THE SOFTWARE AND HARDWARE ACCELERATION FOR INTERIOR TOMOGRAPHY

BY

RUI LIU

A Dissertation Submitted to the Graduate Faculty of

WAKE FOREST UNIVERSITY GRADUATE SCHOOL OF ARTS AND SCIENCES

In Partial Fulfillment of the Requirements

For the Degree of

DOCTOR OF PHILOSOPHY

Biomedical Engineering

December 2016

Winston-Salem, North Carolina

Approved By:

Hengyong Yu, Ph.D., Advisor

Robert J. Plemmons, Ph.D., Chair

Craig A. Hamilton, Ph.D.

Ge Wang, Ph.D.

Guohua Cao, Ph.D.
# TABLE OF CONTENTS

List of Figures .................................................................................................................. 5

List of Tables .................................................................................................................... 13

List of Abbreviations ........................................................................................................ 15

Abstract ............................................................................................................................ 17

Introduction ....................................................................................................................... 19

CHAPTER I. GPU-Based Acceleration for Interior Tomography ..................................... 22

Abstract: ............................................................................................................................. 22

1.1 Introduction ................................................................................................................. 22

1.2 Algorithm Design ....................................................................................................... 25

1.2.1 Imaging Model ....................................................................................................... 25

1.2.2 SART and OS-SART .......................................................................................... 26

1.2.3 CS-based Image Reconstruction ........................................................................... 27

1.2.4 DGT-based sparsity and STF .............................................................................. 27

1.3 GPU Acceleration ..................................................................................................... 29

1.3.1 Parallelization Strategy ....................................................................................... 29

1.3.2 Projection and Backprojection Models ............................................................... 31

1.3.3 Parallelization of TV Minimization ...................................................................... 34

1.3.4 Overall Pseudo-codes ......................................................................................... 36

1.4 Numerical Experiments ............................................................................................. 36
CHAPTER II. GPU-Based Branchless Distance-Driven Projection and Backprojection ........51

Abstract: .............................................................................................................51

2.1 Introduction..................................................................................................52

2.2 Related Works and Background ..................................................................53

2.2.1 Different P/BP Models........................................................................53

2.2.2 GPU Acceleration for CT Reconstruction ............................................55

2.3 Methods .....................................................................................................57

2.3.1 Cone-Beam Geometry ...........................................................................57

2.3.2 Distance Driven Model .........................................................................60

2.3.3 Branchless Model ..................................................................................62

2.3.4 CUDA Implementation of Branchless Model ........................................65

2.3.5 Precision of Hardware Interpolation and Calculation of Integral Images ..............................................................................................................68

2.3.6 Branchless DD Projection based on double-precision interpolation (GPU-DB) ........68

2.3.7 Z-line Backprojection(GPU-BZ) ............................................................69

2.4 Results.......................................................................................................70

2.4.1 Experimental Configuration....................................................................70

2.4.2 Forward and Back-projector as Single Modules ....................................70
3.4.5 SVD Methods for Multiple-resolution Images .................................................. 112

3.5 Discussion and Conclusion ................................................................................. 114

Chapter IV: Evaluation of GPU-Based CT Reconstruction for Obese Patient Problem .... 117

Abstract ..................................................................................................................... 117

4.1. Introduction ......................................................................................................... 118

4.2. Method .............................................................................................................. 121

  4.2.1. Analytical reconstruction algorithm ............................................................... 121

  4.2.2. Iterative reconstruction algorithm ................................................................. 122

  4.2.3. Experiment setup .......................................................................................... 125

4.3. Results .............................................................................................................. 127

  4.3.1. Analytical reconstruction results ................................................................. 127

  4.3.2. Iterative reconstruction results ................................................................. 131

  4.4. Discussion and conclusion ............................................................................. 137

References ............................................................................................................... 139

Curriculum Vitae ..................................................................................................... 149

Educational background ......................................................................................... 149

Academic awards .................................................................................................... 149

Memberships ........................................................................................................... 149

Publications ............................................................................................................ 149

Experiences ............................................................................................................. 151
**List of Figures**

Figure 1. Boundary box based backprojection process. (a) The backprojection is accurate with higher computational cost when the detector size is smaller. (b) The backprojection degenerates to the pixel/voxel driven method with more computational cost when the detector size is large. 31

Figure 2. The thread configurations in fan-beam reconstruction. (a) represents one thread block configuration for back-projection and DGT. (b) represents one thread block configuration for TV descent direction calculation and pseudo-inverse of DGT. ................................................................. 35

Figure 3. Bar chart of the computational cost for different GPU-based reconstruction methods in fan-beam geometry. The left column is for image size 1024×1024, and the right column is for image size 2048×2048 from 50 iterations. The bottom row is the speedup comparison in CPU and GPU. The abscissa indicates different projection views used in the reconstruction. The mandatory axis is the reconstruction time (in seconds). The secondary axis is the speedup factor. .................... 37

Figure 4. A real medical image volume of a patient reconstructed with OS-SART algorithm. (a), (b) and (c) are the transverse, sagittal c) are the transverse, sagittal and coronal planes, respectively. The display window is [0, 1400]. Figure (d)-(f) are the reconstruction differences compared with the FDK method, which is almost unobservable. The bottom line (g) are the pseudo-color volume rendering with of the reconstructed result in different observation positions. .......................................................................................................................................................... 39

Figure 5. Interior volume reconstruction result from the truncated scanning data of a patient with OS-SART algorithm plus TV regularization. The image resolution is 5122 × 64. The image size on transverse plane is 2302 mm2. (a), (b) and (c) are the transverse, sagittal and coronal planes, respectively. The display window is [0, 1400]. (d) is the pseudo-color volume rendering observing from two positions...................................................................................................................................................... 40
Figure 6. Representative results reconstructed from truncated local projections for a modified Shepp-Logan phantom. From the left to right columns; the images were reconstructed from 17, 21, 72, 180 and 360 projections, respectively. The iterations are 30. From the top to bottom rows, the images were reconstructed by the OS-SART, OS-SART with the steepest descent and OS-SART with soft-threshold filtering for TV minimization, respectively. The display window is [0, 1]....

Figure 7. Reconstructed results of a cardiac region from clinical projections. (a) is the reference image, (b) is reconstructed by the SD-based TV minimization, and (c) is reconstructed by the STF-based TV minimization. From (a) to (c), the display window is [0, 1132]. (d) and (e) are image differences between (a), (b) and (a), (c) respectively. The display window is [-571, 580]. The STF-based TV minimization reconstruction result outperforms the image with SD-based TV minimization reconstruction.

Figure 8. Representative results of the OS-SART with SD-based TV minimization (the top row images) and the OS-SART with STF-based TV minimization (the bottom row images) in the transverse (left column), sagittal (middle column) and coronal (right column) views. The display window is [0,1]. The subfigures (a-1) and (d-1) are the magnified parts of (a) and (d). (a-2) and (d-2) are the error images of (a-1) and (d-1) in reference to the original phantom. It can be seen that the STF-based OS-SART can keep more fine details than the SD-based OS-SART.

Figure 9. The SSIM indices of all the slices in transverse, sagittal and coronal observation directions. The SSIM values in the transverse plane (a) always larger than 0.925 indicating promising reconstruction results. Even though the SSIM values in the sagittal plane (b) are a little small, on average, the results are satisfying. Because at the first and last several slices in sagittal and coronal directions, the images are entirely blank, they cause the SSIM values perfectly being 1.
Figure 10. Interior reconstruction results from 50 projections after different iterations. The image size is 256 × 256. (o) is the reference image which was reconstructed by the FBP method from 2200 global projections. (a) to (g) are the reconstructed images from 50 projections after 10, 20, 100, 200, 500, 1000 and 5000 iterations, respectively.

Figure 11. Reconstruction error curves with respect to iterative number. The errors are computed in reference to red circle region in the image (o) in Figure.10. Two subfigure inside the main figure is the local amplification of the main curve. We can observe that when the iterations index is large, the STF-based reconstruction converges slower and slower, sometimes, the RMSE will increase with more iterations.

Figure 12. Circular cone-beam geometry with a curve detector.

Figure 13. Illustration of DD models. (a) DD interpolation model; (b) 2D DD interpolation and (c) 3D brute force DD interpolation.

Figure 14. Illustration for computing the integral over a rectangle region using the integral image. (a) is the image $f_{xi,yj}$ with 4 vertexe. (b) is the corresponding integral image $F(xi,yj)$.

Figure 15. Illustration of branchless DD projection. (a) is branchless DD projection procedure and (b) is one slice of the integral image set as in (a). The rectangle area is determined by four red intersection points.

Figure 16. Illustration of branchless DD backprojection. (a) the selections of different center planes according to current projection view angle, (b) the corresponding projection plane 1 and plane 2 with respect to different center planes. The texels are fetched at the black dots on the integral images.

Figure 17. Software interpolation based branchless DD backprojection. (a) is the original hardware-based interpolation. (b) is the texel positions which are calculated by rounding (blue dots) and fetched, and the texel at red dots are calculated by bilinear interpolation.
Figure 18. Illustration of Z-line based branchless DD backprojection.

Figure 19. Speedup performance with respect to view number. (a) and (b) are the projection and backprojection computational costs with respect to different view numbers. (c) and (d) are the corresponding speedups compared to CPU-32.

Figure 20. Speedup performance with respect to image size along the transverse plane. (a) and (b) are the projection and backprojection computational costs with respect to different image size along in-plane direction. (c) and (d) are the corresponding speedups compared to CPU-32.

Figure 21. Speedup performance with respect to image and detector sizes along Z-direction. (a) and (b) are the projection and backprojection computational costs with respect to image/detector size along the cross-plane direction. (c) and (d) are the corresponding speedups compared to CPU-32.

Figure 22. Numerical simulation results of the modified Shepp-Logan phantom. (a) are the transverse, sagittal and coronal planes reconstructed by the CPU-32 in a display window [0, 1]. (b) are the counterpart of (a) reconstructed by the GPU-BL. (c) and (d) are the corresponding error of (a) and (b) with respect to ground truth in display window [-0.02 0.02]. (e) and (f) are the horizontal and vertical profiles of the center slice of the images.

Figure 23. Numerical simulation results of the FORBID head phantom. (a) is the ground truth. (b) and (c) are the reconstruction results from GPU-BL and CPU-32, respectively. The display window for on the first row images is [1.0,1.2]. (d) is the differences between (b) and (c) in a display window [-0.005,0.005]. (e) and (f) are the reconstruction errors of (b) and (c) with respect to (a) in a display window [-0.02,0.02].

Figure 24. Real phantom reconstruction results. (a) and (b) are reconstructed by the CPU-32 and GPU-BL from high dose (835mAs) projections in a display window [800 1200]HU. (c) is the
difference image between CPU-32 and GPU-BL for low dose case in a display window [-3, 5]HU.

Figure 25. Reconstructed results from a clinical data set. (a) is reconstructed by the FDK algorithm as ground truth. (b) and (c) are reconstructed by the GPU-BL CPU-32, respectively. (d) is the differences between GPU-BL and CPU-32 in a display widow[-0.5, 0.5]HU. (e) and (f) are the differences between the ground truth and GPU-BL and CPU-32 in display window [-20, 20]HU, respectively.

Figure 26. Illustration of physical meaning for SVD assuming an interior scan geometry. The top row is the (1st, 10th, 100th, 200th, 1000th, and 10000th) column vector images of matrix $U$, and the bottom row are corresponding column vector images of matrix $V$.

Figure 27. A schematic diagram for multi-resolution representation of an image and its corresponding system matrix. In (a), the gray part surrounded by the dashed line represents the ROI. The large grids represent coarse pixels while small grids represent fine pixels. (b) is the spy of the combined system matrix $A_c$ where the blue dots represents non-zero elements, the columns of $A_L$ corresponds to fine pixels and the columns of $A_R$ corresponds to coarse pixels.

Figure 28. Numerical phantoms. (a) is a modified high-contrast Shepp-Logan phantom and (b) is a low-contrast FORBILD head phantom. The display windows for these two phantoms are [0, 0.5] and [0.8, 1.5], respectively.

Figure 29. Shepp-Logan phantom reconstruction results from 128 views with different methods. The 1st and 3rd columns are the reconstruction results from noise-free projections, and the 2nd and 4th columns are from the projections assuming $5 \times 10^4$ photons per detector cell in an air scan. The display window is [0.15, 0.25].
Figure 30. Reconstructed Shepp-Logan phantom images from the TSVD method by keeping 10%, 20%, 30%, 60%, 80%, and 90% of singular value magnitudes, respectively. The display window is [0.15, 0.25].

Figure 31. Reconstructed Shepp-Logan phantom images from GSVD method with different values of $\xi$. The display window is [0.15, 0.25].

Figure 32. Interior reconstruction of the Shepp-Logan phantom with 128 views from noise-free/noisy datasets. 1st and 3rd columns are from noise-free projections, and 2nd and 4th columns are from projections assuming $5 \times 104$ photons per detector cell in an air scan. The display window is [0.15, 0.25].

Figure 33. Interior reconstruction of the FORBILD head phantom with noise-free projections (1st and 3rd columns) and noisy projections assuming $5 \times 104$ photons per detector cell (2nd and 4th columns). The display window is [0.8, 1.5].

Figure 34. Real phantom reconstruction results from 246 views (case 1) and 123 views (case 2). The display windows is [-1000, -500]HU.

Figure 35. ROI reconstruction results with a known sub-region from noise-free projections. (a) is the reconstructed Shepp-Logan phantom from SVD in a display window $[0,0.5]$. (b) is the reconstruction error of (a) with respect to the ground truth in a display window $-1.81 \times 10-11,2.13 \times 10-11$. (c) is the counterparts of (a) reconstructed by the SART after 300 iterations in a display window $[0,0.5]$.

Figure 36. Comparison of several algorithms for interior tomography with a known sub-region inside the ROI. From left to right columns, the images are reconstructed from different noise levels. From top to bottom rows, the reconstruction algorithms are TSVD, VSVD, SART with 300 iterations, and GD-Tik with 300 iterations. The display window is [0, 0.5].
Figure 37. Five quantitative criteria comparisons of Figure 36 under six noise levels. The x-axis represents six different noise levels.

Figure 38. The interior reconstruction of a modified Shepp-Logan phantom (255×255) using the multi-resolution pixels and VSVD. The display window for (a) is [0, 0.5]. The dashed circle in (a) indicates the ROI. The ground truth and the reconstructed profiles along the horizontal and vertical lines in (a) are plotted in (b) and (c).

Figure 39. ROI reconstruction of the modified Shepp-Logan phantom (255×255) using a multiple resolution strategy. From left to right columns, six different projection noise levels are applied. The first and second rows are the results from TSVD and VSVD. The third and fourth rows are counterparts of the first and second rows but assuming a known subregion (indicated in Figure 28 (a)). The display window is [0, 0.5].

Figure 40. The transverse, coronal and sagittal planes of the ground truth (left) and GPU-based CB-FBP reconstruction from simulated projections with data truncation (right) in a display window [-500, 500]HU.

Figure 41. GPU-CB-FBP results of 3 clinical datasets for an obese patient. From left to right, the columns are for transverse, coronal and sagittal planes, respectively. Rows 1-3 are the results from three different datasets in a display window [-500, 800]HU. The fourth row is the differences between images reconstructed by the CPU and GPU implementations for the first dataset in a display window [-0.05, 0.05]HU.

Figure 42. The timeline of one inner iteration with one subset in 3 GPUs. The length of the bars indicates the duration of time.

Figure 43. The results reconstructed by TV regularized OS-SART from simulated projections. (a), (b), (c) and (d) are with 5, 15, 25 and 35 iterations, respectively, in a display window [-500, 500]HU. From top to bottom, the rows are for transverse, coronal and sagittal planes,
respectively. The horizontal profiles of the transverse plane indicated along the dashed line at the center of the transverse planes are shown in (e).

Figure 44. The transverse, sagittal and coronal planes reconstructed from the first clinical dataset by the TV-OS-SART in GPU with different iteration numbers and step sizes. The display window is [-500, 800]HU.
List of Tables

Table I. Cone-beam reconstruction geometry configuration for numerical simulation and clinical dataset. ...............................................................................................................................................................38

Table II. The computational cost for STF-based TV minimization in an OS-SART framework with a modified Shepp-Logan phantom. A comparison is presented between CPU (Intel Xeon 3.1GHz, Single Core usage) and GPU (NVIDIA, Tesla K10, single GPU usage) after 50 iterations. .................................................................................................................................................................41

Table III. The computational cost for the projection, backprojection and TV minimization steps in OS-SART, OS-SART with SD-based TV minimization, and OS-SART with STF-based TV minimization .................................................................................................................................................................42

Table IV. The Configuration of GE CT750 HD Geometry .................................................................................................................................72

Table V. Running time of projection and backprojection (unit: s) in GE CT750 HD geometry...74

Table VI. GUPS of Projection and Backprojection. ........................................................................................................................................74

Table VII. Computational costs (in seconds) of various components on multiple GPUs. The image volume is $512 \times 512 \times 64$ with (a) 984, (b) 1968 and (c) 3936 views of size $888 \times 64$ in HD geometry .................................................................................................................................................................................................77

Table VIII. Speedup performance of the FORBILD head phantom reconstruction in one iteration. .................................................................80

Table IX. Quantitative evaluation of the Shepp-Logan phantom reconstruction results with different methods under 4 noise levels in a 128-views standard scan configuration. .........................99

Table X. Quantitative evaluation of the Shepp-Logan phantom reconstruction results with different methods under 4 noise levels in a 128-views interior scan configuration........................................103
Table XI. Quantitative evaluation of the FORBILD head phantom interior reconstruction results with different methods under 4 noise levels in 271-views.......................... 107

Table XII. Quantitative evaluation results of the physical phantom reconstruction with benchmark methods........................................................................................................... 113

Table XIII. Comparison of the TSVD and VSVD results in six different noise levels for the third and the fourth row images in Figure 39. ........................................................................................................................................ 113

Table XIV. The geometrical configuration of a helical scan for numerical simulation of a GE CT scanner. ........................................................................................................................................... 126

Table XV. Reconstruction time for a 512×512×823 volumetric image from simulated projections (888×64×20112) ........................................................................................................................................... 127

Table XVI. Performance comparison of analytical reconstruction for three clinical datasets in CPU/GPU implementation........................................................................................................................................... 131

Table XVII. The total and kernel projection/backprojection times in triple GPUs with 3 different clinical datasets. ........................................................................................................................................... 137
## List of Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2D</td>
<td>Two Dimensional</td>
</tr>
<tr>
<td>3D</td>
<td>Three Dimensional</td>
</tr>
<tr>
<td>AIM</td>
<td>Area Integral Model</td>
</tr>
<tr>
<td>ALARA</td>
<td>As Low As Reasonably Achievable</td>
</tr>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>CB-FBP</td>
<td>Cone Beam Filtered Backprojection</td>
</tr>
<tr>
<td>CG</td>
<td>Conjugate Gradient / Computer Graphics</td>
</tr>
<tr>
<td>COO</td>
<td>Coordinate List</td>
</tr>
<tr>
<td>CS</td>
<td>Compressive Sensing</td>
</tr>
<tr>
<td>CSR</td>
<td>Compressed Sparse Row</td>
</tr>
<tr>
<td>CT</td>
<td>Computed Tomography</td>
</tr>
<tr>
<td>CUDA</td>
<td>Compute Unified Device Architecture</td>
</tr>
<tr>
<td>DC</td>
<td>Direct Current</td>
</tr>
<tr>
<td>DD</td>
<td>Distance Driven</td>
</tr>
<tr>
<td>DGT</td>
<td>Discrete Gradient Transform</td>
</tr>
<tr>
<td>FDK</td>
<td>Feldkamp-Davis-Kress Algorithm</td>
</tr>
<tr>
<td>FFT</td>
<td>Fast Fourier Transform</td>
</tr>
<tr>
<td>FISTA</td>
<td>Fast Iterative Shrinkage-Thresholding Algorithm</td>
</tr>
<tr>
<td>FOV</td>
<td>Field of View</td>
</tr>
<tr>
<td>FPGA</td>
<td>Field-Programmable Gate Array</td>
</tr>
<tr>
<td>GD-Tik</td>
<td>Gradient Descent solving Tikhonov regularized SVD</td>
</tr>
<tr>
<td>GGMRF</td>
<td>Generalized Gaussian Markov Random Field</td>
</tr>
<tr>
<td>GPU</td>
<td>Graphics Processing Unit</td>
</tr>
<tr>
<td>GPU-BL</td>
<td>GPU-based Branchless DD model</td>
</tr>
<tr>
<td>GPU-BZ</td>
<td>GPU-based Z-line DD backprojection</td>
</tr>
<tr>
<td>GPU-DB</td>
<td>GPU-based Double Precision based DD projection</td>
</tr>
<tr>
<td>GSVD</td>
<td>Generalized SVD</td>
</tr>
<tr>
<td>GUPS</td>
<td>Giga-Updates Per Second</td>
</tr>
<tr>
<td>HLSL</td>
<td>High-Level Shader Language</td>
</tr>
<tr>
<td>ICD</td>
<td>Iterative Coordinate Descent</td>
</tr>
<tr>
<td>IR</td>
<td>Iterative Reconstruction</td>
</tr>
<tr>
<td>IRAs</td>
<td>Iterative Reconstruction Algorithms</td>
</tr>
<tr>
<td>LAPACK</td>
<td>Linear Algebra Package</td>
</tr>
<tr>
<td>OMP</td>
<td>Orthogonal Matching Pursuit</td>
</tr>
<tr>
<td>OpenCL</td>
<td>Open Computing Language</td>
</tr>
<tr>
<td>OpenGL</td>
<td>Open Graphical Language</td>
</tr>
<tr>
<td>OS</td>
<td>Ordered Subset</td>
</tr>
<tr>
<td>OS-SART</td>
<td>Ordered Subset-Simultaneous Algebra Reconstruction Technique</td>
</tr>
<tr>
<td>PC</td>
<td>Personal Computer</td>
</tr>
<tr>
<td>PICCS</td>
<td>Prior Image Constrained Compressed Sensing</td>
</tr>
<tr>
<td>P/BP</td>
<td>Projection and Backprojection</td>
</tr>
<tr>
<td>RMSE</td>
<td>Root Mean Square Error</td>
</tr>
<tr>
<td>ROI</td>
<td>Region of Interest</td>
</tr>
<tr>
<td>Acronym</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
</tr>
<tr>
<td>SART</td>
<td>Simultaneous Algebra Reconstruction Technique</td>
</tr>
<tr>
<td>SD</td>
<td>Steepest Descent</td>
</tr>
<tr>
<td>SDK</td>
<td>Software Development Kit</td>
</tr>
<tr>
<td>SF</td>
<td>Separable Footprint</td>
</tr>
<tr>
<td>SIMD</td>
<td>Single Instruction Multiple Data</td>
</tr>
<tr>
<td>SIR</td>
<td>Statistical Iterative Reconstruction</td>
</tr>
<tr>
<td>SM</td>
<td>Streaming Multiprocessor</td>
</tr>
<tr>
<td>SPECT</td>
<td>Single Photon Emission Computed Tomography</td>
</tr>
<tr>
<td>SSIM</td>
<td>Structural Similarity</td>
</tr>
<tr>
<td>STD</td>
<td>Standard Deviation</td>
</tr>
<tr>
<td>STF</td>
<td>Soft-Threshold Filtering</td>
</tr>
<tr>
<td>SVD</td>
<td>Singular Value Decomposition</td>
</tr>
<tr>
<td>TSVD</td>
<td>Truncated SVD</td>
</tr>
<tr>
<td>TV</td>
<td>Total Variation</td>
</tr>
<tr>
<td>VIM</td>
<td>Volume Integral Model</td>
</tr>
<tr>
<td>VSVD</td>
<td>Tikhonov regularized SVD</td>
</tr>
</tbody>
</table>
Abstract

The x-ray CT is with high spatial resolution, temporal resolutions, and high-speed imaging characteristic. The iterative reconstruction algorithms are becoming more important since they can reconstruct better image qualities from incomplete projection data compared to analytical reconstruction algorithms. Iterative reconstruction algorithms also allow more flexible trajectories. However, the high computational cost prevents them from wide applications in clinics. Therefore, it is necessary to develop software and hardware methods to accelerate the iterative reconstruction algorithms. The main methods to accelerate iterative reconstruction algorithms can be either expedite the convergence rate of the algorithm mathematically or directly accelerating the most time-consuming projection / backprojection procedures.

This dissertation focuses on software and hardware acceleration techniques for interior tomography, which is a typical incomplete projection problem with a unique and stable solution under the constraint of certain prior information in the framework of compressive sensing. First, we accelerated and evaluated the ordered subset simultaneous algebraic reconstruction technique for interior tomography by implementing the projection / backprojection procedures with novel algorithms and GPU implementations. The total variation of the region-of-interest (ROI) was minimized for an exact theoretical solution. The performance is comparable to the results reported by peers in the CT field. Then, we proposed a branchless distance driven model to accelerate the projection and backprojection procedures in GPUs. This method is general, and it can be applied to any iterative algorithm in all the tomographic imaging modalities, such as CT, PET, SPECT, etc. The acceleration rate can be 12 folds compared with the state-of-the-art thirty-two-threads CPU implementation. After that, we proposed singular value decomposition based reconstruction algorithms to solve a broad category of objective functions for iterative reconstruction in a non-iterative fashion. Compared to conventional iterative reconstruction algorithm, our method is much more flexible and suitable for ROI reconstruction and interior
Finally, the analytical and iterative reconstruction algorithms were implemented in GPUs for data truncation problem of obese patients. While the approximate analytic reconstruction algorithm outperforms the iterative reconstruction algorithm in terms of computational cost, the iterative algorithm outperforms the analytic algorithm in terms image quality especially when the projection data is suffered from respiration motion. With GPUs acceleration, the reconstruction time of analytical and iterative reconstruction algorithms can be significantly reduced.
Introduction

X-ray computed tomography (CT) is one of the main modern imaging modalities. It is widely applied in clinics[1], pharmaceutical [2], [3] and non-destructive industrial inspection areas [4]. The importance of x-ray CT will not be overestimated for its high temporal and spatial resolutions compared to other imaging modalities. However, x-ray CT radiation accounts for large portion of the ionizing radiation to the population that cannot be ignored [5]. The widely accepted As Low As Reasonably Achievable (ALARA) principle urges the medical community to reduce the unnecessary radiation hazard [4][7].

Many methods have been proposed to reduce the excessive radiation dose. They can be divided into three categories. (1) Reducing the x-ray flux towards the detector: it is usually achieved by controlling the operating current, potential or exposure time. However, it leads to high projection noise and results in degraded image quality. (2) Decreasing the number of x-ray paths across the whole object support, which can be achieved by the short-scan or sparse-view scan. However, this method leads to limited-angle, few-view problems [8]. (3) Irradiate a small region-of-interest (ROI) inside an object named interior scan. In this case, an ROI is small and strictly locates inside the object support. Because only the ROI is illuminated, the radiation dose will be reduced [9]. However, the induced interior problem has been proved being with no unique solution in a general unconstrained setting.

