An Efficient Framework for Compressed Sensing
Reconstruction of Highly Accelerated Dynamic Cardiac MRI

Dissertation

Presented in Partial Fulfillment of the Requirements for the Degree
Doctor of Philosophy in the Graduate School of The Ohio State
University

By

Samuel T. Ting, B.S., M.S.

Graduate Program in Biomedical Engineering

The Ohio State University

2016

Dissertation Committee:
Professor Orlando P. Simonetti, Advisor
Professor Lee C. Potter
Professor Rizwan Ahmad
Professor Jun Liu
© Copyright by

Samuel T. Ting

2016
Abstract

The research presented in this work seeks to develop, validate, and deploy practical techniques for improving diagnosis of cardiovascular disease. In the philosophy of biomedical engineering, we seek to identify an existing medical problem having significant societal and economic effects and address this problem using engineering approaches.

Cardiovascular disease is the leading cause of mortality in the United States, accounting for more deaths than any other major cause of death in every year since 1900 with the exception of the year 1918. Cardiovascular disease is estimated to account for almost one-third of all deaths in the United States, with more than 2150 deaths each day, or roughly 1 death every 40 seconds. In the past several decades, a growing array of imaging modalities have proven useful in aiding the diagnosis and evaluation of cardiovascular disease, including computed tomography, single photon emission computed tomography, and echocardiography. In particular, cardiac magnetic resonance imaging is an excellent diagnostic tool that can provide within a single exam a high quality evaluation of cardiac function, blood flow, perfusion, viability, and edema without the use of ionizing radiation. The scope of this work focuses on the application of engineering techniques for improving imaging using cardiac magnetic resonance with the goal of improving the utility of this powerful imaging modality.
Dynamic cine imaging, or the capturing of movies of a single slice or volume within the heart or great vessel region, is used in nearly every cardiac magnetic resonance imaging exam, and adequate evaluation of cardiac function and morphology for diagnosis and evaluation of cardiovascular disease depends heavily on both the spatial and temporal resolution as well as the image quality of the reconstruction cine images. This work focuses primarily on image reconstruction techniques utilized in cine imaging; however, the techniques discussed are also relevant to other dynamic and static imaging techniques based on cardiac magnetic resonance. Conventional segmented techniques for cardiac cine imaging require breath-holding as well as regular cardiac rhythm, and can be time-consuming to acquire. Inadequate breath-holding or irregular cardiac rhythm can result in completely non-diagnostic images, limiting the utility of these techniques in a significant patient population. Real-time single-shot cardiac cine imaging enables free-breathing acquisition with significantly shortened imaging time and promises to significantly improve the utility of cine imaging for diagnosis and evaluation of cardiovascular disease. However, utility of real-time cine images depends heavily on the successful reconstruction of final cine images from under sampled data. Successful reconstruction of images from more highly undersampled data results directly in images exhibiting finer spatial and temporal resolution provided that image quality is sufficient.

This work focuses primarily on the development, validation, and deployment of practical techniques for enabling the reconstruction of real-time cardiac cine images at the spatial and temporal resolutions and image quality needed for diagnostic utility. Particular emphasis is placed on the development of reconstruction approaches resulting in with short computation times that can be used in the clinical environment.
Specifically, the use of compressed sensing signal recovery techniques is considered; such techniques show great promise in allowing successful reconstruction of highly undersampled data. The scope of this work concerns two primary topics related to signal recovery using compressed sensing: (1) long reconstruction times of these techniques, and (2) improved sparsity models for signal recovery from more highly undersampled data. Both of these aspects are relevant to the practical application of compressed sensing techniques in the context of improving image reconstruction of real-time cardiac cine images. First, algorithmic and implementational approaches are proposed for reducing the computational time for a compressed sensing reconstruction framework. Specific optimization algorithms based on the fast iterative/shrinkage algorithm (FISTA) are applied in the context of real-time cine image reconstruction to achieve efficient per-iteration computation time. Implementation within a code framework utilizing commercially available graphics processing units (GPUs) allows for practical and efficient implementation directly within the clinical environment. Second, patch-based sparsity models are proposed to enable compressed sensing signal recovery from highly undersampled data. Numerical studies demonstrate that this approach can help improve image quality at higher undersampling ratios, enabling real-time cine imaging at higher acceleration rates. In this work, it is shown that these techniques yield a holistic framework for achieving efficient reconstruction of real-time cine images with spatial and temporal resolution sufficient for use in the clinical environment. A thorough description of these techniques from both a theoretical and practical view is provided – both of which may be of interest to the reader in terms of future work.
To my family, especially little Benjamin.
Acknowledgments

I would first and foremost like to thank my advisor Dr. Orlando P. Simonetti for his mentoring me and guiding me through this entire journey. His words have always been apt and timely, pointing me in the right direction throughout the course of my graduate work. I would also like to thank Dr. Yu Ding for his helpful advice during my initial exploratory years. I like to acknowledge Drs. Peter Kellman, Michael Hansen, and Hui Xue for their help with the Gadgetron. I also express my gratitude to Drs. Lee C. Potter and Rizwan Ahmad in providing helpful guidance through the forest of optimization. I would like to specifically acknowledge Debbie Scandling – thank you for helping with all the last minute scheduling of volunteers. I would also like to express my gratitude to the cardiology fellows, specifically Juliana Serafim da Silveira and Jason Craft for taking the time to score images. I would also like to thank Nathan Lambda for the time he took to draw contours. To all the current students in the lab – Juliet Varghese, Ria Mazumdar, David Gross – thank you. Your presence has helped make this journey all the more exciting. I would also like to acknowledge all the past students in our group - Mihaela Jekic, Shivraman Giri, Eric Foster, Vijay Balasubramanian, Jacob Bender, Prashanth Palaniappan, Paras Parikh.

I would like to thank my family, in particular my parents, for their love, care, and prayer throughout this entire journey. Thanks you for being there when I needed to talk, and for praying for me when I could not talk.
To all the young people I have had the privilege of caring for during my journey, thank you. Little do you know that in your allowing me to care for you have I been equally well cared for.
Vita

2004 ........................................... B.S. Electrical Engineering, University of Washington.

2012 ........................................... M.S. Biomedical Engineering, The Ohio State University.

2009 – present ................................. Graduate Research Associate, The Ohio State University.

Publications

Research Publications


Fields of Study

Major Field: Biomedical Engineering
# Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstract</td>
<td>ii</td>
</tr>
<tr>
<td>Dedication</td>
<td>v</td>
</tr>
<tr>
<td>Acknowledgments</td>
<td>vi</td>
</tr>
<tr>
<td>Vita</td>
<td>viii</td>
</tr>
<tr>
<td>List of Tables</td>
<td>xv</td>
</tr>
<tr>
<td>List of Figures</td>
<td>xvi</td>
</tr>
<tr>
<td>1. Introduction</td>
<td>1</td>
</tr>
<tr>
<td>2. Background and Significance</td>
<td>4</td>
</tr>
<tr>
<td>2.1 Prevalence of Cardiovascular Disease Among the US Population</td>
<td>4</td>
</tr>
<tr>
<td>2.2 Introduction to Cardiovascular Disease</td>
<td>6</td>
</tr>
<tr>
<td>2.2.1 Coronary Heart Disease</td>
<td>6</td>
</tr>
<tr>
<td>2.2.2 Heart Failure</td>
<td>10</td>
</tr>
<tr>
<td>2.2.3 Hypertension</td>
<td>16</td>
</tr>
<tr>
<td>2.2.4 Stroke</td>
<td>17</td>
</tr>
<tr>
<td>2.3 Non-Imaging-Based Approaches for Diagnosis of Cardiovascular Disease</td>
<td>17</td>
</tr>
<tr>
<td>2.3.1 Patient History</td>
<td>17</td>
</tr>
<tr>
<td>2.3.2 Electrocardiography</td>
<td>18</td>
</tr>
<tr>
<td>2.3.3 Biomarkers</td>
<td>19</td>
</tr>
<tr>
<td>2.3.4 Clinical Risk Assessment Score</td>
<td>19</td>
</tr>
<tr>
<td>2.3.5 Exercise Treadmill Test</td>
<td>20</td>
</tr>
<tr>
<td>2.4 Imaging-Based Diagnosis of Cardiovascular Disease</td>
<td>20</td>
</tr>
<tr>
<td>Section</td>
<td>Page</td>
</tr>
<tr>
<td>------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>2.4.1 Computed Tomographic Coronary Angiography</td>
<td>21</td>
</tr>
<tr>
<td>2.4.2 Myocardial Perfusion Imaging with Single Photon Emission</td>
<td>22</td>
</tr>
<tr>
<td>Computed Tomography</td>
<td></td>
</tr>
<tr>
<td>2.4.3 Transthoracic Echocardiography</td>
<td>24</td>
</tr>
<tr>
<td>2.4.4 Cardiac Magnetic Resonance Imaging</td>
<td>24</td>
</tr>
<tr>
<td>2.5 Concluding Remarks</td>
<td>29</td>
</tr>
<tr>
<td>3. Cardiac Magnetic Resonance Imaging for Diagnosis of Cardiovascular Disease</td>
<td>30</td>
</tr>
<tr>
<td>3.1 An Introduction to MRI Physics</td>
<td>31</td>
</tr>
<tr>
<td>3.2 An Overview of the Basic Acquisition Techniques Used in Cardiac Imaging</td>
<td>43</td>
</tr>
<tr>
<td>3.2.1 Velocity Imaging</td>
<td>43</td>
</tr>
<tr>
<td>3.2.2 Perfusion Imaging</td>
<td>46</td>
</tr>
<tr>
<td>3.2.3 Late Gadolinium Enhancement</td>
<td>47</td>
</tr>
<tr>
<td>3.2.4 Parameter Mapping Techniques</td>
<td>48</td>
</tr>
<tr>
<td>3.2.5 Cine Imaging</td>
<td>51</td>
</tr>
<tr>
<td>3.3 K-Space and Reconstruction of Cine MRI Data – A Broad Perspective</td>
<td>55</td>
</tr>
<tr>
<td>3.4 Segmented Cine Acquisition</td>
<td>58</td>
</tr>
<tr>
<td>3.5 Real-time, Single-Shot Cine Imaging</td>
<td>61</td>
</tr>
<tr>
<td>3.6 Parallel Image Reconstruction with Multiple Coil Arrays - An Overview of SENSE, GRAPPA, and SPIRiT</td>
<td>65</td>
</tr>
<tr>
<td>3.7 Image Reconstruction from a Linear Algebra Perspective</td>
<td>70</td>
</tr>
<tr>
<td>3.8 Image Reconstruction from an Optimization Perspective</td>
<td>71</td>
</tr>
<tr>
<td>3.9 Parallel Image Reconstruction with Tikhonov Regularization</td>
<td>74</td>
</tr>
<tr>
<td>3.10 Inadequacy of Current Parallel Imaging Techniques for Real-Time Single-Shot Cine Imaging</td>
<td>75</td>
</tr>
<tr>
<td>4. Fundamentals of Signal Recovery with Compressed Sensing for Reconstrution of Highly Accelerated Dynamic Cardiac MRI</td>
<td>77</td>
</tr>
<tr>
<td>4.1 Fundamental Concepts for Compressed Sensing</td>
<td>77</td>
</tr>
<tr>
<td>4.2 A Sparse Review of Optimization Approaches for the Compressed Sensing Problem</td>
<td>87</td>
</tr>
<tr>
<td>4.3 Incoherent Samplings</td>
<td>88</td>
</tr>
<tr>
<td>5. A Fast Implementation for Compressive Recovery of Highly Accelerated Cardiac Cine MRI</td>
<td>92</td>
</tr>
<tr>
<td>5.1 bFISTA: A Fast Implementation for Compressive Recovery of Highly Accelerated Cardiac Cine MRI Using the Balanced Sparse Model</td>
<td>92</td>
</tr>
</tbody>
</table>
7. Fast MRI Reconstruction within the Gadgetron Framework: A Practical Implementation .................................................. 166

7.1 Some Remarks on the Motivation Behind the Gadgetron ........ 167
7.2 An Architectural Overview of the Gadgetron ...................... 168
7.3 File Structure of the Gadgetron Code ............................. 177
7.4 Connecting the Gadgetron to the Client Application ............... 178
  7.4.1 The gadgetron.xml File .................................. 179
  7.4.2 The gadgetron.ini File .................................. 181
  7.4.3 Connection to the Siemens ICE/IDEA Environment .......... 183
  7.4.4 Connection to the Siemens MRI Scanner Environment ...... 188
7.5 Accessing GPU Functionality on the Gadgetron Server Remotely . 189
7.6 Parameter Map Processing - Vendor to ISMRMRD Parameter Conversion ................................................................. 190
7.7 Installation of the MrRecon Codebase ............................. 192
7.8 Interprocess Communication within the Gadgetron Framework .................................................................................. 195
  7.8.1 The process_config and process Functions ................ 198
7.9 XML File Format for the Gadgetron Stream Configuration ....... 200
7.10 Reconstruction within the MrRecon Codebase .................... 203
  7.10.1 Reconstruction using TGRAPPA .......................... 203
  7.10.2 Reconstruction using SPIRiT ............................. 219
  7.10.3 Reconstruction using SENSE ............................. 236
7.11 Detailed Description of MrReconPerSliceRTAccumulatorGadget . 243
7.12 Concluding Remarks ............................................. 253

8. Conclusions and Future Directions .................................... 255

8.1 On the Application to other CMR Imaging Techniques .......... 257
  8.1.1 Preliminary Investigation of Applications to Perfusion Imaging258
  8.1.2 Potential Applications to other CMR Techniques .......... 265
8.2 A Portable Extension of MrRecon to Multiple Dimensional Data . 269
8.3 The Use of Mutual Information in Patch-based Regularization ... 270
8.4 Final Concluding Remarks ....................................... 271

Bibliography ............................................................. 273
List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1</td>
<td>Image Scores (Mean ± Standard Deviation) For Each Method Across All Volunteers, All Views, and All Readers</td>
</tr>
<tr>
<td>5.2</td>
<td>Mean Reconstruction Times For Each Method Across All Volunteers and All Views</td>
</tr>
</tbody>
</table>
# List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1 Cardiovascular disease (CVD) deaths versus cancer deaths by age (United States: 2011). CVD includes <em>International Classification of Diseases, 10th Revision (ICD-10)</em> codes I00 to I99 and Q20 to Q28; cancer, C00 to C97. [1]</td>
<td>5</td>
</tr>
<tr>
<td>2.3 Percentage breakdown of deaths attributable to cardiovascular disease (United States: 2011). Total may not add to 100 because of rounding. Coronary heart disease includes <em>ICD-10</em> codes I20 to I25; stroke, I60 to I69; heart failure, I50; high blood pressure, I10 to I15; diseases of the arteries, I70 to I78; and other, all remaining <em>ICD-10</em> I categories and Q20 to Q28. *Not a true underlying cause. [1]</td>
<td>7</td>
</tr>
<tr>
<td>2.4 Hospital discharges (<em>ICD-9</em>) for the 10 leading diagnostic groups. [1]</td>
<td>8</td>
</tr>
<tr>
<td>2.5 Pathophysiology of coronary heart disease [2]. Left: Example vessel narrowing due to atherosclerotic plaque formation. Right: Anterior view of the heart showing major coronary vessels.</td>
<td>8</td>
</tr>
<tr>
<td>2.6 Example CTCA image showing coronary arteries. Source: <em>Wikipedia</em></td>
<td>21</td>
</tr>
<tr>
<td>2.7 Example MPI SPECT images of the heart. Source: <em>Wikipedia</em></td>
<td>23</td>
</tr>
<tr>
<td>2.8 Example TTE images of the heart, showing the right ventricle (RV), right atrium (RA), left ventricle (LV), left atrium (LA), left ventricular outflow tract (LVOT), and the aortic outflow (Ao). Source: <em>Wikipedia</em></td>
<td>25</td>
</tr>
</tbody>
</table>
2.9 Example images of the heart in the short-axis (a), two-chamber (b), and four-chamber (c) views acquired with cardiac magnetic resonance imaging.

3.1 Example of an MRI scanner. Source: The Ohio State University.

3.2 Diagram of main components of an MRI scanner. (1) Main magnet, (2) Gradient coils, (3) transmit coil (Tx), (4) receive coil array (Rx).

3.3 Block diagram illustrating data acquisition using an MRI scanner.

3.4 Placement of protons in the presence of an external magnetic field \( B_0 \) results in the generation of a net magnetization in a direction parallel to the main magnetic field. Source: [3].

3.5 Energy from an RF pulse at the resonant frequency is absorbed by spins, causing the net magnetization to rotate away from the z axis. Source: [3].

3.6 Illustration of the longitudinal direction (along the direction of the main magnetic field) and the transverse plane (perpendicular to the main magnetic field). Magnetization along the z-direction is referred to as longitudinal magnetization. Magnetization in the xy-plane is referred to as transverse magnetization. Source: [3].

3.7 Longitudinal (T1) relaxation. Application of a 90° RF pulse causes the longitudinal magnetization to become zero. Over time, the longitudinal magnetization will grow back in a direction parallel to the main magnetic field. Source: [3].

3.8 Transverse (T2*) relaxation. Immediately after application of a 90° RF pulse, transverse magnetization is maximized and dephasing begins to occur. The signal from dephasing protons cancel out, reducing the MR signal. Source: [3].

3.9 The concept of precession. Precession of a spinning top and a nuclear spin are similar in that an external force combined with spinning motion causes precession around the direction of the external force. Source: [3].
3.10 Example of a full k-space frame (left column, magnitude component shown), and different specific k-space points (center column, white pixel). K-space points at different locations correspond to plane wave oscillations in the image (signal intensity) domain (right column).

3.11 Spins moving along an external magnetic field gradient acquire a difference in the phase of their rotation (right), whereas stationary spins do not (left). The accumulated phase difference is proportional to the velocity of the moving spin. \( t = \text{time}, V = \text{velocity}, \Phi = \text{phase shift} \). Source: [4]

3.12 Basic principle behind most clinically used phase-contrast sequences. Two acquisitions are performed, each with all parameters kept constant except for the flow-sensitizing bipolar gradients. The data of the two acquisitions are subtracted. The effective flow encoding is achieved by means of the difference in the bipolar gradients of the two acquisitions. This technique eliminates all phase shifts induced by imaging gradients. \( t = \text{time}, V = \text{velocity} \). Source: [4]

3.13 The phase-contrast measurement generates two sets of dynamic images. The magnitude image (left) resembles a normal bright-blood image and is used for anatomic orientation. In the velocity image (right), the gray scale value of each pixel represents the velocity information in that voxel. Black values show flow toward the viewer, whereas white values show flow away from the viewer. Source: [4]

3.14 A typical first-pass perfusion imaging pulse sequence. Cardiac-gated acquisitions are performed by using an electrocardiographic signal. Multiple sections are acquired consecutively (e.g., six sections every two R-R intervals, as shown here). This is repeated continuously during the first pass and washout of the contrast agent. \( T_1 \) contrast is generated by using a saturation preparation (Prep) pulse followed by fast acquisition of each section using a GRE, GRE-EPI, or SSFP sequence. Source: [5]

3.15 Reversible ischemia in a 60-year-old man with a high-grade stenosis in the left anterior descending coronary artery territory. Short-axis cine first-pass perfusion MR images during adenosine stress (left) and rest (right) show a subendocardial segment of hypoperfusion in the anteroseptal LV wall (arrow on left). Source: [5]
3.16 (a) Relaxation of magnetization for myocardium with delayed enhancement compared to normal myocardium. (b) Diagram of delayed enhancement MR imaging sequence illustrating timing of the inversion pulse. Source: [6]

3.17 Acute idiopathic myocarditis in a 30-year-old man with frequent premature ventricular contractions and normal results at coronary angiography. Two-chamber long-axis MR image of the left ventricle (a) and short-axis MR image (b) show extensive scattered delayed enhancement in the anterior, lateral, and inferior wall and apex of the left ventricle and the right ventricular wall (arrows). Source: [6]

3.18 Example of T1 mapping with a Look-Locker (LL) MR imaging sequence in a healthy 34-year-old man with normal myocardium. Source: [7]. (a) LL (inversion time [TI] scout) image series performed at 1.5T show 21 images acquired at a fixed location (short-axis, mid-ventricle) at different TIs ranging from 75 ms (top left) to 695 ms (bottom right), with a 30 ms increase in TI between images. Images were acquired after injection of gadolinium contrast agent. (b) T1 map displays a color scale of the uniform T1 values of the myocardium. (c) Graph of a multiparameter curve-fitting analysis of a selected pixel in the interventricular septum shows how each value is used to derive the contrast-enhanced T1 value (299 ms) for this pixel.

3.19 Example of T1 mapping in a 37-year-old woman with a lateral wall infarction. Source: [7]. Short-axis MOLLI (1.5T) T1 mapping images acquired 26 minutes after gadolinium-based contrast agent administration show a focal area of decreased T1 values in the lateral wall of the left ventricle.

3.20 Example high spatial resolution (1 × 1 mm²) and temporal resolution (11.2 ms) segmented breath-held cine images in the short axis (a, b), two-chamber (c, d), and four-chamber (e, f) orientations at end systole (left column) and end diastole (right column) acquired from the author’s heart with a 45 second breath-hold.

3.21 A typical steady state free precession (SSFP) imaging sequence implementation. Source: [8]
3.22 Example fully sampled (top row) and undersampled (bottom row) k-space trajectories. (Left) Cartesian, (Middle) Radial, (Right) Spiral. Source: [9] ................................. 55

3.23 Characteristic undersampling artifacts (top) and point spread functions (PSF) (bottom) for several k-space sampling trajectories. The fully sampled k-space reference is shown in the left column. The other images show artifacts from an undersampled Cartesian trajectory and its PSF (middle left), undersampled radial trajectory (middle right), and undersampled variable density spiral trajectory (far right). The PSF can help to give some indication as to the form of aliasing introduced due to the particular undersampled k-space trajectory. Source: [9] ................................. 56

3.24 Minimum sampling rate required to acquire different numbers of harmonics, as a function of heart rate. These results are based on calculations using the Nyquist criterion. Source: [10] ......................... 58

3.25 Illustration showing segmented acquisition across five heart beats (5 segments). During each heart beat (or cardiac cycle) a particular segment of k-space corresponding to a specific cardiac phase is collected. The duration of each cardiac cycle (R-R interval) must remain constant between cycles and the respiratory motion between cycles must be suspended to produce a useful image series. ..................... 59

3.26 Schematic cartoon illustrating a segmented acquisition during a breath-hold (i.e. respiratory motion is suspended during acquisition) . . . . . . . . 60

3.27 Schematic cartoon illustrating a segmented acquisition during free-breathing (i.e. respiratory motion is not suspended during acquisition). 61
3.28 Long-axis cardiac MR images acquired in the same 6-mm-thick section during the same phase of the cardiac cycle with different balanced SSFP sequences. (a) Image obtained with a standard breath-hold segmented balanced SSFP technique shows a marked artifact and an indistinct border between the myocardium and the left ventricular cavity because of breathing during image acquisition. (b) Image obtained with a real-time balanced SSFP parallel acquisition technique (time-domain SENSE) with prospective gating shows the elimination of the motion-related artifact. The endocardial contour is blurred, mainly because of a larger pixel size and poorer temporal resolution. Contrast between myocardium and blood also is inferior to that in (a) because of the larger pixel size and lower flip angle. Source: [11].

3.29 Illustration showing a real-time acquisition across five heart beats. During each heart beat (or cardiac cycle) only the necessary k-space data for reconstructing each specific cardiac phase is collected during the temporal duration of the cardiac phase. The time to collect the k-space data for a specific cardiac phase determined the temporal resolution of the resulting cine and is assumed to be short enough so as to “freeze” any cardiac motion.

3.30 Some possible linear combinations of component coil sensitivities for an example eight-coil phased array, approximating several spatial harmonics of frequency $m\Delta k_y$. Weighted individual coil sensitivity profiles are depicted as thin solid lines beneath each component coil. Dashed lines represent the sinusoidal or cosinusoidal weighting functions. Combined sensitivity profiles are indicated by thick solid lines. These combined profiles closely approximate ideal spatial harmonics across the array. Source: [12].

3.31 Component coil sensitivity information for three-element and four-element arrays. (a) Absolute magnitude images. (b) Sensitivity along the diameter indicated by dashed lines in (a). (c) Absolute magnitude images of the abdomen and thorax of a healthy volunteer. (d) Absolute magnitude sensitivity along the dashed lines in (c). Source: [12].

xxi
3.32 Schematic illustrating the different perspectives for understanding the MR image reconstruction problem. MR physics motivates the formulation of a model for the measurement process, which then leads to a reconstruction problem. The reconstruction problem can be reformulated as an optimization problem, which suggests the use of specific optimization techniques to obtain an optimal solution to the MR image reconstruction problem. 72

4.1 Example random sampling trajectory (e) and variable density pseudorandom trajectory (g) along with corresponding point spread functions in (f) and (h). The point spread function demonstrating higher incoherence shows lower side lobe intensities outside the central peak. The side lobes for the pseudo-random trajectory are marginally stronger compared to the random trajectory. Source: [13]. 82

4.2 Comparison of uniformly downsampled (a), PDS (b), and VISTA (c) trajectories at various acceleration rates. Uniformly downsampled trajectories fall on a diagonal rather than horizontal grid in order to satisfy the requirement that the temporal average of all frames generates a fully-sampled k-space. This requirement is additionally satisfied with the VISTA sampling trajectories, but not with the PDS trajectories. Source: [13] 91

5.1 Short-axis views of real-time, free-breathing cines from volunteer A. (a) Individual frames from NLCG-SPIRiT, bFISTA-SPIRiT, and bFISTA-SENSE for six different acceleration rates. The frame with the worst median sharpness measurement is presented from each image series. (b) Temporal profiles along line drawn on the top-left image in panel (a). Tick marks below each profile indicate temporal location of frame presented in panel (a). Compared to the NLCG-SPIRiT results, bFISTA-SPIRiT results achieve a significant reduction in reconstruction time with little appreciable reduction in image quality of dynamic features. bFISTA-SENSE achieves further reduction in reconstruction times with a slight increase in blurring artifacts at rate 10 and above (arrows). 109
5.2 Four-chamber views of real-time, free-breathing cines from volunteer A. (a) Individual frames from NLCG-SPIRiT, bFISTA-SPIRiT, and bFISTA-SENSE for six different acceleration rates. The twentieth frame is presented from each image series. (b) Temporal profiles along line drawn on the top-left image in panel (a). bFISTA-SPIRiT again achieves reduced reconstruction times compared to NLCG-SPIRiT with little difference in image quality for all acceleration rates. bFISTA-SENSE begins to exhibit blurring artifacts at acceleration rates 12 and above (arrows). With bFISTA, the SENSE reconstruction can achieve clinically relevant reconstruction times of 13.8 seconds without a diagnostically significant reduction in image quality.

5.3 Average noise standard deviation and edge sharpness quantification across all volunteers. Noise measurements for each reconstruction method (a) are averaged across both 4CH and SAX views. The left ventricular boundary was used for the sharpness measurement (b). Between all three reconstruction methods, no significant difference in noise level was observed for each volunteer and each acceleration rate (p > 0.7). Between NLCG-SPIRiT and bFISTA-SPIRiT, no significant difference in sharpness was observed for all volunteers and acceleration rates (p > 0.05 for all comparisons). Significant reduction in sharpness was observed between NLCG-SPIRiT and bFISTA-SENSE at rates 7 and above across all volunteers (p < 0.05).

5.4 Phantom RMSE for S-SPIRiT and NLCG with additional noise. NLCG achieves optimal RMSE in 50 seconds, while S-SPIRiT surpasses NLCG in 25 seconds.

5.5 In vivo NLCG (left column) and S-SPIRiT (right column) images. S-SPIRiT shows visibly reduced noise levels in both the lung region (top row) and within the myocardium and blood pool (bottom row).

5.6 $L_2$-error convergence behavior for IST with SURE and NLCG approaches. $L_2$-error is calculated with respect to the reconstruction from fully sampled data. Source: [14]

5.7 The Shepp-Logan phantom of size $N = 128 \times 128$.

5.8 Comparison of rSNR versus undersampling ratio $M/N$ for MFISTA and bFISTA in the Shepp-Logan phantom.
5.9 Example breath-held segmented (left) and free-breathing real-time (right) images at end systole (top) and end diastole (bottom).

5.10 Summary of volumetric parameters (mean ± std. dev.). P-values show no significant difference between breath-held segmented data and breath-held real-time data.

5.11 Scoring of artifact, temporal fidelity, and image quality (mean ± std. dev.) for all volunteers.

5.12 Example breath-held segmented (left) and free-breathing real-time (right) images at end systole (top) and end diastole (bottom).

5.13 Summary of volumetric parameters (mean ± std. dev.). P-values show no significant difference between breath-held segmented data and free-breathing real-time data.

5.14 Scoring of artifact, temporal fidelity, and image quality (mean ± std. dev.) for all volunteers.

6.1 (a) The Shepp-Logan phantom of size $N = 128 \times 128$. (b) The brain image of size $N = 128 \times 128$.

6.2 Recovery SNR versus sampling ratio $M/N$ for the Shepp Logan phantom. Measurements were constructed using the spread-spectrum operator and AWGN at 40 dB SNR, and recovery used UWT-2d-db1 at one level of decomposition.

6.3 Recovery SNR versus sampling ratio $M/N$ for the brain image. Measurements were constructed using the spread-spectrum operator and AWGN at 40 dB SNR, and recovery used UWT-2d-db1 at one level of decomposition.

6.4 (a) Example frame from the digital phantom. (b) Temporal profile for the digital phantom. (c) Example sampling pattern for $M/N = 1/12$.

6.5 Recovery SNR versus sampling ratio $M/N$ for the digital phantom. Measurements were constructed using AWGN at 30 dB SNR, and recovery used UWT-3d-db1 at one level of decomposition.
6.6 (a) Temporal profile from recovery results for the digital phantom with measurement SNR of 30 dB at $M/N = 1/12$. (a) L1, (b) C-L1, (c) CPAR-L1, (d) DPAR-L1 .............................. 158

6.7 (a) Example frame from ground truth image series for dynamic cardiac MRI experiment. (b) Temporal profile for ground truth image series. 158

6.8 Example reconstructed frames and temporal profiles from dynamic cardiac MRI results for $M/N = 1/9$. The same frame as that of the ground truth is used. (a-b) L1 (rSNR: 14.01 dB), (c-d) C-L1 (rSNR: 19.60 dB), (e-f) CPAR-L1 (rSNR: 19.47 dB), (g-h) DPAR-L1 (rSNR: 19.78 dB). 160

6.9 Example exercise stress images from one volunteer in the two-chamber long-axis orientation at acceleration rate 6. APBW shows reduced artifact level compared to FSW. ................................. 163

6.10 Temporal profiles along line indicated in cine frame for each weighting method for the volunteer shown in Figure 6.9. APBW shows improved sharpness in dynamic edges (arrow). ................................. 164

6.11 Example exercise stress images from one volunteer in the short-axis orientation at acceleration rate 8. ................................. 164

6.12 Temporal profiles along line indicated in cine frame for each weighting method for the volunteer shown in Figure 6.11. APBW shows improved sharpness in dynamic edges (arrow). ................................. 165

7.1 Diagram showing the relation between the Gadgetron framework and a vendor-nonspecific MRI system. ................................. 169
7.2 An overview of the Gadgetron architecture. Source [15]. The Gadgetron communicated with a client application through a TCP/IP connection. The client application sends data to the Gadgetron and associated with each data package is a Message ID. Based on the Message ID, control of the socket is handed over to a specific Reader, which is capable of deserializing the incoming data package. The data package is converted to message blocks that are added to the first Gadget’s queue. Data are then passed down the Gadget stream where each Gadget can modify and transform the data. Any Gadget can return images (or partially processed data) to the Gadgetron framework. Based on the Message ID of this return data package, the control of the socket and the data are handed to a particular Writer, which is responsible for writing the return message to the client.

7.3 Each gadget is an active object with at least one thread of execution. The thread is responsible for picking generic data packages off the input message queue and decoding the data packages into the data types that the gadget is expecting. If the data types match the gadget’s signature, the data will be passed on to a data processing function where the data can be modified and/or transformed into new data packages and enqueued on the next gadget’s input queue. Source: [15]

7.4 Root level directory structure for the Gadgetron source code. Not all files and directories are shown.

7.5 Directory structure for code contributions in the MrRecon codebase described in this work. Only directory structure is shown.

7.6 The configuration interface for the PuTTY software.

7.7 The Tunnel configuration panel for the PuTTY software.

7.8 Schematic illustrating communication between the client application and multiple Gadgetron instances. ACE is responsible for handling all intra-communication among Gadgetron instances and between Gadgetron instances and the client application.
7.9 Schematic illustrating interprocessing communication between gadgets within a Gadgetron instance. (1) An ISMRMRD parameter set is first sent to all gadgets after gadget streaming pipeline is initialized. (2) Each message received from the client application is sent to the first gadget on the input side of the streaming pipeline. The ISMRMRD parameter set is processed by the function `config_process` within each gadget. The messages are processed by the function `process` within each gadget.

7.10 Schematic illustrating the general data process in a single gadget. (1) The ACE function `getq` is used to retrieve a message from the gadget’s queue. (2) Relevant data processing is performed using functionality provided by GPU- or CPU-based toolboxes if necessary. (3) Message is enqueued in the child gadget’s queue using the ACE function `putq`.

7.11 XML tree structure for a gadgetronStreamConfiguration.

7.12 Schematic block diagram for `tpatGrappa.xml`. Gadgets from both `gadgetron_mricore` and `mrrecon` DLLs are used. Portions of gadget stream are indicated where data is serialized by (1) k-space line; (2) complete dataset, i.e. across spatial, channel, and frame dimensions for real-time cine; and (3) frames, i.e. across spatial and channel dimensions. Data after sum-of-squares combination (SOSCoilCombine) is serialized by spatial dimensions only. Parameters for the primary reconstruction gadget are summarized.

7.13 Schematic block diagram for `kGrappa.xml`. Gadgets from both `gadgetron_mricore` and `mrrecon` DLLs are used. Portions of gadget stream are indicated where data is serialized by (1) k-space line; (2) complete dataset, i.e. across spatial, channel, and frame dimensions for real-time cine; and (3) frames, i.e. across spatial and channel dimensions. Data after sum-of-squares combination (SOSCoilCombine) is serialized by spatial dimensions only. Parameters for the primary reconstruction gadget are summarized.

7.14 Block schematic diagram for `vSpiritFISTA.xml`. Diagram indicates which DLL each gadget is selected from in addition to the serialization of the data being passed between various gadgets in the configuration stream.
7.15 Block schematic diagram for vSenseFISTA.xml. Diagram indicates which DLL each gadget is selected from in addition to the serialization of the data being passed between various gadgets in the configuration stream. The primary difference between the SENSE reconstruction is the use of the SENSE reconstruction gadget and the omission of the frame merging and GRAPPA reconstruction gadgets.

7.16 Schematic diagram for accumulator gadget. The process function receives an input stream of k-space lines. After each k-space line is received, a check is performed to see whether the k-space lines for a complete slice is received. If so, correct image headers for this slice are formed, and the slice data is sent downstream for further processing. A CLOSE signal from the parent Gadgetron process terminates the gadget once all slices have been received.

8.1 Diagram for SSFPP acquisition (TOP) and 3D reconstruction with SPIRiT (BOTTOM). Color figure available in electronic version.

8.2 Example SR-SPGR, SR-SSFP, SSFPP, and SPGR images at baseline and at peak contrast showing the high SNR and CNR of the SSFPP acquisition technique. Source: [16].

8.3 Example raw 3D SSFPP images from a volunteer using view-shared GRAPPA (top row) and SPIRiT with a 3D kernel (bottom row). Frames are selected from the end of the acquisition, when the volunteer was no longer able to breath-hold. Significant breathing motion during frames 45-48 introduce blurring of the blood-myocardium boundary in the view-shared GRAPPA results. This blurring is reduced in the SPIRiT results (frames 45-48, arrows) in which there was no view sharing across heart beats.
Chapter 1: Introduction

The aim of this dissertation is to develop, validate, and deploy practical approaches for compressed sensing reconstruction of highly accelerated cardiac magnetic resonance (CMR) imaging. These techniques were conceived, developed, and validated as part of the author’s graduate work in biomedical engineering. The presentation of this work in the following chapters closely follows the philosophy of biomedical engineering, i.e. to identify a problem or gap in medicine and utilize engineering approaches to address this gap.

Chapter 2 presents the background and significance of the problem being addressed as well as some motivation for addressing this work. We begin with an overview of the societal impact of cardiovascular disease in the United States as well as the role played by noninvasive diagnostic imaging in mediating the impact of this prevalent disease. We also cover the various imaging modalities currently in use, including SPECT, echocardiography, and computed tomography and their role related to this class of diseases. Specifically, we discuss the use of magnetic resonance imaging (MRI) in the diagnosis of cardiovascular disease as well as some of the advantages that this modality provides beyond other imaging modalities. We also discuss the role played by rest and stress CMR related to cardiovascular disease. Finally, we introduce some of the imaging techniques used in MRI for the diagnosis of this class of diseases.
and provide some motivation regarding the technical challenges that remain in the development and deploying of these techniques.

Chapter 3 presents a brief overview of the various CMR imaging techniques that make this imaging modality a promising tool in the clinical environment. Particular emphasis is placed on cine imaging, and the background pertaining to both breath-held segmented and real-time, single-shot, free-breathing cardiac cine imaging techniques is developed. We present a brief overview of conventional techniques for the reconstruction of real-time, free-breathing cardiac cine and discuss the formulation of fundamental parallel imaging techniques that are used for this reconstruction in addition to the relevant signal model. This chapter serves as the technical foundation upon which the remaining chapters build.

Chapter 4 builds upon the signal model developed in chapter 3 to present a brief overview of compressed sensing techniques from a general signal processing perspective in addition to its specific application to reconstruction of dynamic MRI. We also present a selected review of some of the optimization techniques used in compressed sensing in addition to MRI-specific approaches for satisfying conditions for performing compressed sensing.

Chapter 5 presents our proposed efficient approach for implementing compressed sensing reconstruction of highly accelerated real-time, free-breathing cardiac cine MRI. Smaller studies further investigating this approach and its performance relative to other techniques are also included in this chapter. We also present studies comparing cine images reconstructed using the proposed technique with conventional segmented cine images. The specific aim of this chapter is to demonstrate the practical feasibility of our proposed approach in achieving fast reconstruction times with
adequate diagnostic image quality and illustrate the utility of our approach in the clinical environment.

Chapter 6 presents a further investigation into the sparsity models utilized in compressed sensing reconstruction of dynamic MRI data. We investigate the use of patch-based sparsity promoting models to achieve improved image recovery in the context of both static and dynamic scenarios with the ultimate goal of achieving successful image recovery in the context of more highly undersampled dynamic MRI data. This investigation involves the extension of composite regularization to patch-based regularization. We present feasibility studies demonstrating the utility of a novel patch-based approach with validation in numerical studies. We also present initial results illustrating the potential utility in the context of image recovery of exercise stress MRI data.

In chapter 7 we present a detailed and thorough description of a practical implementation of various aspects of this work within the Gadgetron framework, leading to a practical demonstration of our techniques directly on the scanner in a clinical environment.

Finally, in chapter 8 we conclude with ongoing work and future directions.
Chapter 2: Background and Significance

The aim of this chapter is to provide sufficient motivation from a clinical, societal, and economic perspective for the engineering problem being addressed by this research. We first discuss the prevalence of cardiovascular disease among the US population with a focus specifically on its societal and economic effects. We continue by providing a selective overview of existing noninvasive diagnostic techniques for evaluating this class of diseases. We specifically introduce the imaging technique of cardiovascular magnetic resonance (CMR) imaging and discuss its advantages compared to other existing techniques. Finally, we introduce some specific imaging techniques used in CMR and relevant to this body of research.

2.1 Prevalence of Cardiovascular Disease Among the US Population

In the United States, the class of diseases known as cardiovascular disease (CVD) remains the leading cause of mortality, accounting for more deaths than any other major cause of death in every year since 1900 with the exception of the year 1918 [1]. According to the most recent report from the American Heart Association as of the time of the writing of this work [1], CVD is estimated to account for 32.3% of all deaths in the United States. On average, more than 2150 Americans die of CVD
Figure 2.1: Cardiovascular disease (CVD) deaths versus cancer deaths by age (United States: 2011). CVD includes International Classification of Diseases, 10th Revision (ICD-10) codes I00 to I99 and Q20 to Q28; cancer, C00 to C97. [1]

each day, or roughly 1 death every 40 seconds, and CVD currently claims more lives each year than cancer (Figure 2.1), especially among older populations. In 2011, the annual direct and indirect cost of CVD and stroke in the United States was estimated to be $320.1 billion. By 2030, 43.9% of the US population is projected to have some form of CVD, with total costs projected to increase to over $1 trillion dollars (Figure 2.2) [1].
2.2 Introduction to Cardiovascular Disease

In this section, we provide an overview of some of the major categories of CVD with the primary goal of motivating the significance and relevance of diagnostic imaging modalities.

2.2.1 Coronary Heart Disease

Among the major categories of CVD, coronary heart disease (CHD) is the most common class of CVD [1], accounting for almost 50% of all CVD deaths (Figure 2.3). CHD, also known as coronary artery disease or ischemic heart disease, results due to ischemia (cell starvation due to limited oxygen supply) of the myocardial cells. Ischemia of the myocardium can exhibit itself as limited cardiac function both
at rest and under exercise stress, and prolonged ischemia can lead to myocardial
damage either in the form of myocardial scarring without regrowth or myocardial
necrosis (death). Typical risk factors for CHD involve high blood pressure, smoking,
diabetes, lack of physical exercise, obesity, high blood cholesterol, poor diet, and
excessive alcohol consumption. While numerous studies have been conducted towards
addressing these risk factors, CHD still remains one of the leading causes for hospital
admittance in the United States (Figure 2.4).

CHD is typically caused by atherosclerosis within the coronary arteries that supply
oxygen to the myocardium. Figure 2.5 shows the heart with the major coronary
arteries (left anterior descending (LAD), circumflex (LCX), and the right coronary
Figure 2.4: Hospital discharges (ICD-9) for the 10 leading diagnostic groups. [1]

Figure 2.5: Pathophysiology of coronary heart disease [2]. Left: Example vessel narrowing due to atherosclerotic plaque formation. Right: Anterior view of the heart showing major coronary vessels.
artery (RCA) in addition to the buildup of atherosclerotic plaque which can impede coronary blood supply to the myocardium. Atherosclerosis refers to a disease of the arteries characterized by a thickening of the portion of the arterial vessel wall that is closest to the lumen with plaques consisting primarily of (1) large numbers of abnormal smooth muscle cells, macrophages, and lymphocytes, (2) deposits of cholesterol and other fatty substances both within these cells and extracellularly, and (3) dense layers of connective tissue matrix. Atherosclerosis can reduce coronary blood flow through a number of mechanisms. Extra muscle cells and various deposits within the atherosclerotic plaque can bulge into the lumen, increasing resistance to flow. Dysfunctional endothelial cells in the atherosclerotic region release excess vasoconstrictors and reduced levels of vasodilators. These processes are progressive, occasionally leading to complete occlusion (blockage) of the coronary arteries. However, the primary cause of complete occlusion is through the formation of a blood clot. Typically, atherosclerotic plaques are encapsulated within a fibrous cap, whose rupture can trigger a cascade of events leading to the formation of a blood clot at the site of rupture. This clot formation can suddenly block blood supply to a local region of the myocardium, leading to myocardial infarction, or a heart attack.

The primary indication for CHD precipitating a visit to the emergency department (ED) is chest pain, also known as angina pectoris, or simply, angina. Angina may be classified as being either stable or unstable. Stable angina (SA) typically is precipitated by some form of physical activity with symptoms subsiding several minutes after activity and recurring when activity resumes. Unstable angina (UA), a form of acute coronary syndrome (ACS), is defined as angina pectoris that changes or steadily worsens. UA exhibits at least one of the three following features: (1) it
occurs at rest, (2) it is severe and is of a new onset, (3) it occurs with a crescendo pattern. SA may be associated with plaque formation that is protected with a fibrous cap. UA may be associated with atherosclerosis that yields an active unstable plaque, which can undergo thrombosis to yield acute myocardial ischemia. This myocardial ischemia, if not reversed, can lead to cell necrosis, or a myocardial infarction.

