Chapter 4

Introduction to Parallelizing Compressed Sensing Magnetic Resonance Imaging

Michael Lustig, Kurt Keutzer, and Shreyas Vasanawala

1 Introduction

While Par Lab’s official health application was blood vessel and blood flow modeling for stroke victims (see Chapter 5), our work in accelerating image reconstruction for compressed sensing in magnetic resonance imaging (MRI) was entirely serendipitous. Though, coincidently, the same MRI work plays an important role in a different approach to enable clinical research of assessing blood flow in patients with congenital heart disease [1, 2, 8].

What makes MRI the most powerful and versatile diagnostic imaging modality is the large number of flexible contrast mechanisms. These provide essential clinical information of both anatomy and function. With MRI we can distinguish between white and gray matter in the brain, obtain quantitative measurements of blood flow, measure temperature, measure changes in blood oxygenation which is an indicator of neuronal activation, and even follow metabolic activity. Since its invention more than 30 years ago, MRI has revolutionized the field of diagnostic medicine. Imaging speed is a major part of this revolution as it is essential in many MRI applications. Improvements in MRI hardware and imaging techniques have enabled faster data collection, and hence faster imaging. However, we are currently at the point where fundamental physical and physiological effects limit our ability to simply collect and encode data more quickly. This limits the use of MRI in applications that require rapid imaging of fast-moving objects, such as in the abdomen and the heart.

2 Pediatric Body MRI

A particular application that could greatly benefit from rapid imaging is pediatric body MRI. The evaluation of pediatric health issues poses unique challenges, as many patients have limited ability to communicate symptoms and physical examination can be challenging. Thus, imaging technologies may provide a powerful tool to overcome these limitations. Radiography and CT are rapid and robust modalities; however, children have a significantly elevated estimated risk of cancer induced by ionizing radiation used in these modalities. Estimates of risk of radiation-induced cancer from a typical CT scan are thought to be around 1 in 1000 for young children. Because protocols for diagnosis and monitoring of pediatric malignancies require serial imaging with multiple scans, and because the risks of radiation-induced cancer are likely additive, the overall risk of radiation-induced fatal cancers is greater than 1 in 200. In contrast, MRI offers superb soft tissue characterization with global anatomic assessment, and lacks ionizing radiation. Thus, MRI has the potential to be a dominant pediatric imaging modality [11].
Despite compelling advantages of MRI, several limitations markedly restrict its role in pediatric health. Pediatric MRI exams are long, often in excess of 1 hour. The long image acquisition time results in degraded image quality from motion artifacts, either from voluntary motion, or physiologic motion (respiration, bowel peristalsis). The lengthy exams also limit the availability of MRI given limited magnet time. However, the most significant consequence of long exam times is the need for sedation or general anesthesia. Unfortunately, sedation for MRI adds costs and has significant risks. These risks may deter a physician from requesting an MRI despite its benefits. Thus, the limitations of lengthy examinations and operator dependence of MRI often result in children having exams delayed, deflected to CT, or simply foregone completely.

Figure 1 demonstrates the importance of rapid pediatric MRI. This MR study of a 6-year-old was performed without anesthesia at the insistence of the child’s mother. For the initial portion of the exam, the child was cooperative. The child became restless a few minutes later, resulting in non-diagnostic images. If the entire abdomen could be imaged in less time, this patient could have had a diagnostic MRI without anesthesia. Figure 2 shows that children as young as 4 years of age can be coached to remain in an MRI scanner motionless to the limit of their attention span. Generally, this attention span is 3-5 minutes per year of age. These examples show that with very short exam times, within the child’s attention span, excellent images could be obtained.

3 Rapid MRI with Compressed Sensing and Parallel Imaging

Since the data collection has fundamental limits, many researchers look for methods to reduce the amount of acquired data without degrading the image quality. These reduced sampling methods are based on the fact that MRI data is redundant, so the underlying information may be extracted from fewer measurements than traditionally considered necessary. One of the most significant clinical impacts of reduced sampling methods has been accomplished by parallel imaging with multiple receiver channels. Imaging with multiple channels provides more useful data per MRI acquisition, so fewer acquisitions are needed per scan. Another source of redundancy that has been gaining significant attention is the sparsity and compressibility of various MR signals. This effort has been motivated by the theory of compressed sensing—a general theory for sampling of sparse signal, but in particular applicable to MRI [3].

Medical images, much like natural images taken by digital cameras, can be compressed many-fold—one such example is the popular JPEG compression. The typical paradigm of compression is to first collect all the necessary data and then compress it. The question that arises is why is it necessary to collect so many measurements if most of the data is non-important? Compressed sensing provides a way to address this question. It is a new sampling theory for compressible signals that allows sampling at rates much lower than the Nyquist rate. Compressed sensing implicitly compresses data within the signal acquisition process by obtaining fewer so-called incoherent measurements. This is accomplished through various non-uniform and pseudo-random sampling schemes. Images can be accurately reconstructed from these measurements using non-linear recovery processes that enforce data consistency with the measurements and compressibility of the reconstruction. The practical result of compressed sensing in the context of MRI is that MR images require much less data for reconstruction, and hence can be scanned much faster.

Compressed sensing MRI is an exploding research area, with the number of publications increasing almost exponentially in the past 8 years. Initial studies suggest that compressed sensing can have a tremendous impact on clinical
Figure 2: Another exam showing the motivation and feasibility of this project. The mother of a 4-year-2-month-old girl insisted on MRI without anesthesia. With the skilled help of our child development specialists, the girl cooperated by laying still in an MRI for 10 minutes: high-quality images were obtained (middle). She then became restless (bottom, arrows show motion ghosting) and the exam was terminated. If MRI can be performed rapidly enough, anesthesia is obviated.

imaging. Compressed sensing offers a synergistic enhancement to parallel imaging with a built-in noise reduction. These innovations were well embodied in Michael Lustig’s research as he interviewed for a faculty position in spring of 2009. Applications of compressed sensing to MRI had formed the focus of Lustig’s research during his days as a doctoral student at Stanford University [3]. Immediately after graduation, Lustig continued as a staff researcher and started working very closely with Dr. Shreyas Vasanawala, a pediatric radiologist at Lucile Packard Children’s hospital. Dr. Vasanawala, an MD PhD came originally from the same group as Lustig and just started his position as a faculty. Dr. Vasanawala’s broad background and immense enthusiasm made it possible to rapidly translate the compressed sensing technology into clinical practice for imaging pediatric patients [10].

As it happened Kurt Keutzer was the last person to interview Lustig during Michael’s interview for a faculty position in the EECS Department at Berkeley in the spring of 2009. During this routine interview Keutzer, reasoning that what was lost in image sampling was likely to complicate image reconstruction, asked Lustig: “Doesn’t this undersampling impact reconstruction time? Don’t you write a kind of a computational IOU in this process?” Lustig smiled and with extreme candor replied: “I’m glad this is my last interview slot and nobody has asked me this so far.”

In fact, despite the many advantages, to this point compressed sensing MRI had enjoyed relatively slow adoption by manufacturers and radiologists. In part, the slow clinical adoption rate for any new technique is natural, since its safety and efficacy must be carefully assessed. Subtle artifacts in images can obscure detection of disease or even falsely suggest pathology where none exists (Figure 3). Ensuring that these situations occur very infrequently requires deployment of compressed sensing in the clinic, which is hampered when image reconstruction times are long. In particular, an image reconstruction time longer than about 2 minutes is not acceptable in clinical settings, as immediate feedback on the success of the exam is necessary to make decisions about whether to re-administer contrast medications and to plan imaging planes. At the time of his faculty interview, Lustig’s software took over 2 hours to reconstruct an image that was the result of compressed sensing. This made clinical use entirely infeasible.

There were other motivations for speeding up image reconstruction. Even before reaching the clinical stage, fast image reconstruction would facilitate the optimization of compressed sensing images (Figure 4). The process of adjusting the image reconstruction algorithms and parameters is itself iterative: images must be reconstructed, reviewed carefully, reconstructed again with a slightly different approach, and so on. Further, this optimization must be performed for each clinical application, such as imaging of joints, or the brain, or blood vessels. Naturally, manufacturers are hesitant to invest in such a large undertaking, particularly when it is unclear if the methods are safe and effective. A faster reconstruction permits more refined improvement of the algorithm and its parameters. Thus, fast reconstruction...
is the key for adoption of compressed sensing approaches to MRI.

4 Improving the Performance of Compressed Sensing MRI

Given the strong motivation for performance improvement for compressed sensing MRI, and given the opportunistic attitude prevalent within the Par Lab, Lustig and Keutzer struck up plans for collaboration and interest in the project by graduate student Mark Murphy enabled work to get under way in summer of 2009. Parallelizing image reconstruction for MRI had been considered before [7], but compressed sensing brought new algorithmic and computational challenges. Mark made rapid progress and by the end of the summer it seemed likely that he would succeed in bringing the runtimes within clinical feasibility. By fall of 2009 he had submitted his work that ultimately appeared as [5]. Early in 2010, with the dedicated help of Marcus Alley, a senior researcher at Stanford Radiology, Murphy’s reconstruction was integrated into the clinical workflow at Lucile Packard Children’s Hospital at Stanford and routine clinical work has begun. Our experience with the reconstruction over the next 2 years is described in the conference proceedings article included in this volume.

What makes the problem of fast reconstruction so challenging is that a compressed sensing reconstruction solves a large-scale non-linear optimization. This requires the use of iterative schemes. Each iteration involves operations such as Fourier transforms, sparsifying transforms such as wavelets, and operations that are required for parallel imaging, such as multiplications with coil sensitivity functions or convolutions. These operations by themselves are computationally intensive. In addition, the number of iterations required is a polynomial function of the size of the problems with larger sizes taking many more iterations to converge. As with other Par Lab applications such as content-based image retrieval (CBIR) or speech recognition, our work on parallelizing the application involved developing a software architecture, identifying key computational patterns, and then matching parallelizing strategies to the various computations. At the top level the architecture of this application can be described as a pipe-and-filter architecture in which each filter contains a map of spectral and/or dense linear algebra computations, followed by a
Figure 4: Optimization of image reconstruction parameters. Left image: Zoomed image of a small portion of the liver reconstructed with conventional parallel imaging is too grainy. Middle image: Compressed sensing reconstruction of the same data is not grainy, but the boundary of the liver (arrow) and a branch of an artery to the liver (dashed arrow) are blurred. Right image: After optimizing a parameter of the reconstruction, the image has a nice balance of minimal graininess and sharp delineation of anatomical structures.

reduce into the reconstructed image. Even this simple architecture is sufficient to bring forward the opportunities for significant data-level parallelism.