The recently developed compressive sensing (CS) theory shows that a real signal can be accurately recovered with an overwhelming probability from the amount of data or measurements far less than the Shannon-Nyquist sampling theorem claimed [10], [11]. Inspired by the CS theory, it was proved that the interior problem could be uniquely and stably solved if there is a known subregion or the ROI is piecewise constant/polynomial. Most of the CS-inspired CT reconstruction algorithms are iterative, and iterative reconstruction algorithms (IRAs) are becoming more popular because higher image qualities can be achieved from incomplete
projections. IRAs are flexible to combine with different physical models, statistical models, detector response model, and prior knowledge. The conventional simultaneous algebraic reconstruction technique (SART) framework can be applied in CS-based image reconstruction with TV regularization [19]–[21]. Many other TV optimization methods or applications were also reported, such as PICCS (prior image constrained compressed sensing) algorithm[22], CS-based interior tomography in SIR (statistical iterative reconstruction) [23], and improved TV method in ASD-POCS framework [24]. Despite various advantages of IRAs, the main flaw of IRAs is the reconstruction speed.

To facilitate the clinical applications of IRAs, researchers began to accelerate them by using parallel computing. For example, the Cell processors are applied in general purpose parallel computing [25]. Intel released the Intel Many Integrated Core architecture products [26]. The parallel algorithm development depends on the parallel computing environments (e.g., PC cluster[27], cell engine [28], [29] and FPGA [30], [31]), therefore different implementations will be achieved according to the computing architectures. Also, maintaining and updating the specific designed parallel computing architecture is of high cost and complicate.

This dissertation focuses on the acceleration of interior reconstruction algorithms using both software and hardware techniques. It includes a series of 4 journal papers. Each paper is numbered as an independent chapter. Chapter 1 implements and evaluates the TV-regularized OS-SART algorithm for interior tomography on GPU. This work was published on IEEE Access (Vol. 2, pp. 757–770, 2014). Chapter 2 develops GPU-based branchless distance driven projection and backprojection. This work was submitted to IEEE Transaction on Computational Imaging, and it is pending revision. Chapter 3 proposes singular value decomposition based 2D image reconstruction algorithms for flexible scanning trajectories and incomplete projections. This work was accepted by Journal of X-ray Science and Technology it will appear soon. Chapter 4 evaluates the GPU-based CT reconstruction for morbidly obese patients imaging. It is a typical
application of interior tomography. This work was submitted to the JSM Biomedical Imaging data Papers.
CHAPTER I. GPU-Based Acceleration for Interior Tomography

Abstract:

The compressive sensing (CS) theory shows that real signals can be exactly recovered from very few samplings. Inspired by the CS theory, the interior problem in computed tomography is proved uniquely solvable by minimizing the total variation inside the region of interest if the imaging object is piecewise constant or polynomial. This is called CS-based interior tomography. However, the CS-based algorithms require high computational cost due to their iterative nature. In this paper, a graphics processing unit (GPU)-based parallel computing technique is applied to accelerate the CS-based interior reconstruction for practical application in both fan-beam and cone-beam geometries. Our results show that the CS-based interior tomography is able to reconstruct excellent volumetric images with GPU acceleration in a few minutes.

1.1 Introduction

The x-ray computed tomography (CT) has been an indispensible imaging modality relying on multiple x-ray projections of the subject to reconstruct a two-dimensional (2D) or three-dimensional (3D) distribution of the attenuation coefficients within the subject [45]. Although it is proud of high spatial and temporal resolution [46], CT radiation accounts for a large portion of the ionizing radiation to the population that cannot be underestimated. The patients undergo a CT scan were estimated to be 60 million in 2002 in the United States, which occupied nearly 75% of the radiation exposure and almost 15% of the imaging procedures [47]. The widely accepted As Low As Reasonably Achievable (ALARA) principle urges the medical community to reduce the unnecessary radiation hazard as much as possible [48], [49]. Reducing the x-ray flux towards the detector and decreasing the amount of x-ray paths across the whole object supporting are two common strategies to avoid extra radiation. The former is usually achieved by controlling the operating current, voltage or exposure time but leading to high projection noise and the latter may
produce few-view, limited-angle or truncated projection problems [50]. Conventional iterative reconstruction algorithms are immune to noisy projections in some extent. However, for the highly ill-posed reconstruction problem, additional regularization is necessary for unique and stable solutions. Interestingly, the compressive sensing (CS) theory shows that a real signal can be accurately recovered with an overwhelming probability from the amount of data or measurements far less than the Shannon-Nyquist sampling theorem claimed [51], [52]. The \( \ell_0 \) norm minimization is the core paradigm of CS based signal recovering. Because of the NP-hard characteristic of the \( \ell_0 \) norm minimization, it is usually relaxed to the \( \ell_1 \) norm optimization with solid theoretical supports [53], [54]. Many algorithms have been proposed to solve the \( \ell_1 \) norm optimization problem, such as interior point method [55], gradient projection method [56] and some dedicated algorithms for the CS-based optimization. In the medical imaging field, the total variation (TV, \( \ell_1 \) norm of discrete gradient transform (DGT) of an image) has been widely adopted as a regularization item. The conventional simultaneous algebraic reconstruction technique (SART) framework can be applied for CS-based image reconstruction by adding the TV regularization item [57]–[59]. The TV minimization is achievable either by using the steepest decent (SD) method or by using the soft-threshold filtering (STF) method. Many other TV minimization methods or applications were also reported, such as PICCS (prior image constrained compressed sensing) algorithm [60], CS based interior tomography in SIR (statistical iterative reconstruction) [61], and improved TV method in an ASD-POCS framework [62].

Narrowing down the x-ray beam to focus on a region-of-interest (ROI) is a representative method for dose reduction nominated as interior scan and the corresponding reconstruction is the interior problem. Because the interior problem is generally non-uniquely solvable, the interior scanning is unable to be applied for quantitative analysis based applications in clinics. However, limited by the high resolution detector size and the radiation dose reduction expectation, the interior scan is commonly desirable for many practical applications, such as cardiac CT [63] and Nano-CT [48]. Inspired by the CS theory, the interior problem has been proved uniquely and stably solvable by
minimizing the ROI's TV provided that the imaging object inside the ROI is piecewise constant [64], [65]. This knowledge regularized CT reconstruction algorithm is called CS-based interior tomography and the key is to minimize the TV of the ROI inside the imaging object. The high computational cost (arithmetic operation and memory bandwidth) of the CS-based signal recovering algorithms hinders the sequential implementations of the iterative algorithms been applied in clinics especially for cone-beam spiral CT reconstruction [59]. Both the projection and backprojection processes are categorized as single instruction multiple data (SIMD) [66] computing model. The SIMD model is quite suitable for parallelization without too much complicated communication, synchronization or mutual lock mechanisms. Initially, parallel image reconstruction algorithms were implemented on clusters [67]. Cell processors were also applied in general purpose parallel computing [68]. Very recently, Intel released the Intel MIC (Intel Many Integrated Core) architecture products [69], and software engineers can run their codes on MIC with little or no additional workload. In early years, researchers began to accelerate their algorithms with graphics processing unit (GPU) when its programming interface was published for general purpose computing. Before GPU was programmable for general purpose computing, general algorithms were dedicatedly camouflaged as graphical operations such as texture mapping for parallel acceleration. Analytical reconstruction algorithms were benefited enormously from GPU [70]. Nowadays, two groups of programming interfaces for general computing in GPU have been developed. One is based on computer graphics languages, such as OpenGL [71], CG [72], HLSL [73], etc. The other is designed for high performance GPU computing, such as CUDA [66], OpenCL [74], Brook [75], etc. The CUDA (Compute Unified Device Architecture) [76]–[78] is rapidly exploited in many fields including medical imaging [78]. Comparing with other general GPU computing interfaces, the CUDA has higher performance and is easier to master and more flexible. Therefore, we choose CUDA to implement the CS-based interior reconstruction in GPU. For higher performance, the shared memory, texture memory and constant memory are applied because of their high bandwidth and caching
mechanism in our implementation. In this paper, the SART and OS-SART reconstruction frameworks with TV minimization are implemented in GPU computing to make the CS-based interior tomography practical. In section II, the CT imaging model is reviewed with a concise SART-type reconstruction framework and the STF method. In section III, the implementation details are given. Section IV demonstrates the numerical results in 2D fan-beam and 3D cone-beam geometries. The SD and STF based TV minimization methods are compared in terms of reconstruction accuracy and speed. In section V, we discuss some related issues and conclude this paper.

1.2 Algorithm Design

1.2.1 Imaging Model

A 2D or 3D digital image can be expressed as \( f = (f_{i,j,k}) \in \mathbb{R}^N \), where \( N = I \times J \) for 2D image and \( N = I \times J \times K \) for a 3D image. \( I, J, \) and \( K \) are image pixel number along length, width and height dimensions. In this paper, both \( f_{i,j,k} \) and \( f_n \) are applied for convenience. Therefore, a CT system can be modeled as

\[
p = Af.
\]

Each component of the vector \( p \in \mathbb{R}^M \) is a measured datum, where \( M \) is the total measurements (the product of the projection number and the detector cell number), and \( A \in \mathbb{R}^{M \times N} \) is the system matrix. Typically, the \( n^{th} \) pixel is viewed as a rectangular area with a constant value \( f_n \), and the \( m^{th} \) projection datum \( p_m \) can be viewed as the summation of all the weighted pixel values involving the \( m^{th} \) x-ray. Lots of discrete models have been proposed to calculate the entries of \( A \), such as linear interpolation method, grid method, distance-driven method [79], Siddons’ method [80], area integral method [81], footprint method [82], etc. To balance the speed and accuracy, the Siddons’ method is adopted as the projection model and the pixel-driven method is adopted as the backprojection model in our experiments. Siddons’ algorithm takes the length of the \( m^{th} \) x-ray
penetrating the \(n^{th}\) rectangular pixel/voxel as the element \(a_{m,n}\) of the system matrix \(A\). An additive noise \(e \in \mathbb{R}^N\) is assumed, and the imaging process is finally modeled as

\[
p = Af + e. \tag{2}
\]

### 1.2.2 SART and OS-SART

The SART-type solution for Eq. (2) is expressed as [83]

\[
f^{(l+1)}_n = f^{(l)}_n + \frac{\lambda}{a_{m,n}} \sum_{m=1}^{M} a_{m,n} \left( p_m - A_m f^{(l)} \right), \tag{3}
\]

where \(a_{m,n} = \sum_{m=1}^{M} a_{m,n} > 0\), \(a_{m+} = \sum_{m=1}^{N} a_{m,n} > 0\), \(A_m\) is the \(m^{th}\) row of \(A\), \(l\) is the iteration index and \(0 < \lambda(l) < 2\) is a free relaxation parameter. For simplicity, let \(\Lambda_{m, n}^+ \in \mathbb{R}^N \times \mathbb{R}^N\) be a diagonal matrix with \(\Lambda_{m, n}^+ = \frac{1}{a_{m,n}}\) and \(\Lambda_{m, m}^+ \in \mathbb{R}^M \times \mathbb{R}^M\) be a diagonal matrix with \(\Lambda_{m, m}^+ = \frac{1}{a_{m,n}}\), Eq. (3) is rewritten as

\[
f^{(l+1)} = f^{(l)} + \lambda(l) \Lambda_{m, n}^+ A^T A^{m+} \left( p - Af^{(l)} \right). \tag{4}
\]

The ordered subset (OS) approach accelerates the convergence of SART by one to two orders of magnitude with the cost of inducing image bias [84], [85]. The projection data is divided into \(T\) disjoint subsets \(B_t = \{i_{t1}^1, ..., i_{tK}^T\}\) where \((t = 1, 2, ..., T)\), is the subset index of the projection. The union of these subsets covers the whole projection set. We have \(B_t \cap B_j = \emptyset\) and \(\bigcup_{t=1}^{T} B_t = \{1, 2, ..., M\}\). These subsets alternatively participate the iterations. The OS-SART is represented as [78]

\[
f^{(l+1)}_n = f^{(l)}_n + \sum_{m \in B_t} \sum_{m' \in B_t} a_{m,n} \frac{p_m - A_m f^{(l)}}{a_{m,n}}, \tag{5}
\]

where \(t = (1 \mod T) + 1\). For every sub-step in one iteration, it is suggested that the selected subset should be with the greatest possible angular distance from the previously used subset [84].

In our applications in this paper, the subset size is set as the detector cell number in one view. Meanwhile, the fast weighting technology in the FISTA (fast iterative shrinkage-thresholding
algorithm) is applied to further accelerate the SART-type algorithms with a constant step size [86].

1.2.3 CS-based Image Reconstruction

To further reduce the projections, the CS-based signal reconstruction method is combined with the OS-SART. The paradigm for CS-based signal recovering is a constrained \( \ell_0 \) norm minimization problem defined as

\[
\mathbf{x} = \underset{\mathbf{x}}{\text{arg min}} \| \mathbf{x} \|_0, \quad \text{s.t.} \quad \mathbf{y} = \mathbf{\Phi x},
\]

where \( \mathbf{x} \) is a sparse signal, \( \mathbf{y} \) is the observed data and \( \mathbf{\Phi} \) is the sensing matrix. To address the NP-hard problem and suppress noise, Eq. (6) is usually modified as

\[
\mathbf{x} = \underset{\mathbf{x}}{\text{arg min}} \| \mathbf{x} \|_1, \quad \text{s.t.} \quad \| \mathbf{y} - \mathbf{\Phi x} \|_2^2 \leq \varepsilon,
\]

where \( \varepsilon \) is the measurement error. For a sparser solution, the \( \ell_p \) norm minimization is also investigated. Generally speaking, the smaller the \( p \) is, the fewer measurements are needed for accurate reconstruction [87]. Actually, most of the signals are far-fetched sparse, and the \( \ell_1 \) norm minimization paradigm is prohibited to be applied straightforwardly to medical image reconstruction. Usually, a sparse transform will be employed first to transform the non-sparse signal to an appropriate sparse domain. The CS-based CT image reconstruction paradigm can be finally expressed as

\[
\hat{\mathbf{f}} = \underset{\mathbf{f}}{\text{arg min}} \| \mathbf{\Psi f} \|_1, \quad \text{s.t.} \quad \| \mathbf{p - A f} \|_2^2 \leq \varepsilon.
\]

1.2.4 DGT-based sparsity and STF

The objective function \( \| \mathbf{\Psi f} \|_1 \) in Eq. (8) can be defined as \( \| \mathbf{f} \|_{TV} \) is the discrete gradient transform (DGT) and the image object satisfy the piecewise constant assumption, where \( \| \cdot \|_{TV} \) denotes the \( \ell_1 \) norm of the DGT. The so-called Neumann condition [88] on the boundary is also assumed. With an appropriate Lagrange multiplier \( \sigma \), the problem Eq. (8) can be rewritten as:

27
\[ \hat{f} = \arg \min_{f} \left( \|p - Af\|_2^2 + \sigma \|f\|_{TV} \right). \]  

(9)

Because the two items in Eq. (9) are convex, they can be alternatively minimized to yield an accurate solution. While the OS-SART can be used to minimize \( \|p - Af\|_2^2 \), the conventional SD method can be employed to minimize the TV term. However, the SD based TV minimization sometimes over-enhances the edge regions and generates Gibbs-effects-like artifacts, and the step size needs to be carefully selected to find the minimum value and guarantee fast convergence.

The \( \text{STF} \) (soft threshold filtering) is an alternative choice to minimize TV. Let \( Z = \{\xi_y\}_{y \in \Gamma} \) be a basis in \( \mathbb{R}^N \). \( f \) can be linearly expressed as \( f = \sum_{y \in \Gamma} \langle f, \xi_y \rangle \xi_y \). Let us define an objective function with positive weights \( \Omega = \{\omega_y\}_{y \in \Gamma} \)

\[ \Xi_{\Omega,q}(f) = \|p - Af\|_2^2 + \sum_{y \in \Gamma} 2\omega_y \|\langle f, \xi_y \rangle\|_q^q, \quad q \in [0, 2]. \]  

(10)

When \( q = 1 \), finding the sparsest solution of \( \Xi_{\Omega,1} \) is equivalent to Eq. (9) for DGT [57]. To find the sparse solution of \( \Xi_{\Omega,1} \), we can recursively minimize \( \Xi_{\Omega,1} \) in a STF framework

\[ \hat{f}^{(l+1)} = S_{\Omega,1} \left( \hat{f}^{(l)} + A^T \left( p - A\hat{f}^{(l)} \right) \right), \]  

(11)

where \( \hat{f}^{(l)} \) is the intermediate image and

\[ S_{\Omega,1}(x) = \sum_{y \in \Gamma} S_{\omega_y,1} \left( \langle x, \xi_y \rangle \right) \xi_y \]  

(12)

is the function that performs a soft-threshold filtering. It has been proved by Daubechies \textit{et al.} in [58] that Eq. (11) is convergent. In Eq. (12), \( S_{\omega_y,1} \) is defined as

\[ S_{\omega_y,1}(x) = \begin{cases} 
  x - \omega_y, & \text{if } x \geq \omega_y \\
  0, & \text{if } |x| < \omega_y \\
  x + \omega_y, & \text{if } x \leq -\omega_y
\end{cases} \]  

(13)

However, because the DGT is non-invertible and violates the restricted isometrics property (RIP [51], [89]), the STF method is prohibited to be directly applied for TV minimization. This problem can be addressed by constructing a pseudo inverse of DGT [57], [59].
1.3 GPU Acceleration

1.3.1 Parallelization Strategy

For the CS-based interior tomography algorithm, the projection, backprojection, DGT, pseudo inverse transform, soft-threshold filtering and finding the optimal threshold can be parallelized. Some constants (such as all the trigonometric values of the view angles, coordinates of the source position, aligned voxel coordinates and the detector coordinates in $x$, $y$ and $z$ directions, etc.) can be pre-calculated and symbolically mapped to constant memory to achieve higher caching efficiency. However, this symbolical linking is optional because it costs almost the same clock cycles as fetching the data from the constant memory to calculate the coordinates corresponding to the current voxel or detector cells on the fly.

The onboard device memory usually is sufficient for fan-beam reconstruction unless storing the whole system matrix $A$. Sometimes, it is also possible to compactly store $A$ in the device memory when the image and the projection data are not too large. We compactly stored a $180,000 \times 65,536$ sparse matrix $A$ in COO (Coordinate list, which stores a list of (row, column, value) tuples) format in CPU. The system matrix is transferred to the CSR (Compressed Sparse Row) format after it is loaded into the device memory. The projection process occupies 6.23ms with cuSPARSE (NVIDIA CUDA Sparse Matrix library). Actually, the whole program spent 2.26 seconds including reading matrix to the main memory and transferring to the device memory, etc. Even though a large device memory is provided, the storing of system matrix has to be gingerly designed to be fully compacted and the computational efficiency should be kept. Assuming that the image to be reconstructed is square, if the symmetry of 2D/3D scanning is fully utilized; only the first eighth of the scanning angles needs to be considered when the scanning range is $[0,2\pi]$. The symmetrical relationship between image indices and sinogram indices are defined as follows:
\[(i, j) \rightarrow (\theta, t)\]
\[(R - j, i) \rightarrow (\theta + \pi / 2, t)\]
\[(R - i, R - j) \rightarrow (\theta + \pi, t)\]
\[(j, R - i) \rightarrow (\theta + 3\pi / 2, t)\]
\[(R - j, R - i) \rightarrow (\pi / 2 - \theta, T - t)\]
\[(i, R - j) \rightarrow (\pi - \theta, T - t)\]
\[(j, i) \rightarrow (3\pi / 2 - \theta, T - t)\]
\[(R - i, j) \rightarrow (2\pi - \theta, T - t)\]  \hspace{1cm} (14)

where \(i, j\) represents the pixel indices of the image, \(\theta\) is the current projection angle that can be easily transferred to the angle index, \(t\) is the index of the detector element and \(T\) is the detector resolution, and \(R\) is the image resolution in length or width direction. The pairs \((i, j)\) and \((\theta, t)\) satisfy the Radon transform relationship. In cone-beam geometry, not only the scanning range but also the projection symmetry on the upper and downer parts of the detector can be utilized if there is no offset. However, these symmetrical based tricks are not adopted here due to detector offset need to be considered in practical applications. Typically, a sinogram is divided into adjoining sub-blocks in fan-beam case for parallel projection. Each sub-block corresponds to one thread block in CUDA and the threads in one block can be configured discretionarily. The thread block is divided into \((p_x \times p_y)\) threads, where \(p_x\) addresses the detector indices and \(p_y\) addresses the projection angle indices with built-in variables in CUDA. On one hand, sufficient threads are required to accurately reconstruct the image object. On the other hand, it wastes computational resource if more threads are allocated than what are needed. It is optimal that the detector cell and the projection angle numbers are divisible by \(p_x\) and \(p_y\), respectively. The thread number being multiple of 32 in one block is sensible because the GPU executes 32 threads as a warp simultaneously. Fully occupying 1024 threads in one block is not optimal in Fermi architecture. The maximum warps for one multiprocessor in Fermi architecture is 48 that implies 1536 threads can be executed simultaneously. If the threads block is configured with fully 1024 threads, 1/3 of the computation resource will be idle. In the Kepler architecture, the maximum warp number increases to 64 in one multiprocessor. Therefore, 2048 threads can be executed simultaneously.
Without considering the backward compatible for earlier GPUs, 1024 threads fill up one thread block in Kepler architecture and 512 threads are configured in Fermi architecture.

### 1.3.2 Projection and Backprojection Models

![Boundary box based backprojection process](image)

**Figure 1.** Boundary box based backprojection process. (a) The backprojection is accurate with higher computational cost when the detector size is smaller. (b) The backprojection degenerates to the pixel/voxel driven method with more computational cost when the detector size is large.

Siddons’ algorithm is adopted as the projection model. An element $a_{m,n}$ of system matrix $\mathbf{A}$ is computed as the length of $m^{th}$ x-ray passing through $n^{th}$ pixel/voxel for a 2D/3D image. In conventional CPU and GPU based implementations, additional memories are pre-allocated to store the parameters representing the intersections between an x-ray and the boundary box of each pixel/voxel. The pre-assigned memory should be at least $(N_x + N_y + N_z + 3) \times S_D$ bytes in cone-beam geometry, where $N_x$, $N_y$, and $N_z$ are the pixel numbers of the volume in length, width and height dimensions respectively, and $S_D$ is the data size of each voxel. This may cause trouble for the CUDA programming with GK104 chip because it is banned to dynamically allocate device memory in the executing kernel. It is required to recompile the program if the image resolution changes and it also has to be implemented with macros to allocate the constant memory for storing the parameters. Although the constant expression feature in C++11 standard can solve this
problem with the key word “constexpr”, this feature is not supported in Microsoft Visual Studio 2012 compiler. Therefore, a modified Siddons' algorithm without pre-assigning memory was applied [90]. This algorithm first calculates the intersections of the compact support of the 2D/3D image and the current x-ray path, which is similar to the clipping algorithm in computer graphics [91]. Beginning with the incident point, the algorithm calculates the intersection points for each pixel/voxel for its weighting coefficient $a_{i,j}$. Iteratively, the exit point of the current pixel/voxel is set as the incident point of its neighbor pixel/voxel and this process will be repeated until the current incident point is the exit point located on the boundary of the object. After the current intersection point arrives at the exit point on the compact support, the innate ordered intersection parameters are all generated and the weighting for these intersection voxels can be calculated. The pixel/voxel driven model is adopted to implement backprojection operation. For a given projection angle, mapping a point in the world coordinate to its corresponding detector index can be simplified to a geometrical transform matrix $M$ in flat-panel detector case. This matrix can be constructed as follows. A homogeneous coordinate is assumed in our transform. The source and detector positions are both counter clockwise rotated to make the source on the positive direction of Y axis with a rotation matrix $R_{\theta}$, and the source is moved to the origin of the world coordinate by multiplying a matrix $T_d$. After a perspective transform with $P_d$, the transformed object point is projected to its shadow on the detector. Finally, the shadow index on the detector can be easily calculated with two matrices $T_m$ and $S_u$. For concise, the offset of the detector, which can be easily integrated into the matrix multiplication in $T_m$ is not considered in this derivation. Therefore, the matrices chain can be expressed as
\[ M = S_u \cdot T_m \cdot P_d \cdot T_d \cdot R_\theta \]

\[
= \begin{bmatrix}
\frac{1}{u_x} & 0 & 0 \\
0 & \frac{1}{u_z} & 0 \\
0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
1 & 0 & -m_x \\
0 & 1 & -m_z \\
0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
-D & 0 & 0 & 0 \\
0 & 0 & -D & 0
\end{bmatrix}
\]

\[
\times
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & -d \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
\cos \theta & \sin \theta & 0 & 0 \\
-\sin \theta & \cos \theta & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

where \( u_x \) and \( u_z \) are detector element size, \( m_x \) and \( m_z \) are minimum coordinate of the detector at the initial position, \( D \) is the source to detector distance, \( d \) is the source to iso-center distance and \( \theta \) is the current view angle. Because the ray-driven projection model mismatches the pixel/voxel driven backprojection process which means the system matrix in backprojection is not exactly the transpose of the projection system matrix. This mismatch will induce artifact with the increase of iteration number. Therefore, a method similar to Li et al. [92] is adopted to make the projection and backprojection match. The projection of a voxel on a plane can only be a convex polygon such as hexagon, pentagon or quadrilateral. The ray passing through the voxel must be inside its convex polygon shadow. First, for every voxel the boundary convex polygon is calculated. Then, the rectangle boundary box is calculated from the minimum and maximum projection coordinates of the voxel. Only the detector cells inside the boundary rectangular are considered as demonstrated in Figure 1. Boundary box based backprojection process. (a) The backprojection is accurate with higher computational cost when the detector size is smaller. (b) The backprojection degenerates to the pixel/voxel driven method with more computational cost when the detector size is large. However, if the boundary rectangular is not large enough to contain multiple detector cells, it has no advantage over the simple pixel driven method but needs more computational complexity. When the detector resolution in pitch direction is the same as or smaller than the object resolution in the height dimension, some of the pixels in different slices will never be penetrated by any cone-beam x-ray in projection process. This will result in serious artifacts. To suppress this artifact, when the detector resolution is insufficient for high resolution
reconstruction, we have to compromise the backprojection model to solve the model mismatching problem. Due to the specificity of this algorithm, a single thread is mapped to the vertex of a voxel instead of a single voxel. It is readily appreciated that 8 adjacent voxels share one vertex in three directions, and the redundant coordinate computing will occur if one thread response for one voxel updating. Therefore, the shared memory is applied to minimize latency caused by reading the same data from global memory multiple times and unnecessary computing. The thread block for backprojection is allocated as (8, 8, 8) and the thread index addresses current vertex. Because only the first 7 indices in each dimension response to the voxel updating, each thread block can update 343 voxels.

