ADVANCED COMPUTATIONAL TOOLS FOR REAL-TIME MR IMAGING

by

Haris Saybaşılı

B.Sc., Computer Engineering, Galatasaray University, 2001

M.Sc., Computer Engineering, University of Nice Sophia Antipolis, 2002

Submitted to the Institute of Biomedical Engineering

in partial fulfillment of the requirements

for the degree of

Doctor

of

Philosophy

Boğaziçi University

September 2009

ADVANCED COMPUTATIONAL TOOLS FOR REAL-TIME MR IMAGING

APPROVED BY:

Prof. Dr. Ahmet Ademoğlu .....
(Thesis Advisor)

Doç. Dr. Cengizhan Öztürk .....
(Thesis Co-advisor)

Prof. Dr. Mehmed Özkan .....

Doç. Dr. Can Özturan .....

Yard. Doç. Dr. Daniel Herzka .....

Yard. Doç. Dr. Burak Acar .....

DATE OF APPROVAL: 28.09.2009

ACKNOWLEDGMENTS

I would like to thank Prof. Elliot McVeigh for giving me the opportunity to do research on his Medical Imaging laboratory. I would also like to thank Dr. Andrew Derbyshire, Michael Guttman and Dr. Peter Kellman with whom I have had the pleasure of working, and who have provided much enthusiasm, insight and assistance during my projects. Thanks are also due to Dr. Robert Lederman, who sponsored me for the last one and a half years.

I also thank my wife Beliz, my sister Assist. Prof. Nermin Saybaşılı, my mother Figen Saybaşılı, my parents-in-law Şükran Şenyüz and Prof. Dr. Osman Faruk Şenyüz for their continuing support and advice. Similarly, I thank my brother-in-law Orkun Şenyüz and all my friends, especially Emre Sağlam for their support, and for providing much fun.

Speacial thanks to my beloved father, Prof. Kemali Saybaşılı, who is eternally missed but never forgotten. He always supported me, and guided me in every aspect of my life. He would be more than happy to witness accomplishment of my Ph.D.

Thanks are due to Assoc. Prof. Can Özturan who guided me on many computational problems. Finally I wish to thank Prof. Ahmet Ademoğlu for accepting me to the Ph.D. programme, and to thank Assoc. Prof. Cengizhan Öztürk who directed me to real-time MRI research. I thank both to have provided their support and their guidance during my Ph.D. research.

Many thanks...

ABSTRACT

ADVANCED COMPUTATIONAL TOOLS FOR REAL-TIME MR IMAGING

Real-time Magnetic Resonance Imaging (MRI) has the potential of successfully guiding interventional applications. Overall, the requirements of real-time MRI can be categorized as: (i) fast data acquisition, (ii) fast image reconstruction, and (iii) good image quality.

Fast data acquisition is provided by optimized real-time sequences, by parallel MRI (pMRI) techniques, or by non-Cartesian acquisition schemes (e.g. spiral and radial trajectories). However, fast image reconstruction is non-trivial, especially when computations demanding pMRI methods or non-Cartesian trajectories are involved. Even though signal-to-noise ratio (SNR) can be relatively high during real-time imaging, spatial resolution is limited. Thus, improved visual feedback during real-time MRI guided interventions is a must. This thesis defined three specific aims to improve real-time imaging: (i) real-time image reconstruction for pMRI, (ii) real-time image reconstruction for non-Cartesian trajectories, and (iii) fast MRI post-processing for improved visual feedback during interventions.

Thesis contributions include: (i) real-time hybrid domain TGRAPPA based pMRI reconstruction algorithm (currently the fastest TGRAPPA based algorithm), (ii) first real-time implementation of GRAPPA Operator Gridding algorithm for radial acquisitions, (iii) multi-phase *3D* angiography roadmaps for MRI guided interventions, (iv) improved active device visualization during real-time MRI guided interventions, (v) integration of a real-time active device localizer algorithm.

Keywords: Real-time MRI, pMRI, Non-cartesian trajectories, Parallel processing, MRI guided interventions.

ÖZET

GERÇEK ZAMANLI MR GÖRÜNTÜLEME İÇİN GELİŞMİŞ YAZILIM ARAÇLARI

Gerçek zamanlı Manyetik Rezonans Görüntüleme (MRG) girişimsel uygulamalara kılavuzluk edebilme potansiyeline sahiptir. Genel olarak, gerçek zamanlı MRG'nin gereksinimleri şu şekilde sıralanabilir: (i) hızlı veri alımı, (ii) hızlı görüntüleme, ve (iii) iyi bir görüntü kalitesi sunma.

Hızlı veri alımı eniyilenmiş gerçek zamanlı atım dizileri, paralel MRG (pMRG), ya da kartezyen dışı veri alma yöntemleri ile mümkün olmaktadır. Ancak, hızlı görüntü oluşturmak, yüksek işlem gücü gerektiren pMRG ve kartezyen olmayan veri alım yöntemleriyle oldukça zordur. Ayrıca, gerçek zamanlı görüntülemede yüksek sinyal-gürültü oranı sağlanabilse de görüntü çözünürlüğü sınırlıdır. Dolayısıyla MRG kılavuzlu girişimsel uygulamalarda görsel destek arttırılmalıdır. Tezin ana hedefleri şu şekilde belirlenmiştir: (i) gerçek zamanlı pMRG, (ii) radyal veriyle gerçek zamanlı görüntüleme, ve (iii) girişimsel uygulamalarda görsel desteğin hızlı yöntemlerle iyileştirilmesi.

Tezin sağladığı katkılar, (i) gerçek zamanlı karma uzaylı TGRAPPA tabanlı görüntüleme algoritması (şu an için en hızlı TGRAPPA tabanlı görüntüleme algoritması, (ii) ilk radyal veriler için gerçek zamanlı "GRAPPA Operator Gridding" uygulaması, ve MRG kılavuzlu girişimsel uygulamalar için (iii) üç boyutlu anjiyografi yol haritaları, (iv) iyileştirilmiş aktif aygıt gösterimi ve (v) gerçek zamanlı bir aktif aygıt saptayıcısının görüntüleme yazılımına eklenmesi olarak sıralanabilir.

Anahtar Sözcükler: Gerçek zamanlı MRG, pMRG, Kartezyen dışı veri alımı, Paralel Programlama, MRG kılavuzlu girişimsel uygulamalar.

TABLE OF CONTENTS

ACKNOWLEDGMENTS iii
ABSTRACT iv
ÖZET v
LIST OF FIGURES x
LIST OF TABLES xv
LIST OF SYMBOLS xvi
LIST OF ABBREVIATIONS xvii
1. INTRODUCTION 1
1.1 Parallel MRI 3
1.2 Non-Cartesian Trajectories 4
1.3 Visual Feedback during Real-Time MRI Guided Interventions 6
1.3.1 *3D* Colorized MRA/MRCP Roadmaps 6
1.3.2 Active Device Visualization 7
1.3.3 Real-time Active Device Localization 8
1.4 Overview of the Thesis 8
2. REAL-TIME pMRI RECONSTRUCTION FOR CARTESIAN TRAJECTORIES 11
2.1 Image Domain Reconstruction 14
2.1.1 SENSE 14
2.1.2 TSENSE 16
2.2 *k-space* Domain Reconstruction 18
2.2.1 SMASH 18
2.2.2 GRAPPA 20
2.2.3 TGRAPPA 23
2.3 Towards HTGRAPPA: Real-time Low-latency Parallelized TGRAPPA Based Reconstruction 24
2.3.1 Parallel TGRAPPA Implementation 26
2.3.1.1 Results 27
2.3.1.2 Discussion 29

2.3.2 Highly Parallel Image Domain Real-time TGRAPPA Implementation . . . 31
2.3.2.1 Implementation . . . 33
2.3.2.2 Results . . . 33
2.3.2.3 Discussion . . . 34
2.3.3 HTGRAPPA: Highly Parallel B1-Weighted Image Domain TGRAPPA Based Reconstruction . . . 36
2.3.3.1 Methods . . . 37
2.3.3.2 Implementation . . . 39
2.3.3.3 Results . . . 40
2.3.3.4 Discussion . . . 45
2.4 Conclusion . . . 50
3. REAL-TIME MRI RECONSTRUCTION FOR NON-CARTESIAN TRAJECTORIES . . . 52
3.1 Common Gridding Methods . . . 54
3.1.1 Convolution Gridding . . . 55
3.1.2 BURS . . . 56
3.1.3 INNG . . . 58
3.1.4 DING . . . 60
3.2 Towards Real-time Self-Calibrating GROG . . . 62
3.2.1 SC-GROG . . . 62
3.2.2 RT-GROG . . . 66
3.2.2.1 Implementation . . . 68
3.2.2.2 Results . . . 71
3.2.2.3 Discussion . . . 75
3.3 Conclusion . . . 77
4. IMPROVED VISUAL FEEDBACK DURING REAL-TIME MRI GUIDED INTERVENTIONS . . . 80
4.1 Interventional MRI using Multiple *3D* Angiography Roadmaps with Real-time Imaging . . . 83
4.1.1 Methods . . . 83
4.1.2 Experiments . . . 85

4.1.2.1 Pulmonary Thrombectomy 85
4.1.2.2 TIPS 86
4.1.3 Results 86
4.1.4 Discussion 89
4.2 Improved Visual Feedback with Real-Time Edge Detection during MRI Guided Interventions 90
4.2.1 Edge-Detection 92
4.2.1.1 Canny's Edge Detector 93
4.2.1.2 Deriche's Edge Detector 96
4.2.2 Integration into Custom MRI Reconstruction Software 98
4.2.3 Results 100
4.2.4 Discussion 102
4.3 Real-Time Device Localization Software Integration 104
4.3.1 SnapTo 105
4.3.2 Integration into Custom MRI Reconstruction Software 107
4.3.3 Results 108
4.3.3.1 *In Vitro* Experiments 108
4.3.3.2 *In Vivo* Experiments 109
4.3.4 Discussion 110
4.4 Conclusion 112
5. CONCLUSIONS AND DISCUSSIONS 116
5.1 Discussions 117
5.1.1 Real-time parallel MRI 117
5.1.2 Real-time non-Cartesian imaging 117
5.1.3 Improving visual feedback during real-time MRI guided interventions 118
5.1.3.1 *3D* Colorized MRA/MRCP Roadmaps 118
5.1.3.2 Improved Active Device Visualization 119
5.1.3.3 Real-time Active Device Localization 119
5.2 Future Improvements and Possible Extensions 120
5.2.1 Future improvements 120
5.2.1.1 HTGRAPPA 120

5.2.1.2 RTGROG 121
5.2.1.3 Visual Feedback Improvements During Interventions . 122
5.2.2 Conclusions 124
5.2.2.1 Dynamic Imaging 124
5.2.2.2 New Trends in Parallelism 125
REFERENCES 127

LIST OF FIGURES

Figure 2.1 Image reconstruction using SENSE from $R = 4$ data set acquired with $N_c = 4$ coils [39]. 16

Figure 2.2 TSENSE block diagram [40]. 17

Figure 2.3 (a) $R = 3$, block of size 4×5 (4 lines, 5 points per line). (b) $R = 3$, block of size 2×5 (2 lines, 5 points per line). 21

Figure 2.4 GRAPPA k-space filling scheme is represented for a given coil C (rate 2 data set). Small rectangles represent $[4 \times 5]$ GRAPPA blocks (kernels) of coil C that will be used to estimate its missing samples using data from all coils. Thin lines are unacquired samples, and thick lines represent acquired samples of coil C. The kernel is slided over entire k-space to estimate every missing k-space samples. The sum of all the convolutions (kernel values and acquired samples) at a given location provides one sample estimate at the center of the kernel for coil c. When whole k-space is spanned, full k-space data for coil C is obtained. 21

Figure 2.5 GRAPPA and TGRAPPA acquisition schemes are shown ($R = 4$). In GRAPPA, fully sampled data from the center of the k-space is acquired as a set of ACS lines with each frame. On the other hand, the interleaved acquisition scheme of TGRAPPA provides full set of ACS lines every R frames. 24

Figure 2.6 (a) With each new data set, nc threads are used to reconstruct images. Most recent weight-set is used. (b) One thread is dedicated for GRAPPA weight calculation. It is totally asynchronous to the reconstruction. It updates the weight-set as fast as possible with newly coming data. (c) Another thread is dedicated to the display of the final image. 26

Figure 2.7 Reconstructed images from undersampled cardiac data sets using TGRAPPA (matrix size = 192×144). (a) $R = 2$ (b) $R = 3$ (c) $R = 4$. 27

Figure 2.8	SNR values for different GRAPPA weights parameters	30
Figure 2.9	Image domain TGRAPPA reconstruction scheme is represented for a given coil C. Aliased images from each coil is pixel-wise multiplied with the corresponding image domain weights of coil C. The results of the multiplications are summed up to obtained unaliased image for coil C. Once all coil images are obtained, final image is calculated by RSS combining.	32
Figure 2.10	Block diagram of the reconstruction algorithm is given. Dotted rectangles represents parallel regions. The upper dotted rectangle represents the weight-set calculation process, and the lower dotted rectangle represents the reconstruction process. These processes are computed asynchronously.	34
Figure 2.11	HTGRAPPA reconstruction process is represented. Aliased images from each coil are pixel-wise multiplied with corresponding HTGRAPPA unmixing coefficients, and the results are summed to obtained the final image.	38
Figure 2.12	Block diagram of the HTGRAPPA reconstruction algorithm is given. Dotted rectangles represents parallel regions. The upper dotted rectangle represents noise weighting step. The middle dotted rectangle represents the weight-set calculation process, and the lower dotted rectangle represents the reconstruction process. These processes are computed asynchronously.	41
Figure 2.13	Real-time images reconstructed with HTGRAPPA in 0.008 seconds with acquisition of two interleaved slices (18 acquisition coils, acquisition matrix of 192×108). From left to right: long axis, and short axis heart images. From top to bottom, R=2, R=3 and R=4. Darkening in the myocardium from saturation can be seen where the images intersect, since they are acquired serially.	44

Figure 2.14 HTGRAPPA, TSENSE and TGRAPPA compared. From left to right: TSENSE, TGRAPPA, HTGRAPPA reconstructed images from data acquired with 32 acquisition coils with acquisition matrix of 192×108. Upper images represent large field of view (380), and lower images represent small field of view (320). Note that artifacts from pre-folding are seen with TSENSE, but not with TGRAPPA and HTGRAPPA. 45

Figure 2.15 Series of images (reconstructed from data acquired with R=4, N_c=30, acquisition matrix of 192×108) showing the response of our reconstruction system to dynamic scan plane change. At frame 3, the scan plan was changed perpendicularly. Our system switched to view-sharing reconstruction while the new weight-sets are being calculated. At frame 21, reconstruction switched back to HTGRAPPA when the weight-sets were ready to use. 46

Figure 2.16 Cardiac cine image series reconstructed from rate 4 data set in real-time using HTGRAPPA. 46

Figure 2.17 Performance plots of HTGRAPPA for weight-set calculation and reconstruction with different number of acquisition coils (acquisition matrix of 192×108, acceleration rate of 4). a. Weight-set calculation performances in seconds with block sizes of $[2 \times 5]$ and $[4 \times 5]$. b. Reconstruction performances in seconds. 49

Figure 3.1 GROG gridding for two samples S_1 and S_2 of a projection P_i is represented. Gridding weights G_x, G_y are applied to the samples S_1 and S_2 according to their distances to their closest Cartesian grid neighbors S_1^c and S_2^c, and their contributions to these Cartesian grid points are calculated by $S_1^c = G_x^{m_1} G_y^{n_1} S_1$, $S_2^c = G_x^{m_2} G_y^{n_2} S_2$. 64

Figure 3.2 Relation between local angular weights G_θ per projection and global shifting weights G_x and G_y [28]. 65

Figure 3.3 Comparison of *in vivo* images reconstructed from a radial data set using SC-GROG and conventional gridding [28]. 66

Figure 3.4 Comparison of *in vivo* images reconstructed from a spiral data set using SC-GROG and conventional gridding [28]. 67

Figure 3.5 Block diagram of the RT-GROG reconstruction algorithm is given. Dotted rectangles represent parallel regions. The upper dotted rectangle represents the weight-set calculation process, and the lower dotted rectangle represents the reconstruction process. These processes are computed asynchronously. 69

Figure 3.6 High resolution brain image reconstructed using RT-GROG (400 projections; 256 samples per each). 72

Figure 3.7 Phantom images reconstructed from a 15 coil data set using our real-time GROG implementation. From left to right: a. 256 projections b. 128 projections c. 64 projections. (Image brightness levels are adjusted for easier view of artifacts.) 73

Figure 3.8 Cardiac cine image series reconstructed in real time from a 32 coil data set, 64 projections (128 samples). Acquisition time: 151 ms. Reconstruction time: 69 ms. 73

Figure 3.9 Long axis heart image reconstructed in real-time from 32 coil data set, 96 projections (128 samples). Acquisition time: 226 ms. Reconstruction time: 76 ms. 74

Figure 3.10 Image reconstruction and weight set calculation performances in seconds for 12 coil phantom data and different number of projections (number of samples = 256). 74

Figure 4.1 Images captured from the *3D* rendering for use in MRI-guided TIPS experiment showing multiple colorized MIPs rendered with two orthogonal real-time image slices. Each MIP is indicated by an arrow. a) All MIPs are visualized. b) Only hepatic artery (red) is shown c) Only portal vein (green) is shown. d) Hepatic artery (red) and hepatic vein (yellow) are grouped and shown. 85

Figure 4.2 Colorized *3D* MIP of pulmonary veins (yellow) and arteries (light blue) shown in conjunction with colorized active device (marked with arrows: tip shown in red, shaft shown in green) and two real-time image slices. 87

Figure 4.3 *3D* colored MIP of jugular vein displayed with real-time images for guidance during TIPS procedure for jugular access. 88

Figure 4.4 *3D* colored dynamically updated MIPs displayed in conjunction with real-time multi-slice images during MRI guided TIPS procedure. a) Biliary tree in yellow. b) Active guidewire in blue, hepatic artery in red, hepatic vein in yellow, and portal vein in green. 89

Figure 4.5 Images from pig experiment with three channel guiding catheter that shows selective edge detection feature. a) edge detection is disabled. b) edge detection is enabled on all channels. c) edge detection is enabled only on tip channel. d) edge detection enabled only on middle channel. e) edge detection enabled on tip and on the middle channel, and disabled on shaft channel. 99

Figure 4.6 Phantom images with active device. a) Without edge detection b) With edge detection. 101

Figure 4.7 In vivo pig images acquired during real-time MRI guided vascular intervention. a) Without edge detection b) With edge detection. 101

Figure 4.8 Edge localization for different α values. 103

Figure 4.9 a), b) and c) represent three projection images blended with estimated device trajectories. d) Initial imaging plane plane before executing SnapTo is given. The arrows represents the intersection of the device with the imaging plane in two places. e) Imaging plane after SnapTo execution is given. SnapTo successfully estimates the best *2D* plane that covers the device, and a new imaging plane is prescribed. 109

Figure 4.10 a), b) and c) represent three projection images blended with estimated device trajectories. d) Initial imaging plane plane before executing SnapTo is given. The arrows represents the intersection of the device with the imaging plane in two places. e) Imaging plane after SnapTo execution is given. SnapTo successfully estimated the best *2D* plane that covers the device, and a new imaging plane was prescribed. 110

LIST OF TABLES

Table 2.1	Reconstruction and weight-set calculation performances (for 24 and 32 ACS lines) in seconds for $R = 2$	28
Table 2.2	Reconstruction and weight-set calculation performance in seconds for $R = 3$	28
Table 2.3	Reconstruction and weight-set calculation performance in seconds for $R = 4$	29
Table 2.4	Image reconstruction times in seconds for conventional (conv.) and image domain (img.) TGRAPPA using RSS combining with for different R values and block sizes of $[2 \times 3]$, $[2 \times 5]$, $[2 \times 7]$, $[4 \times 3]$ and $[4 \times 5]$	35
Table 2.5	Weight-set calculation times in seconds for conventional (conv.) and image domain (img.) TGRAPPA using RSS combining for different R values and block sizes of $[2 \times 3]$, $[2 \times 5]$, $[2 \times 7]$, $[4 \times 3]$ and $[4 \times 5]$	35
Table 2.6	Weight-set calculation times (in seconds) for HTGRAPPA for different acceleration (R) values and block sizes of $[2 \times 3]$, $[2 \times 5]$, $[4 \times 3]$ and $[4 \times 5]$ when 16 CPUs are used (18 acquisition coils, acquisition matrix of 192×108, 48 ACS lines).	42
Table 2.7	Reconstruction times (in seconds) for conventional TGRAPPA and for HTGRAPPA for different acceleration (R) values and block sizes of $[2 \times 3]$, $[2 \times 5]$, $[4 \times 3]$ and $[4 \times 5]$ (18 acquisition coils, acquisition matrix of 192×108).	43
Table 3.1	Performance comparison (in seconds) between conventional SC-GROG and RT-GROG on 32 coil data sets with different matrix sizes.	76

LIST OF SYMBOLS

fps	Frames per second
Hz	Hertz
GHz	Gigahertz
mm	Milimeter
ms	Milisecond
T	Tesla

LIST OF ABBREVIATIONS

ACML	AMD Core Math Library
ACS	Auto-Calibration Signal
AMD	Advanced Micro Devices
ATLAS	Automatically Tuned Linear Algebra Software
AUTO-SMASH	Auto Calibrated SMASH
BPE	Bunched Phase Encoding
BURS	Block Uniform Resampling
CG	Conjugate Gradient
CPU	Central Processing Unit
CT	Computerized Tomography
DFT	Discrete Fourier Transform
DICOM	Digital Imaging and Communications in Medicine
DING	Deconvolution-Interpolation Gridding
ECG	Electrocardiography
FFT	Fast Fourier Transform
FLASH	Fast Low Angle Shot
FOV	Field of view
FPGA	Field-Programmable Gate Array
GCC	GNU Compiler Collection
GPU	Graphics Processing Unit
GRAPPA	Generalized Autocalibrating Partially Parallel Acquisitions
GROG	GRAPPA Operator Gridding
HASTE	Half-Fourier Acquisition Single-Shot Turbo Spin-Echo
HTGRAPPA	Hybrid TGRAPPA
INNG	Iterative Next-Neighbor Regridding
LUT	Lookup Table
MIP	Maximum Intensity Projections
MRA	Magnetic Resonance Angiography

MRCP	Magnetic Resonance Cholangiopancreatography
MRI	Magnetic Resonance Imaging
NHLBI	National Heart Lung and Blood Institute
NIH	National Institutes of Health
OpenGL	Open Graphics Library
OpenMP	Open Multi-Processing
pMRI	Parallel Magnetic Resonance Imaging
POV	Point of View
PRAM	Parallel Random Access Model
PROPELLER	Periodically Rotated Overlapping Parallel Lines with Enhanced Reconstruction
Pthreads	Posix Threads
RSS	Root-Sum-Square
RT-GROG	Real-Time GROG
SC-GROG	Self-Calibrating GROG
SENSE	Sensitivity Encoding
SMASH	Simultaneous Acquisition of Spatial Harmonics
SNR	Signal-to-noise ratio
SSFP	Steady-State Free Precession
TGRAPPA	Temporal Generalized Autocalibrating Partially Parallel Acquisitions
TIPS	Transjugular Intrahepatic Portosystemic Shunting
TR	Repetition Time
TSENSE	Temporal SENSE
UNFOLD	Unaliasing by Fourier-encoding the Overlaps using the Temporal Dimension
URS	Uniform Resampling
VD-AUTO-SMASH	Variable Density AUTO-SMASH
XMT	Explicit Multi-Threading

1. INTRODUCTION

Minimally invasive medicine by percutaneous and surgical interventions rapidly progressed over the years. As a result, image based guidance became a must to overcome technical difficulties caused by the increased complexity of such operations. X-ray based imaging, and ultrasound are two of the most popular image guidance modalities during surgical and percutaneous interventions. However, sound reflections in ultrasound, and lack of soft tissue contrast and ionizing radiation in X-ray make them impractical to broadly guide interventions. On the other hand, Magnetic Resonance Imaging (MRI) provides a radiation free imaging modality, with high soft tissue contrast. It provides unrestricted slice orientation in three dimensions. Additionally, MRI also provides functional information, such as the blood flow, tissue oxygenation and perfusion. However, its data acquisition was slow compared to other modalities, and thus could not be used to guide interventional applications until recently. During the last decade, introduction of short, wide bore magnets with high-performance gradient systems, and arrival of MR compatible instruments, such as needles, active guidewires and catheters [1–5] made MRI guided interventions possible [6–10], and highly promising. Active devices possess their own receiver channels, and they are able to acquire MR signals. Signal acquired by the device is used for localization and visualization purposes. That way, active devices can be visualized under MRI, and they can be used during real-time MRI guided interventions.

Real-time MRI was often defined as fast data acquisition to permit dynamic imaging, and to allow better diagnosis by means of functional imaging (e.g. imaging whole heart cycle), and did not cover real-time image reconstruction. The real-time guidance requirement of interventions broadened the term "real-time MRI", by enclosing real-time image reconstruction concept into it. Therefore, in the scope of this thesis, the term real-time MRI implies both fast data acquisition, and fast, low-latency image reconstruction.

Real-time MRI has three major requirements: (i) fast data acquisition, (ii) fast image reconstruction, and (iii) good image quality and improved feedback during interventions. Fast gradients and optimized real-time sequences offered increased data acquisition performance. Nowadays, physiological limits are reached, and thus gradient speeds and real-time sequences can barely provide further improvements. Data acquisition performance can be further improved using parallel MRI (pMRI) methods [11–17]. pMRI involves multiple receiver coils. For accelerated imaging, some of the phase encoding steps are skipped, and k-space is sub-sampled. Subsampling in the k-space causes aliasing in the image domain, therefore unacquired samples must be estimated, or aliasing must be removed in the image domain subsequent to the reconstruction. Estimation of unacquired k-space data samples or removal of the artifacts involves advanced reconstruction methods with high computational loads. Therefore, increased data acquisition performance implies slower image reconstruction. Another way of increasing data acquisition peformance is to use non-Cartesian k-space sampling schemes, such as radial and spiral trajectories which provide more information with less data using a denser sampling scheme in the center of the k-space. Hence, faster data acquisition is possible with such trajectories. However, non-Cartesian sampling schemes involve more computationally demanding image reconstruction algorithms [18–28], and require better artifact suppression methods compared to Cartesian trajectories [27–31]. Therefore, fast image reconstruction is non-trivial when non-Cartesian trajectories are used.

Real-time MR imaging mostly involves single or multiple two dimensional slices for higher frame rates and acquired images are in lower resolution than conventional non-real time MR images. pMRI methods and non-Cartesian sampling schemes also increase image artifact levels. Therefore, providing a good image quality is a real challenge during real-time MRI. Especially when real-time MRI guided interventions are involved, two dimensional images with limited spatial resolution bring forth the need to improve visual feedback by means of post-processing techniques. Otherwise, it may be really hard to distinguish different vascular structures, and the exact position of the active devices on current imaging plane.

High computational demands of pMRI reconstruction methods and non-Cartesian sampling schemes can be addressed with high performance computing (e.g. parallel computing). Optimized and properly parallelized implementations will make the real-time low latency imaging possible, and will increase the practicality of MRI guided interventions. Additionally, real-time feedback can be improved by means of post-processing techniques applied to reconstructed images, and thus better guidance during interventions can be achieved.

During the thesis, three specific aims were defined motivated by the issues described above: (i) real-time image reconstruction for pMRI using parallel processing, (ii) real-time image reconstruction for non-Cartesian trajectories using parallel processing, and (iii) fast MRI post-processing for improved visual feedback during real-time MRI guided interventions. In the following sections, the concept of pMRI, non-Cartesian trajectories and visual feedback improvement challenges during real-time MRI guided interventions, as well as the thesis achievements in each of these areas, will be introduced. At the last section, the overview of the thesis will be presented.

1.1 Parallel MRI

Introduction of pMRI was one of the key advances in MRI methodology in recent years to reduce data acquisition time [12–17]. Parallel MRI involves multiple independent receiver coils, often called phased array coils, to acquire data for faster scanning, and to improve signal-to-noise ratio (SNR) [11]. Multiple small coils, each with their independent receiver channels, are combined in such a way that the combination of their sensitive volumes is approximately equivalent to a single, but a larger coil. Reduced coils size improves SNR by decreasing the noise amplitude of the detected signal. However, the real potential of pMRI lies on faster data acquisition.

Following the development of the phased array coils, their scan time reduction potential is discovered around late 1980's and early 1990's [12–16]. Scan time is linearly proportional to number of phase encoding steps. Thus, if one can decrease the number

of phase encoding steps while keeping the spatial resolution fixed, accelerated data acquisition will be possible, which is the basic idea behind faster imaging with pMRI. Number of skipped phase encoding steps per acquisition determines the “acceleration factor", R. In practice, such an acquisition scheme will subsample the k-space, and thus aliasing artifacts will occur in the image domain, due to the missing phase encode lines. If the sensitivities of each coil in a phased array are known or can be estimated, these sensitivity information can be used to prevent aliasing [32–38] by synthesizing unacquires samples in the k-space, or remove the aliasing [39–43] in the image domain. The most commonly used pMRI reconstruction algorithms are k-space domain (i) Generalized Autocalibrating Partially Parallel Acquisitions (GRAPPA) [36], (ii) Temporal GRAPPA (TGRAPPA) [38], and image domain (i) Sensitivity Encoding (SENSE) [39], (ii) TSENSE [40] algorithms.

k-space domain and image domain parallel MRI reconstruction algorithms show different kind of artifacts [44, 45]. GRAPPA based algorithms are advantageous over SENSE, when small field of views (FOV) are desired (e.g. cardiac imaging), and when exact coil sensitivity profiles are unknown. Detailed comparison of image domain and k-space domain pMRI reconstruction techniques are studied in [46, 47]. TGRAPPA and TSENSE are advantageous for real-time dynamic imaging, especially for real-time MRI guided interventions, as these schemes can track changes in the coil sensitivity profiles to provide better artifact suppression and adaptability.

1.2 Non-Cartesian Trajectories

Cartesian trajectories are most popular trajectories for MR data acquisition. Acquiring equidistant k-space lines permit using highly efficient Fast Fourier Transform (FFT) algorithm to reconstruct images. However, they require relatively long scan times as acquisition of each k-space line necessitates a separate RF excitation pulse. On the other hand, non-Cartesian trajectories may provide denser sampling in the center of the k-space, where the low frequency (information rich) data lies, and thus may permit to obtain more information while acquiring less data than Cartesian trajectories

[48]. However, reconstruction of non-Cartesian sampled data is non-trivial, as FFT cannot be directly applied. Discrete Fourier Transformation (DFT) can perform the reconstruction, but this method is computationally too slow to be practical. The most common method is to resample the non-Cartesian data set onto a Cartesian grid and then to apply the FFT algorithm to reconstruct images. The process of mapping non-Cartesian data set onto a Cartesian grid is called "gridding". Many algorithms are developed to perform gridding [18–28]. The most common gridding algorithm among them is convolution gridding, also called as conventional gridding [18,19]. The common drawbacks of most gridding algorithms are high number of parameters affecting image quality, and non-trivial density compensation [27–30].

The most common non-Cartesian trajectories are spiral and radial trajectories. Other non-Cartesian trajectories such as Periodically Rotated Overlapping Parallel Lines with Enhanced Reconstruction (PROPELLER) [49], Rosette [50] are also proposed. Data acquired using radial and constant angular velocity spiral trajectories can be reconstructed using back-projection reconstruction technique. Gridding can be performed for both spiral and radial data sets after calculating the k-space coordinates of each sample. k-space coordinates determination is quite straightforward for radial trajectories, as it can be directly calculated by sine and cosine of the projection angle. On the other hand, such a calculation requires lots of effort for spiral trajectories as studied in [31].

Seiberlich *et al.* proposed a novel gridding algorithm based on GRAPPA operator formalism [51], GRAPPA Operator Gridding (GROG), in 2007. Unlike other parameter filled gridding methods, GROG is using pMRI principles to calculate gridding weights, and thus no parameters are involved. The only drawback of the method, is the requirement to acquire a cartesian calibration data set prior to non-Cartesian imaging. A year later, Seiberlich *et al.* modified the original GROG algorithm, and proposed a self-calibrating version, SC-GROG [28]. With SC-GROG, additional calibration data set requirement is eliminated, and a totally self-calibrating gridding algorithm is introduced. In SC-GROG, non-Cartesian data themselves are used to determine gridding weights. First, local shifting weights per projection are calculated. Subsequent to that,

a global set of weights in x and y directions are determined. There are no gridding parameters involved, as in original GROG. In summary, SC-GROG is a parameter-free, parallel-imaging based, robust, self-calibrating gridding algorithm that can be applied to grid both undersampled and fully sampled radial or spiral data sets. Additionally, the quality of (SC)-GROG images is on par with conventional gridding [27, 28].

1.3 Visual Feedback during Real-Time MRI Guided Interventions

Recent technological advances made it possible to perform real-time low-latency MR imaging to guide interventions [6–10, 42, 52, 53]. However, real-time pulse sequences are usually restricted to single or multiple *2D* slices to provide necessitated imaging speed. As a result, visual feedback is somehow limited during real-time MRI guided interventions. The key challenges can be described as follows: (i) *2D* images with low spatial resolution makes it harder to distinguish different vascular structures during the intervention; (ii) active device visualization is non-trivial; and (iii) *2D* slices make it harder to localize the active device on the imaging plane.

1.3.1 *3D* Colorized MRA/MRCP Roadmaps

Visualization of structures outside the imaging plane are often needed, such as different portions or different vascular trees. However, *2D* imaging restricts the visual feedback to the selected slice planes. If no visual guidance is available while moving the active devices towards the target area, "blind" attempts can create life-threatening non-target punctures, and significant patient discomfort may occur. For example, percutaneous transjugular intrahepatic portosystemic shunting (TIPS) [54] relieves portal venous congestion due to cirrhosis, and MRI guided intervention is a good candidate to perform it. In this procedure, a catheter traverses hepatic parenchyma to connect a portal vein to a hepatic vein. If no image guidance is available, multiple "blind" attempts are required to traverse the liver from a vein using a sharp needle. Therefore life-

threatening non-target punctures, and significant patient discomfort may occur. The differentiation of biliary tree, hepatic artery, hepatic vein and portal vein is crucial not only during this procedure, but also during other interventions, such as angioplasty, biopsy, stent placement, and electrophysiology procedures. Contrast-enhanced MR angiography offers excellent depiction of such structures but it cannot be performed in real-time and repeated at will. However, if we could import these previously acquired data sets and display them in conjunction with real-time reconstructed *2D* slice images, this drawback would be eliminated [55]. Similar work has been presented for X-ray/Computerized Tomography (CT) and XMR systems in [56–58], and commercially implemented by Siemens (Syngo iPilot tool, Siemens Medical Solutions, Germany) and by Philips Medical Systems (Koninklijke Philips Electronics N.V.). Some of these methods permit using previously acquired MRI data during interventions, but they still necessitate X-ray/CT imaging, and thus imply patient radiation exposure.

Contrast-enhanced MR angiography provides an excellent depiction of vascular structures, but they cannot be performed in real-time and repeated at will. If MR angiography could be performed prior to the intervention, and acquired data could be somehow imported to be displayed in *3D* in conjunction with real-time reconstructed *2D* slice images, it would be possible to provide out of plane anatomical structure visualization.

1.3.2 Active Device Visualization

Despite the fact that active devices possess their own receiver coils [1–5], their visualization is not an easy task. Acquired device data can be reconstructed separately to obtain the gray-scale sensitivity map of the device. However, fusion of a gray-scale active device visualization with grey-scale anatomical images does not add much in terms of visual context. Pseudo-color assignment to each active device channels [53] drastically improves the visual feedback, however, the "colored cloud" that represents the device may occasionally obscure underlying anatomical structures immediately around the device (e.g. mitral valve, aortic valve structures). This problem is worsened

by increased device diameters. Additionally, device coupling occurs inevitably due to dimensional restrictions of small devices and may perturb the colored device map. Showing only the top ridge or specific edges of active device signal may help improve visibility of fine anatomical structures immediately around the device.

1.3.3 Real-time Active Device Localization

Scan plane changes often occur during real-time MRI guided interventions, which may result from active devices out of the imaging plane. Device only projection imaging by switching off the slice-encode gradient makes it possible to relocate the active device by performing some trial-and-error methods [53]. Sathyanarayana *et al.* presents an automatic active device localization method in 2007 [59]. His method imposes particular shape or orientation requirements to be able to localize the device. Other methods to actively visualize catheters during interventional MR angiography procedures are represented in [5, 60]. However, these methods are procedure specific, and imaging speed is very limited. Another automatic device localization algorithm, SnapTo is presented in [61]. This iterative method uses three perpendicular projection images to estimate the device location. Subsequent to the estimation, *3D* trajectory of the device can be drawn, or selected slice can be prescribed to a new location, such that the optimal view of the device will be obtained.

1.4 Overview of the Thesis

The work described in this thesis covers the high performance, highly parallel real-time low-latency image reconstruction algorithm implementations for both pMRI and non-Cartesian trajectories, and visual guidance improvements during real-time MRI guided interventions. This thesis can thus be split into three parts, also representing three specific aims: (i) real-time image reconstruction for pMRI, (ii) real-time image reconstruction for non-Cartesian trajectories, and (iii) improved visual feedback during real-time MRI guided interventions. Motivated by these aims, thesis contributions

include: (i) real-time hybrid domain TGRAPPA based pMRI reconstruction algorithm (currently the fastest TGRAPPA based algorithm) [41–43], (ii) first real-time implementation of self-calibrating GRAPPA Operator Gridding (SC-GROG) algorithm for radial acquisitions, (iii) multi-phase *3D* MR angiography roadmaps for MRI guided interventions [55], (iv) improved active device visualization during real-time MRI guided interventions [62], (v) integration of a real-time active device localizer algorithm. The last three contributions (iii), (iv) and (v) are the tools developed to improve visual feedback during real-time MRI guided interventions, and thus represent third thesis aim.

All the implementations represented in this thesis are integrated into custom real-time MRI software environment introduced by Guttman *et al.* [52, 53], and successfully employed during real-time MRI guided cardiovascular interventions.

Each specific aim is presented in one specific Chapter. Each Chapter is initiated by a brief introduction, where the problem is presented, which is then followed by previous work realized in that field. Subsequent to that, algorithmic and implementation details of the proposed methods are detailed and in depth results are presented. After a discussion section, where the results of the proposed methods are analyzed in detail, a conclusion emphasizing the impact and novelty of the work are given. Third specific aim from which we present three separate contributions, is divided into three different sections, where each contribution is separately studied.

Chapter 2 describes the first contribution of the thesis; real-time low-latency TGRAPPA based pMRI reconstruction scheme HTGRAPPA [41–43]. HTGRAPPA is currently the fastest TGRAPPA based reconstruction algorithm. It is nearly 180 times faster than conventional TGRAPPA when reconstructing rate 4 data set. HTGRAPPA is also commercially implemented (currently only for off-line reconstruction) by Siemens.

Chapter 3 provides the implementation details of the first real-time low-latency implementation of SC-GROG algorithm, RT-GROG. Highly parallelized RT-GROG

permits multi-slice radial imaging scheme for real-time MRI. RT-GROG provided up to 34-fold reconstruction performance increase over conventional SC-GROG during the reconstruction of the 32 coil cardiac data set. Reconstruction was always faster than the data acquisition with RT-GROG.

Chapter 4 concerns the work aimed to improve visual feedback during real-time MRI guided interventions. This Chapter can be divided into three parts: (i) multi-phase *3D* MR angiography roadmaps for MRI guided interventions [55], (ii) improved active device visualization during real-time MRI guided interventions [62], (iii) integration of a real-time active device localizer algorithm. Each part is described in a separate section. First, multi-phase *3D* roadmap generation software is explained. This new software improved the visualization of invasive devices through and between complex vascular spaces, and has the potential of improving patient safety and procedure easiness and efficacy. Our software is employed during several MRI-guided applications, such as pulmonary thrombectomy and TIPS, and is proved to ameliorate visual feedback. Our method can also be utilized during angioplasty, biopsy, stent placement, and electrophysiology procedures. Second, real-time edge detection software aimed to improved device representation is described. Deriche's edge detection algorithm is integrated into our real-time MRI environment to show only the top ridge or specific edges of the active device signal. That way, the obscurity around the device was removed, and visual feedback was drastically improved. Additionally, artifacts due to device coupling was lowered. Finally, real-time integration details of an active device localization algorithm, SnapTo is presented.

In conclusion, Chapter 5 summarizes the work and identifies the principal achievements for each specific aim. In addition, new potential application areas for each aim and its possible extensions are discussed for further development.

2. REAL-TIME pMRI RECONSTRUCTION FOR CARTESIAN TRAJECTORIES

Parallel MRI (pMRI) aims to use spatial sensitivity information of an array of multiple receiver coils to partially replace very time consuming phase encoding step during data acquisition. As a result, faster data acquisition is obtained while maintaining spatial image resolution and image contrast intact. However, reduced phase encoding steps imply subsampling on k-space. Therefore, direct application of inverse Fast Fourier Transform to subsampled data ends up with aliased images.

