Susceptibility Weighted Magnetic Resonance Imaging

A THESIS SUBMITTED TO THE GRADUATE DIVISION OF THE UNIVERSITY OF HAWAIʻI AT MĀNOA IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCES

IN

ELECTRICAL ENGINEERING

May 2011

By

Eric K. Baxter

Thesis Committee:

Victor Stenger, Chairperson
Alek Kavcic
Galen Sasaki

Keywords: GPU spiral recon, relaxometry, quantitative susceptibility, iron concentration

DEDICATION

The culmination of this research and work is dedicated to my loving wife and son for their patience and support during this time along with my father for his example of perseverance and my mother for instilling in me her curiosity and love of science.

ACKNOWLEDGMENTS

My deep appreciation goes to Dr. Victor A. Stenger for allowing me this wonderful opportunity to conduct this research with him and his team. His mentoring is the chief reason for the success of this effort. I would also like to thank Dr. Linda Chang and Dr. Thomas Ernst for their support in bringing this to fruition. My gratitude also goes to Dr. Weiran Deng who was my principle source for learning about the GPU and its programming. Finally, my thanks go to the entire 3T MRI Research Lab team for their friendship and commiseration in working to solve the challenges of MRI.

ABSTRACT

Susceptibility-weighted imaging (SWI) is an important MRI modality for monitoring the progression of disease in the human brain. SWI is particularly sensitive to iron. Iron concentration holds great clinical interest in the progression of many neurological diseases as well as imaging blood. Challenges of these methods include long acquisition and reconstruction times as well as artifacts due to bulk susceptibility variations such as air/tissue boundaries. The scans in this study use spiral trajectories which are efficient but computationally demanding to reconstruct which has prevented spiral adoption in clinical applications.

In this thesis I helped address these concerns with the following four engineering developments:

1. A simple method for T2* relaxometry was tested and analyzed. T2*-weighting imaging proved most effective for detecting these effects.

2. An examination of the benefits and costs of a spiral-in versus spiral-out k-space trajectory is presented. For observing blood vessels in MIPs, spiral-in shows advantages due to T2* relaxometry high-pass filtering.

3. Long scans with high resolution are required to image small iron deposits. High resolution scans become computationally challenging. A GPU-accelerated implementation of the conjugate-gradient least-squares algorithm is presented to reconstruct parallel MRI images quickly and accurately. This algorithm incorporates

environmental corrections including T2* relaxation effects, coil SENSE, and off-resonance field map to improve accuracy. The speed and effectiveness of this GPU implementation is presented here.

4. The CGLS algorithm incorporating Tikhonov regularization is also implemented to solve for quantitative susceptibility as a more direct measurement of iron concentration in tissue. Results of a GPU-accelerated implementation of the susceptibility solver are presented and compared with current literature and my relaxometry results.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 MRI Physics

Magnetic Resonance Imaging (MRI) has become one of the most widely used medical imaging techniques today. MRI makes use of magnetism and radio frequency (RF) transmissions rather than ionizing radiation used by other biomedical scan technologies. This removes concerns of damage to DNA and makes MRI an attractive choice for longitudinal studies such as monitoring the progression of disease and effects of aging. While MRI can be employed in many flexible modalities, the underlying principle remains Nuclear Magnetic Resonance (NMR).

NMR was independently discovered in 1946 by Felix Bloch and Edward Purcell. These two men shared the Nobel Prize for Physics in 1952 for their work. The principle of NMR derives from the observation that many nuclei have both intrinsic magnetic moments and quantum spin. In their natural state most objects do not exhibit any net magnetization because the magnetic moments of each of the nuclei are not aligned. When an external magnetic field, B0, is applied to the object the magnetic moments of the nuclei are forced to align themselves parallel to the external field. This yields a net magnetization from the object.

The application of an external magnetic field will induce a torque on the magnetic moment of the nuclei. This will cause a precession of the moment around the axis of the applied field. The angular momentum of this precession is specific to the nuclei and

proportional to the strength of the applied field. The angular frequency, known as the Larmor frequency is described by: $\omega = -\gamma * B0$. The gyromagnetic ratio, γ, is a isotope-specific ratio of proportionality between angular frequency and the applied field. The hydrogen atom is most commonly imaged due to the abundance of water in the brain and has a gyromagnetic ratio of $\gamma = 42.58$Mhz/T. The precessing nuclei are now susceptible to energy absorption when an electromagnetic pulse is applied at the correct resonant frequency equal to the Larmor frequency. These resonant frequencies fall within the RF range of the electromagnetic spectrum. Applying a RF pulse orthogonal to B0 at the Larmor frequency will cause the nuclei to absorb energy and begin to precess at an angle further away from the direction of the B0 field. This angle is known as the *flip angle* and is proportional to the energy of the RF pulse. The nuclei are now in a higher energy state. When the RF pulse is removed the nuclei will radiate RF energy as their flip angle decreases again toward the axis of B0. This process is known as *relaxation* and yields the means to measure local magnetism through the Maxwell-Faraday equation.

1.2 Signal acquisition

When relaxation occurs the change in magnetization in the spin system causes RF energy to be radiated from each nucleus and return to its normal energy state. In NMR this signal is detected by the principle of electromagnetic induction by a nearby receiver coil. A small current is induced in the coil which is then converted to voltage which is then digitized and stored. The actual electromagnetic response of each coil is dependent of magnetic loading which varies with subjects' physiology and can effect phase measurements between each coil.

1.3 Sampling

Sampling of k-space must be done in such a manner as to allow us to reconstruct an image of the scanned object with sufficient quality. Enough detail must be present in the image and the entire region of interest (ROI) must be displayed for an image to be of diagnostic quality. These two criteria are controlled by k-space sampling.

The finer details of an image are often of great interest. These may be tissue boundaries, blood vessels, or signs of traumatic injury such as hemorrhagic lesions. Small, highly localized, details in images are represented in the higher spatial frequencies. These frequencies correspond to the farther regions of k-space. Most of the energy in the image will be obtained at relatively low frequency regions centered around the k-space. In spiral imaging, the sampling density is highest in these low spatial frequency areas. While having less of the overall signal energy the higher spatial frequency areas contain the more visually interesting details. Unfortunately, reaching these higher frequencies requires either very strong gradients with fast slew rates or additional scan time at more reasonable gradients. Strong gradients with high slew rates may not be realizable as they can induce Peripheral Nerve Stimulation (PNS) and violate Specific Absorption Rate (SAR) limitations. Both of these effects must be mitigated in order to maintain proper safety margins. The alternative is to use smaller gradients and take a longer time in reaching higher spatial frequencies. This is one of the fundamental tradeoffs in obtaining high-resolution MRI images.

The other major sampling consideration for MRI is obtaining a proper Field-of-View (FOV). The FOV is the image's region of support based upon the underlying k-space sampling scheme. The FOV is a consequence of the Nyquist-Shannon sampling

theorem and is defined as $1/\Delta k$. Decreasing the distance between k-space samples will expand the unaliased region of image space, the FOV. Given an object of a known size, it can be determined the appropriate sampling density to use in k-space to establish a sufficient FOV. Of course, increasing the sampling density, thus expanding the FOV, also requires additional samples and longer scan time. If it is desirable to keep scan time short, one must settle for a lower k_{max} which decreases the image resolution. These are the tradeoffs that must be made for an unaliased reconstruction.

1.3.1 Spiral Trajectories

All scans and tests done in this study were performed by spiral k-space trajectories. Spiral trajectories have several advantages that make them very attractive. Spiral sampling is done by oscillating the k_x and k_y gradients to sample in a spiral k-space pattern. Such a trajectory is seen in Figure 1.

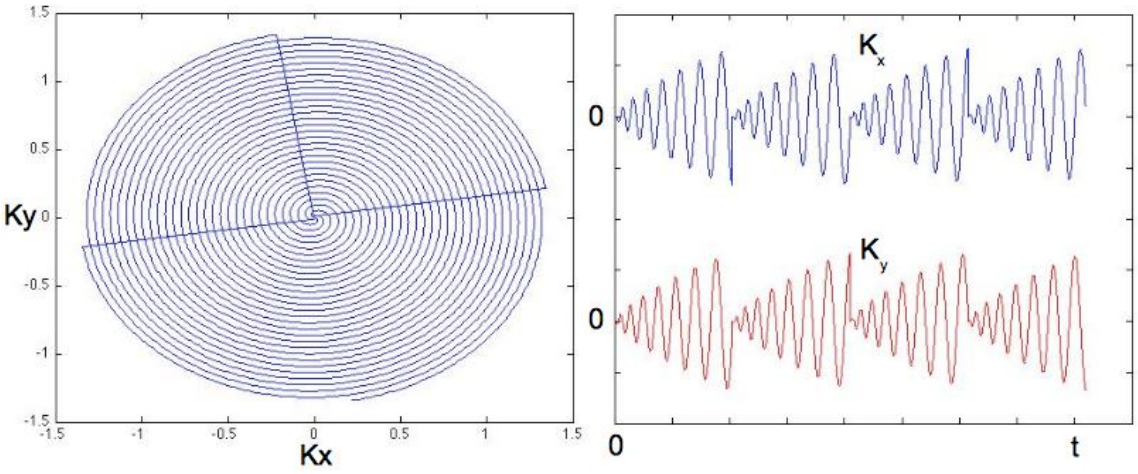

Figure 1: Illustration of spiral trajectory sampling using 4 interleaved spirals. The k-space plane is seen in a) while the behavior of k_x and k_y gradients is seen in b).

Spiral sampling is ideal for quickly and efficiently covering k-space. By sampling in a spiral pattern we can keep the gradients from abrupt changes which could induce PNS in the subject. Maximum slew rates for gradient switching can be constantly

achieved without uncomfortable or potentially dangerous side effects to the subject. The spiral pattern is a very fast method of scanning because the gradients almost constantly operate at their maximum while yielding good coverage of k-space.

While the spiral has many sampling advantages, the non-equispaced sampling locations disallow the use of the FFT and make it computationally demanding to reconstruct into an image. The challenges associated with this are discussed in detail in Chapter 4.

1.4 Imaging

While these signals can be detected more must be done in order to form an image of an object. Spatial information must be encoded for a coherent image to be reconstructed. As stated in Section 1.1, the Larmor frequency is directly proportional to the applied magnetic field. Orthogonal field gradients are applied concurrent with the B0 field. This varies the resonant frequency along the B0 field axis, usually referred to as the z axis. The resonant frequency is now a function of position along the B0 field. This allows one to select a plane perpendicular to the z axis by selecting the specific resonant frequency for that position. All nuclei within the object are subjected to the selective RF pulse but only those precessing within the bandwidth of pulse absorb and then emit RF energy. Any signals detected at the receiving coils are then known to originate from nuclei in that plane. The bandwidth of the RF slice selection pulse dictates the slice thickness and has a powerful effect on the signal-to-noise ratio (SNR). The thickness of scan slices is also dictated by the bandwidth.

Spatial information must also be encoded in the xy in-plane directions. Frequency encoding gradients in the x and y-axis directions are now applied to correlate spatial

position with gradient strength. During a signal acquisition, these gradients, G_x and G_y are varied according to the pattern of trajectory dictated by the user. The gradients are representative of spatial frequencies being sampled as the scan progresses. Plotting the spatial frequencies forms an abstraction known as k-space. For each position in k-space a complex value is sampled by the receiver coils and stored. These data form a two dimensional representation of the Fourier transform of the spatial image of the scanned object. For our purposes a spiral k-space trajectory was used for all scans.

1.4.1 T2* decay

T2* relaxation is due to de-phasing effects of transverse magnetization after the application of the RF excitation pulse. The ferromagnetic properties of iron cause the T2* effect to be accelerated. Iron acts as its own magnetic source in the surrounding tissue and prompts de-phasing to occur more quickly. This effect is illustrated in Figure 2. Clearly T2* is sensitive to the magnetic susceptibility of the tissue being scanned. Fast de-phasing is observed as darker areas in image magnitude compared to surrounding tissues with normal T2* relaxometry.

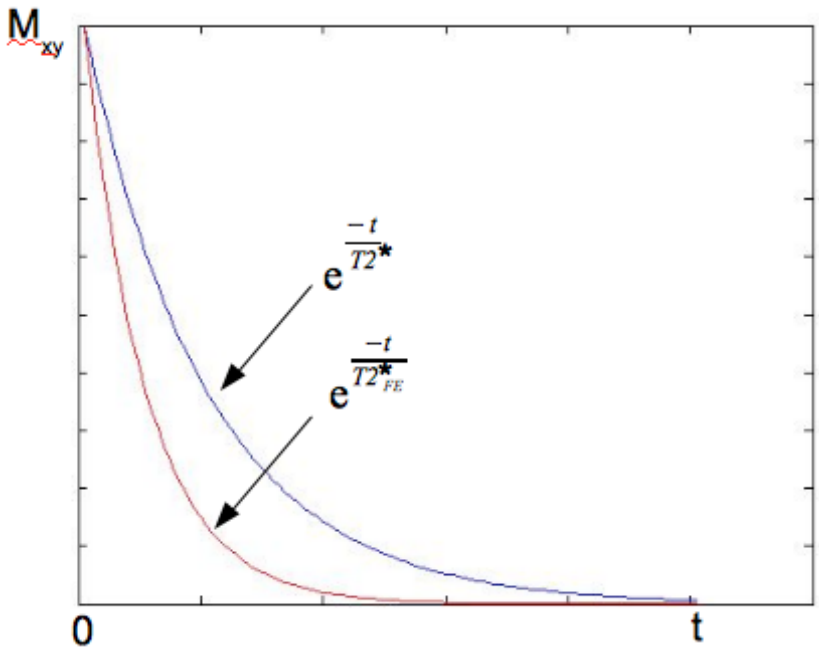

Figure 2: T2* decay illustration. T2* relaxation time is shortened in the presence of iron.

1.4.2 Sensitivity Encoding (SENSE)

In order to reduce the amount of time patients and subjects spend in the scanner parallel imaging techniques (pMRI) have been developed. Time is gained during scans by sampling smaller sets of k-space. Less data is being acquired and so less time is needed for the gradients to complete their decreased trajectories. This undersampling saves scan time but increases the Δk between samples and shrinks the FOV accordingly. Trying to reconstruct an object smaller than the new FOV will result in aliasing of that object in image space. Information from multiple receiver coils is used to combat this.

Sensitivity encoding (SENSE) was first proposed by Pruessman and Weiger (1) as a way to minimize aliasing and increase SNR by using multiple receiver coils to insert some additional spatial information into scan data. A coil's sensitivity is related to its proximity to the signal source, in this case hydrogen protons. A coil's sensitivity is

increased for those parts of the object closest to it and diminishes radially outward. Increasing the number of coils can also increase the reduction factor that can be gained in scan time.

SENSE maps are typically measured by obtaining a fully sampled dataset at low resolution. The relative strength of the signal magnitude can then be determined for each coil at every point within the FOV. This concept is demonstrated in Figure 3.

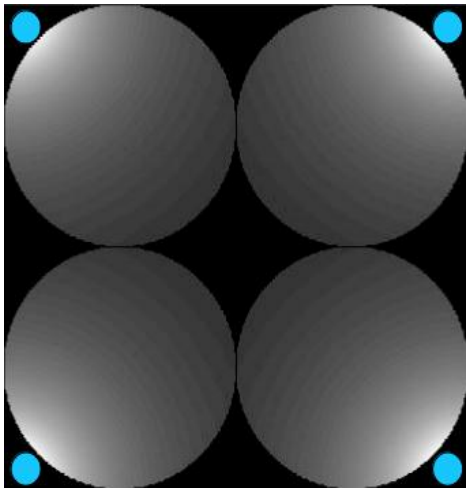

Figure 3: Illustration of coil sensitivities. Coils (blue circles) each have their own sensitivity map.

By knowing the sensitivity map of each coil we can weight the scan data appropriately in reconstruction to significantly reduce aliasing artifacts. A simulated demonstration of SENSE can be seen in Figure 4.

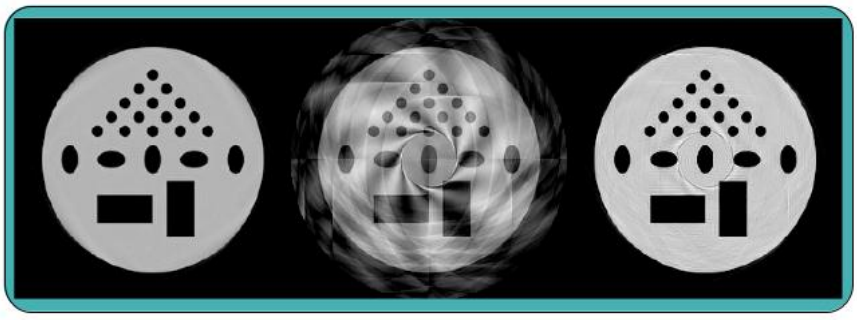

Figure 4: a) fully sampled reconstruction. b) 2x undersampled with no SENSE map. c) 2x undersampled with SENSE map

1.4.3 Field map correction

Ideally the B0 field is homogeneous, however; in practice this is not the case. When performing *in vivo* scans, field inhomogeneities arise near air/tissue boundaries. When imaging the brain, this problem is often observed above the sinus cavities and surrounding ear canals. This disturbance in the field's flatness causes an off-resonance effect to occur in those areas. The Larmor frequency is now regionally varying depending on the physiology of the subject. The larger the off-resonance the faster signal decay will occur in T2 and T2*. The problem is compounded with longer scans as the effect accumulates. This is illustrated in Figure 5. Signal voids appear tissue areas that may be of interest. With knowledge of the off-resonance field map it is possible to apply some correction in off-line image reconstruction.

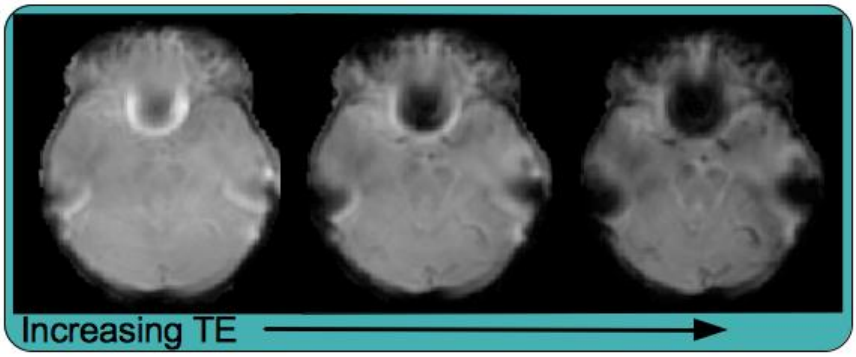

Figure 5: susceptibility artifact worsening with longer TE

Here we represent the spatially varying off-resonance term as $\omega(r)$ later in Equation. 1. Calculation of this parameter is done by performing two scans with a small ΔTE between them. Two phase maps are created from these scans. The difference between these two phase measurements and the known time between them gives us the off-resonance map for our calculations.

1.5 The Scanner

The scanner used for all MRI scans in this study was a Siemens Trio 3T whole-body scanner using 4 head coils. Brain images presented in this paper were all acquired *in vivo* from adult subjects in accordance with IRB standards. Each dataset was scanned using a spiral sequence with multiple interleaves. For the susceptibility-weighted images shown here the slice thickness was 2mm with a 15^{o} flip angle, TE = 30ms, and TR = 1s.