1.3.3 Parallelization of TV Minimization

In 3D case, a general $\ell_p$ norm of DGT is defined as

$$L_p(D(f)) = \left( \sum_{k=1}^{K} \sum_{j=1}^{J} \sum_{i=1}^{I} |D(f_{i,j,k})|^p \right)^{\frac{1}{p}},$$  

(16)

where

$$D(f_{i,j,k}) = \sqrt{(f_{i+1,j,k} - f_{i,j,k})^2 + (f_{i,j+1,k} - f_{i,j,k})^2 + (f_{i,j,k+1} - f_{i,j,k})^2}.$$  

(17)

when $p = 1$, Eq. (16) degenerates to the TV. For an individual DGT value $D(f_{i,j,k})$ at the position $(i, j, k)$, it involves four values ($f_{i+1,j,k}, f_{i,j+1,k}, f_{i,j,k+1}$ and $f_{i,j,k}$) which are sporadically stored in device global memory. Initially, this suggests each thread calculates one DGT value in position $(i, j, k)$ with a simple kernel function. Therefore, the threads configurations in fan-beam and cone-beam cases are the same as the configurations in the corresponding backprojection cases, respectively. In fact, the DGT is not intensive in arithmetic computation but in memory bandwidth. If all the data are stored in the global memory, every thread has to read four discontinuous data from the global memory and adjacent threads in one block have to read the same data which generate latency. Therefore, the volume is divided into overlapped sub volumes
with size $8 \times 8 \times 8$. For one thread block, its indexed sub-volume is copied to the shared memory. In fan-beam case, similar strategies is applied, the image is divided into $32 \times 32$ overlapped sub-image, and for one thread block the corresponding sub-image is loaded into shared memory as shown in Figure 2. Because the value in last index of the sub image is the first index of its adjoining sub image, if the thread in each block is allocated as $(t_x, t_y)$ and the image size is $(N_x, N_y)$ in fan-beam case, the block number for DGT is $\left(\left\lfloor \frac{N_x + t_x - 2}{t_x - 1} \right\rfloor, \left\lfloor \frac{N_y + t_y - 2}{t_y - 1} \right\rfloor\right)$. Similarly, the block number in 3D case can be calculated using a similar formula.

Finding an optimal threshold after DGT is an indispensable step to accelerate the convergence of the STF, which can be achieved by the dichotomy in GPU. The Thrust library is applied for this implementation [91]. We first copied the discrete gradient image $\hat{d}^{(l)}$ to $\hat{d}_c^{(l)}$ and then $\hat{d}_c^{(l)}$ is ascending sorted. The prior knowledge of the intermediate discrete gradient image can be estimated from the roughly reconstructed image by the classical FBP approach. The soft-threshold filtering is applied with respect to each component according to Eq. (13). The pseudo inverse transform will be applied to inversely transform the image from the DGT to image domain, which is implemented in GPU according to Eq. (3.8) to Eq. (3.11) in [57]. The memory bandwidth problem can also be solved by the shared memory similar to the DGT implementation.

Figure 2. The thread configurations in fan-beam reconstruction. (a) represents one thread block configuration for back-projection and DGT. (b) represents one thread block configuration for TV descent direction calculation and pseudo-inverse of DGT.
Different from the DGT transform, the inverse threshold filtering needs more shared memory. In 2D case, the thread block is also configured with size \((32 \times 32)\). Considering that the inverse transform in position \((i,j)\) relates to all its adjacency pixels, only \(30 \times 30\) pixels in the center of the sub image can be calculated by the current thread block as in Figure 2. Therefore the block number should be \(\left\lfloor \frac{N_x+t_x-3}{t_x-2} \right\rfloor, \left\lfloor \frac{N_y+t_y-3}{t_y-2} \right\rfloor\) if the image resolution is \(N_x \times N_y\) and the threads configuration in one block is \((t_x, t_y)\). The block number for 3D pseudo inverse transform can be calculated in a similar fashion.

### 1.3.4 Overall Pseudo-codes

Combining the OS-SART and STF, the CS-based interior tomography can be implemented as the following pseudo codes:

**S1**: Initializing \(l = 0, \hat{f}^{(l)} = 0\) and estimating \(\omega_{\gamma_0}\);

**S2**: Updating \(l \leftarrow l + 1\) and performing OS-SART to update \(\hat{f}^{(l)}\);

**S3**: Performing DGT from \(\hat{f}^{(l)}\) to \(\hat{d}^{(l)}\) corresponding to \(\langle \hat{f}, \zeta \rangle\) in Eq. (13);

**S4**: Performing soft-threshold filtering from \(\hat{d}^{(l)}\) to \(\tilde{d}^{(l)}\) using Eq. (14);

**S5**: Performing pseudo inverse DGT to obtain \(\tilde{f}^{(l)}\);

**S6**: Updating \(\tilde{f}^{(l)} \leftarrow \tilde{f}^{(l)}\);

**S7**: If the stopping criteria are met, output the results; otherwise go to S2.

### 1.4 Numerical Experiments

#### 1.4.1 Platform Configuration and Geometry Parameters

The SART and OS-SART algorithms are implemented in GPU for both fan-beam and cone-beam geometries with CUDA/C++ language. For the CS-based interior tomography, the TV regularization is applied with the aforementioned SD and STF methods. All the experiments are tested on a high-performance workstation configured as follows. Two Intel Xeon CPUs are set up.
with core clock frequency 3.10GHz. Each CPU contains 16 cores. The memory size is 32GBs. The operating system is Microsoft Windows 7 (64-bits Professional version). For GPU computing, NVIDIA Tesla K10 is used including 2 GK104s. Each GPU contains 1536 CUDA cores, and the GPU clock frequency is 745MHz. The device memory clock frequency is 2500MHz and the bus width is 256 bit. The device memory for each GPU is 3GB. For cone-beam reconstruction, the system geometry is configured as in Table I. The system geometry in fan-beam reconstruction is the same as the central slice parameters in cone-beam geometry. In Table I, CATCE means that the value will be different in different groups of experiments and it will be

![Bar chart of the computational cost for different GPU-based reconstruction methods in fan-beam geometry. The left column is for image size 1024×1024, and the right column is for image size 2048×2048 from 50 iterations. The bottom row is the speedup comparison in CPU and GPU. The abscissa indicates different projection views used in the reconstruction. The mandatory axis is the reconstruction time (in seconds). The secondary axis is the speedup factor.](image)

Figure 3. Bar chart of the computational cost for different GPU-based reconstruction methods in fan-beam geometry. The left column is for image size 1024×1024, and the right column is for image size 2048×2048 from 50 iterations. The bottom row is the speedup comparison in CPU and GPU. The abscissa indicates different projection views used in the reconstruction. The mandatory axis is the reconstruction time (in seconds). The secondary axis is the speedup factor.
clarified in the experiments. In the numerical simulations, Poisson noise is assumed [93] and the photon number for each detector element is $10^4$.

**Table I. Cone-beam reconstruction geometry configuration for numerical simulation and clinical dataset.**

<table>
<thead>
<tr>
<th></th>
<th>Simulation Data</th>
<th>Real Patient Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>S2D</td>
<td>1000mm</td>
<td>947mm</td>
</tr>
<tr>
<td>S2O</td>
<td>850mm</td>
<td>539mm</td>
</tr>
<tr>
<td>Detector Type</td>
<td>Equidistant</td>
<td>Equiangular</td>
</tr>
<tr>
<td>Scan range</td>
<td>0 to 360 degrees</td>
<td>-267 to 93 degrees</td>
</tr>
<tr>
<td>View numbers</td>
<td>CATCE</td>
<td>2200</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Detector Parameters</th>
<th>Size</th>
<th>340²mm²</th>
<th>Arc</th>
<th>0.959285172</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Height</td>
<td>65.5296mm</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Entries per row</td>
<td>CATCE</td>
<td></td>
<td>888</td>
</tr>
<tr>
<td></td>
<td>Truncation ratio</td>
<td>50%</td>
<td></td>
<td>43.70%</td>
</tr>
<tr>
<td></td>
<td>Row Number</td>
<td>CATCE</td>
<td>64</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Pitch</td>
<td>CATCE</td>
<td>1.0239mm</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Offset</td>
<td>NONE</td>
<td>0.767925mm</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Image Parameters</th>
<th>Size</th>
<th>200³mm³</th>
<th>500² × 40mm³</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resolution</td>
<td>CATCE</td>
<td>512² × 64</td>
<td></td>
</tr>
<tr>
<td>Slices interval</td>
<td>CATCE</td>
<td>0.625mm</td>
<td></td>
</tr>
</tbody>
</table>

**1.4.2 Reconstruction Time comparison**

Because the CUDA kernels immediately return after they are called, the general timer function cannot be applied to test the performance of a kernel. We imitated the examples in CUDA SDK to test the reconstruction time in fan-beam and cone-beam cases. Visual Profiler 6.0 is also applied for more detailed analysis.
Figure 4. A real medical image volume of a patient reconstructed with OS-SART algorithm. (a), (b) and (c) are the transverse, sagittal and coronal planes, respectively. The display window is [0, 1400]. Figure (d)-(f) are the reconstruction differences compared with the FDK method, which is almost unobservable. The bottom line (g) are the pseudo-color volume rendering with of the reconstructed result in different observation positions.

The OS-SART algorithm for fan-beam reconstruction with STF based TV minimization is tested in double floating precision. To investigate the relationships among image resolutions, view numbers, and the GPU based reconstruction speedups, the image resolution varies from $256^2$ to $2048^2$, the detector resolution ranges from 300 to 2400, and the view number changes from 17 to 360. The total iteration is 50. As summarized in Table II, when the view number is smaller, the speed up is more significant. Larger view numbers or smaller subset sizes indicate more projection and backprojection calls in one sub-loop. When the view number increases, the projection and backprojection change back and forth frequently especially when the subsets are the same size as that of one projection view. Meanwhile, the GPU-computing is suitable for CS-
based interior reconstruction especially when only a few projections are available. With the improvement of the image resolution, the speedup is more evident. This is because when the image is small, the projection and backprojection generally are more bandwidth intensive instead of computing intensive.

Figure 5. Interior volume reconstruction result from the truncated scanning data of a patient with OS-SART algorithm plus TV regularization. The image resolution is $512^2 \times 64$. The image size on transverse plane is $230^2 \text{mm}^2$. (a), (b) and (c) are the transverse, sagittal and coronal planes, respectively. The display window is [0, 1400]. (d) is the pseudo-color volume rendering observing from two positions.

The SD-based TV minimization in the OS-SART framework is also tested to compare with the STF-based algorithm with image resolution $1024^2$ and $2048^2$ in fan-beam geometry. The bar chart in Figure 3 shows the speedup results. The reconstruction time between the OS-SART and the other two TV regularization based algorithms are minimal, and the STF-based TV minimization algorithm runs a little bit faster than the SD-based TV minimization.

A single-floating-precision $512^3$ modified 3D Shepp-Logan phantom is applied to test the speedup performance in cone-beam geometry with the OS-SART algorithm plus TV regularization. A detector resolution of $600^2$ and 100 views are applied. The computational cost for the STF-based and the SD-based TV minimization in the OS-SART framework are respectively 1665.46 and 1791.33 seconds after 85 iterations. For one iteration step, the
computational costs in different algorithms are listed in Table III. The sum of projection, backprojection and regularization time is not exactly the same as the total time because other time is needed for data loading from disk, data transfer, functions calling and so on. The computational cost is relatively small to find the optimal threshold $\omega_{\gamma_0}$ in the OS-SART with STF-based TV minimization. With more detailed analysis, the GPU utilization for the projection and backprojection steps are both 100.0%, the utilization to find the optimal threshold $\omega_{\gamma_0}$ including filtering is 100.0%, while the occupation rate for DGT is only 75%.

**Table II. The computational cost for STF-based TV minimization in an OS-SART framework with a modified Shepp-Logan phantom. A comparison is presented between CPU (Intel Xeon 3.1GHz, Single Core usage) and GPU (NVIDIA, Tesla K10, single GPU usage) after 50 iterations.**

<table>
<thead>
<tr>
<th>View number</th>
<th>Image size</th>
<th>Detector size</th>
<th>Time (s)</th>
<th>Speedup factors</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>CPU</td>
<td>GPU</td>
</tr>
<tr>
<td>17</td>
<td>256$^2$</td>
<td>300</td>
<td>19.8</td>
<td>1.1</td>
</tr>
<tr>
<td></td>
<td>512$^2$</td>
<td>600</td>
<td>70.9</td>
<td>2.5</td>
</tr>
<tr>
<td></td>
<td>1024$^2$</td>
<td>1200</td>
<td>257.7</td>
<td>7.0</td>
</tr>
<tr>
<td></td>
<td>2048$^2$</td>
<td>2400</td>
<td>1066.2</td>
<td>23.4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>View number</th>
<th>Image size</th>
<th>Detector size</th>
<th>Time (s)</th>
<th>Speedup factors</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>CPU</td>
<td>GPU</td>
</tr>
<tr>
<td>180</td>
<td>256$^2$</td>
<td>300</td>
<td>63.5</td>
<td>8.7</td>
</tr>
<tr>
<td></td>
<td>512$^2$</td>
<td>600</td>
<td>320.8</td>
<td>22.7</td>
</tr>
<tr>
<td></td>
<td>1024$^2$</td>
<td>1200</td>
<td>1176.5</td>
<td>65.6</td>
</tr>
<tr>
<td></td>
<td>2048$^2$</td>
<td>2400</td>
<td>4769.9</td>
<td>220.8</td>
</tr>
</tbody>
</table>

A single-floating-precision 512$^2 \times 256$ image volume is also reconstructed with the OS-SART. The projection geometry is in Table I except that the detector resolution changes to 1024 $\times$ 512, and the views increase to 360. An iteration number 10 is applied to this study. The total reconstruction time is 423.20 seconds. In one iteration, 42.3 seconds are needed to project the 512$^2 \times 256$ image volume to 1024 $\times$ 512 $\times$ 360 projections and to backproject the projections. From the performance comparison with different reconstruction scales in image resolution,
detector resolution, and views; we can see that the performance of boundary box based backprojection is widely influenced by the detector resolution. To further accelerate the reconstruction, the dataset is divided into two smaller ones and distributed to dual GPUs evenly. Except for the same geometry configuration, both the volumetric image and the raw projections are halved. An iteration number 30 is assumed. There is no data redundancy to guarantee an accurate reconstruction since the symmetry of circular cone-beam scanning geometry. While the computational cost is 583.46 seconds for single Kepler GK104, the computational cost is reduced to 289.85 seconds with two Kepler GK104s. The GUPS (Giga-updates per second) is also tested for the backprojection step. On average, it takes 7.39 seconds for the backprojection step to update $512^3$ image from 360 views in once iteration. This implies that it spends 20.52ms, and the GUPS is 6.09 for each view.

Table III. The computational cost for the projection, backprojection and TV minimization steps in OS-SART, OS-SART with SD-based TV minimization, and OS-SART with STF-based TV minimization.

<table>
<thead>
<tr>
<th></th>
<th>Total Time</th>
<th>Projection</th>
<th>Backprojection</th>
<th>Regularization</th>
</tr>
</thead>
<tbody>
<tr>
<td>OS-SART</td>
<td>Time: 18.221s</td>
<td>9.479s</td>
<td>6.547s</td>
<td>0s</td>
</tr>
<tr>
<td></td>
<td>Ratio: 100%</td>
<td>59.0%</td>
<td>40.7%</td>
<td>0%</td>
</tr>
<tr>
<td>SD-based</td>
<td>Time: 21.048s</td>
<td>9.472s</td>
<td>6.547s</td>
<td>1.587s</td>
</tr>
<tr>
<td></td>
<td>Ratio: 100%</td>
<td>53.1%</td>
<td>36.7%</td>
<td>8.9%</td>
</tr>
<tr>
<td>STF-based</td>
<td>Time: 19.569s</td>
<td>9.473s</td>
<td>6.547s</td>
<td>0.729s</td>
</tr>
<tr>
<td></td>
<td>Ratio: 100%</td>
<td>55.6%</td>
<td>38.4%</td>
<td>4.30%</td>
</tr>
</tbody>
</table>
Figure 6. Representative results reconstructed from truncated local projections for a modified Shepp-Logan phantom. From the left to right columns; the images were reconstructed from 17, 21, 72,180 and 360 projections, respectively. The iterations are 30. From the top to bottom rows, the images were reconstructed by the OS-SART, OS-SART with the steepest descent and OS-SART with soft-threshold filtering for TV minimization, respectively. The display window is [0, 1].

The recently released unified memory technique in CUDA 6.0 creates a pool of managed memory that is shared between CPU and GPU. It bridges the CPU-GPU gap. This technique makes the programming convenient without considering the data transfer among devices. At the same time, it simplifies the huge volumetric reconstruction when the GPU device memory is insufficient. As the documentation suggested, the raw projection data and the volume to be reconstructed are both declared with keyword “__managed__”. We implemented a miniature of the projection and backprojection with unified memory technique. The performance from one view with the same geometry configuration makes us a little frustrating. Projecting a $512^2 \times 256$ volume to $1024 \times 512$ detector takes 214.312ms (22.5% of the time) and backprojecting the same data costs 736.69ms (77.5% of the time). It needs more intricate implementation details to accelerate the performance which will be an extension of our work.
Figure 7. Reconstructed results of a cardiac region from clinical projections. (a) is the reference image, (b) is reconstructed by the SD-based TV minimization, and (c) is reconstructed by the STF-based TV minimization. From (a) to (c), the display window is [0, 1132]. (d) and (e) are image differences between (a), (b) and (a), (c) respectively. The display window is [-571, 580]. The STF-based TV minimization reconstruction result outperforms the image with SD-based TV minimization reconstruction.

Under the approval of the institutional review board of Wake Forest University Health Sciences, a clinical patient dataset is also reconstructed with the configuration in Table I. To utilize all the GPU resources in our device fully, three GK104 chips including two Tesla K10s and one GeForce GTX 670 are all occupied. This configuration can be approximately viewed as cone-parallel geometry because the divergence angle is small enough to be ignored and the projection data can be evenly distributed to three GPUs without data redundancy. When only SART is used to reconstruct the image volume, the total computational cost for 160 iterations is 1268.46 seconds. This means 7.93 seconds are needed for one iteration on average. On the other hand, if only one core is applied in CPU implementation, 884.07 seconds are required for one iteration. Therefore, GPU implementation can accelerate the reconstruction more than 110 times in this case. The ordered subset technique is also applied to accelerate the convergence. The projection is
divided into 50 subsets, and each subset contains 44 views. Totally, 536.12 seconds are required for 40 iterations which guarantee a promising result as shown in Figure 4, and 13.40 seconds are needed for one iteration. This is caused by the following reason: when the subset number increases, although it can accelerate the convergence with more projection and backprojection processes to traverse all the projection views, it will slow down the execution of one loop. The interior scan is simulated by truncating 43.7% of the clinical dataset as in Table I. Because only the interior part is illuminated due to the projection truncation, the FOV is reduced to $230^2 mm^2$. The projection is divided into 22 subsets for the OS-SART. The computational cost is 161.203 seconds in total for 20 iterations. This means that ~2.5 minutes can give promising interior reconstruction results as shown in Figure 5.

1.4.3 Image Quality

![Image Quality](image)

*Figure 8. Representative results of the OS-SART with SD-based TV minimization (the top row images) and the OS-SART with STF-based TV minimization (the bottom row images) in the transverse (left column), sagittal (middle column) and coronal (right column) views. The display window is [0,1]. The subfigures (a-1) and (d-1) are the magnified parts of (a) and (d), (a-2) and (d-2) are the error images of (a-1) and (d-1) in reference to the original phantom. It can be seen that the STF-based OS-SART can keep more fine details than the SD-based OS-SART.*

First, the image quality of interior reconstruction is evaluated with a modified Shepp-Logan phantom in fan-beam geometry. The iteration number is 30. Some representative interior
reconstruction results are shown in Figure 6. The images reconstructed by the OS-SART from few view projections show serious artifacts inside the internal ROI. The SD-based and STF-based TV minimizations have similar performance after decades of iterations. The slight differences between SD-based and STF-based TV minimizations can be seen from the reconstructions of highly sparse views.

Another group of reconstruction comparison is based on the clinical patient dataset. The sinogram for the central slice in Figure 5 is extracted and uniformly down-sampled from 2200 to 180 views. The iterations number is also 30. The interior parts of the reconstruction results are shown in Figure 7. The RMSE (root-mean-square deviation) is calculated for the ROI indicated by the red circle in Figure 7(a) to evaluate the image quality quantitatively. While the RMSE of Figure 7(b) is 88.28HU, the RMSE of Figure 7(c) is 79.81HU. The SSIM (the structural similarity index) is calculated to evaluate the similarity [94]. The SSIM for the STF-based OS-SART is 0.9140 while the SSIM for the SD-based OS-SART is 0.8723. The two quantitative measurements show that the STF based OS-SART outperforms the SD-based OS-SART in this study.

![SSIM values in transverse, sagittal and coronal direction in all slices](image.png)

*Figure 9. The SSIM indices of all the slices in transverse, sagittal and coronal observation directions. The SSIM values in the transverse plane (a) always larger than 0.925 indicating promising reconstruction results. Even though the SSIM values in the sagittal plane(b) are a little small, on average, the results are satisfying. Because at the first and last several slices in sagittal and coronal directions, the images are entirely blank, they cause the SSIM values perfectly being 1.*
In cone-beam geometry, a $512^3$ single floating precision modified Shepp-Logan phantom is reconstructed from 72 views. The detector resolution is $600^2$ with the simulation configuration summarized in Table I. The reconstructed results in transverse, sagittal and coronal planes are shown in Figure 8. The SSIMs on these three planes are 0.9587, 0.9850 and 0.9690, respectively in the STF-based reconstruction method. In the SD-based reconstruction method, the corresponding SSIM values are 0.9279, 0.9226 and 0.9029, respectively. Therefore, the STF-based reconstruction keeps more fine structures, and the residual errors are smaller than GD-based reconstruction.

For the clinical patient dataset, it can be easily observed from Figure 4 that the differences are tiny between the images reconstructed by the OS-SART and FBP algorithm. Moreover, as shown in Figure 7, the STF-based TV minimization provides better reconstruction result compared to the SD-based TV minimization for interior tomography. To validate the convergence of the STF-based OSSART algorithm, 50 views are uniformly sampled from the sinogram of the central slice to reconstruct an image of $256^2$. Figure 9 shows the reference and seven intermediate images with respect to different iteration numbers. From images Figure 9(b) to Figure 9(h), the image quality improves gradually with the increase of the iteration number. However, when the iterations number is sufficiently large (e.g. 200), the image quality becomes stable. From the convergence curve in Figure 10, one can see that the reconstruction error (the sum of pixel error squares) decreases rapidly in the first decades of iterations, reaches the minimizer after 130 iterations, then increases a little bit and finally becomes stable after 200 iterations.
Figure 10. Interior reconstruction results from 50 projections after different iterations. The image size is $256 \times 256$. (o) is the reference image which was reconstructed by the FBP method from 2200 global projections. (a) to (g) are the reconstructed images from 50 projections after 10, 20, 100, 200, 500, 1000 and 5000 iterations, respectively.

1.5 Conclusion

The x-ray CT is one of the most important imaging modalities for non-destructive diagnosis and image-guided intervention despite the potential radiation risks. To reduce the radiation dose, we have implemented the CS-based interior reconstruction in GPU for fan-beam and cone-beam reconstruction in this paper. The TV regularization is adopted in our work by incorporating the SD or STF methods. These two approaches are both implemented and compared. To test the reconstruction performance, we performed several groups of experiments with different reconstruction parameters from simulated and real datasets in both fan-beam and cone-beam geometries.
Figure 11. Reconstruction error curves with respect to iterative number. The errors are computed in reference to red circle region in the image (o) in Figure 10. Two subfigure inside the main figure is the local amplification of the main curve. We can observe that when the iterations index is large, the STF-based reconstruction converges slower and slower, sometimes, the RMSE will increase with more iterations.

The GPU parallel computing can be used to boost the CS-based interior tomography for practical applications. We implemented the CS-based interior tomography in GPU devices for fast reconstruction. Our experimental results show that the OS-SART with STF-based TV minimization method runs slightly faster than the SD-based TV minimization and reconstruct promising results in fan-beam geometry using one GPU for acceleration. In the cone-beam geometry experiments, the STF-based method outperforms the SD-based method for few-view projections. Comparing to the CPU-based implementation in fan-beam geometry, the GPU implementation speedup is higher when the views are smaller, or the image resolutions are larger in cone-beam case, and the reconstruction speedup with real data is evident. Therefore, the GPU parallelization is suitable for CS-based interior tomography, especially for large-scale volumetric reconstruction. By analyzing the timeline in cone-beam reconstruction, it is found that the projection and backprojection operations dominate the reconstruction cost in the STF-based method. In the near future, we will optimize the implementation and investigate other projection and backprojection models for possible high efficient GPU implementation to reduce the
computational cost further. Other regularization methods will also be studied with implementations in GPU to decrease the number of views for interior reconstruction further. The unified memory technique will be explored to deal with the large volumetric reconstruction cases.

It is not surprising that the STF-based OS-SART outperforms the SD-based OS-SART. The SD-based TV minimization requires tentatively choosing the parameters such as the descent steps and the descent iteration numbers. It is more likely to find the minimum solution with small steps while the convergence is inferior and the descent iteration number influences the reconstruction speed. On the other hand, the deficiency of STF is its convergence rate. To accelerate the convergence, it is necessary to choose the optimized threshold which is related to the dichotomy. The searches range from several times to dozens of times. Actually, there are no too much visual differences between the results reconstructed by the SD-based OS-SART and the STF-based OS-SART. The RMSE and SSIM show that the reconstruction results are comparable with more iteration. For a clinical volumetric reconstruction, several minutes should give promising interior reconstruction results in a realistic setting.
CHAPTER II. GPU-Based Branchless Distance-Driven Projection and Backprojection

Abstract:

The importance of x-ray CT imaging has never been underestimated for its high spatial resolution, high temporal resolution, and fast scanning speed. Projection and backprojection operations are essential in a variety of image reconstruction and physical correction algorithms in CT. The distance-driven (DD) projection and back projection are widely used for their highly sequential memory access pattern and low arithmetic cost. However, the original DD algorithm has an inner loop that adjusts the calculation depending on the relative position between voxel and detector cell boundaries. The irregularity of the branch behavior makes it difficult to implement on massively parallel computing devices such as graphics processing units (GPUs). Such irregular branch behavior can be eliminated by factorizing the DD operation as three branchless steps: integration, linear interpolation, and differentiation, all of which are highly amenable to massive vectorization. However, to our knowledge, the branchless algorithm has only been reported as 1D proof-of-concept examples. It has not been applied to practical CT imaging problems or shown with substantial speed up compared to the standard DD algorithm. In this paper, we propose a highly parallel branchless DD algorithm for 3D cone beam CT. The algorithm utilizes the texture memory and hardware interpolation on GPUs to achieve fast computation speed. When tested with the GE Discovery CT750 HD scanner, the proposed branchless DD algorithm achieved 137-fold speedup for forward projection and 188-fold speedup for backprojection relative to a single-thread CPU implementation. Compared with a state-of-the-art 32-thread CPU implementation, the proposed branchless DD P/BP achieved 8-fold acceleration for forward projection and 10-fold acceleration for back projection. We also implemented iterative reconstruction algorithms with the branchless algorithm and evaluated its
computation time per iteration and computational accuracy under various geometrical configurations. It was demonstrated with both simulation and real datasets that the GPU-based branchless DD algorithm obtained visually identical images as the CPU reference algorithm.