There are several different pMRI reconstruction algorithms aiming to remove aliasing incorporated by reduced phase encodings during data acquisition. In general, pMRI reconstruction algorithms are categorized as; (1) k-space domain reconstruction algorithms (e.g. SMASH [32, 33], GRAPPA [36], TGRAPPA [38]), and (2) image domain reconstruction algorithms (e.g. SENSE [39], TSENSE [40]). Original SMASH and SENSE algorithm require the knowledge of the coil sensitivity profiles, while GRAPPA uses additionally acquired auto-calibration lines from the center of the k-space to calculate its k-space reconstruction weights.

Kellman *et al.* represented TSENSE [40] algorithm in 2001, which removed the coil sensitivity knowledge requirement of the SENSE algorithm. In this method, data acquisition is alternating between even and odd k-space lines. Combination of R (acceleration rate) frames would provide a full k-space data set that could be used to estimate coil sensitivity profiles. Therefore, TSENSE is able to track changes in coil sensitivity profiles during imaging, and this fits well to the dynamic real-time imaging requirements. Similarly, TGRAPPA [38] algorithm is presented by Breuer *et al.* in 2005. TGRAPPA algorithm eliminates the need for additional calibration data set during data acquisition using the same alternating acquisition scheme. Additionally, it is possible to use a fully sampled k-space calibration data set during GRAPPA weight calculation instead of a limited central band in the k-space.

Real-time TSENSE implementation has been previously demonstrated by Guttman *et al.* [52]. However, to the best of our knowledge there was no real-time TGRAPPA implementation that could be deployed in such applications with real-time, low-latency image reconstruction. This is mostly due to the convolution-based (T)GRAPPA reconstruction scheme that requires significantly greater computation than TSENSE. Conventional GRAPPA is a purely k-space based algorithm where both the calculations of weights and the reconstruction take place in k-space. However, as the convolution operation in the k-space domain is equivalent to multiplication in the image-domain, computational demands of the GRAPPA algorithm can be greatly reduced if a hybrid k-space and image-domain reconstruction is used [63–65]. Additionally, this hybrid approach makes it possible to remove dependency of reconstruction speed on GRAPPA kernel size and acceleration rate. TGRAPPA algorithm is beneficial as this method performs better when a smaller field of view (FOV) is desired, and as it is more tolerant of image wrap [44].

Our proposed novel algorithm HTGRAPPA [41–43] is a hybrid algorithm based on TGRAPPA, where weight calculations involve both k-space domain (GRAPPA weight calculation) and image domain (image domain transformation and generation of unmixing coefficients), and where image reconstruction takes place in image domain. But more importantly, HTGRAPPA algorithm is highly parallel. Original TGRAPPA algorithm is sequential, where weights calculation is followed by the reconstruction process. In HTGRAPPA, weights calculation and reconstruction process are decoupled and individually parallelized for high performance. Shared memory architecture is preferred because (HT)GRAPPA algorithm requires to use all-coil data to reconstruct single coil image. Sending all-coil data in a distributed environment will be time consuming due to the data transfer overhead. Because each CPU has access to the same memory location, shared memory architecture is well suited for HTGRAPPA algorithm. The drawback of such a system is that the scalability is very limited. In HTGRAPPA, the number of reconstruction threads is equal to the total number of receiver coils. Each coil has its own reconstruction thread. On the other hand, weights calculation process does not have a fixed number of threads. This dynamic structure allows better adaptation to change in the coil sensitivities and to different computer

architectures. For example this approach makes it possible to deploy HTGRAPPA algorithm in real-time on a standard quad-core desktop computer. OpenMP library is preferred to parallelize weights calculation process because it allows to dynamically change thread number. Lower level Pthreads libary is preferred to parallelize reconstruction process. Additionally, decoupling helps to achieve low-latency real-time image reconstruction. Each newly acquired frame does not require a new set of weights; one fixed set of weights can be used to reconstruct multiple acquisition frames without artifacts. Therefore, in our proposed scheme, reconstruction process uses most recent set of weights to reconstruct images, while weights are continously updated in the background using more recent set of calibration data. Auto-calibration is performed using a time interleaved acquisition scheme similar to that employed in TSENSE and TGRAPPA by combining several R (acceleration factor) consecutive frames.

(T)GRAPPA algorithm provides the final reconstructed image by combining per coil images using root sum of squares (RSS) to avoid dephasing. In HTGRAPPA, RSS combining is replaced with adaptive *B1*-weighted combining [66] for further improved reconstruction performance while improving signal-to-noise ratio (SNR) characteristics. In this approach, k-space domain GRAPPA weights are calculated in the k-space as it was done in GRAPPA, and then transformed into the image domain. After the transformation, these per-coil image domain weights are linearly combined into a composite set of unmixing coefficients using an estimated sensitivity map. These precalculated weights are then applied to the aliased images as pixel-wise multiplication to reconstruct a single unaliased image (as in SENSE) with significant computational reduction over previous GRAPPA implementations. Additionally, noise-weighting is applied to image domain GRAPPA weights in order to improve SNR in the reconstructed images.

In this chapter, first, basic image domain and k-space domain pMRI reconstruction algorithms (T)SENSE and (T)GRAPPA will be briefly introduced. Subsequent to this introduction, our TGRAPPA based ultra-fast hybrid (k-space domain + image domain) pMRI reconstruction algorithm HTGRAPPA will be given in details. Finally, our approach will be discussed by indicating what is missing, and what can be further ameliorated.

2.1 Image Domain Reconstruction

Without any doubt, the most widely used image domain pMRI reconstruction algorithm is the SENSE algorithm. TSENSE is a variant of the SENSE algorithm incorporating temporal filtering for improved real-time imaging.

SENSE algorithm normally requires the knowledge of the coil sensitivity profiles. These profiles can be obtained by a calibration scan right before pMRI using SENSE. The disadvantage of using previously calculated sensitivity profiles is that these profiles may change during imaging due to patient motion, or coil displacement over time. On the other hand, TSENSE uses an interleaved acquisition scheme that makes it possible to properly estimate coil sensitivities from the acquired data itself. That way, sensitivity profiles are automatically updated during imaging, and poor artifact suppression caused by patient or coil movement is avoided. This is especially important for cardiac imaging, where three potential aliasing/artifact sources are present: (1) motion due to breathing, (2) motion due to cardiac cycle, (3) patient movement. If an interventional procedure is involved, the operator can be considered as the fourth source of aliasing/artifact as he or she can perturb patient or coil position. That is why TSENSE is a better fit for real-time dynamic imaging.

SENSE and TSENSE algorithm will be briefly explained in the following sections.

2.1.1 SENSE

k-space location of samples are well known when data is acquired using cartesian trajectories. If an acceleration factor R is used during data acquisition, subsampling in the k-space causes reduced FOV in the image domain. Therefore, resulting reconstructed images are aliased (or folded), where each pixel contains information from R pixels of the desired unfolded image. If one can separate each pixel in the folded image into R distinct pixels and position these properly, aliasing will be removed.

Let C_k represent the coil sensitivity profile of a given coil k. As each pixel is also weighted by the coil sensitivities, one pixel of the image of a given coil k at specific location (x, y) is equal to

$$I_k(x,y) = C_k(x,y_1)\rho(x,y_1) + C_k(x,y_2)\rho(x,y_2) + \ldots + C_k(x,y_R)\rho(x,y_R) \tag{2.1}$$

where ρ represents the desired pixel values. Eq. 2.1 can also be represented in matrix form,

$$I = C\rho \tag{2.2}$$

with C being the matrix of sensitivity profiles of each coil at the R superimposed positions, ρ, the desired pixels and I, the complex pixel values of each coil images. Thus, ρ can be calculated as[1]

$$\rho = (C^H C)^{-1} C^H I \tag{2.3}$$

with generalized inverse of matrix C. Note that matrix inversion in Eq. 2.3 cannot be performed if acceleration rate R is greater than the total number of coils N_c. Figure 2.1 [39] represents basic idea behind SENSE. Final unaliased image is obtained once Eq. 2.3 is repeated for every pixel location in the aliased image. SENSE does not produce per-coil images; final image is obtained using a phased array combiner.

SENSE causes additional SNR loss depending on the coil array geometry factor, generally called g-factor or g:

$$SNR_{sense} = \frac{SNR_{full}}{\sqrt{R}g} \tag{2.4}$$

g represents the encoding efficiency, in other words, measures the efficacy of the matrix inversion in Eq. 2.3.

[1]Noise correlation is normally incorporated into this equation. For simplicity, this is not addressed here. Original SENSE paper investigates this issue in details.

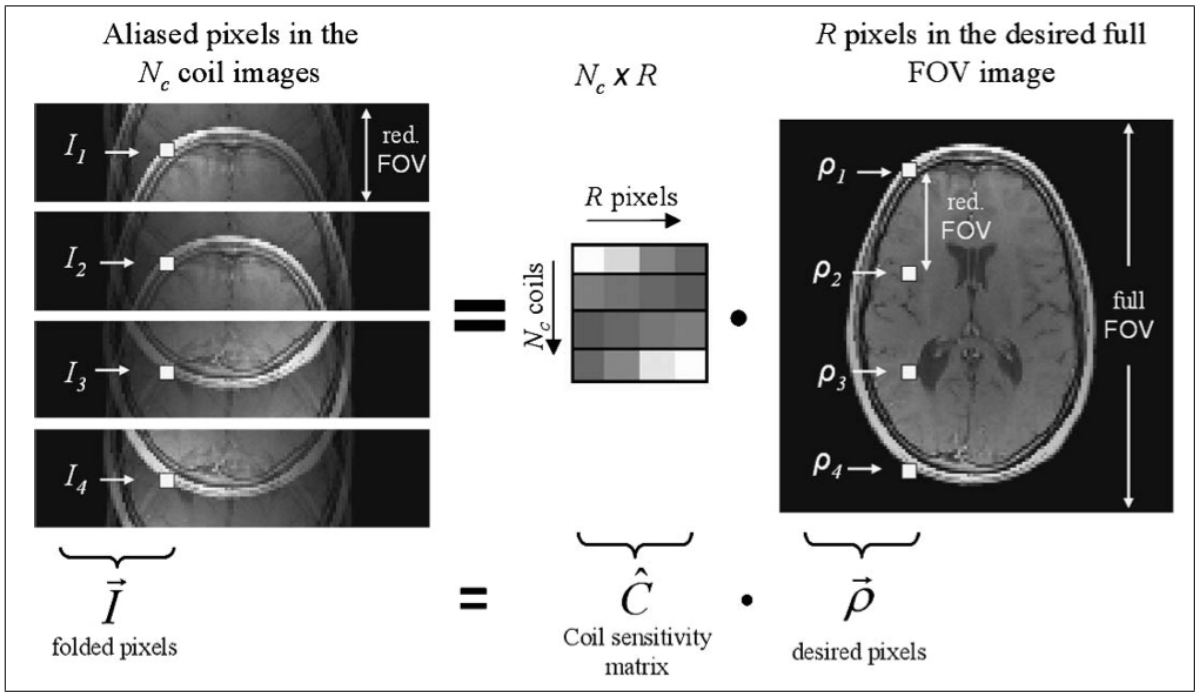

Figure 2.1 Image reconstruction using SENSE from $R = 4$ data set acquired with $N_c = 4$ coils [39].

2.1.2 TSENSE

TSENSE uses "unaliasing by Fourier-encoding the overlaps using the temporal dimension (UNFOLD)"-like [67] interleaved k-space data acquisition scheme to incorporate temporal filtering to spatial encoding. With this acquisition scheme that alternates between odd and even lines, it is possible to temporally frequency-shift the aliasing artifact to the band edge. After this shifting, aliasing can be suppressed by temporal low pass filtering. Spatial filtering accompanied by the temporal filtering improves aliasing artifact rejection, and helps increasing the tolerance on accuracy of the coil sensitivity estimates and tolerance on selection of temporal low-pass filter.

TSENSE is an adaptive algorithm as the coil sensitivity profiles are directly estimated from acquired data set. Thus, separate acquisition of a calibration data set is not required. This adaptivity makes it possible to track changes in the coil sensitivities during dynamic real-time imaging. TSENSE algorithm is well suited to interventional MRI applications where scan plan changes often. Note that, TSENSE

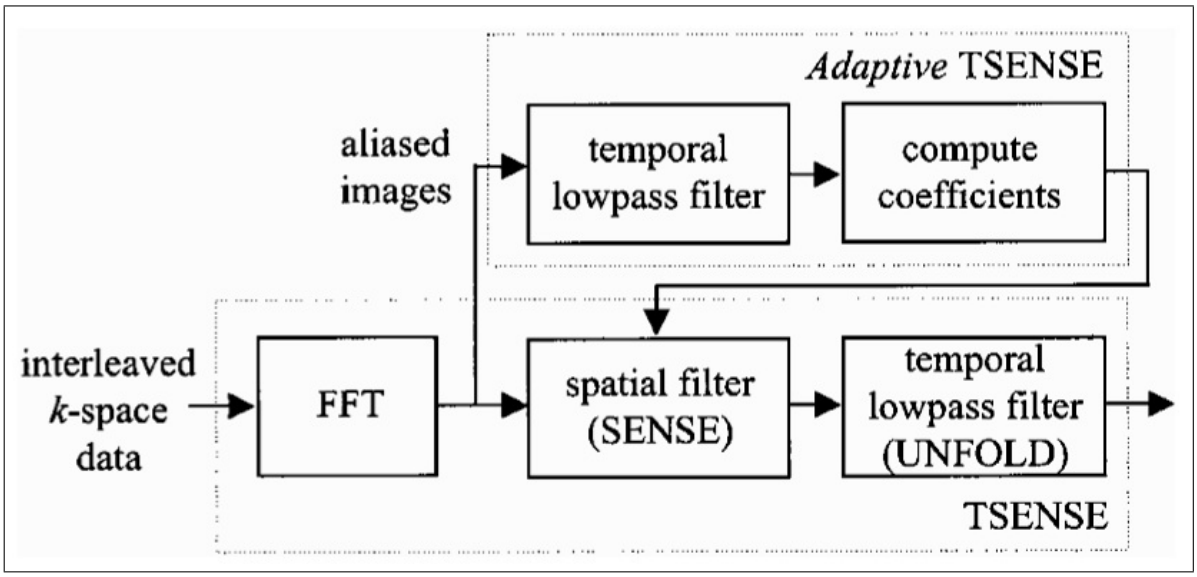

Figure 2.2 TSENSE block diagram [40].

can also be implemented with variable density k-space sampling schemes.

Let $\hat{f}_{sense} = UI$ $(U = (C^H C)^{-1} C^H)$ represents the SENSE estimation process defined in Eq. 2.3. TSENSE estimation process can be defined as

$$\begin{aligned} \hat{f}_{tsense} &= \hat{f}_{sense} * h_{LPF}(t) \\ &= \hat{f}_{unfold} \end{aligned} \tag{2.5}$$

where $h_{LPF}(t)$ is the temporal low-pass filter impulse response. The block diagram of the TSENSE algorithm is given in Figure 2.2 [40].

SNR relation in TSENSE can be expressed as

$$\begin{aligned} SNR_{tsense} &= \frac{SNR_{full}}{\sqrt{R}g}\sqrt{\frac{BW_{full}}{BW_{unfold}}} \\ &= SNR_{sense}\sqrt{\frac{BW_{full}}{BW_{unfold}}} \end{aligned} \tag{2.6}$$

where BW_{full} and BW_{unfold} represent the two sided noise equivalent temporal bandwidths for full and reduced FOV k-space acquisitions. Generally BW_{full}/BW_{unfold} is

around 1.25, and thus TSENSE has a slightly increased SNR compared to SENSE.

TSENSE provides increased artifact suppression when compared to SENSE and UNFOLD alone, and suits well to the real-time dynamic imaging as it is an adaptive algorithm that tracks changes in the coil sensitivity profiles.

2.2 *k-space* Domain Reconstruction

SMASH, introduced by Sodickson *et al.* in 1997 [32], was the first properly experimented (*in vivo*) pMRI reconstruction algorithm. With SMASH, reconstruction takes place in the k-space. SMASH lead to many other algorithms such as AUTO-SMASH [34], variable density AUTO-SMASH (VD-AUTO-SMASH) [35] and Generalized SMASH [37]. But the most popular k-space based pMRI reconstruction algorithm was GRAPPA, introduced by Griswold *et al.* in 2002 [36]. GRAPPA, unlike SMASH, does not require the knowledge of coil sensitivity profiles; acquired data itself is used to calculate reconstruction weights that will estimate unacquired k-space samples.

In the following section, SMASH, GRAPPA and TGRAPPA algorithms will be briefly explained.

2.2.1 SMASH

SMASH algorithm suggests that a linear combination of the estimated coil sensitivies can be used to replace missing phase-encoding lines during accelerated imaging. By combining sensitivity values C_k with linear weights $n_k^{(m)}$, composite sensitivity profiles C_m^{comp} can be obtained where m is the order of the sinusoidal spatial sensitivity variations of the order m:

$$C_m^{comp}(y) = \sum_{k=1}^{N_c} n_k^{(m)} C_k(y) \cong e^{im\Delta k_y y} \tag{2.7}$$

with $\Delta k_y = 2\pi/FOV$. The estimation of the linear weights $n_k^{(m)}$ is realized by a fitting procedure: sensitivity values C_k are fitted to the spatial harmonic $e^{im\Delta k_y y}$.

$S_k(k_y)$, signal of a given coil k can be expressed using the Fourier transformation of the spin density $\rho(y)$ weighted with the coil sensitivity profile $C_k(y)$ of the coil in question:

$$S_k(k_y) = \int dy \rho(y) C_k(y) e^{ik_y y}. \tag{2.8}$$

An expression to generate shifted k-space lines $S(k_y + m\Delta k_y)$ can be derived using Eq. 2.7 and Eq. 2.8 as follows:

$$\begin{aligned} \sum_{k=1}^{N_c} n_k^{(m)} S_k(k_y) &= \int dy \rho(y) \sum_{k=1}^{N_c} n_k^{(m)} C_k(y) e^{ik_y y} \\ &\cong \int dy \rho(y) e^{im\Delta k_y y} e^{ik_y y} = S^{comp}(k_y + m\Delta k_y). \end{aligned} \tag{2.9}$$

This results shows that skipped phase encoding lines can be recovered by linear combination of the receiver coil signals, which is normally realized by magnetic field gradients.

This result indicates that linear combination of component coils can actually be used to generate k-space shifts in almost the same manner as magnetic field gradients in conventional phase-encoding. However, SMASH, in general, strongly depends on coil configurations, hence, requires knowledge of the coil sensitivity profiles for a given configuration. This may be problematic, especially when coil sensitivities cannot be known exactly, or when coil sensitivities change during imaging due to motion. AUTO-SMASH introduced the acquisition of an auto-calibration signal (ACS) during the actual scan to estimate the coil sensitivities. With AUTO-SMASH, $R-1$ calibration lines are acquired in the center of the k-space, where R is the acceleration factor, and they are used to derive linear weights $n_k^{(m)}$. VD-AUTO-SMASH further improves this scheme, and more ACS lines are acquired in the center of the k-space for better reconstruction. Additionally, in VD-AUTO-SMASH, ACS lines can be integrated in the final reconstruction step for further improved image quality.

2.2.2 GRAPPA

GRAPPA [36] is one of the mostly used parallel MR imaging methods derived originally from SMASH like AUTO-SMASH and VD-AUTO-SMASH. The main reason behind these derivation is the fact that SMASH requires prior knowledge of the coil sensitivities, which is not practical most of the time. AUTO-SMASH, VD-AUTO-SMASH and GRAPPA, all of these methods use fully encoded auto-calibration data from the center of the k-space as a reference data for estimating coil sensitivity information that will be used to fill up undersampled k-space. GRAPPA is the most generalized version of them all, as the fitting procedure involves data from all coils. However, acquiring fully sampled auto-calibration data slows down the whole acquisition process which implies longer minimum scan time.

In GRAPPA, estimation of the missing data for a particular coil requires all the acquired data from all the coils. Let N_c be the number of receiver coils and R be the acceleration rate, $(R-1)$ sample estimates must be synthesized for each acquired sample of each coil in order to fully fill out the data set. Blocks (or kernels) of size $[Y \times X]$ are used in the estimation of the missing samples of a given coil. Size of this block compromises the image quality and reconstruction speed. Bigger blocks mean slower but more accurate estimation of the unacquired samples. $[4 \times 5]$ and $[2 \times 5]$ blocks $(R = 3)$ are represented in Figure 2.3.

$R-1$ unacquired points in a block for a given coil c_k are estimated by linear combination of the acquired samples in the corresponding blocks from all coils. This methodology is represented for a given coil C in Figure 2.4 on a rate 2 data set acquired with N coils. Coil C has N distinct blocks of size $[4\times5]$, each dedicated to convolve with one coil data. N convolution operations are performed, and summed up to estimate one skipped sample for coil C. The block is slided over entire k-space. Once whole k-space is spanned, we have a fully sampled data set for coil C. Subsequent to that, inverse Fourier Transform will provide reconstructed image for coil C.

Theoretically, GRAPPA relation can be described as follows. Let $S(c, i, j)$ be

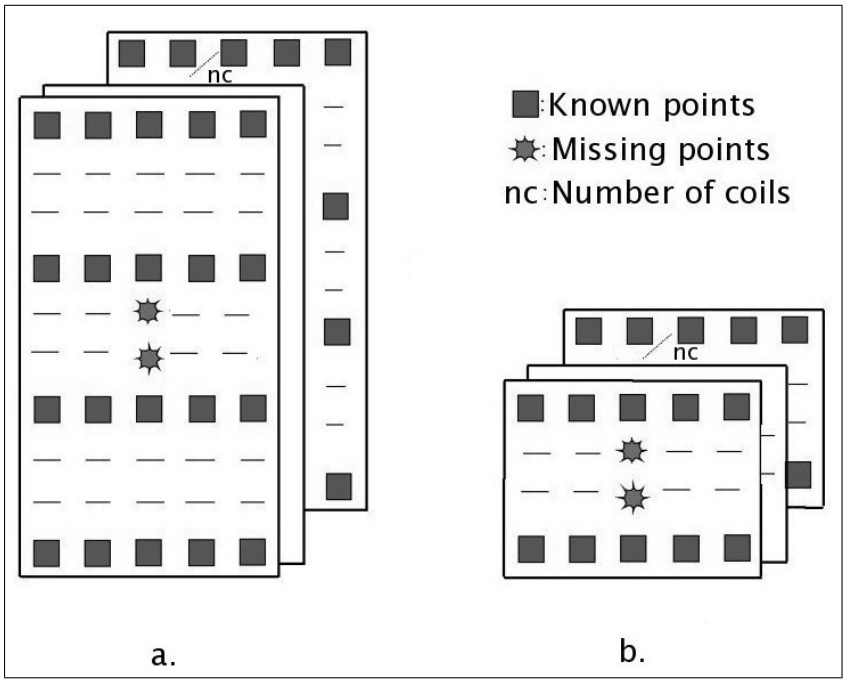

Figure 2.3 (a) $R = 3$, block of size 4×5 (4 lines, 5 points per line). (b) $R = 3$, block of size 2×5 (2 lines, 5 points per line).

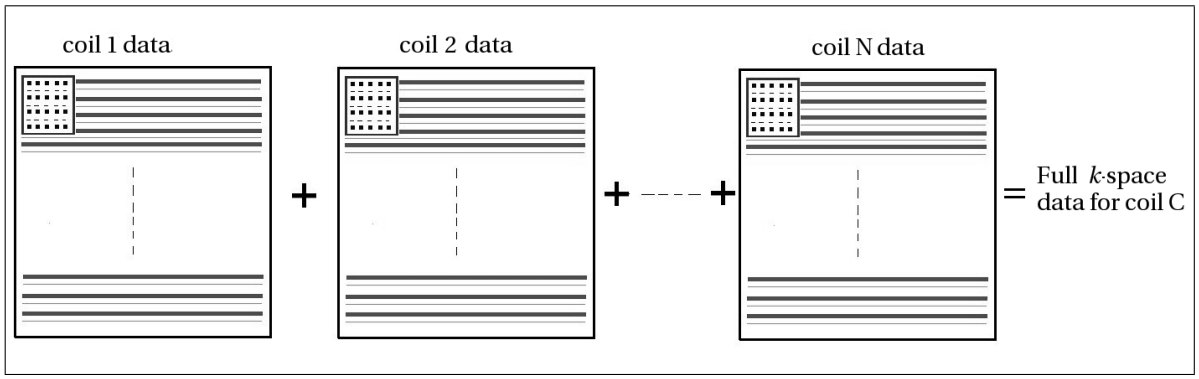

Figure 2.4 GRAPPA k-space filling scheme is represented for a given coil C (rate 2 data set). Small rectangles represent $[4 \times 5]$ GRAPPA blocks (kernels) of coil C that will be used to estimate its missing samples using data from all coils. Thin lines are unacquired samples, and thick lines represent acquired samples of coil C. The kernel is slided over entire k-space to estimate every missing k-space samples. The sum of all the convolutions (kernel values and acquired samples) at a given location provides one sample estimate at the center of the kernel for coil c. When whole k-space is spanned, full k-space data for coil C is obtained.

the (i, j) element of the acquired data in the corresponding block of data from coil c (i.e. the square samples in Figure 2.3). Let $W_{c_k,p}(c, i, j)$ be the set of GRAPPA coefficients used to weight the samples $S(c, i, j)$, and $T_{c_k,p}$ be the pth missing sample of the target block c_k (i.e. the stars in Figure 2.3) to synthesize. This estimation process can be expressed as

$$\sum_{c=1}^{N_c} \sum_{i=1}^{X} \sum_{j=1}^{Y} S(c, i, j)\, W_{c_k,p}(c, i, j) = T_{c_k,p}. \tag{2.10}$$

On the fully sampled auto-calibration signal, the only unknown in Eq. 2.10 are the weighting coefficients $W_{c_k,p}(c, i, j)$. Therefore, these weighting coefficients can be calculated using least squares estimation of the overdetermined system formed by considering all the blocks in the auto-calibration data set. Subsequent to determination of the weights, Eq. 2.10 is used to fill the k-space data for each coil. Once fully sampled data set for all coils are obtained, inverse Fourier Transform is applied individually to each coil data, and per coil images are obtained. These individual coil images are then combined using *root-sum-of-squares* (RSS).

GRAPPA's weight calculation process is computationally demanding, but fortunately there is no need to calculate a new set of weights per frame. A set of fixed GRAPPA weights can be used to estimate unacquired samples of multiple consecutive frames. In theory, one will need a new set of GRAPPA weights only when coil sensitivities are changed; i.e. when scan plane changes during imaging.

Filling up the k-space by estimating the missing samples is computationally less demanding than the weight calculation. However, this is still a slow operation when compared to image domain based parallel imaging methods (e.g. (T)SENSE). This is due to the fact that GRAPPA employs a convolution type process at each sample, as opposed to a pixel-by-pixel multiplication of the image domain based reconstruction methods. Additionally, GRAPPA processes data from all receiver channels, thereby increasing demands on computation and data management. Reconstruction performance of GRAPPA algorithm depends on the number of the receiver coils N_c, acceleration rate R, acquisition matrix size, and the block size. Furthermore, unlike SENSE and

SMASH, GRAPPA reconstructs per coil images and thus final image reconstruction requires an additional RSS combining of all coil images. This extra step drastically decreases the reconstruction performance.

2.2.3 TGRAPPA

The main difference between TGRAPPA and GRAPPA lies on the data acquisition. In GRAPPA, fully sampled central k-space data are acquired as ACS lines with every new acquisition frame. On the other hand, TGRAPPA uses time interleaved acquisition scheme as in UNFOLD and TSENSE. This acquisition scheme makes it possible to obtain a full set of ACS lines every R frames. Additionally, new set of ACS data can be obtained with every acquisition. As a result, TGRAPPA is successful in keeping track of changes in coil sensitivities, and suits very well to the dynamic non-gated real-time imaging. It is also beneficial during real-time MRI guided interventions where slice plane often changes. Change in the slice plane requires to update coil sensitivity profiles during the experiment. Additionally, coil sensitivities may change because of the respiratory movements, or patient motion.

Figure 2.5 compares GRAPPA and TGRAPPA acquisition schemes. In GRAPPA, fully sampled central k-space data are acquired. Because of the additional acquisitions scan time increases. With TGRAPPA acquisition scheme, no additional acquisitions (phase encoding steps) are required. Every consecutive R frames provide fully sampled (same resolution as the actual scan) ACS data set. Consequently, another advantage of TGRAPPA is that the size of the ACS data set is totally flexible. With GRAPPA, number of ACS lines to use is always limited because of the already increased minimum scan time. Thus, data acquisition with TGRAPPA may be tens of milliseconds faster than GRAPPA[2] .

Another advantage of variable ACS data set size is the ability to compromise

[2]Data acquisition speedup of "TGRAPPA versus GRAPPA" depends on the number of centrally acquired GRAPPA ACS lines.

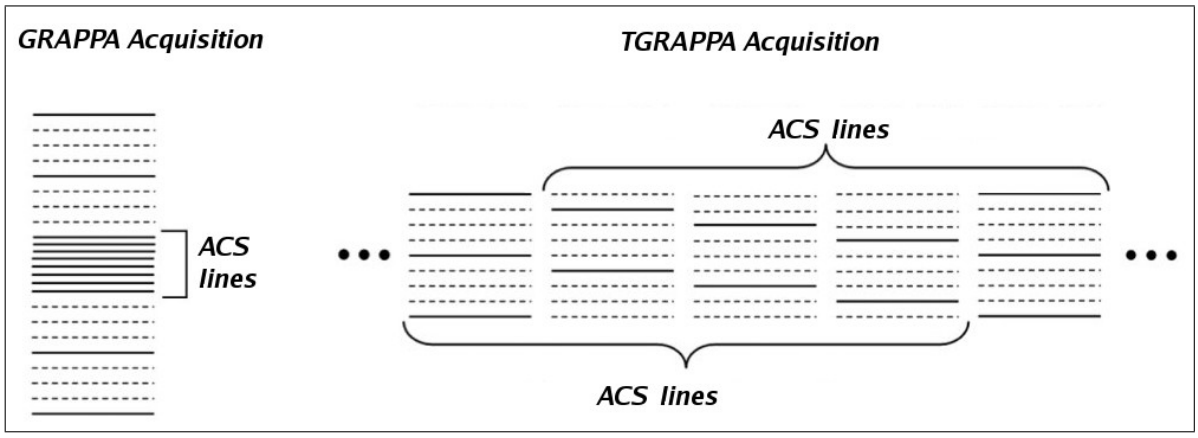

Figure 2.5 GRAPPA and TGRAPPA acquisition schemes are shown ($R = 4$). In GRAPPA, fully sampled data from the center of the k-space is acquired as a set of ACS lines with each frame. On the other hand, the interleaved acquisition scheme of TGRAPPA provides full set of ACS lines every R frames.

between weight update speed and image quality. The more ACS lines are used, the better is the image quality (see Figure 2.8). However, bigger calibration data set implies slower weight calculation. It is possible to dynamically change the number of ACS lines to use to adapt imaging needs. The effect of the number of ACS lines is studied in detail by Breuer *et al.* in original TGRAPPA paper [38].

In practice, an integration window of size $3 \times R$ is used to generate ACS lines. Temporal average of $3 \times R$ frames provides one full set of ACS data. After the initial set of $3 \times R$, every new frame will provide a new set of ACS data, and thus new set of GRAPPA weights.

2.3 Towards HTGRAPPA: Real-time Low-latency Parallelized TGRAPPA Based Reconstruction

GRAPPA algorithm can be beneficial over SENSE for a couple of reasons. First, SENSE requires the knowledge of coil sensitivity profiles. Accurate coil sensitivities cannot be obtained in inhomogeneous regions, such as lungs and abdomen. As a result, SENSE image quality may suffer from inaccurate coil sensitivity maps. On the

other hand, (T)GRAPPA uses global k-space information to generate the reconstruction weights. Thus, it is not affected by local inhomogeneities [68]. Second, if the FOV is smaller than the object, SENSE algorithm exhibits pre-folding artifacts on the reconstructed images. GRAPPA algorithm is free of such artifacts [44].

Real-time implementation of (T)SENSE is previously presented by Guttman *et al.* [52]. However, there was no real-time low latency implementation of (T)GRAPPA algorithm. First, highly parallelized conventional TGRAPPA implementation is developed. However, convolution based, acceleration rate and block size dependent reconstruction scheme is slow to perform low-latency real-time imaging with high acceleration rates ($R > 2$) and high number of receiver coils. Second, highly parallelized image domain TGRAPPA implementation is developed as explained in [63–65]. As the convolution operation is transformed to a simple multiplication, image domain TGRAPPA is much faster, and reconstruction speed is independent of acceleration rate and block size. However, the use of multiple set of weights as in original TGRAPPA, and RSS combination required to obtain final reconstructed images are penalizing the reconstruction speed: real-time imaging with higher rates ($R > 3$) and with high number of receiver coils ($N_c > 15$) is still not practical. We propose a new hybrid domain TGRAPPA based reconstruction scheme, HTGRAPPA, where multiple set of weights per coil are combined into one set of unmixing coefficients. Additionally, RSS combining required in TGRAPPA reconstruction is replaced by a $B1$-weighted combination as in SENSE. Experimental results shows that HTGRAPPA reconstruction algorithm is up to 180 times faster than the conventional, highly parallelized, TGRAPPA algorithm while maintaining all the advantages of GRAPPA algorithm (e.g. no pre-folding artifacts).

In the following sections, the development stages of our real-time low latency highly parallelized TGRAPPA based hybrid domain image reconstruction algorithm HTGRAPPA [41, 42] will be explained in detail.

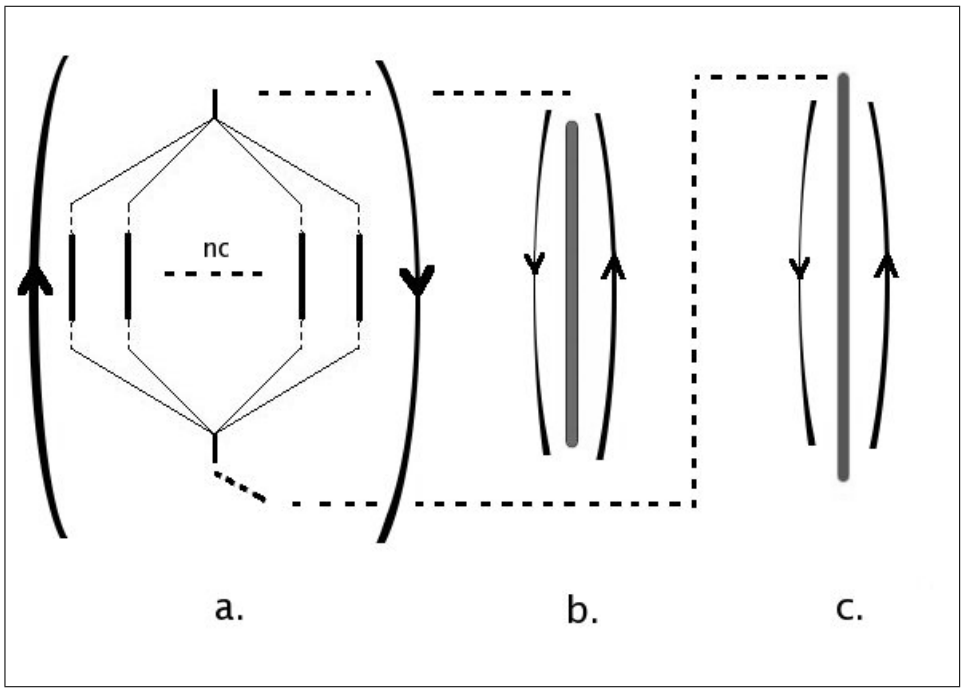

Figure 2.6 (a) With each new data set, *nc* threads are used to reconstruct images. Most recent weight-set is used. (b) One thread is dedicated for GRAPPA weight calculation. It is totally asynchronous to the reconstruction. It updates the weight-set as fast as possible with newly coming data. (c) Another thread is dedicated to the display of the final image.

2.3.1 Parallel TGRAPPA Implementation

A prototype of TGRAPPA algorithm is implemented in *C/C++* for better understanding of this reconstruction scheme. Weight-set calculation and image reconstruction are computed in parallel using 8 dual-core AMD Opteron 8220 processor (2.8 GHz) on Linux operating system with symmetric multi-processing (SMP) configuration. Parallelization in reconstruction is supplied by OpenMP for prototyping. GNU Compiler Collection (GCC) version 4.2.2 used as a compiler, as it has full OpenMP support. Open Graphics Library (OpenGL) used for displaying reconstructed image. Matrix inversion is realized by Automatically Tuned Linear Algebra Software (ATLAS) library (`http://math-atlas.sourceforge.net`).

Weight-set calculation, the most time consuming step of the processing, is performed asynchronously to the image reconstruction. One dedicated thread repeatedly calculates and updates the TGRAPPA weight-set, while images are reconstructed us-

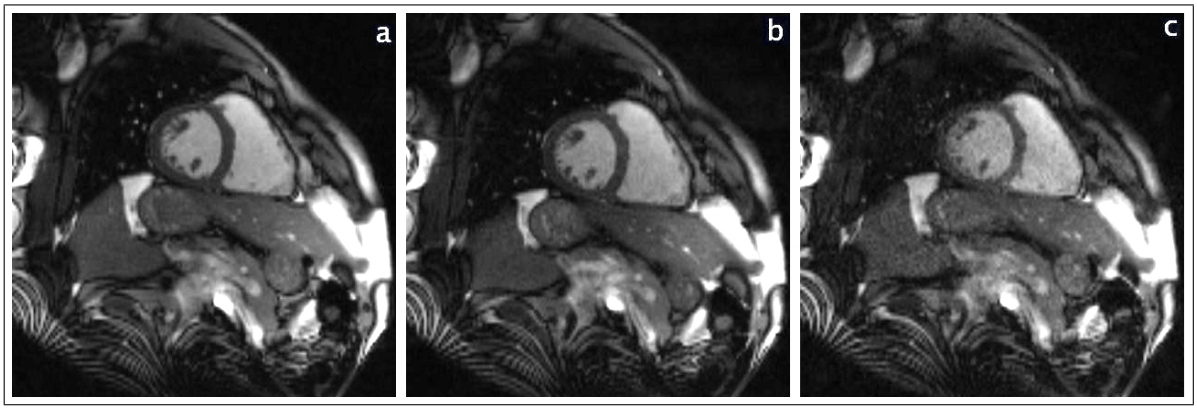

Figure 2.7 Reconstructed images from undersampled cardiac data sets using TGRAPPA (matrix size = 192×144). (a) $R = 2$ (b) $R = 3$ (c) $R = 4$.

ing the most recent weight set and one thread per coil. Another thread is dedicated to display function, and displayed each newly reconstructed image. The main program creates NC OpenMP threads to reconstruct images using the most recent weight-set. The parallelism of the implementation is represented in Figure 2.6.

2.3.1.1 Results. Several real-time cardiac images from a healthy individual are acquired using Siemens Magnetom Espree 1.5 T to validate our implementation. The sequence used is steady-state free-precession (SSFP) with TR=3 ms, flip angle=45 degrees, bandwith=800Hz/pixel with an acquisition matrix of 192×144. S^5FP [69] is used for suppression of fat and flow artifacts. 15 receiver coils are used; 9 elements on the chest and 6 under the spine. In order to improve temporal resolution, acceleration factors of R=2, R=3, and R=4 are used. Images reconstructed using parallel TGRAPPA implementation for validation purposes are given in Figure 2.7 for acceleration rates of 2, 3, and 4.

Several block sizes and different number of ACS lines are used to reconstruct rate 2, rate 3 and rate 4 data sets to perform full performance analysis of our parallelized TGRAPPA implementation. Weight-set calculation and reconstruction performances are calculated in seconds for each case. Table 2.1 represents the performance results on rate 2 data set for block sizes of $[2 \times 3]$, $[4 \times 3]$. 24 and 32 ACS lines are deployed

during weight-set calculation. Performance results for rate 3 data set are given in Table 2.2. Block sizes of [2 × 3], [2 × 5], [4 × 3] and [4 × 5] are used during the reconstruction. GRAPPA weights are calculated using 32 and 48 ACS lines. Table 2.3 depicts performance of our parallelized TGRAPPA implementation on rate 4 cardiac data set. Block sizes of [2 × 3], [2 × 5], [4 × 3] and [4 × 5] are used to reconstruct images. 48 and 60 ACS lines are used as calibration data set during the calculation of the weights.

Table 2.1
Reconstruction and weight-set calculation performances (for 24 and 32 ACS lines) in seconds for $R = 2$

Block Size	Reconstruction	Weight-set	
		24	32
[2 × 3]	0.25	0.31	0.63
[2 × 5]	0.36	0.56	1.46
[4 × 3]	0.45	1.18	1.98
[4 × 5]	0.69	2.53	4.86

Table 2.2
Reconstruction and weight-set calculation performance in seconds for $R = 3$

Block Size	Reconstruction	Weight-set	
		32	48
[2 × 3]	0.32	0.59	0.98
[2 × 5]	0.49	1.6	2.54
[4 × 3]	0.58	1.52	2.76
[4 × 5]	0.94	4.63	8.12

After the performance measurement, SNR values are measured on an accelerated data set for several GRAPPA weights parameters (block size, and number of ACS lines). The results are represented in Figure 2.8. As expected, increased block sizes and number of ACS lines increase the SNR of the reconstructed images, as the accuracy of the calculated reconstruction weights is better. However, increased SNR comes with the cost of slow reconstruction performance. Performance values indicates that real-

Table 2.3
Reconstruction and weight-set calculation performance in seconds for $R = 4$

Block Size	Reconstruction	Weight-set	
		48	60
$[2 \times 3]$	0.35	1.05	1.40
$[2 \times 5]$	0.56	2.73	3.62
$[4 \times 3]$	0.64	2.84	4.93
$[4 \times 5]$	1.06	7.89	32.34

time low latency imaging with high acceleration rates is impossible on our current workstation, even if a smaller block size is preferred by sacrificing SNR.

