson et al., 2011b; Hinds et al., 2011; LaConte, 2011; Lee et al., 2011; McCaig et al., 2011; Rota et al., 2011; Sander and Kandrot, 2010; Shibata et al., 2011; Sitaram et al., 2010; Zotev et al., 2011). However, compared to standard fMRI experiments, rt-fMRI is still limited by processing speed and available software. Continued software and algorithmic development is needed to allow further progress in rt-fMRI and to broaden its use. In this work, we present an open-source rt-fMRI system for biofeedback powered by a novel Graphics Processing Unit (GPU) accelerated motion correction strategy as part of the BioImage Suite project (www.bioimagesuite.org).

In rt-fMRI experiments, tradeoffs exist between less robust results associated with simpler methods and processing delays associated with standard offline methods. Rt-fMRI data must be processed as the images are acquired which is generally at a rate of a volume every 1 or 2 seconds. In standard fMRI experiments, data is processed offline allowing sophisticated processing techniques that, for large datasets, may take hours to days. Thus, some processing tradeoffs must be made, which leads to a divergence between online and offline methods. Due to high computational cost, motion correction tends to be the first area where rt-fMRI methods deviate from standard offline methods. For example, rt-fMRI studies have forgone motion correction (Eklund et al., 2009; Yoo et al., 2004; Yoo and Jolesz, 2002; Yoo et al., 2006), used vendor-specific algorithms (Siemens MoCo) (Hinds et al., 2011), or used different algorithms for online and offline processing (Caria et al., 2010; Hinds et al., 2011; McCaig et al., 2011). Alternatively, methods comparable to standard offline methods can have a processing delay on the order of an image acquisition (Cox and Jesmanowicz, 1999; Mathiak and Posse, 2001).

Freely available and open-source software designed for rt-fMRI is limited. Turbo BRAIN Voyager (www.brainvoyager.com) is widely used but is commercial software. Turbofire (Gembris et al., 2000; Posse et al., 2001) is non-commercial but is not open source. AFNI (Cox et al., 1995) provides rt-fMRI functionality but through building blocks that must be programmed together rather than an all-in-one package. Offline packages like FSL and SPM can be modified for rt-fMRI (Cusack et al., 2011; McCaig et al., 2011), but these methods are not optimized for the computational speed and incremental data challenges of rt-fMRI. Many have instead used custom in-house software, which may not be publicly available or readily tested (Hamilton et al., 2011; McCaig et al., 2011; Yoo et al., 2004). Furthermore, software may be designed specifically for the experiment at hand and couple the image processing with the feedback display. This integrated design can make it difficult to change the feedback for different experiments and may require large amounts of software to be rewritten for each new experiment.

This work contributes to the development of rt-fMRI in two ways: first, by introducing a motion correction approach that provides an estimate of motion with essentially no processing delay and, second, by introducing a modular rt-fMRI system design, inspired from our previous work with image-guided surgery systems (Papademetris et al., 2006; Tokuda et al., 2009). While these two contributions are presented in unison, they are independent from each other. The modular system can be used without the presented motion correction and the motion correction can be used to analyze offline data. For motion correction, the goal was to create a real-time motion correction implementation that, within the time and incremental-data constraints inherent to rt-fMRI, achieves similar performance to standard offline motion correction algorithms. By taking advantage of the interleaved acquisition of fMRI data and GPU acceleration, the algorithm outperforms other methods with little or no processing delay such as zero-order interpolation of motion parameters estimated from the previous frame. Our modular system consists of: (1) a back-end processing component that interfaces with the acquisition/reconstruction setup, and performs motion correction and ROI analysis, and (2) a front-end feedback component that displays feedback to the subject. The division into back-end and front-end components allows the creation of a stable (back-end) system that performs the core processing while remaining unchanged between experiments and a more flexible (front-end) system that simply handles stimulus generation/feedback. Our rt-fMRI system has been shown to be successful in several published and ongoing studies. In this paper, however, we present for the first time: 1) our motion correction algorithm and GPU implementation, 2) a detailed overview of our modular rt-fMRI and implementation, and 3) an evaluation of the motion correction strategy.

2) Overall Approach

2.1) Motion Correction Algorithm

Our strategy takes direct advantage of the interleaved acquisition of fMRI (where the odd-numbered slices are acquired first followed by the even-numbered slices). Our system first waits until all odd slices of frame n are acquired. It then properly orders these odd slices with the even slices of frame n-1 creating frame n-$½$. Frame n-$½$ is then rigidly registered to a reference frame supplied at the start of the experiment. The registration is then run concurrently with the acquisition of the even numbered slices of frame n allowing for little or no processing delay (a short amount of time is needed to apply the resulting transformation once all the data is collected). Once all the even slices are acquired, frame n is assembled and the output of the motion correction is applied.

Similar to most fMRI motion correction algorithms, the motion correction algorithm minimizes a sum of square differences (SSD) objective function, $\sum_x R[x] - T\{I[x]\}$, where x is the voxel index, $R[x]$ is the intensity of reference frame R at x, $I[x]$ is the intensity of frame n-$½$ at x, and $T\{\}$ is a six parameter rigid body transformation matrix. To avoid the influence of background and non-brain voxels on the optimization, only voxel pairs with intensity greater than 5% of the mean intensity of the brain in the reference frame are included in the optimization. A hill climb optimization is used to minimize the SSD objective function (Studholme et al., 1996). To help reduce the effects of local minima, a three stage multi-resolution technique is used. Both images are initially resampled to have isotropic voxel dimensions equal to the minimum voxel dimension. For the first two multi-resolution stages, the isotropic images are smoothed with a Gaussian kernel with a FWHM of 4mm (1st stage) or 2mm (2nd stage) and downsampled by a rate of 4 (1st stage) or 2 (2nd stage). For the 3rd stage, the images are left at the original isotropic dimension and are not smoothed. For each stage, the hill climb algorithm uses 64 steps with a step size of 0.1 of the current voxel dimension. Each stage is initialized with the results of the previous stage.

