Published in final edited form as:

Magn Reson Imaging. 2014 July ; 32(6): 747–758. doi:10.1016/j.mri.2014.02.022.

Real-time imaging with radial GRAPPA: Implementation on a Heterogeneous Architecture for Low-Latency Reconstructions

Haris Saybasili^{1,2}, Daniel A. Herzka², Nicole Seiberlich³, and Mark A. Griswold¹

¹ Case Western Reserve University, Department of Radiology, Cleveland, OH

² Johns Hopkins University School of Medicine, Department of Biomedical Engineering, Baltimore, MD

³ Case Western Reserve University, Department of Biomedical Engineering, Cleveland, OH

Abstract

Combination of non-Cartesian trajectories with parallel MRI permits to attain unmatched acceleration rates when compared to traditional Cartesian MRI during real-time imaging. However, computationally demanding reconstructions of such imaging techniques, such as k -space domain radial generalized auto-calibrating partially parallel acquisitions (radial GRAPPA) and image domain conjugate gradient sensitivity encoding (CG-SENSE), lead to longer reconstruction times and unacceptable latency for online real-time MRI on conventional computational hardware. Though CG-SENSE has been shown to work with low-latency using a general purpose graphics processing unit (GPU), to the best of our knowledge, no such effort has been made for radial GRAPPA. radial GRAPPA reconstruction, which is robust even with highly undersampled acquisitions, is not iterative, requiring only significant computation during initial calibration while achieving good image quality for low-latency imaging applications. In this work, we present a very fast, low-latency, reconstruction framework based on a heterogeneous system using multi-core CPUs and GPUs. We demonstrate an implementation of radial GRAPPA that permits reconstruction times on par with or faster than acquisition of highly accelerated datasets in both cardiac and dynamic musculoskeletal imaging scenarios. Acquisition and reconstructions times are reported.

Keywords

Parallel MRI; Radial GRAPPA; gridding; real-time cardiac imaging; parallel processing; GPGPU programming; low-latency image reconstruction

Introduction

Fast imaging is an ever-present goal in MRI as evidenced by the wide variety of acquisition and reconstruction techniques dedicated to increasing efficiency. The application of fast imaging to dynamic imaging scenarios poses additional constraints including the need for high frame rates with real-time display and low-latency reconstruction to adequately visualize rapidly moving structures. For example, within cardiac MRI, real-time imaging is used as an alternative to segmented breath-hold acquisitions on patients with respiratory impediments or children (e.g. (1,2) among many). More generally, real-time imaging is broadly applied to interventional procedures guidance, during which immediate visual feedback for the device operators is required (3). Similarly, within musculoskeletal kinematic MRI, high-resolution imaging of joints in motion is achieved with segmented acquisitions (4,5) though real-time imaging can be used to interactively assess and adapt joint motion (6). Whether to measure cardiac function or valve performance, to guide procedures throughout the body or directly assess joint structure, good quality imaging with high frame rates in conjunction with low-latency reconstruction and real-time display is needed. In addition to real-time display capabilities, faster-than-acquisition reconstructions would also open new avenues in clinical routines by freeing computational resources for sophisticated and time consuming post-processing algorithms that are otherwise too costly to be integrated into clinical applications.

Efforts to increase acquisition speed for real-time imaging are quite diverse. For example, from the pulse sequence perspective, the transition from spoiled gradient echo (SPGR) to steady-state coherent sequences such as balanced steady-state free precession (bSSFP) provided a significant increase in SNR, imaging efficiency, and achievable acquisition speed (7,8). However, limits in gradient performance and peripheral nerve stimulation prevent pulse sequence structures from being executed faster than currently achievable by standard clinical systems. More recently, reducing data requirements via undersampling with and without parallel imaging (pMRI) has shifted the burden from data acquisition to reconstruction (9-11). Generally, pMRI eliminates artifacts caused by significant undersampling by incorporating coil sensitivities into image reconstruction in either the image domain, (i.e. using, for example, sensitivity encoding, SENSE) (12), or in the Fourier domain (i.e. using simultaneous acquisition of spatial harmonics, SMASH, (13) or generalized auto-calibrating partially parallel acquisitions, GRAPPA (14)). To date, SENSE and GRAPPA are the most commonly used Cartesian pMRI reconstruction algorithms employed in an effort to reduce the time required to acquire a complete image.

Both SENSE and GRAPPA and their variants have been adapted for real-time imaging, with methods such as temporal SENSE (TSENSE), UNFOLD-SENSE, and temporal GRAPPA (TGRAPPA), (15-17) though image reconstruction was originally performed offline due to computational requirements. TSENSE and other variants of these techniques, such as hybrid-domain temporal GRAPPA (HTGRAPPA), have now been shown with real-time image reconstruction and interactive acquisition. (18-21). Even though fast reconstruction rates have been reported with Cartesian imaging (e.g. Ref. (21)), the difficulties introduced by the geometry factor losses associated with highly accelerated Cartesian imaging (22)

have prevented the use of more aggressive sub-sampling with acceleration factors $R > 4$ in combination with real-time low-latency image reconstruction.

Radial acquisitions are more tolerant to undersampling due to the highly oversampled information-rich center of k -space. (23,24) Thus, data acquisition can be faster with radial imaging than with conventional Cartesian imaging even without pMRI. However, the reconstruction of non-Cartesian datasets is computationally more demanding. As data do not lie on a Cartesian grid, the fast Fourier transform (FFT) cannot be directly applied. Hence, samples must be first mapped onto a Cartesian grid, a process exemplified by convolution gridding. (25,26) More recently, methods utilizing pMRI principles to calculate the optimum gridding kernels from the acquired data itself, such as self-calibrating GRAPPA operator gridding (SCGROG) have been proposed. (27,28) Both convolution gridding and SC-GROG have been previously implemented to provide low-latency performance (29,30), demonstrating the feasibility of real-time reconstructions with non-Cartesian acquisitions. However, no efforts were made to deal with the effects of undersampling, possibly resulting in significant aliasing or streaking artifacts.

Several pMRI reconstruction strategies for accelerated radial imaging have been proposed to compensate for more significant undersampling. Among these, k -space domain radial GRAPPA (radial GRAPPA), (31,1) and image domain iterative conjugate gradient SENSE (CG-SENSE), (32) have both delivered good image quality with very high acceleration rates. However, the long reconstruction times associated with the significantly increased computational complexity lead to unacceptable latency on conventional computer hardware with non-optimized implementations. More recently, the use of relatively inexpensive high-performance hardware such as graphical processing units (GPUs) has become more common in image reconstruction. For example, CG-SENSE has been implemented specifically for low-latency real-time display using GPUs (33). No such implementation has been shown for radial GRAPPA, a non-iterative technique shown to yield excellent image quality in conjunction with high acceleration rates.

To the best of our knowledge, we present the first implementation of a fast, low-latency reconstruction platform for radial GRAPPA (1) on heterogeneous computer hardware using multi-core CPUs and a general purpose GPU. In addition, the framework designed to utilize both CPU and GPU computation power appropriately is highlighted. The implementation is demonstrated in a cardiac functional imaging as well as kinematic joint imaging. Performance metrics of reconstruction components are reported and discussed.