The Transactions on Medical Imaging paper that follows in this volume [4] is the culmination of work that began with [5], and focuses on the approach we had taken to significantly reduce the reconstruction times while ensuring scalability of the problem as the dimensions and size of the problem increases. It provides a detailed analysis of current performance and predictions for future performance with ever improving parallel platforms. The outcome is a practical, clinically feasible reconstruction that has been in routine clinical use for the past 3 years. This clinical application is briefly described in the next section, and in more detail in another paper of this volume [9].

5 Clinical Application

The major achievement of this work was the ability to install the reconstruction software at Lucile Packard Children’s Hospital. This has enabled Dr. Shreyas Vasanawala, a radiologist, to pioneer the deployment of the technology of compressed sensing in the clinic. For the first time, compressed sensing accelerated acquisitions were being acquired, and high-quality reconstructions were showing up on the scanner monitor within a couple of minutes. Based upon early experience with a compressed sensing MR data acquisition and image reconstruction technique, a two-fold imaging speedup for anatomic imaging is readily achievable. This two-fold speed up is now in regular use and has yielded multiple benefits. For instance, imaging of the abdomen typically requires a period of suspended respiration to avoid degradation of the images from organ motion. Prior to deployment of compressed sensing, a routine breath-hold to acquire images at diagnostic resolution was 20-30 seconds depending on the size of the child and the specific clinical task. Many children cannot hold their breath that long, particularly if they have diseases of the heart or lungs. However, a 10-15 second breath-hold will yield images at diagnostic resolution and is much more readily achieved. For younger children who need sedation in order to remain still, a shorter period of suspended respiration is more easily and more safely obtained by an anesthesiologist. Thus, from a practical standpoint, there are compelling drivers of adoption of compressed sensing in a pediatric MR imaging practice. The paper [9] describes in retrospect the two years of experience in using compressed sensing in the clinic. It describes the technical approach of accelerating the acquisition, accelerating the reconstruction, and the clinical application.
Figure 5: Compressed sensing MR images of a child born with abnormal connections of blood vessels and chambers in the heart. Left: Color overlay of blood flow on a grayscale anatomic image, with faster flow in the aorta and pulmonary artery (dashed arrows) shown in yellow and red. Oxygenated blood from a portion of the lungs (short arrow) is abnormally flowing to a vessel that should carry deoxygenated blood (thin arrow). Right image: The same child also has an abnormal connection in the heart so that blood starting at the arrow that should only go to the left side of the heart, but can be seen streaming abnormally into both sides.

6 Future Work

There are many promising directions for future work in compressed sensing MRI. With problem sizes that often occur in 3D imaging, imaging with high-density coils (> 16 channels) and dynamic imaging, the system memory and data transfers from memory to the CPU/GPU become major limiting factors. These have to be managed carefully and be considered in the design of the reconstruction algorithms. For example: a desired high-resolution dynamic cardiac MRI scan with \((256 \times 256 \times 128)\) spatial matrix \(\times 24\) times points \(\times 16\) channels results in a staggering problem size of more than 3 billion complex variables and requires 24 Gb just to store the matrix. This requires distributed memory parallel processing solutions. Interestingly, the original health application of blood flow modeling could also have an impact in combination to compressed sensing MRI of blood flow imaging. MRI provides a method to directly measure flow in the body, as shown in Figure 5. At the same time, these acquisitions suffer from low resolution, a very long scan time, and huge amounts of data. Using physical models to constrain blood flow reconstruction in compressed sensing MRI could have huge benefits, improving the resolution and reducing acquisition time. However, the computation barrier is huge. As a first step to combine these approaches, Frank Ong, an undergraduate student at Berkeley, has developed and implemented a scheme to reduce the noise in MRI flow images by enforcing divergence-free properties of blood flow in the reconstruction. This year, Frank took Keutzer’s class on parallel computing and as a class project was able to reduce the reconstruction time by so much that the denoising parameters can be modified interactively with instant feedback [6]. This entire scan described in Figure 4 was completed in under 10 minutes. With the fast compressed sensing reconstruction techniques from the Par Lab, we now have an unprecedented diagnostic window into pediatric heart diseases.

Bibliography


Abstract

We present $\ell_1$-SPIRiT, a simple algorithm for auto calibrating parallel imaging (acPI) and compressed sensing (CS) that permits an efficient implementation with clinically-feasible runtimes. We propose a CS objective function that minimizes cross-channel joint sparsity in the Wavelet domain. Our reconstruction minimizes this objective via iterative soft-thresholding, and integrates naturally with iterative Self-Consistent Parallel Imaging (SPIRiT). Like many iterative MRI reconstructions, $\ell_1$-SPIRiT’s image quality comes at a high computational cost. Excessively long runtimes are a barrier to the clinical use of any reconstruction approach, and thus we discuss our approach to efficiently parallelizing $\ell_1$-SPIRiT and to achieving clinically-feasible runtimes. We present parallelizations of $\ell_1$-SPIRiT for both multi-GPU systems and multi-core CPUs, and discuss the software optimization and parallelization decisions made in our implementation. The performance of these alternatives depends on the processor architecture, the size of the image matrix, and the number of parallel imaging channels. Fundamentally, achieving fast runtime requires the correct trade-off between cache usage and parallelization overheads. We demonstrate image quality via a case from our clinical experimentation, using a custom 3DFT Spoiled Gradient Echo (SPGR) sequence with up to $8 \times$ acceleration via Poisson-disc undersampling in the two phase-encoded directions.

1 Introduction

Imaging speed is a major limitation of MR Imaging, especially in comparison to competing imaging modalities such as Computed Tomography (CT). MR allows much more flexible contrast-generation and does not expose patients to ionizing radiation, and hence does not increase risk of cancer. However, other imaging modalities are substantially more popular, as MR scans are slow, expensive, and in some cases less robust. Patient motion during long scans frequently causes image artifacts, and for uncooperative patients, like children, anesthesia is a frequent solution. Acquisition time in MRI can be reduced by faster scanning or by subsampling. Parallel imaging [34, 32, 15] is a well-established acceleration technique based on the spatial sensitivity of array receivers. Compressed sensing (CS) [7, 10, 23] is an emerging acceleration technique that is based on the compressibility of medical images. Attempts to combine the two have mostly focused on extensions of iterative SENSE [31] with SparseMRI [23]. In [4] Block et al., added total-variation to a SENSE reconstruction from radial sampling, Liang et al., in [21] showed improved acceleration by first performing CS on aliased images and then applying SENSE to unfold the aliasing, Otazo et al. used compressed sensing with SENSE to accelerate first-pass cardiac perfusion [30]. More recently [38, 42] have
presented some improvements, again, using an extension of SENSE. The difficulty in estimating exact sensitivity maps in SENSE has created the need for autocalibrating techniques. One class of autocalibrating algorithms extends the SENSE model to joint estimation of the images and the sensitivity maps [44, 39]. Combination of these approaches with compressed sensing have also been proposed. Knoll et al. [19] proposed a combination with Uecker’s non-linear inversion and Huang et al. [17] proposed a self-feeding SENSE combined with compressed sensing.

A different, yet very popular class of autocalibrating techniques are methods like GRAPPA [15] that do not use the sensitivity maps explicitly. In [24] we proposed an optimized iterative method, SPIRiT, and demonstrated the combination with non-linear regularization. In [22] we presented and extension, \( \ell_1 \)-SPIRiT, that synergistically combines SPIRiT with compressed sensing and in [41, 40] we presented more details and clinical results in pediatric patients.

The combination of compressed sensing with parallel imaging has the advantage of improved image quality, however it comes at a cost. These algorithms involve substantially more computation than direct or iterative linear reconstructions.

In this paper we discuss the \( \ell_1 \)-SPIRiT reconstruction. \( \ell_1 \)-SPIRiT solves a constrained non-linear optimization over the image matrix. The non-linearity of this optimization necessitates an iterative reconstruction, and we describe our simple and efficient POCS algorithm in Section 2.

A recent trend in MRI has been to accelerate reconstructions by implementing and optimizing them for massively parallel processors. Silicon manufacturing technology has recently experienced the end of a trend that produced the incredible pace of computational speed during the 1990’s [12]. In the past decade, all major microprocessor vendors have increased the computational throughput of their designs by introducing programmer-visible parallelism. Intel and AMD provide 4-16 CPU cores per socket, and GPGPUs typically have 16-32 massively multithreaded vector cores per socket. In each case, the computational throughput of the processor is proportional to the number of cores, and future designs will have larger numbers of cores.

This paper discusses the massively parallel implementation of \( \ell_1 \)-SPIRiT on these processors. The resulting sub-minute runtimes demonstrate that computational expense is not a substantial obstacle to clinical deployment of \( \ell_1 \)-SPIRiT. Many previous works have demonstrated substantial improvement in reconstruction runtime using GPUs and multi-core CPUs as parallel execution platforms. Chang and Ji [8] demonstrated multi-channel acceleration by solving SparseMRI reconstruction separately for each channel and reporting 1.6-2.0 acceleration using 4 cores. More recently Kim et al. [18] present a high-performance implementation of a SENSE based compressive sensing reconstruction, describing many low-level optimizations that apply for both CPU and GPU architectures.

Stone et al. [37] describe the implementation of an iterative reconstruction using the Conjugate Gradient (CG) algorithm to solve regularized linear reconstructions for non-Cartesian trajectories. Their implementation relies on a highly optimized GPU implementation of a non-uniform Fourier transform (NDFT) to perform sub-minute non-Cartesian reconstructions. Wu et al. [43, 47] have generalized this work to model other acquisition effects in the NDFT, such as off-resonance and sensitivity encoding. Several other works have discussed the GPU implementation of Gridding [3], a highly accurate NDFT approximation. Sørensen et al. [35] describe an algorithm for obviating potentially expensive synchronization in a GPGPU implementation of Gridding.

Obeid et al. [29] use a spatial-partitioning approach to optimize gridding interpolation, and report 1-30 second runtimes. Nam et al. [1] describe another gridding implementation achieving sub-second interpolations for highly undersampled data. Several other works have presented GPU implementations of Parallel Imaging (PI) reconstructions with clinically-feasible runtimes. Roujol et al. [33] describe GPU implementation of temporal sensitivity encoding (TSENSE) for 2D interventional imaging. Sørensen et al. [36] present a fast iterative SENSE implementation which performs 2D gridding on GPUs. Uecker [39] describes a GPU implementation of a non-linear approach to estimate PI coil sensitivity maps during image reconstruction.