CHAPTER 2

T2/T2* RELAXOMETRY

2.1 Clinical Background

In 1958 Hallgren and Sourander (2) published a landmark paper from a postmortem study of iron distribution throughout the brain. Since then, iron concentration has been the subject of many studies investigating iron and its role in both the aging process and disease progression. Iron's ferromagnetic properties make it an ideal choice for imaging using magnetic resonance techniques *in vivo*. Iron concentration in the brain is of medical interest in many neurodegenerative diseases including Alzheimer's and Parkinson's (3), Huntington's (4) (5), multiple sclerosis (6), HIV (7) and a slew of others. With such a wide-ranging neuropathy it is clear the ability to monitor iron in the brain could be a crucial clinical tool.

Directly imaging iron with MRI can be problematic. The ferromagnetic nature of iron causes de-phasing and signal decay more rapidly than the tissue itself. This causes signal gaps to appear at shorter TEs in areas of higher iron concentrations. This effect is evidenced by a shorter transverse relaxation time, T2, as well as T2*. We expect relaxation times should also decrease, signaling increased iron, as a part of the aging process as well. This would correlate with Hallgren and Sourander's results of elevated iron as a function of age.

2.2 Method

This study scanned 43 healthy HIV-negative and 34 HIV-positive subjects ranging

in age from 18-72 years. To balance age distribution for covariance analysis 34 subjects of each group were compared. The seronegative subjects' ages were 50 ± 12.4 years. The HIV-positive subjects' ages ranged 48 ± 9.7 years. According to (7), HIV-positive subjects should exhibit a quicker T2 and T2* signal decay due to increased iron accumulation. The normal aging process should also evidence this effect. A multi-scan method of increasing echo times was tested to see if this effect could be detected and if so to what extent.

To measure T2 and T2*, relaxation ten scans of both types were done per research subject with increasing TE for each scan. T2-weighted images were obtained at TEs ranging from 8-150ms. T2*-weighted images were obtained at TEs ranging from 3-100ms. Every scan was performed on the Siemens Trio 3T whole-body scanner using four head coils. The scan used four spiral interleaves with a 4s TR and a flip-angle of 90^{o}. The FOV was set at 22cm. Each scan obtained 24 5mm thick slices reconstructed to a 128 x 128 x 24 image matrix.

The image volumes were placed into a 4D matrix, the 4^{th} dimension being TE, for linear registration to a standard MNI brain in the FSL (8) software package. The MNI brain is a commonly used template for standardizing an image space for group analysis. Registration of the scanned images was done in the FSL software package by applying an affine transformation to translate the images into a standard image space. The results were then loaded into MATLAB for numerical analysis.

To determine the actual values of T2 and T2* decay within a voxel an exponential decay was fit for each voxel along the TE dimension of the 4D matrix. This yielded a rate of decay for that subject at the voxel. This was done throughout the whole brain to

produce full mappings of T2 and T2* decay. Example images can be seen in Figure 6.

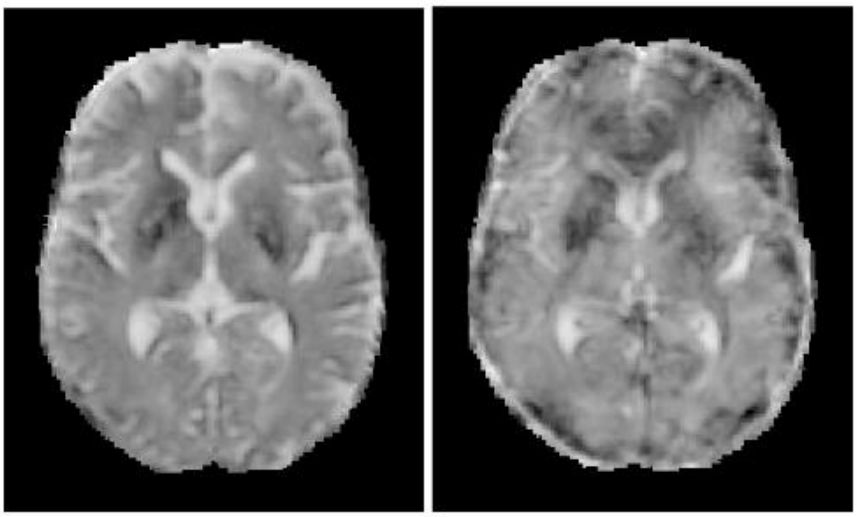

Figure 6: images of typical mappings of a) T2 and b) T2* after using exponential fitting.

2.3 Analysis

If these T2 and T2* mappings are to be useful in monitoring the aging process, we must determine whether we can observe an age-related effect on the relaxometry. Subjects' T2 and T2* mappings were arranged in aged order then used to calculate values for T2-versus-age and T2*-versus-age maps. These maps of the rate of T2 and T2* change are displayed in Figure 7. A blurring effect is evident due to the group averaging that has been done when combining many subjects into a single image. Even with some blurring anatomical structures can still be observed.

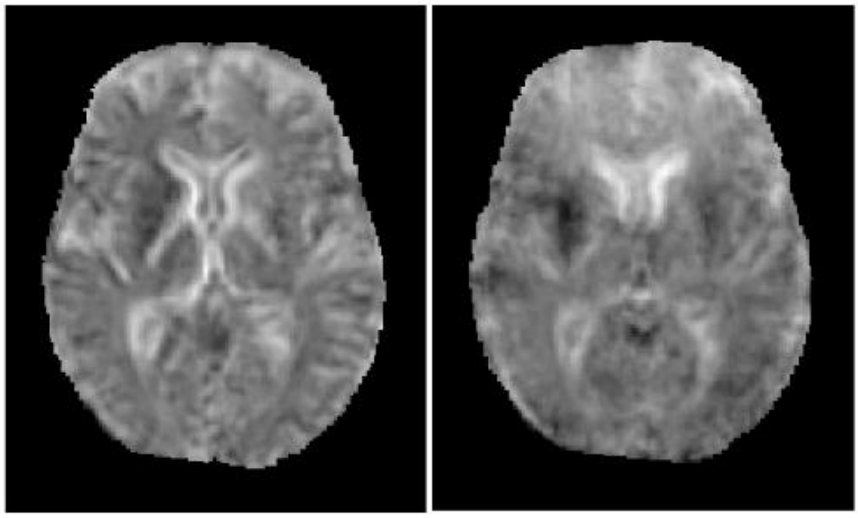

Figure 7: Mapping of a) T2 versus age and b) T2* versus age. Darker regions indicate faster relaxation with increasing age.

As aging occurs, such T2- and T2*-vs-aging maps show the fastest signal relaxation in the putamen and pallidum in both hemispheres. This appears more pronounced in the left hemisphere of the brain. This observation was present for both HIV-positive and negative subject groups.

ANCOVA tests were done to establish whether these observations were statistically significant. The two subject groups' T2 measurements at each voxel in the scan volume were tested co-varying for age. HIV status, age, and their statistical interaction on the relaxometry measures were calculated for each voxel. This created a whole brain map of the statistical significance of the differences observed both between the groups and as a result of the aging process. The age distributions of both groups were matched as closely as possible to increase the certainty of statistical results.

2.4 Results

The ANCOVA calculations were performed in MATLAB using their statistical toolbox and aoctool. A T2 age effect was measured in both putamen, particularly in the

left putamen, with the level of statistical significance being $p<0.0005$ as displayed in Figure 8.

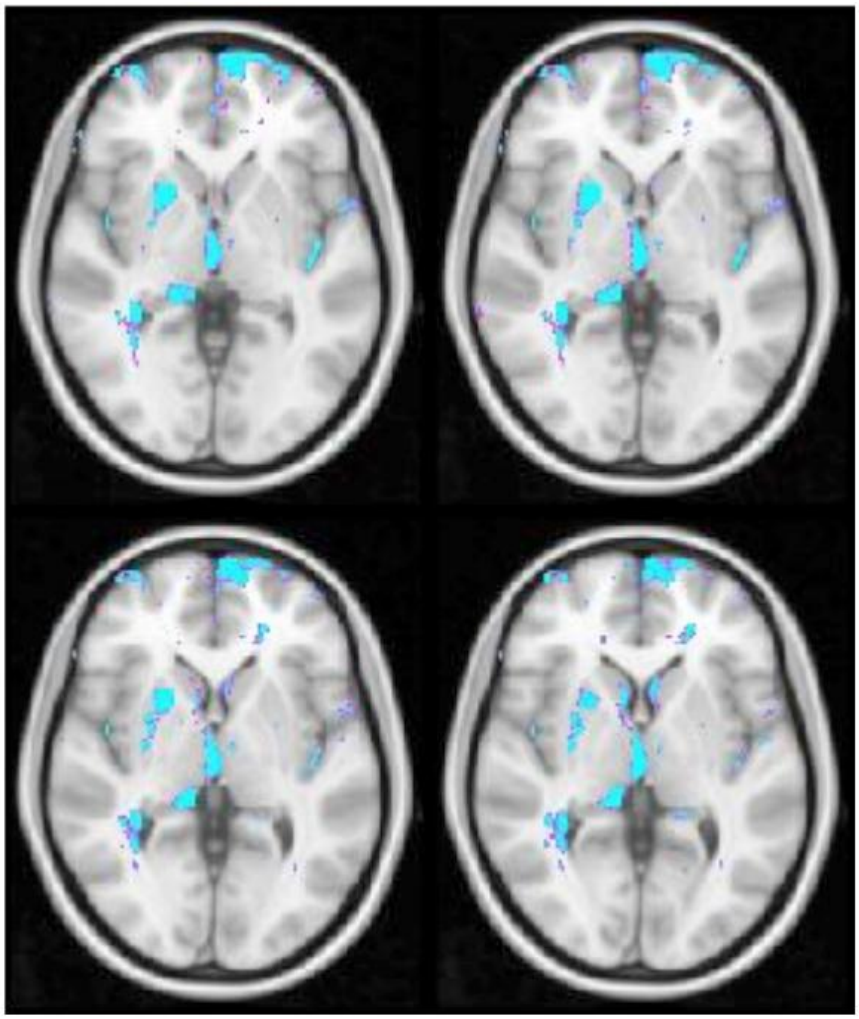

Figure 8: Statistical significance maps of T2 age effect overlaid on the MNI brain template for anatomical reference. Blue regions indicate statistical certainty of $p<0.0005$

No similar regions of statistical significance were observed when testing for an HIV-related effect on the relaxometry measures for T2. Significant age-HIV statistical interaction was not observed in the T2 scans either.

When testing the T2* scans, strong statistical significance for an age-related effect was observed again in the left putamen with a certainty of $p<0.0005$. A mapping of this

result is seen in Figure 9 for slices of interest.

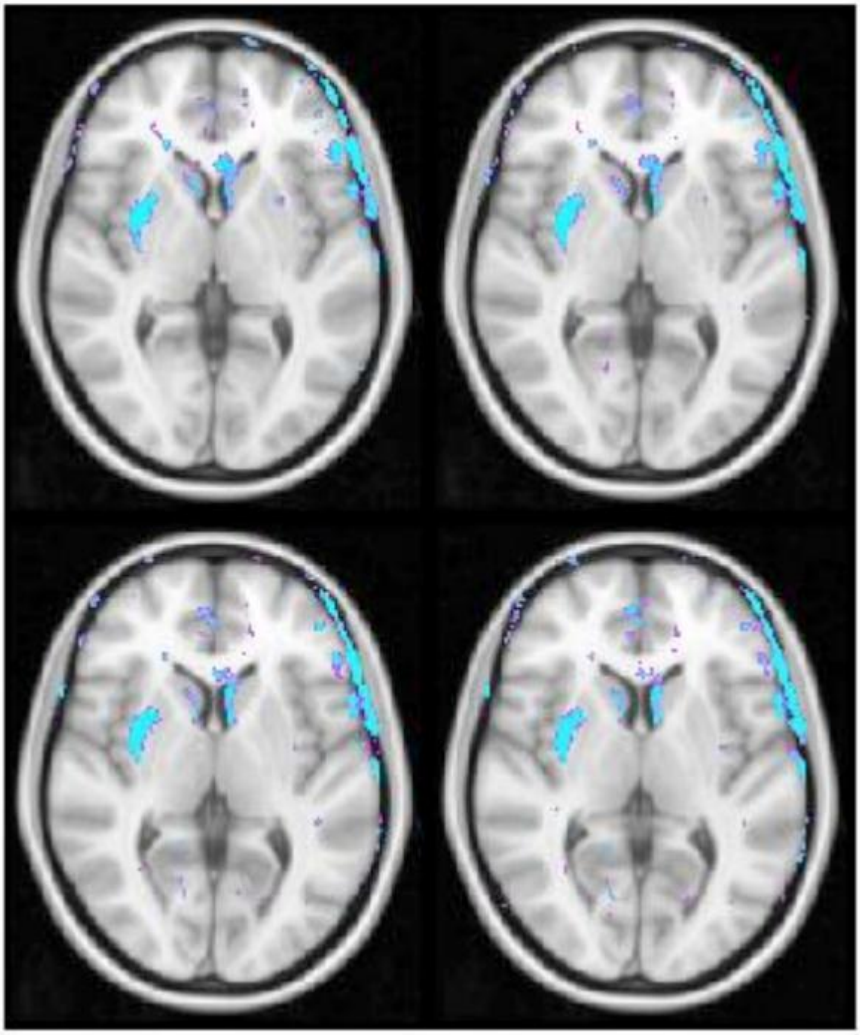

Figure 9: Statistical significance maps of T2* age effect overlaid on the MNI template for anatomical reference. Blue regions indicate statistical certainty of p<0.0005.

In analyzing the HIV-driven effect on T2* measurements some statistical significance was calculated in the pallidum in both hemispheres where $p<0.02$. This mapping can be seen in Figure 10.

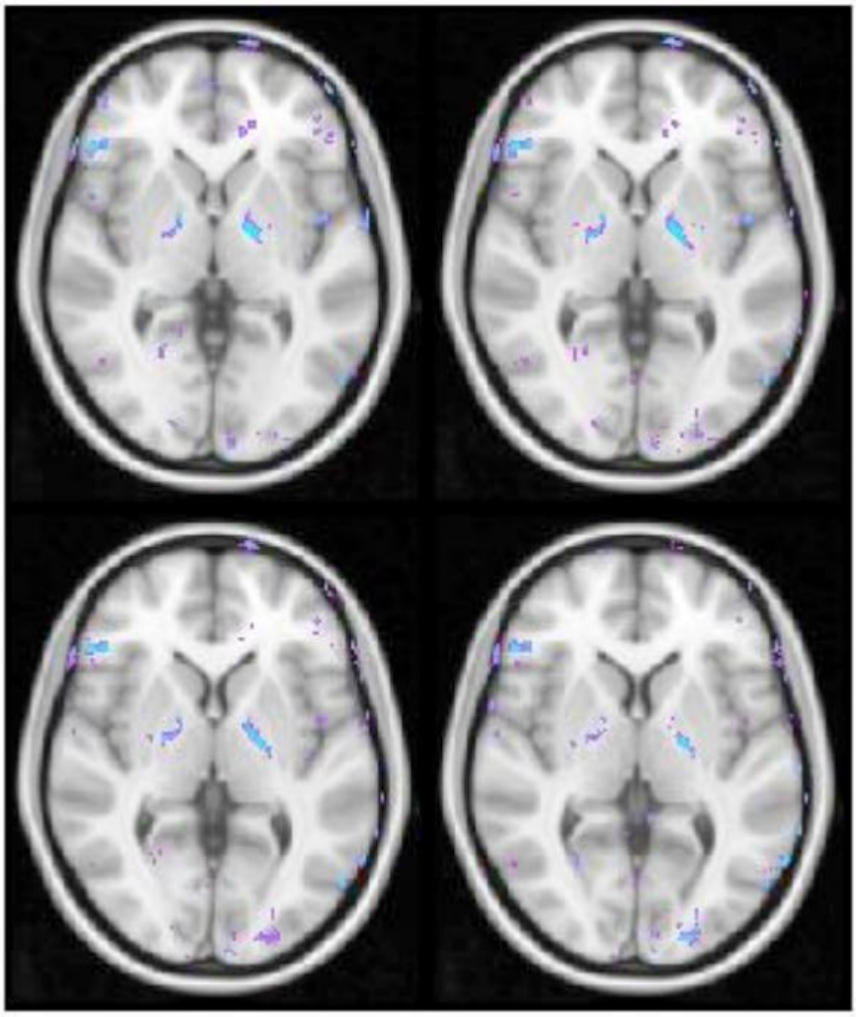

Figure 10: Statistical significance maps of T2* HIV effect overlaid on the MNI template for anatomical reference. Blue regions indicate statistical certainty of p<0.02.

T2* scans were also able to show an age-HIV related interaction in the relaxometry measurements. The result is less significant than age itself but it is still present. A mapping of this significance can be seen in Figure 11 for $p < 0.01$. As with the HIV effect by itself, this interaction was most significant in the pallidum.

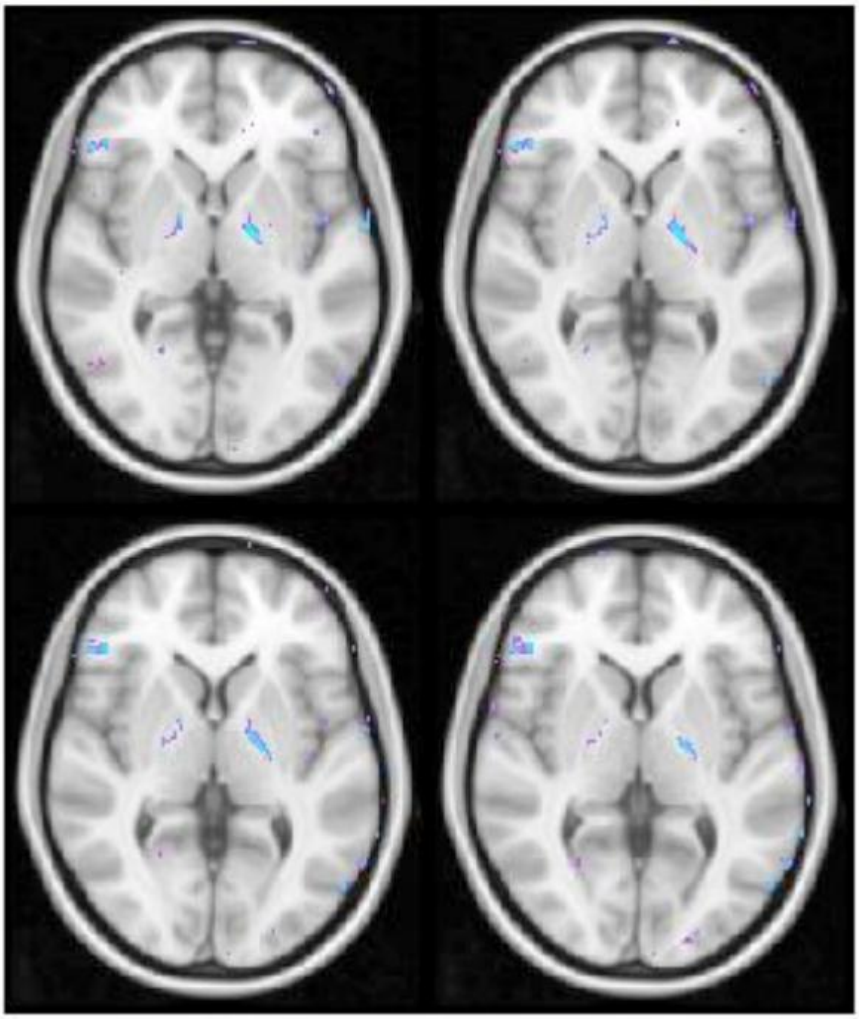

Figure 11: Statistical significance maps of T2* age-HIV interaction effect overlaid on the MNI template for anatomical reference. Blue regions indicate statistical certainty of p<0.01.

