

Summer 8-15-2018

Multi-GPU Acceleration of Iterative X-ray CT Image Reconstruction

Ayan Mitra

Washington University in St. Louis

Follow this and additional works at: https://openscholarship.wustl.edu/eng_etds



Part of the [Electrical and Electronics Commons](#)

Recommended Citation

Mitra, Ayan, "Multi-GPU Acceleration of Iterative X-ray CT Image Reconstruction" (2018). *Engineering and Applied Science Theses & Dissertations*. 373.

https://openscholarship.wustl.edu/eng_etds/373

This Dissertation is brought to you for free and open access by the McKelvey School of Engineering at Washington University Open Scholarship. It has been accepted for inclusion in Engineering and Applied Science Theses & Dissertations by an authorized administrator of Washington University Open Scholarship. For more information, please contact digital@wumail.wustl.edu.

WASHINGTON UNIVERSITY IN ST. LOUIS
School of Engineering and Applied Science
Department of Electrical and Systems Engineering

Dissertation Examination Committee:
Dr. Joseph A. O'Sullivan, Chair
Dr. Mark A. Anastasio
Dr. Martin Arthur
Dr. David G. Politte
Dr. Jeffrey F. Williamson

Multi-GPU Acceleration of Iterative X-ray CT Image Reconstruction
by
Ayan Mitra

A dissertation presented to
The Graduate School
of Washington University in
partial fulfillment of the
requirements for the degree
of Doctor of Philosophy

August 2018
St. Louis, Missouri

© 2018, Ayan Mitra

Table of Contents

List of Figures.....	v
List of Tables	ix
List of Algorithms	x
Acknowledgments	xi
Abstract.....	xiv
Chapter 1: Introduction	1
1.1 Motivation.....	1
1.2 Contributions.....	4
1.3 Outline.....	6
Chapter 2: Background.....	7
2.1 Image Reconstruction Overview.....	7
2.1.1 Reconstruction from Line Integral Data Model	7
2.1.2 Reconstruction from Statistical Data Model	12
2.1.3 Comparison of Analytical and Statistical Iterative Methods.....	15
2.2 System Modeling	17
2.2.1 Branchless Distance-driven Projection	17
2.2.2 Branchless Distance-driven Backprojection	18
2.3 Graphics Processing Unit Architecture	19
2.4 Acceleration of Statistical Iterative-reconstruction Algorithms.....	20
2.4.1 Algorithmic Speedup	21
2.4.2 Hardware Speedup	22
Chapter 3: Multislice Statistical Iterative Helical CT Reconstruction Using GPU.....	24
3.1 Theory	24
3.1.1 Statistical Data Model.....	24
3.1.2 Image Reconstruction Formulation.....	25
3.2 Branchless Distance-driven Projectors	34
3.2.1 Modification of Detector Edge Projections.....	35
3.2.2 Pre-accumulation for Forward Projection	36

3.2.3	Pre-accumulation for Backprojection.....	41
3.2.4	Modified Overlap Computation	42
3.3	CPU Multithreaded Parallelization Scheme for Branchless Distance-driven Projectors	43
3.3.1	Symmetry	44
3.3.2	Multi-threaded Implementation for Forward Projection	46
3.3.3	Multithreaded Implementation for Backprojection	47
3.4	GPU Implementation of Branchless Distance-driven Projectors	49
3.5	Multi-GPU Implementation of Branchless Distance-driven Projectors.....	52
3.6	Experiments	55
3.7	Results	57
3.7.1	Ordered Subsets	57
3.7.2	Phantom	58
3.7.3	Clinical Datasets.....	63
3.7.4	Timing Performance.....	66
3.8	Discussion	71
Chapter 4: Multislice Analytical Helical CT Reconstruction Using GPU		74
4.1	Theory	75
4.2	FDK Reconstruction	77
4.2.1	Data Preprocessing Operations	77
4.2.2	Redundancy Weights	79
4.2.3	Cone-parallel Backprojection.....	80
4.3	GPU Implementation of FDK Backprojection.....	84
4.4	Results	84
4.4.1	Phantom	85
4.4.2	Clinical Datasets.....	87
4.4.3	Timing Performance.....	89
4.5	Discussion	92
Chapter 5: Acceleration of Iterative-reconstruction Algorithms Using Adaptive Auxiliary Variable.....		94
5.1	Theory	95
5.2	Experiments	98
5.3	Results	99

5.3.1	Phantom	99
5.3.2	Clinical Datasets.....	102
5.3.3	Convergence Rate	104
5.4	Discussion	105
Chapter 6: Dual-energy AM Reconstruction Algorithm Using GPU		106
6.1	Dual-energy AM Algorithm.....	107
6.2	Adaptive Auxiliary Variable for Dual Energy	114
6.3	GPU Implementation	118
6.4	Experiments and Reconstructions	119
6.5	Conclusion	126
Chapter 7: Deep Convolutional Neural Network Based Denoising		127
7.1	Theory	128
7.1.1	Deep Neural Networks for X-ray Image Denoising.....	128
7.1.2	Residual Learning and Batch Normalization	129
7.1.3	Proposed Network Model	131
7.2	Experiments	133
7.2.1	CT Noise Model.....	133
7.2.2	Training and Testing Data.....	135
7.2.3	Compared Methods	136
7.3	Results.....	137
7.4	Conclusion	144
Chapter 8: Conclusions and Future Work		146
Appendix A: Derivation of the Penalized AM Algorithm		150
Appendix B: Derivation of Decoupled Dual-energy Surrogate Function.....		153
References		158
Curriculum Vitae		168

List of Figures

Fig. 2.1 Broad classification of X-ray CT reconstruction algorithms.....	7
Fig. 2.2 The geometry of parallel lines and projections used to define the Radon transform.	10
Fig. 2.3 Linear filtered backprojection algorithm for X-ray CT.....	12
Fig. 2.4 CPU vs. GPU architecture	20
Fig. 3.1 The multislice helical geometry used in this dissertation.....	34
Fig. 3.2 (a) Schematic representation of De Man and Basu's [50] 2-D distance-driven method. (b) Schematic representation of our 2-D distance-driven method. (c) Schematic representation of De Man and Basu's [50] 3-D distance-driven method. (d) Schematic representation of our 3-D distance-driven method.	35
Fig. 3.3 (a) Schematic diagram of detector projection on image pixel slab which signifies the area of overlap. (b) Our approach to the calculation of overlap between detector edge projections and image pixel slabs.	43
Fig. 3.4 (a) Axial view of the quarter-rotation symmetry found in helical CT. When an integer number of slices is chosen per quarter rotation of the gantry, the geometry calculations need only be done for just the first quarter rotation of the scan (indicated by a dark solid box). (b) Transverse view of the quarter-rotation symmetry. The projection calculations for each of the slabs shown is identical in the in-plane direction and offset by multiples of Nq in the z direction. An arrow has been drawn for each slab that indicates the direction of in-plane accumulation. The z accumulation is always in the direction of the positive z axis. Similar approach to quarter-rotation symmetry was explored by D. Keesing [65]..	45
Fig. 3.5 Summing of private partial accumulation images on processors 0 and 1 into full-sized accumulation image. At each stage, the shaded block of slices from each processor is simultaneously summed into the full-sized accumulation image.	48
Fig. 3.6 (a) Schematic representation of Multi-GPU implementation of branchless DD projection. (b) Schematic representation of Multi-GPU implementation of branchless DD backprojection.....	53
Fig. 3.7 Schematic representation of iterative algorithm execution between CPU and GPU devices.....	54
Fig. 3.8 Single iteration time for different OS using 3 TITAN X GPUs in parallel.....	58
Fig. 3.9 NCAT phantom reconstruction with voxel size = $1.0 \times 1.0 \times 1.0$ mm. Scan parameters: pitch 1.0, 16×1.5 mm collimation, display window width = 0.01759 mm^{-1} , center = 0.008795 mm^{-1} . (a), (b) Axial slices of the actual phantom. (c), (d) Axial slices of the FDK reconstruction of the phantom with added sinogram noise. (e) and (f) Axial slices of the phantom reconstructed with 10 iterations with 145 ordered subsets and with added noise in sinogram domain.	59
Fig. 3.10 Horizontal profile for different reconstruction images along different lines shown in Figs. 3.9 (a), (c), and (e).....	60

Fig. 3.11 Horizontal profile for different reconstruction images along different lines shown in Figs. 3.9 (b), (d), and (f).....	60
Fig. 3.12 RMSE vs total reconstruction time for different OS configuration using 3 TITAN X GPUs.....	62
Fig. 3.13 PAE in percentage vs total reconstruction time for different OS configuration using 3 TITAN X GPUs	62
Fig. 3.14 SNR in dB vs total reconstruction time for different OS configuration using 3 TITAN X GPUs	63
Fig. 3.15 CNR vs total reconstruction time for different OS configuration using 3 TITAN X GPUs	63
Fig. 3.16 Regularized AM reconstruction using 10 iterations of 145 ordered subsets. Voxel size = $1.0 \times 1.0 \times 1.0$ mm. Scan parameters: 180 mAs, pitch 1.0, 16×1.5 mm collimation. (a) Axial slice of lung with display window width = 0.03 mm^{-1} , center = 0.015 mm^{-1} . (b) Axial slice of abdomen with display window width = 0.007 mm^{-1} , center = 0.021 mm^{-1} . (c) and (d) are coronal views and (e) and (f) are sagittal views with display window width = 0.007 mm^{-1} , center = 0.021 mm^{-1}	64
Fig. 3.17 Regularized AM reconstruction of lung and abdominal slices using 3 TITAN X GPUs. Voxel size = $1.0 \times 1.0 \times 1.0$ mm. Scan parameters: 180 mAs, pitch 1.0, 16×1.5 mm collimation. Axial slice of the lung with display window width = 0.03 mm^{-1} , center = 0.015 mm^{-1} , reconstructed with (a) FDK and (c) 10 iterations of AM using 145 OS. Axial slice of the abdomen with display window width = 0.007 mm^{-1} , center = 0.021 mm^{-1} , reconstructed with (b) FDK and (d) 10 iterations of AM using 145 OS. (e) and (f) are difference images between FDK and 10 AM iteration using 145 OS corresponding to lung and abdomen slices respectively.	65
Fig. 3.18 Plot of I-divergence vs computation time for different ordered subset configurations by using 3 GPUs in parallel.	66
Fig. 3.19 Acceleration of our multi-GPU implementation for complete clinically-sized data.....	67
Fig. 3.20 (a) Forward projection computational times and (b) overall speedup for a different number of pixels along X/Y direction using different hardware configurations.	68
Fig. 3.21 (a) Backprojection computational times and (b) overall speedup for a different number of pixels along X/Y direction using different hardware configurations.	68
Fig. 3.22 (a) Forward projection computational times and (b) overall speedup for different number of image slices using different hardware configurations	69
Fig. 3.23 (a) Backprojection computational times and (b) overall speedup for different number of image slices using different hardware configurations	69
Fig. 4.1 Schematic diagram of the cone beam to parallel fan beam rebinning scheme described in [72]: (a) the native CB geometry; (b) the cone-parallel geometry.	77
Fig. 4.2 Parallel-beam and fan beam geometry.	81
Fig. 4.3 v scaling based on point projection.	82

Fig. 4.4 NCAT phantom reconstruction with voxel size = $1.0 \times 1.0 \times 1.0$ mm. Scan parameters: pitch 1.0, 16×1.5 mm collimation, display window width = 0.01759 mm^{-1} , center = 0.008795 mm^{-1} . (a), (b) Axial slices of the actual phantom. (c), (d) Axial slices of the FDK reconstruction of the phantom.	86
Fig. 4.5 Horizontal profile along the orange line through ideal phantom and noisy FDK reconstruction image shown in Figs. 4.4 (a) and (c).....	86
Fig. 4.6 Horizontal profile along the orange line through ideal phantom and noisy FDK reconstruction image shown in Figs. 4.4 (b) and (d)	87
Fig. 4.7 Clinical abdominal reconstruction using FDK algorithm. Voxel size = $1.0 \times 1.0 \times 1.0$ mm. Scan parameters: 180 mAs, pitch 1.0, 16×1.5 mm collimation. (a) Axial slice of lung with display window width = 0.03 mm^{-1} , center = 0.015 mm^{-1} . (b) Axial slice of abdomen with display window width = 0.007 mm^{-1} , center = 0.021 mm^{-1} . (c) and (d) are coronal views and (e) and (f) are sagittal views with display window width = 0.007 mm^{-1} , center = 0.021 mm^{-1}	88
Fig. 4.8 (a) Backprojection time and (b) 8 core CPU speedup factor for a different number of pixels along X/Y direction.	89
Fig. 4.9 Speedup factor for parallel fan-beam backprojection operation using a different number of GPUs in parallel compared to baseline CPU implementation.....	89
Fig 4.10 (a) Total time and (b) 8 core CPU speedup factor for a different number of pixels along X/Y direction.	90
Fig. 4.11 Speedup factor for total computational time using a different number of GPUs in parallel compared to baseline CPU implementation.....	90
Fig. 4.12 (a) Total time and (b) 8 core CPU speedup factor for a different number of image slices. (c) Speedup factor for total computational time using a different number of GPUs in parallel compared to baseline CPU implementation.....	91
Fig. 5.1 (a) and (c) Linear attenuation coefficient map reconstructed with FDK algorithm for NCAT data in units of mm^{-1} . (b), (d) The values of the auxiliary variable for the corresponding image slice.....	100
Fig. 5.2 Profile along the red dotted line depicted in Fig. 5.1 (c) for images reconstructed using 100 iterations of 5 OS of AM algorithm without (blue) and with (red) adaptive surrogate function.	100
Fig. 5.3 (a), (b) PAE in % vs iteration number for the NCAT phantom with. (c), (d) RMSE vs iteration number for the NCAT phantom.....	101
Fig. 5.4 (a) and (b) SNR vs iteration number for the NCAT phantom. (c) and (d) CNR for the structure in a green dotted box in Fig. 5.1 (c) vs iteration number for the NCAT phantom.	102
Fig. 5.5 (a) and (b) Linear attenuation coefficient map reconstructed with FDK algorithm for real data obtained from Siemens Sensation 16 scanner in units of mm^{-1} . (c) And (d) are the values of the auxiliary variable for the corresponding image slices in units of mm^{-1}	103

Fig. 5.6 Objective function values vs iteration number for Siemens Sensation 16 scanner reconstructed images (a) without ordered subset implementation and with (blue) and without (red) adaptive auxiliary variable, and (b) with 5 ordered subset implementations and with (blue) and without (red) adaptive auxiliary variable.....	104
Fig. 6.1 The phantom linear attenuation coefficient image in mm^{-1} at (a) 53 keV and at (b) 70 keV with four inserts (from the top, a clockwise direction) PMMA, ethanol, methyl ethyl ketone (MEK), and calcium chloride.....	120
Fig. 6.2 Incident spectra.....	123
Fig. 6.3 Attenuation coefficient of the component materials.....	123
Fig. 6.4 Initial (a) $c_1(x)$ and (b) $c_2(x)$ component images reconstructed using FDK algorithm. (c) $c_1(x)$ and (d) $c_2(x)$ component images reconstructed using 400 iterations of 5 OS DE-AM algorithm.....	124
Fig. 6.5 Plot of RMSE between truth image and reconstructed image using 100 iterations of 29 OS DE-AM algorithm vs different energy bins.....	125
Fig. 6.6 Total objective function values vs iteration number for 5 OS implementations of the DE-AM algorithm.....	125
Fig. 7.1 Schematic diagram for batch normalization.....	131
Fig. 7.2 The architecture of our proposed deep CNN.....	132
Fig. 7.3 Noise simulation flowchart.....	134
Fig. 7.4 (a) Clinical abdominal image collected from Siemens Somatom Definition AS scanner. Voxel size $=0.576 \times 0.576 \times 1.0$ mm. Scan parameters: 180 mAs, pitch 0.75, 19×0.6 mm collimation. The abdominal display window is -160 HU to 240 HU. (b) Low-dose noisy image. (c) Denoised image BM3D algorithm. (d) Denoised image with WNNM algorithm. (e) Denoised image with our proposed Deep CNN based method.....	139
Fig. 7.5 (a) Clinical abdominal image collected from Siemens Somatom Definition AS scanner. Voxel size $=0.576 \times 0.576 \times 1.0$ mm. Scan parameters: 180 mAs, pitch 0.75, 19×0.6 mm collimation. The abdominal display window is -160 HU to 240 HU. (b) Low-dose noisy image. (c) Denoised image BM3D algorithm. (d) Denoised image with WNNM algorithm. (e) Denoised image with our proposed Deep CNN based method.....	140
Fig. 7.6 (a) Clinical abdominal image collected from Siemens Somatom Definition AS scanner. Voxel size $=0.576 \times 0.576 \times 1.0$ mm. Scan parameters: 180 mAs, pitch 0.75, 19×0.6 mm collimation. The abdominal display window is -160 HU to 240 HU. (b) Low-dose noisy image. (c) Denoised image BM3D algorithm. (d) Denoised image with WNNM algorithm. (e) Denoised image with our proposed Deep CNN based method.....	141
Fig. 7.7 Intensity profile along the lines in Fig. 7.4.....	142
Fig. 7.8 Intensity profile along the lines in Fig. 7.5.....	143
Fig. 7.9 Intensity profile along the lines in Fig. 7.6.....	143

List of Tables

Table 3.1 Hardware specification of TITAN X	57
Table 3.2 Parameters of measured data and image	58
Table 3.3 Reconstruction times using clinically-sized data and no OS for different CPU and GPU hardware architectures.	67
Table 4.1 Reconstruction times using clinically-sized data and no OS for different CPU and GPU hardware architectures.	87
Table 5.1 Parameters of measured data and image	99
Table 6.1 Execution times by using different CPU and GPU configurations for a single iteration of DE-AM algorithm.....	120
Table 6.2 Variance of different materials in different component images	124
Table 7.1 The PSNR(dB), SSIM, and RMSE values for the 3 image slices shown in the figures previously.....	142

List of Algorithms

Algorithm 3.1 AM algorithm.....	31
Algorithm 3.2 Regularized AM algorithm.....	33
Algorithm 3.3 Branchless distance-driven forward projection.....	47
Algorithm 3.4 Branchless distance-driven backprojection.....	49
Algorithm 3.5 GPU implementation of branchless distance-driven forward projection.....	50
Algorithm 3.6 GPU implementation of branchless distance-driven backprojection	51
Algorithm 4.1 GPU implementation of FDK backprojection.....	84
Algorithm 5.1 Regularized OS-AM algorithm with adaptive auxiliary variable	98
Algorithm 6.1 Regularized DE-AM algorithm.....	113
Algorithm 6.2 Regularized DE-AM algorithm with ordered subsets	114
Algorithm 6.3 Regularized OS-DE-AM algorithm with adaptive auxiliary variable.....	117
Algorithm 6.4 Multi-GPU based computation of incident photon intensity	118

Acknowledgments

It has been an honor and privilege to work with my advisor Dr. Joseph A. O'Sullivan over these past several years. I'm grateful for the trust that he bestowed on me to do much of my work independently. I don't think I'd be here without his patience, support, motivation, and guidance along the way. I was fortunate enough to attend his Biomedical Imaging Technology course at Washington University, which formed a significant base of the theoretical knowledge needed for the completion of this research work. As I reflect on my rich experience as a doctoral student, I don't think I could have asked for a better advisor.

I would also like to sincerely express my deepest gratitude to Dr. David G. Politte, who amid all his commitments, found time to help me with my research. He was extremely gracious to meticulously proofread all my publications and this dissertation. I don't think I would have completed this dissertation without his extensive edits and helpful comments. I would like to thank my committee members Dr. Jeffrey F. Williamson, Dr. Martin Arthur and Dr. Mark A. Anastasio for taking time out of their busy schedule to see me through the process.

I also had the opportunity to collaborate with Dr. Bruce R. Whiting from the Department of Radiology at University of Pittsburgh. His insights, comments and encouragement has enriched my research experience. I am fortunate to have worked with Dr. Craig Abbey from UC Santa Barbara, and Dr. Steven Don from Washington university Medical School and to have taken part in many interesting discussions with them. I would also like acknowledge my current and past lab mates, namely Dr. Soysal Degirmenci, Jingwei Liu, Shuangyue Zhang, Linyun He and Tao Ge. I

also want to thank my extremely supportive friends in St. Louis, USA namely, Dr. Anirban Nandi, Tathagata Banerjee, Prateek Gundannavar Vijay, Ahana Gangopadhyay and Oindrila Chatterjee.

My parents have provided me with unlimited support even through tough times. They would have wanted to see me more frequently, but understood the commitments involving graduate research in a different country. I would like to acknowledge and extend my heartfelt gratitude to my wife Kate Lee Workman who has been there for me every step of the way.

This work was supported in part by NIH grants R01 CA149305 (J. Williamson, PI), R01 561 EB019135 (B. Whiting, PI) and R01 CA212638-02 (J.A. O’Sullivan, PI).

Ayan Mitra

Washington University in St. Louis

August 2018

Dedicated to my parents.

ABSTRACT OF THE DISSERTATION

Multi-GPU Acceleration of Iterative X-ray CT Image Reconstruction

by

Ayan Mitra

Doctor of Philosophy in Electrical Engineering

Washington University in St. Louis, 2018

Professor Joseph A. O’Sullivan, Chair

X-ray computed tomography is a widely used medical imaging modality for screening and diagnosing diseases and for image-guided radiation therapy treatment planning. Statistical iterative reconstruction (SIR) algorithms have the potential to significantly reduce image artifacts by minimizing a cost function that models the physics and statistics of the data acquisition process in X-ray CT. SIR algorithms have superior performance compared to traditional analytical reconstructions for a wide range of applications including nonstandard geometries arising from irregular sampling, limited angular range, missing data, and low-dose CT. The main hurdle for the widespread adoption of SIR algorithms in multislice X-ray CT reconstruction problems is their slow convergence rate and associated computational time.

We seek to design and develop fast parallel SIR algorithms for clinical X-ray CT scanners. Each of the following approaches is implemented on real clinical helical CT data acquired from a Siemens Sensation 16 scanner and compared to the straightforward implementation of the Alternating Minimization (AM) algorithm of O’Sullivan and Benac [1]. We parallelize the computationally expensive projection and backprojection operations by exploiting the massively

parallel hardware architecture of 3 NVIDIA TITAN X Graphics Processing Unit (GPU) devices with CUDA programming tools and achieve an average speedup of 72X over a straightforward CPU implementation. We implement a multi-GPU based voxel-driven multislice analytical reconstruction algorithm called Feldkamp-Davis-Kress (FDK) [2] and achieve an average overall speedup of 1382X over the baseline CPU implementation by using 3 TITAN X GPUs. Moreover, we propose a novel adaptive surrogate-function based optimization scheme for the AM algorithm, resulting in more aggressive update steps in every iteration. On average, we double the convergence rate of our baseline AM algorithm and also improve image quality by using the adaptive surrogate function. We extend the multi-GPU and adaptive surrogate-function based acceleration techniques to dual-energy reconstruction problems as well. Furthermore, we design and develop a GPU-based deep Convolutional Neural Network (CNN) to denoise simulated low-dose X-ray CT images. Our experiments show significant improvements in the image quality with our proposed deep CNN-based algorithm against some widely used denoising techniques including Block Matching 3-D (BM3D) and Weighted Nuclear Norm Minimization (WNNM). Overall, we have developed novel fast, parallel, computationally efficient methods to perform multislice statistical reconstruction and image-based denoising on clinically-sized datasets.

Chapter 1: Introduction

1.1 Motivation

X-ray computed tomography (CT) is a popular noninvasive imaging modality mostly used for the analysis of specific internal anatomical structures and to provide more accurate information regarding those internal regions of interest. X-ray CT is widely used in the medical imaging community to help radiation oncologists devise better treatment plans and physicians detect and diagnose diseases. With the adoption of modern X-ray CT scanners in several areas from medical imaging to security applications, there is a growing challenge to analyze all this new information in a relevant timeframe. In a world where data-generation rates are accelerating faster than modern computing capabilities, and where Moore's law has been stagnant for the last decade, simultaneous adoption of General Purpose computing on Graphics Processing Units (GPGPU), and mathematical optimizations are the industry-wide consensus for bridging the gap between them.

There has been a tremendous advancement in the last few decades in the capabilities of massively parallel graphics hardware. A CPU consists of a few cores with large caches, which are highly optimized for complex sequential operations while GPUs consist of thousands of smaller computational cores designed for handling massively parallel tasks simultaneously and more efficiently. CPU cores are mostly optimized for single-threaded operations where most of the transistor budget is dedicated towards pipelining instructions, and out-of-order execution while leaving fewer resources for the integer and floating-point execution units. GPUs, on the other hand,

have a large portion of the transistor budget dedicated to optimizing the floating-point throughput, rather than generating complex instruction-level parallelism [3]. Modern GPUs rely on large amounts of data transfer bandwidth, device memory, fast read-only texture and shared memory, and thousands of high-performance computational cores clocked at 1.5 GHz to yield massive advantage in computational cost over CPUs. Computationally intensive algorithms like SIR algorithms benefit tremendously in terms of computational time by offloading their most time-consuming parts onto GPU devices.

MBIR algorithms are typically iterative where the next image estimate is computed based on the current image estimate and an error measure between measured data and predicted data from the current image [4]. These algorithms can incorporate the statistics of the measured data, and detector response model, which in turn reduces noise and artifacts in images reconstructed from low-dose X-ray CT measurements [5-9]. Two important components of these algorithms are forward projection, where a reconstructed image is mapped onto the measured data space and backprojection where measured data is mapped onto the image domain. Due to the iterative nature of these algorithms and the high computational burden associated with the implementation of projection and backprojection operations on large data and image volumes, MBIR algorithms are not extensively used in clinical settings.

In the published literature, there are few papers that discuss parallelization strategies for helical CT statistical reconstruction. Much more work has been published on other imaging modalities, for example, in nuclear medicine [10-13] and circular-orbit cone-beam CT (CBCT) [14-17]. In

contrast to helical CT, however, implementations for nuclear medicine and circular orbit CBCT do not need to account for the movement of the scanned object along the z-direction of the scanner during data acquisition. One paper that does address the helical geometry describes a fast analytical backprojection algorithm based on helical symmetry and image rotation [18].

GPUs, therefore, have the potential to facilitate the adoption of complex MBIR algorithms, which can lead to improved images in terms of noise and artifact reduction, improvement of spatial and temporal resolutions [7-9, 19]. They are by far the least costly option for parallel computing, and they can provide large speedups over single-CPU implementations due to their specialized ability to handle arithmetic operations efficiently [7, 20-22]. GPU technology has come a long way, from its invention in the late 1980s to the latest release of GeForce GTX TITAN X GPUs, consisting of 8 billion transistors on a single chip. Modern GPU technologies with their high memory bandwidth and peak arithmetic performance are rapidly outpacing their CPU counterparts [23, 24].

Over the years, several groups have accelerated their iterative-reconstruction algorithm implementations using GPUs. Andreyev et. al [25] have accelerated their blob-based iterative reconstruction using a Tesla GPU. X. Jia et. al [9] implemented a low-dose cone-beam CT reconstruction with total variation regularization on an NVIDIA Tesla C1060 GPU. McGaffin et. al [26] proposed a multi-GPU based fast converging stochastic group ascent algorithm to perform dual maximization and implemented their algorithm on NVIDIA Tesla C2050 GPUs. Meng Wu et. al [27] accelerated separable footprint based projection and backprojection algorithms using NVIDIA Tesla C2050 GPUs. Quivira et. al [28] developed an iterative 3-D reconstruction

algorithm for sparse X-ray CT data on TITAN X GPUs. Due to their inherent parallel architecture, GPUs can provide quite significant performance improvement for algorithms with highly pipelined structure. Current GPUs also provide very high global memory storage, which is ideal for fitting the whole data volume and image array in the GPU itself during kernel execution, in turn eliminating the high latency penalty for accessing external memory. Due to all these advantages, it is quite logical to use GPUs to improve the speed of image reconstruction.

The second line of research for the reduction of the computational time of MBIR problems involve the design of efficient algorithms which amenable to parallelization [29-32]. The optimization framework explored in this work uses a popular linear reconstruction method, Feldkamp-Davis-Kress (FDK), to predict an adaptive and aggressive step size. In mathematical optimization, the optimality of a variable in a certain optimization space is determined by minimizing an objective function or by maximizing the negative of the objective function. A new method named *adaptive surrogate function* is investigated in this dissertation for accelerating the convergence rate of the AM algorithm and is evaluated using a phantom and real clinical data obtained from a Siemens Sensation 16 scanner.

1.2 Contributions

The contributions of the research presented in this dissertation are given below.

- We present a fast-parallel multi-GPU based implementation of branchless distance-driven projectors for helical scanner geometry.

- We propose novel ways to compute the pre-integration part in branchless distance-driven projection and backprojection computation, which eliminates the need for thread synchronization in GPU architecture.
- We present some novel ways to calculate the interpolation step of the branchless distance-driven projection and backprojection operator by directly projecting the detector array to image voxels, which makes our implementation more amenable to GPU thread-based parallelization.
- We derive a precise load sharing mechanism between multiple GPU devices to reduce the downtime of each device.
- We propose a novel adaptive step-size based acceleration technique for our iterative-reconstruction problem which doubles the rate of convergence for both the mono-energy and dual-energy cases.
- We develop novel schemes to accelerate the computational performance of the Feldkamp-Davis-Kress (FDK) reconstruction algorithm using multi GPUs in parallel.
- We implement and validate the above-mentioned multi-GPU based algorithmic acceleration steps on real clinical CT data and computer-generated phantom data.
- We also design and implement a deep Convolutional Neural Network based X-ray CT denoising system and validate the image quality performance of the proposed system on the real clinical dataset.

1.3 Outline

The general outline of this dissertation is as follows: In Chapter 2, we discuss the basic reconstruction problem and our motivation for shifting towards algorithmic and parallel hardware-based speedup. Chapter 3 contains a detailed description and derivation of our parallel multi-GPU based reconstruction algorithm for the mono-energetic model. Chapter 4 presents a multi-GPU based implementation of a popular analytical reconstruction algorithm known as FDK for clinical helical datasets. Next, in Chapter 5, we design a novel adaptive surrogate function and showcase the acceleration of the convergence rate on a multislice clinically-sized mono-energetic dataset. Chapter 6 contains a derivation of a multi-GPU based implementation of a dual-energy reconstruction algorithm and the corresponding adaptive surrogate-function based acceleration technique. In Chapter 7, we propose the deep CNN based X-ray CT image denoising technique and evaluate its performance.

Chapter 2: Background

2.1 Image Reconstruction Overview

Reconstruction algorithms for X-ray CT are broadly classified into the following categories depicted in Fig 2.1.

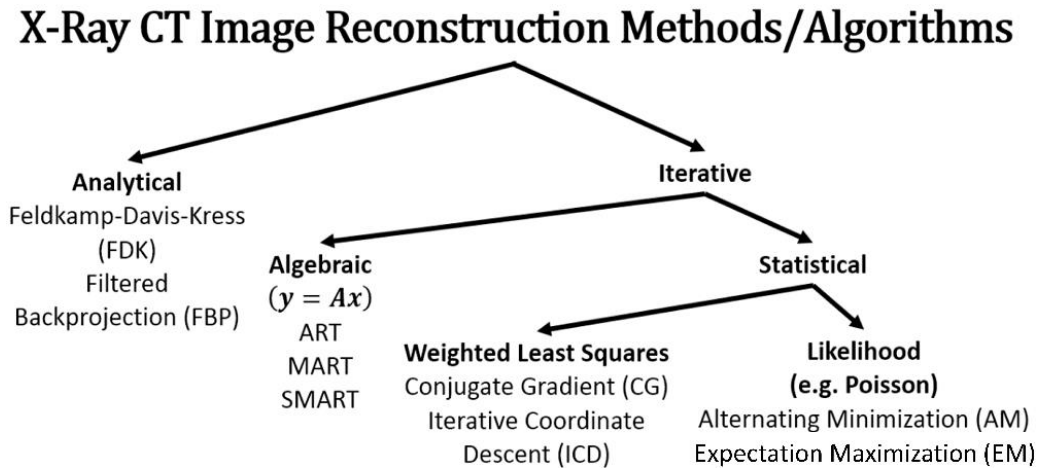


Fig. 2.1 Broad classification of X-ray CT reconstruction algorithms

Analytical algorithms are based on the deterministic line-integral model for measured data while statistical data-driven approaches are based on the arbitrarily accurate model that also accounts for the probability distribution of the measured data.

2.1.1 Reconstruction from Line Integral Data Model

In an X-ray CT system, the X-ray tube generates X-rays that propagate through the object we are trying to image and get attenuated as they travel through its cross-section. The attenuated exit beam

is then detected by the detectors along a straight-line path between the X-ray source and detector.

The detected intensity at any detector can be represented as

$$I_d(y) \triangleq \int_{E_{min}}^{E_{max}} S_0(E) E \exp\left(-\int \mu(s, E) ds\right) dE, \quad (2.1)$$

where $S_0(E)$ is the spectrum of the X-ray source at energy E , y is the source-detector pair, $\mu(s, E)$ is the energy-dependent linear attenuation coefficient along the line between source and detector, and E_{max} and E_{min} are the minimum and maximum energies, respectively, of the range over which the detectors are sensitive. The integration over energy in equation (2.1) complicates the derivation and implementation of algorithms that are based on this model. In order to overcome this issue, we use effective energy, \bar{E} , which is defined as the same measured intensity from a monoenergetic source as is measured using a polyenergetic source. However, this approximation can lead to beam-hardening. We can represent the detected intensity using effective energy as

$$I_d(y) \triangleq I_0(y) \exp\left(-\int \mu(s, \bar{E}) ds\right) \quad (2.2)$$

Given the measurement, $I_d(y)$, we can represent the basic projection measurement, $g_d(y)$, as

$$g_d(y) = -\log\left(\frac{I_d(y)}{I_0(y)}\right) \quad (2.3)$$

$$= \int \mu(s, \bar{E}) ds \quad (2.4)$$

So, we can conclude that the basic CT scanner measurement is actually a line integral of the linear attenuation coefficient $\mu(s, \bar{E})$ at the effective energy of the scanner. However, this approximation can lead to significant image reconstruction errors due to beam hardening [33]. We call this line integral through the object along the path of a collimated X-ray beam the forward projection model. For analytical methods, the forward projection algorithm is derived in continuous space and then

subsequently discretized for practical implementation. The line integral of a 2-D function $f(x, y)$ is given by

$$g(t, \theta) = \int_{-\infty}^{\infty} f(x(s), y(s)) ds \quad (2.5)$$

where for any point s along the line between source and detector,

$$x(s) = t \cos \theta - s \sin \theta, \quad (2.6)$$

$$y(s) = t \sin \theta + s \cos \theta. \quad (2.7)$$

We can alternatively express equation (2.5) as

$$g(t, \theta) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x, y) \delta(x \cos \theta + y \sin \theta - t) dx dy. \quad (2.8)$$

Equation (2.8) is basically the integration of function $f(x, y)$ along the line; hence it is a line integral. $g(t, \theta)$ is called the 2-D Radon transform of $f(x, y)$. The following derivation is based on Kak and Slaney [34]. Since our projection corresponds to a collection of parallel line integrals, they are called parallel ray projections as shown in Fig 2.2. The view angle is θ and the normal vector normal to the direction of projection is denoted by $\hat{n}(\theta)$.

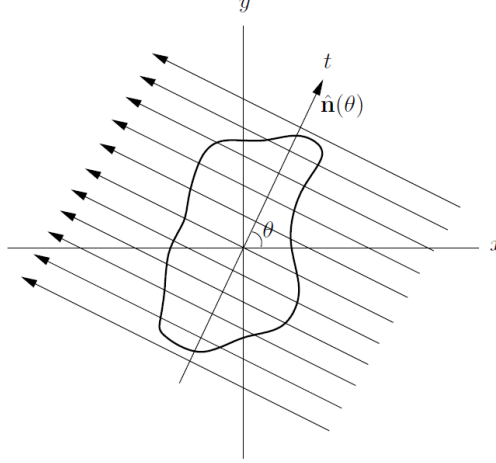


Fig. 2.2 The geometry of parallel lines and projections used to define the Radon transform.

For a fixed θ , $g(t, \theta)$ is called the projection at angle θ for all t . Using the projection slice theorem [35], we can develop the relationship between the 1-D Fourier transform of the projection and the 2-D Fourier transform of the object which is crucial to analytical reconstruction. The relationship is:

$$G(\omega, \theta) = \mathcal{F}_{1D}\{g(t, \theta)\} = \int_{-\infty}^{\infty} g(t, \theta) e^{-j2\pi\omega t} dt \quad (2.9)$$

$$= \iiint_{-\infty}^{\infty} f(x, y) \delta(x \cos \theta + y \sin \theta - t) e^{-j2\pi\omega t} dx dy dt \quad (2.10)$$

$$= \iint_{-\infty}^{\infty} f(x, y) \int_{-\infty}^{\infty} \delta(x \cos \theta + y \sin \theta - t) e^{-j2\pi\omega t} dt dx dy \quad (2.11)$$

$$= \iint_{-\infty}^{\infty} f(x, y) e^{-j2\pi\omega(x \cos \theta + y \sin \theta)} dx dy \quad (2.12)$$

$$= F(\omega \cos \theta, \omega \sin \theta), \quad (2.13)$$

where $j \triangleq \sqrt{-1}$. Equation (2.13), denoted by $F(\omega \cos \theta, \omega \sin \theta)$ is the Fourier transform of projection $g(t, \theta)$ at angle θ and is equal to the 2-D Fourier transform of $f(x, y)$ along the $\hat{n}(\theta)$ direction.

The inverse Fourier transform of $F(\omega \cos \theta, \omega \sin \theta)$ can be expressed in polar coordinates:

$$f(x, y) = \int_0^{2\pi} \int_0^\infty F(\omega \cos \theta, \omega \sin \theta) e^{j2\pi\omega(x \cos \theta + y \sin \theta)} \omega d\omega d\theta. \quad (2.14)$$

Using the projection-slice theorem from equation (2.13) we have

$$f(x, y) = \int_0^{2\pi} \int_0^\infty G(\omega, \theta) e^{j2\pi\omega(x \cos \theta + y \sin \theta)} \omega d\omega d\theta. \quad (2.15)$$

$$= \int_0^{2\pi} \int_{-\infty}^\infty |\omega| G(\omega, \theta) e^{j2\pi\omega(x \cos \theta + y \sin \theta)} d\omega d\theta. \quad (2.16)$$

$$= \int_0^\pi \left[\int_{-\infty}^\infty |\omega| G(\omega, \theta) e^{j2\pi\omega t} d\omega d\theta \right]_{t=x \cos \theta + y \sin \theta} d\theta. \quad (2.17)$$

In equation (2.17) the $|\omega|$ factor is a filter that accentuates high frequencies for each parallel-beam projection. After inverse Fourier transformation, the filtered projection is backprojected by substituting $t = x \cos \theta + y \sin \theta$, which is followed by summation of the filtered projections at all angles. As a result, this approach is termed *filtered backprojection* (FBP) and the high pass filter given by $|\omega|$ is called a ramp filter due to its shape in Fourier space. The ramp filter is carefully apodized to avoid amplification of high-frequency noise in the projection. The apodization filter can also be utilized to control the noise-resolution tradeoff for different imaging needs. The three steps in filtered backprojection are shown in Fig. 2.3

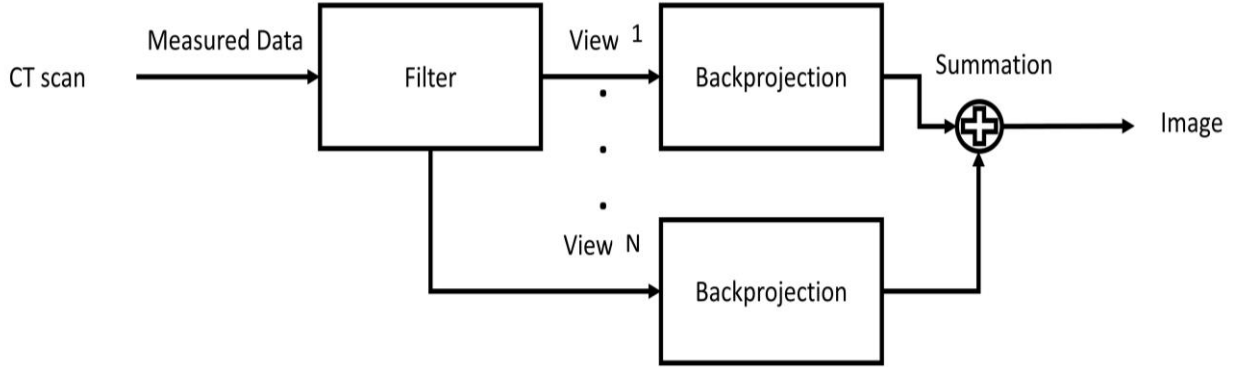


Fig. 2.3 Linear filtered backprojection algorithm for X-ray CT

The inverse Radon transform can also be adapted for use with a fan-beam geometry as shown in [34]. The resulting reconstruction formula for fan-beam is basically a weighted FBP formula. For our helical CT geometry reconstruction discussed in Chapter 4, we use the Feldkamp-Davis-Kress (FDK) algorithm. We rebin our cone-beam to equivalent parallel fan-beam projections and apply the backprojection method discussed previously. However, due to a sampling pattern difference between Cartesian and polar coordinate systems, interpolation can adversely affect the noise-resolution tradeoff.

2.1.2 Reconstruction from Statistical Data Model

In this section, we consider a mono-energetic, scatter-free data model which accounts for the randomness of the measured X-ray photon counts. Detailed data models exist in the literature [1, 36-38] which account more accurately for scatter, noise and beam hardening. At the basis of our statistical model, we assume the number of X-ray photons at each detector follows a Poisson counting process. For X-ray CT, simple Poisson is a good approximation to the more complex

compound Poisson process, which can effectively capture the physics of X-ray CT and scanner geometry since it is an appropriate model for a CT scanner with energy-integrating detectors [39]. If we denote the mean of our measurement data $d(y)$ as $g(y)$, where y is our source-detector pair, then we can represent the probability distribution of a particular measurement $d(y)$ by

$$P(d(y)) = \exp(-g(y)) g(y)^{d(y)} / d(y)! \quad (2.18)$$

Determination of the mean value $g(y)$ requires a forward projection which is basically an integral denoted by equation (2.8). However statistical reconstruction problems are not constrained by the projection slice theorem. The problem can simply be modeled by a discretized system matrix $h(y|x)$ that relates the image space to the data space by matrix vector multiplication as denoted by

$$l(y) = \sum_x h(y|x) \mu(x), \quad (2.19)$$

where $\mu(x)$ is the x – th voxel of the attenuation coefficient image. For transmission tomography, we use Beer’s law,

$$g(y) \triangleq E[d(y)] = I_0(y) e^{-l(y)}, \quad (2.20)$$

where $I_0(y)$ is the mean number of photons detected for y – th source-detector pair in the absence of an attenuating medium. The likelihood function can be expressed mathematically as

$$\hat{\mu} \triangleq \operatorname{argmax}_{\mu \geq 0} \prod_y \exp(-g(y)) g(y)^{d(y)} / d(y)! \quad (2.21)$$

where $\hat{\mu}$ is the maximum likelihood (ML) estimate of the image and the product is taken over all measurements. In order to write equation (2.21) as a product of Poisson probabilities, we assume each measurement is independent. However, it is easier to maximize the log-likelihood function

$$L(d|\mu) \triangleq \operatorname{argmax}_{\mu \geq 0} \sum_y d(y) \log(g(y)) - g(y), \quad (2.22)$$

where we have dropped the term containing $d(y)!$ since it is independent of μ and thus irrelevant to our optimization problem. It has been shown previously that the problem in equation (2.22) can have a guaranteed convergence to a possibly non-unique global maximum [36].

Since our problem can be classified as an ill-posed inverse problem, we may end up overfitting the image to the noisy data. In order to overcome this issue, we modify the likelihood function to be maximized by including a penalty. We can also think of this penalty function as an image prior that enforces local smoothness on the image. One of such choices is the Gibbs potential energy function,

$$U(\mu) \triangleq \sum_{x=1}^N \sum_{x' \in N_x} \phi(\mu(x) - \mu(x')). \quad (2.23)$$

Here, N_x is a local neighborhood of voxels surrounding voxel x , the potential function $\phi(\cdot)$ is often chosen to be a convex function, and the first sum is over all the voxels in the image volume. The introduction of the penalty function from equation (2.23) to our original ML problem in (2.22) makes this a penalized-likelihood (PL) problem. PL is quite useful when the problem is particularly ill-posed.

Numerical solutions for statistical reconstruction problems often use iterative gradient descent methods like Newton's methods to optimize the problem since there exists no closed form solution of the PL problem. Many algorithms have been developed previously to optimize the objective

function for transmission tomography. Lange and Carson proposed an expectation-maximization algorithm [40], Mumcuoglu et. al [41] developed a conjugate-gradient algorithm for computing maximum a-priori posteriori (MAP) estimates for both transmission CT and emission PET. Bouman et. al [42] developed an iterative coordinate-descent (ICD) algorithm which is basically a greedy pixel-wise computation that involves updating each image voxel sequentially. As a result, the ICD algorithm is not amenable to parallelization on GPU devices. Elkbari et. al [4] developed the concept of optimization transfer and surrogate functions which is used later in Chapter 3. O’Sullivan and Benac [1] developed an alternating minimization (AM) algorithm that alternates between exponential and linear family optimization. The proposed method provides a closed-form update for the ML algorithm with guaranteed convergence. For our implementation, we use an AM algorithm with a Huber-type penalty function used previously by [37].

2.1.3 Comparison of Analytical and Statistical Iterative Methods

The main hurdle for the adoption of statistical iterative-reconstruction methods in clinical CT scanners is their high computational burden. Also in most cases, CT scanners collect enough data to enable the use of linear, single-shot reconstruction methods like FBP or FDK to reconstruct high-quality, low-noise images. However, for low-dose CT [6, 43-45], irregular scanner geometries or incomplete data, these linear methods introduce troublesome artifacts, in which case SIR algorithms can be advantageous.

Unlike conventional linear backprojection algorithms, SIR algorithms allow the inclusion of additional information in the reconstruction process including photon statistics, physical properties

of the X-ray beam and image-penalty functions. For low-count photon measurements, Fessler [36] showed that the introduction of the logarithm for the computation of linear projection estimates in equation (2.3) adds systematic bias. However, the lack of linearization for statistical methods gives it an advantage over linear methods. Additionally, FBP algorithms apply the same weight to high variance, i.e. low-dose measurements and low variance measurements since they are unable to utilize the noise model of the measured data. This shortcoming introduces higher noise to images reconstructed from low-dose CT measurements.

For multislice cone-beam CT geometries, most linear algorithms fail to reduce cone-beam artifacts due to the large cone-beam angle. Although the FDK algorithm discussed later in this work somewhat reduces the cone-beam artifact, due to their approximate nature, these artifacts are not completely eliminated. The methods of Hsieh [46] and Katsevich [47] attempt to reduce noise in analytical reconstructions, but in the end, they are of limited utility due to their inability to incorporate measurement statistics. SIR algorithms, on the other hand, are based on a physically realistic model of signal statistics [42, 48, 49]. SIR algorithms attempt to incorporate the nonlinearities of the measurement systems rather than trying to overfit the reconstructed image to a noisy measurement. The non-linear objective function along with the roughness or edge preserving penalty function in SIR algorithms, gives us the leverage to adaptively control the tradeoff between desired resolution and noise tolerance.

This dissertation is focused on the reconstruction time and accuracy of different analytical and SIR algorithms. Although there are significant advantages for using FDK algorithms due to their

impressive computational efficiency, as discussed in this thesis, we believe that the use of multiple GPUs can reduce the reconstruction time of SIR algorithms significantly. As shown in later chapters we can use multiple GPUs and sophisticated parallelization schemes to not only accelerate the linear single-shot backprojection algorithms but we can also apply these techniques to a complex model-based reconstruction problem.

2.2 System Modeling

The system matrix used in iterative reconstruction can be computed either by ray-driven or voxel-driven methods. In a ray-driven method, a weight is assigned to the X-ray beam proportional to the amount of interaction between the beam and voxels it passes through in the object being imaged. On the other hand, in voxel-driven methods, the detector edges are projected to the voxel array along the ray path to compute the system matrix. De Man et. al [50] provides a good review of some available projection and backprojection methods. They also proposed a distance-driven method as a more accurate method to perform forward and backprojection. In the following section, we discuss the proposed distance-driven operators.