2.1 Introduction

X-ray computed tomography (CT) is one of the main modern imaging modalities widely applied in clinical diagnosis [1], pharmaceutical industries [2], [3] and other non-destructive evaluations[4], etc. The importance of x-ray CT has never been underestimated for its high spatial resolution, high temporal resolution, and fast scanning speed. However, modern CT systems require a variety of modeling, correction, and image reconstruction algorithms to generate high-quality images.

Forward and back projection (P/BP) operations are the key components of many CT algorithms. The projection operation is widely applied in the simulation of CT imaging systems or acquisition processes [95]–[97]. Another important application of P/BP is in iterative reconstruction algorithms (IRAs), where P/BP are performed alternatively to find an optimum solution that fits the measurements and prior knowledge according to an objective function. IRAs are becoming more popular because they can improve image quality when the projection data are truncated or very noisy. It is possible to integrate IRAs with physical models, statistical models, detector response model, and prior knowledge. Despite the various advantages of IRAs, they usually demand high computational cost dominated by the P/BP operations.

The distance-driven (DD) model [98], [99] is widely applied in P/BP for their highly sequential memory access pattern and low arithmetic cost on CPU platforms. Its overlapping kernel avoids high-frequency artifacts and ensures that the P/BP operators are matched. One drawback of the original DD algorithm is that its inner loop adjusts the calculation with an if-else branch depending on the relative position between voxel and detector cell boundaries. For fan-beam or
cone-beam tomography, the pattern of voxel and detector cell boundaries is non-uniform when it is mapped to the common axis, resulting in irregularity and poor predictability of the branch behavior, making it difficult to implement on many-core vector computing devices such as GPUs.

To overcome the irregular branch behavior of the original DD algorithm, Basu and De Man proposed a branchless DD algorithm [100] by factorizing the DD operation as three branchless steps: integration, linear interpolation, and differentiation. All three steps are highly parallelizable and appear very suitable for vectorized implementation. However, to our knowledge, the branchless algorithm has only been reported as a 1D proof-of-concept example. It has not been applied to practical CT imaging problems or shown with substantial speed up compared to the standard DD algorithm. In this study, we extend the branchless DD P/BP algorithm [100] to 3D cone-beam CT geometry and implement it on GPU. We also apply the proposed branchless DD algorithm to IRAs from simulation and real CT datasets.

The rest of the paper is organized as follows. In section II, we will briefly review several P/BP models, and some GPU relevant CT image reconstruction works. In section III, the conventional DD algorithm and the branchless DD algorithm will be reviewed. The branchless DD algorithm will be extended to 3D, and its implementation on GPU will be described. In section IV, the proposed GPU branchless DD model will be evaluated for both speed and accuracy. In section V, some related issues will be discussed, and the conclusions will be made.

2.2 Related Works and Background

2.2.1 Different P/BP Models

P/BP models are different in their trade-offs between the computational speed and modeling accuracy. Ray-driven P/BP works by tracing rays through the image. The contribution from a voxel to a ray can be based on the intersection length [80], [101]–[103], or be interpolated based on the distance from the voxel to nearby rays [102], [104]. The ray-driven models usually work
well for forward projection, but they tend to introduce Moiré pattern artifacts in backprojection [98], [99]. Also, ray-driven backprojection algorithms are not straightforward to parallelize. One technique to parallelize the ray-driven backprojection is the boundary box technique [105], where the prospectively projected voxel forms a convex shadow on the detector. The backprojection value from one view is the sum of detector cells inside the convex shadow respectively reweighted by intersection lengths between the voxel and x-ray paths. This method requires the traversal of detector cells inside the shadow which is time-consuming. Furthermore, the ray-driven methods also perform in a non-sequential memory access pattern, and the out-of-order memory writing makes it difficult to be highly parallelized directly.

Pixel-driven P/BP models, on the other hand, work well for backprojection but are less efficient for forward projection. Pixel-driven backprojection is suitable for hardware implementation with specialized circuitry [106] and is widely used for FBP algorithms when the detector array is regular [101], [102], [107]. However, pixel-driven forward projection introduces Moiré pattern artifacts [98], [99], [103]. They are also difficult to parallelize because they access memory in a non-sequential pattern. Although the Moiré pattern artifacts in pixel-driven forward projection can be prevented with sophisticated weighting schemes [108], the increased computational complexity makes it seldom applied. It was also proposed to store the projection weightings to reduce its high computational complexity [109], [110], but it is impractical to be applied in CT imaging for its immense data size. For higher computational efficiency, other pixel- and ray-driven P/BP models were also proposed by simplifying the estimation of the weighting factors [107], [111], [112]. The high-frequency artifacts were also reduced by designing non-standard rectangular pixel shapes [102], [113]–[115].

More accurate P/BP models have been proposed. For example, the area integral model (AIM) [116] or volume integral model (VIM) calculate the projection weighting factor by considering the intersection area or volume between the x-ray path and the pixel/voxel. The Sutherland-
Hodgman clipping algorithm is one common algorithm in computer graphics to calculate the intersection volume [117]. The clipped convex polyhedron will be decomposed into a number of tetrahedrons to calculate the volume size [118]. Both the AIM and VIM describe the intersection weighting factors in high accuracy and ensure matched forward and back operation. However, the difficulty is their high computational complexity. The AIM and VIM are difficult to be implemented and optimized for high computational performance.

It is desirable to develop P/BP models with low computational complexity and high accuracy. Two representative models are the distance driven (DD) model [98], [99] and separable footprint (SF) model [119]. Both DD and SF models can more accurately describe the contribution of a voxel to a detector cell and provide matched forward and back operators. The original DD implementation [99] also offers a highly continuous memory access pattern and low arithmetic cost. However, the thickness of the voxel is ignored. While the SF model takes the thickness of the voxel into consideration, the thickness profile of the voxel along the transverse direction is modeled as a trapezoid. The shadow intensity profile along Z axis can be viewed as a rectangle or a trapezoid depending on the cone angle, but its calculation is more complicated compared with the DD model.

### 2.2.2 GPU Acceleration for CT Reconstruction

The design of parallel algorithms depends on computing architectures. PC clusters [27] are very expensive and takes large space, while FPGAs are difficult to programming or upgrade[30], [31]. In contrast, the GPU is more flexible and receives more attention. It is initially designed to accelerate the rendering of images and video streams outputting on a display device. It features a large number of computing cores and high bandwidth memory bus which are suitable to execute highly data-parallel and arithmetic-intensive algorithms. At the very beginning, the GPU application programming interfaces (APIs) (e.g. DirectX, OpenGL, HLSL and CG API) are only intended for computer graphics applications. In order to implement general purpose algorithms on
GPU, programmers need to acquire the knowledge of computer graphics and represent their algorithms in graphical operation semantics. More recently, to reduce the coding difficulty, generic APIs (i.e. Compute Unified Device Architecture (CUDA)[35], Open Computing Language (OpenCL)[37]) were provided. As a result, researchers and programmers only need to focus on the algorithm design instead of considering how to map their algorithms to graphic operation semantics. In this paper, we choose CUDA to implement the branchless DD P/BP model. The CUDA [35], [40], [41] inherits C/C++ as an extension and is rapidly exploited in several application fields including medical imaging[120], [121].

The GPU based CT reconstruction was first proposed in [122] to accelerate the filtered backprojection algorithm. High performance backprojection algorithms on GPU and other devices have been explored [120], [123], [124]. Rohkol et al.[125] developed the RabbitCT platform providing a benchmark to evaluate the performance of GPU-based analytical reconstruction algorithms. Muller and his partners demonstrated the GPU-accelerated CT reconstruction and achieved high speedups with respect to their CPU counterparts [126], [127]. They accelerated the CT reconstruction on GPU by using the RGBA channels of 2D texture operations [127] or with shading language for backprojection in FDK algorithms[128], [129]. They also investigated the efficiency of ordered subset reconstruction algorithm using conjugate gradient method in GPU[130]. CUDA-based GPU acceleration for CT reconstruction was reported in [120]. Iterative reconstruction methods based on solving the optimization problem in CUDA was reported in [131]. The multiple GPUs acceleration in CUDA for IRAs were proposed in [132], where the task was simply divided into several sub-problems and distributed to multiple GPUs. An abundant of research and literature have been published focusing on the GPU-acceleration of image reconstruction algorithms.
2.3 Methods

2.3.1 Cone-Beam Geometry

The cone-beam geometry with a circular or helical scanning trajectory is applied. In Figure 12, a right-hand Cartesian coordinate system with an arc detector is assumed. The x-ray source and the detector rotate simultaneously around the origin of the system in plane $z = 0$. The distance from the source to the rotation center is $D_{so}$. The x-ray source position $\vec{p}_s$ can be calculated as

$$\vec{p}_s = \begin{pmatrix} -D_{so} \sin \theta \\ D_{so} \cos \theta \\ \theta h \end{pmatrix}.$$  \hfill (18)

The rotation angle $\theta$ is defined counter-clockwise from the direction of $y$-axis and $h$ is the helical pitch. When $h = 0$, it will degrade to the well-known circular scanning trajectory. $(\beta, t)$ denotes the local coordinate on an arc-detector, where $\beta$ is the in-plane fan angle and $t$ is the local $Z$ coordinate (See Figure 12). The coordinate of point $\vec{p}_1$ on the detector can be calculated from

$$\vec{p}_1 = \begin{pmatrix} D_{sd} \sin(\beta + \theta) - D_{so} \sin \theta \\ -D_{sd} \cos(\beta + \theta) + D_{so} \cos \theta \\ t + \theta h \end{pmatrix}.$$  \hfill (19)
where $D_{sd}$ is the source to the detector distance. The x-ray path vector from $\vec{p}_s$ to $\vec{p}_1$ can be normalized as

$$
\vec{e} = \frac{\vec{p}_1 - \vec{p}_s}{\|\vec{p}_1 - \vec{p}_s\|} = \begin{pmatrix}
\cos \varphi \sin (\beta + \theta) \\
-\cos \varphi \cos (\beta + \theta) \\
\sin \varphi
\end{pmatrix},
$$

(20)

where $\varphi$ is the polar angle calculated from

$$
\varphi = \arcsin \left( \frac{t}{\sqrt{D_{sd}^2 + t^2}} \right). 
$$

(21)

Let the function $f(\vec{r})$ be the linear attenuation coefficients to be reconstructed, where $\vec{r} = (x, y, z)$. The line integral of the object is given by

$$
p(\beta, t, \theta) = \int_{L(\beta, t, \theta)} f(\vec{r})dl,
$$

(22)

where $L(\beta, t, \theta)$ represents the x-ray path determined by parameters $\beta, t, \theta$ as well as constants $D_{so}$ and $D_{sd}$.

$$
L(\beta, t, \theta) = \left\{ \vec{p}_s + \vec{s} \vec{e} : s \in [0, L_s] \right\}
$$

(23)

On the one hand, given a rotation angle $\theta$ and a point $\vec{x} = (x, y, z)$ between the source and the detector, its projection position $\vec{p}_1 (\beta, t)$ on the detector can be calculated from

$$
\beta(\theta, x, y) = \arctan \left( \frac{x \cos \theta + y \sin \theta}{D_{so} + x \sin \theta - y \cos \theta} \right),
$$

(24)

and

$$
t(\theta, x, y, z) = \frac{z \cdot D_{sd}}{\sqrt{(D_{so} + x \cos \theta - y \sin \theta)^2 + (x \cos \theta + y \sin \theta)^2}}.
$$

(25)
We assume that the sizes of the detector cells are the same with no gap. The fan angle of each detector cell is $\Delta \beta$ and the height of the detector cell is $\Delta t$. Given the center index of the detector $(i_{c\beta}, i_{ct})$, the indices of the intersection point $\vec{p}_i(\beta, t)$ are

$$i_\beta = \frac{\beta}{\Delta \beta} + i_{c\beta},$$  \hspace{1cm} (26)

$$i_t = \frac{t}{\Delta t} + i_{ct}.$$  \hspace{1cm} (26)

On the other hand, given the source position $\vec{p}_s$, an x-ray path vector $\vec{e}$ and $x$ coordinate $x_v$ on this x-ray path, the corresponding $y$ and $z$ coordinates can be calculated from

$$y_v = D_{so} \cos \theta - \cot(\beta + \theta)(x_v + D_{so} \sin \theta),$$  \hspace{1cm} (27)

$$z_v = \frac{(x_v + D_{so} \sin \theta) \cdot \tan \varphi}{\sin(\beta + \theta)}.$$  \hspace{1cm} (27)

Similarly, given the $y$ coordinate $y_v$ on this x-ray path, $x$ and $z$ can be calculated from

$$x_v = -D_{so} \sin \theta + \tan(\beta + \theta)(-y_v + D_{so} \cos \theta),$$  \hspace{1cm} (28)

$$z_v = \frac{(-y_v + D_{so} \cos \theta) \cdot \tan \varphi}{\cos(\beta + \theta)}.$$  \hspace{1cm} (28)

With the center index of the object $(x_b, y_b, z_b)$ and the voxel size $(\Delta x, \Delta y, \Delta z)$, the index of the point $\vec{v} = (x_v, y_v, z_v)$ can be calculated by

$$i_x = \frac{x_v - x_b}{\Delta x} + N_x / 2,$$

$$i_y = \frac{y_v - y_b}{\Delta y} + N_y / 2,$$

$$i_z = \frac{z_v - z_b}{\Delta z} + N_z / 2,$$  \hspace{1cm} (29)

where $N_x, N_y, N_z$ are number of image pixels along $X, Y, Z$ directions respectively.
2.3.2 Distance Driven Model

Figure 13. Illustration of DD models. (a) DD interpolation model; (b) 2D DD interpolation and (c) 3D brute force DD interpolation.

1D DD MODEL

In Figure 13(a), an object is modeled as a piecewise constant function \( f(x) \). Its values are \( f_i \) and \( f_{i+1} \) over intervals \([x_i, x_{i+1})\) and \([x_{i+1}, x_{i+2})\), respectively. For a detector cell at location \([y_j, y_{j+1})\), the detector measurement is the integral

\[
p_j = \frac{1}{y_{j+1} - y_j} \int_{y_j}^{y_{j+1}} f(x) dx.
\]  

(30)

Because the source signal \( f(x) \) is piecewise constant, the DD model carries out this integration by calculating the extent of overlap [99]

\[
p_j = \frac{x_{i+1} - y_j}{y_{j+1} - y_j} f_i + \frac{y_{j+1} - x_{i+1}}{y_{j+1} - y_j} f_{i+1},
\]  

(31)

assuming \( y_j \in [x_i, x_{i+1}) \) and \( y_{j+1} \in [x_{i+1}, x_{i+2}) \).

2D DD MODEL

The calculation of fan-beam DD projection on one detector cell can be divided into many 1D DD interpolation sub-problems as shown in Figure 13(b). A common axis has to be determined first. Usually, it is parallel to image \( x \) or \( y \)-axis according to the projection angle as described in [99]. The center line of a row of image pixels is represented by the dashed horizontal line. The
intersections of the dashed line with left and right boundaries of each pixel in the row are mapped onto the common axis at locations $x_i, i \in \{1, 2, \ldots, N + 1\}$ where $N$ is the number of pixels in the row. For any detector cell $j$, its left and right boundaries are mapped to the common axis at locations $y_j$ and $y_{j+1}$. The projection $p_j$ is the accumulation of the 1D DD interpolations from all rows of the image reweighted by the slope of the x-ray path and the pixel size.

**3D DD Model**

In the three-dimensional (3D) cone-beam geometry, a common plane is defined first according to the projection angle. Usually, the $xoz$ plane or $yoz$ plane is selected. Then the detector cell and voxel boundaries are mapped to the common plane. The overlap areas are calculated as the product of overlap lengths. The coefficient of the system matrix is the overlap area weighted by voxel size, the ray slope, and normalized by the detector area. The projection of each $xoz$ or $yoz$ image plane at detector $i$ is defined as an extension of equation (30) to a 2D integral

$$p_{mn} = \frac{1}{(u_{m+1}-u_m)(v_{n+1}-v_n)} \int_{u_m \leq u < u_{m+1}} \int_{v_n \leq v < v_{n+1}} f(x, y) dxdy,$$

where $u, v$ are coordinates of detector cell boundaries mapped to the common plane. The 2D integral can be implemented by traversing all the related pixels inside the red rectangle (Figure 13(c)) defined from four intersection points by connecting the source and the center of the detector edges. When the curved detector is applied, detector cell boundary positions become non-uniform when mapped to the common axis or plane. The original DD algorithms has an inner loop that goes through all detector cell/voxel boundaries, and the calculation has to be adjusted depending on whether the next boundary is a voxel or detector cell boundary, hence an if-else branch cannot be totally avoided. Such branch behavior is detrimental to computational efficiency on GPUs because of the divergence of parallel execution paths.
2.3.3 Branchless Model

**BRANCHLESS DD MODEL IN 1D**

A branchless DD algorithm was previously proposed as a variant of implementation of the DD model in which the inner loop is essentially branchless, making it highly parallelizable [100]. In branchless DD, (30) is reformulated as the difference of two indefinite integrals as

\[ p_j = \frac{1}{y_{j+1} - y_j} \left( F(y_{j+1}) - F(y_j) \right), \tag{33} \]

where

\[ F(t) = \int_{-\infty}^{t} f(x)dx + C, \tag{34} \]

is the integral of \( f(x) \) and \( C \) is an arbitrary constant. Because \( f(x) \) represents a spatial restricted image, it is integrable from minus infinity to infinity. \( C \) has no effect on the final result but it is proposed in [100] that the DC component could be subtracted to reduce the dynamic range of \( F(x) \). The advantage of this formulation is that since \( f(x) \) is piecewise constant, \( F(x) \) is piecewise linear, hence \( F(y_j), x_i \leq y_j < x_{i+1} \) can be very efficiently evaluated based on linear interpolation between \( F(x_i) \) and \( F(x_{i+1}) \) by dedicated texture interpolation hardware on GPUs.

More specifically, the piecewise constant function \( f(x) \) can be expressed in a summation form

\[ f(x) = \sum_{i=1}^{l} f_i \cdot \Pi \left( \frac{x - (x_i + x_{i+1})/2}{x_{i+1} - x_i} \right), \tag{35} \]

where \( l \) is the number of pixels, and \( \Pi(\cdot) \) is the rectangular function defined as

\[ \Pi(x) = \begin{cases} 1 & -0.5 \leq x < 0.5 \\ 0 & otherwise \end{cases}. \tag{36} \]

By substituting (35) into (34) and exchanging the order of integration and summation, we have
\[ F(t) = \int_{-\infty}^{t} \sum_{i=1}^{l} f_i \cdot \Pi \left( \frac{x - (x_i + x_{i+1})/2}{x_{i+1} - x_i} \right) dx + C \]

\[ = \sum_{i=1}^{l} f_i \int_{-\infty}^{t} \Pi \left( \frac{x - (x_i + x_{i+1})/2}{x_{i+1} - x_i} \right) dx + C \]

\[ = \sum_{i=1}^{l} f_i \cdot (x_{i+1} - x_i) \cdot \int_{-\infty}^{x_{i+1} - x_i} \Pi(x) dx + C \]

\[ = \sum_{i=1}^{l} f_i \cdot (x_{i+1} - x_i) \cdot R \left( \frac{t - (x_{i+1} + x_i)/2}{x_{i+1} - x_i} \right) + C \]

(37)

where \( R(\cdot) \) is a ramp function of unit height and width defined as

\[ R(x) = \begin{cases} 
0 & x < -0.5 \\
 x + 0.5 & -0.5 \leq x < 0.5 \\
1 & x \geq 0.5 
\end{cases} \]  \hspace{1cm} (38)

\( F(x_i) \) can be computed recursively by

\[ F(x_{i+1}) = \sum_{i=1}^{l} f_i \cdot (x_{i+1} - x_i) \cdot R \left( \frac{x_{i+1} - (x_{i+1} + x_i)/2}{x_{i+1} - x_i} \right) + C \]

\[ = \sum_{i=1}^{l} f_i \cdot (x_{i+1} - x_i) + C \]

\[ = \begin{cases} 
 F(x_i) + f_i \cdot (x_{i+1} - x_i) & i \geq 1 \\
 C & i = 0 
\end{cases} \]  \hspace{1cm} (39)

Evaluating the function at detector position \( y_j \) with \( x_i \leq y_j < x_{i+1} \) yields

\[ F(y_j) = F(x_i) + f_i \cdot (y_j - x_i) \]

\[ = \frac{x_{i+1} - y_j}{x_{i+1} - x_i} F(x_i) + \frac{y_j - x_i}{x_{i+1} - x_i} F(x_{i+1}) \]. \hspace{1cm} (40)

Hence \( F(x) \) is a piecewise linear function which can be calculated by linear interpolation between two nearest knots. On GPU, the linear interpolation can be performed by texture mapping hardware very efficiently.

Overall, calculation of equation (33) is a branchless operation that can be performed by three steps:

1. Integration, which computes \( F(x_i) = \sum_{k=1}^{i} f_k + C \) at knot positions \( x_i \).
(2) Interpolation, which computes $F(y_j)$ through linear interpolation between $F(x_i)$ and $F(x_{i+1})$ where $x_i$ and $x_{i+1}$ are two knots positions and $x_i \leq y_i < x_{i+1}$.

(3) Differentiation which computes $p_j = \frac{(F(y_{j+1}) - F(y_j))}{y_{j+1} - y_j}$.

Figure 14. Illustration for computing the integral over a rectangle region using the integral image. (a) is the image $f(x_i, y_j)$ with 4 vertexe. (b) is the corresponding integral image $F(x_i, y_j)$.

**BRANCHLESS DD MODEL IN 3D**

Similar to (33), the $p_{mn}$ in 3D can be calculated as the difference of four indefinite integrals

$$p_{mn} = \frac{F(u_{m+1}, v_{n+1}) + F(u_m, v_n) - F(u_m, v_{n+1}) - F(u_{m+1}, v_n)}{(u_{m+1} - u_m)(v_{n+1} - v_n)} \quad (41)$$

where

$$F(x, y) = \int_{-\infty}^{x} \int_{-\infty}^{y} f(x', y') dx' dy' \quad (42)$$

is an integral image. Since $f(x, y)$ is piece-wise constant, $F(x, y)$ is piece-wise linear. Based on (41), 3D Branchless DD can also be performed in three steps:

1. Integration. The integral image $F(x, y)$ is evaluated on the voxel grid by summation

$$F(x_i, y_j) = \sum_{x \leq x_i \atop y \leq y_j} f(x', y'). \quad (43)$$

which is the sum of all the pixels from above and left of $(x_i, y_j)$. The integral image can be efficiently calculated in a recursive way.
\[
F(x_i, y_j) = f(x_i, y_j) + F(x_{i-1}, y_j) + F(x_i, y_{j-1}) - F(x_{i-1}, y_{j-1}).
\]

(2) Interpolation. The values of the integral image is evaluated at \(F(u_{m+1}, v_{n+1})\), \(F(u_m, v_n)\), \(F(u_m, v_{n+1})\) and \(F(u_{m+1}, v_n)\) through 2D bilinear interpolation, which are the values of the integral image at the four corners of the detector cell when mapped to the common plane. This bilinear interpolation can be carried out by GPU hardware and avoid branch divergence.

(3) Differentiation. Once the integral image is calculated, evaluating the sum over a horizontal rectangular area in the image does not require to traversal all pixels inside this rectangle but in a constant time from \(F(u_{m+1}, v_{n+1})\), \(F(u_m, v_n)\), \(F(u_m, v_{n+1})\), and \(F(u_{m+1}, v_n)\) on the vertexes of the rectangle inside the integral image. The integral over a rectangular area as illustrated in Figure 14 can be calculated as

\[
\sum_{(x, y) \in ABCD} f(x, y) = F(A) + F(C) - F(B) - F(D).
\]

2.3.4 CUDA Implementation of Branchless Model

*Figure 15. Illustration of branchless DD projection. (a) is branchless DD projection procedure and (b) is one slice of the integral image set as in (a). The rectangle area is determined by four red intersection points.*
We implement the branchless DD projection in CUDA by the following steps. First, two integral images sets are generated for each slide of the image volume \( F_x^{(i)}, i \in \{1, ..., N_x\} \) along the x-axis and \( F_y^{(j)}, j \in \{1, ..., N_y\} \) along y-axis, respectively. For a rotation angle \( \theta \), the x-ray source and the detector cell positions can be calculated with (18) and (19), respectively. The selection of one of the two integral image sets is determined by the projection angle \( \theta \). In Figure 15(a), four vectors \( \vec{e}_L, \vec{e}_R, \vec{e}_U, \vec{e}_D \) are calculated by connecting the x-ray source and four middle points of the detector cell edges. The intersection points of four vectors on the common plane are calculated by (27) or (28). The corresponding pixel indices can be calculated from (29). A rectangular area can be defined from four intersection points (marked by red dots in Figure 15(b)). The texels at the vertexes (marked by blue dots in Figure 15(b)) are fetched, and the 2D DD kernel value is calculated according to (45). The 2D DD kernel value will be scaled by the intersection length/height and the x-ray path slope. The projection value is calculated by accumulating all scaled 2D DD values in the integral image set. In our parallel implementation, one thread

\[ \text{Figure 16. Illustration of branchless DD backprojection. (a) the selections of different center planes according to current projection view angle, (b) the corresponding projection plane 1 and plane 2 with respect to different center planes. The texels are fetched at the black dots on the integral images.} \]
calculates one projection value on one detector cell in one angle. To utilize texture cache and hardware based interpolation, the integral image sets are bound to the texture object.

![Figure 17. Software interpolation based branchless DD backprojection. (a) is the original hardware-based interpolation. (b) is the texel positions which are calculated by rounding (blue dots) and fetched, and the texel at red dots are calculated by bilinear interpolation.](image)

The DD backprojection model can be implemented by the following steps. First, the integral image sets of the projections are generated. Then, for any given voxel, the middle plane of the voxel is decided according to $\theta$ (shown in Figure 16(a)). Four intersections on the detector are determined by connecting the x-ray source with four middle points on the edges of the middle plane. A rectangle region can be determined as in Figure 16(b). Finally, the texels are fetched and the 2D DD kernel value is calculated by (45). The backprojection value is the accumulation of all reweighted 2D DD values. In our implementation, one thread also calculates one backprojection value of one voxel. The integral image of the projection data is bound to the texture object.

The proposed branchless DD implementation was also extended to multiple GPUs. GPUs of different models were used. With a different number of cores and clock rates in these GPUs, the tasks need to be unevenly distributed according to their computational power. The computational power of each GPU is estimated by multiplying the number of CUDA cores and the core clocks. Multiple GPUs P/BP are accomplished with OpenMP+CUDA where OpenMP is used to distribute the tasks to GPUs.
2.3.5 Precision of Hardware Interpolation and Calculation of Integral Images

To utilize Eq. (41) for branchless operation, the integral image sets are mapped to the texture memory for hardware based interpolation. However, the texture memory does not represent texel position in IEEE 754 standard but in a 9-bit fixed point value according to the NVIDIA documentation [133]. Therefore, the minimum relative distinguishable position of two texels is 1/512. Although the inaccuracy of using fixed point values to represent interpolation positions may not cause any visual difference in computer graphics applications, the impact of the precision loss in general purpose computing remains an open question. Calculation of the integral image involves adding numbers with very different dynamic range and may result in a precision loss if performed with single-precision floating point data type. Although (41) can be applied to reduce the dynamic range, it cannot completely solve the problem. Both the hardware based texture interpolation and huge dynamic range will cause the inaccuracy. In our implementation, about 0.2% difference can be observed between the GPU and CPU implementations.