Methods

Theory

Griswold et al. first introduced radial GRAPPA in 2004 (31). In the original formulation, calibration data were segmented in readout and angular encoding directions to have an over-determined system of equations when solving for the radial GRAPPA weights. Once these weights were calculated for each segmented region, each segment was then individually reconstructed. However, for high acceleration rates, determining a proper segment size was challenging: small segments caused noise enhancement, while large segments caused

blurring. For improved image quality, our implementation was based on the improved through-time radial GRAPPA calibration formulation presented by Seiberlich et al. in 2009 (1). In this formulation of radial GRAPPA, several fully-sampled radial datasets are used to obtain multiple repetitions of each kernel geometry in the undersampled dataset. Then, instead of using a large segment in a given k -space frame, as performed in the original radial GRAPPA method, very small segments (generally in readout direction along which the kernel shape is more uniform) and through-time repetitions (same segment location extracted from multiple consecutive frames) are combined to obtain an over-determined systems of equations to solve for the radial GRAPPA weights. Segments are chosen to be small enough to assume that the undersampling degree and direction are constant in each segment, or equivalently, k -space lines are approximately parallel. The combination of in k -space (segments) and multiple repetitions through-time (consecutive frames) increases accuracy in radial GRAPPA weights, thus increasing overall image quality.

Since we are primarily concerned with computational issues in this study, and since this computational load does not necessarily correspond to the undersampling rate with respect to the Nyquist rate, we have introduced a new metric to characterize these reconstruction processes. Let RR represent the reconstruction subsampling rate for radial GRAPPA,

$RR = \frac{N_{p_{calibration}}}{N_{p_{accelerated}}}$, where $N_{p_{calibration}}$ and $N_{p_{accelerated}}$ represent number of radial projections acquired during the calibration and accelerated scans, respectively. Note that $N_{p_{calibration}}$ also represents the number of projections of the radial GRAPPA output images. The improved radial GRAPPA calibration scheme (combination of in k -space and through-time calibration) is depicted in Figure 1 for a reconstruction rate $RR = 2$, using a GRAPPA kernel size of 2×3 , and a segment size of 4×1 (4 segments in readout direction, one segment in angular direction). The GRAPPA kernel used to estimate the missing target sample (red dot, Fig. 1), is assumed to be uniform within the segment. In this example, each segment provides four equations resulting from the four different GRAPPA kernel positions within the segments (in k -space calibration; Fig. 1a). By combining all the information from the same k -space segment over N through-time calibration (Fig. 1b), an over-determined system with $4N$ equations is obtained and used to determine the GRAPPA kernel for a specific target sample. The same process is repeated for all missing samples that make up the undersampled radial k -space with each sample having its own GRAPPA kernel, making the calculation of missing samples easily parallelizable.

Let N_c represent the number of coils. Considering a GRAPPA kernel size of 2-by-3, the estimation process of a missing sample of coil c , $T_c(k_y + \Delta k_y, k_x)$, can be described as

$$T_c(k_y + \Delta k_y, k_x) = \sum_{i=1}^{N_c} \sum_{b_y} \sum_{b_x} S_i(k_y + b_y, k_x + b_x) W_{c,i}(k_y + \Delta k_y, k_x, b_y, b_x) \quad [1]$$

where $S_i(k_y, k_x)$ represents acquired data in k -space position (k_x, k_y) , $W_{c,i}(k_y + \Delta k_y, k_x, b_y, b_x)$ radial GRAPPA weights to estimate an unacquired sample in k -space position $(k_x, k_y + \Delta k_y)$, with $b_x \in [-101]$ and $b_y \in [0RR]$. Note that both S and T vectors are known in the calibration data set. This over-determined system can be described in matrix form as

$$T_{c,k_y+\Delta k_y,k_x}=S_{k_y,k_x}W_{k_y+\Delta k_y,k_x} \quad [2]$$

Using Eq. 2, $W_{k_y+\Delta k_y,k_x}$ can be calculated by

$$W_{k_y+\Delta k_y,k_x}=pinv\left(S_{k_y,k_x}\right)T_{c,k_y+\Delta k_y,k_x} \quad [3]$$

where “*pinv*” stands for pseudo-inverse operator. Once all the weights for all the missing target samples are computed, undersampled radial k -space data can be filled by applying Eq. 1.

Implementation

Radial GRAPPA calibration and reconstruction processes were decoupled to achieve minimum possible latency. The calibration process during which the GRAPPA kernels are determined was implemented in parallel (multi-threaded) on multi-core CPUs, and, the reconstruction process during which missing samples are calculated was implemented on one or more GPU. This separation was made to best utilize available resources: calibration requires computational power for the pseudoinverse and large amounts of memory with speed being a lower priority, while image reconstruction requires computation speed and high degrees of parallelization which are feasible with the large number of cores in a GPU.

Radial GRAPPA uses a distinct kernel per missing sample, yielding a separate least-squares problem per missing k -space point. Therefore, during both calibration and reconstruction, missing samples were processed as individual operations and executed in parallel, setting up an excellent opportunity for parallelization. Given that the Automatically Tuned Linear Algebra Software (ATLAS) (34) library is thread-safe, no explicit synchronization was required during parallel execution. To avoid race conditions during which multiple threads try accessing the same memory location, a separate set of variables was created per execution thread during calibration. During the estimation of missing samples, no synchronization was required since each missing sample received its own radial GRAPPA kernel.

Calibration and image reconstruction were implemented in an asynchronous manner, where calibration was performed first followed by real-time imaging, although the latter was able to start without the completion of the former. The choice of asynchronous calibration for radial GRAPPA was made to insure smooth and efficient transition between the calibration phase and highly accelerated real-time imaging. For example, during calibration, standard gridding could be enabled for reconstruction with some residual artifacts and streaking and lower temporal resolution. During this time, the GRAPPA kernel weights can be calculated in the background once sufficient calibration data has been collected. At the completion of calibration, the estimation of missing k -space points using the calculated GRAPPA kernel weights can be performed, transitioning smoothly and with no interruption of imaging. Demanding synchronization between the two processes would result in a delay between real-time imaging and the weight calculation, making the process inefficient. Furthermore, any fully sampled data acquired after the end of calibration but before the onset of

accelerated imaging could be incorporated into the calibration dataset and a new set of more stable weights could be generated in the background and incorporated into reconstruction seamlessly upon completion. Finally, the asynchronous nature of the weights calculation makes possible the auto-adaptating of weights with variations in coil sensitivity profiles or due to changes in imaging plane, in a manner similar to that of TGRAPPA (17).