This work presents the parallelization of an autocalibrating approach, \( \ell_1 \)-SPIRiT, via multi-core CPUs and GPUs and the resulting clinically-feasible reconstruction runtimes. Moreover we discuss the approach taken to parallelizing the various operations within our reconstruction, and the performance trade-offs in different parallelization strategies. Additionally, we discuss the data-size dependence of performance-relevant implementation decisions. To our knowledge, no previous works have addressed this issue.
2 Iterative Self-Consistent Parallel Imaging Reconstruction (SPIRiT)

SPIRiT is a coil-by-coil autocalibrating parallel imaging method and is described in detail in [24]. SPIRiT is similar to the GRAPPA parallel imaging method in that it uses autocalibration lines to find linear weights to synthesize missing k-space. The SPIRiT model is based on self-consistency of the reconstructed data with the acquired k-space data and with the calibration.

SPIRiT is an iterative algorithm in which in each iteration non-acquired k-space values are estimated by performing a linear combination of nearby k-space values. The linear combination is performed using both acquired k-space samples as well as estimated values (from the previous iteration) for the non-acquired samples. If we denote $x_i$ as the entire k-space grid of the $i^{th}$ coil, then the consistency criterion has a form of a series of convolutions with the so called SPIRiT kernels $g_{ij}$. The SPIRiT kernels are obtained by calibration from auto calibration lines similarly to GRAPPA. If $N_c$ is the total number of channels, the calibration consistency criterion can be written as

$$x_i = \sum_{j=1}^{N_c} g_{ij} \ast x_j.$$ 

The SPIRiT calibration consistency for all channels can be simply written in matrix form as

$$x = Gx,$$

where $x$ is a vector containing the concatenated multi-coil data and $G$ is an aggregated operator that performs the appropriate convolutions with the $g_{ij}$ kernels and the appropriate summations. As discussed in [24], the $G$ operator can be implemented as a convolution in k-space or as multiplication in image space.

In addition to consistency with the calibration, the reconstruction must also be consistent with the acquired data $Y$. This can be simply written as

$$y = Dx,$$

where $D$ is an operator that select the acquired k-space out of the entire k-space grid. In [24] two methods were proposed to find the solution that satisfies the constraints. Here we would like to point out the projection over convex sets (POCS) approach which uses alternate projections that enforce the data consistency and calibration consistency. In this paper we extend the POCS approach to include sparsity constraints for combination with compressed sensing.

As previously mentioned, the convolution kernels $g_{ij}$ are obtained via a calibration from the densely sampled auto-calibration region in the center of k-space, commonly referred to as the Auto-Calibration Signal or ACS lines. In the reconstruction we would like to find $x$ that satisfies $x = Gx$. However in the calibration $x$ is known and $G$ is unknown. We compute the calibration in the same way as GRAPPA [15], by fitting the kernels $g$ to the consistency criterion $x = Gx$ in the ACS lines. Due to noise, data corruption, and ill-conditionedness, we solve this fit in an $\ell_2$-regularized least-squares sense. We compute the calibration once, prior to image reconstruction. Calibration could potentially be improved via a joint estimation of the kernels and images, as has been presented by Zhao et al. as Iterative GRAPPA [46] and in SENSE-like models by Ying et al. [44] and Uecker et al [39]. Joint estimation is more computationally expensive, but its runtime could be improved with techniques similar to those discussed in this paper.

3 $\ell_1$-SPIRiT Reconstruction

Variations of the $\ell_1$-SPIRiT reconstruction have been mentioned in several conference proceedings [22, 26, 20, 41]. More detailed descriptions are given in [24] and in [40]. But for the sake of completeness and clarity we include here a detailed description of the variant that is used in this paper.

$\ell_1$-SPIRiT is an approach for accelerated sampling and reconstruction that synergistically unifies compressive sensing with auto-calibrating Parallel imaging. The sampling is optimized to provide the incoherence that is required for compressed sensing yet compatible to parallel imaging. The reconstruction is an extension of the original SPIRiT algorithm that in addition to enforcing consistency constraints with the calibration and acquired data, enforces joint-sparsity of the coil images in the Wavelet domain. Let $y$ be a the vector of acquired k-space measurements from all the coils, $F$ a Fourier operator applied individually on each coil-data, $D$ a subsampling operator that chooses only
Fast $\ell_1$-SPIRiT Compressed Sensing Parallel Imaging MRI

$x_k$ - image estimate after $k^{th}$ iteration
$y$ - acquired data
$F$ - multi-coil Fourier transform operator
$G$ - SPIRiT operator
$\Psi$ - multi-coil Wavelet transform operator
$D$ - subsampling operator choosing acquired data
$S_\lambda$ - Joint Soft-thresholding

$G \leftarrow \text{AutoCalibrate}(y)$
Initialize $x_0 \leftarrow F^{-1}D^Ty$
for $k = 1, 2, \ldots$ until convergence:

(A) $m_k \leftarrow Gx_{k-1}$
(B) $w_k \leftarrow \Psi^{-1}S_\lambda\{\Psi m_k\}$
(C) $x_k \leftarrow F^{-1}[(I - DT) (FW + DTy)]$

Figure 1: The POCS algorithm. Line (A) performs SPIRiT k-space interpolation, implemented as voxel-wise matrix-vector multiplications in the image domain. Line (B) performs Wavelet Soft-thresholding, computationally dominated by the forward/inverse Wavelet transforms. Line (C) performs the k-space consistency projection, dominated by inverse/forward Fourier transforms.

acquired $k$-space data out of the entire $k$-space grid, $G$ an image-space SPIRiT operator that was obtained from auto-calibration lines, $\Psi$ a wavelet transform that operates on each individual coil separately. $\ell_1$-SPIRiT solves for the multi-coil images concatenated into the vector $x$ which minimizes the following problem:

$$\begin{align*}
\text{minimize}_x & \quad \text{Joint}\ell_1(\Psi x) \\
\text{subject to} & \quad DFx = y \\
& \quad Gx = x
\end{align*}$$

The function Joint$\ell_1(\cdot)$ is a joint $\ell_1$-$\ell_2$-norms convex functional and is described later in more detail. Minimizing the objective (3.1) enforces joint sparsity of wavelet coefficients between the coils. The constraint in (3.2), is a linear data-consistency constraint and in (3.3) is the SPIRiT parallel imaging consistency constraint. The Wavelet transform [5] $\Psi$ is well-known to sparsify natural images, and thus used frequently in Compressive Sensing applications as a sparsifying basis. Just as the Fourier transform, it is a linear operation that can be computed via a fast $O(n \log n)$ algorithm.

As previously mentioned, in this work we solve the above problem via a an efficient POCS algorithm, shown in Figure 1.

This algorithm does not solve the constrained minimization exactly, but instead minimizes the related Lagrangian objective function. The POCS algorithm converges to a fixed-point that satisfies the above constraints, often within 50-100 iterations.

### 3.1 Joint-Sparsity of Multiple Coils

We perform soft-thresholding on the Wavelet coefficients to minimize the $\ell_1$-objective function (3.1). The soft-thresholding function $S_\lambda(x)$ is defined element-wise for $x \in \mathbb{C}$ as:

$$S_\lambda(x) = \frac{x}{|x|} \cdot \max(0, |x| - \lambda)$$

where $|x|$ is the complex modulus of $x$. The parameter $\lambda$ estimates the amplitude of noise and aliasing in the Wavelet basis, and the soft-thresholding operation is a well-understood component of many denoising [11] and compressive sensing algorithms [9].
We use a randomized shifting technique to approximate translation-invariant Wavelets [41, 13], and this has a negligible computational overhead. The individual coil images are sensitivity weighted images of the original image of the magnetization. We assume that coil sensitivities are smooth and do not produce spatial shift of one coil’s image relative to another. Thus, edges in these images appear in the same spatial position, and therefore coefficients of sparse transforms, such as wavelets, exhibit similar sparsity patterns. To exploit this, we use a joint-sparsity model [41, 40]. In compressed sensing, sparsity is enforced by minimizing the $\ell_1$-norm of a transformed image. The usual definition of the $\ell_1$-norm is the sum of absolute values of all the transform coefficients, $\sum_c \sum_r |w_{cr}| = \sum_c \sum_r \sqrt{|w_{cr}|^2}$, where $c$ is the coil index and $r$ is the spatial index. In a joint-sparsity model we would like to jointly penalize coefficients from different coils that are at the same spatial position. Therefore we define a joint $\ell_1$ as:

$$\text{Joint} \ell_1(w) = \sum_r \sqrt{\sum_c |w_{rc}|^2}$$

In a joint $\ell_1$-norm model, the existence of large coefficient in one of the coils, protects the coefficients in the rest of the coils from being suppressed by the non-linear reconstruction. In the POCS algorithm joint sparsity is enforced by soft-thresholding the magnitude of the wavelet coefficients across coils, at a particular position.

### 3.2 Computational Complexity

If $n_c$ is the number of PI channels and $v$ is the number of voxels per PI channel, the computational complexity of our algorithm is:

$$O \left( C_C \cdot n_c^3 + T \cdot ((C_W + C_F) \cdot n_c v \log v + C_S \cdot n_c^2 v) \right)$$

$T$ is the number of iterations the POCS algorithm performs. The algorithm often converges with sufficient accuracy within 50-100 iterations. The constants $C_W$, $C_F$, $C_S$, and $C_C$ indicate that the relative computational cost of the Wavelet transforms, Fourier transforms, and SPIRiT interpolation and calibration are heavily dependent on input data size. Section 6 presents more detailed runtime data.

The $n_c^3$ term represents the SPIRiT calibration, which performs a least-norm least-squares fit of the SPIRiT model to a densely sampled autocalibration region in the center of k-space. Solving each of these systems independently leads to an $O(n_c^4)$ algorithm, which is prohibitively expensive for large coil arrays. Appendix A describes an algorithm that reduces this complexity to $O(n_c^3)$ by re-using a single Cholesky factorization for all channels. This derivation can potentially be used to accelerate the computation of GRAPPA kernels as well.

The $n_c v \log v$ term represents the Fourier and Wavelet transforms, and the $n_c^2 v$ term represents the image-domain implementation of the k-space SPIRiT interpolation. This k-space convolution is implemented as multiplication in the image domain, hence the linearity in $v$ of this term. Due to the $O(n_c^2 v)$ complexity, SPIRiT interpolation is asymptotically the bottleneck of the POCS algorithm. All other operations are linear in the number of PI channels, and at worst log-linear in the number of voxels per channel. There are several proposed approaches that potentially reduce the complexity of the SPIRiT interpolation without degrading image quality. For example ESPRiT [20] performs an eigendecomposition of the G matrix, and uses a rank-one approximation during POCS iterations. Also, coil array compression [6, 45] can reduce the number of parallel imaging channels to a small constant number of virtual channels. Our software includes implementations of both of these approaches, and in practice the $\ell_1$-SPIRiT solver is rarely run with more than 8 channels.