Trilinear interpolation is used when applying the transformation. The motion correction algorithm is implemented on the GPU with NVIDIA`s CUDA programming language (www.nvidia.com/cuda).

2.2 GPU Implementation

Since the most time consuming steps in motion correction are applying the transformation and evaluating the objective function, our GPU implementation primarily focuses on parallelizing these two steps with two kernel functions (GPU functions that are called in parallel). The first kernel function treats each corresponding pair of reference and frame voxels independently and processes the pair by applying the transformation matrix, subtracting the transformed frame n – 1 from the reference image, and squaring the difference. Linear interpolation is performed by fetches to texture memory. The final result of the first kernel function is a squared difference image between the reference image and the transformed frame *n*-½. The second kernel function is designed to sum the squared difference image using a parallel reduction algorithm presented. The output from these kernels functions is the evaluation of SSD metric.

The dimensions (approximately 64×64×32) of fMRI data provide a convenient memory arrangement for the GPU. Once copied to the GPU, frame *n*-½ is arranged in texture memory as a 3D texture with the width, height, and depth of the texture set to the x, y, and z dimensions of the image, respectively. However, since the reference image does not need to be interpolated, it is left simply as an array in global memory. The first kernel function is called using a grid with dimensions equal to the x and y dimensions of the image and blocks with size equal to the z dimension of the image. This memory arrangement was chosen for convenient access to the x, y, z coordinates of a voxel, which are needed for the built-in linear interpolation.

The hill climb optimization is implemented in CUDA on the CPU. As hill climb optimization is generally a serial operation, this design was chosen due to a low expected reduction in processing time for a GPU implementation. As parameters are changed during each step of the optimization, the corresponding transformation matrix is updated and copied from the host memory to the device memory. The two kernel functions described above are called, the SSD metric is copied back to host memory, and the new parameters are kept if the SSD metric is lower. Each step in the optimization involves 12 float copies to the device memory for the transformation matrix (the last row of the matrix is exclude as it remains constant for a 3D transformation matrix) and 1 float copy from device memory (the SSD metric). The CPU optimization is simpler than a corresponding GPU implementation and the added memory transfers between host and device do not significantly decrease performance of our algorithm.

Currently, both the smoothing and resampling operations associated with the multi-resolution optimization are performed on the CPU. The resulting images are then transferred to the GPU to facilitate estimating the motion parameters. Once the optimization is finished for the current stage, the images are smoothed and resampled for the next stage on the CPU and the resulting images are again transferred to the GPU. In total, for each frame-to-frame motion correction, the transform image is transferred to the device three times (one for each resolution level). The smoothed and resampled reference images can be transferred to the device before the rt-fMRI experiment. While both smoothing and resampling can be implemented on the GPU, this CPU implementation accounts for less than 1% of the total computation cost for motion correction of a single frame.

2.3) Modular rt-fMRI System

Our rt-fMRI system consists of two computers, MRI scanner, and reconstruction system as shown in Figure 1. As the scanner acquires a frame of the fMRI time series, each slice in the frame is reconstructed on the fly by the reconstruction system and saved as a file to a directory accessible by the image processing computer. The back-end component, running on the image processing computer, then scans the mounted network directory for each slice (volume) file and analyzes the current frame. The front-end component, running on the feedback computer, receives the results from the back-end component and displays them back to the subject.

The back-end and front-end components were chosen to run on different computers to achieve system stability and flexibility. Typically, the MRI scanner and reconstruction system sit behind a firewall on their own local area network (LAN). With a two-component system, the back-end component can sit behind the firewall, taking advantage of the speed of the LAN, and remain untouched from experiment to experiment. The front-end, which may change between experiments, sits outside the firewall making it flexible to the needs of the experiment. Results are transferred between the two components using a RS-232 serial port.

2.4) Back-end Processing Component

The back-end component is the crux of our system, providing a generic interface between the MRI scanner/reconstruction system and the display component. This component performs four major tasks: scan for new slices and construct a 3D image, perform motion correction using the proposed algorithm, analyze the data, and send the results to the front-end component.

Slice/Volume Retrieval—The back-end component repeatedly scans the remote directory for new data as the data is being reconstructed. If each slice of a volume can be saved in real-time, the Back-End will proceed as outlined in Figure 2. Depending on available data, the back-end component will look for the next slice/volume, perform motion correction, or perform ROI analysis and send the results to the front-end component.

Motion Correction—Motion correction is performed with the algorithm described in Section 2.1. If the scanner is not able to save each slice of an image in real-time, the end-users can specify one of the alternative implementations described in Section 3.1. We have included these alternative implementations to increase the flexibility of the system.

ROI Analysis—The motion corrected frame n is analyzed for feedback by calculating the mean activity in predefined ROIs. The system accepts multiple ROIs for experiments that may involve activation in multiple brain regions or require control regions. To correct for small motion artifacts near brain edges and partial volume effects, any voxels with intensity less than 50% of the average intensity of all the ROIs in the previous frame (frame n-1) are not included in the analysis of frame n and are removed from any future calculations.

Serial Connection—The final step in the workflow for the back-end component is to send the results to the front-end component. Here, the number of ROIs and each ROI mean is sent as a string over serial communication to the front-end. A start and stop code is added to the front and end of each string. The final string sent has the form: R_T_F *#_of_ROIs ROI_Means* R_T_F.

2.5) Front-end Component

While the back-end component is kept constant for all experiments, the front-end component could vary dramatically from application to application. The front-end component is responsible for generating stimuli that drive an fMRI experiment and it also incorporates the results from the back-end component to provide feedback to the subject. Often this feedback is provided by plotting the mean signal intensity of a specific ROI. Scanner drift can be controlled at this stage by displaying the mean signal in an ROI relative to the activity in a control region in a similar manner to that described by de Charms et al (deCharms et al., 2005). In addition, temporal smoothing of the feedback signal and outlier detection can be performed at this stage to increase the robustness of the feedback signal. The front-end component implementation is left up to the end user and allows the use of any computer/ programming language capable of serial port communications. This provides proper interfacing of the feedback with typical systems used to generate fMRI stimuli.