2.3.1.2 Discussion. The results show that increased block size implies slower reconstruction, as expected. Using more ACS lines increase the accuracy of the weights and provides better image quality, however slow down the weight calculation. Additionally, our current workstation cannot handle the overload caused by weight calculation and reconstruction threads. Distributed computing for weight calculation does not help increase the reconstruction performance either, because of the large data to transfer over network ($8 * N_c * N_s * N_{ACS}$ bytes to transfer, where N_c is the number of receiver coils, N_s is the number of samples per phase encoding, N_{ACS} is the number of ACS lines to use).

Bigger block size and bigger ACS data set provide better weight estimates. With $R = 2$, 6 images per second can be reconstructed. This drops to 5 for $R = 3$. With $R = 4$, moderate image quality can be obtained by the rate of 2 images per second. Therefore, it is not possible to reconstruct images in real-time when $R > 2$, as data acquisition is faster than the reconstruction (approximately 10 frames per second (fps) acquisition for rate 3, and 18 fps acquisition for rate 4 with a matrix size of 192×108).

For $R = 2$, 24 ACS lines with $[2 \times 3]$ blocks provide visually acceptable image

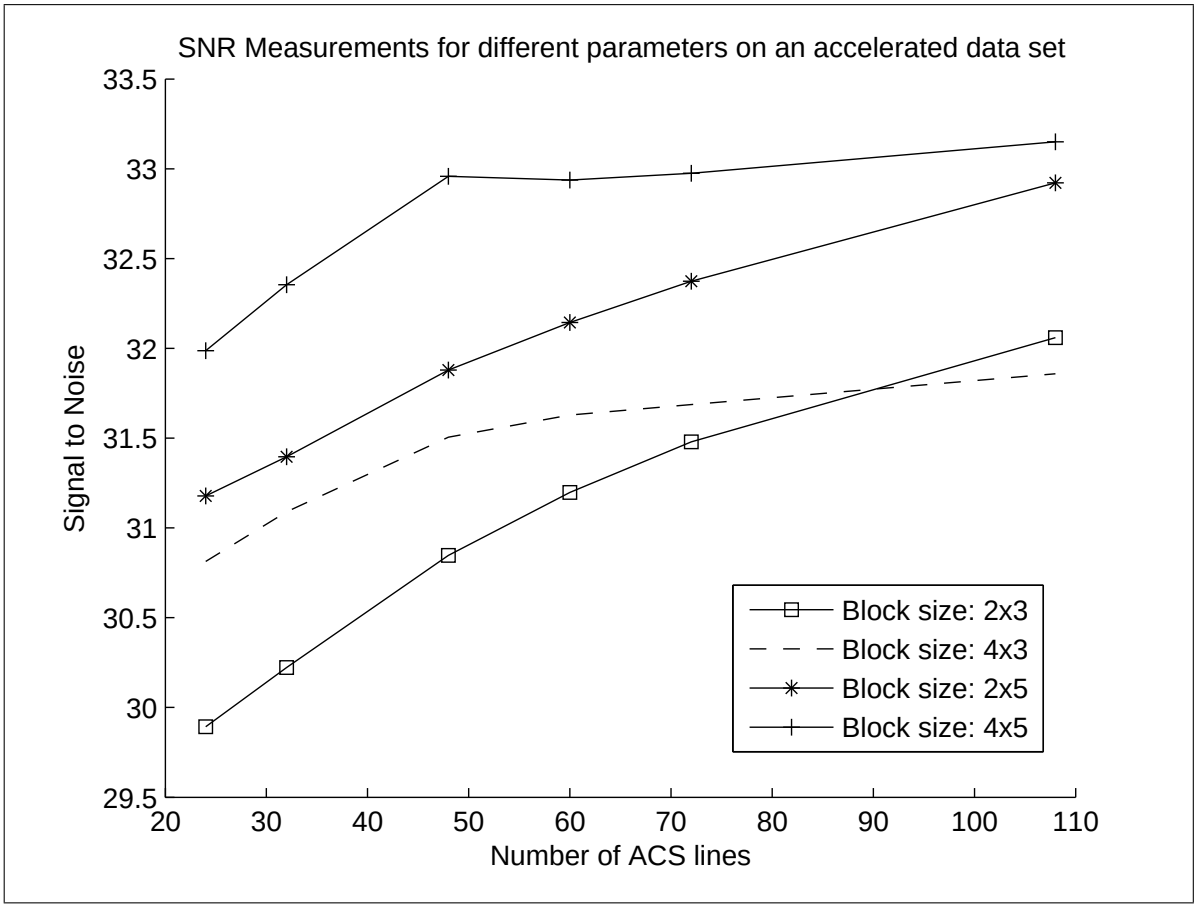

Figure 2.8 SNR values for different GRAPPA weights parameters

.

quality. For $R = 3$, 48 ACS lines with $[2 \times 3]$ blocks reconstruct visually acceptable image quality. Moderate image quality is obtained using 48 ACS lines with block size of $[2 \times 5]$ with rate 4 data set. However, further experiments (see Figure 2.8) show that the best results are obtained with block size of $[4 \times 5]$ and when full ACS data set are used, as stated in original TGRAPPA paper [38].

This prototype provided valuable information about optimum block size and ACS lines number for adequate image quality and reconstruction performance. Additionally, the bottleneck in TGRAPPA algorithm is identified as the convolution-like reconstruction process. Rate, GRAPPA block size dependent reconstruction is found to be slow for low-latency real-time reconstruction when $R > 2$. If GRAPPA weights can be transformed into image domain, convolution-like reconstruction will be performed with simple multiplication. That way, rate and block size dependency of the (T)GRAPPA algorithm can be removed, and faster reconstruction performance can be achieved.

2.3.2 Highly Parallel Image Domain Real-time TGRAPPA Implementation

TGRAPPA algorithm can be applied in the image domain after proper transformation [63–65]. To this end, the $p = [1 : R - 1]$ weight-sets $w_{c_0,p}(c, \cdot, \cdot)$ that relate the acquired data in coil c to the missing data from coil c_0 combined to yield N_c k-space convolution kernels $W^{set}_{c_0 c}(k_x, k_y)$ for each coil, c_0. Center points of these kernel center points must be set to Delta function, $\delta_{c_0 c}$ for preserving original sample values. After setting central values, kernels are then zero filled to the same matrix size as the actual acquisition. Subsequent to zero filling, k-space domain GRAPPA weights can be transformed to the image domain by inverse Fourier Transformation:

$$W^{img}_{c'c} = \mathscr{F}^{-1}\left(W^{set}_{c'c}\right) \tag{2.11}$$

where $W^{img}_{c_0 c}$ represents image domain weights of coil c_0 to apply to data coming from coil c. Once the image domain weights are known, the under-sampled k-space data,

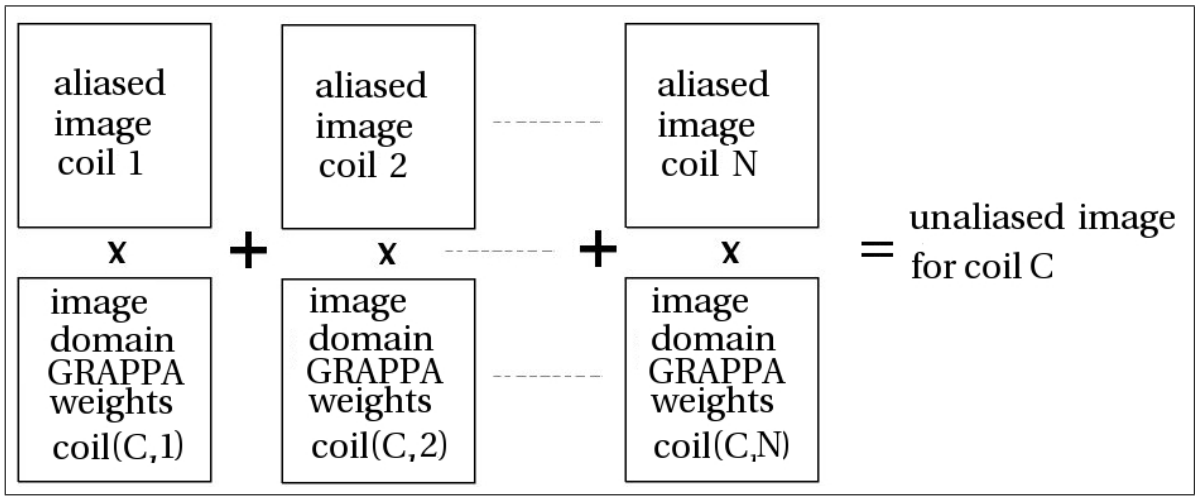

Figure 2.9 Image domain TGRAPPA reconstruction scheme is represented for a given coil C. Aliased images from each coil is pixel-wise multiplied with the corresponding image domain weights of coil C. The results of the multiplications are summed up to obtained unaliased image for coil C. Once all coil images are obtained, final image is calculated by RSS combining.

.

$D_c(k_x, k_y)$ from each coil, $c = [1 : N_c]$, is transformed into image-domain, yielding aliased images $I_c^{aliased}$:

$$I_c^{aliased} = \mathscr{F}^{-1}\left(D_c\right). \tag{2.12}$$

Image domain weights can now be applied to aliased images to reconstruct per coil unaliased images I_{c_0} as follows:

$$I_{c_0} = \sum_{c=1}^{N_c} I_c^{aliased} \cdot W_{c_0 c}^{img} \tag{2.13}$$

where $\cdot$ denotes pixel-by-pixel multiplication operator. One coil image reconstruction process of image domain TGRAPPA algorithm is represented in Figure 2.9. Aliased images from each coil are pixel-wise multiplied with corresponding image domain weights of coil C, and the results are summed up to obtain the unaliased image for coil C, as it is shown in Eq. 2.13.Final image is provided by *Root-Sum-of-Squares* (RSS) combining, as in original (T)GRAPPA algorithm.

2.3.2.1 Implementation. Image domain TGRAPPA algorithm is implemented in *C/C++*, and parallelized for high performance. Weight-set calculation and image reconstruction processes are decoupled and computed in parallel using 8 dual-core AMD Opteron 8220 processor (2.8 GHz) on Linux-2.6.16.46-0.12-smp. Parallelization of the reconstruction is achieved using Pthreads (`http://pasc.org`) library, and parallelization in the weight-set calculation is realized by OpenMP (`http://openmp.org/wp/`) library. The code is compiled on GCC (`http://gcc.gnu.org`) version 4.2.2. OpenGL (`http://www.opengl.org`) library is used for displaying reconstructed images. Matrix inversions, used in the least-squares calculation of the weight-set coefficients from the auto-calibration data, is calculated using the ATLAS library (`http://math-atlas.sourceforge.net`). The FFTW (`http://www.fftw.org`) library is used for Fast Fourier Transformations.

N_c threads are created by Pthreads library and dedicated to reconstruct per coil images. Another thread is created by Pthreads library and is dedicated to weight calculation process. This thread is parallelized using OpenMP library. OpenMP is preferred because it facilitates dynamic thread allocation during the execution. At initialization, or when there is a scan plane change, more threads are assigned to weight calculation process for further increased weight update performance. Weights are calculated in the background, while image reconstruction threads are reconstructing each newly acquired frame using latest set of weights.

Note that, it is possible to dynamically change the number of ACS lines and block size to use. That way, image quality can be adjusted on the fly regarding the instant imaging needs. Block diagram of the implementation is given in Figure 2.10.

2.3.2.2 Results. Several real-time cardiac data set from healthy individuals are acquired using Siemens Magnetom Espree 1.5 T (Siemens Medical Solutions, Erlangen, Germany). The sequence is S^5FP [70] with TR=3 ms, flip angle=45 degrees, bandwith=800Hz/pixel with an acquisition matrix of 192×144. A $SSFP$ sequence is used for suppression of fat and flow artifacts. 15 receiver coils are used; 9 elements on

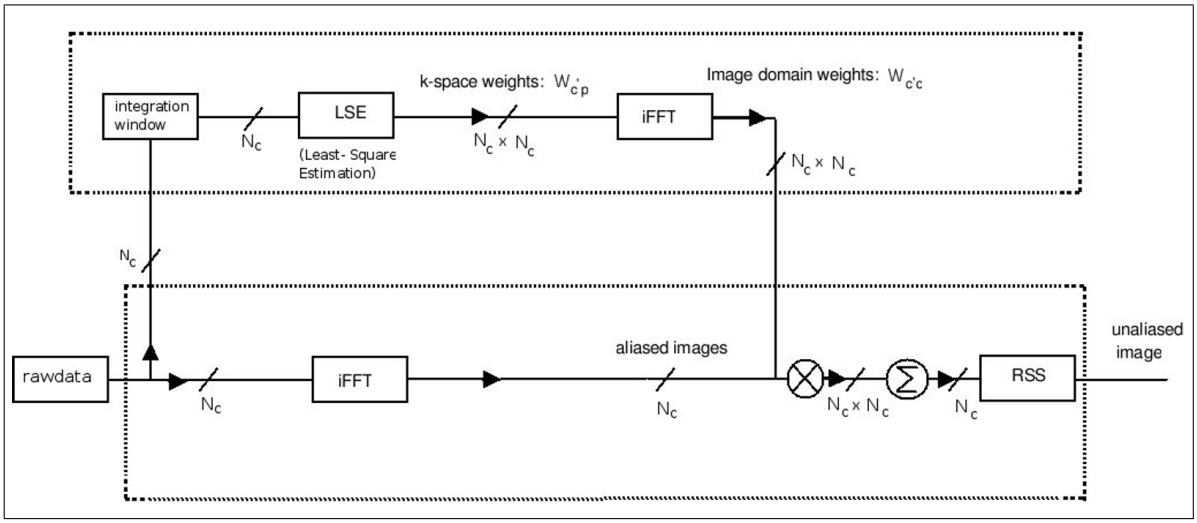

Figure 2.10 Block diagram of the reconstruction algorithm is given. Dotted rectangles represents parallel regions. The upper dotted rectangle represents the weight-set calculation process, and the lower dotted rectangle represents the reconstruction process. These processes are computed asynchronously.

the chest and 6 under the spine. In order to improve temporal resolution, parallel imaging technique TGRAPPA is employed with acceleration factors of R=2, R=3, and R=4.

The performance results for the reconstruction and first time weight-set calculation are given in Table 2.4 and 2.5 for both conventional and image domain TGRAPPA implementation (both parallelized). For both conventional and image domain TGRAPPA implementations, 60 auto-calibration lines are used for weight-set calculation, and acceleration rates of $R = 2$, $R = 3$ and $R = 4$ are used, using block sizes of $[2 \times 3]$, $[2 \times 5]$, $[4 \times 3]$ and $[4 \times 5]$.

2.3.2.3 Discussion. Table 2.4 indicates that image domain TGRAPPA reconstruction outperforms conventional TGRAPPA reconstruction. Furthermore, image domain reconstruction speed remains unaffected by increased acceleration rates and by the block size. As a result, the trade-off between reconstruction speed and image quality is eliminated when image domain reconstruction is preferred. On the other hand, weight calculation slowed down due to the transformation the k-space GRAPPA

Table 2.4
Image reconstruction times in seconds for conventional (conv.) and image domain (img.) TGRAPPA using RSS combining with for different R values and block sizes of $[2 \times 3]$, $[2 \times 5]$, $[2 \times 7]$, $[4 \times 3]$ and $[4 \times 5]$

R	$[2 \times 3]$		$[2 \times 5]$		$[4 \times 3]$		$[4 \times 5]$	
	conv.	img.	conv.	img.	conv.	img.	conv.	img.
2	0.330	0.065	0.540	0.065	0.650	0.065	1.050	0.065
3	0.430	0.065	0.720	0.065	0.840	0.065	1.350	0.065
4	0.500	0.065	0.810	0.065	0.950	0.065	1.550	0.065

Table 2.5
Weight-set calculation times in seconds for conventional (conv.) and image domain (img.) TGRAPPA using RSS combining for different R values and block sizes of $[2 \times 3]$, $[2 \times 5]$, $[2 \times 7]$, $[4 \times 3]$ and $[4 \times 5]$

R	$[2 \times 3]$		$[2 \times 5]$		$[4 \times 3]$		$[4 \times 5]$	
	conv.	img.	conv.	img.	conv.	img.	conv.	img.
2	0.270	0.480	0.650	0.870	0.770	0.990	2.100	2.320
3	0.290	0.520	0.700	0.920	0.820	1.040	2.120	2.340
4	0.320	0.550	0.750	0.970	0.870	1.090	2.160	2.380

weights into the image domain. Additional processing time for this transformation depends on the number of receiver coils and on the image matrix. The total loss of time due to the additional operations is measured as 0.22 seconds, which is considered to be reasonable for the application.

Image domain TGRAPPA reconstruction image quality is on par with conventional TGRAPPA as expected [63–65]. But more importantly, acceleration rate and block size independent reconstruction scheme provided much faster reconstruction rates on our cardiac data set (constant reconstruction performance of 0.065 seconds) compared to conventional TGRAPPA (reconstruction performance as low as 1.550 seconds).

Even though image domain TGRAPPA reconstruction outperforms conven-

tional TGRAPPA, it is still slow for high acceleration rates on our current reconstruction computer. Our sequence can acquire 18 fps on rate 4, while image domain TGRAPPA can only reconstruct 15 fps (0.065 seconds per frame). Therefore, real-time imaging is practical only up to rate 3 on our current 16 core reconstruction workstation. Despite the fact that acceleration rate and GRAPPA block size dependency of the reconstruction are eliminated, final image reconstruction still involves multiple set of weights per coil, and time consuming RSS combination. Blazingly fast reconstruction can be achieved if multiple per coil weights can be combined into one set of combined weights. That way RSS combination can be eliminated in favor of phased array combiner as in SENSE.

2.3.3 HTGRAPPA: Highly Parallel B1-Weighted Image Domain TGRAPPA Based Reconstruction

Despite the fact that image domain TGRAPPA outperforms conventional TGRAPPA, multiple set of weights per coil and RSS combining handicap desired real-time image reconstruction performance. If one can combine multiple set of weights into one weight-set per coil, and eliminate RSS combining, blazingly fast reconstruction speeds can be achieved.

We present a new algorithm, based on TGRAPPA, where multiple per coil weights are combined into one set of "unmixing coefficients" (one set of weights per coil to remove the aliasing in the image domain), and where RSS combining is eliminated. In this approach, k-space domain GRAPPA weights are calculated in the k-space as done in conventional GRAPPA, and then transformed into the image domain. After the transformation, sensitivity maps (so called B1-map estimates) are estimated using ACS data set. Fully sampled ACS data is Fourier transformed to the image domain for each coil. Then each coil images are combined using RSS method, to obtain the final image. Following that, each coils image is pixel by pixel divided to the final image. That way, per sensitivity maps are estimated. These estimates are then applied to per-coil image domain weights to linearly combine them into a composite set of unmixing

coefficients.

Unmixing coefficients are applied to the aliased images as pixel-wise multiplication as it is done in SENSE algorithm. That way, RSS combining is eliminated and a single unaliased image is reconstructed with significant computational reduction over previous GRAPPA implementations. Furthermore, noise-weighting is applied to image domain GRAPPA weights to improve SNR in the reconstructed images. As this method is performed in hybrid domain (k-space domain + image domain), we call it hybrid TGRAPPA (HTGRAPPA). HTGRAPPA algorithm will be explained in detail in the following sections.

2.3.3.1 Methods. First, k-space GRAPPA weights must be transformed into image domain (see Eq. 2.11). Once the transformation is done, these new image domain weights must be combined into one set of weights per coil. That way RSS requirement is eliminated from the reconstruction process.

Composite unmixing coefficients of coil c, u_c^{comp} is calculated by pixel-wise multiplication and summing of each $W_{c'c}^{img}$ by complex conjugate of B1-map estimates $b1_{c'}$. These estimates are obtained using temporal average of ACS data from each coil. This relation can be expressed as

$$u_c^{comp} = \sum_{c'=1}^{N_c} W_{c'c}^{img} \cdot b1_{c'}^{*} \tag{2.14}$$

where $\cdot$ represents pixel-wise multiplication operation. Subsequent to the calculation of composite unmixing coefficients, the under-sampled k-space data, $D_c(k_x, k_y)$ from each coil, $c = [1 : N_c]$, is transformed into image-domain, yielding aliased images $I_c^{aliased}$ as in Eq. 2.12. Final image can then be reconstructed using following equation:

$$I^{final} = \sum_{c=1}^{N_c} I_c^{aliased} \cdot u_c^{comp}. \tag{2.15}$$

Eq. 2.15 represents a phased array combiner as in SENSE unmixing. This relation implies that the reconstruction speed for our implementation does not depend on either R

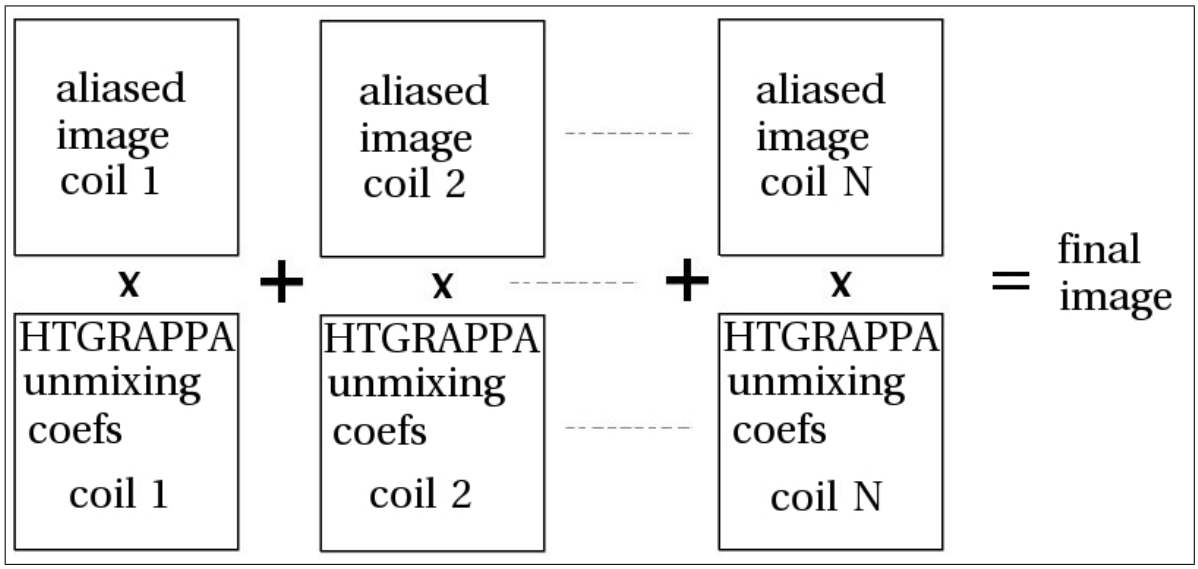

Figure 2.11 HTGRAPPA reconstruction process is represented. Aliased images from each coil are pixel-wise multiplied with corresponding HTGRAPPA unmixing coefficients, and the results are summed to obtained the final image.

or the block size, as in image domain TGRAPPA. Furthermore, computation is greatly reduced as the convolution operation is replaced by a pixel-by-pixel multiplication and summing. Additionally, multi-coil root sum-of-squares magnitude operation is totally eliminated from the final image reconstruction. Figure 2.11 schematically represents the relation given in Eq. 2.15. Note that HTGRAPPA reconstruction is theoretically N_c times faster than image domain TGRAPPA reconstruction (compare Figure 2.9 and Figure 2.11).

SNR of the phased array combined image can be maximized using optimal noise weighting [11]. Let R_n be the noise correlation matrix as defined in [71]:

$$R_{ij} = \frac{1}{N} \sum_{k=1}^{N} n_i(k) n_j^*(k) \tag{2.16}$$

where R_{ij} represents $(i,j)^{th}$ component of matrix R_n, and $n_i(k)$ represents k^{th} noise data sample for coil k. Noise data can be acquired in the presence of solely B_0 field (B_1 field is turned off). Optimum noise weighted composite unmixing coefficients are then computed using $b_1 R_n^{-1}$. Noise weighting improves the SNR of the reconstructed images by a little computation added to weight calculation process.

2.3.3.2 Implementation. HTGRAPPA weight-set calculation and image reconstruction steps are decoupled and implemented in *C/C++* for high performance, as it is done in image domain TGRAPPA implementation. Weight-set calculation and image reconstruction are computed (asynchronously) in parallel using general purpose 8 dual-core AMD Opteron 8220 processor (2.8 GHz) on Linux-2.6.16.46-0.12-smp. Pthreads (http://pasc.org) library is used for the parallelization of the reconstruction, and OpenMP (http://openmp.org) library is used for the parallelization of the HTGRAPPA weight calculation. The code is compiled on GCC (http://gcc.gnu.org) version 4.2.2 for OpenMP support. Matrix inversions are performed using the ATLAS library (http://math-atlas.sourceforge.net). The FFTW library (http://fftw.org) is used for Fast Fourier Transformations. All libraries are specifically compiled from source to provide better optimization.

N_c threads are dedicated to image reconstruction. These threads are created once at the beginning, and the same threads with same variables are used during the whole execution to avoid continuous context switching. Another single dedicated thread is created for weight calculation. This thread is parallelized with OpenMP to update the weights as fast as possible using the most current ACS data. That way, reconstructed images track the changes in the coil sensitivities. Maximum number of CPUs in the system are dedicated to weight calculation at initialization, and when scan-plane is changed, in order to produce useful images as soon as possible. For uninterrupted imaging, view sharing method [72] is used while waiting the first set of weights to be calculated. This method consists of merging the actual acquisition with previously acquired $R-1$ acquisitions to obtain fully sampled k-space data. Then, image is reconstructed using Fourier transform. This method is fast and aliasing is greatly eliminated, but introduces blur due to motion as previously acquired data are used to reconstruct most recent image. Therefore, this method is only used to provide uninterrupted imaging while HTGRAPPA unmixing coefficients are calculated. After the calculation, reconstruction switches back to HTGRAPPA method.

Our implementation permits to dynamically change the number of threads dedicated to weight calculation, as well as the number of ACS lines and GRAPPA block

size to use in the calculation of weights. Dynamically changing the number of threads associated to weight calculation is realized by OpenMP library. OpenMP improves the scalability of our reconstruction program, and makes it possible to dynamically adapt to different imaging needs.

Block diagram of the HTGRAPPA algorithm is presented in Figure 2.12. Every acquired frame is feeded to the integration window to obtain the ACS data, and also inverse Fourier transformed to yield aliased images. At the same time, B1-map estimates and GRAPPA weights are calculated. Subsequent to noise weighted unmixing coefficient calculation, aliasing is removed, and unaliased final image is obtained. Noise data is acquired by turning off the B1 field during the initialization, and when the acceleration rate changes during the intervention. Note that, each dotted rectangle in Figure 2.12 represents individually parallelized steps. Parallelization of the noise correlation matrix is important to provide low-latency reconstruction. If noise correlation matrix calculation is slow, the first couple of frames will be reconstructed with a high latency. Sequential FFT is applied for both reconstruction and weights calculation. If more resources are available, parallel FFT can also be used. For the weights calculation process, updating the integration window, calculation of the least-square estimation for GRAPPA weights, and calculation of final composite unmixing coefficients are realized in parallel using OpenMP fork-join parallelism. ATLAS library is called during least-square estimation to perform matrix operations.

2.3.3.3 Results. Real-time cardiac images from healthy individuals are acquired using a Siemens Magnetom Avanto 1.5 T and short, wide bore Siemens Magnetom Espree 1.5 T (Siemens Medical Solutions, Erlangen, Germany). Imaging parameters are as follows: TR=3ms, flip angle=45 degrees, bandwidth=800Hz/pixel, acquisition matrix=192×108. First cardiac data set are collected using 18 receiver coils; 6 elements on the chest and 12 under the spine. Second cardiac data are acquired using a 32-element array (Invivo Corporation), 16 elements on the chest, 16 elements under the spine. A third cardiac data are acquired using 30 receiver coils. SSFP sequence is used during all acquisitions. The HTGRAPPA imaging technique is employed to improve

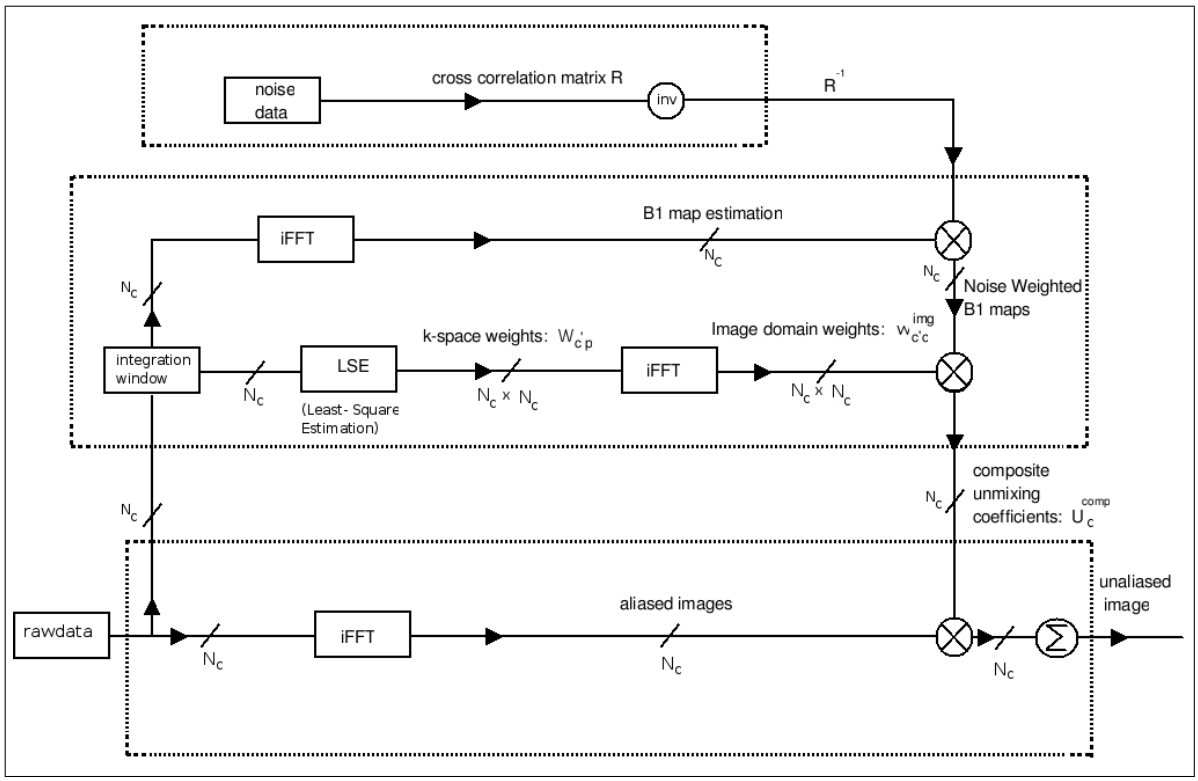

Figure 2.12 Block diagram of the HTGRAPPA reconstruction algorithm is given. Dotted rectangles represents parallel regions. The upper dotted rectangle represents noise weighting step. The middle dotted rectangle represents the weight-set calculation process, and the lower dotted rectangle represents the reconstruction process. These processes are computed asynchronously.

temporal resolution with acceleration factors up to rate, R=4.

HTGRAPPA weight calculation performance results (acquisition matrix of 192× 108, Nc=18) are given in Table 2.6 for acceleration rates of $R = 2, 3, 4$, using various block sizes [2 × 3], [2 × 5], [4 × 3] and [4 × 5]. 48 ACS lines are used during weight calculations. Image domain transformation, generation of composite unmixing coefficients and noise weighting are included in these results. Noise weighting added 0.11 seconds in average to total weight calculation.

The images from the first data set are also reconstructed using conventional TGRAPPA algorithm with asynchronous weight-set calculation, and compared to our hybrid algorithm. Table 2.7 represents side-by-side reconstruction performance comparison (in seconds) of TGRAPPA with HTGRAPPA. As expected, changes in the acceleration rate and GRAPPA block size do not affect HTGRAPPA's reconstruction performance.

Table 2.6
Weight-set calculation times (in seconds) for HTGRAPPA for different acceleration (R) values and block sizes of [2 × 3], [2 × 5], [4 × 3] and [4 × 5] when 16 CPUs are used (18 acquisition coils, acquisition matrix of 192 × 108, 48 ACS lines).

R	[2 × 3]	[2 × 5]	[4 × 3]	[4 × 5]
2	0.495	0.702	0.726	1.320
3	0.510	0.712	0.779	1.339
4	0.530	0.751	0.795	1.428

Images reconstructed in real-time from our first data set ($N_c = 18$, matrix size=192 × 108) are represented in Figure 2.13 for 2 interleaved slices (long and short axis views) using the HTGRAPPA method at acceleration rates 2, 3, and 4. Note that these images are parallel imaging artifact free. Additionally, long axis images that are intentionally prescribed with reduced FOV to cause pre-folding artifacts, are reconstructed robustly, as expected. Small blocks (e.g. [2×3]) reconstruct images with slightly degraded artifact suppression.

Table 2.7
Reconstruction times (in seconds) for conventional TGRAPPA and for HTGRAPPA for different acceleration (R) values and block sizes of [2 × 3], [2 × 5], [4 × 3] and [4 × 5] (18 acquisition coils, acquisition matrix of 192 × 108).

R	[2 × 3]		[2 × 5]		[4 × 3]		[4 × 5]	
	conv.	hyb.	conv.	hyb.	conv.	hyb.	conv.	hyb.
2	0.360	0.008	0.520	0.008	0.630	0.008	1.140	0.008
3	0.430	0.008	0.660	0.008	0.790	0.008	1.310	0.008
4	0.480	0.008	0.750	0.008	0.880	0.008	1.480	0.008

Figure 2.14 compares, from left to right, images reconstructed from our second data set (N_c=32, acquisition matrix of 192 × 108, R=4) using TSENSE, TGRAPPA and HTGRAPPA algorithms. Note that TSENSE and TGRAPPA reconstructions are both off-line, while HTGRAPPA reconstructed images are on-line. Both large and small FOV images are given for comparing the pre-folding artifact robustness of HTGRAPPA with TGRAPPA and TSENSE algorithms. Upper part represents images with larger FOV (380 mm), and lower part represents images with smaller FOV (320 mm).

At initialization, or when slice orientation changes, our implementation automatically switches to view-sharing until the weights are calculated. This behavior is represented in Figure 2.15 using a series of images reconstructed in real-time from our third data set (N_c = 30, acquisition matrix of 192 × 108, R=4) with HTGRAPPA, where slice orientation is changed during scan-time. Reconstruction system switches to view-sharing method while the new weight-set calculation is in progress, and automatically switches back to HTGRAPPA when the new weight-sets are ready to use.

Rate 4 images of full heart cycle (16 images, matrix size 192 × 108, 3/4 partial phase Fourier acquisition) are reconstructed in real time using one fixed set of weights. 20 k-space lines are used to reconstruct each frame. Figure 2.16 shows that HTGRAPPA can reconstruct images of full cardiac cycle without aliasing artifacts or temporal blurring.

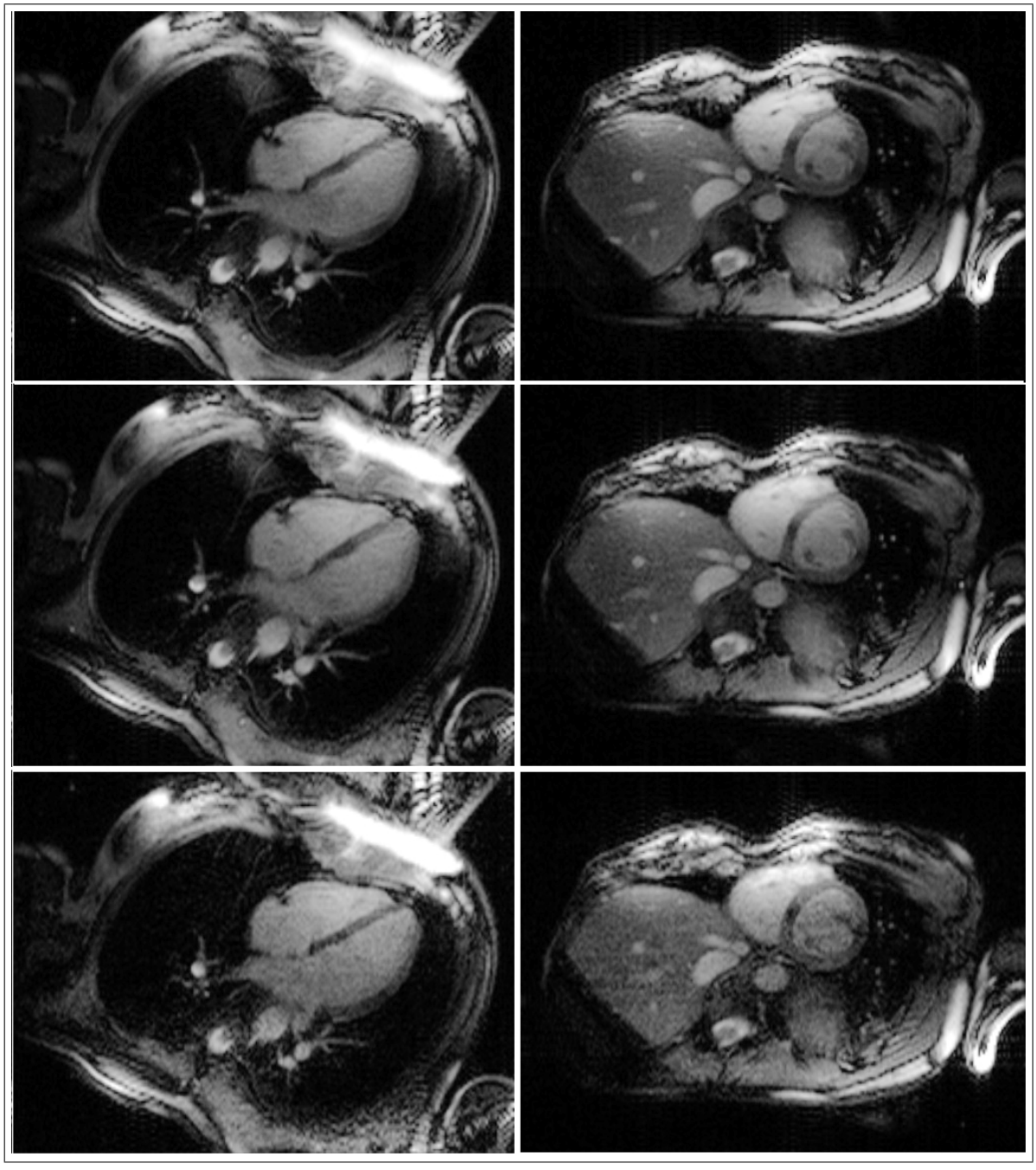

Figure 2.13 Real-time images reconstructed with HTGRAPPA in 0.008 seconds with acquisition of two interleaved slices (18 acquisition coils, acquisition matrix of 192×108). From left to right: long axis, and short axis heart images. From top to bottom, R=2, R=3 and R=4. Darkening in the myocardium from saturation can be seen where the images intersect, since they are acquired serially.

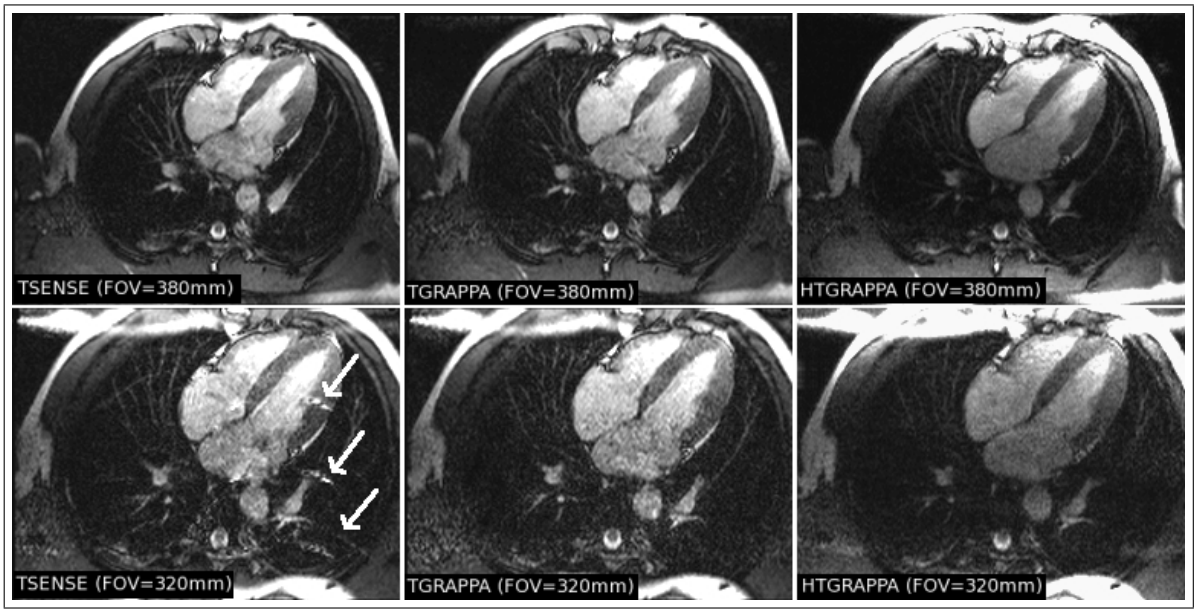

Figure 2.14 HTGRAPPA, TSENSE and TGRAPPA compared. From left to right: TSENSE, TGRAPPA, HTGRAPPA reconstructed images from data acquired with 32 acquisition coils with acquisition matrix of 192×108. Upper images represent large field of view (380), and lower images represent small field of view (320). Note that artifacts from pre-folding are seen with TSENSE, but not with TGRAPPA and HTGRAPPA.