One could solve the $\ell_1$-SPIRiT reconstruction problem (Eqns 3.1-3.3) via an algorithm other than our POCS approach, for example non-linear Conjugate Gradients (NLCG). The computational complexity of alternate algorithmic approaches would differ only in constant factors. The same set of computations would still dominate runtime, but a different number of iterations would be performed and potentially a different number of these operations would be computed per iteration. End-to-end reconstruction times would differ, but much of the performance analysis in this work applies equally well to alternate algorithmic approaches.

### 4 Fast Implementation

The POCS algorithm is efficient: in practice, it converges rapidly and performs a minimal number of operations per iteration. Still, a massively parallel and well-optimized implementation is necessary to achieve clinically feasible
4.1 Parallel Processors

Many of the concerns regarding efficient parallel implementation of MRI reconstructions are applicable to both CPU and GPU architectures. These two classes of systems are programmed using different languages and tools, however have much in common. Figure 2 establishes a four-level hierarchy that one can use to discuss parallelization decisions.
In general, synchronization is more expensive and aggregate data access bandwidth is less at “higher” levels of the hierarchy (i.e. towards the top of Figure 2). For example, Cuda GPGPUs can synchronize threads within a core via a __syncthreads() instruction at a cost of a few processor cycles, but synchronizing all threads within a GPU requires ending a grid launch at a cost of \( \approx 5 \mu s \), or 7,500 cycles. CPU systems provide less elaborate hardware-level support for synchronization of parallel programs, but synchronization costs are similar at the corresponding levels of the processor hierarchy. On CPUs, programs synchronize via software barriers and task queues implemented on top of lightweight memory-system support. With respect to data access, typical systems have \( \approx 10 \) TB/s (\( 10^{13} \) bytes/s) aggregate register-file bandwidth, but only \( \approx 100 \) GB/s (\( 10^{11} \) bytes/s) aggregate DRAM bandwidth. Exploiting locality and data re-use is crucial to performance.

In this work, we do not further discuss cluster-scale parallelization (among Nodes in Figure 2). The CPU parallelization we’ll describe in this section only leverages the parallelism among the multiple Sockets/Cores a single Node. As indicated by Figure 2, parallelization decisions at this level are analogous to decisions among the multiple GPUs in a single system, but we leave more detailed performance analysis of cluster-parallelization to future work.

### 4.2 Data-Parallelism and Geometric Decomposition

The computationally intense operations in MRI reconstructions contain nested data parallelism. In particular, operations such as Fourier and Wavelet transforms are performed over \( k \)-dimensional slices through the \( N \)-dimensional reconstruction volume, with \( k < N \). In most cases the operations are performed for all \( k \)-dimensional (\( k \)-D) slices, providing another source of parallelism to be exploited for accelerating the reconstruction. The \( k \)-D operations themselves are parallelizable, but usually involve substantial synchronization and data-sharing. Whenever possible, it is very efficient to exploit this additional level of parallelism. For the purposes of software optimization, the size and shape of the \( N \)-dimensional (\( N \)-D) data are important. The Geometric Decomposition (GD) design pattern [25] discusses the design of parallel programs in which the data involved have geometric structure. GD suggests the parallelization should follow a division of the data that follows this structure, in order to achieve good caching and inter-thread communication behavior.

Recall from Figure 2 that modern processor architectures provide four levels at which to exploit parallelism. An efficient parallel implementation must decide at which levels of the processor hierarchy to exploit the levels of the nested parallelism in MRI reconstruction.

In volumetric MRI reconstructions, all of these operations are applied to the 4-D array representing the multichannel 3D images. Figure 3 illustrates that the exploitable parallelism of operations over these arrays is two-level: operations like Fourier and Wavelet transforms applied to the individual channels’ images exhibit massive voxel-wise parallelism and require frequent synchronization; but the transforms of the 4-32 channels can be performed independently and in parallel.
In Cartesian acquisitions, the readout direction is never randomly subsampled. Similarly in stack-of-spirals or stack-of-radial acquisitions, the same non-Cartesian sampling of $x - y$ slices is used for every $z$ position. In these cases, the 3D reconstruction can be decoupled into independent 2D reconstructions for each undersampled slice. The resulting reconstruction is not SNR-optimal, since noise is only averaged over the samples within each 2D slice. Additionally, the compressive sensing reconstruction is unable to exploit cross-slice Wavelet-domain sparsity. However, decoupling can provide a substantial performance benefit. Parallelizing over independent 2D reconstructions is very efficient, as the decoupled 2D reconstructions require no synchronization. Our Cuda $\ell_1$-SPIRiT solver is able to run multiple 2D problems simultaneously per GPU in batch mode. Large batch sizes require more GPU memory, but expose more parallelism and can more effectively utilize the GPU’s compute resources.

### 4.3 Size-Dependence and Cache/Synchronization Trade-off

One can produce several functionally equivalent implementations by parallelizing at different levels of the hierarchy in Figure 2. These different implementations will produce identical results\(^1\), but have very different performance characteristics. Moreover, the performance of a given implementation may differ substantially for different image matrix sizes and coil array sizes. In general, the optimal implementation is a trade-off between effective use of the cache/memory hierarchy and amortization of parallelization overheads.

For example, one may choose to exploit the voxel-wise parallelism in an operation only among the vector lanes within a single processor core. The implementation can then exploit parallelism over multiple channels and 2D slices over the multiple cores, sockets, and nodes in the system. Utilizing this additional parallelism will increase the memory footprint of the algorithm, as the working set of many 2D slices must be resident simultaneously. This consideration is particularly important for GPU systems which have substantially less DRAM capacity than CPU systems.

On the other hand, one may leverage voxel-wise parallelism among the multiple cores within a socket, the multiple sockets within the system, or among the multiple nodes. In doing so the implementation is able to exploit a larger slice of the system’s processing and memory-system resources while simultaneously reducing memory footprint and working-set size. The favorable caching behavior of the smaller working set may result in a more efficient implementation. However it is more expensive to synchronize the higher levels of the processing hierarchy. Furthermore for problems with smaller matrix sizes, voxel-wise parallelism may be insufficient to fully saturate the processing resources at higher levels. Even when caching behavior is more favorable, this over-subscription of resources may degrade performance.

Which implementation provides better performance depends both on the size of the input data and on the size of the processor system. When the image matrix is very high-resolution (i.e. has a large number of voxels) or the processing system is relatively small (i.e. a small number of processor cores), then one can expect a high degree of efficiency from exploiting voxel-wise parallelism at higher levels of the hierarchy. If the image matrix is relatively small or the processor system is very large, then one should expect that parallelism from the Channel and Decoupled-2D levels of Figure 3 is more important. As the number of processing cores per system and the amount of cache per core both continue to increase over time, we expect the latter case to become more common in the future.

### 4.4 Parallel Implementation of $\ell_1$-SPIRiT

In the case of $\ell_1$-SPIRiT, there are four operations which dominate runtime: SPIRiT auto-calibration, Fourier transforms during the k-space consistency projection, Wavelet transforms during the joint soft-thresholding, and the image-domain implementation of SPIRiT interpolation. Figure 4 depicts the overall flow of the iterative reconstruction. Note that PI calibration must be performed only once per reconstruction, and is not part of the iterative loop.

**SPIRiT Auto-Calibration** Our $\ell_1$-SPIRiT implementation performs auto-calibration by fitting the SPIRiT consistency model to the densely sampled Auto-Calibration Signal (ACS), which requires solving a least-squares least-norm problem for each PI channel. Note that while we perform the POCS iterations over decoupled 2D slices, we perform calibration in 3D k-space. The SPIRiT interpolation kernels for the 2D problems are computed via an inverse Fourier transform in the readout direction.

\(^1\)Identical up to round-off differences in floating point arithmetic, which is not always associative or commutative
Figure 4: (a) Flowchart of the $\ell_1$-SPIRiT POCS algorithm and the (b) SPIRiT, (c) Wavelet joint-threshold and (d) data-consistency projections

As discussed in Appendix A, the auto-calibration is computationally dominated by two operations: the computation of a rank-$k$ matrix product $A^*A$ and a Cholesky factorization $A = LL^*$, which itself is dominated by rank-$k$ products. Numerical linear algebra libraries for both CPUs and GPUs are parallelized via an output-driven scheme that requires very little inter-thread synchronization. For example, when computing a matrix-matrix product $C = AB$ each element $c_{i,j}$ is computed as an inner product of a row $a_i$ of $A$ with a column $b_j$ of $B$. All such products can be computed independently in parallel, and are typically blocked to ensure favorable cache behavior.

The SPIRiT Operator in Image Space

Figure 4 (b) illustrates the image-domain implementation of SPIRiT interpolation $Gx$. $Gx$ computes a matrix-vector multiplication per voxel – the length $n_c$ (# PI channels) vector is composed of the voxels at a given location in all PI channels. For efficiency in the Wavelet and Fourier transforms, each channel must be stored contiguously – thus the cross-channel vector for each voxel is non-contiguous. Our implementation of the interpolation streams through each channel in unit-stride to obviate inefficient long-stride accesses or costly data permutation. The image-domain implementation is substantially more efficient than the k-space implementation, which performs convolution rather than a multiplication. However, the image-domain representation of the convolution kernels requires a substantially larger memory footprint, as the compact k-space kernels must be zero-padded to the image size and Fourier transformed. Since SPIRiT’s cross-coil interpolation is an all-to-all operation, there are $n_c^2$ such kernels. This limits the applicability of the image-domain SPIRiT interpolation when many large-coil-array 2D problems are in flight simultaneously. This limitation is more severe for the Cuda implementation than the OpenMP implementation, as GPUs typically have substantially less memory capacity than the host CPU system.

Enforcing Sparsity by Wavelet Thresholding

Figure 4 (c) illustrates Wavelet Soft-thresholding. Similarly to the Fourier transforms, the Wavelet transforms are performed independently and in parallel for each channel. Our Wavelet transform implementation is a multi-level decomposition via a separable Daubechies 4-tap filter. Each level of the decomposition performs low-pass and high-pass filtering of both the rows and columns of the image. The number of levels of decomposition performed depends on the data size: we continue the wavelet decomposition until the approximation coefficients are smaller than the densely sampled auto-calibration region. Our OpenMP implementation performs the transform of a single 2D image in a single OpenMP thread, and parallelizes over channels 2D slices.