The T2 age-related effects in the putamen and pallidum are in agreement with current literature such as (9). HIV-driven effects seen in T2* concurs with a similar study done by KA Miszkiel et al., (7). Asymmetry between the hemispheres is a well-known neurological issue. A review of this literature can be found in (10). This may be in part due to handedness, education, linguist skills, or other tasks which are hemispherically preferential.

The clinical importance of these specific results is outside the scope of this

investigation but the agreement with literature serves as validation that the method is sound. Aging effects and disease progression both show statistical significance in specific areas of the brain. However, a relaxometry investigation like this is only useful for groups and cannot be easily applied to the individual. As a surrogate marker for iron concentration, relaxometry is able to show relativistic changes in tissues but yields no way of measuring actual iron density in any given region. Furthermore, large voxel sizes suffer from partial volume effects. This is the averaging of effects within each voxel which can mask out minute changes. Using a higher resolution scan with very accurate image reconstruction is desirable. A fast and accurate method for image reconstruction is necessary to accommodate higher resolutions with more k-space samples. This motivates the use of GPUs (Graphics Processing Units) in Chapters 4-7. Furthermore, quantitative susceptibility measurements are needed to begin measuring actual iron concentrations *in vivo*. An approach to this problem is presented in Chapter 8.

CHAPTER 3

INHERENT HIGH-PASS FILTERING WITH SPIRAL-IN TRAJECTORIES

3.1 T2-Based Spatial Frequency Weighting

T2* signal decay occurs even as a MRI readout is being acquired. This has a filtering effect on any images reconstructed from the k-space data. Spatial frequencies which are obtained early during readout have endured less T2* signal decay and so will be weighted more heavily in image reconstruction. This is demonstrated in figure 12. Notice the classical high-pass filtering shape inherent in the spiral-in acquisition.

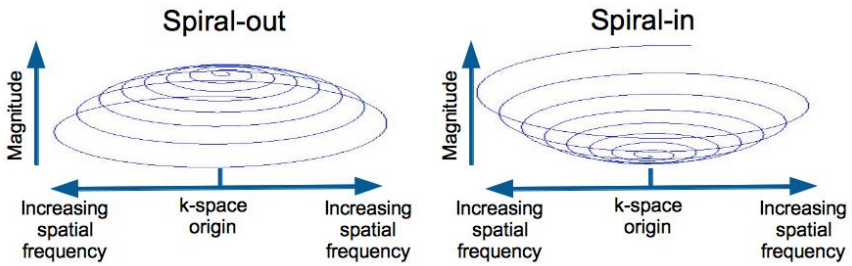

Figure 12: Illustration of the T2* relaxation effect on spiral-out versus spiral-in. Both spirals trace same k-space locations.

It was hypothesized that a spiral-in trajectory would be weighted toward the higher spatial frequencies because they would be acquired before significant T2* decay had occurred. This should give spiral-in images better fine details while only mildly sacrificing overall contrast. We also anticipated that spiral-in scans would also show some signal recovery near air-tissue boundaries were susceptibility artifacts are large. This occurs in regions such as those above the sinuses and around the ear canals. These regions, particularly above the sinuses, are of neurological interest and T2* decay effects present a challenge for imaging.

Images were reconstructed from both spiral-in and spiral-out trajectories and qualitatively compared. Volumes of susceptibility-weighted images (SWIs) for both spiral-in and spiral-out scans consisted of 32 2mm thick slices reconstructed to a 512 x 512 x 32 image. The FOV was set to be 22cm, TE = 30ms, TR = 100ms, with a flip-angle of 30^{o}. The spiral-in was immediately followed by the spiral-out. This gives a nearly identical TE for both trajectories. The k-space trajectory had 36 interleaved spirals of 5,120 samples each. Identical k-space trajectories were traced for both spiral-in and spiral-out. The gradient waveforms shown in Figure 13 show the progression of the spiral-in/out protocol.

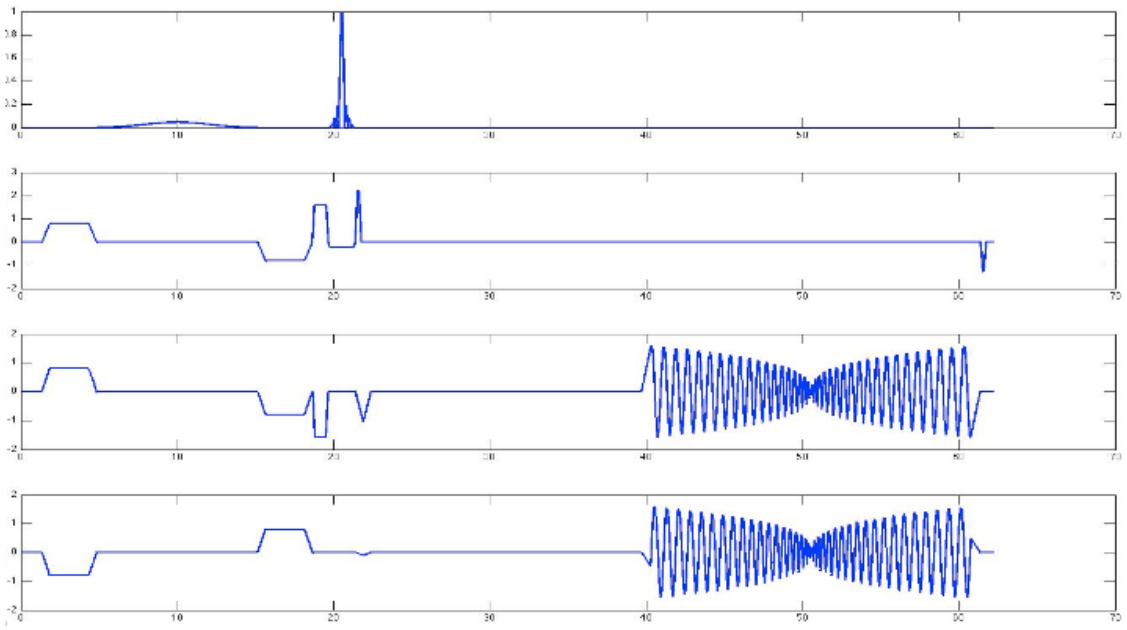

Figure 13: Waveforms of scan protocol used to acquire both spiral-in and spiral-out scans with similar TEs.

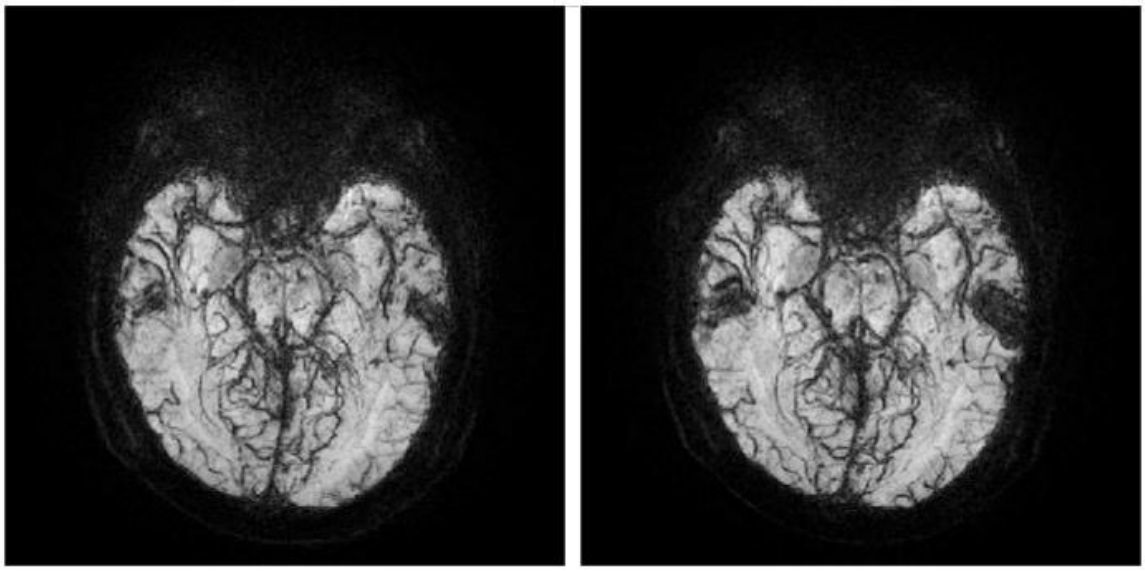

Figure 14: MIPs over 7 slices of a) spiral-in and b) spiral-out acquired scans.

3.2 Results

Spiral-in and spiral-out images were reconstructed using identical methods. Because the images were simultaneously acquired they needed no further steps to make direct comparisons. As theorized, the spiral-in images displayed more fine details than the spiral-out images. This is most clearly seen around the edges of blood vessels. A typical method for viewing blood vessels is to produce what is called a Minimum Intensity Projection (MIP). This is done by simply projecting the minimum intensity value along a desired line of sight into a single plane. Blood vessels show up as dark regions in SWI and so these MIPs will show the connectivity of blood vessels through multiple slices. MIPs taken over the same slices from both spiral-in and spiral-out were compared. Examples of these MIP images can be seen in Figure 14. A zoomed in view of identical areas is presented in Figure 15 for closer inspection of the blood vessels.

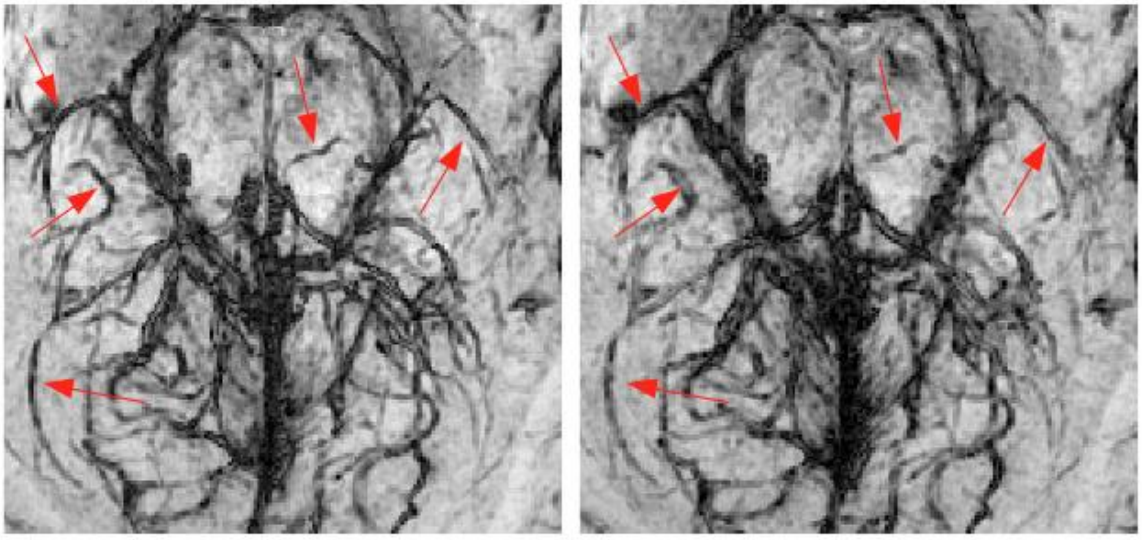

Figure 15: Magnified view of MIPs shown in Figure 29. Arrows indicate regions of better vascular acuity in a) spiral-in scans versus b) spiral-out.

To visualize the signal recovery seen in the spiral-in scans two identical Regions-Of-Interest (ROIs) were drawn on images with large susceptibility artifacts where significant signal loss has taken place. As predicted, more signal was present in the sinus and ear canal regions of the spiral-in acquisitions than the spiral-out. An example of this comparison can be seen in Figure 16.

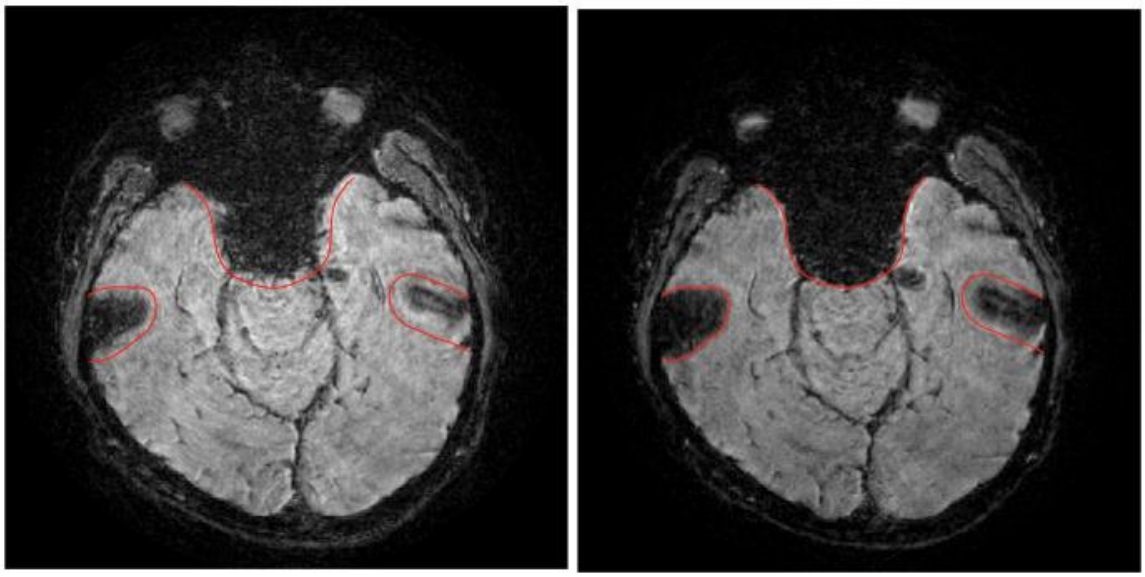

Figure 16: Magnitude images showing susceptibility induced signal decay in both a) spiral-in and b) spiral-out scans. ROIs outline identical regions in both images. Some signal recovery is evident in the spiral-in ROIs.

While spiral-in shows advantages in the magnitude images for signal recovery and vascular detail an analysis of the phase images shows advantages for the spiral-out trajectory. Magnitude imaging is T2*-dependent. The longer the TE or the later after the RF pulse, the more strongly T2* decay affects the image. Magnitude images benefit from quick echo times. To maximize the contrast of a phase image, however, we must allow sufficient time for tissue de-phasing to occur. The higher spatial resolution components of the spiral-out are acquired at a later TE than our spiral-in scans. This gives spiral-out an advantage in exhibiting fine vascular detail in the phase images. It is tempting to keep increasing TE to obtain stronger contrast in phase imaging. The benefit is negated for echo times which are too long by the total de-phasing of the protons. As the TE becomes longer, phase wrapping is exacerbated. Eventually the excessive phase wrapping will foil attempts to high-pass filter them out and obtain a useable image. A balance must be maintained between these criteria.

CHAPTER 4

THE RECONSTRUCTION PROBLEM

4.1 Regridding

Images must be reconstructed by applying an inverse Fourier transform-type operation on the scanned data. Data acquired with non-Cartesian k-space trajectories, such as spirals, can no longer be directly reconstructed into image space by an inverse FFT (Fast Fourier Transform). Non-equispaced sampling schemes must employ a variety of other techniques for image reconstruction.

The most widely used of these techniques is called gridding. Conceptually, gridding refers to the process of resampling the non-Cartesian data onto a Cartesian grid. Once this is done a standard inverse FFT can very quickly and easily reconstruct the image. Unfortunately, the speed and ease at which this can be done comes at a cost. In the process of resampling errors are introduced. An interpolation of data from their present positions to new ones along Cartesian coordinates cannot be perfect. It has been shown in (11) that the optimal gridding operation is convolution with a sinc function before Cartesian resampling. The impracticality of representing a sinc function to an infinite extent requires a more reasonable convolution kernel. In (12), the authors present results from convolving the data with a Kaiser-Bessel windowing function (13) instead. While the Kaiser-Bessel function has a strong central lobe it is accompanied by many smaller sidelobes which can degrade the image. Furthermore, the central lobe exhibits rolloff on its edges which will cause attenuation of the object's edges in the final image. Figure 17 displays this effect. A rolloff correction can be applied to flatten out the central

lobe's profile but this causes the relative strength of the sidelobes to increase. The sidelobes will be a source of aliasing in the reconstructed image. A proposed solution to this problem is to extend the FOV. This has the effect of extending the effective size of the central lobe to encompass the object while aliasing from the sidelobes will be outside our actual region of interest. However, to increase the FOV requires decreasing k-space sample spacings, Δkx and Δky, which lengthens scan time. Ideally, we would like to avoid the need to resample onto a Cartesian grid and its inherent difficulties.