2.2.1 Branchless Distance-driven Projection

For the computation of a ray-driven projection, we can evaluate the contribution of the ray to the voxel by calculating the length of intersection along the ray path [51-54] or interpolate based on the distance of the X-ray beam to nearby beams [52, 55]. However, these ray-driven algorithms are not easy to parallelize, and sometimes introduce moiré patterns in backprojected images [50,

56]. However, voxel-driven projection and backprojection are more suitable for parallel hardware implementation.

One of the state-of-the-art projection algorithms, called distance-driven (DD) projection and backprojection, was proposed by De Man and Basu [50, 56]. In 2006 they proposed an extension to their algorithm called branchless distance-driven projection and backprojection [57] in which they basically parallelized the inner loop of their overlap calculation. They divided the overlap kernel into 3 distinct and independent steps: digital integration, interpolation, and digital differentiation. Schlifske et. al [58] proposed a 2-D extension to the branchless DD algorithm, in which they “pre-integrate” the 2-D image slice of the image volume before projection and after backprojection. In our work, we use a similar method in which we pre-accumulate the image intensities in 4 perpendicular image slabs in a recursive manner before projection in order to accommodate the 3-D helical nature of the data.

2.2.2 Branchless Distance-driven Backprojection

The core calculation of the algorithm is the computation of the overlap between the projection of an individual slab of the image volume onto a 2-D detector array. For our specific reconstruction, we used helical CT geometry. In our work, we have also employed a recursive adjoint accumulation scheme after backprojection to retrieve our final 3-D image volume. Our proposed method of pre-accumulation enables us to employ interpolation directly into the image accumulation array which reduces some of the computational burdens associated with the sequential integration of the original branchless DD method.

We also focus on the parallelization of the branchless DD backprojection over multiple GPUs. We first simplify the overlap computation of the branchless DD algorithm by projecting detector boundaries directly onto the image voxel boundaries. After that, we added a pre-accumulation scheme, which reduces the sequential integration burden on individual GPU threads. Next, we present a pseudocode for the implementation of our proposed algorithm on single and multiple GPUs. Last but not least, we have validated our overall parallelization scheme by reconstructing images from Siemens Sensation 16 helical CT data using the alternating minimization algorithm and its ordered subsets version.

2.3 Graphics Processing Unit Architecture

Graphics processing units (GPU) are specialized devices designed to rapidly manipulate and alter memory to accelerate the creation of images and send them to display devices. Shaped by the fast-growing video game industry that expects a tremendously massive number of floating-point calculations per video frame, there is an active research push to maximize the chip area and power budget dedicated to floating-point calculations. Therefore, modern GPUs are optimized for throughput i.e. the number of tasks processed per unit of time, while CPUs are optimized for low latency and the amount of time needed to perform a complex task. This high value of throughput is achieved by executing a large number of tasks on multiple threads while allowing individual threads to take a potentially much longer time to execute. This design saves chip area and power by allowing pipelined memory channels and arithmetic operations to have long latency. The reduced area and power of memory and arithmetic operations allow designers to pack more cores on a chip to increase the execution throughput. As compared to a normal CPU, more transistors

are devoted to data processing rather than data caching and flow control as shown in Fig. 2.4. DRAM stands for dynamic random-access memory and the ALU stand for arithmetic logic unit.

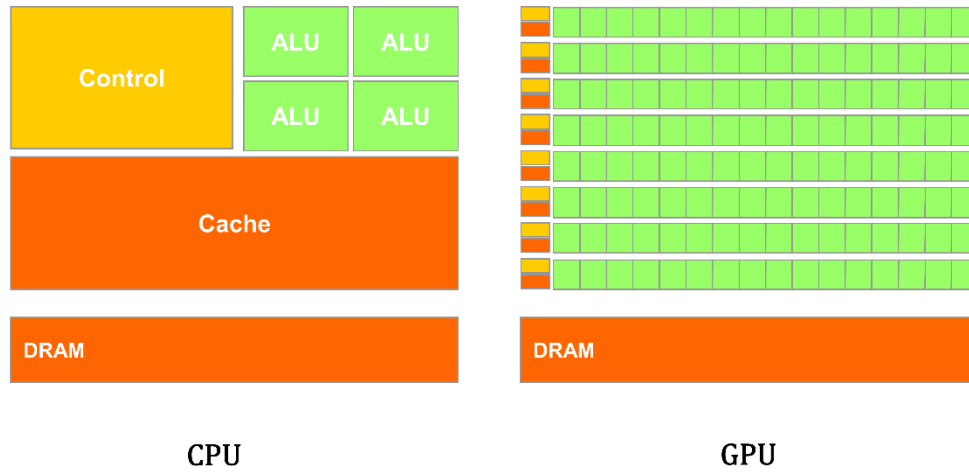


Fig. 2.4 CPU vs. GPU architecture

2.4 Acceleration of Statistical Iterative-reconstruction Algorithms

The majority of the time for the SIR algorithms is spent in the computation of the forward projections and backprojections. Considering the many benefits of the statistical reconstruction, one goal in the research community is to speed up the execution of these methods in order to reconstruct large 3-D volumes in a reasonable timeframe. A variety of acceleration techniques have been developed and can be divided into algorithmic and hardware approaches.

2.4.1 Algorithmic Speedup

Ordered subsets (OS) is a range-decomposition method introduced by Hudson and Larkin [59]. OS are able to speed up the convergence of parallel-update iterative-reconstruction algorithms significantly. An algorithm that utilizes OS iteratively computes image updates using only a subset of the available projection data. During each iteration, the OS algorithm cycles through each subset of data, performing an image update after each sub-iteration. OS can improve the convergence rate by a factor roughly equal to the number of subsets. However, for multi-GPU implementation, there is significant overhead in every OS iteration from combining data from multiple GPUs running in parallel. But, the amount of acceleration achieved using OS overshadows the increase in computational burden due to the OS implementation.

The original OS method removes the monotonic convergence guarantee of most statistical reconstruction algorithms. Convergent OS methods have been developed [60], but their memory demands may be too high for clinical practice. However, even without the convergence guarantee, the original OS method tends to be stable in practice.

Our surrogate-function based optimization technique discussed in Chapter 3 results in an independent parallel voxel-based update step which can be ideal for multi-GPU implementation. However, this kind of first-order surrogate function with the majorization property suffers from a sublinear rate of convergence. Thus, there has been a tremendous amount of research on methods for accelerating convex optimization. One of the most popular acceleration techniques is the Fast

Iterative Shrinkage-Thresholding Algorithm (FISTA) [32, 61]. In this work we propose, a novel adaptive surrogate-function based optimization technique.

The main motivation behind adaptive surrogate-function based acceleration is the fact that the update steps in the original unaccelerated surrogate-function based optimization technique are very conservative. The small update steps guarantee convergence to a global minimum but at a slow rate. The acceleration method we propose computes aggressive update step-size based on the measured sinogram, air scan, and current estimate of the image. After every iteration, we modify the update steps to include the previous update in the image domain. This scheme basically yields step-sizes which are unique to different regions in the image space. Image regions which are most divergent from the converged final image will have larger update steps and vice versa. The adaptive update step computation is independent for each voxel and can be easily implemented in a parallel multi-GPU architecture with negligible computational burden.

2.4.2 Hardware Speedup

In general, the computational burden of tomographic reconstruction is commensurate with the complexity and scale of the physical model underlying the process. In recent years, massively parallel commodity graphics hardware (Graphics Processing Units, GPUs) allowed the use of more complex models while maintaining reasonable execution times. Ultimately, this led to improved images in terms of dose efficiency, noise, artifacts, and spatial resolution, and opened the way to new applications. Nowadays, it is safe to assume that any serious attempt at developing an

advanced reconstruction algorithm for clinical applications requires hardware acceleration, often from massively parallel GPU cores.

With current advances in GPU memory size, we can easily fit the entire clinical image volume on the GPU device memory, which eliminates the high latency penalty of using external CPU memory to retrieve data as in older GPUs. Additionally, GPUs are notoriously slow in executing divergent branches (“if” statements) unless all processor cores take the branch, whereas CPUs are much better at this type of flow control. A modification to the distance-driven projector mentioned above was proposed that eliminates branching [57].

Projection and backprojection operations are often a performance bottleneck in CT reconstruction schemes. Being highly parallel, they are well suited for GPU implementations. The efficiency of projection and backprojection operations is particularly critical in iterative-reconstruction schemes as they are repeated multiple times and often become an overall performance bottleneck. From a GPU perspective, the forward projection is best obtained with a ray-driven approach, where each ray is assigned to an independent GPU thread. For backprojection, a voxel-driven approach is more adapted to the GPU architecture and avoids potential race conditions where two threads could write to the same memory location with unpredictable outcomes. However, a mismatch in projectors might lead to convergence issues in some circumstances [62]. Instead, to ensure convergence we employ voxel-driven projection and backprojection which are the exact adjoints of each other.

Chapter 3: Multislice Statistical Iterative

Helical CT Reconstruction Using GPU

3.1 Theory

Multislice helical CT has been proven to be a successful imaging modality in many clinical applications and is currently in widespread use. This kind of imaging modality is inherently 3D since the X-ray tube continuously projects a cone beam of X-rays through the object being imaged. At the same time, the patient is also translated along the gantry axis. Every detector captures data in a partial rotation of the gantry that corresponds to each image slice. In order to reconstruct an attenuation image from the measured data, we need to model the system geometry accurately. Below, we highlight the main aspects of this process along with the formulation of the fast-parallel statistical iterative reconstruction (SIR) algorithm for multislice helical CT.

3.1.1 Statistical Data Model

In this work, we consider a mono-energetic, noisy and scatter free data model which accounts for the randomness of the measured X-ray photon counts. Detailed data models exist in the literature [1, 36-38] which account for more accurate scatter, noise and beam hardening. At the basis of our statistical model, we assume the photons arrive at the detector array in accordance with a Poisson counting process. Such a model can effectively capture the physics of X-ray CT and scanner geometry while simplifying the 3-D reconstruction algorithm described below.

The 3-D image volume of linear attenuation coefficients in units of mm^{-1} is represented in the vector array μ . The index y refers to a ray path between the X-ray source and a pixel in the multirow detector array. The measured transmission data for the y^{th} source-detector pair, $d(y)$, is modeled as originating from independent Poisson counting processes. In discretized form, the mean value of $d(y)$ is modeled as:

$$g(y; \mu) \triangleq I_0(y) \exp \left(- \sum_x h(y|x) \mu(x) \right) + \beta(y), \quad (3.1)$$

where $I_0(y)$ is the mean number of counts in the absence of an attenuating medium, $\beta(y)$ is the mean number of background events assumed to be nonnegative and known, and $\mu(x)$ is the linear attenuation coefficient in voxel x . The system matrix elements $h(y|x)$ comprise the appropriately discretized point spread function relating the projection space to the image space. If projection y does not pass through voxel x , then $h(y|x)$ is zero. In a simple ray-tracing model, $h(y|x)$ represents the length of intersection between the voxel indexed by x and the ray-path indexed by y . The discretized forward projection operation can therefore be represented by $l(y)$ as:

$$l(y, \mu) \triangleq \sum_x h(y|x) \mu(x). \quad (3.2)$$

3.1.2 Image Reconstruction Formulation

In transmission tomography, the basic goal of image reconstruction is to estimate the spatial distribution of the linear attenuation coefficient, μ , in the scanned object. This can be achieved by maximizing a log-likelihood objective function between measured data and estimated data from our statistical model.

Maximum Likelihood

Using a polyenergetic data model with scatter, O’Sullivan and Benac [1] derived an alternating minimization (AM) algorithm to find the maximum loglikelihood (ML) solution. The problem was formulated as the double minimization of an I-divergence over a linear and an exponential family, thereby resulting in a closed-form update for each iteration. If we assume the individual detector measurements are independent Poisson random variables, the Poisson log-likelihood function is:

$$\mathcal{L}(d; \mu) = \sum_y [d(y) \log(g(y; \mu)) - g(y; \mu)]. \quad (3.3)$$

The objective of our iterative-reconstruction algorithm is to maximize the log-likelihood function in (3.3) subject to $\mu(x)$ being nonnegative, due to the nature of linear attenuation coefficients. It turns out that maximizing $\mathcal{L}(d; \mu)$ is equivalent to minimizing the I-divergence between $d(y)$ and $g(y; \mu)$. In other words,

$$\mu_{ML}^* = \operatorname{argmax}_{\mu \geq 0} \mathcal{L}(d; \mu) = \operatorname{argmin}_{\mu \geq 0} I(d(y) || g(y; \mu)), \quad (3.4)$$

where the I-divergence $I[d(y) || g(y; \mu)]$ is defined as:

$$I[d || g; \mu] \triangleq \sum_y \left[d(y) \log \left(\frac{d(y)}{g(y; \mu)} \right) + g(y; \mu) - d(y) \right]. \quad (3.5)$$

The objective function presented in (3.5) can’t be optimized directly over μ , in part because the optimization space is large. One of the best approaches is to develop surrogate functions that approximate the original function at every iteration and are easy to minimize. This approach leads to iterative algorithms where different surrogate functions are formed and solved at each iteration and yet the original function decreases monotonically.

In order to formulate a surrogate optimization function, we start with a nonnegative initial image, $\mu^0(x)$ where the superscript represents the iteration index, and create surrogate functions for I-divergence at each iteration and update the image by minimizing the surrogate function. Special properties of the surrogate function guarantee a monotonic decrease of the original function, which will be explained later in this section. If we ignore the terms of the I-divergence that do not depend on μ , the objective function to be minimized for the mono-energetic case is:

$$\bar{\mathbb{I}}[d||g;\mu] \triangleq \sum_y [-d(y) \log(g(y;\mu)) + g(y;\mu)]. \quad (3.6)$$

If we replace the estimated mean term $g(y;\mu)$ by $I_0(y)\exp(-\sum_x h(y|x)\mu(x))$, and ignore the term $\sum_y -d(y)\log(I_0(y))$ (which is independent of μ), equation (3.6) becomes:

$$\hat{\mathbb{I}}[d||g;\mu,\hat{\mu}] \triangleq \sum_y d(y) \sum_x h(y|x)\mu(x) + \sum_y I_0(y) \exp\left(-\sum_x h(y|x)\mu(x)\right), \quad (3.7)$$

$$\begin{aligned} &\triangleq \sum_x \mu(x) \sum_y h(y|x)d(y) \\ &+ \sum_y I_0(y) \exp\left(-\sum_x h(y|x)\hat{\mu}(x)\right) \exp\left(-\sum_x h(y|x)(\mu(x) \right. \\ &\quad \left. - \hat{\mu}(x))\right). \end{aligned} \quad (3.8)$$

We define the forward projection of the current image estimate $\hat{\mu}(x)$ as:

$$\hat{q}(y) = I_0(y) \exp\left(-\sum_x h(y|x) \hat{\mu}(x)\right), \quad (3.9)$$

the backprojection of $\hat{q}(y)$ as

$$\hat{b}(x) = \sum_y h(y|x) \hat{q}(y), \quad (3.10)$$

and the backprojection of measured data $d(y)$ as

$$\tilde{b}(x) = \sum_y h(y|x) d(y). \quad (3.11)$$

Therefore, I-divergence can be defined as:

$$\begin{aligned} \hat{I}[d||g; \mu, \hat{\mu}] &= \sum_x \mu(x) \tilde{b}(x) \\ &+ \sum_y \hat{q}(y) \exp\left(-\sum_x h(y|x) (\mu(x) - \hat{\mu}(x))\right). \end{aligned} \quad (3.12)$$

Using the convex decomposition described in Lemma B.0.2 in Appendix B, we can derive the following inequality,

$$\begin{aligned} \hat{I}[d||g; \mu, \hat{\mu}] &\leq \sum_x \mu(x) \tilde{b}(x) \\ &+ \sum_y \hat{q}(y) \sum_x r(y|x) \exp\left(-\frac{h(y|x)}{r(y|x)} (\mu(x) - \hat{\mu}(x))\right), \end{aligned} \quad (3.13)$$

where

$$r(y|x) \geq 0, \forall y, x \quad (3.14)$$

$$\sum_x r(y|x) \leq 1 \forall y. \quad (3.15)$$

If we choose

$$r(y|x) = \frac{h(y|x)}{Z}, \forall x, y \quad (3.16)$$

where Z , also referred to as auxiliary function is set equal to the maximum projection length through the reconstruction cylinder, or

$$Z = \max_y \sum_x h(y|x). \quad (3.17)$$

As a result, we can satisfy the conditions denoted by equations (3.14) and (3.15). Finally, we define the surrogate function of the data fit term $\hat{\mathbb{I}}[d||g; \mu, \hat{\mu}]$ using equations (3.10), (3.13) and (3.16), which gives

$$\hat{\mathbb{I}}[d||g; \mu, \hat{\mu}] = \sum_x \mu(x) \tilde{b}(x) \quad (3.18)$$

$$\begin{aligned} &+ \sum_y \hat{q}(y) \sum_x \frac{h(y|x)}{Z} \exp(-Z(\mu(x) - \hat{\mu}(x))) \\ &= \sum_x \mu(x) \tilde{b}(x) \\ &+ \frac{1}{Z} \sum_x \left(\sum_y \hat{q}(y) h(y|x) \right) \exp(-Z(\mu(x) - \hat{\mu}(x))) \end{aligned} \quad (3.19)$$

$$= \sum_x \mu(x) \tilde{b}(x) + \frac{1}{Z} \sum_x \hat{b}(x) \exp(-Z(\mu(x) - \hat{\mu}(x))). \quad (3.20)$$

The surrogate function has the following majorization properties:

$$I[d||g; \mu] = \hat{I}[d||g; \mu, \mu] \forall \mu, \quad (3.21)$$

$$I[d||g; \mu] \leq \hat{I}[d||g; \mu, \hat{\mu}] \forall \mu, \hat{\mu}. \quad (3.22)$$

Using these two properties from equation (3.21) and (3.22), we can conclude that

$$I[d||g; \hat{\mu}] - I[d||g; \mu] \geq \hat{I}[d||g; \hat{\mu}, \hat{\mu}] - \hat{I}[d||g; \mu, \hat{\mu}]. \quad (3.23)$$

In other words, if one can find some μ that makes the right-hand side of (3.23) positive (some μ that decrease the surrogate-function value), then the original objective function also decreases. This is the key ingredient for forming iterative algorithms using any kind of surrogate functions, including the Jensen type for our case. With a proper choice of $r(y|x)$, the surrogate can be “decoupled”; in other words, minimizing $\hat{I}[d||g; \mu, \hat{\mu}]$ can become N one-dimensional independent convex minimization problems (one for each $\mu(x)$), which are easy to parallelize. In order to solve this surrogate function, we can equate the derivative of this function w.r.t. μ to 0 as, or

$$\frac{\partial \hat{I}[d||g; \mu, \hat{\mu}]}{\partial \mu(x)} = 0 \forall x. \quad (3.24)$$

The derivative of the surrogate function of the I-divergence is

$$\frac{\partial \hat{I}[d||g; \mu, \hat{\mu}]}{\partial \mu(x)} = \tilde{b}(x) - \hat{b}(x) \exp(-Z(\mu(x) - \hat{\mu}(x))) \forall x. \quad (3.25)$$

If we denote the estimate of μ at the k -th iteration by $\hat{\mu}^{(k)}$, then the closed form solution of the maximum-likelihood function from O’Sullivan and Benac [1] can be expressed as

$$\hat{\mu}^{(k+1)}(x) = \left[\hat{\mu}^{(k)}(x) + \frac{1}{Z} \log \frac{\hat{b}^{(k)}(x)}{\tilde{b}(x)} \right]_+ \quad \forall x. \quad (3.26)$$

The $[\cdot]_+$ is shorthand for $\max(\cdot, 0)$. The decoupling steps provide an iterative algorithm that is guaranteed to decrease the objective function monotonically. Also, it creates many one-parameter convex functions (one for each voxel) that can be minimized in parallel using GPU threads. The pseudocode for the unregularized AM algorithm is shown in Algorithm 3.1.

Algorithm 3.1 AM algorithm

Input: $\hat{\mu}^{(0)}(x) = 0, Z = 2 \cdot R_{recon} \in \mathbb{R}_+, d(y), I_0(y) \in \mathbb{R}_+^M$

Precompute $\tilde{b}(x) = \sum_y d(y)h(y|x), \forall x$

for $k = 1, 2, 3, \dots$ **do**

$$\hat{q}^{(k)}(y) = I_0(y) \exp(-\sum_x h(y|x) \hat{\mu}^{(k)}(x)) \quad \forall y$$

$$\hat{b}^{(k)}(x) = \sum_y h(y|x) \hat{q}^{(k)}(y) \quad \forall x$$

$$\hat{\mu}^{(k+1)}(x) = \left[\hat{\mu}^{(k)}(x) + \frac{1}{Z} \log \frac{\hat{b}^{(k)}(x)}{\tilde{b}(x)} \right]_+ \quad \forall x$$

end for

Penalized Likelihood

Since the measured data are noisy, it is necessary to regularize the optimization problem to prevent the algorithm from over-fitting the data through unrealistic images. This necessitates the use of edge-preserving penalty functions to incorporate the neighboring voxel interactions in the algorithm to perform a trade-off between data fitting and image smoothness.

To derive the algorithm for penalized maximum-likelihood estimation, we add a penalty term, $R(\mu)$, to the objective function used in the AM reconstruction, and weight it by a regularization parameter λ , where λ is a nonnegative scalar that reflects the amount of smoothing desired. A larger value will give emphasis to the penalty term (i.e., the prior expectation that the image will be smooth), whereas a smaller value will give more emphasis to the I-divergence term (i.e., the discrepancy between the measured data and the data estimated by the model). The added penalty term is defined as:

$$R(\mu(x)) \triangleq \sum_{x' \in N(x)} \omega(x, x') \psi(\mu(x) - \mu(x')), \quad (3.27)$$

where $R(\mu(x))$ can be interpreted as the log-likelihood term for some prior. For 3-D regularization, we use the 26-voxel neighborhood $N(x)$ surrounding voxel x . The weights $\omega(x, x')$ control the relative contribution of each neighbor. The potential function $\psi(t)$ is a symmetric convex function that penalizes the difference between the values of neighboring voxels. We used an edge preserving penalty function

$$\psi(t) = \left(\left| \frac{t}{\delta} \right| - \log \left(1 + \left| \frac{t}{\delta} \right| \right) \right) \quad (3.28)$$

previously used by other researchers [40, 63, 64] and decouple the image variables of our penalized objective function in such a way that all the voxels can still be updated in parallel. In this penalty, t represents the difference between neighbouring voxel values, and δ is a parameter that controls the transition between a quadratic region (for smaller $\left| \frac{t}{\delta} \right|$) and a linear region (for larger $\left| \frac{t}{\delta} \right|$). For our specific reconstruction, we exclude a few image slices from the beginning and end of the image volume in the penalty calculation because those slices will have severe artifacts due to cone-beam

truncation. Calculating the penalty for those slices could negatively impact reconstruction of the inner slices since the artifacts do not carry any type of structure that can meaningfully be penalized by $R(\mu)$. The overall problem is then to find the penalized-likelihood estimate,

$$\mu_{PML}^* = \underset{\mu \geq 0}{\operatorname{argmin}} I[d||g(\mu)] + \lambda R(\mu) \quad (3.29)$$

The addition of the penalty term eliminates the possibility of using a closed form solution as in equation (3.26). Instead, we use Newton's method on the decoupled I-divergence and penalty surrogate functions as shown in Appendix A. For ordered subsets, used in later sections, we scale down λ by the number of subsets used in that iteration. The pseudocode for the regularized AM algorithm is shown in Algorithm 3.2.

Algorithm 3.2 Regularized AM algorithm

Input: $\hat{\mu}^{(0)}(x) = 0 \in \mathbb{R}_+^N, Z = 2 \cdot R_{recon} \in \mathbb{R}_+, d(y) \in \mathbb{R}_+^M, I_0(y) \in \mathbb{R}_+^M, \lambda \geq 0, \delta > 0$

Precompute $\tilde{b}(x) = \sum_y d(y)h(y|x), \forall x$

for $k = 1, 2, 3, \dots$ **do**

$$\hat{q}^{(k)}(y) = I_0(y) \exp(-\sum_x h(y|x) \hat{\mu}^{(k)}(x)) \forall y$$

$$\hat{b}^{(k)}(x) = \sum_y h(y|x) \hat{q}^{(k)}(y) \forall x$$

$$\hat{\mu}^{(k+1)}(x) = \underset{\mu(x) \geq 0}{\operatorname{argmin}} \tilde{b}(x)(\mu(x) - \hat{\mu}^{(k)}(x)) + \frac{\hat{b}^{(k)}(x)}{Z} \exp(-Z(\mu(x) - \hat{\mu}^{(k)}(x))) +$$

$$\lambda \sum_{x' \in N(x)} \frac{\omega(x, x')}{2} \delta^2 \left(\left| \frac{2\mu(x) - \hat{\mu}^{(k)}(x) - \hat{\mu}^{(k)}(x')}{\delta} \right| - \log \left(1 + \left| \frac{2\mu(x) - \hat{\mu}^{(k)}(x) - \hat{\mu}^{(k)}(x')}{\delta} \right| \right) \right)$$

end for

3.2 Branchless Distance-driven Projectors

The geometry of our helical multislice CT scanner is shown in Fig. 3.1. The X-ray source rotates at a radius of R_f and the detector array rotates along the same direction at a radius of R_d from the isocenter. For the point $P(x, y, z)$ on the bold line in Fig. 3.1, β is the view angle, γ is the fan angle and η is the cone angle. z_{feed} is the axial distance travelled by the patient bed in one complete rotation of the X-ray source detector pair.

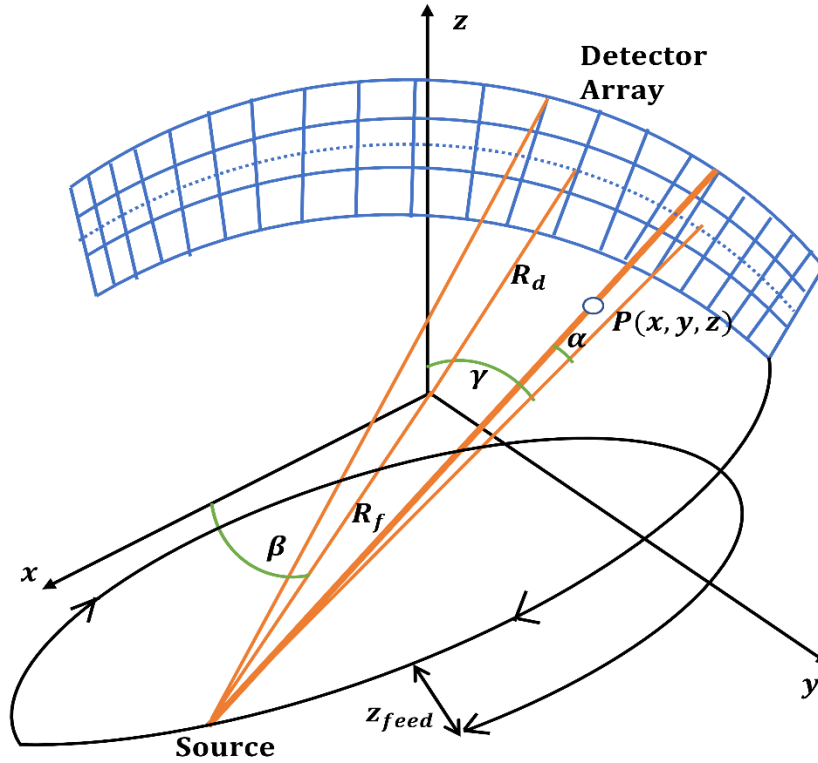


Fig. 3.1 The multislice helical geometry used in this dissertation.

3.2.1 Modification of Detector Edge Projections

The core calculation of the algorithm is the computation of the overlap between the projection of an individual slab of the image volume onto a 2-D detector array.

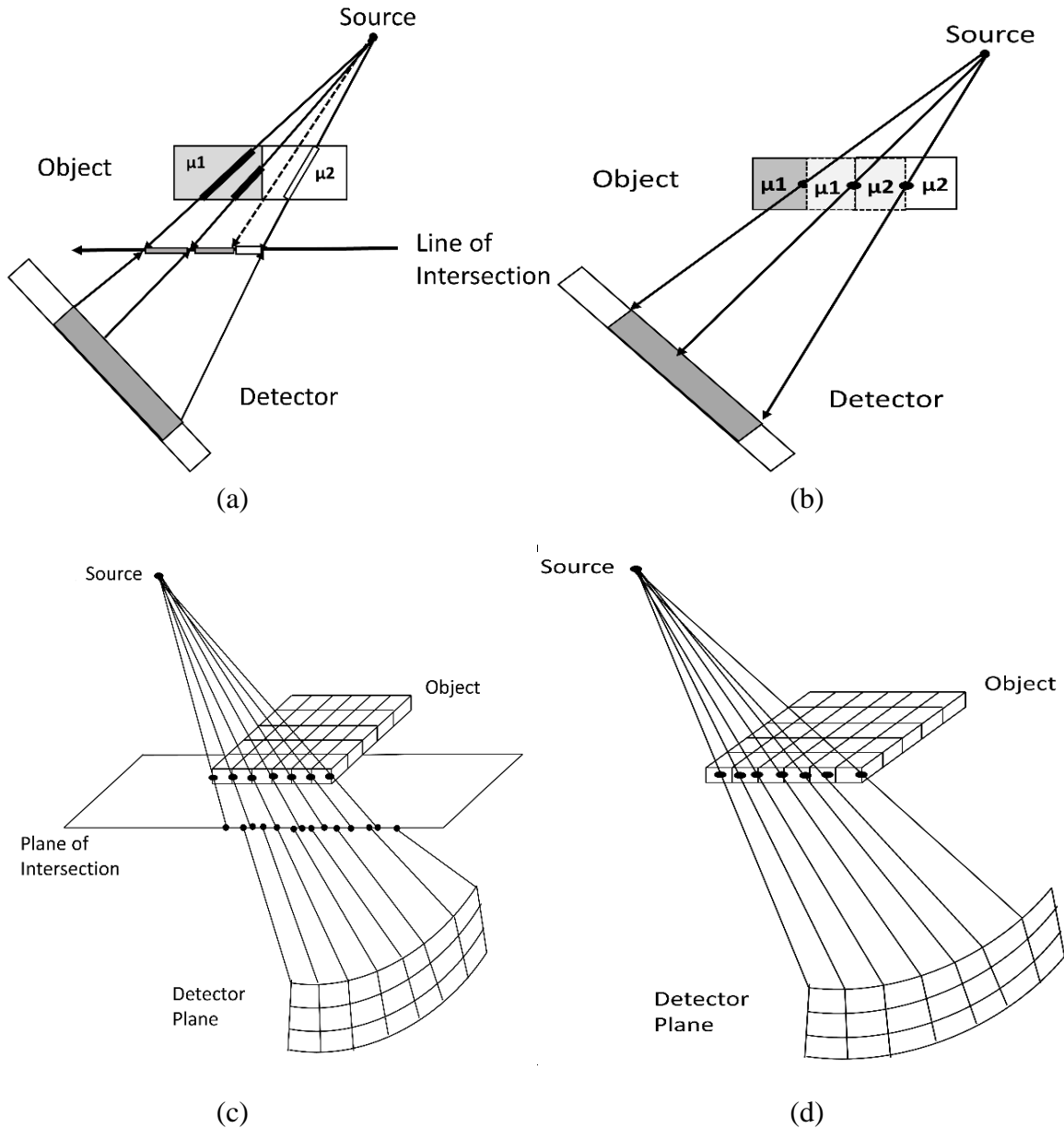


Fig. 3.2 (a) Schematic representation of De Man and Basu's [50] 2-D distance-driven method. (b) Schematic representation of our 2-D distance-driven method. (c) Schematic representation of De Man and Basu's [50] 3-D distance-driven method. (d) Schematic representation of our 3-D distance-driven method.

In our algorithm, the overlap calculations are performed directly at the level of the slab of interest. This differs slightly from the method proposed by De Man and Basu [57], where the overlap calculations are performed in the xz or yz plane passing through the origin. In that case, both the flattened voxel edges and detector edges would need to be projected onto the plane passing through the origin. In our implementation, the only projection calculations are from the detector edges to the slab. The coordinates of the source-to-detector ray intersections with the flattened image voxel array or slabs determine the 2-D rectangular region of the slab that contributes to each detector element. These rays are constructed using the edges of each detector element. For the completion of an X-ray projection image for a particular view angle, all the slab contributions are aggregated for a particular detector array. The contribution is also scaled by the length of the intersection of the ray through that slab. For our particular reconstruction, we assumed the slabs are flat and of uniform thickness.

3.2.2 Pre-accumulation for Forward Projection

First, we consider the contribution from a 1-D pixel array (i.e., one slab of a 2-D image) to a detector element at a fixed view angle. The pixels are uniformly spaced and represent a piecewise continuous function, $f(x)$, using a rectangle basis of unit width [57],

$$f(x) \triangleq \sum_i f_i \phi(x - i), \quad (3.30)$$

where

$$\phi(x) = \begin{cases} 0 & x < -0.5 \\ 1 & -0.5 \leq x \leq 0.5 \\ 0 & x > 0.5 \end{cases} \quad (3.31)$$

We wish to find the total contribution of the pixel array to detector element k with edges $x = u_1$ and $x = u_2$. This is mathematically expressed as:

$$g_k \triangleq \frac{1}{u_2 - u_1} \int_{u_1}^{u_2} f(x) dx = \frac{F(u_2) - F(u_1)}{u_2 - u_1}, \quad (3.32)$$

where

$$F(u) \triangleq \int_{-\infty}^u f(x) dx. \quad (3.33)$$

Let $K \triangleq \lfloor u \rfloor$, i.e. floor (u). Plugging it into (3.29), equation (3.32) can be rewritten as

$$F(u) = \sum_i f_i \int_{-\infty}^u \phi(x - i) dx \quad (3.34)$$

$$= \sum_{i=0}^{K-1} f_i \int_{-\infty}^K \phi(x - i) dx + f_K \int_K^u \phi(x - K) dx \quad (3.35)$$

$$= \sum_{i=0}^{K-1} f_i + (u - K) f_K. \quad (3.36)$$

Next, we can define an accumulated pixel array,

$$A[m] \triangleq \sum_{i=0}^{m-1} f_i. \quad (3.37)$$

We can rewrite equation (3.35) using (3.36) as follows:

$$F(u) = A[K] + (u - K)f_K \quad (3.38)$$

$$= A[K] + (u - K)(A[k + 1] - A[K]). \quad (3.39)$$

Now $F(u)$ can be calculated simply in terms of the pre-accumulated array A , and the original pixel values f_i are no longer needed. In fact, (3.39) is nothing more than linear interpolation into array A . The final step to calculate g_k is to substitute the value of $F(u)$ from equation (3.39) to equation (3.32). Now we consider the actual contribution from a 2-D slab to a detector element k with edges $x = u_1, x = u_2, y = v_1$, and $y = v_2$ as shown in Fig. 3.3.

$$g_k \triangleq \frac{1}{(u_2 - u_1)(v_2 - v_1)} \int_{u_1}^{u_2} \int_{v_1}^{v_2} f(x, z) dz dx. \quad (3.40)$$

We can define a continuous-coordinate slab using separable rectangular functions as:

$$f(x, z) \triangleq \sum_i \sum_j f_{ij} \phi(x - i) \phi(z - j). \quad (3.41)$$

We can represent in-plane calculations for each basis position j in the z direction as:

$$F_j(u) = A_j[K] + (u - K)(A_j[K + 1] - A_j[K]), \quad (3.42)$$

where

$$A_j[m] \triangleq \sum_{i=0}^{m-1} f_{ij}. \quad (3.43)$$

This leads to

$$g_k = \frac{1}{(u_2 - u_1)(v_2 - v_1)} \sum_j F_j(u_2) - F_j(u_1) \int_{v_1}^{v_2} \phi(z - j) dz \quad (3.44)$$

$$g_k = \frac{G(u_1, u_2, v_2) - G(u_1, u_2, v_1)}{(u_2 - u_1)(v_2 - v_1)}, \quad (3.45)$$

where

$$G(u_1, u_2, v) = \sum_j F_j(u_2) - F_j(u_1) \int_{-\infty}^v \phi(z - j) dz. \quad (3.46)$$

Similarly, we can define an accumulated voxel array in the z direction

$$C_{u_1, u_2}[n] \triangleq \sum_{j=0}^{n-1} B_j(u_1, u_2). \quad (3.47)$$

Analogous to (3.39) we define $J \triangleq \lfloor v \rfloor$. We can then write

$$G(u_1, u_2, v) = C_{u_1, u_2}[J] + (v - J)(C_{u_1, u_2}[J + 1] - C_{u_1, u_2}[J]). \quad (3.48)$$

We can also write $\sum_j F_j(u_2) - F_j(u_1)$ as weighted sum of few elements of $A_j[m]$,

$$B_j(u_1, u_2) = \sum_m \omega_m A_j[m], \quad (3.49)$$

Where ω_m is nonzero for up to four distinct values of m , as determined by (3.39) and (3.42).

Therefore, the slab can be pre-accumulated in both the x and z directions, as shown below:

$$C_{u_1, u_2}[n] = \sum_{j=0}^{n-1} \sum_m \omega_m A_j[m], \quad (3.50)$$

$$= \sum_m \omega_m \sum_{j=0}^{n-1} \sum_{i=0}^{m-1} f_{i,j}, \quad (3.51)$$

$$= \sum_m \omega_m S[m, n], \quad (3.52)$$

where

$$S[m, n] \triangleq \sum_{j=0}^{n-1} \sum_{i=0}^{m-1} f_{i,j}. \quad (3.53)$$

Finally, this accumulation can be written in recursive form for faster calculation as follows:

$$S[m, n] = \sum_{j=0}^{n-1} A_j[m], \quad (3.54)$$

$$= \sum_{j=0}^{n-2} A_j[m] + A_{n-1}[m], \quad (3.55)$$

$$= S[m, n-1] + \sum_{i=0}^{m-1} f_{i,n-1}. \quad (3.56)$$

For the projection model, as shown above, we pre-accumulate original pixel values in a recursive manner to a pre-accumulation array corresponding to four perpendicular slabs, each contributing to a different orientation of our view angle.

3.2.3 Pre-accumulation for Backprojection

Backprojection for the distance-driven kernel is defined as the transpose of the forward projection operator. Using flow graph reversal, the transpose of the entire kernel can be done by transposing each sub-operation and performing them in the reverse order, i.e.:

- (a) Transposed digital differentiation,
- (b) Transposed linear interpolation or “anterpolation”,
- (c) Transposed integration,

By writing out the 2-D slab accumulation operation (3.56) in matrix form, it can be shown that the transpose of slab accumulation is

$$f_{i,j}^* = \sum_{n=j+1}^{N_z} \sum_{m=i+1}^{N_x} S[m,n], \quad (3.57)$$

where N_x and N_z are the number of voxels in the two directions, respectively. This operation can also be written recursively for faster calculation. If we let

$$D[i,n] \triangleq \sum_{m=i+1}^{N_x} S[m,n], \quad (3.58)$$

then

$$f_{i,j}^* = \sum_{n=j+1}^{N_z} D[i, n], \quad (3.59)$$

$$= \sum_{n=j+2}^{N_z} D[i, n] + D[i, j+1], \quad (3.60)$$

$$= f_{i,j+1}^* + D[i, j+1], \quad (3.61)$$

$$= f_{i,j+1}^* + \sum_{m=i+1}^{N_x} S[m, j+1]. \quad (3.62)$$

For transposed digital integration, we perform the similar recursive post accumulation technique over the accumulated backprojection array to retrieve the individual voxel values from the 2-parallel pair of mutually perpendicular slabs.

3.2.4 Modified Overlap Computation

After the pre-accumulation, the original voxel values are no longer required. In fact, we perform direct interpolation of detector edges onto this accumulation array for both forward projection and backprojection, which gives us a big boost on the time performance over the sequential computation of digital integration for every region of overlap.

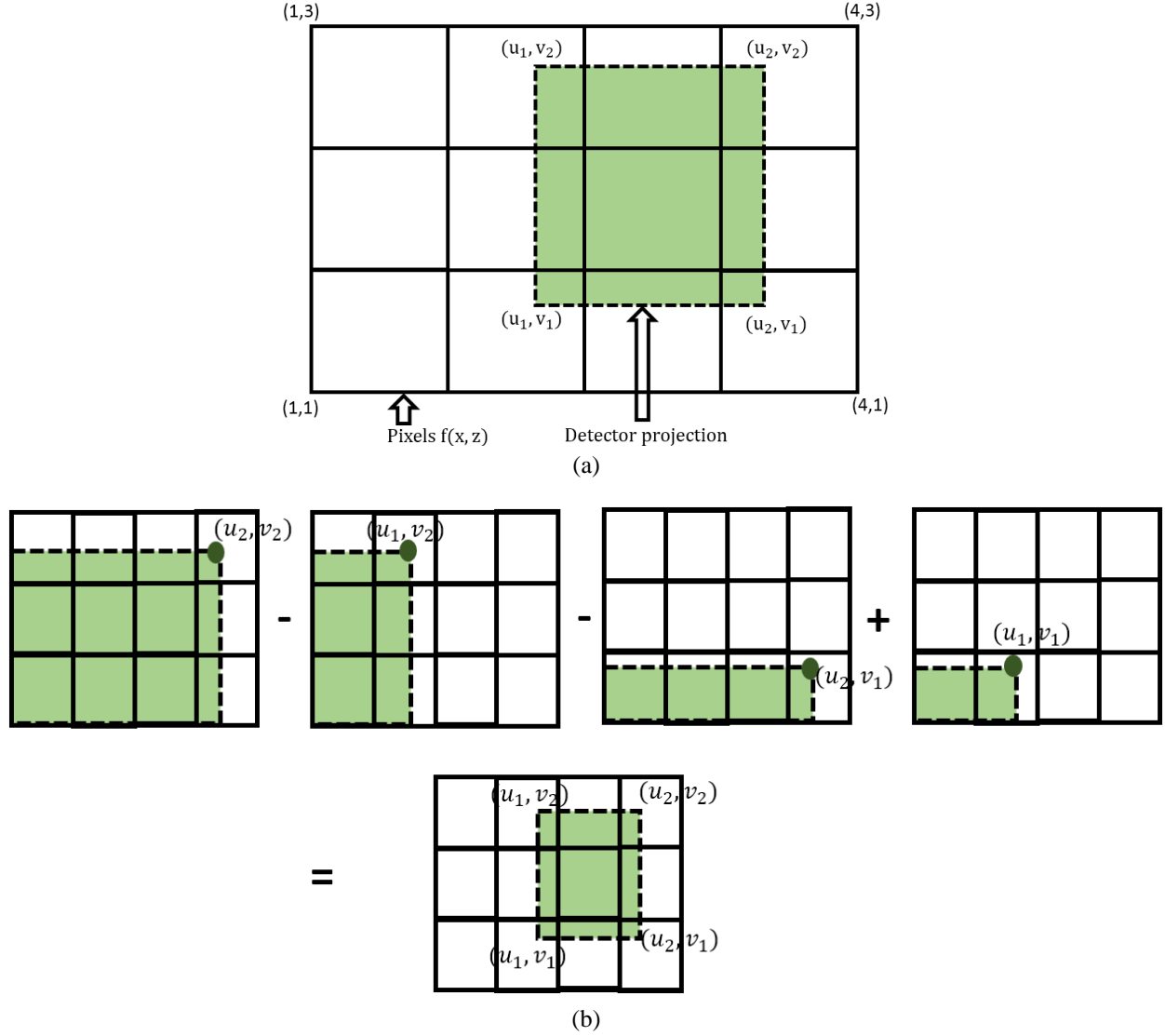


Fig. 3.3 (a) Schematic diagram of detector projection on image pixel slab which signifies the area of overlap. (b) Our approach to the calculation of overlap between detector edge projections and image pixel slabs.

3.3 CPU Multithreaded Parallelization Scheme for Branchless Distance-driven Projectors

Before performing interpolation and differentiation, we determine which part of the algorithm could be divided into independent processes to run on a single GPU thread. The way branchless

projection methods are structured, the interpolation and digital differentiation for each slab at each quarter rotation are independent of one another, so it can be implemented on a single GPU thread.

3.3.1 Symmetry

It was determined that the source-detector edge intersections with each slab (to find u_1 ; u_2 ; v_1 ; and v_2) need to be calculated only for the first quarter rotation of the gantry regardless of the length of the scan. For this symmetry to be valid, an integer number of image slices must correspond to the distance the bed travels in a quarter rotation of the gantry. This is actually not much of a restriction, as any helical pitch may be used, and the reconstruction slice thickness can be made arbitrarily small. In fact, it becomes even less limiting for scanners with larger axial coverage since they have a higher travel per rotation at a given pitch. The other constraint (which seems to always be satisfied in practice) is for the number of views per rotation to be a multiple of four.

The quarter-rotation symmetry is illustrated in Fig. 3.4(a) for an example where the bed translates two slices per quarter-rotation (denoted by N_q in the Fig. 3.4). The solid box indicates the portion of the scan (i.e., the first quarter rotation) for which the intersection calculations must be computed, while the dashed boxes represent the remaining symmetric quarter rotations. Also appearing in this figure are two diagonal lines, which correspond to the axial coverage of the cone-beam at each view angle. Fig. 3.4(b) shows the four-fold rotational symmetry in the xy plane for an arbitrary view angle and its $\frac{\pi}{2}$ -rotated offsets. A symmetric source-detector ray within each view is also shown. This symmetry is used in conjunction with the appropriate slice offset to identify the correct

region of the slab in each quarter rotation. Note that the top half of Fig. 3.4 lines up vertically with the bottom half.

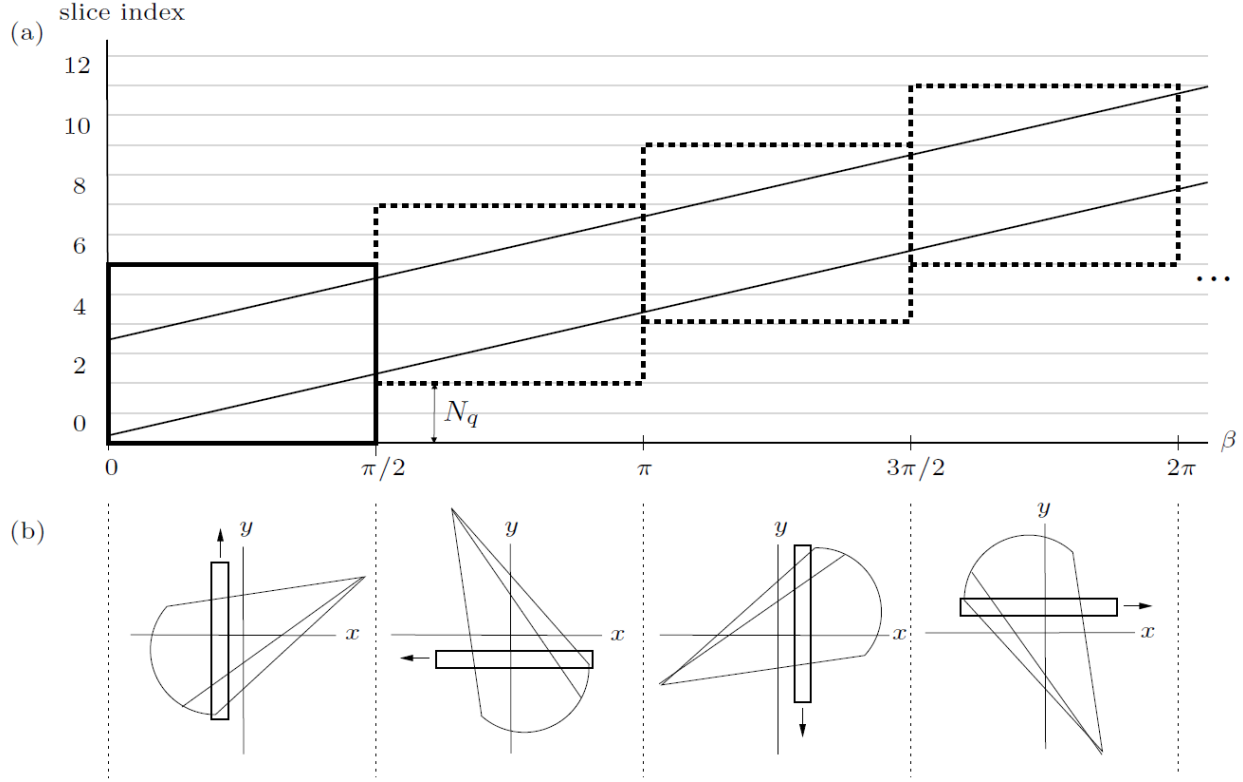


Fig. 3.4 (a) Axial view of the quarter-rotation symmetry found in helical CT. When an integer number of slices is chosen per quarter rotation of the gantry, the geometry calculations need only be done for just the first quarter rotation of the scan (indicated by a dark solid box). (b) Transverse view of the quarter-rotation symmetry. The projection calculations for each of the slabs shown is identical in the in-plane direction and offset by multiples of N_q in the z direction. An arrow has been drawn for each slab that indicates the direction of in-plane accumulation. The z accumulation is always in the direction of the positive z axis. Similar approach to quarter-rotation symmetry was explored by D. Keesing [65].