2.6) System Implementation

The back-end component is implemented as part of the BioImage Suite project, a comprehensive, multi-platform, open-source image analysis suite. End-users interface with the back-end component via a custom TCL command line script. The inputs to the back-end component include the subject's fMRI volume used as the reference image during motion correction with the same dimension as the images acquired in real time, an image in the same space as the reference volume defining the ROIs to be analyzed, the number of frames in the fMRI experiment, and the path to the directory where the reconstructed slices will be saved by the scanner. This script implements the scanning of the remote directory and the assembly of the fMRI volumes. It also interfaces with the C++ code, which implements the motion correction and ROI analysis via custom C++/TCL wrapper functions. ROI analysis is performed using a standard algorithm within BioImage Suite. The presented motion correction algorithm is accessible to the main script by CUDA/C++ wrapper functions. By wrapping the hardware acceleration in this manner, the algorithm gains access to all of BioImage Suite's processing pipelines and testing framework. Joshi et al (Joshi et al., 2011) provides more details on BioImage Suite's integration of CUDA, C++, and TCL and testing framework.

Since the front-end component can be designed by the end user, we show two examples of front ends used with our system to highlight the advantages of separating the display component from the image processing component. Additionally an example front-end witten in Matlab is provided with BioImage Suite. The two front-end components differ in programming language, ROI signal processing, and stimuli shown to the subject. A front-end component written in E-Prime and that performs additional computations to increase robustness of the feedback is shown in Figure 3a. This biofeedback display was designed for a study in which subjects were trained to control self-referential thinking associated with activation of the posterior cingulate cortex (PCC). Here, ROI means with greater than a 10% change from the previous volume were treated as outliers and replaced by the previous measurement. Additionally, the ROI means were temporally smoothed based on the last 5 values with a zero mean unit variance Gaussian kernel. The signal is shown to the subject as a histogram of percent signal change in the PCC relative to a baseline task and is updated every 2 seconds. A second example of different front-end components is written in Matlab and the feedback provided to the subject includes multiple stimuli shown simultaneously (Figure 3b). This biofeedback display was designed for a study in which subjects were trained to control a region of the orbitofrontal cortex involved in contamination anxiety while viewing anxiety-provoking stimuli (Hampson et al., 2011b). At the top of the display, a large image presents scenes that are either neutral or designed to provoke contamination anxiety. On the left of the display, a color-coded arrow indicates whether the subject should

rest, attempt to increase activity, or decrease activity. At the bottom of the display, a color-coded line graph provides real-time feedback. The images and arrows change every 26 seconds, and the feedback is updated for each volume acquired.

Our current implementation uses a Siemens 1.5T Sonata human MRI system. In interleaved mode, this scanner acquires first the odd-numbered slices, then the even-numbered slices. Other MRI systems can be used in place of the Siemens system. Essentially, the only requirement is that the reconstruction software reconstructs and stores the volumes in real-time in a directory accessible by our software. If only whole volumes can be saved in real-time , then the proposed motion correction would not work. To use the proposed motion correction, the scanner also needs to acquire and save the slices in real-time and in an interleaved fashion. If the proposed algorithm cannot be used, then either prospective motion correction such as PACE or the frame *n* approach described below (Section 3.1) can be used with the software. We are currently testing our software on a Siemens Tim Trio 3T, a Verio 3T, and a Magnetom 7T at two research centers.

3) RESULTS

3.1) Evaluation of Motion Correction

Competing Algorithms—The lack of ground truth data complicates the evaluation of the accuracy of motion correction algorithms on empirical data. Thus, we evaluated our algorithm by demonstrating that it provides results that are as consistent with the results of offline methods as these results are comparable with each other. Our goal is to approach online performance in a real-time specific system as opposed to improve on online methods. We test three different implementations:

1. The frame *n-½* approach where we use our prediction of the motion from the frame *n*.
2. The frame *n* scenario approach where we use all the data from frame *n*. This approach represents the theoretical upper bound in terms of motion estimation quality but it creates a delay in feedback caused by motion correction. Most rt-fMRI systems currently use this type of approach.
3. The frame *n-1* approach where we use the frame *n-1* data to estimate motion for frame *n*. This approach uses no data from frame *n*, but can also achieve no delays associated with motion correction (similar to our presented approach).

Data and "Gold Standard" Generation—The presented motion correction algorithm was compared to FSL MCFLIRT (http://www.fmrib.ox.ac.uk/fsl) and SPM Realign (www.fil.ion.ucl.ac.uk/spm) using rt-fMRI data consisting of 8 subjects attempting to increase activity in the Supplemental Motor Area (SMA) while receiving feedback of SMA activity over four neurofeedback session. Complete details about this sample can be found elsewhere (Hampson et al., 2011a). SPM and FSL were used with their default parameters for rt-fMRI data with the exceptions that tri-linear interpolation for SPM instead of the default spline-based interpolation. For each subject and each session, each run was motion corrected to a short functional series acquired immediately before neurofeedback instead of the default volume for each program.

Each motion correction was performed and statistical maps comparing the increased task activation relative to baseline were created for each session and motion correction. These maps were thresholded at $p<0.05$, creating binary images showing significant clusters. Overlap (the union of the two images divided by the intersection of the two images; also known as the Jaccard index) of the significant clusters was used to evaluate the consistency

of the algorithms. A higher overlap of significant clusters indicates that the two motion correction algorithms produced similar results and were more consistent with each other.

As we aim to show that the presented algorithm is as consistent with the results of offline methods as the two offline methods are with each other, we compared the overlap of the presented algorithm and the offline methods with the overlap between the two offline methods. For each pair of motion correction algorithms, overlap of significant clusters was computed for each session of feedback resulting in four measures of overlap for each pair of motion correction algorithms. Two-sample t-tests were used to compare the overlap of each implementation of the presented algorithm (see above). Significance was assessed at $p<0.05$. Our gold standard, the average overlap of significant clusters between SPM and FSL, was 80.2%.