2.2.2 Heart Failure

Another significant category of CVD is heart failure (Figure 2.3). Heart failure, or congestive heart failure, can be defined broadly as the failure of the pumping mechanism of the heart, or the failure of the heart to supply sufficient blood flow to meet the needs of the body. While the primary effect of heart failure is reduced cardiac output, the effects of reduced cardiac output may not necessarily be severe or life-threatening. The factors and physiological processes responsible for the onset of heart failure and the change from stable to life-threatening conditions are complex. In this section we aim to present a brief overview of some of the aspects of cardiovascular pathophysiology relevant to the role played by various imaging modalities in clinical evaluation.

Heart failure can be classified in a number of meaningful ways, some of which we proceed to delineate here. Chronic heart failure refers to a long term condition of reduced cardiac output, typically with stable symptoms. Alternatively, acute heart failure refers to the rapid onset of relevant conditions. Heart failure may also be classified as compensated, referring to the onset of conditions due to stressors or defects in which the heart is still able to function, albeit perhaps non-optimally. Decompensated heart failure refers to the onset of conditions in response to stressors
or defects that leads to the functional deterioration of the heart due to its inability
to adequately compensate for these stressors or defects.

An additional classification of heart failure is the differentiation between left-sided
failure and right-sided failure. *Left-sided* heart failure refers to failure in the left
side of the heart (left ventricle and atrium) and can have downstream effects in the
brain and rest of the body and upstream effects within the pulmonary vasculature.
Downstream effects can include dizziness and confusion as well as cool extremities
at rest. Upstream effects within the respiratory system can include increased rate
of breathing and increased work of breathing. Patients can also develop pulmonary
edema. Patients may also exhibit shortness of breath upon exertion or, in severe cases,
at rest also. Additional symptoms include increased breathlessness while lying flat,
sudden night-time attacks of severe breathlessness, and exercise intolerance. *Right-
sided* heart failure refers to failure in the right side of the heart (right ventricle and
atrium) and can result primarily in peripheral edema.

In particular, acute decompensated heart failure (ADHF) refers to the rapid on-
set of functional deterioration of the heart to the point that the heart is no longer
able to meet the demands of the body through adequate blood supply. This serious
and life threatening condition requires specific attention both in terms of diagnosis
and treatment. Patients exhibiting ADHF typically have presented previously with
chronic compensated heart failure, which frequently can deteriorate into ADHF. The
causes for this deterioration can vary considerably; however, a few common causes
can be a simultaneous illness such as pneumonia, myocardial infarction, arrhythmias,
hypertension, amyloidosis, or the patient’s inability to comply with a fluid, dietary,
or medication restriction [17]. While not necessarily precipitating ADHF onset, these
conditions can increase cardiac workload and thereby increase potential for ADHF onset. The presence of these conditions can result in one or more of the following changes in the heart itself, which can lead to ADHF onset:

- *Reduced Force of Contraction.* In a healthy heart, increased ventricular filling leads to increased contraction via the Frank-Starling mechanism, and this leads to a rise in cardiac output. However, in patients with heart failure, this mechanism can fail since ventricular filling can exceed cardiac muscle contraction, reducing cardiac efficiency.

- *Reduced Stroke Volume.* This can be the result of either a failure in the systolic phase or a failure in the diastolic cardiac phase, or both. In the former, this can be due to reduced contractility. In the latter, this can be due to reduced ventricular filling due to wall stiffening.

- *Reduced Reserve Capacity.* Due to increasing cardiac stress in meeting normal metabolic demands, the reserve capacity of the heart to increase cardiac output in response to increased oxygen demand can be reduced. This is typically exhibited as exercise intolerance in patients with heart failure.

- *Increased Heart Rate.* In an attempt to compensate for heart failure by maintaining blood pressure and perfusion, the sympathetic nervous system may increase heart rate in order to maintain cardiac output. While this may help initially, this response can place further strain on the myocardium, thereby increasing coronary perfusion demands and worsening cardiac ischemia. Such systemic response can also lead to increased risk of potentially fatal arrhythmias.
• *Hypertrophy.* Cardiac responses in trying to maintain cardiac output in the presence of heart failure can lead to enlargement of terminally differentiated heart muscle fibers. Such hypertrophy can contribute to increased wall stiffness and reduced ability of the myocardium to relax during diastole.

• *Enlargement of the Ventrices.* In order to maintain cardiac output, ventricular volume may increase in a spherical manner. Such volumetric increase can result in reduced stroke volume due to mechanical and contractile inefficiency.

These factors can lead to ADHF onset by placing additional strain on the heart, increasing the risk of cardiac arrest and leading to a generally reduced blood supply to the rest of the body. Additionally, patients with chronic heart failure can exhibit the following physiological compensations:

• Due to reduced cardiac output, arterial blood pressure is reduced, destimulating baroreceptors in the carotid sinus and aortic arch which links to the nucleus tractus solitarius in the brain. This region in the brain increases sympathetic activity, causing a release of catecholamines into the blood stream and ultimately leading to systemic arterial vasoconstriction. While blood pressure is maintained through this mechanism, peripheral resistance is increased, increasing cardiac workload. Additionally, catecholamine release leads to increased heart rate, improving cardiac output by increasing cardiac contraction at the cost of additional increased cardiac workload.

• Increased sympathetic response causes secretion of vasopressin (antidiuretic hormone, ADH) by the hypothalamus, resulting in increased blood volume and pressure due to increased fluid retention of the kidneys.
Reduced kidney perfusion stimulates the release of renin, which catalyses the production of angiotension, a potent vasopressor. Angiotensin and its metabolites cause further vasoconstriction and stimulate the release of aldosterone from the adrenal glands. This increases blood volume by promoting salt and fluid retention in the kidneys.

Chronically high levels of circulating neuroendocrine hormones (catecholamines, renin, angiotensin, and aldosterone) can affect the myocardium directly, leading to long term structural remodelling of the heart.

Reduced perfusion in skeletal muscle can lead to atrophy, leading to increased muscular fatigability and decreased muscular peak strength, ultimately contributing to increased exercise intolerance.

These conditions serve to increase peripheral resistance and blood volume, further straining the heart and accelerating deterioration of the myocardium. Increased cardiac workload can lead to increased calcium afterload and increased cytosolic calcium entry via increase in cyclic adenosine monophosphate (cAMP). Increased myocyte calcium entry augments myocardial contractility and impairs myocardial relaxation. Such calcium overload due to increased cardiac workload can induce arrhythmias and lead to sudden death. Additionally, increased myocardial contractility and reduced myocardial relaxation leads to increased myocardial energy expenditure and further reduction in cardiac output. Increased myocardial energy expenditure can lead to myocardial cell death or apoptosis, and this can lead to further reduced cardiac output and worsening heart failure, perpetuating a downward cycle of increased neuro-hormonal stimulation and further adverse hemodynamic and myocardial responses.
Vasoconstriction and fluid retention due to physiological compensation can increase hydrostatic pressure in the capillaries, shifting hydrostatic forces to favor increased interstitial fluid formation and resulting in edema. In right-sided heart failure, this typically occurs in the extremities; however, in left-sided heart failure, this can result in pulmonary edema. Pulmonary edema leads to reduced spare capacity for ventilation, increased stiffening of the lungs, and reduced gas exchange efficiency due to increased distance between air and blood within the lungs. As a result, patients with pulmonary edema can exhibit shortness of breath typically while lying flat.

Heart failure can result due to either systolic dysfunction or diastolic dysfunction, with the former being more easily recognized. Heart failure due to systolic dysfunction can be described simplistically as the failure of the pump mechanism of the heart. In this condition, ventricular contraction is attenuated, resulting in inadequate stroke volume and as a result, inadequate cardiac output. Typically, heart failure due to systolic dysfunction is characterized by an ejection fraction lower than 45%. In general, systolic heart failure can result from dysfunction or destruction of cardiac myocytes. Congenital diseases such as Duchenne muscular dystrophy can negatively affect the molecular structure of cardiac myocytes. Myocytes can also be adversely damaged by inflammation (e.g. myocarditis), infiltration (e.g. amyloidosis), or toxins or other pharmaceutical agents. However, the most common cause of heart failure due to systolic dysfunction is infarcted tissue and scar formation due to ischemia. Myocardial infarction results in dead myocytes being replaced by scar tissue, adversely affecting myocardium function. Additionally, with heart failure due to systolic dysfunction, inadequate ventricular emptying during systole results in elevated end-diastolic pressure and volume. This increased pressure and volume can be transmitted upstream to
the atrium and the circulatory system. In the left heart, this results in increased left atrial pressure and volume, which is then transmitted to the pulmonary vasculature. As a result, heart failure due to systolic dysfunction can lead to pulmonary edema due to extravasation of fluid into the lung parenchyma. In the right heart, this results in increased pressure in the systemic venous circulation and systemic capillary beds and exhibits itself as peripheral edema.

Heart failure due to diastolic dysfunction can be described as the failure of the ventricle to adequately relax during each cardiac cycle, perhaps due to a stiffer ventricular wall. Diastolic dysfunction typically results in inadequate ventricular filling which then results in inadequate stroke volume. End-diastolic pressures may also be elevated. Diastolic dysfunction can be caused by processes similar to systolic dysfunction, and symptoms are identical to those of systolic dysfunction (pulmonary and peripheral edema). However, diastolic dysfunction may be present at earlier stages of heart failure than systolic dysfunction and may be present asymptotically. Often, diastolic dysfunction may exhibit itself only under conditions of extreme physiological stress even with preserved systolic function. While the patient with diastolic dysfunction may be completely asymptomatic at rest, they may be extremely sensitive to increases in heart rates; and sudden tachycardia (due to physiological responses to exercise, fever, or dehydration or due to pathological tachyarrhythmias) can result in rapid onset pulmonary edema.

2.2.3 Hypertension

Hypertension, or high blood pressure, refers to the condition where the pressure in the arteries is consistently above the normal range. Hypertension is classified as a
sustained systolic/diastolic blood pressure of 140/90 mm Hg. While the causes are not entirely known, high blood pressure can lead to atherosclerosis, and increased risk for stroke or heart attack.

2.2.4 Stroke

Ischemic stroke occurs when a blood vessel supplying the brain gets blocked, typically due to a blood clot. This can lead to cell death within the brain. A hemorrhagic stroke occurs when a blood vessel within the brain bursts, typically due to uncontrolled hypertension. Permanent damage can result if excessive cell death within the brain occurs; however, injured cells can repair themselves, allowing for rehabilitation after a stroke event. Major risk factors for stroke include the following: hypertension, smoking, diabetes, high cholesterol, heart disease, and atrial fibrillation.

2.3 Non-Imaging-Based Approaches for Diagnosis of Cardiovascular Disease

In this section, we consider some of the non-imaging-based approaches for diagnosis and evaluation of various forms of CVD.

2.3.1 Patient History

Patient history is critically important in the initial evaluation of patients presenting with chest pain and can particularly be useful in initial risk stratification for ACS. Objective evidence of ACS is typically present in only a few patients and is therefore used to stratify patients into higher- and lower-risk groups, thereby enabling the targeting of the appropriate level of additional diagnostic testing.
A review of patient history is typically carried out through a thorough questionnaire. Since patients rarely consider their symptoms to be “pain,” questions are typically targeted to achieve a description of their “discomfort,” the characteristics of which may be useful in risk stratification.

Symptoms described as pressure, tightness, squeezing, or indigestion, or those similar to prior ACS events are considered typical and can be used to identify patients at increased risk for ACS. Atypical symptoms such as stabbing, pleuritic, and pinprick discomfort, are usually associated with a noncardiac etiology and can be used to identify patients at lower risk for ACS [18]. Risk factors for CHD, e.g. low levels of physical activity, use of tobacco, and high levels of alcohol consumption, can predict long-term risk for ACS; however, they are of limited value in the ED for identifying patients with ACS presenting with acute symptoms [19, 20] and are outweighed by other factors such as electrocardiography (ECG) results, history of CHD, and age.

2.3.2 Electrocardiography

The use of electrocardiography (ECG) is regarded as the easiest and most important tool for early risk stratification of ACS. ECG is considered one of the “vital signs” for patients presenting with chest pain, and current recommendations suggest that it should be performed within 10 minutes of initial presentation to the ED [21]. Presence of ST-segment elevation should prompt consideration for immediate reperfusion therapy. ST-segment depression is associated with a marked increase in risk for myocardial infarction (MI) and ischemic complications [22, 23]. An increase of as little as 0.5 mm of ST-segment depression predicts increased risk for MI and death;
this risk increases with increased extent of depression [23]. T-wave inversion is usually considered consistent with ischemia; however, risk is lower than with ST-segment depression [22].

Despite its importance, ECG exhibits a number of limitations, including a relatively low diagnostic sensitivity for ACS, in particular, unstable angina. Ischemic changes are apparent at the time of presentation to the ED in only 20% to 30% of patients who have an acute MI, with the remaining patients showing normal findings on ECG at presentation [24, 25].

2.3.3 Biomarkers

In the event of myocardial injury, biomarkers are released into the blood stream; such biomarkers can be used to assess ACS as well as other categories of CVD. Current recommendations advise that all patients reporting to the ED with suspected ACS undergo cardiac biomarker sampling [21]. Sampling should be performed first at baseline, and again 6 to 8 hours later if baseline data is negative. Troponin is considered to be the standard biomarker for identifying MI because troponin I and troponin T are not present in healthy persons. Troponin exhibits superior sensitivity and specificity compared to other biomarkers and can help identify patients with increased short- and long-term risk of cardiac events [21]. Creatine kinase and creatine kinase MB can also be used; however, because of less than optimal sensitivity and specificity, current recommendations indicate troponin as the preferred biomarker.

2.3.4 Clinical Risk Assessment Score

The thrombolysis in myocardial ischemia (TIMI) risk score is an assessment integrating information from seven risk factors that can be used for stratification of
patients [26]. These risk factors are independent and equally-weighted: (1) cardiac biomarker elevation, (2) ST-segment depression, (3) age $\geq 65$ years, (4) aspirin use within the past week, (5) 2 or more episodes of chest pain in the past 24 hours, (6) 3 or more of the standard CHD risk factors, and (7) known coronary stenosis of 50% or more. Increased TIMI score is associated with increase incidence of death, MI, and need for urgent revascularization. In patients with low TIMI scores (e.g. 0-1), further evaluation is necessary because the event rate in this group is not negligible [27].

2.3.5 Exercise Treadmill Test

Exercise treadmill testing (ETT) is a standard procedure for risk stratification of chest pain [21, 28]. ETT has the advantages of a relatively modest cost, high availability, ease of performance, and the ability to provide important prognostic information. Selection of this test is based on the patient’s ability to exercise and normal findings in baseline ECG that allows interpretation of exercise-induced ST-segment alterations. If ECG findings are not interpretable, ETT may also be combined with imaging techniques. Failure in ETT to reproduce chest pain symptoms prompting the initial ED visit reduces the likelihood of ACS; therefore, ETT can be used as a negative predictive tool. Although the sensitivity of ETT in detecting CHD is lower than that of imaging techniques [29], the relative cost effective of this techniques provides advantages over other techniques [30, 31].

2.4 Imaging-Based Diagnosis of Cardiovascular Disease

In this section, we consider some of the imaging-based techniques for diagnosis and evaluation of CVD.
2.4.1 Computed Tomographic Coronary Angiography

Computed Tomographic Coronary Angiography (CTCA) utilizes computed tomography (CT) techniques to provide anatomical as well as functional information regarding the coronary arteries and is considered to be the clinical gold standard for measuring lumen narrowing (Figure 2.6). Acquisition of CT data is synchronized with a surface ECG, and whole-heart coverage can be achieved within a single heart beat, or within less than one second. Acquisition is typically performed during injection of
iodine-based contrast agents, with whole body radiation dose of at least 1-2 millisieverts (mSv) for a CTCA compared to a typical background radiation dose of 2.4 mSv per year. CTCA can provide anatomical information useful for diagnosis of CHD based on narrowing of lumen. However, it should be noted that lumen narrowing may not necessarily be a sufficient indication of risk for ACS, as sufficient hemodynamic pressure gradient across the stenotic region may render the lumen narrowing physiologically insignificant. In particular, Pijls et al. [32] showed that patients exhibiting visually apparent but functionally nonsignificant stenosis do not benefit from revascularization. Therefore, performing invasive procedures on such patient populations not only is an unnecessary economic burden but also unnecessarily exposes patients to ionizing radiation and procedural risks. This also suggests that techniques that provide not only anatomical information but also functional information may be particularly useful for identifying patients at risk for ACS.

2.4.2 Myocardial Perfusion Imaging with Single Photon Emission Computed Tomography

Myocardial perfusion imaging using nuclear SPECT (Single Photon Emission Computed Tomography) involves the injection of a radioisotope (typically thallium-201 or technetium-99m sestamibi) into a patient. Figure 2.7 shows example MPI images of the heart. The radioisotope is taken up by cardiac myocytes and is distributed within the intracellular region. Gamma ray emission from the radioisotopes is then detected by cameras. Regions with diminished perfusion due to a coronary stenosis tend to exhibit reduced radioisotope uptake by myocytes, leading to visible perfusion defects in MPI that manifests as regions of reduced gamma ray counts. MPI may be performed both at rest and stress. Resting MPI has been shown to
accurately identify low- and high-risk patients presenting with chest pain [33]. While technetium-99m protocols can be performed over the course of two days for rest and stress, comparable diagnostic accuracy can be achieved with one-day rest-stress protocols [34]. One-day protocols can also be performed using different radioisotopes at rest and stress, relying on the different photon energies of thallium-201 and technetium-99m, with the not insignificant tradeoff of a significantly higher radiation dose with dual-isotope protocols. MPI techniques offer the ability to provide a wide variety of diagnostic information pertaining to evaluation of CVD, including myocardial blood flow, myocardial metabolism, and ventricular function. However, MPI using SPECT involves a significant amount of ionizing radiation (10-28 mSv [35]). This radiation dose is in the range where harmful effects are possible and must be taken into account especially in a patient population that is likely to undergo other diagnostic tests and procedures that may involve additional radiation doses. Additionally, MPI images are
of poor spatial resolution (6-10 mm) and can exhibit soft tissue attenuation artifacts [36].

2.4.3 Transthoracic Echocardiography

Transthoracic echocardiography (TTE) utilizes internal tissue reflection of ultrasound waves to generate cine loops of cardiac wall motion and is a valuable tool for assessing cardiac function, an important target for diagnosing CVD. An example image acquired using TTE is shown in Figure 2.8. Ischemia induced regional wall motion abnormalities can be detected using TTE almost immediately after ischemia onset and preceding ECG alteration and symptoms [37]. TTE can adequately identify wall motion abnormalities and therefore predict the presence of coronary artery disease; however, its effectiveness in diagnosis of an acute event is only moderate because such functional abnormalities present themselves also in myocardial infarction and scar tissue [38]. TTE has a number of limitations that can limit diagnostic accuracy in some patient populations, including poor acoustic windows due to large body habitus, prior cardiothoracic surgery, or lung disease, and poor visualization of the apex of the left ventricular posterolateral wall [39]. TTE is also highly dependent on operator skill to acquire proper cardiac views, especially when combined with exercise stress [39]. Additionally, TTE images typically exhibit poor signal-to-noise ratio and tissue contrast, leading to high inter-observer variability in interpreting the images [39, 40, 41].

2.4.4 Cardiac Magnetic Resonance Imaging

Cardiac magnetic resonance (CMR) imaging is a rapidly developing imaging modality that has superior image quality compared to most other noninvasive imaging
Figure 2.8: Example TTE images of the heart, showing the right ventricle (RV), right atrium (RA), left ventricle (LV), left atrium (LA), left ventricular outflow tract (LVOT), and the aortic outflow (Ao). Source: Wikipedia
CMR imaging techniques are particularly useful in the diagnosis and evaluation of various stages of heart failure, and we briefly summarize the utility of a number of CMR techniques in this paragraph. Cine imaging can be used for evaluating cardiac parameters relevant for quantifying cardiac output in addition to qualitatively evaluating regional wall motion and identifying potential regions with perfusion defects or tissue necrosis. Myocardial edema imaging and late post-gadolinium imaging can also be used to identify regions of myocardial injury or fibrosis. This can be very useful in evaluating risk factors for decompensation into ADHF in otherwise stable patients. Additionally, flow measurement via phase contrast techniques can be used to identify significant valvular stenosis or insufficiency and can be useful in evaluating
cardiac workload. We provide a more thorough overview of a number of these CMR techniques in the following chapter.

As discussed in previous sections, the factors leading to and resulting from the development of ADHF are complex and have a large number of physiological effects. We provide here a few remarks that seek to clearly delineate the specific factors that play a role in the use of CMR imaging techniques in this specific patient population.

Firstly, patients at risk for heart failure or presenting with various stages of heart failure may also present with or be at risk for potentially life-threatening arrhythmias. From the perspective of CMR techniques, this introduces two main considerations. First, care must be taken to ensure that external factors do not conspire to increase patient cardiac workload. External factors may include the use of exercise stress techniques, patient claustrophobia, or general patient nervousness throughout the exam process. Such external factors may serve to increase risk for life-threatening arrhythmias during an exam through increased cardiac workload. Additionally, such factors can limit or perhaps even exclude the use of various CMR techniques for patient evaluation. Second, if the patient presents with arrhythmias, resulting irregular cardiac cycle duration can affect technical aspects of CMR imaging techniques - this aspect will be described in more detail later on.

Secondly, patients at risk for heart failure or presenting with various stages of heart failure may present with shortness of breath or be unable to hold their breath for the duration required by some CMR techniques. Additionally, such patients may be severely limited in the number of breath-hold acquisitions they are able to undergo. This may also affect or even exclude the utility of various technical approaches for CMR imaging.
Therefore, from a clinical perspective, patients presenting with various stages of heart failure can exhibit either the presence of arrhythmias or the inability to breath-hold, or both. The presence of arrhythmias results in irregularities in cardiac rhythm, while the inability to breath-hold results in cardiac variation due to respiratory motion. The existence of one or both of these factors eliminates the possibility of using segmented techniques for cine, phase contrast, and late post-gadolinium enhanced CMR techniques.

Compared to other diagnostic imaging modalities, CMR imaging offers the promise of a single, non-invasive test not involving radiation for accurately evaluating cardiovascular morphology, function, perfusion, viability, and flow, and a number of recent studies [49, 50, 51] indicate that this promise is being fulfilled. However, despite growing evidence of its many advantages over existing modalities, CMR does not play a significant role in diagnostic cardiology. A major reason for this is due to the inefficiencies of current CMR image acquisition methods. Currently, most CMR acquisition techniques utilize a segmented acquisition introduced over 20 years ago [52] due to its support for very high spatial and temporal resolution. However, the “one slice per breath-hold” approach utilized in segmented CMR leads to lengthy exam times of up to one hour, making the exam difficult for both the operator and the patient and limiting patient throughput, which can lead to increased cost for diagnosis of CVD using this highly promising imaging modality. Segmented CMR can also be unreliable in patients presenting with cardiac arrhythmias and/or the inability to repeatedly suspend respiration for scans that may last up to 20 seconds long. As a result, the conventional segmented CMR approach is impractical in up to 30% of patient populations. A rapid CMR exam offering excellent diagnostic information while
not requiring patient breath-hold or regular cardiac rhythm would prove cost-effective for improving evaluation diagnosis of cardiovascular disease using imaging modality.

Real-time CMR [53, 54, 55, 10] can address the difficulties of segmented CMR by utilizing accelerated imaging techniques to enable free-breathing acquisition. However, achieving spatial and temporal resolutions comparable to those in segmented CMR can be challenging, and the various techniques for achieving such spatial and temporal resolutions can be computationally time-consuming. This serves as the primary aim for this body of work, i.e. we seek to investigate various approaches for achieving target spatial and temporal resolutions using real-time CMR, identify an approach that allows us to achieve these targets with potential for a fast, efficient implementation, and implement this approach, demonstrating a practical approach for real-time CMR meeting target spatial and temporal resolutions within a clinically reasonable amount of time.

2.5 Concluding Remarks

In this chapter, we have provided some of the clinical and societal motivation for considering the role played by real-time CMR in the diagnosis of cardiovascular disease. The focus of this chapter has been to provide a higher scope picture of the problem at hand. In the following chapters, we provide a more detailed development of the specific topics necessary for addressing the problem posed in this chapter from an engineering perspective.
Chapter 3: Cardiac Magnetic Resonance Imaging for Diagnosis of Cardiovascular Disease

The aim of this chapter is to describe in further detail the use of cardiac magnetic resonance (CMR) imaging in diagnosis of cardiovascular disease (CVD). Cardiac magnetic resonance imaging, and in particular, cardiac cine magnetic resonance imaging, is widely used for evaluation of all forms of CVD, and in this chapter we aim to describe in some detail the role played by CMR in the evaluation of CVD. In this chapter we also shift to introducing the problem not from a medical and societal perspective but from an engineering perspective.

We begin by covering the basics of magnetic resonance imaging (MRI) physics from an acquisition perspective. We continue with an overview of MRI acquisition techniques using in cardiac imaging, including both static and dynamic imaging techniques. We then continue with an overview of reconstruction techniques, with a particular focus on the notion of k-space and methods for image signal recovery from an undersampled k-space. With this foundational understanding of both MRI physics and reconstruction, we review the segmented cine technique and discuss its advantages and drawbacks. We also discuss the real-time, single-shot cine technique and the conventional parallel image reconstruction techniques necessary for making this approach feasible. We also describe the basic signal model for MRI and formulate
these reconstruction techniques in the context of this model. This model will also serve as the foundation for introducing signal recovery within the compressed sensing framework in the following chapter.

3.1 An Introduction to MRI Physics

In this section, we provide an overview of the MRI physics with the goal of providing sufficient background for understanding (1) the basic acquisition techniques used in CMR, (2) the concept of k-space, and (3) the reconstruction of acquired MRI data.

An example of a physical MRI scanner in the clinical setting is shown in Figure 3.1. Schematically, an MRI scanner typically contains the components shown in Figure
Figure 3.2: Diagram of main components of an MRI scanner. (1) Main magnet, (2) Gradient coils, (3) transmit coil (Tx), (4) receive coil array (Rx).
3.2. Most scanners include the following physical components: (1) main magnet, (2) gradient coils \((G_x, G_y, G_z)\), (3) the transmit coil \((Tx)\), and (4) the receive coil array \((Rx)\). C-shaped static magnet designs exist; however, the main magnet is typically a superconducting magnet of toroidal shape, with common clinical field strengths of 1.5T and 3T. The gradient coils in each of the three physical directions \((x, y, \text{ and } z)\), with the convention that the orientation of the main magnetic field pointing in the \(z\) direction) are used to introduce variations (typically linear) to the main magnetic field in the three physical directions. RF energy is transmitted into the object being observed via a single channel, circularly polarized transmit coil that is inserted within the bore of the MRI scanner (Figure 3.1). The receive coil array (consisting of one or more coils) is used to detect energy emitted by the object being observed. Most modern MRI scanners use this arrangement, i.e. a whole body transmitter and a surface receiver array, although transmit coil arrays are available at higher field strengths (i.e. 7.0T).
Figure 3.4: Placement of protons in the presence of an external magnetic field $B_0$ results in the generation of a net magnetization in a direction parallel to the main magnetic field. Source: [3].
Figure 3.3 shows a block diagram illustrating the process of data acquisition using an MRI scanner. By weight, the water content in humans is approximately 60%. Therefore, for clinical imaging in humans, the primary source of signal in MRI is the nuclear magnetic spin of the $^1\text{H}$ nuclei of water, i.e., the proton signal in water. In the following, we briefly describe the measurement process carried out in MRI. When an object being imaged is placed within the bore of the MRI scanner, the nuclear magnetic spins of the $^1\text{H}$ nuclei tend to align along the direction of the main magnetic field. Thermal equilibrium dictates that a non-equal proportion of the spins align in one direction parallel to the field compared to the opposing direction, yielding a net ensemble of spins oriented parallel to the main magnetic field. This macroscopic ensemble of spins forms the primary signal source in MRI (Figure 3.4).

Radio frequency (RF) energy at the resonant frequency of the $^1\text{H}$ nuclei is transmitted into the ensemble of spins, or magnetization. This shifts the magnetization into a higher energy state, corresponding to a “tipping” of the magnetization away from the direction of the main magnetic field (Figure 3.5). While in the quantum
Figure 3.6: Illustration of the longitudinal direction (along the direction of the main magnetic field) and the transverse plane (perpendicular to the main magnetic field). Magnetization along the \( z \)-direction is referred to as \textit{longitudinal} magnetization. Magnetization in the \( xy \)-plane is referred to as \textit{transverse} magnetization. Source: [3].
Figure 3.7: Longitudinal (T1) relaxation. Application of a 90° RF pulse causes the longitudinal magnetization to become zero. Over time, the longitudinal magnetization will grow back in a direction parallel to the main magnetic field. Source: [3].

Figure 3.8: Transverse (T2*) relaxation. Immediately after application of a 90° RF pulse, transverse magnetization is maximized and dephasing begins to occur. The signal from dephasing protons cancel out, reducing the MR signal. Source: [3].
mechanical sense, this notion of “tipping” is not strictly true, it serves as a useful analogy for a conceptual understanding of how the macroscopic ensemble of magnetization is manipulated via RF energy. Quantum mechanically, the notion of “tipping” corresponds to a shift in the probability of measuring one of the two possible energy states of the hydrogen nuclei. This is referred to as a generation of transverse magnetization (Figure 3.6). When transmission of RF energy ceases, the magnetization relaxes, or returns to the direction of the main magnetic field, resulting in the reformation of longitudinal magnetization (Figure 3.7). This pathway for reformation of longitudinal magnetization (and thereby reduction of transverse magnetization) is referred to as T1 relaxation, with relaxation time constant $T_1$ corresponding to the decay of the transverse magnetization generated by the initial RF pulse. Another mechanism by which generated transverse magnetization can decay is through dephasing of the individual spins in the transverse plane (Figure 3.8); this is referred to as $T_2^*$ relaxation. Various mechanisms for this dephasing exist, including spin-spin interactions, magnetic field inhomogeneities, magnetic susceptibility effects, and chemical shift effects. The dephasing due only to spin-spin interactions is referred to as $T_2$ relaxation.

RF energy can be transmitted via the transmit coil through a series of one or more pulses. RF energy is transmitted at the resonant frequency of the main magnetic field, therefore spins that are “on-resonance,” i.e. their local field strength corresponds to that of the main magnetic field, will experience a transmission of energy such that spins are tipped away from the main magnetic field. Spins that are “off-resonance” will experience a reflection of the RF energy since their local magnetic field does not correspond to the main magnetic field, and hence these spins will not be tipped away.
from the main magnetic field. The resonant frequency at which energy is transmitted into the spin system is governed by the Larmor equation (3.1), where \( \gamma \) denotes the gyromagnetic ratio and \( B_0 \) denotes the magnetic field, typically corresponding to the main magnetic field in the absence of gradients. The gyromagnetic ratio is isotope specific; for hydrogen nuclei (i.e. protons), \( \gamma = 42.576 \text{ MHz/T} \). Therefore, for a magnetic field strength of 1.5T, RF energy transmitted at a frequency of 63.864 MHz will result in a transmission of energy into the spin system, leading to a tipping of the ensemble magnetization away from the main magnetic field.

\[
\omega = \gamma \cdot B_0
\]  

(3.1)

The three gradient coils serve to introduce spatial variations in the main magnetic field (in most cases, a linear variation is relevant), and can therefore be used to spatially modulate RF energy transmission. Spatial variations in the main magnetic field adjust the local field experienced by each spin. Therefore, only those spins whose Larmor frequency corresponds to that of the RF pulse being played out (i.e. they are on-resonance) will experience a transmission of energy, leading to generation of transverse magnetization in those spatial locations. For most imaging purposes, gradient waveforms are played out during RF pulses in order to achieve slice or slab selection. Slice or slab selection can be used to select regions of interest where one would like to measure physiological information.

In addition to being used during RF transmission, gradient waveforms are played out during the measurement process in order to acquire the raw MR data. Individual spins not aligned with the direction of the main magnetic field (as in after an RF pulse) will precess around the main magnetic field due to the net force exerted by the field
Figure 3.9: The concept of precession. Precession of a spinning top and a nuclear spin are similar in that an external force combined with spinning motion causes precession around the direction of the external force. Source: [3].
The precession frequency depends on the local field strength experienced by a particular spin and is again governed by the Larmor equation. Therefore, the playing out of linear gradient waveforms imposes a relative difference in accumulated phase between spins within a volume of interest. The concept here is comparable to that used in optical interferometry, with the exception that RF wavelengths are used here.

Due to precessional frequencies that vary spatially with local field strength, the playing out of linear gradient waveforms while acquiring data via an analog-to-digital converter (ADC) allows for the encoding of spatial information in the signal measured by the receiver coil array. Gradient waveforms played out over time result in spatially and temporally varying accumulation of phase in the spins due to their precessing with frequency proportional to the local magnetic field strength. Relative phase accumulation across the object being imaged allows for amplitude and phase measurements of the spatial frequency components of proton signal density in the object. Hence, the raw MR data is acquired in the spatial Fourier domain, commonly referred to as the \emph{k-space domain}. In the k-space domain, each individual complex-valued pixel represents a spatial sinusoidal plane wave or oscillation in proton signal density. The magnitude component represents the amplitude of the plane wave, and the phase component represents the phase offset of the plane wave. Distance from the origin determines the frequency of this plane wave oscillation, and the direction of the vector from the origin to the pixel determines the orientation of this plane wave oscillation (Figure 3.10).

The diagram showing the amplitudes and timings of the RF pulses and gradients in addition to when the signal is measured by the analog/digital converter is called
Figure 3.10: Example of a full k-space frame (left column, magnitude component shown), and different specific k-space points (center column, white pixel). K-space points at different locations correspond to plane wave oscillations in the image (signal intensity) domain (right column).
Figure 3.11: Spins moving along an external magnetic field gradient acquire a difference in the phase of their rotation (right), whereas stationary spins do not (left). The accumulated phase difference is proportional to the velocity of the moving spin. $t = \text{time}, V = \text{velocity}, \Phi = \text{phase shift}$. Source: [4]

the sequence diagram, and the design of the RF pulses and gradient waveforms for encoding physiologically useful information is referred to as pulse sequence design.

3.2 An Overview of the Basic Acquisition Techniques Used in Cardiac Imaging

In the following sections, we present a brief overview of the various techniques for applying CMR to evaluate cardiac morphology, function, perfusion viability, and flow. We also mention some of the challenges posed in terms of image reconstruction. The focus of this work is specifically on cine imaging; however, we provide a description of the other major CMR techniques in order to illustrate the potential utility of CMR in providing a wide variety of diagnostic information in a single patient study.

3.2.1 Velocity Imaging

CMR can be used to quantify blood velocity and flow through the use of phase contrast MRI (PC-MRI) [56, 57, 58, 59]; these dynamic imaging techniques can be useful for evaluating left ventricular filling and valvular function. PC-MRI is based
Figure 3.12: Basic principle behind most clinically used phase-contrast sequences. Two acquisitions are performed, each with all parameters kept constant except for the flow-sensitizing bipolar gradients. The data of the two acquisitions are subtracted. The effective flow encoding is achieved by means of the difference in the bipolar gradients of the two acquisitions. This technique eliminates all phase shifts induced by imaging gradients. $t = $ time, $V = $ velocity. Source: [4]

Figure 3.13: The phase-contrast measurement generates two sets of dynamic images. The magnitude image (left) resembles a normal bright-blood image and is used for anatomic orientation. In the velocity image (right), the gray scale value of each pixel represents the velocity information in that voxel. Black values show flow toward the viewer, whereas white values show flow away from the viewer. Source: [4]
Figure 3.14: A typical first-pass perfusion imaging pulse sequence. Cardiac-gated acquisitions are performed by using an electrocardiographic signal. Multiple sections are acquired consecutively (e.g., six sections every two R-R intervals, as shown here). This is repeated continuously during the first pass and washout of the contrast agent. $T_1$ contrast is generated by using a saturation preparation (Prep) pulse followed by fast acquisition of each section using a GRE, GRE-EPI, or SSFP sequence. Source: [5]

on the observation that spins moving through a magnetic gradient can accumulate a phase difference that is proportional to the velocity of the spin (Figure 3.11). Bipolar gradients are used to encode this spatial motion of spins within the phase of the signal measured by the receiver coil array (Figure 3.12). PC-MRI generates a dynamic series of complex-valued images, with the magnitude component showing physiological structure and the phase component encoding velocity measurements (Figure 3.13). At each spatial location, velocity measurements may be acquired in a single direction or in all three Cartesian directions. Depending on whether one, two, or three velocity directions are measured, the amount of data needed to be collected can be very large, motivating the need for accelerated image acquisition [60] in order to maintain adequate or even improve spatial and temporal resolutions.
3.2.2 Perfusion Imaging

CMR can be used to quantify myocardial perfusion through the use of perfusion imaging [61, 62, 63, 64]. In perfusion imaging, snapshots of multiple slices or a volume of the heart are taken at a specific phase of the cardiac cycle over multiple heart beats (typically 40-50 heart beats) beginning shortly after gadolinium contrast injection. Perfusion imaging is a dynamic imaging technique, with the goal being to capture the perfusion dynamics as the contrast agent initially perfuses the myocardium (first pass) as well as shortly thereafter (wash out). Figure 3.14 shows the basic sequence diagram for a typical perfusion sequence. Snapshot images for multiple slices are acquired over multiple heart beats using either a gradient echo (GRE), gradient echo with echo planar imaging (GRE-EPI), or steady-state free precession (SSFP) sequence. Imaging is typically performed at rest and at stress in order to identify perfusion defects. Adequately perfused myocardium will show increased signal intensity with
arrival of contrast agent, and poorly perfused tissue will show reduced signal intensity with delayed arrival of contrast agent. Therefore, perfusion imaging, particularly at stress, can differentiate ischemic myocardium from normally perfused tissue. Figure 3.15 show example images from a perfusion cine both at rest and post adenosine stress. Due to the need to image a large region of the heart in a snapshot duration (e.g. a single cardiac phase), perfusion imaging greatly benefits from accelerated imaging techniques [65].

3.2.3 Late Gadolinium Enhancement

Static imaging techniques based on late gadolinium enhancement (LGE) [66, 67, 68], or delayed enhancement imaging, can be used to identify regions of myocardial infarction as well as other inflammatory or infectious diseases of the myocardium. In LGE imaging, a gadolinium contrast agent is injected and imaging using a $T_1$
Figure 3.17: Acute idiopathic myocarditis in a 30-year-old man with frequent premature ventricular contractions and normal results at coronary angiography. Two-chamber long-axis MR image of the left ventricle (a) and short-axis MR image (b) show extensive scattered delayed enhancement in the anterior, lateral, and inferior wall and apex of the left ventricle and the right ventricular wall (arrows). Source: [6]

A weighted sequence is performed 10-30 minutes after contrast injection. The utility of this approach is based on the observation that compared to normal myocardium, abnormal myocardium due to disease can exhibit different $T_1$ relaxation times due to retained contrast agent (Figure 3.16a). This leads to shortening of $T_1$ relaxation times and therefore increased signal intensity in $T_1$ weighted images. Adequately timing the inversion pulse (Fig. 3.16b) can maximize signal contrast between normal and abnormal myocardium. Figure 3.17 shows example LGE images in the two-chamber long-axis view and the short-axis view.

### 3.2.4 Parameter Mapping Techniques

As seen from LGE imaging, physiological changes in myocardial tissue can have effects on spin relaxation times, which can lead to diagnostically useful qualitative
Figure 3.18: Example of T1 mapping with a Look-Locker (LL) MR imaging sequence in a healthy 34-year-old man with normal myocardium. Source: [7].

(a) LL (inversion time [TI] scout) image series performed at 1.5T show 21 images acquired at a fixed location (short-axis, mid-ventricle) at different TIs ranging from 75 ms (top left) to 695 ms (bottom right), with a 30 ms increase in TI between images. Images were acquired after injection of gadolinium contrast agent. (b) T1 map displays a color scale of the uniform T1 values of the myocardium. (c) Graph of a multiparameter curve-fitting analysis of a selected pixel in the interventricular septum shows how each value is used to derive the contrast-enhanced T1 value (299 ms) for this pixel.
Figure 3.19: Example of T1 mapping in a 37-year-old woman with a lateral wall infarction. Source: [7]. Short-axis MOLLI (1.5T) T1 mapping images acquired 26 minutes after gadolinium-based contrast agent administration show a focal area of decreased T1 values in the lateral wall of the left ventricle.
changes in signal intensity. Static parameter mapping techniques seek to quantitatively measure the decay time constants for these relaxation processes, yielding additional diagnostically useful information. More specifically, $T_1$, $T_2$, and $T_2^*$ parameter mapping techniques seek to obtain a pixel-wise quantitative map of decay constants, providing a potentially robust approach for quantitatively characterizing myocardial disease onset and progression [69, 70, 71, 72, 73]. In $T_2$ and $T_2^*$ mapping, snapshot images are acquired at specific echo times, and a pixel-wise curve fitting is performed to obtain time constants. In $T_1$ mapping, the decay time constant for the magnetization to reform along the main magnetic field is measured. An example of the $T_1$ mapping process is shown for a healthy patient in Figure 3.18. An example $T_1$ mapping is shown in Figure 3.19 for a patient with a myocardial infarction. Typically, very basic acceleration of image acquisition via parallel imaging (e.g. GRAPPA [74] or SENSE [75]) is used; however, compressed sensing techniques have been proposed for improving parameter mapping techniques [76].

3.2.5 Cine Imaging

The most commonly used technique for evaluating cardiac function with CMR is cine imaging. Cine imaging is a dynamic imaging technique that refers to the capturing of a series of images of the same slice or slab over a period of time such that this image series adequately characterizes motion of anatomical features within the region. In order to adequately capture dynamic features using cine imaging, adequate spatial and temporal resolution is required. Although 3D single slab techniques are possible, cine imaging typically consists of multiple single slice acquisitions consisting
Figure 3.20: Example high spatial resolution ($1 \times 1 \text{ mm}^2$) and temporal resolution (11.2 ms) segmented breath-held cine images in the short axis (a, b), two-chamber (c, d), and four-chamber (e, f) orientations at end systole (left column) and end diastole (right column) acquired from the author’s heart with a 45 second breath-hold.
of a short axis stack of slices as well as two- and four- chamber and other long axis slice orientations (Figure 3.20).

Cine imaging can be used to generate a number of both qualitative and quantitative features useful in evaluation of disease onset or progression. Standard short- and long- axis views can be used to identify regions of ischemic or non-viable myocardium through identification of left ventricle (LV) wall segments with abnormal or impaired contraction during systole (the phase of the cardiac cycle in which the LV is maximally contracted). Qualitative evaluation of relaxation during diastole (the phase of the cardiac cycle in which the LV is maximally relaxed) can also be obtained. Inadequate relaxation during diastole can be indicative of diastolic dysfunction, and inadequate contraction during systole can be indicative of systolic dysfunction. Both conditions can be used for diagnosis of various stages of heart failure. Volumetric parameter measurements derived from cine images at the systolic and diastolic phases of the cardiac cycle can also be useful diagnostically. End diastolic volume (EDV) refers to the volume of blood contained in the LV at end diastole, and end systolic volume (ESV) refers to the volume of blood contained in the LV at end systole. The stroke volume (SV) refers to the difference between EDV and ESV. The ejection fraction (EF) is given by the ratio (EDV-ESV)/EDV and is a measure of how well the LV is pumping oxygen-enriched blood to the rest of the circulatory system. The volumetric parameter measurements derived from cine images are of great clinical utility and also depend heavily on adequate spatial and temporal resolution of the acquired cine images. Quantitative analysis using cine images can provide stress and strain measurements for the myocardium. Cine imaging of the valves is helpful in identifying various morphological diseases of the valves. Identification of key anatomical features
captured by cine imaging in the standard views is useful in diagnosis of a wide variety of congenital heart diseases.

Typical cine acquisitions use the balanced steady-state free precession (SSFP) pulse sequence, which is illustrated in Figure 3.21. This sequence has high signal-to-noise ratio due to a short echo time, and good blood-to-myocardium contrast.