Calibration data composed of multiple fully sampled through-time frames can occupy a significant amount of memory space. For example, a 15-coil calibration dataset for a 144 projection image and 80 calibration frames would require ~400 MB of memory space per slice (144 double oversampled readout points per projection, 8 bytes per complex point). Because of the high memory requirements, the calibration process was best suited for implementation on the multi-core CPUs taking advantage of parallel processing (multi-threading). In addition, radial GRAPPA kernel weights themselves require a large amount of memory: with $RR=9$, a 15-coil set of radial GRAPPA weights occupies ~400 MB of memory per slice ($2 \times 3 \times 15 \times 15$ weights per missing point, $144 \times (9-1)$ missing projections, 8 bytes per complex weight). A standard GPU would be capable at most a few sets of weights. On the other hand, the process of reconstructing missing points requires comparatively little memory and can be easily partitioned. Thus, this process was implemented on a GPU, thereby benefiting from its very high-throughput computation characteristics and its fine-grained parallelism. It is interesting to note that because of the way the reconstruction equations are structured into matrix operations, using higher acceleration rates leads to fewer matrix operations on larger matrices that estimate all missing points simultaneously, which leads to better GPU resource utilization and faster execution.

Since radial data are sampled on a non-Cartesian grid, gridding needs to be included and optimized as part of the low-latency implementation. Conventional convolution gridding (25) is the standard solution for this process, though newer implementations taking advantage of the principles of parallel imaging such as GROG (27,28) and its real-time implementation RTGROG (30) are also available. Here, both conventional convolution gridding and RT-GROG were implemented specifically for the heterogeneous platform. For convolution gridding, the look-up-table (LUT) and coordinates were pre-computed in parallel on the CPUs, while itself gridding was performed in real-time on the GPU. The RT-GROG implementation was based on previous work by Saybasili et al. (30), though ported to the GPU for increased speed. The shifting weights and LUT computations were executed on the CPU, while the gridding was carried out on the GPU. The two processes were asynchronous permitting flexible configuration and a fully auto-calibrated reconstruction scheme. The projections acquired as part of the radial GRAPPA calibration dataset were also used to calibrate initial RT-GROG gridding weights, and these weights were continuously updated during the actual reconstruction. For both conventional gridding and the RT-GROG implementation, a grid-centric approach was used, where each grid point gathered non-Cartesian samples located in kernel radius distance to itself, instead of mapping non-Cartesian samples to grid locations. As each grid point executed the gridding algorithm locally, the output was independent of processing order (i.e. no race condition was present) and thus no synchronization was required. For each grid point, a list of non-Cartesian samples that would be mapped to it was calculated during object creation and passed to the GPU. A GPU thread was associated to each grid point, and gridding was

performed individually for each grid point. Please note that the same RT-GROG weights could be used after the initial calibration (28).

Software: Reconstruction Framework

An external reconstruction framework with multi-slice capabilities was developed to facilitate deployment of fast imaging tools during real-time MRI, as well as associated method for prototyping and debugging. Image reconstruction and data processing was modeled as a pipeline that was configured with a series of *Services*. A pipeline manager object was provided to automatically configure a requested pipeline to reconstruct images, or to process data. Each *Service* had an *execute()* and an optional *executeAsync()* methods that would be run asynchronously to each other in a dedicated portable operating system interface (POSIX) thread during the execution of the pipeline. To enable asynchronous execution on the GPU each *Service* was assigned with two separate CUDA streams. This asynchronous multithreading was provided by the pipeline manager to each *Service* in a transparent manner. A TCP/IP server (*mri_services*) was developed to host the pipeline manager and to serve remote requests accordingly. In addition to Radial GRAPPA and RT-GROG *services* (*RadialGrappaService* and *RTGrogService* respectively), several other *Services* were also developed; such as (i) *FFTOpService*: FFT, FFTshift operations, (ii) *RSSCombineService*: root-sum-of-squares (RSS) combination of all-coil images (and cropping if necessary), (iii) *PermuteDataService*: permute input data. A *ConvolutionGridderService*, was implemented allowing a flexible choice of gridding method during online reconstructions. A 4×4 Kaiser-Bessel kernel was chosen as the gridding kernel. Kernel parameters were selected according to (26).

When the network connection was established between the client (e.g. the MRI scanner or a Matlab client) and the *mri_services*, pipeline initialization information was sent first, followed by each *Service's* configuration parameters (e.g. reconstruction rate, number of projection, etc). After initialization, raw data were sent for image reconstruction/data processing with very low-latency. The pipeline manager's state (its configuration, as well as each *Service's* configuration and variables) was preserved until a reconfiguration request was received from the client. In this way, immunity to data acquisition discontinuities was guaranteed; multiple accelerated data acquisitions after one pMRI calibration scan was possible (e.g. multiple accelerated imaging sessions with different TR/TE values after only one pMRI calibration scan using previously calculated radial GRAPPA weights). Finally, a separate instance of the pipeline manager was assigned to each slice in the acquisition. Thus, the calibration process for each slice was executed in a separate thread, asynchronously to other slices. This setup permitted the use different reconstruction pipelines for different slices if necessary.

To avoid unnecessary CPU to GPU memory transfers, all the variables were kept on the GPU memory after initial transfer of the raw MRI data from CPU memory to the GPU memory at the initialization of the pipeline. Once the final *Service* of the reconstruction pipeline was executed on the GPU, its results were transferred back to CPU memory. A pinned (i.e. non-swappable) memory was used when possible to increase the memory

transfer rates, as instructed in the CUDA programming manual (41). Additionally, shared memory was used on CUDA kernels if possible to increase overall performance (41).

Finally, the client code for *mri_services* was developed for the Siemens imaging platform. The user interface (UI) of an interactive real-time product sequence was modified to easily configure the external reconstruction pipeline while setting the MRI acquisition parameters. Immediately after starting the scan, a network connection was established to the external reconstruction workstation and a remote reconstruction pipeline was configured. Reconstructions were performed online on an external workstation, and the images were sent back to scanner for display. This reconstruction process is depicted in Fig. 2.

The schematic description of the reconstruction pipeline used during accelerated real-time online radial imaging is depicted in Fig. 3. Undersampled radial data were fed to *RadialGrappaService* and missing samples were estimated. *RadialGrappaService*'s fully sampled data output was in native scanner raw data order (projections, coils, samples as dimension ordering). However, *RTGrogService* processed data for each coil independently. To properly convert data between these two *services*, fully sampled data were then permuted and sorted by *PermuteDataService* to improve memory coalescence for faster execution of *RTGrogService* on the GPU: samples to be gridded were sorted while permuting data dimensions in a way to maximize memory read performance on the GPU as described in (41). After gridding the radial data by *RTGrogService*, *FFTopService* and *RSSCombineService* were applied to obtain final combined image.

Hardware

The primary reconstruction hardware in this work was an 8x AMD Opteron 8439 SE (48 cores) workstation with a general purpose GPU (Tesla C1060, NVIDIA). The workstation was equipped with 128 GB of memory and 3 TB of disk space. Network connections were established by a direct gigabit Ethernet connection between the scanner and the workstation. Additionally, offline reconstruction tests were also carried out on an alternative platform consisting of a 64-bit Linux workstation with an Intel Xeon X5660 CPUs (6 cores, 12 threads), 48 GB of RAM and a relatively new GPU (Fermi M2090, NVIDIA). This platform was not available for online imaging experiments and was therefore only used for evaluation of the reconstruction pipeline.