Results—Our optimal GPU-accelerated real-time, no delay, frame *n-½* approach was shown to be effective as it produced similar results to the gold standard (offline) algorithms and best-case baseline (using frame n which adds delay to the feedback system) and exhibited statistically significant improvement over the approach using frame *n-1*. The mean overlap over all subjects and sessions are summarized in Table 1. We note that the average overlap between SPM and FSL was 80.2%. The overlap for frame *n* with FSL and SPM was 80.35% and 81.64%. The approach using our frame *n-½* approach performed similarly with average overlaps of 78.76% when compared to FSL and 78.95% when compared to SPM. The approach using frame *n-1* performed significantly worse ($p<0.05$) with average overlaps of 68.69% when compared to FSL and 69.07% when compared to SPM. Several representative slices containing activation in the SMA are shown in Figure 3. In this figure, all five motion correction algorithms are shown to reduce edge artifacts, and increase activation in the target region when compared to no motion correction. However, the frame *n-1* approach produces less significant results in the SMA compared to the other four algorithms.

On a Linux workstation with a 3.16 GHz Intel Xeon CPU and a GTX 260 Nvidia GPU, the average time for frame-to-frame motion correction using the proposed method was 559 msec where each frame was a 64×64×23 image. As this processing time is well within our goal of being less than the time needed to collect the even slices at a TR of 1.5 seconds or greater, we did not optimize the parameters used during the hill climb optimization. Instead, we used relatively conservative numbers (in terms of speed) to ensure proper accuracy of the motion correction. We note that the optimization started to converge after a few steps. A faster implementation with only a slight reduction in accuracy is possible by reducing the number of step in the optimization if a further reduction in time is needed for the experiment at hand. However, this was not needed to achieve our time goals.

3.2) Rt-fMRI Studies

Our rt-fMRI system has been shown to be successful in several published and ongoing studies. Hampson et al (Hampson et al., 2011a) used the software described here in a study in which subjects underwent four rt-fMRI sessions in order to learn to control activation in the (SMA). It was shown that, after some biofeedback, subjects displayed significant control of the SMA and that resting state functional connectivity between the SMA and subcortical regions decreased after rt-fMRI training. Furthermore, two ongoing studies are currently using the presented system. The first is a study to evaluate rt-fMRI as a training tool for meditation using the front end shown in Figure 3a. The second study uses the front end in Figure 3b to train subjects to control regions within the orbitalfrontal cortex involved in contamination-related anxiety, with potential long-term applications for obsessive-compulsive disorder (OCD) (Hampson et al., 2011b).

4) DISCUSSION

We have developed a novel motion correction strategy and an open-source implementation of a real-time fMRI system that is freely available in the BioImage Suite project. The motion correction strategy takes advantage of the interleaved acquisition of fMRI data and GPU acceleration to achieve minimum processing time. Using only the odd slices of the current frame, the frame-to-frame registration time (~0.5s) is less than that needed to collect the even slices (½ TR; 1s for our data). As a result, an estimate of the motion parameters is available at the end of an image acquisition with no time delay. Our real-time motion correction approach is shown to be comparable to standard offline motion correction algorithms. Furthermore, this strategy is more effective than using the previous frame to estimate motion for the current frame (which also has essentially zero processing delay).

In general, motion correction algorithms can be classified as either prospective or retrospective. Retrospective algorithms are the most common and, in the simplest case, involve correcting for motion of a given frame of an fMRI time-series by using that frame for motion correction. Examples of retrospective algorithms include SPM and FSL motion correction as well as the frame *n* implementation of our algorithm. Prospective algorithms involve correcting motion of a given frame based on previous estimates of motion. These algorithms such as 3D PACE are most useful when the motion estimates are then used to update the scanning parameters to account for the change in head position. Without this update of parameters, estimating motion for the current frame based on the motion of the previous frames is generally not practical as retrospective algorithm will perform at least as well due to the added information about the actual motion in the current frame. Rt-fMRI is one application where using prospective algorithms without updating scan parameters is practical as any reduction in processing time will allow more timely feedback. The proposed algorithm for the frame *n-1* implementation is a prospective algorithm and is similar to 3D PACE (Thesen et al., 2000) with the exception of not updating scanning parameters.

If slices of a volume are available in real-time, the proposed algorithm blurs the line between prospective and retrospective algorithms as it both estimates motion before all the data is collected and uses information from the current volume in the estimate of motion. A key insight of the frame *n-½* implementation is that the information in the odd slices can be used to increase the accuracy of the motion correction. While the fame *n-1* implementation is helpful in reducing motion artifacts, this approach is limited because prediction of the future can never be completely accurate as the subject may move in completely different direction between temporally adjacent frames. The additional information contained in the odd slices of the current frame allows the frame *n-½* implementation to be much closer to the frame *n* implementation and significantly better than the frame *n-1* implementation.

Our system is modular, consisting of 1) a back-end processing component responsible for motion correction and ROI analysis and 2) a front-end feedback component responsible for interpreting the results and displaying feedback to the subject. The benefit of a two-component system is that it allows: 1) the back-end component to become a permanent fixture within the MRI scanner and the reconstruction system firewall and 2) the front-end component to be flexible to the end-user choice of computer and feedback design. This approach has been successful for image-guided neurosurgery where one component (the stereotactic navigation system) must remain stable from procedure to procedure and the other must be flexible as new imaging algorithms are developed and integrated into the surgery (Papademetris et al., 2006; Tokuda et al., 2009). We are currently adapting the modular framework to real-time neurofeedback systems based on Near Infrared Spectroscopy (NIRS) and Dense Electroencephalography (EEG). The modular framework will allow the use of the same front-ends to be used for any feedback device while only

modifying the back-end to integrate with NIRS or EEG speaking to the flexibility of this approach.