We will present performance results for two alternate GPU implementations of the Wavelet transform. The first parallelizes a 2D transform over multiple cores of the GPU, while the second is parallelized only over the vector lanes within a single core. The former is a finer-grained parallelization with a small working set per core, and permits an optimization that greatly improves memory system performance. As multiple cores share the transform for a single 2D transform, the per-core working set fits into the small L1-cache of the GPU. Multiple Cuda thread blocks divide the work of the convolutions for each channel’s image, and explicitly block the working data into the GPU’s local store.
Table 1: Table of dataset sizes for which we present performance data. \(n_x\) is the length of a readout, \(n_y\) and \(n_z\) are the size of the image matrix in the phase-encoded dimensions, and \(n_c\) is the number of channels in the acquired data. Performance of SPIRiT is very sensitive to the number of channels, so we present runtimes for the raw 32-channel data as well as coil-compressed 8- and 16- channel data.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>(n_x)</th>
<th>(n_y)</th>
<th>(n_z)</th>
<th>(n_c)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>192</td>
<td>256</td>
<td>58</td>
<td>8, 16, 32</td>
</tr>
<tr>
<td>B</td>
<td>192</td>
<td>256</td>
<td>102</td>
<td>8, 16, 32</td>
</tr>
<tr>
<td>C</td>
<td>192</td>
<td>256</td>
<td>152</td>
<td>8, 16, 32</td>
</tr>
<tr>
<td>D</td>
<td>192</td>
<td>256</td>
<td>190</td>
<td>8, 16, 32</td>
</tr>
<tr>
<td>E</td>
<td>320</td>
<td>260</td>
<td>250</td>
<td>8, 16, 32</td>
</tr>
<tr>
<td>F</td>
<td>320</td>
<td>232</td>
<td>252</td>
<td>8, 16, 32</td>
</tr>
</tbody>
</table>

Parallelism from multiple channels is exploited among multiple cores when a single 2D transform cannot saturate the entire GPU. The latter parallelization exploits all voxel-wise parallelism of a 2D transform within a single GPU core, and leverages the channel-wise and slice-wise parallelism across multiple cores. The working set of a single 2D image does not fit in the l1 cache of the GPU core and we cannot perform the explicit blocking performed in the previous case.

Enforcing k-space acquisition consistency Figure 4 (d) illustrates the operations performed in the k-space consistency projection. The runtime of this computation is dominated by the forward and inverse Fourier transforms. As the FFTs are performed independently for each channel, there is \(n_c\)-way embarrassing parallelism in addition to the voxel-wise parallelism within the FFT of each channel. FFT libraries typically provide APIs to leverage the parallelism at either or both of these levels. The 2D FFTs of phase-encode slices in \(\ell_1\)-SPIRiT are efficiently parallelized over one or a few processor cores, and FFT libraries can very effectively utilize voxel-wise parallelism over vector lanes. We will present performance results for the GPU using both the Plan2D API, which executes a single 2D FFT at a time, and the PlanMany API which potentially executes many 2D FFTs simultaneously. The latter approach more easily saturates the GPU’s compute resources, while the former approach is a more fine-grained parallelization with potentially more efficient cache-use.

5 Methods

Section 6 presents performance results for a representative sample of datasets from our clinical application [41]. We present runtimes for a constant number of POCS iterations only, so the runtime depends only on the size of the input matrix. In particular, our reported runtimes do not depend on convergence rates or the amount of scan acceleration. We present performance results for six datasets, whose sizes are listed in Table 1.

We present several performance metrics of interest. First, we shall discuss the end-to-end runtime of our reconstruction to demonstrate the amount of wall-clock time the radiologist must wait from the end of the scan until the images are available. This includes the PI calibration, the POCS solver, and miscellaneous supporting operations. To avoid data-dependent performance differences due to differing convergence rates, we present runtime for a constant (50) number of POCS iterations.

To demonstrate the effectiveness of our \(O(n^3)\) calibration algorithm, we compare its runtime to that of the “obvious” implementation which uses ACML’s implementation of the Lapack routine cposv to solve each coil’s calibration independently. The runtime of calibration does not depend on the final matrix size, but rather on the number of PI channels and the number of auto-calibration readouts. We present runtimes for calibrating \(7 \times 7 \times 7\) kernels averaged over a variety of ACS sizes.

We also present per-iteration runtime and execution profile of the POCS solver for several different parallelizations, including both CPU and GPU implementations. The per-iteration runtime does not depend on the readout length or the rate of convergence. Since we decouple along the readout dimension, POCS runtime is simply linear in \(n_x\).

Presenting per-iteration runtime allows direct comparison of the different performance bottlenecks of our multiple implementations.
Additionally, we explore the dependence of performance on data-size by comparing two alternate implementations parallelized for the GPU. The first exploits voxel-wise parallelism and channel-wise parallelism at the Socket-level from Figure 2, and does not exploit Decoupled-2D parallelism. This implementation primarily synchronizes via ending Cuda grid launches, incurring substantial overhead. However, the reduced working-set size increases the likelihood of favorable cache behavior, and enables further caching optimizations as described in Section 4.4. Fourier transforms are performed via the 2D API, which expresses a single parallel FFT per grid launch. Fermi-class GPUs are able to execute multiple grid launches simultaneously, thus this implementation expresses channel-wise parallelism as well.

The second implementation exploits voxel-wise parallelism only within a core of the GPU, and maps the channel-wise and Decoupled-2D parallelism at the Socket-level. This implementation is able to use the more efficient within-core synchronization mechanisms, but has a larger working set per core and thus cannot as effectively exploit the GPU’s caches. It also launches more work simultaneously in each GPU grid launch than does the first implementation, and can more effectively amortize parallelization overheads. Fourier transforms are performed via the p1anMany API, which expresses the parallelism from all FFTs across all channels and all slices simultaneously.

All performance data shown were collected on our dual-socket × six-core Intel Xeon X5650 @2.67GHz system with four Nvidia GTX580s in PCI-Express slots. The system has 64GB of CPU DRAM, and 3GB of GPU DRAM per card (total 12 GB). We leverage Nvidia’s Cuda [28] extensions to C/C++ to leverage massively parallel GPGPU processors, and OpenMP2 to leverage multi-core parallelism on the system’s CPUs. Additionally, multiple OpenMP threads are used to manage the interaction with the system’s multiple discrete GPUs in parallel.

We leverage freely available high-performance libraries for standard operations: ACML3 for linear system solvers and matrix factorizations, FFTW4 and CUFFT5 for Fourier transforms.

6 Performance Results

Figures 5-10 present performance data for our parallelized \( \ell_1 \)-SPIRiT implementations.

Figure 5 shows stacked bar charts indicating the amount of wall-clock time spent during reconstruction of the six clinical datasets, whose sizes are listed in Section 5. The POCS solver is run with a single 2D slice in flight per GPU. This configuration minimizes memory footprint and is most portable across the widest variety of Cuda-capable GPUs. In the common case, it provides highest performance. Thus it is the default in our implementation. The stacked bars in Figure 5 represent:

**3D Calibration**: The SPIRiT calibration that computes the SPIRiT \( G \) operator from the ACS data as described in Section 3. Figure 9 presents more analysis of this portion.

**POCS**: The per-slice 2D data are reconstructed via the algorithm described in Figure 1. Figure 6 presents a more detailed analysis of the runtime of this portion.

**other**: Several other steps must also be performed during the reconstruction, including data permutation and IFFT of the readout dimension.

Figures 6 and 7 show the contribution of each individual algorithmic step step to the overall runtime of a single iteration of the 2D POCS solver. In Figure 6, the solver is parallelized so that a single 2D problem is in-flight per GPU. In Figure 7, a single 2D problem is in flight per CPU core. The stacked bars in Figure 6 and Figure 7 are:

**FFT**: The Fourier transforms performed during the k-space consistency projection.

**SPIRiT Gx**: Our image-domain implementation of the SPIRiT interpolation, which performs a matrix-vector multiplication per voxel.

**Wavelet**: The Wavelet transforms performed during wavelet soft-thresholding.

**other**: Other operations that contribute to runtime include data movement, joint soft-thresholding, and the the k-space projection excluding Fourier transforms.

Figure 8 compares the runtime of the Parallel GPU and CPU POCS implementations to the runtime of a sequential C++ implementation, using high-performance libraries and compiled with full compiler optimization. The reported speedup is computed as the ratio of the sequential runtime to the parallel runtime.

Figure 9 demonstrates the runtime of the efficient Cholesky-based SPIRiT calibration algorithm described in Appendix A. The left graph compares the runtime of of our efficient \( O(n^3) \) calibration to the naïve \( O(n^4) \) algorithm.
Fast $\ell_1$-SPIRiT Compressed Sensing Parallel Imaging MRI

The right plot shows what fraction of the efficient algorithm’s runtime is spent in the matrix-matrix multiplication, the Cholesky factorization, and the other BLAS2 matrix-vector operations.

7 Discussion

Figures 5 and 6 present performance details for the most portable GPU implementation of the POCS solver which runs a single 2D slice per GPU. As shown in Figure 5, our GPU-parallelized implementation reconstructs datasets A-D at 8 channels in less than 30 seconds, and requires about 1 minute for the larger E and F datasets. Similarly, our reconstruction runtime is 1-2 minutes for all but the 32-channel E and F data, which require about 5 minutes. Due to the $O(n^3)$ complexity of calibration, calibration requires a substantially higher fraction of runtime for the 32-channel reconstructions, compared to the 8 an 16-channel reconstructions. Similarly, Figure 6 shows that the $O(n^2)$ SPIRiT interpolation is a substantial fraction of the 32-channel POCS runtimes as well. Figures 5 and 6 demonstrate another important trend of the performance of this GPU implementation. Although dataset D is $4 \times$ larger than dataset A, the 8-channel GPU POCS runtimes differ only by about 10%. The trend is clearest in the performance of the Fourier and Wavelet transforms, whose runtime is approximately the same for datasets A-D. This is indicative of the inefficiency of the CUFFT library’s Plan2D API for these small matrix sizes. In a moment we’ll discuss how an alternate parallelization strategy can substantially improve efficiency for these operations.
Figure 7: Per-iteration runtime and execution profile of the multi-core CPU 2-dimensional POCS solver.

Figure 8: Speedup of parallel CPU and GPU implementations of the POCS solver over the optimized sequential C++ baseline.

Figure 9: 3D SPIRiT Calibration runtimes, averaged over a wider variety of auto-calibration region sizes than that represented by Table 1. Calibration is always performed on the CPU via optimized library routines, using all available threads.