Use of quarter-rotation symmetry requires that the image volume be accumulated in the four $\frac{\pi}{2}$ -rotated orientations. (Image accumulation refers to 2-D accumulations according to $S[m, n]$ within each slab, but not across slabs.) The direction of image accumulation in x or y is indicated by the arrows in Fig. 3.4(b). Accumulation in the z direction is always in the direction of the positive z

axis. Therefore, four full-sized accumulation images reside in memory during forward or backprojection.

3.3.2 Multi-threaded Implementation for Forward Projection

This section discusses our method for parallelizing the forward projection in the helical orbit geometry. The fact that the system matrix is symmetric for each quarter rotation makes it quite natural to implement parallelism at the granularity of a quarter rotation of data. Each processor or core is assigned a contiguous group of projections whose cardinality is a multiple of the number of views in a quarter rotation. This design allows for theoretically perfect load balancing (in the absence of memory-related latencies) during forward and backprojection since each processor essentially makes use of the same number of nonzero $h(y|x)$ elements. Locality is inherent in this framework too; each quarter rotation of data is related to a local neighborhood of slices, as seen in Fig. 3.4(a).

The pseudocode for the parallelized forward projection is shown in Algorithms 3.3. The set Z_p refers to the range of voxel locations in the z direction that contribute to view index p . The set of symmetric view indices corresponding to quarter-rotation offsets of p on an individual processor are denoted Q_p .

Algorithm 3.3 Branchless distance-driven forward projection

```

Perform 2-D accumulation of  $\mu$  for each quarter rotation according to equation (3.56)
begin parallel region
  for  $p \in$  views within first quarter-rotation do
    for each slab in accumulation, image do
      for  $u = 1:N_{columns}$  do
        determine if the channel contribution to the slab is nonzero
        interpolate slab at detector column edge
        differentiate the value of the interpolation
        for  $v = 1:N_{rows}$  do
          interpolate column differentiation results at detector row edge
          for  $q \in Q_p$  do
            differentiate row interpolation values at row edges
            accumulate the differentiation value to the projection array
          end for
        end for
      end for
    end for
  end for
  Weight projection by lengths of intersection through the single slab ( $\forall q \in Q_p$ )
end for
end parallel region

```

3.3.3 Multithreaded Implementation for Backprojection

If we were to perform backprojection directly into the shared full-sized accumulation images, we would have serious memory contention issues since multiple processors would be writing to the same array elements simultaneously. Instead, each processor performs backprojection to its own private accumulation image arrays (of reduced size compared to the full-sized arrays). This eliminates any need for synchronization during the backprojection of a processor's set of views.

It is easiest to illustrate this concept with an example. Referring to Fig. 3.5, suppose there are two processors; the first one is assigned $\beta \in [0, \pi)$ and the second one is assigned $\beta \in [\pi, 2\pi)$. It can be observed that processor 0 only ever needs to access slices 0 – 2, while processor 1 only ever

needs to access slices 1 – 3. Therefore, each partial accumulation image consists of three slices, and each processor can easily determine what its starting slice index should be.

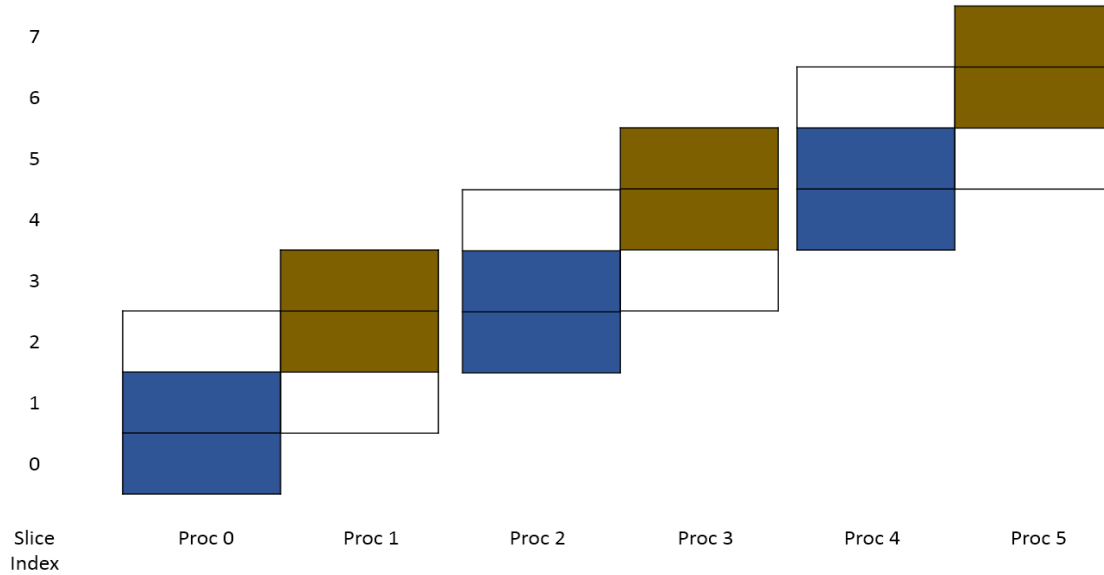


Fig. 3.5 Summing of private partial accumulation images on processors 0 and 1 into full-sized accumulation image. At each stage, the shaded block of slices from each processor is simultaneously summed into the full-sized accumulation image.

Once each processor is done backprojecting its set of views, the partial accumulation image arrays need to be summed into the shared, full-sized accumulation image arrays. After each block, a barrier synchronization construct is used to ensure each processor has finished summing the current block of slices to the full-sized arrays.

The pseudocode for the parallelized backprojections is shown in Algorithm 3.4

Algorithm 3.4 Branchless distance-driven backprojection**Begin parallel region** **for** $p \in$ views within first quarter-rotation **do** **for** each slab in accumulation, image **do** **for** $u = 1:N_{column}$ **do**

determine if the channel contribution to the slab is nonzero

for $v = 1:N_{rows}$ **do**

adjoint differentiate the corresponding element in projection array in the row direction

for $q \in Q_p$ **do**

interpolate results for corresponding row edge

end for

adjoint differentiate for corresponding detector column edge for all relevant column edges

interpolate result for corresponding detector column edge to slab

end for

interpolate result for last detector column edge to slab

end for **end for** **end for****end parallel region**

perform 2D adjoint accumulation for every quarter according to equation (3.62)

sum the four adjoint accumulation images.

3.4 GPU Implementation of Branchless Distance-driven Projectors

In our GPU based parallel implementation of branchless DD projection, each thread calculates a single partial projection element for specific view angle. The pre-accumulation is done before the start of forward projection in GPU. CPU threads are very efficient in handling serial operations like summation, however, we can harness the block reduction algorithms in CUDA to perform our pre-accumulation on GPUs. For each flattened slab of the volumetric image, pixels are accumulated in a vertical and horizontal direction similar to the method used by Rui Lui et al. [66]. The pre-accumulated images are mapped on to read-only texture memory due to their super-fast

memory access capabilities. Another motivation behind the use of texture memory is due to the fact that bilinear interpolation step can be performed really fast due to their inherent hardware architecture being specifically designed for pixel rasterization and rendering.

A basic pseudocode of the 3-D implementation of forward projection described in Algorithm 3.5.

Algorithm 3.5 GPU implementation of branchless distance-driven forward projection

```

Perform 2-D accumulation of  $\mu$  for each quarter rotation according to equation (3.56)
number of GPU threads launched = number of views within 1st quarter rotation  $\times$  number of
slabs in the accumulated image  $\times$  number of quarter rotations assigned to each GPU
begin GPU kernel
  for all GPU blocks in parallel do
    for all threads in a block do
      begin GPU thread calculation
        for every detector column
          determine if the channel contribution to the slab is nonzero
          interpolate slab at detector column edge
          differentiate the value of the interpolation
        for every detector row
          interpolate column differentiation results at detector row edge
          differentiate row interpolation values at row edges
          accumulate the differentiation value to the corresponding element in projection
          array
        end for
      end for
    end of GPU thread calculation
  end for
end for
weight projection by lengths of intersection through the slab
end kernel

```

The multislice 3-D backprojection is also computed in a similar fashion on GPU devices using the CUDA programming language. In our implementation, a single thread computes the pre-accumulated partial voxel value for every flattened slab. The projection values are mapped into

texture memory for backprojection as well. Use of GPU texture memory (cache on-chip read-only memory) provides us fast read-only access and computationally efficient bilinear interpolation. The accumulation step is computed separately after all the partial pre-accumulated values for each voxel are gathered on the CPU from multiple GPU devices.

A basic pseudocode of the 3-D implementation of backprojection described in Algorithm 3.6

Algorithm 3.6 GPU implementation of branchless distance-driven backprojection
<p>Number of GPU threads launched = Number of views within 1st quarter rotation × number of slabs in the accumulated image × number of quarter rotations assigned to each GPU</p> <p>begin GPU kernel</p> <p>for all GPU blocks in parallel do</p> <p> for all threads in a block do</p> <p> begin GPU thread calculation</p> <p> weight projection by lengths of intersection through the slab</p> <p> for each detector column</p> <p> determine if the channel contribution to the slab is nonzero</p> <p> for every detector row</p> <p> adjoint differentiate the corresponding element in projection array in the row direction</p> <p> interpolate results for corresponding row edge</p> <p> end for</p> <p> interpolate results for the last row edge</p> <p> adjoint differentiate for corresponding detector column edge for all relevant column edges</p> <p> interpolate result for corresponding detector column edge to slab</p> <p> end for</p> <p> interpolate result for last detector column edge to slab</p> <p> end of GPU thread calculation</p> <p> end for</p> <p>end for</p> <p>end kernel</p> <p>perform 2-D adjoint accumulation for every quarter according to equation (3.62)</p> <p>sum the four adjoint accumulation images.</p>

3.5 Multi-GPU Implementation of Branchless Distance-driven Projectors

Each GPU is assigned a contiguous group of projections whose cardinality is a multiple of the number of views in a quarter rotation. This design allows for theoretically perfect load balancing (in the absence of memory-related latencies) during forward and backprojection since each GPU essentially makes use of the same number of nonzero $h(y|x)$ elements. The full-sized accumulation images and the projection data corresponding to each subset are stored in GPU Global memory. In our approach, we systematically add slices with minimal synchronization overhead between the devices. We have also determined the maximum block size that can be summed concurrently by all devices.

Forward projection is straightforward in terms of global memory access, since each device stores values in separate portions of the projection data array, and access to the accumulation image is read-only. However, if we were to perform backprojection directly into the full-sized accumulation images, we would have serious memory contention issues since multiple devices would be writing to the same array elements simultaneously. Instead, each device performs backprojection to its own private accumulation image arrays (of reduced size compared to the full-sized arrays). This eliminates any need for synchronization during the backprojection of a device's set of views. Once each device is done backprojecting its set of views, the partial accumulation image arrays are summed into the full-sized accumulation image arrays. Fig. 3.6 illustrates the process by which non-overlapping groups of slices from each partial array can be added simultaneously without

memory contention. After each block, a barrier synchronization construct is used to ensure each device has finished summing the current block of slices to the full-sized arrays.

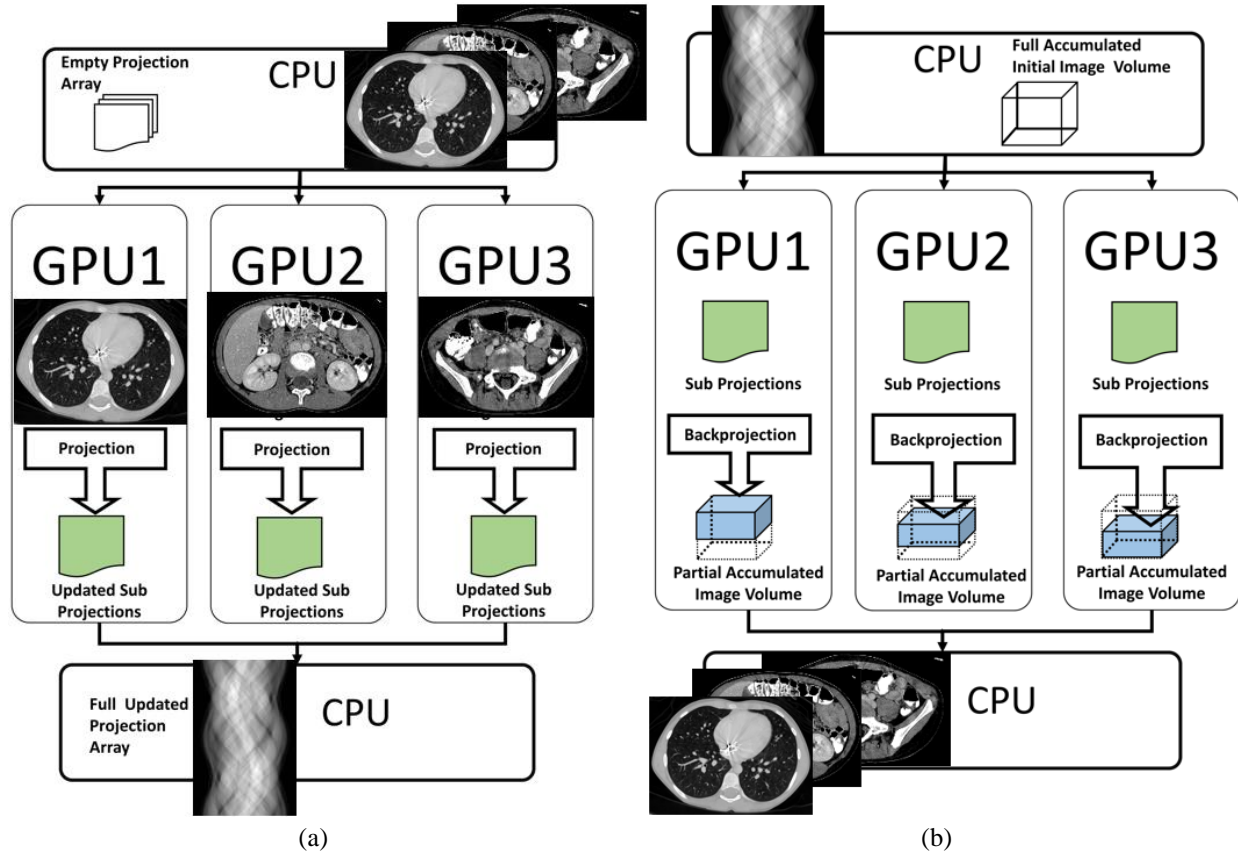


Fig. 3.6 (a) Schematic representation of Multi-GPU implementation of branchless DD projection. (b) Schematic representation of Multi-GPU implementation of branchless DD backprojection.

However, these two approaches create the following constraints on several parameters as follows:

- Total number of views must be a multiple of the number of views in the one-quarter rotation.
- Total number of quarter rotations must be a multiple of the number of GPU devices.
- The number of subsets must divide into the number of views per quarter rotation evenly.

For measured data where these constraints were not satisfied, we pad the measured sinograms with zeros to increase the number of views.

To minimize the overhead time that occurs in data copying, kernel launch, etc., we create the same number of CPU threads as the number of GPUs to be utilized. Each of the threads interacts with an individual GPU. Each of them copies input data from the CPU to the GPU, executes the kernel, and copies results back to the CPU. The host CPU waits for all GPU devices to complete and merges results into one.

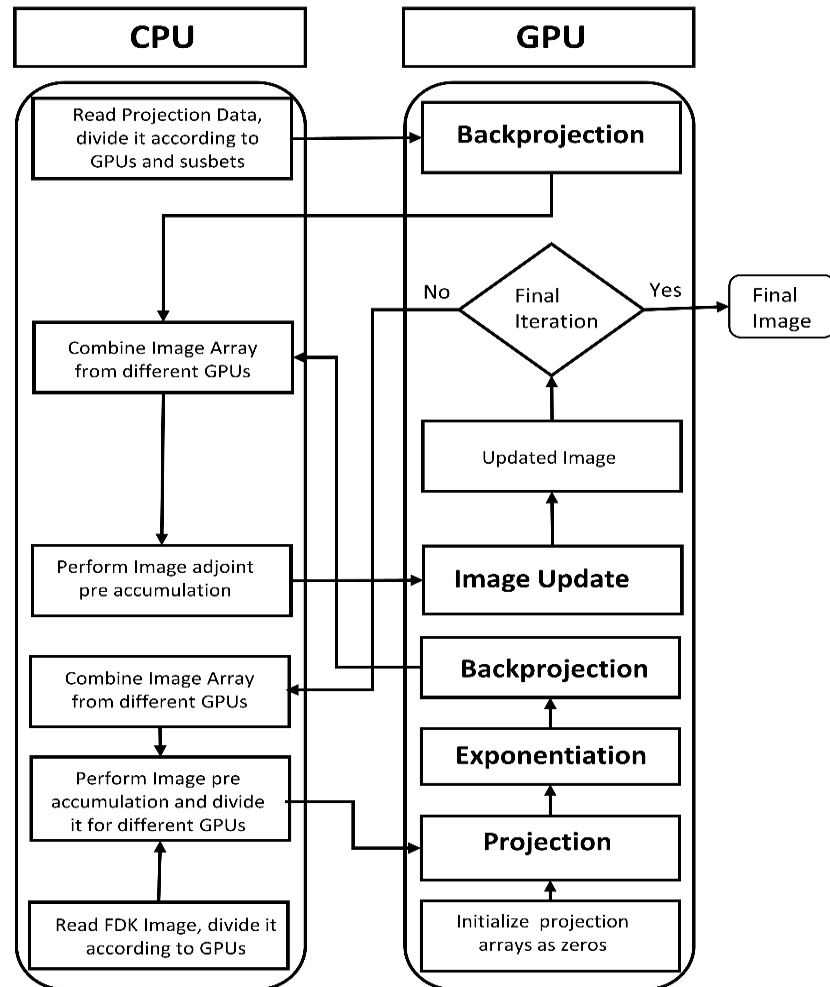


Fig. 3.7 Schematic representation of iterative algorithm execution between CPU and GPU devices

3.6 Experiments

We implemented our multi-threaded CPU algorithm using OpenMP, an industry-standard parallel computing library designed for shared memory systems. The C code was compiled using the Intel Compiler 18.0 with certain optimizations enabled. The code was run on an 8-core Intel *i7* – 5960X (3.0 GHz, 1333 MHz front-side bus) with 64 GB RAM (1.2 GHz). The operating system running on this machine was Microsoft Windows 7. For our multi-GPU implementation, we used 3 NVIDIA GeForce GTX TITAN X GPUs.

Phantom and clinical data were acquired from a Siemens Somatom Sensation 16 scanner (Siemens Medical Solutions, Forchheim, Germany) without using the flying focal spot mode. The scanner acquires 1160 views per rotation, using a $16 \text{ row} \times 672 \text{ channel}$ curved detector array. The distance between the source and isocenter is 570 mm, and the distance between the source and detector is 1040 mm. Data for the clinical abdominal scan and phantom scan were collected from 12 gantry rotations with pitch = 1.0 and $16 \times 1.5 \text{ mm}$ collimation at isocenter. The size of the reconstructed images is $512 \times 512 \times 164$ with $1.0 \times 1.0 \times 1.0 \text{ mm}$ voxels.

Using data from both clinical abdominal scan and phantom scan, we performed a reconstruction using AM reconstruction without ordered subset (OS) and a various number of OS configuration e.g. 5 OS, 29 OS, and 145 OS. The maximum number of allowable subsets for our implementation is 290 which was determined by the number of views or measurements per quarter rotation. However, this choice of OS produces only one view per quarter rotation which was deemed too

aggressive as it resulted in some unwanted image artifacts. In order to accelerate the convergence of our AM algorithm, we have initialized the AM iterations using multi-GPU implementation of the helical FDK image. The fast-parallel multi-GPU implementation scheme for helical FDK reconstruction is presented in the subsequent chapter.

Finally, we performed timing tests using the full-scale abdominal dataset to quantify the performance of our computational approach on the clinically-sized dataset. The reconstruction was done using one iteration of AM without ordered subsets, and one iteration of 5 OS and 29 OS. In the OS cases, the image update was performed for subsets which had an impact on timing performance due to the need for more pre-accumulation and more frequent synchronization. To generate a speedup bar representation a baseline serial version of AM algorithm was written and compiled without OpenMP.

To compare both time performance and image quality, we start with an Intel Core *i7 – 5960X* with 8 cores, 16 hyper-threads, clocked at 3 GHz, with 20 MB caches and 64 GB of memory. For our GPU implementation, we used GeForce GTX TITAN X. TITAN X is based on Maxwell architecture with 3072 CUDA cores and 24 streaming multiprocessors (SMs) running at 1.2 GHz. Each block contains 65536 registers and 48 KiB of shared memory. Some of the highlights of TITAN X hardware are shown in Table 3.1.

Single precision	7.468 TeraFLOP/s
Double precision	233.376 GigaFLOP/s
Multiprocessors	24
Clock rate	1.216 GHz
Global Memory bandwidth	336.48 GB/s
L2 Cache size	3MiB
CUDA cores	3072
Shared memory per block	48KiB

Table 3.1 Hardware specification of TITAN X

3.7 Results

3.7.1 Ordered Subsets

The use of ordered subsets has a significant advantage in increasing the convergence rate. It should be noted that OS implementation is not guaranteed to converge monotonically with increasing numbers of iterations. So, we could devise an adaptive scheme where we reduce the number of ordered subsets at higher iterations. For our current medical abdominal dataset, our 29-ordered subset tends to converge after about 80-100 iterations. For further improvement in our image reconstruction, we can use AM algorithm without ordered subset for subsequent iterations after 29 OS-AM implementation. We have also observed that higher number of OS generates more overhead computation due to GPU device synchronization, image volume pre-accumulation, and CPU to GPU memory transfer. The total backprojection volume doesn't change with increase in OS, as a result, the inter-device memory transfer time, and pre-accumulation time increases linearly with increase in the number of OS. This phenomenon is evident in Fig. 3.8. Although higher OS requires higher computation time per iteration, the overall speedup in the acceleration of objective function, PAE, and RMSE convergence rate dominates over increase in per iteration computation time as evident in Fig. 3.12, 3.13, 3.14, 3.15 and 3.18.

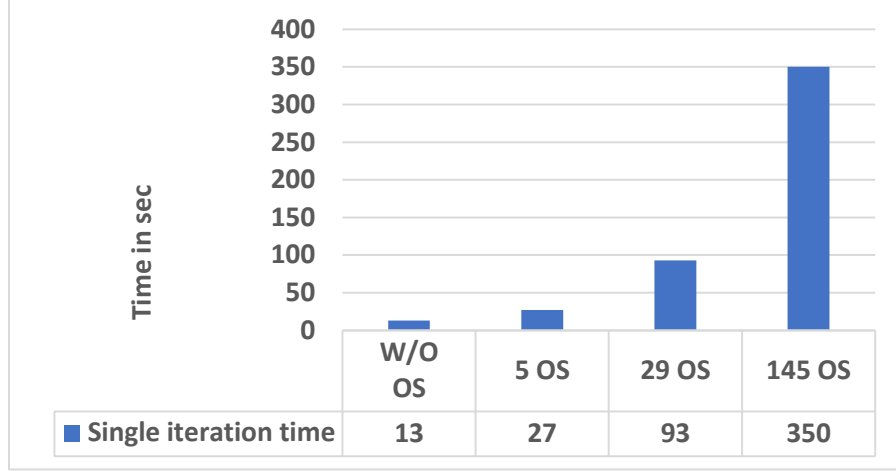


Fig. 3.8 Single iteration time for different OS using 3 TITAN X GPUs in parallel

3.7.2 Phantom

To generate synthetic sinogram from the NCAT phantom image volume, we use the MATLAB 2017b `poissrnd` function. Noisy photon count data were generated by sampling a Poisson pdf with data mean given by $g(y; \mu)$ from equation (3.1) where we have ignored the background intensity $\beta(y)$. The parameters of the measured data and reconstructed images are shown in Table 3.2. The incident photon incident was considered to be 10000 for all measurement views.

No. of views	13920
No. of detector channels	672
No. of detector rows	16
No. of image slices	164
No. of pixels/slice	512x512

Table 3.2 Parameters of measured data and image

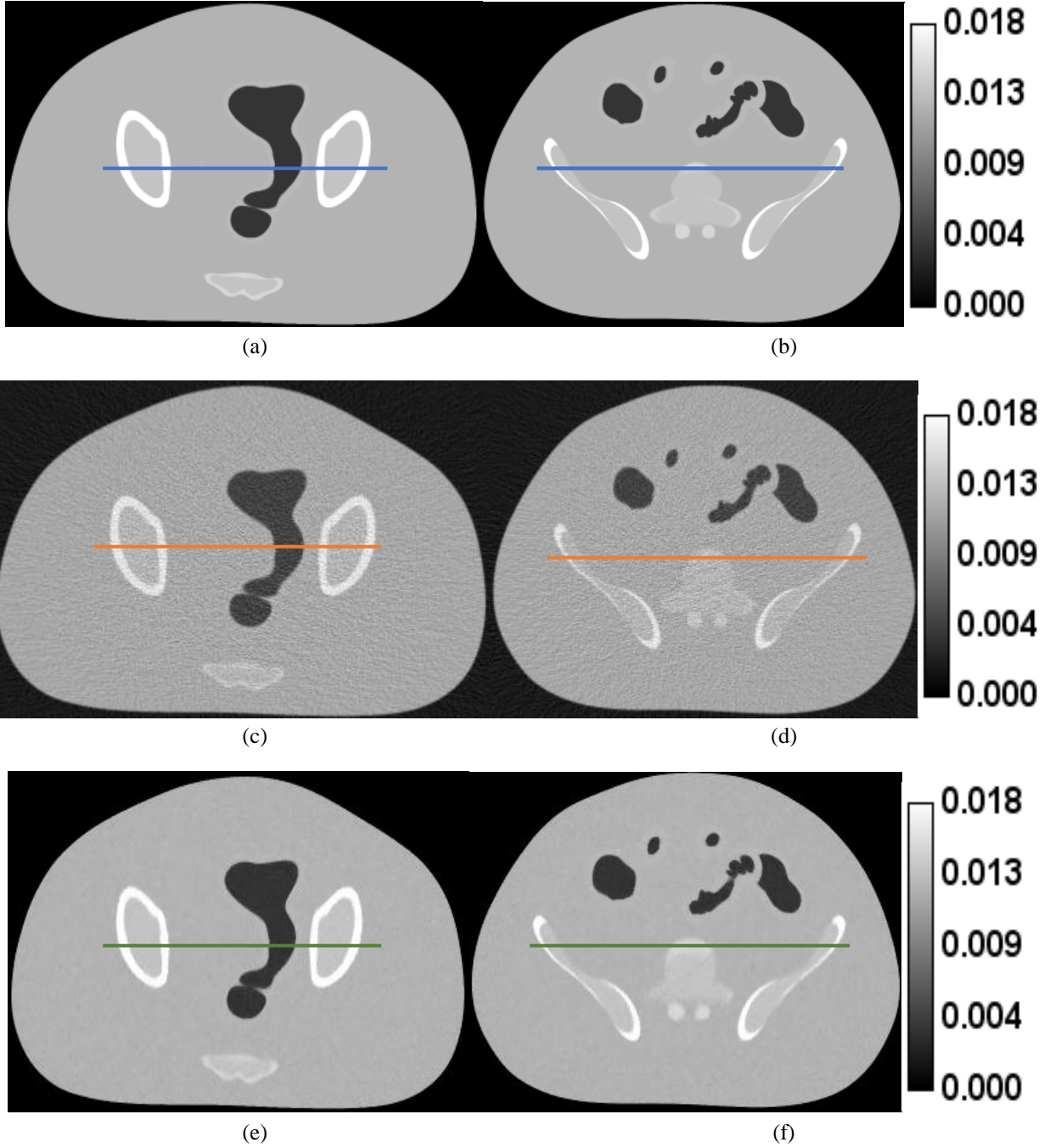


Fig. 3.9 NCAT phantom reconstruction with voxel size = $1.0 \times 1.0 \times 1.0$ mm. Scan parameters: pitch 1.0, 16×1.5 mm collimation, display window width = 0.01759 mm^{-1} , center = 0.008795 mm^{-1} . (a), (b) Axial slices of the actual phantom. (c), (d) Axial slices of the FDK reconstruction of the phantom with added sinogram noise. (e) and (f) Axial slices of the phantom reconstructed with 10 iterations with 145 ordered subsets and with added noise in sinogram domain.

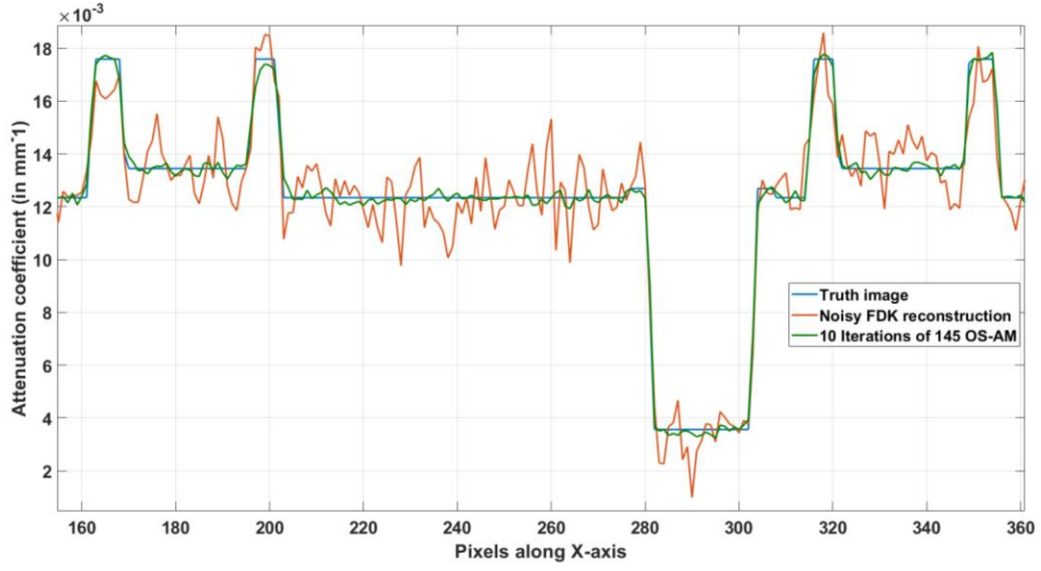


Fig. 3.10 Horizontal profile for different reconstruction images along different lines shown in Figs. 3.9 (a), (c), and (e)

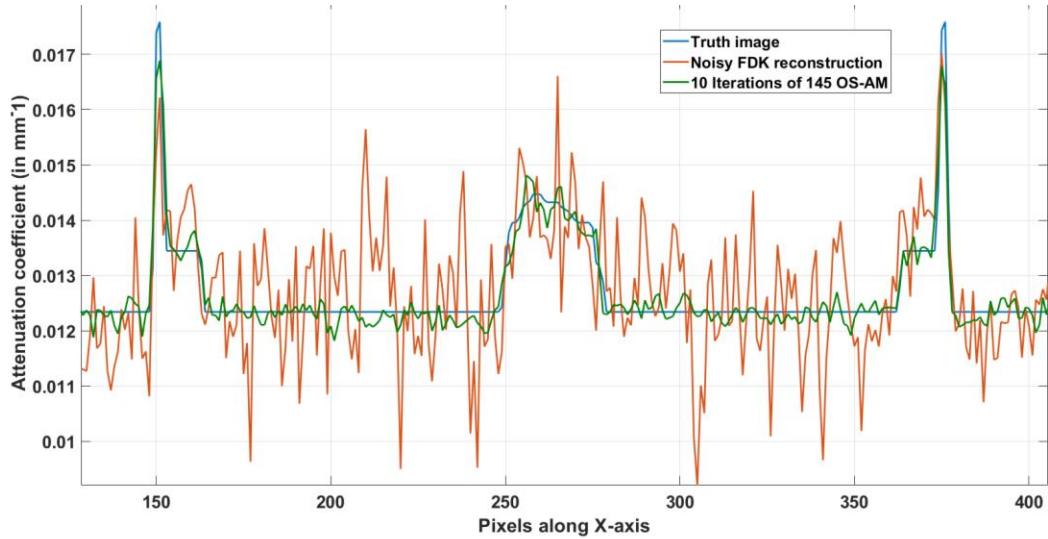


Fig. 3.11 Horizontal profile for different reconstruction images along different lines shown in Figs. 3.9 (b), (d), and (f)

To quantify the effects of the mismatch between the algorithm and the data models, the following quantities were measured on the reconstructed images. In the following definition, N denotes the total number of voxels in the image volume, $\hat{\mu}^{(k)}(x)$ is the reconstructed image, $\hat{\mu}^{true}(x)$ is the

phantom image from which the synthetic projection data were generated. This measure is termed as Percent absolute error (PAE):

$$\text{PAE} = 100 \times \frac{1}{N} \sum_{x=1}^N \left| \frac{\hat{\mu}^{(k)}(x)}{\hat{\mu}^{true}(x)} - 1 \right|. \quad (3.62)$$

We use Root mean square error (RMSE), and Signal-to-noise ratio (SNR) as image quality parameter defined as:

$$\text{RMSE} = \sqrt{\frac{1}{N} \sum_{x=1}^N [\hat{\mu}^{true}(x) - \hat{\mu}^{(k)}(x)]^2} \quad (3.63)$$

$$\text{SNR} = 10 \times \log_{10} \left[\frac{\sum_{x=1}^N [\hat{\mu}^{true}(x)]^2}{\sum_{x=1}^N [\hat{\mu}^{true}(x) - \hat{\mu}^{(k)}(x)]^2} \right]. \quad (3.64)$$

We also use Contrast-to-noise ratio (CNR) as an image quality estimate defined as:

$$\text{CNR} = \frac{(\hat{\mu}_s^{(k)} - \hat{\mu}_b^{(k)})}{\hat{\sigma}_b^{(k)}}, \quad (3.65)$$

where $\hat{\mu}_s^{(k)}$ is the mean attenuation coefficient of a defined structure in the region of interest, $\hat{\mu}_b^{(k)}$ is the mean attenuation coefficient of the image background surrounding the structure, and $\hat{\sigma}_b^{(k)}$ is the standard deviation of the noise calculated from the pixel values outside of the targeted region of interest. The structure of phantom used for this analysis is denoted by the green dotted line in Fig. 3.9 (a). Pixels surrounding this structure is considered as background.

However, for real data, there is no true image that can be used to calculate the image quality parameters discussed before. Instead, we use the total value of the objective function from equation (3.5) as our performance measure.

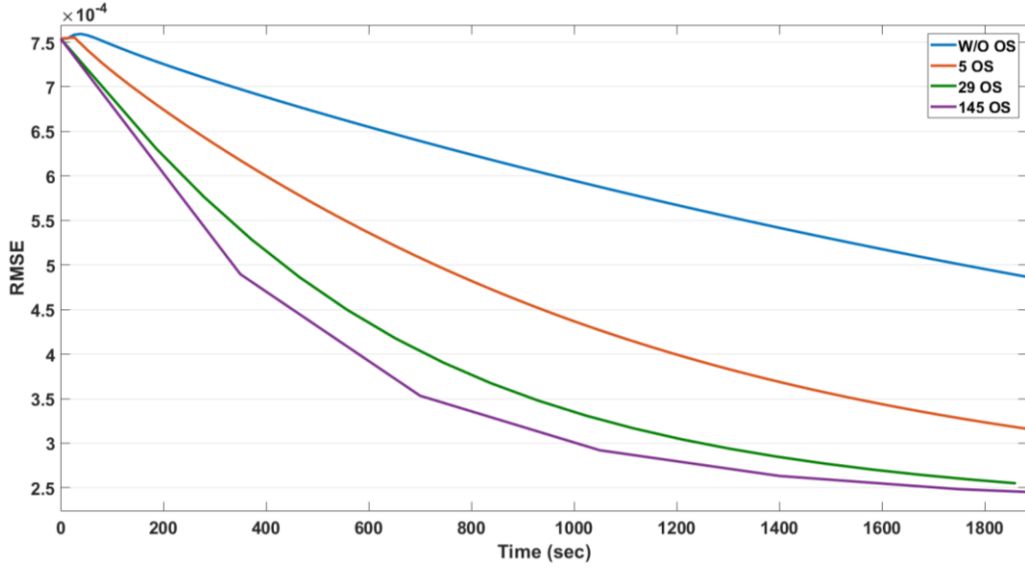


Fig. 3.12 RMSE vs total reconstruction time for different OS configuration using 3 TITAN X GPUs

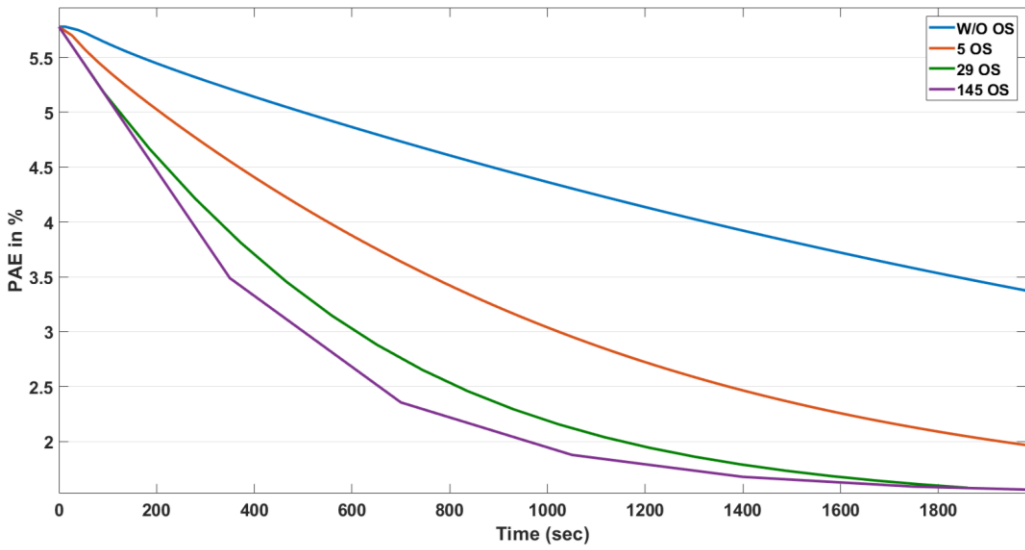


Fig. 3.13 PAE in percentage vs total reconstruction time for different OS configuration using 3 TITAN X GPUs

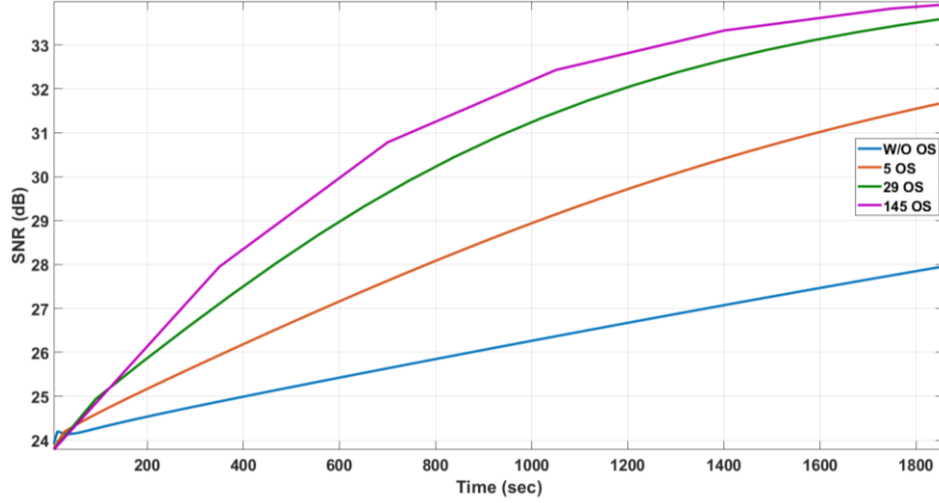


Fig. 3.14 SNR in dB vs total reconstruction time for different OS configuration using 3 TITAN X GPUs

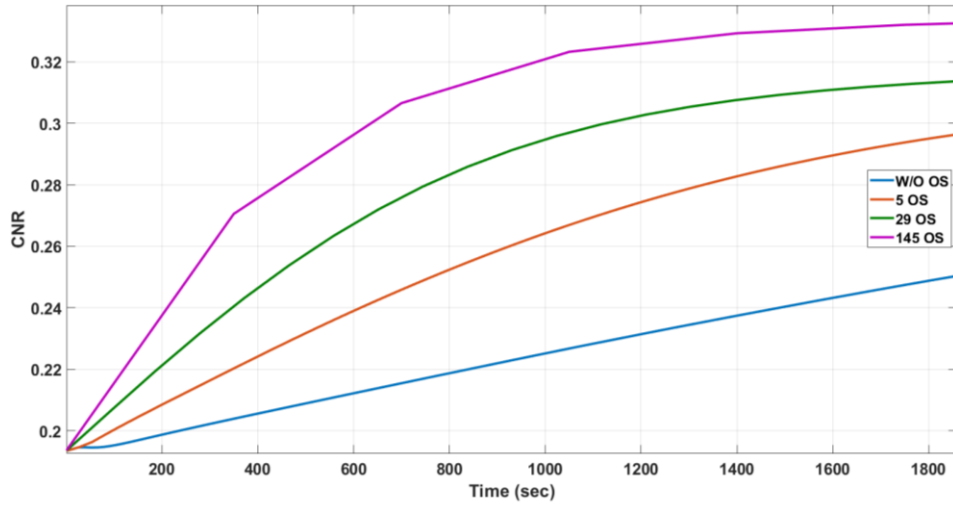


Fig. 3.15 CNR vs total reconstruction time for different OS configuration using 3 TITAN X GPUs

3.7.3 Clinical Datasets

Fig. 3.16 shows axial, coronal, and sagittal views of the abdominal images reconstructed using 10 iterations of 145 OS AM algorithm with regularization parameters: $\lambda = 100$, and $\delta = 0.0002$. The sinogram data used in this reconstruction was obtained from Siemens Sensation 16 scanner at 90 kVp.

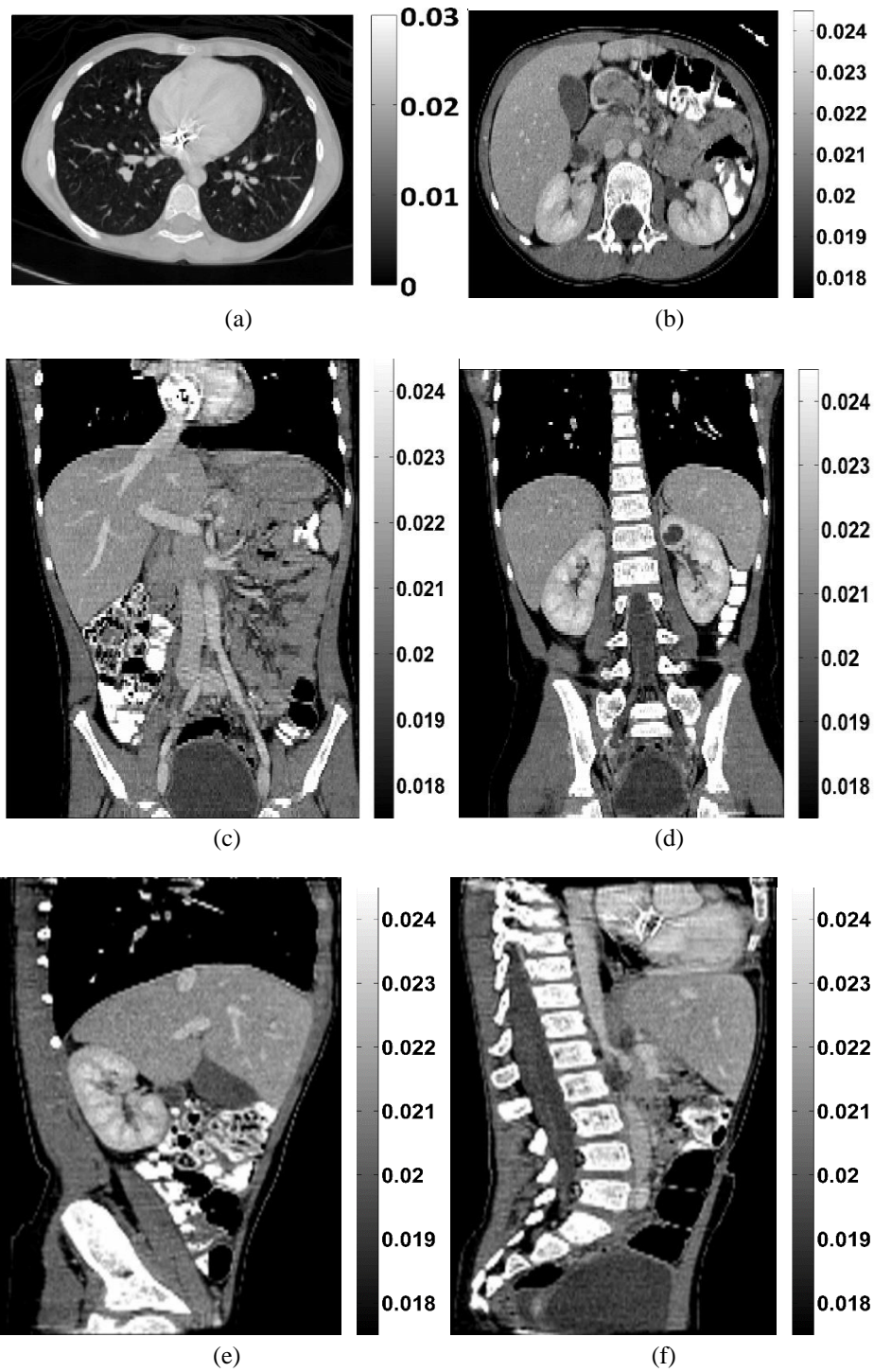


Fig. 3.16 Regularized AM reconstruction using 10 iterations of 145 ordered subsets. Voxel size = $1.0 \times 1.0 \times 1.0$ mm. Scan parameters: 180 mAs, pitch 1.0, 16×1.5 mm collimation. (a) Axial slice of lung with display window width = 0.03 mm^{-1} , center = 0.015 mm^{-1} . (b) Axial slice of abdomen with display window width = 0.007 mm^{-1} , center = 0.021 mm^{-1} . (c) and (d) are coronal views and (e) and (f) are sagittal views with display window width = 0.007 mm^{-1} , center = 0.021 mm^{-1} .