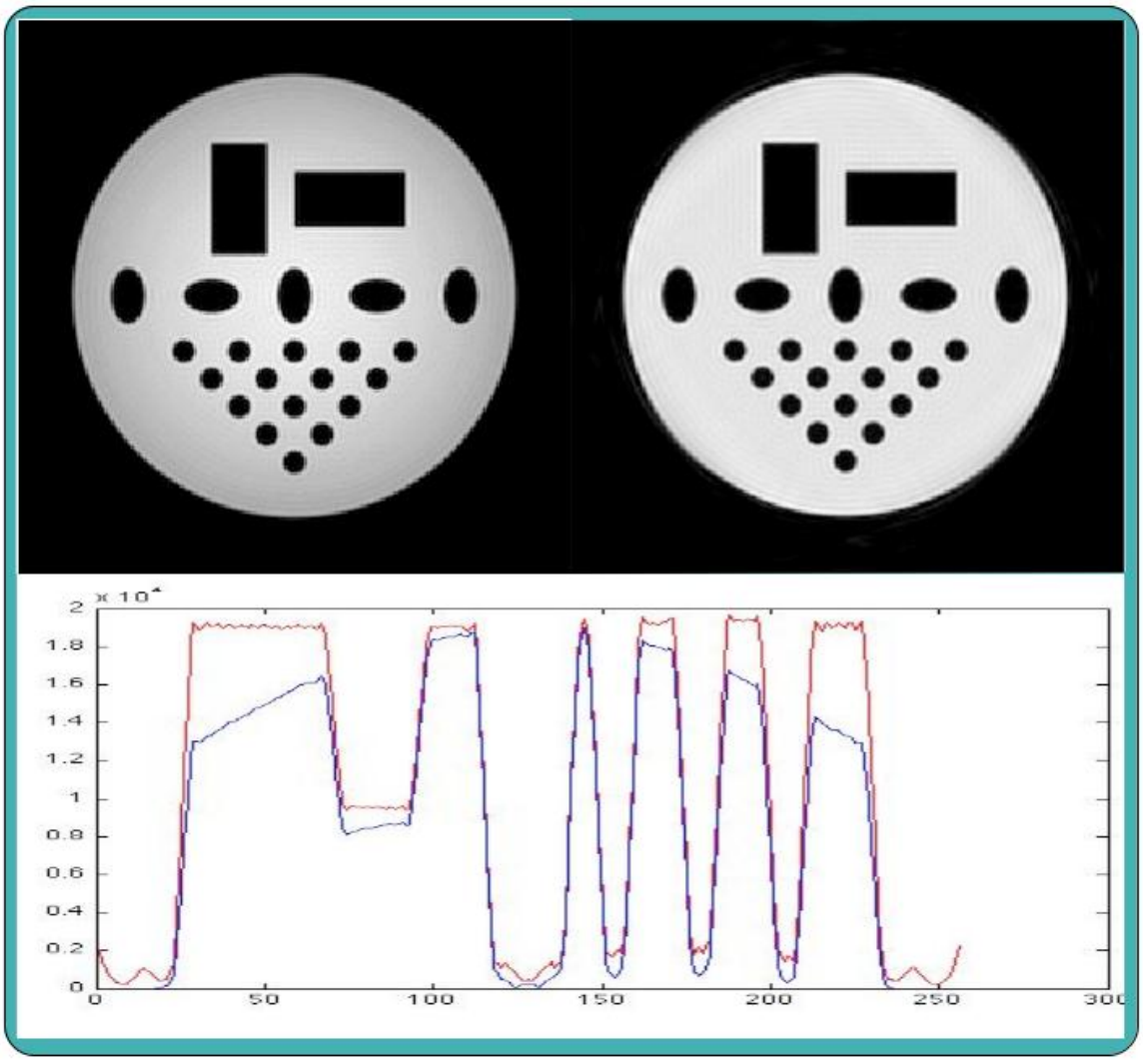

Figure 17: a) gridded reconstruction with attenuation of object edges. b) CGLS reconstruction with no edge attenuation. c) overlaid crossection of object magnitude for a (blue), b (red).

4.2 The Conjugate-Gradient Least-Squares Approach

The conjugate-gradient least-squares (CGLS) approach to the image reconstruction problem is essentially an inverse linear system optimization problem. Eqn. 1 describes the relationship between the acquired signal data and the image to be reconstructed.

$$S_N(k(t)) = \int_{FOV} C_N(\vec{r})\hat{f}(\vec{r})\, e^{-i[\vec{r}\cdot k(t)+\omega(\vec{r})]} d\vec{r} \qquad [1]$$

Here, $S_N(k(t))$ denotes the signal measured at each receiver coil, *N*, as a function of time. The vector $\vec{r}$ contains each spatial coordinate in the final reconstructed image. The variable $\hat{f}$ represents the complex values of the resultant image for which we are trying to solve. $C_N(\vec{r})$ contains the coil sensitivity values based on spatial location. The k-space trajectory is seen in *k(t)*. The integral is taken over the entire FOV.

A discrete version of Equation. 1 can be written as a simple linear system of the form

$$\boldsymbol{K}\hat{f} = S \qquad [2]$$

The matrix, ***K,*** contains the complex exponential Fourier transform components defined by the FOV spatial coordinates and the k-space trajectory sample locations. Also within the encoding matrix are the coil sensitivities and field map information. In the linear system the Fourier transformation has already been applied to the scanned object producing the time-dependent signal measured at the coils. The ***K*** matrix is almost never directly invertible. In order to resolve the image we apply the conjugate-gradient least-squares solver. We now wish to perform the following minimization:

$$\min_{\hat{f}} \left\| \boldsymbol{K}\hat{f} - S \right\|_2^2 \quad [3]$$

An iterative approach is used to resolve the image incrementally closer to the actual image. Such an algorithm was implemented on CUDA-enabled GPUs and tested for execution speed and image quality verses a CPU-based implementation. The exact algorithm followed for both can be seen in Appendix A. This algorithm requires the calculation of each element in the ***K*** matrix, which can be quite large. A computationally efficient method for populating these values is required for any implementation to be of practical use. The architecture of Graphics Processing Units (GPUs) has proven very beneficial for such operations.

CHAPTER 5

GPU ARCHITECTURE

5.1 Design Philosophy

The philosophy behind the GPU architecture has been massive parallelization. While traditional pipelined CPUs operate on only a thread at a time at different pipeline stages, GPUs are capable of performing operations on many threads simultaneously. This is ideal for calculating results which are each independent. The GPU design was born from the need for faster graphics calculations involving many matrix and vector operations. These devices require fast memory accesses, good availability of floating-point ALUs, and minimal synchronization. Such devices have already been shown effective in accelerating tailored RF pulse design (14) and (15).

High memory bandwidth is essential for GPU applications as all input data and output results must be passed between the GPU and the host system and be able to accommodate results from the massive parallelism of the GPU. This often means writing to many sequential memory addresses. Therefore, an emphasis is put on one- and two-dimensional memory locality for optimal speed. The GPUs themselves contain several levels of memory. Each have their own advantages and must be used properly to avoid potential bottlenecks as described in Section 5.2.

GPUs contain many identical units known as stream processors. These processors are designed to operate in parallel and yield high through put of arithmetic instructions. The management of these resources is typically left to the GPU device itself. The GPUs operate in tandem with their host CPU machines which largely control the logical flow of a given algorithm.

5.2 GPU Memory Types

5.2.1 Global Memory

The GPU's global memory is the largest available on the device. The host machine will perform memory copies into this area of the GPU before any further processing occurs. These take place in 32-, 64,-, or 128-byte memory transactions. This storage area is typically implemented with DRAM. Unfortunately, the tradeoff for large storage capacity is slower bandwidth and long latencies. Accesses to global memory must be minimized in order to avoid creating bottlenecks in execution. All GPUs used in this study had 512MB of global memory available on-chip.

5.2.2 Constant Memory

The constant memory also physically resides within the global memory but is also loading into a cache to speed read accesses. In order to achieve fetch acceleration, the constant cache is read-only during kernel executions. If a kernel requests a value not currently in the cache, a miss occurs and necessitates a read from the slower global memory. Constant memory accesses will usually be faster, and never worse than, global memory accesses. Constant memory can cache 64KB of data.

5.2.3 Texture Memory

Texture memory is also a kind of cache of global memory space. Usage of this area is ideal when repeatedly accessing data with high one- or two-dimensional locality. Texture memory is optimized for broadcast to many stream processors. Memory addressing is performed outside the kernel execution. The performance of this cache achieves a constant latency. Only when a cache miss occurs does the access slow to

global memory performance. Texture memory can contain 2^{27} bytes which is approximately 128 MB.

5.2.4 Shared Memory

Shared Memory resides within the stream processors themselves. If care is taken to avoid access collisions, this memory, along with registers, is the fastest available storage on the GPU. Their spatial proximity to the processors allows shared memory to be faster than any other memory. The scope of shared memory only extends to the current threadblocks being executed. Once the threadblocks are done and the kernel finishes execution the shared memory is cleared. Maximizing the use of shared memory is ideal for reducing computation time but size limitations must be observed. Only 8KB of shared memory was available on the GPU devices in this study.

5.2.5 Registers

Registers are also only accessible within their threadblocks. These have similar latency to shared memory but are typically used for storing temporary values. It is common for the compiler to determine the exact use of registers as needed in execution but they can also be specified by code. Registers are relatively few in number and their use must be kept to a minimum. If too many registers are demanded by a kernel it may force a reduction in warp size, the number of threads that can be executed in parallel. Overuse of registers can also force the compiler to use the slower global memory as a stand-in. This causes a dramatic reduction in performance and should be avoided.

5.3 Computational Units

The GPU devices used in this study were NVIDIA Tesla C1060s, each GPU

device used in this study was of compute capability 1.3. This refers to the architectural version of the NVIDIA hardware. This hardware includes special computational cores and functional units to carry out specific common tasks. These units include 8 CUDA cores used for both integer and single-precision (SP) floating-point arithmetic along with one double-precision floating-point unit. Two special floating-point units (SFUs) are present in the architecture and are useful for calculating the results of certain functions such as trigonometric or exponential operations. A warp scheduler is available to control the execution of the warps across numerous multiprocessors. Each of these devices contains 30 multiprocessors and 240 CUDA cores, (16).

5.4 The Test System

All CPU tests were performed on a 2.66 GHz quad-core Intel Xeon machine with 3GB of system RAM running the Mac OS X 10.6 operating system. Experiments for performing reconstructions using GPU device cards were performed on a Linux PC with a 2.66 GHz quad-core Xeon central processor with 32GB DRAM. Through the PCI express 2.0 slots, the machine was installed with four NVIDIA Tesla C1060 GPU devices. Each of the Tesla devices contains 240 CUDA cores, running on a 1.3 GHz clock with 4 GB of GDDR3 on-card memory with a memory bandwidth of 102 GB/sec. Both the Xeon CPU and Tesla C1060 graphics cards are capable of IEEE-754 Standard for single-precision (SP) floating point format. The Tesla C1060 also includes capability for double-precision (DP) floating point format as well.

CHAPTER 6

CGLS IMPLEMENTATION ON GPU

6.1 Implementation Philosophy

6.1.1 Issues to be Addressed

A few guiding principles were established to ensure the fastest reconstruction and quality of results. The matrix-vector multiplications are the most computationally intensive instructions in the CGLS algorithm. This is especially true as k-space trajectory readouts become longer or a higher spatial resolution is required. These two parameters dictate the size of the encoding matrix, $\boldsymbol{K}$. If we wish to reconstruct to an image matrix of size *MxM* (single slice) pixels with a readout length of *T* samples and employ *N* number of coils the resultant encoding matrix is of size *4xMxMxTxN* elements. The 4x multiplier is necessary because of the need to split the matrix into both real and imaginary parts in the following manner:

$$\begin{pmatrix} \boldsymbol{K}_R & -\boldsymbol{K}_I \\ \boldsymbol{K}_I & \boldsymbol{K}_R \end{pmatrix} \quad [4]$$

Each of these elements is stored as a SP float and therefore occupies four times the number of bytes as elements. Clearly for high resolution MRI scans with long readouts it is vitally important to calculate the elements of the encoding matrix in the most efficient manner possible.

Explicitly storing the $\boldsymbol{K}$ matrix is also out of the question. MRI datasets of even modest resolutions and readouts quickly overwhelm the GPU's global memory space. Even if sufficient memory was available, the matrix is used twice per CGLS iteration, necessitating a great deal of fetching from global memory which introduces a very

constrictive bottleneck. The solution to these issues is to avoid explicitly storing the encoding matrix itself but instead to calculate matrix elements on an as-needed basis.

The image quality that results from GPU computation must also be carefully tested against traditional reconstruction techniques. Because of the asynchronous nature of threadblock scheduling in the GPU, long floating-point summations are not guaranteed to yield the same results as those performed in a deterministic order on a CPU. It is possible to structure the algorithm such that threadblocks are executed in a consistent order on the GPU but this would be detrimental to the speed advantages from massive parallelism. This issue is discussed in further detail in Section 5.2.1.

Another question that needs to be addressed is whether or not SP floating-point calculations are sufficient for the purpose of the algorithm. While it's true the algorithm could simply be extended to operate with double-precision (DP) floating-point representation, the limited availability of DP calculation units present a bottleneck for execution. As seen in section 5.3, the number of SP units available is 8 verses only 1 DP unit. This introduces congestion for these resources when using DP. It is shown in section 7.2.1 that with proper care potential limitations of SP can be avoided without resorting to DP and drastically slowing down reconstruction. The latest generation of GPU devices with compute capability 2.0 has additional DP units which should help to alleviate this limitation.

6.1.2 Algorithm Design Requirements

MR image reconstruction often requires very large encoding matrices as well as transposed versions for each iteration of CGLS execution. The matrix can quickly overwhelm all available device memory if stored explicitly. It is therefore necessary to

calculate elements of the matrix on an as-needed basis. It is very important that these values be calculated quickly so that the algorithm can finish in a reasonable amount of time. As seen below, six principles were proposed as design goals in order to streamline the algorithm.

1) Use fastest available memory
2) Reuse calculations as much as possible
3) Ensure all kernel inputs are cached
4) Use fast Special Function Units
5) Maximize parallelism and occupancy
6) Optimize blocksize

The fastest available memory within the GPU is the shared memory, described in Section 5.2.4. This memory was used to calculate and store the elements of the encoding matrix as needed in the CGLS algorithm. Using this level of memory minimizes accesses to global memory which would cause a couple orders of magnitude slowdown in execution. Since this memory is cleared when a threadblock has completed execution no extra code was needed to prepare the shared memory for use again.

Ideally one can reuse elements of the $\boldsymbol{K}$ matrix once they have been calculated in a particular sub-block. It can be observed from Equation 4 that the upper left quadrant of the matrix and the lower right quadrant are identical. Therefore the algorithm implements the calculation of only the upper left quadrant values but multiplies these values with both the real and imaginary parts of the data. A similar observation is made for the lower left quadrant and the upper right quadrant, differing only by sign. These symmetries allow us to only calculate half of the elements of the full matrix. This

provides a very large advantage.

The third design principle is to cache all kernel inputs. Size limitations in cached memories force careful use of these tools. Vectors of long k-space trajectories or high spatial resolutions must be stored in the slower global memory. When a CUDA kernel is about to execute the sections of the k-space trajectory and the spatial coordinates necessary for that particular section of the encoding matrix are cached into the texture memory, described in Section 5.2.3. The memory reads within the kernel itself are all to the cached texture memory to feed the inputs to the encoding matrix element calculations. These accesses are quite fast and are in accordance with the strong spatial localities which the texture memory fetches are optimized for. This type of memory is the fastest available which can accommodate the necessary data sizes. Binding the texture cache to global memory locations just before kernel execution minimizes cache misses which would drastically hurt speed performance.

Part of the CUDA architecture is the SFU (Special Function Unit). These are specialized ALUs with built-in SP floating-point functions such a trigonometric and exponential functions. These units operate more quickly than the typical arithmetic functions by sacrificing some accuracy. Truncated Taylor series are used in the case of sine and cosine functions used in the CGLS implementation. These functions are well-behaved and no limitations are necessary to restrict their outputs to acceptable ranges. The qualitative and quantitative effect the reduction in numerical accuracy has on the final image reconstruction requires careful analysis.

Parallelism is the most important advantage offered by a GPU implementation. Maximizing the amount of parallel execution can have a tremendous effect on

performance. The GPUs used in this study are capable of executing 512 threads per multiprocessor simultaneously. This number may be decreased by overuse of registers and shared memory. A balance must be struck between using the ultra-fast shared memory and registers against maximizing the amount of parallelism. The CUDA Occupancy Calculator (17) was used to assist in analyzing the appropriate allocation of these resources. Ideally 100% occupancy can be achieved without reducing register or shared memory use to a detrimental level.

The dimensions of threadblocks in the kernel execution are also important to speed. User code can vary the size of threadblocks while keeping the number of threads in a threadblock constant to maximize occupancy. The optimal size of the blocks is algorithm dependent and must be experimentally determined. The performance variations of different sizes are largely dictated by the spatial locality of the memory accesses in the kernel. Memory collisions caused by non-optimal block sizes force serialized access and cause large delays.

Some of the simple matrix-vector operations necessary for the CGLS algorithm are already freely available in code libraries. For this program the freely available CUBLAS (CUDA Basic Linear Algebra Subprograms) were used to implement vector-vector dot product operations as well as vector copies from one variable to another for each iteration.

The GPU implementation of the CGLS algorithm presented here employs all of the techniques described above. The performance impact of these optimization principles is analyzed in the Chapter 7.

6.1.3 The CPU Implementation

Many of the optimizations described in Section 6.1.2 are not available in a CPU implementation of the same algorithm. This is simply due to the serialized nature of the CPU execution pipeline. The operational flow of the CPU code mirrors that of the GPU implementation as much as possible. CUBLAS functions available to the GPU execution are replaced with CBLAS (Complex Basic Linear Algebra Subprograms) library. All program inputs are read from command line parameters and data files in identical ways between the two versions. Close similarity of the code adaptations is necessary to properly analyze advantages and disadvantages of GPU or CPU execution.

6.2 Reading Inputs

6.2.1 The RAW file

Code to read in all data from disk drives was identical in both CPU and GPU implementations of the program. Raw data files from the scanner were used as data input to the algorithm. These files have no preprocessing and are exactly as copied from the scanner computer. The first 48 bytes of the file are dedicated to scan parameter values which are used to parse the rest of the data. This header is followed by the measured data from the MRI signal acquired during the scan in SP floating-point representation.

The header contains information about the arrangement of data in the file these include the number of receiver coils, slices, spiral interleaves, and k-space samples per interleave as well as gradient levels and slew rates. Image domain parameters such as FOV and an appropriate image matrix size are given. All of these values are read into variables with global scope to be used throughout the image reconstruction process.

The scan data must now be read in and careful parsing must take place to arrange the data into the most appropriate ordering to use in the CGLS algorithm.

A rearrangement of the raw data as it is read in is motivated by a desire to simplify array indexing which would otherwise be quite complicated. This is especially important within the kernel executions. Thread IDs are used inside the CUDA kernels as indexes for k-space and spatial location arrays. By arranging the raw data into a favorable organization up front the indexes can be more straightforward and prevent extra indexing calculations inside CUDA kernels which would need to execute for every thread.

6.2.2 The SENSE file

Coil sensitivities are also included in an input file given to the CGLS code. The SENSE file contains image information to properly weight those parts of the image that were in closest proximity to that particular receiver coil during scan. This file is also organized such that indexing is simplified within the CUDA kernel. The SENSE coefficients from each coil are duplicated so that there are now both real and imaginary versions of the coil information. This requires slightly more global memory to store the vector which is now twice as long but saves overly complicated array indexing when applying the coefficients to the encoding matrix. This arrangement is performed only once rather than repeated index calculations for each thread during parallel execution. All data from the SENSE file are read in as SP floats. Figure 18 displays the on-disk SENSE file data arrangement and the duplication that takes place. The SENSE data can now be applied by point-wise multiplication when operating in the spatial image domain.

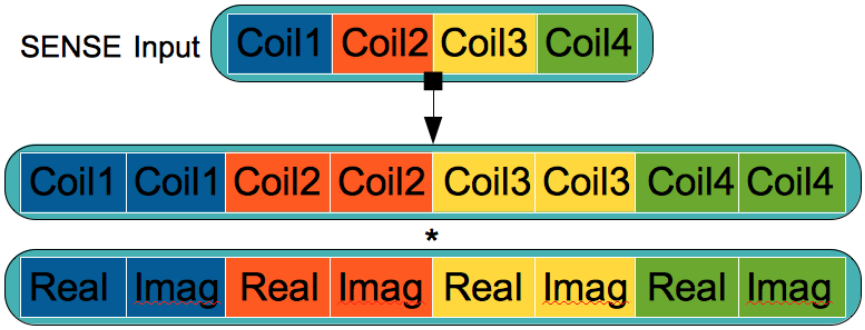

Figure 18: Illustration of SENSE input rearrangement for optimal implementation.