Figure 10: Data Size dependence of performance and comparison of alternate parallelizations of the POCS solver.
Figures 7 presents the averaged per-iteration execution profile of the OpenMP-parallelized CPU POCS solver, which uses an #pragma omp for to perform a single 2D slice’s reconstruction at a time per thread. The relative runtimes of the Fourier and Wavelet transforms are more balanced in the CPU case. In particular, the CPU implementation does not suffer from low FFT performance for the small data sizes. The FFT is run sequentially within a single OpenMP thread, and it incurs no synchronization costs or parallelization overhead.

Figure 8 presents the speedup of the multi-GPU solver and the multicore CPU solver over a sequential C++ implementation. Note that the 4-GPU implementation is only about 33% faster than the 12-CPU implementation for the smallest data size (dataset A at 8 channels), while for the larger reconstructions the GPU implementation is 5×-7× faster. The OpenMP parallelization consistently gives 10×-12× speedup over sequential C++, while the multi-GPU parallelization provides 30×-60× speedup for most datasets.

Figure 9 demonstrates the enormous runtime improvement in SPIRiT calibration due to our Cholesky-based algorithm described in Section 3 and derived in Appendix A. The runtime of our calibration algorithm is dominated by a single large matrix-matrix multiplication, Cholesky decomposition, and various BLAS2 (Matrix-vector) operations. For 8 channel reconstructions, the O(n^4) algorithm is faster by 2−3×, while it is 10× faster for 32 channel data. In absolute terms, 8-channel calibrations require less than 10 seconds when computed via either algorithm. However, 32 channel calibrations run in 1-2 minutes via the Cholesky-based algorithm, while the O(n^4) algorithm runs for over 15 minutes.

Figure 10 provides a comparison of alternate parallelizations of the POCS solver and the dependence of performance on data size. The “No Batching” implementation exploits the voxel-wise and channel-wise parallelism within a single 2D problem per GPU Socket. The remaining bars batch multiple 2D slices per GPU Socket. The top bar graph shows runtimes for a small 256 × 58 image matrix, and the bottom graph shows runtimes for a moderately sized 232 × 252 matrix. Both reconstructions were performed after coil-compression to 8 channels.

Fourier transforms in the “No Batching” implementation are particularly inefficient for the small data size. The 256 × 58 transforms for the 8 channels are unable to saturate the GPU. The “Batched 1x” bar uses the P1anMany API rather than the P1an2D API. This change improves FFT performance, demonstrating the relative ineffectiveness of the GPU’s ability to execute multiple grid launches simultaneously. Performance continues to improve as we increase the number of slices simultaneously in-flight, and the FFTs of the small matrix are approximately 5× faster when batched 32×. However, for the larger 232 × 252 dataset, 32× batching achieves performance approximately equal to the non-batched implementation. That the 1× batched performance is worse than the non-batched performance likely indicates that the larger FFT is able to exploit multiple GPU cores.

Our Wavelet transforms are always more efficient without batching, as the implementation is able to exploit the GPU’s small scratchpad caches (Cuda __shared__ memory) as described in Section 4.4. The Wavelet transform performs convolution of the low-pass and high-pass filters with both the rows and the columns of the image. Our images are stored in column-major ordering, and thus we expect good caching behavior for the column-wise convolutions. However, the row-wise convolutions access the images in non-unit-stride without our scratchpad-based optimizations. Comparing the runtimes of the “No Batching” and “Batched 1x” Wavelet implementations in Figure 10 shows that our cache optimization can improve performance by 3×-4×. This is a sensible result, as we use 4-tap filters and each pixel is accessed 4 times per convolution. The cache optimization reduces the cost to a single DRAM access and 3 cached accesses.

Performance can be improved by choosing different parallelization strategies for the various operations. In particular, the best performance would be achieved by using a batched implementation of the Fourier transforms, while using the un-batched implementation of the Wavelet transforms. Such an implementation would still require the larger DRAM footprint of the batched implementation, as multiple 2D slices must be resident in GPU DRAM simultaneously. However it could achieve high efficiency in the Wavelet transform via the caching optimization, and also in the Fourier transforms via higher processor utilization. Although our current implementation does not support this hybrid configuration, Figure 11 shows that it could perform up to 2× faster for the 256 × 58 dataset.

Moore’s Law scaling will result in higher core counts in future architectures. Per Gustafson’s law [16], efficient utilization of future architectures will require increased problem size. In our context, we can increase problem size via larger batch sizes. Per-batch reconstruction time will remain constant, but total reconstruction time will be inversely proportional to batch size. Thus batching potentially provides linear performance scaling with increased core counts.
Figure 11: Performance achievable by a hybrid parallelization of the POCS solver on the $256 \times 58$ dataset.

8 Image Quality

We present more comprehensive evaluation of image quality in prior works [40], and present in Figure 12 a case demonstrating the clinical advantage that high-performance reconstruction can provide. Our 3-Dimensional Compressed Sensing pulse sequence is a modified 3DFT spoiled gradient-echo (SPGR) sequence which undersamples in both of the phase-encoded dimensions ($y$) and ($z$). Acquisitions are highly accelerated, with $4 \times 8 \times$ undersampling of phase encodes. Our clinical imaging is performed using 3T and 1.5T GE systems with a with 32-channel pediatric torso coil. Typical accelerated scan times are 10-15 seconds, and typical ARC [2] reconstruction times are 30-60 seconds. In some cases, we perform partial k-space acquisition in the readout direction. The $\ell_1$-SPIRiT solver is still able to decouple the 2D reconstructions as described in Section 4.2, using only the acquired portion of the readout. Subsequently, we perform Homodyne reconstruction [27] to estimate the missing portion of readout, preventing blur and phase from appearing in the final images [23]. Our reconstruction is performed on-line with coil compression, producing sub-minute runtimes for matrix sizes typically acquired in the clinic. Total latency from scan completion to image availability is 2-3 minutes, 20-70 seconds of which are the POCS solver. The remainder of the reconstruction time is spent performing Grad-Warp [14] and Homodyne processing steps, in addition to file transfers between the scanner and our reconstruction system.

9 Conclusion

We have presented $\ell_1$-SPIRiT, a compressive sensing extension to the SPIRiT parallel imaging reconstruction. Our implementation of $\ell_1$-SPIRiT for GPGPUs and multi-core CPUs achieves clinically feasible sub-minute runtimes for highly accelerated, high-resolution scans. We discussed in general terms the software implementation and optimization decisions that contribute to our fast runtimes, and how they apply for the individual operations in $\ell_1$-SPIRiT. We presented performance data for both CPU and GPU systems, and discussed how a hybrid parallelization may achieve faster runtimes. Finally, we present an image quality comparison with a competing non-iterative Parallel Imaging reconstruction approach.

In the spirit of reproducible research, the software described in this paper is available at: http://www.eecs.
Fast $\ell_1$-SPIRiT Compressed Sensing Parallel Imaging MRI

Figure 12: Comparison of GE Product ARC (Autocalibrating Reconstruction for Cartesian imaging) [2] reconstruction (left images) with our $\ell_1$-SPIRiT reconstruction (right images). Both reconstructions use the same subsampled data, and require similar runtimes. These MRA images of a 5 year old patient were acquired with the 32-channel pediatric torso coil, have FOV $28 \times 22.4 \times 17.6$ cm and spatial resolution $0.55 \times 0.55 \times 0.8$ mm. The images were sampled and reconstructed in a $192 \times 256 \times 220$ image matrix, were acquired with $7.2 \times$ acceleration, via undersampling $3.6 \times$ in the $y$-direction and $2 \times$ in the $z$-direction. The pulse sequence used a 15 degree flip angle and a TR of 3.9 ms. The $\ell_1$-SPIRiT reconstruction shows enhanced detail in the mesenteric vessels in the top images, and and the renal vessels in bottom images.

berkeley.edu/~mlustig/Software.html

Bibliography


Appendix A  $O(n^3)$ SPIRiT Calibration

As discussed in Section 3, our $\ell_1$-SPIRiT calibration solves a least-norm, least-squares (LNLS) fit to the fully-sampled auto-calibration signal (ACS) for each channel’s interpolating coefficients. As each channel’s set of coefficients interpolates from each of the the $n_c$ channels, the matrix used to solve this system has $O(n_c^2)$ columns. Solving the $n_c$ least-squares systems independently requires $O(n_c^4)$ time, and is prohibitively expensive. This section derives our algorithm for solving them in $O(n_c^3)$ time using a single matrix-matrix multiplication, single Cholesky factorization, and several inexpensive matrix-vector operations.

We construct a calibration data matrix $A$ from the calibration data in the same manner as GRAPPA [15] and SPIRiT [24] calibration: each row of $A$ is a window of the ACS the same size as the interpolation coefficients. This matrix is Toeplitz, and multiplication $Ax$ by a vector $x \in \mathbb{C}^{n_c \cdot n_k}$ computes the SPIRiT interpolation: $y = \sum_{i=1}^{n_c} x_i \ast ACS_i$.

The LNLS matrices for each channel differ only by a single column from $A$. In particular, there is a column of $A$ that is identical to the ACS of each coil. Consider the coil corresponding to column $i$ of $A$, and let $b$ be that column. We define $N = A - be_i' - A$ with the $i$th column zeroed out. We wish to solve $Nx = b$ in the least-norm least-squares sense, by solving $(N^*N + \varepsilon I)x := \tilde{N}x = N^*b$. The runtime of our efficient algorithm is dominated computing the product $N^*N$ and computing the Cholesky factorization $LL^* = N$.