Software

C++ was chosen as the main implementation language. During the development of the reconstruction framework, high-level libraries were avoided maintain simplicity. To achieve the primary goal of low latency, only low-level system libraries were used, which resulted in a portable system, requiring little maintenance. The ATLAS library 3.9.24 (<http://math-atlas.sourceforge.net>) and AMD Core Math Library (ACML) 4.0 (<http://developer.amd.com/libraries/acml/>) were used for matrix operations (e.g. matrix multiplications, matrix inversions). CPU parallelization was achieved using OpenMP (<http://openmp.org>) and native POSIX threads (Pthreads). Pthreads mutex and condition variables were used for thread synchronization and inter-thread communications. The CUDA 4.0 (<http://developer.nvidia.com/cuda-toolkit-40>) toolkit was utilized for GPU programming. All the

binaries were compiled in 64 bits and executed on a Fedora Core 13 64-bit Linux distribution (<http://fedoraproject.org/>) with Linux kernel version 2.6.34.9-69. GNU Compiler Collection (GCC) version 4.4.5 was used to compile GPU codes. The following compiler flags were used for GCC: “-O3 -fomit-frame-pointer -mtune=native -ffast-math -mfpmath=sse -msse3 -fopenmp”. The NVCC compiler provided by the CUDA toolkit was used to compile GPU-related code. Low-level UNIX sockets (TCP/IP) were deployed for network connections between the external reconstructor and the scanner. The CUFFT library was used to perform FFT operations on the GPU.

Data Acquisition and Imaging

Data were acquired on a 1.5 T wide-bore scanner (Espree, Siemens, Erlangen, Germany). The standard 15 channel combined body and spine arrays were employed to acquire data. Following validation using bottle phantoms, real-time images of four healthy subjects were reconstructed online, in real-time. The human subject protocol was approved by the Institutional Review Board and written consent was obtained from all subjects. First, a fully sampled calibration dataset was acquired. Once calibration was completed, accelerated datasets with various acceleration factors $R = \frac{\text{image matrix size}}{N_p}$, where N_p is the number of projections acquired. Data acquisition parameters for each dataset are given below.

Phantom data—Radial acquisition matrix of the calibration data = 128 projections $\times$ 256 readout points with oversampling, radial acquisition matrix of the undersampled data = 16 projections $\times$ 256 readout points with oversampling (reconstruction rate RR = 8, acceleration factor R = 8), reconstructed image matrix size = 128 $\times$ 128, TR = 2.64 ms, TE = 1.32 ms, readout bandwidth (BW) = 558 Hz/px, FOV = 400 mm, slice thickness = 10 mm, flip angle 70 degrees, sequence: steady-state free precession (SSFP), number of coils = 15. Additional phantom data were acquired using 32 acquisition coils with the same imaging parameters for offline test purposes.

Cardiac data—Two different cardiac datasets at different imaging resolutions were acquired. *Standard resolution*: Radial acquisition matrix of the calibration data = 144 projections $\times$ 256 readout points with oversampling, radial acquisition matrix of the undersampled data = 24/16 projections $\times$ 256 readout points with oversampling (reconstruction rate RR = 6/9, and acceleration rate R = 5.3/8), image matrix size = 128 $\times$ 128, TR = 2.64 ms, TE = 1.32 ms, BW = 1115 Hz/px, FOV = 200-300 mm, slice thickness = 8 mm, flip angle = 70 degrees, sequence: steady-state free precession (SSFP), number of coils = 15. *High Resolution*: Radial acquisition matrix of the calibration data = 192 projection $\times$ 384 readout points with oversampling, radial acquisition matrix of the undersampled data = 24 projections $\times$ 384 readout points with oversampling (reconstruction rate RR = 8, acceleration rate R=8), image matrix size = 192 $\times$ 192, TR = 3.2 ms, TE = 1.6 ms, BW = 1115 Hz/px, FOV = 300 mm, slice thickness = 8 mm, flip angle = 70 degrees, sequence: steady-state free precession (SSFP), number of coils = 15.

Ankle data—Radial acquisition matrix of the calibration data = 144 projections $\times$ 512 readout points with oversampling, radial acquisition of the undersampled data = 24 projections, $\times$ 512 readout points with oversampling (reconstruction rate RR=6, acceleration

factor $R=10.6$) image matrix size = 256×256 , TR = 11.28 ms, BW = 685 Hz/px, FOV = 210 mm, slice thickness = 5 mm, flip angle = 15 degrees, sequence: spoiled gradient echo (GRE), number of coils = 15.

RESULTS

Our implementation was validated on a 15 coil, $RR/R=8/8$ ($128/8 = 16$ projections, 256 readout points with oversampling) cylindrical phantom dataset. First, the numerical correctness was tested: a calibration frame image was directly compared to the $RR/R=8/8$ accelerated frame image in term of root-mean-square-error (RMSE). RMSE was calculated as 1.46%, which was comparable to the results presented by Seiberlich et al. in (1). After the numerical validation, three cylindrical phantoms were used to validate our implementation on moving objects; two phantoms were placed vertically, while the third one was placed horizontally between the other two. An oblique scan plane was prescribed to visualize the middle phantom entirely. While scanning, the middle phantom was manually moved out of the imaging plane. Real-time, online reconstructed images were able to visualize the movement of the phantom with great clarity. Multiple consecutive imaging frames, depicting the middle phantom moving out of the imaging plane, are given in Fig. 4.

For *in vivo* validation, ungated, free-breathing short-axis cardiac data were acquired using 15 coils were reconstructed online, in real-time. A full FOV still frame, and representative, cropped images series during different cardiac cycles for three datasets are presented in Fig. 5. Fig 5a displays accelerated images acquired with $RR/R=6/5.3$ and 144 projections with 256 readout points including oversampling, The FOV was 200 mm and the 24 projections acquired were reconstructed to a 128×128 matrix size faster than they were acquired (39.5 ms vs. 62 ms). Fig 5b displays images acquired with $RR/R=8/8$ (192 projections, 384 readout points, 300 mm FOV, image matrix size 192×192) which were acquired in 72 ms (24 projections) and reconstructed in 67.2 ms. Finally, Fig 5c displays images acquired with $RR/R=9/8$ (144 projections, 256 readout points, 300 mm FOV image matrix size 128×128) using 16 projections which were acquired in 42.2 ms and reconstructed faster in 35.2 ms. In addition to the above data sets, several high resolution datasets were acquired from a healthy volunteer's ankle using a GRE sequence with $RR/R=6/10.6$ ($144/6 = 24$ projections). Images were reconstructed online, in real-time on our external workstation. For each case, calibration data were acquired with the subject in motion. A series of real-time reconstructed consecutive frames are presented in Fig. 6, along with data acquisition and reconstruction times.