6.2.3 The field map file

The field map input file is simply a vector representation of the off-resonance frequencies calculated using the method described in Section 1.4.3. These measurements are independent of the number of coils so the same correction values were applied to each coil. These values are stored in global memory and then loaded into texture cache for fast retrieval.

6.3 Generating K-space, spatial coordinates, and T2 Decay

6.3.1 Spatial coordinate calculations

The two domains needed for the population of the encoding matrix are the k-space trajectory and the image-domain coordinates to be reconstructed. The image coordinates are derived directly from the desired reconstruction resolution and the FOV. The x and y vectors to specify these locations will be identical and can be easily calculated by determining the image dimensions and equally dividing the FOV between these locations.

6.3.2 K-space trajectory calculations

The k-space sample locations are also calculated based on scan parameters which are read from the raw data file's header. The spiral trajectory code in the reconstruction

code matches code used in the MRI scan computer. The code used in the GPU reconstruction algorithm is an adaptation of the spiral trajectory code from Douglas C. Noll of the University of Pittsburgh and John Pauly of Stanford University. Calculations to create the k_x and k_y vectors are based on scan parameters such as the number of spiral interleaves, the desired FOV, the sampling frequency, and maximum gradient slew rates. These calculations are done only once during the algorithm and are completed very quickly.

6.3.3 T2* decay vector

A vector to quantify the T2* decay must be calculated for the encoding matrix as well. This vector takes into account the relative amount of signal decay that has occurred throughout a single ADC readout. The T2* decay effect is addressed in more detail in chapter 2. Accounting for T2* attenuation properly weights the sampled data along a readout. This becomes especially important with long readouts. Knowing the k-space sampling period, the echo time (TE), and the number of samples in each readout the vector is calculated and plotted in Figure 19.

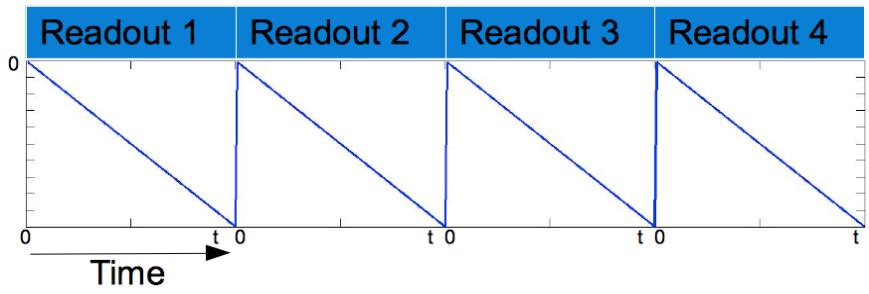

Figure 19: Plot of T2* decay vectors applied to encoding matrix

6.4 CGLS Calculations

All of the steps described above occur identically in the CPU and GPU

implementations of the reconstruction code. At this point there is some divergence as the GPU version begins to take a multithreaded approach to the CGLS calculations. Copies of the data described in Sections 6.2 and 6.3 are moved into global memory on the GPU device. This is a straightforward memory copy and requires negligible time in the algorithm.

In accordance with the design principles laid out in Section 6.1.2, the vectors used to populate the elements of the encoding matrix are cached into texture memory. Texture memory is the fastest available storage for data of this size. The sizes of the thread- and gridblocks needed to perform the matrix-vector multiplication step, $d = K^H m$, are now determined. The Tesla C1060 GPU used in this study allows threadblocks of up to 512 total threads. Grid sizes are limited to 65,535 blocks in either the first or second grid dimension. The grid limitation can present a problem when reconstructing very large datasets.

Two approaches can be taken when constrained by the grid size limitation. First, the threadblock sizes can be increased in that dimension if doing so does not create a problem along the other dimension. This solution, however, can have profound performance implications when the threadblock dimensions are not optimal. The second approach is to simply divide the problem into manageable grid sizes and execute the kernel for as many grids are necessary. In this case, two sets of grid dimensions are created. One set will be maximized along the problematic dimension. The other set will be scaled back along the same dimension to take care of any residual threads not calculated in the maximum sized grids. This creates some additional overhead but is not as drastic a performance cost as altering the threadblock dimensions.

Threadblock sizes were chosen to maximize multiprocessor occupancy and allow the calculations to be performed in shortest time possible. All threadblocks were set to execute 512 threads. This is a convenient value since data in the k-space spirals is acquired in 1024 sample interleaved chunks and makes sizing easy. The dimensions of the blocks were determined experimentally.

6.4.1 Multiplying along k-space

The first matrix-vector operation to be performed for the CGLS solver requires the calculation of the inner product of the encoding matrix, ***K***, against the raw k-space sampled scan data. This is the first step of the algorithm which will make extensive use of parallel execution on the GPU. A kernel is launched to perform the multiplication including the population of the elements of ***K***. The inputs of this kernel are the spatial and k-space trajectory vectors as well as the SENSE, field map, and T2 decay vectors all previously cached into the texture memory.

A block of shared memory identical to the size and dimensions of the threadblock is allocated to store the elements of ***K*** needed for that particular block, part of the overall matrix. The corresponding section of raw scan data required is also copied into another portion of shared memory. Shared memory is the fastest available and will be cleared when the block's execution is completed.

The elements of ***K*** are now calculated without the trigonometric functions required to split the values into real and imaginary parts. This allows for the reuse of these elements for either situation. Included in this calculation are the field map and T2 decay terms. The SENSE component has been factored out to be applied after the matrix-vector multiplication is complete. A point-wise multiplication can quickly apply

the SENSE coefficients to the matrix-vector product. This greatly reduces the number of floating-point operations necessary. Figure 20 shows how the multiplication takes place.

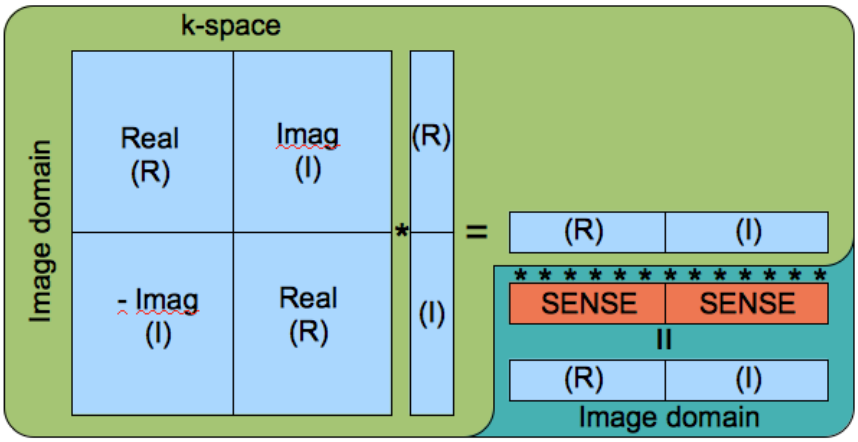

Figure 20: Illustration of the matrix-vector multiplication along k-space with point-wise multiplication with SENSE

In order to form the real and imaginary components of the matrix trigonometric functions are necessary. For the real components a simple cosine function is needed. In the imaginary components a sine function is required and the sign changes as well. These functions are implemented in the SFUs of the GPU as truncated Taylor series. A shorter truncation than normal allows for faster calculation at the expense of numerical accuracy. The limited availability of SFUs for DP floating point is a principle reason for remaining in SP floating-point representation.

The fast trigonometric functions are applied to the real and imaginary components then these two pieces and partial summations already computed across the row are added. This step requires a write to global memory. This is performed for both real and imaginary portions of the product. Several syncthreads() commands are needed in the kernel to keep threads synchronized for each step. The kernel code described here is seen in Figure 21.

```
__global__ void MultKSpace(float *d_in, float *d_out, int vec_length, int nxy, int offset) {
  int ix=blockIdx.x*blockDim.x+threadIdx.x;
  int iy=blockIdx.y*blockDim.y+threadIdx.y;

  int row = threadIdx.x;
  int col = threadIdx.y;

  __shared__ float s_Avec[BLOCKSIZE2][BLOCKSIZE];
  __shared__ float s_m[BLOCKSIZE][2];

  s_m[col][0] = tex1Dfetch(t_in,iy+offset);
  s_m[col][1] = tex1Dfetch(t_in,iy+offset + vec_length/2);
  __syncthreads();

  s_Avec[row][col] = (tex1Dfetch(t_kx,iy+offset)*tex1Dfetch(t_x,ix)) +
                     (tex1Dfetch(t_ky,iy+offset)*tex1Dfetch(t_y,ix)) +
                     (tex1Dfetch(t_fldmap,ix)*tex1Dfetch(t_samptime,iy+offset));
  __syncthreads();
  for (short int e=0; e < BLOCKSIZE; e++) {
    d_out[ix] += __cosf(s_Avec[row][e])*s_m[e][0] +
                 __sinf(s_Avec[row][e])*s_m[e][1];
    __syncthreads();
    d_out[ix + nxy*nxy] += -__sinf(s_Avec[row][e])*s_m[e][0] +
                           __cosf(s_Avec[row][e])*s_m[e][1];
    __syncthreads();
  }
}
```

Figure 21: CUDA kernel code implemented to perform matrix-vector multiplication along k-space.

A separate kernel is used to perform the point-wise multiplication to apply the SENSE coefficients to the image-domain result. The non-SENSE product is pre-cached into the texture memory for the multiplication to take place in the quickest way possible.

6.4.2 Multiplying along image space

The multiplication along image space is very similar to the multiplication that takes place along k-space. The difference here is mainly how the SENSE information is incorporated into the encoding matrix. Multiplying along the image-domain coordinates will yield a representation of the data in k-space. The SENSE coefficients can no longer simply be point-wise multiplied to the result because they are an image-domain

representation. The values must now be incorporated directly into the matrix-vector multiplication. As in Section 6.4.1, the k-space, image-domain coordinates, SENSE and field map values along with the now image-domain transformed scan data are cached into the texture memory.

The threadblock size is optimized again for best speed performance and the kernel is launched. Here again it may be necessary to employ multiple grids if any grid dimensions are too large for the GPU. A block of shared memory is allocated to the storage of the ***K*** matrix elements. The transformed scan data is also read into shared memory. As before, the ***K*** values are calculated with the exception of applying the trigonometric functions. The fast trigonometric functions are again applied as needed to account for the real and imaginary components along with the appropriate sign changes. The sine and cosine are again implemented with the SFUs. After the trigonometric functions are applied the SENSE coefficients are multiplied to the result. Both the real and imaginary portions as well as any partial sums previously calculated are summed and written to global memory. The layout of these operations is displayed in Figure 22.

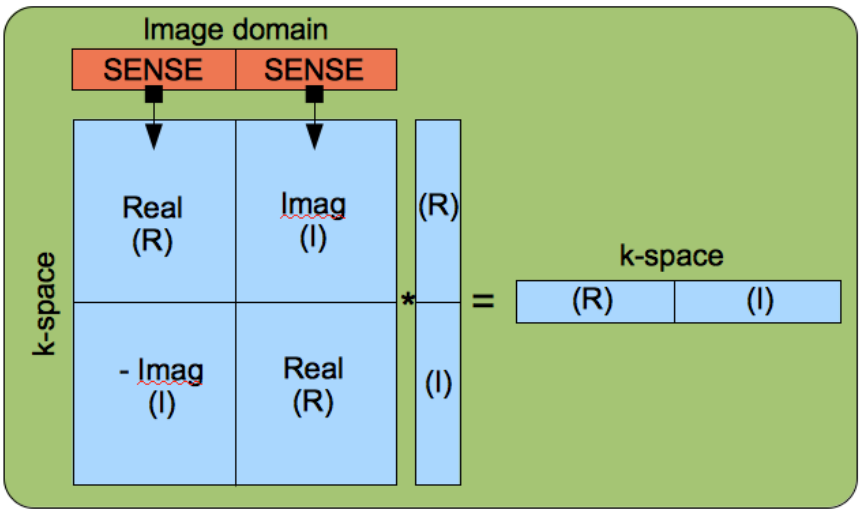

Figure 22: Illustration of the matrix-vector multiplication along the spatial domain with SENSE incorporated into the encoding matrix.

Again syncthreads() commands are necessary to avoid creating memory race conditions. The kernel code governing these calculations is given below in Figure 23.

```
__global__ void MultXSpace(float *d_d, float *d_p, int vec_length, int nxy, int offset) {
  int ix=blockIdx.x*blockDim.x+threadIdx.x;
  int iy=blockIdx.y*blockDim.y+threadIdx.y;

  int row = threadIdx.x;
  int col = threadIdx.y;

  __shared__ float s_Avec[BLOCKSIZE2][BLOCKSIZE];
  __shared__ float s_d[BLOCKSIZE][2];
  __syncthreads();

  s_d[col][0] = tex1Dfetch(t_in2,iy+offset);
  s_d[col][1] = tex1Dfetch(t_in2,iy+offset + nxy*nxy);

   s_Avec[row][col] = (tex1Dfetch(t_kx,ix)*tex1Dfetch(t_x,iy+offset)) +
                      (tex1Dfetch(t_ky,ix)*tex1Dfetch(t_y,iy+offset)) +
                       (tex1Dfetch(t_fldmap,iy+offset)*tex1Dfetch(t_samptime,ix));
   __syncthreads();
   for (short int e=0; e < BLOCKSIZE; e++) {
     d_p[ix] += tex1Dfetch(t_sense,iy+offset) *
                  (__cosf(s_Avec[row][e])*s_d[e][0] -
                   __sinf(s_Avec[row][e])*s_d[e][1]);
     __syncthreads();
     d_p[ix + vec_length/2] += tex1Dfetch(t_sense,iy+offset) *
                                 (__sinf(s_Avec[row][e])*s_d[e][0] +
                                  __cosf(s_Avec[row][e])*s_d[e][1]);
     __syncthreads();
   }
}
```

Figure 23: CUDA kernel code implementing matrix-vector multiplication along the spatial domain incorporating both SENSE and the field map.

6.4.3 Miscellaneous calculations

Other than the major matrix-vector calculations there are several more steps to complete in the CGLS algorithm. Several memory copies are required as well as calculating L2 norms. These are implemented by using CUBLAS library tools. The

CUBLAS library is freely available and gives the user access to common linear algebra functions in very streamlined implementations.

The steps described above are completed for each of the coils used in the scan. Originally, the attempt was made to implement a CGLS algorithm which would automatically combine the separate images from each coil into a single image within the conjugate-gradient itself. This approach yielded poor results. It was found that without phase unwrap filtering the images could not be easily combined. Each coil detects a different phase image. These images are similar but phase wrapping complicates their combination if no filtering is done. If the coils are directly combined without unwrapping the phase images, phase cancellations occur and yield an image full of artifacts. If the phase is unwrapped and then combined during the CGLS iterations, then the algorithm is attempting to converge to a wrapped version of the raw scan data. Incorporating the unwrap filter would require filtering of both the raw data itself before any other CGLS calculations are done. This would be costly in time and more complicated than simply applying the filter afterward. The two methods yield mathematically identical results. Allowing each coil to be reconstructed separately then combined in the last step is more straightforward, faster to compute, and more easily debugged for errors in particular coils.

6.4.4 Output

At the completion of each iteration the current solution is copied from the GPU's global memory to the host system's central RAM, then to the hard drive. This output file is in SP floating-point representation. Each coil is represented separately with the first half of the values for that coil being the real component while the second half contains

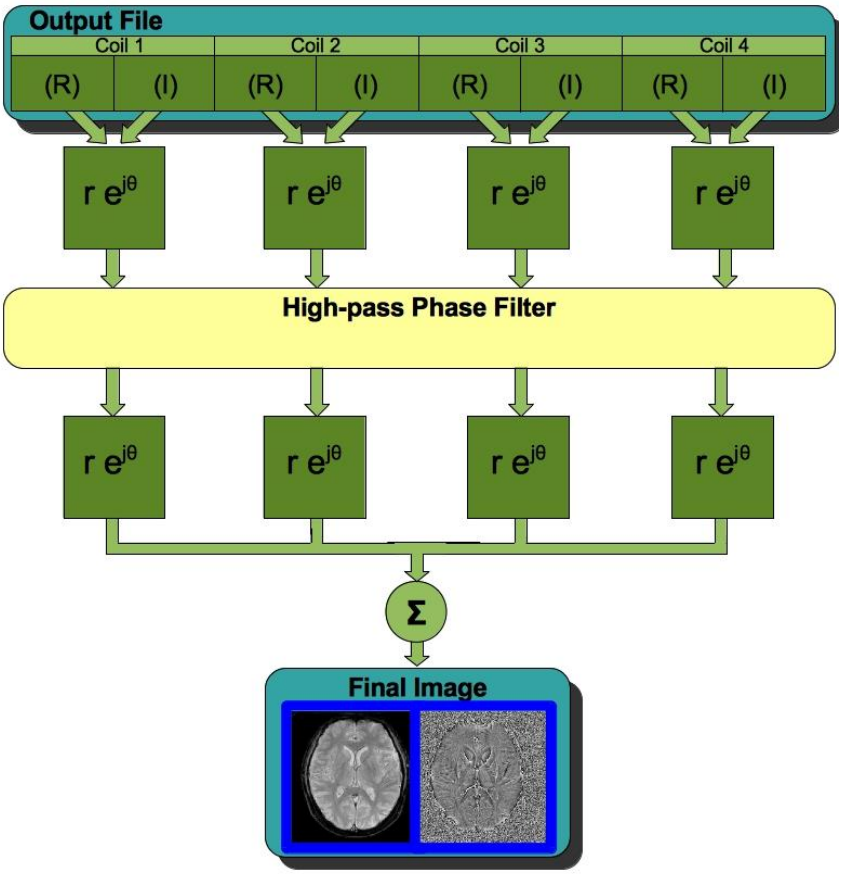

Figure 24: Flow of algorithm to combine data from multiple coils into a single output image.

the imaginary values. The solutions for each coil are concatenated as seen in figure 24.

In order to combine the coils into a single composite image the phase of each coil must be unwrapped. The real and imaginary components are first combined into their

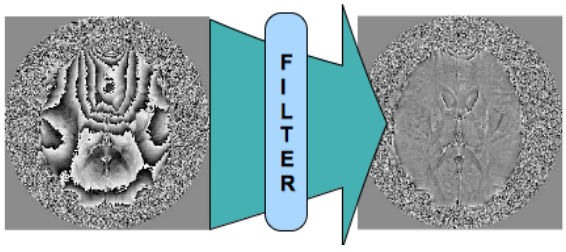

Figure 26: Demonstration of phase wrapping and final phase image after high-pass filtering.

magnitude/phase representation. A high-pass filter is applied to each coil's phase image to remove the low frequency phase wraps which obscure the underlying tissue-contrasted phase. A simple Fermi filter is employed to perform this operation. Both the wrapped and unwrapped images are demonstrated in Figure 25.