Cine imaging is the primary focus of this body of work. However, many of the techniques proposed and discussed here may be relevant to other (dynamic as well as static) CMR techniques. We allocate a small portion of space later in this work to some studies investigating reconstruction methods applied to some of the other CMR techniques. We now proceed to cover more specific details from an engineering and signal processing perspective related to image reconstruction of cardiac cine images.
Because of the serial way in which gradient waveforms must be played out temporally, MRI exhibits the unique characteristic compared to other imaging modalities in that the raw data, commonly referred to as k-space, measured in the spatial Fourier domain must be sampled sequentially, i.e. point by point. In contrast, with an X-ray machine, the signal intensity at all spatial locations is measured simultaneously upon emission of radiation by the signal source.

The manner in which the gradient waveforms are played out determines the sampling trajectory in k-space. In this work, we are primarily concerned with trajectories in two spatial dimensions, although the work proposed can readily be extended to
Figure 3.23: Characteristic undersampling artifacts (top) and point spread functions (PSF) (bottom) for several k-space sampling trajectories. The fully sampled k-space reference is shown in the left column. The other images show artifacts from an undersampled Cartesian trajectory and its PSF (middle left), undersampled radial trajectory (middle right), and undersampled variable density spiral trajectory (far right). The PSF can help to give some indication as to the form of aliasing introduced due to the particular undersampled k-space trajectory. Source: [9]
three spatial directions. In two spatial dimensions, the most common k-space trajectory is a Cartesian trajectory (Figure 3.22, left), in which k-space points are sampled on a Cartesian grid. Non-Cartesian trajectories are also used in MRI, e.g. radial (Figure 3.22, middle) and spiral (Figure 3.22, right). Figure 3.23 shows the aliasing artifacts introduced for various k-space undersampling trajectories, as well as the corresponding point spread functions, which can be used to characterize these artifacts and can be useful in evaluation of reconstruction approaches. Cartesian trajectories exhibit the particular advantages of allowing a rectangular field-of-view (FOV) as well as robustness against trajectory inaccuracy. Radial trajectories can be advantageous in dynamic scenarios in that every k-space line traverses the center region, thereby encoding low frequency motion information. Spiral trajectories can provide for more efficient coverage of k-space, which can be beneficial for dynamic imaging scenarios. In this work, we focus primarily on Cartesian trajectories. With an additional gridding step, this work can be readily extended to non-Cartesian trajectories as well.

According to studies by Setser et al. [10], the minimum sampling rate for adequately sampling the cardiac motion at rest using the Nyquist criterion corresponds to a temporal resolution of around 40 ms (Figure 3.24), with increased temporal resolution for higher heart rates. Therefore, clinical guidelines specify that cardiac cine imaging for adequate characterization of wall motion and cardiac function should achieve a temporal resolution of better than 45 ms and a spatial resolution of better than $2.5 \times 2.5$ mm$^2$ [77, 78]. It should be noted that, since the extent of traversal in k-space determines spatial resolution, there exists an inherent tradeoff between temporal and spatial resolution.
In order to achieve adequate spatial resolution and temporal resolution at typical image sizes of approximately $160 \times 120$, it is not possible to acquire all the necessary k-space data for the reconstruction of a single cardiac phase in a single heart beat using a straightforward Fast Fourier transform (FFT). In the following two sections, we describe two techniques that are used clinically to address this issue.

### 3.4 Segmented Cine Acquisition

In segmented cine imaging [52, 79], the acquisition of the necessary k-space data for reconstruction by FFT is divided across multiple heart beats. Figure 3.25 shows an illustration of this multiple heart beat acquisition scheme. This immediately introduces the constraints that each heart beat must be similar, i.e. the R-to-R interval, or the duration of each heart beat or cardiac cycle, must be similar. Additionally,
Figure 3.25: Illustration showing segmented acquisition across five heart beats (5 segments). During each heart beat (or cardiac cycle) a particular segment of k-space corresponding to a specific cardiac phase is collected. The duration of each cardiac cycle (R-R interval) must remain constant between cycles and the respiratory motion between cycles must be suspended to produce a useful image series.
the respiratory motion must in some way be suspended, i.e. the respiratory phase must be held constant across heart beats. Therefore, it becomes immediately apparent that segmented techniques will fail in the presence of arrhythmias (irregular cardiac rhythm) and in the presence of breathing motion. Hence, segmented cine is performed under breath-held conditions (typically end-expiration).

During the temporal footprint for a particular cardiac phase, segmented cine acquires a segment of the k-space for that specific cardiac phase (Figure 3.25). Under breath-held conditions (Figure 3.26), segments are acquired at approximately the same respiratory phase; hence different k-space segments for a specific cardiac phase acquired across multiple heart beats can be pieced together to form a complete cardiac phase. However, in the presence of free-breathing motion (Figure 3.27), segments from multiple heart beats come from different respiratory phases. Therefore, combining segments from multiple heart beats with differing respiratory phases will generate non-diagnostic images (Figure 3.28). In the presence of arrhythmias, variation in the
duration of each cardiac cycle will result in further reduction in the image quality of segmented cines.

By increasing the number of cardiac cycles over which a segmented acquisition is performed (i.e. increasing the breath-hold duration), high temporal and spatial resolution cine images can be acquired. Directly reducing the number of k-space lines acquired per segment (and thereby increasing the number of heart-beats needed to traverse the entire k-space for a cardiac phase) can increase the temporal resolution. Increasing k-space coverage can improve spatial resolution, at the cost of temporal resolution. As a result, segmented cine acquisition is commonly used clinically due to its excellent support for high spatial and temporal resolutions.

### 3.5 Real-time, Single-Shot Cine Imaging

The number of patients either exhibiting arrhythmias or unable to breath-hold can be significant. In a recent survey conducted at the Ross Heart Hospital at the Ohio State University over a six month period, 221 out of 933 (23.7%) adult cardiac
Figure 3.28: Long-axis cardiac MR images acquired in the same 6-mm-thick section during the same phase of the cardiac cycle with different balanced SSFP sequences. (a) Image obtained with a standard breath-hold segmented balanced SSFP technique shows a marked artifact and an indistinct border between the myocardium and the left ventricular cavity because of breathing during image acquisition. (b) Image obtained with a real-time balanced SSFP parallel acquisition technique (time-domain SENSE) with prospective gating shows the elimination of the motion-related artifact. The endocardial contour is blurred, mainly because of a larger pixel size and poorer temporal resolution. Contrast between myocardium and blood also is inferior to that in (a) because of the larger pixel size and lower flip angle. Source: [11].
patients presented with either severe arrhythmias or the inability to breath-hold, and as a result, real-time, single-shot imaging was used to conduct cardiac studies instead of segmented cine imaging.

Provided techniques allowing reconstruction of real-time, single-shot image acquisitions [80, 53], cine imaging in the presence of arrhythmias and under free-breathing conditions can be made possible. Real-time, single-shot imaging utilizes accelerated imaging to acquire snapshot images fast enough to escape the effects of cardiac and respiratory motion, thereby addressing the difficulties of segmented cine techniques by allowing continuous free-breathing and being impervious to cardiac arrhythmias. In real-time imaging, only the necessary k-space lines for reconstruction of a given cardiac phase are acquired within the temporal footprint of that cardiac phase (Figure 3.29). Hence, within a single heart beat, all k-space data necessary for reconstruction of each cardiac phase is collected; this is in contrast to segmented imaging, where the k-space data is distributed across multiple heart beats.

For cine imaging using SSFP, the typical time (repetition time, or $T_R$) to acquire one line of k-space for a Cartesian trajectory ranges from 2-4 ms, with a shorter $T_R$ being better for purpose of temporal resolution, efficiency, and to avoid artifacts due to SSFP acquisition. Therefore, in order to achieve real-time, single-shot imaging, not all k-space lines for a given cardiac phase can be acquired if a reasonable temporal resolution is to be achieved. Hence, some form of advanced image reconstruction is needed to estimate or interpolate the missing unacquired k-space data to produce a diagnostically relevant image. From a general signal processing perspective, the focus of this work pertains to reconstruction methods for cardiac cine imaging, in
Figure 3.29: Illustration showing a real-time acquisition across five heart beats. During each heart beat (or cardiac cycle) only the necessary k-space data for reconstructing each specific cardiac phase is collected during the temporal duration of the cardiac phase. The time to collect the k-space data for a specific cardiac phase determined the temporal resolution of the resulting cine and is assumed to be short enough so as to “freeze” any cardiac motion.
particular, efficient techniques designed to achieve a practical implementation in the clinical setting.

In the next few sections, we will cover the basic, commercially available parallel imaging techniques for image reconstruction of dynamic real-time cardiac cine, with a particular focus on developing the signal model necessary for introducing compressed sensing techniques. It is important to note here that current parallel imaging techniques yield inadequate spatial and temporal resolution (Figure 3.28), leading either to significant blurring due to low spatial and temporal resolution or increased noise levels and artifacts due to inadequate reconstruction at higher acceleration rates. Previously proposed compressed sensing techniques can achieve the necessary spatial and temporal resolution for real-time cardiac cine; yet these techniques are often accompanied by impractically long reconstruction times.

3.6 Parallel Image Reconstruction with Multiple Coil Arrays - An Overview of SENSE, GRAPPA, and SPIRiT

Parallel image reconstruction of MRI data became feasible only with the introduction of multiple coil arrays [81, 12, 82, 75, 83, 84, 74]. The use of multiple coil arrays enables the acquisition of redundant data channels, each modulated by individual coil sensitivity maps. It is this variation in coil sensitivity maps that provides the physical model for the introduction of parallel MRI (pMRI) reconstruction techniques such as SENSE and GRAPPA.

Intuitively, both SENSE and GRAPPA reconstruction techniques stem from the observation that individual coil elements within a coil array exhibit a spatially varying sensitivity to the RF energy emitted by nuclear spins that can be used to invert the aliasing introduced by sub-sampling of k-space (or sampling at a frequency lower than
Figure 3.30: Some possible linear combinations of component coil sensitivities for an example eight-coil phased array, approximating several spatial harmonics of frequency $m\Delta k_y$. Weighted individual coil sensitivity profiles are depicted as thin solid lines beneath each component coil. Dashed lines represent the sinusoidal or cosinusoidal weighting functions. Combined sensitivity profiles are indicated by thick solid lines. These combined profiles closely approximate ideal spatial harmonics across the array. Source: [12].
Figure 3.31: Component coil sensitivity information for three-element and four-element arrays. (a) Absolute magnitude images. (b) Sensitivity along the diameter indicated by dashed lines in (a). (c) Absolute magnitude images of the abdomen and thorax of a healthy volunteer. (d) Absolute magnitude sensitivity along the dashed lines in (c). Source: [12].
the Nyquist frequency). Sodickson and Manning recognized that a weighted linear combination of the coil sensitivities can be used to construct the spatial harmonic components in the image domain that correspond to k-space pixel values [12] (Figure 3.30). In their work, coil sensitivity maps were manually extracted from reference acquisitions and used to reconstruct separate sub-sampled k-space data. An example of their coil sensitivity map estimation is shown in Figure 3.31.

The signal model for pMRI may be motivated from a mathematical perspective as follows. We seek to recover a signal $x \in \mathbb{C}^N$ from a noisy signal $y \in \mathbb{C}^M$ based on the following signal model:

$$y = Ax + n,$$

(3.2)

where $A \in \mathbb{C}^{M \times N}$ is some measurement operator determined by the physics of the MRI measurement. Therefore, the general goal of pMRI techniques such as SENSE [75] and GRAPPA [74] is to recover the signal $x$ given the measurements $y$. In other words, we seek to invert the operator $A$ in the presence of noise.

With SENSE, Pruessmann et al. [75] recognized that modulation of image signal intensity by individual coil sensitivity information can be used to reverse the aliasing introduced by Fourier undersampling. From a signal model perspective, we can define an operator $C$ which takes the coil-combined image data (i.e. the signal intensity of the physical object itself) and modulates this data (pointwise multiplication) by the sensitivity maps for each of the $n_c$ coils, thereby generating coil-by-coil data. A subsampled Fourier transform operator can then be used to generate the measured signal $y$ in k-space for each coil. Hence, for SENSE, the signal model can be expressed
in more detail as the following:

$$y = F_s C x_s + n,$$  \hspace{1cm} (3.3)

where $x_s$ is the coil-combined data and $y$ is the multiple coil measured data.

With GRAPPA, Griswold et al. \cite{74} recognized that the coil sensitivity maps are spatially smoothly varying. Therefore, the pointwise modulation by the coil sensitivity maps in the image domain corresponds to convolution with some kernel of finite support in the k-space domain. GRAPPA recognizes that each unacquired point may be a linear combination of neighboring acquired points across all coils. This eliminates the need to explicitly determine the coil sensitivity maps; instead convolution kernels are estimated. For GRAPPA, the signal model may be expressed in a slightly different manner:

$$y = F_s x_m + n,$$  \hspace{1cm} (3.4)

where $F_s$ is our previously defined subsampled Fourier transform operator and $y$ is our coil-by-coil measured data. Unlike the SENSE signal model which seeks to reconstruct the coil-combined data $x_s$, the GRAPPA model seeks to reconstruct the coil-by-coil input data, as denoted by $x_m$. This is in contrast to SENSE techniques, where the coil-combined source data is reconstructed. While the coil sensitivity maps are not explicitly used in the signal model for GRAPPA, they are implicitly used in the estimation of the GRAPPA convolution kernels. We present the different signal models in (3.3) and (3.4) to highlight this difference. Since the physics behind the MRI measurements in both cases remains the same, the general MRI signal model presented in (3.2) can also be used without loss of generality.
SENSE and GRAPPA were the initially proposed techniques that served to help introduce real-time, single-shot imaging into the clinical environment, and both techniques are commonly available on commercial MRI scanners. Because of its close relation to GRAPPA, we briefly mention another pMRI technique termed SPIRiT [85]. GRAPPA seeks to estimate convolution kernels such that each unacquired point is a linear combination of a neighborhood consisting only of acquired points. In contrast, SPIRiT seeks to estimate dense kernels, such that the data at a given k-space location (whether acquired or unacquired) can be written as a linear combination of a neighborhood consisting of both acquired and unacquired points.

3.7 Image Reconstruction from a Linear Algebra Perspective

In considering pMRI techniques and their practical implementation, it is helpful to develop a linear algebra perspective for solving the pMRI reconstruction problem, i.e. finding $x$ given $y$ in (3.2). In this section we briefly cover some of the fundamental items related to this problem from a linear algebra perspective.

In basic linear algebra, the problem of solving $y = Ax$ is a frequently considered problem. If $A$ is square and non-singular, then an inverse $A^{-1}$ exists, and the solution is found in a straightforward manner by multiplying both sides by the inverse:

$$x = A^{-1}y.$$  \hfill (3.5)

However, if $A$ is non-square (e.g. for the pMRI problem where typically $M \neq N$), then the direct inverse does not exist. Provided the inverse of $A^H A$ exists, the Moore-Penrose inverse, or pseudo-inverse, may then be used, leading to the following
approximate solution:

\[
\tilde{x} = (A^H A)^{-1} A^H y, \tag{3.6}
\]

where \(A^H\) denotes the Hermitian conjugate of \(A\). The pseudo-inverse solution corresponds to the least-squares solution. Given the model in (3.2), this is what is typically used to perform either a SENSE or GRAPPA reconstruction. For a SENSE reconstruction, this form of inversion is used to estimate the unfolding matrix for inverting the aliasing due to k-space undersampling. For both GRAPPA and SPIRiT, this form of inversion is used for estimation of the convolution kernels in k-space.

### 3.8 Image Reconstruction from an Optimization Perspective

In general, the model generated by MR physics motivates the reconstruction problem, and this reconstruction problem can be reformulated as an optimization problem, to which optimization techniques can be applied to obtain an optimal solution. This hierarchy is illustrated in Figure 3.32.

From an optimization perspective, we can formulate the pMRI reconstruction problem as the task of finding the \textit{optimal} solution \(\tilde{x}\) that minimizes some error, or cost function. For the general MRI signal recovery problem, i.e. for some measurement operator \(A\), this may be expressed as follows:

\[
\tilde{x} = \arg\min_x \| A x - y \|_2^2. \tag{3.7}
\]

In other words, we seek to find the best estimate of \(x\) that minimizes the \(\ell_2\) error (or Euclidean distance) compared to the measured data \(y\). For a SENSE reconstruction, this may be expressed in more detail as the following:

\[
\tilde{x}_s = \arg\min_{\tilde{x}_s} \| F_s C \tilde{x}_s - y \|_2^2, \tag{3.8}
\]
Figure 3.32: Schematic illustrating the different perspectives for understanding the MR image reconstruction problem. MR physics motivates the formulation of a model for the measurement process, which then leads to a reconstruction problem. The reconstruction problem can be reformulated as an optimization problem, which suggests the use of specific optimization techniques to obtain an optimal solution to the MR image reconstruction problem.
where $F_s$ is the subsampled Fourier transform operator and $C$ is the coil sensitivity map operator as defined previously.

K-space-based approaches utilizing convolution kernels can likewise be formulated as optimization problems. For GRAPPA, we can express the reconstruction as a constrained optimization problem:

$$
\hat{k}_m = \arg\min_{k_m} \| (G - I)k_m \|_2^2 \quad \text{s.t.} \quad F_s k_m = y, \quad (3.9)
$$

where $G$ corresponds to the GRAPPA convolution kernels.

Similarly, for SPIRiT, we have the following optimization problem:

$$
\tilde{k}_m = \arg\min_{k_m} \left[ \| F_s k_m - y \|_2^2 + \lambda \| (G - I) x \|_2^2 \right], \quad (3.10)
$$

where $G$ now denotes the dense SPIRiT convolution kernels and $\lambda$ is a tuning parameter adjusting between the contributions from the data fidelity and self-consistency terms. Note that all of these problem formulations involve quadratic terms and therefore can be expressed as some form of the general optimization problem presented in (3.7).

In their unregularized forms, both SENSE and GRAPPA reconstructions can be computed analytically to obtain a closed-form solution in a clinically reasonable amount of time. Therefore, both SENSE and GRAPPA are commonly used clinically for reconstruction of accelerated MRI data. In contrast, due to the dense SPIRiT kernel, it can be extremely memory intensive to obtain the closed-form solution for the SPIRiT reconstruction problem. Therefore, iterative approaches are typically used to obtain the SPIRiT solution. This iterative approach can lead to long reconstruction times, and as a result, SPIRiT has yet to see widespread use in the clinical environment.
provided accurate estimation of coil sensitivity maps, SENSE can produce images with optimal signal-to-noise ratio (SNR) [86]. However, errors in estimation of coil sensitivity maps can lead to reconstruction errors that exhibit themselves as coherent ghosting artifacts. This can become an issue especially in dynamic imaging, where motion between frames can introduce errors in coil sensitivity map estimation. Hence the quality of SENSE reconstruction can suffer particularly in scenarios where severe motion is present or where it is otherwise difficult to obtain an accurate estimation of coil sensitivity maps.

K-space-based approaches based on estimation of convolution kernels (e.g. GRAPPA and SPIRiT) can be more robust in these scenarios, generating images that, while exhibiting slightly lower SNR, do not exhibit coherent ghosting artifacts.

3.9 Parallel Image Reconstruction with Tikhonov Regularization

The basic formulation for image reconstruction using GRAPPA and SENSE yield the least squares solution in the absence of noise. In the presence of noise, or in scenarios where the acceleration rate is high, it may be difficult to obtain the least squares solution for the image reconstruction problem, yielding a solution that may exhibit excess noise or high artifact levels. In this case, Tikhonov regularization can help the reconstruction. For the SENSE formulation, adding Tikhonov regularization yields the following optimization problem:

\[
\tilde{x}_s = \arg\min_{x_s} \| F_s C x_s - y \|_2^2 + \lambda \| x_s \|_2^2.
\] (3.11)

The Tikhonov regularized formulations for GRAPPA and SPIRiT likewise include this additional quadratic term. As the regularization term for Tikhonov regularization is
quadratic, it does not affect the existence of an analytical closed-form solution for SENSE and GRAPPA.

3.10 Inadequacy of Current Parallel Imaging Techniques for Real-Time Single-Shot Cine Imaging

As mentioned previously, adequate spatial and temporal resolutions are necessary for cine imaging using CMR to be able to adequately characterize cardiac morphology, function, and wall motion. Current guidelines recommend an in-plane spatial resolution of at least $2.5 \times 2.5 \text{ mm}^2$ with a temporal resolution of better than 45 ms to adequately resolve wall motion. Imaging at conditions other than rest, e.g. in exercise stress imaging [87, 88], can require higher spatial and temporal resolutions as the heart rate under peak stress can more than double for a human heart.

Segmented cine techniques provide an excellent framework for achieving these target spatial and temporal resolutions; however, they operate within the breath-hold paradigm, limiting CMR utility in a significant population. Additionally, breath-hold scans can be time-consuming, leading to long scan times, especially in uncooperative patients. This reduces patient throughput and has reduced the cost effectiveness of CMR, despite its demonstrated clinical utility.

However, within the paradigm of pMRI techniques based on SENSE and GRAPPA, real-time, single-shot cine imaging approaches remain inadequate in achieving satisfactory spatial and temporal resolutions. Therefore, despite its promising clinical utility, real-time, single-shot cine imaging is still not routinely used in the clinical setting, and when it is used, it is often used as a method of last resort. Since spatial and temporal resolutions can be typically traded, the effectiveness of a pMRI technique can be quantified in general from the acceleration rate $R$, where $1/R$ corresponds to
the fraction of data acquired. For example, for cine data collected at acceleration rate 2, only one half of the necessary data for generating the MR images is actually acquired. Clinical protocols for real-time, single-shot imaging using GRAPPA typically achieve acceleration rates 2-3; higher acceleration rates yield unacceptable image quality. For exercise stress cine imaging, the acceleration rates typically do not exceed rate 3, due to excessive breathing motion and high heart rates.

The inadequacy of current pMRI reconstruction techniques for real-time, single-shot cine imaging serves as the primary motivation for this body of work. As an example, acceleration rates ranging from 8-10 would be required to achieve the above mentioned $2.5 \times 2.5$ mm$^2$ in-plane spatial resolution with a temporal resolution of 45 ms, assuming a $340 \times 340$ mm$^2$ field of view and a 3 ms T$_R$. We note briefly that single-shot techniques are also used in other CMR imaging techniques – including both static and dynamic techniques; therefore, the approaches discussed in this work are also relevant to these other techniques.

In the subsequent chapters, we introduce our approach for efficiently utilizing compressed sensing techniques for improving reconstruction image quality at higher acceleration rates, thereby paving the way for real-time, single-shot cine imaging at higher spatial and temporal resolutions. We briefly highlight that we seek efficient approaches that yield a practically feasible reconstruction in terms of computation time in order to allow for clinical deployment.
Chapter 4: Fundamentals of Signal Recovery with Compressed Sensing for Reconstruction of Highly Accelerated Dynamic Cardiac MRI

In this chapter, we cover some of the relevant topics for implementing signal recovery using the compressed sensing (CS) framework for the pMRI problem posed in the previous chapter. The primary aim of this chapter is to provide sufficient background specifically related to CS for the description of our efficient CS-based approach for reconstructing highly accelerated dynamic cardiac cine MRI in the following chapter.

We begin by first presenting fundamental concepts pertaining to the CS framework, starting with the signal model introduced in the previous chapter. We continue by presenting a brief overview of some of the numerous optimization algorithms that can be used for implementing CS. Finally, we conclude with some discussion pertaining to k-space sampling trajectories amenable to introducing incoherence as well as the specific k-space sampling trajectory used in this work.

4.1 Fundamental Concepts for Compressed Sensing

As developed in the previous chapter, the reconstruction problem for MRI can be expressed in the context of the following signal model:

\[ y = Ax + n, \]  \hspace{1cm} (4.1)
where \( A \in \mathbb{C}^{M \times N} \) is some measurement operator that in the context of MRI typically consists of some subsampled Fourier operator as well as modeling the use of a multiple coil receiver array. Given a series of \( T \) images, with \( T = 1 \) for the static imaging scenario, and \( N_p \) pixels in each image, we have that \( N = T \cdot N_p \), and \( x \in \mathbb{C}^N \). If a receiver coil array consisting of \( N_c \) coils is used to perform the measurement, and the Fourier sub-sampling is such that \( M_p \leq N_p \) k-space locations are sampled for each coil, then \( M = N_c \cdot M_p \), and \( y \in \mathbb{C}^M \). The acceleration rate is defined as \( R = N_p/M_p \).

Given the model (4.1) with \( N_c = 1 \) and \( M_p \leq N_p \), the linear system is underconstrained, and no unique solution exists by calculation of the Moore-Penrose inverse. However, in the case of parallel imaging, where multiple coils are used, \( N_c > 1 \), and for \( R \leq N_c \), the linear system is overconstrained, i.e. \( M > N \) and there are more equations in the linear system than unknowns, although not all equations may be linearly independent. It is this fact that makes parallel imaging possible. For \( M > N \), a unique least-squares solution may then be obtained via the Moore-Penrose inverse.

The condition number of \( A \) presents a practical limitation on the acceleration rate, \( R \), even when \( M > N \). The condition number of the linear operator \( A \) is given by the following:

\[
\kappa(A) = \frac{\sigma_{\text{max}}(A)}{\sigma_{\text{min}}(A)},
\]

where \( \sigma_{\text{max}}(A) \) and \( \sigma_{\text{min}}(A) \) are the maximal and minimal singular values of \( A \), respectively. A large condition number results in high noise-sensitivity in the pseudoinverse of \( A \). As \( R \) is increased, the noise sensitivity of an ill-conditioned \( A \) results in an ineffectual reconstruction while the number of equations still outnumbers the number of unknown pixels. Further, although an increase in the number of coils adds
additional equations, the condition number of the resulting $A$ matrix may show little or no improvement for large $N_c$.

Signal recovery via CS \cite{89, 90} provides the means for exploiting sparsity to achieve higher acceleration rates than have been obtained with parallel imaging alone. Given the model in (4.1), the CS framework claims that recovery of the original signal $x \in \mathbb{C}^N$ from $y \in \mathbb{C}^M$ with $M < N$ is possible provided certain conditions for the encoding matrix $A$ and the original signal $x$ are met. We state these conditions concisely in the following and develop them in addition to their relevance to MRI in the subsequent paragraphs:

- (1) The original data $x$ must exhibit sparsity in either the original domain or some transform domain.

- (2) The encoding matrix $A$ must exhibit incoherence, or randomness.

For signal recovery via CS, nonlinear approaches must be used, leading to a direct effect on the computational feasibility of any practical implementation using CS.

Condition (1) requires that the original data be sparse in some domain, that is, the data exhibits a small number $K < N$ of non-zero data points within this domain; such datasets are termed $K$-sparse datasets or signals. In synthesis CS, the unknown image $x$ is assumed to be well approximated by a sparse sum of $K$ columns of some dictionary, $\Psi$. In contrast, in analysis CS, the transform $\Psi x$ is assumed to be sparse, for a possibly tall and thin matrix $\Psi$. In the context of MRI signal recovery, these two forms of CS can produce significantly differing results; a further discussion on this topic is deferred until a later portion of this work.
The concept underlying sparsity relates to the compressibility of information. Sparse data implies that by selecting only the non-zero data points, one is able to compress this dataset without losing information. Conversely, if one is able to measure only the sparse component of the signal, i.e. only the non-zero portion of the dataset, one has succeeded in recovering the necessary information for reconstruction of the complete dataset.

The benefits of exploiting signal sparsity can be understood by considering the successful use of data compression in the field of image processing. Natural images are well-known to be compressible with little or no visual loss of information; and this serves as the motivation behind image compression tools such as JPEG and JPEG-2000, which utilize the discrete cosine transform and the wavelet transform. In these transform domains, natural image content can be represented as a vector of sparse coefficients, and image compression can be achieved by storing only the significant coefficients in these domains for later image reconstruction. Conversely, the nonlinear processing in a CS framework serves, in effect, to identify the location of significant information in the sparse domain.

Image reconstruction can benefit from signal sparsity as it becomes necessary then to only successfully recover the significant coefficients in a given sparse transform domain. In the context of MRI, this sparsity condition can potentially be achieved directly in the image domain (e.g. for MR angiography) or in the wavelet domain. Domains incorporating the temporal dimension can also be useful for improving signal recovery especially in dynamic imaging scenarios.

Condition (2) pertains to the encoding matrix $A$. For stable recovery, the operator $A$ cannot map two different sparse images to the same measurement vector. That
is, distinct sparse vectors must remain distinct under the action of $A$. While this condition, known as a restricted isometry [91], is not tractably verifiable, a matrix $A$ constructed via random Gaussian entries satisfies the condition with exceedingly high probability for large $M$ and $N$. For sparsity in the natural basis, $I$, a testable sufficient condition of $A$ is low coherence, where coherence is defined as the maximum absolute inner product of any two normalized columns of $A$.

In general, the encoding matrix denotes a mathematical model used to characterize a sampling process. Specifically related to MRI, this encoding matrix refers to the image acquisition scheme used to acquire the MR data. This can involve any of the parameters used for acquiring MRI data. One very interesting approach for introducing incoherence is through the variation of acquisition parameters such as repetition time, echo time, and flip angle; this serves as the motivation behind magnetic resonance fingerprinting [92, 93]. In the context of dynamic MRI, we are primarily interested in using the k-space sampling trajectory for obtaining the measured data to introduce incoherence. The incoherence of the trajectory may be measured qualitatively by examining the point spread function (PSF) of the k-space sampling trajectory. Examples of a random sampling pattern and a variable density sampling scheme used in this work along with the corresponding PSFs are shown in Figure 4.1. An ideal PSF corresponding to maximal incoherence shows minimal side lobe intensities.

Hence, the CS framework suggests that we sample k-space in a random fashion such that structured artifacts are not introduced in the target domain by the undersampling. The primary goal here is to avoid introducing coherent artifacts in the
Figure 4.1: Example random sampling trajectory (e) and variable density pseudorandom trajectory (g) along with corresponding point spread functions in (f) and (h). The point spread function demonstrating higher incoherence shows lower side lobe intensities outside the central peak. The side lobes for the pseudo-random trajectory are marginally stronger compared to the random trajectory. Source: [13].
target domain. In other words, by introducing a random sampling trajectory, all artifacts introduced by the undersampling become “noise-like.” As we have mentioned previously, this can be achieved in various ways; however, in the context of dynamic MRI, the most straightforward way for introducing incoherence is through alteration of the k-space trajectory.

We pause briefly to highlight the fact that the CS framework prescribes a theoretical lower bound, proportional to $K \log N$, on the number of measurements necessary to successfully recover a $K$-sparse signal provided that the conditions of signal sparsity and measurement incoherence are satisfied [94]. Furthermore, a nonlinear optimization algorithm should be used to recover this sparse signal. However, the nonlinear optimization is much more computationally expensive than linear inversion. Moreover, noise robustness can be poor in the face of incoherent $A$ or lack of sparsity in $x$.

From the perspective of a practical implementation of a CS framework for the reconstruction of dynamic MRI data, these two issues of computational efficiency and robustness become critical. More specifically, we are primarily interested in *efficient* approaches for practically implementing CS-based signal recovery that can result in sufficiently fast reconstruction times so as to be relevant *in the clinical setting*. We are also interested in the *robustness* of the approach in that signal recovery should robustly generate results of expected image quality and exhibiting expected diagnostic characteristics.

We now proceed to cover some of the relevant mathematical details necessary for introducing our CS-based holistic framework for signal recovery of dynamic cardiac cine MRI. As discussed in the previous chapter, reconstruction techniques for MRI
can be expressed generally as the following optimization problem:

\[
\hat{x} = \text{argmin}_x \|Ax - y\|_2^2,
\]

where \( y \) denotes the noisy measured data and \( x \) is the original data that we seek to estimate. The operator \( A \) here denotes the subsampled Fourier operator. The function \( f(x) := \|Ax - y\|_2^2 \) is often referred to as a cost function. Problem (4.3) is the unregularized reconstruction problem. Various regularization terms can be added to the cost function that can be used to incorporate prior information or knowledge regarding the data. Previously, we have seen that the use of Tikhonov regularization adds an additional quadratic term to the cost function and can be helpful in aiding recovery in the presence of noise.

In order to formulate the relevant optimization problem for the CS framework, we briefly review the definition of the \( \ell_p \)-norm. The \( \ell_p \)-norm of a vector \( x \) with components \( x_k \) is defined as follows:

\[
\|x\|_p = \left( \sum_k |x_k|^p \right)^{1/p}.
\]

For \( p = 2 \), it is straightforward to see we obtain the Euclidean norm. While technically not a norm, it is helpful to consider the limit as \( p \to 0 \):

\[
\|x\|_0 = \text{The number of nonzero components in } x.
\]

This definition of the \( \ell_0 \)-norm serves the useful purpose of defining an optimally sparse signal.

Given a measurement matrix \( A \), the CS framework seeks to find the sparsest solution consistent with the measured data \( y \), and the optimization problem can be
expressed as the following *constrained* problem:

\[
\tilde{x} = \arg\min_x \|x\|_0 \quad \text{s.t.} \quad \|Ax - y\|_2 < \epsilon,
\]  

(4.6)

for some \(\epsilon > 0\). If we are seeking sparsity in some transform domain \(\Psi\), we have a similar optimization problem:

\[
\tilde{x} = \arg\min_x \|\Psi x\|_0 \quad \text{s.t.} \quad \|Ax - y\|_2 < \epsilon.
\]  

(4.7)

Therefore, (4.7) presents the optimization problem that must be solved in order to obtain the optimal solution in the CS paradigm.

However, it is well-known that the problem of minimizing the \(\ell_0\)-norm is NP-hard [95, 96], and there exist no computationally tractable algorithms to practically obtain this solution. Convex relaxation may be used to replace the \(\ell_0\)-norm with the \(\ell_1\)-norm for which gradient descent methods can be applied. This results in the following practically realizable optimization problem:

\[
\tilde{x} = \arg\min_x \|\Psi x\|_1 \quad \text{s.t.} \quad \|Ax - y\|_2 < \epsilon.
\]  

(4.8)

The solution to (4.8) coincides with the unique solution to (4.7) if \(\Psi x\) is \(K\)-sparse and the measurement matrix satisfies what is known as the *restricted isometry property adapted to D* (D-RIP) [97] for some dictionary \(D\) which here corresponds to the transform domain \(\Psi\). The D-RIP requires that the measurement matrix \(A\) should be close to an isometry for signals \(x\) with \(\Psi x\) \(K\)-sparse. That is, \(A\) should affect Euclidean distances minimally. Formally, this may be expressed as the following requirement for all \(K\)-sparse \(\Psi x\):

\[
(1 - \delta)\|x\|_2^2 \leq \|Ax\|_2^2 \leq (1 + \delta)\|x\|_2^2,
\]  

(4.9)
for some isometry constant $\delta$, $0 < \delta < 1$. The methodical construction of the measurement matrices $A$ that satisfy the D-RIP as well as the tightness of these bounds is an area of active research in both the mathematical and signal processing communities. The theoretical foundations of this task may also be relevant to the optimal selection of k-space sampling trajectories in the practical application to MRI.

Given that the conditions prescribed by the CS framework are satisfied, practical compressive recovery can be accomplished using iterative algorithms to solve (4.8). While (4.8) is expressed as a constrained optimization problem, the unconstrained analogue is often considered:

$$\hat{x} = \arg\min_x \|Ax - y\|_2^2 + \lambda \|\Psi x\|_1,$$

with $\lambda$ now being a tuning parameter balancing the contributions from the data fidelity term and the $\ell_1$-regularization term. The presence of the $\ell_1$-norm eliminates the possibility of using linear solvers to achieve signal recovery. Therefore, the practical implementation of CS techniques requires the use of nonlinear iterative approaches. A wide variety of such techniques exist; we provide a selected overview of these techniques in a later section of this chapter.

The use of nonlinear iterative techniques to achieve MRI signal recovery within the CS paradigm can introduce additional conditions, under which these techniques may or may not converge to a satisfactory solution. Each method requires the tuning of various threshold parameters as well as the identification of optimal gradient step-sizes; the selection of these parameters as well as the robustness of our selection can be extremely sensitive to specific applications. Additionally, some methods may be more computationally intensive, leading to longer reconstruction times and limiting clinical relevance given current hardware capabilities.
4.2 A Sparse Review of Optimization Approaches for the Compressed Sensing Problem

It is well outside the scope of this work to provide a thorough overview of the various optimization approaches for the CS problem. However, we highlight a number of approaches that are relevant in the context of the CS problem for reconstruction of dynamic MRI data, along with a few remarks.

The most standard approach for finding a solution to the optimization problem posed by CS is the nonlinear conjugate gradient (NLCG) method. This method computes the gradient of each term within the cost function for the CS problem. Taking a step in the direction *conjugate* to the descent directions from all previous iterations can improve convergence. As the problem is nonlinear, a step size must then be calculated in order to ensure proper minimization of the cost function. The step size can be calculated by a number of approaches, all of which require multiple evaluations of the cost function. Hence, the NLCG method can be viewed as an approach that can exactly find the global solution to the nonlinear CS problem; however, this approach can be considerably computationally intensive. We briefly highlight here that the NLCG method can be used to solve the *analysis* CS problem; this will be further developed in a later chapter.

While the NLCG approach provides a suitable approach for the CS problem in the context of the dynamic MRI reconstruction problem, this approach can converge slowly in certain contexts. Additionally, the use of a line search can result in high computational requirements, especially in the context of large datasets. Other techniques such as the augmented Lagrangian [98] and split Bregman [99] methods can achieve improved convergence compared to NLCG methods.
Of particular interest are a class of algorithms termed iterative/shrinkage thresholding (IST) algorithms [100, 101]. This class of optimization algorithms calculates the gradient of the quadratic terms in the cost function for the CS problem, and implements the nonlinear regularization term via a shrinkage/thresholding operation. Provided that the proximal operator for this term has an efficient closed-form expression [102], this operation can efficiently approximate the nonlinear term in the CS cost function. The fast iterative shrinkage/thresholding algorithm (FISTA, [102]) achieves improved convergence over other IST algorithms and is of particular interest in this work. We reserve further discussion related to this topic for a later chapter.

4.3 Incoherent Samplings

In this section, we provide a brief overview of the various approaches for introducing incoherence in the context of the CS problem for reconstruction of dynamic MRI data.

As previously discussed, incoherence needed by the CS problem can be mapped to incoherence in the k-space sampling trajectory. Therefore, a suitable sampling trajectory in k-space is one in which random samples are acquired in k-space. However, the physics of MRI acquisition would require the use of single-point imaging in order to achieve this ideal degree of randomness, which would be prohibitively slow.

In the case of Cartesian sampling, where lines of k-space are acquired, we are then limited to introducing randomness in the additional spatial dimension and the temporal dimension. Therefore, the problem of incoherence reduces to the generation of a sampling trajectory in at least two dimensions, where the other (fully-sampled) spatial dimension is perpendicular to this plane.
In the context of dynamic MRI, a random 2D sampling can still provide a high degree of incoherence [103]; however, such sampling patterns can generate inconsistent results due to irregular gaps between samples or clustering in the sampling pattern. In the context of parallel MRI (pMRI), large gaps can lead to ill-conditioning of the underlying inverse problem [104], and clustering of k-space samples can reduce acquisition efficiency. Pseudo-random sampling can still provide a high degree of incoherence while regulating gaps between k-space sampling. For pMRI, empirical studies have demonstrated that the use of a Poisson disk sampling (PDS) can generate superior results compared to random sampling [104, 105], and in the context of static MRI applications, PDS remains a commonly used pseudo-random sampling scheme [105, 106]. However, in the context of dynamic MRI, PDS cannot be readily extended to the spatiotemporal 2D sampling trajectory, because a constant temporal resolution across frames cannot be maintained. Attempting to enforce this constraint destroys the uniform sampling density variation in PDS.

In this work, we utilize the variable density spatiotemporal acquisition (VISTA, [13]) sampling trajectory. We summarize the advantages of this approach in the following discussion and refer the reader to [13] for specific implementation details. As in PDS or other pseudo-random approaches, VISTA provides (1) uniform coverage of the k-space domain with bounded distance between consecutive samples to limit eddy currents, (2) incoherence, and (3) frequent sampling of the central region of k-space, which is useful in highly undersampled signal recovery problems. The iterative procedure used to generate a VISTA sampling also provides the additional flexibility to (1) maintain a constant temporal resolution by fixing the number of k-space lines acquired per frame and (2) guarantee a fully-sampled, time-averaged k-space. The
latter is useful for the estimation of convolution kernels in the case of GRAPPA and SPIRiT and also for the estimation of coil sensitivity maps in the case of SENSE. Additionally, a specific undersampling ratio, or acceleration rate, can be specified by specifically defining the number of samples that are to be distributed using the VISTA approach. Figure 4.3 shows a comparison between typical uniformly downsampled trajectories, PDS trajectories, and VISTA trajectories for various acceleration rates.
Figure 4.2: Comparison of uniformly downsampled (a), PDS (b), and VISTA (c) trajectories at various acceleration rates. Uniformly downsampled trajectories fall on a diagonal rather than horizontal grid in order to satisfy the requirement that the temporal average of all frames generates a fully-sampled k-space. This requirement is additionally satisfied with the VISTA sampling trajectories, but not with the PDS trajectories. Source: [13]
Chapter 5: A Fast Implementation for Compressive Recovery of Highly Accelerated Cardiac Cine MRI

The goal of this chapter is to present our primary contribution in terms of a holistic framework for a fast and practical implementation of compressive recovery for highly accelerated cardiac cine imaging. In this chapter, we focus primarily on the design and validation of this approach; the detailed practical implementation will be outlined in a subsequent chapter. We begin by presenting an algorithmic development of our approach as well as validation within a volunteer study. We then follow with additional studies that serve to algorithmically compare the performance of our approach with other optimization approaches. Finally, we conclude with a number of studies comparing this approach to conventional segmented cine techniques.

5.1 bFISTA: A Fast Implementation for Compressive Recovery of Highly Accelerated Cardiac Cine MRI Using the Balanced Sparse Model

In this section, we describe our approach for a fast implementation for compressive recovery of highly accelerated cardiac cine MRI using the balanced sparse model.
5.1.1 Background

Technological advances in both signal processing and hardware capabilities over the past two decades have resulted in significant progress in the field of magnetic resonance imaging (MRI). In the specific domain of dynamic cardiac cine imaging, breath-hold segmented techniques using balanced steady-state free precession (SSFP) have long been considered the clinical gold standard for evaluating cardiac function, offering good spatial resolution, high blood-to-myocardium contrast, and excellent image quality [79]. However, for patients presenting with arrhythmias or limited breath-hold capacity, real-time cardiac cine MRI [80, 53] under free-breathing conditions has shown greater clinical utility compared to breath-hold segmented techniques [53]. Real-time cine achieves the necessary spatial and temporal resolutions through the successful reconstruction of undersampled k-space data, and it was not until the introduction of parallel MRI (pMRI) techniques that real-time cine became feasible. However, commercially available pMRI techniques [107, 108] typically yield low achievable acceleration rates and as a result generally fail to achieve the minimum in-plane spatial resolution ($2.5 \times 2.5 \text{ mm}^2$) and temporal resolution (45 ms) necessary for maintaining fidelity of cardiac wall motion [77, 78].

Recent advances in the use of compressed sensing (CS) techniques for MRI [109] and its combination with pMRI techniques [65] have enabled real-time cardiac cine imaging at sufficiently high spatial and temporal resolutions for clinically relevant evaluation of cardiac function. However, few of these approaches have been deployed in the clinical environment, largely due to lengthy reconstruction times. We have previously shown that the combination of an incoherent sampling scheme and analysis
CS accelerated the real-time acquisition by a factor of 15 and thus enabled reconstruction with high spatial \(1.5 \times 2 \text{ mm}^2\) and temporal \(28 \text{ ms}\) resolutions \[13\]. This approach utilized a nonlinear conjugate gradient (NLCG) optimization algorithm to directly solve the analysis CS problem \[110\]. When combined with a CPU-based implementation, this resulted in lengthy reconstruction times. Furthermore, the implementation of this approach in the MATLAB programming environment prevented direct reconstruction within the scanner environment, diminishing its clinical utility. In this work, we seek to increase the clinical utility of this approach by introducing algorithmic and implementation modifications to reduce the reconstruction time, ultimately demonstrating an efficient reconstruction within the scanner environment in a manner invisible to the operator.