2.3.3.4 Discussion. HTGRAPPA provides real-time low latency image domain reconstruction that is unaffected by increased acceleration rates and increased block sizes. Therefore, superior artifact suppression with bigger GRAPPA kernels can be achieved without sacrificing imaging speed. On our current 16 cores AMD workstation, 125 fps reconstruction performance is achieved while reconstructing 18 coil cardiac data set with acquisition matrix of 192×108. Evidently, HTGRAPPA can easily handle real-time reconstructions of rate 4 data set (acquired with 16 fps). However, additional calculations must be performed during the weight calculation (e.g. image domain transformation of the weights, B1-map estimation, generation of noise weighted unmixing coefficients). Extra time required for these calculations depends on the number of receiver coils and on the acquisition matrix size. On our 18 coil data set, we experienced additional 0.42 seconds during the calculation of composite unmixing coefficients. 0.76 extra seconds are required to calculate these coefficients on our 32 coil data set. We consider this to be reasonable given the benefits of HTGRAPPA reconstruction scheme: more than 120 fps reconstruction performance is achieved when reconstructing 18 coil rate 4 data set, and more than 75 fps reconstruction performance was reached when

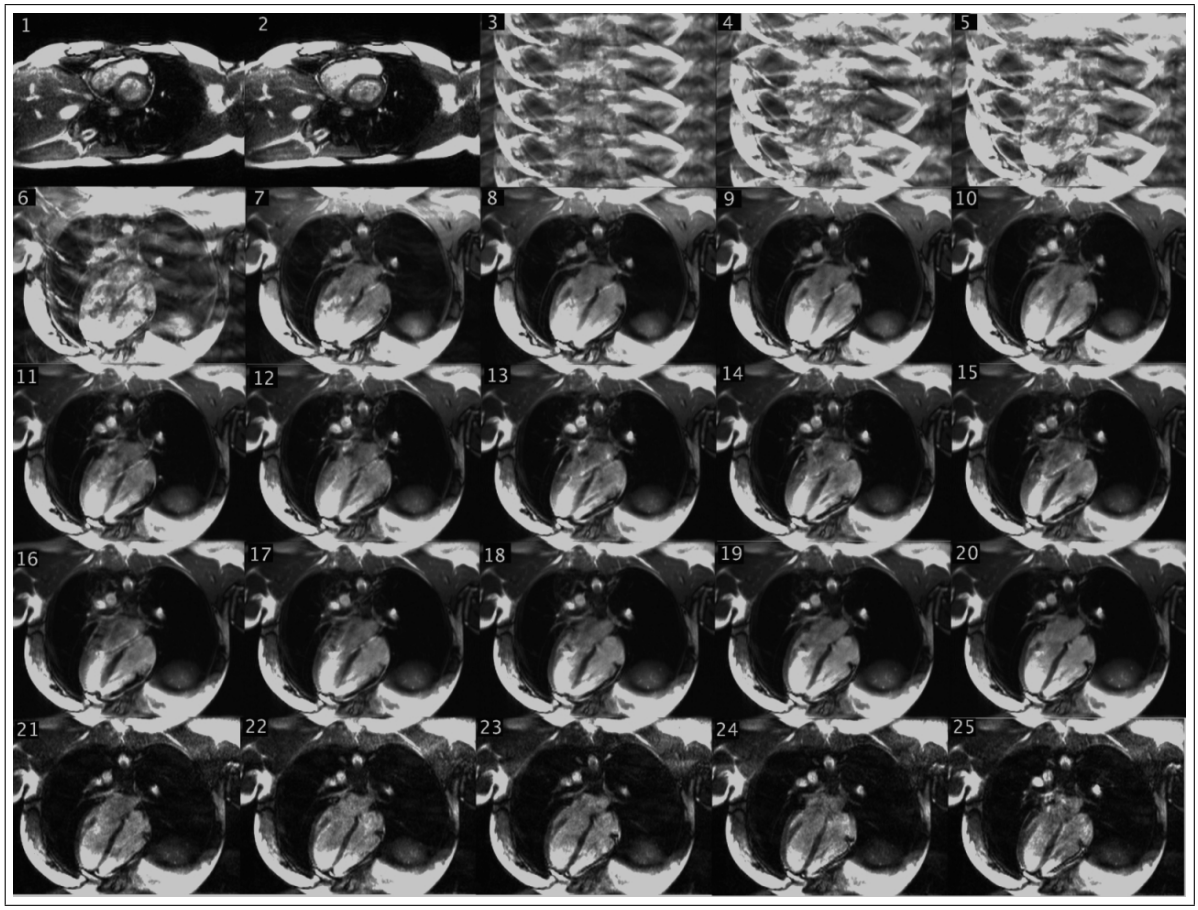

Figure 2.15 Series of images (reconstructed from data acquired with R=4, N_c=30, acquisition matrix of 192×108) showing the response of our reconstruction system to dynamic scan plane change. At frame 3, the scan plan was changed perpendicularly. Our system switched to view-sharing reconstruction while the new weight-sets are being calculated. At frame 21, reconstruction switched back to HTGRAPPA when the weight-sets were ready to use.

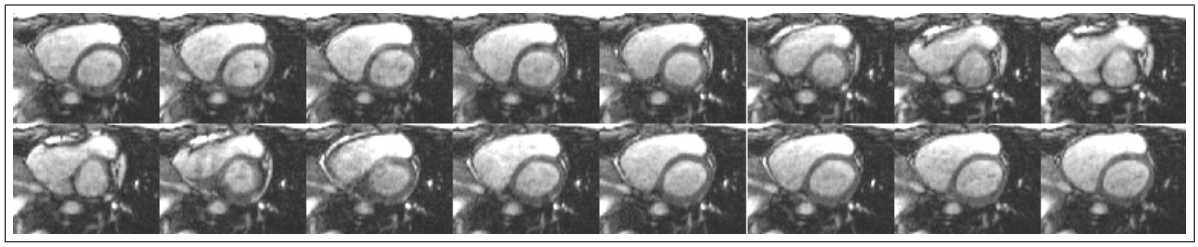

Figure 2.16 Cardiac cine image series reconstructed from rate 4 data set in real-time using HTGRAPPA.

reconstructing rate 4, 30 coil data set. Furthermore, one set of fixed composite unmixing coefficients can accurately reconstruct multiple acquisition frames as shown in Figure 2.16.

HTGRAPPA weight calculation performance values indicated in Table 2.6 are obtained with all threads (reconstruction + weight calculation) are sharing system central processing units (CPU). Disabling reconstruction threads increase weight calculation performance by 10%. On the other hand, disabling weight calculation threads increase reconstruction performance by 40%. These results justify that the weight computation is the computationally most demanding part of the algorithm. As the reconstruction is much faster than the data acquisition, reconstruction threads are idle most of the time. Therefore much of the system resources are used by the weight calculation. HTGRAPPA weight calculation can be implemented using a GPU to leave all CPU resources to the reconstruction threads. That way, HTGRAPPA can even run on real-time on a standard dual-core desktop PC.

Reconstruction performances of TGRAPPA and HTGRAPPA on our 18 coil cardiac data set are given in Table 2.7. Even with acceleration rate 2 and with relatively small block size of $[2 \times 3]$, TGRAPPA is four times slower than HTGRAPPA. By the increased acceleration rates and block sizes, conventional TGRAPPA becomes too slow to reconstruct images in real-time. Nearly 180 times faster reconstruction performances are achieved using HTGRAPPA when compared to conventional, highly parallel TGRAPPA algorithm (see section 2.3.1) on rate 4 data set. This blazingly fast reconstruction performance is the direct result of the (1) acceleration rate and block size independent reconstruction scheme, (2) combining multiple set of weights per coil into one set of weights per coil, and (3) elimination of RSS combining from reconstruction process.

Figure 2.14 represents large (upper images) FOV (380 mm) and small (lower images) FOV (320 mm) images reconstructed with TSENSE, TGRAPPA and HTGRAPPA algorithms (from left to right). Note that TSENSE and TGRAPPA reconstructions are off-line, while HTGRAPPA reconstructed images on-line. Image quality

of HTGRAPPA is on par with TGRAPPA and TSENSE. Furthermore, HTGRAPPA images do not exhibit artifacts from pre-folding as is the case for TSENSE when smaller FOV are used. 30 coil rate 4 data set can be reconstructed in real-time without sacrificing image quality with HTGRAPPA algorithm.

Real-time imaging is a must for MRI guided interventions. During interventions, scan plane changes very often to keep track of invasive device positions, and to find the best view of the anatomical structures of interest. This implies frequent changes in the coil sensitivities that can be problematic for pMRI algorithms. pMRI algorithms require some time to adapt to these changes, and thus aliasing cannot be removed during the transition. HTGRAPPA overcomes this issue by automatically switching to view sharing method when new weights are needed. Weights for the new scan plane are calculated in the background. Reconstruction switches back to HTGRAPPA when the new weights are ready to deploy. That way, uninterrupted parallel imaging is achieved. Figure 2.15 represents the response of our implementation when scan plane is dynamically changed during real-time imaging. Scan plane is rotated 90 degrees at frame 3, and reconstruction switches to view sharing. New weights are calculated when frame 21 is acquired, and reconstruction automatically switches to HTGRAPPA in a totally transparent manner for the operator. Calculation of the new weights may be quickened by (1) decreasing the block size, (2) decreasing the number of ACS lines, or (3) by increasing number of threads associated to weight calculation, if there are enough system resources.

Non-triggered free breathing cardiac cine images across a single heart cycle reconstructed using rate 4 parallel imaging are shown in Figure 2.16. Partial phase Fourier acquisition is used with TR=3 ms, and 20 k-space lines are acquired per frame. Total acquisition time per frame is $3 \times 20 = 60$ ms (16.67 fps). The average heart-beat of the volunteer is approximately 60 beats per second. This result shows that a fixed set of unmixing coefficients may be used to reconstruct multiple acquisition frames without aliasing artifacts or temporal blurring.

Disproportional drops in the weight calculation are experienced when number of

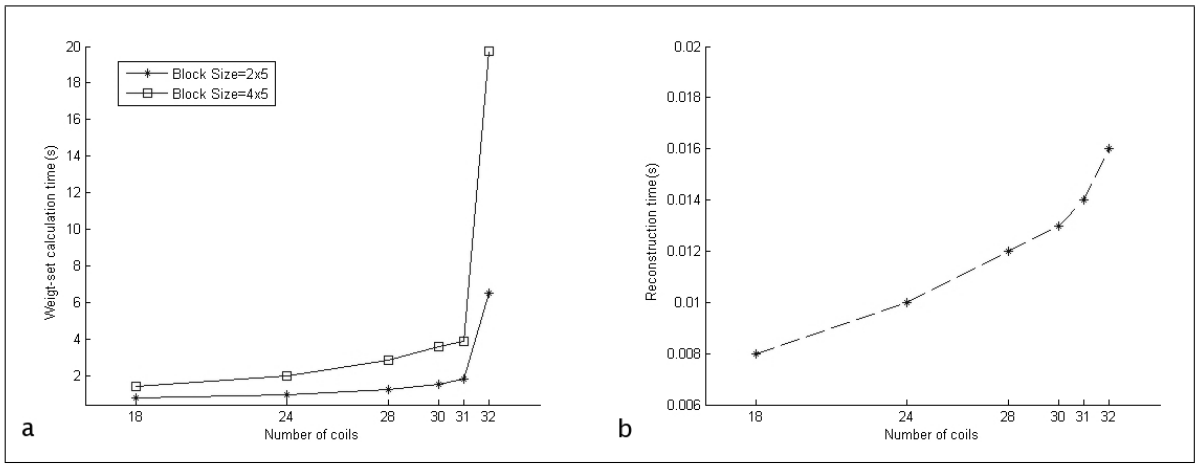

Figure 2.17 Performance plots of HTGRAPPA for weight-set calculation and reconstruction with different number of acquisition coils (acquisition matrix of 192×108, acceleration rate of 4). a. Weight-set calculation performances in seconds with block sizes of $[2 \times 5]$ and $[4 \times 5]$. b. Reconstruction performances in seconds.

acquisition coils exceeded 31. Rate 4 data set with matrix size of 192×108 are acquired with different number of receiver coils to investigate this performance drop. Block sizes of $[4 \times 5]$ and $[2 \times 5]$ and 48 ACS lines are used for the weight-set calculation. The source of the problem turned out to be increased "L2 cache misses" during the execution. Results represented in Figure 2.17 shows that our current computer hardware is not capable of handling more than 31 coil data in real-time when bigger blocks are used. Image quality is still acceptable with block size of $[2 \times 5]$. A new computer with advance architecture will improve the performance of HTGRAPPA.

A significant benefit of TGRAPPA is the tolerance to pre-folding within the prescribed FOV of the auto-calibration data. Furthermore, acceleration rate and block size independent reconstruction scheme provides better image quality with faster reconstruction rates. The flexible general purpose software implementation using parallel threads permits the scaling of the number of CPU's to readily trade off cost and speed.

Pre-folding tolerant, acceleration rate and block size independent reconstruction using HTGRAPPA makes it possible to perform low-latency real-time imaging without sacrificing image quality. Our implementation is flexible as it is implemented on a general purpose architecture. Additionally, dynamic structure of weight calculation (e.g.

changing block size, number of ACS lines, number of threads on the fly) provides highly adaptive real-time imaging. Our approach can also be fully or partially implemented using low cost graphics cards, as it is done in [73] for the SENSE algorithm.

2.4 Conclusion

GRAPPA algorithm can be beneficial over SENSE when (1) accurate coil sensitivities cannot be obtained in inhomogeneous regions, such as lungs and abdomen, and (2) smaller FOV are desired. Even though real-time implementation of (T)SENSE is previously presented by Guttman *et al.* [52], there was no real-time low latency implementation of (T)GRAPPA algorithm, due to the convolution-type reconstruction scheme, and RSS combining required to obtain the final image.

First, highly parallelized conventional TGRAPPA implementation is developed as a prototype. The effects of block size and number of ACS lines on the image quality are experimented. Additionally, this prototype allows us to pinpoint why TGRAPPA reconstruction is slow: convolution based, acceleration rate and block size dependent reconstruction scheme was slow to perform low-latency real-time imaging with high acceleration rates ($R > 2$) and high number of receiver coils. If one can transform GRAPPA weights to the image domain, convolution operations can be replaced by simple multiplication. Hence, faster reconstruction performances can be achieved [63–65]. To this end, highly parallel image domain TGRAPPA algorithm is implemented in *C++*, and deployed during several real-time experiments. Results show that image domain TGRAPPA is much faster, and acceleration rate and block size dependency of the reconstruction are removed. Constant reconstruction performances for a given number of receiver coils and acquisition matrix are achieved. However, the overall performance does not meet our expectations, which is real-time low latency imaging with higher acceleration rates ($R \geq 4$), and with higher number of acquisition coils ($N_c \geq 15$). The use of multiple set of weights as in original TGRAPPA, and RSS combination required to obtain final reconstructed images are penalizing the reconstruction speed. If one can combine multiple set of image domain weights per coil into one set of weights per coil,

and if one can eliminate RSS combining step from the reconstruction, blazingly fast reconstruction speeds can be achieved. To this end, a new, B1-weighted image domain TGRAPPA based hybrid domain algorithm HTGRAPPA is developed.

HTGRAPPA is a highly parallel, hybrid TGRAPPA implementation for low latency, real-time imaging algorithm where weight calculation and reconstruction processes were decoupled and individually parallelized. For improved reconstruction speeds, k-space domain convolution type operations are converted to pixel-wise multiplication in the image domain after proper transformation: (1) GRAPPA weights are transformed to image domain, (2) multiple image domain weights per coil are combined using B1-map estimates, (3) RSS combining are replaced by a phased array combiner as in SENSE. Additionally, noise weighting is applied to HTGRAPPA weights for improved SNR on the reconstructed images. HTGRAPPA provides a constant reconstruction speed for a given acceleration rate and block size. Our implementation can keep pace with acquisition rates up to $R = 8$ on our reconstruction workstation, which is not previously possible to achieve using conventional TGRAPPA reconstruction in our tests. Additional steps are added to the calculation of the weights. The weight sets are calculated in the background, asynchronous to image reconstruction. Furthermore, our implementation can respond to interactive changes in slice plane by first producing view-shared images, then resuming HTGRAPPA once new weights have been calculated. That way, low latency, uninterrupted interactive real-time MRI is permitted. Up to 120 fps reconstruction performance is achieved. HTGRAPPA is nearly 180 times faster than conventional, highly parallel TGRAPPA when reconstructing rate 4 data set. Additionally, block size and number of ACS lines can be dynamically selected regarding the imaging needs. It is possible to quicken weight calculation if more CPUs are available. HTGRAPPA does not exhibit pre-folding artifacts when small field of views are used, and it is totally auto-calibrated. HTGRAPPA is currently the fastest TGRAPPA based reconstruction algorithm.

3. REAL-TIME MRI RECONSTRUCTION FOR NON-CARTESIAN TRAJECTORIES

Non-Cartesian trajectories are proven to be advantageous for MR data acquisition for many clinical applications [49, 74, 75]. Despite all the benefits of using non-Cartesian sampling schemes, Cartesian trajectories have gained more widespread use in the clinic for their low computational requirements and thus their faster reconstruction performances. Reconstruction of non-Cartesian sampled data is non-trivial, as FFT cannot be directly applied. DFT can reconstruct such data, but this method is computationally too slow to be practical. If radial or constant angular velocity spiral trajectories are used, back-projection reconstruction algorithm can be also applied. The most common method is to resample the non-Cartesian data set onto a Cartesian grid and then apply FFT algorithm to reconstruct images. The process of mapping non-Cartesian data set onto a Cartesian grid is called "gridding".

Straightforward interpolation for gridding causes high level of artifact, and thus is not preferred [48]. The most common gridding algorithm is the "convolution gridding", also called as conventional gridding [19]. This method is relatively fast compared to other methods, and provides adequate image quality. It applies convolution to k-space data to perform gridding. Choice of convolution kernel is a trade-off between fast processing and image quality. Additionally, the effects of uneven sampling density caused by non-Cartesian sampling schemes must be removed by applying density compensation. Subsequent to gridding, Fast Fourier Transform (FFT) is applied to reconstruct images. Several other methods, Uniform Resampling (URS)/Block Uniform Resampling (BURS) [24, 25], Iterative Next-Neighbor Regridding (INNG) [26] and Deconvolution-Interpolation Gridding (DING) [21] have been proposed to perform gridding, all being computationally expensive. With these methods, image quality depends on several gridding parameters.

In 2007, a novel gridding algorithm, GRAPPA Operator Gridding (GROG)

[27] is proposed by Seiberlich, *et al.* This algorithm is based on GRAPPA Operator formalism [51]. To our knowledge, GROG is the first pMRI based gridding algorithm where coil sensitivity profiles are used to calculate gridding weights (or k-space shifting weights) in x and y directions. These weights are then applied to non-Cartesian samples for appropriate mapping to a Cartesian grid. GROG has a simple density compensation scheme, where pixel values are simply averaged. An additional Cartesian data set must be acquired as a calibration data set to calculate gridding weights, as in GRAPPA [36]. In 2008, a self-calibration version of GROG (SC-GROG) is presented by Seiberlich, *et al.* [28]. In SC-GROG, Cartesian calibration data set requirement is eliminated, and the non-Cartesian samples themselves are used as a calibration data set to calculate k-space domain shifting weights. There are no parameters, no additional data sets to deal with. A new set of distinct gridding weights can be calculated for each acquisition frame. SC-GROG is a parameter-free, parallel-imaging based, robust, self-calibrating gridding algorithm that can be applied to grid both undersampled and fully sampled data sets. Additionally, the quality of (SC)-GROG images is on par with conventional gridding [27, 28].

SC-GROG is a good candidate for real-time imaging. Its self-calibrating scheme benefits from pMRI, and at the same time it eliminates all the parameters of the conventional gridding algorithm. Convolution kernel selection is not needed anymore, as a new custom kernel per acquired frame can be generated. Additionally, kernel size is fixed to one with (SC)-GROG algorithm. Furthermore, density compensation is straightforward. Even though conventional gridding has some high performance implementations [76–78], there was no real-time oriented low-latency implementation for SC-GROG algorithm. Weights calculation, gridding and reconstruction processes are all sequential in the conventional (SC)-GROG algorithm.

We present the first real-time highly parallel implementation of the SC-GROG algorithm for multi-slice radial acquisitions, real-time GROG (RT-GROG). Unlike previous work on conventional gridding that involve Graphics Processing Unit (GPU) programming, a general purpose multi-core architecture is preferred for better scalability, and portability. SC-GROG algorithm is deeply analyzed, and modified to track

changes in the coil sensitivities for dynamic imaging, as it is done in [38,40,42]. For that purpose, SC-GROG's weight-set calculation and image reconstruction steps are decoupled into two separate processes, and then parallelized on algorithmic level. GROG algorithm is based on GRAPPA operator formalism. Thus, as in original GRAPPA algorithm, gridding of one coil data requires all-coil data. Shared memory computer architecture is preferred to prevent communication overhead, as in HTGRAPPA. In this scheme, weight calculation process updates the weights in the background, while reconstruction process uses latest set of gridding weights to grid non-Cartesian data and reconstruct images. Furthermore, per sample *2D* weights that are normally calculated during gridding are pre-calculated and stored in a lookup table (LUT) by the weight-set calculation process. Provided LUT access eliminated the per sample *2D* weight calculation from the reconstruction process, and increased gridding (and thus reconstruction) performance is obtained. Our test results indicate that with RT-GROG, image reconstruction is always faster than the data acquisition. It is important to note that our implementation can also be applied to constant-angular-velocity and reordered constant-linear-velocity spiral trajectories [28]. 34-fold performance increase is experienced over highly parallel SC-GROG, when our proposed RT-GROG scheme is used.

In this Chapter, first, basic gridding methods conventional gridding, BURS and DING will be briefly presented. Subsequent to this introduction, SC-GROG algorithm will be explained. Following that, our real-time highly parallelized SC-GROG implementation RT-GROG will be presented in details. Finally, our proposed real-time implementation method will be deeply discussed.

3.1 Common Gridding Methods

Several methods have been proposed to perform gridding. In this section, most well-known gridding algorithms will be briefly described. First, most popular conventional gridding method will be explained. Second, BURS algorithm will be presented. Following that, iterative algorithms INNG and DING will be described.

3.1.1 Convolution Gridding

Optimal gridding kernel is shown to be a *sinc* function by O'Sullivan in 1985 [18]. Following the convolution of the non-Cartesian k-space with a *sinc* function, data can then be sampled onto a Cartesian grid. However, *sinc* function has infinite extent, and thus gridding of each non-Cartesian k-space sample requires a multiplication by the *sinc* function, which is not practical at all. Convolution kernel selection for gridding is analyzed in detail in [19], and authors are concluded that Kaiser-Bessel window is the most practical convolution kernel. With such a kernel, gridding accuracy can be controlled by the free parameter β. The choice of β values is also discussed in [19].

Usage of a non-*sinc* function for the convolution decreases image quality. This is due to the fact that resampling of the convolved k-space data introduces aliasing. This aliasing can be further reduced by using higher oversampling ratios. Oversampling pushes aliased replicates apart from the image. After Fourier transform, excess field of view (FOV) can be discarded to remove the aliased part of the image [48].

Let $S(k_{xj}, k_{yj})$ be the acquired non-Cartesian k-space data, and $g_{2D}(k_x, k_y)$ be the *2D* gridding kernel at specific location (k_x, k_y). Gridded *2D* resampled data $S^{(c)}(m\Delta k_{xj}, n\Delta k_{yj})$ can be obtained by:

$$S^{(c)}(m\Delta k_{xj}, n\Delta k_{yj}) = \sum_j S(k_{xj}, k_{yj}) g_{2D}(m\Delta k_x - k_{xj}, m\Delta k_y - k_{yj}) \Delta k_j^{(s)} \qquad (3.1)$$

where $\Delta k_j^{(s)}$ represents the density compensation factor. Density compensation is required to correct uneven sampling density of non-Cartesian trajectories. It can be approximated by the differential area expression for the k-space trajectory. In the case of Cartesian sampling, density compensation is omitted, because $\Delta k_j^{(s)}$ is the same for each sample. In *2D* polar coordinates, differential area is represented by $kdkd\theta$. Therefore, for radial trajectories $\Delta k_j^{(s)} = k_j \Delta k_j \Delta\theta_j$. If Δk_j and $\Delta\theta_j$ are the same for each all samples, then $\Delta k_j^{(s)} = k_j$. The same density compensation function can also be used for *2D* spiral data acquisition. If a constant velocity spiral trajectory is used, $\Delta\theta_j$ is not constant, and must be properly estimated. A detailed information about density

compensation for spiral trajectories can be found in [29]. Final image is obtained by

$$I = \mathscr{F}^{-1}(S^{(c)})/G \tag{3.2}$$

where $\mathscr{F}^{-1}$ represents *2D* inverse Fourier transform, and $G = \mathscr{F}^{-1}(g)$. As a finite convolution function is used, any roll-off in the central lobe of g will be seen as an attenuation towards the side of the reconstructed image [19]. Dividing the image by G will correct this roll-off, and will provide final image.

Convolution gridding is the most popular gridding algorithm. However, image quality depends on several parameters: (i) convolution kernel choice, (ii) kernel parameters, (iii) density compensation function. This method uses a priori information. No information is extracted neither from the acquired data, nor from the acquisition coils.

3.1.2 BURS

BURS algorithm examines the gridding operation as an inverse problem. Let us assume that we have N samples $f(k_1), \ldots, f(k_N)$ on a uniform grid, that we arranged in a vector $x = \{f(k_1), \ldots, f(k_N)\}$. What we want to calculate is the nonuniform samples stacked in a vector $b = \{f(K_1), \ldots, f(K_M)\}$ using vector x. Let us define a matrix A, the matrix of interpolation (gridding) coefficients:

$$Ax = b. \tag{3.3}$$

Non-Cartesian acqusition provides us with vector b. Therefore, if we can find a matrix A obeying $A^{\dagger}A = I_N$ († stands for pseudo-inverse operator, and I_N is the identity matrix of size $(N \times N)$), vector x can be calculated. Note that, the pseudo-inverse $A^{\dagger}$ provides the optimal solution to Eq. 3.3 in the minimal-norm least-square sense. In practice, $A^{\dagger}$ is precalculated and stored in a LUT, and then is used during the reconstruction. In practice, the matrix A can have very large dimensions that makes the inversion difficult. In addition, the calculation of the vector x will involve huge amount of operations [24].

Typically, the coefficients from a given row i of matrix $A^\dagger$ will have a sinc-like shape centered around k_i. Therefore, if each row of the matrix $A^\dagger$ has mostly zeros and only the coefficients in the close vicinity of k_i (in a radius δ_k) are chosen to be non-zero, the computational burden of calculation x can be decreased. This is the basic idea of BURS algorithm. BURS can be summarized as follows: [24]

1. write down the matrix A given by Eq. 3.3.
2. choose δ_k and Δ_k, which is the radius to choose the Cartesian grid points to consider ($\delta_k < \Delta_k$).
3. create a zero-filled matrix $A_2^\dagger$ of size $N \times M$.
4. for i=1:N (for each uniform grid point k_i)
   (a) select the M_i' measurement samples $f(K_m)$ such that $|K_m - k_i| < \delta_k$.
   (b) select the N_i' uniform grid points $f(k_n)$ such that $|k_n - k_i| < \Delta_k$.
   (c) set up matrix A' (appropriate submatrix of A) using selected points.
   (d) calculate $(A')^\dagger$
   (e) isolate the row of $(A')^\dagger$ corresponding to the location k_i.
   (f) plug-in this row into the appropriate locations in the ith row of $A^\dagger$
5. the uniform samples are given by $x = A^\dagger b$.
6. image is reconstructed using Fourier transform.

Note that the estimation of $A^\dagger$ is realized off-line. Image reconstruction in BURS involves step 5 and step 6. As $A^\dagger$ is a sparse matrix, inverse operations are efficient. BURS does not require any pre- or post-compensation operations, opposite to conventional gridding. In many situations, the number of points M_i' within the radius δ_k can have large variation in k-space. In places where k-space is oversampled (i.e. center of the k-space), M_i' can be too large. This implies a large matrix inversion, and increased number of multiplication during the resampling step. In order to overcome this issue,

number of measurements that will be used in the computation of each point k_i can be limited to a certain maximum value, M'_{max}.

BURS algorithm is as effective as conventional gridding. In terms of computational complexity, BURS can be more efficient as it does not require to interpolate onto a larger Cartesian grid, and no pre- or post-compensation steps are involved. However, this efficiency depends on multiple parameters: (i) the choice of radius δ_k and Δ_k, and (ii) the choice of M'_{max}. Additionally, BURS method is sensitive to noise. Regularization techniques can help reduce the effects of noise, however k-space data signal-to-noise ratio (SNR) is not known in advance. Therefore, it is often quite difficult to apply accurate regularization [26].

3.1.3 INNG

INNG is next-neighbor regridding based iterative gridding algorithm. Basically, next-neighbor regridding algorithm involves the following steps [26]: (i) density compensation, (ii) distribution of the non-Cartesian data into a larger rescaled matrix, (iii) *2D* inverse Fourier transform of this large matrix, and (iv) cropping the original-sized image matrix from the center. Rescaling factor s is defined as 2^m, where m is a positive integer. Original k-space coordinate is multiplied by s and rounded to determine target Cartesian grid location. If more than one non-Cartesian sample are mapped to the same Cartesian grid location, the mean value is stored. With this method, image quality is highly dependent on the density compensation function, and rescaling parameter s [26]. INNG is an iterative algorithm that extends next-neighbor regridding algorithm. INNG does not require to perform density compensation, but necessitates to perform number of multiple FFTs of larger scaled matrices.

Let assume a Cartesian grid of size $N \times N$. Supposing a $N \times N$ Cartesian grid size, and $sN \times sN$ large rescaled matrix size, image reconstruction process of INNG is as follows:

1. distribute k-space data to the large rescaled matrix
2. obtain initial image by a *2D* inverse Fourier transform of the rescaled matrix
3. while [(updated image) - (image from previous iteration)] $< \epsilon$
   (a) zero-fill the rescaled matrix, except at the central $N \times N$ portion
   (b) apply *2D* Fourier transform to the resultant matrix and obtain updated data matrix
   (c) restore the data of the updated data matrix at the original locations within the rescaled matrix
   (d) apply *2D* inverse Fourier transform to obtain the updated image
4. obtain final image

INNG algorithm imposes two constraints on the data at each iteration: finite-support constraint (corresponds to process 3a), and data-consistency constraint (corresponds to process 3c).

Two FFTs are required per iteration with INNG algorithm. This can be problematic in terms of computation, especially if a large rescaling factor is used. The authors described here a facilitated INNG algorithm where computational load is reduced. In this method, consecutively increasing scaling factors are used during the reconstruction of the intermediate images. A larger scaling factor is used to reconstruct final image (e.g. 8) to reduce the gridding errors. This method is shown to be computationally more efficient, however implies higher memory space. To further decrease the computational load, they developed a block INNG (BINNG) [26] algorithm where each acquired k-space region is partitioned into blocks. Subsequent to partitioning, INNG algorithm is applied to each block.

There are several drawbacks for (B)INNG algorithm. First, an appropriate scaling factor must be chosen. Second, and more importantly, setting the stopping criteria for this algorithm is not an easy task, which is the case for many iterative algorithms.

3.1.4 DING

DING is an iterative algorithm that utilizes conjugate gradient (CG) technique to perform gridding. It involves solving a sparse system of linear equations, which is equivalent to deconvolution operation [21].

Let $S(k)$ designate continuous Fourier transform of the object to be imaged, and $S_s(k)$ be the discrete sampled k-space data. If the nyquist criterion is satisfied while sampling the k-space, sampling theorem states that $S(k)$ can be exactly reconstructed if $S_s(k)$ is convolved by a *sinc* function. For improved computational efficiency, the authors replaced the infinite *sinc* function with a small window $C(k)$ of finite support. Let $D(k)$ be the deconvolution of $S(k)$ with $C(k)$. Thus, $S(k)$ is defined as

$$S(k) = D(k) * C(k) = \int D(k')C(|k - k'|)dk' \tag{3.4}$$

where $*$ is the convolution operation. If Eq. 3.4 is calculated on a rectilinear grid,

$$D_r(k) = (S(k) * C(k)^{-1})\Lambda(k) \tag{3.5}$$

with $\Lambda(k) = \Sigma_{n=1}^{N}\delta(k - k_n)$ being the grid sampling function, and with $D_r(k) = D(k)\Lambda(k)$ (uniformly sampled $D(k)$). Eq. 3.5 is equivalent to the following expression in the image domain:

$$d_r(r) = \frac{s(r)}{c(r)} * \Lambda(r) \tag{3.6}$$

where the lowercase variables represent the inverse Fourier transformed variables. $d_r(r)$ represents the reconstructed image, and $s_r(r)$ represent desired image. These two images can be related by the modulation function $1/c(r)$ and the aliasing untrusted by the grid sampling function. If the central part of the image is multiplied by $c(r)$ and the result is truncated to the desired field of view (FOV), the intensity modulation effect can be eliminated. Therefore, final image estimate $s_{est}(r)$ is obtained by

$$s_{est}(r) = \left[\frac{s(r)}{c(r)} * \Lambda(r)\right] c(r)\Pi_{FOV}(r) \tag{3.7}$$

where $\Pi_{FOV}(r)$ represents the rectangular function that truncates the object to the desired FOV.

A discretized version of Eq. 3.4 will be solved, with $S(k)$ measured only at L points. This will provide the following sparse system:

$$S(k_j)_{j=1:L} \approx \sum_n D(k_n)C(|k_j - k_n|). \tag{3.8}$$

In matrix form, Eq. 3.8 can be expresses as

$$s_s = Cd_r \tag{3.9}$$

where C is the $L \times N$ interpolation matrix, d_r is the vector of the stacked uniform samples of D. Therefore, DING algorithm involves the following steps [21]:

1. solve Eq. 3.9 for d_r with CG technique (iterative)
2. perform an inverse Fourier transform to the image domain
3. multiply the result by $c(r)$ to correct the effects of deconvolution

DING uses a Keiser-Bessel kernel as an interpolation window. Window parameter β must be properly calculated for efficient gridding. Optimal β values for different window sizes are listed in [21]. DING does not need any density compensation function, as it is the part of the deconvolution process. Additionally, it does not require grid subsampling, which is needed for conventional gridding. In terms of computation, DING results with 8 times more computational load than conventional gridding. The choice of β value, stopping condition for the iteration and the increased computational load decreases the practicability of DING during real-time imaging.

3.2 Towards Real-time Self-Calibrating GROG

Self-calibrating, parameter free gridding algorithm had the potential to suit well to the real-time MRI applications (e.g. real-time MRI guided interventions). However, no real-time parallelized implementation was available. In the following sections, first, SC-GROG algorithm will be explained in detail. Second, our real-time SC-GROG implementation RT-GROG will be described. Finally, the experimental results of RT-GROG will be deeply discussed.

3.2.1 SC-GROG

SC-GROG algorithm uses pMRI principles to calculate the kernels required to map non-Cartesian samples to a cartesian grid. These kernels, or SC-GROG gridding weights, are calculated from the data itself, thus they are highly optimized. SC-GROG weights are also considered as k-space shifting operators. First, a local shifting operator is calculated for each projection. This local shifting operator can be used to shift each sample to the next sample's position. Then, general shifting operators for whole k-space in x and y directions are deduced from the local shifting operators. These general shifting operators are then used to grid non-Cartesian samples.

SC-GROG algorithm briefly described above can be technically described as follows. SC-GROG uses $N_c \times N_c$ weight-set to map a non-Cartesian sample to its closest neighbor on a Cartesian grid, where N_c represents number of the receiver coils used during the acquisition. For simplicity, let G_1 designate unitary weight-set (for shifts of Δk) in one dimension. The weight-set for smaller shifts is calculated by

$$G_m = (G_1)^m, \quad -1 < m < 1. \tag{3.10}$$

Equation 3.10 can be generalized for gridding of a *2D* k-space sample $s(k_x, k_y)$ acquired with all receiver coils, using the properties of the GRAPPA operator described

in [51]:

$$s(k_x + \delta x, k_y + \delta y) = (G_x)^{\delta x}(G_y)^{\delta y} s(k_x, k_y) \tag{3.11}$$

where δx and δy is the distance of this non-Cartesian sample to its nearest Cartesian grid location in x and y direction, G_x and G_y are the unitary weight-sets in x and y directions, and, finally, $s(k_x+\delta x, k_y+\delta y)$ represents the gridded signal at the Cartesian location $(k_x+\delta x, k_y+\delta y)$. This operation is schematized in Figure 3.1. For the sample S_1, the distances to the nearest Cartesian grid point are calculated as m_1 in x direction and n_1 in y direction. SC-GROG weights $G_x^{m_1}$ and $G_y^{n_1}$ are then calculated and applied to the sample S_1 for each coil to realize gridding operation. The same steps are also used for the non-Cartesian sample S_2. If multiple non-Cartesian samples are mapped to the same Cartesian location, the results are simply averaged.

Calculation of G_x and G_y are not straight-forward. First, an angular weight-set G_{θ_i} per projection ($i = 1, \ldots, N_p$ with N_p being number of projections) must be calculated. G_θ represents unitary shifting operator of size $N_c \times N_c$ along the given radial ray: $s(\theta, r+1) = G_\theta s(\theta, r)$. Therefore this weight-set can be calculated as follows:

$$G_\theta = pinv(s(\theta, r))s(\theta, r+1) \tag{3.12}$$

where $pinv$ represents the pseudo-inverse operator, $s(\theta, r)$ represents multichannel data of projection θ at position r. Each set of G_θ can be decoupled into two distinct shifting weight-sets in x and y direction, $G_\theta = G_x^m G_y^n$ where m and n represent shifts in x and y directions between each consecutive sample along given projection:

$$\begin{aligned} G_{\theta_1} &= G_x^{m_1} G_y^{n_1} \\ G_{\theta_2} &= G_x^{m_2} G_y^{n_2} \\ &\ldots \\ G_{\theta_{N_p}} &= G_x^{m_{N_p}} G_y^{n_{N_p}}. \end{aligned} \tag{3.13}$$

Figure 3.2 [28] represents the relation between local angular weights G_θ and global shifting weights G_x and G_y. Eq. 3.13 can be linearized by taking the matrix logarithm

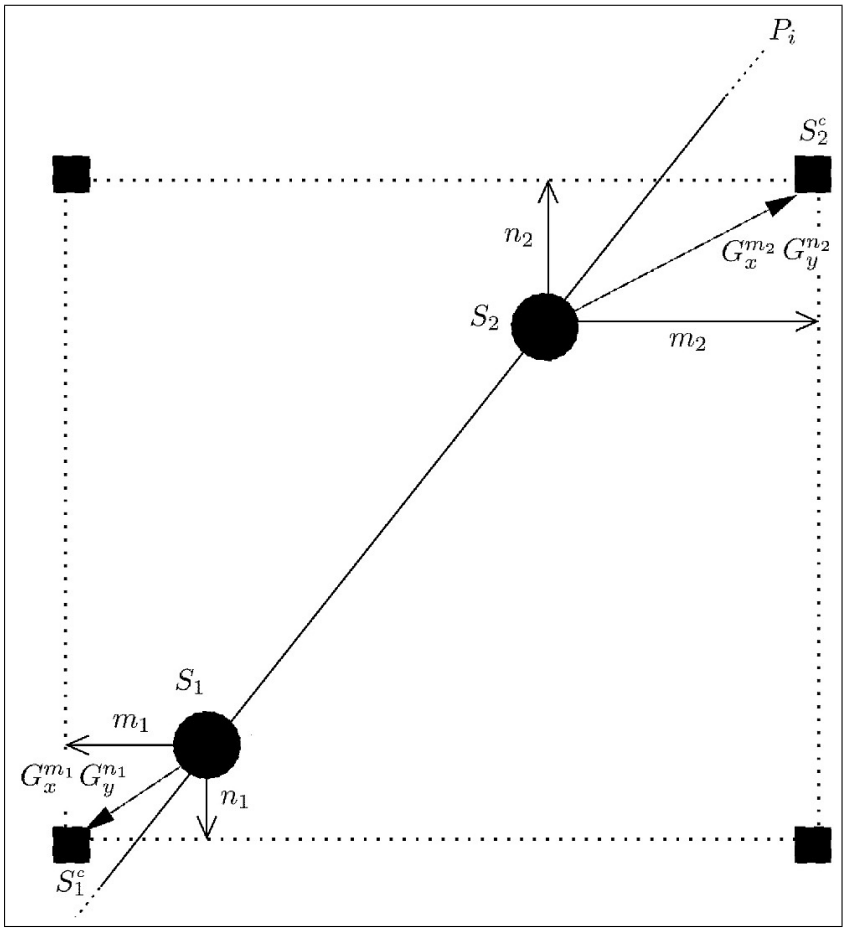

Figure 3.1 GROG gridding for two samples S_1 and S_2 of a projection P_i is represented. Gridding weights G_x, G_y are applied to the samples S_1 and S_2 according to their distances to their closest Cartesian grid neighbors S_1^c and S_2^c, and their contributions to these Cartesian grid points are calculated by $S_1^c = G_x^{m_1} G_y^{n_1} S_1$, $S_2^c = G_x^{m_2} G_y^{n_2} S_2$.