Calibration and reconstruction performance of the phantom and cardiac datasets is depicted in Fig. 7 for varying number of fully sampled frames used in calibration and reconstruction rates. Data acquisition times are also given in the reconstruction plots to better represent the capabilities of our software implementation on our Linux workstation. Reconstruction plots represent full reconstruction times, including CPU/GPU memory transfers. Please note that $RR=12$ ($144/12 = 12$ projections) and $RR=16$ ($144/16 = 9$ projections and $128/16 = 8$ projections, respectively) data sets were synthetically created to give an overall idea about execution performance with higher rates. Individual reconstruction *Service* performances for

different radial GRAPPA reconstruction rates are depicted in Table 1 for various imaging parameters.

DISCUSSION

The use of external reconstruction platforms in MRI has significant history, (35-40) with each newer platform incorporating faster, more powerful computer hardware and resulting in faster image reconstruction and display. More complex algorithms are now feasible for real-time imaging as shown here. Furthermore, the use of GPU hardware which has been made readily available and inexpensive through video display and gaming, is now almost a certainty in MR image reconstruction. Achieving the same number of cores on purely CPU-based systems is simply economically unfeasible. Hence, generation of reconstruction platforms based on heterogeneous system is an active area of research ready for exploration. This implementation represents an efficient option aimed at maximum and flexible utilization of hardware and targeted at minimum image display latency. Despite the fact that our hardware system was four years old, we managed to reconstruct images acquired in 42 ms in only 35.2 ms. Other frameworks (e.g. Ref (40)), have also taken similar approaches, albeit with the use of higher level libraries that provide more extensive functionality but may limit portability over time. Additionally, the flexibility incorporated into this reconstruction framework could easily be deployed in reconstruction of Cartesian datasets or data acquired with other non-Cartesian imaging sequences. Finally, reconstructions that outpace acquisition open the doors for significant post-processing, especially when using newer hardware with faster CPUs and more powerful GPUs. This additional speed could easily be used to maintain real-time display and more involved rendering or for more involved post-processing such as required for phase contrast imaging as used in real-time flow evaluation.

Faster-than-acquisition reconstructions are beneficial in many ways. MRI guided interventional procedures require low-latency real-time reconstructions to provide proper visual feedback to the operator. If the image reconstruction cannot keep up with the data acquisition, image display latency will increase and impair the visual guidance. That may lead to severe complications; even jeopardize the patient's life. The benefits of a faster-than-acquisition reconstructions for general imaging applications -that do not necessitate low-latency real-time image display, would be two-fold. First, immediate feedback would allow detecting imaging problems on the fly. That way, MRI technician could re-run the same study with correct protocol/imaging parameters without losing valuable scan time. If the reconstruction were very slow, it might be even necessary to reschedule the patient for a future date that would imply repositioning the patient and relocalization of the slice planes of interest. Detecting imaging errors in real-time would improve diagnostic efficiency while decreasing costs. Second, fast reconstructions would free computational resources and valuable time for the clinical integration of sophisticated postprocessing algorithms that are omitted from clinical routines due to strict time constraints. Our implementation was optimized for higher radial GRAPPA reconstruction rates by improving computer resource utilization with increased rate of acceleration: weights calculation for (RR-1) missing samples in a GRAPPA block and their estimation during reconstruction were computed at once. As lower reconstruction rates required more matrix operations on smaller matrices during both calibration and reconstruction, system load was non-optimal due to lower

resource utilization. On the other hand, higher reconstruction rates used fewer matrix operations on bigger matrices, which leads to better resource utilization, and more optimal system loading. Hence, high performance was guaranteed for highly accelerated radial imaging. Performance results justified that both reconstruction and calibration performances were better for higher reconstruction rates (Fig. 7).

The small FOV short axis images represented in Fig. 5a clearly depicts that radial GRAPPA could reconstruct data sets where the object of interest is much bigger than the selected FOV. Since radial GRAPPA is a k -space based technique and folding-over is only observed in the image domain, no fold-over or streaking artifacts can be seen. Additionally, a change in the object of interest (e.g. phantom moving out of the imaging plane depicted in Figure 4) would not require a re-calibration for radial GRAPPA, since both the calibration and the reconstruction processes are based solely on coil sensitivity profiles: no re-calibration would be required unless there is a change in coil sensitivities (e.g. change in the imaging plane). As can be seen in Figure 4, even though the objects to be imaged were drastically changed when the middle phantom was moved out of the imaging plane, radial GRAPPA could still reconstruct images since the coil sensitivity profiles were remained unchanged.

CPU reconstruction times were not deeply investigated in this work since the aim was to provide low-latency real-time reconstructions. CPU performances for RT-GROG previously presented in (30) clearly indicates that a low-latency reconstruction with high acceleration rates would be impossible on the CPU since just the gridding would take longer than the data acquisitions during an accelerated scan. A CPU version of the radial GRAPPA weight estimation process (executed in parallel using all 48 cores) was found to be approximately five times slower on our workstation when compared to the GPU implementation. Please note that even though this performance was not sufficient for low-latency imaging, it was still adequate for offline reconstructions that normally took about a second per frame in MATLAB.

For GPU implementations of both convolution gridding and RT-GROG, a grid-centric gridding approach was preferred to avoid race conditions that could result in outputs which are dependent on execution order. Gridding maps that contained the list of non-Cartesian samples assigned to each grid point were calculated at the time the object was created, requiring approximately 120 ms. As the data acquisition for a fully sampled calibration frame was always >300 ms, and since the reconstruction of a calibration frame involved only gridding (which was on the order of tens of milliseconds), the calculation of the gridding maps did not result in performance penalties, nor any additional latency. Since the raw data matrix sizes of the calibration data and the output of radial GRAPPA were the same (as were the k -space coordinates of each sample), these maps were also used during accelerated imaging. Though this approach worked sufficiently well for this application, it would be unfeasible in current acquisitions involving golden angle radial imaging, where the trajectories changes from frame to frame, and which would require the calculation of one map per frame. For such a case, a non-Cartesian sample based gridding approach for the GPU should be used, as described in (29), or a separate gridder could be assigned to each golden-angle radial spoke. However, since radial GRAPPA requires fixed k -space

coordinates for all the frames during data acquisition, only one gridding map was needed throughout a study.

The performance of both the RT-GROG and convolution gridding were compared with convolution gridding completing execution approximately 2.5 ms faster than RT-GROG when mapping non-Cartesian samples to Cartesian grid locations on a 15-coil radial data set with radial acquisition matrix of 144 projections and 256 readout samples. This performance difference was due to the increased memory access on the GPU when utilizing RT-GROG, which requires access to all coil data to generate each individual coil's gridded samples while convolution gridding does not have inter-coil interactions. In general, this performance loss was considered negligible in this implementation, especially when considering the advantages of RT-GROG (i.e. self-calibration that provides optimum gridding weights, trivial density compensation). However, with larger datasets acquired with more coils, the disparity in performance would increase: RTGROG would use 32 coefficients per sample, while conventional gridding would only use 16 coefficients (4×4 kernel size) per sample during gridding. Hence, these results suggest that for datasets with more coils, standard convolution gridding is preferable. High-level libraries were avoided during the development of our multi-threaded reconstruction framework to improve code portability and to the reach minimum latency possible. Though the reconstruction environment was developed for the Unix environment, it could be ported to any operating system with little modification. Additionally, this framework could be easily extended to provide support for multi-GPU systems, where each GPU would have its own reconstruction pipeline. Finally, using this framework in a distributed environment (e.g. a cluster of nodes connected via network), each node could be configured to have its own reconstruction pipeline per GPU and could perform partial reconstructions, which could then be combined to obtain the final image.