Numerous techniques have been proposed for accelerating CS recovery for pMRI, both from an algorithmic as well as computational perspective. Murphy et al. \[111\] presented parallelization strategies involving multi-core CPUs and GPUs to accelerate CS recovery for the k-space-based pMRI problem, achieving reconstruction times of 2-3 minutes for a single 3D volume. Ramani et al. demonstrated that an augmented Lagrangian framework \[98\] can converge faster than NLCG and mFISTA \[112\] approaches and reported times on the order of a minute for the image-based CS pMRI reconstruction of a single slice in 3D brain studies. More recently, Xue et al. have demonstrated distributed k-space-based reconstruction \[113\] using a multi-node parallel computing topology; this allowed for simultaneous multi-slice reconstruction of a stack of nine cardiac short axis cine slices with reconstruction times of four minutes when 6 nodes are used and one minute when 19 nodes are used.
Our contributions in this work are two-fold. First, we propose a new approach to perform analysis compressive recovery for sparsifying transforms with tight-frame representation. This approach uses the balanced sparse model [114] to achieve an approximate yet computationally efficient solution to the analysis CS problem and is termed the balanced fast iterative/shrinkage thresholding algorithm (bFISTA). In contrast to other approaches used for analysis CS, the iterative procedure prescribed in bFISTA is computationally more efficient as it requires only one forward and one backward computation of the sparsifying transform in each iteration. We apply bFISTA to two formulations of pMRI. We demonstrate that, compared to a NLCG technique, the use of bFISTA to solve k-space-based and image-based pMRI problems can achieve a significant reduction in reconstruction time. Second, we implement solutions to these two classes of pMRI problems within a GPU-based framework [15] that demonstrates fast reconstruction directly in a manner fully integrated with the MRI acquisition software and invisible to the operator.

The remainder of this section proceeds as follows: we present two main problem formulations for pMRI, review analysis and synthesis CS for pMRI, present our algorithmic framework for bFISTA, discuss our GPU implementation, describe our experimental methodology, present our results from in vivo studies, and conclude with further discussion of our results.
5.1.2 Problem Formulation for Unregularized pMRI

In general, the pMRI problem for cine imaging can be formulated into two related but distinct inverse problems. The first formulation employs a coil-by-coil interpolation in k-space. The interpolation weights are estimated from fully sampled calibration data. GRAPPA [74] and SPIRiT [85] are two well known k-space-based methods, with the former typically solved analytically and the latter iteratively. The second formulation directly reconstructs a single coil-combined image from undersampled k-space of multiple coils or channels. The coil sensitivity maps used in this formulation are estimated from fully sampled calibration data. SENSE [75] is the most commonly used image-based method and typically admits a closed-form solution.

The unregularized k-space-based formulation for SPIRiT can be expressed as follows:

$$\tilde{k}_m = \arg\min_{k_m} \| [S^H \lambda (G - I)^H]^H k_m - [I \ 0]^H y \|_2^2,$$

(5.1)

with $\tilde{k}_m$ denoting the estimated multiple channel k-space data. Similarly, the unregularized formulation for SENSE can be expressed as follows:

$$\tilde{x}_s = \arg\min_{x_s} \| SFC x_s - y \|_2^2,$$

(5.2)

with $\tilde{x}_s$ now denoting the estimated single channel image data. In both equations (5.1) and (5.2), $y$ denotes the noisy, acquired k-space data, $S$ denotes the undersampling operator, $F$ represents the frame-by-frame forward 2D Fourier transform, $G$ represents the frame-by-frame application of the k-space interpolation kernels, and $\lambda$ is a tuning parameter balancing term-wise contribution where relevant. The operator $C$ denotes the estimated coil sensitivity maps such that its application generates image data across all coils and all frames.
By letting \( k_m = F x_m \), with \( x_m \) here denoting the multiple channel data in the image domain, the k-space-based formulation in equation (5.1) can be equivalently expressed in a manner that seeks to recover multiple channel image data:

\[
\tilde{x}_m = \arg\min_{x_m} \left\| \left[ S^H \lambda (G - I)^H \right]^H F x_m - \left[ I \ 0 \right]^H y \right\|_2^2.
\] (5.3)

Equation (5.2) seeks to reconstruct a single image from multiple data channels, whereas equation (5.3) seeks to reconstruct the coil-by-coil images. Letting \( d \) represent the generic data to be estimated in either of the previous problem formulations, and letting \( Q \) represent either \( \text{SFC} \) in equation (5.2) or \( \left[ S^H \lambda (G - I)^H \right]^H F \) in equation (5.3), the problem formulation for unregularized pMRI may be expressed as the following:

\[
\tilde{d} = \arg\min_{d} \|Qd - \bar{y}\|_2^2,
\] (5.4)

where \( \bar{y} = [y^H \ 0^H]^H \) is a vector that augments \( y \) with zeros such that the number of elements corresponds to the number of rows in \( Q \).

In the case of dynamic MRI, we are interested in recovering a sequence of \( T \) images. Hence, equation (5.4) can be extended such that \( d = [d_1^H, \ldots, d_T^H]^H \) and \( y = [y_1^H, \ldots, y_T^H]^H \). In this case, \( Q = D(Q_1, \ldots, Q_T) \). Here \( D \) is an operator that places the operators \( Q_i \) along the diagonal entries of a matrix with zero elements elsewhere, such that each operator \( Q_i \) is applied to the \( i^{th} \) image. With dynamic MRI, it is typically advantageous to introduce \( \ell_1 \) regularization in some sparsifying domain; this leads to an iterative approach for both the k-space-based and image-based methods.
5.1.3 Synthesis and Analysis Compressed Sensing for pMRI

Compressed sensing techniques can be used to obtain solutions to both the k-space-based and image-based pMRI problems, as posed in equation (5.4) with suitable selection of $Q$, via the use of $\ell_1$ regularization within a sparse domain. For any generic sparsifying frame representation, we can define an analysis operator $\Psi$ such that $\alpha = \Psi d$ is sparse. Conversely, the synthesis operator $\Phi$ can be defined such that $d = \Phi \alpha$. Given these definitions, two approaches to CS for pMRI become readily apparent. Analysis CS utilizes the operator $\Psi$ to solve the following optimization problem with regularization parameter $\mu$:

$$\tilde{d} = \arg\min_d \|Qd - \bar{y}\|_2^2 + \mu \|\Psi d\|_1.$$

(5.5)

Conversely, synthesis CS utilizes the operator $\Phi$ and seeks the solution $\alpha$ via the following problem:

$$\tilde{\alpha} = \arg\min_{\alpha} \|Q\Phi \alpha - \bar{y}\|_2^2 + \mu \|\alpha\|_1,$$

(5.6)

with $\tilde{d} = \Phi \tilde{\alpha}$. For an image-based method, i.e. equation (5.2), the operators $\Psi$ and $\Phi$ are applied to a single image, whereas for a k-space-based method, e.g. equation (5.3), they are applied on a channel-by-channel basis, leading to significant computational differences between the two approaches. In the context of dynamic imaging, where it is necessary to recover multiple frames, image-based methods are less memory intensive and can be computationally more efficient.

If $\Psi$ is invertible and $\Phi = \Psi^{-1}$, the analysis and synthesis formulations are equivalent [110]. For redundant $\Psi$, the two formulations give different results, and the solution of synthesis CS and analysis CS are distinct in general. Recent empirical
evidence suggests that the analysis approach may outperform the synthesis approach [110]. For example, $\Psi$ based on the undecimated wavelet transform (UWT) [115] has been shown to produce images with higher quality compared to the more common decimated transform.

In the context of this work, a redundant representation based on the UWT is used for $\Psi$, motivating our goal in seeking a solution to the analysis CS problem. For dynamic cine imaging, we also seek to apply $\Psi$ in the 3D (two spatial and one temporal) domain in order to take advantage of structure in both the spatial and temporal dimensions. The high memory and computational requirement of a 3D UWT further motivates our goal in seeking an efficient solution to the analysis CS problem. In particular, we seek approaches that minimize the number of times $\Psi$ must be applied, as this significantly affects computation time.

5.1.4 bFISTA - A Fast Solver for Compressive Recovery using the Balanced Sparse Model

Numerous optimization techniques can be used to solve the CS problems posed in equations (5.5) and (5.6). The nonlinear conjugate gradient (NLCG) method can readily be used to obtain solutions to both problems. However, the use of a line search where the objective function is calculated at each step in the line search leads to multiple applications of $\Psi$ and long computation times. The fast iterative/shrinkage thresholding algorithm (FISTA) [102] achieves quadratic convergence rate through the use of a specific linear combination of the previous two iterates. However, since the proximal operator for FISTA does not have a closed-form solution for the analysis CS problem [116], it cannot be directly applied to equation (5.5). Recent extensions of FISTA, e.g. mFISTA [112] as well as dFISTA and sFISTA [116], can be used to solve
the analysis CS problem, but these approaches require multiple calculations of the 
operator $\Psi$ at each iteration. In this section, we propose balanced FISTA (bFISTA), 
a fast solver based on a balanced sparse model [114] that achieves a solution very close 
to that of the analysis CS problem while maintaining the computational efficiency of 
the original FISTA approach, with the accompanying constraint that $\Psi$ be a tight 
frame.

Let $\Psi$ be the analysis operator for a tight frame, i.e. $\Psi^H \Psi = I$; and let 
$\Phi = (\Psi^H \Psi)^{-1} \Psi^H$ be the corresponding synthesis operator. Then the notion of a pro-
jection operator $P := \Psi \Phi$ onto the range of $\Psi$ such that $P \alpha = \alpha$ for $\alpha \in \text{range}(\Psi)$ 
can be defined. We can then formulate equation (5.5) as a constrained synthesis CS 
problem:

$$\tilde{\alpha} = \arg\min_{\alpha \in \text{range}(\Psi)} \|Q \Phi \alpha - \bar{y}\|_2^2 + \mu \|\alpha\|_1^1.$$  \hspace{1cm} (5.7)

Using the projection operator, a related unconstrained problem with weight parameter 
$\beta$ can be obtained:

$$\tilde{\alpha} = \arg\min_{\alpha} \|Q \Phi \alpha - \bar{y}\|_2^2 + \beta \|(I - P) \alpha\|_2^2 + \mu \|\alpha\|_1^1.$$  \hspace{1cm} (5.8)

This problem, which we term the balanced CS problem, is known as a balanced sparse 
model [114] and can be seen as balancing between both the analysis and synthesis 
models. With $\beta \to 0$, the synthesis model is achieved, and with $\beta \to \infty$, the analysis 
model is achieved, i.e. the optimal solutions to equations (5.7) and (5.8) coincide. 
In practice, it is not possible to select $\beta = \infty$, and any model for which $\beta \neq 0$ 
may be viewed as a balanced sparse model. Augmented Lagrangian approaches as 
well as the alternating directional method of multipliers can be applied [117] directly 
to equation (5.8), with $\beta$ playing the role of a tuning parameter that is manually
selected; however, these approaches are computationally costly due to the need for multiple calculations of the operator $\Psi$. We choose to apply FISTA [102] directly to equation (5.8) and seek to select a suitable $\beta$ that yields a computationally efficient iterative procedure.

By merging the first two quadratic terms in equation (5.8) into a single quadratic term, we can apply FISTA directly to equation (5.8) to obtain a solution for the balanced CS problem. For a tight-frame representation, $\Phi = \Psi^H$ and $P^H = P$. FISTA then generates the following iterative procedure with shrinkage operator $T$ and gradient step size $\gamma$:

$$\alpha_{k+1} = T_{\gamma \mu} [\alpha_k - \gamma (\Psi Q^H (Q \Psi^H \alpha_k - \bar{y}) + \beta (I - \Psi \Psi^H) \alpha_k)].$$  \hspace{1cm} (5.9)

In order to obtain a computationally efficient solution, we choose here to set $\beta = 1/\gamma$. Therefore, at this point we trade the goal of obtaining a solution to the analysis CS problem for computational efficiency. Multiplying both sides by $\Psi^H$, letting $d_k = \Psi^H \alpha_k$, and simplifying, we obtain the final bFISTA step for a tight-frame representation with analysis operator $\Psi$:

$$d_{k+1} = \Psi^H T_{\gamma \mu} [\Psi \left( d_k - \gamma (Q^H (Q d_k - \bar{y})) \right)].$$  \hspace{1cm} (5.10)

The direct use of FISTA [102] in solving equation (5.8) guarantees a quadratic convergence rate provided $\gamma \leq 1/L$, where $L$ is the Lipschitz constant for the gradient term.

With a suitable selection of the tight-frame representation, the bFISTA procedure specified in equation (5.10) can be used to obtain approximations to the analysis CS problems for $\ell_1$-regularized SPIRiT and SENSE as posed in equation (5.5). Compared to mFISTA and its variants sFISTA and dFISTA [116], in addition to the method
proposed in [117], which require three or more calculations of $\Psi$ or its adjoint operator in each iteration, bFISTA requires two calculations, matching that of the original FISTA approach proposed for synthesis CS. The method proposed by Khare et al. [14] also approximates the solution to the analysis CS problem using the iterative procedure given in equation (5.10); however, a different linear combination of the previous two iterates is used. Their approach, however, does not establish a connection between equation (5.10) and the balanced CS problem that equation (5.10) is attempting to solve.

5.1.5 Gadgetron Implementation

To accelerate reconstruction, the Gadgetron framework [15] was used to implement $\ell_1$-regularized SPIRiT (bFISTA-SPIRiT) and SENSE (bFISTA-SENSE) using the bFISTA solver. The Gadgetron provides a convenient way for implementing reconstruction code directly in the scanner environment, using hardware that provides sufficient computational power for accelerating reconstruction. Built-in GPU-based functions for parallel implementation of basic arithmetic element-wise $n$-dimensional array operations were used whenever possible in our reconstruction implementations. Data matrices used for estimating kernel operators were formed using CUDA-based kernels for rearranging the fully-sampled time-averaged k-space (FTAK). Data fidelity and kernel operators were implemented using GPU-based arithmetic operations provided by the Gadgetron. A 2D FFT combined with pointwise multiplication and summing across coils was used to implement kernel operators in the image domain for fast computation. The undecimated wavelet transform was implemented as multiple convolution filters for each wavelet band. For short filter lengths (e.g. in Daubechies 1),
GPU implementation using convolution can be more efficient than pointwise multiplication in the Fourier domain. GPU parallelization was across coil-wise spatiotemporal voxel location with looping across wavelet bands. Precomputation of voxel indices was used to reduce memory accesses and improve computation time.

5.1.6 Methods

5.1.6.1 Generation of VISTA Sampling Patterns

All data were acquired using the VISTA [13] sampling pattern. This pseudo-random variable-density k-t sampling scheme can achieve a high degree of incoherence relative to a random sampling while also minimizing clustering of sampling points and guaranteeing a fully-sampled time-averaged k-space to facilitate pMRI reconstruction. Complete two-dimensional VISTA sampling patterns for acceleration rates 2 through 15 were generated using the code provided in [13] for a k-t grid of dimensions $n_{PE} \times 32$, with $120 \leq n_{PE} \leq 200$ being the number of phase encoding lines. All VISTA sampling patterns were pre-computed and stored as lookup tables that were read circularly during acquisition.

5.1.6.2 Data Acquisition

All aspects of this human study were approved by the local Institutional Review Board. Four healthy volunteers (labeled A, B, C, and D) provided written consent to participate in this study. Scanning was performed at 3T (Tim Trio, Siemens Healthcare, Erlangen, Germany) using a 32-channel cardiac array. For each volunteer, real-time cardiac cine data were acquired under free-breathing conditions in the 4-chamber long axis and short axis views at eleven different acceleration rates ($R = 3 - 12, 15$) using the previously described VISTA sampling patterns. An SSFP acquisition
was used with other relevant imaging parameters specified as follows: matrix size: 224×144; field of view (FOV): 360×288 mm^2 for volunteers A, B, and C and 400×320 mm^2 for volunteer D; slice thickness: 8 mm; TE: 1 ms; TR: 2.7-2.8 ms; receiver bandwidth: 1500 Hz/pixel; flip angle: 60°; 48 temporal frames. For volunteers A, B, and C, the spatial resolution was 1.6×2 mm^2, and for volunteer D, the spatial resolution was 1.8×2.2 mm^2. Temporal resolutions ranged from 134.4 ms at rate 3 to 33.6 ms at rate 12 and 26.9 ms at rate 15. For each volunteer all 22 datasets (2 views, 11 rates) were collected in a single session.

5.1.6.3 Data Reconstruction

Each acquired dataset was reconstructed using three methods: (1) a nonlinear conjugate gradient (NLCG) implementation of ℓ₁-SPIRiT (NLCG-SPIRiT), (2) our bFISTA implementation of ℓ₁-SPIRiT (bFISTA-SPIRiT), and (3) our bFISTA implementation of ℓ₁-SENSE (bFISTA-SENSE). We reiterate briefly that NLCG-SPIRiT is used to solve the analysis CS problem in equation (5.5), while bFISTA-SPIRiT and bFISTA-SENSE are used to solve the balanced CS problem in equation (5.8), with β suitably selected for computational efficiency.

NLCG-SPIRiT was implemented according to [13], with relevant tuning parameters λ = 1 × 10^{-2} and μ = 5 × 10^{-5} and a stopping criterion of a 1 × 10^{-4} percent difference in the normalized cost function. SPIRiT kernels of size 9×9 estimated from the FTAK were used, and an initialization consisting of view-sharing across three adjacent frames followed by a GRAPPA reconstruction with a 2×11 kernel size was used to aid convergence. The sum-of-squares method was used to combine final coil-wise reconstructed images.
For bFISTA-SPIRiT, the kernel size, initialization, and stopping criterion were identical to NLCG-SPIRiT. Tuning parameters were set as follows: \( \lambda = 1 \times 10^{-2} \) and \( \mu = 3 \times 10^{-2} \). The regularization parameter \( \mu \) was adjusted such that similar sharpness measurements (described in a following section) compared to NLCG-SPIRiT were observed in multiple test datasets. The parameter was then kept fixed during reconstruction of volunteer data.

For bFISTA-SENSE, coil sensitivity maps were estimated from the FTAK using a method by Walsh et al. \[118\]. The zero-filled k-space data was used as an initialization, and the stopping criterion was identical to that used in the previous two methods. The tuning parameter \( \mu \) was tuned in a manner identical to bFISTA-SPIRiT and was set to be \( 1 \times 10^{-3} \).

For all three reconstruction methods, structure across the two spatial dimensions as well as the time dimension was jointly exploited via \( \ell_1 \) regularization in the joint 3D spatiotemporal domain using a weighted 3D redundant Haar wavelet with a single-level decomposition as the sparsifying transform, \( \Psi \) \[119\]. In order to utilize the higher temporal correlation resulting from higher temporal resolution, detail subbands along the time dimension were weighted by the net acceleration rate \( R \), i.e. regularization strength for the temporal detail subbands was increased by a factor of \( R \). The presented approach is also amenable to auto-tuning of subband weights \[120\].

For the SPIRiT reconstruction methods, a joint sparsity model employing a hybrid \( \ell_2-\ell_1 \) norm for the wavelet coefficients was used \[105, 121\]. This allowed us to exploit similar sparsity patterns in the wavelet domain across coils that results from edges in individual coil images appearing at similar spatial positions.
5.1.6.4 Implementation of Reconstruction Methods

All reconstruction methods were implemented on an Intel Core i7 workstation with 64GB system memory and a NVIDIA GeForce GTX Titan with 6GB memory. SPIRiT-NLCG was written in MATLAB (MathWorks, Natick, Massachusetts, USA, v2013a) using CPU-based code. bFISTA-SPIRiT and bFISTA-SENSE were implemented within the Gadgetron framework [15] using GPU-based code. To aid reconstruction, all data were cropped in image space to approximately half the original FOV in the frequency dimension and 32-channel data were compressed to 12 virtual channels [122].

5.1.6.5 Image Scoring

All data sets (4 volunteers, 2 views, 3 reconstruction methods, and 11 acceleration rates) were scored for overall image quality by three expert readers – a physicist with over 20 years of CMR experience, a cardiologist with level III certification in CMR, and a radiologist with 5 years of CMR experience – using a 5-point Likert scale (1 = very poor, nondiagnostic image quality, 5 = excellent image quality), with a score of 3 or greater indicating adequate diagnostic image quality.

5.1.6.6 Quantitative Image Analysis

Regularized iterative reconstruction methods inevitably involve trade-offs between reconstruction time, noise, and image sharpness. For each of the acquired data sets, we evaluate each of these metrics. For each of the three reconstruction methods, total reconstruction time was automatically recorded. For the NLCG reconstruction, total reconstruction time referred to the time from when the raw data was loaded to the time when final reconstructed images were made available. For the bFISTA
reconstructions within the Gadgetron framework, total reconstruction time referred to the time from when all k-space lines are received by the workstation to the time when reconstructed images are serialized and sent back to the scanner.

The method proposed in [123] was used to quantify sharpness for each short-axis dataset. Left-ventricular boundaries were manually selected by the user, and image intensity profiles normal to the edges were parametrically modelled using a sigmoid function [124, 125]. To improve measurement accuracy, all frames from each short-axis dataset were used in the sharpness measurement.

While there are no standard methods for quantifying noise in scenarios where noise is correlated across space and time due to spatiotemporal regularization, we utilize the method proposed in [13]. This method extends the image difference-based method [126] to minimize the impact of correlation across neighboring frames. The following steps are performed for this noise quantification method: (1) an image region with either static or minimal signal is selected, (2) data from this region are organized as a 2D matrix, such that the $i^{th}$ column contains all pixels from the selected region in the $i^{th}$ frame, (3) columns in this matrix are randomly ordered so as to minimize the effect of correlation between adjacent frames, (4) the forward difference is computed across columns by subtracting the $i^{th}$ column from the $(i + 1)^{th}$ column, (5) the resulting matrix is then rearranged into a 1D vector, (5) the standard deviation of the vector is defined as noise.

5.1.6.7 Statistical Analysis

All comparisons between the three reconstruction methods were made using one-way ANOVA with a Tukey-Kramer correction for multiple comparisons. Reconstruction times, sharpness measurements, noise measurements and image quality scores
were pooled across all volunteers and all views. Image quality measurements were additionally pooled across the three readers. Statistical comparisons were then made between the three reconstruction methods for each acceleration rate.

5.1.7 Results

Figure 5.1 shows reconstruction results for the short axis view for volunteer A. Figure 5.1a shows individual frames for each of the three reconstruction methods and six of the eleven acceleration rates. Individual frames correspond to the frames with the worst (lowest) median sharpness measurement for the left-ventricular boundary. Figure 5.1b shows the corresponding temporal profiles for the three reconstruction methods and six acceleration rates along the solid line indicated in the top-left image of Figure 5.1a. Tick marks at the bottom of each profile indicate the temporal location of the corresponding frame displayed in Figure 5.1a. Figure 5.2 shows reconstruction results for the long axis view for the same volunteer.

Across all acceleration rates, little visual difference between NLCG-SPIRiT and bFISTA-SPIRiT can be observed. Therefore, between NLCG-SPIRiT and bFISTA-SPIRiT, a substantial reduction in reconstruction time can be achieved with little visible impact on image quality. Up to acceleration rate 8, temporal profiles for bFISTA-SENSE correspond closely to the other two methods; however, differences in terms of resolving of fine spatiotemporal details (arrows) can be observed at higher acceleration rates. As a result, with bFISTA-SENSE, clinically reasonable reconstruction times can be achieved with a slight visible difference in image quality.
Figure 5.1: Short-axis views of real-time, free-breathing cines from volunteer A. (a) Individual frames from NLCG-SPIRiT, bFISTA-SPIRiT, and bFISTA-SENSE for six different acceleration rates. The frame with the worst median sharpness measurement is presented from each image series. (b) Temporal profiles along line drawn on the top-left image in panel (a). Tick marks below each profile indicate temporal location of frame presented in panel (a). Compared to the NLCG-SPIRiT results, bFISTA-SPIRiT results achieve a significant reduction in reconstruction time with little appreciable reduction in image quality of dynamic features. bFISTA-SENSE achieves further reduction in reconstruction times with a slight increase in blurring artifacts at rate 10 and above (arrows).
Figure 5.2: Four-chamber views of real-time, free-breathing cines from volunteer A. (a) Individual frames from NLCG-SPIRiT, bFISTA-SPIRiT, and bFISTA-SENSE for six different acceleration rates. The twentieth frame is presented from each image series. (b) Temporal profiles along line drawn on the top-left image in panel (a). bFISTA-SPIRiT again achieves reduced reconstruction times compared to NLCG-SPIRiT with little difference in image quality for all acceleration rates. bFISTA-SENSE begins to exhibit blurring artifacts at acceleration rates 12 and above (arrows). With bFISTA, the SENSE reconstruction can achieve clinically relevant reconstruction times of 13.8 seconds without a diagnostically significant reduction in image quality.
Figure 5.3 shows results of noise and sharpness measurements for each of the three reconstruction methods averaged across all volunteers and views. Edge sharpness was measured along the left-ventricular boundary of the short-axis view; noise was estimated in both the short-axis and four-chamber views. Noise and sharpness measurements were separately performed for each of the four volunteers. Between NLCG-SPIRiT and bFISTA-SPIRiT, no significant difference in sharpness was observed for all acceleration rates and all volunteers. A significant reduction in image sharpness for bFISTA-SENSE compared to NLCG-SPIRiT was observed at acceleration rates higher than 7 ($p < 0.05$) across all volunteers. However, image sharpness shows no significant reduction up to acceleration rate 7, corresponding to a temporal resolution of 57.6 ms. Comparisons of noise measurements between the three reconstruction methods showed no statistical difference at all acceleration rates ($p > 0.7$).

Results of overall image quality assessments are shown in Table 5.1. With the exception of the comparison between bFISTA-SENSE and NLCG-SPIRiT at acceleration rates 5, 6, 9 and 12, no significant differences in the scores were observed between the three reconstruction methods for each acceleration rate ($p > 0.05$). For bFISTA-SPIRiT, at rates 11 and 12, only one of the 24 scores was below 3. Evaluation of bFISTA-SENSE indicate that image quality is slightly lower compared to both NLCG-SPIRiT and bFISTA-SPIRiT, with 3 scores at rate 11 and 2 scores at rate 12 being below 3.

Recorded reconstruction times for each of the three methods are summarized in Table 5.2, with FISTA reconstruction methods showing significantly reduced times compared to the NLCG method ($p < 0.05$). For an image series with 48 frames
Figure 5.3: Average noise standard deviation and edge sharpness quantification across all volunteers. Noise measurements for each reconstruction method (a) are averaged across both 4CH and SAX views. The left ventricular boundary was used for the sharpness measurement (b). Between all three reconstruction methods, no significant difference in noise level was observed for each volunteer and each acceleration rate ($p > 0.7$). Between NLCG-SPIRiT and bFISTA-SPIRiT, no significant difference in sharpness was observed for all volunteers and acceleration rates ($p > 0.05$ for all comparisons). Significant reduction in sharpness was observed between NLCG-SPIRiT and bFISTA-SENSE at rates 7 and above across all volunteers ($p < 0.05$).

and a matrix size of $224 \times 144$, our implementation of bFISTA-SPIRiT within the Gadgetron allows us to achieve a 24-fold reduction in average reconstruction time from 3330.8 seconds (55.5 minutes) to 140.7 seconds (2.35 minutes) compared to the previous CPU-based implementation of NLCG within the MATLAB environment (Table 5.2). The implementation of bFISTA-SENSE achieved a mean reconstruction time of 13.8 seconds – a 10-fold further reduction in reconstruction time compared to bFISTA-SPIRiT and an overall 240-fold improvement compared to NLCG-SPIRiT.
Table 5.1: Image Scores (Mean ± Standard Deviation) For Each Method Across All Volunteers, All Views, and All Readers

<table>
<thead>
<tr>
<th>Rate</th>
<th>NLCG-SPIRiT</th>
<th>bFISTA-SPIRiT</th>
<th>bFISTA-SENSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>3.08 ± 0.76</td>
<td>3.00 ± 0.58</td>
<td>2.79 ± 0.58</td>
</tr>
<tr>
<td>4</td>
<td>3.33 ± 0.75</td>
<td>3.17 ± 0.62</td>
<td>3.04 ± 0.61</td>
</tr>
<tr>
<td>5</td>
<td>3.46 ± 0.58</td>
<td>3.38 ± 0.48</td>
<td>3.05 ± 0.45†</td>
</tr>
<tr>
<td>6</td>
<td>3.71 ± 0.68</td>
<td>3.41 ± 0.57</td>
<td>3.25 ± 0.43†</td>
</tr>
<tr>
<td>7</td>
<td>3.71 ± 0.68</td>
<td>3.63 ± 0.56</td>
<td>3.29 ± 0.61</td>
</tr>
<tr>
<td>8</td>
<td>3.50 ± 0.71</td>
<td>3.54 ± 0.49</td>
<td>3.33 ± 0.55</td>
</tr>
<tr>
<td>9</td>
<td>3.71 ± 0.73</td>
<td>3.50 ± 0.58</td>
<td>3.12 ± 0.44†</td>
</tr>
<tr>
<td>10</td>
<td>3.63 ± 0.70</td>
<td>3.75 ± 0.66</td>
<td>3.42 ± 0.64</td>
</tr>
<tr>
<td>11</td>
<td>3.50 ± 0.76</td>
<td>3.33 ± 0.62</td>
<td>3.08 ± 0.57</td>
</tr>
<tr>
<td>12</td>
<td>3.58 ± 0.70</td>
<td>3.42 ± 0.64</td>
<td>3.13 ± 0.60†</td>
</tr>
<tr>
<td>15</td>
<td>3.50 ± 0.65</td>
<td>3.33 ± 0.55</td>
<td>3.17 ± 0.55</td>
</tr>
</tbody>
</table>

†Significant difference observed compared to NLCG-SPIRiT (p ≤ 0.05).

5.1.8 Discussion

In this work, we have proposed bFISTA, an efficient version of FISTA that utilizes the balanced sparse model [114] to achieve an approximate solution to the analysis CS recovery problem. This approach is relevant for both k-space-based and image-based pMRI reconstruction. The analysis CS formulation of the pMRI problem in equation (5.5) is of particular interest due to benefits pertaining to image quality; this serves as our primary motivation for seeking computationally efficient approaches for solving this problem. Extensions of FISTA such as mFISTA [112] as well as dFISTA and sFISTA [116] are relevant to the analysis CS problem, yet they lead to iterative procedures that are more computationally intensive than the original FISTA approach proposed for synthesis CS. Such increased computation can become
Table 5.2: Mean Reconstruction Times For Each Method Across All Volunteers and All Views

<table>
<thead>
<tr>
<th>Rate</th>
<th>Mean Reconstruction Time (Seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>NLCG-SPIRiT</td>
</tr>
<tr>
<td>3</td>
<td>3147.9</td>
</tr>
<tr>
<td>4</td>
<td>3123.6</td>
</tr>
<tr>
<td>5</td>
<td>3373.2</td>
</tr>
<tr>
<td>6</td>
<td>3425.2</td>
</tr>
<tr>
<td>7</td>
<td>3337.4</td>
</tr>
<tr>
<td>8</td>
<td>3432.7</td>
</tr>
<tr>
<td>9</td>
<td>3433.3</td>
</tr>
<tr>
<td>10</td>
<td>3399.6</td>
</tr>
<tr>
<td>11</td>
<td>3419.0</td>
</tr>
<tr>
<td>12</td>
<td>3409.9</td>
</tr>
<tr>
<td>15</td>
<td>3137.1</td>
</tr>
<tr>
<td>MEAN</td>
<td>3330.8</td>
</tr>
</tbody>
</table>

significant in large-scale problems, e.g. reconstruction of dynamic MRI. In 2012, Khare et al. [14] proposed an iterative/shrinkage thresholding approach to solve the analysis CS problem. Their approach is similar to bFISTA, except that a different weighting is used to combine the previous two iterates. In this work, we establish that bFISTA and the method by Khare et al. do not solve the analysis CS problem but instead solve the problem given by the balanced CS model for a specific selection of $\beta$.

Our approach, bFISTA, matches the computational complexity of the original FISTA method, although it works only for redundant tight-frame representations.
Our GPU-based implementation within the Gadgetron framework provides a convenient avenue for effectively utilizing the highly parallelized computation power provided by GPUs for significantly accelerating reconstruction, while also allowing us to demonstrate online reconstruction within the scanner environment in a manner invisible to the operator. Raw data was piped out of the scanner reconstruction chain across the medical network to the GPU system; and reconstructed images were sent back directly into the DICOM patient database on the scanner, with the only noticeable latency being due to image acquisition and reconstruction.

Evaluation by expert readers indicates that bFISTA-SPIRiT can reconstruct images with equivalent image quality compared to the significantly slower NLCG-SPIRiT method. Therefore, for the same reconstruction method (e.g. SPIRiT), the balanced sparse model used in bFISTA can achieve a solution very close to that of NLCG, which uses the analysis model. In fact, for each acceleration rate, the percentage of image quality scores less than 3 (unacceptable diagnostic quality) for bFISTA-SPIRiT was always less than or equal to that of NLCG-SPIRiT.

Comparison in image sharpness between NLCG-SPIRiT and bFISTA-SPIRiT methods shows no significant difference in sharpness measurements in the left ventricular myocardium border (Figure 5.3b). With bFISTA-SPIRiT, we are able to achieve a spatial resolution of $1.6 \times 2 \text{ mm}^2$ and a temporal resolution of 44.8 ms at an acceleration rate of 9, thereby meeting minimum clinical guidelines for spatial and temporal resolution [77, 78] with no reduction in image quality or sharpness and an average reconstruction time that can be faster than other methods [127]. With
bFISTA-SENSE, we achieve average reconstruction times of 13.8 seconds, approaching reconstruction times of traditional pMRI methods that utilize a closed-form solution. At an acceleration rate of 9, bFISTA-SENSE can achieve spatial and temporal resolutions meeting clinical guidelines with slightly reduced yet still diagnostic image quality and a modest reduction in sharpness compared to NLCG-SPIRiT. It should be noted that by relaxing the spatial resolution slightly, bFISTA-SENSE may achieve satisfactory spatial and temporal resolution while maintaining overall image quality and sharpness.

Both bFISTA-SPIRiT and bFISTA-SENSE target the balanced CS problem for reconstruction of dynamic real-time cine. These approaches maintain comparable image quality compared to NLCG-SPIRiT, which targets the corresponding analysis CS problem, while additionally achieving significantly reduced reconstruction times. bFISTA-SPIRiT can achieve this reduction with no observable impact on image quality or image boundary sharpness. Switching the reconstruction methods from the k-space-based SPIRiT reconstruction to an image-based SENSE reconstruction eliminates the need for coil-by-coil processing, resulting in faster reconstruction times with a marginal reduction in image quality. Provided accurate coil sensitivity maps, image-based SENSE reconstruction can generate images with optimal signal-to-noise ratio (SNR); however, accurate measurement of coil sensitivity maps may not be possible especially in real-time, single-shot imaging or imaging in the presence of severe breathing motion [128]. In this case, kernel-based methods (e.g. GRAPPA as well as SPIRiT) are generally more robust, generating images with fewer artifacts compared to methods involving the explicit use of sensitivity maps [86]. This may serve to explain the slight reduction in image quality for bFISTA-SENSE compared to
bFISTA-SPiRiT and NLCG-SPiRiT. In this study, the same coil sensitivity maps and kernels were used to reconstruct each image frame; frame-by-frame estimation could help improve image quality. While bFISTA-SENSE can achieve faster reconstruction times, in some scenarios slightly reduced image quality may become manifest in the form of either additional blurring of dynamic image features or increased artifact levels.

We note briefly that, unlike segmented cine imaging, with real-time cine imaging the acquisition time does not directly depend on the acceleration rate. Therefore, the operator is free to choose the scan duration. With all other acquisition parameters held constant, the direct effect of the acceleration rate is on temporal resolution, or acquisition time per frame. Hence, with real-time cine imaging, higher acceleration rates are used not to reduce total scan time, but to reduce acquisition time per frame, or increase temporal resolution.

Over the range of acceleration rates considered in this study, noise measurements in the reconstructed images stayed relatively constant. For the SPiRiT-based methods, image sharpness improved at lower acceleration rates and remained constant at higher acceleration rates. This type of behavior has been observed in previous studies [13]. In these scenarios, higher acceleration rates result in a finer temporal resolution which leads to reduced blurring. The increased temporal resolution at higher acceleration rates can offset blurring due to image recovery from highly undersampled data. With the SENSE-based method, image sharpness shows a similar trend at lower acceleration rates. At higher acceleration rates however, the increased temporal resolution is not sufficient enough to offset blurring due to image recovery. This leads
to the observed reduction in image sharpness across the entire range of acceleration rates.

In all the methods described in this work, the need to tune free parameters and set stopping criteria can limit the robustness of these techniques in the clinical environment. In our study, a fixed set of parameters was selected for the NLCG method based on experience, and the corresponding parameters for bFISTA were tuned using a test dataset such that image sharpness matched NLCG results. These parameters were then fixed and used for reconstruction of data acquired for each volunteer. However, this approach provides no guarantee of robustness in patient populations. Additionally, there exists no guarantee that the chosen parameters optimally take advantage of varying levels of sparsity between the spatial and temporal dimensions. In this case, efficient methods for automatically tuning free parameters [120] will be necessary and are the subject of future work.

5.1.9 Conclusions

In this work, we describe bFISTA, an approach for using the FISTA framework to achieve an approximate yet computationally efficient solution to the analysis compressive sensing recovery problem in the context of pMRI. We further describe two approaches for applying bFISTA to the fast reconstruction of variable density incoherent spatiotemporally sampled cardiac cine MRI. The first approach, bFISTA-SPIRiT, significantly reduces reconstruction times compared to a NLCG-based implementation. This is achieved by a GPU-based C++ implementation within the Gadgetron framework that also allows for reconstruction directly in the scanner environment. No significant change in noise level, image quality, or boundary sharpness was observed
with this approach. The second approach, bFISTA-SENSE, achieves further reduction in reconstruction time with comparable noise level, a slightly reduced yet still diagnostic image quality, and a slightly decreased boundary sharpness. We validate these techniques using in \textit{vivo} data and demonstrate that bFISTA can be applied to pMRI problems for the successful reconstruction of dynamic cardiac cine data. Our \textit{in vivo} studies demonstrate that the reconstructed data meet the necessary spatial and temporal resolution requirements for adequate ventricular function evaluation within a reasonable reconstruction time, making our approach relevant for deployment within the clinical environment.

5.2 Other Studies Relevant to the Application of bFISTA for Reconstruction of Highly Accelerated Cardiac MRI

In this section, we present a number of smaller studies relevant to our proposed method, bFISTA, for reconstructing highly accelerated cardiac MRI.

5.2.1 On the Convergence of bFISTA Compared to a Nonlinear Conjugate Gradient Approach

In this section, we present a small study examining the performance of bFISTA compared to a nonlinear conjugate gradient implementation in the context of the reconstruction of a static water bottle phantom in addition to the reconstruction of data from an exercise stress study using cardiac cine MRI.

The work presented in this study has been previously published as the following reference:

We present the published work in its entirety in the following sections and include additional remarks in the final section. In the published work, the approach termed S-SPIRiT refers to the application of bFISTA to the SPIRiT approach [85].

5.2.1.1 Purpose

The high spatial and temporal requirements of real-time cardiac MR imaging under free-breathing conditions can be challenging to meet with current GRAPPA [74] and SENSE [75] reconstruction techniques. SPIRiT [85] improves upon GRAPPA by taking advantage of additional self-consistency constraints between acquired and unacquired data. Yet typical iterative solutions for SPIRiT with nonlinear regularization are susceptible to local minima issues due to ill-conditioning of the underlying optimization problem. The nonlinear conjugate gradient (NLCG) solver can be computationally intensive for SPIRiT with $\ell_1$ regularization and is difficult to tune to specific scenarios. We propose shrinkage SPIRiT (S-SPIRiT), a FISTA-based [102] implementation of $\ell_1$-regularized SPIRiT that provides improved robustness to suboptimal parameter tuning and reduced computational requirements compared to NLCG techniques.

5.2.1.2 Methods

Given the cost function for unregularized SPIRiT,

$$g(x) = \|Fx - y\|_2^2 + \lambda\|Gx - x\|_2^2,$$

(5.11)
the $\ell_1$ regularized formulation is then

$$f(x) = g(x) + \mu \| \Psi x \|_1,$$

(5.12)

with $F$, $G$, and $\Psi$ denoting respectively the k-space sampling operator, the SPIRiT self-consistency operator, and a sparse transform operator; $x$ and $y$ denoting respectively the reconstructed and acquired data; and $\lambda$ and $\mu$ denoting tuning parameters. FISTA [102] minimizes $f(x)$ by first computing a gradient step

$$x_{k+1} \leftarrow x_k - \nabla_x g(x_k),$$

(5.13)

and then performing a shrinkage operation $T_\mu$ within the sparsifying domain:

$$x_{k+1} \leftarrow \Psi^{-1} T_\mu \Psi (x_k - \nabla_x g(x_k)).$$

(5.14)

Fast $\mathcal{O}(1/k^2)$ convergence is achieved by using a particular linear combination of the previous two iterations and the input to the next iteration. In S-SPIRiT, we apply FISTA to the standard SPIRiT cost function to achieve $\ell_1$ regularization within the wavelet domain. The low cost of the gradient calculation for FISTA significantly reduces computational time compared to the multiple line searches required for the NLCG approach. In S-SPIRiT, we also use a dynamically updated wavelet thresholding parameter based on the ratio of the noise variance to subband signal as estimated from the wavelet coefficients [129], eliminating the need to tune an additional parameter and providing robustness in the presence of suboptimal parameter tuning and high noise levels.

S-SPIRiT was tested in both phantom and in vivo studies. In the phantom study, 80 frames of an axial slice of a static spherical water bottle phantom were acquired (Siemens, Avanto 1.5T, 12 channels) with no parallel acceleration. Data were retrospectively randomly downsampled in both spatial directions to rate 6. A $7 \times 7$ SPIRiT
kernel was estimated using the temporal average of all frames. Both S-SPIRiT and NLCG reconstructions were performed in two scenarios: (1) without additional noise and (2) with additional complex Gaussian noise. For optimal performance, NLCG reconstruction utilized both wavelet domain and total variation (TV) regularization. Per iteration RMS error between results and the full k-space data was measured. Tuning parameters were optimized for the noiseless scenario and remained unchanged in the noisy scenarios. For the in vivo study, free-breathing cardiac cine data (uniformly downsampled, temporally-interleaved Cartesian trajectory, 32 channels) were acquired at rest (rate 6, 256 frames, 4-chamber view) and after treadmill exercise stress (rate 5, 50 frames, short axis view) at 1.5T (Siemens, Avanto) from one healthy volunteer. All aspects of this in vivo study were approved by the local Institutional Review Board. Data were first reconstructed using GRAPPA with a $4 \times 5$ kernel estimated from the temporal average of all frames. GRAPPA results were used to initialize both S-SPIRiT and NLCG reconstructions. A $7 \times 7$ SPIRiT kernel estimated using the temporal average of the initially acquired data was used. The minimum of the cost function was used as a stopping criterion for both reconstructions. For optimal performance, NLCG reconstruction used both wavelet and TV regularization terms. Average SNR based on random matrix theory [130] was measured for both reconstruction methods. All reconstructions were performed using Matlab 2012a on an Intel Core i5 workstation with 16 Gb memory.

5.2.1.3 Results

Figure 5.4 shows RMS error relative to the full k-space phantom data for both S-SPIRiT and NLCG in the noisy scenario. The use of dynamic wavelet thresholding provides an additional degree of robustness, allowing S-SPIRiT to achieve reduced
Figure 5.4: Phantom RMSE for S-SPIRiT and NLCG with additional noise. NLCG achieves optimal RMSE in 50 seconds, while S-SPIRiT surpasses NLCG in 25 seconds.
Figure 5.5: *In vivo* NLCG (left column) and S-SPIRiT (right column) images. S-SPIRiT shows visibly reduced noise levels in both the lung region (top row) and within the myocardium and blood pool (bottom row).
RMSE compared to NLCG in a shorter amount of time. Figure 2 shows a comparison between \textit{in vivo} NLCG and S-SPIRiT results at rest during systole, where cardiac motion is most pronounced. At rest, S-SPIRiT produced a 14.80% SNR improvement over NLCG. Post exercise stress, S-SPIRiT produced a 33.44% improvement over NLCG. For the phantom study, each NLCG iteration took 5 seconds, whereas each S-SPIRiT iteration took on average 0.9 seconds. In the \textit{in vivo} study, NLCG required on average 16 seconds per iteration and typically converged within 10 iterations for a per frame reconstruction time of about 160s. S-SPIRiT required 3.5 seconds per iteration and converged within 25 iterations, or about half the reconstruction time of NLCG.