2.3.6 Branchless DD Projection based on double-precision interpolation (GPU-DB)

To mitigate the potential precision loss due to GPU hardware fixed-point interpolation and the calculation of the integral image, we also develop a branchless DD implementation based on a software double-precision interpolation (referred as GPU-DB). In this method, after determining the rectangle area as in Figure 15, for every vertex of the rectangle, the texels at its four nearest neighbors are fetched and interpolated in software to calculate the value of that vertex as illustrated in Figure 17. Both the projection and image volumes are stored in double precision floating-point datatype. However, the NVIDIA GPU does not support double-precision floating-point texture, and the double-precision floating-point number is cast as a structure composed of two integral numbers and bound to the texture. In the kernel function, the structure composed with two integrals can be cast and interpolated. This method can avoid the inaccuracy caused by the hardware interpolation and integral image limitations. However, because of 4 times texture
fetching, slow software interpolation and inefficient GPU computing in double-precision floating-point number, the proposed method is much slower than the original branchless DD method. The speed performance can only be comparable with multi-threads CPU implementation.

2.3.7 Z-line Backprojection (GPU-BZ)

![Diagram of Z-line Backprojection](image)

*Figure 18. Illustration of Z-line based branchless DD backprojection.*

In our GPU-BL, both the volume and the projection data are stored in order of priority on the Z-direction. For backprojection, a column of continuously stored voxels shares the same middle plane. Every central plane of the voxel indicates four texture accesses. Adjacent voxels shares edges resulting in redundant texture accesses as illustrated in *Figure 18*. For example, in the conventional branchless DD, to calculate the backprojection of a column of three voxels, totally 12 texture accesses and 9 +/- operations are required as

\[
\begin{align*}
V_1 &= p(x_3) + p(x_2) - p(x_1) - p(x_4), \\
V_2 &= p(x_5) + p(x_4) - p(x_3) - p(x_6), \\
V_3 &= p(x_7) + p(x_6) - p(x_5) - p(x_8),
\end{align*}
\]

(46)

where 4 texture accesses are redundant in total.

In Z-line backprojection, the calculation of (46) is decomposed into two steps. The first step is to fetch the texels and calculate the differences along transverse plane...
\[ G_1 = p(x_2) - p(x_1), \]
\[ G_2 = p(x_4) - p(x_3), \]
\[ G_3 = p(x_6) - p(x_5), \]
\[ G_4 = p(x_8) - p(x_7). \]

(47)

The second step is to calculate results by subtracting adjacent differences along \( Z \) direction

\[ V_1 = G_1 - G_2, \]
\[ V_2 = G_2 - G_3, \]
\[ V_3 = G_3 - G_4. \]

(48)

Therefore, the total texture accesses is reduced to 8 times and the +/- operations are reduced to 7 times. This implementation strategy in the backprojection is referred to as GPU-BZ.

2.4 Results

2.4.1 Experimental Configuration

The GPUs used in our experiments were GM200 configured in the NVIDIA GeForce Titan X; GK104 configured in the NVIDIA Tesla K10 and the NVIDIA GeForce GTX 670. For GeForce Titan X, the GM200 contains 3072 cores with the core clock 1.0GHz, and the device memory is 12GB with a 336.5 GB/sec bandwidth. While for Tesla K10, two GK104 GPUs are configured. Each GK104 contains 1536 CUDA cores with the core clock 745MHz, and the device memory is 4GB with a 320GB/sec bandwidth. For GTX 670, the GK104 GPU contains 1344 CUDA cores with the core clock 915MHz, and the device memory is 2GB with a 192GB/s bandwidth.

Two Intel Xeon CPUs are configured and each CPU contains 8 physical cores (16 logical cores with hyper-threading) with 3.1GHz core clock. The GPU-BL is implemented with CUDA 7.5 runtime API. The CPU reference DD is implemented in ANSI C routine. POSIX threads are used to parallelize the CPU DD. To evaluate the performance of the GPU-BL, numerical simulations and image reconstructions from clinical applications are performed.

2.4.2 Forward and Back-projector as Single Modules

The speed of GPU-BL was compared with several DD implementations including single-thread CPU DD (referred as CPU-1), 8-thread CPU DD (referred as CPU-8), 32-thread CPU DD
(referred as CPU-32), GPU-DB (for projection) and GPU-BZ (for backprojection) The comparison was based on the GE CT750 HD scanner. The configuration of this geometry is summarized as in Table IV.

Figure 19. Speedup performance with respect to view number. (a) and (b) are the projection and backprojection computational costs with respect to different view numbers. (c) and (d) are the corresponding speedups compared to CPU-32.

**Speedup Ratios**

The P/BP speedup ratios of different DD implementation are summarized in Table V. The speedsups are calculated with respect to CPU-1. Both projection and backprojection achieved about 8-fold accelerations in CPU-8 which agrees well with the number of threads used. The hyper-threading technique of the Intel Xeon CPU provided slightly greater acceleration than 8-fold because the number of physical cores was larger than the number of threads. However, this improvement was only 5% to 15%. Therefore, the 32-threads implementation only achieved about 17-fold acceleration with 16 physical cores. The GPU-BL projection achieved 137-fold acceleration while the GPU-BL backprojection achieved almost 188-fold accelerations. We can also observe that the GPU-BL forward projection was slower than GPU-BL back-projection. This
was because two integral image sets were calculated before projection and only one integral image set was calculated before backprojection.

![Graphs showing projection and backprojection computational costs and corresponding speedups.](image)

*Figure 20. Speedup performance with respect to image size along the transverse plane. (a) and (b) are the projection and backprojection computational costs with respect to different image size along in-plane direction. (c) and (d) are the corresponding speedups compared to CPU-32.*

The GPU-DB projection performed a little worse than CPU-32. As what we have mentioned, the GPU-DB required 4 times more texture memory accesses compared with the GPU-BL and the texel position calculation was achieved by software interpolation instead of the hardware interpolation. Meanwhile, the gaming-oriented GeForce Titan X is not optimized for double-precision floating-point arithmetic. The GPU-BZ did not outperform the GPU-BL because the configuration of the threads block was kept the same as in GPU-BL. With the same threads block configuration, GPU-BZ occupied too much shared memory to store the intermediate results $G_k (k = 0, \ldots, N_2)$ and it caused data latency. To avoid this situation, the number of threads in one block has to be reduced. However, if the number of threads in one block is too small, the warp occupancy will be low resulting in a large portion of cores idle in the streaming multiprocessor (SM). In fact, it is very difficult to balance the SM occupancy and data latency.

*Table IV. The Configuration of GE CT750 HD Geometry*
<table>
<thead>
<tr>
<th>Parameters</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source-to-iso-center distance</td>
<td>541mm</td>
</tr>
<tr>
<td>Source-to-detector distance</td>
<td>949mm</td>
</tr>
<tr>
<td>In-plane detector cell size</td>
<td>1.0239mm</td>
</tr>
<tr>
<td>Cross-plane detector cell size</td>
<td>1.0963mm</td>
</tr>
<tr>
<td>Number of detector columns</td>
<td>888</td>
</tr>
<tr>
<td>Number of detector rows</td>
<td>64</td>
</tr>
<tr>
<td>Reconstruction FOV</td>
<td>$500 \times 500mm^2$</td>
</tr>
<tr>
<td>Detector offset</td>
<td>$(-1.28,0)mm$</td>
</tr>
</tbody>
</table>

We further analyzed the kernel functions of GPU-BL by the CUDA Visual Profiler version 7.5. With the CT geometry described in Table IV, the projection kernel took 1.016 seconds while the backprojection kernel took 0.807 seconds. The SMs in both projection and backprojection are fully utilized. The performance of P/BP kernel was mainly limited by the memory bandwidth. Actually, except for the kernel functions in P/BP, generating integral image sets also occupied a significant portion of the computational time. In our implementation, the integral images were first generated along the horizontal direction, and then along the vertical direction slice by slice. It approximately occupied 0.13 seconds and 0.15 seconds for the projection and backprojection, respectively. About 0.4 seconds was required for the P/BP to transfer the data from/to the host memory.
We listed the corresponding Gigabytes Updates Per Second (GUPS) that is independent of the problem size in the first order approximation. The giga updates is the total number of ray or voxel updates divided by $1024^3$. According to the GUPS calculation formula in [134],

$$GUPS = \frac{512^3 \times 360}{1024^3} / \text{time},$$

(49)

where the detector cell number is set to $512 \times 512$, the image size is set to $512^3$ and the number of views is 360. In this configuration, the GUPS of several competitive methods are listed in Table VI. The GUPS results are consistent with the speedup performance in Table V.

Table V. Running time of projection and backprojection (unit: s) in GE CT750 HD geometry.

<table>
<thead>
<tr>
<th></th>
<th>CPU-1</th>
<th>CPU-8</th>
<th>CPU-32</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Projection</strong></td>
<td>195.59 (1.00x)</td>
<td>24.10 (8.12x)</td>
<td>11.48 (17.04x)</td>
</tr>
<tr>
<td><strong>Backprojection</strong></td>
<td>195.44 (1.00x)</td>
<td>24.89 (7.85x)</td>
<td>11.48 (17.02x)</td>
</tr>
<tr>
<td><strong>GPU-BL</strong></td>
<td>1.42 (137.74x)</td>
<td>13.46 (14.53x)</td>
<td>N/A</td>
</tr>
<tr>
<td><strong>GPU-DB</strong></td>
<td>1.04 (187.92x)</td>
<td>N/A</td>
<td>1.54 (126.91x)</td>
</tr>
</tbody>
</table>

Table VI. GUPS of Projection and Backprojection.

<table>
<thead>
<tr>
<th></th>
<th>CPU-32</th>
<th>GPU-BL</th>
<th>GPU-DB</th>
<th>GPU-BZ</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Projection</strong></td>
<td>1.57</td>
<td>17.65</td>
<td>1.34</td>
<td>N/A</td>
</tr>
<tr>
<td><strong>Backprojection</strong></td>
<td>1.49</td>
<td>17.93</td>
<td>N/A</td>
<td>11.58</td>
</tr>
</tbody>
</table>

**Speed with Respect to Number of Views**

Fixing the image size and the detector size, the speedup performances of GPU-BL was also evaluated by increasing the number of views from 100 to 11,000. In our implementation, we did not use the asynchronous technique to hide the data transfer latency. The CPU-1 and CPU-8 were not tested here because of their lower speeds. The P/BP time and speedup ratios with respect to CPU-32 are shown in Figure 19. We can see the time for projection and backprojection almost linearly increased with respect to the number of views. When the number of views was small, the speedup was low because of the overhead for data transfer between CPU and GPU. When the
number of views is increased, the relative overhead of data transfer became small compared to the computation time on GPU, and the benefit of GPU acceleration dominates. The speedup factor was up to 10X for projection and 11X for backprojection. With more views, the speedup gradually increased and reached a plateau.

Figure 21. Speedup performance with respect to image and detector sizes along Z-direction. (a) and (b) are the projection and backprojection computational costs with respect to image/detector size along the cross-plane direction. (c) and (d) are the corresponding speedups compared to CPU-32.

Speed with Respect to Image Size In-Plane

The speeds of different P/BP implementations were evaluated with respect to different image sizes in-plane. With the numbers of views and detector columns fixed, we increased the image size from $256^2$ to $1408^2$ with a step size of 128 pixels along both $X$ and $Y$ directions. The CPU-32 was also used as the reference to calculate the speedups. The P/BP time and speedups are shown in Figure 20. We can see that the computational time of CPU-32 in both projection and backprojection increased approximately as a quadratic function which was consistent with the computational scale. However, the computational time of projection with GPU-BL and GPU-DB
increased approximately linearly. In branchless DD projection, both the size and the number of integral images increases linearly with in-plane image size, thus the computational complexity of the integration step is quadratic with respect to the in-plane image size. However, the complexity of both the interpolation and differentiation steps is constant for each integral image slice and the computational time of these two steps scales linearly with the number of image slices. Since the computation time of branchless DD projection is dominated by the interpolation and differentiation steps, its overall complexity is approximately linear with respect to in-plane image size within practical configurations. For the above reasons, although the GPU-DB was slower compared to the CPU-32 when the image volume was small, the computational complexity of GPU-DB was lower than CPU-32. From the speedup charts, with the increase of image size, we can see that the projection speedup of GPU-BL is greater for larger in-plane image sizes. However, we can also observe that the computational time of backprojection with GPU-BL and GPU-BZ increased approximately quadratically. This is because one thread calculates the backprojection value of one voxel in GPU-BL, or one column of voxels along the Z direction in our GPU-BZ. When the projection data size is fixed and the image size increases quadratically, the number of threads will also increase quadratically. Therefore, the computational times of both GPU-BL and GPU-BZ increase quadratically.

**Speed with Respect to Image and Detector along Z direction**

The speed of different P/BP implementations were also evaluated with respect to image and detector sizes along the Z direction. The image and detector sizes in Z were increased simultaneously from 64 to 480 with a step size of 32. We also applied the CPU-32 as the baseline to calculate the speedup ratio. As shown in Figure 21, the computational time of projection and backprojection both linearly increased with respect to the numbers of pixels and detector cells along the Z direction. This linear increase in computation time was consistent among all DD implementations. The sizes of image and detector cell along Z-direction did not affect the speedup ratio.
Table VII. Computational costs (in seconds) of various components on multiple GPUs. The image volume is 512 × 512 × 64 with (a) 984, (b) 1968 and (c) 3936 views of size 888 × 64 in HD geometry.

<table>
<thead>
<tr>
<th>Number of GPUs</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Projection</strong></td>
<td>1.478</td>
<td>1.107</td>
<td>1.102</td>
<td>1.101</td>
</tr>
<tr>
<td><strong>Projection Kernel</strong></td>
<td>1.009</td>
<td>0.703</td>
<td>0.579</td>
<td>0.457</td>
</tr>
<tr>
<td><strong>Backprojection</strong></td>
<td>1.041</td>
<td>0.953</td>
<td>0.944</td>
<td>0.886</td>
</tr>
<tr>
<td><strong>Backprojection Kernel</strong></td>
<td>0.618</td>
<td>0.454</td>
<td>0.346</td>
<td>0.272</td>
</tr>
</tbody>
</table>

(A) 984 Views

<table>
<thead>
<tr>
<th>Number of GPUs</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Projection</strong></td>
<td>2.550</td>
<td>2.045</td>
<td>1.762</td>
<td>1.581</td>
</tr>
<tr>
<td><strong>Projection Kernel</strong></td>
<td>2.012</td>
<td>1.411</td>
<td>1.184</td>
<td>0.897</td>
</tr>
<tr>
<td><strong>Backprojection</strong></td>
<td>1.933</td>
<td>1.504</td>
<td>1.351</td>
<td>1.261</td>
</tr>
<tr>
<td><strong>Backprojection Kernel</strong></td>
<td>1.260</td>
<td>0.832</td>
<td>0.703</td>
<td>0.552</td>
</tr>
</tbody>
</table>

(B) 1968 Views

<table>
<thead>
<tr>
<th>Number of GPUs</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Projection</strong></td>
<td>4.685</td>
<td>3.513</td>
<td>2.841</td>
<td>2.542</td>
</tr>
<tr>
<td><strong>Projection Kernel</strong></td>
<td>4.029</td>
<td>2.853</td>
<td>2.305</td>
<td>1.791</td>
</tr>
<tr>
<td><strong>Backprojection</strong></td>
<td>3.579</td>
<td>2.754</td>
<td>2.253</td>
<td>1.842</td>
</tr>
<tr>
<td><strong>Backprojection Kernel</strong></td>
<td>2.693</td>
<td>1.746</td>
<td>1.461</td>
<td>1.174</td>
</tr>
</tbody>
</table>

(C) 3936 Views

**Multi-GPU Branchless DD**

The performance of multi-GPU-BL is summarized in Table VII. Multiple-GPUs computation cannot provide much higher speedups compared to single GPU when the number of views was small. The GPUs with weaker computational power had lower bandwidths which made the data transfer more time consuming. Furthermore, the calculation of integral images also occupied considerable computational cost. Although more GPUs were employed, the total cost to compute the integral images did not decrease. When the number of views was large, the P/BP time was reduced significantly.

**2.4.3 Iterative Reconstruction**

**Simulation data**

Noiseless numerical simulations were conducted to investigate the performance of GPU-BL for IRAs. In our numerical simulations, a monochromatic x-ray cone-beam CT with equal-angular
detector was assumed. A modified Shepp-Logan phantom and a FORBILD head phantom were employed, and the image sizes are 512×512×64. A full scan projection dataset was evenly acquired with 2200 views. A ray-tracing model with 4×4 subsamples was applied to simulate the projection. In each view, every 4×4 neighboring detector cells were averaged to form an 888×64 projection. The linear attenuation coefficients of the two phantoms were reconstructed by an OS-SART algorithm with the proposed GPU-BL projector. The final results were compared with the ground truth and the CPU-32 based OS-SART results. The number of ordered subsets was 10. The maximum iteration number was 40.

The reconstructed modified Shepp-Logan from CPU-32 and GPU-BL are shown in Figure 22(a) and Figure 22(b), respectively. The corresponding error images relative to the ground truth are presented in Figure 22(c), and Figure 22(d) in a much narrower display window and the representative profiles along horizontal and vertical directions of the center transverse plane are shown in Figure 22(e) and Figure 22(f). We can see that the reconstruction errors are greater at the top and bottom slices. This is because the top and bottom slices cannot be completely irradiated in circular cone-beam geometry. From Figure 22(e) and Figure 22(f), we can see that the profiles match the ground truth very well except some tiny Gibbs artifact at the edges of the object. This is the property of the OS-SART without any regularization.
Figure 22. Numerical simulation results of the modified Shepp-Logan phantom. (a) are the transverse, sagittal and coronal planes reconstructed by the CPU-32 in a display window [0, 1]. (b) are the counterpart of (a) reconstructed by the GPU-BL. (c) and (d) are the corresponding error of (a) and (b) with respect to ground truth in display window [-0.02 0.02]. (e) and (f) are the horizontal and vertical profiles of the center slice of the images.

The reconstructed FORBILD head images in transverse, sagittal and coronal planes are shown in Figure 23(b) and Figure 23(c) in a display window [1.0, 1.2]. The corresponding reconstruction errors in a narrower window are shown in Figure 23(e) and Figure 23(f). As indicated by the arrows in Figure 23(b) and Figure 23(c), the reconstruction errors can be observed in sagittal and
coronal planes from both GPU-BL and CPU-32. As aforementioned, the mismatch between GPU-BL and CPU-32 is about 0.2% (see Figure 23(d) in a much narrower display window). The computation time per iteration of FORBILD head phantom reconstruction is summarized in Table VIII.

![Figure 23. Numerical simulation results of the FORBILD head phantom. (a) is the ground truth. (b) and (c) are the reconstruction results from GPU-BL and CPU-32, respectively. The display window for on the first row images is [1.0,1.2]. (d) is the differences between (b) and (c) in a display window [-0.005,0.005]. (e) and (f) are the reconstruction errors of (b) and (c) with respect to (a) in a display window [-0.02,0.02].](image)

Table VIII. Speedup performance of the FORBILD head phantom reconstruction in one iteration.

<table>
<thead>
<tr>
<th></th>
<th>Projection</th>
<th>Backprojection</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU-32</td>
<td>24.13</td>
<td>26.63</td>
</tr>
<tr>
<td>GPU-BL</td>
<td>2.75</td>
<td>2.25</td>
</tr>
<tr>
<td></td>
<td>Total 2.78</td>
<td>Kernel 1.92</td>
</tr>
</tbody>
</table>
Real Phantom Reconstruction

We also acquired real phantom data from a GE Discovery CT750 HD system (GE Healthcare, Waukesha, WI) with 888 detector channels and 984 views per rotation. We scanned an oval phantom with a quality-assurance insert and two Teflon rods on both sides. The data were acquired with a 32-row helical scan with a pitch of 31/32 at 120 kVp, 835 mA, and 1-second rotation. Images were reconstructed on a grid of 512×512×64 with a field-of-view of 50 cm, an in-plane pixel size of 0.98 mm², and a slice thickness of 0.625 mm. All reconstructions are used $q$-GGMRF regularization[135]. All reconstructions were based on a preconditioned conjugate gradient algorithm initialized with standard FBP images[136]. The reconstruction is presented in Figure 24. The differences between Figure 24(a) and Figure 24(b) are hardly noticeable unless displayed with a much narrower window in Figure 24(c). The RMSE of GPU-BL with respect to CPU-32 was 0.57 HU. The differences between Figure 24(d) and Figure 24(e) are almost imperceptible, although there is the small difference inside the red rectangle under very careful comparison.

Figure 24. Real phantom reconstruction results. (a) and (b) are reconstructed by the CPU-32 and GPU-BL from high dose (835mAs) projections in a display window [800 1200]HU. (c) is the difference image between CPU-32 and GPU-BL for low dose case in a display window [-3, 5]HU.
**Clinical Application**

![Reconstructed results from a clinical data set.](image)

Figure 25. Reconstructed results from a clinical data set. (a) is reconstructed by the FDK algorithm as ground truth. (b) and (c) are reconstructed by the GPU-BL CPU-32, respectively. (d) is the differences between GPU-BL and CPU-32 in a display window [-0.5, 0.5]HU. (e) and (f) are the differences between the ground truth and GPU-BL and CPU-32 in display window [-20, 20]HU, respectively.

Finally, we evaluated the GPU-BL using clinical patient datasets. With the approval of Wake Forest University School of Medicine, a clinical patient dataset with 2,200 views was acquired and reconstructed by the SART algorithm in the same geometry as in *Table IV*. A 512×512×64 volumetric image was reconstructed. We used the Titan X to perform our experiments. The FISTA technique[137] is also applied for acceleration. The FDK reconstruction was also provided as a reference in *Figure 25*(a). The reconstruction results from the GPU-BL and CPU-32 are shown in *Figure 25*(b) and *Figure 25*(c), respectively. The differences between GPU-BL and CPU-32 are shown in *Figure 25*(d) in a narrow display window [-0.5, 0.5] HU. The differences relative to the FDK reconstruction are shown in *Figure 25*(e) and *Figure 25*(f) in a window of [-]
20, 20] HU. One can also see the greater difference at the top and bottom slices of the reconstructed image volume. The SSIM [138] of the center slice of GPU-BL result relative to the FDK reconstruction is 0.9905.

2.5 Discussion and Conclusions

In this paper, we proposed a branchless DD P/BP algorithm for 3D cone beam CT. The proposed algorithm eliminated the irregular branch behavior of the original DD algorithm and made the DD operation highly amenable to massive vectorization of GPUs. For a GE Discovery CT750 HD system, the proposed method achieved 137-fold speedup for projection and 188-fold speedup for backprojection compared with a single thread CPU implementation. Compared with a state-of-the-art 32-thread CPU implementation, the proposed branchless DD P/BP achieved 8-fold acceleration for forward projection and 10-fold acceleration for back projection. Our implementation of branchless DD also fully leveraged the cache mechanism and the hardware interpolation supported by the texture memory in GPUs.

Some loss of precision of the branchless algorithm can be caused by the integration and interpolation steps when implemented with the single-precision data type and hardware based interpolation. However, our reconstruction results from simulation, and real data showed that the precision loss was small and the GPU-based branchless algorithm obtained visually identical images as the CPU reference algorithm.

This study focused on algorithm development, and the GPU implementation was not fully optimized. In particular, the integration step of branchless DD occupied a considerable portion of the computational time. However, highly optimized GPU algorithms [139] exist for computing the integral of 2D images and can be readily incorporated for further acceleration. Our implementation of iterative reconstruction involved repeated data transfer between GPU device memory and CPU host memory. Some standard GPU programming techniques can be used in the
future to reduce or hide this data transfer latency. The overhead of data transfer could also be eliminated by implementing the complete iterative reconstruction in GPUs.
Chapter III. Singular Value Decomposition-based 2D Image Reconstruction for Computed Tomography

Abstract

Singular value decomposition (SVD)-based 2D image reconstruction methods are developed and evaluated for a broad class of inverse problems for which there are no analytical solutions. The proposed methods are fast and accurate for reconstructing images in a non-iterative fashion. The multi-resolution strategy is adopted to reduce the size of the system matrix to reconstruct large images using limited memory capacity. A modified high-contrast Shepp-Logan phantom, a low-contrast FORBILD head phantom, and a physical phantom are employed to evaluate the proposed methods with different system configurations. The results show that the SVD methods can accurately reconstruct images from standard scan and interior scan projections. They outperform other benchmark methods. The general SVD method outperforms the other SVD methods. The truncated SVD and Tikhonov regularized SVD methods accurately reconstruct a region-of-interest (ROI) from an internal scan with a known sub-region inside the ROI. Furthermore, the SVD methods are much faster and more flexible than the benchmark algorithms, especially in the ROI reconstructions in our experiments.

3.1 Introduction

Although computed tomography (CT) has been widely applied for medical diagnosis, its potential radiation hazard cannot be ignored. CT scanning accounted for approximately 75% of the ionizing radiation exposure in 2002 in the United States [140]. Many methods have been proposed to reduce the excessive radiation dose, including three representative methods. The first is to reduce the number of photons emitted from the x-ray source; however, the noisy projections that are acquired degrade the image quality. The second method is to reduce the number of views.
However, the reconstructed images contain severe streak artifacts. The third method is to illuminate a small region of interest (ROI) inside the object [9], which is referred to as an interior scan. However, due to the nonlocal property of the inverse Radon transform, no theoretical exact local reconstruction can be achieved solely from projections that pass through the ROI [141]. Based on prior knowledge, several theoretically exact solutions have been developed, which are referred to as interior tomography [9], [142]–[146]. In the early 1990s, Gel’fand and Graev established a relationship between the Hilbert transform of an image along a line and the corresponding backprojection of the differentiated projections [147][148]. The interior problem was proven to be uniquely solvable if there is a known sub-region inside the ROI [144], [145], which is referred to as knowledge-based interior tomography.

To date, two major reconstruction frameworks have been developed for knowledge-based interior tomography. One is an iterative framework that is based on projection onto convex sets [144]. An $l_1$ norm minimization of a sparse transform is usually applied as the regularization term. Iterative reconstruction algorithms involve multiple time-consuming projection and backprojection procedures, which limits their general applications. The other reconstruction framework is to use the singular value decomposition (SVD) method to reconstruct an image on 1D PI-segments based on the Hilbert transform relationship [149], [150]. Because this approach allows an interior reconstruction without iteration, it is much faster than its iterative counterparts. However, a rebinning procedure is usually required to convert the reconstructed images to Cartesian coordinates. Thus, the spatial resolution is compromised.

The greater capacity of computer memory motivates us to develop SVD-based 2D image reconstruction methods. These methods can perform fast image reconstructions for a broad class of inverse problems that have no analytical solutions. This includes, but is not limited to, few-view global reconstruction, local reconstruction from truncated projections, and irregular
projection geometry. In particular, this paper emphasizes interior tomography as a practical application.