The modularity of the proposed system is in contrast to highly integrated systems such as the one present in (LaConte, 2011). More integrated systems can provide faster processing and quicker feedback which are both important features of a rt-fMRI system. Nevertheless, these systems can be hard to implement at other research centers. While the presented motion correction algorithm requires an increased level of integration by having each slice reconstructed in real time, the presented frame work is flexible to presentation of stimuli both in terms of presentation software and type and to choices of motion correction. If speed and simple integration is critical, the frame *n-1* approach can be used. If accuracy and simple integration is critical, the frame *n* approach can be used. If speed and accuracy is critical but more complex integration is possible, the frame *n-½* approach can be used.

Rt-fMRI is a promising field of study limited by several technical challenges. The first rt-fMRI studies typically used computer clusters and high-end workstations to meet the high computational and minimum processing-time demands of a rt-fMRI experiment (Bagarinao et al., 2003; James T, 1999). However, GPUs now provide an ideal, inexpensive solution to meet the computational demands for rt-fMRI experiments as most processing algorithms, including motion correction and sliding-scale correlation, can be easily parallelized. Our hope is that continuing development of open-source rt-fMRI algorithms and software (such as those presented here) will improve accessibility to rt-fMRI, and eventually, lead to standard software packages similar to those used in non-real-time fMRI research (such as SPM, AFNI, or FSL). Future work will include the development of additional algorithms that allow for more complex analyses. Additions of recursive General Linear Models (Hinds et al., 2011; Nakai et al., 2006) to model drift and additional regressors, linear prediction to offset delays in the hemodynamics of fMRI, independent component analysis (ICA) based functional connectivity (Esposito et al., 2003), and whole-brain classification techniques (LaConte, 2011; LaConte et al., 2007; Sitaram et al., 2010) could enable new study designs and continued development of this exciting technology.

Acknowledgments

We thank J. Brewer and P. Worhnsky for the development of the front-end used for the current meditation study and for the example feedback shown in Figure 3a. We also thank E. Finn for her helpful comments on the manuscript. This study was funded by the Dana foundation (M. Hampson) and NIH (R01 EB006494, R03 EB012969, RO1 EB009666, R01 NS051622, R21 MH090384).

References

Bagarinao E, Matsuo K, Nakai T. Real-time functional MRI using a PC cluster. Concepts in Magnetic Resonance Part B: Magnetic Resonance Engineering. 2003; 19B:14–25.

Caria A, Sitaram R, Birbaumer N. Real-Time fMRI: A Tool for Local Brain Regulation. The Neuroscientist. 2011

Caria A, Sitaram R, Veit R, Begliomini C, Birbaumer N. Volitional Control of Anterior Insula Activity Modulates the Response to Aversive Stimuli. A Real-Time Functional Magnetic Resonance Imaging Study. Biological Psychiatry. 2010; 68:425–432. [PubMed: 20570245]

Christopher deCharms R. Applications of real-time fMRI. Nat Rev Neurosci. 2008; 9:720–729. [PubMed: 18714327]

Cox RW, Jesmanowicz A. Real-time 3D image registration for functional MRI. Magnetic Resonance in Medicine. 1999; 42:1014–1018. [PubMed: 10571921]

Cox RW, Jesmanowicz A, Hyde JS. Real-Time Functional Magnetic Resonance Imaging. Magnetic Resonance in Medicine. 1995; 33:230–236. [PubMed: 7707914]

Cusack R, Veldsman M, Naci L, Mitchell DJ, Linke AC. Seeing different objects in different ways: Measuring ventral visual tuning to sensory and semantic features with dynamically adaptive imaging. Human Brain Mapping. 2011 n/a-n/a.

deCharms RC. Reading and controlling human brain activation using real-time functional magnetic resonance imaging. Trends in Cognitive Sciences. 2007; 11:473–481. [PubMed: 17988931]

deCharms RC, Christoff K, Glover GH, Pauly JM, Whitfield S, Gabrieli JDE. Learned regulation of spatially localized brain activation using real-time fMRI. NeuroImage. 2004; 21:436–443. [PubMed: 14741680]

deCharms RC, Maeda F, Glover GH, Ludlow D, Pauly JM, Soneji D, Gabrieli JDE, Mackey SC. Control over brain activation and pain learned by using real-time functional MRI. Proceedings of the National Academy of Sciences of the United States of America. 2005; 102:18626–18631. [PubMed: 16352728]

Eklund A, Ohlsson H, Andersson M, Rydell J, Ynnerman A, Knutsson H. Using Real-Time fMRI to Control a Dynamical System by Brain Activity Classification. Medical Image Computing and Computer-Assisted Intervention – MICCAI 2009. 2009; 5761:1000–1008.

Esposito F, Seifritz E, Formisano E, Morrone R, Scarabino T, Tedeschi G, Cirillo S, Goebel R, Di Salle F. Real-time independent component analysis of fMRI time-series. NeuroImage. 2003; 20:2209–2224. [PubMed: 14683723]

Gembris D, Taylor JG, Schor S, Frings W, Suter D, Posse S. Functional magnetic resonance imaging in real time (FIRE): Sliding-window correlation analysis and reference-vector optimization. Magnetic Resonance in Medicine. 2000; 43:259–268. [PubMed: 10680690]

Goebel R, Zilverstand A, Sorger B. Real-time fMRI-based brain computer interfacing for neurofeedback therapy and compensation of lost motor functions. Imaging in Medicine. 2011; 2:407–415.