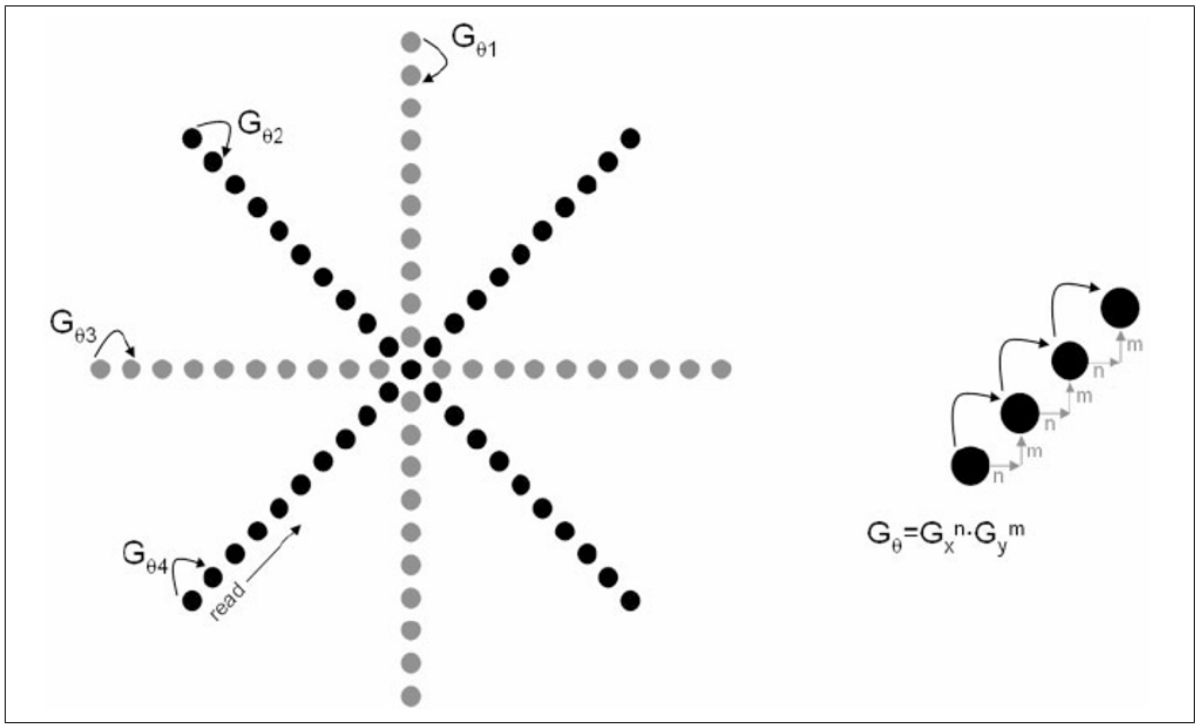

Figure 3.2 Relation between local angular weights G_θ per projection and global shifting weights G_x and G_y [28].

of each side:

$$\begin{aligned} ln(G_{\theta_1}) &= m_1 ln(G_x) + n_1 ln(G_y) \\ ln(G_{\theta_2}) &= m_2 ln(G_x) + n_2 ln(G_y) \\ &\cdots \\ ln(G_{\theta_{N_p}}) &= m_{N_p} ln(G_x) + n_{N_p} ln(G_y). \end{aligned} \tag{3.14}$$

After linearization, Eq. 3.14 can be reorganized as a matrix equation and solved for $ln(G_x)$ and $ln(G_y)$. Subsequently, G_x (and G_y) is calculated by taking the matrix exponentials, $G_x = exp(ln(G_x))$ (and $G_y = exp(ln(G_x))$ respectively). Once the weight-sets are calculated, gridding can be performed using Eq. 3.11.

Figure 3.3 [28] represents *in vivo* images reconstructed from a radial data set using both SC-GROG and conventional gridding. Figure 3.4 [28] compares *in vivo*

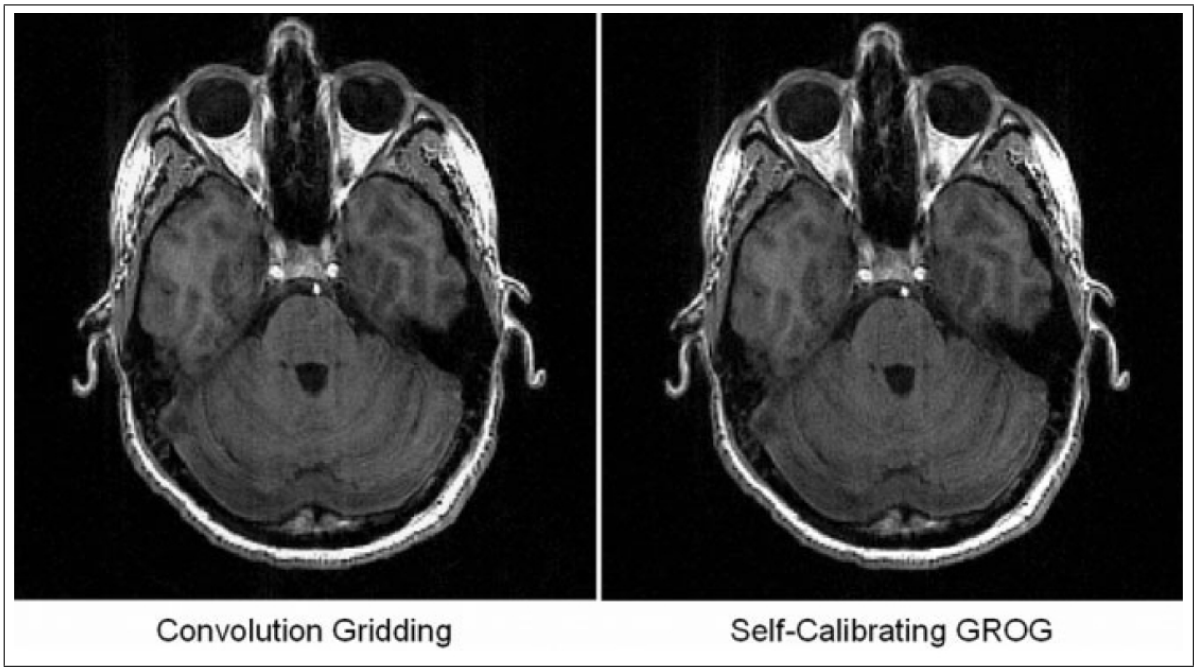

Figure 3.3 Comparison of *in vivo* images reconstructed from a radial data set using SC-GROG and conventional gridding [28].

images reconstructed from a spiral data set using SC-GROG and conventional gridding. Note that, the images are visually identical.

Gridding using Eq. 3.11 requires to calculate $(G_x)^{\delta x}(G_y)^{\delta y}$ for each pixel. Therefore, with conventional SC-GROG, floating point power of two matrices must be calculated and then multiplied to obtained one pixel gridding weights. This step drastically increases computational requirements of the SC-GROG algorithm. If one can avoid per pixel weight calculation during the reconstruction, greatly improved reconstruction will be obtained.

3.2.2 RT-GROG

The bottleneck in SC-GROG algorithm is the per pixel weight calculation during reconstruction. Our real-time implementation decouples weight calculation and reconstruction processes, as it is done in [42]. Additionally, per pixel gridding weights

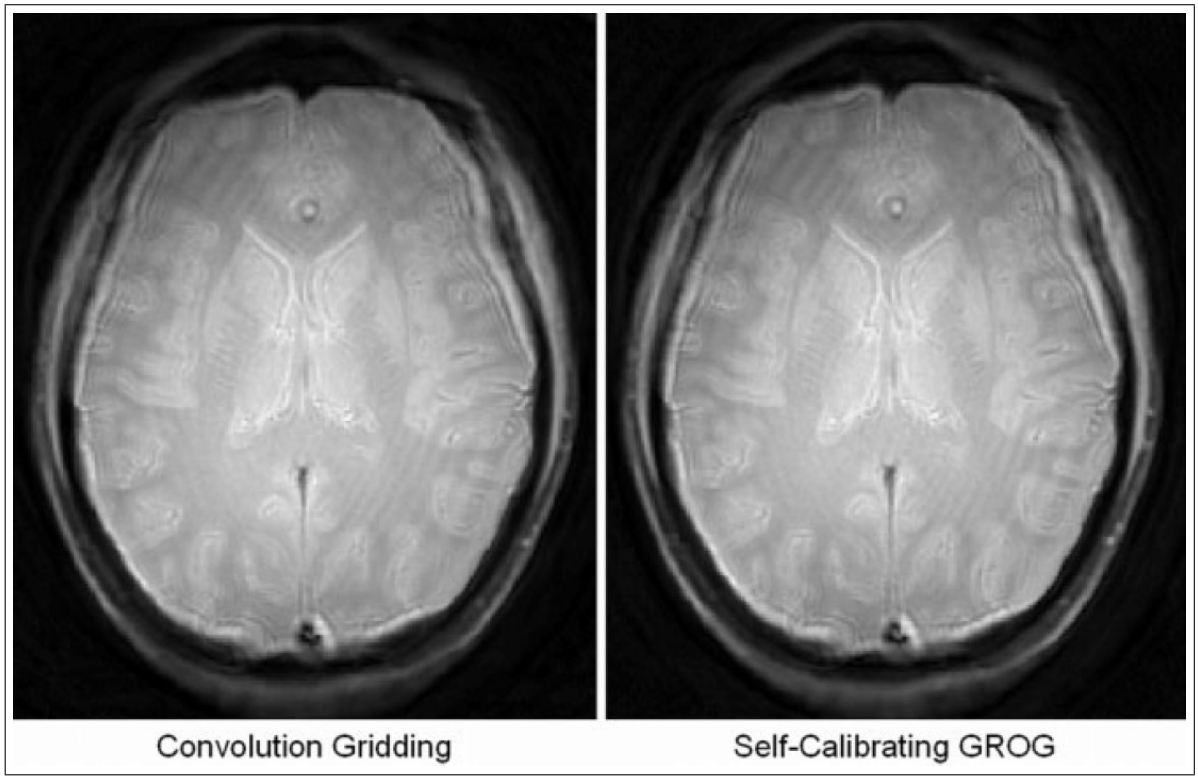

Figure 3.4 Comparison of *in vivo* images reconstructed from a spiral data set using SC-GROG and conventional gridding [28].

are pre-evaluated for discretized distances in x and y directions in the weight calculation process. Subsequent to the evaluation, calculated weights are stored in a LUT for faster access by the reconstruction process. That way, highly increased reconstruction performances are achieved.

In the following section, our implementation details, as well as the experimental results will be addressed in details.

3.2.2.1 Implementation. The SC-GROG algorithm is implemented in $C++$ and parallelized using Pthreads (http://pasc.org) and OpenMP (http://openmp.org). The weight set calculation and reconstruction processes are decoupled to execute asynchronously in parallel. The weight calculation thread continuously updates the calibration data set and recalculates the weights in the background to track changes in the coil sensitivities, while the reconstruction threads grid non-Cartesian samples using the latest weights and perform real-time image reconstruction. N_c reconstruction threads are initially created by the Pthreads library, and the same threads with the same allocated memory variables are used during the whole execution in order to prevent continuous context switching. Unnecessary context switching during the execution must be avoided to provide a computational acceleration. The weight set calculation process is parallelized using OpenMP library to be able to dynamically change the number of threads devoted to the weight set calculation. At initialization, and after a scan-plane change, new weights must be calculated as quickly as possible. Therefore, all system CPUs are devoted to multi-threaded weight-set calculation process in such cases for faster calculation. The number of threads used for the weight set calculation may be decreased dynamically for devoting more resources to image reconstruction. OpenMP based parallelization for weight calculation improves the scalability and adaptability of our reconstruction program to different environments and imaging needs.

Block diagram of the RT-GROG algorithm is presented in Figure 3.5. Every acquired frame is feeded to the integration window to obtain the calibration data. First local shifting weights G_θ are calculated, and then local shifting weights G_x and G_y.

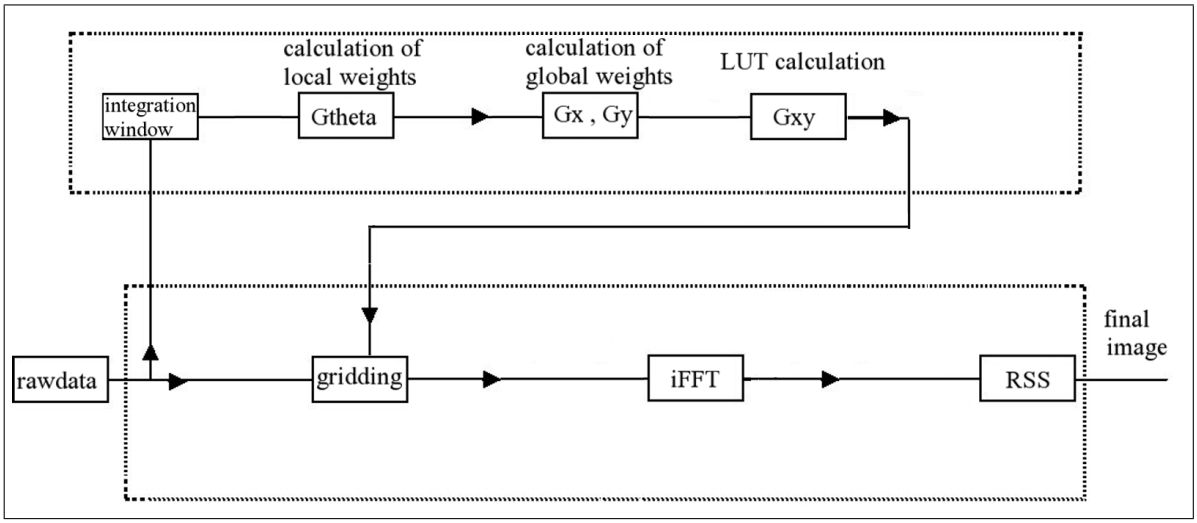

Figure 3.5 Block diagram of the RT-GROG reconstruction algorithm is given. Dotted rectangles represent parallel regions. The upper dotted rectangle represents the weight-set calculation process, and the lower dotted rectangle represents the reconstruction process. These processes are computed asynchronously.

Subsequently, LUT is computed for every possible combination. Reconstruction process (lower block in Figure 3.5) uses most recent LUT to grid and to reconstruct images. ATLAS and ACML libraries are deployed during weights calculation process. Local and global weights, and LUT calculation are based on OpenMP fork-join parallelism. Each reconstruction thread applies sequential FFT. If more ressources are available, parallel FFT can also be deployed.

The code is computed in parallel using 8 dual-core AMD Opteron 8220 processor (2.8 GHz) on Linux-2.6.16.46-0.12-smp. Pseudo-inverse operations and calculation of the eigenvectors is realized using the AMD Core Math Library (ACML) http://developer.amd.com/acml). For the matrix inversions, the ATLAS library is used (http://math-atlas.sourceforge.net). The FFTW library (http://fftw.org) was used for Fast Fourier Transformations. The code is compiled on GCC (http://gcc.gnu.org) version 4.2.2. Matrix power operations in Eq. 3.13, matrix logarithm operations in Eq. 3.14, and matrix exponential operations to solve for the base weights G_x and G_y are implemented in $C++$ using OpenMP assuming that the $1D$ SC-GROG weights G_x, G_y and their logarithms are diagonalizable matrices. Let M be a diagonalizable square matrix, and V matrix of eigenvectors of M. M can be projected into another space

where its projection M' will be a diagonal matrix:

$$M' = V^{-1}MV. \tag{3.15}$$

Therefore,

$$\begin{aligned} ln(M) &= Vln(M')V^{-1} \\ exp(M) &= Vexp(M')V^{-1} \\ M^n &= V(M')^nV^{-1} \end{aligned} \tag{3.16}$$

All of the calculations using M' in Eq. 3.16 can thus be realized by replacing every number on the diagonal of M' by its logarithm, its exponential, or its power, respectively. All projections are used in the calculation of the $1D$ weights G_x and G_y to achieve the best accuracy. In our implementation, we assume a step size of 0.1 (or 1/10 of unit in k-space) for distance measurements between a non-Cartesian sample and its closest Cartesian grid neighbor (e.g. the distance of 0.123 is rounded to 0.1, while 0.167 is rounded to 0.2). By this assumption, all possible combinations of per sample *2D* weights $(G_x)^m(G_y)^n$ for $m, n = \{-0.5, -0.4, \ldots, 0, 0.1, \ldots, 0.5\}$ are pre-calculated and stored in a LUT $GxGy$ by the weight set calculation threads following the calculation of the 1D gridding weights. That way, Eq. 3.11 is simplified to

$$s(k_x + \delta x, k_y + \delta y) = GxGy(\delta x, \delta y)s(k_x, k_y), \tag{3.17}$$

which eliminates per sample $(G_x)^{\delta x}(G_y)^{\delta y}$ calculation from the reconstruction process. Reconstruction threads only calculate the distance $(\delta x, \delta y)$ of each sample to its closest Cartesian grid neighbor, and retrieve the appropriate precalculated *2D* weight-set from LUT. As a result, greatly improved reconstruction performance is achieved.

Our implementation is employed during both *in vitro* and *in vivo* experiments. First, an existing brain data set is reconstructed to test the implementation. For further validation, several sets of phantom data are acquired using a short, wide bore Siemens Magnetom Espree 1.5 T (Siemens Medical Solutions, Erlangen, Germany). *2D* radial TrueFisp sequence with flip angle=45 degrees, TR=3.76 ms, FOV=340 mm is used. A

total of $6-15$ receiver coils are used during the acquisition of $50-400$ projections (256 samples per projection). Images are reconstructed in real-time. Experiment results are analyzed in terms of performance and image quality.

Following *in vitro* validation, several real-time multi-slice cardiac images of a healthy subject are acquired using a Siemens Magnetom Avanto 1.5 T (Siemens Medical Solutions, Erlangen, Germany) using the same sequence, but this time with fewer projections (64 and 96) and fewer samples (128 and 192) using 32 receiver coils (Invivo Corporation). The human subject protocol (NCT00720460) is approved by the NHLBI Institutional Review Board. All subjects are consented to participate in writing. The TR is 2.36 ms for 128 sample acquisitions, and 2.86 ms for 192 sample acquisitions. Short and long axis cardiac images are reconstructed in real-time.

3.2.2.2 Results. Brain image from high resolution radial data set (400 projections; 256 samples per each) is reconstructed to test the implementation, and given in Figure 3.6.

Phantom images are reconstructed from the acquired data with $6-15$ coils for $50-400$ projections in order to test our algorithm. Figure 3.7 represents images reconstructed from data acquired with 15 coil with $N_p = 256$, 128, 64, from left to right. A higher number of projections provided better images with fewer artifacts.

Short axis and long axis heart data are acquired with 32 coils. A total of 96 projections with 192 samples and 64 projections with 128 and 192 samples are acquired. Figure 3.8 represents real-time reconstructed cine images of a beating heart from a 32 coil data set on short axis (64 projections, 128 samples). These images indicate that a fixed set of SC-GROG weights may be used to grid multiple consecutive radial acquisitions. In Figure 3.9, a long axis heart image reconstructed in real-time from a 32 coil data set with 96 projections (128 samples) is represented.

The performances of the image reconstruction and weight set calculation (in

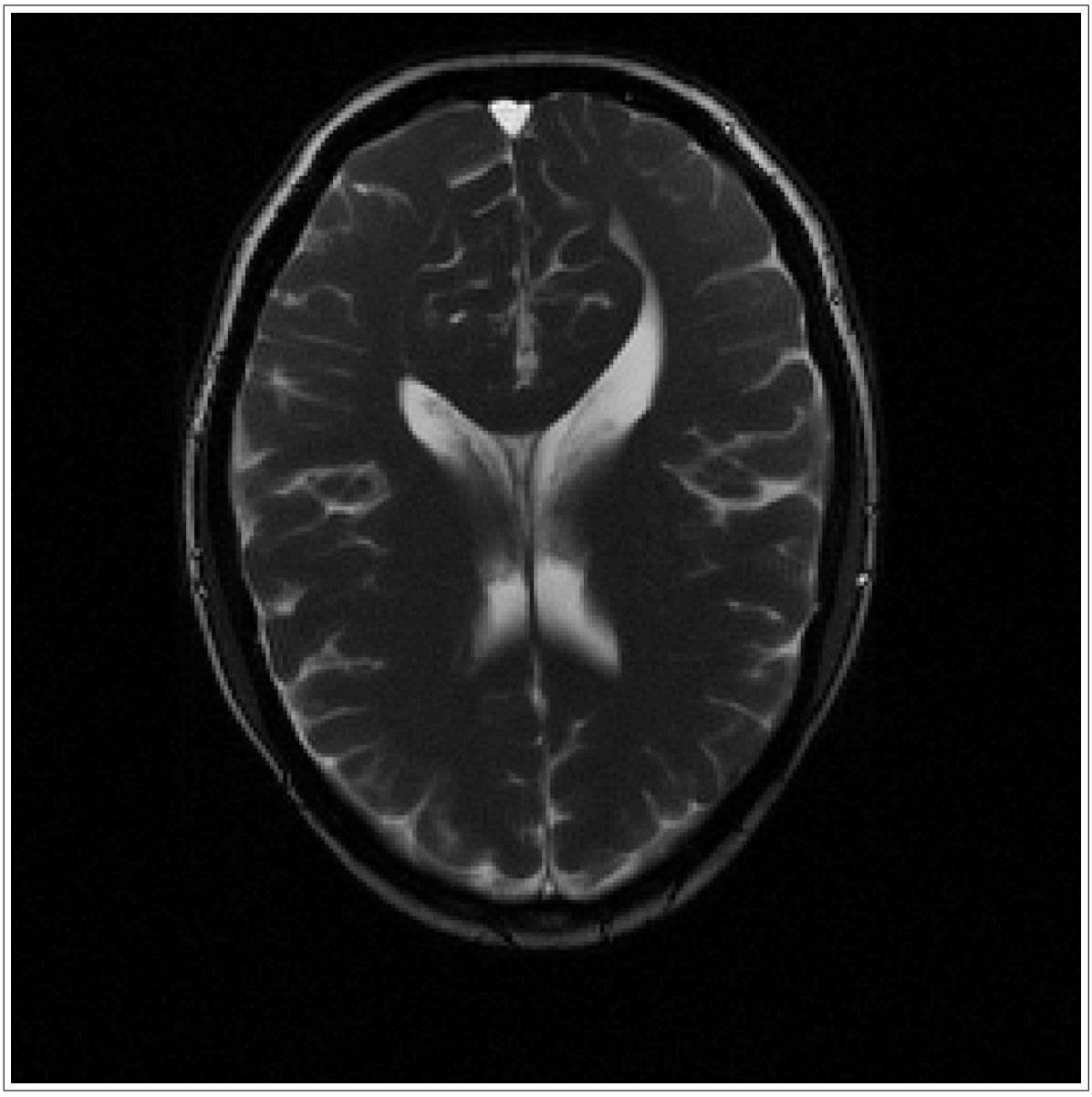

Figure 3.6 High resolution brain image reconstructed using RT-GROG (400 projections; 256 samples per each).

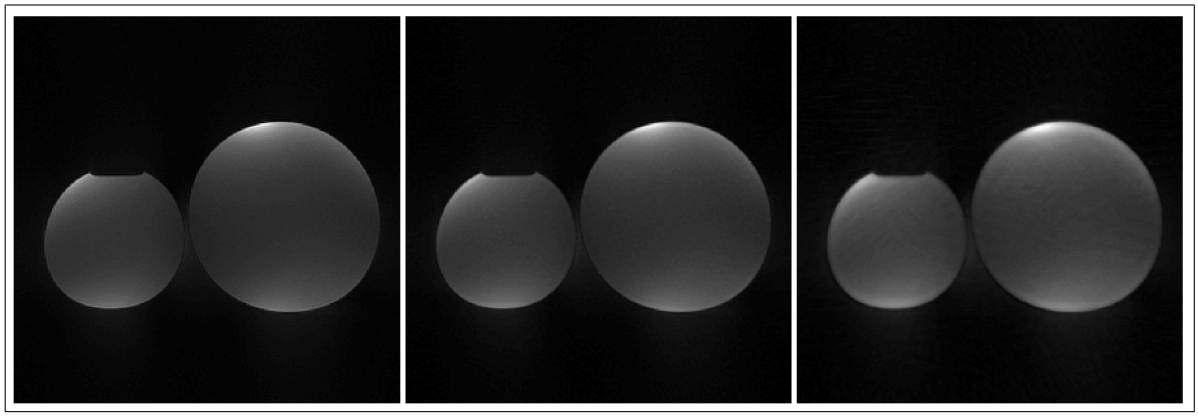

Figure 3.7 Phantom images reconstructed from a 15 coil data set using our real-time GROG implementation. From left to right: a. 256 projections b. 128 projections c. 64 projections. (Image brightness levels are adjusted for easier view of artifacts.)

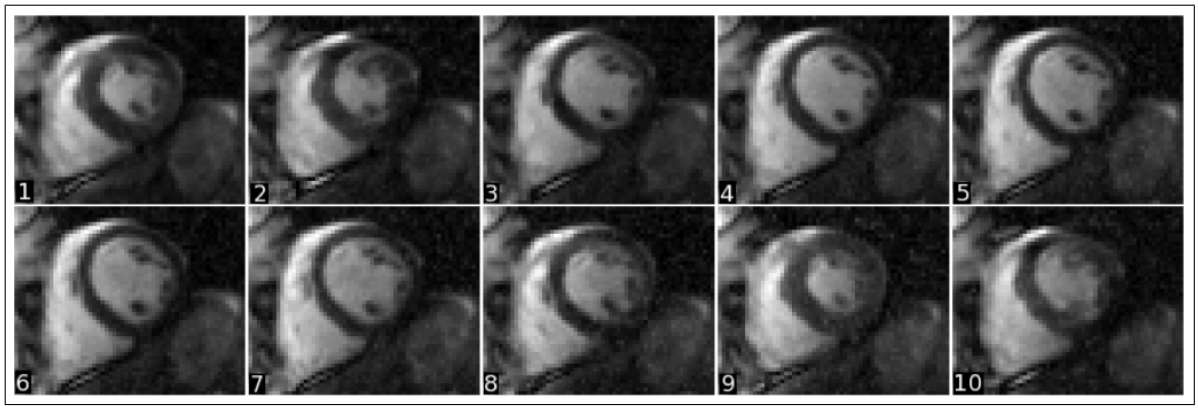

Figure 3.8 Cardiac cine image series reconstructed in real time from a 32 coil data set, 64 projections (128 samples). Acquisition time: 151 ms. Reconstruction time: 69 ms.

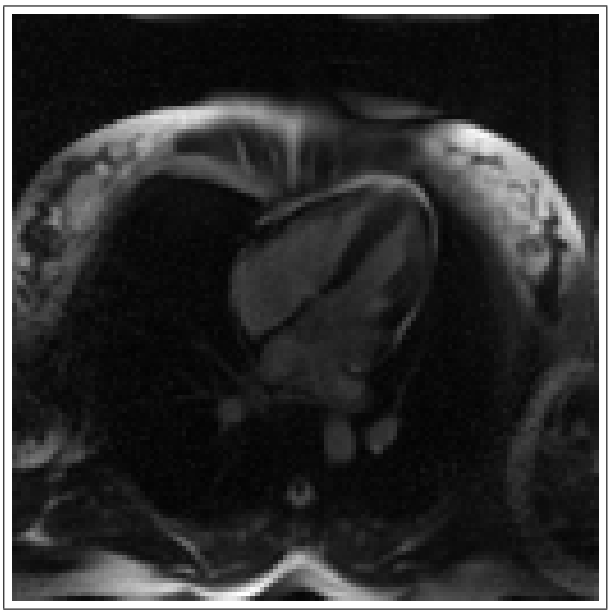

Figure 3.9 Long axis heart image reconstructed in real-time from 32 coil data set, 96 projections (128 samples). Acquisition time: 226 ms. Reconstruction time: 76 ms.

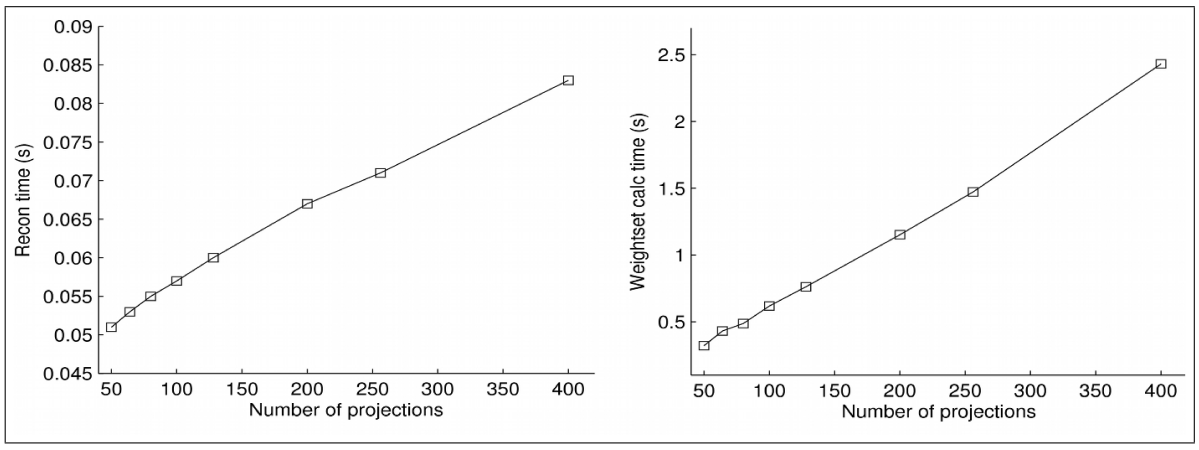

Figure 3.10 Image reconstruction and weight set calculation performances in seconds for 12 coil phantom data and different number of projections (number of samples = 256).

seconds) using phantom data acquired with 12 coils for different number of projections are given in Fig 3.10. With the 32 coil cardiac data set, the reconstruction is up to 3-times faster than data acquisition. For acquisitions of 128×96 (192×96, 128×64 respectively), the reconstruction rate is 13.2 fps (10.0 fps, 14.5 fps respectively), while the data acquisition rate is 4.4 fps (3.6 fps, 6.6 fps, respectively).

3.2.2.3 Discussion. SC-GROG provides a fast, robust and self-calibrating gridding method that perfectly fits real-time applications. Image reconstruction speed is between $52 - 83$ ms ($12.0 - 19.2$ fps) on our 12 coil phantom data set, while the acquisition is between $188 - 1504$ ms ($0.6 - 5$ fps). 32 coil cardiac data images are reconstructed between $69 - 101$ ms ($10.0 - 14.5$ fps), while data acquisition speeds are between $151 - 274$ ms ($3.6 - 6.6$ fps). Weight calculation speeds are between $321 - 2431$ for the phantom data, and $294 - 646$ ms for the cardiac data. Reconstruction is always faster than the data acquisition, and weight update speeds are acceptable. These results prove that RT-GROG suits well real-time imaging applications. View sharing method can be used to improve image resolution, but it will also introduce blurring due to motion. Other pMRI methods can be incorporated into the SC-GROG algorithm for undersampled data acquisitions, to estimate the missing k-space samples prior to gridding [79, 80]. Note that, our implementation can also be applied to constant-angular-velocity and reordered constant-linear-velocity spiral trajectories [28].

MR image reconstruction implicitly assumes that signal encoding employs perfectly linear gradients. In practice, gradient non-linearities, spatially varying inhomogeneities and time-varying eddy-currents cause deviations from this ideal. Furthermore, MR gradient coils have independent errors and delays that can cause deviations from the expected k-space trajectory. In general, it is quite difficult to be certain of the particular k-space location for a sample and to ascribe a unique value for it. Prior experience with non-Cartesian MR reconstruction suggests that the k-space location must be provided with significantly better precision than 1 k-space sample, but there is little to be gained by using better than 1/10 k-space sample precision.

To investigate this, frames are reconstructed using conventional GROG and RT-GROG. The signal difference between these images is less than 40% of the standard deviation of the noise of the original image, supporting the assertion that artifacts from using discrete GROG coefficients in the LUT are below the noise floor and essentially negligible. Additionally, 34-fold performance increase is observed when reconstructing of 32 coil, 192×96 data over conventional SC-GROG (also implemented in parallel) when our RT-GROG scheme is used. A performance comparison between conventional SC-GROG and RT-GROG is given in Table 3.1. A detailed comparison of the GROG and conventional gridding approaches for non-Cartesian MRI reconstruction is given in the original journal papers on GROG [27, 28] in terms of performance and image quality.

Table 3.1
Performance comparison (in seconds) between conventional SC-GROG and RT-GROG on 32 coil data sets with different matrix sizes.

	128×64	128×96	192×64	192×96
SC-GROG	1.412	1.972	2.285	3.4525
RT-GROG	0.069	0.076	0.101	0.123

Pre-computation of *2D* weights for our LUT scheme adds 75 ms (on average) to the weight calculation time on our 32 coil cardiac data set. As we obtain 34-fold reconstruction performance increase by using the LUT scheme and as one fixed set of SC-GROG coefficients can be used to grid multiple consecutive acquisition frames, this additional 75 ms is considered to be reasonable.

Our results emphasize 32-coil acquisitions. Additionally, a 16 core workstation is used during imaging. It may be difficult to provide such a workstation on a basic clinical environment. However, $10 - 12$ coil acquisitions provide decent image quality, and imply faster reconstruction performance. We believe that $10 - 12$ coil data sets can be reconstructed in real-time on a modern quad-core workstation. A GPU implementation can further decrease implementation costs. Scanner manufacturers are now providing multi-core reconstruction computers, therefore our reconstruction scheme

can be employed for clinical purposes with minimum effort. Additionally, OpenMP parallelism of the weight set calculation process may permit real-time synchronous SC-GROG reconstruction where a new set of weights per acquisition can be calculated without delays, if enough CPU resources are dedicated to the weight set calculation process.

3.3 Conclusion

Despite all their benefits, non-Cartesian trajectories cannot become popular due to their high computational requirements. As the data is not sampled on a Cartesian grid, FFT algorithm cannot be applied directly. First, the non-Cartesian data must be mapped onto a Cartesian grid. This operation is called "gridding". Many gridding methods have been proposed in the literature, conventional gridding, BURS, INNG, DING and many others, where image quality depends on several gridding parameters. Additionally, uneven sampling density in the k-space must be corrected by applying density compensation, which is a non-trivial task.

Convolution gridding is the most popular gridding algorithm. Image quality of this method is highly parameter dependent. First, a good convolution kernel must be selected. Second a good density compensation function must be used. Other additional parameters are also involved regarding the convolution kernel choice. More importantly, a priori information is used for the gridding process; information from the acquired data is not involved in any part of this algorithm.

BURS can be more efficient than conventional gridding in terms of computational complexity, as it does not require to interpolate onto a larger Cartesian grid, and no pre- or post-compensation steps are involved. However, this efficiency is very parameter dependent. The choice of radius δ_k and Δ_k has a direct effect on the accuracy of interpolation coefficients. Computational complexity highly depends on the choice of M'_{max}. M'_{max} can also affect the accuracy of the interpolation coefficients, as it limits the number of measurements for the computation of a given point k_i. Additionally,

this algorithm is very sensitive to noise, and proper regularization is not an easy task.

(B)INNG is an iterative algorithm, where density compensation is not required. With INNG algorithm, computational load and memory requirements can be relatively high depending on the rescaling factor, and matrix size. BINNG further decreases the computational requirements. However, image quality depends on the choice of the scaling factor, and the stopping criteria for the iteration.

DING is an iterative algorithm that utilizes conjugate gradient (CG) technique to perform gridding. It involves solving a sparse system of linear equations, which is equivalent to deconvolution operation. DING employs a Keiser-Bessel kernel as an interpolation window, and thus, the window parameter β must be properly calculated for efficient gridding. DING does not necessitate any density compensation function, as it is the part of the deconvolution process. Additionally, it does not require grid subsampling, which is needed for conventional gridding. However, in terms of computational load, DING is 8 times more demanding than conventional gridding. The choice of β value, stopping condition for the iteration and the increased computational load decreases the practicability of DING for real-time scenarios.

SC-GROG, is a self-calibrating gridding algorithm where no additional calibration data set is required. Gridding weights are calculated based on pMRI principles, using multiple coil information and acquired non-Cartesian k-space data set. More importantly, there are no parameters involved, and kernel size is fixed to 1, and thus, has low computational requirements. A new set of distinct gridding weights can be calculated for each frame. SC-GROG can be applied to grid both undersampled and fully sampled data sets. Furthermore, density compensation is straightforward. The quality of (SC)-GROG images is on par with conventional gridding.

Being a parameter-free, parallel-imaging based, robust, self-calibrating gridding algorithm, SC-GROG is a good candidate for real-time imaging. Its self-calibrating scheme benefits from pMRI, and at the same time does not involve any parameters at all. Convolution kernel selection is not needed anymore, as a new custom kernel

per acquired frame can be generated. Even though conventional gridding has some high performance implementations [76–78], there was no real-time oriented low-latency implementation for SC-GROG algorithm.

We present a parallelized real-time SC-GROG implementation for multi-slice radial acquisitions for real-time MRI. The weight-set calculation and the image reconstruction steps are decoupled, and parallelized to run in real-time on a general purpose multi-core architecture. Each process runs asynchronously. Weights calculation threads continuously updates the weights in the background, and the reconstruction threads are using the most recent weights to grid and reconstruct images. Per sample gridding weight calculation step is eliminated from the reconstruction process by using a LUT. Subsequent to the calculation of gridding weights G_x and G_y, all possible combinations of per-sample *2D* weights $(G_x)^m(G_y)^n$ are evaluated for $m, n = \{-0.5, -0.4, ...0, 0.1, ...0.5\}$ and stored in the LUT by the weight set calculation process. In this way, reconstruction performance is drastically improved. Our implementation is auto-calibrated, and supports multiple slice acquisitions. Additionally, there are no parameters. All the necessary information is extracted from the acquired data itself, rather than using some *a priori* information. Highly parallel architecture of our implementation overcomes slice plane changes during real-time MRI. If required, It is possible to change number of threads associated to weight calculation dynamically. RT-GROG provides up to 34-fold reconstruction performance increase over conventional SC-GROG during the reconstruction of the 32 coil cardiac data set. Reconstruction is always faster than the data acquisition with RT-GROG. For improved image quality on under-sampled data sets, other pMRI methods can be incorporated into the SC-GROG algorithm in order to estimate the missing k-space samples prior to gridding, as explained in [79, 80].

4. IMPROVED VISUAL FEEDBACK DURING REAL-TIME MRI GUIDED INTERVENTIONS

Real-time MRI guided interventions gained a huge potential for clinical application by current technological developments, such as short, wide bore magnets, high-performance gradient systems, phased-array receivers [11] and arrival of MR compatible instruments, such as needles, active guidewires and catheters [1–3, 10]. Furthermore, real-time pulse sequences were further improved, and new imaging techniques based on parallel MRI (pMRI) principles were developed by which real-time MRI guided interventions became more applicable [32–42, 52, 69].

Interventions require fast data acquisition to obtain real-time feedback. This requirement is partly fulfilled by efficient real-time pulse sequences. As the gradients reached to the physical limits, further improvement in the imaging frame rate were achieved by parallel imaging [11], which employed multiple receiver coils to replace skipped phase encoding steps during partial data acquisition (see Chapter 2). Efficient k-space domain [33, 36] and image domain [39] pMRI reconstruction algorithms were proposed. These algorithms require significant amount of computation resources to provide the necessary reconstruction speed with reasonable image quality. Recent technological advances have allowed parallel MRI reconstruction in real-time on a single memory space multi-processor workstations [42, 52], and on a GPU [73].