5.2.1.4 Conclusion

Compared to NLCG techniques, S-SPIRiT can provide an efficient implementation of $\ell_1$-regularized SPIRiT with additional robustness towards suboptimal parameter tuning and high noise levels. S-SPIRiT may be a practical way to achieve improved image quality beyond GRAPPA in the context of free-breathing real-time cardiac MR imaging.

5.2.1.5 Further Comments

The primary goal of this study was to illustrate the computational differences between NLCG and a FISTA-based approach on the same physical hardware. From this perspective, two aspects are important to consider. The first aspect is the per-iteration computation time. This small study serves to demonstrate that per-iteration, FISTA is more computationally efficient than NLCG due to the elimination of the line search. The second aspect is the overall computation time to reach optimal
image quality (as quantified by RMSE in the case of the phantom study) or to reach convergence (as quantified by a stopping criterion in the case of the \textit{in vivo} study). In the case of the phantom study, a FISTA-based approach can achieve a better image quality as quantified by RMSE with a faster total reconstruction time compared to NLCG (Figure 5.4). Given a certain stopping criterion, in the \textit{in vivo} study, a FISTA-based approach can also achieve improved image quality compared to NLCG (Figure 5.5). Taking into account a few key caveats discussed below, this suggests that a FISTA-based approach can exhibit improved holistic performance (e.g. taking into account both efficiency and quality) compared to a NLCG approach.

It is important to note a few differences in this small study. For all reconstruction approaches, a uniformly downsampled k-space trajectory was used. Therefore, the condition of incoherence as quantified by a random or pseudo-random k-space trajectory is not satisfied here. This can have the impact of negatively affecting algorithmic performance by the introduction of local minima in the optimization problem. In this study, an adaptive wavelet denoising process was used to adjust the shrinkage parameter in each iteration. This can also impact algorithmic convergence for both FISTA-based methods and NLCG methods. In the later work presented earlier in this chapter related to bFISTA, we utilize fixed weights that produce consistent results in small volunteer populations. However, for the case of exercise stress imaging described here, dynamic denoising techniques may prove to be useful. In a subsequent chapter, we explore further dynamic, spatially-varying techniques for regularization in the context of bFISTA.

We note briefly here that a similar approach with a dynamic denoising process was proposed by Khare et al. [14] and applied to static brain imaging. In their work they
Figure 5.6: $L_2$-error convergence behavior for IST with SURE and NLCG approaches. $L_2$-error is calculated with respect to the reconstruction from fully sampled data. Source: [14]
examine the use of an iterative approach similar to FISTA but with a differing linear combination and apply this to the SENSE reconstruction problem. We include their convergence plots in Figure 5.6 as a comparison to the results obtained in this study. Qualitatively, the convergence results they obtain using their iterative approach match those reported in this study.

5.2.2 On the Algorithmic Performance of bFISTA Compared to MFISTA

In this section, we describe and present some results from a small study comparing the algorithmic performance of the previously described bFISTA approach compared to MFISTA [112]. The MFISTA optimization algorithm can be applied to solve the analysis CS problem as reiterated below:

$$\tilde{d} = \arg\min_d \|Qd - y\|_2^2 + \mu R(\Psi d).$$  \hspace{1cm} (5.15)

Here, $Q$ is some operator which will be described further in the experimental details, and $R$ is some regularizer. In this section, we consider the generalized LASSO regularizer (L1), \( R(\Psi d) = \|\Psi d\|_1 \) and examine the algorithmic performance of both bFISTA and MFISTA.

5.2.2.1 Experimental Details

We consider the signal recovery problem in (5.15) in the context of the standard Shepp-Logan phantom (Figure 5.7). Specifically, we attempt to recover an image signal $d \in \mathbb{R}^N$ of dimensions $N = 128 \times 128$ from noisy measurements $y \in \mathbb{R}^M$ with $M/N$ being the undersampling ratio and $M < N$ according to the following signal model:

$$d = Qy + w,$$  \hspace{1cm} (5.16)
for some added zero-mean, white and circular Gaussian noise \( w \) with noise variance \( \sigma^2 \). In this setting, we use a “spread spectrum” operator \([131]\) of the form \( Q = DFC \) with \( C \in \mathbb{R}^{N \times N} \) being a diagonal matrix with independently and identically distributed (i.i.d.) equiprobable \( \pm 1 \) entries, \( F \in \mathbb{C}^{N \times N} \) being the discrete Fourier transform (DFT), and \( D \in \mathbb{R}^{M \times N} \) being a row-selection operator selecting \( M \) rows of \( FC \in \mathbb{C}^{N \times N} \) uniformly at random. The measurement signal-to-noise ratio (SNR) is defined to be \( \|y\|_2^2 / (M\sigma^2) \), and the recovery SNR of the signal estimate \( \hat{d} \) given the noiseless gold standard \( d \) is defined as \( \|d\|_2^2 / \|d - \hat{d}\|_2^2 \).

All experiments were conducted in the MATLAB programming environment. Signal recovery via both bFISTA and MFISTA were run for the above experimental setup using an inner-outer iteration structure. The maximum number of outer, reweighting iterations for all experiments was set to 48, and the maximum number of inner iterations was set to 60, with early termination if \( \|d^{(t)} - d^{(t-1)}\|_2 / \|d^{(t)}\|_2 < 1 \times 10^{-6} \).
Figure 5.8: Comparison of rSNR versus undersampling ratio $M/N$ for MFISTA and bFISTA in the Shepp-Logan phantom.

For each optimization algorithm, plots of recovery SNR (rSNR) versus undersampling ratio ($M/N$) were generated.

5.2.2.2 Results

Figure 5.8 compares rSNR for bFISTA and MFISTA for the generalized LASSO L1 regularizer. At undersampling ratios close to 1, bFISTA performs slightly better than MFISTA (higher recovery SNR). For undersampling ratios below 0.5, performance of bFISTA and MFISTA are comparable. However, per iteration, bFISTA requires fewer computations of the sparsifying transform, and therefore can be more computationally efficient than MFISTA per iteration.
5.2.3 On the Comparison of a bFISTA Implementation of SPIRiT to Conventional Segmented Cine

In this study, the bFISTA approach described previously as implemented within the Gadgetron framework and deployed within the scanner environment was applied to SPIRiT with the goal of the comparison to conventional segmented cine of quantitative volumetric parameters.

The work presented in this study has been previously published as the following reference:

S.T. Ting, Y. Ding, H. Xue, S. Giri, N. Jin, R. Ahmad, O.P. Simonetti. Sub-30ms real-time, free-breathing cardiac cine with VISTA sampling and SPIRiT reconstruction: A comparison with conventional segmented cine. International Society for Magnetic Resonance in Medicine, 22nd Annual Scientific Session, Milan, Italy. May 2014.

5.2.3.1 Purpose

Combined with Variable density Incoherent Spatiotemporal Acquisition (VISTA, [132]), SPIRiT [85] reconstruction can provide an avenue for highly accelerated real-time, free-breathing cardiac imaging. However, practical clinical implementation requires efficient optimization algorithms and a pathway for direct reconstruction on the scanner in a time efficient manner. We achieve this by using the Fast Iterative/Shrinkage Thresholding Algorithm (FISTA, [102]) to reduce the computational cost of SPIRiT (see previous section), and by implementing this technique using GPU processing within the Gadgetron framework [15]. We demonstrate a clinically practical implementation of real-time cine accelerated to rate 15 with good image quality
and volumetric parameters that closely match those obtained using conventional segmented cine techniques.

5.2.3.2 Reconstruction

SPIRiT reconstruction was implemented within the Gadgetron framework using a $7 \times 7$ kernel. The VISTA sampling pattern was used to provide sufficient spatiotemporal incoherence for subsequent reconstruction. Spatiotemporal $\ell_1$-regularization within the 3D wavelet domain was implemented to take advantage of this sampling, allowing reconstruction of highly undersampled data. FISTA enables highly efficient iterative optimization by eliminating the need for a line search along the gradient direction, both reducing per-iteration computation time and improving convergence. All real-time data were reconstructed within the Gadgetron framework using an Intel Core i7 workstation with 64 Gb system memory and an NVIDIA GeForce GTX Titan with 6 Gb memory. Sliding window view-sharing across three frames and subsequent GRAPPA reconstruction using a $2 \times 11$ kernel was used as an initialization for the FISTA algorithm. The GRAPPA and SPIRiT kernels were estimated from the temporal average of all frames. A minimum change in the cost function $1 \times 10^{-4}$ was used as a stopping criterion.

5.2.3.3 Volunteer Study

All aspects of this human study were approved by the local Institutional Review Board. Short axis stacks (10-12 slices) were acquired at rest from six healthy volunteers (Siemens, Tim Trio, 3T, 32 channels) using (1) a conventional segmented cine technique (breath-held, 40 phases, acceleration rate: 3, 15 segments, 10 lines/segment) and (2) our real-time technique (free-breathing, 40 phases, acceleration rate:
Acquisition parameters common to both techniques were as follows: FOV: 240×240 mm², slice thickness: 8 mm, matrix size: 150 × 160, bandwidth (BW): 1488 Hz/pixel, TR/TE: 2.8 ms / 1.26 ms, FA: 50°, temporal resolution: 28.2 ms. To eliminate the need for registration between slices, a third series was acquired using the real-time technique during breath-hold for calculation of volumetric parameters. Datasets were visually scored by a qualified reader for image quality (1: nondiagnostic, 2: poor, 3: adequate, 4: good, 5: excellent), temporal fidelity of wall motion (1: nondiagnostic, 2: poor, 3: adequate, 4: good, 5: excellent), and artifact level (1: nondiagnostic, 2: severe, 3: moderate, 4: mild, 5: minimal). End-expiration diastolic and systolic phases were estimated from breath-held series (segmented and real-time), and endocardial contours were drawn within Argus (Siemens) and used to estimate end diastolic volume (EDV), end systolic volume (ESV), stroke volume (SV), and ejection fraction (EF).

5.2.3.4 Results

Figure 5.9 shows breath-held segmented cine images and free-breathing real-time images at end diastole and end systole. Paired t-test results show no significant differences in EDV, ESV, SV, and EF between breath-held segmented and breath-held real-time series (Figure 5.10). Figure 5.11 shows visual scoring (mean and standard deviation) for all datasets. While all scores were at least adequate in all real-time series (including free-breathing), paired t-test results indicate significant differences (1) for all metrics between the breath-held segmented and free-breathing real-time series (p = 0.025 for all comparisons) and (2) in artifact between breath-held segmented and breath-held real-time datasets (p = 0.012). No significant differences in any scores were observed between free-breathing and breath-held real-time images.
Figure 5.9: Example breath-held segmented (left) and free-breathing real-time (right) images at end systole (top) and end diastole (bottom).

<table>
<thead>
<tr>
<th>REAL-TIME</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>EDV (mL)</td>
<td>ESV (mL)</td>
<td>SV (mL)</td>
<td>EF (%)</td>
</tr>
<tr>
<td></td>
<td>119.23 ± 14.32</td>
<td>52.82 ± 6.32</td>
<td>66.42 ± 14.83</td>
<td>55.26% ± 6.73%</td>
</tr>
<tr>
<td>SEGMENTED</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>119.82 ± 16.04</td>
<td>52.15 ± 6.82</td>
<td>67.67 ± 17.22</td>
<td>55.89% ± 7.62%</td>
</tr>
<tr>
<td>p</td>
<td>0.598</td>
<td>0.487</td>
<td>0.330</td>
<td>0.406</td>
</tr>
</tbody>
</table>

Figure 5.10: Summary of volumetric parameters (mean ± std. dev.). P-values show no significant difference between breath-held segmented data and breath-held real-time data.
Average reconstruction time for each 40-frame image series within the Gadgetron framework was reduced to 56.2 seconds using FISTA from a previously reported 20 minutes using NLCG techniques [132] within MATLAB. With the use of multiple GPUs, we expect additional reduction in computation time.

5.2.3.5 Conclusion

Combined with VISTA and spatiotemporal regularization, the FISTA implementation of SPIRiT within the Gadgetron framework provides a clinically practical avenue for highly accelerated, real-time, free-breathing cardiac imaging in terms of overall image quality, quantification of volumetric parameters, and reasonable reconstruction time.

5.2.4 On the Comparison of a bFISTA Implementation of SENSE to Conventional Segmented Cine

In this study, the bFISTA approach described previously as implemented within the Gadgetron framework and deployed within the scanner environment was applied to SENSE with the goal of the comparison to conventional segmented cine of quantitative volumetric parameters.
The work presented in this study has been previously published as the following reference:


5.2.4.1 Purpose

The combination of Variable density Incoherent Spatiotemporal Acquisition (VISTA, [13]) with 3D spatiotemporal regularization has been shown to achieve comparable image quality relative to convention segmented cine techniques when combined with a SPIRiT reconstruction (see previous section). We combine these techniques with an iterative SENSE [75] reconstruction to achieve further speed improvements, allowing an avenue for highly accelerated real-time, free-breathing cardiac cine imaging with minimal reconstruction latency. We achieve $< 10$ second reconstruction times using a GPU-based implementation of the Fast Iterative Shrinkage Thresholding Algorithm (FISTA, [102]) and demonstrate a clinically practical implementation of real-time free-breathing cine meeting target guidelines for temporal and spatial resolution ($< 40$ ms and $< 2 \times 2$ mm$^2$, respectively) with sufficient diagnostic quality compared to segmented cine.

5.2.4.2 Reconstruction

SENSE reconstruction was implemented within the Gadgetron framework [15] with coil sensitivity maps estimated using a method by Walsh et al. [118]. The VISTA sampling pattern was used to provide sufficient spatiotemporal incoherence for
subsequent reconstruction. Spatiotemporal $\ell_1$-regularization within the 3D wavelet domain was implemented to take advantage of this sampling. Forward and backward wavelet transforms were applied to a single coil-combined channel, significantly reducing per-iteration computation time compared to coil-by-coil reconstruction methods. An efficient GPU-based wavelet transform implementation eliminated the need to perform an initial FFT in the readout direction for reduction of computation time, allowing us to take advantage of sparsity within all three spatiotemporal dimensions. FISTA enabled highly efficient iterative optimization by eliminating the need for a line search along the gradient direction, both reducing per-iteration computation time and improving convergence. All real-time data were reconstructed using an Intel Core i7 workstation with 64 Gb system memory and an NVIDIA GeForc GTX Titan with 6 Gb memory. No coil compression or image cropping was used to reduce computation time. The zero-filled k-space data was used as an initialization. A minimum change in the cost function of $1 \times 10^{-6}$ was used as a stopping criterion.

5.2.4.3 Volunteer Study

All aspects of this human study were approved by the local Institutional Review Board. Short axis (10-12 slices), 2 chamber and 4 chamber views were acquired at rest from 5 healthy volunteers (Siemens, Avanto, 1.5T 12 channels) using our real-time technique (free-breathing, 40 phases, rate 12) as well as a conventional segmented technique (breath-held at end-expiration, 40 phases, GRAPPA rate 2 with 28 reference lines). Common parameters between the two techniques were as follows: FOV $287 \times 340$ mm$^2$, slice thickness: 8 mm, matrix size: $156 \times 192$, spatial resolution: $1.8 \times 1.8$ mm$^2$, BW: 1447 Hz/pixel, TE/TR: 1.1 ms / 2.5 ms, FA: 70-79°, temporal resolution: 32.5 ms. A respiratory bellows signal was used to acquire real-time short
axis stacks during end-expiration. Reconstruction time for real-time data was automatically recorded from the time all acquired data were received until the time images were sent back to the scanner. Datasets were visually scored by two qualified readers for image quality (1: nondiagnostic, 2: poor, 3: adequate, 4: good, 5: excellent), and artifact level (1: nondiagnostic, 2: severe, 3: moderate, 4: mild, 5: minimal). Scores greater than or equal to 3 indicated diagnostic quality. Argus (Siemens) was used to estimate end diastolic volume (EDV), end systolic volume (ESV), stroke volume (SV) and ejection fraction (EF).

5.2.4.4 Results

Figure 5.12 shows example images from one volunteer for the short-axis view at both systole and diastole. Mean reconstruction time for SENSE was 7.8 seconds for each 40-frame image series, or 194 ms per frame. Even with a six-fold reduction in scan time and under free-breathing conditions, paired t-test results show no significant differences in EDV, ESV, SV, and EF between real-time and segmented data (Figure 5.13). Figure 5.14 shows mean image scores across all volunteers for all views. While paired t-test results indicated significant differences in scoring metrics between real-time and segmented data for both readers (p < 0.004 for all comparisons), all scores were at least adequate for real-time data, with the exception of three 2 chamber views in which a single reader gave a score of 2. In each of these cases, significant field-of-view wrapping was apparent, which introduces errors in coil sensitivity map estimation.
Figure 5.12: Example breath-held segmented (left) and free-breathing real-time (right) images at end systole (top) and end diastole (bottom).

<table>
<thead>
<tr>
<th>REAL-TIME</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>EDV (mL)</td>
<td>102.8 ± 19.2</td>
<td>44.8 ± 13.2</td>
<td>58.0 ± 8.3</td>
<td>57.2% ± 7.4%</td>
</tr>
<tr>
<td>ESV (mL)</td>
<td>45.0 ± 14.4</td>
<td>57.3 ± 8.6</td>
<td>56.9% ± 8.1%</td>
<td></td>
</tr>
<tr>
<td>SV (mL)</td>
<td>57.3 ± 8.6</td>
<td>56.9% ± 8.1%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EF</td>
<td>57.2% ± 7.4%</td>
<td>56.9% ± 8.1%</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 5.13: Summary of volumetric parameters (mean ± std. dev.). P-values show no significant difference between breath-held segmented data and free-breathing real-time data.
Figure 5.14: Scoring of artifact, temporal fidelity, and image quality (mean ± std. dev.) for all volunteers.

5.2.4.5 Conclusion

Combined with VISTA and 3D spatiotemporal regularization, the FISTA implementation of SENSE within the Gadgetron framework provides a clinically practical avenue for highly accelerated, real-time, free-breathing cardiac imaging in terms of overall image quality, quantification of volumetric parameters and low-latency reconstruction time.

The goal of this chapter is to introduce and present our contributions pertaining to the use of improved regularizers to account for variations in the sparsity model used for signal recovery of dynamic MRI. While the specific application of the techniques discussed in this chapter is cardiac cine MRI, the relevance of this work is general and can be applied to a wide variety of signal recovery problems.

6.1 Patch-Based Adaptive Regularizers for $\ell_1$ Compressive Recovery of Cardiac Cine MRI

In this section, we present a detailed investigation of our patch-based regularization techniques for compressive recovery of cardiac cine MRI.

6.1.1 Introduction

Magnetic resonance imaging (MRI) offers several distinct advantages over other medical imaging modalities, including the use of non-ionizing radiation and no known side effects on the human body. Dynamic MRI has shown particular utility especially in imaging the cardiac region. Real-time cine imaging [80, 53] allows for dynamic imaging of the heart under free-breathing conditions, greatly increasing the clinical utility of this imaging modality. Commonly used real-time imaging techniques utilize
parallel MRI (pMRI) techniques [107, 108] to achieve successful image recovery from undersampled k-space data, allowing increased spatial and temporal coverage of cardiac features. The introduction of compressed sensing (CS) [94, 133] has sparked a great amount of research activity in dynamic MRI related to image recovery at higher undersampling ratios. Effective application of CS to dynamic MRI requires that (1) the image to be recovered be sparse in some transform domain; (2) the artifact introduced due to undersampling in the k-space domain be incoherent, or noise-like, in this sparse transform domain; and (3) image recovery be achieved via a nonlinear method enforcing both image sparsity and data consistency [90]. To meet the second requirement for dynamic MRI, a high level of incoherence can be achieved by using a Cartesian trajectory with a pseudo-random sampling [13]. Recent developments related to fast recovery algorithms for large inverse problems [102, 112] have led to significant progress towards the third requirement. For MRI, the first requirement is typically satisfied by recognizing that most medical images are compressible in the discrete Wavelet domain [85]. However, successful image recovery via CS at high undersampling ratios can be highly dependent on the efficient use of varying levels of sparsity within the transform domain. In this work, we investigate the use of patch-based adaptive regularizers for improving CS-based image recovery of dynamic MRI.

6.1.2 Related Work

Many approaches have been proposed to solve the CS recovery problem in the general context of image processing. One commonly used scheme is LASSO [134] (also
known as basis pursuit denoising [135]). This approach, commonly used in synthesis-based CS, assumes the signal itself to be sparse and uses a single scalar regularization weight to balance the trade-off between enforcing of image sparsity and data consistency. In contrast, the generalized LASSO [136] assumes the data to be sparse in a transform domain, leading to analysis-based CS [110]. These approaches do not require any prior statistical information and can be conveniently solved via standard convex optimization tools; however, at high undersampling ratios, the reconstruction quality is often far from optimal. Algorithms incorporating statistical knowledge regarding the data, such as Bayesian CS recovery using message passing [137, 138], can achieve better reconstruction quality, but this requires knowledge of the statistical prior; mismatch between the selected prior and the underlying statistical model can result in additional error that is further amplified in linear inverse problems [139].

Recently [120], the use of composite regularizers has been proposed to take advantage of the observation that a signal \( \mathbf{x} \) often exhibit sparse representations in multiple dictionaries \( \Psi_d \) but with varying levels of sparsity across these dictionaries. In [120], the LASSO algorithm with L1 regularizer \( \|\Psi \mathbf{x}\|_1 \) is extended to the composite regularizer of the form \( \sum_d \lambda_d \|\Psi_d \mathbf{x}\|_1 \) with self-adjusting regularization weights \( \lambda_d \). This same family of composite regularizers is also applied to the well-known iteratively reweighted L1 algorithm [140], and in both cases the use of composite regularizers achieves significantly improved recovery SNR when compared to non-composite counterparts.

Nonlocal patch-based approaches [141] take advantage of pattern redundancy between image patches, leading to a reordering that produces a sparser representation compared to conventional CS techniques [142]. Similarity between image patches was
exploited in block-matching and 3D filtering for image denoising [141] and was used for image reconstruction from undersampled Fourier measurements [143]. In all of these results, it has been shown that patch-based methods can better preserve image details and edges compared to conventional approaches.

6.1.3 Contributions

The aim of this work is to investigate patch-based adaptive regularizers (PAR) in the context of image recovery for dynamic MRI. We propose two related approaches. In the first approach, a composite patch-based adaptive regularizer (C-PAR) is used, in that the sparse coefficients are segmented into non-overlapping patches and a single regularization weight is adaptively determined for each patch. In the second approach, a denoising patch-based adaptive regularizer (D-PAR) is used, in that for each sparse coefficient, a single regularization weight is adaptively determined based on a patch centered around that coefficient. We show through both static and dynamic numerical studies, that these two approaches can achieve improved image quality compared to the well-known LASSO approach as well as non-patch-based composite regularizers.

The remainder of this section is structured as follows. We review the basic signal model for pMRI and describe our proposed regularizers in this context. We then describe the numerical studies used to test our proposed approaches. Finally, we present the results of our numerical studies and conclude with some discussion of our results.

6.1.4 Theory

Mathematical notation in this section follows general typographical conventions. Bold capital letters represent matrices, and bold lowercase letters represent vectors.
Italic type is used for all letters representing variables, parameters, and elements of matrices and vectors. The names of well-known functions are written in lowercase upright letters, e.g. \( \log(\cdot) \). Certain important constructs are referred to by blackboard-bold letters, e.g. \( \mathbb{C} \) for the set of complex numbers. Subscripts denote indices for elements of matrices and vectors. A superscript in parentheses denotes the iteration number of a given iterative method.

### 6.1.5 Problem Formulation for Unregularized pMRI

Signal recovery for MRI may be formulated as the problem of recovering a signal (or image) from the following noisy measurements:

\[
\mathbf{y} = \Phi \mathbf{x} + \mathbf{w} \in \mathbb{C}^M, \tag{6.1}
\]

where \( \mathbf{x} \in \mathbb{C}^N \) is a vectorized \((N_1 \times N_2)\)-pixel image with \( N = N_1 \times N_2 \), \( \Phi \in \mathbb{C}^{M \times N} \) is a known sub-sampled Fourier measurement operator and \( \mathbf{w} \in \mathbb{C}^M \) is additive noise. In general, we are interested in the case where \( M \ll N \).

In the case of dynamic MRI, we are interested in recovering a sequence of \( T \) images, e.g. \( \mathbf{x} = [\mathbf{x}_1^H, \cdots, \mathbf{x}_T^H]^H \in \mathbb{C}^N \), where \( N = N_1 N_2 \times T \), from a sequence of measured images, \( \mathbf{y} = [\mathbf{y}_1^H, \cdots, \mathbf{y}_T^H]^H \in \mathbb{C}^M \), given a known measurement operator \( \Phi = \mathcal{D}(\Phi_1, \cdots, \Phi_T) \). Here \( H \) denotes the conjugate transpose, and \( \Phi_i \in \mathbb{R}^{(M/T) \times N_1 N_2} \). The operator \( \mathcal{D} \) places the operators \( \Phi_i \) along the diagonal entries of a matrix with zero elements elsewhere, such that each operator \( \Phi_i \) is applied to the \( i^{th} \) image. Again, we are typically interested in scenarios where \( M \ll N \). The signal
problem may then be expressed in detail as the following:

\[
\begin{bmatrix}
\mathbf{y}_1 \\
\vdots \\
\mathbf{y}_T
\end{bmatrix} =
\begin{bmatrix}
\Phi_1 & \mathbf{0} \\
\vdots & \ddots \\
\mathbf{0} & \Phi_T
\end{bmatrix}
\begin{bmatrix}
\mathbf{x}_1 \\
\vdots \\
\mathbf{x}_T
\end{bmatrix} +
\begin{bmatrix}
\mathbf{w}_1 \\
\vdots \\
\mathbf{w}_T
\end{bmatrix}
\] 

(6.2)

Parallel MRI (pMRI) seeks to recover a dynamic image sequence through the use of multi-coil arrays. This problem may, in general be formulated as two related but distinct inverse problems. The first formulation employs a coil-by-coil interpolation of k-space. Two example k-space methods are GRAPPA [74] and SPIRiT [85]. In the case of SPIRiT, the measurement operator \(\Phi\) is defined as \(\Phi_k = \mathcal{D}(S_1, \cdots, S_T)\), where \(S_i\) denotes a k-space sampling operator that can vary with time. The interpolation kernels are estimated from full-sampled calibration data, generating an additional interpolation operator \(G := \mathcal{D}(G_1, \cdots, G_T)\). The unregularized optimization problem for the SPIRiT method may then be expressed as follows, where we seek to recover the multiple-channel k-space data \(k_m\):

\[
\hat{k}_m = \arg\min_{k_m} \left\| \Phi_k^H \lambda (G - I)^H k_m - \left[ I \mathbf{0} \right]^H \mathbf{y} \right\|_2^2.
\] 

(6.3)

Here, \(\lambda\) is the parameter for balancing between the contributions of the measurement and interpolation operators. By letting \(k_m = Fx_m\), with \(x_m\) here denoting the multiple channel data in the image domain, the k-space-based formulation in (6.3) can be equivalently expressed in a manner that seeks to recover multiple channel image data:

\[
\hat{x}_m = \arg\min_{x_m} \left\| \Phi_k^H \lambda (G - I)^H Fx_m - \left[ I \mathbf{0} \right]^H \mathbf{y} \right\|_2^2.
\] 

(6.4)

The second formulation seeks to recover a single image sequence from undersampled multiple-coil k-space data. SENSE is the most commonly used image-based
method. In this case, $\Phi_x = D(S_1 FC_1, \cdots, S_T FC_T)$, where $F$ is the 2D forward Fourier transform, and $C_1, \cdots, C_T$ denote the coil sensitivity maps estimated from the fully-sampled calibration data. The optimization problem for these image-based methods can be expressed as follows, where $x$ denotes the recovered single-channel image data:

$$\tilde{x} = \arg\min_x \| \Phi_x x - y \|_2^2. \quad (6.5)$$

Denoting $Q$ as either $[\Phi_k^H \lambda G^H]^H F$ in (6.4) or $\Phi_x$ in (6.5) and letting $d$ denote the data to be recovered, the unregularized pMRI problem for both SPIRiT and SENSE may be unified as follows:

$$\tilde{d} = \arg\min_d \| Qd - \bar{y} \|_2^2, \quad (6.6)$$

where $\bar{y} = [y^H \ 0^H]^H$ is a vector that augments $y$ with zeros such that the number of elements corresponds to the number of rows in $Q$. This allows us to concisely consider the compressive recovery problem for pMRI with $\ell_1$-regularization.

6.1.5.1 Composite Regularization for Analysis Compressed Sensing pMRI

Compressed sensing approaches can be used to obtain solutions to the pMRI problem posed in (6.6) via the use of $\ell_1$-regularization within a sparse domain. For any generic sparsifying frame representation, let $\Psi$ be an analysis operator such that $\alpha = \Psi d$ is sparse for a given signal $d$. Let $\Psi_i$ denote one of the $D$ sub-dictionaries of $\Psi$ for $i = 0, \cdots, D - 1$. Then the generalized LASSO approach for analysis CS seeks to solve the following optimization with manually-tuned regularization parameter $\mu$:

$$\tilde{d} = \arg\min_d \gamma \| Qd - \bar{y} \|_2^2 + \mu \sum_i \| \Psi_i d \|_1^1. \quad (6.7)$$
Here, $\gamma$ is a tuning parameter reflecting knowledge of the noise level. The composite regularization approach proposed in [120, 144] recognizes that the different sub-dictionaries $\Psi_i$ may exhibit varying levels of sparsity; therefore, accounting for this sparsity variation through the use of individually tuned regularization parameters $\mu_i$ may improve signal recovery. Hence, the composite regularization approach for analysis CS seeks to solve the following optimization problem:

$$\hat{d} = \arg\min_d \gamma \|Qd - \tilde{y}\|_2^2 + \sum_i \mu_i \|\Psi_id\|_1.$$  \hspace{1cm} (6.8)

While the individual weights may be manually selected by the user [13], they may also be estimated by iteratively maximizing the log-likelihood function $L(\mu_i|\Psi_id)$. For an i.i.d. Laplacian prior on $\Psi_id$, maximizing the log-likelihood function yields the following estimate for the composite regularization parameters [120]:

$$\hat{\mu}_i = \frac{N}{\sum_{j=1}^N \|\Psi_id\|_j + \epsilon},$$  \hspace{1cm} (6.9)

where $[\Psi_id]_j$ denotes the $j^{th}$ coefficient of $\Psi_id$ and $N$ is the total number of coefficients in that sub-dictionary. Here, $\epsilon > 0$ is a small regularization term whose role is to prevent the denominator from becoming zero.

6.1.5.2 Patch-Based Adaptive Regularization for Analysis Compressed Sensing pMRI

Motivated by the observation that the level of sparsity can exhibit spatial variation within a sparsifying frame representation, we introduce the notion of patch-based adaptive regularizers in this section.

Let $P_x(\Psi_id, B)$ be a patch-extraction operator that extracts the transform coefficients in the neighborhood $B$ centered around the location $x$ from a sub-dictionary of...
the transform dataset denoted by $\Psi_i d$. For example, in the two-dimensional case and a square patch of size $2M + 1$, then $\mathcal{B} = [-M, M] \times [-M, M]$, and $\mathbf{P}_{x}(\Psi_i d, \mathcal{B})$ extracts the transform coefficients from the neighborhood $x + \mathcal{B}$ for the $i^{th}$ sub-dictionary $\Psi_i$. This definition can be readily extended to three dimensions, which will be necessary for image recovery in the context of dynamic MRI.

Patch-based adaptive regularization (PAR) may be introduced through a diagonal matrix for each sub-dictionary $\Psi_i d$. The optimization problem may then be expressed as follows:

$$\hat{d} = \arg\min_d \gamma \|Qd - \bar{y}\|_2^2 + \sum_i W_i \|\Psi_i d\|_1,$$

(6.10)

where $W_i$ is a diagonal weighting matrix that adjusts the regularization strength for each coefficient in the $i^{th}$ sub-dictionary. It is straightforward to see that for generalized LASSO, $W_i = \mu I$, and that for composite regularization, $W_i = \mu_i I$.

Therefore, for PAR, we seek to introduce a suitable weighting matrix $W_i$ for each sub-dictionary that adjusts the regularization strength for each patch. For notational simplicity, we consider the two-dimensional case here; the extension to additional dimensions is straightforward.

PAR may be implemented in two ways. In the first approach, patches are defined to be adjacent and non-overlapping; we term this approach composite patch-based adaptive regularization (C-PAR). Given a patch size of $2M + 1$ and a neighborhood $\mathcal{B}$ as defined previously, each sub-dictionary $\Psi_i d$ may be divided into adjacent, non-overlapping patches of size $2M + 1$. Let $x_p$ be the center location of one of these patches in a sub-dictionary $\Psi_i d$. Denoting the diagonal element in the weighting matrix $W_i$ corresponding to the weight for the $j^{th}$ element in the patch covering the neighborhood $x_p + \mathcal{B}$ as $\hat{\mu}_{i,j}(x_p + \mathcal{B})$, each weight may be adaptively updated using
composite regularization via the following rule:

\[ \hat{\mu}_{i,j}(x_p + B) = \frac{N_p}{\sum_{k=1}^{N_p} |[P_{x_p}(\Psi_i d, B)]_k| + \epsilon}, \]  

(6.11)

where \([P_{x_p}(\Psi_i d, B)]_k\) denotes the \(k^{th}\) coefficient in the patch centered at \(x_p\) and \(N_p\) denotes the number of coefficients in the patch. Hence, the same regularization weight as calculated by (6.11) is applied to every coefficient within the patch.

In the second approach, patches are allowed to be overlapping such that the regularization weight for each coefficient is estimated by considering a neighborhood surrounding the coefficient; we term this approach denoising patch-based adaptive regularization (D-PAR). Denoting the \(j^{th}\) diagonal element in the weighting matrix \(W_i\) as \(\hat{\mu}_{i,j}\), each diagonal element is individually updated using composite regularization according to the following rule:

\[ \hat{\mu}_{i,j} = \frac{N_p}{\sum_{k=1}^{N_p} |[P_x(\Psi_i d, B)]_k| + \epsilon}, \]  

(6.12)

where \(x\) now denotes the location of the \(j^{th}\) element.

PAR may be incorporated into the optimization routine using an inner/outer loop structure, with the inner loop solving the convex optimization for a given value of \(\hat{\mu}_{i,j}\) and the outer loop updating the value of \(\hat{\mu}_{i,j}\).

### 6.1.6 Numerical Results

We now present results from our numerical studies examining the performance of our proposed methods C-PAR and D-PAR. In the following sections, we describe and present results for three experiments that focus on the problem of recovering an \(N\)-pixel image or image series \(x\) from \(M\) noisy compressed measurements \(y = \Phi x + w\) with \(M \ll N\). In the first experiment, we recover the Shepp-Logan phantom and a
generic brain image. In the second experiment, we recover a dynamic image series designed to mimic a cardiac MRI image sequence. In the third experiment, we recover a dynamic in vivo cardiac MRI image sequence.

In each of the three experiments, we compare our two proposed methods with (1) the standard non-composite L1 regularizer as well as its composite analogue proposed in [120]. We term these two methods “L1” and “C-L1” respectively, and we term our two proposed methods “CPAR-L1” and “DPAR-L1.”

6.1.6.1 Experimental Setup

For the first experiment, in which we recover a static 2D image, a “spread spectrum” operator [131] of the form $\Phi = DFC$ is used, with $C \in \mathbb{R}^{N \times N}$ being a diagonal matrix with i.i.d. equiprobable $\pm 1$ entries, $F \in \mathbb{C}^{N \times N}$ being the discrete Fourier transform (DFT), and $D \in \mathbb{R}^{M \times N}$ being a row-selection operator selecting $M$ rows of $FC \in \mathbb{C}^{N \times N}$ uniformly at random.

For the second and third experiments, in which we recover a dynamic image series, the measurement operator $\Phi$ is constructed using randomly sub-sampled Fourier measurements at each time point with a varying sampling pattern across time.

In all experiments, added noise $w$ was zero-mean, white, and circular Gaussian (i.e., independent real and imaginary components of equal variance). Denoting the noise variance by $\sigma^2$, we define the measurement signal-to-noise ratio (SNR) as $\|y\|_2^2/(M\sigma^2)$ and the recovery SNR of the signal estimate $\hat{x}$ as $\|x\|_2^2/\|x - \hat{x}\|_2^2$.

For all experiments, the parameter $\gamma$ was set to be $1/\sigma^2$ and the regularization term $\epsilon$ was set to $1 \times 10^{-2}$. For the L1 regularizer, in order to ensure that the selected value for $\gamma$ corresponded to the oracle-tuned value, experiments were run using values
of $\gamma \times 2/3$ and $\gamma \times 3/2$. No significant variation in recovery SNR curves was observed, and therefore $\gamma$ was kept fixed to a value of $1/\sigma^2$.

An image-based approach to reconstruction is used in all experiments (e.g. $Q = \Phi_x$). FISTA [102] is used to implement the optimization needed for all methods. The maximum number of outer, reweighting iterations for C-L1, CPAR-L1, and DPAR-L1 was set to 48, and the maximum number of inner FISTA iterations was set to 60, with early termination if $\|x^{(t)} - x^{(t-1)}\|_2 / \|x^{(t)}\|_2 < 1 \times 10^{-6}$.

### 6.1.6.2 Shepp-Logan and Brain Image Recovery

For the first experiment, we examine the algorithmic performance compared to sampling ratio $M/N$ in recovering 2D static images according to the signal model presented in (6.1). We use the Shepp-Logan phantom and a generic brain image with $N = 128 \times 128$ as shown in Figure 6.1. Compressed noisy measurements $y$ were constructed using the spread-spectrum $\Phi$ and AWGN $w$ with a measurement SNR of 40 dB for both images. The analysis operator $\Phi \in \mathbb{R}^{4N \times N}$ was constructed using the undecimated db1 2D wavelet transform (UWT-2d-db1) with a single level of decomposition. The C-L1 method treated each sub-band of the wavelet transform as a separate sub-dictionary $\Psi_i$, while CPAR-L1 and DPAR-L1 further divide each sub-dictionary into patches for patch-based composite regularization. Both CPAR-L1 and DPAR-L1 utilized a patch size of 7.

Figure 6.2 shows the recovery SNR versus sampling ratio $M/N$ for the Shepp-Logan phantom, and Figure 6.3 shows the same for the brain image. For both the Shepp-Logan phantom and the brain image, CPAR-L1 and DPAR-L1 outperform L1 for $M/N \geq 0.175$, with DPAR-L1 slightly outperforming CPAR-L1 in the Shepp-Logan phantom and CPAR-L1 slightly outperforming DPAR-L1 in the brain image.
Figure 6.1: (a) The Shepp-Logan phantom of size $N = 128 \times 128$. (b) The brain image of size $N = 128 \times 128$.

In the Shepp-Logan phantom, CPAR-L1 and DPAR-L1 outperform C-L1 for $M/N \geq 0.175$, and they outperform C-L1 in the brain image for $M/N \geq 0.5$.

6.1.6.3 Recovery of a Dynamic Digital Phantom

In the second experiment, we examine algorithmic performance compared to sampling ratio $M/N$ in the context of recovering a dynamic digital phantom [145] using the signal model presented in (6.2) with $N = N_1 \times N_2 \times T = 144 \times 144 \times 40$. The phantom consisted of both dynamic and static features (Figure 6.4) with the dynamic ellipses exhibiting a periodic motion. To simulate experimental conditions typically used in reconstruction of dynamic MRI, 12 uniformly spaced circular receiver coils were simulated around the phantom using the Biot-Savart law.

The measurement operator $\Phi$ was constructed from $N_1 N_2 \times N_1 N_2$ 2D DFT sampling operators that pseudo-randomly selected $M/N \times N_2$ rows from each image.
Figure 6.2: Recovery SNR versus sampling ratio $M/N$ for the Shepp Logan phantom. Measurements were constructed using the spread-spectrum operator and AWGN at 40 dB SNR, and recovery used UWT-2d-db1 at one level of decomposition.

frame. The sampling pattern is varied with time, and regions containing most of the signal energy (e.g. low frequencies) are sampled more densely according to a variable density sampling scheme [13]. Sampling ratio ranged from $M/N = 1/2$ to $M/N = 1/15$. Compressed noisy measurements $y$ were created by adding AWGN $w$ with measurement SNR of 30 dB. The analysis operator $\Phi \in \mathbb{R}^{8N \times N}$ was constructed using the undecimated db1 3D wavelet transform (UWT-3d-db1) with a single level of decomposition. Signal recovery was performed using L1, C-L1, CPAR-L1, and DPAR-L1.
Figure 6.3: Recovery SNR versus sampling ratio $M/N$ for the brain image. Measurements were constructed using the spread-spectrum operator and AWGN at 40 dB SNR, and recovery used UWT-2d-db1 at one level of decomposition.

Figure 6.5 shows the recovery SNR versus sampling ratio $M/N$ for the dynamic digital phantom for a measurement SNR of 30 dB. CPAR-L1 and DPAR-L1 outperform both L1 and C-L1 at all undersampling ratios. Figure 6.6 shows the temporal profiles from the results for L1, C-L1, CPAR-L1, and DPAR-L1. From both recovery SNR plots and temporal profiles, CPAR-L1 and DPAR-L1 exhibit similar performance.
Figure 6.4: (a) Example frame from the digital phantom. (b) Temporal profile for the digital phantom. (c) Example sampling pattern for $M/N = 1/12$.

### 6.1.6.4 Recovery of a Dynamic Cardiac MRI Dataset

In the third experiment, we examine algorithmic performance and image quality in the context of recovering an \textit{in vivo} dynamic cardiac MRI dataset using the signal model presented in (6.2) with $N = N_1 \times N_2 \times T = 256 \times 258 \times 38$. A single, fully sampled, breath-held, segmented acquisition was acquired at 3T (Siemens, Tim Trio) using a 32 channel coil array from one healthy volunteer. Informed consent was obtained in accordance with the local Institutional Review Board. Relevant scan parameters are as follows: field of view: $256 \times 258 \text{ mm}^2$, matrix size: $256 \times 256$, spatial resolution: $1 \times 1 \text{ mm}^2$, slice thickness: 6 mm, 6 lines/segment, temporal resolution: 11.2 ms.

The fully sampled dataset was used as the ground truth. Figure 6.7 shows an example frame as well as a temporal profile from the ground truth. As in the previous
Figure 6.5: Recovery SNR versus sampling ratio $M/N$ for the digital phantom. Measurements were constructed using AWGN at 30 dB SNR, and recovery used UWT-3d-db1 at one level of decomposition.

In the experiment, the measurement operator $\Phi$ was constructed from $N_1N_2 \times N_1N_2$ 2D DFT sampling operators that pseudo-randomly selected $M/N \times N_2$ rows from each image frame. No additional noise was added to the acquired data. A variable density sampling pattern similar to that used in the dynamic digital phantom experiment was used with sampling ratios $M/N = 1/9$.

Figure 6.8 shows reconstruction results for the experiment with sampling ratio $M/N = 1/9$. Example reconstructed frames corresponding to the same frame displayed in Figure 6.7a are shown as well as temporal profiles. The temporal profile for the L1 results show signal variation across frames. This is lessened in the C-L1 results.
Figure 6.6: (a) Temporal profile from recovery results for the digital phantom with measurement SNR of 30 dB at $M/N = 1/12$. (a) L1, (b) C-L1, (c) CPAR-L1, (d) DPAR-L1.

Figure 6.7: (a) Example frame from ground truth image series for dynamic cardiac MRI experiment. (b) Temporal profile for ground truth image series.
with the accompanying tradeoff of increased blurring of dynamic features. Dynamic features are restored in CPAR-L1 and DPAR-L1 results while maintaining reduced signal variation across frames. Recovery SNR in region of interest surrounding the heart compared to the ground truth for L1, C-L1, CPAR-L1, and DPAR-L1 are as follows, respectively: 14.01 dB, 19.60 dB, 19.47 dB, 19.78 dB.