Hamilton JP, Glover GH, Hsu J-J, Johnson RF, Gotlib IH. Modulation of subgenual anterior cingulate cortex activity with real-time neurofeedback. Human Brain Mapping. 2011; 32:22–31. [PubMed: 21157877]

Hampson M, Scheinost D, Qiu M, Bhawnani J, Lacadie CM, Leckman JF, Constable RT, Papademetris X. Biofeedback of Real-Time Functional Magnetic Resonance Imaging Data from the Supplementary Motor Area Reduces Functional Connectivity to Subcortical Regions. Brain Connectivity. 2011a; 1:91–98. [PubMed: 22432958]

Hampson M, Stoica T, Saksa J, Scheinost D, Qiu M, Bhawnani J, Pittenger C, Papademetris X, Constable RT. Real-time fMRI biofeedback targeting the orbitofrontal cortex for contamination anxiety. Journal of Visual Experiments Accepted. 2011b

Hinds O, Ghosh S, Thompson TW, Yoo JJ, Whitfield-Gabrieli S, Triantafyllou C, Gabrieli JDE. Computing moment-to-moment BOLD activation for real-time neurofeedback. NeuroImage. 2011; 54:361–368. [PubMed: 20682350]

James T,V. Real-Time fMRI Paradigm Control, Physiology, and Behavior Combined with Near Real-Time Statistical Analysis. NeuroImage. 1999; 10:91–106. [PubMed: 10417244]

Joshi A, Scheinost D, Okuda H, Belhachemi D, Murphy I, Staib L, Papademetris X. Unified Framework for Development, Deployment and Robust Testing of Neuroimaging Algorithms. Neuroinformatics. 2011; 9:69–84. [PubMed: 21249532]

LaConte SM. Decoding fMRI brain states in real-time. NeuroImage. 2011; 56:440–454. [PubMed: 20600972]

LaConte SM, Peltier SJ, Hu XP. Real-time fMRI using brain-state classification. Human Brain Mapping. 2007; 28:1033–1044. [PubMed: 17133383]

Lee S, Ruiz S, Caria A, Veit R, Birbaumer N, Sitaram R. Detection of Cerebral Reorganization Induced by Real-Time fMRI Feedback Training of Insula Activation. Neurorehabilitation and Neural Repair. 2011; 25:259–267. [PubMed: 21357528]

Mathiak K, Posse S. Evaluation of motion and realignment for functional magnetic resonance imaging in real time. Magnetic Resonance in Medicine. 2001; 45:167–171. [PubMed: 11146500]

McCaig RG, Dixon M, Keramatian K, Liu I, Christoff K. Improved modulation of rostrolateral prefrontal cortex using real-time fMRI training and meta-cognitive awareness. NeuroImage. 2011; 55:1298–1305. [PubMed: 21147230]

Nakai T, Bagarinao E, Matsuo K, Ohgami Y, Kato C. Dynamic monitoring of brain activation under visual stimulation using fMRI--The advantage of real-time fMRI with sliding window GLM analysis. Journal of Neuroscience Methods. 2006; 157:158–167. [PubMed: 16765449]

Papademetris X, Vives KP, DiStasio M, Staib LH, Neff M, Flossman S, Frielinghaus N, Zaveri H, Novotny EJ, Blumenfeld H, Constable RT, Hetherington HP, Duckrow RB, Spencer SS, Spencer DD, Duncan JS. Development of a research interface for image guided intervention: initial application to epilepsy neurosurgery. Biomedical Imaging: Nano to Macro, 2006. 3rd IEEE International Symposium on. 2006:490–493.

Phan KL, Fitzgerald DA, Gao K, Moore GJ, Tancer ME, Posse S. Real-time fMRI of cortico-limbic brain activity during emotional processing. NeuroReport. 2004; 15

Posse S, Binkofski F, Schneider F, Gembris D, Frings W, Habel U, Salloum JB, Mathiak K, Wiese S, Kiselev V, Graf T, Elghahwagi B, Grosse-Ruyken M-L, Eickermann T. A new approach to measure single-event related brain activity using real-time fMRI: Feasibility of sensory, motor, and higher cognitive tasks. Human Brain Mapping. 2001; 12:25–41. [PubMed: 11198103]

Rota G, Handjaras G, Sitaram R, Birbaumer N, Dogil G. Reorganization of functional and effective connectivity during real-time fMRI-BCI modulation of prosody processing. Brain and Language. 2011; 117:123–132. [PubMed: 20888628]

Rota G, Sitaram R, Veit R, Erb M, Weiskopf N, Dogil G, Birbaumer N. Self-regulation of regional cortical activity using real-time fMRI: The right inferior frontal gyrus and linguistic processing. Human Brain Mapping. 2009; 30:1605–1614. [PubMed: 18661503]

Sander, J.; Kandrot, E. CUDA by Example: An Introduction to General-Purpose GPU Programming. Addison-Wesley Professional; 2010.

Shibata K, Watanabe T, Sasaki Y, Kawato M. Perceptual Learning Incepted by Decoded fMRI Neurofeedback Without Stimulus Presentation. Science. 2011; 334:1413–1415. [PubMed: 22158821]

Sitaram R, Lee S, Ruiz S, Rana M, Veit R, Birbaumer N. Real-time support vector classification and feedback of multiple emotional brain states. NeuroImage. 2010; 56:753–765. [PubMed: 20692351]

Studholme C, Hill DL, Hawkes DJ. Automated 3-D registration of MR and CT images of the head. Med Image Anal. 1996; 1:163–175. [PubMed: 9873927]

Thesen S, Heid O, Mueller E, Schad LR. Prospective acquisition correction for head motion with image-based tracking for real-time fMRI. Magn Reson Med. 2000; 44:457–465. [PubMed: 10975899]

Tokuda J, Fischer GS, Papademetris X, Yaniv Z, Ibanez L, Cheng P, Liu H, Blevins J, Arata J, Golby AJ, Kapur T, Pieper S, Burdette EC, Fichtinger G, Tempany CM, Hata N. OpenIGTLink: an open network protocol for image-guided therapy environment. The International Journal of Medical Robotics and Computer Assisted Surgery. 2009; 5:423–434.

Weiskopf N, Mathiak K, Bock SW, Scharnowski F, Veit R, Grodd W, Goebel R, Birbaumer N. Principles of a brain-computer interface (BCI) based on real-time functional magnetic resonance imaging (fMRI). Biomedical Engineering, IEEE Transactions on. 2004; 51:966–970.