Direct gigabit Ethernet connection provided nearly seamless data transfer performance. While transferring 15 coil calibration frames (acquisition times ~ 380 ms), data transfer latencies were approximately 30 ms between the scanner and our external reconstruction workstation. Data transfer latency was linearly proportional to the reconstruction rate: one frame raw data transfer between the scanner and the external workstation of a $RR/R=9/8$ dataset (with 16 projections and 256 samples) took approximately 3.5 ms. As the cropped RSS combined images were much smaller in size (single precision real-valued images) compared to raw MRI data (multi-coil, single precision complex-valued data), return image transfer latencies were negligible (less than 1 ms). The gigabit network connection permitted a very low-latency data transfers, though if necessary, the connection could be sped up nearly 10-fold with already-available 10 gigabit Ethernet hardware. However, in the current scanner configuration this was not allowed due to lack of hardware support. As future scanners are likely to provide this connection support, latency due to data transfer will ultimately be negligible or at least outpace increases in data size to the use of more receive coils.

For the images shown here, we used RT-GROG for gridding as it is an auto-calibrated, parameter-free process with straightforward density compensation. However, the reconstruction pipeline allowed for seamless switching between RT-GROG and convolution

gridding during real-time imaging. Radial GRAPPA calibration data were also used to calculate RT-GROG gridding weights. Both calibration processes ran on separate threads (asynchronously), and RTGROG calibration was significantly faster than radial GRAPPA calibration (<200 ms), and added virtually no performance penalties to the overall calibration process. Given that the difference in time between RT-GROG and conventional gridding is only a small fraction of the total reconstruction time, we believe that both methods are viable options for real-time imaging and display.

The CPUs and GPUs of our external reconstruction workstation were approximately 3 years old at the time of this work. We had the opportunity to test our implementation (offline) on a newer 64 bit Linux workstation with an Intel Xeon X5660 CPUs (6 cores, 12 threads), 48 GB of RAM and a NVIDIA Fermi M2090 GPU. While the calibration performance was similar to the results seen in this work, the image reconstruction was significantly faster (up to 2.6×). The 15 coil, RR/R=9/8 (16 projections) radial cardiac data seen in Figure 5c were reconstructed in 12.5 ms, compared to 35.2 ms (full reconstruction time of a RSS combined image, including CPU/GPU data transfers), which was 3.4× faster than the data acquisition (42.2 ms). These results indicate that our results are reproducible on a modest (cheaper) modern workstation. The older online workstation could not match the acquisition speeds when reconstructing 32 coil datasets *(RR/R = 8/8, 16 projections acquisition time 42.2 ms per frame), as the reconstruction of a single image took 127.75 ms (on average). However, our test results on the newer workstation (described above) were promising: the same 32-coil data set was reconstructed in 42.1 ms per frame (on average). Though a performance increase with this updated hardware was achieved, it was not available for real-time reconstructions on the MR system.

CONCLUSION

In this work, a fast, parallelized radial GRAPPA implementation on a heterogeneous hardware platform (multiple multi-core CPUs and a GPU) designed for highly accelerated real-time, online, low-latency MRI applications was presented. The radial GRAPPA method was decoupled into reconstruction and calibration processes to best utilize hardware and achieve high performance. The GPU was used preferably for low-latency computation during image reconstruction, while the CPUs were utilized for the memory-intensive calibration process. To support real-time image display, a generic and highly flexible multi-threaded framework was also developed, yielding low-latency image reconstruction. Both bSSFP and SPGR sequences were tested during dynamic imaging of the heart and foot, respectively. The reconstruction framework was able to generate images more rapidly than data acquisition, even when collecting data with a high temporal resolution of 35 ms.

Supplementary Material

Refer to Web version on PubMed Central for supplementary material.

REFERENCES

1. Seiberlich N, Ehse P, Duerk J, Gilkeson R, Griswold MA. Improved radial GRAPPA calibration for real-time free-breathing cardiac imaging. *Magn Reson Med*. 2011; 65:492–505. [PubMed: 20872865]
2. Seiberlich N, Lee G, Ehse P, Duerk JL, Gilkeson R, Griswold M. Improved temporal resolution in cardiac imaging using through-time spiral GRAPPA. *Magn Reson Med*. 2011; 66(6):1682–8. [PubMed: 21523823]
3. Kahlert P, Parohl N, Albert J, Schafer L, Reinhardt R, Kaiser GM, McDougall I, Decker B, Plicht B, Erbel R, Eggebrecht H, Ladd ME, Quick HH. Towards real-time cardiovascular magnetic resonance guided transarterial corevalve implantation: in vivo evaluation in swine. *J Card Magn Reson*. 2012; 14:21.
4. Sheehan FT, Zajac FE, Drace JE. Using cine phase contrast magnetic resonance imaging to non-invasively study in vivo knee dynamics. *J Biomech*. 1997; 31(1):21–26. [PubMed: 9596534] N4
Seisler AR, Sheehan FT. Normative three-dimensional patellofemoral and tibiofemoral kinematics: a dynamic, in vivo study. *IEEE Trans Biomed Eng*. 2007; 54(7):1333–41. [PubMed: 17605365]
5. Sheehan FT. The finite helical axis of the knee joint (a non-invasive in vivo study using fast-PC MRI). *J Biomech*. 2007; 40(5):1038–47. [PubMed: 17141789]
6. Sayin O, Saybasili H, Guo L, Carrino JA, Sheehan FT, Griswold MA, Seiberlich N, Herzka DA. Flexible real-time imaging of highly-dynamic knee joint motion. *Proceedings of ISMRM*. 2013 #3483.
7. Carr JC, Simonetti O, Bundy J, Li D, Pereles S, Finn P. Cine MR angiography of the heart with segmented true fast imaging with steady-state precession. *Radiology*. 2001; 219:828–834. [PubMed: 11376278]
8. Derakhshan, Griswold M, Nour GS, Sunshine JL, Duerk JL. Resolution enhanced T1-insensitive steady-state imaging. *Magn Reson Med*. 2012; 68(2):421–9. [PubMed: 22131264]
9. Hutchinson M, Raff U. Fast mri data acquisition using multiple detectors. *Magn Reson Med*. 1998; 6:87–91. [PubMed: 3352509]
10. Roemer PB, Edelstein WA, Hayes CE, Souza SP, Mueller OM. The NMR phased array. *Magn Reson Med*. 1990; 16:192–225. [PubMed: 2266841]
11. Carlson JW, Minemura T. Imaging time reduction through multiple receiver coil data acquisition and image reconstruction. *Magn Reson Med*. 1993; 29:681–687. [PubMed: 8505905]
12. Pruessmann KP, Weiger M, Scheidegger MB, Boesiger P. SENSE: Sensitivity encoding for fast MRI. *Magn Reson Med*. 1999; 42(5):952–962. [PubMed: 10542355]
13. Sodickson DK, Manning WJ. Simultaneous acquisition of spatial harmonics (SMASH): fast imaging with radiofrequency coil arrays. *Magn Reson Med*. 1997; 38:591–603. [PubMed: 9324327]
14. Griswold MA, Jakob PM, Heidemann RM, Nittka M, Jellus V, Wang K, Kiefer B, Haase A. Generalized autocalibrating partially parallel acquisitions (GRAPPA). *Magn Reson Med*. 2002; 47:1202–1210. [PubMed: 12111967]
15. Kellman P, Epstein FH, McVeigh ER. Adaptive sensitivity encoding incorporating temporal filtering (TSENSE). *Magn Reson Med*. 2001; 45:846–852. [PubMed: 11323811]
16. Madore B. UNFOLD-SENSE: A Parallel MRI Method with Self-Calibration and Artifact Suppression. *Magn Reson Med*. 2004; 52:310–320. [PubMed: 15282813]
17. Breuer FA, Kellman P, Griswold MA, Jakob PM. Dynamic autocalibrated parallel imaging using temporal GRAPPA (TGRAPPA). *Magn Reson Med*. 2005; 53:981–5. [PubMed: 15799044]
18. Bock M, Muller S, Zuehlsdorff S, Speier P, Fink C, Hallscheidt P, Umthum R, Semmler Wolfhard. Active catheter tracking using parallel MRI and real-time image reconstruction. *Magn Reson Med*. 2006; 55(6):1454–1459. [PubMed: 16683261]
19. Guttman MA, Kellman P, Dick AJ, Lederman RJ, McVeigh ER. Realtime accelerated interactive MRI with adaptive TSENSE and UNFOLD. *Magn Reson Med*. 2003; 50:315–321.10. [PubMed: 12876708]