CHAPTER 7

RECONSTRUCTION RESULTS AND DISCUSSION

7.1 Testing

Image reconstruction tests were performed from a variety of datasets to evaluate the GPU implementation of the CGLS reconstruction algorithm. Simulated datasets were generated using MATLAB code to mimic scanner raw data output. These were used to test the code under conditions which may have not been practical in the scanner. The MATLAB code is capable of creating datasets of any number of coils with any image or k-space parameters desired. Noise of different types and strengths can also be applied to the simulated datasets. This allows us to evaluate the reconstruction under known conditions.

Actual scanner data was also reconstructed and analyzed. These datasets were acquired by sequences currently used in the lab's other research protocols. Brain scans were taken of human test subjects in accordance with IRB regulations. These scans are real-world test conditions with all of the associated confounding factors including physiological differences increasing B0 field inhomogeneity, T2 decay, coil SENSE, and signal noise. Parallel MRI scans were performed using 4 head coils.

7.2 Quantitative Analysis

The Normalized Root Mean-Squared Error (NRMSE) was used as a quantitative measure for how well the reconstructions have performed. Several simulated phantom objects were reconstructed with both the CPU and GPU implementations of the CGLS reconstruction algorithm. Despite having identical algorithms, reconstructions resulting

from the two approaches are not identical. The average NRMSE between CPU and GPU CGLS reconstructions on the same dataset was 0.07. The source of these variations appears to be two fold. First, the fast trigonometric functions used in the GPU algorithm sacrifice accuracy for speed. In our tests, the effects of this tradeoff were negligible as the calculation of the trigonometric functions was not a major bottleneck for speed. A slowdown of 2% was measured for the GPU algorithm when not using the fast trigonometric functions without improving the error observed earlier.

The second source of variation in reconstruction results arises from the nature of floating-point operations. In a CPU serial-pipeline environment the addition of values represented as floats takes place in a deterministic order, yielding a consistent result. The dynamic thread scheduling of the GPU architecture means that without explicit code to order the pairwise additions, which would incur additional latency, there is nothing guaranteeing that results from all threads are added in the same pairing. It is important to note that while the CPU and GPU solutions may differ slightly, neither one can be said to be more correct than the other. Additionally, the error bound of floating-point arithmetic on a set of values is independent of the data distribution (18) based on the assumptions of Wilkinson's standard model (19). Therefore, results from both the CPU and GPU implementation, while different, agree very closely. No visual distinction was observed.

7.2.1 Sufficiency of SP floating-point representation

As alluded to in section 6.1.1, it is appropriate to ask whether or not SP is sufficient for the needs of the CGLS algorithm. SP was chosen over DP based on consideration for data sizes and availability of computational resources in the GPU. Of particular concern are datasets with very long k-space trajectories or high-resolution

reconstructed images. Either of these situations necessitates long reduction operations when performing matrix-vector calculations which could cause overflow problems when using SP. This would occur when trying to represent a number whose value exceeds 10^{38}. Assuming a k-space trajectory with 1 million points (about 1/3 more than our longest scans), we can calculate that the average value required in a sum, such as occurs in the dot-product operations, to cause an overflow exception is on the order of 10^{32}. This is many orders of magnitude beyond anything in our scans or the algorithm. Therefore, single-precision floats are sufficient to avoid overflow issues and this issue has never been observed in actual scan data.

To test the algorithm to its fullest, a pathological simulated dataset was generated to force a SP overflow when performing the matrix-vector operations. This introduced NaN values into calculated results in the recursion, rendering the output images useless. However, including a simple scaling factor on the raw data as soon as it is read from the file allows the values to be scaled to avoid an overflow occurrence while preserving reconstructed image quality and contrast. SP is therefore sufficient for the algorithm.

This is advantageous as DP instructions in the GPU are significantly more computationally costly. SP functions require 28 bytes of local memory, doubles require 44 bytes (20). Executing instructions for each thread using single-precision requires 4 clock cycles, whereas double-precision requires 32. Also, for CUDA-enabled architectures of compute capability less than 2.0, each multiprocessor contains 2 single-precision functional units but only 1 DP functional unit. SP representation is the best choice because it requires less memory, fewer clock cycles, and makes better use of available functional units.

7.3 Image Quality

Evaluating the quality of the image reconstructions is a subjective comparison and does not easily lend itself to a direct process. Three criteria were established to judge the relative quality of the different reconstruction methods. These are:

1) Contrast

2) Blur

3) Noise

Image contrast must be strong enough to enable the differentiation of tissue types and blood vessels and to be able to discern anatomical structures. The resultant images must not be blurry but have sharply localized tissue boundaries and features. Noise must be rejected sufficiently to allow for a clear view of the underlying image.

7.3.1 Effectiveness of SENSE

The inclusion of coil sensitivities into the reconstruction improves SNR and adds some regularization to the linear inverse problem. This allows us to try to reconstruct images from undersampled k-space. As discussed in Section 1.4.2, when we undersample k-space we are effectively reducing the FOV and therefore introducing aliasing artifacts into our image. By including SENSE we are including additional weighting information in our calculations which can mitigate this aliasing.

To easily test the effectiveness of including the coil sensitivities we generated a k-space trajectory in the normal fashion then ignored every other interleave. This has the effect of undersampling k-space by a factor of 2. It is important to note that this is not the typical undersampling scheme. The usual approach to undersampling will still include a full or nearly full sampling of values near the k-space origin. This gives the benefit of

eliminating aliasing artifacts stemming from the lower spatial frequencies where these components would be stronger than those from the higher spatial frequencies. A comparison of reconstructions using no SENSE map and one employing the appropriate SENSE map are seen in Figure 26.

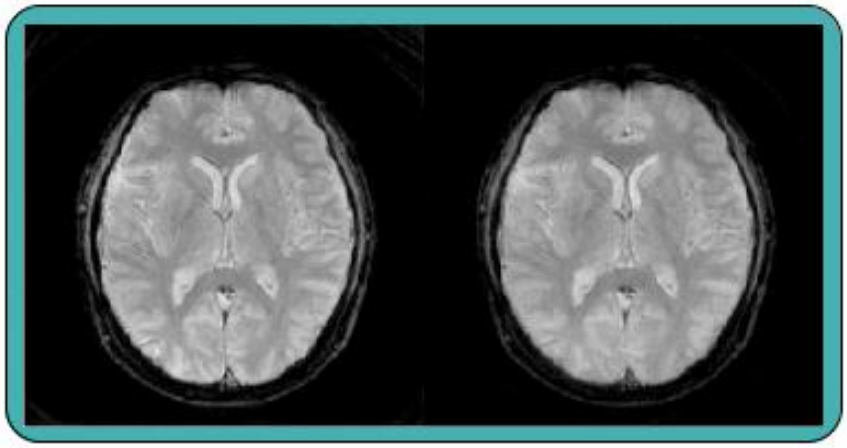

Figure 27: a) Comparison of fully sampled images reconstructed with SENSE and b) without SENSE. Note better tissue contrast with SENSE.

Applying a SENSE map to the 2x undersampled case we can still see some degradation of the image but many of the aliasing effects have been reduced. An example of a 2x undersampled image reconstruction using SENSE is seen in Figure 27. Notice that aliasing has been pushed mostly to the edges of the FOV. These aliasing effects could be further reduced by fully sampling the area around the k-space origin while undersampling the more outlying spatial frequencies.

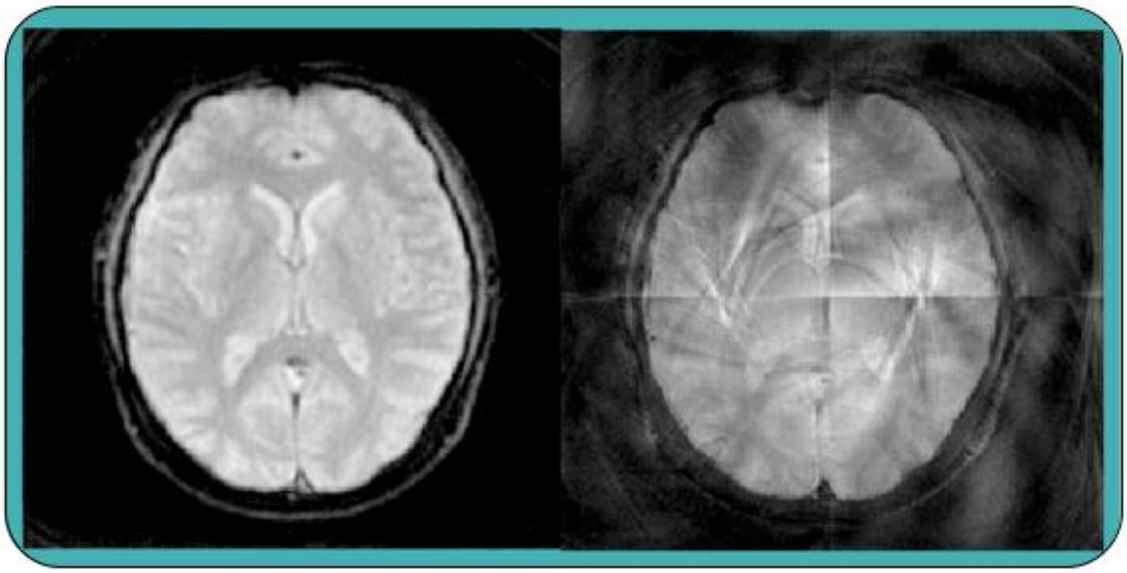

Figure 28: 2x undersampled reconstructing a) with SENSE and b) without SENSE. The undersampling scheme used here also undersampled the center of k-space introducing strong aliasing effects.

By reducing the number of data samples by a 2x factor this also reduces the calculation time by a factor of 2x as well.

Further undersampling is possible but the amount of aliasing increases accordingly. This diminishes the utility of the reconstructed images if the amount of undersampling is too great. However, an accurate SENSE map with information from several coils this problem can be mitigated to some extent. To test our implementation of the GPU-accelerated CGLS solver a fully sampled image was reconstructed using a four coil SENSE map. This was considered as our benchmark image against which we compared various undersampling rates both with and without SENSE maps. The NRMSE was calculated between our undersampled reconstructions and the fully sampled ideal. The results of these measurements are displayed in Figure 28. The tests show increasing NRMSE as the undersampling factor increases but the error is significantly less using an accurate multiple coil SENSE map. This demonstrates the advantage of

parallel MRI which can achieve higher SNR and allow undersampling. This is also an advantage of the CGLS algorithm over gridding techniques which cannot perform SENSE reconstruction with undersampling.

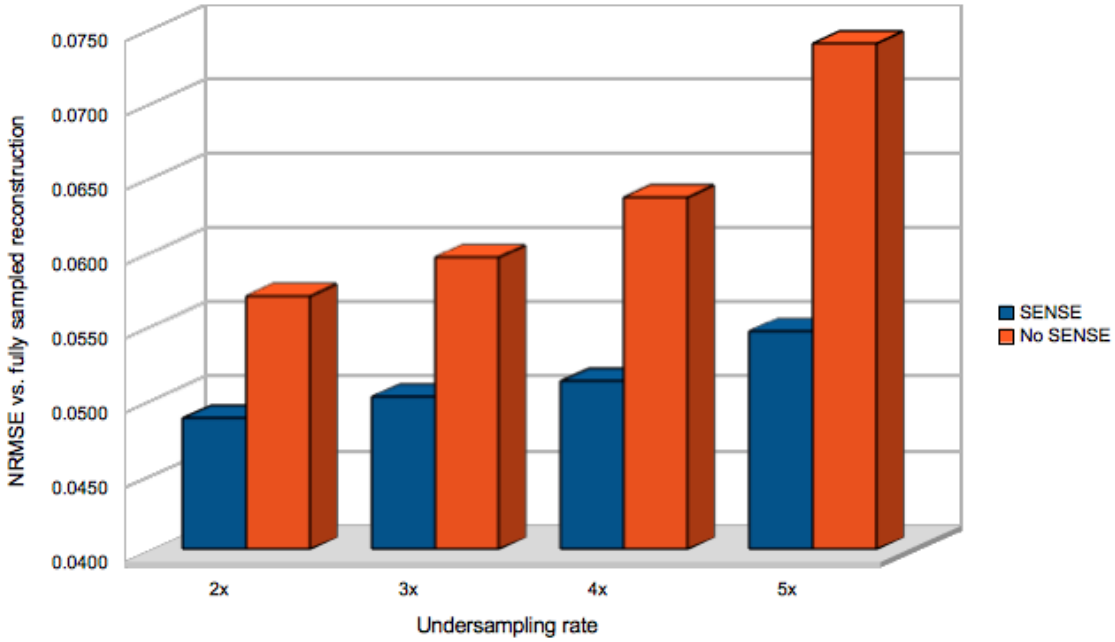

Figure 29: NRMSE measurements of 4 different undersampling rates with SENSE (blue) and without SENSE (orange).

7.3.2 Effectiveness of the field map

To accurately reconstruct an MRI image we must also take into account inhomogeneity in the B0 field. These off-resonance effects occur near air/tissue boundaries. This is most frequently seen in areas above the sinus cavities and around the ear canals in brain scans. So far in the reconstructions we have assumed that the underlying B0 field is homogeneous. Because of air/tissue boundary susceptibility off-resonances this is a poor assumption. If this effect is not taken into account areas with strong inhomogeneity are somewhat blurred in the image reconstructions. We can see the effect of the field map represented as ω in Equation 1.

A common method for measuring the field map is to take two scans with a small ΔTE between them. This allows a phase difference to accrue between the scans. In a completely homogeneous case with no off-resonance the difference should be the same everywhere. In the presence of inhomogeneity the phase difference, ΔΦ, is increased. The images in Figure 29 show the reconstruction without a field map, the calculated field map itself, and the reconstruction applying that field map.

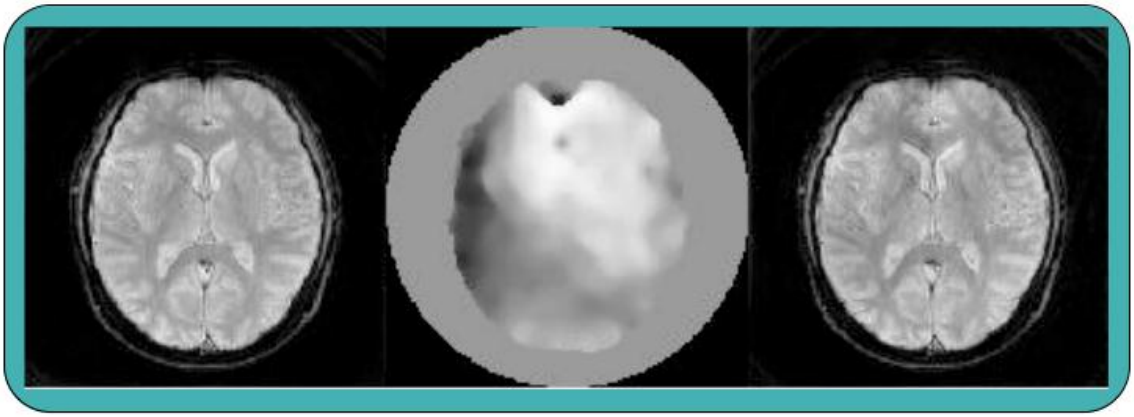

Figure 30: a) image reconstruction with field map applied b) calculated field map c) image reconstruction with no field map applied.

Notice the images are still very similar but some signal recovery has occurred in the frontal area above the sinus.

7.3.3 Noise rejection

With any real-world MRI scan noise is present. The two CGLS algorithms were tested against each other to ensure the GPU version is as effective as the CPU approach in rejecting noise as well as to demonstrate the ability of the CGLS algorithm itself. Simulated phantom datasets were generated which allow us to control noise levels for testing. Each dataset was tested on both the GPU and GPU implementations of the CGLS algorithm. An ideal dataset with no noise was created with otherwise identical

parameters for comparison. The results of this comparison are seen in Figure 30.

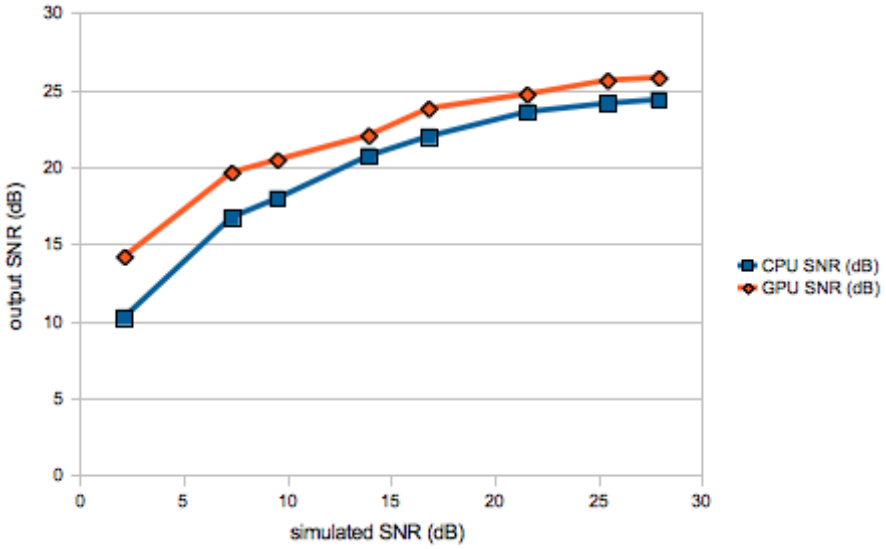

Figure 30: SNR of reconstructed images plotted against the SNR of the given input for CPU (blue squares) and GPU (red triangles)

7.4 Speed

The reconstructed images from the CPU and GPU versions of the CGLS algorithm are intended to be identical. This is not precisely the case for reasons outlined in Section 7.2.1 but the two images should be indistinguishable to the eye. The advantage of employing the parallelism of the GPU is speed. Many parallel threads calculating simultaneously should enjoy greater throughput than a CPU implementation executing one or a few threads at a time.

To measure the algorithm's acceleration on the GPU several simulated phantom datasets were generated in MATLAB and reconstructed on both the CPU and GPU code versions. The length of the k-space trajectories and final image resolution were varied to increase the size of the encoding matrix. Times were measured by start/stop commands within the code to measure total execution times of the CGLS iterations. Reading and

processing of program inputs were not included in these measurements because the two code versions are identical in this respect and doing these tasks requires negligible time in the overall execution.

The vast majority of execution time is spent performing the matrix-vector multiplications necessary for the CGLS solver. This is also the easiest code to execute in parallel. The acceleration afforded by the GPU is clearly demonstrated in Figure 31. The GPU reconstruction shows acceleration factors ranging from 72-98x over the CPU implementation.