Weiskopf N, Sitaram R, Josephs O, Veit R, Scharnowski F, Goebel R, Birbaumer N, Deichmann R, Mathiak K. Real-time functional magnetic resonance imaging: methods and applications. Magnetic Resonance Imaging. 2007; 25:989–1003. [PubMed: 17451904]

Yoo S-S, Fairneny T, Chen N-K, Choo S-E, Panych LP, Park H, Lee S-Y, Jolesz FA. Brain-computer interface using fMRI: spatial navigation by thoughts. NeuroReport. 2004; 15

Yoo S-S, Jolesz FA. Functional MRI for neurofeedback: feasibility studyon a hand motor task. NeuroReport. 2002; 13

Yoo S-S, O'Leary HM, Fairneny T, Chen N-K, Panych LP, Park H, Jolesz FA. Increasing cortical activity in auditory areas through neurofeedback functional magnetic resonance imaging. NeuroReport. 2006; 17

Zotev V, Krueger F, Phillips R, Alvarez RP, Simmons WK, Bellgowan P, Drevets WC, Bodurka J. Self-Regulation of Amygdala Activation Using Real-Time fMRI Neurofeedback. PLoS ONE. 2011; 6:e24522. [PubMed: 21931738]

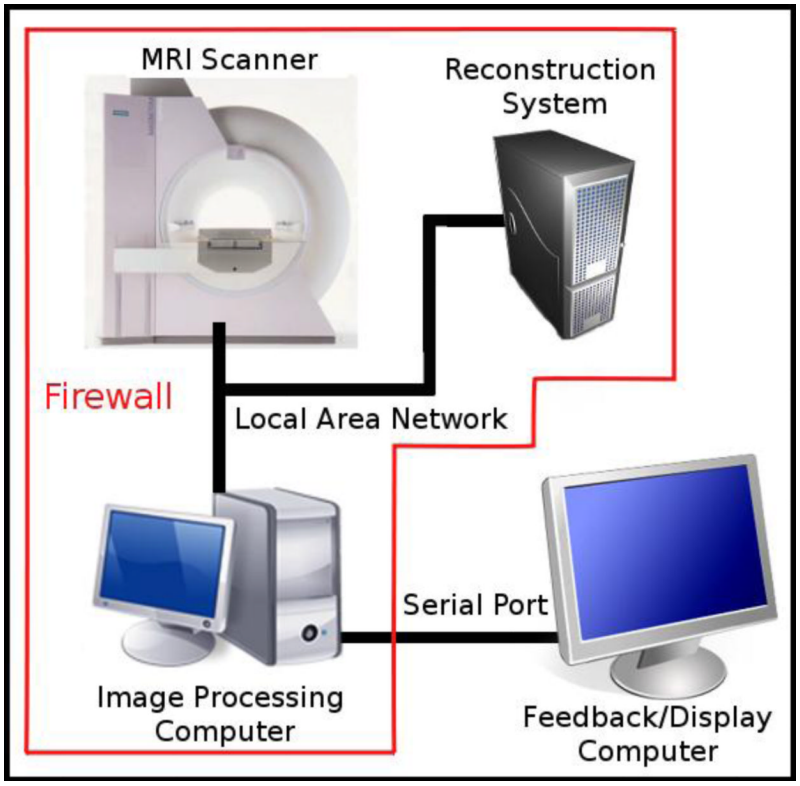

Figure 1. Schematic of the Rt-fMRI Setup

Our setup consists of the MRI scanner, image reconstruction system, image processing computer, and the feedback/display computer. The MR data are processed by the reconstruction system creating an image of each slice/volumes that is written to a file. The slices/volumes are retrieved by the image processing computer via LAN and processed in real time using BioImage Suite. The results are sent to the feedback computer via serial port and shown to the subject.

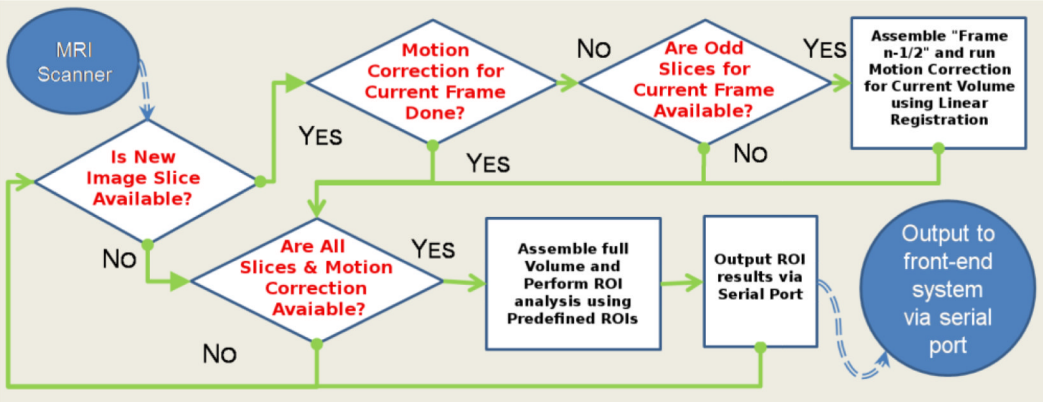

Figure 2. Flow Chart of the Back-End Processing Component

The rt-fMRI Back-End sits in between the MRI scanner and Front-End display. If eacsh slice of a volume can be saved in real-time, the Back-End will proceed as outline above. First, if all odd slices are acquired, frame *n*-½ is created and motion correction is performed. Second, if all slices are acquired and the motion correction is finished, the motion correction is applied, ROI analysis is performed, and the results are sent the Front-End. Finally, if neither set of slices are acquired, the Back-End continues to poll for new slices. If each slice of a volume cannot be saved in real-time, the Back-end will proceed in a similar manner with the exception of using frame *n* for motion correction.