6.1.7 Discussion and Conclusions

Motivated by the observation that a given signal $\mathbf{x}$ admits sparse representations in a sparsifying frame representation with varying levels of sparsity across and also within sub-dictionaries, we propose two approaches for introducing patch-wise adaptive regularization for signal recovery of (approximately) sparse signals from noisy linear measurements. The first approach, CPAR-L1, introduces patch-wise regularization for $\ell_1$ penalties of the form (6.10) in the context of adjacent and non-overlapping patches. This is a direct extension of a recently proposed composite regularization technique [120] that accounts for varying levels of sparsity across but not within sub-dictionaries. The second approach, DPAR-L1, introduces patch-wise regularization for $\ell_1$ penalties of the form (6.10) but with overlapping patches. This approach is similar to the denoising technique proposed in [146]. Whereas their proposed technique utilized a Gaussian prior, in this work, a Laplacian prior on the distribution of sparse coefficients is used to estimate the regularization weights.

Static numerical experiments indicate that DPAR-L1 slightly outperforms CPAR-L1 in the case of the Shepp-Logan phantom, whereas the converse is true in the case of the brain image. However, this difference is reduced when moving to studies using both the dynamic digital phantom and in vivo dynamic MRI data. However, the
Figure 6.8: Example reconstructed frames and temporal profiles from dynamic cardiac MRI results for $M/N = 1/9$. The same frame as that of the ground truth is used. 

(a-b) L1 (rSNR: 14.01 dB), (c-d) C-L1 (rSNR: 19.60 dB), (e-f) CPAR-L1 (rSNR: 19.47 dB), (g-h) DPAR-L1 (rSNR: 19.78 dB).

The use of non-overlapping patches may be of interest in particular when combining this technique with nonlocal shrinkage approaches [147] or approaches for deriving mutual information based on the Kullback-Leibler distance [148].
The in vivo experiment using dynamic MRI data demonstrate that both CPAR-L1 and DPAR-L1 can achieve the same amount of reduction in temporal signal variation compared to C-L1. Furthermore, additional fine details lost in the C-L1 signal recovery are preserved when PAR is used. This can be especially useful in reconstruction of highly undersampled dynamic MRI for preserving fine details necessary for diagnostic evaluation.

6.2 Patch-based Adaptive Iteratively-Reweighted Sparse Compressive Recovery for Reconstruction of Highly Accelerated Exercise Stress Cardiac Cine

In this section, we present some preliminary results pertaining to the application of our patch-based regularization approaches in the context of compressive image recovery of highly accelerated exercise stress cardiac cine images. Portions of this work have been accepted for presentation at the SCMR 19th Annual Scientific Sessions, Los Angeles, CA, USA, January 2016.

6.2.1 Introduction

Real-time exercise stress cardiac magnetic resonance imaging is challenging due to exaggerated breathing motion and high heart rates. Through improvements in image reconstruction, it may be possible to improve the reliability and diagnostic accuracy of this difficult imaging application.

Like most natural images, dynamic cardiac images possess a rich structure that can be exploited to aid image reconstruction by enforcing sparsity in an appropriate transform domain, e.g. in the undecimated wavelet transform (UWT) domain.
When using UWT or its decimated counterpart, standard techniques achieve $\ell_1$ regularization through the use of a single weighting rule (regularization strength) across different sub-bands [85]. Since the level of sparsity varies across sub-bands, it has been shown that iteratively adapting the individual regularization strengths for each sub-band can improve the recovery process [120].

However, levels of sparsity may vary significantly not only between but also within sub-bands. Therefore, taking advantage of this finer-grained variation may further improve reconstruction results, especially in scenarios where severe motion is present.

### 6.2.2 Purpose

In this work, we demonstrate that the use of a patch-based, iteratively reweighted approach for compressive recovery, in which weighting rules are individually determined for spatiotemporal patches, can improve image reconstruction of exercise stress cardiac images relative to standard compressive recovery techniques.

### 6.2.3 Methods - Volunteer Study

Exercise stress cine images in the short-axis and long-axis orientation were acquired from three healthy volunteers using a 1.5T (Siemens, Avanto) scanner with a 32-channel body coil array. All aspects of this volunteer study were approved by the local Institutional Review Board. Data were collected at acceleration rates 6 and 8 using a VISTA [13] sampling pattern.

Acquired data were reconstructed using a SENSE-based reconstruction with $\ell_1$ regularization in the 3D spatiotemporal discrete wavelet domain. Two approaches were used for adjusting $\ell_1$ regularization in the sparse domain. In the first approach (FSW - fixed single weighting), a fixed weighting rule was used across all
sub-dictionaries. In the second approach (APDW - adaptive patch-based weighting), subdictionaries were divided into 7x7x7 patches, and weighting rules were calculated for each patch using an adaptive method [120].

All reconstruction code was implemented in MATLAB using an Intel Core i5 with 16 Gb of system memory. An inner and outer iteration structure was used for the iterative reconstruction with the adaptive weights being updated for each outer iteration. A total of 60 inner iterations and 48 outer iterations was used.

6.2.4 Results

For the volunteer study, Figures 6.9 and 6.10 show example frames and temporal profiles from data collected at acceleration rate 6 in the two-chamber long-axis orientation. Figures 6.11 and 6.12 show example frames and temporal profiles from data collected at acceleration rate 8 in the short-axis orientation.

In both examples, the use of adaptive patch-based weighting can help improve overall image quality by reducing artifact level and improving sharpness of dynamic edges.
Figure 6.10: Temporal profiles along line indicated in cine frame for each weighting method for the volunteer shown in Figure 6.9. APBW shows improved sharpness in dynamic edges (arrow).

Figure 6.11: Example exercise stress images from one volunteer in the short-axis orientation at acceleration rate 8.
Figure 6.12: Temporal profiles along line indicated in cine frame for each weighting method for the volunteer shown in Figure 6.11. APBW shows improved sharpness in dynamic edges (arrow).

6.2.5 Conclusions

Patch-based, adaptive, iteratively reweighted compressive recovery techniques can be used to take advantage of structured sparsity in exercise stress cardiac MRI, leading to improved quality compared to standard techniques.
Chapter 7: Fast MRI Reconstruction within the Gadgetron Framework: A Practical Implementation

The primary purpose of this chapter is to provide a thoroughly exhaustive (and potentially exhausting) description of the practical implementation of the reconstruction approaches described in this work as implemented within the Gadgetron framework [15]. The secondary purpose of this chapter is to provide a thorough technical description of the code contributions by the author within the Gadgetron framework. We seek to provide the reader a general overview of the role played by code written within the Gadgetron framework in enabling a practical, feasible reconstruction for real-time, cardiac cine MRI. Ultimately, we seek also to provide the interested reader a thorough description (though perhaps not documentation) of the author’s code contributions within this framework throughout the course of his graduate studies. This includes, but is not limited to, the code specifically related to the approaches discussed in this work. The purpose here is then to provide a description thorough enough such that the interested reader may begin an independent study of the code itself with the goal of further development.

As of the writing of this work, the latest code for the Gadgetron framework may be found at the following website: http://gadgetron.github.io. As the Gadgetron
framework is a rapidly moving development target, should this website not be available, the interested reader may be able to obtain more information by contacting the authors of [15].

The code discussed in this chapter was written using version 3.9.1 of the Gadgetron (SHA-1 Hash: ac50185da333298a60a1f3f33655df87c3c2a60c). As development of the Gadgetron framework is still rapidly moving, it is not guaranteed that the code will immediately be compatible with the latest version of the Gadgetron due to rapidly changing application programming interfaces (APIs). While we present some portions of code within this chapter, for brevity’s sake, we choose not to include the entire codebase as an inline reference. The code in its entirety can be made available upon request of the author.

7.1 Some Remarks on the Motivation Behind the Gadgetron

As discussed in detail in [15], the Gadgetron framework, or simply, the “Gadgetron,” is a flexible system for creating a streaming data processing pipeline for processing raw input data to generate reconstructed images. Therefore, it is a flexible code framework for implementing image reconstruction algorithms. The primary motivation behind the Gadgetron draws from the observation that most “proof-of-concept” implementations of proposed reconstruction algorithms within the MRI community are within closed “offline” programming environments that do not allow for implementation directly within the scanner environment. In other words, the process for reconstructing data typically involves (1) capturing the raw data from the MRI scanner, (2) physically transporting the raw data to an external machine, (3) generating reconstructed images within the offline programming environment, and
generating final images for viewing or analysis. This final step typically does not occur within the environment of the patient database used clinically. This laborious process is often considered secondary to the potential advantages of fast scientific prototyping of reconstruction algorithms within the offline programming environment. However, this also becomes a barrier or hindrance to practical implementation with the goal of clinical relevance of the proposed algorithms.

The Gadgetron framework therefore seeks to address this issue by providing a programming environment amenable implementation of reconstruction algorithms directly within the scanner environment. The framework is modular, in that all data processing occurs on an external computer outside the vendor-specific MRI computing environment. This also addresses a second issue of code reproducibility and distribution of code between multiple vendors.

7.2 An Architectural Overview of the Gadgetron

In this section, we seek to give a broad architectural overview of the Gadgetron in terms of its relation to a vendor-nonspecific MRI system as well as a general view of the processing pipeline utilized within the framework itself.

Figure 7.1 shows the reconstruction system for a vendor-nonspecific MRI system. The complete MRI system can be viewed from a system engineering perspective as a measurement device (MRI scanner) that generates raw spectroscopic data. This data must be processed in some way to generate user-relevant data (either in terms of images or measurements). This processing is carried out by an additional computer system within the MRI system itself. The code framework and processing pipeline are often proprietary, and their design is vendor-specific. Therefore, code written for
Figure 7.1: Diagram showing the relation between the Gadgetron framework and a vendor-nonspecific MRI system.
a specific vendor in general may not run on an MRI system from another vendor. Logically, this pipeline may be viewed as a “stream” of modules that operate on the data to eventually produce user-relevant data that is then sent to the viewing console for user evaluation. This process is denoted by the Product Reconstruction box in the figure.

The Gadgetron framework exists on an external computer separate from the MRI system, as denoted by the Gadgetron Reconstruction box in Figure 7.1. Physically, this external computing environment can be connected to the scanner via a TCP/IP network connection; the manner in which this must physically be accomplished depends on the MRI machine vendor as well as the setup of the medical network environment. We cover some of the details pertaining to this aspect of the Gadgetron in a later section. Logically, the Gadgetron framework serves to bypass vendor-specific (product) modules in order to implement an independent reconstruction processing pipeline consisting of “Gadgets” within its own programming environment. This is implemented by inserting two vendor-specific modules within the product reconstruction pipeline, termed in Figure 7.1 “Gadgetron Emitter” and “Gadgetron Injector.” The Gadgetron Emitter plays the role of redirecting data from a certain point in the product reconstruction pipeline to the Gadgetron. In our context this data is raw k-space data; however, data in other formats at other points in the reconstruction pipeline can also be redirected. It may also be possible to redirect data from multiple points in the processing pipeline to one or more instances of the Gadgetron framework that exist either on the same or different physical computers. The Gadgetron Injector plays the role of inserting the final processed data from the Gadgetron back into the product reconstruction pipeline. Typically this occurs
at the end of the product reconstruction pipeline, essentially bypassing the product reconstruction. Again, one is not restricted to a single instance of the **Gadgetron Injector**; it may be an interesting exercise to consider the use of multiple instances at different points in the product reconstruction pipeline.

The code for implementing and inserting both the **Gadgetron Emitter** and **Gadgetron Injector** exists within the vendor-specific programming environments, and as such, are likewise vendor-specific. All code implementations described in this work are within the Siemens ICE programming environment, and as such, there exists a small amount of code for inserting these modules. There also exists a small amount of code for mapping Siemens-specific parameters to parameters that can be understood by the Gadgetron. Due to intellectual property constraints, we refrain from discussing specific implementation details pertaining to this mapping code, which is typically made available via vendor-specific channels. Other vendors have their own vendor-specific parameters, so this mapping is also vendor specific. In an attempt to somehow standardize the format for raw MRI data, the ISMRM (International Society for Magnetic Resonance in Medicine) raw data (ISMRMRD) format was proposed at the ISMRM Sedona 2013 workshop. The code for this standardized ISMRMRD format is publicly available at the following website: gadgetron.github.io. If the website has moved or is otherwise not available, the code may be obtained by contacting the authors of [15]. The Gadgetron framework reads raw data in the ISMRMRD format, and one of the primary roles played by the vendor-specific mapping code is to map the vendor-specific data to this ISMRMRD format.

Schematically, the processing pipeline for the Gadgetron is shown in Figure 7.2. The Gadgetron connects to the client application via a TCP/IP connection. The
Figure 7.2: An overview of the Gadgetron architecture. Source [15]. The Gadgetron communicated with a client application through a TCP/IP connection. The client application sends data to the Gadgetron and associated with each data package is a Message ID. Based on the Message ID, control of the socket is handed over to a specific Reader, which is capable of deserializing the incoming data package. The data package is converted to message blocks that are added to the first Gadget’s queue. Data are then passed down the Gadget stream where each Gadget can modify and transform the data. Any Gadget can return images (or partially processed data) to the Gadgetron framework. Based on the Message ID of this return data package, the control of the socket and the data are handed to a particular Writer, which is responsible for writing the return message to the client.
client application is typically, but not always, the MRI scanner environment itself. The vendor-specific code within the scanner environment is responsible for packaging raw measured data into messages which are then sent to the Gadgetron via TCP/IP. Once received by the Gadgetron, the Message ID is parsed, and a specific reader is used to read the data contained in the message. The processing pipeline is itself a stream of gadgets, the order of which is specified by an XML file. Once the raw data is parsed by a specific low-level reader, it is sent to the first gadget in the stream for processing. This process of receiving and parsing messages is carried out by the GadgetStreamController until the end of the data stream from the client application is reached.

As data is received by the gadget stream, it is processed until final image data is generated. The specific design of this gadget stream as well as the shared toolboxes utilized is flexible. Each gadget itself can return processed (or unprocessed) data to the GadgetStreamController independently. Typically, processed data is returned by the final gadget in the stream. When a message is received by the GadgetStreamController, it is again parsed for a Message ID which serves to identify which low-level writer should be used to write the data out to the TCP/IP connection. The role of the writer is then to parse the data and prepare it for sending back to the client application.

The vendor-specific code responsible for inserting the Emitter and the Injector provides the interface for converting a vendor-specific parameter specification to the standardized ISMRMRD format. Therefore, within the GadgetStreamController, the readers read using the ISMRMRD format, and the writers write using the ISMRMRD format.
Of particular importance to the design of a reconstruction pipeline within the Gadgetron framework is the XML configuration file that is sent by the client application. This configuration file defines the desired reconstruction pipeline that should be loaded by the Gadgetron, including both data serialization (Writer) and deserialization (Reader) components as well as the specific gadgets to be loaded. This file also contains parameters specific to each reconstruction.

To summarize, we present an overview of the communication protocol as presented in the original paper by Hansen et al. [15]:

- A connection request from a client is received and a new instance of a GadgetStreamController is created.

- The client sends an XML configuration file defining the desired reconstruction pipeline to the Gadgetron. The GadgetStreamController loads the required data serialization and deserialization components and Gadgets from shared libraries as specified in the configuration file.

- The client sends an XML file with parameters for the reconstruction. This XML document is passed to each individual Gadget, which can interpret this document in the context of the function it is responsible for.

- The client starts streaming data to the GadgetStreamController, which uses its sets of Readers to deserialize the data and pass it on to the Gadgets.

- Images (or partially reconstructed datasets) are returned to the GadgetStreamController from the Gadgets. The GadgetStreamController uses Writers to serialize the reconstructed images and pass them back to the client.
Figure 7.3: Each gadget is an active object with at least one thread of execution. The thread is responsible for picking generic data packages off the input message queue and decoding the data packages into the data types that the gadget is expecting. If the data types match the gadget’s signature, the data will be passed on to a data processing function where the data can be modified and/or transformed into new data packages and enqueued on the next gadget’s input queue. Source: [15]
Internally, the general processing pipeline within a gadget itself can be described as in Figure 7.3. Each gadget itself contains a message queue. The role played by the gadget is then to receive messages from this queue that are relevant to the gadget. Relevant messages are then dequeued and processed within the gadget. Once processing is complete, the message is then placed on the queue for a downstream gadget.

The general design of the Gadgetron framework is such that common functionality provided by the framework is implemented in shared toolboxes, while streaming data processing is implemented in gadgets. We summarize a few key aspects of the functionality provided by the Gadgetron framework via shared toolboxes:

- **Multidimensional arrays.** The Gadgetron framework provides a templated multidimensional array (`NDArray`) that can be instantiated on both the CPU (`hoNDArray`) and the GPU (`cuNDArray`). Operators exist to transparently handle memory transfers between available computational resources. In addition to base multidimensional array implementations, the Gadgetron also includes extensive libraries providing functionality for operating on these arrays, including pointwise arithmetic operations as well as other functionality relevant to multidimensional arrays (e.g. permuting of dimensions, summing across a specific dimension, reshaping, etc.). These operators are implemented both for CPU and GPU.

- **Fast Fourier transforms.** Multiple toolbox components provide access to various forms of the Fast Fourier Transform (FFT). CPU-based FFT is made available using the FFTW3 library and GPU-based FFT is made available using the CUFFT library from NVIDIA. A GPU implementation of the nonuniform FFT
is also included. Functionality is provided for extending the raw FFT functions within the context of multidimensional arrays.

- **Iterative Solvers** The Gadgetron also includes toolboxes for implementing various iterative optimization algorithms, e.g. conjugate gradient, Split-Bregman, Augmented Lagrangian, etc.

- **Image Processing Operators** Operators for implementing various image transforms are also included, e.g. total variation, wavelet transform, etc.

- **Image Registration and Motion Correction** Toolboxes for implementing various image registration and motion correction techniques based on optical flow on both the CPU and GPU are also available within the Gadgetron.

A large (and still growing) number of gadgets are available within the Gadgetron for implementing specific reconstruction methods for specific acquired MRI data. Functionality exists within the current Gadgetron code for processing other non-Cartesian trajectories, including both radial and spiral trajectories. Functionality exists also for reading in echo-planar-imaging (EPI) data. A thorough description of all the functionality contained within the Gadgetron framework is well outside the scope of this document; we refer the reader to the code itself and excellent examples of the various available functionality within the code itself.

### 7.3 File Structure of the Gadgetron Code

All references to files in the Gadgetron code are relative to the parent directory for the Gadgetron source code. As of the revision referred to at the beginning of this
chapter, the root directory structure for the source code should be according to that shown in Figure 7.4.

All code contributions relevant to this work are located within the gadgets/MrRecon directory. The complete directory structure for this sub-directory is shown in Figure 7.5. For brevity’s sake, only directories shown. The vast majority of the gadgets written for this project exist in the directory gadgets/MrRecon. Other gadgets specific to the processing of segmented data are located in the directory gadgets/MrRecon/MrReconSegmented. Gadgets specific to processing of segmented $T_2^*$ data are located in gadgets/MrRecon/MrReconT2StarSegmented. CPU specific functions are located in the directory gadgets/MrRecon/toolboxes/cpu; GPU specific functions are located in the directory gadgets/MrRecon/toolboxes/gpu. Some of the code in this directory may still be incomplete and is not guaranteed to be functional. However, the specific code related to the methods described in this work should be functional.

7.4 Connecting the Gadgetron to the Client Application

We dedicate a small amount of space in terms of connecting the Gadgetron framework to the client application. In the context of the code described in this work, two client applications were considered: (1) Connection to the Siemens ICE/IDEA Environment, and (2) Connection to the Siemens MRI Scanner Environment. We first describe the two primary files for specifying the connection between the Gadgetron and the client application and then continue to describe various details relevant to connecting the Gadgetron to the two previously mentioned client applications. We refer to the computer containing the Gadgetron framework as the server and the
computer containing the client application as the client; the server and the client may exist on physically distinct or identical computers.

### 7.4.1 The gadgetron.xml File

The gadgetron.xml file exists on the server and primarily contains the port number that the Gadgetron executable (gadgetron.exe) will be listening on for connection requests and received messages (Line 6). The contents of this file are shown below.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<gadgetronConfiguration xsi:schemaLocation="http://gadgetron.sf.net/gadgetron/gadgetron.xsd"
    xmlns="http://gadgetron.sf.net/gadgetron"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <port>9002</port>
</gadgetronConfiguration>
```

Figure 7.4: Root level directory structure for the Gadgetron source code. Not all files and directories are shown.

1. Figure 7.4: Root level directory structure for the Gadgetron source code. Not all files and directories are shown.
2. The gadgetron.xml file exists on the server and primarily contains the port number that the Gadgetron executable (gadgetron.exe) will be listening on for connection requests and received messages (Line 6). The contents of this file are shown below.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<gadgetronConfiguration xsi:schemaLocation="http://gadgetron.sf.net/gadgetron/gadgetron.xsd"
    xmlns="http://gadgetron.sf.net/gadgetron"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <port>9002</port>
</gadgetronConfiguration>
```
Figure 7.5: Directory structure for code contributions in the MrRecon codebase described in this work. Only directory structure is shown.
This file exists in the source code as `apps/gadgetron/gadgetron.xml.example` and must be renamed to `gadgetron.xml`. While not specifically relevant to this work, the `cloudBus` entries pertain to the instantiation of a network of Gadgetron instances for distributed Gadgetron computation, either in a physical network or within a cloud computing environment [113]. Therefore, for setting up a distributing Gadgetron reconstruction environment, the entries under this section allow the specification of not only a port number, but also an IP address. Note also, that different Gadgetron instances can conceivably be run within the same logical computing environment listening for relevant connections/data by using different port numbers.

### 7.4.2 The gadgetron.ini File

The `gadgetron.ini` file exists on the client. (In this case, either the Siemens ICE/IDEA environment or the Siemens MRI Scanner Environment; however, similar information must in some way be provided for other client applications.) A generic example of the contents of this file is shown below.

```ini
[gadgetron]
hostname=10.192.32.88
port=9002

[tunnel]
open_tunnel=false
rsakeyfile=keyfile
sshusername=gtuser
sshhost=localhost
localhostname=192.168.2.1
```
The primary purpose of this file is to specify the location of the server on the TCP/IP network. Hence, the relevant entries in this file are contained in the [gadgetron] section: hostname, which can either be the domain name for the server or the IP address; and port, which specifies the port number of the Gadgetron instance on the server.

In order to provide a stable connection that is not affected by administrative restrictions placed on the network, it may be convenient to set up an SSH connection between the client and the server. The Gadgetron provides code to perform this automatically; settings for configuring this are found in the [tunnel] section. In this work, the SSH tunnel is set up manually, so we have not worked with the specific configuration settings in this section.

The Gadgetron now has the capability of loading specific binary compiled code corresponding to a Gadgetron instance within the Siemens vendor-specific reconstruction environment. This is a binary .img that must be compiled within the Linux environment and accessible locally by the client. The path entry chroot_image_path in the [chroot] section specifies the location of this file. To the best of our knowledge, this functionality is available only within the context of Siemens MRI scanners; however, we expect that this functionality can be readily extended to other vendors. The image file essentially corresponds to a virtual machine that is dynamically loaded immediately after data acquisition. The instance of the virtual machine is closed once...
reconstruction is completed and all data has been sent back to the client. This may be of interest if it becomes necessary to deploy a specific set of reconstruction code on multiple scanners at different sites.

7.4.3 Connection to the Siemens ICE/IDEA Environment

The Siemens ICE/IDEA Environment is a programming environment provided by Siemens that allows pulse sequence and reconstruction developers to implement and test pulse sequences and reconstruction code in an environment mirroring that of the physical scanner environment that exists on a separate standalone computer environment. Therefore the Siemens-specific Gadgetron interface code (hencefore termed, “IceGadgetron” code) can be compiled and deployed within the ICE/IDEA environment, allowing for the testing of reconstruction code within the Gadgetron framework provided a raw data file in the Siemens format. The specific implementation details for enabling communication between the ICE/IDEA environment within the Ohio State University Wexner Medical Center (OSUWMC) medical network are described in this section.

We assume that the ICE/IDEA environment exists on a computer (either physical or virtual) with IP address \( A \). We also assume that the Gadgetron framework exists on a computer (either physical or virtual) with IP address \( B \) and is listening on port \( p_B \). Ideally, within an open TCP/IP network without any administrative restrictions, the communication between the ICE/IDEA environment and the Gadgetron framework should work in a straightforward manner provided that the hostname and port entries are set correctly in the gadgetron.ini file. Given these settings, the
IceGadgetron code within the standalone client will attempt to set up a connection with the server located at the IP address specified by the hostname entry at the port number specified by the port entry. Once a connection is established, all communication relevant to performing the Gadgetron reconstruction will commence, and the connection will remain open until the reconstruction is complete.

In the gadgetron.ini file, the hostname entry is the specific IP address of the server. As of the time of the writing of this document, this setting allows communication between the client and the server when (1) the client and server exist on separate physical computers and when (2) the client and the server exist on the same physical computer (i.e. the ICE/IDEA environment is run in one console window, and the Gadgetron instance is run in a separate console window). In the second scenario, should the loopback interface be enabled, one may utilize localhost or the loopback address 127.0.0.1 in the hostname field; this configuration was used in the context of this work until network configurations were changed by medical center administration. In the case of the first scenario, the physically recognized IP address for the server must be used. It can be helpful to use the ping command to ensure that the server can be reached by the client.

Provided no administrative restrictions or other network configurations hinder communication between the client and the server the above should work without any issues. However, port restrictions imposed by medical center administration or network address translation (NAT) devices can hinder communication. In this case, it is helpful to utilize an SSH connection via port forwarding. In this case the PuTTY software can be used (Figure 7.6). In the session section of the configuration window, the Host Name and Port must be configured. Here, the port refers not
to the port that Gadgetron data will be received on; rather, it refers to the port for the SSH connection, which is not used here. The relevant portion of the SSH configuration pertains to *tunneling*, in which any data sent to a port on the *client* side is *tunneled* to the receiving port on the *server* side, allowing the bypassing of any network configurations that may hinder port-to-port communication. This can help aid the stability of the connection in the environment of a medical network, with ever changing network configurations and network security restrictions. Tunnel settings can be configured and stored in the *Tunnel* configuration panel located under *Connection → SSH → Tunnels* (Figure 7.7). In the case of the example configuration shown here, two tunnel configuration rules are set. The first rule is from port 9002 on the local loopback interface to port 9002 located on a *server* located at the IP address 10.192.32.141. The second rules explicitly specifies the local IP address but is otherwise identical to the first setting. It is helpful to have both tunnel rules enabled in case the loopback interface is not enabled. The tunnel settings within PuTTY are relative to the *client* side, i.e. data is sent from the *client* to the *server*.

In order to establish an SSH connection, an SSH server must be setup on the Gadgetron *server*. In the case of a Gadgetron installation within a Linux environment (e.g. an Ubuntu environment), the SSH server is typically already installed and needs no further configuration. However, within the Windows environment, at least at the time of this writing, it is typically necessary to set up an SSH server. In our work we have used the Bitvise SSH Server version 5.59; however, any SSH server software may be used. The primary settings on the server side are that tunnelling should be enabled and adequate directory access to the specific user account should be provided.
Figure 7.6: The configuration interface for the PuTTY software.
Figure 7.7: The Tunnel configuration panel for the PuTTY software
SSH connections are only relevant when the client and server exist on separate physical computers and communicate via a network with configurations that do not allow port to port communication. Typically, when the client and the server exist on the same physical computer, this is not necessary as communication between ports on the same IP address are typically not restricted. It may be possible that in rare scenarios where loopback communication traverses the medical network, communication is restricted. In this case, the use of tunnelling via SSH may provide a more stable connection.

7.4.4 Connection to the Siemens MRI Scanner Environment

As the standalone Siemens ICE/IDEA environment closely mirrors the physical Siemens Scanner environment, most configuration settings for the former also work for the latter client application. IceGadgetron is compiled and installed in a similar fashion; and the same gadgetron.ini file exists within this installation. Therefore, the specification for the IP address and port number for the Gadgetron server remains unchanged. Data communication is between the physical MRI scanner environment and the Gadgetron server, which may exist in disparate locations within the medical network. As such, it is highly recommended that tunnelling via an SSH connection is used to ensure communication stability. Within the Siemens ICE environment, the Linux-based ssh command is available. We refer the reader to the manpage for the ssh command in order to bind a local port to a host port. The command that we have used in this work for establishing the SSH connection with tunneling is shown below.

```
ssh -L 192.168.1.2:9002:10.192.32.88:9002 gadgetron@10.192.32.88
```
In this case, the Gadgetron server is located at the IP address 10.192.32.88 and is listening on port number 9002. An SSH server is set up with user account `gadgetron`. For a Gadgetron server located at a different IP address or a different SSH user account, the command should be adjusted accordingly.

The client should be the computer system on which the pulse sequence is first executed (i.e. the host computer). In some scenarios, this may be different from the computer on which the reconstruction is performed. Therefore, it is important to specify the correct IP address here.

### 7.5 Accessing GPU Functionality on the Gadgetron Server Remotely

We add a brief remark here to highlight a particular known issue pertaining to GPU access when remotely starting a Gadgetron instance. Ideally, a Gadgetron instance should be started prior to conducting data acquisition/reconstruction by physically opening a command prompt window on the physical computer that is to serve as the Gadgetron server. In this scenario, all functionality (i.e. both CPU and GPU functionality) should be available to the executable. However, under various circumstances (e.g. a crash forcing termination of the Gadgetron instance, or running of the Gadgetron instance in an environment not allowing persistent existence of the instance), it may be necessary to start a Gadgetron instance via a remote connection. We emphasize that it is not possible to access GPU functionality via the establishing of a remote connection using the remote desktop protocol (RDP) – i.e. through the use of the Windows remote desktop functionality. A Gadgetron instance can be started via this approach; however, GPU functionality will not be available, and the Gadgetron will crash when trying to access GPU functionality. However, code
requiring only CPU functionality will run without issue. If GPU functionality is
desired, it is necessary to start the Gadgetron instance via a remote connection that
uses a VNC connection (i.e. Teamviewer or an equivalent software). This issue has
been known in the GPU community for some time and pertains to the fact that a
connection via RDP swaps out the underlying video driver, impacting access to GPU
computational capacity. It is possible that with the professional line of video cards,
this issue has been addressed. However, at the time of the writing of this work, with
the current consumer line of video cards used, a VNC connection is necessary.

7.6 Parameter Map Processing - Vendor to ISMRMRD Parameter Conversion

As part of initializing a client-server communication channel between a client ap-
plication and the Gadgetron server, an XML file consisting of acquisition parameters
is sent to the server. This parameter file is generated via the vendor-specific inter-
face code (in the case of this work, the IceGadgetron code) and formatted at a low
level into a vendor specific XML that is then sent to the server.

The contents of this file include parameters relevant to the MRI acquisition itself
in addition to other patient specific data entered via the MRI host console. Such
parameters can be important and relevant to the reconstruction side and may be used
to indicate the type of reconstruction desired, e.g. various parameters can indicate the
type of k-space trajectory (Cartesian versus non-Cartesian), the sampling pattern, as
well as basic scan information such as slice thickness and field-of-view. Essentially,
every parameter related to the acquisition itself can be included in this XML file
through modification of the vendor-specific code.
The initial raw XML file containing the acquisition parameters varies from vendor to vendor; therefore, initial processing must be performed to convert this parameter set to a set of parameters as specified by the ISMRMRD format. This is performed as a two-step process. In the first step, the input XML file is mapped via a direct parameter map as specified by a separate XML file. Typically, this is a one-to-one mapping of the input parameter set into a sub-tree within the Gadgetron parameter space, e.g. for a Siemens parameter set, all parameters would be mapped into a Siemens sub-tree. In the second step, an XML parser is used to parse this vendor specific parameter tree to generate a new parameter tree that is compatible with the ISMRMRD format. As most of this code involves vendor-specific acquisition parameters, it is not possible here to provide examples. However, examples for the case of parameter map processing within the Siemens environment can be found within the IceGadgetron code that is made available via Siemens distribution channels.

In the context of image reconstruction within the Gadgetron framework, various reconstruction algorithms may require user input on the acquisition side. For example, it may be necessary to specify that a VISTA [13] sampling trajectory is used when performing a cardiac cine acquisition in order to inform the Gadgetron framework that a specific type of reconstruction is needed. This may require a modification on the sequence side to include a checkbox that the user may select if a VISTA trajectory is used. In this case, the question then becomes an issue of how the user input is mapped to an input that the Gadgetron can recognize and act upon. The user flag may be mapped to a vendor specific parameter that is then mapped by the initial Gadgetron parameter map. The Gadgetron XML parser may then further parse this input to set a variable that can be read by the relevant gadgets used for reconstruction.
In another example, certain acquisitions may require knowledge of certain acquisition parameters that are not typically mapped via a “typical” parameter map. Knowledge of the location of the vendor-specific parameters can indicate the specific modifications that must be made to the parameter map in order to provide access to these acquisition parameters within the Gadgetron framework.

Often it is the case that, with a new version of the Gadgetron framework, even though the released code may compile without issue, the executable may crash at runtime. In performing error tracking, one should keep in mind that any mismatches in the parameter mapping and parsing may result in extremely cryptic output, or no output at all. In some cases, it may be possible that reconstruction commences and completes without any fatal errors; however, incorrect mapping may result in spurious errors in reconstructed results that may be difficult to trace. Even though these files are not a central part of the code framework from the perspective of implementing reconstruction methods, errors in these files can result in cryptic and often difficult-to-identify error output downstream. Therefore, it is recommended that the attentive Gadgetron developer always keep in mind changes that may need to be made to these files.

7.7 Installation of the MrRecon Codebase

In this section, we provide details pertaining to the installation of the code contributions discussed in this work, referred to as the MrRecon codebase.

It is assumed that the reader has a working installation of the Gadgetron code that matches the version indicated at the beginning of the chapter. We reiterate that subsequent versions of the Gadgetron code may require that modifications be
made to the MrRecon code in order for it to compile and run. The Gadgetron framework currently uses the CMAKE build system, allowing for cross-compatible build environments, that for our primary purposes, include both the Windows and Linux environments. It is most straightforward to install the Gadgetron framework and all the necessary libraries within the Linux environment. As the OSUWMC primarily prefers the Windows environment, the description here focuses on this environment.

Installation of the Gadgetron framework within the Windows environment can be quite involved, as it requires the compiling of several library dependencies that can themselves be quite involved. We refer the interested reader to the Gadgetron website for specific Gadgetron installation instructions in both environments.

Installation of the MrRecon codebase involves copying the MrRecon directory into the gadgets directory of the Gadgetron codebase. Modification of the CMakeLists.txt file within the gadgets directory is necessary in order for the CMAKE system to include the MrRecon code in the generated MSVC solution. This can be done by ensuring that the code specified below is present somewhere in the CMakeLists.txt file.

```
add_subdirectory(MrRecon)
```

Currently, the MrRecon codebase requires the CULA Dense library in order to compile. This library is in addition to the recommended libraries for installation of the Gadgetron framework. In addition, the MrRecon codebase utilizes the following libraries: CUDA, MKL, Armadillo. In the initial Gadgetron installation, these libraries are optional; however, for the MrRecon codebase, they are required.
When including the MrRecon code in the CMAKE build system, a number of additional projects are created within the resulting MSVC solution, each including the prefix MrRecon_ to indicate that it is part of the MrRecon codebase:

- MrRecon
- MrRecon_gpucore
- MrRecon_cpucore
- MrReconSegmented
- MrReconT2StarSegmented
- MrRecon_mnwavelet
- MrRecon_timing

Of these seven additional projects, the majority of the reconstruction gadgets are contained within the MrRecon project. Core toolboxes for the GPU and CPU are included respectively in the projects MrRecon_gpucore and MrRecon_cpucore. Additional gadgets for segmented data and segmented $T_2^*$ data are in the projects MrReconSegmented and MrReconT2StarSegmented. A gadget for performing timing measurements for an entire reconstruction process is included in the project MrRecon_timing. The project MrRecon_mnwavelet is incomplete and is intended to be a general toolbox for implementing $M$-level $N$-dimensional wavelet transforms.

Code specifically related to the implementation of bFISTA within the Gadgetron framework is contained primarily in the projects MrRecon, MrRecon_gpucore, and MrRecon_cpucore. Some code exists within these projects pertaining to patch-based
regularization techniques; however, this code cannot be considered complete. Development related to these techniques began initially within the Gadgetron framework and was subsequently moved to the MATLAB programming environment in order to aid code prototyping and iterative revision of the techniques. The remaining MrRecon projects fall outside the specific scope of the methods discussed in this work; however, we will dedicate a small amount of space in a subsequent section to describing the code contained in these projects.

7.8 Interprocess Communication within the Gadgetron Framework

In this section, we cover some of the details related to communication between processes within the Gadgetron framework. The primary goal of this section is to provide a basic understanding of communication between gadgets both in terms of image (or signal) data as well as configuration parameters.

All interprocess communication within the Gadgetron framework is implemented via the ADAPTIVE Communication Environment (ACE, [149]). A schematic illustrating the role played by ACE in client/server communication is shown in Figure 7.8. The ACE libraries enable network communication between the client application and the Gadgetron server as well as communication both between multiple Gadgetron instances (either on the same or different physical computer) and also within a Gadgetron instance. ACE is used to establish a main Gadgetron process that is responsible for listening for an incoming TCP/IP connection on the specific Gadgetron port specified in the gadgetron.xml file. ACE is also used for setting up the streaming architecture for the Gadgetron. This is accomplished by parsing an XML file that is loaded upon receiving a connection request from a client application. This XML
Figure 7.8: Schematic illustrating communication between the client application and multiple Gadgetron instances. ACE is responsible for handling all intra-communication among Gadgetron instances and between Gadgetron instances and the client application.

Figure 7.9: Schematic illustrating interprocessing communication between gadgets within a Gadgetron instance. (1) An ISMRMRD parameter set is first sent to all gadgets after gadget streaming pipeline is initialized. (2) Each message received from the client application is sent to the first gadget on the input side of the streaming pipeline. The ISMRMRD parameter set is processed by the function \texttt{config\_process} within each gadget. The messages are processed by the function \texttt{process} within each gadget.
Figure 7.10: Schematic illustrating the general data process in a single gadget. (1) The ACE function `getq` is used to retrieve a message from the gadget’s queue. (2) Relevant data processing is performed using functionality provided by GPU- or CPU-based toolboxes if necessary. (3) Message is enqueued in the child gadget’s queue using the ACE function `putq`.

The `file` specifies the chain of gadgets that should be loaded by the Gadgetron process. The order of the gadgets determines how the pipeline architecture should be set up, with data being passed down the pipeline from gadget to gadget. A summary of this process is shown in Figure 7.9. Currently, the Gadgetron framework only supports a serial chain of gadgets; however, the ACE library provides functionality for setting up more complex streaming architectures, including branching within the streaming architecture.

The use of ACE functionality directly within gadget-specific code can be unwieldy; therefore, functionality is provided within the Gadgetron framework to abstract away from the ACE interface itself. Gadgets themselves are classes derived from the `ACE_Task` class, which provides basic multi-threaded task functionality. `ACE_Task` provides multi-threaded execution functionality as well as basic message communication between tasks themselves. In the context of developing gadgets within the Gadgetron,
some primary relevant functionality is summarized in the following. The `msg_queue` is a queue of messages associated with the specific task. Messages can be retrieved from this queue using the `getq` functionality. Messages can also be placed on the queue by a parent task via the `putq` functionality. Message dequeuing (i.e. retrieval) and enqueuing (i.e. transmission) form the basic initial and final steps in most typical gadgets written within the Gadgetron. In other words, the process flow within a gadget often involves the following steps: (1) dequeue a message from the current task’s queue, (2) process the message, (3) enqueue the message in the sibling’s queue. Currently, the Gadgetron supports only single parents and single children; siblings have not yet been implemented. This process is summarized in Figure 7.10.

7.8.1 The `process_config` and `process` Functions

Within the `gadget` class itself, the parent `ACE_Task` expects two primary functions: (1) the `process_config` function and (2) the `process` function.

The `process_config` function is called within each gadget before any streaming data is passed through the streaming architecture, i.e. at the point of initializing the gadget chain. The primary role of the `process_config` function is to perform any initialization necessary for the gadget. The `process_config` received an ACE message block `mb` which is a block of memory that contains a header following the ISMRMRD format. Deserialization of this header enables parsing via an XML parser, allowing access to relevant acquisition parameters. Therefore, the `process_config` function is typically used to perform any initialization within the gadget relevant to the acquisition parameters. For example, in the context of cine imaging, the `process_config` function can be used to determine (1) the dimensions of the data
that is eventually to be received, (2) whether it is a real-time or segmented acquisition, (3) what kind of sampling pattern was used to acquired the data, as well as many other properties.

The kind of parameters that can be parsed in this function depend on the specific parameter map and parser that is used. Often errors related to attempting to parse non-existent parameters can occur here, resulting in the Gadgetron framework crashing before any data itself is processed. This can be a helpful indication that issues may exist with the parameter map and parser.

The \texttt{process} function is called within each gadget each time a message is received on the gadget’s message queue. The primary role of the \texttt{process} function is to perform any relevant data processing on the message received. The function takes as its input an ACE message block $\texttt{mb}$. The message block is a generic chunk of memory that in itself contains no type information. Therefore, before any data processing, the message block must be parsed in order to determine the memory contents. Parsing functionality is achieved via type-casting. The Gadgetron implements a container class \texttt{GadgetronContainerMessage} which is a generic container that can be used to hold any data structure defined within the Gadgetron framework. Gadgetron messages can also be chained together; enqueueing one message on a child’s message queue also enqueues all child messages.

The generic \texttt{process} function takes only one input, a single ACE message block. Therefore, at the beginning of each \texttt{process} function, parsing must be implemented to suitably identify messages within this message block. This can quickly become a tedious task, so functionality has been implemented within the Gadgetron for implementing \texttt{process} functions that take as input multiple \texttt{GadgetContainerMessage}
variables. In other words, these class definitions already include code that parses
the input message block into multiple Gadgetron messages. The Gadgetron provides
functionality for gadgets that take in one, two, or three messages; it is straightforward
to provide functionality further extending this functionality. Further details on
gadget implementation as well as class definitions can be found in the header file
apps/gadgetron/Gadget.h.

7.9 XML File Format for the Gadgetron Stream Configuration

Before describing the reconstruction methods implemented in the MrRecon code-
base, we first describe the XML file format for configuration of the Gadgetron stream
for data processing. The specific XML file to be loaded must be specified for each
connection made to a Gadgetron server, and this is typically done via the XML
file sent during the initial connection request. The gadget stream is loaded accord-
ing to the tree structure within this XML file. Currently, the instantiated gadget
stream is static for the duration of the Gadgetron connection; dynamic reordering of
gadgets within the stream has not yet been implemented. The XML files for con-
figuration of the Gadgetron gadget stream consists of an XML tree with root node
gadgetronStreamConfiguration. Under this node, one or more of each of the fol-
lowing three nodes must be present: (1) the reader node, (2) the writer node, and
(3) the gadget node. A summary of the gadgetronStreamConfiguration XML tree
is shown in Figure 7.11.

The reader node is responsible for handling incoming data from the client appli-
cation. The writer node is responsible for writing data out to the client application.
Note that neither nodes are responsible for intra-gadget communication; this is set
Figure 7.11: XML tree structure for a gadgetronStreamConfiguration.
up initially by the parent Gadgetron process and is handled internally by each gadget itself. The **writer** and **reader** nodes are responsible solely for client/server communication. Multiple **writer** or **reader** nodes may be present, but at least one of each node must be present. Within either type of node, three fields must be present. The **slot** is a numerically unique identifier for the particular reader or writer. The **dll** specifies which DLL should be loaded by the Gadgetron process to access functionality, and the **classname** specifies the specific C++ class that implements this functionality.

In general, generic **reader** and **writer** nodes already exist within the Gadgetron framework. Therefore, it will in general not be necessary to write additional readers or writers. However, the flexibility exists, should there be a need to process different data.