A small deviation from the desired device trajectory may result in a severe complication during interventions. Active guidewires that acquire MR data [1–3] facilitate device tracking in real-time. Furthermore, pseudo color assignment to each active device channel dramatically improves the device visualization during interventions [10, 53]. However, real-time pulse sequences are usually restricted to single or multiple *2D* slices to provide necessitated imaging speed. Visualization of structures outside the imaging plane are often needed, such as different portions of or different vascular trees. Contrast-enhanced MR angiography offers excellent depiction of such

structures but cannot be performed in real-time and repeated at will. However, if we could import these data sets and display them in conjunction with real-time reconstructed *2D* slice images, this drawback would be eliminated. Our custom MR data reconstruction and visualization environment dynamically renders multiple colorized MIPs in real-time using previously acquired magnetic resonance angiography (MRA) and magnetic resonance cholangiopancreatography (MRCP) data, and displays them interactively with multiple slice real-time MR images using alpha-blending [55]. Our real-time imaging environment also colorizes image data from active invasive devices and blends them with surface coil images. That way, multiple active invasive devices can be visualized and localized in the additional context of multiple vascular structures represented by uniquely colorized dynamic MIPs. As a result, the advantages of real-time imaging and non-real-time imaging were combined during real-time MRI guided interventions.

Uniquely coloring each device channel signal [53] improves the visualization of the devices by providing a colorful roadmap integrated with grayscale anatomical MR images. However, different environmental factors, especially during *in vivo* experiments, (e.g noise, motion) add up and colored device representation may occasionally obscure underlying anatomical structures immediately around the device (e.g. mitral valve, aortic valve structures); this is critical especially if the device diameter is 2 mm or larger. Signal from device channels can be interactively scaled to adjust brightness and visibility. But this scaling also increases the noise level, and cannot prevent artifacts due to device coupling. Device coupling occurs inevitably due to dimensional restrictions of small devices and may cause difficulties for operators in determining the actual position/orientation of them. If one can show only the top ridge or specific edges of active device signal, visibility of fine anatomical structures immediately around the device can dramatically improve.

Edge detection is a well established subject in image processing [81]. Golden rules in edge detection are described by Canny in 1983 [82]. Deriche develop a computationally efficient recursive edge detection algorithm [83] based on Canny's method. This recursive approach allows implementing infinite impulse response filters, while

dramatically reduces computational cost [84]. We integrate Deriche's edge detection algorithm to our in-house developed real-time MRI environment to show only the top ridge or specific edges of the active device signal during real-time MRI guided interventions. A dedicated edge detector is assigned to each device channel. That way, edges are detected only on interactively selected channels. Subsequent to detection, edges are colored for further improved visual feedback, and displayed by alpha-blending them with real-time reconstructed anatomical MR images. Our results shows that edge detection on data from device channels is helpful in recovering otherwise somehow obscured anatomical structures, and it drastically improves visual feedback during real-time MRI guided interventions.

Active device localization during real-time MRI guided interventions is a big challenge. Scan plane changes often occur during intervention during when device can be out of the FOV. This is somewhat addressed by device-only projection imaging by switching off the slice-encode gradient. Slice positioning for better device visualization is generally realized by trial-and-error methods. What we need is (i) to automatically localize the device, (ii) to prescribe a given slice to that location, and (iii) to show the estimated location/trajectory of the device in interactive *3D* display window. For that purpose, an already existing Matlab code for automatic device repositioning is converted to *C++*, and integrated into our custom MRI reconstruction software as a separate thread. Three perpendicular projection images are acquired and fed to the device localization thread by pressing a button. Subsequent to projection image acquisition, normal real-time imaging is performed, while device location is estimated in the background. When the estimate is ready, a given slice is prescribed to that location for maximum view of the device. Additionally, this estimate was used to draw the device on our *3D* window for further improved visual guidance, in conjunction with real-time acquired multi-slice images.

In this Chapter, thesis contributions achieved on improving visual guidance during real-time MRI guided interventions will be explained. First, our fully MRI based, multiple colorized *3D* angiography roadmaps scheme will be explained in details. Second, improved device visualization by edge detection methods will be presented.

After giving the integration details of the device localization software to our custom MRI reconstruction environment, the impact of these contributions, as well as their weaknesses will be discussed.

4.1 Interventional MRI using Multiple *3D* Angiography Roadmaps with Real-time Imaging

We aim to enhance visual feedback by overlaying colorized multiphase MR angiography (MRA) and cholangiopancreatography (MRCP) roadmaps. That way, anatomical context can be drastically improved. For that purpose, time-resolved MRA and respiratory-gated MRCP are acquired prior to real-time imaging in a pig model. MRA and MRCP data are cropped and then processed in our custom real-time MRI reconstruction and visualization workstation where they are displayed as Maximum Intensity Projections (MIPs) [55]. Distinctly colored MIPs are rendered in three dimensions (*3D*) and employed in conjunction with real-time multi-slice imaging data using alpha blending. Interactive rotation allows different views of the combined data. Our software facilitates the differentiation of different vascular structures during real-time imaging. Overall, highly increased visual feedback is obtained.

In the following sections, the implementation details, and the results of our software will be discussed.

4.1.1 Methods

Animal protocols are approved by the institutional animal care and use committee in accordance with the National Institutes of Health (NIH) guidelines. Pigs undergo ketamine and/or telezol/xylazine induction followed by endotracheal intubation, mechanical ventilation, and $1-2\%$ isoflurane. During femoral artery and venous access, percutaneous techniques are used.

Time-resolved MRA of the great vessels and respiratory-gated MRCP images of a pig are acquired prior to the real-time MRI guided procedure, using a short, wide bore Siemens Espree 1.5 T MRI scanner (Siemens Medical Solutions, Erlangen, Germany). During a breath hold, the MRA is acquired with a *3D* Fast Low Angle Shot (FLASH) sequence with the following parameters: TR = 2.8 ms, TE = 1.2 ms, flip angle = 24 degrees, FOV = 327×327 mm, and voxel size = 2.3 mm × 1.3 mm ×2 mm. For MRCP imaging, a respiratory navigator gated fast spin echo sequence (Half-Fourier Acquisition Single-Shot Turbo Spin-Echo (HASTE)) is used with the following parameters: TR = 2773 ms, TE = 620 ms, flip angle = 180 degrees, FOV = 380×380 mm, and voxel size = 1.0 mm ×1.0 mm ×1.5 mm. Real-time SSFP images are acquired with TR = 3 ms, flip angle = 45 degrees, FOV = 340×255 mm with acquisition matrix of 192×108. 18 receiver coils are used during real-time imaging, and HTGRAPPA [42] is employed as a parallel imaging technique with acceleration rates of 2 and 3. Image reconstruction and display is performed on a general purpose Linux workstation with two dual-core AMD Opteron 2216 processors and with a NVIDIA Quadro FX graphics card.

The structures of interest are manually cropped from previously acquired multi-phase MRA and MRCP data using Siemens Syngo *3D* tools (Leonardo, Siemens). Subsequent to cropping, results are exported as DICOM image series and then loaded onto the real-time MRI workstation at the beginning of the intervention. SGI's Volumizer library (http://www.sgi.com/products/software/volumizer) is used to render *3D* colorized MIPs in real-time. These colorized MIPs are displayed in *3D* in conjunction with real-time reconstructed multi-slice images and colorized active guidewire images during the intervention.

This capability is implemented in our interactive real-time MRI scan control and display system described in [53]. The MIPs are dynamically re-rendered from scratch according to the new point of view (POV) as the user rotates the scene. Our implementation allows multiple multiple angiography data sets. Each MIP corresponding to a given angiographic data set can be individually turned on or off. Joint visualization is possible by interactively grouping MIPs of interest. With these capabilities, our improved real-time visualization interface can show only what is needed at a particular

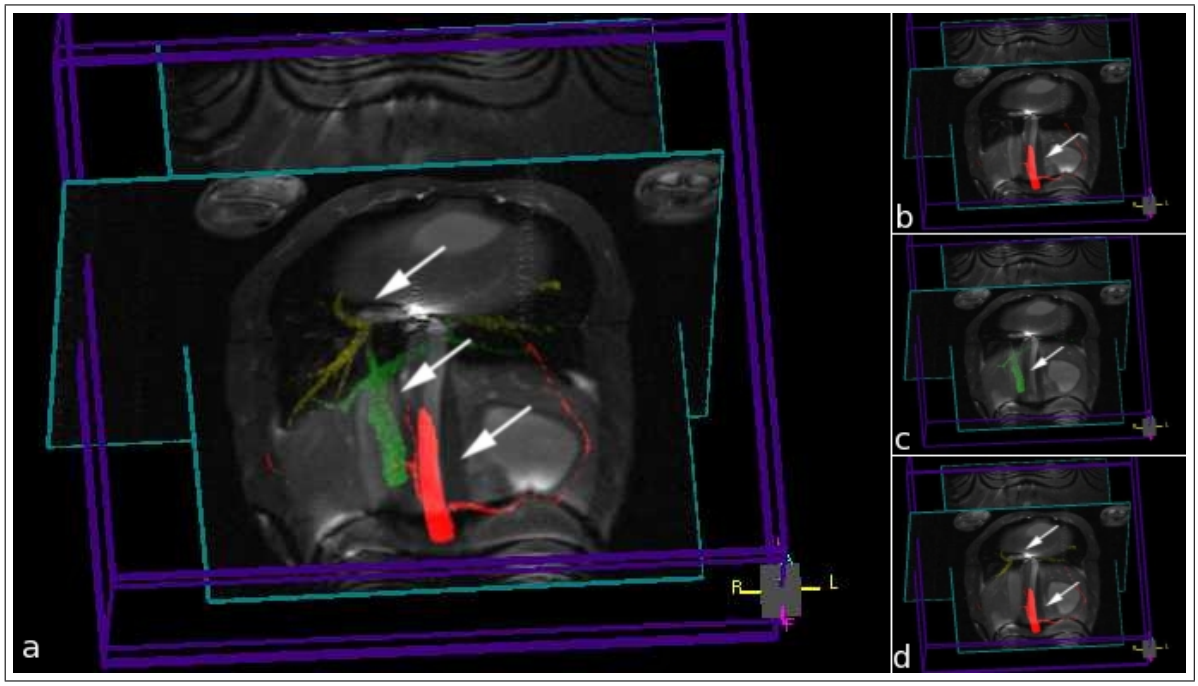

Figure 4.1 Images captured from the *3D* rendering for use in MRI-guided TIPS experiment showing multiple colorized MIPs rendered with two orthogonal real-time image slices. Each MIP is indicated by an arrow. a) All MIPs are visualized. b) Only hepatic artery (red) is shown c) Only portal vein (green) is shown. d) Hepatic artery (red) and hepatic vein (yellow) are grouped and shown.

stage of a procedure to avoid extra clutter in the rendering (see Figure 4.1).

4.1.2 Experiments

The *3D* colorized roadmaps utility is deployed in real-time during (i) interventional pulmonary thrombectomy, and (ii)interventional transjugular intrahepatic portosystemic shunt (TIPS) procedures.

Angiographic data are acquired prior to the invasive portion of the experiment, and further processed for the generation of related *3D* MIPs.

4.1.2.1 Pulmonary Thrombectomy. Venous thromboembolism occurring in the pulmonary arteries is a common cause of death [85]. Massive pulmonary thromboem-

bolism causes hemodynamic compromise or collapse, and mechanical treatment options are limited [85]. New real-time MRI techniques are being developed at the National Institutes of Health (NIH) to relieve vascular obstruction and rescue patients suffering massive pulmonary thromboembolism. Differentiation of different venous structures are crucial to visualize the boundaries with respect to *3D* anatomical locations during this procedure. Real-time imaging fused with multiple colorized angiographic roadmaps may improve efficacy of this treatment by providing improved visual feedback.

4.1.2.2 TIPS. Percutaneous transjugular intrahepatic portosystemic shunting [54] relieves portal venous congestion due to cirrhosis. In this procedure, a catheter traverses hepatic parenchyma to connect a portal vein to a hepatic vein. If no image guidance is available, multiple "blind" attempts are required to traverse the liver from a vein using a sharp needle. Therefore life-threatening non-target punctures, and significant patient discomfort may occur. TIPS has several critical stages: (i) initial venous access, (ii) angioplasty, and (iii) stent-graft insertion. Differentiation of different venous, arterial and biliary structures, and knowledge of their *3D* relations are crucial during the procedure which can be difficult under X-ray fluoroscopy. The biliary tree and hepatic artery must be avoided while placing a shunt between the portal and hepatic veins. This procedure may also benefit from multiple colorized angiography roadmaps. TIPS is typically performed using X-ray fluoroscopic guidance but has been reported in humans using a combined "double-donut" MRI and X-ray system [86].

4.1.3 Results

Figure 4.2 shows the angiography MIPs of pulmonary veins (in yellow) and pulmonary arteries (in light blue) with colored active catheter image (arrow) blended with the real-time multi-slice images. The active catheter contains separate receiver channels for the tip and the shaft. The small coil at the tip is represented by red color, and the longer shaft coil is represented by green. The visibility and differentiation of the pulmonary arteries and veins are provided by using separately colored MIPs during

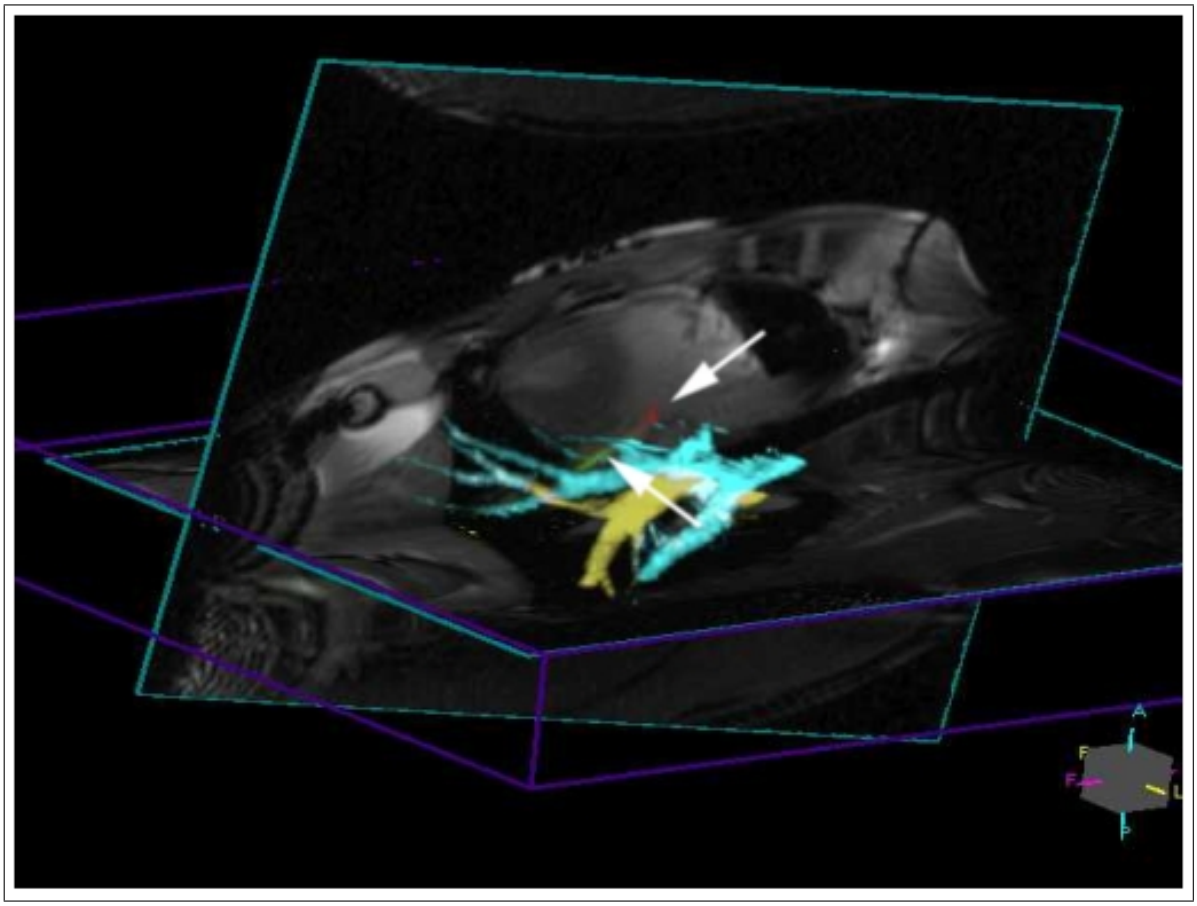

Figure 4.2 Colorized *3D* MIP of pulmonary veins (yellow) and arteries (light blue) shown in conjunction with colorized active device (marked with arrows: tip shown in red, shaft shown in green) and two real-time image slices.

the experiment. Pulmonary artery branches are accessed via femoral veins under the guidance of *3D* MRA MIPs.

Figure 4.3 shows dynamically rendered *3D* jugular vein MIP (in yellow) overlaid on real-time multi-slice images. In Figure 4.4, dynamically updated *3D* MIPs of biliary tree (in yellow), active guidewire (in blue), hepatic artery (in red), hepatic vein (in yellow), and portal vein (in green) are displayed in conjunction with real-time reconstructed multi-slice images. Real-time guidance during (i) jugular access, (ii) safe traversal to the liver avoiding biliary tree and hepatic artery, and (iii) percutaneously connecting hepatic and portal vein, is provided by *3D* MRA MIPs.

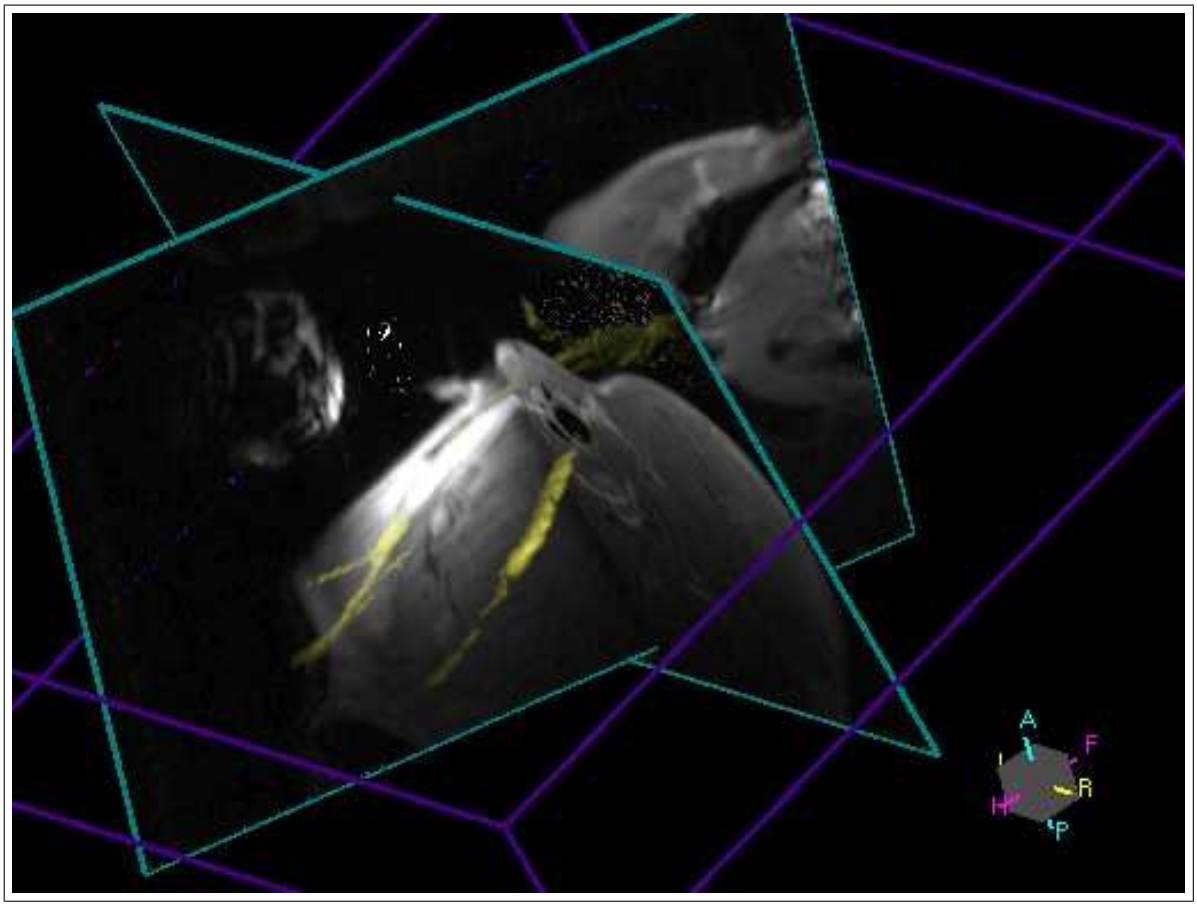

Figure 4.3 *3D* colored MIP of jugular vein displayed with real-time images for guidance during TIPS procedure for jugular access.

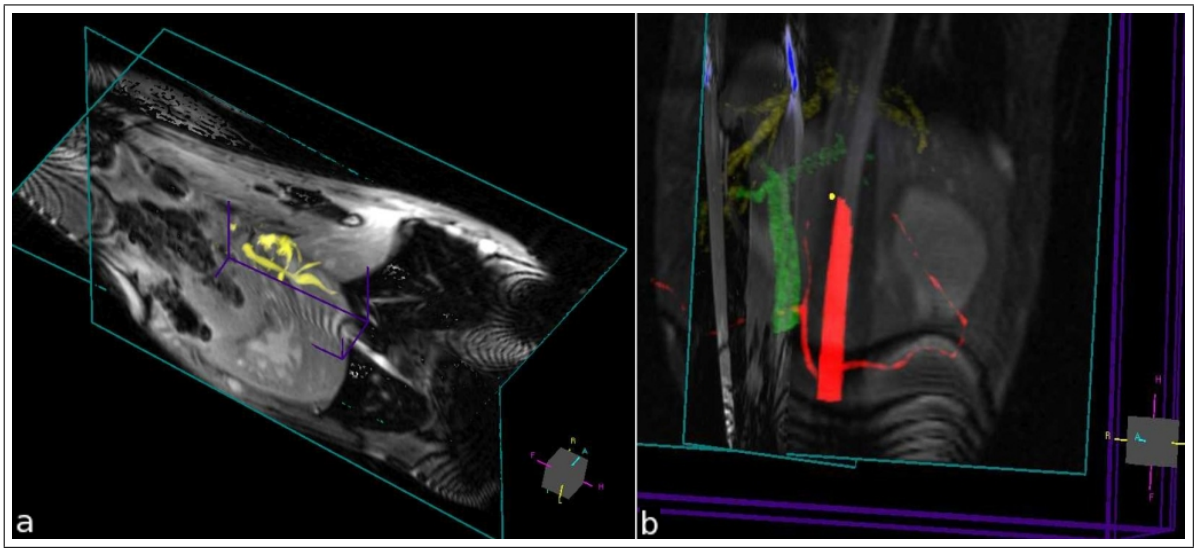

Figure 4.4 *3D* colored dynamically updated MIPs displayed in conjunction with real-time multi-slice images during MRI guided TIPS procedure. a) Biliary tree in yellow. b) Active guidewire in blue, hepatic artery in red, hepatic vein in yellow, and portal vein in green.

4.1.4 Discussion

2D imaging is used during real-time MRI guided vascular interventions in general, for faster imaging. Therefore visual feedback is limited. Previously acquired, non-real-time data sets, such as MRA or MRCP can be used to enhance visual feedback and thus can improve the anatomical context during the intervention.

We present a software module that dynamically creates and colorize *3D* MIPs using previously acquired MRA and MRCP data sets. Subsequent to the acquisition of MRA and MRCP data sets, vascular structures of interest are cropped, and exported as DICOM image series. These image series are then loaded into our real-time reconstruction environment, prior to the intervention. *3D* MIPs are rendered in distinct colors and visualized on our *3D* display window with partial image transparency. That way, multiple distinct data sets can be visualized in conjunction with each other, and drastically increased visual feedback can be provided during the intervention. All angiographic data sets are loaded to the system at the beginning of the invasive portion of the procedure, but it is possible to selectively display MIPs of interest required for a particular step of the procedure. Similar work has been presented for X-ray/CT and

XMR systems in [56–58], and commercially implemented by Siemens (Syngo iPilot tool, Siemens Medical Solutions, Germany) and by Philips Medical Systems (Koninklijke Philips Electronics N.V.). Some of these methods permit using previously acquired MRI data during interventions, but they still necessitate to acquire X-ray/CT imaging, and thus imply patient radiation exposure. Our method totally avoids radiation, as it is totally MRI based. Additionally, *3D* MRA/MRCP roadmaps and real-time reconstructed multi-slice images share the same coordinate system (if acquired by the same scanner) and, therefore, do not require registration.

Roadmaps obtained using previously acquired data have some inherent limitations. First, they may become outdated. That is why it is better to acquire angiographic data just before the intervention. Second, the roadmaps are static and thus may misalign with "live" images due to cardiac, respiratory, or other patient motion. Breath-holding, electrocardiography-gating (ECG-gating), more sophisticated motion correction techniques, or periodic intraprocedural updates, might enhance the accuracy of our *3D* roadmap utility.

4.2 Improved Visual Feedback with Real-Time Edge Detection during MRI Guided Interventions

Device visualization is a real challenge during real-time MRI guided interventions. Passive devices can be visualized if they comply with some physical criterion, i.e. material type or size. However, these devices provide limited visual feedback. On the other hand, active devices have their own receiver channels to acquire MR signal that can be further interpreted in order to localize the device. Active device visualization by assigning pseudo colors to each device channel has been described previously [53]. In that approach, device channel data is reconstructed separately using view sharing method, and then reconstructed images are colored according to the pseudo color assigned to that channel. Subsequent to coloring, these images are displayed in conjunction with real-time reconstructed images using alpha-blending, and visual feedback

for device visualization is improved.

Uniquely coloring each device channel signal improves the visualization of the devices by providing a colorful roadmap integrated with grayscale anatomical MR images. However, different environmental factors, especially during *in vivo* experiments, (e.g noise, motion) add up and colored device representation may occasionally obscure underlying anatomical structures immediately around the device (e.g. mitral valve, aortic valve structures); this is critical especially if the device diameter is 2 mm or larger. Signal from device channels can be interactively scaled to adjust brightness and visibility. But this scaling increases the noise level, and cannot prevent artifacts due to device coupling. Device coupling occurs inevitably due to dimensional restrictions of small devices and may cause difficulties for operators in determining the actual position/orientation of them. If one could show only the top ridge or specific edges of active device signal, visibility of fine anatomical structures immediately around the device could dramatically improve.

Edge detection is a well established subject in image processing. Detailed analysis and comparison of different edge detection methods are previously studied in [81]. Canny described the golden rules for a perfect edge detector, and developed an efficient implementation [82]. Couple of years later, Deriche extended Canny's approach, and developed a computationally more efficient recursive edge detection algorithm [83]. This recursive approach allows to implement filters with infinite impulse response, while dramatically reducing computational cost of edge detection [84].

We integrate Deriche's edge detection algorithm to our in-house developed real-time MRI environment to show only the top ridge or specific edges of the active device signal during real-time MRI guided interventions. A dedicated Deriche edge detector is assigned to each device channel. That way, edges are detected only on interactively selected channels. Subsequent to detection, edges are colored for further improved visual feedback, and displayed by alpha-blending them with real-time reconstructed anatomical MR images, as done in [53]. Our results shows that edge detection on device channels data is helpful in recovering otherwise obscured anatomical structures,

and it drastically improves visual feedback during real-time MRI guided interventions. Experiments show that edge detection is also helpful to decrease the device coupling artifacts to some extent.

In the following sections, the concept of edge detection will be briefly explained, and, Canny's and Deriche's edge detectors will be introduced. After theoretical introduction, the integration details, and experimental results will be given. Finally, our method will be discussed to reveal advantages, drawbacks, and potential improvements.

4.2.1 Edge-Detection

Edge detection techniques are useful in computer vision field to determine physical, photometrical and geometrical properties of objects in an image. Physical edges in an image are represented by changes in the intensity functions, steps, lines and junctions [81].

Step edges are represented by gray-level discontinuities in an image. Detection of edges generally involves: (i) smoothing, (ii) differentiation, and (iii) edge localization. Smoothing is used to reduce the noise in the original image, and thus ensures accurate edge detection. However, it also incorporates information loss, due to the filtering. Therefore, there is a trade-off between reduced noise and amount of information. Differentiation is used to calculate desired derivatives of the image. The most common differentiation operations are gradient operators, Laplacian and the second-order directional derivative.

Two dimensional gradient operators are first-order operators that can be represented by the vector $g = (\frac{\partial}{\partial x}, \frac{\partial}{\partial y})$. The modulus of the gradient is $|\vec{g}| = \sqrt{(\frac{\partial}{\partial x})^2 + (\frac{\partial}{\partial y})^2}$, and the direction of the gradient is $\Phi = arctg(\frac{\partial}{\partial x}/\frac{\partial}{\partial y})$. In most of the edge detection techniques, the direction of the gradient, which is perpendicular to the edge orientation, is used to detect edges. For a computationally more efficient calculation of the modulus, $|\vec{g}| = |\frac{\partial}{\partial x}| + |\frac{\partial}{\partial y}|$ or $|\vec{g}| = max(|\frac{\partial}{\partial x}|, |\frac{\partial}{\partial y}|)$ can also be used. However, these

decrease the accuracy of the edge detection [81].

The second order operators are defined as $\frac{\partial^2}{\partial x^2}$, $\frac{\partial^2}{\partial x \partial y}$, and $\frac{\partial^2}{\partial y^2}$. Laplacian operator $\nabla = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$, and the second-order directional derivative along the gradient direction $\frac{\partial^2}{\partial \vec{n}^2} = \frac{\partial^2}{\partial x^2} cos^2(\Phi) + \frac{\partial^2}{\partial xy} sin(\Phi) cos(\Phi) + \frac{\partial^2}{\partial y^2} sin^2(\Phi)$ are the most commonly used second-order operators.

Edge localization determines edges, while suppressing false edges to increase SNR. One of the most efficient ways to label edges is to extract local maxima of the gradient modulus [81]. If gradient approach is used, a given pixel p is a local maximum if the value of $|\vec{g_p}| > |\vec{g_{n_1}}|$ and $|\vec{g_p}| > |\vec{g_{n_2}}|$, where $|\vec{g_{n_i}}|$ represents i^{th} neighboring pixel at the same distance on either side of p along Φ. If a second-order detector is used, the sign of p is compared with the sign of neighboring pixels to the left and below. If the sign differs, a zero-crossing is reached [81].

4.2.1.1 Canny's Edge Detector. Canny described the criterias for a good edge detector as follows [82]:

1. Good detection: low probability of failing to detect real-edge points, and low probability of detecting false edges.
2. Good localization: the points labelled as edges must be as close as possible to the center of the true edge.
3. Only one response must be provided to a single edge.

where criterion 1 and criterion 2 represent a trade-off.

Assume that A is the amplitude of the step, and n_0^2 is the variance of the input white noise. The input signal $I(x)$ can then be represented as $Au_{-1}(x) + n(x)$, where u_{-1} represents the unit step function. If the noisy edge is convolved with a

spatial anti-symmetric function $f(x)$, the maxima in the output of this convolution ($\theta(x_0) = \int_{-\infty}^{+\infty} I(x)f(x_0 - x)dx$) will detect edges [83].

Criterion 1 corresponds to maximize SNR. It defines SNR as the quotient of the step only response and the square root of mean-squared noise response:

$$SNR = \frac{A}{n_0} \frac{\int_{-\infty}^{0} f(x)dx}{\sqrt{\int_{-\infty}^{+\infty} f^2(x)dx}} = \frac{A}{n_0}\Sigma \tag{4.1}$$

Best operator $f(x)$ for detection can be found by maximizing Σ.

Criterion 2 corresponds to minimizing the variance σ^2 of the zero-crossing position. The best operator $f(x)$ can be calculated using

$$L = \frac{A}{n_0} \frac{|f'(0)|}{\sqrt{\int_{-\infty}^{+\infty} f^2(x)dx}} = \frac{A}{n_0}\lambda \tag{4.2}$$

by maximizing λ.

Criterion 3 corresponds to avoid multiple outputs to a single edge. Number of peaks in the output must be limited to decrease the probability of detecting one edge more than once. Let x_{max} be the distance between peaks in the noise response of f defined as

$$x_{max} = kW = 2\pi \left[\frac{\int_{-\infty}^{\infty} f'^2(x)dx}{\int_{-\infty}^{\infty} f''^2(x)dx} \right] \tag{4.3}$$

where W describe the width of the operator, and k some fraction.

Having described all the criteria, Canny deduce that the best operator to detect edges can be found by maximizing the product $\Sigma\lambda$ under the constraint of Eq. 4.3. He later finds that his maximization leads to the solution $f(x)$, as

$$2f(x) - 2\lambda_1 f''(x) + 2\lambda_2 f''''(x) + \lambda_3 = 0 \tag{4.4}$$

with $\Delta = \lambda_2 - \lambda_1^2/4$, and where α and ω are real, such that

$$\begin{aligned} \alpha^2 - \omega^2 &= \frac{\lambda_1}{2\lambda_2} \\ 4\alpha^2\omega^2 &= \frac{-\lambda_1^2 + 4\lambda_2}{4\lambda_2^2}. \end{aligned} \tag{4.5}$$

General solution in the range of $[0, W]$ can be expressed as

$$\begin{aligned} f(x) &= a_1 e^{\alpha x} sin(w)x + a_2 e^{\alpha x} cos(w)x + \\ & \quad a_3 e^{-\alpha x} sin(w)x + a_4 e^{-\alpha x} cos(w)x + C \end{aligned} \tag{4.6}$$

with boundary conditions of $f(0) = 0$, $f(W) = 0$, $f'(0) = S$ and $f'(W) = 0$, where S defines the slope of the function $f(x)$ at the origin. This solution can be extended to range $[-W, W]$ by using $f(x) = -f(-x)$, as $f(x)$ is anti-symmetric. a_i for $i = \{1, 2, 3, 4\}$ can be determined by the boundary conditions. Canny finds that the largest value for the constant k is 0.58, and its performance is $\Sigma\lambda = 1.12$.

Canny observes that the shape of the optimal edge detection operator is approximately the first derivative of a Gaussian, which is computationally more efficient: $f(x) = \frac{-x}{\sigma^2} e^{-x^2/2\sigma^2}$. Note that overall performance index for the first derivative of the Gaussian is 20% less than the optimal operator.

Canny edge detection involves following steps:

1. convolve the image with a symmetric *2D* Gaussian
2. differentiate the result in two directions
3. estimate the edge direction from the gradient of the smoothed image intensity surface
4. suppress the non-maxima in gradient direction from the gradient magnitude

4.2.1.2 Deriche's Edge Detector. Deriche extends Canny's approach by using an infinite extent recursive filtering approach [83,84]. This approach results with a computationally more efficient edge detector, while conforming with Canny's best detector criteria definitions.

Canny implements his design using a finite extent anti-symmetric filter. An infinite extent anti-symmetric will also lead to the same general solution represented in Eq. 4.6, but with the following boundary conditions: $f(0) = 0$, $f(+\infty) = 0$, $f'(0) = S$, and $f'(+\infty) = 0$. Under these conditions, following solution is obtained [83]:

$$f(x) = -ce^{-\alpha x} sin(\omega)x \tag{4.7}$$

where α, ω, and c are positive real numbers. Using the anti-symmetry of $f(x)$, Eq. 4.7 can be expressed for any x as follows:

$$f(x) = -ce^{-\alpha|x|} sin(w)x \tag{4.8}$$

If Fourier transform is applied to that filter,

$$F(u) = \frac{4ic\alpha\omega u}{(\alpha^2 + \omega^2) + u^4 + 2u^2(\alpha^2 - \omega^2)}. \tag{4.9}$$

where u represents the spatial frequency.

Deriche evaluates the integral in the performance criterion, and obtaines the following results [83]:

$$\begin{aligned} |f'(0)| &= \omega c \\ \int_{-\infty}^{0} f(x)dx &= \frac{\omega c}{\alpha^2 + \omega^2} \\ \int_{-\infty}^{+\infty} f^2(x)dx &= \frac{\omega^2 c^2}{2\alpha(\omega^2 + \alpha^2)} \\ \int_{-\infty}^{+\infty} f'^2(x)dx &= \frac{\omega^2 c^2}{2\alpha} \\ \int_{-\infty}^{+\infty} f''^2(x)dx &= \frac{\omega^2 c^2(5\alpha^4 + \omega^4 + 6\alpha^2\omega^2)}{2\alpha(\omega^2 + \alpha^2)}. \end{aligned} \tag{4.10}$$

Therefore, following overall performance index for his edge detection operator are ob-

tained:

$$\begin{aligned} \lambda &= \sqrt{2\alpha} \\ \Sigma &= \sqrt{\frac{2\alpha}{\alpha^2+\omega^2}} \\ \Sigma\lambda &= \frac{2\alpha}{\sqrt{\omega^2+\alpha^2}} \\ k &= \sqrt{\frac{\alpha^2+\omega^2}{5\alpha^2+\omega^2}}. \end{aligned} \tag{4.11}$$

Deriche obtaines four interesting cases by setting $\alpha = m\omega$, where m is a real valued parameter [83]:

1. **Case 1:** $m = \sqrt{3}$, $\lambda = \sqrt{2\alpha}$, $\Sigma \cong \sqrt{\frac{3}{2\alpha}}$, $\Sigma\lambda \cong 1.732$, and $k \cong 0.5$. Hence, for the same value of k, the performance of the proposed method is nearly two times more than the first derivative of a Gaussian.

2. **Case 2:** $m = 1$, $\lambda = \sqrt{2\alpha}$, $\Sigma \cong \sqrt{\frac{1}{\alpha}}$, $\Sigma\lambda \cong 1.414$, and $k \cong 0.58$. Hence, the performance of the optimum operator used by Canny is 25% worse than Deriche's operator.

3. **Case 3:** $m << 1$, $\lambda = \sqrt{2\alpha}$, $\Sigma \cong m\sqrt{\frac{2}{\alpha}}$, $\Sigma\lambda \cong 2m$, and $k \cong 1$. This case represents an ideal response measure for k. However, the product $\Sigma\lambda$ becomes much less then unity.

4. **Case 4:** $m >> 1$, $\lambda = \sqrt{2\alpha}$, $\Sigma \cong \sqrt{\frac{2}{\alpha}}$, $\Sigma\lambda = 2$, $k = \frac{1}{\sqrt{5}} \cong 0.44$. This case represents the best trade-off.

As $\Sigma\lambda$ is maximum when $\omega = 0$, and $sin(\omega x) \cong \omega x$ for very small ωx, Deriche's optimum filter can be rewritten as

$$g(x) = -cxe^{-\alpha|x|} \tag{4.12}$$

which is a solution of the Eq. 4.4, when $\Delta = 0$. Following these results, Deriche evaluates the overall performance index for his edge detector [83]:

$$\lambda = \sqrt{2\lambda}$$

$$\begin{aligned} \Sigma &= \sqrt{\frac{2}{\alpha}} \\ \Sigma\lambda &= 2 \\ k &= 0.44 \end{aligned} \tag{4.13}$$

Results in Eq. 4.13 imply that Deriche's edge detection operator outperforms Canny's operator. Additionally, this approach is simpler and involves only one parameter, α, with which the accuracy of edge localization can be controlled in favor of higher SNR, or better edge detection.

Deriche implements this operator using recursive filtering, for drastically decreased computational requirements over conventional approach [81, 83, 84].

4.2.2 Integration into Custom MRI Reconstruction Software

C code for Deriche's recursive edge detection algorithm [83, 84] is obtained from Project Epidaure's website (`http://www.sop-inria.fr/epidaure/software/basics/edges.php`) (currently known as Project ASCLEPIOS) INRIA, Sophia-Antipolis, France. The code is modified and ported to C++, for easier integration. A *C++* class, *EdgeDetector*, that can perform edge detection per device channel data is developed, and integrated into our custom MRI reconstruction software [62].

An instance of the *EdgeDetector* class is created once, and the same instance is used during the whole execution, for optimized memory and resource management. *EdgeDetector* permits enabling/disabling edge detection on the fly during real-time imaging. Additionally, selective edge detection can be applied; such that only selected channel data will be fed to the edge detector. Selective edge detection feature is represented in Figure 4.5. It is also possible to adjust the filter parameter α dynamically on the fly, for increased adaptivity.