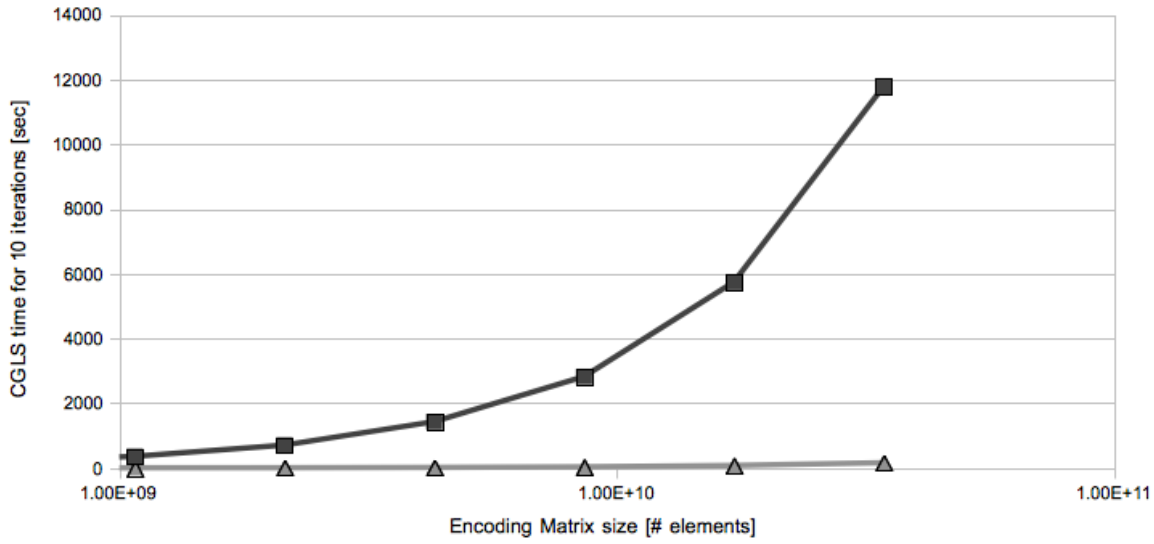

Figure 31: Time comparison of CGLS algorithm implemented on a CPU (squares) and the Tesla C1060 GPU (triangles) for increasing encoding matrix sizes.

7.5 CGLS Convergence

It is important to understand the convergence behavior of the CGLS algorithm to develop reasonable stopping criteria. CGLS can be performed for an arbitrary number of iterations but we would like to know the point of diminishing returns where taking the time to calculate additional iterations will not yield any appreciable benefit. To test this many experiments were performed on simulated images of various resolutions versus an

idealized version of these images. This allows us to calculate the NRMSE of each iteration against the ideal. The expected L-curve convergence behavior was observed and can be seen in Figure 32. In repeated experiments of different scan parameters the lowest NRMSE consistently occurred on or just after iteration number corresponding to $\sqrt{res}$. This is advantageous since increasing the image resolution will more slowly increase the number of iteration required for convergence.

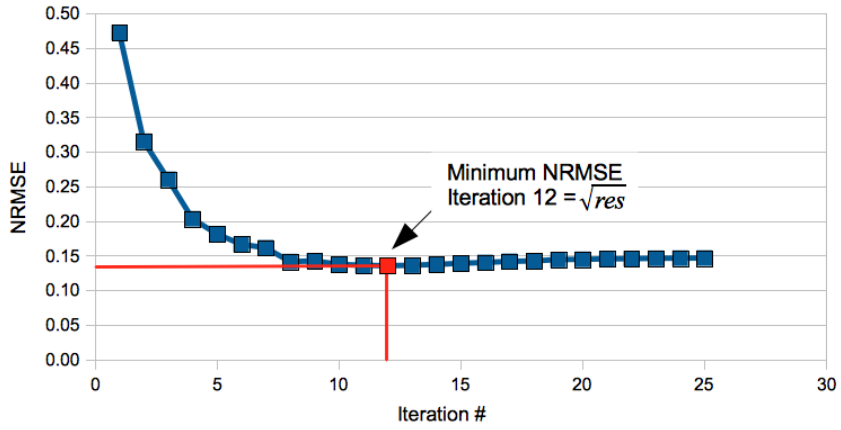

Figure 32: plot of NRMSE at each iteration tested against an idealized reconstruction.

It should be stressed that low NRMSE does not necessarily mean that the image is the best one. An analysis of error does not necessarily correspond with human perception of the highest quality image. A simple inspection of different iterations revealed good correlation between low NRMSE and image quality with no discernible difference in iterations beyond the lowest NRMSE. For an idea of the improvement in image quality versus the number of iterations see Figure 33.

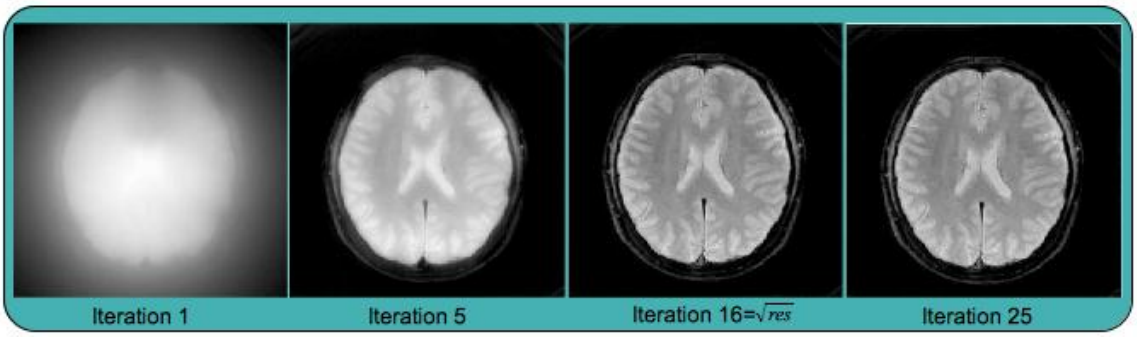

Figure 33: demonstration of CGLS convergence at several iterations.

7.6 GPU Usage

Taking full advantage of the GPU's parallel architecture is paramount in the algorithm's performance. In an ideal situation, maximum occupancy of the multiprocessors will give the greatest instruction throughput. The CUDA Occupancy Calculator (17) was used to evaluate potential block sizes and register and shared memory usages. The Occupancy Calculator is a spreadsheet-based tool provided by NVIDIA which allows the user to select a GPU compute capability along with block sizes along with shared memory and register requirements. The user can see what effect changes in these parameters will have on multiprocessor occupancy and what may be causing any limitation.

An option is available with the CUDA nvcc compiler to display actual kernel usage of important computing resources. These include usage of local, shared, and constant memories along with register and the overall occupancy achieved by each kernel in the code. Results from these measurements were used to help analyze and reduce bottlenecks limiting occupancy of the multiprocessors.

Initially the CUDA kernels achieved 67% occupancy on the multiprocessors. This was seen to be limited by the number of registers being used. After some code

adjustments to reduce the register usage and increase use of shared memory occupancy now achieves 100% during all kernel executions. The limitation is now simply due to the number of blocks per multiprocessor. This number can vary depending on the compute capability of the GPU. Testing the same design parameters on the latest GPU architecture, compute capability 2.0 the results still show 100% occupancy. The code is well designed to reach maximum parallelism in all CUDA GPUs currently available and for the foreseeable future.

7.7 Multiple GPUs

The ability to quickly and easily install multiple GPU devices in a single computer is another advantage of the GPU approach. Several GPU cards may be installed and increase the parallelism available to the algorithm. In contrast, CPU clusters are needed to extend their computational power but these are much more complicated to set up and maintain. A 4 GPU Tesla C1060 system, such as used in this study, is capable of 4 tera-FLOPS (floating-point operations per second). A CPU cluster of similar capability would cost roughly 20 times this configuration (14).

The multi-GPU environment can be easily programmed with the CUDA language extensions. CPU clusters are not as easily programmable and their use has a much higher learning curve. Any changes to the number or configuration of a CPU cluster may necessitate code changes to adjust to the new environment. Meanwhile CUDA greatly simplifies the problem and very easily accommodates upgrades and changes.

The system in this study used 4 Tesla C1060 GPU devices. There are many different ways the GPUs could be used together in the reconstruction problem. It is desirable to maintain maximum occupancy across all of these devices and each of the

multiprocessors within them. Minimizing data sharing between GPUs is important for performance reasons. Interdependence between the devices could create a situation where one device is waiting on another for some result. This would prolong latency and slow down the overall performance of the algorithm. Also, data sharing between GPUs must be transferred on the host's data bus. Such bus accesses are much slower than internal memory accesses within a single GPU. An optimal solution maintains independence of the GPUs while using all of them to their fullest extent.

Dividing the reconstruction problem evenly across the GPU devices by scan slice is the most efficient and straightforward approach. Slices are acquired independently in the MRI scan and can also be reconstructed separately. It might seem ideal to divide the computation by coil. This is possible but makes reaching the goal of reusing sections of the encoding matrix as much as possible more complicated. For instance, the field map of each slice is unique. If the problem is divided by coil field maps for every slice will need to be loaded into every device. This takes additional time and memory. Splitting the problem by reconstructing one whole slice per device ensures the independence of the GPUs, uses the minimum memory space, and requires the least control overhead by the host machine.

This solution was testing by reconstructing a volume of slices allocating each slice to a separate GPU. This approach is illustrated in Figure 34.

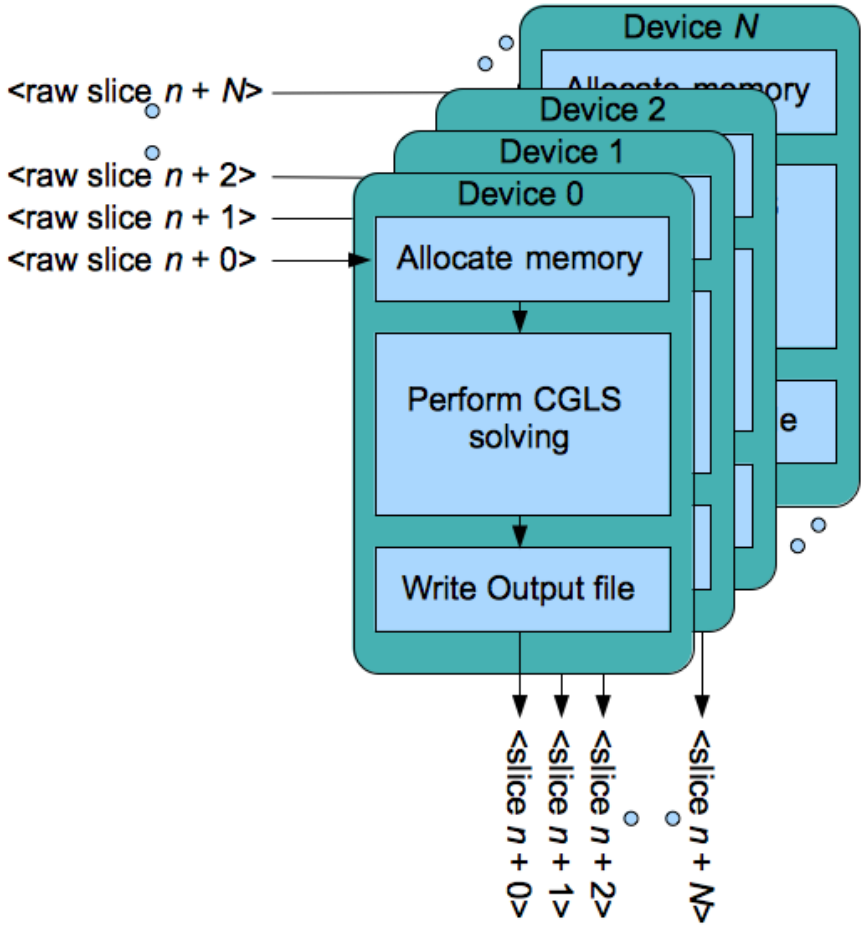

Figure 34: Illustration of CGLS implementation with multiple GPUs.

In reconstructing a large volume of many slices a new slice would begin reconstruction on a GPU as soon as the previously assigned one was done. The time required to reconstruct the entire volume should be reduced by a factor of *N* number of GPUs used. Figure 35 below shows very good results from this arrangement. A multi-slice dataset of a set size was reconstructed on a varying number of GPUs. As the number of GPUs used in the test were increased, the total reconstruction time decreased in close agreement with the expected 1/*N* rate. There is some cost associated with using multiple GPUs. This is due to control overhead created by the host communicating with

the GPUs. Per slice this overhead never exceeded 13% in any of the tests. This means that while using multiple GPUs each slice may take slightly longer to reconstruct than it would only using one GPU but the total throughput is roughly *N* times higher.

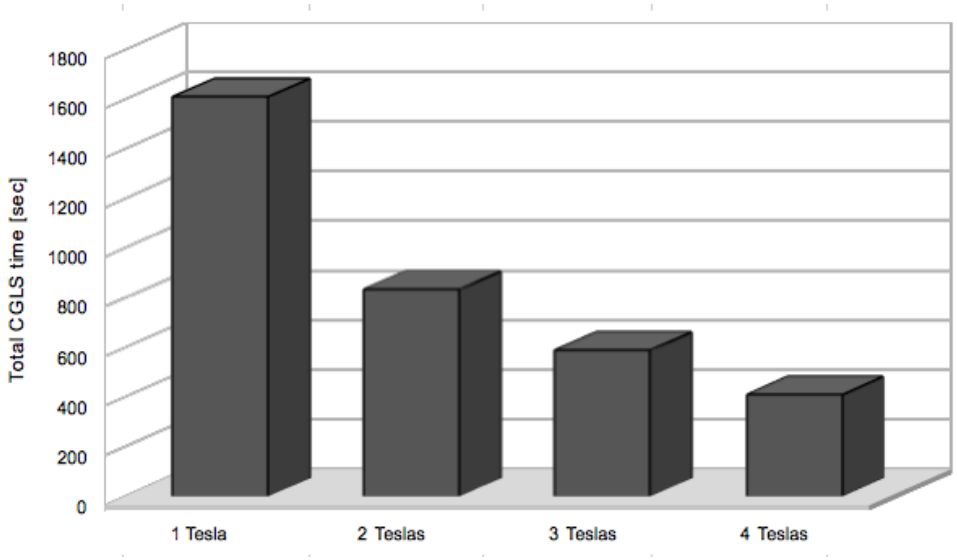

Figure 35: graph of CGLS reconstruction time for a 10 slice volume using several GPUs in parallel.

CHAPTER 8

SUSCEPTIBILITY MAPPING

8.1 Introduction

As we have seen, analysis of iron concentration is of great importance in monitoring the progression of disease in the brain. The ability to specifically quantify and analyze the distribution of ferromagnetic materials in human tissue could provide useful contrast and important metrics for the study of disease. Of particular interest are artificial magnetic contrasts such as gadolinium. Such paramagnetic contrasts are used to increase tissue contrast by changing a tissue's natural relaxation time.

Traditional methods relying predominantly on tissue contrast from image magnitude is limited when imaging ferromagnetic molecules due to strong T2* effects which cause rapid signal decay. By creating a susceptibility map we can quantify how much iron may be present in a given voxel.

8.2 Susceptibility Mapping

Susceptibility mapping can be done by considering a detailed fieldmap as the result of the sum of all dipole-dipole interactions within the FOV. The interactions of the dipoles are governed by Maxwell's magnetostatic field equation (21) and the Lorentz sphere correction. These principles are employed to calculate the localized magnetic fields created by dipole moments. The dipole response is the spatial domain is described by Equation 5 below.

$$d(r) = \frac{3\,cos^2(\theta) - 1}{4\pi r^3} \qquad [5]$$

To calculate the field map based upon a susceptibility distribution this dipole response is convoluted with the susceptibility map. This convolution is done in the spatial domain. This formulation is seen in Equation 6 (22) where it can be seen that the change in the B0 field is a convolution of the dipole response with the distribution of susceptibility. Alternately, this can be done by a point-wise multiplication of the Fourier transformed dipole response by the Fourier transformed susceptibility map. This formulation is seen in Equation 7. An example of the spatial dipole response is seen in Figure 36.

$$\Delta B(r) = \frac{1}{4\pi}\int_{r\neq r'} \chi(r')\frac{3\,cos^2(\theta\,)-1}{|r'-r|^3}d^3r' \quad [6]$$

$$\Delta B(r) = F^{-1}\left\{\left(\frac{1}{3}-\frac{k_z^2}{k^2}\right)^{-1} F(\chi)\right\} \quad [7]$$

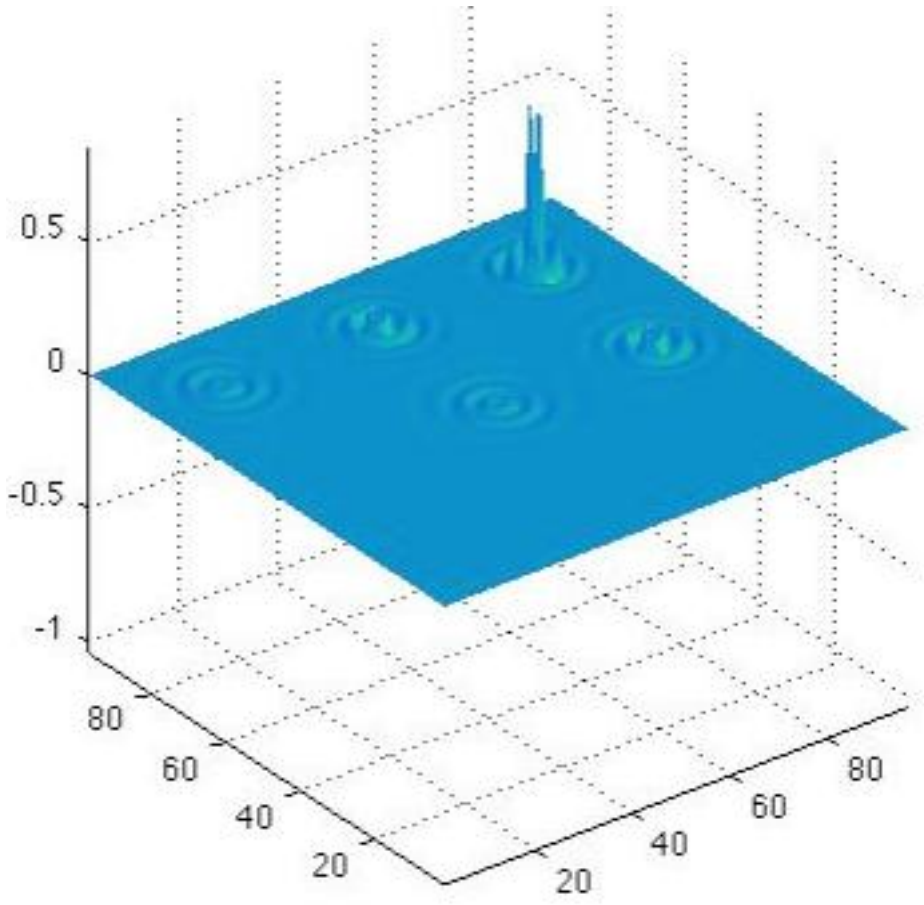

Figure 36: Surface plot of the dipole response through 5 slices.

At first, this point-wise frequency multiplication relationship makes the inverse problem of solving for the susceptibility map from the field map appear to be a straightforward operation. Unfortunately, the dipole response does not lend itself to easy inversion. At points where $k^2 = 3k_z^2$ the dipole response becomes zero. In the spatial domain, this corresponds to the "magic" angle of 54.7^o between two dipoles and the B0 applied field. When performing the inverse of the frequency domain relationship these zeros cause the problem to be ill-posed. It has been suggested that an appropriate discretization can avoid these zeros while substituting very small coefficients, (23). This avoids some computational problems but further problems remain. The small values near

the zeros cause strong noise amplification in the solution when the inversion is performed. Another approach is necessary.