Several SVD-based methods have been proposed for single photon emission computed tomography (SPECT) reconstruction [151]–[155]. Smith et al. [152] utilized a generalized matrix inverse to estimate the source activity distributions from SPECT. The SVD of the system matrix provided considerable insight into the SPECT image reconstruction problem. Tradeoffs between the resolution and error in estimating the source voxel intensities were also discussed in [14]. Zeng and Gullberg [153] applied SVD to analyze and reconstruct images. They concluded that problems might be ill-conditioned with truncated projections and that the reconstruction artifacts are due to the mismatch between the measured data and modeled projections. Wiener filtering and truncated inverse filtering were discussed in the search for optimal restoration with an SVD pseudo inversion method in [154]. Verhoeven [155] compared five CT algorithms for limited data problems, including SVD methods. SVD-based image reconstruction techniques have also been applied in other fields [156], [157]. However, a systematic evaluation of SVD methods for 2D CT image reconstruction has not been reported.

This paper is organized as follows. Section II presents mathematical models of CT imaging and SVD methods. The details of the algorithm implementation are presented in section III, and section IV shows the results. Finally, relevant issues are discussed in section V.

### 3.2 Mathematical Theory

#### 3.2.1 Imaging model and SVD method

In a discrete CT system, both the image and the projection are modeled as vectors. The projection procedure is modeled as

$$ \mathbf{A}\mathbf{f} = \mathbf{p}. $$

(50)
where \( \mathbf{f} \in \mathbb{R}^N \) is the image, \( \mathbf{p} \in \mathbb{R}^M \) is the projection, and the projection procedure is simplified to a linear transform that is represented by a system matrix \( \mathbf{A} \in \mathbb{R}_{+}^{M \times N} \), where \( \mathbb{R} \) is the real number domain, and \( \mathbb{R}_{+} \) represents a set of all nonnegative real numbers. An intuitive way to find the solution to Eq. (50) is

\[
\hat{\mathbf{f}} = \mathbf{A}^\dagger \mathbf{p},
\]  

where \( \mathbf{A}^\dagger \) is the (pseudo) inverse of \( \mathbf{A} \). According to SVD, \( \mathbf{A} \) can be decomposed as

\[
\mathbf{A} = \mathbf{U} \Sigma \mathbf{V}^T \in \mathbb{R}_{+}^{M \times N},
\]

where \( T \) represents the transpose of a matrix, both \( \mathbf{U} \in \mathbb{R}^{M \times M} \) and \( \mathbf{V} \in \mathbb{R}^{N \times N} \) are unitary matrices, and \( \Sigma \in \mathbb{R}^{M \times N} \) is a uniquely determined diagonal matrix, whose diagonal entries \( \sigma_i (i \in \{1, 2, \ldots, \min(M, N)\}) \) are singular values of \( \mathbf{A} \). The (pseudo) inverse of \( \mathbf{A} \) can be calculated as

\[
\mathbf{A}^\dagger = \mathbf{V} \Sigma^\dagger \mathbf{U}^T,
\]

where \( \Sigma^\dagger \) is the (pseudo) inverse of \( \Sigma \), which can be formed by replacing each non-zero diagonal entry \( \sigma_i \) by its reciprocal.

<table>
<thead>
<tr>
<th>Column Index of ( \mathbf{U} ) and ( \mathbf{V} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
<tr>
<td><img src="image1.png" alt="Image" /></td>
</tr>
</tbody>
</table>

Figure 26. Illustration of physical meaning for SVD assuming an interior scan geometry. The top row is the \((1^{st}, 10^{th}, 100^{th}, 200^{th}, 1000^{th}, \text{and } 10000^{th})\) column vector images of matrix \( \mathbf{U} \), and the bottom row are corresponding column vector images of matrix \( \mathbf{V} \).
We can interpret the SVD of $\mathbf{A}$ in terms of eigen-decomposition. The columns of $\mathbf{U}$ are eigenvectors of $\mathbf{A}\mathbf{A}^T$, and the columns of $\mathbf{V}$ are eigenvectors of $\mathbf{A}^T\mathbf{A}$. In the CT reconstruction problem, if the column of $\mathbf{U}$ is backprojected and then projected, we can obtain the same projection at a constant scale. Using interior scan geometry to project a 128×128 image to 128 detector cells with 128 views, several representative columns of $\mathbf{U}$ and $\mathbf{V}$ are depicted as 2D images in Figure 26. The first column of $\mathbf{U}$ is the projection of a constant image, and the first column of $\mathbf{V}$ is the backprojection of a constant projection data set.

3.2.2 Truncated SVD and Tikhonov regularization method for image reconstruction

Small singular values magnify projection noise. Therefore, severe artifacts will appear if the (pseudo) inverse is directly applied for image reconstructions. To suppress noise, we need to find a threshold to truncate small singular values. Given the magnitude ratio of singular values, the truncation position can be calculated according to

$$
\hat{i} = \arg \min_i \left| \sum_{j=1}^i \sigma_j \right| - r,
$$

(54)

where $\bigoplus \sigma$ is the sum of all of the singular values, and $r$ is a target ratio. If $\varepsilon(=\sigma_i)$ is a selected threshold, truncated SVD (TSVD) is usually applied to suppress the noise [158]. The pseudo-inverse of $\Sigma$ is defined with $\varepsilon$ as

$$
\sigma_i^{-1} = \begin{cases} 
1/\sigma_i & \sigma_i > \varepsilon \\
0 & \sigma_i \leq \varepsilon 
\end{cases}
$$

(55)

A more general regularization method can be modeled as [158]

$$
\|\mathbf{A}f - \mathbf{p}\|_2^2 + \xi^2 \|\Phi f\|_2^2,
$$

(56)

where $\xi$ is a parameter that balances the data fidelity and regularization terms, and $\Phi$ represents a transform of $\mathbf{f}$. If $\Phi = \mathbf{I}$, Eq. (56) can be explicitly solved by defining the pseudo inverse of $\Sigma$. 

89
Eq. (57) is the solution of Tikhonov regularization [159], which is referred to as VSVD. Both TSVD and VSVD play similar roles in smoothing reconstructed images.

### 3.2.3 Generalized SVD method for image reconstruction

When \( \Phi \neq I \), generalized SVD (GSVD[160]) can be used to solve Eq. (56). Both \( A \in \mathbb{R}^{M \times N} \) and \( \Phi \in \mathbb{R}^{P \times N} \) are real matrices with \( M \geq N \geq P \) and \( \text{rank}(\Phi) = P \). In GSVD, \( A \) and \( \Phi \) can be decomposed as

\[
A = U \begin{pmatrix} \Lambda & 0 \\ 0 & I_{N-P} \end{pmatrix} B^{-1},
\]

\[
\Phi = V \begin{pmatrix} M & 0 \end{pmatrix} B^{-1}.
\]

\( B \) is an invertible matrix. Both \( \Lambda \) and \( M \) are diagonal matrices,

\[
\Lambda = \text{diag} \left( \lambda_1, \lambda_2, \ldots, \lambda_P \right), \quad 0 \leq \lambda_1 \leq \ldots \leq \lambda_P \leq 1,
\]

\[
M = \text{diag} \left( \mu_1, \mu_2, \ldots, \mu_P \right), \quad 1 \geq \mu_1 \geq \ldots \geq \mu_P \geq 0.
\]

The values \( \lambda_i \) and \( \mu_i \) are normalized so that \( \lambda_i^2 + \mu_i^2 = 1 \). Generalized singular values are defined as

\[
\gamma_i = \frac{\lambda_i}{\mu_i},
\]

in a decreasing manner on the index \( i \). A property of GSVD is

\[
B^T A^T A B = \begin{pmatrix} \Lambda^2 & 0 \\ 0 & I \end{pmatrix},
\]

\[
B^T \Phi^T \Phi B = \begin{pmatrix} M^2 & 0 \\ 0 & 0 \end{pmatrix}.
\]

The objective function Eq. (56) can be written in quadratic form as

\[
\hat{f} = \arg \min_x \left\| \begin{pmatrix} A \\ \xi \Phi \end{pmatrix} f - \begin{pmatrix} p \\ 0 \end{pmatrix} \right\|_2^2
\]

The optimal solution can be determined by setting the first derivative \( f \) with respect to zero,
\[
\left( A^T A + \xi^2 \Phi^T \Phi \right) f = A^T p. \quad (63)
\]

Therefore,

\[
\hat{f} = \left( A^T A + \xi^2 \Phi^T \Phi \right)^{-1} A^T p. \quad (64)
\]

Eq. (61) is applied to calculate \((A^T A + \xi^2 \Phi^T \Phi)^{-1}\) with the following steps

\[
B^T \left( A^T A + \xi^2 \Phi^T \Phi \right) B = B^T A^T A B + \xi^2 B^T \Phi^T \Phi B = \begin{pmatrix} A^2 + \xi^2 M^2 & 0 \\ 0 & 0 \end{pmatrix},
\]

\[
\left( A^T A + \xi^2 \Phi^T \Phi \right) = \left( B^T \right)^{-1} \left( \begin{pmatrix} A^2 + \xi^2 M^2 & 0 \\ 0 & 0 \end{pmatrix} \right) B^{-1}, \quad (65)
\]

\[
\left( A^T A + \xi^2 \Phi^T \Phi \right)^{-1} = \left( B^{-1} \right)^T \left( \begin{pmatrix} A^2 + \xi^2 M^2 & 0 \\ 0 & 0 \end{pmatrix} \right)^{-1} B^{-1} = \left( B^{-1} \right)^T \Sigma^T B^{-1}.
\]

where \(\Sigma^T\) is a diagonal matrix that can easily be calculated. \(B\) is a full rank matrix, and its inverse needs to be calculated only once. In this paper, the Laplacian operator \(\nabla (= \Phi)\) is applied.

### 3.2.4 Interior reconstruction with a known sub-region

Let \(\Omega = \{1, 2, ..., N\}\) be the index set of pixels. If we treat pixels as independent variables, \(A\) can be modified column by column. When a sub-region inside the ROI is known, the pixels can be grouped as

\[
f = \begin{pmatrix} f_K \\ f_{\Omega \setminus K} \end{pmatrix} \in \mathbb{R}^N, \quad (66)
\]

where \(f_K\) and \(f_{\Omega \setminus K}\) represent unknown and known pixels, respectively, and \(K\) is the index set of known pixels. Correspondingly, the system matrix can be divided into two sub-matrices

\[
A = [A_K \ A_{\Omega \setminus K}], \quad (67)
\]

The projection procedure is

\[
Af = [A_K \ A_{\Omega \setminus K}] \begin{pmatrix} f_K \\ f_{\Omega \setminus K} \end{pmatrix} = A_K f_K + A_{\Omega \setminus K} f_{\Omega \setminus K} = p. \quad (68)
\]

We immediately arrive at the following linear matrix equation
\[ A_{\Omega \setminus K} f_{\Omega \setminus K} = p - A_K f_K. \]  

(69)

Only the sub-matrix \( A_{\Omega \setminus K} \) needs to be decomposed with SVD for the image reconstruction

\[ A_{\Omega \setminus K} = U_{\Omega \setminus K} \Sigma_{\Omega \setminus K} V_{\Omega \setminus K}^T \]  

(70)

### 3.2.5 Multi-resolution image reconstruction with SVD methods

#### Figure 27. A schematic diagram for multi-resolution representation of an image and its corresponding system matrix. In (a), the gray part surrounded by the dashed line represents the ROI. The large grids represent coarse pixels while small grids represent fine pixels. (b) is the spy of the combined system matrix \( A_C \) where the blue dots represent non-zero elements, the columns of \( A_L \) corresponds to fine pixels and the columns of \( A_R \) corresponds to coarse pixels.

Reconstructing a high-resolution image by SVD requires extensive memory resources. For example, to reconstruct a 512×512 image, \( A \) in a non-compressed format occupies 512 GB of double floating point data when the number of detector cells and the number of views are both 512. This challenging requirement inspires us to reconstruct the image using a multi-resolution scheme. In interior tomography, the ROI is strictly inside the object support. Therefore, using the multi-resolution strategy to compress the system matrix for ROI reconstruction will save space. We represent the object with fine pixels inside the ROI and with coarser pixels outside the ROI. Let \( A_c \) be an interior scan of a low resolution image and \( A_f \) be the same scan of a high resolution image. The columns of \( A_f \) that correspond to the pixels inside the ROI are extracted to form \( A_L \),
and the columns of \( A_c \) that correspond to the pixels outside the ROI are extracted to form \( A_R \).

The interior scan system matrix of a dual-resolution image is constructed by merging \( A_L \) and \( A_R \)

\[
A_M = [A_L, A_R].
\]  

(71)

The multi-resolution strategy provides a way to compress the system matrix. For example, if a 256×256 image is reconstructed using SVD methods, the size of \( A \) is \( 256^2 \times 256^2 \times 8 \) bytes = 32 GB of double floating point data. Performing SVD requires at least 96 GB of memory. In Eq. (71), the pixel size is 0.176 cm × 0.176 cm inside the ROI and 0.528 cm × 0.528 cm outside the ROI. Therefore, one coarse pixel covers nine fine pixels. A 153×153 fine grid is used to cover the ROI, and every 3×3 pixels outside the ROI is merged into one coarse pixel (see Figure 27(a)). Consequently, 41,616 pixels outside the ROI are reduced to 4,624 pixels. The number of columns of \( A \) is reduced to \( 153 \times 153 + 4,624 = 28,033 \). Let the number of detector cells be 255 and the number of views be 110. The multi-resolution system matrix \( A_M \) is shown in Figure 27(b), where the blue dots represent non-zero elements. More complicated system matrices can be constructed in similar ways. To our knowledge, the SVD-based tomographic reconstruction method has not been thoroughly investigated due to the massive size of a solid CT system matrix [151], [161].

Compared to our previous SVD algorithm on 1D PI-segments [150], we refer to our method as SVD-based 2D tomography. In the following sections, we perform a comprehensive evaluation to demonstrate the merits of the algorithm in terms of the system flexibility, fast computation, and accurate reconstruction.

### 3.3 Algorithm and Computer Implementation

#### 3.3.1 Benchmark Algorithms and System Configuration

In this paper, the system matrices for the SVD reconstruction are generated using the area integral model (AIM) [116]. We evaluated TSVD, VSVD, and GSVD. The performances of the SVD methods were compared to the filtered backprojection algorithm (FBP [141] with appropriate
extrapolation [162], [163] to deal with the data truncation), in which the R-L kernel was applied [164]. The simultaneous algebraic reconstruction technique (SART) [83] and total variation (TV) regularized SART (SART-TV) [165] were also applied. All of the experiments were performed on a high-performance workstation with dual 3.10 GHz Intel Xeon CPUs and 128 GB of memory. SVD of the system matrices was implemented in C with the Intel MKL library, which provides a linear algebra package (LAPACK) that includes the SVD of a matrix. The decomposed results were loaded into Matlab for reconstruction and evaluation. For SART and SART-TV, the projection and backprojection were both implemented by matrix-vector multiplication with $\mathbf{A}$ because matrix-vector multiplication is fully optimized in Matlab.

![Image](image.png)

*Figure 28. Numerical phantoms. (a) is a modified high-contrast Shepp-Logan phantom and (b) is a low-contrast FORBILD head phantom. The display windows for these two phantoms are [0, 0.5] and [0.8, 1.5], respectively.*

### 3.3.2 Data Acquisition

**Numerical simulation**

We employed a modified high-contrast Shepp-Logan phantom and a low-contrast FORBILD head [166] phantom *(Figure 28)*. The two phantoms are compactly supported in a 50.0 cm×50.0 cm rectangular region. The reconstructed image is a 128×128 mesh grid. The distance from the x-ray source to the iso-center is 53.85 cm, and the distance from the source to the detector is 94.67 cm.
cm. An equiangular fan-beam geometry is applied. The fan angle is 0.9592 radians and covers the entire object, and we decrease the fan angle to 0.4796 radians to simulate an interior scan. In our numerical simulations, the standard scan (the entire object was inside the FOV) and interior scan with a circular trajectory are applied using overdetermined system matrices. The number of detector cells and the number of views are both 128.

For the Shepp-Logan phantom, projections were generated by analytically computing overlapping lengths between each x-ray path and ellipsoids. Inconsistencies exist between the analytically calculated intersection lengths and the system matrix coefficients. These inconsistencies were viewed as discretization errors. The discretization errors could also be simulated by using different models to generate the projection and SVD-based reconstructions. To further evaluate the discretization errors, the distance-driven model [79], [167], [168] is adopted to generate projections of the FORBILD head phantom. Poisson distributions were used to model the number of photons on each detector cell [5]. Five, 50, and 200 (× 10^4) photons per detector cell were used to simulate air scans for different noise levels without a bowtie filter.

**Real physical phantom**

A physical phantom is also used to evaluate the SVD methods. The original projection data were acquired on a GE Discovery CT750 HD scanner with a circular scanning trajectory with a radius of 538.5 mm. This geometry defines an FOV with a radius of 249.2 mm. For one rotation, 984 projections are evenly sampled. A total of 888 detector cells is used. The sinogram of the central slice from the equiangular fan-beam geometry is obtained after appropriate pre-processing. The memory size would not allow us to occupy all of the projection data to perform the SVD-based reconstructions. Therefore, the projection data are downsampled. In case 1, the number of detector cells is reduced to 222 by averaging every four adjacent detector cells, and the number of views is downsampling to 246 by discarding three of every four views. In case 2, the number of
detector cells is reduced to 444 by averaging every two adjacent detector cells, and the number of views is downsampled to 123 by discarding 7 of every 8 views.

### 3.3.3 Quantitative Evaluation Methods

The root mean square error (RMSE) and structural similarity (SSIM[138]) are applied to evaluate the reconstructed images inside the ROI quantitatively. The RMSE in our experiment is calculated by

\[
RMSE = \sqrt{\sum_{j \in \text{ROI}, j \notin \text{Known}} (f_j^\ast - f_j^\star)^2 / N_R},
\]

where \(f_j^\ast\) is the ground truth, \(f_j^\star\) is the reconstructed attenuation coefficient, and \(N_R\) is the number of unknown pixels inside the ROI. In addition, the maximum error and standard deviation (STD) inside the rectangular areas marked E and G in Figure 28 are also measured in several experiments.

### 3.4 Results

#### 3.4.1 Shepp-Logan Phantom Reconstruction Results

**Standard Scan Geometry**

The Shepp-Logan phantom is reconstructed from a standard scan dataset. 128 views are uniformly distributed in the scan range \([0, 2\pi]\). The parameters are \(r = 71\% (\varepsilon = 21.3)\) for TSVD and \(\xi = 3.311\) for VSVD. The maximum numbers of iterations for SART and SART-TV are 300. The gradient descent method is applied to minimize the TV, and the step sizes are carefully tuned. Reconstructed images are shown in Figure 29. A narrow display window \([0.15, 0.25]\) is used to determine the differences between the reconstructed images from the proposed and benchmark methods. We observe that FBP reconstructs the worst images with severe artifacts around the object support. SART-TV reconstructs smooth images. SART-TV can visually eliminate artifacts around the object; however, the parameters must be carefully tuned to
avoid removing the high-frequency information (see the three tiny ellipsoids at the bottom of the Shepp-Logan phantom in Figure 28).

![Figure 29. Shepp-Logan phantom reconstruction results from 128 views with different methods. The 1st and 3rd columns are the reconstruction results from noise-free projections, and the 2nd and 4th columns are from the projections assuming $5 \times 10^4$ photons per detector cell in an air scan. The display window is $[0.15, 0.25]$.](image)

Optimizing the parameters for regularized iterative reconstruction algorithms is a challenging problem. Note that the TV minimization assumes a piecewise constant image model, which is more suitable for reconstruction of the Shepp-Logan phantom. It is not surprising that SART-TV reconstructs images with the smallest maximum error, average error, and STD in $E$. Both TSVD and VSVD can suppress streak artifacts somewhat compared to FBP, and they retain more high-frequency information than SART/SART-TV. TSVD can suppress artifacts outside the object support better than SART, while more high-frequency artifacts occur inside the object support.
than with SART. The VSVD reconstruction has a similar image quality to SART but with more high-frequency information. GSVD outperforms TSVD and VSVD. GSVD not only suppresses artifacts around the object support but also retains more high-frequency information with smooth images than TSVD and VSVD. However, we can observe undershoots inside the skull. After evaluating hundreds of $\xi$, undershoots are inevitable under the smallest RMSE/largest SSIM criteria. The RMSE and SSIM are unsatisfying if we vary other parameters to avoid undershoots.

The quantitative evaluations (*Table IX*) show that GSVD provides the smallest RMSE. Although GSVD is inferior to SART-TV in terms of the criteria described above, SART-TV requires 300 iterations to outperform GSVD, which is time-consuming. Although many iterative reconstruction algorithms can converge to the optimal solution of the objective function in tens of iterations, they are still slower than the SVD-based methods, which only require one matrix-vector multiplication, for small image reconstructions.

![Reconstructed Shepp-Logan phantom images from the TSVD method by keeping 10%, 20%, 30%, 60%, 80%, and 90% of singular value magnitudes, respectively. The display window is [0.15, 0.25].](image)

*Figure 30. Reconstructed Shepp-Logan phantom images from the TSVD method by keeping 10%, 20%, 30%, 60%, 80%, and 90% of singular value magnitudes, respectively. The display window is [0.15, 0.25].*
Table IX. Quantitative evaluation of the Shepp-Logan phantom reconstruction results with different methods under 4 noise levels in a 128-views standard scan configuration.

<table>
<thead>
<tr>
<th># of photons ($\times 10^4$)</th>
<th>FBP</th>
<th>SART</th>
<th>SART-TV</th>
<th>TSVD</th>
<th>VSVD</th>
<th>GSVD</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>RMSE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No noise</td>
<td>0.1125</td>
<td>0.0982</td>
<td>0.0945</td>
<td>0.0958</td>
<td>0.0993</td>
<td><strong>0.0931</strong></td>
</tr>
<tr>
<td>5</td>
<td>0.1127</td>
<td>0.0983</td>
<td>0.0946</td>
<td>0.0960</td>
<td>0.0995</td>
<td><strong>0.0932</strong></td>
</tr>
<tr>
<td>50</td>
<td>0.1125</td>
<td>0.0982</td>
<td>0.0946</td>
<td>0.0958</td>
<td>0.0993</td>
<td><strong>0.0931</strong></td>
</tr>
<tr>
<td>200</td>
<td>0.1125</td>
<td>0.0982</td>
<td>0.0945</td>
<td>0.0958</td>
<td>0.0993</td>
<td><strong>0.0931</strong></td>
</tr>
<tr>
<td></td>
<td>Maximum error in E</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No noise</td>
<td>0.0118</td>
<td>0.0092</td>
<td><strong>8.91e-04</strong></td>
<td>0.0463</td>
<td>0.0180</td>
<td>0.0080</td>
</tr>
<tr>
<td>5</td>
<td>0.0198</td>
<td>0.0092</td>
<td><strong>9.18e-04</strong></td>
<td>0.0463</td>
<td>0.0181</td>
<td>0.0080</td>
</tr>
<tr>
<td>50</td>
<td>0.0118</td>
<td>0.0092</td>
<td><strong>8.92e-04</strong></td>
<td>0.0463</td>
<td>0.0181</td>
<td>0.0080</td>
</tr>
<tr>
<td>200</td>
<td>0.0125</td>
<td>0.0092</td>
<td><strong>8.92e-04</strong></td>
<td>0.0463</td>
<td>0.0180</td>
<td>0.0080</td>
</tr>
<tr>
<td></td>
<td>Average Error in E</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No noise</td>
<td>0.0028</td>
<td>0.0022</td>
<td><strong>3.66e-4</strong></td>
<td>0.0115</td>
<td>0.0051</td>
<td>0.0015</td>
</tr>
<tr>
<td>5</td>
<td>0.0057</td>
<td>0.0022</td>
<td><strong>3.57e-4</strong></td>
<td>0.0116</td>
<td>0.0052</td>
<td>0.0015</td>
</tr>
<tr>
<td>50</td>
<td>0.0035</td>
<td>0.0022</td>
<td><strong>3.57e-4</strong></td>
<td>0.0116</td>
<td>0.0052</td>
<td>0.0015</td>
</tr>
<tr>
<td>200</td>
<td>0.0030</td>
<td>0.0022</td>
<td><strong>3.58e-4</strong></td>
<td>0.0115</td>
<td>0.0051</td>
<td>0.0015</td>
</tr>
<tr>
<td></td>
<td>STD</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No noise</td>
<td>0.0036</td>
<td>0.0027</td>
<td><strong>1.65e-4</strong></td>
<td>0.0147</td>
<td>0.0047</td>
<td>0.0020</td>
</tr>
<tr>
<td>5</td>
<td>0.0072</td>
<td>0.0028</td>
<td><strong>1.69e-4</strong></td>
<td>0.0147</td>
<td>0.0047</td>
<td>0.0021</td>
</tr>
<tr>
<td>50</td>
<td>0.0045</td>
<td>0.0027</td>
<td><strong>1.66e-4</strong></td>
<td>0.0147</td>
<td>0.0047</td>
<td>0.0020</td>
</tr>
<tr>
<td>200</td>
<td>0.0038</td>
<td>0.0028</td>
<td><strong>1.65e-4</strong></td>
<td>0.0146</td>
<td>0.0047</td>
<td>0.0020</td>
</tr>
<tr>
<td></td>
<td>SSIM</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No noise</td>
<td>0.5299</td>
<td>0.6144</td>
<td><strong>0.8100</strong></td>
<td>0.5537</td>
<td>0.5638</td>
<td>0.7449</td>
</tr>
<tr>
<td>5</td>
<td>0.5147</td>
<td>0.6143</td>
<td><strong>0.8100</strong></td>
<td>0.5537</td>
<td>0.5638</td>
<td>0.7449</td>
</tr>
<tr>
<td>50</td>
<td>0.5286</td>
<td>0.6143</td>
<td><strong>0.8100</strong></td>
<td>0.5537</td>
<td>0.5638</td>
<td>0.7449</td>
</tr>
<tr>
<td>200</td>
<td>0.5296</td>
<td>0.6143</td>
<td><strong>0.8099</strong></td>
<td>0.5537</td>
<td>0.5638</td>
<td>0.7449</td>
</tr>
</tbody>
</table>
TSVD is applied to reconstruct images from noise-free projections by keeping different magnitude ratios of the singular values. Figure 30 shows the reconstructed results from keeping $r = 10\%, 20\%, 30\%, 60\%, 80\%,$ and $90\%$ of the singular value magnitudes. A very blurred contour of the Shepp-Logan phantom is produced by keeping $10\%$ of the singular value magnitudes. The overall shape of the Shepp-Logan phantom can be observed with $30\%$ of the singular values, but it contains visible ring artifacts. With more singular values, additional details can be reconstructed; however, artifacts caused by the inconsistencies also become significant. When all of the singular values are involved, artifacts cover all of the image information. We also evaluate VSVD with hundreds of different values of $\xi$. Figure 31 shows 6 representative results from different values of $\xi$. With larger values of $\xi$, the image becomes smoother with less high-frequency information, and there is a DC down-shift trend.