The **gadget** node specifies which gadget should be loaded by the Gadgetron parent process. Multiple **gadget** entries may exist in a single XML file; the order of the gadgets in the streaming architecture is determined by the order of the **gadget** entries in the XML file. Within each **gadget** node, three fields are required: (1) the **name** field is a unique text identifier for the gadget, (2) the **dll** field specifies the DLL to be loaded by the Gadgetron process, and (3) the **classname** field indicates which C++ class should be instantiated. An optional fourth field, the **property** field, can also be present and is used to pass parameters to the gadget. Within the **property** field, two entries should be present: (1) **name** is the name of the property, and (2) **value** is the value of the property. In order for the properties to be correctly accessible by the gadget, the **GADGET_PROPERTY** macro must be correctly declared within the gadget header file.
Therefore, the primary components from a development perspective for implementing a reconstruction stream architecture involves the `gadgetronStreamConfiguration` XML file and the specific class definitions for each gadget within the reconstruction stream architecture. The format of these class definitions is up to the developer; however, in the `MrRecon` codebase, we follow the general convention that a single gadget consists of a `.cpp` file containing the class implementation and a corresponding `.h` file containing the class definition.

### 7.10 Reconstruction within the MrRecon Codebase

In this section, we begin to describe in detail the reconstruction code contained within the `MrRecon` codebase. We focus primarily on three types of reconstruction within this section: (1) reconstruction using TGRAPPA [74], (2) reconstruction using SPIRiT [85], and (3) reconstruction using iterative SENSE based on [75]. While TGRAPPA reconstruction is not specifically discussed in other portions of this work, it serves as a useful foundation for describing the implementation of various gadgets used elsewhere in this codebase.

#### 7.10.1 Reconstruction using TGRAPPA

In this section, we describe two approaches for performing a TGRAPPA reconstruction. The first is based on the HTGRAPPA method [150], which performs the TGRAPPA reconstruction in the image domain and can be computationally efficient but cannot be used for nonuniform Cartesian trajectories (e.g. the VISTA sampling [13]). The second implementation performs the TGRAPPA reconstruction in the k-space domain, applying the relevant kernel to interpolate a specific block of k-space. This approach is general and can be used for variable density sampling patterns. The
k-space implementation can be utilized as an initialization in other reconstruction approaches such as the SPIRiT and SENSE approaches to be discussed later.

These implementations draw heavily from code within the Gadgetron framework itself for implementing a GRAPPA (not TGRAPPA) reconstruction aimed at interactive real-time imaging in which the orientation of the slice plane can be adjusted dynamically with the GRAPPA reconstruction code also adjusting the kernel estimation dynamically.

Because the focus of this work is primarily real-time cine imaging, the implementations here are not GRAPPA and do not utilize any reference scan (i.e. separate ACS data or a fully-sampled ACS region acquired in each frame). Rather, the TGRAPPA approach is used, in which a full-sampled ACS dataset is generated from the temporal average of all acquired k-space frames. Modification of the code to support a separate ACS scan or a fully-sampled ACS region should be relatively straightforward.

The relevant XML files specifying the gadget chain for TGRAPPA reconstruction are the files tpatGrappa.xml and kGrappa.xml located in the directory gadgets/MrRecon/config. For clarity, we term the image-based TGRAPPA implementation as HTGRAPPA and the k-space-based GRAPPA implementation as KGRAPPA. The first XML file is used to implement HTGRAPPA, and the second file is used to implement KGRAPPA. The XML file for the HTGRAPPA gadget chain is shown in full detail below.

```xml
<?xml version="1.0" encoding="UTF-8"?>
  <reader>
    <!-- HTGRAPPA gadget chain details here -->
  </reader>
</gadgetronStreamConfiguration>
```
<slot>1008</slot>
<dll>gadgetron_mricore</dll>
<classname>GadgetIsmrmrdAcquisitionMessageReader</classname>
</reader>

<writer>
<slot>1004</slot>
<dll>gadgetron_mricore</dll>
<classname>MRIImageWriterCPLX</classname>
</writer>

<writer>
<slot>1005</slot>
<dll>gadgetron_mricore</dll>
<classname>MRIImageWriterFLOAT</classname>
</writer>

<writer>
<slot>1006</slot>
<dll>gadgetron_mricore</dll>
<classname>MRIImageWriterUSHORT</classname>
</writer>

<gadget>
<name>MRINoiseAdjust</name>
<dll>gadgetron_mricore</dll>
<classname>NoiseAdjustGadget</classname>
</gadget>

<gadget>
<name>PCA</name>
<dll>gadgetron_mricore</dll>
<classname>PCACoilGadget</classname>
</gadget>

<gadget>
<name>CoilReduction</name>
<dll>gadgetron_mricore</dll>
<classname>CoilReductionGadget</classname>
<property><name>coils_out</name><value>16</value></property>
</gadget>

<gadget>
<name>RemoveROPartialFourier</name>
<dll>mrrecon</dll>
<classname>RemoveROPartialFourierGadget</classname>
</gadget>

<gadget>
<name>RemoveROOversampling</name>
<dll>gadgetron_mricore</dll>
<classname>RemoveROOversamplingGadget</classname>
</gadget>
</gadget>

<gadget>
<name>PerSliceAccumulator</name>
<dll>mrrecon</dll>
<classname>MrReconPerSliceRTAccumulatorGadget</classname>
<property>
  <name>maxReconedFrames</name>
  <value>-1</value>
</property>
</gadget>

<gadget>
<name>CropImageSeries</name>
<dll>mrrecon</dll>
<classname>CropImageSeriesGadget</classname>
<property>
  <name>roCropBottom</name>
  <value>1</value>
</property>
<property>
  <name>roCropTop</name>
  <value>1</value>
</property>
</gadget>

<gadget>
<name>CoilCompressionImageSeries</name>
<dll>mrrecon</dll>
<classname>CoilCompressionImageSeriesGadget</classname>
<property>
  <name>Coils</name>
  <value>12</value>
</property>
</gadget>

<gadget>
<name>GRAPPA Recon</name>
<dll>mrrecon</dll>
<classname>RtHTGrappaReconGadget</classname>
<property>
  <name>kernelSizeCOL</name>
  <value>5</value>
</property>
<property>
  <name>kernelSizeLIN</name>
  <value>4</value>
</property>
<property>
  <name>sendOnAcqData</name>
  <value>true</value>
</property>
<property>
  <name>tikhonovReg</name>
  <value>1e-4</value>
</property>
</gadget>

<gadget>
<name>Padding Gadget</name>
<dll>mrrecon</dll>
<classname>PadToReconSizeGadget</classname>
<property>
  <name>padFE</name>
  <value>false</value>
</property>
<property>
  <name>padPE</name>
  <value>true</value>
</property>
</gadget>

<gadget>
<name>SendOutByPhase</name>
<dll>mrrecon</dll>
<classname>SendOutByPhaseGadget</classname>
<property>
  <name>truncateFirstFrames</name>
  <value>0</value>
</property>
</gadget>
The XML file for the KGRAPPA gadget chain is shown in full detail below.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<gadgetronStreamConfiguration xsi:schemaLocation="http://gadgetron.sf.net/gadgetron␣gadgetron.xsd"
</gadgetronStreamConfiguration>
```
<reader>
  <slot>1008</slot>
  <dll>gadgetron_mricore</dll>
  <classname>GadgetIsmmrdAcquisitionMessageReader</classname>
</reader>

<writer>
  <slot>1004</slot>
  <dll>gadgetron_mricore</dll>
  <classname>MRIImageWriterCPLX</classname>
</writer>

<writer>
  <slot>1005</slot>
  <dll>gadgetron_mricore</dll>
  <classname>MRIImageWriterFLOAT</classname>
</writer>

<writer>
  <slot>1006</slot>
  <dll>gadgetron_mricore</dll>
  <classname>MRIImageWriterUSHORT</classname>
</writer>

<gadget>
  <name>MRINoiseAdjust</name>
  <dll>gadgetron_mricore</dll>
  <classname>NoiseAdjustGadget</classname>
</gadget>

<gadget>
  <name>PCA</name>
  <dll>gadgetron_mricore</dll>
  <classname>PCACoilGadget</classname>
</gadget>

<gadget>
  <name>CoilReduction</name>
  <dll>gadgetron_mricore</dll>
  <classname>CoilReductionGadget</classname>
  <property>
    <name>coils_out</name>
    <value>16</value>
  </property>
</gadget>

<gadget>
  <name>RemoveROPartialFourier</name>
  <dll>mrrecon</dll>
  <classname>RemoveROPartialFourierGadget</classname>
</gadget>

<gadget>
<name>RemoveROOversampling</name>
<dll>gadgetron_mricore</dll>
<classname>RemoveROOversamplingGadget</classname>
</gadget>

<name>PerSliceAccumulator</name>
<dll>mrrecon</dll>
<classname>MrReconPerSliceAccumulatorGadget</classname>

<property>
  <name>maxReconedFrames</name>
  <value>48</value>
</property>

<name>GRAPPA Recon</name>
<dll>mrrecon</dll>
<classname>RtKGrappaReconGadget</classname>

<property>
  <name>kernelSizeCOL</name>
  <value>5</value>
</property>

<property>
  <name>kernelSizeLIN</name>
  <value>4</value>
</property>

<property>
  <name>sendOnAcqData</name>
  <value>true</value>
</property>

<property>
  <name>tikhonovReg</name>
  <value>1e-3</value>
</property>

<name>SendOutByPhase</name>
<dll>mrrecon</dll>
<classname>SendOutByPhaseGadget</classname>

<property>
  <name>truncateFirstFrames</name>
  <value>0</value>
</property>

<property>
  <name>truncateLastFrames</name>
  <value>0</value>
</property>

<name>SOSCombine</name>
<dll>mrrecon</dll>
<classname>SOSCombineGadget</classname>

<!--
<name>FFT</name>
<dll>gadgetroncore</dll>
<classname>FFTGadget</classname>
-->
7.10.1.1 Description of the HTGRAPPA XML File

In this section we provide a detailed description of the HTGRAPPA XML file. As the readers and writers are fairly standardized within the Gadgetron framework, we do not allocate space for describing their functionality. A schematic block diagram for this XML file is shown in Figure 7.12. Of particular note in this diagram, gadgets from the \texttt{gadgetron\_mricore} DLL file are provided by the Gadgetron framework; gadgets from the \texttt{mrrecon} DLL file are provide by the \texttt{MrRecon} codebase. Also indicated in the stream diagram are various forms of data passed between gadgets.

\texttt{MRINoiseAdjustGadget} performs a noise adjustment for each k-space line received. For the particular scanner environment (i.e. the Siemens environment), each message sent to the \texttt{server} consists of a single acquisition line of k-space. If multiple channels are used, then this line consists of multiple channels. Therefore, for each
Figure 7.12: Schematic block diagram for *tpatGrappa.xml*. Gadgets from both *gadgetron_mricore* and *mrrecon* DLLs are used. Portions of gadget stream are indicated where data is serialized by (1) k-space line; (2) complete dataset, i.e. across spatial, channel, and frame dimensions for real-time cine; and (3) frames, i.e. across spatial and channel dimensions. Data after sum-of-squares combination (SOSCoilCombine) is serialized by spatial dimensions only. Parameters for the primary reconstruction gadget are summarized.
k-space line received, a noise adjustment is performed. This is similar to the noise-
prewhitenning described in [75]. This gadget only functions if a noise scan is detected. If so, then the noise covariance between the channels is calculated and this is used to scale all subsequent data lines received such that the noise level is constant across all channels.

Following this gadget are two commented out gadgets, PCACoilGadget, and Coil-
ReductionGadget. The purpose of these two gadgets is to perform coil reduction in the principal component domain. In other reconstruction code within MrRecon, we have implemented our own coil compression that takes into account all k-space data received. The implementation here is performed on a per k-space line basis.

If the k-space data is acquired using partial Fourier (in the frequency encoding direction since at this point we are only dealing with a single line of k-space), then the next gadget, RemoveROPartialFourierGadget correctly determines the center k-space sample location and suitably zero-pads the k-space line such that this partial Fourier format is accounted for. Note that this definition of partial Fourier is in the frequency encoding direction; this is more commonly referred to as an asymmetric echo in the MRI community, with partial Fourier referring to asymmetric sampling in the phase encoding direction.

The next gadget, RemoveROOversamplingGadget, removes the two-fold oversampling in the frequency encoding direction, effectively reducing the field-of-view (FOV) by a factor of two. This is accomplished by a Fourier transform along the frequency encoding direction into the image domain, cropping by a factor of 2, and returning back to the k-space domain. Note that the FOV field in the header information should be adequately updated at some point in the gadget chain in order to account for this
twofold FOV cropping; otherwise, the images may not display correctly or at all on the user console.

The next gadget, MrReconPerSliceRTAccumulatorGadget, is responsible for binning the individual k-space lines initially received by the Gadgetron server into a multidimensional dataset corresponding to an MR dataset. In other words, this gadget seeks to accumulate k-space lines until a complete dataset is detected. Once a complete dataset is detected (in this case, a single slice, which may consist of two or three spatial dimensions across multiple time points and multiple channels), that dataset is sent on for further downstream processing. The gadget will continue to receive individual k-space lines until another complete dataset is received or the end of the acquisition is detected. The functionality within the accumulator is relatively involved, and we reserve a specific section later on for a more thorough description of this and related accumulator gadgets.

CropImageSeriesGadget can be used for performing cropping of the (now multidimensional) dataset in the readout, or frequency encoding, direction. This can be useful for reducing computation time by only reconstructing relevant regions of the FOV. The gadget takes in two parameters:

- **roCropBottom**: This parameter specifies the ratio of the bottom half (i.e. the region indexed from \([0, \cdots, \frac{n_{FE}}{2} - 1]\)) of the image that should be kept,

- **roCropTop**: This parameter specifies the ratio of the top half (i.e. the region indexed from \([\frac{n_{FE}}{2}, \cdots, n_{FE}]\)) of the image that should be kept.
Here, \( n_{FE} \) is the dimension size in the frequency encoding direction. For example, setting \( \text{roCropBottom} = 0.5 \) and \( \text{roCropTop} = 0.5 \) will discard the top and bottom quarters of the image in the frequency encoding direction.

\textbf{CoilCompressionImageSeriesGadget} is used to perform coil compression on the image series. The primary difference between this gadget and the coil compression gadget described earlier is that the principle components here take into account the entire image series, in contrast to a single k-space line. This can help improve coil compression. This gadget takes in one parameter, \textit{Coils}, which is used to specify the number of output coils to generate.

\textbf{RtHTGrappaReconGadget} implements the HTGRAPPA reconstruction. It takes in two parameters for the kernel size. The parameter \textit{kernelSizeCOL} is used to indicate the size of the kernel in the frequency encoding direction. The parameter \textit{kernelSizeLIN} is used to indicate the size of the kernel in the phase encoding direction. The parameter \textit{sendOnAcqData} is a Boolean flag that determines whether or not the original acquired k-space data should be sent on to the downstream gadgets. The parameter \textit{tikhonovReg} is used to specify the amount of Tikhonov regularization for estimating the kernel. Image reconstruction utilizes both the GPU and CPU, with kernel estimation being on the GPU and GRAPPA unwrapping being performed on the CPU. The following general steps are implemented:

- First, the average all k-space is obtained.

- Second, the GRAPPA kernels are estimated by (1) forming the system matrices, and (2) performing matrix inversion. Matrix inversion is implemented using CUBLAS routines. The implementation closely follows that of the interactive GRAPPA implementation provided by the Gadgetron framework.
• Third, HTGRAPPA kernels are obtained via GRAPPA unmixing. Application of the HTGRAPPA kernels implicitly combines the coils, so the coil sensitivity maps are used to again generate separate channels in case this is needed for further downstream processing. The application of the sensitivity maps can be omitted if no further downstream processing is needed.

• Fourth, pointwise multiplication and sum operations are performed to perform the HTGRAPPA reconstruction.

GPU estimation of the kernels is implemented on the GPU in the file located at gadgets/MrRecon/toolboxes/gpu/tpatHTgrappa.cu and the corresponding header file in the same directory location. Application of the kernel (or interpolation weights) is implemented in the file located at gadgets/MrRecon/GrappaWeights.cpp and the corresponding header file in the same directory location. This portion of code is itself a modification of the original GRAPPA code contained in the Gadgetron framework for interactive real-time imaging. The corresponding files for the original GRAPPA code are at the following locations:

• gadgets/grappa/GrappaGadget.{cpp, h}

• toolboxes/mri/pmri/gpu/htgrappa.{cu, h}

PadToReconSizeGadget implements zero-padding such that the output matches the recon size as specified by the header information. In some cases, it may be possible that the matrix size of the data does not match the recon size as specified by the header information. The consequence of this is that the reconstruction may successfully be completed, yet the images may not display correctly on the user console due to a mismatch in the header information. Using this gadget to perform a suitable
zero-padding addresses this issue. Note that this is not an issue when performing an offline Gadgetron reconstruction outside of the scanner environment.

SendOutByPhaseGadget deserializes the reconstructed image data, and sends out individual messages corresponding to the specific reconstructed cardiac phases. The parameter truncateFirstFrames can be used to truncate the image series at the beginning; the parameter truncateLastFrames can be used to truncate the image series at the end. This can be useful when it becomes necessary to compare the image series with other acquisitions, e.g. comparing to segmented cine images.

SOSCombineGadget performs a sum-of-squares coil combination. Here we use the actual sum-of-squares combination. There exists another coil combination approach in the Gadgetron framework that takes into account phase variation.

The remaining gadgets, ExtractGadget, AutoScaleGadget, FloatToUShortGadget, and ImageFinishGadget, perform post reconstruction data processing such the data is converted to a format expected by the MRI scanner environment. These gadgets may not be necessary in later Gadgetron versions.

7.10.1.2 Description of the KGRAPPA XML File

The KGRAPPA XML file is identical to the HTGRAPPA XML file with the two following distinctions: (1) the coil compression functionality provided by the Gadgetron via PCACoilGadget and CoilReductionGadget is used, and (2) the Rt-KGrappaReconGadget is used for KGRAPPA reconstruction. The schematic block diagram for this stream configuration file is shown in Figure 7.13. Note that many of the gadgets used in the HTGRAPPA configuration could also be used here due to the modular implementation. For example, it is possible to utilize the Gadgetron implementation of coil compression here instead of that implemented within MrRecon. The
primary difference between the \texttt{RtKGrappaReconGadget} and the \texttt{RtHTGrappaReconGadget} is that the KGRAPPA gadget expects two variables to be set by the preceding accumulator gadget that indicate (1) the minimum gap size between acquired lines within phases, and (2) the maximum gap size between acquired lines within phases. A detailed description of this follows in the section related to accumulator gadgets. This allows the KGRAPPA gadget to perform multiple kernel estimations corresponding to multiple kernel sizes. Image reconstruction is then performed in the k-space domain on a block-by-block basis, utilizing the corresponding kernel for that particular gap size. This results in a slower reconstruction time; however, this has the advantage of being flexible for both uniformly downsampled and nonuniformly downsampled Cartesian trajectories. This gadget will be used as an initialization for the SPIRiT reconstruction that is described in the following section.

GRAPPA kernel estimation for KGRAPPA is again performed on the GPU, and k-space estimation is performed of the CPU. It may be possible to obtain further speed improvements by implementing the k-space estimation on the GPU; however, in our work, we have observed that the time for kernel estimation dominates the time for k-space estimation, especially when multiple kernels need to be estimated. The GPU code for the kernel estimation can be found in the following files:

- \texttt{gadgets/MrRecon/toolboxes/gpu/kgrappa.\{cu, h\}}

Again, the kernel estimation code closely follows that of the original GRAPPA implementation for interactive real-time imaging, with the notable exception that no further processing is done once the interpolation weights are obtained.

In both TGRAPPA implementations, we have provided functionality for implementing Tikhonov regularization when performing kernel estimation. This did not
Figure 7.13: Schematic block diagram for kGrappa.xml. Gadgets from both gadgrtron_mricore and mrrecon DLLs are used. Portions of gadget stream are indicated where data is serialized by (1) k-space line; (2) complete dataset, i.e. across spatial, channel, and frame dimensions for real-time cine; and (3) frames, i.e. across spatial and channel dimensions. Data after sum-of-squares combination (SOSCoilCombine) is serialized by spatial dimensions only. Parameters for the primary reconstruction gadget are summarized.
exist in the original GRAPPA implementation provided by the Gadgetron framework.

The code for the Tikhonov regularization is found in the file `gadgets/MrRecon/toolboxes/gpu/gpuutils.cu` and is shown below.

```c
__global__ void tikhonovKernel(complext<float>* AHA, float thresh, unsigned int N) {
    unsigned long idx_in = blockIdx.x*blockDim.x+threadIdx.x;
    if (idx_in == 1) {
        float traceA = 0;
        for (int r = 0; r < N; r++) {
            traceA += AHA[N*r+r].vec[0];
        }
        float incValue = thresh * traceA / N;
        for (int r = 0; r < N; r++) {
            AHA[N*r+r].vec[0] = AHA[N*r+r].vec[0] + incValue;
        }
    }
}

bool performTikhonovReg(cuNDArray<complext<float>>* AHA, float thresh) {
    dim3 blockDim(512,1,1);
    dim3 gridDim(1,1,1);
    unsigned int N = AHA->get_size(0);
    tikhonovKernel<<< gridDim, blockDim >>>(AHA->get_data_ptr(), thresh, N);
    return true;
}
```

### 7.10.2 Reconstruction using SPIRiT

In this section, we describe the implementation of SPIRiT reconstruction within the `MrRecon` codebase. The implementation described here corresponds primarily
to the bFISTA method as applied to SPIRiT and discussed in a previous chapter. However, this specific implementation refers primarily to the selection of the optimization algorithm. Other algorithms can be used; where relevant, we mention their implementations within the codebase.

The gadget stream configuration XML file for the primary SPIRiT reconstruction implemented in the MrRecon codebase is `vSpiritFISTA.xml`, located in the directory `gadgets/MrRecon/config`; the listing for this XML file is included in its entirety below.

```xml
<?xml version="1.0" encoding="UTF-8"?>
  <reader>
    <slot>1008</slot>
    <dll>gadgetron\_mricore</dll>
    <classname>GadgetIsmrmrdAcquisitionMessageReader</classname>
  </reader>

  <writer>
    <slot>1004</slot>
    <dll>gadgetron\_mricore</dll>
    <classname>MRIImageWriterCPLX</classname>
  </writer>

  <writer>
    <slot>1005</slot>
    <dll>gadgetron\_mricore</dll>
    <classname>MRIImageWriterFLOAT</classname>
  </writer>

  <writer>
    <slot>1006</slot>
    <dll>gadgetron\_mricore</dll>
    <classname>MRIImageWriterUSHORT</classname>
  </writer>

  <!--
  <gadget>
    <name>MRINoiseAdjust</name>
    <dll>gadgetron\_mricore</dll>
    <classname>NoiseAdjustGadget</classname>
  </gadget>
  -->

</gadgetronStreamConfiguration>
```
<gadget>
  <name>PCA</name>
  <dll>gadgetron_mricore</dll>
  <classname>PCACoilGadget</classname>
</gadget>

<gadget>
  <name>CoilReduction</name>
  <dll>gadgetron_mricore</dll>
  <classname>CoilReductionGadget</classname>
  <property><name>coils_out</name><value>12</value></property>
</gadget>

<gadget>
  <name>RemoveROPartialFourierAndOversampling</name>
  <dll>mrrecon</dll>
  <classname>RemoveROPartialFourierAndOversamplingGadget</classname>
</gadget>

<gadget>
  <name>PerSliceAccumulator</name>
  <dll>mrrecon</dll>
  <classname>MrReconPerSliceRTAccumulatorGadget</classname>
  <property><name>maxReconedFrames</name><value>-1</value></property>
  <property><name>correctFrameNumbers</name><value>true</value></property>
</gadget>

<gadget>
  <name>PrintTime</name>
  <dll>mrrecon</dll>
  <classname>PrintTimeGadget</classname>
  <property><name>isEndTime</name><value>false</value></property>
  <property><name>statusString</name><value>Elapsed Reconstruction Time</value></property>
  <property><name>fileName</name><value>reconTime.log</value></property>
</gadget>

<gadget>
  <name>CropImageSeries</name>
  <dll>mrrecon</dll>
  <classname>CropImageSeriesGadget</classname>
  <property><name>roCropBottom</name><value>1</value></property>
  <property><name>roCropTop</name><value>1</value></property>
  <property><name>performTiming</name><value>true</value></property>
</gadget>
<gadget>
  <name>CoilCompressionImageSeries</name>
  <dll>mrrecon</dll>
  <classname>CoilCompressionImageSeriesGadget</classname>
  <property><name>Coils</name><value>12</value></property>
  <property><name>performTiming</name><value>true</value></property>
</gadget>

<gadget>
  <name>Frame Merge</name>
  <dll>mrrecon</dll>
  <classname>FrameMergeGadget</classname>
  <property><name>performMerge</name><value>true</value></property>
  <property><name>performTiming</name><value>true</value></property>
</gadget>

<!-- these buffers are necessary on 3gb card -->

<gadget>
  <name>Recon Buffer</name>
  <dll>mrrecon</dll>
  <classname>BufferGadget</classname>
</gadget>

<gadget>
  <name>GRAPPA Recon</name>
  <dll>mrrecon</dll>
  <classname>RtKGrappaReconGadget</classname>
  <property><name>kernelSizeCOL</name><value>5</value></property>
  <property><name>kernelSizeLIN</name><value>4</value></property>
  <property><name>sendOnAcqData</name><value>true</value></property>
  <property><name>tikhonovReg</name><value>5e-4</value></property>
  <property><name>performTiming</name><value>true</value></property>
</gadget>

<!--
<gadget>
  <name>Spatial Median Filter</name>
  <dll>mrrecon</dll>
  <classname>SpatialMedianFilterGadget</classname>
  <property><name>performFilter</name><value>false</value></property>
  <property><name>filterSize</name><value>3</value></property>
</gadget>

222
<gadget>
  <name>Temporal Filter </name>
  <dll>mrrecon</dll>
  <classname>TemporalFilterGadget </classname>
  <property><name>blurringWeight </name><value>5</value></property>
</gadget>

<gadget>
  <name>Recon Buffer </name>
  <dll>mrrecon</dll>
  <classname>BufferGadget </classname>
</gadget>

<gadget>
  <name>SPIRIT Recon</name>
  <dll>mrrecon</dll>
  <classname>SpiritReconGadget </classname>
  <property><name>reconMethod</name><value>2</value></property>
  <property><name>kernelSizeCOL</name><value>9</value></property>
  <property><name>kernelSizeLIN</name><value>9</value></property>
  <property><name>tikhonovReg</name><value>5e-3</value></property>
  <!-- NLCG: weight for L1 Joint Sparsity gradient term -->
  <property><name>transformWeight</name><value>4e-5</value></property>
  <!-- current values -->
  <property><name>shrinkageWeight</name><value>3e-5</value></property>
  <property><name>shrinkageTemporalScalingWeight</name><value>1</value></property>
  <!-- if true, use L12 shrinkage to impose joint sparsity across coils -->
  <property><name>useL12Shrinkage</name><value>true</value></property>
  <property><name>fidelityWeight</name><value>1</value></property>
  <!-- Wavelet Sparsity Options -->
  <property><name>shrinkageTemporalScalingWeight</name><value>1</value></property>
</gadget>

Data Consistency Weight
<fidelityWeight>1</fidelityWeight>

<!-- if true, use L12 shrinkage to impose joint sparsity across coils -->
<useL12Shrinkage>true</useL12Shrinkage>
<!-- if true, use SCoRe reweighting to adjust weights for shrinkage in each band -->
<property><name>useScoreReweighting</name><value>false</value></property>

<property><name>CalculateAfterNIter</name><value>10</value></property>

<property><name>ScoreGlobalWeight</name><value>1</value></property>

<!-- if true, use Adaptive Restart if cost function goes up -->
<property><name>AdaptiveRestart</name><value>false</value></property>

<!-- if true, use the actual calculated accel. rate rather than the nominal rate -->
<property><name>useActualPERate</name><value>true</value></property>

<!-- Self-consistency Weight -->
<property><name>spiritKernelWeight</name><value>1e-2</value></property>

<!-- Initialization with L1 POCS -->
<property><name>useL1POCSInitialization</name><value>true</value></property>
<property><name>l1pocs_maxIter</name><value>12</value></property>
<property><name>l1pocs_normThresh</name><value>1e-3</value></property>
<property><name>l1pocs_waveletWeight</name><value>8e-4</value></property>

<!-- Kernel Reestimation Mode - 0: none. 1: average-all. 2: frame-by-frame. -->
<property><name>kernelReestimationMode</name><value>2</value></property>

<!-- Wavelet Regularization Relaxation -->
<property><name>waveletRelaxationIterDuration</name><value>16</value></property>
<property><name>waveletRelaxationStartingFactor</name><value>10</value></property>
Temporal Gaussian filter

<property><name>useTemporalGaussianFilter</name><value>true</value></property>
<property><name>temporalGaussianFilterBlurringWeight</name><value>16</value></property>

Stopping Criterion

<property><name>minimumNormThresh</name><value>0</value></property>
<property><name>maximumIterations</name><value>300</value></property>
<property><name>minimumIterations</name><value>20</value></property>
<property><name>objectiveTolerance</name><value>1e-3</value></property>
<property><name>acqLinesObj</name><value>0</value></property>
<property><name>writeOutRawData</name><value>true</value></property>
<property><name>fixAcquiredData</name><value>true</value></property>
<property><name>performTiming</name><value>true</value></property>

Padding Gadget
<dll>mrrecon</dll>
<classname>PadToReconSizeGadget</classname>
<property><name>padFE</name><value>true</value></property>
<property><name>padPE</name><value>false</value></property>
<property><name>performTiming</name><value>true</value></property>

PrintTime
<dll>mrrecon</dll>
<classname>PrintTimeGadget</classname>
<property><name>isEndTime</name><value>true</value></property>
<property><name>statusString</name><value>SPIRiT Reconstruction</value></property>
<property><name>fileName</name><value>reconTime.log</value></property>
<gadget>
  <name>SendOutByPhase</name>
  <dll>mrrecon</dll>
  <classname>SendOutByPhaseGadget</classname>
  <property>
    <name>separateSeries</name>
    <value>true</value>
  </property>
  <property>
    <name>truncateFirstFrames</name>
    <value>0</value>
  </property>
  <property>
    <name>truncateLastFrames</name>
    <value>1</value>
  </property>
  <property>
    <name>truncateToCardiacCycle</name>
    <value>true</value>
  </property>
  <property>
    <name>truncateToFirstCardiacCycle</name>
    <value>true</value>
  </property>
  <property>
    <name>triggerOffset</name>
    <value>2</value>
  </property>
</gadget>

<!--
<gadget>
  <name>ConstAutoScale</name>
  <dll>mrrecon</dll>
  <classname>ConstAutoScaleGadget</classname>
</gadget>
-->

<gadget>
  <name>SOSCombine</name>
  <dll>mrrecon</dll>
  <classname>SOSCombineGadget</classname>
</gadget>

<gadget>
  <name>Extract</name>
  <dll>gadgetron_mricore</dll>
  <classname>ExtractGadget</classname>
</gadget>

<gadget>
  <name>AutoScale</name>
  <dll>gadgetron_mricore</dll>
  <classname>AutoScaleGadget</classname>
</gadget>

<gadget>
  <name>FloatToShort</name>
  <dll>gadgetron_mricore</dll>
  <classname>FloatToUShortGadget</classname>
</gadget>
Other XML configuration files implementing the SPIRiT reconstruction exist in this directory. \texttt{vSpirit.xml} implements the SPIRiT reconstruction using HT-GRAPPA and therefore works only for uniformly downsampled trajectories. \texttt{vSpiritFISTA\_T2Star.xml} utilizes the accumulator gadget specific for $T_2^*$ data and implements a SPIRiT reconstruction in that context. \texttt{vSpiritNLCG.xml} is intended to implement a SPIRiT reconstruction using the nonlinear conjugate gradient (NLCG) method. Any XML file of the form \texttt{vSpiritFISTA\_<date>.xml} is an older version containing older reconstruction parameters. The most current version of the SPIRiT reconstruction is contained in \texttt{vSpiritFISTA.xml}; other variants may be informative, but may also require some modification in order to obtain a successful reconstruction. The block schematic diagram for \texttt{vSpiritFISTA.xml} is shown in Figure 7.14. Many of the gadgets used here are similar or identical to those used in the GRAPPA reconstruction methods described earlier. We restrict our description to the gadgets here that have not been described previously in discussing the GRAPPA methods.

\texttt{RemoveROPartialFourierAndOversamplingGadget} combines the two previously described gadgets, \texttt{RemoveROOveramplingGadget} and \texttt{RemovePartialFourierGadget}, into a single gadget that deals with the processing of data in the frequency encoding direction. The primary difference with this approach is that capability for ensuring consistency in the locations of the non-acquired data points in the frequency encoding is maintained by imposing a zero-value condition on these locations. This ensures
Figure 7.14: Block schematic diagram for vSpiritFISTA.xml. Diagram indicates which DLL each gadget is selected from in addition to the serialization of the data being passed between various gadgets in the configuration stream.
that iterative procedures further downstream treat these locations as unacquired locations. With separate distinct gadgets for both operations, numerical inaccuracies in oversampling removal results in non-zero values at non-acquired locations; due to the structure of downstream reconstruction code, errors due these non-zero values persist.

**PrintTimeGadget** occurs twice in the configuration file for SPIRiT reconstruction, and is used to perform timing measurements for the entire reconstruction. Of particular note is the fact that this gadget occurs twice within the gadget stream configuration. Since gadgets are implemented as class instantiations, this is possible with the Gadgetron and often useful. Once could conceivable use this flexibility to perform multiple reconstruction passes using the same reconstruction approach but with different input parameters. While the Gadgetron provides its own timing measurement functionality, it is challenging to perform timing measurements across multiple gadgets with multiple datasets passing through the stream. (Acquired datasets can include cine data for multiple slices; such datasets are sent through serially, and it is up to the reconstruction code to properly handle such datastreams.) File I/O for writing time stamps and timing measurements are provided via this gadget.

Another gadget that appears multiple times within this configuration stream is the **BufferGadget**. The purpose of this gadget is to buffer data at a certain point until all data processing has completed up until that point. This is important especially in the processing of large datasets using GPU processing, as is the case in a multiple gadgets in this gadget stream. As data processing occurs within each gadget asynchronously, it may be the case that one gadget may be processing data at the same time as another gadget. Since memory is relatively more constrained on the GPU compared to the
CPU, this can create issues with memory intensive processing. If no code for memory management of GPU processing is implemented, this can result in fatal crashes that suspend data processing. The BufferGadget essentially serves as a mutual exclusion lock on the GPU and can help improve the stability. However, excessive use of this gadget throughout the gadget stream can have the impact of increasing data processing latency due to the forced serialization of multiple datasets passing through the gadget stream. This can have an impact on reconstruction latency especially in scenarios where multiple slices are reconstructed in sequence (e.g. the acquisition and reconstruction of a stack of short-axis cine slices).

Two additional gadgets referred to in the XML file that can be used are the SpatialMedianFilterGadget and the TemporalFilterGadget. The former implemented a frame-by-frame median filter, and the latter implements a pixel-wise temporal filter. Both can be used optionally to improve the initialization passed to the SPIRiT reconstruction gadget. The temporal filter functionality has been incorporated into the SPIRiT reconstruction gadget itself. Previous tests using the spatial median filter seem to indicate that this has little impact on reconstruction results in the context of our work; however, the functionality exists and may be useful in other applications.

The FrameMergeGadget performs a sliding window view-sharing. Multiple routines exist for 3-, 5- and all-frame merging. The specific routine is hard-coded in the gadget code and cannot be dynamically changed. Currently, following the approach in [13], a 3-frame sliding window view-sharing approach is used. In this case, for each pixel, no operation is performed if the point is acquired. However, if the point is not acquired, an average of the acquired points at the corresponding location in the
nearby frames is calculated and assigned to this location. These points only serve the purpose of initialization points; they are not labelled as originally acquired points in the dataset. This approach can be useful in aiding reconstruction especially at high acceleration rates where the distance between acquired k-space lines can be quite large. Implementation of this code occurs on the GPU, and different frame-merging routines exist as CUDA routines.

This gadget stream configuration uses the KGRAPPA reconstruction method as an initialization for the SPIRiT reconstruction.

We now dedicate the remainder of this section to describing the SPIRiT reconstruction gadget SpiritReconGadget. We first provide a description of the input parameters needed by the gadget, in particular (1) their definitions, (2) whether or not the specific parameters is in use, and (3) how the parameter is used.

The parameter reconMethod specifies the particular iterative method (optimization algorithm) to be used to perform the reconstruction. Valid options are as follows:

- 0: Use a POCS (Projection-onto-convex-sets) method [85]
- 1: Use a NLCG method
- 2: Use a FISTA approach.

As the primary focus of this work has been on the FISTA approach, this remains the most complete reconstruction method in terms of code implementation. It is not guaranteed that either the POCS or NLCG method work out-of-the-box in the current MrRecon code. All approaches utilize the GPU for processing.

The parameters kernelSizeCOL and kernelSizeLIN specify the size of the SPIRiT kernel to be used given a kernelSizeCOL × kernelSizeLIN SPIRiT kernel. The values
must be odd-valued, and in principle, a rectangular kernel should be supported. Kernel estimation within the SPIRiT reconstruction code closely follows that described for the GRAPPA reconstruction code with small modifications to account for the dense kernel used. Kernel estimation is performed on the GPU.

The parameter $tikhonovReg$ adjusts the amount of Tikhonov regularization used in estimating the SPIRiT kernel. Note that there exists a Tikhonov regularization parameter for both the initial GRAPPA reconstruction and the SPIRiT reconstruction. These two parameters do not need to be the same. Adjusting the regularization parameter in either reconstruction can impact overall blurring and artifact levels introduced by the complete reconstruction method.

The parameter $fidelityWeight$ adjusts the contribution from the data fidelity term in the SPIRiT reconstruction [85].

The parameter $transformWeight$ adjusts the contribution from $\ell_1$ regularization in the sparse transform domain $\Psi$ for the NLCG optimization approach, only. Different parameter entries are used for the FISTA optimization method and will be described shortly. In the NLCG approach, $\ell_1$ regularization occurs as a term within the gradient step; this parameter adjusts the contribution from this term.

The next two parameters $shrinkageWeight$ and $shrinkageTemporalScalingWeight$ adjust the contribution from the $\ell_1$ regularization in the sparse transform $\Psi$ for the FISTA optimization approach. In our work, a 3D single level wavelet transform is used as the sparse transform domain, resulting in eight different wavelet bands. The parameter $shrinkageWeight$ is a global multiplicative factor adjusting the scaling for each of the bands. The high temporal frequency bands (4 bands out of 8) are automatically scaled by a factor proportional to the input acceleration rate
(actual rate as calculated by the accumulator gadget). The parameter `shrinkage-TemporalScalingWeight` is an additional multiplicative scaling factor for the high temporal frequency bands. Note that, modulo scaling by the acceleration rate, the total scaling factor applied to the high temporal frequency bands is therefore the product of these two input parameters. In the default mode, manually tuned weights are used for weighting the regularization strengths for each of the wavelet bands. These are currently hardcoded as follows: [1 1 1 R R R R], where R refers to the input acceleration rate.

The following parameter `useL12Shrinkage` is a Boolean property that toggles whether joint sparsity across coils should be used [111]. Joint sparsity adjusts the shrinkage calculation such that the $\ell_2$ norm across coils is taken into account. Therefore pixel locations that are zero-valued across all coils are assumed to be noise and are therefore pushed towards zero during the shrinkage operation, whereas pixel locations whose $\ell_2$ norm across coils is not zero are considered to contain signal.

The following parameter `useScoreReweighting` is a Boolean flag that toggles use of SCoRe reweighting [120], also referred to as composite regularization, across the sparse transform sub-dictionaries. If set to a value of `false` then the hard-coded fixed regularization weights are used; if set to `true` composite regularization is used. The following two parameters are composite regularization related parameters. The first parameter, `CalculateAfterNIter`, takes an integer value. This value is the number of iterations after which SCoRe reweighting should be used. During these initial iterations, the values of the regularization weights are kept fixed. Currently, the values for each band are fixed to 1. The second parameter, `ScoreGlobalWeight`, is a float value that is a global multiplicative scaling factor on the SCoRe weights.
The next parameter **AdaptiveRestart** is a Boolean property toggling whether adaptive restart [151] should be used. In the FISTA algorithm, the weighting of the linear combination depends on an iteratively calculated weighting. If adaptive restart is used, the iterative weight calculation for the linear combination is reset should the cost function ever increase, which can occur since we are essentially using a second order acceleration technique in FISTA.

The parameter **useActualPERate** is a Boolean flag toggling whether the actual acceleration rate (in the phase encoding direction for 2D cine) should be used in place of the nominal accelerate rate as reported in the acquisition parameters. In the case of variable density sampling, these two values can differ, leading to differing regularization strengths being applied. The actual acceleration rate is calculated by the accumulator gadget as serial k-space lines are received and binned within a specific dataset. This acceleration rate is then stored and passed downstream to the reconstruction gadgets.

The parameter **spiritKernelWeight** adjusts the contribution from the SPIRiT self-consistency term in the SPIRiT reconstruction problem [85].

The following four parameters pertain to the use of an \( \ell_1 \) POCS initialization prior to the use of either FISTA or NLCG in the SPIRiT reconstruction. POCS initialization follows that of [85] in that regularization in the 2D wavelet domain is used. In this case, we are using a 2D redundant db1 wavelet transform with 3 levels of decomposition. The wavelet transform is implemented on the GPU. The Boolean flag **useL1POCSInitialization** toggles whether this module should be enabled. If it is **false**, then the input initialization provided to the gadget will be used. If it is **true**, then POCS will be run using the input initialization as its input, with
the result being passed downstream within the gadget for further processing. The parameter `l1pocs_maxIter` specifies the number of POCS iterations that should be run. The parameter `l1pocs_normThresh` specifies the stopping criterion in terms of the $\ell_2$ norm difference between iterates. The parameter `l1pocs_waveletWeight` specifies the strength of regularization in the 2D wavelet domain that should be used.

After an initial reconstruction (i.e. GRAPPA and/or POCS), it is often beneficial to re-estimate the SPIRiT kernels. This can be especially useful in highly dynamic scenarios, such as high heart rates or heavy breathing motion. In this case, the following property `kernelReestimationMode` can be adjusted. A value of 0 indicates no re-estimation of the SPIRiT kernels. A value of 1 indicates re-estimation of the average-all SPIRiT kernel, i.e. all frames of the input initialization to the FISTA/NLCG solver are averaged together in the time dimension, and a kernel is estimated from the (now single-frame) resulting k-space. A value of 2 indicates re-estimation of “frame-by-frame” kernels. In this case, each (fully-sampled) frame is treated as an input for the kernel estimation, and therefore a stack of kernels is generated, with each kernel being applied to the corresponding frame in the FISTA/NLCG solver.

Wavelet relaxation, in which the regularization strength is temporally increased and then iteratively reduced, often can help aid convergence of optimization algorithms. The following two parameters implement this functionality for SPIRiT reconstruction. The first parameter, `waveletRelaxationIterDuration` is the number of iterations over which wavelet relaxation should occur. During this period of iterations, a global multiplicative scaling factor for the regularization strength is imposed that
starts at the value specified by `waveletRelaxationStartingFactor` and decreases exponentially to a value of 1 over `waveletRelaxationIterDuration` iterations.

### 7.10.3 Reconstruction using SENSE

In this section, we describe the implementation of SENSE reconstruction within the `MrRecon` codebase. The implementation described here corresponds primarily to the bFISTA approach as applied to SENSE and discussed in a previous chapter. The implementation described here is similar in approach to that for SPIRiT described in the previous section. The primary difference is that neither frame merging nor a GRAPPA initialization is used, and the SPIRiT reconstruction gadget is replaced by a SENSE reconstruction gadget. There exist two XML configuration files for the SENSE reconstruction in the `MrRecon` codebase. The first file, found in `gadgets/MrRecon/config/vSenseFISTA.xml` implements the bFISTA approach for SENSE reconstruction. The second file, found in `gadgets/MrRecon/config/vSenseFISTA-prev.xml` is an older version, containing older parameters, that may be useful as a reference.

The gadget stream configuration XML file `vSenseFISTA.xml` is shown in its entirety below.