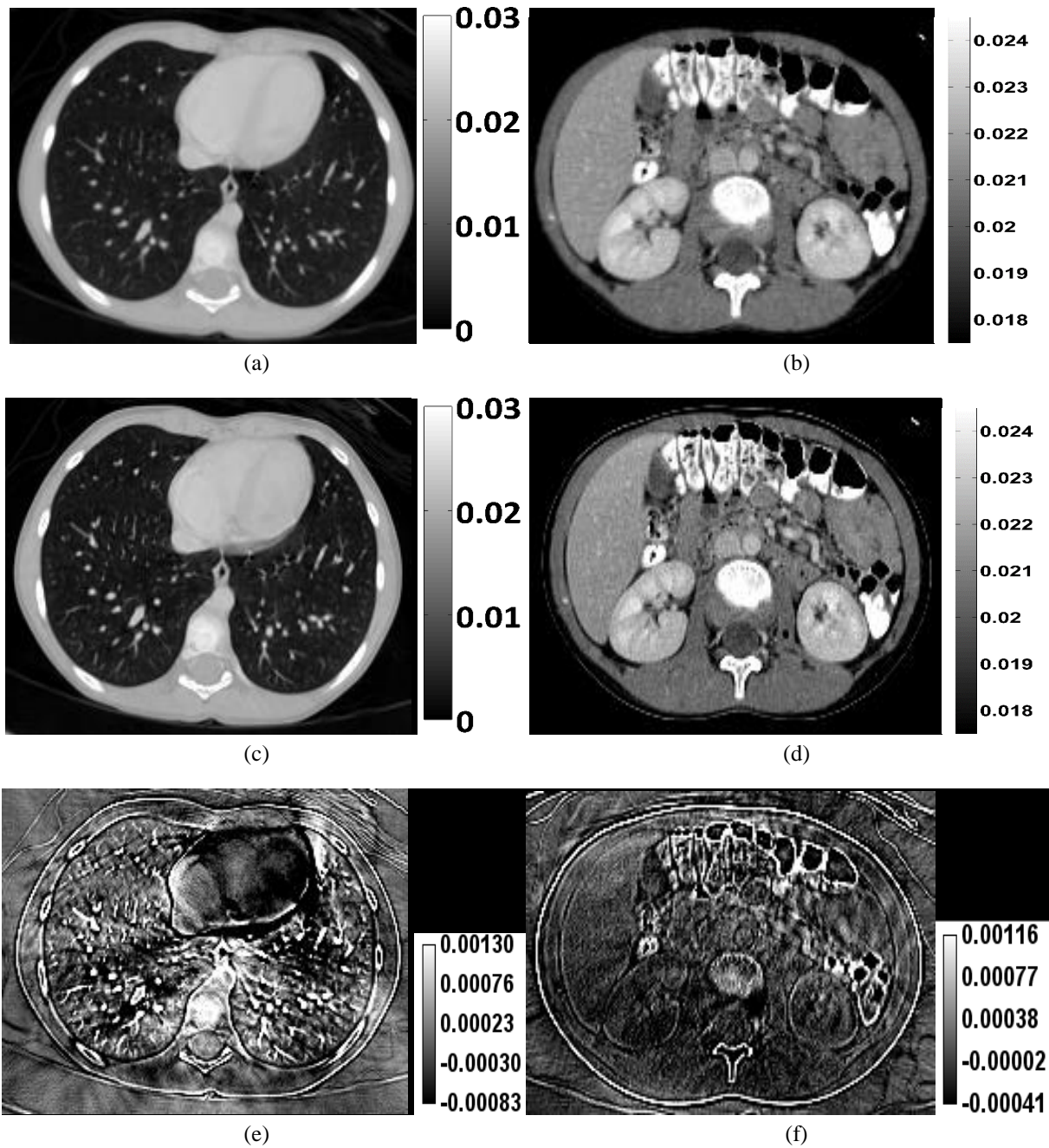


Fig. 3.17 Regularized AM reconstruction of lung and abdominal slices using 3 TITAN X GPUs. Voxel size = $1.0 \times 1.0 \times 1.0$ mm. Scan parameters: 180 mAs, pitch 1.0, 16×1.5 mm collimation. Axial slice of the lung with display window width = 0.03 mm^{-1} , center = 0.015 mm^{-1} , reconstructed with (a) FDK and (c) 10 iterations of AM using 145 OS. Axial slice of the abdomen with display window width = 0.007 mm^{-1} , center = 0.021 mm^{-1} , reconstructed with (b) FDK and (d) 10 iterations of AM using 145 OS. (e) and (f) are difference images between FDK and 10 AM iteration using 145 OS corresponding to lung and abdomen slices respectively.

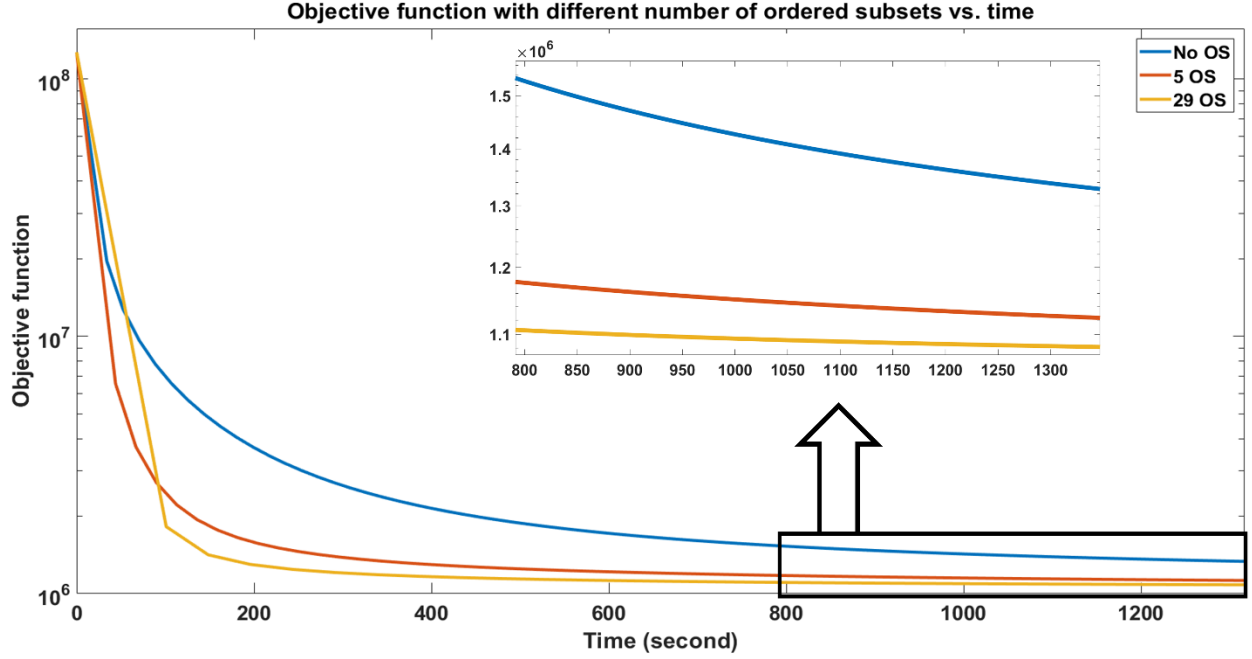


Fig. 3.18 Plot of I-divergence vs computation time for different ordered subset configurations by using 3 GPUs in parallel.

From Fig. 3.16 we can clearly conclude that AM algorithm improves image resolution and enhance edges. We can also observe the lung and heart motion from Fig. 3.17 (a) and (c). Lung nodules are more prominent using our iterative reconstruction approach which can lead to better diagnosis of tumors presents in lungs.

3.7.4 Timing Performance

We have used abdominal dataset as a benchmark for determining the timing performance of our multi-threaded CPU and multi-GPU implementation. The wall clock time to run one iteration of AM algorithm without ordered subset on a standalone CPU core without multi-threading was 433 seconds for projection and 435 seconds for backprojection with a total time of 882 seconds. On the other hand, if we compiled the code with OpenMP using 8 cores with 2 hyperthreads per core,

the total time for a single iteration is reduced to 190 seconds. Using the Intel Thread Profiler, we have determined that in case of our multi-threaded CPU implementation, 96.2% of the execution time was in parallel while the rest was spent in barrier synchronization of different threads. This profiler result confirms the efficacy of our load balancing scheme within each iteration.

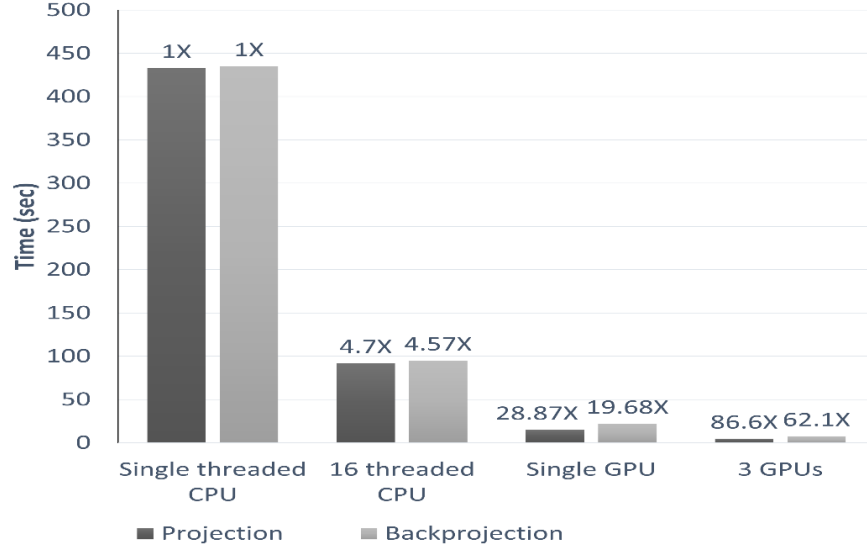


Fig. 3.19 Acceleration of our multi-GPU implementation for complete clinically-sized data

Operations	Execution Time (seconds)			
	Single-threaded CPU	16-threaded CPU	Single-GPU	Multi-GPU
Pre- accumulation	8.1	1.7	0.570	0.21
Projection	433	92	15	4.7
Exponentiation	1.1	0.25	0.07	0.029
Backprojection	435	95	22	7.6
Image Update	4.8	1.2	0.17	0.06
Total	882	190.15	37.81	12.6

Table 3.3 Reconstruction times using clinically-sized data and no OS for different CPU and GPU hardware architectures.

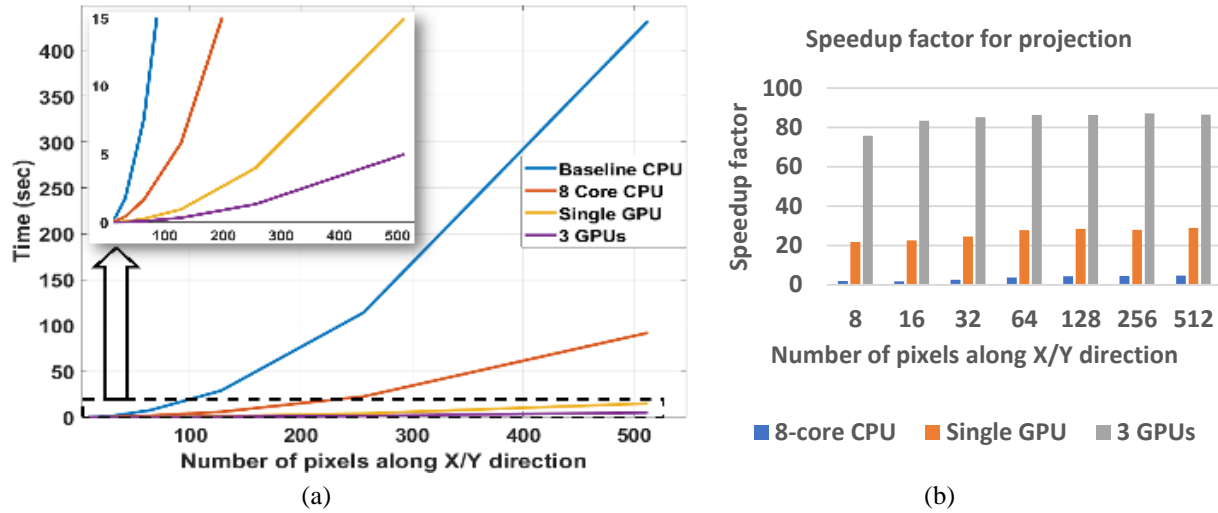


Fig. 3.20 (a) Forward projection computational times and (b) overall speedup for a different number of pixels along X/Y direction using different hardware configurations.

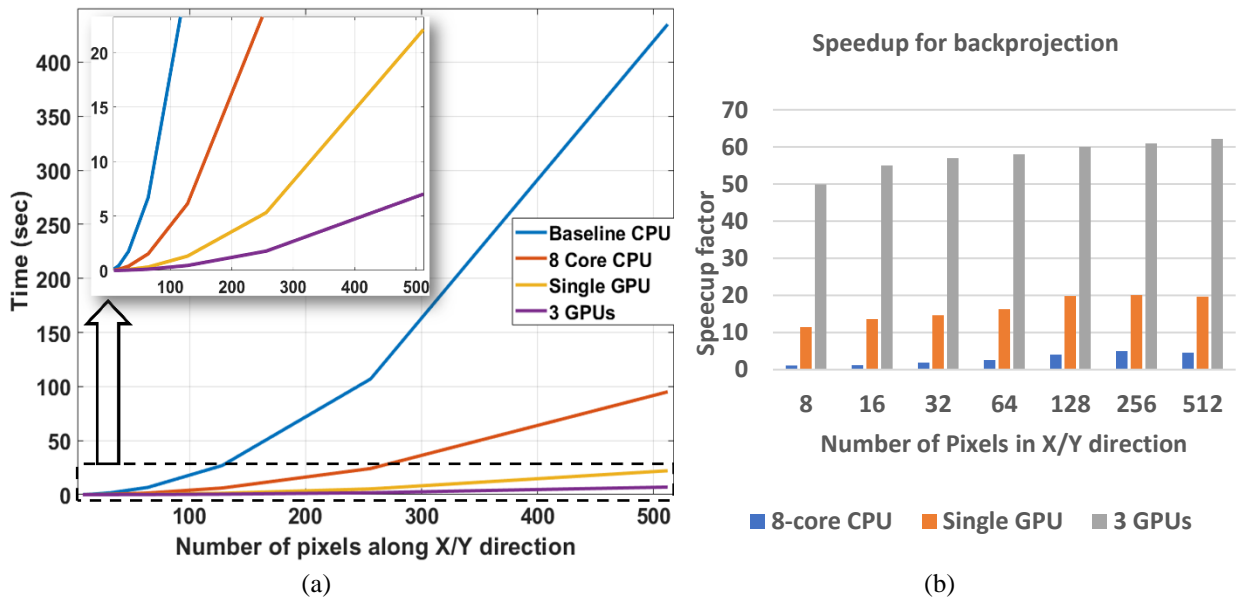


Fig. 3.21 (a) Backprojection computational times and (b) overall speedup for a different number of pixels along X/Y direction using different hardware configurations.

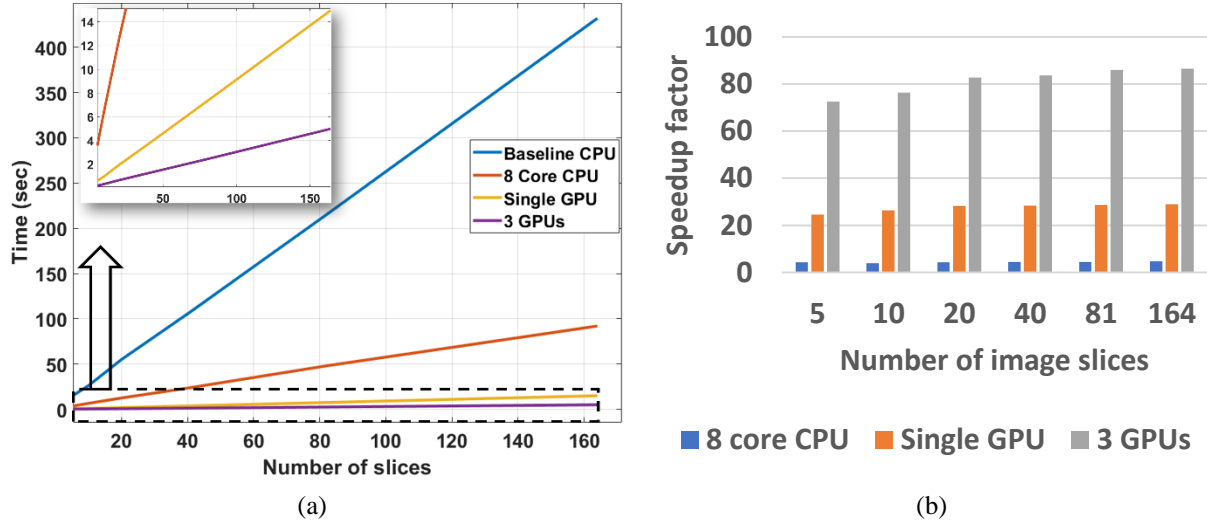


Fig. 3.22 (a) Forward projection computational times and (b) overall speedup for different number of image slices using different hardware configurations

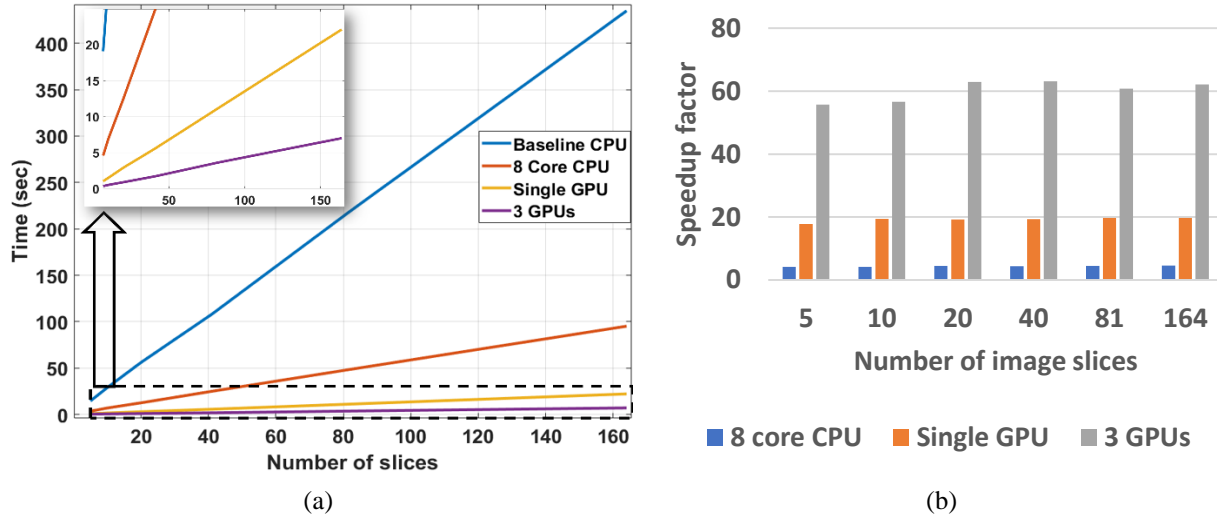


Fig. 3.23 (a) Backprojection computational times and (b) overall speedup for different number of image slices using different hardware configurations

The leftmost bar in Fig. 3.19 is the execution time of the baseline serial version and the remaining bars show runtimes for the specific optimizations using multiple CPU threads and multiple GPU devices. Table 3.3 shows the time of execution of each component of our algorithm with different

hardware configurations. For the baseline serial version, we ran our projector algorithms on a single CPU core with nested for loops representing the parallel GPU threads. For multithreaded CPU implementation, each CPU core launches 2 hyper-threads for every logical processor in the core. Each hyper-thread basically acts as a standalone GPU device. Instead of parallel GPU threads, we used a corresponding number of nested for loops. We also used a barrier synchronization to wait for every CPU thread to finish its projection and backprojection in their private projection and image accumulation arrays respectively. To calculate the parallelization efficiency of the multi-threaded CPU version we define our speedup ratio according to Amdahl's law as follows

$$S = \frac{T_1}{T_N} < \frac{1}{(f + \frac{1-f}{N})} < \frac{1}{f} \text{ as } N \rightarrow \infty, \quad (3.63)$$

where, T_1 and T_N are elapsed times of 1 and N workers. f is the fraction of the code that is not parallelizable. The parallel efficiency is then defined as,

$$E = S/N. \quad (3.64)$$

From our experimentation with $N = 16$ CPU threads, we get $S = T_1/T_N = 4.7$ for the projection operation. As a result, $f = 0.1603$ and parallel efficiency is $E = 0.2963$. So, we can conclude, our multi-threaded CPU implementation can achieve a maximum speedup of 6.2 times for the projection operation for the clinically-sized dataset.

3.8 Discussion

We have observed from Fig. 3.19 that using multiple GPUs to reconstruct images gives us better performance in computational cost compared to our best available CPU configuration. Our primary contribution is a novel approach of pre-accumulation for projection (see equation 3.56) and adjoint pre-accumulation for backprojection (see equation 3.62) in the setting of the three-dimensional branchless DD algorithm. We observe that computing times linearly decrease with increasing the number of GPUs. Since we can divide the projection array according to its number of ordered subsets and the number of GPU devices available, the effective size of the projection array we pass to the GPUs is much smaller than the size of the partial image accumulation array. As a result, the backprojection operator accumulates and write the result on a much bigger image accumulation array than the projection array is read from. So, the time required for backprojection is higher than for forward projection. The difference is much more significant when we use more ordered subsets since the number of subsets only reduces the volume of projection array keeping the size of partial accumulation array unchanged.

The time needed to combine partial image accumulation arrays from different GPU devices after every backprojection increases the iteration time for ordered subset configurations. For ordered subset implementation, we also need to perform measured data backprojection after every subset iteration since all the measured data backprojection arrays for every subset cannot be saved in our device memory. In Figure 3.18, we show the change in objective function values (defined in equation 3.6) with iteration number for various ordered subset configurations. Since minimizing the objective function values will maximize the log-likelihood between the measured data and our

estimated data by the model, we can use this distance method to estimate the accuracy and noise reduction of our reconstruction. The objective function value at 0th iteration of Fig. 3.18 denotes the value of the objective function between measured data and projection sinogram of FDK reconstruction of the data. The significant decrease in the objective function values clearly illustrates the improvement in image quality with our proposed reconstruction algorithm. In the end, we can clearly conclude that our optimizations are effective and that our multi-GPU approach is beneficial for both forward and backprojection cases.

For the calculation of speedup using different hardware configurations and different scan geometries, the single-threaded CPU implementation was considered as the baseline. We observed from Fig. 3.20 and Fig. 3.21 that computational time for both multithreaded CPU and GPU configurations increase quadratically w.r.t. baseline CPU implementation for a different number of pixels along X/Y direction. The number of pixels along X/Y direction determines the size of the flattened slab. The amount of computation for every GPU thread launched is directly proportional to the size of the flattened slab. As a result, the computational time increases quadratically with the number of pixels along the perpendicular dimensions of the slab. However, the speedup is small for small image volume due to overhead for data transfer between CPU and GPU. As the image volume increases, the relative contribution of the overhead is decreased and the actual computation time of projection and backprojection kernel dominates. Thus, we observe a steady increase in the speedup factor with increasing image volume.

The computation time increases linearly with the number of views and the number of image slices as seen in Fig. 3.22 and Fig. 3.23. The slow initial speedup can again be attributed due to overhead for data transfer between CPU and GPU. So, the speedup factor increases slightly with increasing number of image slices. For the brevity of this thesis, we have only shown the computational time and speedup factor for the variation of image slices. Since the number of minimum image slice is directly proportional to the number of views, we can observe a similar trend if we varied the number of views.

We can expect to reduce run times with more GPUs (see Fig. 3.19), which opens the door to exciting new possibilities in clinical settings. For precision critical applications we can use the double precision floating-point arithmetic with TITAN Z GPUs, with some performance degradation compared to our single precision TITAN X GPUs.

Chapter 4: Multislice Analytical Helical CT Reconstruction Using GPU

In this chapter, we present the details of the efficient fully 3-D reconstruction framework using an analytical method. The main motivation for the multi-GPU implementation of analytical reconstruction is twofold. Firstly, the voxel-driven analytical reconstruction approach can be easily parallelized over multiple GPU threads and across multiple GPU devices. As a result, the total reconstruction time for a clinically sized data is < 2 seconds using 3 TITAN X GPUs in parallel. Naturally, we can use the images reconstructed using analytical methods as the initial input for our iterative reconstruction problem. This approach accelerates the convergence rate of our SIR algorithms. Secondly, we can use these algorithms to calculate aggressive update step described in Chapter 5. On average, this aggressive update step method reduces the total computation time by 50% without adding any significant computational burden.

The structure of this chapter is described as follows: Section 4.1 describes the scanner geometry and Feldkamp-Davis-Kress (FDK) algorithm overview. Section 4.2 describes the changes we have proposed to cone-beam geometry to make it more amenable to multi-GPU based parallelization. Section 4.3 describes our fast-parallel multi-GPU based implementation of the FDK algorithm. Section 4.4 describes the experiments we have conducted to demonstrate the improvement in performance of our parallel implementation.

4.1 Theory

Within the class of analytical reconstruction algorithms, there is a further dichotomy between so-called exact algorithms and approximate algorithms. For exact reconstruction, many techniques based on PI-lines have been developed [67-69]. There are also newer approaches that use differentiated backprojection onto PI-lines or other lines, which require the subsequent application of the inverse Hilbert transform and interpolation [70].

The approximate reconstruction algorithms are not mathematically exact and therefore suffer from cone-beam and windmill artifacts in the presence of high-contrast objects, which worsen with increasing z-direction distance from the central transverse plane. However, these algorithms offer more practical implementations and can more readily incorporate redundant data into the reconstruction (for better dose utilization). Among these, the Adaptive multiple plane reconstruction (AMPR) method rebins the data into oblique planes that best fit the helix, upon which 2-D FBP is performed; the reconstructed tilted slices are then interpolated in the z-direction to form an image volume with uniform spacing [71]. Helical FDK algorithms form another class of approximate methods, in which a voxel- and view-dependent weighting function is applied in the process of performing 3-D backprojection; this weight normalizes the contribution from redundant data [72-74]. The algorithms differ in terms of dose utilization, redundancy weighting function, and whether the algorithm operates in the native geometry or a rebinned geometry, etc. We present a similar multi-GPU implementation of the (3-D)-weighted cone-beam filtered backprojection algorithm published by Tang et al. [75]. The major aspects of the algorithm and

other pertinent references can be found in that paper. In this section, some of the specifics will be addressed.

The geometry of our helical multislice CT scanner is described in Fig. 3.1 in Chapter 3. A key element of most helical FDK algorithms (including this one) is that a fixed angular interval along the helix is chosen to reconstruct each slice. In other words, reconstruction of a slice at $z = z_0$ is done by backprojecting a symmetric set of views on both sides of the slice in which the center view's z source position $z_{src} = z_0$. The interval is fixed to 2π in our algorithm, which simplifies the redundancy weighting and leads to good image quality.

The overall backprojection expression is given below

$$\mu(x, y, z) = \frac{\pi}{2\theta_m} \int_{-\theta_m}^{\theta_m} \frac{R_f}{\sqrt{R_f^2 + \hat{\eta}^2}} \omega_{3d}(\theta, \hat{t}, \hat{\eta}) \tilde{\rho}(\theta, \hat{t}, \hat{\eta}) d\theta \quad (4.1)$$

where $\omega_{3d}(\theta, \hat{t}, \hat{\eta})$ is the redundancy weighting function, $\tilde{\rho}(\theta, \hat{t}, \hat{\eta})$ is the radially-filtered projection data, and $\theta_m = \pi$. The $(\hat{t}, \hat{\eta})$ coordinates specify the location on the detector where the point (x, y, z) is projected at view angle θ . In this expression, the θ interval is implicitly defined such that $\theta = 0$ intersects the helix at slice location z . In this work, a mapping from cone angle η to linear coordinate v on the detector is used. This is done with respect to the isocenter of the scanner, resulting in the relationship $v = R_f \tan \eta$.

4.2 FDK Reconstruction

4.2.1 Data Preprocessing Operations

The chosen FDK algorithm operates in the cone-parallel geometry. Therefore, the first step is to perform row-wise fan-beam to parallel-beam rebinning, which transforms the data to the correct geometry. Our algorithm uses linear interpolation for the azimuthal and radial resampling operations. A schematic diagram of the helical source trajectory and projection data acquisition in the native cone-beam geometry are shown in Fig. 4.1 (a). The corresponding row-wise fan-beam to the parallel-beam rebinning scheme is depicted in Fig. 4.1 (b).

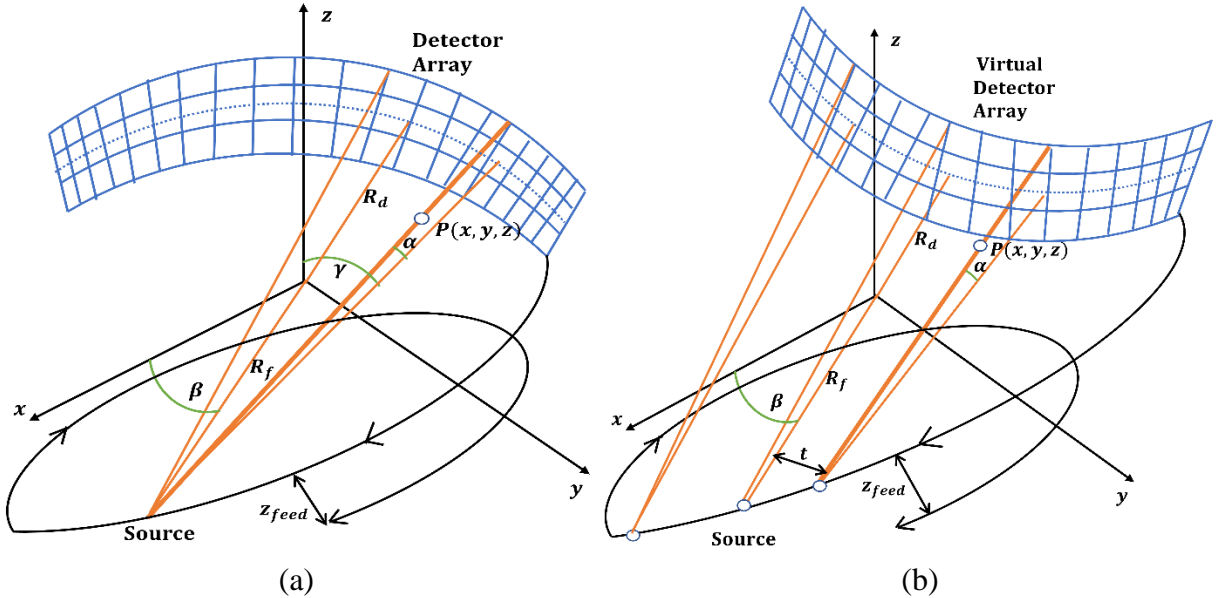


Fig. 4.1 Schematic diagram of the cone beam to parallel fan beam rebinning scheme described in [72]: (a) the native CB geometry; (b) the cone-parallel geometry.

The X-ray source rotates at a radius of R_f and the detector array rotates along the same direction at a radius of R_d from the isocenter. For the point $P(x, y, z)$ on the bold line in Fig. 4.1 (a), β is

the view angle, γ is the fan angle and η is the cone angle. z_{feed} is the axial distance travelled by the patient bed in one complete rotation of the X-ray source detector pair. The ray originating from focal point S and passing through point $P(x, y, z)$ can be uniquely determined in the new cone-parallel rebinned geometry by view angle β , cone angle η and orthogonal distance from iso-ray (namely orthogonal iso-distance) t . In Fig. 4.1 (b) we can also notice that the curvature of the virtual detector array is inverted.

Depending on the particular scan geometry and the choice of θ_m , the backprojection may require data that was not physically measured in the v direction, i.e. $|\hat{\eta}|$ may be greater than the cone angle of the scanner. For the redundancy weights (discussed below) to work properly, all data must be available for the entirety of the backprojection operation. To fulfill this requirement, extrapolation of rows using constant extension is performed as a preprocessing step [76]. In particular, at $(\theta, t) = (\theta_m, 0)$, $z_{src} = \frac{\theta_m z_{feed}}{2\pi}$. This is the farthest z distance from the source to the slice (ignoring the tilt of the cone-parallel projection in z). The largest $|\hat{\eta}|$ value will be obtained at this z_{src} position, and with minimum in-plane source-to-voxel distance $(R_f - R_{FOV})$ due to magnification. Therefore, at isocenter, the physical height of the detector including the required extension can be shown to be

$$H = \frac{\theta_m z_{feed} R_f}{\pi(R_f - R_{FOV})}. \quad (4.2)$$

The ramp filtering procedure can be found in [2]. Two basic frequency-domain apodization windows have been implemented, but certainly, others can be added as needed. The existing windows are:

- Hamming window: $0.54 + 0.46 \cos(\pi\omega)$
- Hann window: $0.5 + 0.5 \cos(\pi\omega)$

where $\omega \in [0, 1]$ is the normalized frequency.

As shown in equation (4.1), cosine weighting for the cone angle is only needed in the v direction, since the X-ray source in the cone-parallel geometry only diverges in that direction. This factor, $R_f / \sqrt{R_f^2 + \hat{\eta}^2}$ is approximated in the preprocessing stage by pre-multiplying the projection data by the cosine of the cone angle for the center v position of each detector row (as opposed to having it remain a voxel-dependent quantity).

4.2.2 Redundancy Weights

Unlike the circular-orbit FDK algorithm, the voxels in a helical scan are not illuminated uniformly from all view directions. Therefore, redundancy weights are needed during backprojection to normalize the contribution of the measurements to each voxel in the image volume. In the case of the 2-D parallel-beam coordinate system, there exists a complementary ray (also known as the conjugate ray) that is co-linear with the primary ray but comes from the opposing view at $\theta + \pi$. Now consider the 3-D cone-parallel projection of a point (x, y, z) ; it will land on the detector at radial coordinate \hat{t} . In the helical geometry, the opposing view will have a z offset due to the

moving source in the z direction. Therefore, the complementary ray is still co-linear with the primary ray when projected onto the xy plane, but in z , they only intersect at the point (x, y, z) . In fact, the cone angle is likely to be different for the primary and complementary ray. With the choice of $\theta_m = \pi$, the 3-D weighting strategy in [75] is to use both the primary and complementary ray in the reconstruction. The ray with the smaller cone angle is weighted more heavily, as that should reduce the cone angle artifacts in the reconstruction. Similarly, the ray whose z_{src} position is closer to the slice is weighted more heavily. Using the mapping from η to v , the 3-D weighting function implemented in our algorithm is

$$\omega_{3d}(\theta, t, v) \triangleq \frac{\omega_{2d}(\theta, t)|v_c|^{kh}}{\omega_{2d}(\theta, t)|v_c|^{kh} + \omega_{2d}(\theta_c, t_c)|v|^{kh}}, \quad (4.3)$$

where the subscript c refers to the complementary ray, the kh parameter is currently fixed to 2.0, and

$$\omega_{2d}(\theta, t) \triangleq \begin{cases} 1 + \theta/\pi & \text{if } -\pi \leq \theta < 0 \\ 1 - \theta/\pi & \text{if } 0 < \theta < \pi \end{cases}. \quad (4.4)$$

The in-plane parallel-beam complementary ray coordinates are simply $\theta_c = \theta + \pi$, and $t_c = -t$. The v_c coordinate can also be determined directly from the primary ray coordinates, as will be explained at the end of the next section.

4.2.3 Cone-parallel Backprojection

This section addresses the calculation of (\hat{t}, \hat{v}) based on the cone-parallel projection of point (x, y, z) from view angle θ . Once (\hat{t}, \hat{v}) is known, bilinear interpolation is performed on the discrete 2-D detector array to determine the projection data value.

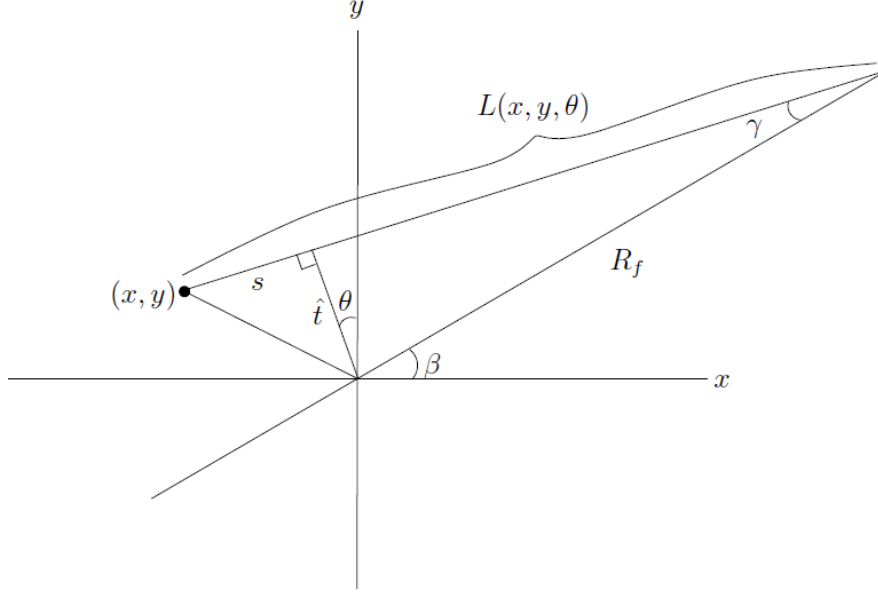


Fig. 4.2 Parallel-beam and fan beam geometry.

Figure 4.2 shows the parallel-beam geometry as well as the fan-beam coordinates for reference.

The relationship between the coordinate systems can be expressed as

$$\beta = \theta + \gamma \quad (4.5)$$

$$\gamma = \sin^{-1}(t/R_f). \quad (4.6)$$

From the Fig. 4.2, it is also possible to calculate \hat{t} and $L(x, y, \theta)$:

$$\hat{t}(x, y, \theta) = y \cos \theta - x \sin \theta \quad (4.7)$$

$$L(x, y, \theta) = \sqrt{R_f^2 - \hat{t}^2} + s \quad (4.8)$$

$$= \sqrt{R_f^2 - \hat{t}^2} - x \cos \theta - y \sin \theta \quad (4.9)$$

The in-plane source-to-voxel length $L(x, y, \theta)$ is used to calculate the projection in the z direction, as shown in Fig. 4.3.

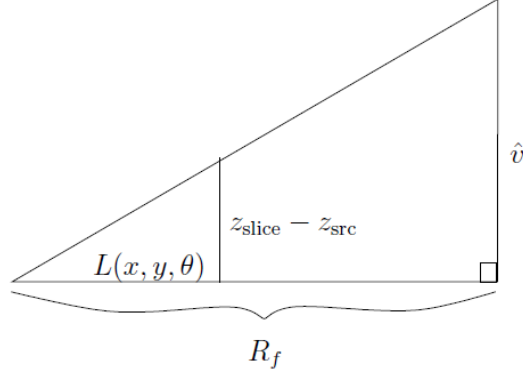


Fig. 4.3 v scaling based on point projection.

The similar triangles allow for the calculation of \hat{v} as:

$$\hat{v}(x, y, z_{slice}, \theta) = \frac{R_f}{L(x, y, \theta)} (z_{slice} - z_{src}). \quad (4.10)$$

In the cone-parallel geometry, the source is distributed along the helix for each projection, thereby giving the projection a tilt in the z direction. z_{src} is therefore a function of θ and t . First consider the native cone-beam geometry, where

$$z_{src}(\beta) = z_{src,0} + \frac{z_{feed}}{2\pi} \beta \quad (4.11)$$

and $z_{src,0}$ is the z source position for the first view of the scan. Replacing θ according to (4.5) and (4.6),

$$z_{src}(\theta, t) = z_{src,0} + \frac{z_{feed}}{2\pi} [\theta + \sin^{-1}(t/R_f)]. \quad (4.12)$$

Note that there are three contexts for the θ variable in this chapter:

- Local angular coordinate centered about z_{slice} and used for z -related calculations, e.g., (4.1).
- Angle used for in-plane calculations, e.g., (4.7), (4.9).

- Global angular coordinate used to keep track of source position in the global z coordinate system.

Our algorithm defines two separate variables as follows:

$$\theta_{axial}(p) \triangleq p|\Delta\theta| \quad (4.13)$$

$$\theta_{in-plane}(p) \triangleq p\Delta\theta + \theta_0 \quad (4.14)$$

where p is the view index, $\Delta\theta$ is the signed view angle spacing, and θ_0 is the starting in-plane angle. The absolute value operator is used in equation (4.13) since z_{src} is defined to increase with increasing view index, regardless of the gantry rotation direction.

For the weighting function $\omega_{3d}(\theta, t, v)$, it was noted that the v_c coordinate must be calculated. This can be obtained in a few steps. First, using Fig. 4.3 again, the chord length along the ray (for a circle of radius R_f) is $\sqrt{R_f^2 - t^2}$. Therefore, the complementary in-plane source-to-voxel length is

$$L_c = 2\sqrt{R_f^2 - t^2} - L. \quad (4.15)$$

From there, it is straightforward to calculate v_c :

$$z_{src,c} = z_{src,0} + \frac{z_{feed}}{2\pi} [\theta_c + \sin^{-1}(t_c/R_f)] \quad (4.16)$$

$$v_c = \frac{R_f}{L_c} (z_{slice} - z_{src,c}). \quad (4.17)$$

4.3 GPU Implementation of FDK Backprojection

The GPU implementation of our backprojection algorithm is shown below

Algorithm 4.1 GPU implementation of FDK backprojection

Number of GPU threads launched = Number of pixels in a slice \times number of slices assigned to each GPU

begin GPU kernel

for all GPU blocks in parallel **do**

for all threads in a block **do**

begin GPU thread calculation

 compute the view range for current reconstruction slice

for every view within our reconstruction slice range

 determine the range of channels contributing to our reconstruction slice

for every channel within our reconstruction slice range

 determine the range of detector rows contributing to our reconstruction slice

for every detector row within reconstruction slice range

 use bilinear interpolation to obtain projection data value at every detector coordinate for the current view

 calculate corresponding redundancy weight

 compute the normalized 3-D weight

 accumulate weighted projection data to current voxel

end for

end for

end for

end of GPU thread calculation

end for

end for

end kernel

4.4 Results

We implemented our multi-threaded CPU algorithm using OpenMP, an industry-standard parallel computing library designed for shared memory systems. The C code was compiled using the Intel Compiler 18.0 with certain optimizations enabled. The code was run on an 8-core Intel *i7* – 5960X (3.0 GHz, 1333 MHz front-side bus) and 64 GB RAM (1.2 GHz). The operating system

running on this machine was Microsoft Windows 7. For our multi-GPU implementation, we used 3 NVIDIA GeForce GTX TITAN X GPUs.

The clinical data were acquired on a Siemens Somatom Sensation 16 scanner (Siemens Medical Solutions, Forchheim, Germany) without using the flying focal spot mode. The scanner acquires 1160 views per rotation, using a 16 row \times 672 channel curved detector array. The distance between the source and isocenter is 570 mm, and the distance between the source and detector is 1040 mm.

4.4.1 Phantom

We use an NCAT phantom image as the ideal (truth) image. To generate synthetic noisy sinogram from the NCAT phantom image volume, we use the MATLAB 2017b `poissrnd` function. Noisy photon count data were generated by sampling a Poisson pdf with data mean given by $g(y; \mu)$ from equation (3.1) where we have ignored the background intensity $\beta(y)$. The parameters of the measured data and the reconstructed image is shown previously in Table 3.2. The incident photon incident was considered to be 10000 for all measurement views.

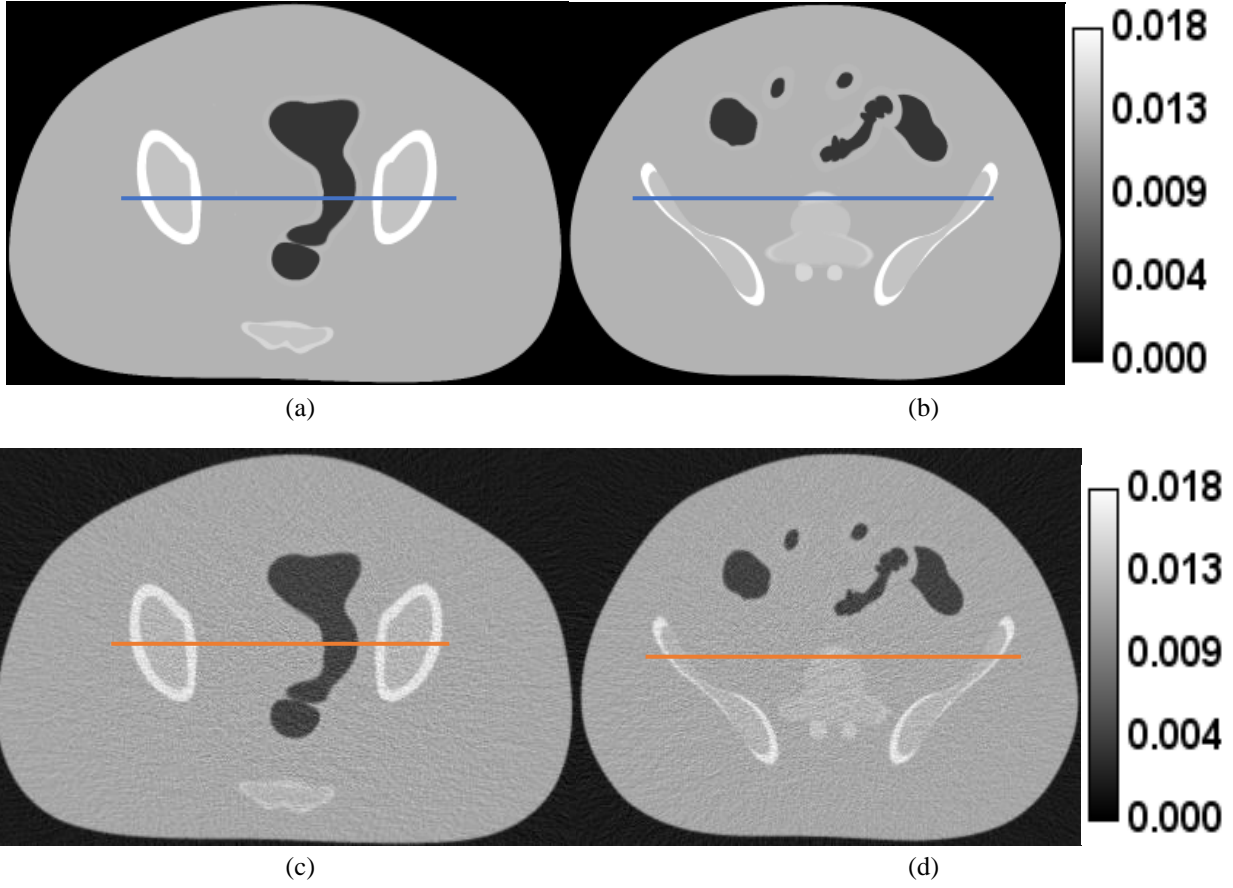


Fig. 4.4 NCAT phantom reconstruction with voxel size = $1.0 \times 1.0 \times 1.0$ mm. Scan parameters: pitch 1.0, 16×1.5 mm collimation, display window width = 0.01759 mm^{-1} , center = 0.008795 mm^{-1} . (a), (b) Axial slices of the actual phantom. (c), (d) Axial slices of the FDK reconstruction of the phantom.

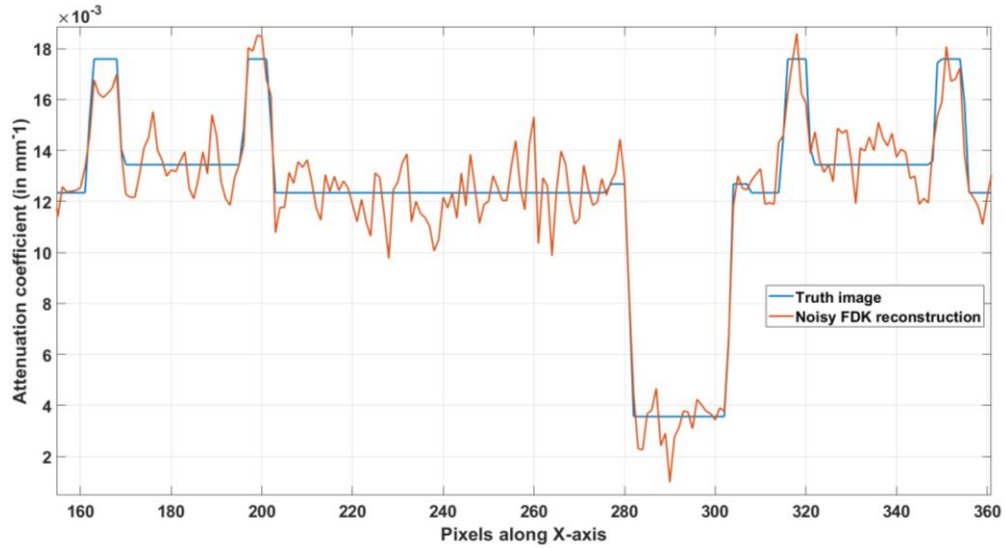


Fig. 4.5 Horizontal profile along the orange line through ideal phantom and noisy FDK reconstruction image shown in Figs. 4.4 (a) and (c)

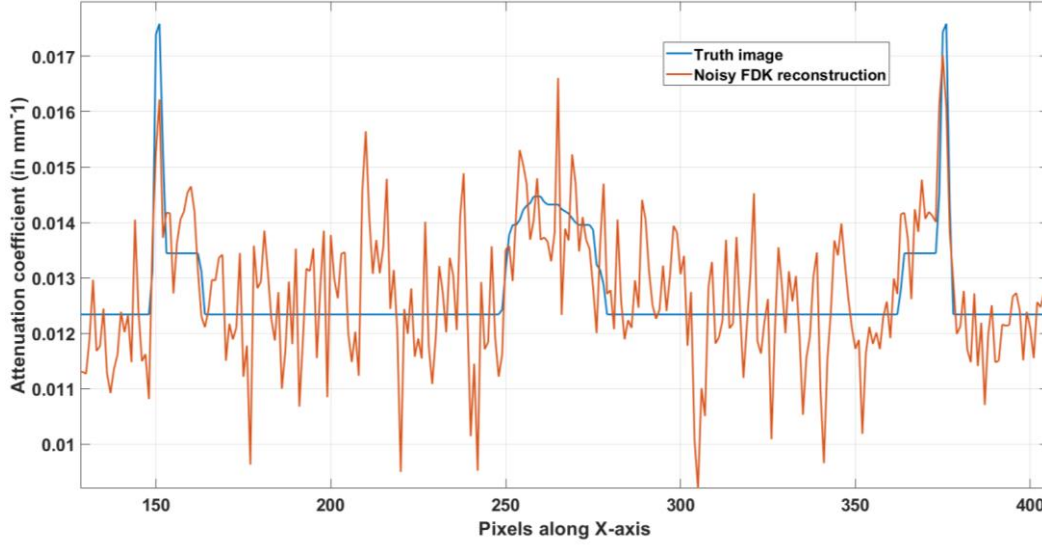


Fig. 4.6 Horizontal profile along the orange line through ideal phantom and noisy FDK reconstruction image shown in Figs. 4.4 (b) and (d)

To quantify the effects of the mismatch between the algorithm and the data models, Percent absolute error (PAE), Root mean square error (RMSE) and Signal-to-noise ratio (SNR) defined in equation (3.62), (3.63) and (3.64) respectively.