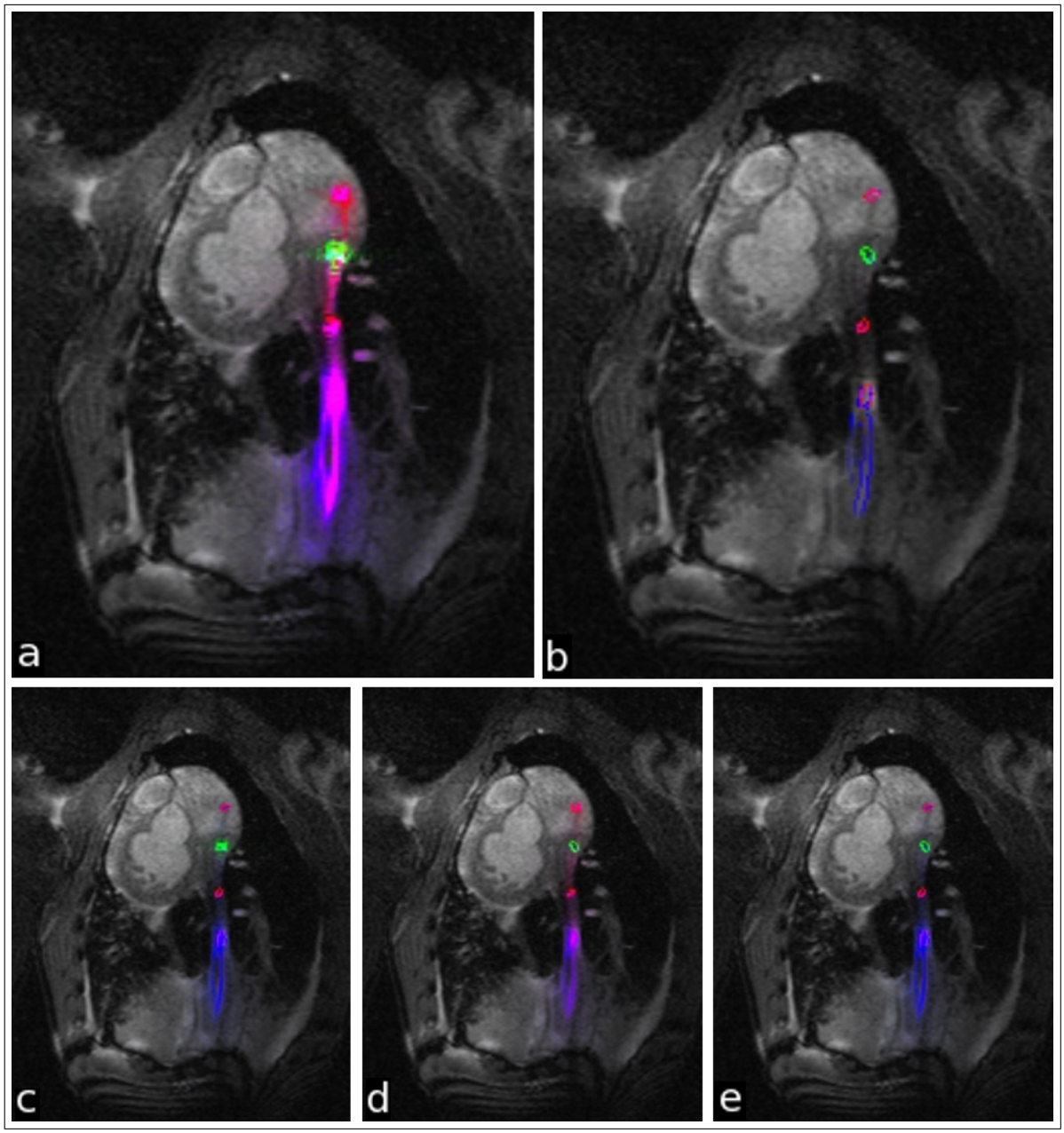

Figure 4.5 Images from pig experiment with three channel guiding catheter that shows selective edge detection feature. a) edge detection is disabled. b) edge detection is enabled on all channels. c) edge detection is enabled only on tip channel. d) edge detection enabled only on middle channel. e) edge detection enabled on tip and on the middle channel, and disabled on shaft channel.

4.2.3 Results

Real-time images are acquired both *in vitro* and *in vivo* in order to validate our software. Animal imaging protocols for pigs are approved by the NHLBI Animal Care and Use Committee. Phantom and animal experiments are performed in a short, wide bore Siemens Espree 1.5 T MRI scanner (Siemens Medical Solutions, Erlangen, Germany). A SSFP sequence with TR=3 ms, matrix size of 192×108 was used during real-time imaging. Acceleration factors of R=$1, 2, 3$ are used. Images are reconstructed on our custom 8 dual-core AMD Opteron (8220) reconstruction workstation.

Two active devices [87] are used during the experiments. During phantom imaging, a 0.035" active guidewire with two separate channels is used. A three channel 7 Fr guiding catheter is used during real-time *in vivo* experiment. Edge detection module is applied during both experiments. Images with and without edge detection are collected, and compared.

Figure 4.5 and Figure 4.6 represent a 0.035" active guidewire with two separate channels in a phantom without and with edge detection. Figure 4.7 represents *in vivo* results of 7 Fr guiding catheter from a pig experiment, with and without edge detection on device channels. During *in vivo* vascular interventions, false edges are occasionally detected on the shaft channel due to motion (e.g. cardiac and respiratory). In that case, edge detection on the channel in question can be interactively disabled. Edge detection on the distal tip channels works without failure during our experiments, and greatly improves visualization of anatomical structures in the presence of active devices. Additionally, edge detection helps to remove device coupling related image artifacts between individual device channels, and hence provides cleaner images.

Edge detection performances are measured and analyzed. In average, edge detection on a given channel takes 6 ms. Therefore, $6-18$ ms are required to detect edges depending on the number of independent device channels. Imaging speed is unaffected up to acceleration rate 3. Imaging lag is experienced on higher rates.

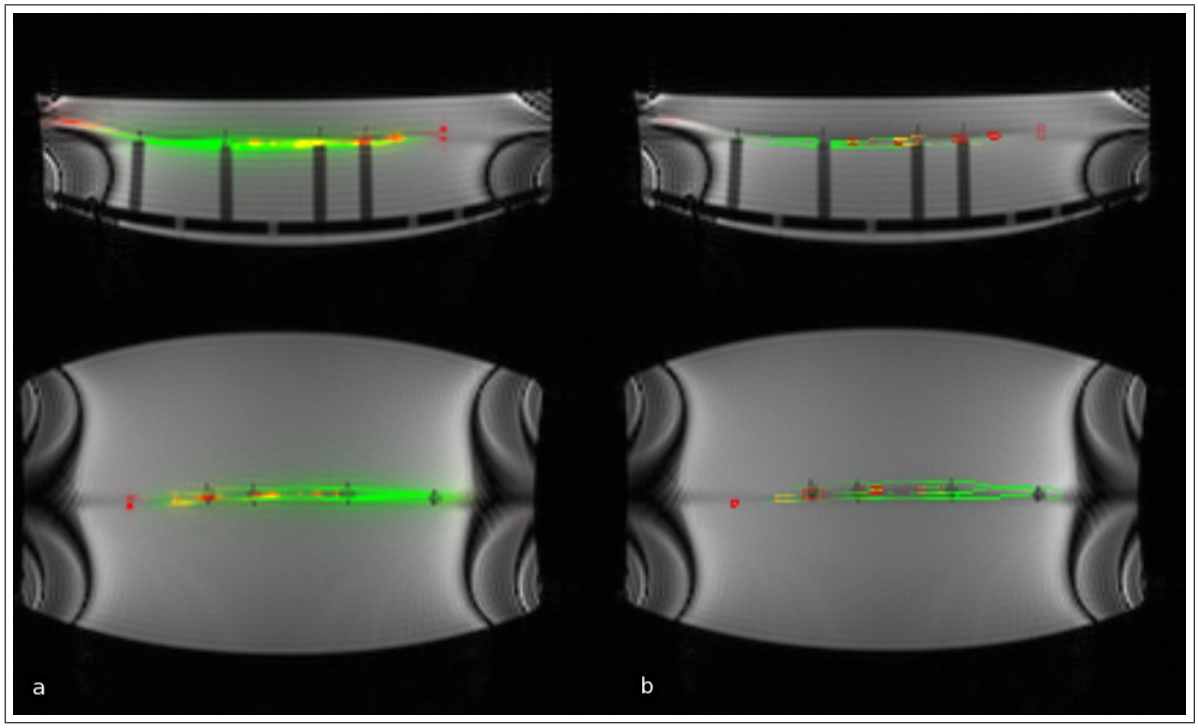

Figure 4.6 Phantom images with active device. a) Without edge detection b) With edge detection.

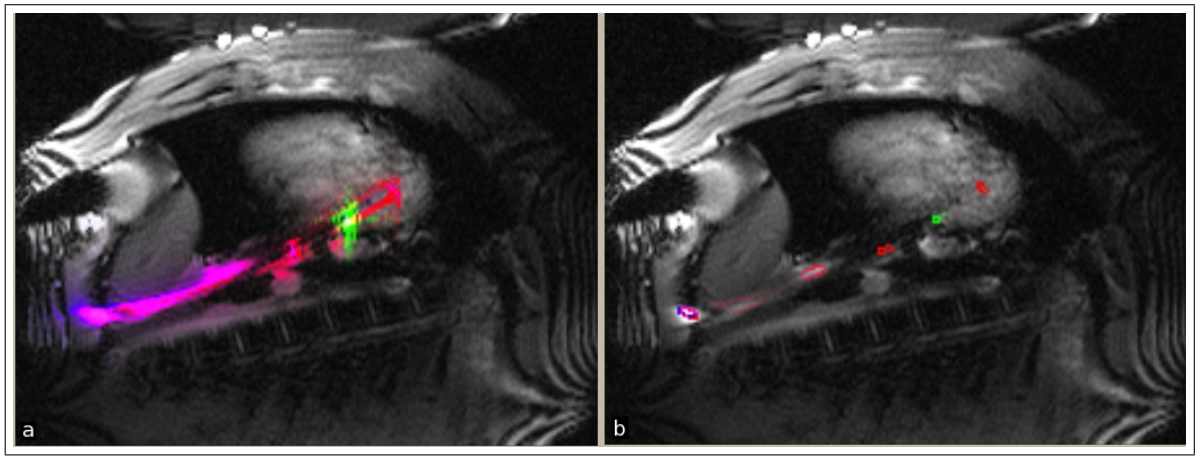

Figure 4.7 In vivo pig images acquired during real-time MRI guided vascular intervention. a) Without edge detection b) With edge detection.

4.2.4 Discussion

Real-time edge detection on device channels is successfully employed during both *in vitro* and *in vivo* experiments. It helps recover obscurity caused by "colored cloud" that represents device channels which is occurring when pseudo-coloring method is used, and improved the visualization of the anatomical context. Furthermore, it helps decreasing device channel coupling artifacts that is normally occurring during active device enabled vascular interventions.

Deriche's recursive edge detector permits controlling the accuracy of the localization using a single parameter, α [83]. Decreasing α lowers the the accuracy of edge localization, and *vice versa*. Increased localization accuracy also increases detection of false edges. Figure 4.8 represents results of edge localization with different α values ($\alpha \in \{0.5, 0.8, 1.0, 1.3, 1.5, 1.7\}$) on a phantom data. $\alpha = 1.3$ gives the best results during our experiments.

Edge detector on device shaft channel may detect false edges when the device undergoes through excessive motion (e.g. near the heart). This is due to the perturbation in the received signal by motion. In such a case, edge detection on the device channel in question can be interactively disabled, and the default pseudo-colored visualization scheme can be used instead. Thresholding with hysteresis [82] is a method that can improve the elimination of false edges. However, this method requires to select two threshold values (low and high thresholds). As all the methods involving thresholding, the choice of threshold values are non-trivial. Considering the fact that the image intensity and signal content changes from frame to frame (minimum 3 frames per seconds are acquired by default), the probability that one set of thresholds will not fit to consecutive set of frames is really high. Additionally, the signal content acquired using device channels is really low, and further diminishes according to the distance from the device. Hence, we considered thresholding impractical in that case. Note that, edge detection on tip channel never failed during *in vivo* animal experiments.

Curve fitting can be employed to improve device shaft visualization. Shaft device

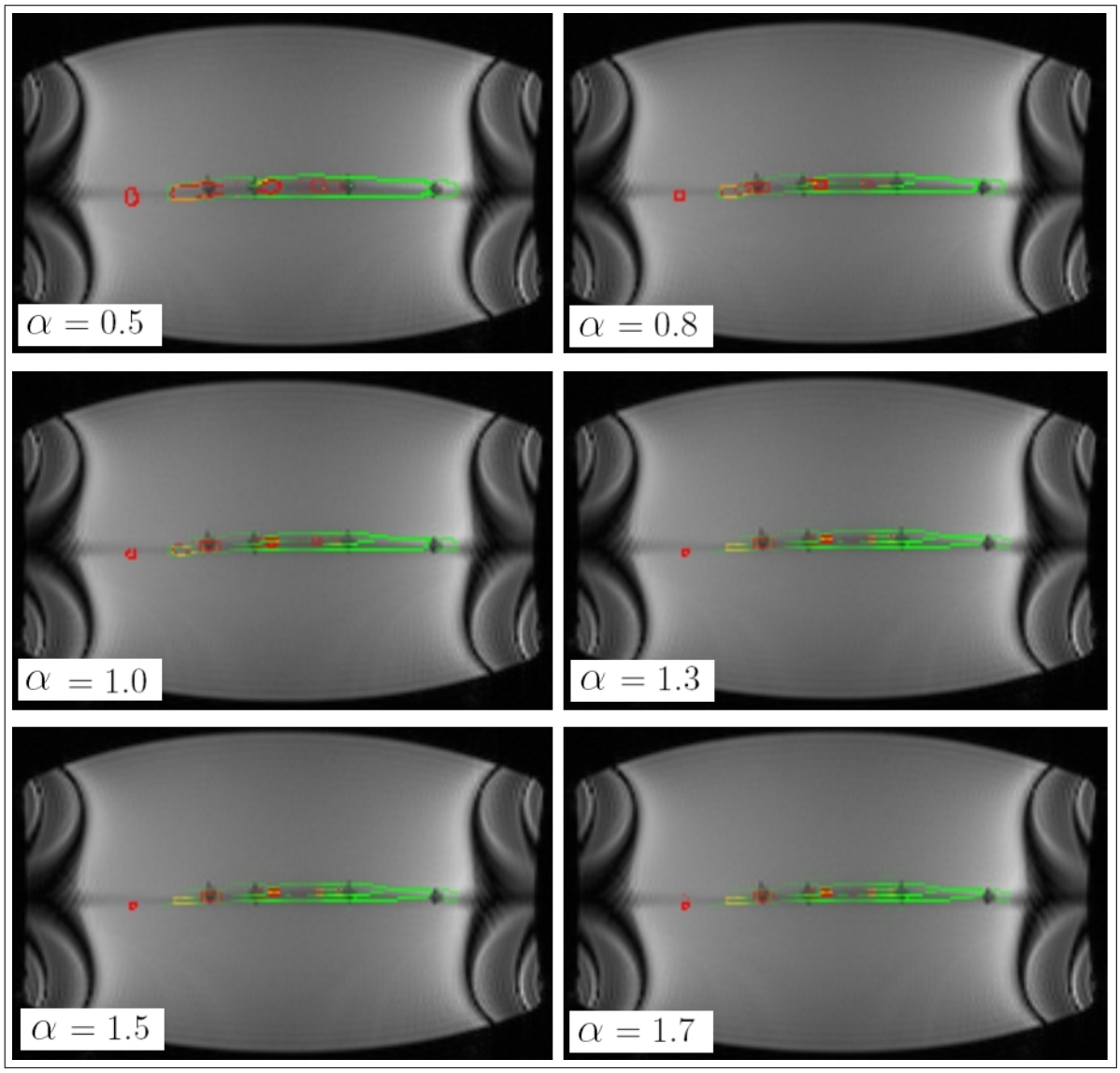

Figure 4.8 Edge localization for different α values.

data can be thresholded to eliminate noise and weak signals. Following thresholding, some morphological operators [88] , e.g. "thinning" can be applied to get only the middle portion of the device, as a one pixel wide line. Using this line, a curve that will represent the shaft can be fitted, and displayed in conjunction with detected edges on other channels. Note that, synthetic representation by thresholding may cause inaccurate fitting. If desired, all device channel data can be combined and "thinning" operation can be applied to obtain a curve that fit the whole device. If desired, whole device curve and detected edges on the tip channel can be combined, and displayed together for improved visualization.

Integrated edge detection feature is able to detect edges on selected device channels (up to three channel active device) during accelerated real-time MRI, up to rate 3. Higher rates are not possible to achieve when edge detection is enabled, as our system could not keep up with the acquisition rate. However, device channel data is really low in terms of SNR, especially when acceleration rate is greater than 2, so little is gained by enabling edge detection on images with high noise content. Additionally, during accelerated imaging view sharing method is used on device channel data, as pMRI techniques could not be applied. View sharing adds additional artifacts, such as aliasing, and blurring due to motion. Therefore, rates higher than 2 are considered to be non-practical for a good device visualization.

4.3 Real-Time Device Localization Software Integration

Active device visualization during real-time MRI guided interventions by assigning pseudo-colors to each individual device channels is previously described [53]. While this method provides drastically improved visual feedback, colored cloud representation of the active device may obscure surrounding anatomical context. In section 4.2, an improved visualization scheme that employs edge detection is presented. That way, obscurity caused by the pseudo-colored active device signal is eliminated, and improved anatomical context is obtained.

Even though pseudo-coloring and edge detection helps to improve visual feedback, if the *2D* slice positioning is not carefully selected - such as to show the tip and the shaft of the device, device visualization is not possible. This scenario happens often during interventions, as scan plane may be continuously changed to visualize target tissue or structure. As a result, a blind trial-and-error may be required to find the best *2D* slices for optimal device visualization. Projection imaging on the device channels by turning off the slice selection gradient facilitates to locate the best slice positioning.

An automatic device localization and slice repositioning algorithm, SnapTo, is previously presented in [61]. This algorithm basically works by *3D* parametrization of an active device using MR projection images. Three orthogonal projection images are acquired to estimate the location of the device. This algorithm is implemented in Matlab (`http://www.mathworks.com`).

This Matlab prototype is converted to *C++* to permit real-time custom reconstruction software integration, and tested during both *in vivo* and *in vitro* experiments. In the following sections, SnapTo algorithm will be briefly explained, and the integration details will be given. After presenting the experimental results, the advantages and the drawbacks of this method will be discussed.

4.3.1 SnapTo

SnapTo [61] automatically locates the active device by estimating the *3D* trajectory of the curve using three perpendicular projection images of the active device channels. This step involves a predictor-corrector iterative method. Subsequent to estimating *3D* coordinates of the device, a curve is fitted to the device, for a better representation of its *3D* location.

k-space data per projection is acquired from device channels, and projection images are reconstructed using inverse Fourier transform. Let $I_i(x, y)$ designate the image for projection i, where $i \in \{1, 2, 3\}$. n^{th} power of each image i_i is calculated

to increase the intensity gap between strong and weak pixel values[3] . That way, the sparsity of the images is improved. I_i' being the nth power of I_i, the *3D* volume approximation for the device is then calculated by

$$V_{3D}(x, y, z) = \prod_{i=1}^{3} I_i'(x, y). \tag{4.14}$$

If we consider a *2D* plane in the approximated *3D* volume that is perpendicular to the device curve, local maximums of this *2D* plane correspond to the coordinates at which the *2D* plane is penetrated by the device. A predictor-corrector iterative algorithm is employed to calculate the device curve estimation.

The iteration starts using the point p_0, which is the voxel of maximum magnitude in the *3D* volume. The direction of the curve d_0 at the initial point p_0 is calculated using principal eigenvector of the moment-of-inertia tensor of the local volume around p_0. Let assume that the iterator previously calculated the point p_i. The next point p_{i+1} is calculated as follows:

1. calculate the predicted point $q = p_i + sd_i$
2. find the *2D* slice $S(x, y)$, perpendicular to the device curve, centered at point q
3. find the centroid (x^*, y^*) of the slice S
4. calculate point p_{i+1} by correcting q by the value of the centroid
5. calculate the curve direction for point p_{i+1}

s being the step-size of the iterator. Iteration stops when one of the ending criteria are reached: (i) the magnitude of the point falls below a threshold when projected onto a perpendicular plane, or (ii) if the curve loops over itself. When the iterator stops, device coordinate estimates are obtained. The *2D* slice that best fits the calculated device curve coordinates in a least-square sense provides us the best plane to visualize the device.

[3]Generally $n = 4$.

4.3.2 Integration into Custom MRI Reconstruction Software

SnapTo algorithm is initially implemented in Matlab. It is converted to *C* or *C++* to be integrated into our custom real-time MRI reconstruction software. *C++* is preferred over *C* for its improved encapsulation, and transparency. ACML library and ATLAS library are used for matrix operations.

A posix thread dedicated to SnapTo calculation is created during the initialization, and all the necessary variables are created. That way, SnapTo is performed asynchronously to the real-time image reconstruction. Real-time imaging is paused only while three SnapTo projections are acquired. Imaging resumes promptly after SnapTo data acquisition. Otherwise, imaging will be delayed while SnapTo calculation is in progress. Creation of all the variables during the initialization permits to eliminate unnecessary context-switching that will decrease execution performance. A special keyboard key is designated to initiate SnapTo calculation. By pressing the key,

1. SnapTo thread is waked up
2. a message is sent to the sequence to acquire SnapTo data
3. SnapTo data is acquired, and sent to the reconstruction workstation
4. imaging automatically resumes, while device coordinates are calculated asynchronously at the background
5. when calculation is finished:
   (a) estimated *3D* curve of the device is drawn on the *3D* display
   (b) the best *2D* plane containing the device is estimated
   (c) if requested, selected slice is prescribed to this *2D* plane for best view of the device, by communicating with the sequence

and SnapTo thread is sent to sleep until a new request is received. For debugging purposes, images from SnapTo data set are displayed on separate *2D* display windows.

A network socket communication channel is used as a communication link between the reconstruction workstation and the scanner.

SnapTo requires additional data acquisitions to collect three perpendicular slices. However, device localization must be realized without stopping the current imaging protocol, as time is crucial during an intervention. Additionally, stopping the actual scan, localizing the device and restarting the scan would not meet the practicality requirements. Our asynchronous threaded scheme eliminated this drawback from the reconstruction side. Our real-time sequence is also modified to dynamically switch to SnapTo acquisition mode. Subsequent to SnapTo data acquisition, our sequence switches to normal real-time imaging, as done in the reconstruction side.

4.3.3 Results

SnapTo is tested on both phantom data and during a pig experiment. Animal protocols are approved by the institutional animal care and use committee in accordance with the National Institutes of Health (NIH) guidelines.

Data is acquired using short, wide bore Siemens Espree 1.5 T scanner (Siemens Medical Systems, Erlangen, Germany). A SSFP sequence with TR=3 ms is used to acquire real-time data. An aortic phantom is used (Shelly Medical Imaging Technologies, London, Canada) during the phantom experiments. A custom made single channel active guidewire is used during both experiments.

4.3.3.1 *In Vitro* Experiments. At the beginning of the experiment, a *2D* slice that is nearly perpendicular to the device is prescribed. By pressing the SnapTo keyboard key, three perpendicular projection images are acquired, and real-time imaging resumes. Device is estimated in each projection images, and displayed on the debugging windows. Subsequent to the estimation of the best *2D* slice that covered the active device, current imaging slice is prescribed to the new coordinates calculated by SnapTo.

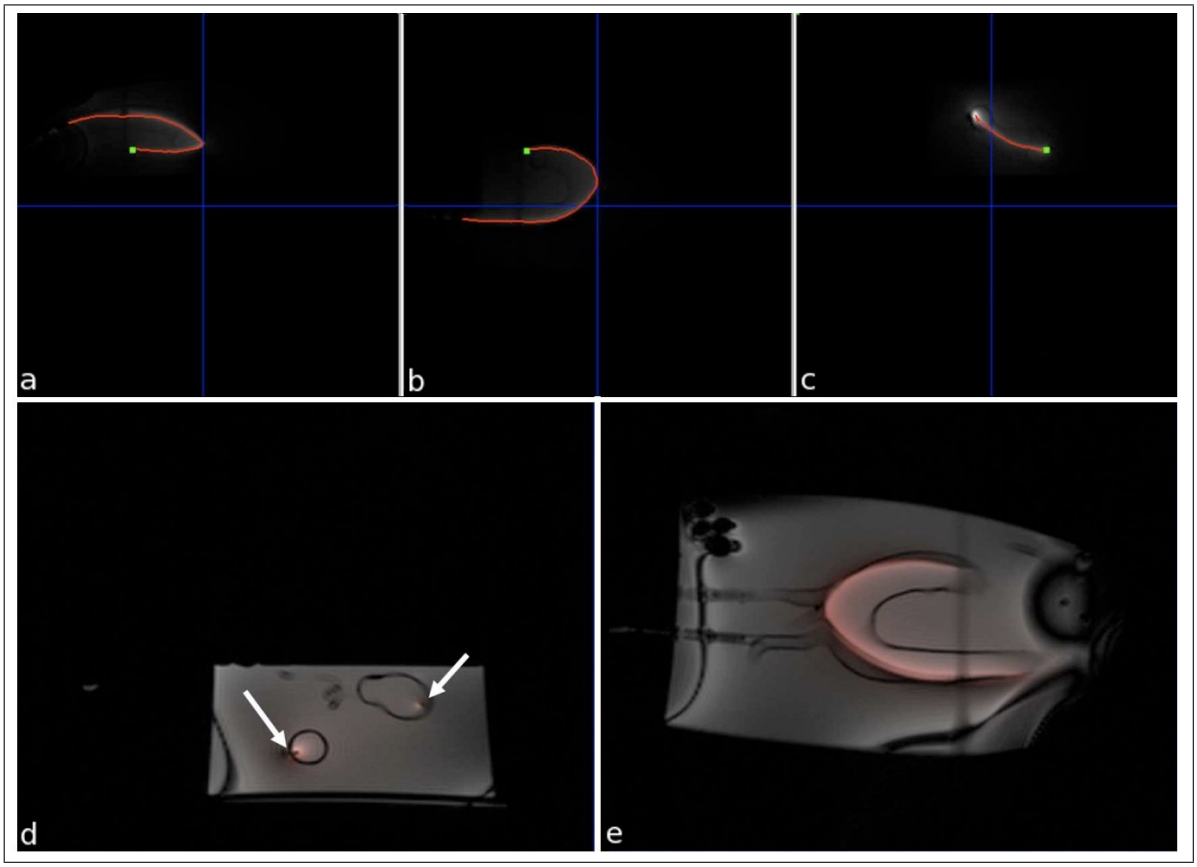

Figure 4.9 a), b) and c) represent three projection images blended with estimated device trajectories. d) Initial imaging plane plane before executing SnapTo is given. The arrows represents the intersection of the device with the imaging plane in two places. e) Imaging plane after SnapTo execution is given. SnapTo successfully estimates the best *2D* plane that covers the device, and a new imaging plane is prescribed.

Figure 4.9 represents debugging windows per projection at the top, and imaging slice before and after the localization at the bottom. SnapTo successfully estimates the coordinates of the optimum *2D* slice, and a new slice plane is prescribed accordingly.

4.3.3.2 *In Vivo* Experiments. During a real-time MRI guided pig experiment, an imaging plane that does not cover the device is selected. Then, SnapTo is initiated, and three perpendicular projections are acquired accordingly. Following the estimation of the device location, its *3D* representation is drawn, and a new slice plane is prescribed for optimum view of the active device. Figure 4.10 represents the experimental results during the pig experiment. SnapTo successfully estimates the best *2D* plane that

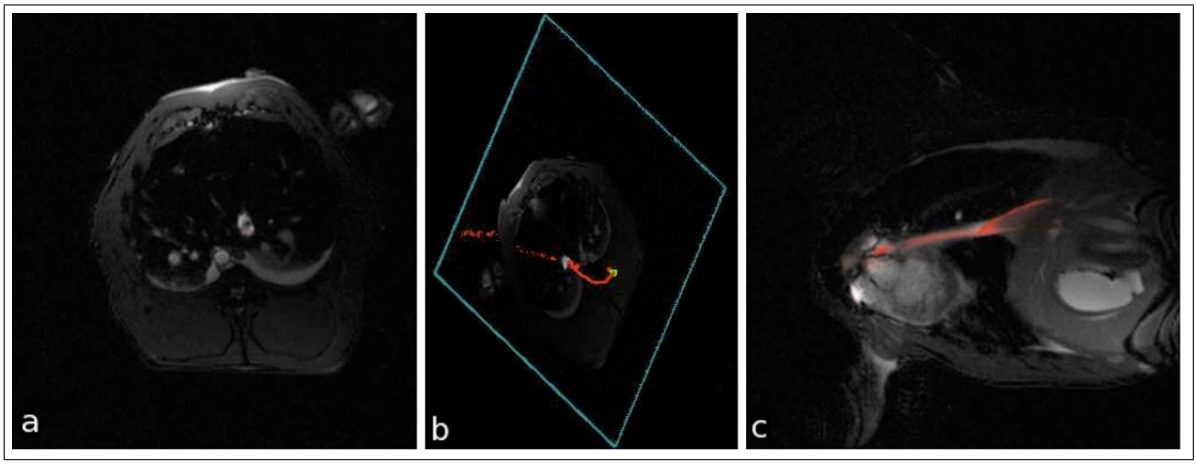

Figure 4.10 a), b) and c) represent three projection images blended with estimated device trajectories. d) Initial imaging plane plane before executing SnapTo is given. The arrows represents the intersection of the device with the imaging plane in two places. e) Imaging plane after SnapTo execution is given. SnapTo successfully estimated the best *2D* plane that covers the device, and a new imaging plane was prescribed.

covered the device. Subsequently, estimated *3D* trajectory of the active device is drawn on our *3D* display windows, and a new imaging plane is prescribed accordingly.

4.3.4 Discussion

An automatic active device localization software is integrated into our custom real-time MRI reconstruction software. Only three perpendicular projections are required in order to estimate the device coordinates. Following the estimation, a *3D* trajectory estimate of the active device can be drawn on our *3D* display window blended with real-time reconstructed multi-slice images. Unlike previous work [59], this method does not require any knowledge about the shape of the device, nor requires any geometric orientation. Furthermore, this method can be also used when multiple active devices are present. As each device has its own receiving channel, they can be identified correctly.

A dedicated thread is created once, and the same variables are used over and over again. That way, unnecessary context switching is avoided. Furthermore, asynchronous

execution made it possible to resume real-time imaging promptly following the SnapTo data acquisition. Our sequence is modified for the data acquisition. Currently, slightly bigger FOVs (around 400mm) are required. However, further improvement in the sequence at the estimation side may easily overcome this drawback.

Three perpendicular k-space data set are acquired in less than a second. Following the data acquisition and reconstruction of the projection images, approximately 250 ms is required to estimate the coordinates of the device, and the best *2D* scan plane that will cover the active device. This method may be used to automatically track the device during whole intervention. However, this will require further optimizations and parallelizations on the SnapTo algorithm, as 250 ms is still long enough to provide practical continuous device tracking.

During experiments, the estimation of active device coordinates occasionally fails when the device is looped over itself. In such a case, the iteration can be restarted by modifying the stopping conditions. As with all iterative algorithms, the choice of stopping conditions has a great effect on the accuracy of the results. It would be better to automatically decide on stopping conditions. Additionally, wrong estimation of the active device tip location occurs occasionally. When this happens, the direction of the tip can be interactively reversed, and iteration can be restarted.

Currently, SnapTo acquisition does not support accelerated imaging, and data acquisition is not limited to only device channels. All coil projection data is acquired, and sent to our custom reconstruction workstation over network. If only device channel data can be acquired, or sent over network, faster, low-latency data acquisition will be possible. Furthermore, if any accelerated imaging is possible, data acquisition will be even faster, and more practical, seamless localization will be possible. View-sharing method can help to improve data acquisition performance until a better accelerated imaging method is incorporated into SnapTo algorithm. If these drawbacks can be addressed, parallelization and further optimizations can make it possible to evolve this algorithm for automatic continuous device tracking during real-time MRI guided interventions.

4.4 Conclusion

Fast imaging using pMRI techniques and efficient real-time sequences make it possible to perform MRI guided interventional procedures. *2D* slices with decreased spatial resolutions are generally acquired during MRI guidance for improved image update speeds. As a result, visualization of the structures outside the imaging plane, such as different portions of or different vascular trees, and active devices becomes a difficult task. Assigning pseudo colors to each individual device channels improves device visualization, but the "colored cloud" that represents the device may obscure surrounding anatomical structures. Additionally, active device localization is a huge challenge during real-time MRI guided interventions, and is often performed by the operator using some trial-and-error methods, such as changing slice positioning blindly, or by using the guidance of projection imaging.

Contrast-enhanced MR angiography offers excellent depiction of vascular structures but cannot be performed in real-time and repeated at will. However, previously acquired MRA or MRCP data sets can be post-processed in order to generate *3D* colorized roadmaps. Blended display of these *3D* colorized roadmaps with *2D* slice images reconstructed in real-time, and with colorized active device representation drastically improves the anatomical context and relative positioning of the devices with the structures of interest. We demonstrate a new scheme, integrated interactive colorized MIP display of multiple angiographic data sets, that allows *3D* visualization of vascular structures during real-time MRI guided procedures. The utilization of the roadmaps improves the visualization of invasive devices through and between complex vascular spaces, with the potential of improving patient safety, procedure easiness and efficacy. Our software is employed during several MRI-guided applications, such as pulmonary thrombectomy and TIPS, and was proved to ameliorate visual feedback. Our method can also be utilized during angioplasty, biopsy, stent placement, and electrophysiology procedures. Furthermore, previously acquired CT data, or similar volumetric data from any other imaging modality (e.g. *3D* ultrasound), can also be used to generate similar colorized *3D* MIPs for deployment during real-time MRI guided interventions after an initial registration process. Static roadmaps have some inherent limitations: (i)

they may become outdated, (ii) respiratory, cardiac or other patient motion can cause misalignments with real-time images. The effect of the first drawback can be minimized, if the angiographic data is acquired just before the intervention. For the latter, breath-holding, ECG-gating or other motion compensation techniques, or periodic intraprocedural updates might enhance the accuracy of the static MIPs. Additionally, if a 4D angiographic data are acquired, time variant roadmaps may be generated. However, these dynamic roadmaps may require further processing for keeping up with the imaging rate.

Active device visualization by assigning pseudo colors to each device channel has been described previously [53]. While this method improves the visualization of devices, this "colored cloud" may occasionally obscure underlying anatomical structures (e.g. mitral valve, aortic valve structures); this is critical especially if the device diameter is 2 mm or larger. We integrate Deriche's edge detection algorithm to our real-time MRI environment to show only the top ridge or specific edges of the active device signal. Our implementation permits selective edge detection on desired device channels. That way, the obscurity around the device is removed, and visual feedback is drastically improved. Additionally, artifacts due to device coupling is lowered. Deriche's edge detector permitted controlling edge localization accuracy by using one parameter, σ. $\sigma = 1.3$ is considered the best compromise between localization accuracy and low fast edge detection. Edge detection performance is measured as 6 ms per device channel. Our system can keep up with the acquisition up to acceleration rate of 3 while edge detection is enabled. However, when acceleration rate exceeds 2, increased noise, and additional blurring artifacts introduced by view-sharing methods make the edge detection feature impractical. During the experiments, false edge detection on device shaft channel increases significantly, when the device undergoes through excessive motion (e.g. near the heart). In such a case, edge detection on that channel can be interactively disabled, and the default pseudo-colored visualization scheme can be used instead. Thresholding with hysteresis is a method that can improve the elimination of false edges. However, this method requires to select two threshold values (low and high thresholds). Image intensity and signal content changes from frame to frame. Therefore the probability that one set of thresholds will fit consecutive set of frames is

really low, and excessive information loss can happen. Hence, we consider thresholding impractical in that case. Note that, edge detection on tip channel never failed during *in vivo* animal experiments. Another improvement can be fitting a curve to the device shaft and display it fused with edges of the tip channel to improve visualization. If desired, all device channel data can be combined and a curve can be fit to whole device. This however, will require additional processing, as combined data from all channels may contain spatial discontinuities.

Dynamic scan plane changes occur often during real-time MRI guided intervention, to better visualize the anatomical context of interest, and active devices in use. Sometimes the selected *2D* slice plane does not properly cover active devices, and thus active devices cannot be visualized as desired. Some blind trial-and-error methods, or projection imaging on device channels [53] is used to locate the device. These methods require some time until the best *2D* plane is chosen. An automatic device localization and slice repositioning algorithm, SnapTo, previously implemented in Matlab (`http://www.mathworks.com`), is converted to *C++* and integrated into our custom MRI reconstruction software. A dedicated thread is created once, and the same variables are used over and over again. That way, unnecessary context switching is avoided. Furthermore, asynchronous execution makes it possible to resume real-time imaging promptly following the SnapTo data acquisition. Additionally, our sequence is modified and optimized to acquire SnapTo data set without interrupting real-time imaging. Three projections are required to estimate the device coordinates. Following the estimation, a *3D* trajectory estimate of the active device can be drawn on our *3D* display. The best *2D* slice plane is calculated using estimated coordinates, and a new imaging plane is automatically prescribed to that location. This method does not require any knowledge about the shape of the device, nor require any geometric orientation. Furthermore, this method can also be used when multiple active devices are present. Following drawbacks are experienced during experiments: (i) the estimation may fail if the the device is looped over itself, (ii) the orientation of the device tip might be occasionally estimated wrongly, (iii) SnapTo does not support accelerated imaging, (iv) when SnapTo data is acquired, acquired data for all coils are actually sent over network to the custom reconstruction workstation. For the first case, re-execution of

the iteration with modified stopping conditions may help. For the second, the orientation of the estimated top position may be interactively reversed. View-sharing method can be incorporated for limited accelerated imaging support in short-term. Long-term plans may involve a better accelerated imaging method to improve data acquisition performance. Furthermore, if only device channel data can be sent over network, data transfer latency will be drastically lowered, considering the fact that up to 32 receiver coils in total can be used in our current scanner. By addressing all these drawbacks, and by parallelization and further optimization of SnapTo algorithm, it will be possible to evolve this algorithm for automatic continuous device tracking during real-time MRI guided interventions.

Visual feedback during real-time MRI guided interventions is improved: (i) a new software module to create interactive multi-phase, colorized *3D* angiography roadmaps using previously acquired MRA/MRCP data sets; (ii) Deriche's recursive edge detection algorithm integrated for real-time selective edge detection on multiple device channel data sets; and (iii) an active device localization software integrated into our custom MRI reconstruction environment.

5. CONCLUSIONS AND DISCUSSIONS

The objective of the work described in this thesis is to develop and implement highly parallel, real-time low-latency parallel MR and non-Cartesian imaging, and to improve the visual feedback during real-time MRI guided interventions. To this end, following advanced software tools have been developed:

- Real-time MRI:
    1. HTGRAPPA: Real-time B1-weighted image domain TGRAPPA reconstruction [42]
    2. RT-GROG: Parallelized self-calibrating GROG for real-time MRI
- Improved visual feedback during real-time MRI guided interventions:
    1. Multi-Phase *3D* Angiography Roadmaps for Real-Time MRI-Guided Procedures [55]
    2. Real-time edge detection for improved device visualization during RT-MRI guided interventions [62]
    3. Real-time implementation of an active device localization software

Methods developed during the thesis are integrated into a custom MRI reconstruction software [52, 53], and tested during real-time MRI guided interventional procedures. HTGRAPPA algorithm is also implemented by Siemens (Siemens Medical Solutions, Erlangen, Germany) to run on conventional MR environment (currently for off-line reconstruction only).

5.1 Discussions

5.1.1 Real-time parallel MRI

HTGRAPPA is highly parallel, hybrid TGRAPPA implementation for low latency, real-time imaging algorithm where weight calculation and reconstruction processes were decoupled and individually parallelized. For improved reconstruction speeds, k-space domain convolution type operations are converted to pixel-wise multiplication in the image domain after proper transformation. Additionally, noise weighting is applied to HTGRAPPA weights for improved SNR on the reconstructed images. HTGRAPPA provides a constant reconstruction speed for a given acceleration rate and block size. The weight sets are calculated in the background, asynchronous to image reconstruction. Furthermore, our implementation can respond to interactive changes in slice plane by first producing view-shared images, then resuming HTGRAPPA once new weights have been calculated. That way, low latency, uninterrupted interactive real-time MRI is permitted. Up to 120 frames per second reconstruction performance is achieved. HTGRAPPA is nearly 180 times faster than conventional TGRAPPA when reconstructing rate 4 data set. Additionally, block size and number of ACS lines can be dynamically selected regarding the imaging needs. HTGRAPPA does not exhibit prefolding artifacts when small field of views are used, and it is totally auto-calibrated.

5.1.2 Real-time non-Cartesian imaging

Highly parallelized RT-GROG permits multi-slice radial imaging scheme for real-time MRI. The weight-set calculation and the image reconstruction steps are decoupled to run asynchronously, and parallelized to run in real-time on a general purpose multi-core architecture. While weight calculation threads continuously update the weights in the background, the reconstruction threads are using the most recent weights to grid and reconstruct images. Per sample gridding weight calculation step is eliminated from the reconstruction process by using a LUT. In this way, reconstruction performance is drastically improved. Our implementation is auto-calibrated, and

supports multiple slice acquisitions and there are no parameters. All the necessary information is extracted from the acquired data itself, rather than using some a priori information. Highly parallel architecture of our implementation overcomes slice plane changes during real-time MRI. RT-GROG provides up to 34-fold reconstruction performance increase over conventional SC-GROG during the reconstruction of the 32 coil cardiac data set. Reconstruction is always faster than the data acquisition with RT-GROG.

5.1.3 Improving visual feedback during real-time MRI guided interventions

Visual feedback improvement contributions are three-fold: (i) multi-phase *3D* MR angiography roadmaps for MRI guided interventions [55], (ii) improved active device visualization during real-time MRI guided interventions [62], (iii) integration of a real-time active device localizer algorithm.