If we formulate the forward problem as a convolution in the spatial domain and express this relationship in a matrix-vector format we get eqn. 8.

$$D\chi = \psi \qquad [8]$$

Here, *D*, is the dipole response matrix. Each column represents a different voxel location. The same is true for each row. At the matrix' intersection of a particular row and column is the dipole response between them based upon their Euclidean distance and the angle between them in relation to the external B0 applied field. The variable, χ, is the susceptibility mapping to be solved for. The measured field map is represented here by ψ.

Eqn. 8 is in the familiar system form for our CGLS linear system solver. In this spatial formulation the "magic" angle zero values are still present but the problem is now analogous to an undersampled dataset. We can now apply techniques from chapters 5-7 to solve for the susceptibility map. Since the problem is ill-conditioned additional constraints to regularize the system are required in our solver.

A method known as COSMOS (Calculation Of Susceptibility through Multiple Orientation Sampling) has been suggested to improve the condition number of the dipole response matrix. By performing multiple scans of an object each with a different orientation to the B0 field the intersection of ill-posed magic angles in the linear system can be minimized. Liu et al. (24) analyze optimal sampling orientations needed to stabilize this problem. From this work at least three scans are needed at widely different angles, (0^0, 60^0, and 120^0). These angles are unrealistic for a human subject within the

MRI scanner. Additionally, the multiple scans needed increase the time the subject is in the scanner.

A more practical approach is to use a typical regularization technique on the single-scan data. Tikhonov regularization has been suggested in (22) to provide data-driven stability to the inverse solver. This approach was taken here to minimize the propagation of noise from the ill-posed magic angle components. The minimization problem can now be formulated as seen in eqn. 9.

$$\min_{\chi} \|D\chi - \psi\|_2^2 + \lambda\|\chi\|_2 \quad [9]$$

The regularization parameter is used to tune the strength of the penalty provided by χ itself. The calculated susceptibility is useful for the penalty because noisier high frequency regions of the mapping will be more heavily penalized. The familiar CGLS algorithm can now be implemented to solve this problem.

8.3 Implementation of CGLS for Susceptibility Mapping

The form of eqn. 9 can be rewritten to a more direct form for implementation of the CGLS solver. We will now work with the matrix vector relationship seen in eqn. 10.

$$\min_{\chi} \left\| \begin{bmatrix} \boldsymbol{D} \\ \lambda \end{bmatrix} \chi - \begin{bmatrix} \psi \\ 0 \end{bmatrix} \right\|_2^2 \quad [10]$$

Since χ is in the form of a diagonal matrix in toward left-hand side of the equation the implementation of the multiplication is a point-wise operation. This keeps the regularization from significantly slowing our calculation time. Another implementation of the CGLS algorithm was coded for the GPU with the same design goals as Section 6.1.2. The length of intermediate variable *p* in the CGLS algorithm was doubled to accommodate the regularization values. This did not stress global memory resources on

the GPU.

Two major issues need to be addressed to accelerate calculation speed as much as possible. First, the size of the dipole response matrix is extremely large and quite dense. This matrix contains the interaction coefficients for each dipole acting on every other dipole in the system. For a volume of dimensions *MxMxN* the size of this matrix is $(MxMxN)^2$. The time needed to calculate such a large matrix is exacerbated by the complexity of calculating each value. This involves calculating both the Euclidean distance between the dipoles and the angle between them with respect to the B0 field. These values must then be combined into the dipole response equation seen in Equation 5. Also, calculation of the angle requires prior calculation of the distance this introduces some serialization. Once these are calculated, the overall dipole response can be calculated. While each dipole-dipole response can be calculated independently, the components of each calculation must be calculated in sequence.

8.4 Susceptibility Results

The susceptibility linear solver was tested using phase maps from our SWI sequences. The maps have an in-plane resolution of 1.4mm and a slice thickness of 2mm. A scaling z-ratio of slice thickness:in-plane resolution is used to calculate distances between voxels. Phase maps were unwrapped using a high-pass Fermi filter. The phase maps were then scaled according to the proposed relation: $\omega = -\phi/(\gamma*TE*B0)$. This provides an estimate from a single echo, (22). The CGLS algorithm can now take this scaled phase map along with the z-ratio to calculate susceptibility maps. An example of such a map is seen in Figure 37a).

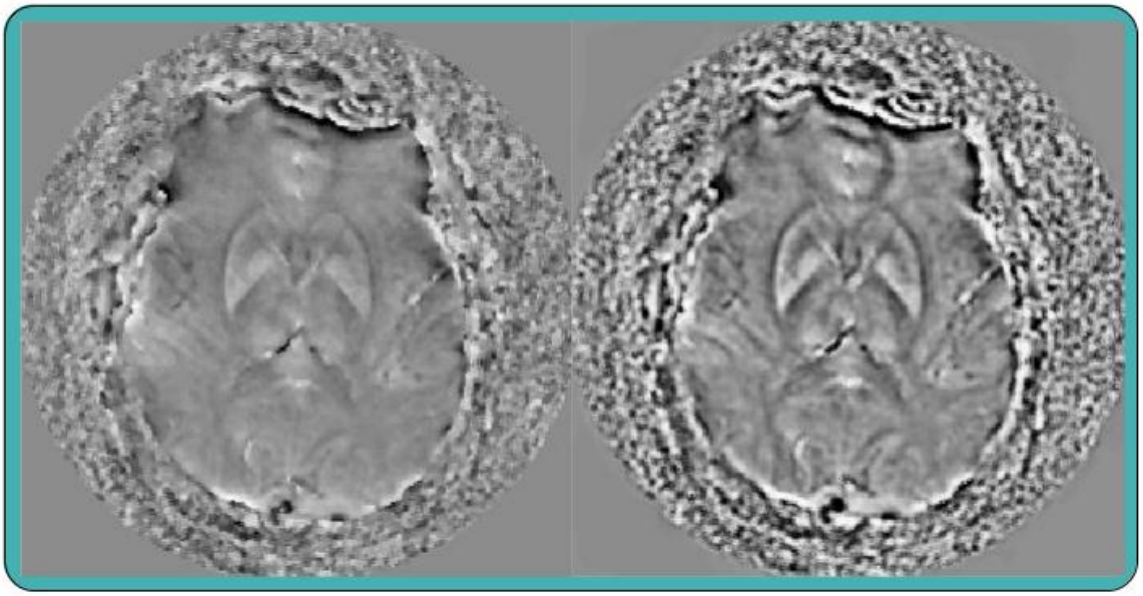

Figure 37: a) Example of phase image input into susceptibility calculations. b) Resultant susceptibility map calculated using 5 surrounding slices. Brighter areas indicate stronger susceptibility.

It was observed in several of the subjects that susceptibility was highest in the putamen and pallidum of both hemispheres. This is especially true in the left hemisphere, which is consistent with both our own findings as detailed in Section 2.4 and our review of the literature. Our results also reveal higher susceptibility in the regions above the sinus where the air/tissue boundary increases the off-resonance.

With the calculated susceptibilities we can now make some differentiation of the overall magnetic properties of each voxel. Diamagnetic materials account for 99% of human tissue, (25). Our results concur that the majority of tissue in the brain is diamagnetic. The ferromagnetic iron of interest exhibits positive susceptibility relative to the B0 field. Visualizing only those measurements with positive susceptibility produces in the image seen in Figure 38. The ferromagnetic effect of iron is very clearly seen in the putamen and pallidum with a preference for the left hemisphere. This correlates with our T2/T2* relaxometry results from Section 2.4.

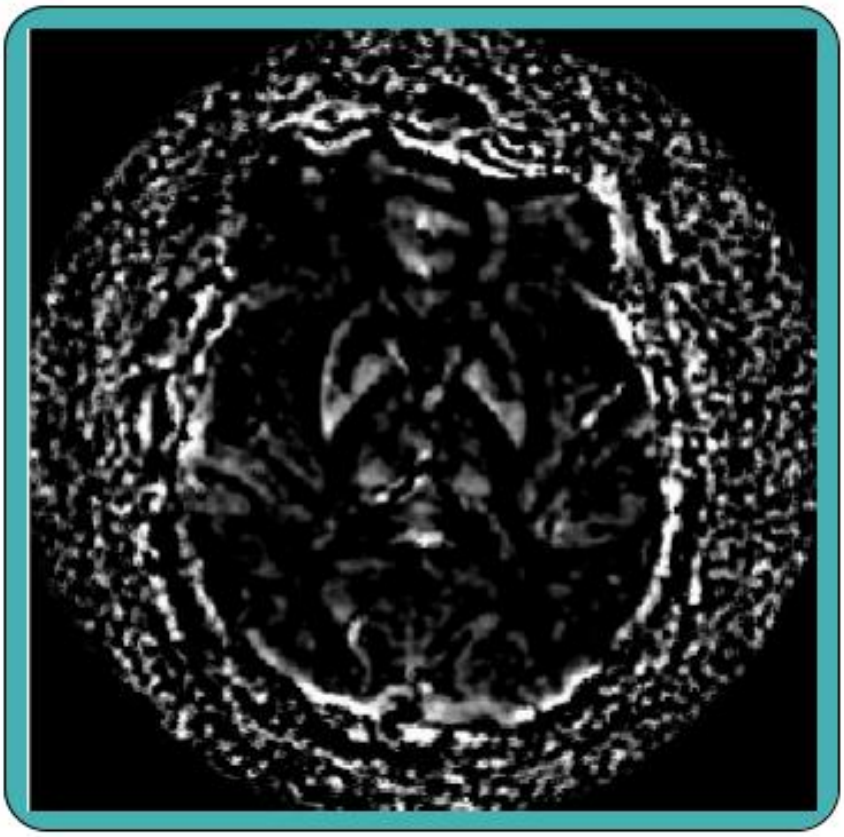

Figure 38: Mapping of regions with susceptibility calculated to be ferromagnetic.

This method allows us to image only the ferromagnetic contributions to the resultant field map. Partial volume effects are still present but the high image resolutions afforded by CGLS reconstruction helps to minimize this confounding factor. Mapping the susceptibility in this way provides high spatial specificity and a more direct measurement of iron concentration.

CONCLUSION

The relaxometry method implemented here does show utility as a tool to monitor the progression of neurodegenerative disease. The T2* shortening effect associated with increased iron concentration was observed both as a function of the normal aging process and HIV. The areas in which this effect was measured agree well with other clinical research results. While this relaxometry approach was effective in observing statistically significant changes in populations with many subjects, it is only a surrogate measurement which remains sensitive to off-resonance and partial volume effects.

High resolution, higher quality images were obtained by employing a GPU-accelerated CGLS algorithm. The algorithm was shown to be capable of producing good quality images by using SENSE undersampling and off-resonance field map correction. This GPU achieved a 72-98x speed advantage over a CPU implementation. The resultant phase images are well-suited for susceptibility mapping.

Using the new high-quality phase images, a GPU-accelerated susceptibility inverse calculator was implemented using another CGLS approach. The GPU cuts iteration time considerably, making a previously intractable problem more practical for high resolution images over multiple slices. The resultant susceptibility maps provide a more direct and quantitative measure of iron concentration with great spatial specificity.

APPENDIX A

CONJUGATE-GRADIENT LEAST-SQUARES ALGORITHM

In order to solve a system of linear equations of the form Equation 2, the CGLS algorithm uses the following iterative approach.

Initialization:

$$d = A^H * m \qquad r = m, \qquad \rho_0 = d^H * d, \qquad b = 0$$

Iterations until convergence or maximum specified iterations reached:

$$p = A * d$$

$$\alpha = \rho_k / p^H * p$$

$$b = b + \alpha * d$$

$$r = r + \alpha * p$$

$$s = A^H * r$$

$$\rho_{k-1} = \rho_k, \qquad \rho_k = s^H * s, \qquad \beta = \rho_k / \rho_{k-1}$$

$$s = s + \beta * d$$

$$d = s$$

GLOSSARY

CGLS:

Conjugate-Gradient Least Squares. The conjugate-gradient least squares algorithm iteratively applies forward and backward matrix/vector operators along with error residuals to solve an optimization problem relating two transform domains.

Flip angle:

The angle between the axis of a magnetic moments precession and the main B0 field. This angle is induced by the application of the RF pulse to apply a torque around an axis parallel to the B0 field.

Larmor frequency:

The angular frequency of a nuclear magnetic moment in an applied field. The Larmor frequency of hydrogen nuclei is 42.58Mhz/T.

MIP:

Minimum Intensity Projection. A MIP is created by retaining the value of minimum intensity along a given view, such as along the z-axis across multiple slices. Such images are often used to visualize vascular connectivity.

MRI:

Magnetic Resonance Imaging. Magnetic Resonance Imaging makes use of an applied magnetic field, magnetic field gradients, and radio frequency excitation pulses to create and measure signals which can be transformed into an image of the scanned object.

Precession:

The wobbling of an axis of rotation under the influence of an applied torque tracing a conical space, as in the action of a top.

SENSE:

Sensitivity Encoding. SENSE is used in parallel MRI to incorporate multiple coil sensitivity information to assist in image reconstruction. This method allows for proper regularization of undersampled data and can decrease scan time.

SWI:

Susceptibility-Weighted Imaging. This imaging modality makes use of T2*-driven tissue contrast which is sensitive to localized magnetic susceptibility of tissues and blood.

T2 :

The exponential time constant for a 63% relaxation of transverse magnetization due to proton dephasing.

T2*:

The exponential time constant for spin-spin relaxation 63% of transverse magnetization. T2* is sensitive to localized inhomogeneities in the magnetic field. This is the principle contrast in susceptibility-weighted imaging.

BIBLIOGRAPHY

1. *SENSE: Sensitivity Encoding for fast MRI.* **Pruessman KP, Weiger M, Scheidegger MB, Boesiger P.** 5, Nov. 1999, Magn. Reson. Med., Vol. 42, pp. 952-962.

2. *The effects of age on the non-haemin iron in the human brain.* **Hallgren, B. and Sourander, P.** 1, Oct. 1958, Journal of Neurochemistry, Vol. 3, pp. 41-51.

3. *Transferrin and Iron in Normal, Alzheimer's Disease, and Parkinson's Disease Brain Regions.* **Loeffler, D.A., et al., et al.** 2, Aug. 1995, Journal of Neurochemistry, Vol. 65, pp. 710-716.

4. *Increased Basil Ganglia Iron Levels in Huntington's Disease.* **G. Bartzokis, MD, et al., et al.** 5, 1999, Archives of Neurology, Vol. 56, pp. 569-574.

5. *Myelin Breakdown and Iron Changes in Huntington's Disease: Pathogenesis and Treatment Implications.* **Bartzokis, G., et al., et al.** 10, 2007, Neurochemical Research, Vol. 32, pp. 1655-1664.

6. *Characterizing Iron Deposition in Multiple Sclerosis Lesions using Susceptibility-Weighted Imaging.* **Haacke, E.M., et al., et al.** 3, Feb. 2009, Magn. Reson. in Medicine, Vol. 29, pp. 537-544.

7. *The Measurement of R2, R2*, and R2' in HIV-infected Patients using the Prime Sequence as a measure of Brain Iron Deposition.* **Miszkiel, K.A., et al., et al.** 10, Apr. 1997, Magn. Reson. Imaging, Vol. 15, pp. 1113-1119.

8. *FSL: New tools for functional and structural brain image analysis.* **Smith, S., et al., et al.** 2001, NeuroImage, Vol. 13.

9. *Age distribution and iron dependancy of the T2 relaxtion time in the globus pallidus and putamen.* **Schenker, C., et al., et al.** 2, 1993, Neuroradiology, Vol. 35, pp. 119-124.

10. *Mapping Brain Assymetry.* **Toga, A.W. and Thompson, P.M.** Jan. 2003, Nature Reviews Neuroscience, Vol. 4, pp. 37-48.

11. *A fast sinc function gridding algorithm for Fourier inversion in computer tomography.* **O'Sullivan, J.** 1985, IEEE Trans. Med. Imaging, Vols. MI-4, pp. 200-207.

12. *Selection of a convolution function for Fourier inversion using gridding.* **Jackson, J.I., et al., et al.** 3, Sept. 1991, IEEE Trans. Medical Imaging, Vol. 10, pp. 473-478.

13. **Kaiser, J.F.** Digital Filters. [book auth.] Kuo F.F. and Kaiser J.F. *System Analysis by Digital Computer.* New York : Wiley, 1966, 7.

14. *Accelerated multidimensional radiofrequency pulse design for parallel transmission using concurrent computation on multiple graphics processing units.* **Deng, W., Yang, C. and Stenger, V.A.** 2, Nov. 2010, Magn. Reson. in Medicine, Vol. 65, pp. 363-369.

15. **Cai, Wei.** *Optimization of a GPU Implementation of Multii-Dimensional RF Pulse Design Algorithm.* Electrical Engineering, University of Hawaii at Manoa. Honolulu, HI : s.n., 2011.

16. **NVIDIA Corporation.** NVIDIA Tesla C1060 Computing Processor. [Online] Feb. 2011. http://www.nvidia.com/object/product_tesla_c1060_us.html.

17. —. CUDA Occupancy Calculator. *CUDA Toolkit 3.2.* [Online] Jan. 2011. http://developer.nvidia.com/object/cuda_3_2_downloads.html.

18. **Tasche, M. and Zeuner, H.** Roundoff Error Analysis for Fast Trigonometric Transforms. [book auth.] G.A. Anastassiou. *Handbook of Analytic-Computational Methods in Applied Mathematics.* Boca Raton : CRC Press, 2000, 8.

19. **Wilkinson, J.H.** *Rounding Errors in Algebraic Processes.* Mineola : Courier Dover Publications, 1994.

20. **NVIDIA Corporation.** CUDA C Programming Guide. *CUDA Toolkit 3.2.* [Online] Jan. 2011. http://developer.download.nvidia.com/compute/cuda/3_2_prod/toolkit/docs/CUDA_C_Programming_Guide.pdf.

21. *Classical Electrodynamcis, 3rd Ed.* **Jackson, J.D.** 9, Sept. 1999, American Journal of Physics, Vol. 67, p. 841.

22. *Nonlinear regularization for per voxel estimation of magnetic susceptibility distributions from MRI field maps.* **Kressler, B., et al., et al.** 2, Feb. 2010, IEEE Trans. on Medical Imaging, Vol. 29, pp. 273-281.

23. *Imaging iron stores in the brain using magnetic resonance imaging.* **Haacke, E.M., et al., et al.** 1, 2005, Magn. Reson. Imaging, Vol. 23, pp. 1-25.

24. *Calculation of susceptibility through multiple orientation sampling (COSMOS): A method for conditioning the inverse problem from measured magnetic field map to susceptibility source image in MRI.* **Liu, T., et al., et al.** 2009, Magn. Reson. Imaging, Vol. 61, pp. 196-204.

25. **Atlas, S.W.** Magnetic Resonance Imaging of the Brain and Spine, 4th Ed. *Magnetic Resonance Imaging of the Brain and Spine, 4th Ed.* s.l. : Lippincott Williams & Wilkins, 2008, Vol. 1 & 2, p. 646.