	PAE in %	RMSE	SNR in dB
Fig. 4.4 (a) and (c)	5.6983	7.245E-04	23.1523
Fig. 4.4 (b) and (d)	5.7545	7.561E-04	23.6975

Table 4.1 Reconstruction times using clinically-sized data and no OS for different CPU and GPU hardware architectures.

4.4.2 Clinical Datasets

The details of our clinical dataset are described in Chapter 3.6. In this chapter we present axial, sagittal and coronal slices of our helical FDK reconstruction in Fig. 4.7.

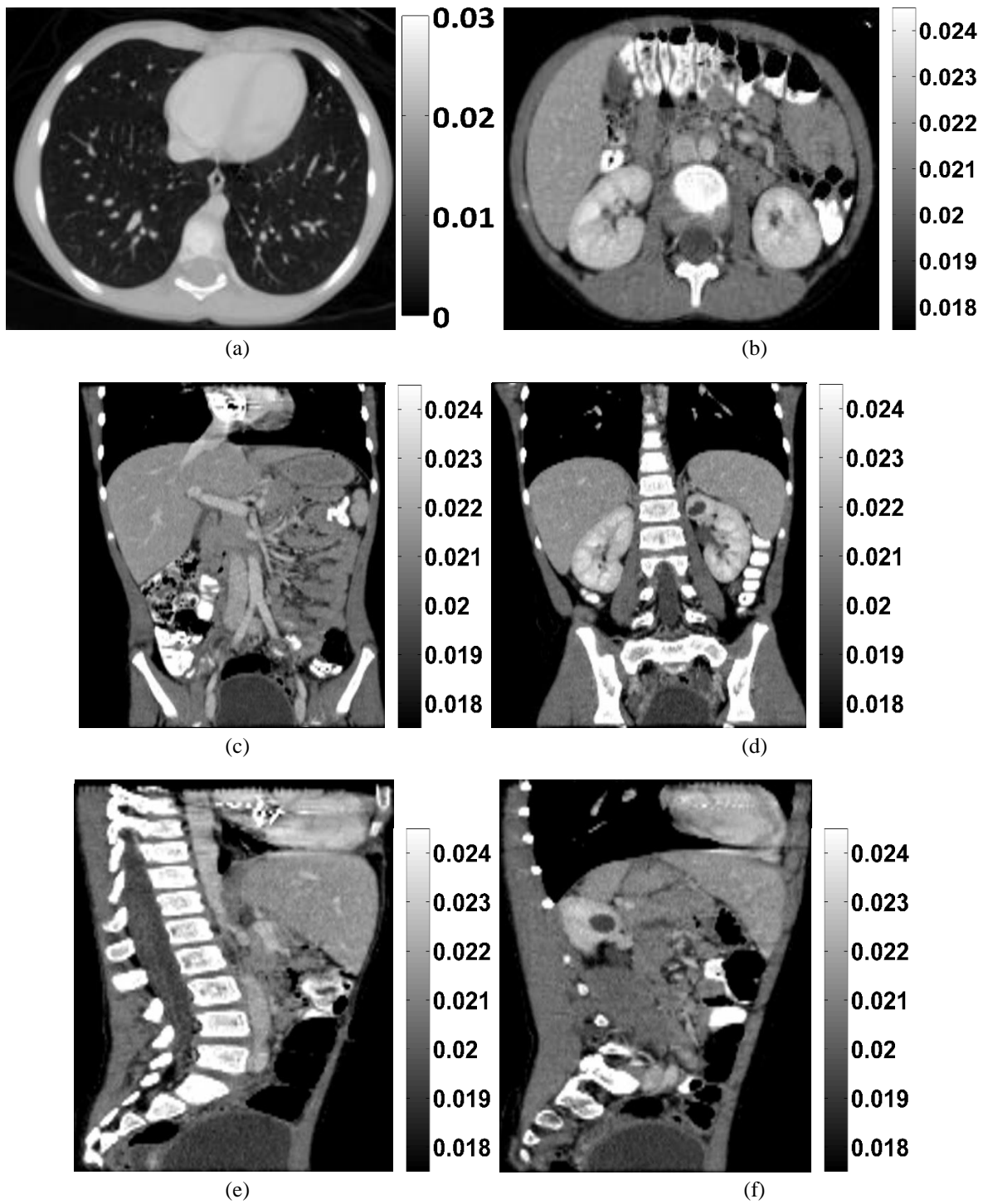


Fig. 4.7 Clinical abdominal reconstruction using FDK algorithm. Voxel size = $1.0 \times 1.0 \times 1.0$ mm. Scan parameters: 180 mAs, pitch 1.0, 16×1.5 mm collimation. (a) Axial slice of lung with display window width = 0.03 mm^{-1} , center = 0.015 mm^{-1} . (b) Axial slice of abdomen with display window width = 0.007 mm^{-1} , center = 0.021 mm^{-1} . (c) and (d) are coronal views and (e) and (f) are sagittal views with display window width = 0.007 mm^{-1} , center = 0.021 mm^{-1} .

4.4.3 Timing Performance

This section addresses the timing performance of our FDK algorithm implementation for different hardware configurations.

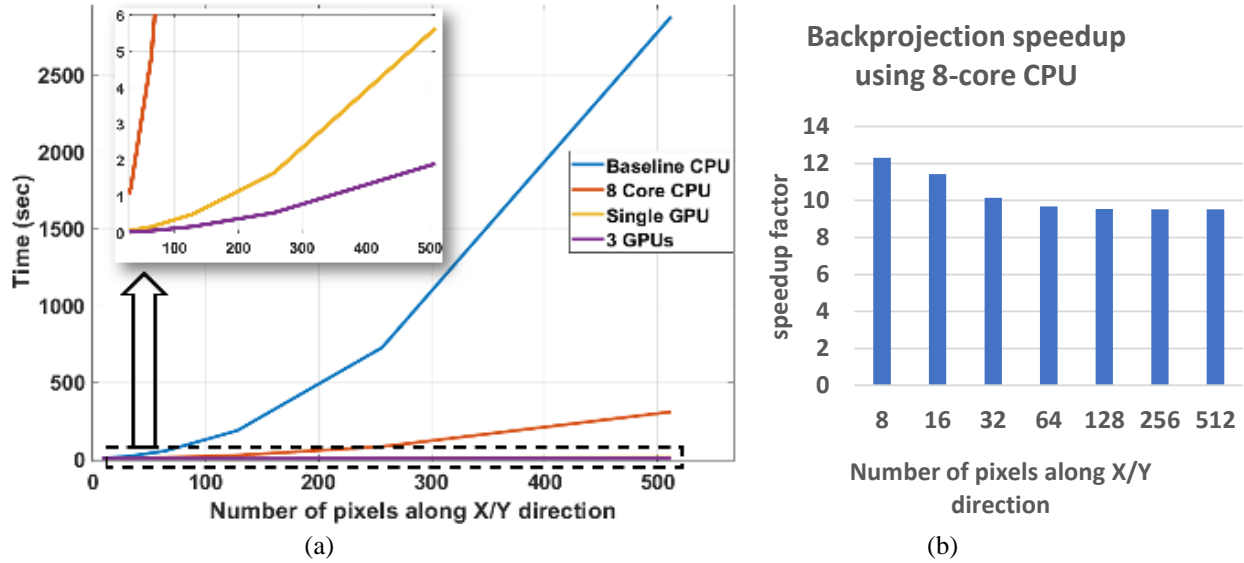


Fig. 4.8 (a) Backprojection time and (b) 8 core CPU speedup factor for a different number of pixels along X/Y direction.

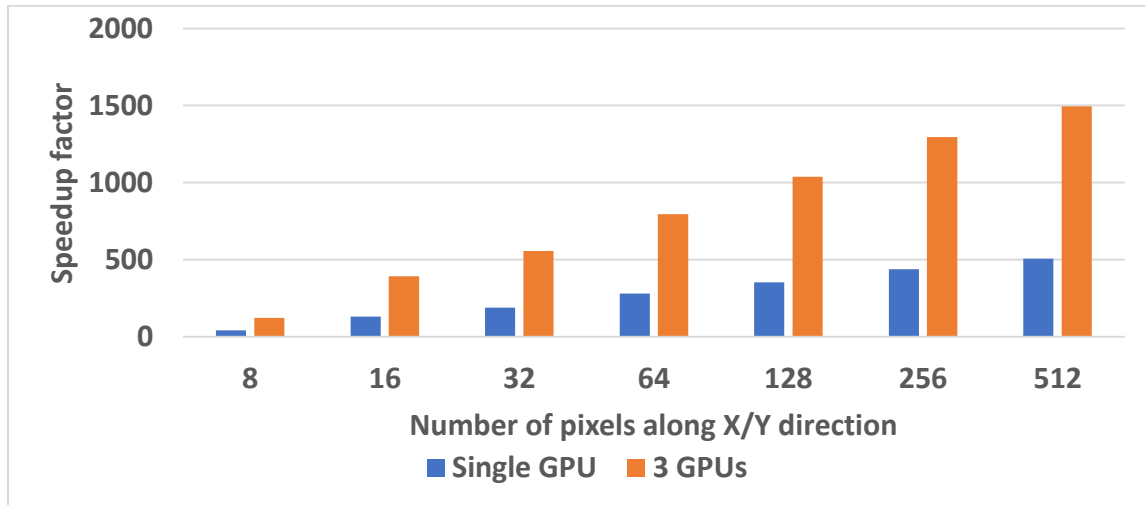


Fig. 4.9 Speedup factor for parallel fan-beam backprojection operation using a different number of GPUs in parallel compared to baseline CPU implementation.

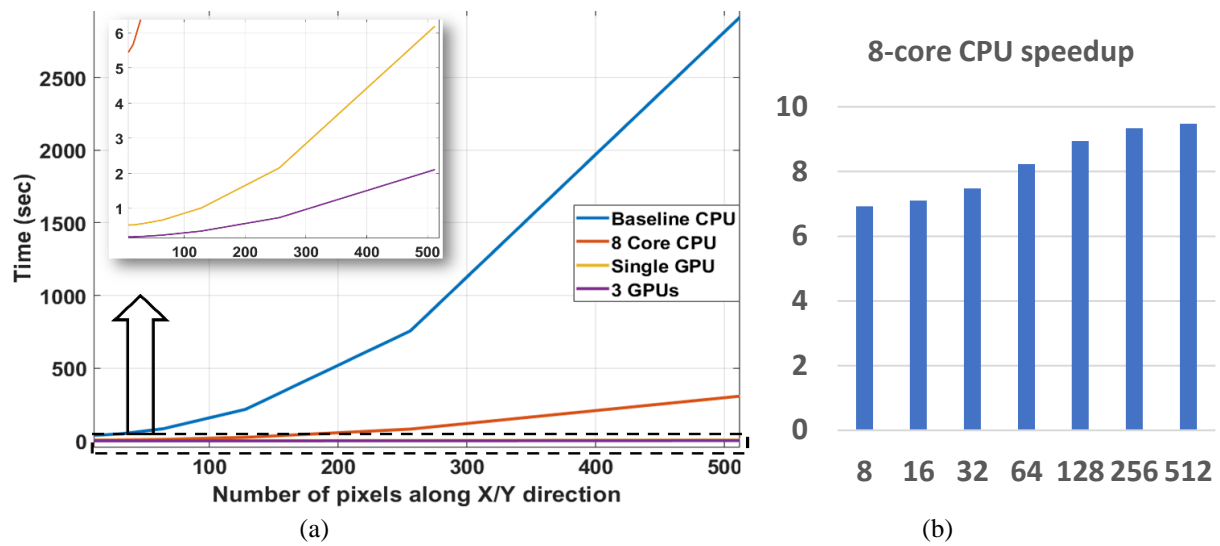


Fig 4.10 (a) Total time and (b) 8 core CPU speedup factor for a different number of pixels along X/Y direction.

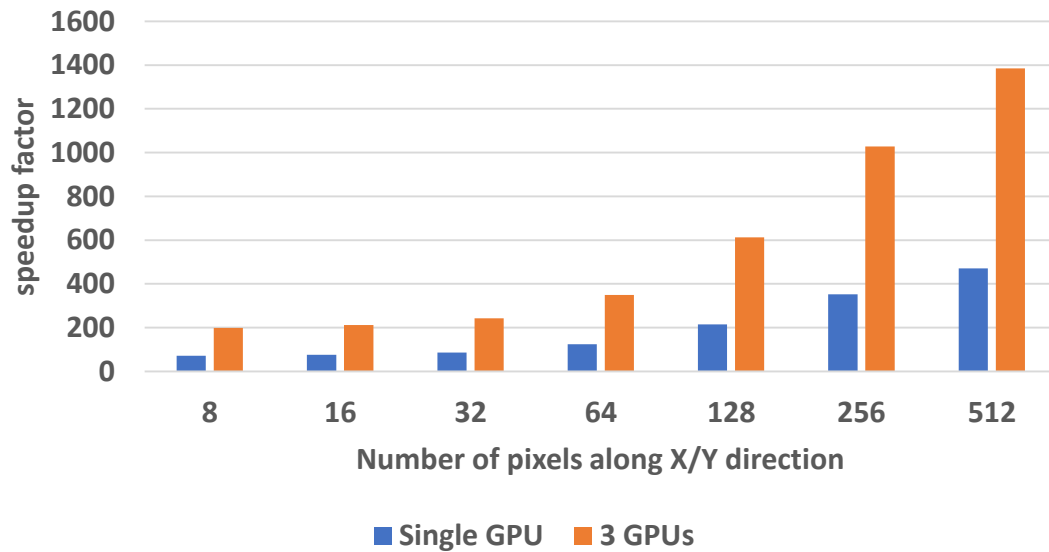
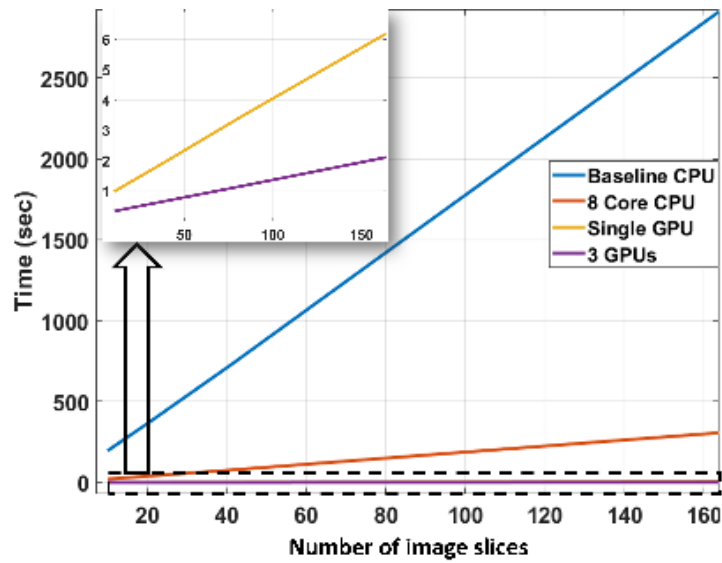
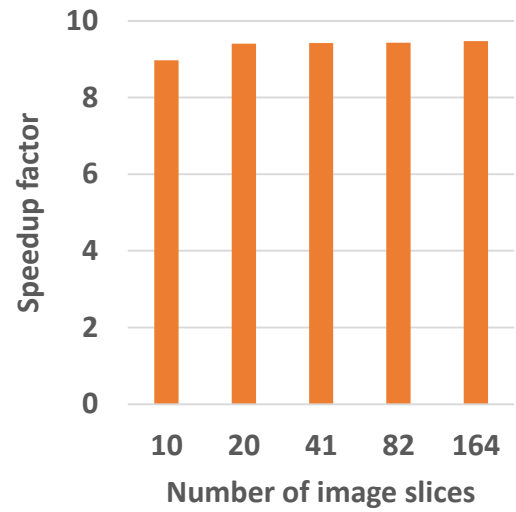


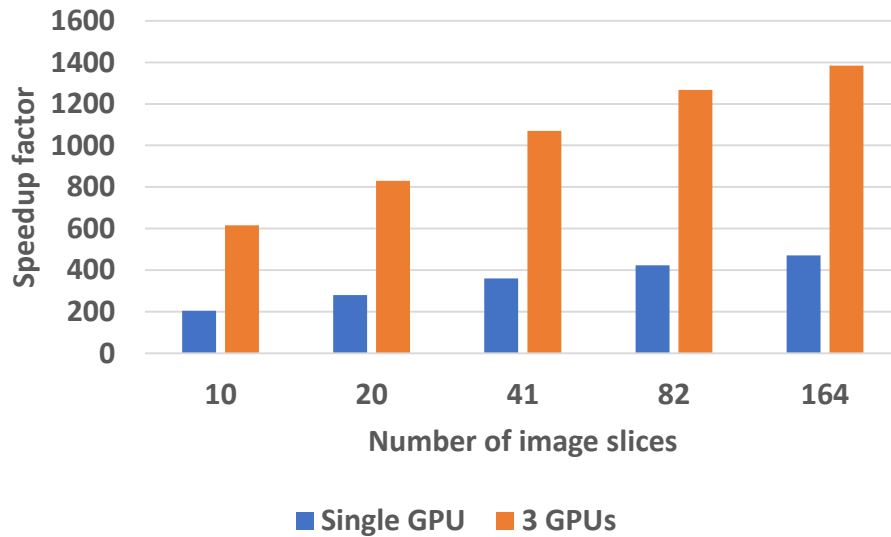
Fig. 4.11 Speedup factor for total computational time using a different number of GPUs in parallel compared to baseline CPU implementation



(a)



(b)



(c)

Fig. 4.12 (a) Total time and (b) 8 core CPU speedup factor for a different number of image slices. (c) Speedup factor for total computational time using a different number of GPUs in parallel compared to baseline CPU implementation.

4.5 Discussion

The total reconstruction time using 3 GPUs in parallel including data read from hard disk drive was less than 2.2s for a clinically-sized data. Compared to other existing methods in the literature, the computational performance of our multi-GPU algorithm is quite competitive. Since we do not have the exact hardware and GPU configuration so a fair and exact comparison would be hard to execute. The speedup factors of different hardware configurations and different scan geometries, were baselined against the reconstruction time of single-threaded CPU implementation. We can clearly observe that in Fig. 4.8 (a), computational time for both CPU and GPU configurations increased in a quadratic fashion consistent with our algorithm. The number of pixels along X/Y direction determines the size of the flattened slab. The amount of computation of every GPU thread launched is directly proportional to the size of the flattened slab. As a result, the computational time increases quadratically with the number of pixels along the perpendicular dimensions of the slab. However, the speedup is small for small image volume due to overhead for data transfer between CPU and GPU. However, when the image volume increases, the relative contribution of the overhead is decreased and the total computation time of backprojection kernel dominates. As a result, we can observe a steady increase in the speedup factor for larger image volume in Fig 4.8 (b) and Fig. 4.9.

When we change the number of views or the number of image slices, there is a linear trend in the increase of computation time as seen in Fig. 4.10. The slow initial speedup in Fig. 4.10 (b) and Fig. 4.11 can also be attributed due to the overhead for data transfer between CPU and GPU. For the brevity of this thesis, we have only shown the computational time and speedup factor for the

variation of image slices. Since the number of minimum image slice is directly proportional to the number of views, we can observe a similar trend if we varied the number of views. When we vary the number of detector rows, as seen in Fig. 4.12, we see a linear increase in the computational time for backprojection. However, when the number of detector rows is small there is significant overhead for data transfer. So, the speedup factor increases slightly with an increase in the number of detector rows.

The 3-D helical FDK reconstruction algorithm presented in this dissertation has significantly low computational burden as presented in Fig. 4.8 (a), Fig. 4.10 (a) and Fig. 4.12 (a). Our approach also improves reconstruction accuracy due to 1-D tangential ramp filtering and no interpolation along the z-axis for this filter [75]. The 3-D weighting function $\omega_{3d}(\theta, t, v)$ described in equation (4.3) is ray dependent and enables our algorithm to reach reconstruction accuracy comparable to exact cone-beam reconstruction algorithms like Katsevich algorithm [47, 67, 77]. The voxel-driven reconstruction method and cone-beam to parallel-beam rebinning approach allows us to update image voxels independently on GPU threads without any thread synchronization. We can also divide the projection data equally on multiple-GPU devices and simultaneously update image voxels over multiple devices in parallel. The detailed algorithm presented in algorithm 4.1 gives us close to 1300X speedup using 3 TITAN X GPUs in parallel over baseline single-threaded CPU implementation. Consequently, we can use this approach without any significant computational burden to calculate initial image estimate for SIR algorithms described in Chapter 3 and aggressive update steps for adaptive surrogate functions described in Chapter 5.

Chapter 5: Acceleration of Iterative-reconstruction Algorithms Using Adaptive Auxiliary Variable

The main hurdle for the adoption of SIR algorithms in practice is the iterative nature of these algorithms and high computation time. The actual computation time required varies with the field of application, the volume of the measured data, and the level of accuracy desired in the reconstructed images. In security applications, the reconstruction time of three-dimensional image volumes must satisfy the rate at which bags travel through the scanner. For many medical applications, the time depends on the availability of radiologists, which can vary widely. There are various pathways to decrease the time in iterative image reconstruction. One of the most effective pathways is to use multiple Graphics processing units (GPUs) to parallelize the computationally intensive parts of the algorithm. [9, 27, 58, 66, 78, 79]. A second pathway is to use advanced algorithms from convex optimization theory [32]. A third pathway is to accelerate the convergence rate of existing algorithms by sacrificing guaranteed convergence properties [29-31]. A new method in the third category, named *adaptive auxiliary variable* is investigated in this article for accelerating the convergence rate of the AM algorithm using a phantom and real clinical data obtained from Siemens Sensation 16 scanner.

In our current work, we first assume a Poisson distribution model for the measured transmission data. Then we calculate a maximum-likelihood estimate between the measured data and data model by reformulating the estimation problem as a double minimization of an I-divergence

problem. A Huber-type penalty is then added to the divergence term. Finally, we formulate an objective function with the I-divergence and regularization terms. As the optimization space is quite large, we have reformulated the objective function as an N one-dimensional convex optimization problem where N is the number of voxels of the image being reconstructed. We then provide pseudo-codes for the general AM algorithm and its accelerated version with the ordered-subset technique. Next, we derive our proposed auxiliary variable based acceleration method and present a pseudocode for its efficient parallel implementation. Finally, we have validated our proposed acceleration technique with NCAT phantoms and Siemens Sensation 16 helical scan data by comparing the convergence rates of straightforward implementation of the AM algorithm with its accelerated version.

5.1 Theory

The AM algorithm in closed form solution yields additive updates for the linear attenuation coefficient values with step-sizes or auxiliary variables that are chosen to guarantee convergence. This guaranteed convergence criterion results in step sizes that are unnecessarily conservative. Therefore, to accelerate the convergence of our algorithm, we will try to choose bigger step sizes using adaptive auxiliary variables $Z(x)$ such that $r(y|x) = \frac{h(y|x)}{Z(x)}$.

For the derivation of these so-called adaptive auxiliary variables, we start with data fit term surrogate function from equation (3.12) in Chapter 3,

$$\begin{aligned}\hat{\mathbb{I}}[d||g; \mu, \hat{\mu}] &= \sum_x \mu(x) \tilde{b}(x) + \sum_y I_0(y) \exp\left(-\sum_x h(y|x) \hat{\mu}(x)\right) \\ &\times \sum_x \frac{h(y|x)}{Z(x)} \exp(-Z(x)(\mu(x) - \hat{\mu}(x))).\end{aligned}\tag{5.1}$$

The derivative of this function with respect to $\mu(x)$ would be,

$$\begin{aligned}\frac{\partial \hat{\mathbb{I}}[d||g; \mu, \hat{\mu}]}{\partial \mu(x)} &= \tilde{b}(x) \\ &- \sum_x h(y|x) I_0(y) \exp\left(-Z_j(\mu(x) \right. \\ &\left. - \hat{\mu}(x))\right) \exp\left(-\sum_x h(y|x) \hat{\mu}(x)\right)\end{aligned}\tag{5.2}$$

Now if our current estimate of $\mu(x)$ at k -th iteration is $\hat{\mu}^{(k)}(x)$ and if we denote $Z(x)$ as $Z^{(k)}(x)$ then we can write

$$\left. \frac{\partial \hat{\mathbb{I}}[d||g; \mu, \hat{\mu}]}{\partial \mu(x)} \right|_{\mu(x)=\hat{\mu}^{(k)}(x)} = 0 \quad \forall x \tag{5.3}$$

$$\Rightarrow Z^{(k)}(x) = \frac{\log\left(\frac{\sum_y h(y|x) I_0(y) \exp(-\sum_x h(y|x) \hat{\mu}(x))}{\tilde{b}(x)}\right)}{\hat{\mu}^{(k)}(x) - \hat{\mu}(x)} \quad \forall x \tag{5.4}$$

Since we are minimizing the surrogate function around $\hat{\mu}(x)$ so any non-negative value for this variable can be used. The inverse of the auxiliary variable basically acts as the weight in closed form update. So, if we can effectively reduce the value of $Z(x)$, we can accelerate the convergence of our algorithm. One such choice would be to make $\hat{\mu}(x) = 0 \quad \forall x$. Thus, our auxiliary variable can be written as:

$$Z^{(k)}(x) = \frac{\log\left(\frac{\sum_y h(y|x)I_0(y)}{\tilde{b}(x)}\right)}{\hat{\mu}^{(k)}(x)} \quad \forall x. \quad (5.5)$$

Now we denote the back projection of incident photon intensity as follows

$$\tilde{b}_0(x) = \sum_y h(y|x)I_0(y). \quad (5.6)$$

Then adaptive auxiliary can be denoted as

$$Z^{(k)}(x) = \frac{\log\left(\frac{\tilde{b}_0(x)}{\tilde{b}(x)}\right)}{\hat{\mu}^{(k)}(x)} \quad \forall x. \quad (5.7)$$

According to our previous estimate of Z from equation (3.16), we can use the length of reconstruction diameter as a threshold for our proposed adaptive auxiliary variable $Z^{(k)}(x)$.

$$Z^{(k)}(x) = \begin{cases} \frac{\log\left(\frac{\tilde{b}_0(x)}{\tilde{b}(x)}\right)}{\hat{\mu}^{(k)}(x)} & \text{if } \frac{\log\left(\frac{\tilde{b}_0(x)}{\tilde{b}(x)}\right)}{\hat{\mu}^{(k)}(x)} < 2 * R_{recon}, \frac{\tilde{b}_0(x)}{\tilde{b}(x)} > 1, \hat{\mu}^{(k)}(x) > 0 \\ 2 * R_{recon} & \text{else} \end{cases} \quad (5.8)$$

We have ignored nonpositive values of $Z^{(k)}(x)$ by the inequalities $\frac{\tilde{b}_0(x)}{\tilde{b}(x)} > 1$, and $\hat{\mu}^{(k)}(x) > 0$ in equation (5.8). Also, it's evident from equation (5.7) that both backprojection arrays can be precomputed. So, the adaptive nature of the auxiliary variable comes from the fact that after each iteration, the denominator is updated with the current estimate of the reconstructed image. For parallel processing units like GPUs, this step doesn't add any significant burden to the overall computation time since the computation of each element of the auxiliary variable is independent of each other and GPU threads can compute all the elements efficiently.

The regularized AM algorithm with ordered subset is described in Algorithm 5.1 with initial image estimate derived from FDK algorithm.

Algorithm 5.1 Regularized OS-AM algorithm with adaptive auxiliary variable	
Input: $\hat{\mu}^{(0,0)}(x) = \hat{\mu}^{FDK}(x) \in \mathbb{R}_+^N$, $d(y)$, $I_0(y) \in \mathbb{R}_+^M$, $\lambda \geq 0$, $\delta > 0$, $\mathbb{Y}_l \forall$ subset index $l = 0, 1, 2, \dots (L-1)$.	
Precompute $\tilde{b}^l(x) = \sum_{y \in \mathbb{Y}_l} d(y)h(y x)$, $\forall l$ and x	
Precompute $\tilde{b}(x) = \sum_y d(y)h(y x)$, $\forall x$	
Precompute $\tilde{b}_0(x) = \sum_y d(y)I_0(y)$, $\forall x$	
Precompute $Z^0(x) = \begin{cases} \frac{\log(\frac{\tilde{b}_0(x)}{\tilde{b}(x)})}{\hat{\mu}^{FDK}(x)} & \text{if } \frac{\log(\frac{\tilde{b}_0(x)}{\tilde{b}(x)})}{\hat{\mu}^{FDK}(x)} < 2 * R_{recon}, \frac{\tilde{b}_0(x)}{\tilde{b}(x)} > 1, \hat{\mu}^{FDK}(x) > 0 \\ 2 * R_{recon} & \text{else} \end{cases} \quad \forall x$	
For iteration: $k = 1, 2, 3, \dots$ do	
for $l = 0, 1, 2, \dots (L-1)$ do	
$\hat{q}^{(k,l)}(y) = I_0(y)\exp(-\sum_y h(y x)\hat{\mu}^{(k,l)}(x))$ for every $y \in \mathbb{Y}_l$	
$\hat{b}^{(k,l)}(x) = \sum_y h(y x) \hat{q}^{(k,l)}(y) \quad \forall x$	
$\hat{\mu}^{(k,l+1)}(x) = \underset{\mu(x) \geq 0}{\operatorname{argmin}} \tilde{b}^{(k)}(x)(\mu(x) - \hat{\mu}^{(k,l)}(x)) + \frac{\tilde{b}^{(k,l)}(x)}{Z^{(k)}(x)} \exp\left(-Z^{(k)}(x)(\mu(x) - \hat{\mu}^{(k,l)}(x))\right) + \frac{\lambda}{L} \sum_{x' \in N(x)} \frac{\omega(x, x')}{2} \delta^2 \left(\left \frac{2\mu(x) - \hat{\mu}^{(k,l)}(x) - \hat{\mu}^{(k,l)}(x')}{\delta} \right - \log\left(1 + \left \frac{2\mu(x) - \hat{\mu}^{(k,l)}(x) - \hat{\mu}^{(k,l)}(x')}{\delta} \right \right) \right)$	
end for	
$\hat{\mu}^{(k+1,0)}(x) = \hat{\mu}^{(k,L)}(x) \quad \forall x$	
$Z^{(k+1)}(x) = \begin{cases} \frac{\log(\frac{\tilde{b}_0(x)}{\tilde{b}(x)})}{\hat{\mu}^{(k+1,0)}(x)} & \text{if } \frac{\log(\frac{\tilde{b}_0(x)}{\tilde{b}(x)})}{\hat{\mu}^{(k+1,0)}(x)} < 2 * R_{recon}, \frac{\tilde{b}_0(x)}{\tilde{b}(x)} > 1, \hat{\mu}^{(k+1,0)}(x) > 0 \\ 2 * R_{recon} & \text{else} \end{cases} \quad \forall x$	
end for	

5.2 Experiments

To generate synthetic sinogram from the NCAT phantom image volume, we add a Poisson noise to the forward projection data of the phantom image using equation (3.1). We use the NCAT

phantom image volume and MATLAB 2017b poissrnd function to generate our noisy estimation of the sinogram. Noisy photon count data were generated by sampling a Poisson pdf with data mean given by $g(y; \mu)$ from equation (3.1) where we have ignored the background intensity term $\beta(y)$. The parameters of the measured data and reconstructed images are shown in Table 5.1.

No. of views	13920
No. of detector channels	672
No. of detector rows	16
No. of image slices	164
No. of pixels/slice	512x512

Table 5.1 Parameters of measured data and image

To quantify the effects of the mismatch between the algorithm and the data models, we use PAE, RMSE and SNR metrics defined in equation (3.62), (3.63) and (3.64) respectively.

5.3 Results

5.3.1 Phantom

Since we start our iterative algorithm with initial image estimate derived from the linear reconstruction algorithms like FBP or FDK, we can use this initial image estimate to precompute the initial values of the auxiliary variable. The value of $Z^{FDK}(x)$ is shown in Fig. 5.1 (b) and 5.1 (d) for reconstructed data using NCAT phantom, where $\hat{\mu}^{(k+1)}(x) = \hat{\mu}^{FDK}(x)$. The region of the image with higher attenuation coefficients show a lower value of the auxiliary variable which in turn results in higher update steps and vice-versa.

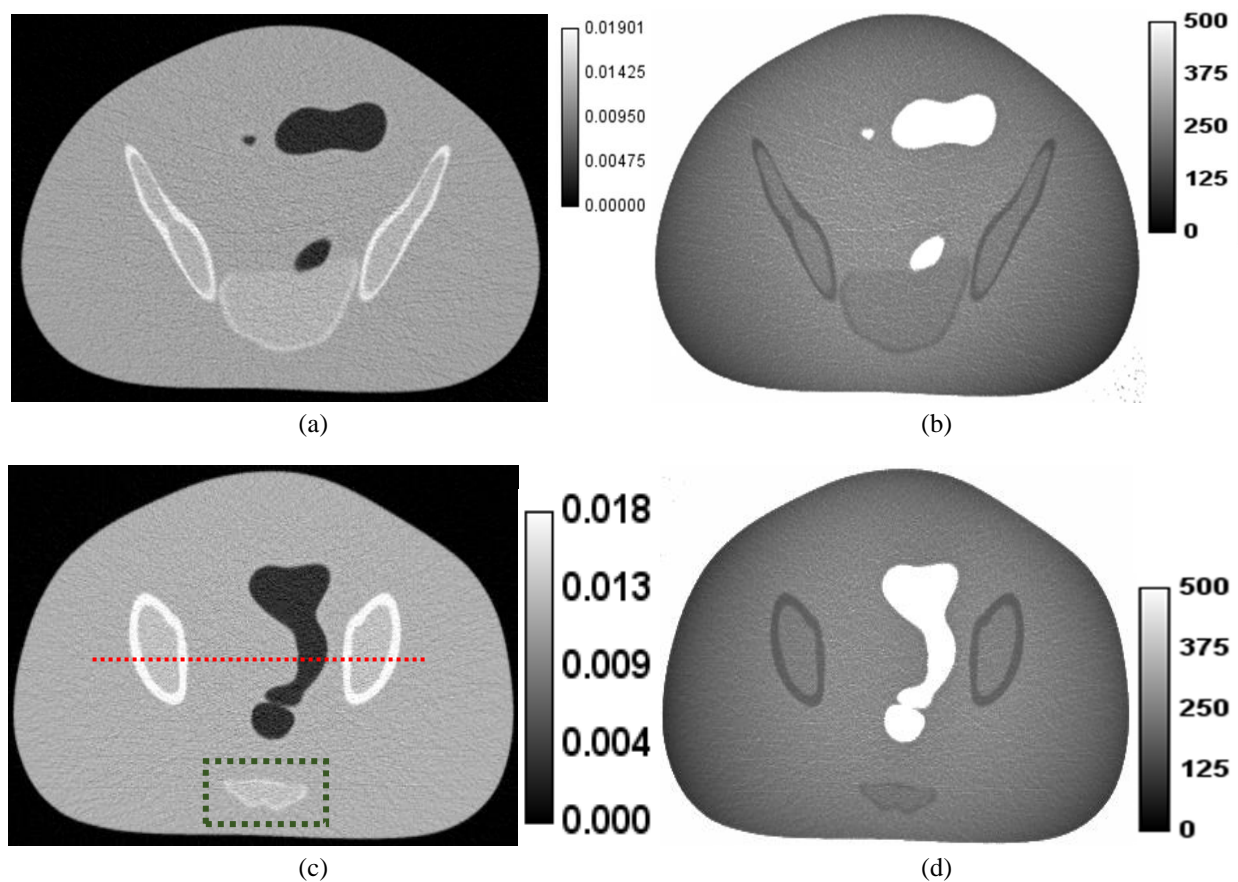


Fig. 5.1 (a) and (c) Linear attenuation coefficient map reconstructed with FDK algorithm for NCAT data in units of mm^{-1} . (b), (d) The values of the auxiliary variable for the corresponding image slice.

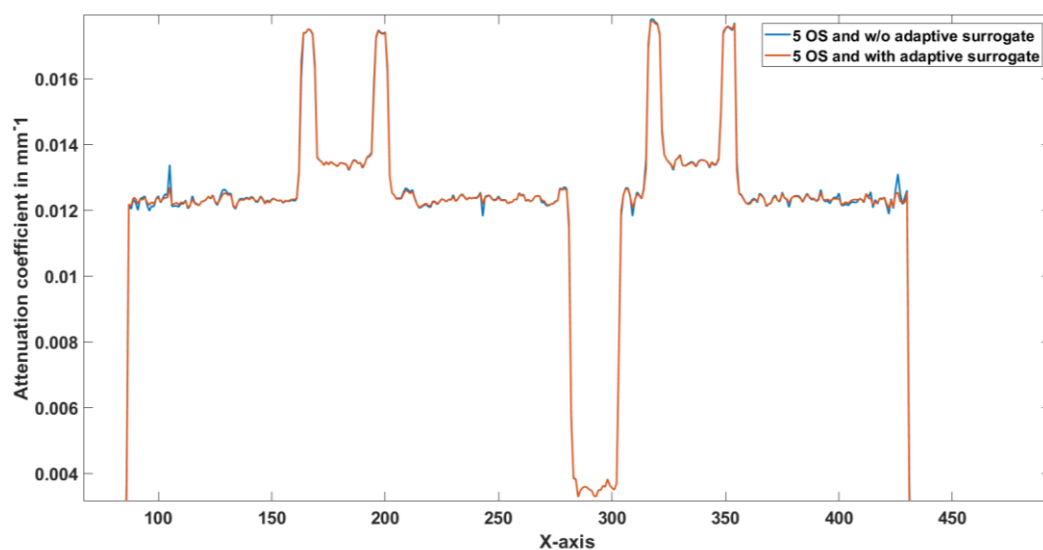


Fig. 5.2 Profile along the red dotted line depicted in Fig. 5.1 (c) for images reconstructed using 100 iterations of 5 OS of AM algorithm without (blue) and with (red) adaptive surrogate function.

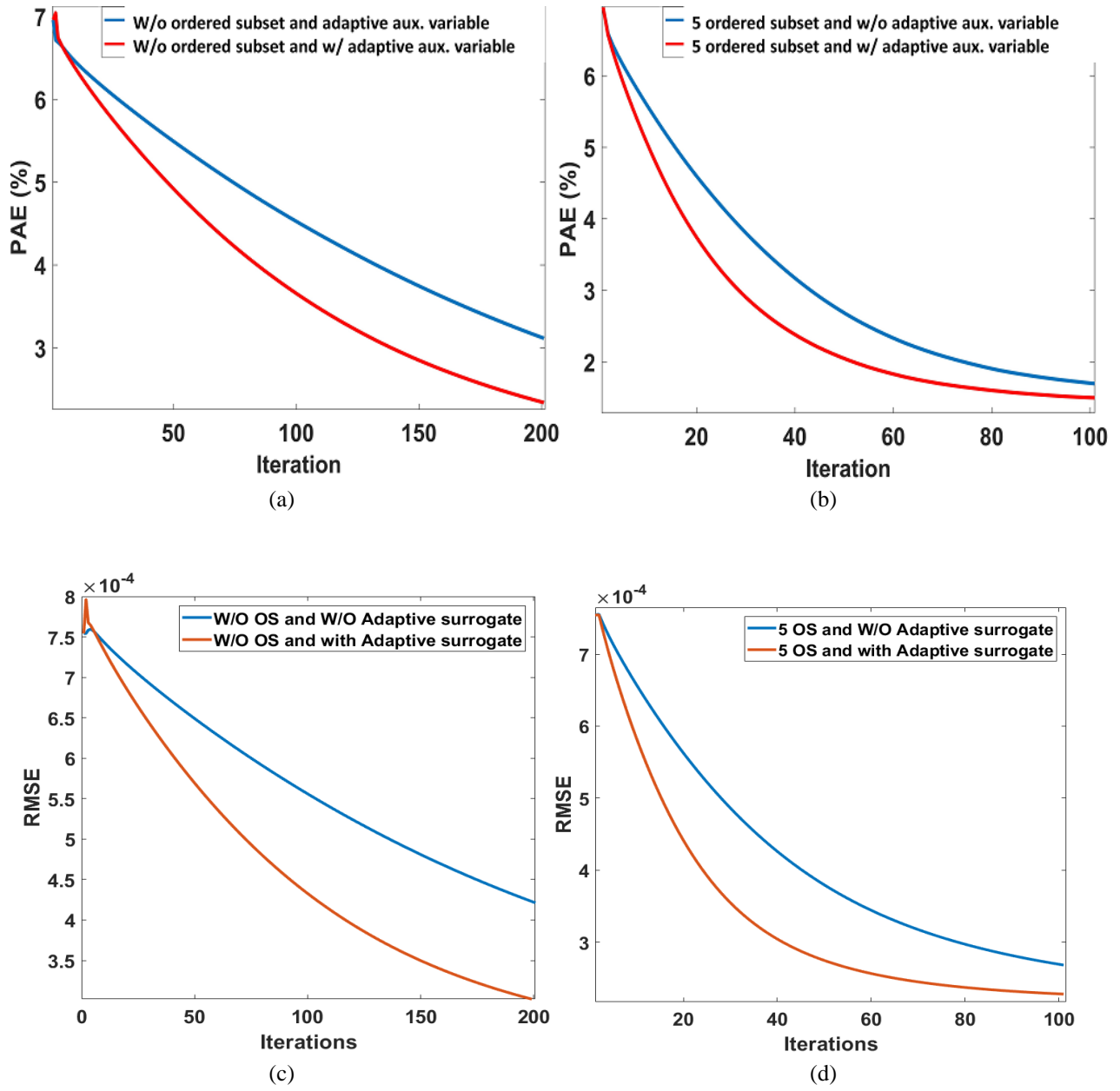


Fig. 5.3 (a), (b) PAE in % vs iteration number for the NCAT phantom with. (c), (d) RMSE vs iteration number for the NCAT phantom.

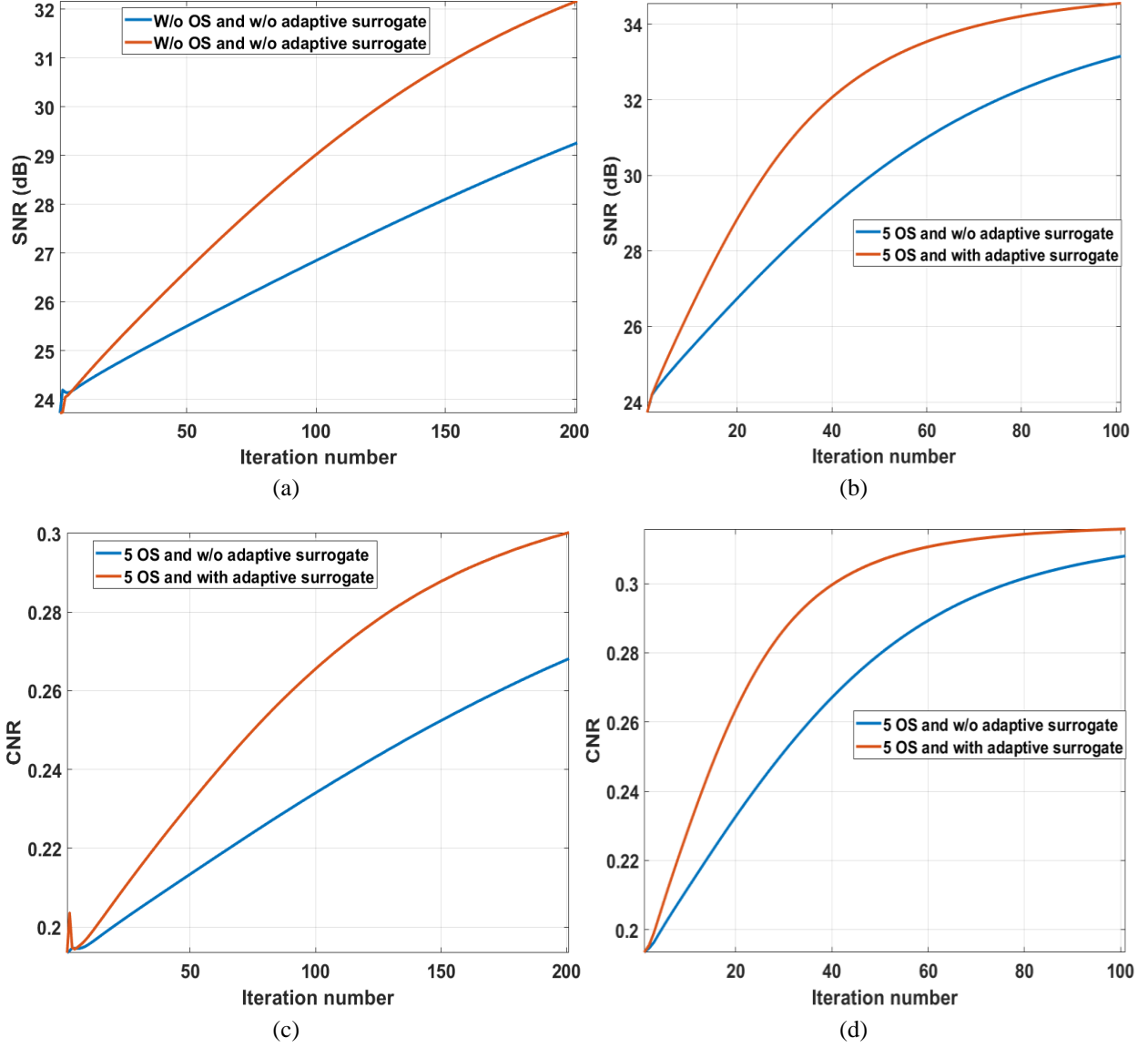


Fig. 5.4 (a) and (b) SNR vs iteration number for the NCAT phantom. (c) and (d) CNR for the structure in a green dotted box in Fig. 5.1 (c) vs iteration number for the NCAT phantom.

5.3.2 Clinical Datasets

The clinical data were acquired on a Siemens Somatom Sensation 16 scanner (Siemens Medical Solutions, Forchheim, Germany) without using the flying focal spot mode. The scanner acquires 1160 views per rotation, using a 16 row \times 672 channel curved detector array. The distance

between the source and isocenter is 570 mm, and the distance between the source and detector is 1040 mm. A lung slice and an abdominal slice is shown in Fig. 5.5 (a) and 5.5 (b) respectively. The value of $Z^{FDK}(x)$ is shown in Fig. 5.5 (b) and 5.5 (d) for reconstructed data using Siemens Sensation 16 scanner, where $\hat{\mu}^{(k+1)}(x) = \hat{\mu}^{FDK}(x)$.