5.1.3.1 *3D* Colorized MRA/MRCP Roadmaps. Contrast-enhanced MR angiography provides and excellent depiction of vascular structures, but they cannot be performed in real-time and repeated at will. MR angiography is performed prior to the intervention, and acquired data is loaded into our custom MRI reconstruction software as DICOM image series. These image series are then dynamically rendered as multiple colorized *3D* Maximum Intensity Projections (MIPs) in real-time. These colorized *3D* roadmaps are then displayed in conjunction with multiple slice real-time MR images using alpha-blending [55]. This new software improves the visualization of invasive devices through and between complex vascular spaces, and has the potential of improving patient safety and procedure easiness and efficacy. Our software is employed during several MRI-guided applications, such as pulmonary thrombectomy and TIPS to ameliorate visual feedback. Our method can also be utilized during angioplasty, biopsy, stent placement, and electrophysiology procedures. Furthermore, previously acquired Computerized Tomography (CT) data, or similar volumetric data from any

other imaging modality (e.g. *3D* ultrasound), can also be used to generate similar colorized *3D* MIPs for deployment during real-time MRI guided interventions after an initial registration process.

5.1.3.2 Improved Active Device Visualization. Active device visualization during real-time MRI guided interventions is possible by assigning unique pseudo-colors to each active device channels [53]. In this method, active device data is reconstructed separately and per channel grayscale sensitivity maps are obtained. Subsequent to that, these sensitivity maps are colorized, and displayed in conjunction with real-time reconstructed multi-slice images. Even though visual context is drastically improved, colored cloud that represents the active device may occasionally obscure the underlying anatomical structures. In order to avoid this drawback and to improve anatomical context, real-time edge detection module is integrated into our custom reconstruction software to show only the top ridge or specific edges of the active device signal. Deriche's edge detection algorithm is preferred for its computational efficiency. Our implementation permits selective edge detection on desired device channels. That way, the obscurity around the device is removed, and visual feedback is drastically improved. Additionally, artifacts due to device coupling is lowered.

5.1.3.3 Real-time Active Device Localization. Real-time active device localization is a real challenge during MRI guided interventions. SnapTo algorithm is integrated as a separate thread.Asynchronous execution makes it possible to resume real-time imaging promptly following the SnapTo data acquisition. Additionally, our sequence is modified and optimized to acquire SnapTo data set without interrupting real-time imaging. Three projections are required to estimate the device coordinates. Following the estimation, a *3D* trajectory estimate of the active device can be drawn on our *3D* display. The best *2D* slice plane is calculated using estimated coordinates, and a new imaging plane is automatically prescribed to that location. This method does not require any knowledge about the shape of the device, nor any geometric orientation. Furthermore, this method can also be used when multiple active devices are

present.

5.2 Future Improvements and Possible Extensions

In this section, first, the drawbacks of proposed methods or possible improvements for them will be discussed. Subsequent to that, suitable, novel applications and novel fields related with the thesis will be identified.

5.2.1 Future improvements

The benefits and potential drawbacks of the methods proposed in this thesis are analyzed in detail in the previous Chapters. Overall summary of thesis contributions and possible further improvements for each will be given in this section.

5.2.1.1 HTGRAPPA. HTGRAPPA provides up to 180x faster reconstruction performances compared to TGRAPPA. However, weight-set calculation is slowed down around 0.6 seconds. Considering the fact that one set of fixed unmixing coefficients can be used to reconstruct multiple consecutive frames [42], the decrease in weight-set update rate is acceptable during real-time MRI guided interventions. Additionally, it is shown that our three year old 8x dual-core workstation is not fast enough to update the unmixing coefficients in a timely manner, when 32 coil acquisitions and big GRAPPA kernel sizes are involved (see Figure 2.17). A modern similar multi-core architecture can provide further increased reconstruction and weight-set update performances, and thus none of these drawbacks will be experienced. It is also possible to implement decoupled weight-set update process on different parallel architectures. This idea will be more generally discussed in section 5.2.2.2.

Currently available regularization schemes for GRAPPA algorithm give imaging slice plane dependent results, and image quality highly depends on the regularization

parameter [89]. Qu *et al.* propose an improved discrepancy-based regularization scheme in [89]. However, their method is only evaluated on rate two data sets, and requires the estimation of the error on the auto-calibration data set, which is non-trivial. Additionally, analyzed data sets are not appropriate for real-time MRI because of the large matrix size.Therefore, the extent of their work is somewhat limited to off-line, low temporal resolution imaging. Further analysis of this method for real-time imaging may provide new avenues for a better regularization scheme that suits real-time low-latency GRAPPA imaging, and may be incorporated into HTGRAPPA algorithm.

Miao *et al.* proposes an improved GRAPPA reconstruction scheme, where anisotropic GRAPPA kernels are used to reconstruct sub-sampled data sets [90]. They aim to improve the model that relates the acquired samples to the missing samples by using the the pattern of high power signals in the center of the k-space. To this end, they fit a bivariate Gaussian function to the auto-calibration data set magnitudes in order to capture the anisotropic pattern. Subsequent to that, they threshold this function and obtain the kernel elements. Their results are promising, however thresholding parameters may highly affect the results, and thus may be impractical during dynamic real-time imaging. If self-calibrated, GRAPPA reconstruction may benefit from their proposed approach. Another method to decrease the noise content during GRAPPA reconstruction is described by Guo *et al.* in [91]. However, their methods employ a "partial low-pass filtering-like" scheme on high noise content areas of the reconstructed images, and therefore introduces blurring and information loss.

5.2.1.2 RTGROG. Results show that RT-GROG provides 34-fold increase in reconstruction performance over SC-GROG when reconstructing 32 coil data sets. Gridding weights update speeds are reasonable, and image reconstruction is always faster than the data acquisition. If weight calculation is implemented in the Graphics Processing Units (GPU), it may be possible generate new gridding weights per frame, without slowing down the data acquisition. Additionally a recent multi-core workstation may greatly improve the performance results. Different parallel architectures may also help to increase reconstruction and weights update performances. This issue will

be discussed in section 5.2.2.2.

Currently, SC-GROG algorithm performs only gridding. Subsampled k-space data set are not considered during the reconstruction. View-sharing method can be used to improve spatial resolution, however this method is not considered because of the motion blur. Seiberlich *et al.* describes a new method that combined GROG with a pseudo-Cartesian GRAPPA algorithm to reconstruct undersampled data set in a more efficient way [79]. This method requires to determine the patterns in the gridded data set that will relate filled-up samples with the empty samples in the Cartesian grid. Pattern choice and pattern determination methods are non-trivial tasks, and increase the overall computational requirements for both reconstruction and weight calculation. A year later, another method, where the generalized sampling theorem is fused with GROG to reconstruct undersampled non-Cartesian data sets, is presented by Seiberlich *et al.* [80]. This method combines bunched phase encoding (BPE) [92], GROG, and an iterative conjugate gradients method to provide an improved reconstruction scheme for undersampled non-Cartesian data sets. Despite the promising results, this method is computationally expensive. However, more sophisticated parallel architectures and methodologies may permit real-time implementations in the near future (see section 5.2.2.2 for more details).

5.2.1.3 Visual Feedback Improvements During Interventions. A software module that generates *3D* colorized MRA/MRCP roadmaps is presented in section 4.1. The main drawback of this method is the static nature of the *3D* MIPs. Time variant, dynamic *3D* MIPs may be possible with a *4D* angiographic data acquisition. Or, a real-time registration algorithm may be implemented to dynamically reshape the MIPs according to the motion in the acquired frame. That way, a more dynamic feedback can be obtained. Additionally, even though our method is 100% based on MRI imaging, data from other imaging modalities (such as ultrasound, computerized tomography (CT)) can also be used to create these roadmaps after proper registration.

Section 4.2 describes the implementation and integration details of the real-

time edge detection software aimed to show only the top ridge or specific edges of an active device during real-time MRI guided interventions. While visual feedback is drastically improved, acceleration rates greater than two is not practical for device visualization. This is mainly due to the view-sharing reconstruction on device channel data sets. If multi-channel acquisition on active devices would be permitted, accurate pMRI reconstruction algorithms can be employed to reconstruct aliasing free images of active device data sets. Additionally, edge detector on shaft channel occasionally detects false edges due to signal perturbation caused by excessive motion (e.g. near the heart). In such a case, edge detector on the channel in question can be dynamically disabled, and default colored representation is used instead. Additionally, a curve may be fitted to the shaft channel data. That way, this channel can be represented by a thin line, instead of a colored cloud. However, curve fitting is non-trivial during real-time imaging due to unexpected signal fluctuations and motions. Inaccurate fitting may cause unreliable device representation, and thus can be misleading.

SnapTo algorithm integration is detailed in section 4.3. This method permits to avoid trial-and-error methods that are generally required to locate the device using projection imaging. Right now, SnapTo does not support accelerated imaging, which is also due to the one channel acquisition scheme of current active devices. Furthermore, even though SnapTo requires three projection data sets on only active devices, all coil data is sent over network to our reconstruction workstation. If data from imaging coils can be eliminated, SnapTo data acquisition performance can be greatly increased. SnapTo algorithm can also be employed to perform automatic device tracking in real-time. However, current implementation is serial, and is not fast enough to keep up with the real-time data acquisition. Thus proper parallelization (see section 5.2.2.2) will be required to increase calculation performance, and further improvements on sequence side will be necessary.

5.2.2 Conclusions

New MR imaging modalities, single-chip parallel architectures may provide new research avenues, therefore may extend the work described in this thesis. Faster data acquisition schemes can be implemented, and faster imaging speeds can be obtained. Following sections briefly depict these novelties.

5.2.2.1 Dynamic Imaging. In recent years, new dynamic imaging methodologies are proposed, such as *k-t* approaches [93–97], and *compressed sensing* [98–100]. Both methods provide fast data acquisition schemes.

$k-t$ methods combines the concept of transform coding with parallel MRI to provide faster data acqusition [93–97]. Increased acquisition speeds are achieved by employing redundant information in typical time series of acquired images. $k-t$ SENSE method requires an additional training data set obtained by low-resolution image series of dynamic object to be investigated. This calibration data set is used to constraint the reconstruction adaptively. A real-time low-latency implementation on GPU is described by Hansen *et al.* [73] in 2008. Huang *et al.* presents $k-t$ GRAPPA reconstruction scheme in 2005 [94], in which additional training data set acquisition required in $k-t$ SENSE method is eliminated. To the best of our knowledge, there is no real-time implementation for $k-t$ GRAPPA, due to convolution-based reconstruction scheme. It may be possible to implement it for low-latency real-time imaging as done with $k-t$ SENSE using novel parallel, high performance architectures.

Another newly emerging popular fast imaging method is compressed sensing [98–100]. The basic idea behind compressed sensing is that an undersampled signal can be reconstructed from its spectrum if: (i) the signal has a sparse representation in a transformed domain, (ii) undersampling based aliasing artifacts are incoherent in that transform domain, and (iii) a non-linear reconstruction scheme is used to enforce the sparsity of the image representation and consistency with acquired data. Compressed sensing is an iterative method, and computationally very demanding. To date, cur-

rently available computer hardwares are not fast enough to perform real-time imaging using compressed sensing reconstruction. New parallel architectures and programming models (see section 5.2.2.2), and effective programming may permit real-time low-latency sparse MR imaging.

5.2.2.2 New Trends in Parallelism. Physical limitations in parallel architectures lead to single-chip parallelism approach. As a result, new parallel programming challenges have been arising, and new programming models have been proposed that can further improve real-time MR imaging performance.

Cilk [101] is a multihreaded parallel programming language that extends *C* language that is available for more than a decade. Being highly effective for exploiting dynamic, highly asynchronous parallelism, Cilk environment may be alternatively deployed to implement the work represented in this thesis.

Several new single-chip parallel architectures have been proposed recently and gained huge interest, such as GPU [102–105], Cell Broadband Engine [106–110], and Explicit Multi-Threading (XMT) [111–116]. GPU architecture was widely deployed in MR imaging in the recent years [73, 77, 78, 117].

GPU architecture is application specific, and has its own programming model. GPU programming model is hard to program, and data partitioning is a challenge. Additionally, GPU has a very limited memory. Overall, it does not perfectly fit general purpose programming. However, it provides really high performance gains while decreasing the architectural cost, if programmed properly and if deployed on an appropriate application.

Cell architecture is composed of 8 threading units controlled by a CPU. It has high memory bandwith, however its memory technology is very expensive. Additionally, double ring interconnection network makes the scaling really hard. However, it is much faster than recent multi-core general purpose CPUs. It has also its own pro-

gramming model to perform general purpose programming.

XMT is a novel architecture, developed at the University of Maryland at College Park [111–116]. XMT is a general-purpose computer architecture aimed to reduce single task completion time. Its Parallel Random Access Model (PRAM) based programming environment permits easy application development. XMT is available as a field-programmable gate array based Paraleap prototype, and currently supports only fixed-point calculations. Preliminary results show that it provides up to 30-fold increased performance (in term of cycle count) when compared to state-of-the-art multicore architectures [116]. Easy programming, and high performance capabilities makes this architecture highly valuable for general purpose programming.

An hybrid approach can also be considered, where distributed systems (e.g. clusters of computers), SMP based on single-chip multi-core and/or GPUs are merged together to form a heterogeneous parallel processing environment for high performance computing. Currently, such an hybrid scheme has been chosen by IBM to build *Roadrunner* [118] super computer. Roadrunner currently is the world's fastest computer. Its hybrid design consists of a dual-core AMD Opteron processors and an attached Cell Broadband Engine processor per node.

New high performance computing tools described above will make it possible to implement in real-time more sophisticated and computationally more demanding imaging algorithms. As a result, the scope of the MRI guided interventions will be greatly enhanced.

REFERENCES

1. Atalar, E., P. A. Bottomley, O. Ocali, L. C. Correia, M. D. Kelemen, J. A. Lima, and E. A. Zerhouni, "High resolution intravascular mri and mrs by using a catheter receiver coil.," *Magn Reson Med*, Vol. 36, pp. 596–605, Oct 1996.

2. Ocali, O., and E. Atalar, "Intravascular magnetic resonance imaging using a loopless catheter antenna.," *Magn Reson Med*, Vol. 37, pp. 112–118, Jan 1997.

3. Serfaty, J. M., X. Yang, P. Aksit, H. H. Quick, M. Solaiyappan, and E. Atalar, "Toward mri-guided coronary catheterization: visualization of guiding catheters, guidewires, and anatomy in real time.," *J Magn Reson Imaging*, Vol. 12, pp. 590–594, Oct 2000.

4. Beyersdorff, D., A. Winkel, B. Hamm, S. Lenk, S. A. Loening, and M. Taupitz, "Mr imaging-guided prostate biopsy with a closed mr unit at 1.5 t: initial results.," *Radiology*, Vol. 234, pp. 576–581, Feb 2005.

5. Quick, H. H., M. O. Zenge, H. Kuehl, G. Kaiser, S. Aker, S. Massing, S. Bosk, and M. E. Ladd, "Interventional magnetic resonance angiography with no strings attached: wireless active catheter visualization.," *Magn Reson Med*, Vol. 53, pp. 446–455, Feb 2005.

6. Ozturk, C., M. Guttman, E. R. McVeigh, and R. J. Lederman, "Magnetic resonance imaging-guided vascular interventions.," *Top Magn Reson Imaging*, Vol. 16, pp. 369–381, Oct 2005.

7. Lufkin, R. B., "Interventional mr imaging.," *Radiology*, Vol. 197, pp. 16–18, Oct 1995.

8. Lufkin, R. B., D. H. Gronemeyer, and R. M. Seibel, "Interventional mri: update.," *Eur Radiol*, Vol. 7 Suppl 5, pp. 187–200, 1997.

9. Debatin, J. F., and G. Adam, eds., *Interventional magnetic resonance imaging*, Springer Netherlands, 2004.

10. Lederman, R. J., "Cardiovascular interventional magnetic resonance imaging.," *Circulation*, Vol. 112, pp. 3009–3017, Nov 2005.

11. Roemer, P. B., W. A. Edelstein, C. E. Hayes, S. P. Souza, and O. M. Mueller, "The nmr phased array.," *Magn Reson Med*, Vol. 16, pp. 192–225, Nov 1990.

12. Hutchinson, M., and U. Raff, "Fast mri data acquisition using multiple detectors.," *Magn Reson Med*, Vol. 6, pp. 87–91, Jan 1988.

13. Carlson, J. W., and T. Minemura, "An algorithm for NMR imaging reconstruction based on multiple RF receiver coils," *J Magn Reson*, Vol. 74, pp. 376–380, 1987.

14. Kwiat, D., S. Einav, and G. Navon, "A decoupled coil detector array for fast image acquisition in magnetic resonance imaging.," *Med Phys*, Vol. 18, no. 2, pp. 251–265, 1991.

15. Carlson, J. W., and T. Minemura, "Imaging time reduction through multiple receiver coil data acquisition and image reconstruction.," *Magn Reson Med*, Vol. 29, pp. 681–687, May 1993.

16. Ra, J. B., and C. Y. Rim, "Fast imaging using subencoding data sets from multiple detectors.," *Magn Reson Med*, Vol. 30, pp. 142–145, Jul 1993.

17. Larkman, D. J., and R. G. Nunes, "Parallel magnetic resonance imaging.," *Phys Med Biol*, Vol. 52, pp. R15–R55, Apr 2007.

18. O'Sullivan, J. D., "A fast sinc function gridding algorithm for fourier inversion in computer tomography.," *IEEE Trans Med Imaging*, Vol. 4, no. 4, pp. 200–207, 1985.

19. Jackson, J. I., C. H. Meyer, D. G. Nishimura, and A. Macovski, "Selection of a convolution function for fourier inversion using gridding [computerised tomography application].," *IEEE Trans Med Imaging*, Vol. 10, no. 3, pp. 473–478, 1991.

20. Rasche, V., R. Proksa, R. Sinkus, P. Bornert, and H. Eggers, "Resampling of Data Between Arbitrary Grids Using Convolution Interpolation," *IEEE Transaction on Medical Imaging*, Vol. 18, May 1999.

21. Gabr, R. E., P. Aksit, P. A. Bottomley, A.-B. M. Youssef, and Y. M. Kadah, "Deconvolution-interpolation gridding (DING): accurate reconstruction for arbitrary k-space trajectories.," *Magn Reson Med*, Vol. 56, no. 6, pp. 1182–1191, 2006.

22. Dale, B., M. Wendt, and J. L. Duerk, "A rapid look-up table method for reconstructing MR images from arbitrary K-space trajectories.," *IEEE Trans Med Imaging*, Vol. 20, no. 3, pp. 207–217, 2001.

23. Beatty, P. J., D. G. Nishimura, and J. M. Pauly, "Rapid gridding reconstruction with a minimal oversampling ratio.," *IEEE Trans Med Imaging*, Vol. 24, no. 6, pp. 799–808, 2005.

24. Rosenfeld, D., "An optimal and efficient new gridding algorithm using singular value decomposition.," *Magn Reson Med*, Vol. 40, no. 1, pp. 14–23, 1998.

25. Rosenfeld, D., "New approach to gridding using regularization and estimation theory.," *Magn Reson Med*, Vol. 48, no. 1, pp. 193–202, 2002.

26. Moriguchi, H., and J. L. Duerk, "Iterative next-neighbor regridding (INNG): improved reconstruction from nonuniformly sampled k-space data using rescaled matrices.," *Magn Reson Med*, Vol. 51, no. 2, pp. 343–352, 2004.

27. Seiberlich, N., F. A. Breuer, M. Blaimer, K. Barkauskas, P. M. Jakob, and M. A. Griswold, "Non-Cartesian data reconstruction using GRAPPA operator gridding (GROG).," *Magn Reson Med*, Vol. 58, no. 6, pp. 1257–1265, 2007.

28. Seiberlich, N., F. Breuer, M. Blaimer, P. Jakob, and M. Griswold, "Self-calibrating GRAPPA operator gridding for radial and spiral trajectories.," *Magn Reson Med*, Vol. 59, no. 4, pp. 930–935, 2008.

29. Hoge, R. D., R. K. Kwan, and G. B. Pike, "Density compensation functions for spiral mri.," *Magn Reson Med*, Vol. 38, pp. 117–128, Jul 1997.

30. Pipe, J. G., and P. Menon, "Sampling Density Compensation in MRI: Rationale and an Iterative Numerical Solution," *Magnetic Resonance in Medicine*, Vol. 41, 1999.

31. Tan, H., and C. H. Meyer, "Estimation of k-space trajectories in spiral mri.," *Magn Reson Med*, Vol. 61, pp. 1396–1404, Jun 2009.

32. Sodickson, D. K., and W. J. Manning, "Simultaneous acquisition of spatial harmonics (smash): fast imaging with radiofrequency coil arrays.," *Magn Reson Med*, Vol. 38, pp. 591–603, Oct 1997.

33. Sodickson, D. K., M. A. Griswold, and P. M. Jakob, "Smash imaging.," *Magn Reson Imaging Clin N Am*, Vol. 7, pp. 237–54, vii–viii, May 1999.

34. Jakob, P. M., M. A. Griswold, R. R. Edelman, and D. K. Sodickson, "Auto-smash: a self-calibrating technique for smash imaging. simultaneous acquisition of spatial harmonics.," *MAGMA*, Vol. 7, pp. 42–54, Nov 1998.

35. Heidemann, R. M., M. A. Griswold, A. Haase, and P. M. Jakob, "Vd-auto-smash imaging.," *Magn Reson Med*, Vol. 45, pp. 1066–1074, Jun 2001.

36. Griswold, M. A., P. M. Jakob, R. M. Heidemann, M. Nittka, V. Jellus, K. Wang, B. Kiefer, and A. Haase, "Generalized autocalibrating partially parallel acquisitions (GRAPPA)," *Magn Reson Med*, Vol. 47, 2002.

37. Bydder, M., D. J. Larkman, and J. V. Hajnal, "Generalized smash imaging.," *Magn Reson Med*, Vol. 47, pp. 160–170, Jan 2002.

38. Breuer, F. A., P. Kellman, M. A. Griswold, and P. M. Jakob, "Dynamic autocalibrated parallel imaging using temporal GRAPPA (TGRAPPA)," *Magn Reson Med*, Vol. 53, 2005.

39. Pruessmann, K. P., M. Weiger, M. B. Scheidegger, and P. Boesiger, "SENSE: Sensitivity encoding for fast MRI.," *Magn Reson Med*, Vol. 42, no. 5, pp. 952–962, 1999.

40. Kellman, P., F. H. Epstein, and E. R. McVeigh, "Adaptive sensitivity encoding incorporating temporal filtering (TSENSE).," *Magn Reson Med*, Vol. 45, no. 5, pp. 846–852, 2001.

41. Saybasili, H., P. Kellman, J. A. Derbyshire, E. R. McVeigh, and M. A. Guttman, "Parallelized hybrid TGRAPPA reconstruction for real-time interactive MRI.," *Proc. of MICCAI 2008, Lecture Notes in Computer Science, Springer-Verlag*, Vol. 5242, no. Pt 2, pp. 163–170, 2008.

42. Saybasili, H., P. Kellman, M. A. Griswold, J. A. Derbyshire, and M. A. Guttman, "HTGRAPPA: Real-time b(1)-weighted image domain tgrappa reconstruction.," *Magn Reson Med*, Vol. 61, pp. 1425 – 1433, Apr 2009.

43. Saybasili, H., P. Kellman, M. A. Griswold, J. A. Derbyshire, and M. Guttman, "HTGRAPPA: TGRAPPA based b1-weighted image domain reconstruction for real-time mri," in *Proceedings of ISMRM 2009, Honolulu, HI*, 2009.

44. Griswold, M. A., S. Kannengiesser, R. M. Heidemann, J. Wang, and P. M. Jakob, "Field-of-view limitations in parallel imaging.," *Magn Reson Med*, Vol. 52, pp. 1118–1126, Nov 2004.

45. Goldfarb, J. W., "The sense ghost: field-of-view restrictions for sense imaging.," *J Magn Reson Imaging*, Vol. 20, pp. 1046–1051, Dec 2004.

46. Blaimer, M., F. Breuer, M. Mueller, R. M. Heidemann, M. A. Griswold, and P. M. Jakob, "Smash, sense, pils, grappa: how to choose the optimal method.," *Top Magn Reson Imaging*, Vol. 15, pp. 223–236, Aug 2004.

47. Preibisch, C., T. Wallenhorst, R. Heidemann, F. E. Zanella, and H. Lanfermann, "Comparison of parallel acquisition techniques generalized autocalibrating partially parallel acquisitions (grappa) and modified sensitivity encoding (msense) in functional mri (fmri) at 3t.," *J Magn Reson Imaging*, Vol. 27, pp. 590–598, Mar 2008.

48. Bernstein, M. A., K. F. King, and X. J. Zhou, *Handbook of MRI Pulse Sequences*, Elsevier Academic Press, 2004.

49. Pipe, J. G., "Motion correction with PROPELLER MRI: application to head motion and free-breathing cardiac imaging.," *Magn Reson Med*, Vol. 42, no. 5, pp. 963–969, 1999.

50. Noll, D. C., "Multishot rosette trajectories for spectrally selective mr imaging.," *IEEE Trans Med Imaging*, Vol. 16, pp. 372–377, Aug 1997.

51. Griswold, M. A., M. Blaimer, F. Breuer, R. M. Heidemann, M. Mueller, and P. M. Jakob, "Parallel magnetic resonance imaging using the GRAPPA operator formalism.," *Magn Reson Med*, Vol. 54, no. 6, pp. 1553–1556, 2005.

52. Guttman, M. A., P. Kellman, A. J. Dick, R. J. Lederman, and E. R. McVeigh, "Real-time accelerated interactive MRI with adaptive TSENSE and UNFOLD.," *Magn Reson Med*, Vol. 50, no. 2, pp. 315–321, 2003.

53. Guttman, M. A., C. Ozturk, A. N. Raval, V. K. Raman, A. J. Dick, R. DeSilva, P. Karmarkar, R. J. Lederman, and E. R. McVeigh, "Interventional cardiovascular procedures guided by real-time mr imaging: an interactive interface using multiple slices, adaptive projection modes and live 3d renderings.," *J Magn Reson Imaging*, Vol. 26, pp. 1429–1435, Dec 2007.

54. Roessle, M., K. Haag, A. Ochs, M. Sellinger, G. Noeldge, J. M. Perarnau, E. Berger, U. Blum, A. Gabelmann, and K. Hauenstein, "The transjugular intrahepatic portosystemic stent-shunt procedure for variceal bleeding.," *N Engl J Med*, Vol. 330, pp. 165–171, Jan 1994.

55. Saybasili, H., A. Z. Faranesh, C. E. Saikus, C. Ozturk, R. J. Lederman, and M. Guttman, "Multi-phase 3d angiography roadmaps for real-time mri-guided procedures," in *Proceedings of ISMRM 2009, Honolulu, HI*, 2009.

56. Soederman, M., D. Babic, R. Homan, and T. Andersson, "3d roadmap in neuroangiography: technique and clinical interest.," *Neuroradiology*, Vol. 47, pp. 735–740, Oct 2005.

57. Rhode, K. S., D. L. G. Hill, P. J. Edwards, J. H. Hipwell, D. Rueckert, G. I. Sanchez-Ortiz, S. Hegde, V. Rahunathan, and R. Razavi, "Application of xmr 2d-3d registration to cardiac interventional guidance," in *MICCAI (1)*, pp. 295–302, 2003.

58. Rhode, K. S., M. Sermesant, D. Brogan, S. Hegde, J. Hipwell, P. Lambiase, E. Rosenthal, C. Bucknall, S. A. Qureshi, J. S. Gill, R. Razavi, and D. L. G. Hill, "A system for real-time xmr guided cardiovascular intervention.," *IEEE Trans Med Imaging*, Vol. 24, pp. 1428–1440, Nov 2005.

59. Sathyanarayana, S., P. Aksit, A. Arepally, P. V. Karmarkar, M. Solaiyappan, and E. Atalar, "Tracking planar orientations of active mri needles.," *J Magn Reson Imaging*, Vol. 26, pp. 386–391, Aug 2007.

60. Quick, H. H., H. Kuehl, G. Kaiser, D. Hornscheidt, K. P. Mikolajczyk, S. Aker, J. F. Debatin, and M. E. Ladd, "Interventional mra using actively visualized catheters, truefisp, and real-time image fusion.," *Magn Reson Med*, Vol. 49, pp. 129–137, Jan 2003.

61. George, A., C. E. Saikus, J. A. Derbyshire, A. Z. Faranesh, O. Kocaturk, R. J. Lederman, and M. A. Guttman, "Fast, robust 3d visualization and automatic slice repositioning ("snap-to") for mr-guided interventions using active device "profiling"," in *Proceedings of ISMRM 2009, Honolulu, HI*, 2009.

62. Saybasili, H., C. Ozturk, A. Ademoglu, R. J. Lederman, and O. Kocaturk, "Real-time edge detection for improved device visualization during RT-MRI guided interventions," in *Proceedings of ESMRMB 2009, Antalya, Turkey*, 2009.

63. Griswold, M. A., "Advanced k-space techniques," in *Second International Workshop on Parallel MRI*, (Zurich), 2005.

64. Wang, J. M., and B. Zhang, "Fast generalized autocalibrating partially parallel acquisition image reconstruction algorithm for magnetic resonance imaging," 2005.

65. Brau, A. C., P. J. Beatty, S. Skare, and R. Bammer, "Comparison of reconstruction accuracy and efficiency among autocalibrating data-driven parallel imaging methods," *Magn Reson Med*, Vol. 59, pp. 382–395, 2008.

66. Walsh, D. O., A. F. Gmitro, and M. W. Marcellin, "Adaptive reconstruction of phased array MR imagery.," *Magn Reson Med*, Vol. 43, no. 5, pp. 682–690, 2000.

67. Madore, B., G. H. Glover, and N. J. Pelc, "Unaliasing by fourier-encoding the overlaps using the temporal dimension (unfold), applied to cardiac imaging and fmri.," *Magn Reson Med*, Vol. 42, pp. 813–828, Nov 1999.

68. Martin Blaimer, Felix Breuer, Matthias Mueller, Robin M. Heidemann, Mark A. Griswold, and Peter M. Jakob, "SMASH, SENSE, PILS, GRAPPA. How to Choose the Optimal Method," *Top Magn Resonance Imaging*, Vol. 15, no. 4, 2004.

69. Derbyshire, J. A., D. A. Herzka, and E. R. McVeigh, "S5fp: spectrally selective suppression with steady state free precession.," *Magn Reson Med*, Vol. 54, pp. 918–928, Oct 2005.

70. Derbyshire, J. A., D. A. Herzka, and E. R. McVeigh, "S5FP: Spectrally selective suppression with steady state free precession," *Magn Reson Med*, Vol. 54, pp. 918–928, 2005.

71. Kellman, P., and E. R. McVeigh, "Image reconstruction in snr units: a general method for snr measurement.," *Magn Reson Med*, Vol. 54, pp. 1439–1447, Dec 2005.

72. Fredrickson, J. O., and N. J. Pelc, "Temporal resolution improvement in dynamic imaging.," *Magn Reson Med*, Vol. 35, pp. 621–625, Apr 1996.

73. Hansen, M. S., D. Atkinson, and T. S. Sorensen, "Cartesian sense and k-t sense reconstruction using commodity graphics hardware.," *Magn Reson Med*, Vol. 59, pp. 463–468, Mar 2008.

74. Glover, G. H., and J. M. Pauly, "Projection reconstruction techniques for reduction of motion effects in MRI.," *Magn Reson Med*, Vol. 28, no. 2, pp. 275–289, 1992.

75. Meyer, C. H., B. S. Hu, D. G. Nishimura, and A. Macovski, "Fast spiral coronary artery imaging.," *Magn Reson Med*, Vol. 28, no. 2, pp. 202–213, 1992.

76. Liao, J. R., "Real-time image reconstruction for spiral mri using fixed-point calculation.," *IEEE Trans Med Imaging*, Vol. 19, pp. 690–698, Jul 2000.

77. Schiwietz, T., T. chiun Chang, P. Speier, and R. Westermann, "MR image reconstruction using the GPU," in *Proceedings of SPIE*, 2006.

78. Stone, S. S., J. P. Haldar, S. C. Tsao, W.-m. W. Hwu, Z.-P. Liang, and B. P. Sutton, "Accelerating advanced mri reconstructions on GPUs," in *CF '08: Proceedings of the 5th conference on Computing frontiers*, (New York, NY, USA), pp. 261–272, ACM, 2008.

79. Seiberlich, N., F. Breuer, R. Heidemann, M. Blaimer, M. Griswold, and P. Jakob, "Reconstruction of undersampled non-Cartesian data sets using pseudo-Cartesian GRAPPA in conjunction with GROG.," *Magn Reson Med*, Vol. 59, no. 5, pp. 1127–1137, 2008.

80. Seiberlich, N., F. A. Breuer, P. Ehses, H. Moriguchi, M. Blaimer, P. M. Jakob, and M. A. Griswold, "Using the GRAPPA operator and the generalized sampling theorem to reconstruct undersampled non-Cartesian data.," *Magn Reson Med*, 2009.

81. Ziou, D., and S. Tabbone, "Edge detection techniques - an overview," *Intern Journ of Pattern Recogn and Image Analysis*, Vol. 8, pp. 537–559, 1998.

82. Canny, J. F., "Finding edges and lines in images," Master's thesis, M.I.T. Artif. Intell. Lab., Cambridge, MA, 1983.

83. Deriche, R., "Using canny's criteria to derive a recursively implemented optimal edge detector," *International Journal of Computer Vision*, Vol. 1, pp. 167–187, June 1987.

84. Monga, O., R. Deriche, G. Malandain, and J.-P. Cocquerez, "Recursive filtering and edge tracking: two primary tools for 3-d edge detection," *Image and Vision Computing*, Vol. 9, pp. 203–214, August 1991.

85. Pelage, J.-P., M. E. Hajjam, C. Lagrange, T. Chinet, A. Vieillard-Baron, S. Chagnon, and P. Lacombe, "Pulmonary artery interventions: an overview.," *Radiographics*, Vol. 25, no. 6, pp. 1653–1667, 2005.

86. Kee, S. T., A. Ganguly, B. L. Daniel, Z. Wen, K. Butts, A. Shimikawa, N. J. Pelc, R. Fahrig, and M. D. Dake, "Mr-guided transjugular intrahepatic portosystemic shunt creation with use of a hybrid radiography/mr system.," *J Vasc Interv Radiol*, Vol. 16, pp. 227–234, Feb 2005.

87. Kocaturk, O., C. E. Saikus, A. Z. Faranesh, M. A. Guttman, K. Ratnayaka, and R. J. Lederman, "An active two channel guidewire for interventional cardiovascular mri," in *Proceedings of ISMRM 2009, Honolulu, HI*, 2009.

88. Serra, J., *Image Analysis and Mathematical Morphology*, Ac. Press, London, 1982.

89. Qu, P., C. Wang, and G. X. Shen, "Discrepancy-based adaptive regularization for grappa reconstruction.," *J Magn Reson Imaging*, Vol. 24, pp. 248–255, Jul 2006.

90. Miao, J., W. C. K. Wong, and D. L. Wilson, "Anisotropic kernel support for improved fast grappa imaging," in *Proceedings of ISMRM 2009, Honolulu, HI*, 2009.

91. Guo, W., and F. Huang, "A local mutual information guided denoising technique and its application to self-calibrated partially parallel imaging.," *Med Image Comput Comput Assist Interv Int Conf Med Image Comput Comput Assist Interv*, Vol. 11, no. Pt 2, pp. 939–947, 2008.

92. Moriguchi, H., and J. L. Duerk, "Bunched phase encoding (bpe): a new fast data acquisition method in mri.," *Magn Reson Med*, Vol. 55, pp. 633–648, Mar 2006.

93. Tsao, J., P. Boesiger, and K. P. Pruessmann, "k-t blast and k-t sense: dynamic mri with high frame rate exploiting spatiotemporal correlations.," *Magn Reson Med*, Vol. 50, pp. 1031–1042, Nov 2003.

94. Huang, F., J. Akao, S. Vijayakumar, G. R. Duensing, and M. Limkeman, "k-t grappa: a k-space implementation for dynamic mri with high reduction factor.," *Magn Reson Med*, Vol. 54, pp. 1172–1184, Nov 2005.

95. Tsao, J., S. Kozerke, P. Boesiger, and K. P. Pruessmann, "Optimizing spatiotemporal sampling for k-t blast and k-t sense: application to high-resolution real-time cardiac steady-state free precession.," *Magn Reson Med*, Vol. 53, pp. 1372–1382, Jun 2005.

96. Tsao, J., M. S. Hansen, and S. Kozerke, "Accelerated parallel imaging by transform coding data compression with k-t sense.," *Conf Proc IEEE Eng Med Biol Soc*, Vol. 1, p. 372, 2006.

97. Xu, D., K. F. King, and Z.-P. Liang, "Improving k-t sense by adaptive regularization.," *Magn Reson Med*, Vol. 57, pp. 918–930, May 2007.

98. Gamper, U., P. Boesiger, and S. Kozerke, "Compressed sensing in dynamic mri.," *Magn Reson Med*, Vol. 59, pp. 365–373, Feb 2008.

99. Ji, J. X., C. Zhao, and T. Lang, "Compressed sensing parallel magnetic resonance imaging.," *Conf Proc IEEE Eng Med Biol Soc*, Vol. 2008, pp. 1671–1674, 2008.

100. Lustig, M., D. Donoho, and J. M. Pauly, "Sparse mri: The application of compressed sensing for rapid mr imaging.," *Magn Reson Med*, Vol. 58, pp. 1182–1195, Dec 2007.

101. Supercomputing Technologies Group, Massachusetts Institute of Technology Laboratory for Computer Science, *Cilk 5.4.6 Reference Manual*, nov 2001.

102. Moreland, K., and E. Angel, "The FFT on a GPU," in *Proc. of the ACM SIGGRAPH/EUROGRAPHICS Conf. on Graphics Hardware (HWWS'03)*, pp. 112–119, 2003.

103. Fialka, O., and M. Cadik, "FFT and convolution performance in image filtering on GPU," in *10th International Conference on Information Visualization (IV'06)*, pp. 609–614, 2006.

104. Govindaraju, N. K., B. Lloyd, Y. Dotsenko, B. Smith, and J. Manferdelli, "High performance discrete fourier transforms on graphics processors," in *Proc. of the IEEE/ACM Conf. on Supercomputing (SC'08)*, pp. 1–12, 2008.

105. Nukada, A., Y. Ogata, T. Endo, and S. Matsuoka, "Bandwidth intensive 3-D FFT kernel for GPUs using CUDA," in *Proc. of the IEEE/ACM Conf. on Supercomputing (SC'08)*, 2008.

106. Chow, A., G. Fossum, and D. Brokenshire, "A programming example: Large FFT on the Cell broadband engine," in *Global Signal Processing Expo (GSPx)*, 2005.

107. Greene, J., and R. Cooper, "A parallel 64k complex FFT algorithm for the IBM/Sony/Toshiba Cell broadband engine processor," in *Global Signal Processing Expo (GSPx)*, 2005.

108. Bader, D., and V. Agarwal, "FFTC: Fastest Fourier Transform on the IBM Cell broadband engine," in *14th IEEE Int. Conf. on High Performance Computing (HiPC'07)*, 2007.

109. Williams, S., J. Shalf, L. Oliker, S. Kamil, P. Husbands, and K. Yelick, "The potential of the cell processor for scientific computing," in *ACM Proc. of the 3rd Conf. on Comput. Frontiers (CF'06)*, pp. 9–20, 2006.

110. Bader, D. A., V. Agarwal, and S. Kang, "Computing discrete transforms on the cell broadband engine," *Parallel Comput.*, Vol. 35, no. 3, pp. 119–137, 2009.

111. Vishkin, U., G. Caragea, and B. Lee, *Parallel Computing: Models, Algorithms and Applications*, ch. Models for advancing PRAM and other algorithms into parallel programs for a PRAM-On-Chip platform. CRC press, 2008.

112. Tzannes, A., C. Caragea, A. Balkan, and U. Vishkin, "XMT-C tutorial." `http://www.umiacs.umd.edu/users/vishkin/XMT/tutorial4xmtc2out-of2.pdf`, Nov 2008.

113. Tzannes, A., C. Caragea, A. Balkan, and U. Vishkin, "XMT-C manual." `http://www.umiacs.umd.edu/users/vishkin/XMT/manual4xmtc1out-of2.pdf`, Jan 2009.

114. "Software release of the explicit multi-threading (XMT) programming environment." `http://www.umiacs.umd.edu/users/vishkin/XMT/sw-release.html`, Aug 2008.

115. Naishlos, D., J. Nuzman, C.-W. Tseng, , and U. Vishkin, "Evaluating the XMT parallel programming model.," in *Proc. 6th Workshop on High-Level Parallel Programming Models and Supportive Environments (HIPS-6)*, 2001.

116. Caragea, G., A. Saybasili, X. Wen, and U. Vishkin, "Performance potential of an easy-to-program PRAM-On-Chip prototype versus state-of-the-art processor," in *Proc. 21st ACM Symp. on Parallel Algorithms and Archit. (SPAA'09)*, 2009.

117. Roujol, S., B. D. D. Senneville, E. Vahala, T. S. Sorensen, C. Moonen, and M. Ries, "Online real-time reconstruction of adaptive TSENSE with commodity CPU / GPU hardware," in *Proceedings of ISMRM 2009, Honolulu, HI*, 2009.

118. "Roadrunner: Hardware and software overview," January 2009. `http://www.redbooks.ibm.com/redpapers/pdfs/redp4477.pdf`.