![Figure 31. Reconstructed Shepp-Logan phantom images from GSVD method with different values of $\xi$. The display window is $[0.15, 0.25]$.](image)

**Interior Scan Geometry**

The experiments described above are repeated with the interior scan geometry. The fan angle is decreased to $0.4796$ radians, and the detector cells and views remain the same. The same-sized
Shepp-Logan phantom is reconstructed. Because of data truncation, the projections are preprocessed according to [24] and [25] for FBP with appropriate extrapolation. Similarly, four noise levels (no noise, $5 \times 10^4$, $50 \times 10^4$, and $200 \times 10^4$ photons per detector cell) are applied.

![Figure 32. Interior reconstruction of the Shepp-Logan phantom with 128 views from noise-free/noisy datasets. 1st and 3rd columns are from noise-free projections, and 2nd and 4th columns are from projections assuming $5 \times 10^4$ photons per detector cell in an air scan. The display window is [0.15, 0.25].](image)

The parameters are $r = 0.71$ ($\varepsilon = 21.3$) for TSVD, $\xi = 3.4$ for VSVD and $\xi = 1.9$ for GSVD.

*Figure 32 shows the reconstructed ROIs from the noise-free/noisy datasets in a narrow display window [0.15, 0.25]. The FBP results show little because the extrapolation downshifts the DC. Without regularization, SART provides visually satisfying results with truncation artifacts. In contrast, SART-TV reconstructs smoother images but eliminates higher frequency information when the parameters are not tuned carefully. Two small ellipsoids at the center of the FOV are*
blurred because of the regularization in SART-TV. However, the noise reduction advantage of SART-TV makes the image piecewise constant and more similar to the ground truth.

Figure 33. Interior reconstruction of the FORBILD head phantom with noise-free projections (1st and 3rd columns) and noisy projections assuming $5 \times 10^4$ photons per detector cell (2nd and 4th columns). The display window is [0.8, 1.5].

Both TSVD and VSVD can reconstruct the FOV very well but result in bright truncation artifacts around the FOV edge. This is a result of the inconsistency between the analytical projection calculation and the discrete system matrix estimation. Truncating small singular values in TSVD or modifying reciprocals of singular values in VSVD fluctuates the reconstructed values around the FOV boundary. GSVD outperforms all of the other methods. It faithfully reconstructs the high-frequency information (see small ellipsoids at the center of the Shepp-Logan phantom) and suppresses truncation artifacts around the FOV. The quantitative evaluations are summarized in Table X. With the exception of SART-TV, which has the best STD, GSVD provides the best
Table X. Quantitative evaluation of the Shepp-Logan phantom reconstruction results with different methods under 4 noise levels in a 128-views interior scan configuration.

<table>
<thead>
<tr>
<th># of photons</th>
<th>FBP</th>
<th>SART</th>
<th>SART-TV</th>
<th>TSVD</th>
<th>VSVD</th>
<th>GSVD</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>RMSE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No noise</td>
<td>0.1135</td>
<td>0.0254</td>
<td>0.0274</td>
<td>0.0308</td>
<td>0.0304</td>
<td><strong>0.0199</strong></td>
</tr>
<tr>
<td>$5 \times 10^4$</td>
<td>0.1139</td>
<td>0.0261</td>
<td>0.0275</td>
<td>0.0325</td>
<td>0.0320</td>
<td><strong>0.0211</strong></td>
</tr>
<tr>
<td>$50 \times 10^4$</td>
<td>0.1135</td>
<td>0.0255</td>
<td>0.0274</td>
<td>0.0310</td>
<td>0.0306</td>
<td><strong>0.0200</strong></td>
</tr>
<tr>
<td>$200 \times 10^4$</td>
<td>0.1139</td>
<td>0.0254</td>
<td>0.0274</td>
<td>0.0308</td>
<td>0.0305</td>
<td><strong>0.0199</strong></td>
</tr>
</tbody>
</table>

| Maximum error in E |      |      |         |      |      |      |
| No noise | 0.0127 | 0.0247 | 0.0134 | 0.0358 | 0.0458 | **0.0113** |
| $5 \times 10^4$ | 0.0307 | 0.0278 | 0.0130 | 0.0408 | 0.0517 | **0.0182** |
| $50 \times 10^4$ | 0.0140 | 0.0251 | 0.0133 | 0.0326 | 0.0445 | **0.0112** |
| $200 \times 10^4$ | 0.0109 | 0.0247 | 0.0133 | 0.0353 | 0.0464 | **0.0107** |

| Average Error in E |      |      |         |      |      |      |
| No noise | 0.0034 | 0.0108 | 0.0112 | 0.0148 | 0.0155 | **0.0034** |
| $5 \times 10^4$ | 0.0071 | 0.0109 | 0.0110 | 0.0157 | 0.0165 | **0.0048** |
| $50 \times 10^4$ | 0.0041 | 0.0108 | 0.0112 | 0.0150 | 0.0157 | **0.0034** |
| $200 \times 10^4$ | 0.0035 | 0.0108 | 0.0112 | 0.0148 | 0.0156 | **0.0030** |

| STD in E |      |      |         |      |      |      |
| No noise | 0.0035 | 0.0049 | **0.0011** | 0.0077 | 0.0087 | 0.0042 |
| $5 \times 10^4$ | 0.0090 | 0.0068 | **0.0012** | 0.0095 | 0.0100 | 0.0060 |
| $50 \times 10^4$ | 0.0043 | 0.0051 | **0.0012** | 0.0077 | 0.0082 | 0.0041 |
| $200 \times 10^4$ | 0.0036 | 0.0050 | **0.0012** | 0.0077 | 0.0089 | 0.0038 |

| SSIM |      |      |         |      |      |      |
| No noise | 0.8076 | 0.8567 | 0.8800 | 0.8490 | 0.8474 | **0.9672** |
| $5 \times 10^4$ | 0.7895 | 0.8519 | 0.8801 | 0.8409 | 0.8356 | **0.9579** |
| $50 \times 10^4$ | 0.8050 | 0.8562 | 0.8795 | 0.8476 | 0.8460 | **0.9677** |
| $200 \times 10^4$ | 0.8078 | 0.8566 | 0.8801 | 0.8486 | 0.8468 | **0.9700** |
quantitative image quality of all of the benchmark methods. The constant shift of FBP makes it worst in terms of the RMSE. TSVD slightly outperforms VSVD in terms of RMSE. Both TSVD and VSVD suppress high-frequency information but in different ways; TSVD suppresses the high-frequency information by abandoning the corresponding singular values, whereas VSVD weakens the high-frequency information by modifying the magnitudes of the reciprocals of the singular values.

3.4.2 FORBILD Head Phantom Reconstruction

The 128×128 FORBILD head phantom is reconstructed to evaluate the SVD methods for low-contrast objects from an interior scan. In many real applications, the ROI is not accurately located at the center of the object. In this case, we relocate the ROI, extrapolate the projection dataset, and enlarge the reconstructed image size to cover the entire object. A standard scan is performed with a 128-element detector, and 271 views are acquired. Poisson noise is added to simulate four different noise levels. We also apply the extrapolation to deal with the projection truncation for FBP [162], [163]. The parameters are \( r = 90\% \) (\( \varepsilon = 27.2 \)) for TSVD, \( \xi = 4.4 \) for VSVD and \( \xi = 1.5 \) for GSVD. Figure 33 shows the reconstructed ROIs in a [0.8, 1.5] display window. After extrapolation, FBP can reconstruct a visually acceptable image. All of the methods can reconstruct the ROI except for low contrast eyeball details because of noise and data truncation. GSVD reconstructed the best image quality without truncation artifacts, while the other non-SVD-based methods require extrapolation or regularization to eliminate severe truncation artifacts. Furthermore, only GSVD accurately reconstructs the FORBILD head phantom boundary. The quantitative evaluations of the reconstructed results are summarized in Table XI and show that GSVD provides the best image quality compared to the other benchmark methods. SART-TV reconstructs the ROI with the smallest STD in region G. When the ROI only occupies a very small region of the object support, the truncation artifacts in SART and SART-TV are significant. The SVD methods can reduce truncation artifacts very well even without prior
information or extrapolation. Both TSVD and VSVD reconstruct images with more high frequency information and noise. When the Poisson noise is high, VSVD slightly outperforms TSVD.
Case 1: 222 detector cells with 246 views

Case 2: 444 detector cells with 123 views

Figure 34. Real phantom reconstruction results from 246 views (case 1) and 123 views (case 2). The display windows is [-1000, -500]HU.
Table XI. Quantitative evaluation of the FORBILD head phantom interior reconstruction results with different methods under 4 noise levels in 271-views.

<table>
<thead>
<tr>
<th># of photons</th>
<th>FBP</th>
<th>SART</th>
<th>SART-TV</th>
<th>TSVD</th>
<th>VSVD</th>
<th>GSVD</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>RMSE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No noise</td>
<td>0.1178</td>
<td>0.2263</td>
<td>0.2207</td>
<td>0.1479</td>
<td>0.1406</td>
<td>0.0818</td>
</tr>
<tr>
<td>$5 \times 10^4$</td>
<td>0.1224</td>
<td>0.2299</td>
<td>0.2227</td>
<td>0.1660</td>
<td>0.1517</td>
<td>0.1183</td>
</tr>
<tr>
<td>$50 \times 10^4$</td>
<td>0.1183</td>
<td>0.2267</td>
<td>0.2210</td>
<td>0.1528</td>
<td>0.1424</td>
<td>0.0815</td>
</tr>
<tr>
<td>$200 \times 10^4$</td>
<td>0.1179</td>
<td>0.2265</td>
<td>0.2208</td>
<td>0.1528</td>
<td>0.1424</td>
<td>0.0838</td>
</tr>
<tr>
<td></td>
<td>Maximum Error in G</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No noise</td>
<td>0.1597</td>
<td>0.2271</td>
<td>0.1862</td>
<td>0.2712</td>
<td>0.2336</td>
<td>0.1193</td>
</tr>
<tr>
<td>$5 \times 10^4$</td>
<td>0.2141</td>
<td>0.2812</td>
<td>0.2170</td>
<td>0.3348</td>
<td>0.2418</td>
<td>0.1763</td>
</tr>
<tr>
<td>$50 \times 10^4$</td>
<td>0.1638</td>
<td>0.2492</td>
<td>0.1914</td>
<td>0.2778</td>
<td>0.2314</td>
<td>0.1329</td>
</tr>
<tr>
<td>$200 \times 10^4$</td>
<td>0.1552</td>
<td>0.2366</td>
<td>0.1864</td>
<td>0.2782</td>
<td>0.2202</td>
<td>0.1142</td>
</tr>
<tr>
<td></td>
<td>Average Error in G</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No noise</td>
<td>0.0476</td>
<td>0.1384</td>
<td>0.1383</td>
<td>0.0630</td>
<td>0.0573</td>
<td>0.0308</td>
</tr>
<tr>
<td>$5 \times 10^4$</td>
<td>0.0527</td>
<td>0.1388</td>
<td>0.1388</td>
<td>0.0811</td>
<td>0.0623</td>
<td>0.0397</td>
</tr>
<tr>
<td>$50 \times 10^4$</td>
<td>0.0481</td>
<td>0.1392</td>
<td>0.1389</td>
<td>0.0614</td>
<td>0.0571</td>
<td>0.0332</td>
</tr>
<tr>
<td>$200 \times 10^4$</td>
<td>0.0478</td>
<td>0.1388</td>
<td>0.1385</td>
<td>0.0595</td>
<td>0.0546</td>
<td>0.0311</td>
</tr>
<tr>
<td></td>
<td>STD in G</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No noise</td>
<td>0.0330</td>
<td>0.0345</td>
<td><strong>0.0166</strong></td>
<td>0.0692</td>
<td>0.0592</td>
<td>0.0386</td>
</tr>
<tr>
<td>$5 \times 10^4$</td>
<td>0.0462</td>
<td>0.0527</td>
<td><strong>0.0239</strong></td>
<td>0.0931</td>
<td>0.0659</td>
<td>0.0511</td>
</tr>
<tr>
<td>$50 \times 10^4$</td>
<td>0.0344</td>
<td>0.0366</td>
<td><strong>0.0184</strong></td>
<td>0.0647</td>
<td>0.0580</td>
<td>0.0421</td>
</tr>
<tr>
<td>$200 \times 10^4$</td>
<td>0.0332</td>
<td>0.0349</td>
<td><strong>0.0167</strong></td>
<td>0.0614</td>
<td>0.0542</td>
<td>0.0381</td>
</tr>
<tr>
<td></td>
<td>SSIM</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No noise</td>
<td>0.7815</td>
<td>0.713</td>
<td>0.7863</td>
<td>0.6639</td>
<td>0.6914</td>
<td><strong>0.8707</strong></td>
</tr>
<tr>
<td>$5 \times 10^4$</td>
<td>0.7341</td>
<td>0.6968</td>
<td>0.7490</td>
<td>0.6445</td>
<td>0.6805</td>
<td><strong>0.8106</strong></td>
</tr>
<tr>
<td>$50 \times 10^4$</td>
<td>0.7757</td>
<td>0.7104</td>
<td>0.7802</td>
<td>0.6718</td>
<td>0.6927</td>
<td><strong>0.8647</strong></td>
</tr>
<tr>
<td>$200 \times 10^4$</td>
<td>0.7801</td>
<td>0.7121</td>
<td>0.7848</td>
<td>0.6766</td>
<td>0.6990</td>
<td><strong>0.8685</strong></td>
</tr>
</tbody>
</table>
3.4.3 Physical Phantom reconstruction

A real physical phantom is reconstructed in a 150×150 grid that covers an area of 500.0mm × 500.0mm with a pixel size of 3.33 mm×3.33 mm. Because there is no ground truth and this phantom is actually piecewise constant, we apply SART-TV with carefully tuned parameters to reconstruct the standard image from all of the available projections. The reconstructed results from the different methods are shown in Figure 34. The display window is [-1000, -500] HU. In case 1, although FBP reconstructs the image with visible artifacts, it has the best ability to resolve the details (small tubes inside red circles). Visually, the tiny tube structures can be maintained in TSVD and VSVD at the cost of greater streak artifacts. Both SART and GSVD slightly blur the small tubes. SART has the best performance at suppressing streak artifacts. The SVD methods can also inhibit streak artifacts, but their performance is not as good as SART. Only GSVD is comparable to SART. In case 2, with fewer views, FBP reconstructs the image with more significant artifacts. However, all of the other methods can clearly reconstruct the tiny tube structures, although they are not as obvious as in FBP. We can observe streak artifacts in the piecewise part inside the phantom in the results from SART, TSVD, and VSVD. These streak artifacts are suppressed by the regularization in GSVD. However, the tube structures are slightly blurred in GSVD. The quantitative results in Table XII show that the smallest RMSE was obtained by SART in case 1. However, the VSVD-reconstructed image has the best SSIM. In case 2, GSVD has the best performance in both RMSE and SSIM.

Figure 35. ROI reconstruction results with a known sub-region from noise-free projections. (a) is the reconstructed Shepp-Logan phantom from SVD in a display window [0,0.5]. (b) is the reconstruction error of (a) with respect to the ground truth in a display window $[-1.81 \times$
10^{-11}, 2.13 \times 10^{-11}]. (c) is the counterparts of (a) reconstructed by the SART after 300 iterations in a display window [0, 0.5].

3.4.4 SVD Methods with a Known Sub-region in the ROI

Eqs. (66) to (70) show that the interior problem can be solved uniquely and stably by the SVD methods when a sub-region is known inside the ROI. SVD only needs to be performed on \( \mathbf{A}_{\Omega \setminus K} \).

Because the compact support of the object is usually known or measurable, we can treat pixels outside the compact support as the known part. In this experiment, a 215\times215 image matrix is reconstructed from noise-free projections. The known sub-region is a 64.6mm\times62.5mm rectangular region (Figure 28(a)) that includes 24,086 known pixels and 22,139 unknown pixels. There are 324 detector cells, and 60 views are uniformly sampled in a standard scan range. When there is no discretization error and noise, SVD can exactly reconstruct the image (Figure 35(a)). The reconstruction error in a much narrower display window is shown in Figure 35(b). Although the top and bottom parts of the image outside the ROI are noisy, the pixels inside the compact support are accurate. The horizontal striped errors can be viewed as numerical limitations. Figure 35(a) can be regarded as an exact reconstruction. Compared to the SVD methods, SART cannot accurately reconstruct the entire image (Figure 35(c)). Both the TSVD and VSVD methods are applied for noisy projections. We also solved Eq. (56) with the same parameter \( \xi \) for VSVD in an iterative fashion, which is referred to as GD-Tik. The aforementioned projections with six noise levels are applied for the evaluation. Threshold values of \( \varepsilon \) (0.063, 0.060, 0.058, 0.0571, 0.0565 and 0.0560 for six noise levels) and the parameter \( \xi \) (0.0527, 0.0502, 0.0483, 0.0467, 0.046 and 0.0454 for six noise levels) for VSVD are empirically chosen. A maximum of 300 iterations is used for GD-Tik and SART. The known sub-region is enforced in every iteration. The interior reconstruction results are shown in Figure 36. A wider display window [0, 0.5] is applied to better show the reconstructed images with the known part inside the FOV. Artifacts around the boundaries of the known sub-region are visible in all of the reconstructed results, and they are
more obvious in the results from the iterative methods. They are caused by the high frequency disturbances of the jumps between the known and unknown parts and the sensitivity of human visual perception to image edges. *Figure 37* shows the quantitative evaluations. VSVD outperforms TSVD with all noise levels. Because the VSVD results can be viewed as the results of GD-Tik with an infinite number of iterations, the performance of GD-Tik is comparable to that of VSVD in some criteria. Although SART outperforms TSVD and VSVD, for the case of $5 \times 10^4$ photons, it requires at least 24 (6.03 seconds) iterations to outperform TSVD and 55 (12.54 seconds) iterations to outperform VSVD. For the case of $200 \times 10^4$ photons, SART requires 43 iterations (10.03 seconds) to outperform TSVD and 101 iterations (22.53 seconds) to outperform VSVD. GD-Tik requires 144 (15.82 seconds) and 321 (35.32 seconds) iterations to minimize Eq. (6) for $5 \times 10^4$ and $200 \times 10^4$ photons, respectively. Combining the merits of the SVD methods and SART, we can use results with more high frequency information and noise from the SVD methods to initialize the iterative reconstruction algorithms for the best performance in terms of the image quality and computational cost.
Figure 36. Comparison of several algorithms for interior tomography with a known sub-region inside the ROI. From left to right columns, the images are reconstructed from different noise levels. From top to bottom rows, the reconstruction algorithms are TSVD, VSVD, SART with 300 iterations, and GD-Tik with 300 iterations. The display window is $[0, 0.5]$.
3.4.5 SVD Methods for Multiple-resolution Images

Figure 37. Five quantitative criteria comparisons of Figure 36 under six noise levels. The x-axis represents six different noise levels.

We evaluate SVD methods with a multi-resolution strategy. Because GSVD requires twice the memory to perform and we were limited by the memory of our current workstation, we did not apply GSVD in this experiment. The inconsistencies between the matrices for the projection generation and image reconstruction are treated as discretization errors. In this study, VSVD is used ($\xi = 2.98$) to calculate the pseudo inverse of $A_M$. The reconstructed results are shown in Figure 38. As we expect, the images within the ROI can be accurately reconstructed with high resolution. The pixels outside the ROI are not important for the interior tomography.
Table XII. Quantitative evaluation results of the physical phantom reconstruction with benchmark methods.

<table>
<thead>
<tr>
<th>Projection</th>
<th>FBP</th>
<th>SART</th>
<th>TSVD</th>
<th>VSVD</th>
<th>GSVD</th>
</tr>
</thead>
<tbody>
<tr>
<td>222×246</td>
<td>75.46</td>
<td><strong>45.13</strong></td>
<td>45.33</td>
<td>50.36</td>
<td>45.29</td>
</tr>
<tr>
<td></td>
<td>0.6710</td>
<td>0.8726</td>
<td>0.7929</td>
<td><strong>0.8814</strong></td>
<td>0.8676</td>
</tr>
<tr>
<td>444×123</td>
<td>174.40</td>
<td>54.39</td>
<td>62.17</td>
<td>56.37</td>
<td><strong>45.16</strong></td>
</tr>
<tr>
<td></td>
<td>0.3152</td>
<td>0.8166</td>
<td>0.7321</td>
<td>0.8314</td>
<td><strong>0.8386</strong></td>
</tr>
</tbody>
</table>

Similarly, projections with six noise levels are evaluated in the multiple-resolution TSVD and VSVD. We chose \( \varepsilon = 21.1 \) and \( \xi = 2.3 \) for all noise levels. Two multiple-resolution-based reconstruction methods are also evaluated using a known sub-region inside the ROI. The reconstructed results are shown in Figure 39, and the quantitative evaluation results are listed in Table XII. Although the differences in image quality are not obvious in Figure 39, the results in Table XIII clearly show that VSVD outperforms TSVD.

![Figure 38](image)

Figure 38. The interior reconstruction of a modified Shepp-Logan phantom (255×255) using the multi-resolution pixels and VSVD. The display window for (a) is \([0, 0.5]\). The dashed circle in (a) indicates the ROI. The ground truth and the reconstructed profiles along the horizontal and vertical lines in (a) are plotted in (b) and (c).

Table XIII. Comparison of the TSVD and VSVD results in six different noise levels for the third and the fourth row images in Figure 39.

<table>
<thead>
<tr>
<th>Photon ( N(\times 10^4) )</th>
<th>5</th>
<th>10</th>
<th>20</th>
<th>50</th>
<th>100</th>
<th>200</th>
</tr>
</thead>
<tbody>
<tr>
<td>TSVD</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VSVD</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>RMSE in ROI</td>
<td>Max Error in ROI</td>
<td>Max Error in E</td>
<td>Avg Error in E</td>
<td>STD in E</td>
<td></td>
</tr>
<tr>
<td>----------------</td>
<td>--------------</td>
<td>-----------------</td>
<td>-----------------</td>
<td>---------------</td>
<td>------------</td>
<td></td>
</tr>
<tr>
<td></td>
<td>TSVD</td>
<td>VSVD</td>
<td>TSVD</td>
<td>VSVD</td>
<td>TSVD</td>
<td>VSVD</td>
</tr>
<tr>
<td></td>
<td>0.0750</td>
<td><strong>0.0653</strong></td>
<td>0.333</td>
<td><strong>0.2406</strong></td>
<td>0.2532</td>
<td><strong>0.2564</strong></td>
</tr>
<tr>
<td></td>
<td>0.0748</td>
<td><strong>0.0649</strong></td>
<td>0.3311</td>
<td><strong>0.2435</strong></td>
<td>0.2551</td>
<td><strong>0.2400</strong></td>
</tr>
<tr>
<td></td>
<td>0.0747</td>
<td><strong>0.0648</strong></td>
<td>0.3368</td>
<td><strong>0.2463</strong></td>
<td>0.2520</td>
<td><strong>0.2481</strong></td>
</tr>
<tr>
<td></td>
<td>0.0746</td>
<td><strong>0.0647</strong></td>
<td>0.3311</td>
<td><strong>0.2486</strong></td>
<td>0.2494</td>
<td><strong>0.2397</strong></td>
</tr>
<tr>
<td></td>
<td>0.0746</td>
<td><strong>0.0647</strong></td>
<td>0.3253</td>
<td><strong>0.2397</strong></td>
<td>0.2507</td>
<td><strong>0.2413</strong></td>
</tr>
<tr>
<td></td>
<td>0.0745</td>
<td><strong>0.0644</strong></td>
<td>0.3250</td>
<td><strong>0.2404</strong></td>
<td>0.2511</td>
<td><strong>0.2406</strong></td>
</tr>
</tbody>
</table>

### 3.5 Discussion and Conclusion

In this paper, we proposed and evaluated SVD-based CT reconstruction methods. GSVD outperforms the other benchmark methods in both standard scan and interior scan cases. The SVD methods can accurately reconstruct the ROI regardless of where it is located. An overdetermined system matrix is necessary for an accurate reconstruction. For noise-free projections, if the system matrix is overdetermined and the pixel is illuminated by at least one x-ray path, the pixel can be accurately reconstructed even if it is outside the ROI. For noisy projections, the ROI can be accurately reconstructed by GSVD, TSVD, and VSVD. However, the pixels outside of the ROI can no longer be accurately reconstructed. These conclusions are consistent with theoretical results from [6] and [20]. When the matrix was overdetermined, Hansen derived perturbation bounds by establishing the stability of the SVD method to recover all of the variables. When the number of measurements is smaller than the number of unknown variables (pixels), the system matrix is underdetermined, Hansen’s results cannot be directly
applied [159], and SVD does not guarantee accurate reconstruction even for noise-free projections. However, if the object support and a sub-region are known, the system matrix can be grouped into known and unknown parts. If the system matrix that corresponds to the unknown part is overdetermined, the SVD methods allow accurate reconstructions of noise-free projections. Meanwhile, the pseudo-inverse of the singular value matrix can be modified to obtain a regularized solution.

Currently, the SVD of a fairly large matrix is still a difficult problem due to the high computational complexity and the large memory requirements. To resolve the interior issue of a high-resolution image, we propose to represent the image using a multi-resolution scheme. This method reduces the size of the system matrix and provides a practical way to reconstruct a high-

Figure 39. ROI reconstruction of the modified Shepp-Logan phantom (255×255) using a multiple resolution strategy. From left to right columns, six different projection noise levels are applied. The first and second rows are the results from TSVD and VSVD. The third and fourth rows are counterparts of the first and second rows but assuming a known subregion (indicated in Figure 28 (a)). The display window is [0, 0.5].
resolution ROI with limited memory capacity. The proposed methods provide a fast image reconstruction tool. SVD only needs to be performed once when the CT geometry is fixed.

In our experiments, the SVD methods had a lower computational cost than the other methods. Reconstructing a 128×128 image required approximately 0.2 seconds. For SART, it consumed 63.7 seconds for 300 iterations. Furthermore, the computational costs of the projection and backprojection operations depend on the scanning geometry configuration, the projection and backprojection models, and their implementation efficiency. The computational time of the SVD methods was also less than FBP in our experiments because the SVD methods can fully utilize the computational resources to accelerate matrix-vector multiplication. Although the SVD methods have greater computational complexities than FBP, FBP does not have the flexibility to deal with interior scans and other irregular scanning geometries, and the system matrix can be freely modified to perform different reconstruction tasks, including ROI reconstruction.