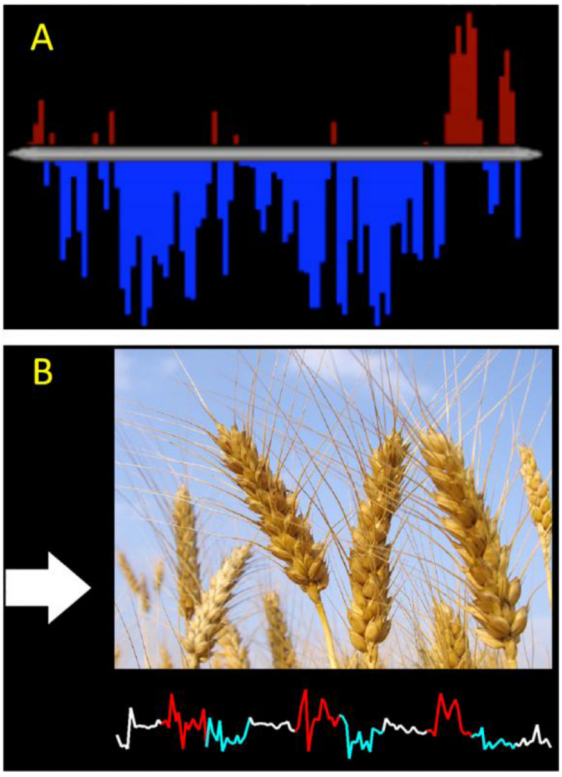

Figure 3. Example Front-End components

A) This Front-End is written in E-Prime and shows feedback from the PCC after addition signal processing. The BOLD time-course from the PCC is shown as a bar graph with red bars representing an increase over baseline and blue bars representing a decrease under baseline. B) This Front-End is written is Matlab and shows both activation of the OFC cortex and images designed to provoke or subdue contamination related anxiety (the wheat in this example). These Front-Ends used in ongoing rt-fMRI experiments show the flexibility of our system to feedback design.

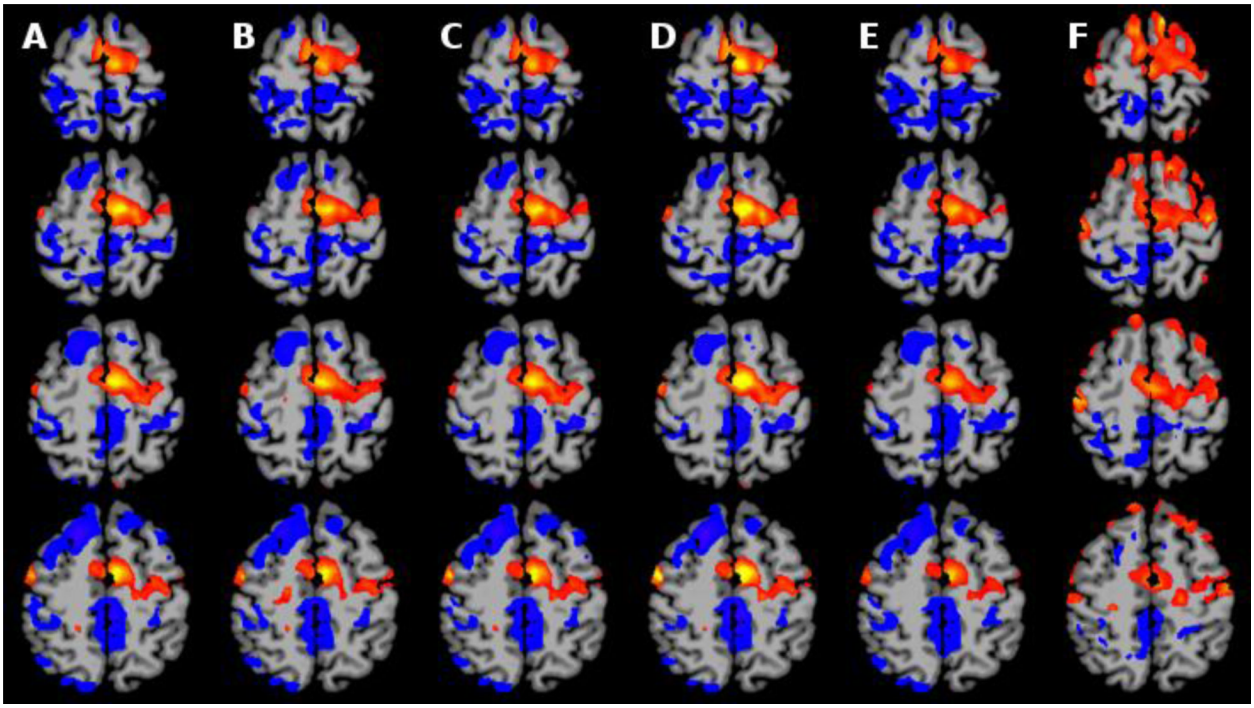

Figure 4. Comparison 5 Different Motion Correction Algorithms on Empirical Rt-fMRI Data

A) SPM Realign and B) FSL MCFLIRT are two standard motion correction algorithms used for comparison. The proposed algorithm using C) Frame *n*, D) Frame *n-½*, and E) Frame *n-1* to estimate motion for Frame *n*. F) Results from data without motion correction. The five motion correction algorithms look similar to each other. All show a reduction in motion artifacts near the edges and deactivation in Frontal Lobe not seen in the uncorrected data.

Table 1
Overlap of Significant Cluster Produce by Different Motion Corrections

SPM and FSL columns represent each software packages standard motion correction algorithm with default parameters. The *n*, *n-½* and *n-1* columns represent the presented motion correction algorithm using frame *n*, *n-½*, and *n-1* to correct for the motion in frame *n*. Please note the dramatic (and statistically significant deterioration) between the two approach that do not add delay to the system, our proposed *n-½* approach and the commonly used *n-1* approach.

	FSL	**N**	**N-½**	**N-1**
SPM	80.20	81.68	78.95	69.07
FSL		80.35	78.76	68.69
N			87.96	75.29
N-½				76.05