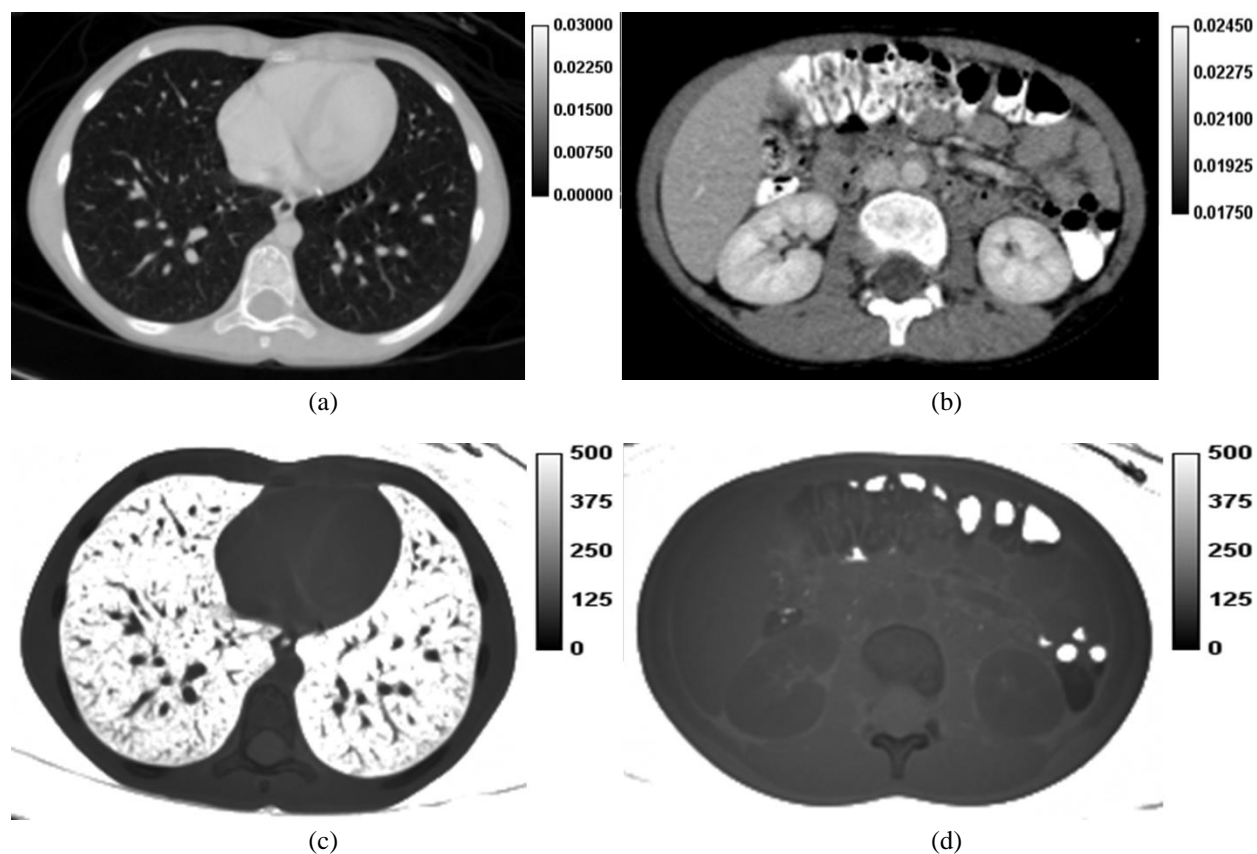


Fig. 5.5 (a) and (b) Linear attenuation coefficient map reconstructed with FDK algorithm for real data obtained from Siemens Sensation 16 scanner in units of mm^{-1} . (c) And (d) are the values of the auxiliary variable for the corresponding image slices in units of mm.

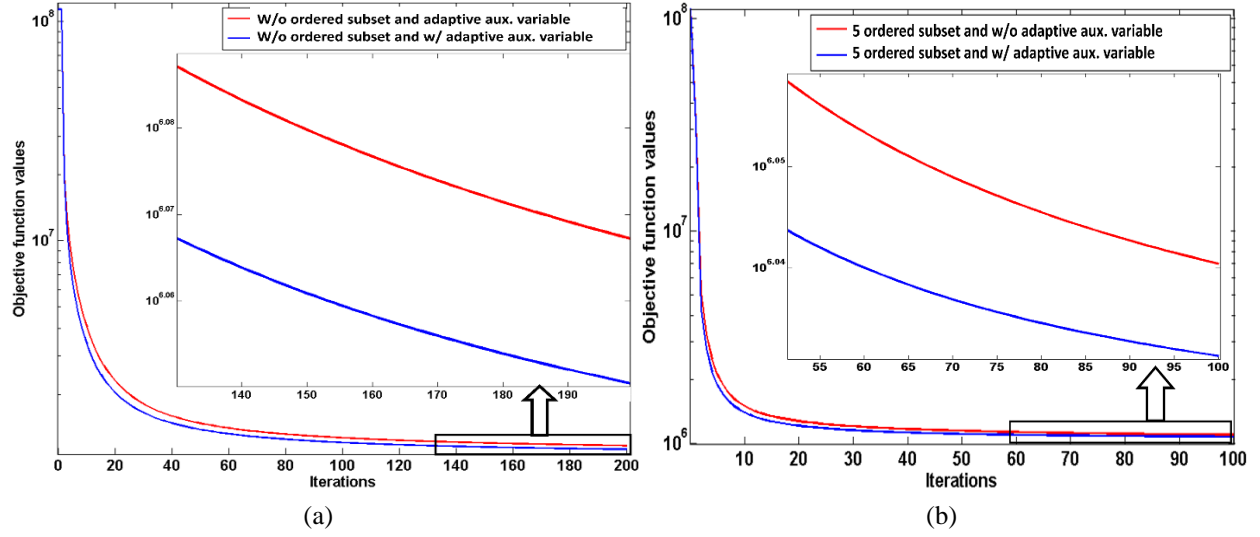


Fig. 5.6 Objective function values vs iteration number for Siemens Sensation 16 scanner reconstructed images (a) without ordered subset implementation and with (blue) and without (red) adaptive auxiliary variable, and (b) with 5 ordered subset implementations and with (blue) and without (red) adaptive auxiliary variable.

5.3.3 Convergence Rate

The RMSE and PAE values (defined in equation (3.62) and (3.63) respectively) for the phantom reconstruction shows 2X increase in the convergence rate. The increase in convergence rate is estimated from the Fig. 5.3, 5.4 and 5.6, by comparing the number of iterations needed by the standard AM algorithm and the adaptive surrogate function based AM algorithm to reach the same values of the objective function or another image quality metric. The objective function value also shows a 2X increase in the convergence rate for the clinical dataset as well. Even with the addition of OS, we can still achieve the same amount of acceleration in convergence rate. However, for higher OS like 29 OS, the rate of acceleration slows down faster than other OS configurations. The main reason for this change can be attributed to the fact that the adaptive surrogate function is not updated for a significant number of image update steps in 29 OS configurations.

5.4 Discussion

In this chapter, we have proposed a novel approach to adaptively compute the additive step in the AM algorithm. We have observed that our approach of using adaptive auxiliary variable combined with OS creates no extra computation cost compared to the straightforward implementation of the OS-AM algorithm. From the image quality assessment parameters, we can conclude that our proposed adaptive auxiliary variable technique shows an average of 2X increase in convergence rate for every OS configuration. We can expect to achieve further acceleration with the addition of other acceleration methods like Nesterov's momentum-based acceleration techniques.

Chapter 6: Dual-energy AM Reconstruction

Algorithm Using GPU

Dual-energy X-ray CT (DECT) reconstruction algorithms have the potential to improve the image contrast and reduce the artifacts [80, 81], which can be highly useful in different clinical applications including radiation dose reduction [82], material decomposition [83, 84] and energy selective imaging. In proton therapy dose prediction analysis, the stopping power of high energy proton beam depends on the estimates of electron density and mean excitation energy. The electron density and the mean excitation energy is derived from a mono-energy estimation of X-ray CT scanning introduces uncertainties due to the beam hardening effect and the method by which the electron density is converted to CT number. Mono-energy estimation fail to disambiguate the Hounsfield unit (HU) degeneracy on density and tissue composition. In order to accurately estimate these parameters, Dual-energy CT (DECT) image reconstructions are widely used in this domain [85-89]. DECT has the potential to reduce range uncertainties by estimating two independent parameters, which can resolve the dependence of photon stopping power on density and tissue composition.

The dual-energy alternating minimization (DE-AM) described in this chapter is an extension of the AM algorithm proposed by O’Sullivan and Benac [1] and discussed in Chapter 3. Simulated data reconstructed in this chapter consists of four inserts suspended in water with calcium chloride and polystyrene used as basis vector material. The DE-AM algorithm combined with ordered subsets produced slow convergence rate along with high computational time.

Motivated by studies demonstrating the slow convergence of the DE-AM algorithm, we have proposed a novel adaptive auxiliary variable based acceleration step which estimates an aggressive update step based on the initial estimate computed using the linear analytical methods like FDK. We have applied this acceleration method to simulated data generated using Siemens Sensation 16 helical scan geometry. Along with algorithmic acceleration steps, we have also proposed fast-parallel multi-GPU based computation of dual-energy alternating minimization algorithm.

6.1 Dual-energy AM Algorithm

At the basis of our statistical model, we assume that the photons arrive at the detectors in accordance with a Poisson counting process. Let the 3-D image volume of linear attenuation coefficients (in mm^{-1}) be represented by the vector μ . Let y denote a ray path between the X-ray source and a pixel in the multi-row detector array, x denote a voxel in the image volume and X-ray spectra by $j \in \{1,2\}$. The measured transmission data, $d_j(y)$, is modeled as originating from independent Poisson counting processes. In discretized form, the mean value of $g_j(y; \mu)$ is

$$g_j(y; \mu) = \sum_E I_{0j}(y, E) \exp \left[- \sum_x h(y|x) \mu(x, E) \right] + \beta_j(y), \quad (6.1)$$

where the outer sum is over discrete energies of the X-ray photons. $I_{0j}(y, E)$ is the mean number of counts in the absence of an attenuating medium for X-ray photon energy E (nominally with units of keV), $\beta_j(y)$ is the mean number of background events assumed to be nonnegative and known. The summation in the exponent represents the forward projection of the attenuation function. The system matrix elements $h(y|x)$ comprise the appropriately discretized point-spread

function relating the projection space to the image space. If projection y does not pass through voxel x , then $h(y|x)$ is zero. The attenuation function $\mu(x, E)$ (in mm^{-1}) is indexed by image space coordinates, x , and by X-ray photon energy, E . We envision a small number, M , of different types of materials indexed by m ,

$$\mu(x, E) = \sum_{m=1}^M \mu_m(E) c_m(x) \quad (6.2)$$

with known linear attenuation coefficients $\mu_m(E)$ in mm^{-1} and relative partial densities $c_m(x)$ [90]. This two parameter Basis vector model (BVM) assumes that attenuation coefficients of unknown materials are linear combinations of the corresponding radiological quantities of dissimilar basis substances i.e. polystyrene, calcium chloride [91]. For pure linear combinations, the relative partial densities are nonnegative and sum to one. Our model allows the values of $c_m(x)$ to be nonnegative, and does not enforce a sum constraint in order to allow the $\mu_m(E)$ to merely span the set of allowable attenuation functions $\mu(x, E)$. Our model for $\mu(x, E)$ in equation (6.2) is equivalent to having terms $(\mu/\rho)(x, E)\rho(x, E)$, where $(\mu/\rho)(x, E)$ is the mass attenuation coefficient (usually given in cm^2/g and $\rho(x, E)$ is the partial density (in g/cm^3 , with $h(y|x)$ in cm) of the m -th constituent. The model (6.2) is related to others in the literature [4, 92-95].

For our Alternating minimization (AM) algorithm, we use the maximum-likelihood solution derived by O'Sullivan and Benac [1]. The problem was formulated as the double minimization of an I-divergence over a linear and exponential family, thereby resulting in a closed-form update for each iteration. The objective function to be minimized for the poly-energetic case is

$$I[d||g] \triangleq \sum_{j=1}^2 \sum_y \left[d_j(y) \log \left(\frac{d_j(y)}{g_j(y; \mu)} \right) + g_j(y; \mu) - d_j(y) \right]. \quad (6.3)$$

The objective function presented in (6.3) can't be optimized directly over μ since the optimization space is large. One of the best approaches is to develop surrogate functions that approximate the original function at every iteration and are easy to minimize. This approach leads to iterative algorithms where different surrogate functions are formed and solved at each iteration and yet the original function decreases monotonically. The decoupled objective function as derived in Appendix B is:

$$\begin{aligned} \hat{I}[d||g; c, \hat{c}] = & \sum_E \sum_{j=1}^2 \sum_y \sum_x \sum_{m=1}^M \left[\hat{p}_j^{(k)}(y, E) \mu_m(E) h(y|x) c_m(x) \right. \\ & \left. + \frac{h(y|x) \mu_m(E)}{Z_m(x)} \hat{q}_j^{(k)}(y, E) \exp \left(-Z_m(x) (c_m(x) - \hat{c}_m^{(k)}(x)) \right) \right]. \end{aligned} \quad (6.4)$$

We define the forward projection of current image estimate at energy level E , $\hat{\mu}^{(k)}(E)$ as:

$$\hat{q}_j^{(k)}(y, E) = I_{0j}(y, E) \exp \left[- \sum_x \sum_{m=1}^M h(y|x) \mu_m(E) c_m(x) \right]. \quad (6.5)$$

The data forward projection is defined as:

$$\hat{p}_j^{(k)}(y, E) = \hat{q}_j^{(k)}(y, E) \frac{d_j(y)}{\sum_{E'} \hat{q}_j^{(k)}(y, E')}. \quad (6.6)$$

Next the back projections $\hat{b}_j^{(k)}(x, E)$ and $\tilde{b}_j^{(k)}(E)$ of the current estimates $\hat{q}_j^{(k)}(y, E)$ and $\hat{p}_j^{(k)}(y, E)$ are calculated as:

$$\tilde{b}_{j,m}^{(k)}(x) = \sum_y \sum_E \mu_m(E) h(y|x) \hat{p}_j^{(k)}(y, E) \quad (6.7)$$

$$\hat{b}_{j,m}^{(k)}(x) = \sum_y \sum_E \mu_m(E) h(y|x) \hat{q}_j^{(k)}(y, E). \quad (6.8)$$

If we replace the estimates from equations (6.7) and (6.8) to equation (6.4), we can rewrite our data fit term surrogate function as:

$$\begin{aligned} \hat{l}[d||g; c, \hat{c}] = & \sum_{j=1}^2 \sum_x \sum_{m=1}^M \left[c_m(x) \tilde{b}_{j,m}^{(k)}(x) \right. \\ & \left. + \frac{\hat{b}_{j,m}^{(k)}(x)}{Z_m(x)} \exp \left(-Z_m(x) (c_m(x) - \hat{c}_m^{(k)}(x)) \right) \right]. \end{aligned} \quad (6.9)$$

In order to derive the closed form solution of the surrogate function presented equation (6.9), we equate its derivative w.r.t. $c_m(x)$ to 0.

$$\frac{\partial \hat{l}[d||g; c, \hat{c}]}{\partial c_m(x)} = \sum_{j=1}^2 \tilde{b}_{j,m}^{(k)}(x) - \hat{b}_{j,m}^{(k)}(x) \exp \left(-Z_m(x) (c_m(x) - \hat{c}_m^{(k)}(x)) \right) = 0, \quad (6.10)$$

$$\Rightarrow c_m(x) = \left[\hat{c}_m^{(k)}(x) - \frac{1}{Z_m(x)} \log \left(\frac{\sum_{j=1}^2 \tilde{b}_{j,m}^{(k)}(x)}{\sum_{j=1}^2 \hat{b}_{j,m}^{(k)}(x)} \right) \right]. \quad (6.11)$$

Finally, the updated estimate, $\hat{c}_m^{(k+1)}(x)$, is calculated iteratively in closed form solution,

$$\hat{c}_m^{(k+1)}(x) \triangleq \left[\hat{c}_m^{(k)}(x) - \frac{1}{Z_m(x)} \log \left(\frac{\sum_{j=1}^2 \tilde{b}_{j,m}^{(k)}(x)}{\sum_{j=1}^2 \hat{b}_{j,m}^{(k)}(x)} \right) \right]. \quad (6.12)$$

The function $Z_m(x)$ is a precomputed normalization function, which can be freely chosen subject to the constraints reviewed by O’Sullivan et al.[1]

$$Z_m(x) = Z = \max_{y,E} \sum_m \sum_x \mu_m(E) h(y|x). \quad (6.13)$$

Since it’s an ill-posed inverse problem, we add a penalty term, $R(\mu)$, to the objective function used in the AM reconstruction, and weight it by a regularization parameter λ , where λ is a scalar that reflects the amount of smoothing desired. A larger value will give emphasis to the penalty term (i.e., the prior expectation that the image will be smooth), whereas a smaller value will give more emphasis to the I-divergence term (i.e., the discrepancy between the measured data and the data estimated by the model). The added penalty term is defined as

$$R(\mu(x)) = \lambda \sum_{x' \in N(x)} \omega(x, x') \psi(\mu(x) - \mu(x')), \quad (6.14)$$

where $R(\mu)$ can be interpreted as the log-likelihood term for some prior. For 3-D regularization, we use the 26-voxel neighborhood $N(x)$ surrounding voxel x . The weights $\omega(x, x')$ control the relative contribution of each neighbor. The potential function $\psi(t)$ is a symmetric convex function that penalizes the difference between the values of neighboring voxels. For computational simplicity, we use a modified potential function used by Lange [37]

$$\psi(t) \triangleq \delta^2 \left[\left| \frac{t}{\delta} \right| - \log \left(1 + \left| \frac{t}{\delta} \right| \right) \right], \quad (6.15)$$

where δ is a parameter that controls the transition between a quadratic region (for smaller $\left| \frac{\mu(x) - \mu(x')}{\delta} \right|$) and a linear region (for larger $\left| \frac{\mu(x) - \mu(x')}{\delta} \right|$). For our specific reconstruction, we exclude

a few image slices from the beginning and end in the penalty calculation because those slices will have severe artifacts due to cone-beam truncation. Calculating the penalty for those slices could negatively impact reconstruction of the inner slices since the artifacts do not any type of structure that can meaningfully be penalized by $R(\mu)$. The overall problem is then to find the penalized-likelihood estimate,

$$\mu_{PML}^* = \underset{\mu \geq 0}{\operatorname{argmin}} I[d||g(\mu)] + \lambda R(\mu), \quad (6.16)$$

where λ is a scalar value that controls the desired smoothness. This approach is also called *penalized maximum likelihood estimation*. It is worth noting that (6.3) is a special case of (6.16) when $\lambda = 0$.

Implementation of the Regularized DEAM Algorithm

The decoupling steps provide an iterative algorithm that is guaranteed to decrease the objective function monotonically. Also, it creates many one-parameter convex functions (one for each voxel) that can be minimized in parallel using GPU threads. The pseudocode for the regularized AM algorithm is shown in Algorithm 6.1.

Algorithm 6.1 Regularized DE-AM algorithm

Input: $\hat{c}_m^0(x) \in \mathbb{R}_+^N, Z = 2 \cdot R_{recon} \times \max_E \sum_m \mu_m(E), d_j(y), I_{0j}(y) \in \mathbb{R}_+^M, \lambda \geq 0, \delta > 0.$

for $k = 1, 2, 3, \dots$ **do**

$$\hat{q}_j^k(y, E) = I_{0j}(y, E) \exp[-\sum_m \mu_m(E) \sum_x h(y|x) \hat{c}_m^k(x)]$$

$$\hat{p}_j^{(k)}(y, E) = \hat{q}_j^{(k)}(y, E) \frac{d_j(y)}{\sum_{E'} \hat{q}_j^{(k)}(y, E')}$$

$$\tilde{b}_{j,m}^k(x) = \sum_y \sum_E \mu_m(E) h(y|x) \hat{p}_j^{(k)}(y, E)$$

$$\hat{b}_{j,m}^k(x) = \sum_y \sum_E \mu_m(E) h(y|x) \hat{q}_j^{(k)}(y, E)$$

$$\hat{c}_m^{k+1}(x) = \underset{c_m(x) \geq 0}{\operatorname{argmin}} \sum_{j=1}^2 \tilde{b}_{j,m}^k(x) (c_m(x) - \hat{c}_m^k(x)) + \sum_{j=1}^2 \frac{\hat{b}_{j,m}^k(x)}{Z} \exp\left(-Z(c_m(x) - \right.$$

$$\left. \hat{c}_m^k(x))\right) + \lambda \sum_{x' \in N_x} \frac{\omega_{xx'}}{2} \delta^2 \left(\left| \frac{2c_m(x) - \hat{c}_m^k(x) - \hat{c}_m^k(x')}{\delta} \right| - \log \left(1 + \left| \frac{2c_m(x) - \hat{c}_m^k(x) - \hat{c}_m^k(x')}{\delta} \right| \right) \right)$$

end for

Acceleration methods**Ordered Subsets**

Ordered subsets is a widely-used technique to increase the convergence speed by using a subset of data at each sub-iteration. The subsets are constructed to be balanced, disjoint, and exhaustive. If the data is partitioned into L number of subsets, at sub-iteration l , a surrogate function for the data-fitting term with only data indices in the corresponding subset is created and minimized with a proportional regularization term. Since the original data-fitting term for which we create surrogate functions changes at each iteration, there is no guaranteed convergence. Denoting all source-detector pairs as \mathbb{Y} and source-detector pairs in subset l as \mathbb{Y}_l for $l = 0, 1, \dots, (L-1)$, the regularized ordered subsets algorithm (OS-AM) is presented in Algorithm 6.2.

Algorithm 6.2 Regularized DE-AM algorithm with ordered subsets

Input: $\hat{c}_m^0(x) \in \mathbb{R}_+^N, Z = 2 \cdot R_{recon} \times \max_E \sum_m \mu_m(E), d_j(y), I_{0j}(y) \in \mathbb{R}_+^M, \lambda \geq 0, \delta > 0, \forall_l \forall l = 0, 1, \dots (L-1).$

for $k = 1, 2, 3, \dots$ **do**

for $l = 0, 1, 2, \dots (L-1)$ **do**

$$\hat{q}_j^{(k,l)}(y, E) = I_{0j}(y, E) \exp \left[- \sum_m \mu_m(E) \sum_x h(y|x) \hat{c}_m^{(k,l)}(x) \right]$$

$$\hat{b}_m^{(k,l)}(x) = \sum_y \sum_E \mu_m(E) h(y|x) \hat{q}_j^{(k,l)}(y, E)$$

$$\hat{p}_j^{(k,l)}(y, E) = \hat{q}_j^{(k,l)}(y, E) \frac{d_j^l(y)}{\sum_{E'} \hat{q}_j^{(k,l)}(y, E')}$$

$$\tilde{b}_{j,m}^k(x) = \sum_y \sum_E \mu_m(E) h(y|x) \hat{p}_j^{(k,l)}(y, E)$$

$$\begin{aligned} \hat{c}_m^{k+1}(x) = \operatorname{argmin}_{c_m(x) \geq 0} & \tilde{b}_m^l(x) \left(c_m^{(k,l)}(x) - \hat{c}_m^{(k,l)}(x) \right) + \frac{\hat{b}_m^{(k,l)}(x)}{Z} \exp \left(-Z \left(c_m^{(k,l)}(x) - \right. \right. \\ & \left. \left. \hat{c}_m^{(k,l)}(x) \right) \right) + \lambda \sum_{x' \in N_x} \frac{\omega_{xx'}}{2} \delta^2 \left(\left| \frac{2c_m^{(k,l)}(x) - \hat{c}_m^{(k,l)}(x) - \hat{c}_m^{(k,l)}(x')}{\delta} \right| - \log \left(1 + \right. \right. \\ & \left. \left. \left| \frac{2c_m^{(k,l)}(x) - \hat{c}_m^{(k,l)}(x) - \hat{c}_m^{(k,l)}(x')}{\delta} \right| \right) \right) \end{aligned}$$

end for

$$\hat{c}_m^{(k+1,0)}(x) = \hat{c}_m^{(k,L)}(x)$$

end for

6.2 Adaptive Auxiliary Variable for Dual Energy

The AM algorithm in closed form solution yields additive updates for the linear attenuation coefficient values with step sizes or auxiliary variables that are chosen to guarantee convergence.

This guaranteed convergence criterion results in step sizes that are unnecessarily conservative. For

the derivation of these so-called adaptive auxiliary variables, we start with data fit term surrogate function

$$\begin{aligned} \hat{\mathbb{I}}[d||g; c, \hat{c}] = & \sum_{j=1}^2 \sum_x \sum_{m=1}^M \left[c_m(x) \tilde{b}_{j,m}^{(k)}(x) \right. \\ & \left. + \frac{\hat{b}_{j,m}^{(k)}(x)}{Z_m(x)} \exp(-Z_m(x)(c_m(x) - \hat{c}_m(x))) \right]. \end{aligned} \quad (6.17)$$

The derivative of this function with respect to $c_m(x)$ would be,

$$\begin{aligned} \frac{\partial \hat{\mathbb{I}}[d||g; c, \hat{c}]}{\partial c_m(x)} &= \sum_{j=1}^2 \tilde{b}_{j,m}^{(k)}(x) - \hat{b}_{j,m}^{(k)}(x) \exp(-Z_m(x)(c_m(x) - \hat{c}_m(x))) \\ &= 0. \end{aligned} \quad (6.18)$$

Now if we equate our previous estimate to be $\hat{c}_m(x) = 0$, we can write

$$\hat{b}_{j,m}^{(k)}(x) \Big|_{\hat{c}_m(x)=0} = \sum_y \sum_E \mu_m(E) h(y|x) I_{0j}(y, E). \quad (6.19)$$

Our current estimate $c_m(x)$ at k -th iteration is $\hat{c}_m^{(k)}(x)$ and estimate of $Z_m(x)$ at k -th iteration is $Z_m^{(k)}(x)$. As a result, we can write

$$\frac{\partial \hat{\mathbb{I}}[d||g; c, \hat{c}]}{\partial c_m(x)} \Big|_{c_m(x)=\hat{c}_m^{(k)}(x), \hat{c}_m(x)=0} = 0 \quad \forall x \text{ and } m = 1 \dots M, \quad (6.20)$$

$$\begin{aligned} \Rightarrow \sum_{j=1}^2 \tilde{b}_{j,m}^{(k)}(x) - \sum_{j=1}^2 \sum_y \sum_E \mu_m(E) h(y|x) I_{0j}(y, E) \exp\left(-Z_m(x) \hat{c}_m^{(k)}(x)\right) \\ = 0, \end{aligned} \quad (6.21)$$

$$\Rightarrow Z_m^{(k)}(x) = \frac{\log\left(\frac{\sum_{j=1}^2 \sum_y \sum_E \mu_m(E) h(y|x) I_{0j}(y, E)}{\sum_{j=1}^2 \tilde{b}_{j,m}^{(k)}(x)}\right)}{\hat{c}_m^{(k)}(x)} \quad \forall x \text{ and } m. \quad (6.22)$$

We can also rewrite the numerator as follows:

$$\tilde{b}_{0j,m}(x) = \sum_y \sum_E \mu_m(E) h(y|x) I_{0j}(y, E), \quad (6.23)$$

$$\Rightarrow Z_m^{(k)}(x) = \frac{\log\left(\frac{\sum_{j=1}^2 \tilde{b}_{0j,m}(x)}{\sum_{j=1}^2 \tilde{b}_{j,m}^{(k)}(x)}\right)}{\hat{c}_m^{(k)}(x)} \quad \forall x \text{ and } m = 1 \dots M. \quad (6.24)$$

If we put constraints then we can rewrite adaptive auxiliary variable as

$$Z_m^{(k)}(x) = \begin{cases} \frac{\log\left(\frac{\sum_{j=1}^2 \tilde{b}_{0j,m}(x)}{\sum_{j=1}^2 \tilde{b}_{j,m}^{(k)}(x)}\right)}{\hat{c}_m^{(k)}(x)} & \text{if } \frac{\log\left(\frac{\sum_{j=1}^2 \tilde{b}_{0j,m}(x)}{\sum_{j=1}^2 \tilde{b}_{j,m}^{(k)}(x)}\right)}{\hat{c}_m^{(k)}(x)} < Z, \\ Z & \text{else} \end{cases} \quad (6.25)$$

where

$$Z = 2 \cdot R_{recon} * \max_E \sum_m \mu_m(E). \quad (6.26)$$

The OS-DE-AM algorithm with adaptive step-size is presented in Algorithm 6.3.

Algorithm 6.3 Regularized OS-DE-AM algorithm with adaptive auxiliary variable

Input: $\hat{c}_m^{(0,0)}(x) \in \mathbb{R}_+^N, Z = 2 \cdot R_{recon} \times \max_E \sum_m \mu_m(E), d_j(y), I_{0j}(y) \in \mathbb{R}_+^M, \lambda \geq 0, \delta > 0, \forall_l \forall l = 0, 1, \dots (L - 1).$

Precompute $\hat{p}_j^0(y, E) = \hat{q}_j^0(y, E) \frac{d_j(y)}{\sum_{E'} \hat{q}_j^0(y, E')} \forall j, y, E$

Precompute $\tilde{b}_{j,m}^{(0)}(x) = \sum_y \sum_E \mu_m(E) h(y|x) \hat{p}_j^{(0)}(y, E) \forall j, x, m$

Precompute $\tilde{b}_{0j,m}(x) = \sum_y \sum_E \mu_m(E) h(y|x) I_{0j}(y, E) \forall j, x, m$

Precompute $Z_m^{(0)}(x) = \begin{cases} \frac{\log\left(\frac{\sum_{j=1}^2 \tilde{b}_{0j,m}(x)}{\sum_{j=1}^2 \tilde{b}_{j,m}^{(0)}(x)}\right)}{\hat{c}_m^{(FDK)}(x)} & \text{if } \frac{\log\left(\frac{\sum_{j=1}^2 \tilde{b}_{0j,m}(x)}{\sum_{j=1}^2 \tilde{b}_{j,m}^{(0)}(x)}\right)}{\hat{c}_m^{(FDK)}(x)} < Z \\ Z & \text{else} \end{cases} \forall x \text{ and } m$

for $k = 1, 2, 3, \dots$ **do**

for $l = 0, 1, 2, \dots (L - 1)$ **do**

$$\hat{q}_j^{(k,l)}(y, E) = I_{0j}(y, E) \exp\left[-\sum_m \mu_m(E) \sum_x h(y|x) \hat{c}_m^{(k,l)}(x)\right]$$

$$\tilde{b}_m^{(k,l)}(x) = \sum_{j=1}^2 \sum_y \sum_E \mu_m(E) h(y|x) \hat{q}_j^{(k,l)}(y, E)$$

$$\hat{p}_j^{(k,l)}(y, E) = \hat{q}_j^{(k,l)}(y, E) \frac{d_j^l(y)}{\sum_{E'} \hat{q}_j^{(k,l)}(y, E')}$$

$$\tilde{b}_m^{(k,l)}(x) = \sum_{j=1}^2 \sum_y \sum_E \mu_m(E) h(y|x) \hat{p}_j^{(k,l)}(y, E)$$

$$\begin{aligned} \hat{c}_m^{k+1}(x) = \underset{c_m(x) \geq 0}{\operatorname{argmin}} & \tilde{b}_m^{(k,l)}(x) \left(c_m^{(k,l)}(x) - \hat{c}_m^{(k,l)}(x) \right) + \frac{\tilde{b}_m^{(k,l)}(x)}{Z} \exp\left(-Z \left(c_m^{(k,l)}(x) - \right. \right. \\ & \left. \left. \hat{c}_m^{(k,l)}(x) \right) \right) + \lambda \sum_{x' \in N_x} \frac{\omega_{xx'}}{2} \delta^2 \left(\left| \frac{2c_m^{(k,l)}(x) - \hat{c}_m^{(k,l)}(x) - \hat{c}_m^{(k,l)}(x')}{\delta} \right| - \log\left(1 + \right. \right. \\ & \left. \left. \left| \frac{2c_m^{(k,l)}(x) - \hat{c}_m^{(k,l)}(x) - \hat{c}_m^{(k,l)}(x')}{\delta} \right| \right) \right) \end{aligned}$$

end for

$$\hat{c}_m^{(k+1,0)}(x) = \hat{c}_m^{(k,L)}(x)$$

$$\tilde{b}_{j,m}^k(x) = \sum_y \sum_E \mu_m(E) h(y|x) \hat{p}_j^k(y, E)$$

$$Z_m^{(k+1)}(x) = \begin{cases} \frac{\log\left(\frac{\sum_{j=1}^2 \tilde{b}_{0j,m}(x)}{\sum_{j=1}^2 \tilde{b}_{j,m}^{(k)}(x)}\right)}{\hat{c}_m^{(k+1,0)}(x)} & \text{if } \frac{\log\left(\frac{\sum_{j=1}^2 \tilde{b}_{0j,m}(x)}{\sum_{j=1}^2 \tilde{b}_{j,m}^{(k)}(x)}\right)}{\hat{c}_m^{(k+1,0)}(x)} < Z \\ Z & \text{else} \end{cases} \forall x \text{ and } m$$

end for

6.3 GPU Implementation

In order to utilize this parallel architecture of GPU devices, we have presented a scheme to compute the energy integrating incident photon intensity in Algorithm 6.4.

Algorithm 6.4 Multi-GPU based computation of incident photon intensity

Input: $\sum_x h(y|x)\hat{c}_m^{(k,l)}(x) \in \mathbb{R}_+^M, d_j(y), I_{0j}(y) \in \mathbb{R}_+^M, \forall l = 0, 1, \dots (L-1)$.

Begin parallel region for every element in measurement array

For every energy E **do**

$$\hat{q}_j^{(k,l)}(y, E) = I_{0j}(y, E) \exp \left[- \sum_m \mu_m(E) \sum_x h(y|x) \hat{c}_m^{(k,l)}(x) \right]$$

$$\widehat{ps}_j^{(k,l)}(y, E) = \hat{q}_j^{(k,l)}(y, E) d_j^l(y)$$

$$\widehat{pt}_j^{(k,l)}(y) += \mu_m(E) \widehat{ps}_j^{(k,l)}(y, E)$$

$$\widehat{qt}_j^{(k,l)}(y) += \mu_m(E) \hat{q}_j^{(k,l)}(y, E)$$

$$\widehat{qs}_j^{(k,l)}(y) += \hat{q}_j^{(k,l)}(y, E)$$

End for

$$\widehat{pt}_j^{(k,l)}(y) = \widehat{pt}_j^{(k,l)}(y) \frac{1}{\widehat{qs}_j^{(k,l)}(y)}$$

End Parallel Region

$$\tilde{b}_{j,m}^{(k,l)}(x) = \sum_y h(y|x) \widehat{pt}_j^{(k,l)}(y)$$

$$\hat{b}_{j,m}^{(k,l)}(x) = \sum_y h(y|x) \widehat{qt}_j^{(k,l)}(y)$$

GPU devices accelerate computational performance when each thread in the device perform an independent operation on an independent element of an array. The absence of device and thread synchronization yield the fastest acceleration. In the pseudocode mentioned above, each of the N -GPUs operates on $1/N$ -times the measured data. Each thread of each GPU operates on independent elements of the measured data which eliminates the slowdown related to GPU thread synchronization. We can store the whole projection data on TITAN X GPU texture memory for fast read only memory access. We can use local memory to store the accumulation values of each energy and equate the value to projection element array stored in global memory. Since each GPU thread writes data on independent and unique elements in projection array, this mitigates the need of atomic operations which speeds up the computation. The most computationally intensive parts are still the projection and backprojection operations. However, we use the same parallelization techniques mentioned in Chapter 3 to accelerate our reconstruction.

6.4 Experiments and Reconstructions

We have used a water phantom with four insets depicted in Fig. 6.1 as a benchmark for determining the timing performance of our multi-threaded CPU and multi-GPU implementation. For the entire data volume using 13920 views, 672×16 detector elements, the total computational time of this energy-dependent accumulation for 120 individual energies are 0.12 seconds compared to 20 seconds for baseline CPU implementation. The wall clock time to run one iteration of AM algorithm without ordered subset on a standalone CPU core without multi-threading was 433 seconds for every projection and 435 seconds for every backprojection with a total time of 1782

seconds. On the other hand, if we compiled the code with OpenMP using 8 cores with 2 hyperthreads per core, the total time for a single iteration reduced to 384 seconds. Using the Intel Thread Profiler, we have determined that in case of our multithreaded CPU implementation, 96.2% of the execution time was in parallel while the rest was spent in barrier method based synchronization over different threads. This profiler result confirms the efficacy of our load balancing scheme within each iteration.

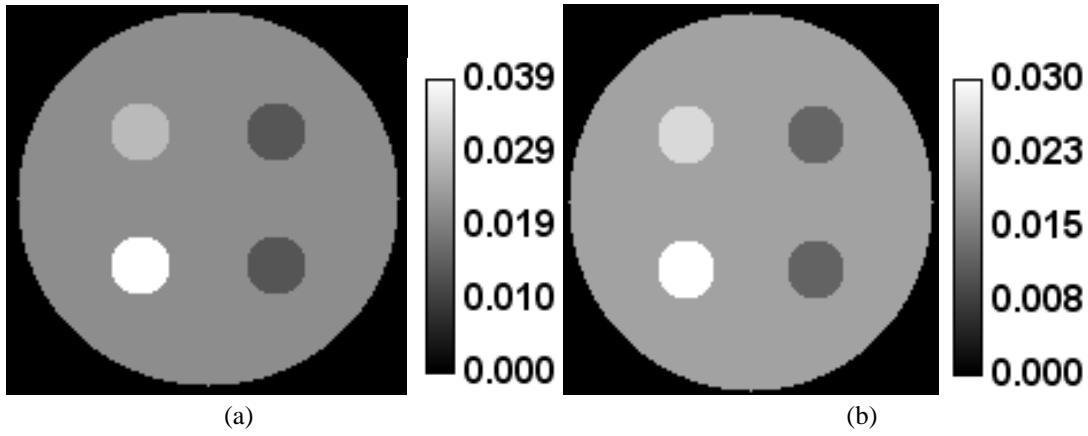


Fig. 6.1 The phantom linear attenuation coefficient image in mm^{-1} at (a) 53 keV and at (b) 70 keV with four inserts (from the top, a clockwise direction) PMMA, ethanol, methyl ethyl ketone (MEK), and calcium chloride.

Operations	Execution Time (seconds)			
	Single-threaded CPU	16-threaded CPU	Single GPU	Multi GPU
Pre- accumulation ($\times 2$)	8.1×2	1.7×2	0.570×2	0.21×2
Projection ($\times 2$)	433×2	92×2	15×2	4.7×2
Exponentiation	20	4	0.37	0.12
Backprojection ($\times 2$)	435×2	95×2	22×2	7.6×2
Image Update ($\times 2$)	4.8×2	1.2×2	0.17×2	0.06×2
Total	1781.8	383.8	75.85	25.26

Table 6.1 Execution times by using different CPU and GPU configurations for a single iteration of DE-AM algorithm

X-rays emitted from tubes are not monoenergetic, instead, the distribution of the photon energies obeys a spectrum [96, 97]. Figure 6.2 shows incident X-ray spectrum corresponding to 90 kVp and 140 kVp. The photons at lower energies are more likely to be absorbed as the linear attenuation coefficient of the material is higher for low photon energies. Therefore, as photons penetrate through an object, the mean photon energy coming out of the object is higher. This is referred as beam hardening phenomenon and it is the source of many image artifacts, such as cupping artifact and streaking artifact.

All 3-D images presented in Fig. 6.1 are $512 \times 512 \times 164$ in size with pixel size of $1\text{mm} \times 1\text{mm} \times 1\text{mm}$. All 3-D simulations use $I_0 = 100000$ which corresponds to the number of unattenuated photons. The two component materials used are calcium chloride ($c_1(x)$) and polystyrene ($c_2(x)$). The attenuation coefficient spectra for the two components are shown in Figure 6.3. The initial images shown in Fig. 6.4 (a) and (b) are reconstructed with FDK algorithm and then converted to component coefficient images. The coefficients for the conversion from linear attenuation coefficient to component coefficient are computed using water equivalent attenuation corresponding to tube voltages 90 kVp and 140 kVp. From equation (6.6), we can denote the forward projection of the data mean estimate as:

$$\tilde{q}_j^{(k)}(y) = \sum_E I_{0j}(y, E) \exp \left[- \sum_x h(y|x) \mu_{water}(E, x) \right], \quad (6.26)$$

where $\mu_{water}(E, x)$ is the energy dependent attenuation coefficient map in mm^{-1} for a phantom image made of water. Now we can perform backprojections of $\tilde{q}_j^{(k)}(y)$ using FDK algorithm for

the 90 kVp and 140 kVp spectra. From the NIST X-Ray Mass Attenuation Coefficient table for water, we can estimate the two corresponding keV energy bins where the attenuation coefficient of water is approximately equal to our FDK reconstruction for the two energy spectra. For our specific spectra shown in Fig. 6.2, 55 keV and 70 keV are the two water equivalent energy bins for the 90 kVp and 140 kVp spectra respectively. For these two energy bins, we use the corresponding attenuation coefficients of calcium chloride and polystyrene and use BVM described in equation (6.2) to estimate the initial $c_1^{FDK}(x)$ and $c_2^{FDK}(x)$ images shown in Fig. 6.4 (a) and (b).

Figure 6.4 (c) and (d) give the reconstructed component images obtained by using 400 iterations of 5 OS unregularized DE-AM algorithm with noiseless data. Unregularized DE-AM algorithm produce images with a large bias for the estimations of high density material Calcium Chloride as shown in Fig. 6.4 (c) and (d). Higher standard deviations are observed for edge regions of $c_1(x)$ and reconstructions for $c_2(x)$ tend to have more uniform standard deviations over the whole region, except for calcium chloride and PMMA, which have relatively higher attenuation coefficients. In Fig. 6.5, we have plotted the RMSE value between ideal phantom image and 29 OS-DE-AM reconstructed image for different energy bins. The RMSE value for Calcium chloride and PMMA are higher compared to all other materials due to their relatively higher attenuation coefficients.

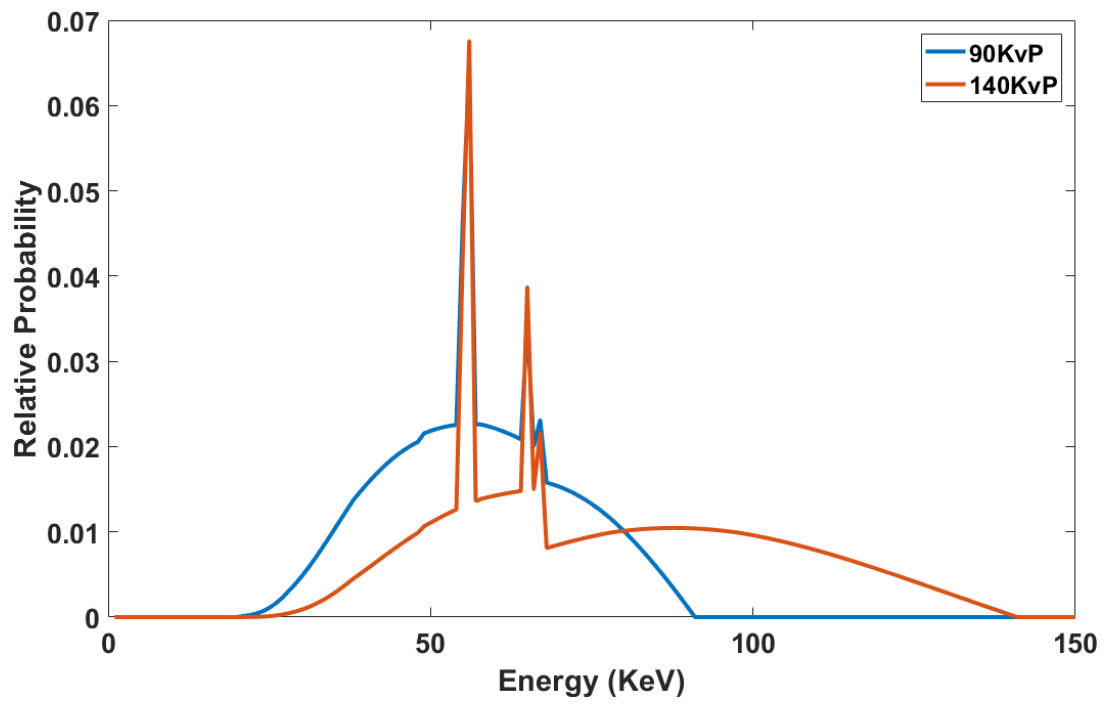


Fig. 6.2 Incident spectra

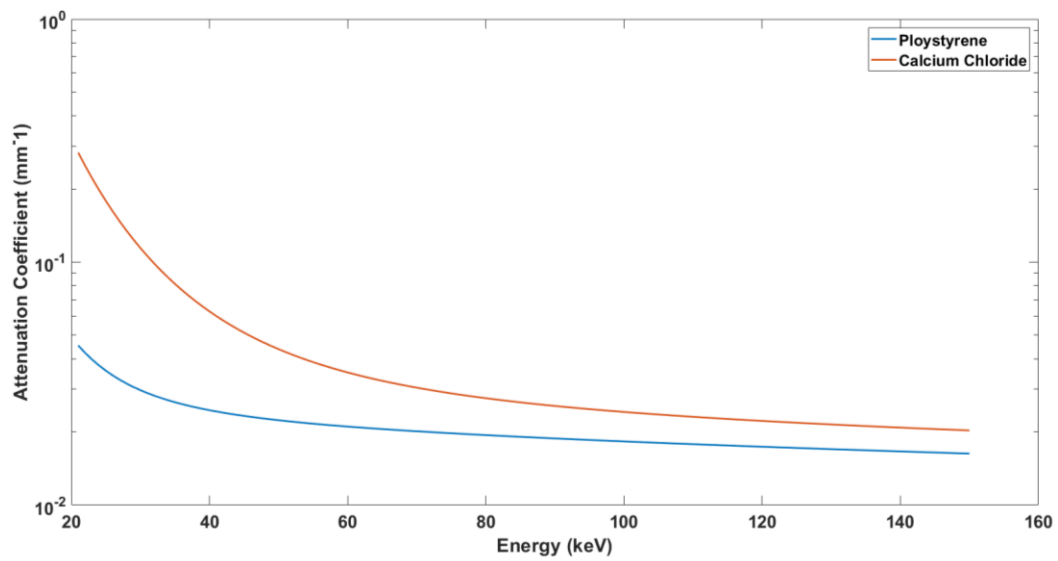


Fig. 6.3 Attenuation coefficient of the component materials

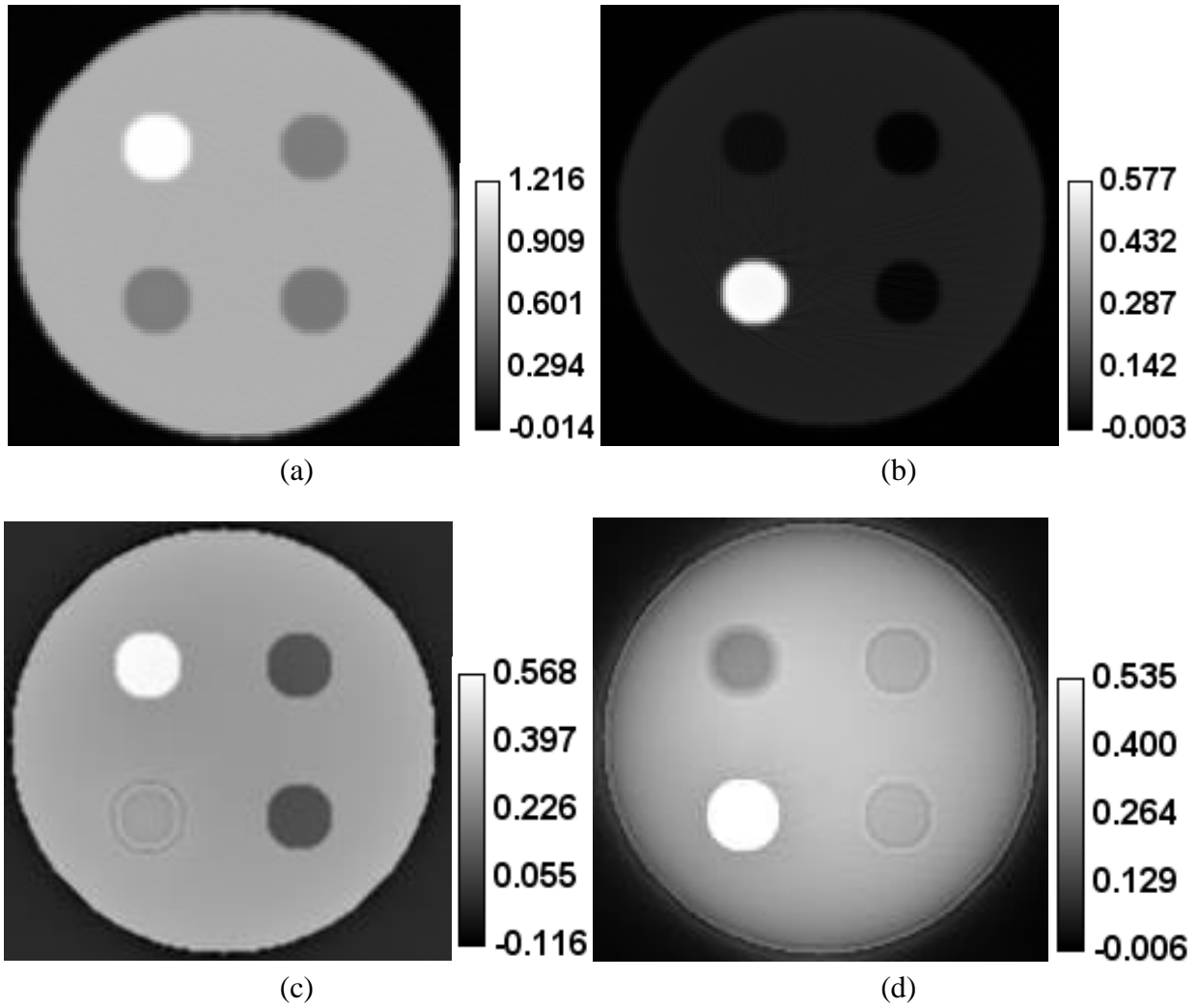


Fig. 6.4 Initial (a) $c_1(x)$ and (b) $c_2(x)$ component images reconstructed using FDK algorithm. (c) $c_1(x)$ and (d) $c_2(x)$ component images reconstructed using 400 iterations of 5 OS DE-AM algorithm.