20. Roujol S, de Senneville BD, Vahala E, Sørensen TS, Moonen C, Ries M. Online real-time reconstruction of adaptive TSENSE with commodity CPU/GPU hardware. *Magn Reson Med*. Dec; 2009 62(6):1658–64. [PubMed: 19902515]
21. Saybasili H, Kellman P, Griswold MA, Derbyshire JA, Guttman MA. HTGRAPPA: Real-time B1-weighted image domain TGRAPPA reconstruction. *Magn Reson Med*. 2009; 61:1425–1433. [PubMed: 19353673]
22. Wiesinger F, Boesiger P, Pruessmann KP. Electrodynamics and ultimate SNR in parallel MR imaging. *Magn Reson Med*. 2004; 52(2):376–390. [PubMed: 15282821]
23. Wild JM, Paley MNJ, Kasuboski L, Swift A, Fichele S, Woodhouse N, Griffiths PD, van Beek E. Dynamic radial projection MRI of inhaled hyperpolarized ^3He Gas. *Magn Reson Med*. 2003; 49:991–997. [PubMed: 12768575]
24. Larson AC, White RD, Laub G, McVeigh ER, Li D, Simonetti OP. Self-gated cardiac Cine MRI. *Magn Reson Med*. 2004; 51:93–102. [PubMed: 14705049]
25. O'Sullivan JD. Fast sinc function gridding algorithm for fourier inversion in computer tomography. *IEEE Trans Med Imaging*. 1985; 4:200–207. [PubMed: 18243972]
26. Jackson JI, Meyer CH, Nishimura DG, Macovski A. Selection of a convolution function for fourier inversion using gridding [computerised tomography application]. *IEEE Trans Med Imaging*. 1991; 10:473–478. [PubMed: 18222850]
27. Seiberlich N, Breuer F, Blaimer M, Barkauskas K, Jakob PM, Griswold MA. Non-Cartesian data reconstruction using GRAPPA operator gridding (GROG). *Magn Reson Med*. 2007; 58(6):1257–65. [PubMed: 17969027]
28. Seiberlich N, Breuer F, Blaimer M, Jakob PM, Griswold MA. Self-calibrating GRAPPA operator gridding for radial and spiral trajectories. *Magn Reson Med*. 2008; 59(4):930–935. [PubMed: 18383296]
29. Sørensen TS, Schaeffter T, Noe KØ, Hansen MS. Accelerating the nonequispaced fast Fourier transform on commodity graphics hardware. *IEEE Trans Med Imag*. 2008; 27:538–547.
30. Saybasili H, Derbyshire JA, Kellman P, Griswold MA, Ozturk C, Lederman RJ, Seiberlich N. RT-GROG: Parallelized self-calibrating GROG for real-time MRI. *Magn Reson Med*. 2010; 64:306–312. [PubMed: 20577983]
31. Griswold MA, Blaimer M, Heidemann RM, Speier P, Kannengeiser S, Nittka M, Breuer F, Mueller M, Jakob PM. Rapid evaluation of cardiac function using undersampled radial TrueFISP with GRAPPA. *Proc 12th Annual Meeting of the ISMRM, Kyoto, Japan*. 2004:737.
32. Pruessmann KP, Weiger M, Bornert P, Boesiger P. Advances in sensitivity encoding with arbitrary k -space trajectories. *Magn Reson Med*. 2001; 46:638–651. [PubMed: 11590639]
33. Sørensen TS, Atkinson D, Schaeffter T, Hansen MS. Real-time reconstruction of sensitivity encoded radial magnetic resonance imaging using a graphics processing unit. *IEEE Trans Med Imag*. 2009; 28:1974–1985.
34. Dongarra, J.; Petite, A.; Whaley, R. Automatic tuned linear algebra software. 2012. <http://math-atlas.sourceforge.net>. Accessed date
35. Kerr AB, Pauly JM, Hu BS, Li KC, Hardy CJ, Meyer CH, Macovski A, Nishimura DG. Real-time interactive MRI on a conventional scanner. *Magn Reson Med*. Sep; 1997 38(3):355–67. (Erratum in *Magn Reson Med* 1998 Dec;40(6):952-5). [PubMed: 9339436]
36. Guttman MA, Lederman RJ, Sorger JM, McVeigh ER. Real-time volume rendered MRI for interventional guidance. *J Card Magn Reson*. 2002; 4:431–442.
37. Santos JM, Wright GA, Pauly JM. Flexible real-time magnetic resonance imaging framework. In *Proc IEEE Eng Med Biol Soc*. 2004; 2:1048–51.
38. Schiwietz T, Chang T, Speier P, Westermann R. MR image reconstruction using the GPU. In *Proc SPIE*. 2006; 6142:1279–1290.
39. Uecker M, Zhang S, Frahm J. Nonlinear inverse reconstruction for real-time MRI of the human heart using undersampled radial FLASH. *Magn Reson Med*. Jun. 2010; 63(6):1456–62. [PubMed: 20512847]
40. Hansen MS, Sørensen TS. Gadgetron: An open source framework for medical image reconstruction. *Magn Reson Med*. Jul 12.2012 doi: 10.1002/mrm.24389. [Epub ahead of print].