Our derivation begins by noting that:

$$\tilde{N} = N^*N + \varepsilon I \equiv (A - be_i')(A - be_i')^* + \varepsilon I = A^*A + \varepsilon I - be_i' - eb_i^*$$

Where we have defined $\tilde{b} = A^*b$ with entry $i$ multiplied by $\frac{1}{2}$ to avoid adding $eb_i^*be_i'$. If we have a Cholesky
factorization $LL^* = A^*A + \varepsilon I$:

$$
\tilde{N} = LL^* - \tilde{b}e' - \tilde{eb}^*
= LL^{-1}(LL^* - \tilde{b}e' - \tilde{eb}^*)L^*
= L(I - \tilde{b}e^* - \tilde{eb}^*)L^*
$$

Where we’ve defined $\hat{b} = L^{-1}\tilde{b} = L^{-1}A^*b$, and $\hat{e} = L^{-1}e$. These vectors can be computed with BLAS2 triangular solves and matrix-vector multiplications. In fact, we can aggregate the $b$’s and $e$’s from all parallel imaging channels into matrices and compute all $\hat{b}$’s and $\hat{e}$’s with highly efficient BLAS3 solves. Now, to solve the system of equations $\tilde{N}x = \tilde{b}$:

$$
x = \tilde{N}^{-1}\tilde{b}
= (L(I - \hat{b}e^* - \hat{eb}^*)L^*)^{-1}\tilde{b}
= L^*(I - \hat{b}e^* - \hat{eb}^*)^{-1}L^{-1}\tilde{b}
$$

It remains to compute the inverse of $(I - \hat{b}e^* - \hat{eb}^*)$. We can define two matrices $\hat{B}, \hat{E} \in \mathbb{C}^{n \times 2}$, where $\hat{B} = -\left(\hat{b}, \hat{e}\right)$, and $\hat{E} = \left(\hat{e}, \hat{b}\right)$. Using the Sherman-Morrison-Woodbury identity:

$$(I - \hat{b}e^* - \hat{eb}^*)^{-1} = (I + \hat{B}\hat{E}^*)^{-1}
= I - \hat{B}(I + \hat{E}^*\hat{B})^{-1}\hat{E}^*$$

Note that $I + \hat{E}^*\hat{B}$ is a $2 \times 2$ matrix that is very inexpensive to invert. Thus we have our final algorithm:

$$
L \leftarrow \text{chol}(A^*A + \varepsilon I)
\hat{b} \leftarrow L^{-1}A^*b
\hat{e} \leftarrow L^{-1}e
\hat{B} \leftarrow -\left(\hat{b}, \hat{e}\right)
\hat{E} \leftarrow \left(\hat{e}, \hat{b}\right)
\hat{x} \leftarrow L^{-*}(I - \hat{B}(I + \hat{E}^*\hat{B})^{-1}\hat{E}^*)\hat{b}
$$
Practical Parallel Imaging Compressed Sensing MRI:
Summary of Two Years of Experience in Accelerating Body
MRI of Pediatric Patients.

S.S. Vasanawala, M.J. Murphy, M.T. Alley, P. Lai,
K. Keutzer, J.M. Pauly, and M. Lustig

©2011 Institute of Electrical and Electronic Engineers, Inc. All rights reserved. Reprinted with the permission of the IEEE. The original article appears as “Practical Parallel Imaging Compressed Sensing MRI: Summary of Two Years of Experience in Accelerating Body MRI of Pediatric Patients” by Shreyas Vasanawala, Mark Murphy, Marcus Alley, Annie Lai, Kurt Keutzer, John Pauly, and Michael Lustig, for the 2011 8th IEEE International Symposium on Biomedical Imaging: From Nano to Macro, Proceedings, pages 1039-1043, March 30 2011 - April 2 2011. Personal use of this material is permitted. However, permission to reuse this material for any other purpose must be obtained from the IEEE. DOI 10.1109/ISBI.2011.5872579

Abstract

For the last two years, we have been experimenting with applying compressed sensing parallel imaging for body imaging of pediatric patients. It is a joint-effort by teams from UC Berkeley, Stanford University and GE Healthcare. This paper aims to summarize our experience so far. We describe our acquisition approach: 3D spoiled-gradient-echo with Poisson-disk random undersampling of the phase encodes. Our reconstruction approach: $\ell_1$-SPIRiT, an iterative autocalibrating parallel imaging reconstruction that enforces both data consistency and joint-sparsity in the wavelet domain. Our implementation: an on-line parallelized implementation of $\ell_1$-SPIRiT on multi-core CPU and General Purpose Graphics Processors (GPGPU) that achieves sub-minute 3D reconstructions with 8-channels. Clinical results showing higher quality reconstruction and better diagnostic confidence than parallel imaging alone at accelerations on the order of number of coils.

1 Introduction

Magnetic resonance (MR) imaging offers superb soft-tissue characterization with global anatomic assessment, has no ionizing radiation, and, thus, has the potential to be a dominant pediatric imaging modality [10]. However, a major limitation of MR imaging is slow imaging speed relative to computed tomography (CT). The resulting motion artifacts and frequent need for anesthesia often result in preference by radiologists and referring clinicians for CT, given its relative ease of use and robustness.

For the last two years we have been experimenting with accelerating acquisitions of pediatric body MRI using the combination of compressed sensing and parallel imaging. Our aim is to achieve fast and robust pediatric MRI that will reduce the need for general anesthesia in pediatric patients and make MRI a viable alternative to CT. This paper aims to describe our experience with compressed sensing in clinical practice. We describe our acquisition approach, reconstruction methods, implementation and results in clinical settings.

2 Approach

At first, our initial efforts were focussed on compressed sensing MRI alone [7]. In the last several years, our approach was to combine compressed sensing with a robust coil-by-coil autocalibrating parallel imaging (acPI) reconstruction [2]. A successful CS reconstruction has three main requirements: (i) sparsity of representation, (ii) incoherent
Practical Parallel Imaging Compressed Sensing MRI

sampling, and (iii) non-linear sparsity enforcing reconstruction. In order to synergistically combine CS with parallel imaging, we reconsidered these requirements in the context of (acPI) and proposed: a modified sparsity model for multiple coil images, an incoherent sampling scheme for imaging with multiple receivers, and an acPI reconstruction that exploits both imaging with multiple coils and the sparsity information. The acPI method is SPIRiT [8] which is based on self-consistency with the calibration and data acquisition. It is a method that exhibits higher accuracy and better noise performance than GRAPPA [5]. It supports arbitrary sampling, and provides a framework to incorporate sparsity constraints, which is essential for combination with CS.

2.1 Incoherent Sampling

In general, random sampling of $k$-space provides the high degree of incoherence needed for compressed sensing. However, pure random sampling is not optimized for parallel imaging with multiple receivers. Random sampling tends to produce sampling patterns with either large gaps or bunched samples. With multiple coil imaging, close samples in $k$-space are naturally correlated. This correlation enables the recovery of missing samples in parallel imaging. This means that bunched samples are “wasteful” as they provide little additional information on the signal. On the other hand, large gaps reduce the reconstruction conditioning of the parallel imaging. Random sampling with minimum distance between samples is called Poisson-disk sampling [3]. Sampling according to a Poisson-disk distribution provides high degree of incoherence and at the same time uniform distance between samples. In addition, this approach also provides flexibility for fractional and anisotropic acceleration (using ellipsoids rather than discs), resulting in a better fit to different coil array geometries.

Figure 1: An example of a 4x4 and a 2.2x4.3 2D accelerated Poisson disc sampling patterns and their associated point spread functions. The incoherent aliasing appears beyond the Nyquist-rate supported field of view. Non-isotropic FOV can be used to adapt to coil array geometries.
2.2 Sparsity of Multiple Coils

The individual coil images are sensitivity weighted images of the original image of the magnetization. Edges in these images appear in the same spatial position, and therefore coefficients of sparse transforms, such as wavelets, exhibit similar sparsity patterns. To exploit this, we use a joint-sparsity model. In compressed sensing, sparsity is enforced by minimizing the $\ell_1$-norm of a transformed image. The usual definition of the $\ell_1$-norm is the sum of absolute values of all the transform coefficients, $\sum_c \sum_r |w_{cr}| = \sum_c \sum_r \sqrt{|w_{rc}|^2}$, where $c$ is the coil index and $r$ is the spatial index. In a joint-sparsity model we would like to jointly penalize coefficients from different coils that are at the same spatial position. Therefore we define a joint $\ell_1$ as: $\text{Joint} \ell_1(w) = \sum_r \sqrt{\sum_c |w_{rc}|^2}$. In a joint $\ell_1$-norm model, the existence of large coefficient in one of the coils, protects the coefficients in the rest of the coils from being suppressed by the non-linear reconstruction.

2.3 $\ell_1$-SPIRiT

SPIRiT is an autocalibrating parallel imaging reconstruction method. It is a generalization of GRAPPA. In SPIRiT, an interpolation kernel, $G$, is calibrated from a fully sampled calibration area in the center of k-space. The missing k-space samples are reconstructed such that they are consistent with the interpolator and the acquired data. Let $x$ be the desired full k-space for all coils, $y$ the acquired k-space, $D$ be an operator that chooses acquired k-space points out of the entire grid, then SPIRiT solves for $x$ that satisfies both $Gx = x$ and $Dx = y$. To combine SPIRiT with CS we also require that the solution has a small Joint $\ell_1$-norm [11, 6]. Therefore we solve for,

$$\begin{align*}
\text{minimize} & \quad \text{Joint} \ell_1(\Psi x) \\
\text{s.t.} & \quad Gx = x \\
& \quad Dx = y.
\end{align*}$$ (2.1)

3 implementation

3.1 Pulse Sequence and Sampling

A standard three-dimensional (3D) spoiled gradient-recalled acquisition in the steady state sequence (SPGR) was modified to include a Poisson disc undersampling distribution of the phase-encodes. The advantage of the Poisson-disk sampling is that data can be reconstructed by using product parallel imaging reconstruction, such as autocalibrating reconstruction for cartesian sampling (ARC; GE Healthcare) [1], a variant of GRAPPA. The resulting images are more immune to calibration error, as any residual aliasing will be incoherent. This ability acts as a “safety net” when testing in patients. Our Poisson-disk implementation is based on the algorithms described in [http://www.devmag.org.za/articles/55-POISSON-DISK-SAMPLING/](http://www.devmag.org.za/articles/55-POISSON-DISK-SAMPLING/) and is based on the algorithm in [3]. The algorithm was modified to enable variable density [7] Poisson-disk for better CS reconstruction and support for Poisson ellipse for supporting unisotropic accelerations. For simplicity of the reconstruction, the Poisson-disk points that are generated are gridded to the closest grid point, and lie on a Cartesian grid. For large matrix size and high-acceleration the properties of the Poisson-disk are fully preserved. For smaller size matrix and low-acceleration they are only approximate. In all the acquisitions a fully sampled window of at least $24 \times 20$ are acquired for the purpose of autocalibration.

3.2 $\ell_1$-SPIRiT Implementation

To implement the optimization problem in Eq. (2.1) we chose a projection over convex sets (POCS) approach. This is an effective implementation that requires very simple operations: convolutions, Fourier and Wavelet transforms and soft-thresholding. The algorithm is the following:

1. Preparations:
   i. Normalize the scale of the data
   ii. Calibrate a 3D kernel from autocalibration lines in 3D k-space
   iii. Compute an inverse Fourier transform of the data in the readout direction to create many separable 2D problems.