	PMMA	Ethanol	MEK	CaCl
$c_1(x)$ Image	6.9686×10^{-4}	4.2806×10^{-4}	3.7981×10^{-4}	2.8867×10^{-4}
$c_2(x)$ Image	5.6032×10^{-4}	2.0106×10^{-4}	1.7828×10^{-4}	0.0041

Table 6.2 Variance of different materials in different component images

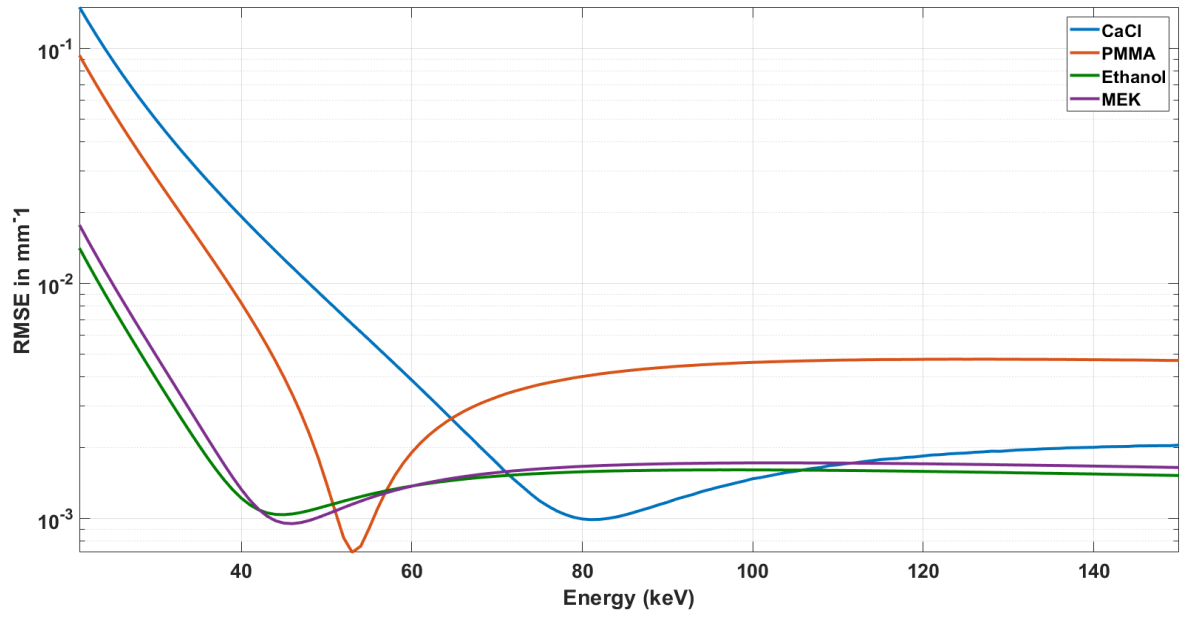


Fig. 6.5 Plot of RMSE between truth image and reconstructed image using 100 iterations of 29 OS DE-AM algorithm vs different energy bins.

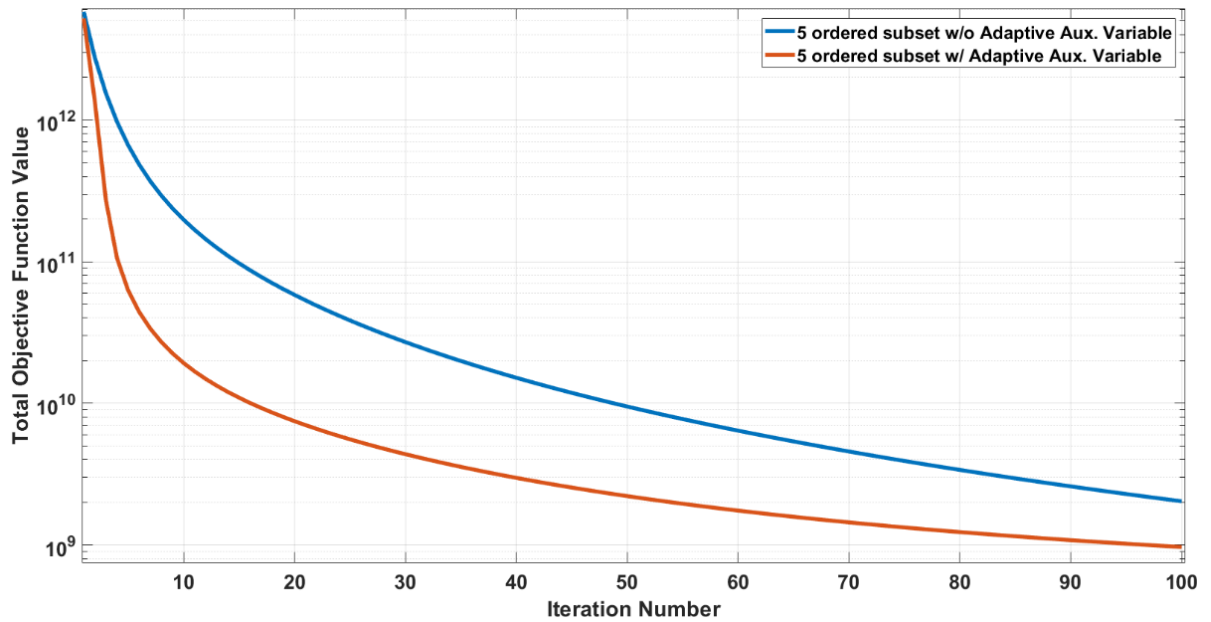


Fig. 6.6 Total objective function values vs iteration number for 5 OS implementations of the DE-AM algorithm.

6.5 Conclusion

In this chapter, DE-AM algorithms were used to reconstruct 3-D images from data simulated with the geometry of the Siemens Sensation 16 scanner. We have shown significant improvement in computational time compared to baseline CPU implementation. We have proposed a novel approach to adaptively compute the additive step in the DE-AM algorithm. We have observed that our approach of using adaptive auxiliary variable combined with OS creates no extra computation cost compared to the straightforward implementation of the OS-AM algorithm. From the Fig. 6.6, we can conclude that our proposed adaptive auxiliary variable technique shows an average of 2X increase in convergence rate for 5 OS configuration.

Chapter 7: Deep Convolutional Neural

Network Based Denoising

In order to reduce the potential radiation risk, low-dose CT has gained increased attention in medical imaging community. Currently, patients go through multiple X-ray CT scans during image-guided radiation therapy, which elevates the potential risk for tissue damage and radiation-induced cancer [98, 99]. However, simply lowering the radiation dose will significantly degrade the image quality. Therefore, there is increasing demand for fast image reconstruction algorithms that can produce higher quality images in clinically relevant time. In this chapter, we explore the deep Convolutional neural network (CNN) as a noise reduction strategy for low-dose CT. A deep convolutional neural network is used to map low-dose CT images towards its corresponding normal-dose counterparts using recently proposed residual learning method [100]. Qualitative results demonstrate a great potential of the proposed method for artifact reduction and structure preservation. In terms of the quantitative metrics, the proposed method has shown a substantial improvement on PSNR, RMSE, and SSIM than the competing state-of-art methods like Block matching 3D (BM3D) [101] and Weighted nuclear norm minimization (WNNM) [102]. Furthermore, the speed of our method is significantly faster than the iterative and linear reconstruction methods discussed in previous chapters.

7.1 Theory

7.1.1 Deep Neural Networks for X-ray Image Denoising

Most clinical X-ray CT scanners currently being used employ some version of analytical reconstruction algorithms like FBP or FDK. However, in low-dose X-ray CT, the linear reconstruction algorithms introduce severe artifacts typically due to beam hardening, photon starvation, scatter and other causes which reduces the diagnostic reliability. Therefore, high quality diagnostically relevant low-dose X-ray CT reconstruction is a topic of major research effort. In previous chapters, we have observed that model-based image reconstruction problems perform reliably well but they are still computationally expensive even with the introduction of multiple GPUs in parallel. As a result, we have explored the possibility of leveraging the tremendous potential of artificial intelligence especially deep convolutional neural networks to perform X-ray CT image denoising.

The concept of the first feedforward supervised deep multilayer perceptron was introduced by Alexey Ivakhnenko in 1965 [103]. Other researchers subsequently used deep learning in computer vision, speech recognition problems, however, their application and adoption were somewhat limited by the astronomically high computational cost. In 2009, NVIDIA was involved in what was called the “big bang” of deep learning, as deep-learning neural networks were trained with NVIDIA Graphics processing units (GPUs). GPUs speed up training algorithms by orders of magnitude, reducing running times from weeks to days. In May 2016, IEEE Transactions on Medical Imaging published a special issue on “Deep Learning in Medical Imaging” [104]

containing 18 special issue articles that outlined the tremendous potential of deep learning based algorithms in the medical imaging domain. Over the year several researchers have tried to harness the sophisticated pattern recognition power of deep networks and apply that to low-dose CT denoising field [105-109]. Deep Convolutional Neural Network (CNN) can easily learn high dimensional features through a hierarchical framework. The main advantage of this approach is the low computational burden along with seamless integration with the post-processing workflow from CT scanner reconstruction without ever accessing the sinogram itself.

In this work, we treat the learning problem as a discriminative one i.e. separating the noise from the noisy image by feedforward CNN instead of learning over a generative adversarial model with the predefined image prior. We use deep architecture to extract high-level image patterns and characteristics [110], batch normalization [100, 111], and residual learning [111, 112] to speed up our learning rate. We have also parallelized our algorithm and implemented it on NVIDIA TITAN X GPUs to reduce computational time. The main advantage of our design is the use of residual learning to learn and extract the pattern of noise itself instead of learning complex organ structures typically present in X-ray CT images.

7.1.2 Residual Learning and Batch Normalization

The main motivation for the use of deep residual learning proposed by Kaiming et. al [112] stems from the increased difficulty in training deeper networks. They reformulated their learning problem as a residual function with reference to the layer inputs, instead of learning the unreferenced function. With growing evidence in favor of residual mapping being easier to learn

rather than original unreferenced mapping, residual networks can learn residual mapping in a few stacked layers thereby increasing training accuracy with increasing network depth. Leveraging this residual network strategy, we can form deep CNN which can easily learn complex noise patterns present in X-ray CT measurements arising from various factors like a cone-beam artifact, detector edge response, beam hardening and scatter. In our approach, we use a single residual unit to predict the residual image similar to the methods used by Kai Zhang et al. [111].

One of the major problems in training deep networks is the fact that the distribution of the internal hidden network's input changes during training which slows down learning rate and requires careful initialization of parameters. The change in mean and standard deviation of the internal hidden layer non-linearity input for each mini-batch during training is known as internal covariance shift [100]. Batch Normalization (BN) is therefore used to reduce the internal covariance shift by introducing a normalization step and performing the performing the normalization for each mini batch of our training CNN model. Batch normalization has shown to increase learning rate, quantitative accuracy and reduce overall dependence to accurate initialization of parameters [100]. We have shown a schematic diagram of our batch normalization implementation in Fig. 7.1. The "Layer" in Fig 7.1 can be any hidden layer in our network. The output of this network is denoted by the vector x . The mean and standard deviation of this output over a mini-batch can be represented by μ and σ respectively. The distribution of x could change over different mini-batch training which can introduce internal covariance shift. In order to solve this problem, we add two other additional terms γ and β , which act as the new standard deviation

and mean over different mini-batches. Therefore, batch normalization only adds two extra parameters per activation layer and they can be easily updated with back-propagation.

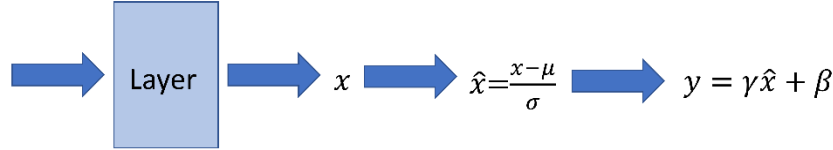


Fig. 7.1 Schematic diagram for batch normalization

We have proposed that addition of both batch normalization and residual learning can enhance the Deep CNN performance on learning complex X-ray CT noise pattern and at the same time result in the fast, robust and stable training regimen. In the subsequent chapters, we have discussed the details of our training network and the performance of our network on simulated low-dose X-ray CT noise.

7.1.3 Proposed Network Model

In this section, we discuss the rationale behind our proposed network architecture and training parameters. Following the improved results from using very small (3×3) convolutions filters for deep network architecture [113], we adopt this architecture instead of pooling layers. Therefore, the size of our receptive field is $(2D + 1) \times (2D + 1)$ for a network of depth D . Higher receptive depth field is advantageous in capturing high level X-ray CT image details and texture information. For our general image denoising task, we set a receptive field size of 41×41 with corresponding network depth of 20.

The input to our Deep CNN is a noisy low-dose X-ray CT reconstructed image denoted by $\mu_{LD}(x)$, where x denotes the voxel indices. We can represent our noisy observation as follows

$$\mu_{LD}(x) = \mu_{HD}(x) + \beta(x) \quad (7.1)$$

where $\mu_{HD}(x)$ is the equivalent high dose (clean) image and $\beta(x)$ is the added measurement noise.

The noise model is described in the following chapter but for our current analysis, we can assume it as an additive noise model. Our deep CNN residual learning is trained on the residual mapping $\beta(x)$. We have used averaged mean squared error as our error estimate for training purposes

$$\mathcal{E}(\Theta) = \frac{1}{2N} \sum_{x=1}^N \|\mathcal{R}(\mu_{LD}(x); \Theta) - (\mu_{LD}(x) - \mu_{HD}(x))\|^2 \quad (7.2)$$

where Θ denote all the training parameters, $\mathcal{R}(\cdot)$ is the residual mapping function consisting of all network layer weights and bias terms, $\mathcal{E}(\cdot)$ is the error function, and N is the total number of voxels.

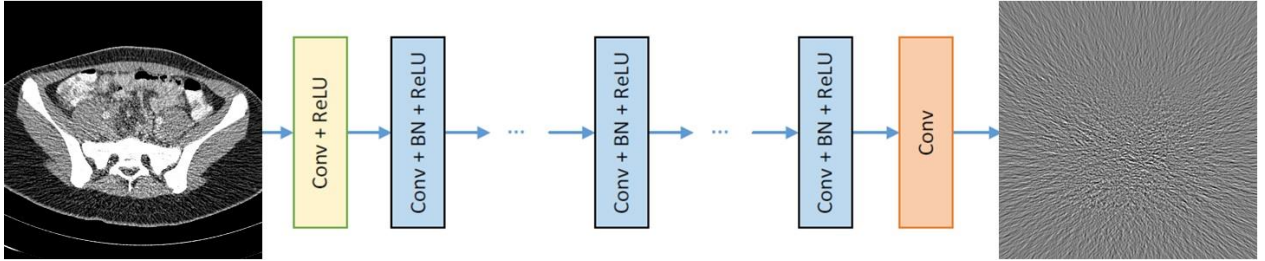


Fig. 7.2 The architecture of our proposed deep CNN

For a given depth D , we have three different layers shown in different colors in Fig. 7.2. The first layer is called Conv+ReLU which stands for a combination of convolutional (Conv) and Rectified Linear Unit (ReLU) layers. Each of these layers consists of a standard ReLU ($\max(0, \cdot)$) function and 64 filters of size 3×3 used to generate 64 feature maps. Conv+BN+ReLU are the next

$(D - 2)$ layers consisting of 64 filters of size 3×3 used to generate 64 feature maps, batch normalization, and ReLU. The last Conv layer consists of a filter of size 3×3 to reconstruct the residual image output.

For our optimization problem, we use a mini batch Stochastic gradient descent (SGD) method known as ADAM [114]. The main advantage of using Adam’s SGD algorithm is that the hyperparameters have intuitive interpretations and they require minimal tuning. Adam optimization with batch normalization and residual learning paradigm have shown to produce faster convergence and better denoising performance for Gaussian noise compared to other state-of-the-art denoising networks [111].

7.2 Experiments

7.2.1 CT Noise Model

The noise model for this study was developed by Dr. Bruce R. Whiting with the support of the NIH grant “Measuring the Impact of Noise on CT Readers”, 5-R01-EB019135-03. The overall noise consists stochastic acquisition noise [38] (both quantum and electronic) since these kinds of noise are directly related to radiation exposure. The basic acquisition noise model in sinogram domain can be treated as a random point process due to little temporal and spatial correlation between measurements [115, 116]. However, in X-ray CT image domain, the noise model is non-local and correlated over many pixels, which makes the standard denoising algorithms like BM3D and WNNM quite ineffective [117].

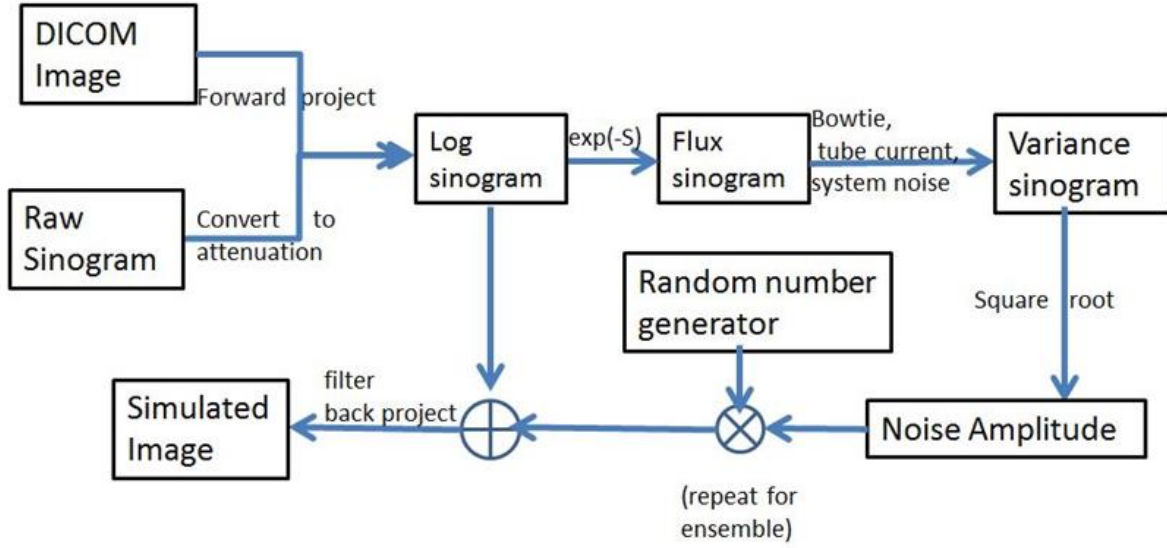


Fig. 7.3 Noise simulation flowchart

The amount of synthetic noise β added to the high dose image is computed by equating the Noise equivalent quanta (NEQ) of the target low-dose scan image (reduced by a predetermined factor ρ) depicted by the LHS of equation (7.3) to the NEQ of the high dose depicted by the RHS of equation (7.3) with some added noise β .

$$\frac{q^2}{q + \beta(g, d, \rho) + \beta_s} = \frac{(q\rho)^2}{q\rho + \beta_s}, \quad (7.3)$$

where q is the flux, β_s is the system noise, g is the gantry index, and d is detector index. The magnitude of $\beta(g, d, \rho)$ can be reformulated as done previously [118],

$$\beta(g, d, \rho) = p(d) \times Q_0 \times T(g, d) \times \left(\frac{1}{\rho} - 1\right) + \beta_s \times \left(\frac{1}{\rho^2} - 1\right) \quad (7.4)$$

where p is the bowtie profile, Q_0 is flux, and T represents tube current modulation. In the data flow described in Fig. 7.3, synthetic noise is injected to create a simulated image. In order to create an ensemble, the random noise generation step is repeated for every image slice.

7.2.2 Training and Testing Data

The data used in this study were collected as a part of the NIH grant “Measuring the Impact of Noise on CT Readers”, 5-R01-EB019135-03, Bruce R. Whiting P.I. We have collected X-ray CT images consisting of 60 appendicitis cases and 60 non-appendicitis cases from Siemens Somatom Definition AS scanner. Scan parameters: tube current = 180 mAs; pitch = 0.75; collimation = 19×0.6 mm. Each of these 3D X-ray image volumes on average consists of 400 slices. However, we have only used 20 non-appendicitis cases and 20 appendicitis cases with a total of ~ 16000 image slices. The noise level introduced in the image was varied using the parameter ρ using the equation (7.4). The choice of the parameter ρ was selected from the noise observer study with a small random fluctuation. We use a patch size of 40×40 and crop 128×1600 patches to train the model.

For testing our deep CNN denoising performance, we use 3 new appendicitis cases out of the remaining 20 appendicitis cases. We initialize the weights by the method in [119] and use Adam’s SGD with weight decay of 0.0001, a momentum of 0.9 and a mini-batch size of 128. We train 50 epochs for our deep CNN models. The learning rate was decayed exponentially from $1e^{-1}$ to $1e^{-4}$ for the 50 epochs. We use the MatConvNet package [120] to train the proposed deep CNN models.

All the experiments were carried out using the MATLAB (R2017b) environment running on a PC with 8-core Intel *i7 – 5960X* (3.0 GHz, 1333 MHz front-side bus), 64 GB RAM (1.2 GHz) and a NVIDIA TITAN X GPU. It takes about one and a half day to run our algorithm for 50 epochs on the specified dataset.

7.2.3 Compared Methods

We compared the proposed deep CNN method with two state-of-the-art, non-local similarity-based denoising methods: BM3D [101] and WNNM [102]. In BM3D, the image denoising is based on nonlocal image modeling, principal component analysis, and local shape-adaptive anisotropic estimation. The nonlocal image modeling was exploited by grouping similar image patches in 3-D groups. WNNM algorithm on the other hand, iteratively found an analytical fixed-point solution of the data fidelity term constructed over the noisy image and approximate low-noise solution. Experimental results clearly showed that the proposed WNNM algorithm outperformed BM3D in terms of both quantitative measure and visual perception quality. The implementation codes were downloaded from the authors' websites and the default parameter settings were used in our experiments.

7.3 Results

In order to compare the performance of our Deep CNN based denoising technique with other existing methods, we use Peak signal-to-noise ratio (PSNR), Structural similarity (SSIM), and Root mean square error (RMSE) as image quality metrics. Given a high dose (clean) image K of size $M \times N$, and it's denoised estimate I , the RMSE is defined as

$$RMSE = \sqrt{\frac{1}{MN} \sum_{i=1}^M \sum_{j=1}^N (K(i,j) - I(i,j))^2}. \quad (7.5)$$

If we define the maximum intensity of the denoised image as MAX_I , then PSNR can (in dB) is defined as:

$$PSNR = 10 \cdot \log_{10} \left(\frac{MAX_I^2}{\frac{1}{MN} \sum_{i=1}^M \sum_{j=1}^N (K(i,j) - I(i,j))^2} \right). \quad (7.6)$$

$$PSNR = 20 \cdot \log_{10}(MAX_I) - 10 \cdot \log_{10} \left(\frac{1}{MN} \sum_{i=1}^M \sum_{j=1}^N (K(i,j) - I(i,j))^2 \right). \quad (7.7)$$

The difference between SSIM and other techniques mentioned previously such as RMSE or PSNR is that these approaches estimate absolute errors; while, SSIM is a perception-based method that incorporates perceptual phenomena such as luminance masking, and contrast masking terms. SSIM considers image degradation as a perceived change in structural information. Structural information is based on the concept that when pixels are spatially close to each other, they have strong interdependencies. These dependencies carry important information about the structure of

the objects in the visual scene. Luminance masking is a phenomenon whereby image distortions tend to be less visible in bright regions, while contrast masking is a phenomenon whereby distortions become less visible where there is a significant activity or "texture" in the image. The SSIM index is calculated on various windows of an image. The measure between two windows x and y of common size $N \times N$ is:

$$SSIM(x, y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)}, \quad (7.8)$$

where μ_x and μ_y are the means of all the pixels in the window x and y respectively, σ_x^2 and σ_y^2 are the variance in the windows x and y respectively, and σ_{xy} is the covariance between x and y . $c_1 = (k_1L)^2$ and $c_2 = (k_2L)^2$ are used to stabilize the division with weak (small) denominator. L is the dynamic range of the pixel-values (typically this is $2^{\text{\#bits per pixel}} - 1$). The default values of k_1 and k_2 are 0.01 and 0.03 respectively.

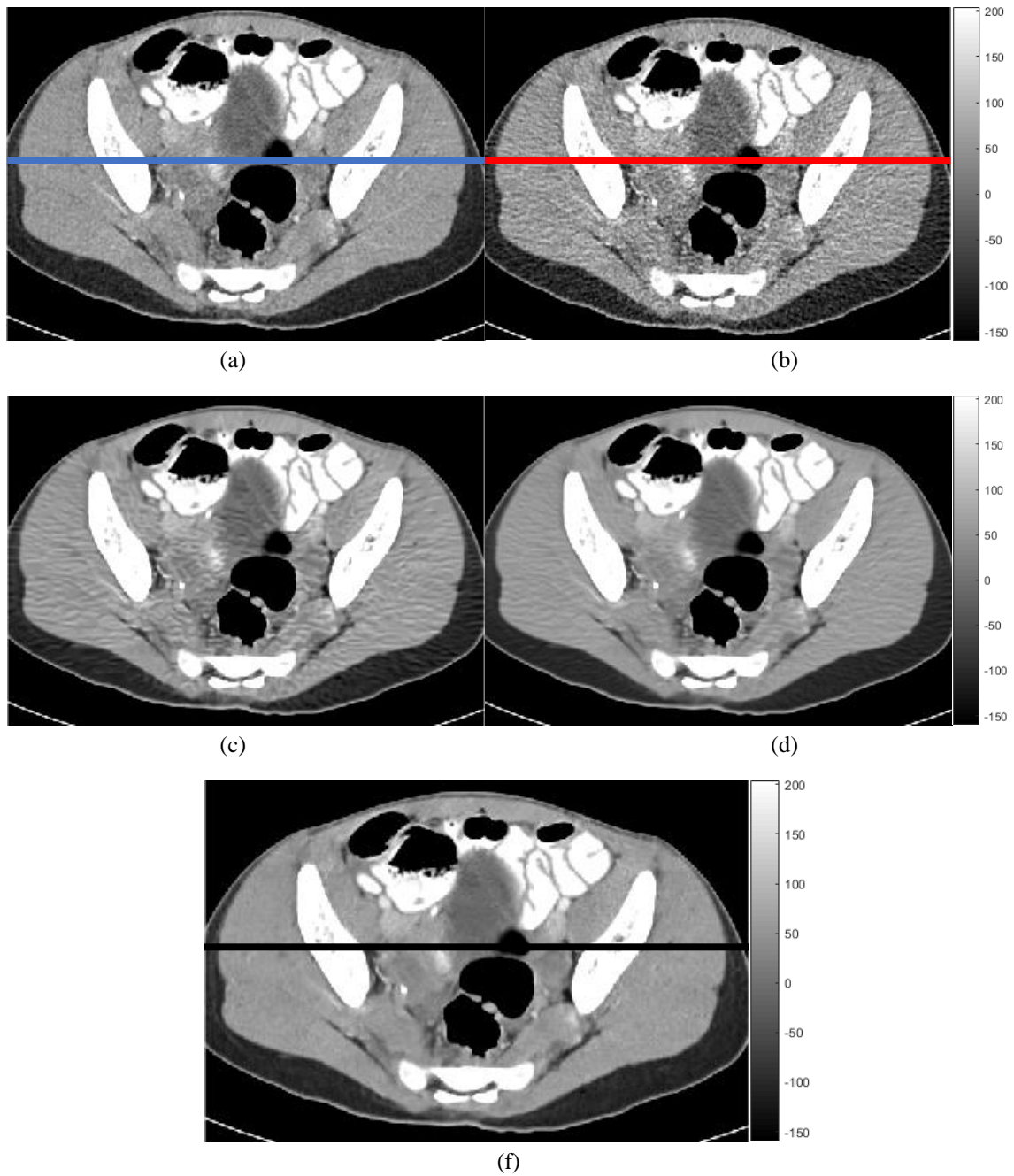


Fig. 7.4 (a) Clinical abdominal image collected from Siemens Somatom Definition AS scanner. Voxel size $=0.576 \times 0.576 \times 1.0$ mm. Scan parameters: 180 mAs, pitch 0.75, 19×0.6 mm collimation. The abdominal display window is -160 HU to 240 HU. (b) Low-dose noisy image. (c) Denoised image BM3D algorithm. (d) Denoised image with WNNM algorithm. (e) Denoised image with our proposed Deep CNN based method.

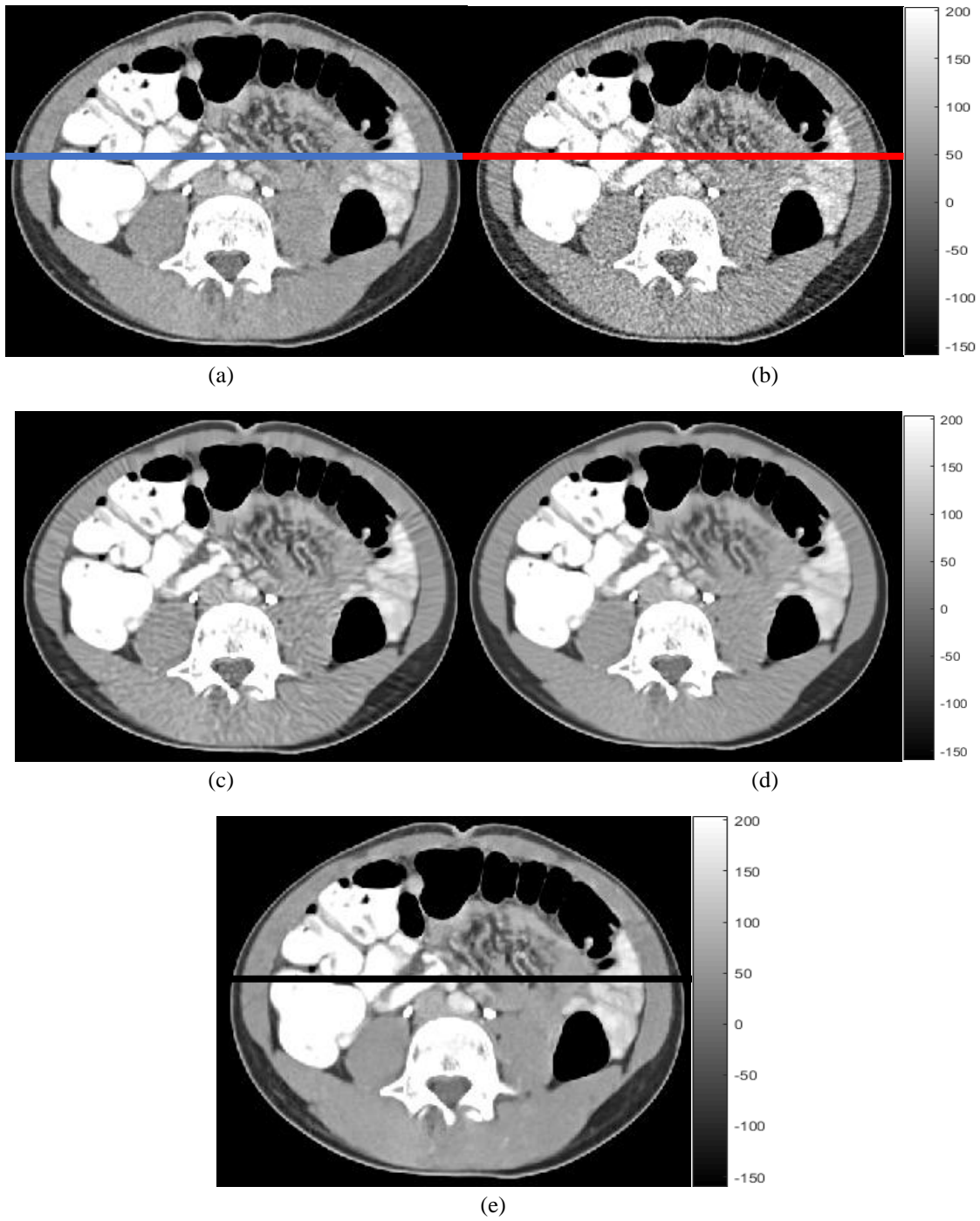


Fig. 7.5 (a) Clinical abdominal image collected from Siemens Somatom Definition AS scanner. Voxel size $= 0.576 \times 0.576 \times 1.0$ mm. Scan parameters: 180 mAs, pitch 0.75, 19×0.6 mm collimation. The abdominal display window is -160 HU to 240 HU. (b) Low-dose noisy image. (c) Denoised image BM3D algorithm. (d) Denoised image with WNNM algorithm. (e) Denoised image with our proposed Deep CNN based method.

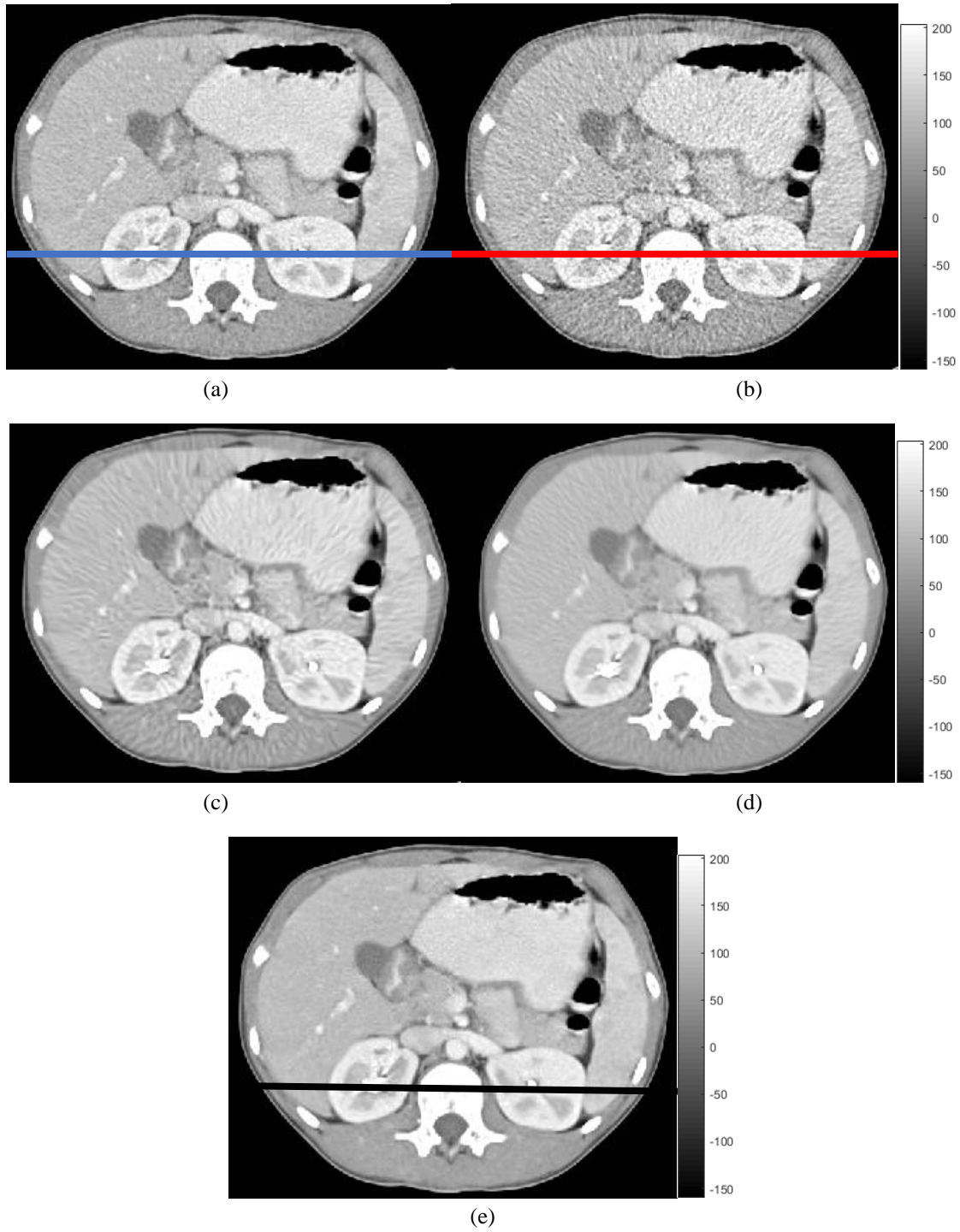


Fig. 7.6 (a) Clinical abdominal image collected from Siemens Somatom Definition AS scanner. Voxel size $= 0.576 \times 0.576 \times 1.0$ mm. Scan parameters: 180 mAs, pitch 0.75, 19×0.6 mm collimation. The abdominal display window is -160 HU to 240 HU. (b) Low-dose noisy image. (c) Denoised image BM3D algorithm. (d) Denoised image with WNNM algorithm. (e) Denoised image with our proposed Deep CNN based method.

Figure No. 7.4	PSNR (dB)	SSIM	RMSE
BM3D	25.2158	0.9121	13.9878
WNNM	25.6885	0.909	13.2470
Deep CNN	27.1579	0.9225	11.1853

Figure No. 7.5	PSNR (dB)	SSIM	RMSE
BM3D	25.6644	0.9452	13.2837
WNNM	25.9677	0.9398	12.8279
Deep CNN	27.5299	0.9514	10.7163

Figure No. 7.6	PSNR (dB)	SSIM	RMSE
BM3D	26.3476	0.9562	12.2789
WNNM	26.39	0.9529	12.2131
Deep CNN	27.9087	0.9614	10.2591

Table 7.1 The PSNR(dB), SSIM, and RMSE values for the 3 image slices shown in the figures previously.

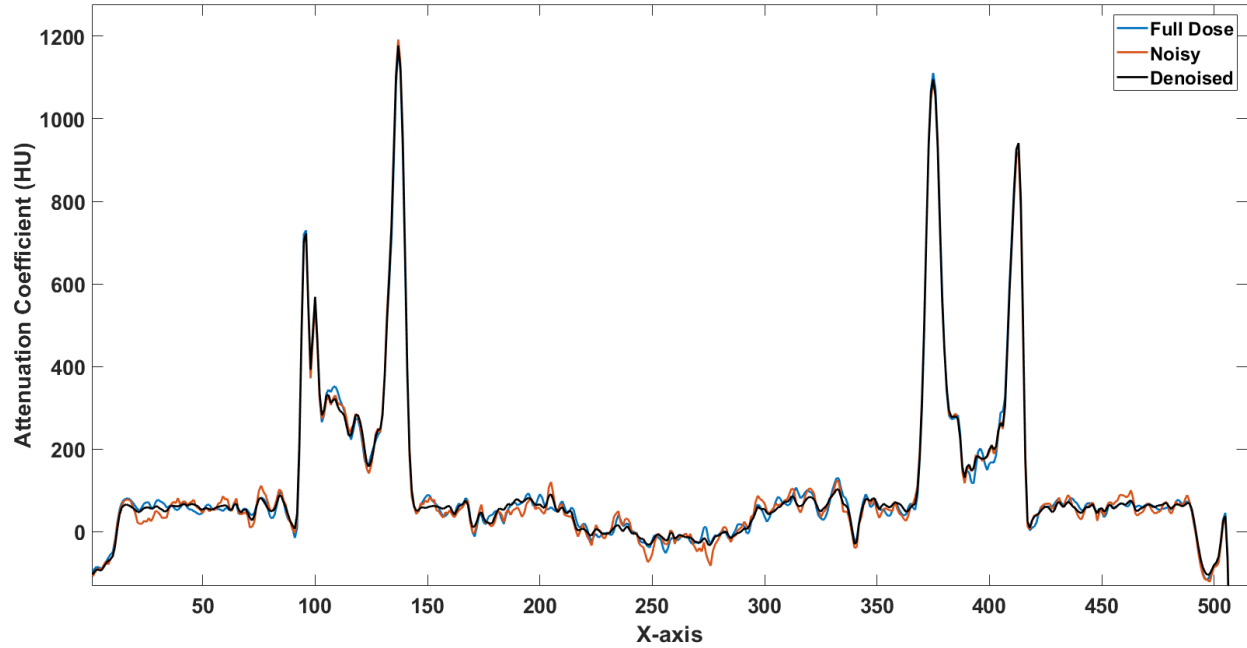


Fig. 7.7 Intensity profile along the lines in Fig. 7.4

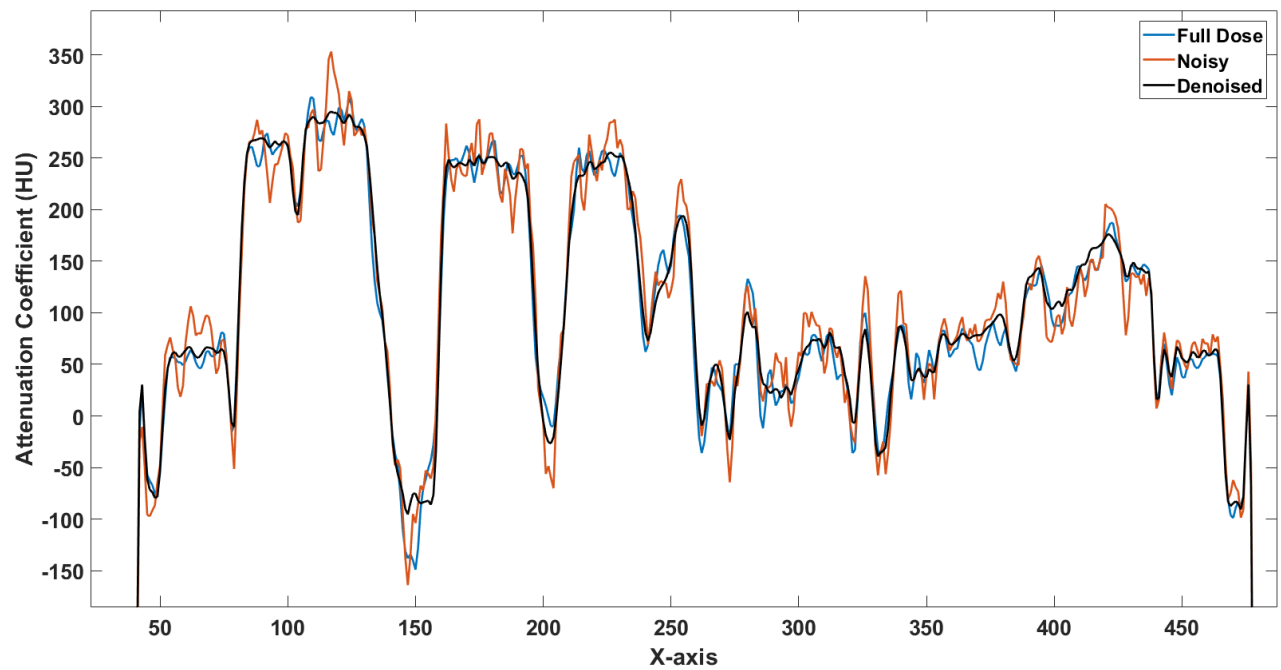


Fig. 7.8 Intensity profile along the lines in Fig. 7.5

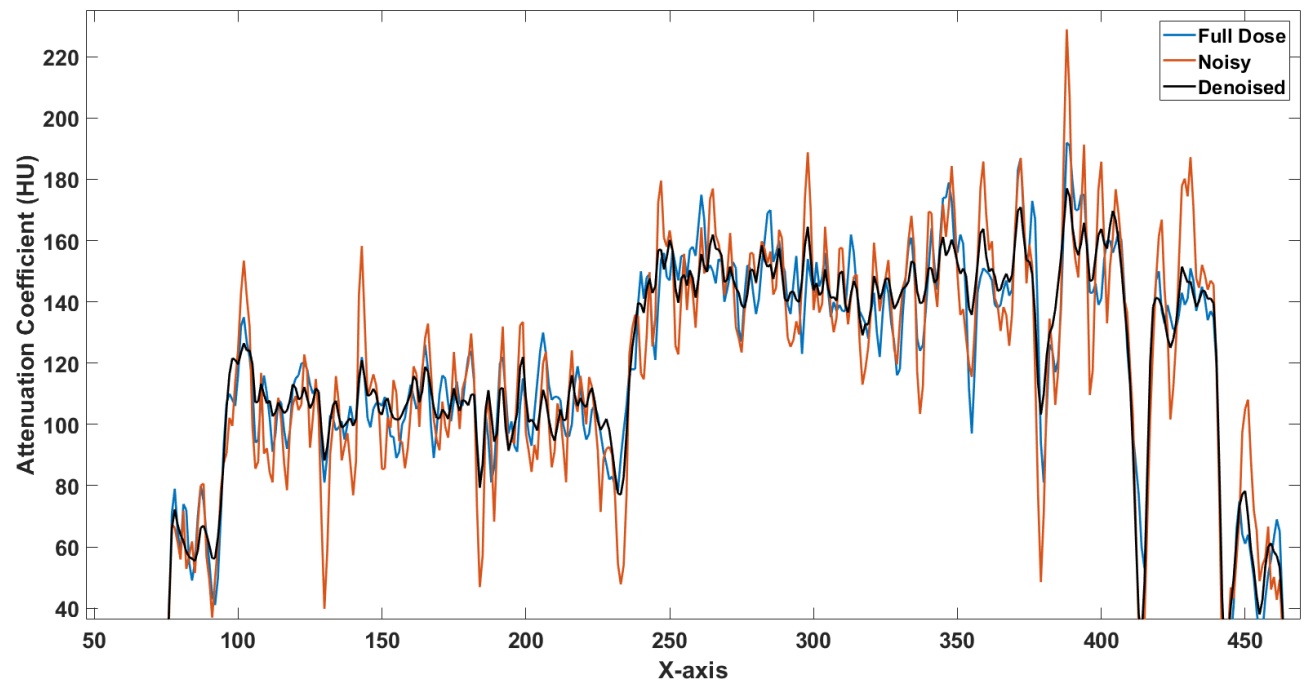


Fig. 7.9 Intensity profile along the lines in Fig. 7.6

7.4 Conclusion

Extensive experimental results have demonstrated that the proposed method produces superior image denoising performance in terms of RMSE, PSNR and SSIM metrics compared to traditional methods like BM3D and WNNM. Our deep CNN learns to distinguish the structural information of the object from various noise intensity. However, it should be noted that some texture information may be lost as demonstrated in Fig. 7.4 (e). Further validation maybe required through reader study to conclude the clinical applicability of our Deep CNN based denoising algorithm.