41. Cuda, C. [July 2013] Programming Guide Version 4.0. http://docs.nvidia.com/cuda/pdf/CUDA_C_Programming_Guide.pdf. Published: July 2013.

FIGURE 1.

Depiction of the through-time radial GRAPPA calibration process for a GRAPPA kernel size of 2×3 (reconstruction rate $RR=2$), and a segment size of 4×1 (4 kernel repetitions in readout direction, one in projection direction). The GRAPPA kernel is assumed to be uniform in this small segment. (a) During calibration, the GRAPPA kernel is translated within the segment to provide four equations. (b) The same segment location repeated through N calibration data frames is used to obtain an over-determined system ($4 \times N$ equations) to solve for the weights for this specific geometry (Eq. [1]).

FIGURE 2.

The real-time, online reconstruction process is presented above. When the “scan” button is pressed on the user interface of the scanner, a connection to the external reconstruction workstation is established. The external reconstruction pipeline configuration is obtained from the sequence UI. If the pipeline was not previously configured, or a re-configuration request is submitted, configuration information is sent to the external reconstruction service for pipeline initialization. After configuration, raw MR data are sent (online, in real-time) and images are reconstructed externally with very low-latency. In the “*MRI_SERVICES*” frame, light gray boxes depict jobs performed by “*external pipeline manager*”. Once the raw data arrives, *MRI_SERVICES* passes it to the *pipeline manager*, which will start the reconstruction chain. Each service in the pipeline passes its results to the next service while asynchronous calculations are performed in the background on a separate thread (if any). The output of the last service provides the final image that is sent back to the scanner for display.

FIGURE 3.

The default external reconstruction pipeline is represented. Incoming raw data frames are first passed through the radial GRAPPA reconstruction process. If this is a fully-sampled data set, it is passed to the asynchronous calibration thread. Otherwise, k -space is filled up by estimating missing samples of the raw data. Second, fully sampled k -space data (output of radial GRAPPA) are permuted and sorted before gridding. The RT-GROG service applies gridding, and updates its gridding weights in the background asynchronously. Convolution gridding can alternatively replace the RT-GROG block. Gridded data are then transferred to the image domain by FFT service, which will be cropped and combined by the last service in the chain. The combined image is then sent to the scanner for display. Gray boxes depict implemented *Services*.

FIGURE 4.

Phantom imaging validation using 15 coils, $RR/R=8/8$ (16 projections) data demonstrated good image quality with low latency. The bottles and the structure of the phantoms provide the sharp edges and clear structures necessary to spot reconstruction errors and any induced artifacts. Three cylindrical phantoms were deployed: two placed facing up, and one placed longitudinally between them. Following the calibration, middle phantom was manually moved out of the imaging plane during accelerated imaging. All the images were reconstructed online, in real-time with radial GRAPPA. This image series were acquired in 1774 ms (one accelerated frame acquisition time = 42 ms, 42 frames in total). Movement of the phantom was clearly observed during real-time imaging experiment. Please note that the artifacts on the side phantoms were due to SSFP banding and field changes.

FIGURE 5.

Cardiac images with different imaging parameters (e.g. FOV, image matrix and acceleration rates) were reconstructed in real time. For each case, a still frame and a series of images depicting a cardiac cycle are given. Note that in (c) every other frame is shown as the frame rate is too fast to span a full cycle in 8 frames.

FIGURE 6.

Real-time (online) reconstructed GRE images of the ankle depict good clarity of the joint in motion. (a) Ankle and (b) forefoot images reconstructed from 24 projections (144 projections $\times$ 512 readout points with oversampling reconstruction image matrix size=256 $\times$ 256, RR/R = 6/10) acquired in 270.7 ms and reconstructed in 90.6 ms.

FIGURE 7.

Performance plots for radial GRAPPA calibration, and reconstruction (from receiving the undersampled data to final combined image, including CPU/GPU memory transfers) versus acquisition (for accelerated scans). Lower reconstruction rates required more matrix operations on smaller matrices during both calibration and reconstruction, and therefore system load was non-optimal due to lower resource utilization. Higher reconstruction rates used fewer matrix operations on bigger matrices, which lead to better resource utilization, and more optimal system loading. That way, better performance on higher reconstruction rates was guaranteed.

(a) (Top) Radial GRAPPA weights calculation on the phantom data (15 coils, radial acquisition matrix of the calibration data: 128 projections $\times$ 256 readout samples with oversampling), image matrix size=128 $\times$ 128, $RR / R \in [2 / 2, 4 / 4, 8 / 8, 16 / 16]$ (64, 32, 16, and 8 projections, respectively), number of repetitions $nrep \in [30 \ 40 \ 50 \ 60]$. (Bottom) Image reconstruction performances on the same data sets. (b) (Top) Radial GRAPPA weights calculation on a cardiac dataset (15 coils, radial acquisition matrix of the calibration data: 144 projections $\times$ 256 readout samples with oversampling), image matrix size=128 $\times$ 128, $RR / R \in [2 / 1.8, 3 / 2.7, 4 / 3.5, 6 / 5.3, 8 / 7.1, 9 / 8, 12 / 10.6, 16 / 14.2]$, (72, 48, 36, 24, 18, 16, 12 and 9 projections, respectively) number of repetitions $nrep \in [30 \ 40 \ 50 \ 60 \ 70 \ 80]$. (Bottom) Image reconstruction performances on the same data sets.

TABLE 1

Execution performances in milliseconds for individual *Services* of the reconstruction pipeline.

Service	Reconstruction Performance (in milliseconds) Imaging Parameters (Radial acquisition matrices for calibration data, and various reconstruction rates)								
	<i>128×256</i>			<i>144×256</i>					
	<i>RR=2 (64p)</i>	<i>RR=4 (32p)</i>	<i>RR=8 (16p)</i>	<i>RR=2 (72p)</i>	<i>RR=3 (48p)</i>	<i>RR=4 (36p)</i>	<i>RR=6 (24p)</i>	<i>RR=8 (18p)</i>	<i>RR=9 (16p)</i>
RadialGRAPPA	34.0	18.8	10.9	38.2	26.9	21.1	15.2	12.1	11.1
Permute&Sort		5.60					6.31		
RTGROG		13.97					15.10		
FFT Operations		1.13					1.13		
RSS&Crop		0.12					0.12		

A 15 coil data sets with 256 readout points for different reconstruction rates was used for these measurements. For each RR, the corresponding number of acquired projections (*p*) are also presented. Please note that only radial GRAPPA *Service* performance depends on the reconstruction rate.