2. For each readout position:
   i. Compute a 2D kernel, $G$, for the current readout position from the 3D kernel.
   ii. Initialize: $x_0 = 0$, $\lambda = \text{"big"}$
iii. SPIRiT:
Compute $x_{i+1} = Gx_i$

iv. CS Joint $\ell_1$ Projection:
$x_{i+1} = \Psi^{-1}\text{JointSoftThresh}(\Psi x_{i+1}, \lambda)$

v. Data consistency:
$x_{i+1} = (I - D^T D)x_{i+1} + D^T y$

vi. Adjust $\lambda$, Repeat iii-v

The JointSoftThresh operation is applied at each wavelet coefficient location. At each location it operates jointly on the coefficients from all coils. It computes:

$$\text{JointSoftThresh}(w, \lambda) = w/\|w\| \cdot \{\|w\| - \lambda\}_+.$$  

There are several implementation details that are worth noting: a) Since the readout direction is fully sampled, we compute the inverse Fourier transform and work on many separable 2D reconstruction problems. This reduces the complexity, simplifies the implementation and is easy to parallelize. b) The acquired data is unchanged. The algorithm is only used to extrapolate missing data in $k$-space. We have found that in doing so we consistently get better depiction of features and more natural looking images at the expense of slightly increasing noise. c) We use continuation of the soft-thresholding parameter from high penalty to a very low penalty, modifying it during the iterations. This often leads to much faster convergence of the iterations. We also found that radiologists prefer that images are not denoised. Therefore the $\ell_1$-norm penalty is set such that minimal final denoising is performed. d) Using orthogonal wavelets often results in some blocky artifacts. Translation invariant wavelets can mitigate this, but with significant increase in computation. Alternatively, one can use randomized shifting as suggested by [4] to approximate translation invariant wavelets. We have found that this approach significantly reduces the artifacts and produces much better diagnostic quality images.

### 3.3 Parallel Processing

The $\ell_1$-SPIRiT POCS algorithm is very effective: it produces high-quality images usually after 50-100 iterations. Surprisingly, it is also relatively fast, in that it requires a very small number of operations: each iteration requires 2 (forward and inverse) Fourier Transforms, the SPIRiT $k$-space convolution operator is implemented inexpensively as element-wise multiplication in the image domain, 2 Wavelet transforms (forward and inverse), and a very fast joint soft-threshold operation. Despite this inherent efficiency, the algorithm is still far more expensive than non-iterative reconstruction algorithms. However, we have demonstrated that the algorithm is very amenable to a massively parallel implementation [9].

We have implemented the POCS algorithm in both OpenMP and Nvidia’s Cuda, and deployed it for on-line reconstructions on a dual-socket six-core 2.67 GHz Intel Westmere system with four Nvidia Tesla C1060’s. The calibration implementation relies on Lapack [http://www.netlib.org/lapack] routines from AMD’s ACML [http://www.amd.com/acml]. The POCS iterations are the most expensive step of the reconstruction, accounting typically for 95% of the runtime. As mentioned above, we decouple the 3D reconstruction into many independent 2D problems by inverse Fourier transforming along the fully sampled readout dimension. Both our OpenMP and our Cuda implementation execute multiple 2D problems in parallel. OpenMP executes each 2D reconstruction as a task, and we have as many 2D problems in flight simultaneously as there are CPU cores. Our deployed Cuda implementation concurrently solves as many 2D problems as there are GPUs in the system.\(^1\) Within each 2D problem, we leverage vector-parallelism within the Fourier, Wavelet, interpolation, and thresholding operations to parallelize among thread blocks and threads within a GPU. Efficient utilization of memory bandwidth is crucial to high performance, and wherever possible our implementation coalesces DRAM accesses and caches data in the GPU’s scratchpad (shared) memories.

This parallel implementation provides clinically-useful runtimes of the efficient POCS $\ell_1$-SPIRiT algorithm. Figure 2 shows the runtime of the Cuda POCS implementation running on the four GPUs in our recon system. Even for the largest dataset, which is representative of our highest-resolution scans on 32-channel coils, the POCS iterations run for less than 3 minutes. For typically sized 8-channel scans, runtimes are typically 30 seconds or less. Note that the Tesla C1060 GPUs in our current reconstruction machine are almost two years old: the reconstruction may run up to twice as fast on newer, Fermi-class GPUs.

\(^1\)Our next release concurrently solves as many 2D problems as there are Streaming Multiprocessors in the system’s GPUs.
### Matrix Size POCS Runtime

<table>
<thead>
<tr>
<th>Matrix Size</th>
<th>POCS Runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>172×230×188×32</td>
<td>169.9 s</td>
</tr>
<tr>
<td>192×320×110×32</td>
<td>120.7 s</td>
</tr>
<tr>
<td>320×206×108×12</td>
<td>59.3 s</td>
</tr>
<tr>
<td>192×320×66×8</td>
<td>24.4 s</td>
</tr>
</tbody>
</table>

Figure 2: $\ell_1$-SPIRiT POCS runtime for four representative scans: two large 32-channels, a high-resolution 12-channel scan, and a smaller 8-channel scan.

### 3.4 Clinical Applications

Our CS 3D SPGR sequence was installed first on a 1.5T GE HDxt scanner and later on a 3T GE MR750. We first focused on applications that did not require intravenous contrast so that acceleration limits of the sequence could be explored by repeated acquisitions. These included whole brain imaging, and with the addition of fat suppression, volumetric cartilage imaging. The speed of the reconstruction facilitated iterative image acquisition, image evaluation, and parameter adjustments. We found we could double our imaging speed while maintaining diagnostic image quality [11].

The next application that we explored was magnetic resonance cholangiopancreatography (MRCP). MRCP exams are focused on the bile ducts of the liver and the duct that drains pancreatic fluids into the bowel. These ducts are rather small, filled with fluid that has a long T1 and T2 relaxation time, and are moving due to respiration. Thus, we sought to accelerate the MR acquisition such that high resolution could be maintained, but the scan completed in a breath-hold.

Thus, we modified our SPGR sequence to fully refocus magnetization in each repetition interval, yielding a sequence with T2/T1 contrast. In this case, the bile ducts are bright, but the remainder of human tissue for the most part is flat in contrast. Thus, this application is well suited to compressed sensing: sparse images, a pressing need for encoding speed, and inherent contrast such that acquisition could be repeated in an individual to optimize the technique. With this approach, we found we could achieve acceleration factors in excess of six, obtaining isotropic submillimeter resolution of the upper abdomen in a breath-hold.

We then turned attention to MRI exams enhanced with intravenous contrast. Here the introduction of new techniques is more challenging, as contrast may only be given once and has a rapid transit through the circulatory system. Therefore, only one acquisition can be obtained. Further, it has to be obtained in a manner that ensures high quality diagnostic images with no chance of compromising patient care. In this case, we focused our efforts on magnetic resonance angiography (MRA). Here again, small vessels have to be delineated rapidly, both for breath-holding, and now also because intravenous contrast has a short vascular residence time. Further, the application is similar to MRCP, as the images are relatively sparse, containing vessels and a flat background.

Our initial MRA exams were performed in patients who needed intravenous contrast for delayed contrast-enhanced imaging, but not MRA. This experience revealed that a doubling of imaging speed could be obtained with good image quality, or alternatively, higher resolution images could be obtained in the same scan time. Thus, we now routinely employ CS in our MRA exams. For children in whom a long breath-hold is a challenge, CS enables speed. For others, higher resolution is obtained.

### 4 Experiments and Results

We recently performed a study in which 34 pediatric patients who required an MRI as part of their routine clinical care were enrolled. For these patients, scans were performed with an eight channel coil and at double to triple the speed we would ordinarily employ in routine clinical practice using traditional reconstruction methods. Images were were reconstructed with routine parallel imaging algorithm as well as with $\ell_1$-SPIRiT and then presented to two radiologists to compare image quality and delineation of various anatomic structures. $\ell_1$-SPIRiT image quality was consistently rated the same as or better than that of parallel imaging image quality (Wilcoxon and symmetry tests, $p < .001$), and this effect was strongest for those cases with higher accelerations. A reassuring result was that out of 325 structures
Figure 3: 6 year old female with a transplanted kidney. Top: Cropped images from a 13 second 3 Tesla acquisition with 32 channels and an acceleration factor of 6. The high acceleration factor permits a 320x320 matrix with 2 mm slice thickness. Note improved delineation of artery to the kidney (small white arrow), head of the pancreas (dashed arrow), small vessels in the liver (black arrow) with compressed sensing (CS) reconstruction than parallel imaging (PI) reconstruction. The fast acquisition also permits capturing fast perfusion dynamics, as seen by the differences in contrast enhancement of the splenic red and white pulp tissues (big white arrow). Bottom: Maximum intensity projections (MIP) highlight improved image quality with decreased noise afforded by the CS reconstruction.

While the preceding results were acquired at 1.5 Tesla field strength and eight-channel coils, the technical advances in reconstruction speed have permitted clinical deployment at 3 Tesla field strength with 32-channel coils. As children have smaller size than adults, the higher signal at 3 Tesla and with higher density receive coils greatly improves ability to resolve small anatomic structures. An example is shown in Figure 3. Figure 4 shows an example of a state-of-the-art reconstruction with 32 channels and an even higher 8-fold acceleration.
Figure 4: Reconstruction example with $\ell_1$-SPIRiT using a dedicated 32 channel pediatric body coil. 0.875/1.6 mm in-plane/slice resolution, 8-fold accelerated acquisition of a first pass contrast MR angiography of a 6 year old patient. Pediatric patients have smaller vessels and faster circulation than adults and require much faster imaging. (a) Volume rendering (b) Maximum intensity projection (MIP) and (c) Zoomed MIP showing extraordinary level of details. (d) Our unique variable-density Poisson-disk sampling pattern, optimized for CS with parallel imaging. The data was acquired within 16 seconds compared to 2 min that are required for Nyquist sampling. Due to the rapid acquisition there is no venous contamination in the image. The lack of venous contamination, along with the high SNR of the source images, enables good quality volume rendering.

5 Conclusion

We have presented a simple and effective approach to combining parallel imaging and compressed sensing. We have implemented a clinical pulse sequence and a fast on-line compressed sensing parallel imaging reconstruction. These are installed at Lucile Packard Children’s Hospital and have been used clinically. Our results and experience show that the combination of compressed sensing and autocalibrated imaging is indeed feasible in a clinical setting. The
solution presented above requires a small investment in additional computer hardware, but enables faster and/or higher resolution MRI compared to parallel imaging alone. This approach is of great value for pediatric imaging, but can be used in many other applications.

6 Acknowledgements

The authors are grateful for the support of GE Healthcare, the John and Tashia Morgridge Foundation, and the NIH (R01-EB009690, RR09794-15).

Bibliography