Traditional methods like BM3D and WNNM lose image resolution when the noise in the image is strong as shown in Fig. 7.4 and Fig. 7.6. BM3D and WNNM methods work best for Fig. 7.6. One possible explanation for this observation is that most of the abdominal image consists of soft tissue and both BM3D and WNNM work best for a uniform tissue region. From Fig. 7.6, we can demonstrate that our deep CNN is able to describe the details of the vessels in the liver. In Fig. 7.7, 7.8, and 7.9 our proposed network suitably reduces noise and describes the peak points. On average, over the 3 independent test cases consisting of 1328 image slices in total, our proposed denoising method outperforms BM3D method by almost 2.2dB and WNNM method by 1.7dB. The SSIM and RMSE metric also shows better performance with our denoising method.

In addition to visual quality, another important aspect of an image restoration method is the testing speed. We use the NVIDIA cuDNN-v5 deep learning library to accelerate the GPU computation of the proposed Deep CNN. We have ignored data transfer between CPU and GPU from our model

execution time. With a single TITAN X GPU, we can run our Deep CNN based denoising algorithm on an 512×512 image with an average time of 53ms whereas BM3D takes on average 2.85s and WNNM takes on average 773s on a CPU. With GPU acceleration, BM3D may run slightly faster than our Deep CNN implementation, however, the image quality enhancement is significantly better with our method.

Chapter 8: Conclusions and Future Work

In this work, we have developed multislice fully 3D spiral/helical X-ray CT reconstruction algorithm for both analytical and statistical methods. For statistical reconstruction, we have compared our estimated projection data against the measured data to compute the next image iterate estimate. If we could formulate an accurate system model, the reconstructed images would have little bias. However, we have tried to match the system matrix for projection and backprojection algorithms to ensure that they are the exact transpose of each other. We have validated this claim by running our alternating minimization algorithm without any ordered subset for 5000 iterations on the clinically-sized dataset. After 5000 iterations, we have seen a steady pattern of increase in the objective function. However, this pattern changes when we switch our computation to double precision. Hence, we can conclude that use of single-precision floating-point arithmetic in the image-estimate step creates rounding errors which make the objective function diverge.

The raw CT data derived from the scanner has been preprocessed to mitigate the effects of detector sensitivity variation, beam hardening, and X-ray tube current modulation. The main focus of our work has mostly been devoted to accurate reconstruction models rather than the preprocessing steps. We have split the measured data and image volume into different CPU cores and GPU devices in such a way that the overhead due to memory and device synchronization, and data transfer is minimized. We have also ensured every CPU core and GPU device performs the exact amount of computations so that the ordered subsets (OS) can be executed in a computationally

efficient way. We have also proposed novel surrogate function, which decreases our original objective function faster than Jensen-type surrogate functions used in previous literature. We have shown that ordered subsets along with adaptive surrogate functions can significantly decrease the convergence rate. We have met the convergence criteria in 60 – 80 iterations using the adaptive surrogate function and a large number (29) of ordered subsets. Although the total computation time for the converged image using our methods on a clinically-sized dataset is 900 – 1000 times higher than analytical methods like FDK, it is still promising since as the total computation time is < 30 minutes.

Regarding the regularization that was added to the AM algorithm, choosing suitable regularization parameters is notoriously difficult (especially in 3D). One could possibly test a range of parameters by performing several “trial” reconstructions on a down sampled dataset, or on just a few slices, and then attempt to scale the parameters accordingly for the full-scale problem. Other more systematic methods exist but are also more computationally demanding.

We have also observed a slow convergence of high frequencies using our alternating minimization algorithm. We can see a striped pattern in coronal and sagittal view of our helical CT reconstruction. We have also observed that these stripes are inclined towards helical trajectory. As iteration progresses, these striped patterns gradually disappear. In a helical CT scan, different voxels are seen a different number of times, and as a result, they are illuminated differently which can be the reason for these helical scan artifacts. The analytical FDK algorithm takes care of these artifacts by employing different weights for different view angle and voxel as discussed in Chapter

4. However, the absence of these weighting functions in AM algorithm can cause these artifacts. Since in AM algorithm, the voxel update step is the ratio of two backprojection images, these artifacts don't cancel out since they are dependent on view angles. Weighting function similar to ones employed in FDK algorithm can be added to branchless distance-driven projection and backprojection algorithms to get rid of these artifacts in early iterations.

The adaptive auxiliary variable derived for single-energy and dual-energy reconstruction problems has some drawbacks too. The condition of guaranteed convergence is absent for these types of surrogate functions. As a result, they should be carefully applied only to initial iterations. The update step in adaptive step size derivation doesn't consider the term with penalty function. We have avoided this step due to a slightly higher computational burden. However, for accurate results in case of noisy measurement, we can modify the computation step for the adaptive update function, for mono-energy as

$$Z^{(k)}(x) = \begin{cases} \frac{\log\left(\frac{\tilde{b}_0(x)}{\tilde{b}(x)}\right)}{\hat{\mu}_A^{(k)}(x)} & \text{if } \frac{\log\left(\frac{\tilde{b}_0(x)}{\tilde{b}(x)}\right)}{\hat{\mu}_A^{(k)}(x)} < 2 * R_{recon}, \frac{\tilde{b}_0(x)}{\tilde{b}(x)} > 1, \hat{\mu}^{(k)}(x) > 0 \\ 2 * R_{recon} & \text{else} \end{cases} \quad (8.1)$$

where,

$$\hat{\mu}_A^{(k)}(x) = \hat{\mu}^{(k)}(x) + \lambda \sum_{x' \in N_x} \frac{\omega_{xx'}}{2} \delta^2 \left(\left| \frac{2\hat{\mu}^{(k)}(x)}{\delta} \right| - \log \left(1 + \left| \frac{2\hat{\mu}^{(k)}(x)}{\delta} \right| \right) \right). \quad (8.2)$$

For dual energy, we can modify our adaptive surrogate function based update step as:

$$Z_m^{(k)}(x) = \begin{cases} \frac{\log\left(\frac{\sum_{j=1}^2 \tilde{b}_{0j,m}(x)}{\sum_{j=1}^2 \tilde{b}_{j,m}^{(k)}(x)}\right)}{\hat{c}_{mA}^{(k)}(x)} & \text{if } \frac{\log\left(\frac{\sum_{j=1}^2 \tilde{b}_{0j,m}(x)}{\sum_{j=1}^2 \tilde{b}_{j,m}^{(k)}(x)}\right)}{\hat{c}_{mA}^{(k)}(x)} \leq Z \\ Z & \text{else} \end{cases} \quad (8.3)$$

$$\hat{c}_{mA}^{(k)}(x) = \hat{c}_m^{(k)}(x) + \lambda \sum_{x' \in N_x} \frac{\omega_{xx'}}{2} \delta^2 \left(\left| \frac{2\hat{c}_m^{(k)}(x)}{\delta} \right| - \log \left(1 + \left| \frac{2\hat{c}_m^{(k)}(x)}{\delta} \right| \right) \right). \quad (8.4)$$

For the deep CNN based denoising algorithm, we assume the images are 2D even though they are reconstructed from a spiral scan. As we have discussed before, the spiral scanning introduces its own artifacts. However, we haven't incorporated that into the noise model for our analysis. To the best of my knowledge, 3-D multislice X-ray CT images haven't been denoised with neural networks by other research groups. Further work is needed to model 3-D nature of the noise statistics and incorporate that into our residual denoising method.

In conclusion, the work in this dissertation has described a solid computational foundation for multi-GPU based X-ray CT reconstruction problems upon which many improved techniques can be tested in a short amount of time. It is anticipated that the multi-GPU based reconstruction and denoising methods described in this thesis will be used in future projects.

Appendix A: Derivation of the Penalized AM Algorithm

Using the convex decomposition lemma, for any convex function $f(\cdot)$, we can write

$$f[\alpha t_1 + (1 - \alpha)t_1] \leq \alpha f(t_1) + (1 - \alpha)f(t_1), \text{ where } 0 \leq \alpha \leq 1 \quad (\text{A.1})$$

Using this property,

$$\begin{aligned} \psi(\mu(x) - \mu(x')) &= \psi \left\{ \alpha \left[\frac{1}{\alpha} (\mu(x) - \hat{\mu}(x)) + (\hat{\mu}(x) - \hat{\mu}(x')) \right] \right. \\ &\quad \left. + (1 - \alpha) \left[\frac{-1}{(1 - \alpha)} (\mu(x') - \hat{\mu}(x')) + (\hat{\mu}(x) - \hat{\mu}(x')) \right] \right\} \end{aligned} \quad (\text{A.2})$$

$$\begin{aligned} &\leq \alpha \psi \left[\frac{1}{\alpha} (\mu(x) - \hat{\mu}(x)) + (\hat{\mu}(x) - \hat{\mu}(x')) \right] \\ &\quad + (1 - \alpha) \psi \left[\frac{-1}{(1 - \alpha)} (\mu(x') - \hat{\mu}(x')) + (\hat{\mu}(x) - \hat{\mu}(x')) \right]. \end{aligned} \quad (\text{A.3})$$

To simplify equation (A.3), let $\alpha \triangleq 1/2$ to obtain

$$\begin{aligned} \psi(\mu(x) - \mu(x')) &\leq \frac{1}{2} \psi[2(\mu(x) - \hat{\mu}(x)) + (\hat{\mu}(x) - \hat{\mu}(x'))] \\ &\quad + \frac{1}{2} \psi[-2(\mu(x') - \hat{\mu}(x')) + (\hat{\mu}(x) - \hat{\mu}(x'))] \end{aligned} \quad (\text{A.4})$$

$$\begin{aligned} &= \frac{1}{2} \psi[2\mu(x) - \hat{\mu}(x) - \hat{\mu}(x')] + \frac{1}{2} \psi[2\mu(x') - \hat{\mu}(x) - \hat{\mu}(x')] \\ &\quad \hat{\mu}(x')]. \end{aligned} \quad (\text{A.5})$$

We have exploited the evenness of the potential $\psi(\cdot)$ to derive equation (A.5) from equation (A.4).

We plug this surrogate for $\psi(\mu(x) - \mu(x'))$ into (3.27), and define the modified penalty function $\hat{R}(\mu)$ by ignoring the part independent of $\mu(x)$ as follows

$$\begin{aligned} \hat{R}(\mu) = \sum_x \sum_{x' \in N(x)} \frac{\omega(x, x')}{2} \delta^2 \left(\left| \frac{2\mu(x) - \hat{\mu}(x) - \hat{\mu}(x')}{\delta} \right| \right. \\ \left. - \log \left(1 + \left| \frac{2\mu(x) - \hat{\mu}(x) - \hat{\mu}(x')}{\delta} \right| \right) \right) \end{aligned} \quad (\text{A.6})$$

So, we want to solve the penalized-likelihood function as follows

$$\frac{\partial \hat{\mathcal{I}}[d||g; \mu, \hat{\mu}]}{\partial \mu(x)} + \lambda \frac{\partial \hat{R}(\mu)}{\partial \mu(x)} = 0 \quad \forall x. \quad (\text{A.7})$$

The derivative of the surrogate of I-divergence is determined to be

$$\frac{\partial \hat{\mathcal{I}}[d||g; \mu, \hat{\mu}]}{\partial \mu(x)} = \tilde{b}(x) - \hat{b}(x) \exp(-Z(\mu(x) - \hat{\mu}(x))) \quad \forall x. \quad (\text{A.8})$$

The derivative of the penalty term is

$$\frac{\partial \hat{R}(\mu)}{\partial \mu(x)} = \sum_{x'} \omega(x, x') \frac{\partial \psi(t)}{\partial t} \bigg|_{t=2\mu(x) - \hat{\mu}(x) - \hat{\mu}(x')} \quad (\text{A.9})$$

Replacing the values of the derivatives from equation (A.8) and (A.9) into (A.7) we can write,

$$\begin{aligned} \tilde{b}(x) - \hat{b}(x) \exp(-Z(\mu(x) - \hat{\mu}(x))) \\ + \sum_{x'} \omega(x, x') \delta \left(1 + \frac{1}{1 + \frac{t}{\delta}} \right) \bigg|_{t=2\mu(x) - \hat{\mu}(x) - \hat{\mu}(x')} = 0 \end{aligned} \quad (\text{A.10})$$

Since there is no closed form solution of equation (A.10) so we use Newton's method to solve for $\mu(x)$. Using newton's method, we can write

$$\begin{aligned} & \hat{\mu}^{(k+1)}(x) \\ &= \hat{\mu}^{(k)}(x) - \gamma \left[\frac{\partial^2 \left(\hat{I}[d||g; \mu, \hat{\mu}] + \lambda \hat{R}(\mu) \right)}{\partial \mu^2(x)} \right]^{-1} \frac{\partial \left(\hat{I}[d||g; \mu, \hat{\mu}] + \lambda \hat{R}(\mu) \right)}{\partial \mu(x)} \end{aligned} \quad (\text{A.11})$$

where,

$$\begin{aligned} & \frac{\partial \left(\hat{I}[d||g; \mu, \hat{\mu}] + \lambda \hat{R}(\mu) \right)}{\partial \mu(x)} \\ &= \tilde{b}(x) - \hat{b}(x) \exp(-Z(\mu(x) - \hat{\mu}(x))) \end{aligned} \quad (\text{A.12})$$

$$\begin{aligned} & + \sum_{x'} \omega(x, x') \delta \left(1 + \frac{1}{1 + \frac{t}{\delta}} \right) \Bigg|_{t=2\mu(x) - \hat{\mu}(x) - \hat{\mu}(x')} \\ & \frac{\partial^2 \left(\hat{I}[d||g; \mu, \hat{\mu}] + \lambda \hat{R}(\mu) \right)}{\partial \mu^2(x)} \\ &= -Z \hat{b}(x) \exp(-Z(\mu(x) - \hat{\mu}(x))) \\ & - \sum_{x'} \omega(x, x') \frac{1}{\left(1 + \frac{t}{\delta} \right)^2} \Bigg|_{t=2\mu(x) - \hat{\mu}(x) - \hat{\mu}(x')}. \end{aligned} \quad (\text{A.13})$$

The γ term represents step size.

Appendix B: Derivation of Decoupled Dual-energy Surrogate Function

For the derivation of the decoupled dual-energy surrogate function, we start with our original goal of minimizing I-divergence over $c_{m,j} \geq 0$,

$$I[d||g] \triangleq \sum_{j=1}^2 \sum_y \left(d_j(y) \log \frac{d_j(y)}{g_j(y:c)} + g_j(y:c) - d_j(y) \right), \quad (\text{B.1})$$

where

$$g_j(\mu, c) \triangleq \sum_E q_j(y:E) \quad (\text{B.2})$$

$$\begin{aligned} \varepsilon_j = \left\{ q_j: q_j(y, E) = I_{0j}(y, E) \exp \left(- \sum_x \sum_m h(y|x) \mu_m(E) c_m(x) \right), E \right. \\ \left. \neq 0, q_j(y, 0) = \beta_j(y) \right\}. \end{aligned} \quad (\text{B.3})$$

The exponential family ε_j defines the model used for the data.

The main difficulty in solving the original objective function denoted by equation (B.1) is the summation over all the energies inside the “log” denominator. In order to decouple our computation of the summation over energy part, we would need to move the denominator part out of logarithm.

Lemma B.0.1 *The I-divergence (B.1) can be written in the variational form*

$$I[d||g] = \min_{p_j \in \mathcal{L}(d_j)} \sum_{j=1}^2 I[p_j||q_j], \quad (\text{B.4})$$

where

$$I[p_j||q_j] = \sum_E \sum_y \left(p_j(y, E) \log \frac{p_j(y, E)}{q_j(y, E)} + q_j(y, E) - p_j(y, E) \right), \quad (\text{B.5})$$

$$\mathcal{L}(d_j) = \left\{ p_j(y, E) \geq 0: \sum_E p_j(y, E) = d_j(y) \right\}. \quad (\text{B.6})$$

In order to prove this lemma, we start with Lagrange multipliers to enforce equality in equation (B.6).

$$\begin{aligned} L_j = \sum_E \sum_y \left(p_j(y, E) \log \frac{p_j(y, E)}{q_j(y, E)} + q_j(y, E) - p_j(y, E) \right) \\ + \lambda_j(y) \left(\sum_E p_j(y, E) - d_j(y) \right). \end{aligned} \quad (\text{B.7})$$

Minimizing over $p_j(y, E)$ and solving for $\lambda_j(y)$ to enforce the equality in Equation (B.6) yields $p_j(y, E) = 0$ if $q_j(y, E) = 0$ (defining $I[0||0] = 0$) and if $q_j(y, E) \neq 0$

$$p_j(y, E) = d_j(y) \frac{q_j(y, E)}{\sum_{E'} q_j(y, E')}. \quad (\text{B.8})$$

Substituting this expression of $p_j(y, E)$ back into the I-divergence in equation (B.5) produces the lemma B.0.1.

Therefore, we can express the original maximum-likelihood estimation problem in (B.1) as a double minimization problem over the exponential and linear family with the inequality constraint $c_m(x) \geq 0$ for all (m, x) . However, there is still difficulty inside the exponential term in equation (B.3) since the optimization space is really large. So, to tackle this issue, we employ the following convex decomposition lemma.

Lemma B.0.2 *Suppose that f is a convex function defined on a convex cone $\mathcal{D} \subset \mathbb{R}^n$. Given $x_i \in \mathcal{D}, i = 1, 2, \dots,$*

$$f\left(\sum_i x_i\right) \leq \sum_i r_i f\left(\frac{1}{r_i} x_i\right) \quad (\text{B.9})$$

for all $r \in P$, with $r_i > 0$ for all i . If f is strictly convex, equality holds if and only if $(1/r_i)x_i = x$ is independent of i .

By applying Lemma B.0.2 to our objective function in (B.5), we have

$$\begin{aligned}
& \sum_{j=1}^2 \sum_y \sum_E \sum_x \sum_m \hat{p}_j(y, E) h(y|x) \mu_m(E) c_m(x) \\
& + \sum_{j=1}^2 \sum_E \sum_y I_{0j}(y, E) \exp \left(- \sum_m \sum_x h(y|x) \mu_m(E) c_m(x) \right) \\
& = \sum_{j=1}^2 \sum_y \sum_E \sum_x \sum_m \hat{p}_j(y, E) h(y|x) \mu_m(E) c_m(x) \tag{B.10}
\end{aligned}$$

$$\begin{aligned}
& + \sum_{j=1}^2 \sum_E \sum_y \hat{q}_j(y, E) \exp \left(- \sum_m \sum_x h(y|x) \mu_m(E) (\hat{c}_m(x) \right. \\
& \quad \left. - c_m(x)) \right), \\
& \leq \sum_{j=1}^2 \sum_y \sum_E \sum_x \sum_m \left\{ \hat{p}_j(y, E) h(y|x) \mu_m(E) c_m(x) \right. \\
& \quad \left. + r(x, m|y, E) \hat{q}_j(y, E) \exp \left[\frac{h(y|x) \mu_m(E)}{r(x, m|y, E)} (\hat{c}_m(x) - c_m(x)) \right] \right\}, \tag{B.11}
\end{aligned}$$

for all $r(x, m|y, E) > 0$ such that

$$\sum_x \sum_{m=1}^M r(x, m|y, E) \leq 1 \forall (y, E). \tag{B.12}$$

Note the inequality in (B.11); this minor extension of the convex decomposition lemma is valid due to the possibility of adding a dummy x variable (again denoted 0) such that $\hat{c}_m(0) - c_m(0) = 0$ for each m . Equality is achieved in (B.11) if $\frac{h(y|x) \mu_m(E)}{r(x, m|y, E)} (\hat{c}_m(x) - c_m(x))$ is only a function of

(y, E) . One clear possibility for this is if the algorithm converges and $\hat{c}_m(x) = c_m(x)$. To derive an alternating minimization algorithm for X-ray transmission CT, set

$$r(x, m|y, E) = \frac{h(y|x)\mu_m(E)}{Z_m(x)}, \quad (\text{B.13})$$

where $Z_m(x)$ are chosen to enforce the constraint (B.11). In general, the $Z_m(x)$ must be large enough, one such choice being

$$Z_m(x) = Z_0 = \max_{y,E} \sum_x \sum_{m=1}^M h(y|x)\mu_m(E). \quad (\text{B.14})$$

The resulting decoupled objective function is

$$\begin{aligned} \sum_{j=1}^2 \sum_y \sum_E \sum_x \sum_m \left\{ \hat{p}_j(y, E) h(y|x) \mu_m(E) c_m(x) \right. \\ \left. + \frac{\hat{q}_j(y, E) h(y|x) \mu_m(E)}{Z_m(x)} \exp[Z_m(x)(\hat{c}_m(x) - c_m(x))] \right\}. \end{aligned} \quad (\text{B.15})$$

References

- [1] J. A. O'Sullivan and J. Benac, "Alternating minimization algorithms for transmission tomography," *IEEE Transactions on Medical Imaging*, vol. 26, no. 3, pp. 283-297, 2007.
- [2] A. C. Kak and M. Slaney, *Principles of computerized tomographic imaging*. SIAM, 2001.
- [3] A. R. Brodtkorb, T. R. Hagen, and M. L. Sætra, "Graphics processing unit (GPU) programming strategies and trends in GPU computing," *Journal of Parallel and Distributed Computing*, vol. 73, no. 1, pp. 4-13, 2013.
- [4] I. A. Elbakri and J. A. Fessler, "Statistical image reconstruction for polyenergetic X-ray computed tomography," *IEEE transactions on medical imaging*, vol. 21, no. 2, pp. 89-99, 2002.
- [5] K. Li, J. Tang, and G. H. Chen, "Statistical model based iterative reconstruction (MBIR) in clinical CT systems: experimental assessment of noise performance," *Medical physics*, vol. 41, no. 4, 2014.
- [6] P. J. Pickhardt *et al.*, "Abdominal CT with model-based iterative reconstruction (MBIR): initial results of a prospective trial comparing ultralow-dose with standard-dose imaging," *American journal of roentgenology*, vol. 199, no. 6, pp. 1266-1274, 2012.
- [7] X. Zhao, J.-j. Hu, and P. Zhang, "GPU-based 3D cone-beam CT image reconstruction for large data volume," *Journal of Biomedical Imaging*, vol. 2009, p. 8, 2009.
- [8] X. Jia, B. Dong, Y. Lou, and S. B. Jiang, "GPU-based iterative cone-beam CT reconstruction using tight frame regularization," *Physics in Medicine & Biology*, vol. 56, no. 13, p. 3787, 2011.
- [9] X. Jia *et al.*, "GPU-based fast low-dose cone beam CT reconstruction via total variation," *Journal of X-ray science and technology*, vol. 19, no. 2, pp. 139-154, 2011.
- [10] M. Schellmann, T. Kesters, and S. Gorlatch, "Parallelization and runtime prediction of the listmode osem algorithm for 3d pet reconstruction," in *Nuclear Science Symposium Conference Record, 2006. IEEE*, 2006, vol. 4, pp. 2190-2195: IEEE.
- [11] M. Tianyu, Z. Rong, and J. Yongjie, "Communication optimization and auto load balancing in parallel OSEM algorithm for fully 3-D SPECT reconstruction," in *Nuclear Science Symposium Conference Record, 2005 IEEE*, 2005, vol. 5, pp. 2695-2699: IEEE.
- [12] M. D. Jones and R. Yao, "Parallel programming for OSEM reconstruction with MPI, OpenMP, and hybrid MPI-OpenMP," in *Nuclear Science Symposium Conference Record, 2004 IEEE*, 2004, vol. 5, pp. 3036-3042: IEEE.

- [13] C. A. Johnson and A. Sofer, "A data-parallel algorithm for iterative tomographic image reconstruction," in *Frontiers of Massively Parallel Computation, 1999. Frontiers' 99. The Seventh Symposium on the*, 1999, pp. 126-137: IEEE.
- [14] M. Knaup, W. A. Kalender, and M. Kachelrieß, "Statistical cone-beam CT image reconstruction using the cell broadband engine," in *Nuclear Science Symposium Conference Record, 2006. IEEE*, 2006, vol. 5, pp. 2837-2840: IEEE.
- [15] M. Kachelrieß, M. Knaup, and O. Bockenbach, "Hyperfast parallel-beam and cone-beam backprojection using the cell general purpose hardware," *Medical Physics*, vol. 34, no. 4, pp. 1474-1486, 2007.
- [16] T. M. Benson and J. Gregor, "Framework for iterative cone-beam micro-CT reconstruction," *IEEE transactions on nuclear science*, vol. 52, no. 5, pp. 1335-1340, 2005.
- [17] J. Kole and F. Beekman, "Parallel statistical image reconstruction for cone-beam x-ray CT on a shared memory computation platform," *Physics in Medicine & Biology*, vol. 50, no. 6, p. 1265, 2005.
- [18] S. Steckmann, M. Knaup, and M. Kachelrieß, "Algorithm for hyperfast cone-beam spiral backprojection," *Computer methods and programs in biomedicine*, vol. 98, no. 3, pp. 253-260, 2010.
- [19] B. Jang, D. Kaeli, S. Do, and H. Pien, "Multi GPU implementation of iterative tomographic reconstruction algorithms," in *Biomedical Imaging: From Nano to Macro, 2009. ISBI'09. IEEE International Symposium on*, 2009, pp. 185-188: IEEE.
- [20] N. Gac, S. Mancini, M. Desvignes, and D. Houzet, "High speed 3D tomography on CPU, GPU, and FPGA," *EURASIP Journal on Embedded systems*, vol. 2008, p. 5, 2008.
- [21] J. Kole and F. J. Beekman, "Evaluation of accelerated iterative x-ray CT image reconstruction using floating point graphics hardware," *Physics in Medicine & Biology*, vol. 51, no. 4, p. 875, 2006.
- [22] F. Xu and K. Mueller, "Real-time 3D computed tomographic reconstruction using commodity graphics hardware," *Physics in Medicine & Biology*, vol. 52, no. 12, p. 3405, 2007.
- [23] H. Scherl, B. Keck, M. Kowarschik, and J. Hornegger, "Fast GPU-based CT reconstruction using the common unified device architecture (CUDA)," in *Nuclear Science Symposium Conference Record, 2007. NSS'07. IEEE*, 2007, vol. 6, pp. 4464-4466: IEEE.
- [24] J. D. Owens, M. Houston, D. Luebke, S. Green, J. E. Stone, and J. C. Phillips, "GPU computing," *Proceedings of the IEEE*, vol. 96, no. 5, pp. 879-899, 2008.

- [25] A. Andreyev, A. Sitek, and A. Celler, "Acceleration of blob-based iterative reconstruction algorithm using Tesla GPU," in *Nuclear Science Symposium Conference Record (NSS/MIC), 2009 IEEE*, 2009, pp. 4095-4098: IEEE.
- [26] M. G. McGaffin and J. A. Fessler, "Alternating dual updates algorithm for X-ray CT reconstruction on the GPU," *IEEE transactions on computational imaging*, vol. 1, no. 3, pp. 186-199, 2015.
- [27] M. Wu and J. A. Fessler, "GPU acceleration of 3D forward and backward projection using separable footprints for X-ray CT image reconstruction," in *Proc. Intl. Mtg. on Fully 3D Image Recon. in Rad. and Nuc. Med*, 2011, vol. 6, p. 021911.
- [28] F. Quivira, S. Bedford, R. Moore, J. Beaty, and D. Castañón, "Sparse Data 3-D X-ray reconstructions on GPU processors," *Electronic Imaging*, vol. 2016, no. 19, pp. 1-5, 2016.
- [29] S. Degirmenci, D. G. Politte, C. Bosch, N. Tricha, and J. A. O'Sullivan, "Acceleration of iterative image reconstruction for x-ray imaging for security applications," in *Computational Imaging*, 2015, p. 94010C.
- [30] S. Degirmenci, *A General Framework of Large-Scale Convex Optimization Using Jensen Surrogates and Acceleration Techniques*. Washington University in St. Louis, 2016.
- [31] C. Bosch, S. Degirmenci, J. Barlow, A. Mesika, D. G. Politte, and J. A. O'Sullivan, "Optimizing convergence rates of alternating minimization reconstruction algorithms for real-time explosive detection applications," in *SPIE Defense+ Security*, 2016, pp. 98470P-98470P-17: International Society for Optics and Photonics.
- [32] Q. Xu, D. Yang, J. Tan, A. Sawatzky, and M. A. Anastasio, "Accelerated fast iterative shrinkage thresholding algorithms for sparsity-regularized cone-beam CT image reconstruction," *Medical physics*, vol. 43, no. 4, pp. 1849-1872, 2016.
- [33] R. A. Brooks and G. Di Chiro, "Beam hardening in x-ray reconstructive tomography," *Physics in medicine & biology*, vol. 21, no. 3, p. 390, 1976.
- [34] A. C. Kak and M. Slaney, *Principles of computerized tomographic imaging*. IEEE press, 1988.
- [35] M. Levoy, *Volume rendering using the Fourier projection-slice theorem*. Computer Systems Laboratory, Stanford University, 1992.
- [36] J. A. Fessler, "Statistical image reconstruction methods for transmission tomography," *Handbook of medical imaging*, vol. 2, pp. 1-70, 2000.
- [37] K. Lange, "Convergence of EM image reconstruction algorithms with Gibbs smoothing," *IEEE transactions on medical imaging*, vol. 9, no. 4, pp. 439-446, 1990.

- [38] B. R. Whiting, P. Massoumzadeh, O. A. Earl, J. A. O'Sullivan, D. L. Snyder, and J. F. Williamson, "Properties of preprocessed sinogram data in x-ray computed tomography," *Medical physics*, vol. 33, no. 9, pp. 3290-3303, 2006.
- [39] G. M. Lasio, B. R. Whiting, and J. F. Williamson, "Statistical reconstruction for x-ray computed tomography using energy-integrating detectors," *Physics in medicine & biology*, vol. 52, no. 8, p. 2247, 2007.
- [40] K. Lange and R. Carson, "EM reconstruction algorithms for emission and transmission tomography," *J Comput Assist Tomogr*, vol. 8, no. 2, pp. 306-16, 1984.
- [41] E. U. Mumcuoglu, R. Leahy, S. R. Cherry, and Z. Zhou, "Fast gradient-based methods for Bayesian reconstruction of transmission and emission PET images," *IEEE transactions on Medical Imaging*, vol. 13, no. 4, pp. 687-701, 1994.
- [42] C. A. Bouman and K. Sauer, "A unified approach to statistical tomography using coordinate descent optimization," *IEEE Transactions on image processing*, vol. 5, no. 3, pp. 480-492, 1996.
- [43] A. K. Hara, R. G. Paden, A. C. Silva, J. L. Kujak, H. J. Lawder, and W. Pavlicek, "Iterative reconstruction technique for reducing body radiation dose at CT: feasibility study," *American Journal of Roentgenology*, vol. 193, no. 3, pp. 764-771, 2009.
- [44] P. Prakash *et al.*, "Reducing abdominal CT radiation dose with adaptive statistical iterative reconstruction technique," *Investigative radiology*, vol. 45, no. 4, pp. 202-210, 2010.
- [45] G. S. Desai, R. N. Uppot, W. Y. Elaine, A. R. Kambadakone, and D. V. Sahani, "Impact of iterative reconstruction on image quality and radiation dose in multidetector CT of large body size adults," *European radiology*, vol. 22, no. 8, pp. 1631-1640, 2012.
- [46] J. Hsieh, "Adaptive streak artifact reduction in computed tomography resulting from excessive x-ray photon noise," *Medical Physics*, vol. 25, no. 11, pp. 2139-2147, 1998.
- [47] A. Katsevich, "Analysis of an exact inversion algorithm for spiral cone-beam CT," *Physics in Medicine & Biology*, vol. 47, no. 15, p. 2583, 2002.
- [48] K. Lange and J. A. Fessler, "Globally convergent algorithms for maximum a posteriori transmission tomography," *IEEE Transactions on Image Processing*, vol. 4, no. 10, pp. 1430-1438, 1995.
- [49] J. A. Fessler, "Hybrid Poisson/polynomial objective functions for tomographic image reconstruction from transmission scans," *IEEE Transactions on Image Processing*, vol. 4, no. 10, pp. 1439-1450, 1995.
- [50] B. De Man and S. Basu, "Distance-driven projection and backprojection in three dimensions," *Physics in medicine and biology*, vol. 49, no. 11, p. 2463, 2004.

- [51] G. Zeng and G. Gullberg, "Ray-driven backprojector for backprojection filtering and filtered backprojection algorithms," in *Proceedings of the 1993 IEEE Nuclear Science Symposium & Medical Imaging Conference*, 1994: Publ by IEEE.
- [52] W. Zhuang, S. Gopal, and T. Hebert, "Numerical evaluation of methods for computing tomographic projections," *IEEE Transactions on Nuclear Science*, vol. 41, no. 4, pp. 1660-1665, 1994.
- [53] R. L. Siddon, "Fast calculation of the exact radiological path for a three-dimensional CT array," *Medical physics*, vol. 12, no. 2, pp. 252-255, 1985.
- [54] G. T. Herman, *Fundamentals of computerized tomography: image reconstruction from projections*. Springer Science & Business Media, 2009.
- [55] P. M. Joseph, "An improved algorithm for reprojecting rays through pixel images," *IEEE transactions on medical imaging*, vol. 1, no. 3, pp. 192-196, 1982.
- [56] B. De Man and S. Basu, "Distance-driven projection and backprojection," in *Nuclear Science Symposium Conference Record, 2002 IEEE*, 2002, vol. 3, pp. 1477-1480: IEEE.
- [57] S. Basu and B. De Man, "Branchless distance driven projection and backprojection," in *Computational Imaging*, 2006, p. 60650Y.
- [58] D. Schlifske and H. Medeiros, "A fast GPU-based approach to branchless distance-driven projection and back-projection in cone beam CT," in *Proc. SPIE*, 2016, vol. 9783, p. 97832W.
- [59] H. M. Hudson and R. S. Larkin, "Accelerated image reconstruction using ordered subsets of projection data," *IEEE transactions on medical imaging*, vol. 13, no. 4, pp. 601-609, 1994.
- [60] S. Ahn, J. A. Fessler, D. Blatt, and A. O. Hero, "Convergent incremental optimization transfer algorithms: application to tomography," *IEEE Trans Med Imaging*, vol. 25, no. 3, pp. 283-96, Mar 2006.
- [61] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM journal on imaging sciences*, vol. 2, no. 1, pp. 183-202, 2009.
- [62] G. L. Zeng and G. T. Gullberg, "Unmatched projector/backprojector pairs in an iterative reconstruction algorithm," *IEEE transactions on medical imaging*, vol. 19, no. 5, pp. 548-555, 2000.
- [63] J. A. Fessler, E. P. Ficaro, N. H. Clinthorne, and K. Lange, "Grouped-coordinate ascent algorithms for penalized-likelihood transmission image reconstruction," *IEEE transactions on medical imaging*, vol. 16, no. 2, pp. 166-175, 1997.

- [64] H. Erdogan and J. A. Fessler, "Ordered subsets algorithms for transmission tomography," *Physics in medicine and biology*, vol. 44, no. 11, p. 2835, 1999.
- [65] D. Keesing, "Development and implementation of fully 3D statistical image reconstruction algorithms for helical CT and half-ring PET insert system," 2009.
- [66] R. Liu, L. Fu, B. De Man, and H. Yu, "GPU-based branchless distance-driven projection and backprojection," *IEEE transactions on computational imaging*, vol. 3, no. 4, pp. 617-632, 2017.
- [67] A. Katsevich, "An improved exact filtered backprojection algorithm for spiral computed tomography," *Advances in Applied Mathematics*, vol. 32, no. 4, pp. 681-697, 2004.
- [68] F. Noo, J. Pack, and D. Heuscher, "Exact helical reconstruction using native cone-beam geometries," *Physics in Medicine & Biology*, vol. 48, no. 23, p. 3787, 2003.
- [69] Y. Zou and X. Pan, "Exact image reconstruction on PI-lines from minimum data in helical cone-beam CT," *Physics in Medicine & Biology*, vol. 49, no. 6, p. 941, 2004.
- [70] H. Schöndube, K. Stierstorfer, and F. Noo, "Accurate helical cone-beam CT reconstruction with redundant data," *Physics in Medicine & Biology*, vol. 54, no. 15, p. 4625, 2009.
- [71] S. Schaller, K. Stierstorfer, H. Bruder, M. Kachelriess, and T. Flohr, "Novel approximate approach for high-quality image reconstruction in helical cone-beam CT at arbitrary pitch," in *Medical Imaging 2001: Image Processing*, 2001, vol. 4322, pp. 113-128: International Society for Optics and Photonics.
- [72] X. Tang, J. Hsieh, A. Hagiwara, R. A. Nilsen, J.-B. Thibault, and E. Drapkin, "A three-dimensional weighted cone beam filtered backprojection (CB-FBP) algorithm for image reconstruction in volumetric CT under a circular source trajectory," *Physics in Medicine & Biology*, vol. 50, no. 16, p. 3889, 2005.
- [73] K. Taguchi, B.-S. S. Chiang, and M. D. Silver, "A new weighting scheme for cone-beam helical CT to reduce the image noise," *Physics in Medicine & Biology*, vol. 49, no. 11, p. 2351, 2004.
- [74] K. Stierstorfer, A. Rauscher, J. Boese, H. Bruder, S. Schaller, and T. Flohr, "Weighted FBP—a simple approximate 3D FBP algorithm for multislice spiral CT with good dose usage for arbitrary pitch," *Physics in Medicine & Biology*, vol. 49, no. 11, p. 2209, 2004.
- [75] X. Tang, J. Hsieh, R. A. Nilsen, S. Dutta, D. Samsonov, and A. Hagiwara, "A three-dimensional-weighted cone beam filtered backprojection (CB-FBP) algorithm for image reconstruction in volumetric CT—helical scanning," *Physics in Medicine and Biology*, vol. 51, no. 4, p. 855, 2006.

- [76] K. Taguchi, B.-S. S. Chiang, and M. D. Silver, "A new weighting scheme for cone-beam helical CT to reduce the image noise," *Physics in Medicine and Biology*, vol. 49, no. 11, p. 2351, 2004.
- [77] A. Katsevich, "Theoretically exact filtered backprojection-type inversion algorithm for spiral CT," *SIAM Journal on Applied Mathematics*, vol. 62, no. 6, pp. 2012-2026, 2002.
- [78] A. Mitra, D. G. Polite, B. R. Whiting, J. F. Williamson, and J. A. O'Sullivan, "Multi-GPU Acceleration of Branchless Distance Driven Projection and Backprojection for Clinical Helical CT," *Journal of Imaging Science and Technology*, vol. 61, no. 1, pp. 10405-1-10405-13, 2017.
- [79] A. Mitra, S. Degirmenci, D. G. Polite, and J. A. O'Sullivan, "Fast Parallel GPU Implementation for Clinical Helical CT using Branchless DD," in *GPU Technology Conference*, 2016, p. 6234.
- [80] L. Yu, A. N. Primak, X. Liu, and C. H. McCollough, "Image quality optimization and evaluation of linearly mixed images in dual-source, dual-energy CT," *Medical physics*, vol. 36, no. 3, pp. 1019-1024, 2009.
- [81] J. Hsieh, "Computed tomography: principles, design, artifacts, and recent advances," 2009: SPIE Bellingham, WA.
- [82] D. Marin *et al.*, "Low-tube-voltage, high-tube-current multidetector abdominal CT: improved image quality and decreased radiation dose with adaptive statistical iterative reconstruction algorithm—initial clinical experience," *Radiology*, vol. 254, no. 1, pp. 145-153, 2009.
- [83] X. Liu, L. Yu, A. N. Primak, and C. H. McCollough, "Quantitative imaging of element composition and mass fraction using dual-energy CT: Three-material decomposition," *Medical physics*, vol. 36, no. 5, pp. 1602-1609, 2009.
- [84] T. R. Johnson *et al.*, "Material differentiation by dual energy CT: initial experience," *European radiology*, vol. 17, no. 6, pp. 1510-1517, 2007.
- [85] T. Tsunoo, M. Torikoshi, Y. Ohno, K. Uesugi, and N. Yagi, "Measurement of electron density in dual-energy x-ray CT with monochromatic x rays and evaluation of its accuracy," *Medical physics*, vol. 35, no. 11, pp. 4924-4932, 2008.
- [86] M. Bazalova, J.-F. Carrier, L. Beaulieu, and F. Verhaegen, "Dual-energy CT-based material extraction for tissue segmentation in Monte Carlo dose calculations," *Physics in Medicine & Biology*, vol. 53, no. 9, p. 2439, 2008.
- [87] R. E. Alvarez and A. Macovski, "Energy-selective reconstructions in x-ray computerised tomography," *Physics in Medicine & Biology*, vol. 21, no. 5, p. 733, 1976.

- [88] M. Torikoshi *et al.*, "Electron density measurement with dual-energy x-ray CT using synchrotron radiation," *Physics in Medicine & Biology*, vol. 48, no. 5, p. 673, 2003.
- [89] B. Schaffner and E. Pedroni, "The precision of proton range calculations in proton radiotherapy treatment planning: experimental verification of the relation between CT-HU and proton stopping power," *Physics in Medicine & Biology*, vol. 43, no. 6, p. 1579, 1998.
- [90] J. F. Williamson *et al.*, "Prospects for quantitative computed tomography imaging in the presence of foreign metal bodies using statistical image reconstruction," *Medical physics*, vol. 29, no. 10, pp. 2404-2418, 2002.
- [91] D. Han, J. V. Siebers, and J. F. Williamson, "A linear, separable two-parameter model for dual energy CT imaging of proton stopping power computation," *Medical physics*, vol. 43, no. 1, pp. 600-612, 2016.
- [92] B. De Man, J. Nuyts, P. Dupont, G. Marchal, and P. Suetens, "An iterative maximum-likelihood polychromatic algorithm for CT," *IEEE transactions on medical imaging*, vol. 20, no. 10, pp. 999-1008, 2001.
- [93] I. A. Elbakri and J. A. Fessler, "Statistical X-ray-computed tomography image reconstruction with beam-hardening correction," 2001: SPIE.
- [94] P. Sukovic and N. H. Clinthorne, "Penalized weighted least-squares image reconstruction for dual energy X-ray transmission tomography," *IEEE transactions on medical imaging*, vol. 19, no. 11, pp. 1075-1081, 2000.
- [95] C. H. Yan, R. T. Whalen, G. S. Beaupre, S. Y. Yen, and S. Napel, "Reconstruction algorithm for polychromatic CT imaging: application to beam hardening correction," *IEEE Transactions on medical imaging*, vol. 19, no. 1, pp. 1-11, 2000.
- [96] J. M. Boone and A. E. Chavez, "Comparison of x-ray cross sections for diagnostic and therapeutic medical physics," *Medical Physics*, vol. 23, no. 12, pp. 1997-2005, 1996.
- [97] J. Boone *et al.*, "Handbook of Medical Imaging: Volume 1. Physics and Psychophysics," ed: SPIE press, Bellingham, 2000.
- [98] M. M. Rehani *et al.*, "Managing patient dose in computed tomography," *Ann ICRP*, vol. 30, no. 4, pp. 7-45, 2000.
- [99] M. W. Kan, L. H. Leung, W. Wong, and N. Lam, "Radiation dose from cone beam computed tomography for image-guided radiation therapy," *International Journal of Radiation Oncology* Biology* Physics*, vol. 70, no. 1, pp. 272-279, 2008.
- [100] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*, 2015, pp. 448-456.

- [101] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "BM3D image denoising with shape-adaptive principal component analysis," in *SPARS'09-Signal Processing with Adaptive Sparse Structured Representations*, 2009.
- [102] S. Gu, L. Zhang, W. Zuo, and X. Feng, "Weighted nuclear norm minimization with application to image denoising," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 2862-2869.
- [103] A. G. e. Ivakhnenko and V. G. Lapa, "Cybernetic predicting devices," PURDUE UNIV LAFAYETTE IND SCHOOL OF ELECTRICAL ENGINEERING1966.
- [104] H. Greenspan, B. van Ginneken, and R. M. Summers, "Guest editorial deep learning in medical imaging: Overview and future promise of an exciting new technique," *IEEE Transactions on Medical Imaging*, vol. 35, no. 5, pp. 1153-1159, 2016.
- [105] E. Kang, J. Min, and J. C. Ye, "A deep convolutional neural network using directional wavelets for low-dose X-ray CT reconstruction," *Medical physics*, vol. 44, no. 10, 2017.
- [106] H. Chen *et al.*, "Low-dose CT via convolutional neural network," *Biomedical optics express*, vol. 8, no. 2, pp. 679-694, 2017.
- [107] H. Chen *et al.*, "Low-dose CT with a residual encoder-decoder convolutional neural network," *IEEE transactions on medical imaging*, vol. 36, no. 12, pp. 2524-2535, 2017.
- [108] G. Wang, "A perspective on deep imaging," *IEEE Access*, vol. 4, pp. 8914-8924, 2016.
- [109] X. Yang *et al.*, "Low-dose x-ray tomography through a deep convolutional neural network," *Scientific reports*, vol. 8, no. 1, p. 2575, 2018.
- [110] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097-1105.
- [111] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang, "Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising," *IEEE Transactions on Image Processing*, vol. 26, no. 7, pp. 3142-3155, 2017.
- [112] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770-778.
- [113] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.
- [114] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

- [115] J. Wang, T. Li, H. Lu, and Z. Liang, "Penalized weighted least-squares approach to sinogram noise reduction and image reconstruction for low-dose X-ray computed tomography," *IEEE transactions on medical imaging*, vol. 25, no. 10, pp. 1272-1283, 2006.
- [116] J. Hsieh, O. Gurmen, and K. F. King, "Investigation of a solid-state detector for advanced computed tomography," *IEEE transactions on medical imaging*, vol. 19, no. 9, pp. 930-940, 2000.
- [117] A. Britten, M. Crotty, H. Kiremidjian, A. Grundy, and E. Adam, "The addition of computer simulated noise to investigate radiation dose and image quality in images with spatial correlation of statistical noise: an example application to X-ray CT of the brain," *The British journal of radiology*, vol. 77, no. 916, pp. 323-328, 2004.
- [118] P. Massoumzadeh, S. Don, C. F. Hildebolt, K. T. Bae, and B. R. Whiting, "Validation of CT dose-reduction simulation," *Medical physics*, vol. 36, no. 1, pp. 174-189, 2009.
- [119] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1026-1034.
- [120] A. Vedaldi and K. Lenc, "Matconvnet: Convolutional neural networks for matlab," in *Proceedings of the 23rd ACM international conference on Multimedia*, 2015, pp. 689-692: ACM.

Curriculum Vitae

Ayan Mitra

Degrees

Ph.D., Electrical Engineering, Washington University in St. Louis, St. Louis, MO, U.S.A., 2018.

B.E., Electrical Engineering, Jadavpur University, Kolkata, India, 2013.

Selected Publications

A. Mitra, D. G. Politte, J. A. O’Sullivan, “Accelerating Iterative Image Reconstruction via Adaptive Surrogate Functions,” Proc. of Electronic Imaging 2018 (in print).

S. Don, D. G. Politte, R. Holdener, **A. Mitra**, C. Abbey, B.R. Whiting, “Stochastic Noise Tolerance in Pediatric Appendicitis CT,” SPR Annual Meeting 2018 (accepted).

A. Mitra, D. G. Politte, B. R. Whiting, J. F. Williamson, J. A. O’Sullivan, “MultiGPU Acceleration of Branchless Distance Driven Projection and Backprojection for Clinical Helical CT,” Journal of Imaging Science and Technology, 61(1), page 1-13, 2017.

A. Mitra, D.G. Politte, J.A. O’Sullivan, “MultiGPU Acceleration of Iterative X-Ray CT Image Reconstruction,” GPU technology Conference 2017.

A. Mitra, S. Degirmenci, D.G. Politte, J.A. O’Sullivan, “Fast Parallel GPU Implementation for Clinical Helical CT using Branchless DD,” GPU technology Conference 2016.

A. Mitra, A. Roy, “A high regulated low ripple DC power supply based on LC filter and IGBT,” International Journal of Power Electronics and Drive System (IJPEDS), Vol. 3, No. 1, March 2013, pp. 30-40, ISSN: 2088-8694.

**Publications in
Preparation**

A. Mitra, D. G. Politte, B. R. Whiting, J. F. Williamson, J. A. O’Sullivan, “Accelerating Dual Energy Iterative Image Reconstruction via Adaptive Surrogate Functions.”

August 2018