```xml
<reader>
  <slot>1008</slot>
  <dll>gadgetron_mricore</dll>
  <classname>GadgetIsmrmrdAcquisitionMessageReader</classname>
</reader>
```
Figure 7.15: Block schematic diagram for vSenseFISTA.xml. Diagram indicates which DLL each gadget is selected from in addition to the serialization of the data being passed between various gadgets in the configuration stream. The primary difference between the SENSE reconstruction is the use of the SENSE reconstruction gadget and the omission of the frame merging and GRAPPA reconstruction gadgets.
<writer>
<slot>1004</slot>
<dll>gadgetron_mricore</dll>
<classname>MRIImageWriterCPLX</classname>
</writer>

<writer>
<slot>1005</slot>
<dll>gadgetron_mricore</dll>
<classname>MRIImageWriterFLOAT</classname>
</writer>

<writer>
<slot>1006</slot>
<dll>gadgetron_mricore</dll>
<classname>MRIImageWriterUSHORT</classname>
</writer>

<!--
<gadget>
<name>MRINoiseAdjust</name>
<dll>gadgetron_mricore</dll>
<classname>NoiseAdjustGadget</classname>
</gadget>
-->

<!--
<gadget>
<name>PCA</name>
<dll>gadgetron_mricore</dll>
<classname>PCACoilGadget</classname>
</gadget>
-->

<gadget>
<name>CoilReduction</name>
<dll>gadgetron_mricore</dll>
<classname>CoilReductionGadget</classname>
<property>
<name>coils_out</name><value>12</value></property>
</gadget>

<!--
<gadget>
<name>RemoveROPartialFourierAndOversampling</name>
<dll>mrrecon</dll>
<classname>RemoveROPartialFourierAndOversamplingGadget</classname>
</gadget>
-->

<gadget>
<name>PerSliceAccumulator</name>
<dll>mrrecon</dll>
<classname>MrReconPerSliceRTAccumulatorGadget</classname>
<property>
<name>maxReconedFrames</name><value>-1</value></property>
<property>
<name>correctFrameNumbers</name><value>true</value></property>
</gadget>
<gadget>
  <name>PrintTime</name>
  <dll>mrrecon</dll>
  <classname>PrintTimeGadget</classname>
  <property><name>isEndTime</name><value>false</value></property>
  <property><name>statusString</name><value>Elapsed Reconstruction Time</value></property>
  <property><name>fileName</name><value>reconTime.log</value></property>
</gadget>

<gadget>
  <name>CropImageSeries</name>
  <dll>mrrecon</dll>
  <classname>CropImageSeriesGadget</classname>
  <property><name>roCropBottom</name><value>1</value></property>
  <property><name>roCropTop</name><value>1</value></property>
  <property><name>performTiming</name><value>true</value></property>
</gadget>

<gadget>
  <name>CoilCompressionImageSeries</name>
  <dll>mrrecon</dll>
  <classname>CoilCompressionImageSeriesGadget</classname>
  <property><name>Coils</name><value>12</value></property>
  <property><name>performTiming</name><value>true</value></property>
</gadget>

<gadget>
  <name>Frame Merge</name>
  <dll>mrrecon</dll>
  <classname>FrameMergeGadget</classname>
  <property><name>performMerge</name><value>true</value></property>
  <property><name>performTiming</name><value>true</value></property>
</gadget>

<!-- these buffers are necessary on 3gb card -->

<gadget>
  <name>Recon Buffer</name>
  <dll>mrrecon</dll>
  <classname>BufferGadget</classname>
</gadget>

-->
<gadget>
    <name>SENSE Recon</name>
    <dll>mrrecon</dll>
    <classname>SenseReconGadget</classname>
    <property><name>reconMethod</name><value>1</value></property>
    <property><name>fidelityWeight</name><value>1</value></property>
    <property><name>shrinkageWeight</name><value>1e-3</value></property>
    <property><name>shrinkageTemporalScalingWeight</name><value>1</value></property>
    <property><name>minimumNormThresh</name><value>1e-9</value></property>
    <property><name>maximumIterations</name><value>150</value></property>
    <property><name>minimumIterations</name><value>20</value></property>
    <property><name>objectiveTolerance</name><value>1e-3</value></property>
    <property><name>waveletRelaxationIterDuration</name><value>6</value></property>
    <property><name>waveletRelaxationStartingFactor</name><value>10</value></property>
    <property><name>waveletRelaxationIterStepSize</name><value>3</value></property>
    <property><name>waveletRelaxationContFactor</name><value>5</value></property>
    <property><name>waveletRelaxationPowerFactor</name><value>4</value></property>
    <property><name>useScoreReweighting</name><value>false</value></property>
    <property><name>CalculateAfterNIter</name><value>10</value></property>
</gadget>
<property><name>ScoreGlobalWeight</name><value>3e-3</value></property>

<!-- if true, use Adaptive Restart if cost function goes up -->
<property><name>AdaptiveRestart</name><value>false</value></property>

<!-- Other Flags -->
<property><name>useActualPERate</name><value>true</value></property>
<property><name>writeOutRawData</name><value>true</value></property>
<property><name>performTiming</name><value>true</value></property>

</gadget>

<gadget>
<name>Padding Gadget</name>
<dll>mrrecon</dll>
<classname>PadToReconSizeGadget</classname>
<property><name>padFE</name><value>true</value></property>
<property><name>padPE</name><value>false</value></property>
<property><name>performTiming</name><value>true</value></property>
</gadget>

<gadget>
<name>PrintTime</name>
<dll>mrrecon</dll>
<classname>PrintTimeGadget</classname>
<property><name>isEndTime</name><value>true</value></property>
<property><name>statusString</name><value>SENSE Reconstruction</value></property>
<property><name>fileName</name><value>reconTime.log</value></property>
</gadget>

<gadget>
<name>SendOutByPhase</name>
<dll>mrrecon</dll>
<classname>SendOutByPhaseGadget</classname>
<property><name>separateSeries</name><value>true</value></property>
</gadget>
<property><name>truncateFirstFrames</name><value>0</value></property>
<property><name>truncateLastFrames</name><value>1</value></property>
<property><name>truncateToCardiacCycle</name><value>true</value></property>
<property><name>truncateToFirstCardiacCycle</name><value>true</value></property>
<property><name>triggerOffset</name><value>2</value></property>
</gadget>
<gadget>
<name>SOSCombine</name>
<dll>mrrecon</dll>
<classname>SOSCombineGadget</classname>
</gadget>
<gadget>
<name>Extract</name>
<dll>gadgetron_mricore</dll>
<classname>ExtractGadget</classname>
</gadget>
<gadget>
<name>ConstAutoScale</name>
<dll>mrrecon</dll>
<classname>ConstAutoScaleGadget</classname>
</gadget>
<gadget>
<name>AutoScale</name>
<dll>gadgetron_mricore</dll>
<classname>AutoScaleGadget</classname>
</gadget>
<gadget>
<name>FloatToShort</name>
<dll>gadgetron_mricore</dll>
<classname>FloatToUShortGadget</classname>
</gadget>
<gadget>
<name>ImageFinishUSHORT</name>
<dll>gadgetron_mricore</dll>
<classname>ImageFinishGadgetUSHORT</classname>
</gadget>
</gadgetronStreamConfiguration>
A schematic diagram showing the gadget chain is shown in Figure 7.15.

The gadget properties that are required for the SenseReconGadget have already been discussed in the context of the SPIRiT reconstruction gadget stream configuration XML file. Therefore, we do not allocate space to discuss the properties present in this gadget. Rather, in the following, we highlight the differences between the SENSE reconstruction and the SPIRiT reconstruction. The primary difference is that the SENSE reconstruction implemented here admits the use of only the bFISTA optimization approach. In principle, an NLCG approach is straightforward to implement; however, the current iteration of this code does not provide this implementation. Additionally, because there exists no initialization, no options are provided for reestimation of coil sensitivity maps. Reestimation of coil sensitivity maps can conceivably be implemented by chaining two instances of the SENSE reconstruction gadgets, with the latter gadget taking the result of the former gadget as an initialization.

7.11 Detailed Description of MrReconPerSliceRTAccumulatorGadget

In this section, we provide a detailed description of the MrReconPerSliceRTAccumulatorGadget. For brevity, we refer to this gadget as the RT accumulator gadget. There exist multiple additional accumulator gadgets within the MrRecon code:

- MrReconPerSliceSegAccumulatorGadget
- MrReconPerSliceT2StarAccumulatorGadget
- MrReconPerSliceAccumulatorGadget
MrReconPerSliceSegAccumulatorGadget contains slight modifications for allowing the reading of segmented data, which uses an additional segment index. MrReconPerSliceT2StarAccumulatorGadget contains modifications for reading of $T_2^*$ data, which also uses a different index. MrReconPerSliceAccumulatorGadget is a generic accumulator gadget corresponding to an older version of the RT accumulator gadget. As the structure of each of these accumulator gadgets is similar, we focus our discussion on the RT accumulator gadget. In the SENSE reconstruction, the zero-filled k-space data is used as a direct initialization for the bFISTA optimization, and as a result, there exist no options for POCS initialization or temporal filtering within the SENSE gadget properties.

As the name suggests, the RT accumulator gadget is an accumulator gadget that works for real-time acquisitions. All sampling trajectories are supported, including uniform downsampled (i.e. TGRAPPA), or uniform downsampled with center k-space region (i.e. GRAPPA), or a nonuniformly downsampled trajectory (i.e. VISTA [13]). In principle, radial trajectories should be supported provided downstream gadgets know how to interpret the data.

Figure 7.16 shows a general block diagram for data flow in the RT accumulator gadget. The structure of the data flow for the various other accumulator gadgets is similar. The RT accumulator gadget receives a stream of individual k-space lines (i.e. one phase encoding line for Cartesian, a radial line for Radial, or a spiral for Spiral). Each k-space line is indexed by a 10-dimensional array in the ISMRMRD format that specifies the particular location of the line. For real-time, single-shot cine, only a few are relevant: COL, LIN, PAR, PHA, and SLICE. COL refers to the frequency encoding direction; LIN refers to the phase encoding direction; PAR refers
Figure 7.16: Schematic diagram for accumulator gadget. The \texttt{process} function receives an input stream of k-space lines. After each k-space line is received, a check is performed to see whether the k-space lines for a complete slice is received. If so, correct image headers for this slice are formed, and the slice data is sent downstream for further processing. A \texttt{CLOSE} signal from the parent Gadgetron process terminates the gadget once all slices have been received.

To the partition direction in the case of 3D imaging; and PHA refers to the cardiac phase. For multiple slices, the SLICE index is also incremented. For the segmented accumulator gadget, the primary difference here is that the SEG index is also used to specify the specific segment being acquired. It is up to the vendor-specific code to perform a proper mapping from vendor-specific indices to the 10-dimensional array as specified by the ISMRMRD format. The number of channels is provided by in the process configuration and is not indexed for each k-space line received.

Based on the input configuration that is processed by the \texttt{process\_config} function, an initial buffer corresponding to the size of the data is allocated, and each k-space line is placed in its proper location. After each k-space line is received, the SLICE index is checked to see if a complete slice has been received (i.e. all phases for
a single slice). If this is the case, a special function is called to process this slice and send it downstream for further processing. In this processing, proper image headers containing the necessary information for downstream processing are created. If it is necessary to store phase-specific image header information, this is taken into account at this point. Physiological information is also archived so as to ensure that final data returned to the scanner is compatible.

Once the last line of k-space is sent out by the Gadgetron client, a terminate signal is sent to the server. This signal is received by each gadget asynchronously, and the default close function forces each gadget to shutdown once all processing is complete. As the accumulator’s functionality is atomized by k-space line, it is possible for the gadget to terminate in the middle of receiving k-space lines for a certain slice, thereby causing the gadget to not send the final slice. Therefore, modifications are made to the gadget’s close function to detect if we are still receiving the data for the last slice. Once the last line for this slice has been received, the close function makes a function call to send out the last slice and then allows the parent process to proceed with termination.

The above provides a description for the accumulator gadget that encompasses the general functionality of all accumulators within the MrRecon codebase. Additional functionality in terms of various data processing necessary for downstream gadgets is described in the remainder of this section.

The RT accumulator provides functionality for tracking exactly which k-space lines in the phase encoding (PE) direction are received. This is implemented in the code from the file gadgets/MrRecon/MrReconPerSliceRTAccumulatorGadget.cpp shown
if (phase < acquiredLines_.size()) {
    // It exists. So push acquired line number on to the corresponding entry
    acquiredLines_[phase].push_back(m1->getObjectPtr()->idx.kspace_encode_step_1);
} else {
    // It doesn’t, so insert a new entry
    std::vector<unsigned int>* lineNoEntry = new std::vector<unsigned int>;
    lineNoEntry->push_back(m1->getObjectPtr()->idx.kspace_encode_step_1);
    acquiredLines_.push_back(*lineNoEntry);
}

The variable acquiredLines_ is declared in the file gadgets/MrRecon/MrReconPerSliceRTAccumulatorGadget.h as follows:

std::vector<std::vector<unsigned int>> acquiredLines_;

Therefore, the variable acquiredLines_ plays the role of a variable length list of lists; with the inner list containing an arbitrary number of k-space line indices corresponding the acquired k-space lines for a given phase, and the outer list containing the acquired phases. Elements in this double-layered list are dynamically allocated, so specific care in memory deallocation is necessary in order to avoid memory leaks. This data structure provides a flexible way for storing any sampling trajectory in the context of 2D Cartesian sampling. In principle, a straightforward extension to radial sampling trajectories can be made. Additionally, with an additional layer, an extension to 3D Cartesian sampling can be made, allowing the tracking of sampling trajectories for multidimensional datasets. This index of the acquired k-space lines is
included within the message chain passed on to all downstream gadgets, and can be useful in downstream processing.

As the accumulator gadget is responsible for “deserializing” individual k-space lines into logical “datasets,” this gadget also plays the role of defining the gadget chain convention to be adhered to in all downstream gadgets. This gadget chain is in general specified in the header comments of each gadget .cpp and .h file. In some cases, gadgets add their own messages to this chain, but in most cases, the accumulator gadget defines this message chain. The message chain definition as specified in the RT accumulator gadget is shown in the comments from the file gadgets/MrRecon/MrReconPerSliceRTAccumulatorGadget.cpp:

```
2 * MrReconPerSliceRTAccumulatorGadget.cpp
3 *
4 * Samuel Ting, 05/07/2013
5 *
6 * 2014/02/24: All accumulators should create m5. So the order
7 * of the messages
8 * should be as follows:
9 *
10 * m1: dummy image header containing some useful information
11 * (e.g. matrix size, FOV) but not frame specific
12 *
13 * m2: image data - this is the data that the gadgets work on,
14 * not necessarily
15 * the original acquired data
16 *
17 * m3: image header buffer - this buffer contains image headers
18 * with frame specific information
19 *
20 * m4: acquired lines - a buffer of the corresponding acquired
21 * PE line indices
22 *
23 * m5: acquired data - this is the original acquired data -
24 * needed by some
25 * gadgets where the k-space lines are restored
```

Message m1 contains image header information that is used by all downstream gadgets for reconstruction and data processing. In general, this message contains global (non-frame specific) image information that corresponds to the entire dataset (e.g.
matrix size, recon size, FOV, etc.). Eventually, the relevant information from this message will be transferred to individual frame-specific image headers when data is deserialized into frames. Care must be taken to ensure that information within this header matches that provided by the scanner. Any unexpected mismatches can result in the images not displaying on the user console, even with a successful reconstruction that generates image files in an offline environment. Adjustments can be made to data within this message; however, the changes should be expected. For example, if the matrix size is reduced (e.g. an FOV reduction), the corresponding adjustment to the FOV field must be made. This can be made at any point within the gadget stream, but it must be made prior to returning images to the client application.

Of particular note, within this message are various user-defined integer fields. Currently the accumulator gadgets utilize the first four integer fields as shown in the following comments from gadgets/MrRecon/MrReconPerSliceRTAccumulator-Gadget.cpp:

```plaintext
* current user_int usage:

* user_int[0] : minPE_gap_ : minimum gap in PE VISTA line sampling;
  * used for 2-line GRAPPA

* user_int[1] : maxPE_gap_ : maximum gap in PE VISTA line sampling;
  * used for 2-line GRAPPA

* user_int[2] : flag for determining whether this is the last packet sent out in cases where maxReconedFrames_ is used;
  * used in bufferRecon gadget.

* user_int[3] : flag for indicating which GPU to send this packet to if on a multi-gpu setup. This gadget
```
* will only set this to -1, the GpuSplitGadget will correctly set it for each packet based on the desired number of GPUs and number of available GPUs.

Fields `user_int[0]` and `user_int[1]` are used to determine the minimum and maximum kernel sizes that need to be estimated in the context of a two-line GRAPPA reconstruction used as initialization for the SPIRiT reconstruction.

Field `user_int[2]` is used in the case where frames are sent out in batches. It may be the case that the size of the dataset is too large to be processed in a single chunk by downstream gadgets. In this case, the `maxReconFrames` gadget property for the accumulator gadget can be used to process the dataset in chunks. In this case, the user field is used to signal to downstream gadgets that this is the last chunk being sent downstream. In our work, this functionality is not frequently used, and therefore it is not guaranteed that downstream gadgets will properly interpret this flag.

Field `user_int[3]` provides the capability to flag specific packets for processing on specific GPUs. Within most gadgets, functionality is provided to run on a specific GPU and listen for corresponding packets. As most of the work described here has been implemented on a single GPU, this functionality has not been used too frequently.

Message `m2` contains the physical k-space data. This is typically a multidimensional data that for the real-time accumulator is a five dimensional multidimensional array with dimensions as follows: `[COL LIN PAR CHA PHA]`. COL refers to the frequency encoding direction, LIN refers to the phase encoding direction, and CHA refers to the channel dimension. PAR refers to the partition dimension and in the case of 2D dynamic imaging is typically 1.
Message m3 is a buffer containing the specific image headers from each phase. Its type definition is specified in gadgets/MrRecon/MrReconPerSliceRTAccumulator-Gadget.h:

```cpp
std::vector<ISMRMRD::ImageHeader> headerMessages_;```

Each k-space line received by the accumulator gadget contains a header message of the format ISMRMRD::AcquisitionHeader that contains information specific to the acquisition. When the first line of each phase is received (In the ISMRMRD format, this is actually indicated as the first acquisition in a slice.), a new header of type ISMRMRD::ImageHeader is created, and relevant imaging parameters are copied from the acquisition header. Therefore, only one image header is created and buffered for each input phase that corresponds to the first received line of k-space for that phase. It may be more accurate to create the header from the center line of k-space as this is the most relevant k-space line from a signal energy perspective. This buffer of ISMRMRD::ImageHeader headers is used to create the final deserialized series of messages corresponding to the image phase in the gadget SendOutByPhaseGadget. This helps to ensure that information expected by the client application is suitably provided by the Gadgetron server. Of particular note, this buffer of headers contains the physiological time stamps. This information is used by the gadget SendOutByPhaseGadget to detect phases corresponding to a one or more complete cardiac cycles.

Message m4 contains the data structure acquiredLines_ containing the indices for the acquired phase encoding lines as described previously.

Message m5 contains the original acquired data. This is guaranteed to be the original unmodified k-space data as received by the accumulator gadgets. Initially, this data is identical to the contents of message m2; however, all downstream processing
is expected to be performed on message \( m_2 \). It is implicitly expected that all downstream gadgets do not modify the data contained in message \( m_5 \). Most reconstructions require the original k-space data for maintaining data fidelity, and modification of this data may effect unexpected changes in reconstruction results.

In the final portion of this section, we describe some details pertaining to the `process_config` function used in the RT accumulator gadget. The details here are also relevant to the function as utilized in other accumulator gadgets. The primary purpose of the `process_config` function as used here is to parse the relevant limits for the 9-dimensional data structure as specified by the ISMRMRD data format. The dimensions of this data structure are defined as follows:

\[
\begin{bmatrix}
\text{Line} & \text{Partition} & \text{Average} & \text{Slice} & \text{Contrast} & \text{Phase} & \text{Rep} & \text{Set} & \text{Seg}
\end{bmatrix}
\]

The Line counter here refers to the phase-encoding direction. These encoding counters do not take into account the frequency encoding dimension or the channel dimension. More details can be found by studying the header file `ismrmrd.h` in the ISMRMRD code. For real-time cine data, we are primarily interested in the following indices: Line, Partition, Slice, and Phase. The additional dimensions Col (frequency encoding dimension) and Cha (channel dimension) are also used. The Col dimension is determined by parsing the encoding k-space information provided to the function. The channel dimension is not assigned until the first k-space line is received. At this point, the number of active channels is assigned to this dimension. The segmented cine accumulator uses the additional Seg dimension; and the \( T_2^* \) accumulator uses the Contrast dimensions. These modifications are highly dependent on the mapping provided in the vendor-specific interface code between the client application and
the Gadgetron server. Therefore, modifications may be necessary in the context of the programming environments for MRI scanners provided by different vendors.

The `process_config` function is also responsible for identifying the particular parallel imaging (specifically, the trajectory) used in the acquisition. Currently, specific fields can be set by the sequence code for indicating the parallel imaging used. This is detected by the accumulator gadget. However, this input currently is used only to provide user feedback identifying the type of acquisition. The `MrRecon` code currently does not depend on these fields for identifying the type of acquisition.

### 7.12 Concluding Remarks

This concludes our (perhaps inadequate) attempt at providing a detailed and thorough description of the role played by the Gadgetron framework in the broader context of this work. The Gadgetron framework provides a convenient avenue for bringing our proposed reconstruction methods from the “benchside” to the “bedside.” While this goal can be similarly achieved through vendor-specific channels, the use of the Gadgetron framework allows the additional flexibility of code portability across vendors and across MRI machines via the medical network. Additionally, the Gadgetron framework can be straightforwardly deployed on any manner of commercially available computer hardware. This allows a much faster cycle in terms of providing adequate computational power compared to working within a vendor-specific hardware environment.

We have attempted to provide both a high-level perspective as well as a detailed, “developer-oriented” perspective of the functionality provided by the `MrRecon` code-base. Arguably, our attempt at providing a detailed perspective of the functionality
is incomplete. To account for this, we recommend that the interested reader study
the code itself in detail. Since the Gadgetron framework is a rapidly evolving code-
base, the current version of the MrRecon may not work with the latest version of the
Gadgetron. Therefore, we have refrained from providing a complete archive of the
code in this work. However, this code can be made available upon request from the
author.
Chapter 8: Conclusions and Future Directions

In this work, we have presented a holistic framework for applying compressed sensing signal recovery in the context of reconstruction of highly accelerated real-time, single-shot, free-breathing dynamic cardiac cine MRI. The specific context of our work has been related to the imaging of a single slice within the heart and great vessels. We have demonstrated that, from an optimization perspective, our approach is computationally efficient, yielding reconstruction times on the order of tens of seconds that are clinically relevant for compressed sensing approaches. Our approach, termed bFISTA, applies FISTA to the balanced sparse model for compressed sensing to achieve reconstruction results comparable to that obtained using the analysis model and a less computationally efficient nonlinear conjugate gradient approach. The method proposed in this work achieves the same computational efficiency as that of the original FISTA optimization, which was initially applied to the synthesis model.

In addition to testing and validating our approach in a MATLAB programming environment, we have also proposed, implemented, and demonstrated a working implementation of our technique within the Gadgetron framework, which enables direct
reconstruction within the scanner environment itself using sufficiently powerful com-
putational hardware to enable fast reconstruction times. While numerous optimiza-
tion algorithms exist for solving the compressed sensing problem in the context of
reconstruction of dynamic MRI data, the primary challenge addressed in this work
has been to select a particular class of algorithms and utilize it to achieve a practical,
working implementation allowing fast reconstruction within the clinical environment.
This has been a long-standing practical issue in terms of demonstrating the clinical
utility of advanced MRI reconstruction approaches based on compressed sensing.

We have additionally demonstrated that, for implementations of both SPIRiT and
SENSE reconstruction techniques, our bFISTA approach achieves similar diagnostic
image quality compared to conventional segmented cine imaging. As discussed in
the introductory chapters, segmented cine imaging bears the significant limitation
of requiring breath-holds and regular cardiac rhythm. In contrast, real-time, free-
breathing cine acquisitions do not have these requirements, and when combined with
adequate reconstruction techniques, can allow for cine imaging at spatial and temporal
resolutions that provide equivalent diagnostic utility compared to segmented cine
imaging. In this work, we have demonstrated that the bFISTA approach can serve
as a reconstruction technique for achieving real-time cine imaging at the necessary
spatial and temporal resolutions for adequate diagnostic utility. Given that this
holistic compressed sensing framework can provide the adequate spatial and temporal
resolutions necessary in dynamic cine imaging with a practical reconstruction time,
the utility of real-time cine imaging in the clinical environment is greatly increased,
potentially allowing for increased use of real-time techniques in place of segmented
techniques to increase CMR utility especially in patient populations where segmented imaging is not possible.

In this work, we have also presented preliminary results investigating the use of advanced sparsity models that may show potential benefit for image recovery in highly dynamic MRI scenarios. Specifically, we have investigated the use of patch-based regularization for improving the signal recovery model, leading to an improved sparsity model that may be useful for image recovery of both resting and exercise stress cine MRI. This work was implemented and validated in the context of static two-dimensional scenarios as well as retrospectively downsampled segmented cine data and shows promise in improving reconstruction techniques for real-time cine imaging.

In the following sections, we present and discuss a number of topics suggesting future directions extending the practical contributions of this work. Specifically, we discuss the application of iterative compressed sensing reconstruction to other CMR imaging techniques, including a small initial study investigating an application to CMR perfusion imaging. We also discuss some of the potential future directions from the perspective of both the practical implementation within the Gadgetron framework as well as theoretical exploration from the perspective of signal recovery models.

8.1 On the Application to other CMR Imaging Techniques

The primary focus of this work has been on its application to dynamic cine imaging primarily in a single slice (i.e. two spatial dimensions). However, this work can likewise be extended to not only dynamic but also static CMR imaging techniques in two or more measurement dimensions (i.e. spatial, temporal, as well as other additional
measurement dimensions). In this section, we discuss the potential application of this body of work to other CMR imaging techniques. We first motivate this by discussing an earlier preliminary study applying iterative reconstruction to perfusion imaging and then continue with additional remarks regarding the potential application of our paradigm of compressed sensing reconstruction to other CMR techniques.

8.1.1 Preliminary Investigation of Applications to Perfusion Imaging

In this section, we present a small study investigating the application of iterative reconstruction techniques similar to that discussed in this work in the context of three-dimensional myocardial perfusion imaging. In perfusion imaging, snap-shot images of a particular volume (either a slab or slice) are acquired across multiple heartbeats at the same cardiac phase during contrast injection. Therefore, a dynamic cine series is generated showing the contrast wash-in and wash-out for a given volume and cardiac phase across multiple heartbeats. Iterative techniques can therefore be applied within this context with the goal of achieving successful image recovery at higher undersampling ratios compared to previous techniques.

The specific iterative algorithms used in this small study are precursors to the main contributions in this work, and serve only as a motivation for further work in this direction. Additionally, this study serves as an initial exploration of application of iterative reconstruction techniques to 3D perfusion imaging. While we do not discuss 2D perfusion imaging in this section, the work discussed in previous chapters can be directly applied in the context of 2D perfusion imaging.

The work presented in this study has been previously published as the following reference:

8.1.1.1 Introduction and Purpose

Steady-State First Pass Perfusion (SSFPP, [16]) imaging is a recently proposed acquisition technique (Figure 8.1, top) for first-pass contrast-enhanced MRI myocardial perfusion imaging (MPI) using 3D imaging techniques. SSFPP utilizes the steady-state characteristics of the SSFP sequence in place of the typical saturation recovery (SR) preparation to produce excellent signal-to-noise and contrast-to-noise ratio (SNR/CNR) compared to existing acquisition techniques (Figure 8.2). In 3D imaging, increased data requirements lead to a lengthened acquisition window, increasing sensitivity to cardiac motion. While view-sharing across temporal frames from different heartbeats can help mitigate the increased temporal requirements of 3D imaging, this also increases sensitivity to respiratory motion. Combined with the high inherent SNR of 3D imaging, the increased acquisition efficiency of SSFPP and improved image characteristics can be combined with acceleration techniques to shorten the acquisition window and eliminate the need for view-sharing across heartbeats in 3D MPI.

In this work, we present our initial experience in combining SSFPP with a 3D implementation of SPIRiT [85] to improve robustness of 3D MPI to the motion sensitivity that results from a longer acquisition window and view-sharing across heartbeats.
Figure 8.1: Diagram for SSFPP acquisition (TOP) and 3D reconstruction with SPIRiT (BOTTOM). Color figure available in electronic version.
Figure 8.2: Example SR-SPGR, SR-SSFP, SSFPP, and SPGR images at baseline and at peak contrast showing the high SNR and CNR of the SSFPP acquisition technique. Source: [16]

Figure 8.3: Example raw 3D SSFPP images from a volunteer using view-shared GRAPPA (top row) and SPIRiT with a 3D kernel (bottom row). Frames are selected from the end of the acquisition, when the volunteer was no longer able to breath-hold. Significant breathing motion during frames 45-48 introduce blurring of the blood-myocardium boundary in the view-shared GRAPPA results. This blurring is reduced in the SPIRiT results (frames 45-48, arrows) in which there was no view sharing across heart beats.
8.1.1.2 Methods

Resting perfusion images (50 frames, 300-340 ms acquisition time per 3D frame, gated to diastole) were acquired in one healthy volunteer at 1.5T (Siemens, Avanto) using SSFPP imaging (FA: 40°, resolution: 2.2 x 2.8 x 8 mm³, 160 x103 x 6 matrix, 33.3% slab oversampling, TR: 2.7 ms, Multihance [0.1 mmol/kg]). All aspects of this human study were approved by the local Institutional Review Board. A TWIST [152] trajectory (Figure 8.2, Bottom, A) was used. A central core volume of 18 k-space lines (4% of the total k-space volume) was acquired during each heartbeat. Outside of this core volume, lines were acquired in a pseudo-random manner such that view-sharing between 3 adjacent frames (temporal footprint of 3 heart-beats) yielded a GRAPPA trajectory (rate 3 in PE direction with 24 fully-sampled reference lines) for each slice (Figure 8.2, Bottom, B). An effective acceleration rate of 7.30 was achieved using the TWIST trajectory. Data were reconstructed in two ways. First, a GRAPPA reconstruction was used (Figure 8.2, Bottom, C). For each frame, the core volume of data was retained while view-sharing with the two adjacent frames was used in the outer volume to obtain a complete GRAPPA trajectory. View-sharing reduced the effective acceleration rate to 2.68. Slice-by-slice GRAPPA reconstruction within each temporal frame was performed using a 4 x 5 kernel estimated from the temporal average of all frames. Second, a 3D implementation of SPIRiT was used (Figure 3d). The 3D kernel (7 x 7 x 3) was estimated from the same k-space volume used for GRAPPA kernel estimation and then used to iteratively reconstruct each temporal frame without incorporation of shared data from adjacent heartbeats. For simplicity and to reduce computation time, an image-based POCS solver was used. The minimum RMS difference between iterations was used as a stopping point, and
the GRAPPA reconstruction was used to initialize the SPIRiT reconstruction. Average SNR based on random matrix theory [153] and mean edge sharpness based on a sigmoid fitting function [123] were measured for both reconstruction methods. All reconstructions were performed using Matlab 2012b on an Intel Core i5 workstation with 16Gb memory.

8.1.1.3 Results

Figure 4 shows images reconstructed using view-shared GRAPPA (top row) and SPIRiT (bottom row) for a series of frames acquired during free-breathing. Even with a 60% reduction in the acquired data used for reconstruction, SPIRiT provided a 4.45% SNR improvement over view-shared GRAPPA results. Edge sharpness measurement at the boundary between the left and right ventricles (Frame 48, arrow) showed a 9.5% improvement in mean sharpness in frames with breathing motion (Frame 48). In frames without motion (Frame 49), there was no significant reduction in edge sharpness. However, some blurring remains in frames where motion is most severe (Frame 47).

8.1.1.4 Conclusions

We used SSFP with SPIRiT to improve the robustness of 3D myocardial perfusion imaging in the presence of breathing motion. Compared to view-shared GRAPPA techniques, blurring due to breathing motion in reconstructed images was visibly reduced. The use of nonlinear conjugate gradient techniques with $\ell_1$-regularization and more optimal incoherent sampling schemes may further reduce the effect of residual breathing motion in 3D MPI through the use of higher acceleration rates.
8.1.1.5 Additional Concluding Remarks

The work presented in this small study plays the role of an initial exploration into applying iterative reconstruction techniques in the context of dynamic 3D data, in this case, perfusion data. It is necessary to note a few key differences between this work and the direct extension of the bFISTA discussed in previous chapters.

First, a TWIST sampling trajectory was used. Therefore, an extension of the VISTA sampling trajectory to three spatial dimensions must be adequately explored. The TWIST sampling trajectory is a sampling scheme that does not necessarily incorporate the optimal degree of incoherence for a compressed sensing based reconstruction. This can lead to additional residual artifacts in the final reconstructed images. An extension of the VISTA sampling trajectory to three spatial dimensions would provide the optimal amount of incoherence for a compressed sensing reconstruction, while also meeting the necessary requirements for a pMRI-based reconstruction.

Second, a POCS optimization without any $\ell_1$ regularization was used to solve the SPIRiT reconstruction problem with a SPIRiT kernel in three spatial dimensions. To the best of our knowledge, reconstruction using a three-dimensional SPIRiT kernel has not been thoroughly explored. Given no memory limitations, the ideal scenario would be to explore a three-dimensional SPIRiT reconstruction using a nonlinear conjugate gradient approach with spatial- and temporal- regularization in a four-dimensional wavelet domain. However, this approach, at least computationally, will be initially extremely slow, and it remains a considerable challenge in terms of efficiently implementing this approach within the constraints of finite memory capacity. Another approach would be to utilize a SENSE-based iterative reconstruction, in this case the memory constraints, while still not insignificant, are reduced.
8.1.2 Potential Applications to other CMR Techniques

As discussed in the previous section, iterative reconstruction techniques can be applied to perfusion imaging both for 2D and 3D techniques. In the 2D scenario, the iterative reconstruction techniques discussed in this work (i.e. a VISTA sampling trajectory with a FISTA-based optimization approach and either a SENSE or SPIRiT reconstruction) can readily be applied to the 2D perfusion imaging scenario. This would potentially enable successful image recovery at higher undersampling ratios. The reduced acquisition time per slice could then be used for greater coverage of the heart during each heartbeat or for improving the spatial resolution of the acquisition. Three dimensional perfusion techniques (as discussed in the pilot study earlier in this chapter) could also benefit from this paradigm of signal recovery, providing potentially more benefits due to the use of k-space correlations in the third spatial dimension. However, a significant practical barrier exists in terms of the reconstruction time and memory requirements for what is essentially a four dimensional problem (three spatial dimensions plus one temporal dimension). A more efficient implementation within the Gadgetron framework may also be promising in overcoming these barriers from both a computational and practical perspective.

The direct extension of this body of work to an additional spatial dimension (i.e. three spatial plus one temporal dimension) also opens the door for potential benefits in the context of 3D cine imaging. Cine imaging in three spatial dimensions is a direct extension of the contributions in this work to 2D cine imaging. The primary barriers involve the practical extension of the code contributions in the Gadgetron framework towards utilizing an additional spatial dimension - in particular, the wavelet transform utilized. By virtue of this extension additional factors pertaining to memory
requirements in addition to computational efficiency must also be considered if the goal of practical implementation is to be achieved. However, should these barriers be overcome, the extension to 3D cine imaging could have significant impact in enabling whole heart cine imaging at higher spatial and temporal resolutions. The extension of these techniques could be applied both in the context of segmented breath-hold imaging (e.g. whole heart coverage in a single breath-hold), or perhaps to achieve real-time, free-breathing cine imaging of the whole heart.

Three dimensional extensions of this work may also be relevant in the context of dynamic 3D angiography, which seeks to image the vascular system consisting of the arterial and venous system. Successful image recovery at higher undersampling ratios using the techniques discussed here could enable shorter acquisition times, which can reduce susceptibility to bulk motion that can introduce artifacts in the final reconstructed images. Additionally, the use of these techniques could also enable acquisitions at higher spatial resolutions, potentially enabling imaging of smaller vessels within the vascular system. Provided the conditions of incoherence are still satisfied, combination of these reconstruction techniques with more efficient non-Cartesian trajectories (e.g. spiral trajectories) could allow for further improvements via either increased spatial resolution or reduced acquisition times. Combination with different magnetization preparation pulses could also open the way for promising application of compressed sensing techniques in this context.

Flow imaging is a particularly promising and also challenging dynamic application in which the techniques described in the preceding chapters could be applied. In the extension to the highest dimensional scenario, one could consider the measurement of the three-direction velocity on a pixel-wise basis in a three-dimensional volume over
time. In this case, the problem dimensionality increases substantially to seven dimensions (i.e. three velocity dimensions, three spatial dimensions, and one temporal dimension), and sparsity in all seven dimensions could potentially be exploited. The use of patch-wise regularization extensions may also be potentially promising. However, the primary issue then becomes a matter of implementation, as a seven-dimensional wavelet transform (should an attempt be made to take advantage of sparsity in all dimensions be made) is significantly expensive both in terms of computational power and also physical memory. A three-dimensional wavelet transform combined with different transforms (e.g. principal component analysis or the Karhunen-Loeve transform) applied in the other dimensions may provide a more suitable sparsity model than a higher dimensional wavelet transform. Therefore, incrementally expanding the dimensionality of the problem by taking advantage of sparsity in a small number of additional dimensions may yield reconstruction benefits while still allowing for a practical clinically relevant reconstruction given currently available hardware. For example, it may be greatly beneficial to take advantage of the sparsity in a single velocity dimension, thereby increasing the dimensionality of the problem by only one dimension.

Static late gadolinium enhancement (LGE) imaging techniques could also potentially benefit from the techniques discussed in this work, despite the particular focus on dynamic imaging. In LGE techniques, a specific magnetization preparation pulse is combined with image acquisition at a single cardiac phase acquired at a specific time point after the completion of the preparation pulse. The duration of the time between the end of the preparation pulse and image acquisition determines the contrast between viable and non-viable myocardium. This image acquisition is the acquisition
of a single frame with high spatial resolution being the most desirable. Typically, high spatial resolution can be achieved through the use of parallel imaging techniques (i.e. GRAPPA or SENSE) with a fully-sampled center k-space region. High spatial resolution is achieved at the cost of SNR, which is often gained back through averaging acquisitions over multiple cardiac cycles in the presence of a breath-hold. In this case, we are considering the acquisition of snapshot images of the same slice location and cardiac phase over multiple cardiac cycles in the presence of suspended respiratory motion. Therefore, although the final reconstructed result is a static image in two spatial dimensions, the actual acquired data exists in three dimensions, with the third dimension being the time dimension across multiple cardiac cycles. From this perspective, a compressed sensing reconstruction scheme could potentially be used in place of averaging, allowing higher spatial resolutions by enabling successful reconstruction from more greatly undersampled data. Combined with motion correction techniques, this approach could potentially also be applied to LGE imaging over multiple cardiac cycles without suspending breathing motion (i.e. free-breathing). It is entirely possible that utilizing a compressed sensing scheme could achieve better preservation of SNR levels, leading to increased spatial resolution through more efficient tradeoff with SNR.

Parameter mapping techniques could also potentially benefit directly from the sampling scheme and iteration reconstruction proposed in this work by treating the dimension of decay (e.g. the echo time in $T_2$ or $T_2^*$ mapping) as an additional dimension of the sparsity that plays the role of the time dimension. As the combination of the VISTA sampling scheme and a FISTA-based reconstruction using SENSE or SPIRiT can enable image recovery from higher undersampling ratios, this could be
used to shorten the acquisition time for each image used to estimate the decay time
constants. This could potentially allow for the acquisition of more time points along
the dimension of decay, enabling improved parametric curve fitting. Alternatively, the
shortened acquisition time could allow for the introduction of more complex magneti-
zation preparation pulses prior to image acquisition that could be used for additional
robustness against flow or motion artifacts.

8.2 A Portable Extension of MrRecon to Multiple Dimensional
Data

Extension of the techniques described in this work to higher dimensional problems
using other CMR techniques may require significant modification to the current imple-
mentation within the Gadgetron framework. The reconstruction code contained in the
MrRecon codebase is primarily suitable for reconstruction of dynamic two-dimensional
MRI data. Therefore, in additional to the theoretical framework necessary for en-
abling compressed sensing reconstruction in the context of other CMR applications
described in the previous section, a portable extension to multiple dimensions will be
necessary to enable practical implementations of reconstruction for these applications.
This may involve redesigning the framework to utilize a general multidimensional
input/output interface. It may also be more beneficial and efficient to depend upon
the toolboxes developed within the Gadgetron framework itself.

Of particular note is the need to develop and implement a flexible GPU-based
toolbox for implementing $M$-level $N$-dimensional wavelet transforms for any arbi-
trary wavelet filter length. This was the initial purpose of the incomplete toolbox
that exists in the location gadgets/MrRecon/toolboxes/mnWavelet. A particularly
challenging aspect of this extension is to adequately manage the memory constraints
while maintaining a fast computation speed. It is possible that multiple GPUs may be necessary, in which case it will become necessary to perform management of memory transfer between cards. Completion of this project will greatly speed implementation of multi-dimensional applications of bFISTA within the Gadgetron.

8.3 The Use of Mutual Information in Patch-based Regularization

The quality of final recovered images in the compressed sensing recovery problem depends on the signal model used. LASSO-based regularization models [134, 136] utilize the prior information that supposes that the entire image signal is sparse in some transform domain. Recently proposed composite regularizers [120] utilize prior information that supposes that the level of sparsity varies across the different sub-bands or sub-dictionaries within this sparse domain. In the signal model for these approaches, the variation in sparsity across sub-bands is taken into account; however, variation across the spatial and/or temporal dimensions (or other measurement dimensions) is not taken into account. Therefore, the use of patch-based adaptive regularization may potentially be beneficial in enabling improved image reconstruction from more highly undersampled data by further extending the sparsity model. Another additional extension may be the combination of patch-based adaptive regularization with mutual information based on the Kullback-Leibler divergence in order to achieve non-local shrinkage. While computationally intensive to compute, the mutual information between patches may be used as prior information based on the notion that similarly sparse patches in the measurement dimensions could be processed similarly to further improve reconstruction results. However, achieving a practical efficient implementation may be particularly daunting, as the computation of mutual information between
patches scales quadratically with the number of patches. Yet, in highly dynamic scenarios such as in exercise stress cine imaging, it is possible that this approach may yield significant improvements in image quality.

8.4 Final Concluding Remarks

In this body of work, we have demonstrated the application of signal processing techniques to improving the reconstruction of real-time, free-breathing cardiac cine MRI from a practical perspective. The general focus of this work has been with the goal of achieving a practical implementation of cine reconstruction that can be realized in the clinical environment. From a practical perspective, this focus includes the efficient reconstruction of cine images with minimal effect on diagnostic reconstruction quality. Therefore, the scope of this work from an engineering perspective involves the use of a suitable compressed sensing scheme in conjunction with an efficient optimization algorithm. This particular signal processing paradigm is then applied to specific reconstruction techniques for cine reconstruction and implemented on computer hardware that provides for efficient reconstruction times.

Admittedly, the scope of the contributions in this work has been limited to a specific CMR application using a specific implementation paradigm both from the theoretical and practical perspective. However, the potential utility of this work is expected to beneficially address a longstanding practical issue with promising compressed sensing techniques in that there exist significant barriers to practical implementation due to high computational requirements. We look forward to further exploration by the research community of the numerous directions beyond the topics discussed in this work, including the following: (1) the use of different optimization
algorithms that might offer more efficient computation or improved convergence, (2) the use of different sampling trajectories for more efficient k-space coverage, and (3) the use of a more efficient, or perhaps more flexible or extensible, frameworks for practical implementation. Additionally, the scope of the application of this work can be extended to CMR techniques beyond cine imaging as discussed in the previous sections of this chapter, and we look forward to seeing further practical, clinically relevant applications of compressed sensing image reconstruction in the domain of CMR techniques.
Bibliography


[20] Selker HP, Griffith JL, D’Agostino RB. A tool for judging coronary care unit admission appropriateness, valid for both real-time and retrospective use. A
time-insensitive predictive instrument (TIPI) for acute cardiac ischemia: a mul-


[46] Kim RJ, Wu E, Rafael A, Chen EL, Parker Ma, Simonetti O, Klocke FJ, Bonow RO, Judd RM. The use of contrast-enhanced magnetic resonance imaging to


[55] Plein S, Smith WH, Ridgway JP, Kassner A, Beacock DJ, Bloomer TN, Sivananthan MU. Qualitative and quantitative analysis of regional left ventricular wall

278


281


286