In conclusion, SVD methods were developed for fast and accurate reconstruction of a broad class of inverse problems that have no analytical solutions. The reconstruction can be rapidly simplified to matrix-vector multiplication. The system matrix can be compressed in both the column and row dimensions by representing the image in a multi-resolution scheme. Additional theoretical analysis and extensive experiments will be conducted to further improve both the SVD performance and the quality of the reconstructed image.
Chapter IV: Evaluation of GPU-Based CT Reconstruction for Obese Patient Problem

Abstract

The obese population is increasing in the United States. There have been modest improvements in scanner hardware and image processing to address some specific challenges associated with imaging of the morbidly obese patients. However, most legacy CT systems lack capabilities to provide sufficient delivery of image-based diagnosis in this increasing subset of the population. One of the most common problems is the projection data truncation in CT imaging due to the massive girths of obese patients. In the past decade, it was proved that the image could be accurately and stably reconstructed from locally truncated projections if certain prior knowledge is known, and this technique is named interior tomography. To overcome the time-consuming issue of the iterative algorithms, we apply GPU techniques to speedup the reconstruction process. In this paper, we evaluate the GPU-based CT reconstruction algorithms (one analytic algorithm and one iterative reconstruction algorithm) for obese patients with both simulated and real clinical datasets. While the approximate analytic reconstruction algorithm outperforms the iterative reconstruction algorithm in terms of computational cost, the iterative algorithm outperforms the analytic one in terms of image quality especially when the projection data is suffered from patient motion, which can happen with increased exposure times required to deliver sufficient x-ray photons in morbidly obese patients. The unsolved issues for iterative reconstruction algorithms lay on not only the reconstruction speed but also the design of the objective function especially the parameter to balance data fidelity and the regularization terms.
4.1. Introduction

The x-ray computed tomography (CT) is an important imaging modality widely applied for clinical diagnosis[46], image-guided surgeries [169] and pharmaceutical industries [170]. Its high spatial resolution, temporal resolution, and fast imaging speed fosters a broad class of applications. However, we cannot disregard the concerns of radiation dose from x-ray CT. Several strategies can be applied to reduce the radiation dose. Most commonly used strategy involves reducing the flux or energy from the x-ray source by controlling the current/voltage or shortening the exposure time for each projection view. However, this introduces stronger noise in projection data. The second strategy is to decrease the number of projection views for reconstruction. Conventionally, it needs about one thousand projections to reconstruct streak-artifact-free images using analytical reconstruction algorithms. If the view number is insufficient, the streak artifacts will be severe. The third strategy is to narrow down the beam to irradiate a region-of-interest (ROI) instead of the entire cross-section of the anatomy. However, this kind of interior problem has no unique solution according to the classical CT theory [171]. In 2007, it was proven that the interior problem can be uniquely and stably solved with a known sub-region inside the ROI [5]–[7] referred to interior tomography. Because the interior tomography demands less projection dataset, it has the capability to handle larger objects, minimize radiation dose, enhance temporal resolution, reduce system cost and suppress scattering artifacts [48].

When scanning obese patients, often the scan field of view is smaller than the patient’s girth, thus inducing truncation artifacts. We hypothesize that in this circumstance, interior tomography can be immensely beneficial in obese patients. The obese population is swiftly increasing during the last three decades [172]–[176] in the United States (U.S.). Currently, more than 65% of the U.S. adults are considered overweighted with 30% of the adults, and 16.9% of the children considered obese[172]–[174], [177], [178]. The high prevalence of obesity among children and adults over the last decades has brought several challenges to the delivery of healthcare including
increased demand for imaging of obese patients to diagnose obesity-related comorbidities. Although some modern MDCT scanners allow scanning of subjects weighing as much as 800 lbs with more efficient detector array system and use of iterative reconstruction techniques to reduce image noise, a vast majority of legacy CT scanners are not optimized to address the imaging challenges in morbidly obese patients [179].

There are several medical imaging problems in obese patients. The weight and girth of obese patients seriously preclude the regular imaging systems such as CT [172], [179]. Although some vendors offer CT scanners with larger gantry aperture and greater weight limits for gantry table, weights of morbidly obese patients often exceed the limitation of patient-bed in most legacy CT scanners. This makes the patient-bed unstable and puts patients into risks. Meanwhile, the wobbly patient-bed disturbs the imaging procedure. Obese patients imaging may also suffer from other challenges such as photon starvation, beam hardening, and truncation artifacts. Often, scan parameters are adjusted (i.e. reduction in helical pitch and/or increase in gantry rotation time) to allow sufficient x-ray photons to pass through morbidly obese patients, who may not be able to hold their breath during scanning which results in motion artifacts. The massive girth may impede inclusion of the entire trunk circumference in the scan field of view (FOV) and result in the interior problem [179], [180]. This results in truncation artifacts which can lead to missed abnormalities due to image quality deterioration. Very few scanners have extended scan FOV (up to 70 cm instead of most commonly applied 50 cm) to address truncation artifacts.

Iterative reconstruction (IR) algorithms provide solutions for data truncation of interior problem engendered by the long girth of obese patients. Compared to analytical reconstruction algorithms, IR algorithms allow more flexible scanning trajectories, have a higher capability to suppress noise and entail fewer projection data. Governing by their objective functions, IR algorithms reconstruct better image quality for different scanning protocols, such as sparse-view projections, limited-angle projections, low-dose scan and data truncation. However, IR algorithms
need multiple projection and backprojection (P/BP) procedures to find the optimal images by minimizing the objective function. The P/BP operations are time-consuming. At the same time, the total reconstruction time is also affected by the convergence rates of the algorithms. To accelerate the reconstruction, we can both design algorithms with higher convergence rates and can accelerate the P/BP procedures. Some high convergence rate algorithms require large memory such as singular value decomposition (SVD)-based algorithm [110], and some of them are arduous to be parallelized (such as ICD method[181]). Accelerating the P/BP procedures is more straightforward. Different P/BP models compromise between the accuracy and efficiency. The design of parallel algorithms hinges on computing architectures. Compared to PC clusters and FPGAs, the GPU is more flexible. It is easy to maintain and update [27], [30], [31]. The GPU was originally designed for fast graphical rendering. It features a great number of computing cores and high bandwidth memory bus. This architecture is appropriate to execute highly data-parallel-computing-intensive algorithms. Some general purpose GPU programming languages were provided to release the researcher/programmers from obscure graphics syntax programming, and CUDA (Compute Unified Device Architecture) and OpenCL (Open Computing Language) are two main APIs. In this paper, we will adopt CUDA to implement image reconstruction algorithms to solve the obese patient problem. CUDA [35], [40], [41] inherits C/C++ as an extension and is rapidly exploited in many areas including medical imaging [120], [121].

Fast backprojection algorithms on GPU and other parallel devices were reported in [120], [123], [124]. A benchmark platform RabbitCT developed by Rohkol et al. [125] was applied to evaluate the speedup performance of GPU-based backprojection. Muller and his collaborators are pioneers of GPU-based CT reconstruction algorithms [126], [127]. Before the general purpose GPU computing language was widely used, they accelerated the FDK algorithm in GPU utilizing RGBA channels of 2D texture operations [127] with shading language for backprojection [128],

120
The CUDA-based GPU acceleration for CT reconstruction was reported in [120]. In [131] and [182], and CUDA-based IR methods were investigated to solve the optimization problem.

In this paper, we would like to evaluate the GPU-based CT reconstruction algorithms for data truncation of the obese patient problem and compare their speedup performance. The paper is organized as follows. In section II, we will present the mathematical model, reconstruction algorithms, the experiment environment and data acquisition. In section III, the reconstructed volumetric images from analytical and iterative methods will be presented and evaluated. In the last part, we will discuss some related issues and make a conclusion.

4.2. Method

4.2.1. Analytical reconstruction algorithm

In this paper, we apply the 3D-weighted cone-beam filtered backprojection (CB-FBP) algorithm [183] to reconstruct the volumetric images analytically. This algorithm is suitable for helical scanning trajectory. It is derived from the well-known FDK algorithm [184]. Similar to the FDK, CB-FBP can be summarized into four steps. The first step is to rebin the projection data by interpolating from a helical geometry to a cone-parallel geometry. This parallel geometry can ameliorate noise uniformity and accelerate image generation speed. The second step is to reweight the projection data according to projection view and fan angles. By exploiting the dependency of the cone angle of a ray, the reconstruction accuracy can be improved significantly after the 3D reweighting. In the third step, the filtering will be performed along the tangential direction of the helical source trajectory as the penultimate step. The projection data is filtered with Hilbert filter using FFT (Fast Fourier Transform). The final step is the standard backprojection.

To reduce the coupling between different components, the rebinning, reweighting, filtering and backprojection procedures are separately implemented in GPU. In each step, the projection
data is first transferred from the host memory to the device memory; after calculation, the results are transferred back to the host memory. Although it is less efficient compared to that all steps are performed in the device memory, the time of data transfer can be ignored because all steps are only performed once. Furthermore, it is hard to extend or update the tightly coupled implementation. We can also see that the filtering step with FFT engages much larger device memory than other steps because of the data padding. When the device memory is insufficient, the whole reconstruction task has to be divided into several sub-tasks.

4.2.2. Iterative reconstruction algorithm

A three-dimensional (3D) volumetric image can be represented as a vector \( \mathbf{f}_{i,j,k} \in \mathbb{R}^{I \times J \times K} \), where \( I, J, \) and \( K \) are pixel numbers along three dimensions of the image, which can also be expressed as \( \mathbf{f}_n \in \mathbb{R}^N \) where \( N = I \times J \times K \). In iterative reconstruction algorithms, the data acquisition process can be modeled as a linear matrix equation

\[
p = \mathbf{A}\mathbf{f} + \mathbf{e},
\]

where \( \mathbf{A} \) is the system matrix, \( \mathbf{e} \in \mathbb{R}^N \) is the additive noise, \( \mathbf{p} \in \mathbb{R}^M \) is the projection data, \( M \) is the total number of sampling which can be calculated as the product between detector cell number and view number, and \( N \) is the pixel number of the volumetric image. For all the IR algorithms, the objective function can be expressed in a general form

\[
\hat{\mathbf{f}} = \arg \min_{\mathbf{f}} \| \mathbf{A}\mathbf{f} - \mathbf{p} \|_W^2 + R(\mathbf{f}),
\]

where \( \mathbf{W} \) is the covariance matrix of \( \mathbf{p} \) (or the data-based approximate covariance [109]). When the noise level is low, \( \mathbf{W} \) can be viewed as identity (\( \mathbf{W} = \mathbf{I} \)). \( R(\mathbf{f}) \) is a regularization term. Because of the enormous size of the projection and the image, it is unfeasible to explicitly store the system matrix \( \mathbf{A} \) in the memory to calculate \( \mathbf{A}\mathbf{f} \). The projection process \( \mathbf{A}\mathbf{f} \) and backprojection process \( \mathbf{A}^\top \mathbf{p} \) are calculated on-the-fly. Many P/BP models were proposed to compromise between the computational complexity and the accuracy, such as ray-driven model[80], area integral
model[81], distance driven (DD) model[79], [167] and separable footprint model [82]. The DD model is immensely employed because of its high precision with matched forward and backward operators as well as low computational complexity. We apply the GPU-based branchless DD model to accelerate the P/BP procedures in the IR algorithms [185] in this paper.

In 2009, it was proved that the interior problem can be uniquely solved if the ROI is piecewise constant [146]. The key is to minimize the total variation (TV) of the ROI. Therefore, a TV-regularized OS-SART (Ordered Subset-Simultaneous Algebra Reconstruction Technique) algorithm is employed to evaluate GPU-based interior tomography implementation for obese patient image reconstruction. The data fidelity term $\|Af - p\|_W^2$ is solved by SART-type algorithm

$$f^{(i+1)}_n = f^{(i)}_n + \frac{\lambda}{\alpha_{n+}} \sum_{m=1}^M a_{m,n} (p_m - A_m f^{(i)})_n,$$

which can be vectorized as

$$f^{(i+1)} = f^{(i)} + \lambda \sum_{b=1}^N (A^{+}_b A^T A^{+}_b (p - A f^{(i)}),$$

where $A^{+}_b$ is a sub matrix of $A$ by selecting the corresponding $b$ rows of $A$. Similarly, $p_b$ means selecting the corresponding elements of $p$. $A^{+}_b$ and $A^{+}_b$ are the corresponding diagonal
matrices for normalization with respect to the sub-matrix $A_{b(0)}$. When all the subset inside $B_t$ are employed, the image $f$ has been updated $t$ times.

The CS theory implies that most of the real signals are sparse in the certain particular domain. The CS-based sparse signal recovery paradigm can be represented as

$$\mathbf{x} = \arg\min_{\mathbf{x}} \|\mathbf{x}\|_0 \quad s.t. \quad \mathbf{p} = \Psi \mathbf{x},$$

(78)

where $\mathbf{x}$ is the sparse signal, and $\mathbf{p}$ is the observed signal, and $\Psi$ is referred to a sensing matrix. However, most of the practical signals (e.g. images) are not sparse in spatial domain. Furthermore, the minimization of $\|\cdot\|_0$ is an NP-hard problem, which can only be approximately solved with some greedy algorithms such as the OMP algorithm[45]. The CS-based signal recovery paradigm cannot be directly applied for interior tomography. We decompose $\Psi$ into the manipulation of the system matrix $A$ and the pseudo inverse of the sparse transform $S^\dagger$. Meanwhile, the NP-hard $\ell_0$-norm is replaced by $\ell_1$-norm. Therefore, we have

$$\mathbf{x} = \arg\min_{\mathbf{x}} \|\mathbf{x}\|_1 \quad s.t. \quad \mathbf{p} = AS^\dagger \mathbf{x}.$$

(79)

Let $S^\dagger \mathbf{x} = f$, therefore $Sf = x$. The objective function can be further converted to

$$\hat{f} = \arg\min_{f} \|Sf\|_1 \quad s.t. \quad \mathbf{p} = Af.$$

(80)

The constrained $\ell_1$-norm optimization can be approximately re-written as an unconstrained optimization problem

$$\hat{f} = \arg\min_{f} \lambda \|Sf\|_1 + \frac{1}{2} \|Af - \mathbf{p}\|_2^2,$$

(81)

where $\|Sf\|_1$ is the regularization term corresponding to the generalized form $R(f)$ in Eq.(2). Actually, $S$ is not necessarily a linear transform, for example, the widely used discrete gradient transform whose $l_1$-norm is TV. The regularized term TV can be defined as
\[ \| \mathbf{f} \|_{TV} = \sum_{k=1}^{K-1} \sum_{j=1}^{J-1} \sum_{i=1}^{I-1} \sqrt{(f_{i+1,j,k} - f_{i,j,k})^2 + (f_{i,j+1,k} - f_{i,j,k})^2 + (f_{i,j,k+1} - f_{i,j,k})^2}. \] (82)

The TV regularized objective function can be expressed as

\[ \hat{\mathbf{f}} = \arg \min_{\mathbf{f}} \lambda \| \mathbf{f} \|_{TV} + \frac{1}{2} \| \mathbf{A} \mathbf{f} - \mathbf{p} \|_{2}^2, \] (83)

where \( \lambda \) is the weighting factor to balance the data fidelity and regularization terms. In this paper, the objective function for obese patient problem is Eq. (83). Eq.(83) can be minimized by either using a gradient descent based method or soft-threshold filtering (STF) method. Because it is hard to estimate the prior TV for the reconstruction from truncated dataset, we apply the gradient descent method. To minimize the TV term, the first derivate of TV term have to be calculated according to

\[ \frac{\partial \| \mathbf{f} \|_{TV}}{\partial f_{i,j,k}} = \frac{-2(f_{i+1,j,k} - f_{i,j,k}) - 2(f_{i,j+1,k} - f_{i,j,k}) - 2(f_{i,j,k+1} - f_{i,j,k})}{\sqrt{(f_{i+1,j,k} - f_{i,j,k})^2 + (f_{i,j+1,k} - f_{i,j,k})^2 + (f_{i,j,k+1} - f_{i,j,k})^2 + \nu}} + \frac{2(f_{i,j,k} - f_{i-1,j,k})}{\sqrt{(f_{i,j,k} - f_{i-1,j,k})^2 + (f_{i-1,j+1,k} - f_{i-1,j,k})^2 + (f_{i-1,j,k+1} - f_{i-1,j,k})^2 + \nu}} + \frac{2(f_{i,j,k} - f_{i,j-1,k})}{\sqrt{(f_{i,j,k} - f_{i,j-1,k})^2 + (f_{i,j,k} - f_{i,j-1,k})^2 + (f_{i,j-1,k+1} - f_{i,j-1,k})^2 + \nu}} + \frac{2(f_{i,j,k} - f_{i,j,k+1})}{\sqrt{(f_{i,j,k} - f_{i,j,k+1})^2 + (f_{i,j,k} - f_{i,j,k+1})^2 + (f_{i,j,k+1} - f_{i,j,k+1})^2 + \nu}}, \] (84)

where \( \nu \) is a tiny positive constant to prevent the denominator being 0.

4.2.3. Experiment setup

4.2.3.1. Experiments environment configuration

All experiments are accomplished on a high-performance workstation in which two Intel Xeon CPUs are configured. Each CPU contains eight physical cores with hyper-threads technique, and the core clock rate is 3.1GHz. The host memory is 128GB. The GPUs in our experiments are the NVIDIA GeForce Titan X (Maxwell architecture) and Tesla K10 (Kepler Architecture). The
GeForce Titan X contains one GM200 GPU with 3072 cores. The core clock is 1.0GHz. It is accommodated with 12GB device memory. The Tesla K10 includes dual GK104 GPUs, and each GPU contains 1536 cores, and the core clock rate is 745MHz. The device memory for every GPU in Tesla K10 is 4GB.

4.2.3.2. Numerical simulation

The numerically simulated projections were generated using helically forward projecting a clinical 512×512×823 volumetric image, voxel size of 0.98mm. A standard 64-slice GE CT geometry was assumed. The beam collimation was 39.375mm. Over a 360-degree range of full scan, 984 projections were uniformly acquired. The radius of the scanning trajectory was 538.5mm. The detector contains 888 detector cells and 64 slices. Other geometrical parameters are summarized in Table 1. The truncated projections were achieved by discarding 111 cells on both sides of the detector. The same volumetric images were reconstructed from both untruncated and truncated projections.

*Table XIV. The geometrical configuration of a helical scan for numerical simulation of a GE CT scanner.*

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source-to-isocenter distance</td>
<td>538.52mm</td>
</tr>
<tr>
<td>Source-to-detector distance</td>
<td>946.75mm</td>
</tr>
<tr>
<td>In-plane detector cell size</td>
<td>1.0239mm</td>
</tr>
<tr>
<td>Cross-plane detector cell size</td>
<td>1.0963mm</td>
</tr>
<tr>
<td>Number of detector columns</td>
<td>888</td>
</tr>
<tr>
<td>Number of detector rows</td>
<td>64</td>
</tr>
<tr>
<td>Offset of the object</td>
<td>(0,0,0)</td>
</tr>
<tr>
<td>Reconstruction FOV</td>
<td>250mm</td>
</tr>
<tr>
<td>Detector offset</td>
<td>(-1.28,0)mm</td>
</tr>
<tr>
<td>Number of views per rotation</td>
<td>984</td>
</tr>
</tbody>
</table>
4.2.3.3. Clinical data
A clinically obese (body mass index >30 kg/m²) adult female patient was scanned three times at Wake Forest University Baptist Hospital on a 64-detector row MDCT (GE Discovery CT750 HD) scanner. The patient underwent scanning for the symptom of shortness of breath to assess for pulmonary embolism. The patient was scanned in a helical scanning mode as summarized in Table XV. The beam collimation for two datasets was 39.375mm and 55.0mm for the third dataset. The numbers of views were 8139, 12950 and 6420, respectively.

4.3. Results

4.3.1. Analytical reconstruction results
4.3.1.1. Numerical simulation results
The reconstructed volumetric image is 512×512×823. Because the CPU-based image reconstruction is time-consuming, we only applied GPU-based CB-FBP. The isotropic pixel size is 0.98 mm. The total reconstruction times are summarized in Table XVI for all the steps. We can see that most of the time is consumed by the backprojection. The representative transverse, sagittal and coronal planes of the ground truth and the reconstructed images are shown in Figure 40 in a display window [-500, 500]HU. Because of data truncation, a bright ring is generated around the FOV in the transverse plane. The image intensity inside the FOV is a little bit small. In both sagittal and coronal planes, the image intensity is not uniform, and it results in stripe artifacts. If certain extrapolation method is employed to smooth the truncated projections to zero, those artifacts can be partially reduced.

<table>
<thead>
<tr>
<th></th>
<th>Rebining</th>
<th>Filtering</th>
<th>Backprojection</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time consumption</td>
<td>4.65s</td>
<td>8.52s</td>
<td>134.70s</td>
<td>152.87s</td>
</tr>
</tbody>
</table>
Figure 40. The transverse, coronal and sagittal planes of the ground truth (left) and GPU-based CB-FBP reconstruction from simulated projections with data truncation (right) in a display window [-500, 500]HU.

4.3.1.2. Clinical application results

The clinical datasets were reconstructed by three different implementations of the CB-FBP algorithm, which are single thread CPU implementation (CPU1-CB-FBP), thirty-two-threads CPU implementation (CPU32-CB-FBP) and GPU-based implementation (GPU-CB-FBP). The reweighted-and-filtered projections are mapped to the texture memory for higher speedup performance, and the texture memory allows hardware-based interpolation when fetching the texels located at a floating point address. All the reconstructed volumetric images are $512 \times 512 \times 512$ in $500 \times 500 \times 500$ mm$^3$. The representative transverse, sagittal and coronal planes of the reconstructed 3D images are shown in Figure 41 in a display window [-500, 800]HU. The patient trunk in the transverse plane cannot be completely reconstructed because of data truncation. Fortunately, the truncated parts are fat which does not contribute information related to pulmonary embolism. Although the patient was asked to hold breath during the CT scan, she
could not hold her breath. Therefore, we can observe some distortions caused by the respiration motion in sagittal and coronal planes indicated by yellow arrows in Figure 41. The difference between CPU and GPU implementations is tiny, and it is hard to be observed. The difference images between GPU and CPU reconstruction results for the first dataset are also shown in the fourth row of Figure 41 in a much narrower display window ([−0.05, 0.05]HU). The hardware based interpolation sacrifices partial accuracy by representing the floating point addresses with a 9-bit fixed point value. This will cause certain accuracy loss. The difference images between CPU and GPU are obvious at the edge of the sagittal/coronal planes in a helical scanning pattern, which can be ignored. Because of the data truncation, cuFFT in GPU and CPU based FFT will have larger deviation.

We summarize the computational costs of CPU1-CB-FBP, CPU32-CB-FBP and GPU-CB-FBP implementations in Table XVI. Compared to CPU1-CB-FBP, the speedup of CPU32-CB-FBP can be 15 folds because the physical number of cores is only 16. The speedup performance of GPU can be up to 362 folds. Limited by the device memory, the projections cannot be totally loaded in one time for filtering with the cuFFT library. For example, the second projection set occupies about 2.66GB with 12,590 views. It will be a significant memory burden if both the projections and the whole volumetric image are loaded into the memory simultaneously. To solve this problem, we divided the reconstruction task into several sub-tasks. Each time, only a subset of the volume is reconstructed. Therefore, if the number of sub-tasks is large, less time is required for each sub-task. However, the total time is expected to be longer because of the overlapping deliberation when splitting the projection data and image volume.
Figure 41. GPU-CB-FBP results of 3 clinical datasets for an obese patient. From left to right, the columns are for transverse, coronal and sagittal planes, respectively. Rows 1-3 are the results from three different datasets in a display window [-500 800]HU. The fourth row is the differences.
between images reconstructed by the CPU and GPU implementations for the first dataset in a display window [-0.05, 0.05]HU.

Table XVI. Performance comparison of analytical reconstruction for three clinical datasets in CPU/GPU implementation.

<table>
<thead>
<tr>
<th>Reconstruction time (unit: s) &amp; speedup</th>
<th>Volume 1</th>
<th>Volume 2</th>
<th>Volume 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single thread CPU</td>
<td>17896.56</td>
<td>1.00x</td>
<td>1.00x</td>
</tr>
<tr>
<td>Multi threads CPU</td>
<td>1257.24</td>
<td>14.23x</td>
<td>1679.07</td>
</tr>
<tr>
<td>GPU</td>
<td>58.39</td>
<td>306.5x</td>
<td>78.96</td>
</tr>
</tbody>
</table>

4.3.2. Iterative reconstruction results

4.3.2.1. Numerical simulation results

In the GPU-based implementations for TV regularized OS-SART, both Titan X, and Tesla K10 are applied for speedup. Different types of GPUs in our workstation can be assigned to different task scales according to their computational powers. The computational power of each GPU can be estimated by multiplying the number of CUDA cores and the core clock rate, while the capacities of the device memory are not taken into consideration for simplification. According to the NVIDIA Profiler software, the task scales for each GPU are carefully calibrated. Multi-GPUs communicate through OpenMP. Because the performance differences between Tesla K10 and Titan X are huge, the computational cost can only be moderately reduced compared to single Titan X implementation. With the OS number of 20, Titan X is allocated with 684 views, and each GPU in Tesla K10 is allocated with 165 views per subset iteration. The corresponding slice indices for each GPU can be calculated according to the number of views and the geometry configuration. The whole sub-volume allotted in each GPU have to be completely covered by rotating the source point and the detector simultaneously along the helical scanning trajectory. For backprojection, Titan X is assigned with 451 slices and each GPU in Tesla K10 is assigned 186 slices to reconstruct the entire 823 slices. The first and the last indices of the projection views
can be similarly calculated according to the sub-volume slices range and the system configuration. We only applied Titan X to calculate the discrete gradient of TV term while both the projection and image update are finished in CPU with OpenMP.

![Image](image.png)

*Figure 42. The timeline of one inner iteration with one subset in 3 GPUs. The length of the bars indicates the duration of time.*

We plotted the time profile with NVIDIA Profiler in *Figure 42*. It shows the time bar of one subset iteration. The duration is 4.12 seconds. Each outer iteration requires about 88 seconds for 20 subsets. In Titan X, the projection kernel occupies 151.32ms; the backprojection kernel occupies 1.43s, and TV kernel occupies 173.919ms. We can see several gaps between different kernels. They represent the synchronization and host-device / device-host memory data transfers (indicated by khaki bars). With more OSs, the convergence rate is improved by scarifying the speedup performance in iterations. One shortcoming of the OS-SART is that more OS occupies large host memory to store the row sum and column sum of the subsystem matrix calculated by projecting/backprojecting the image/projection with value one. Of course, this process can be calculated on the fly and the computational time will double.

The reconstruction results along the transverse, sagittal and coronal planes after 5, 15, 25 and 35 iterations are shown in *Figure 43* with a [-500, 500]HU display window. We notice that TV regularized OS-SART can reconstruct visually satisfying results after fifteen iterations if twenty subsets are applied. From the horizontal profile along the transverse plane indicated in *Figure 43(e)*, we can see that the quantitative values inside the FOV are asymptotic to the ground truth with more iterations, while larger bias can be observed outside the FOV. Of course, from the
profile we can also see that the reconstructed images inside the FOV have a little bit bias from the ground truth. In interior tomography, it requires many iterations to eliminate this low-frequency shift, and the regularization parameter has to be tuned carefully.
Figure 43. The results reconstructed by TV regularized OS-SART from simulated projections. (a), (b), (c) and (d) are with 5, 15, 25 and 35 iterations, respectively, in a display window [-500, 500]HU. From top to bottom, the rows are for transverse, coronal and sagittal planes, respectively. The horizontal profiles of the transverse plane indicated along the dashed line at the center of the transverse planes are shown in (e).
4.3.2.2. Clinical application results

Figure 44. The transverse, sagittal and coronal planes reconstructed from the first clinical dataset by the TV-OS-SART in GPU with different iteration numbers and step sizes. The display window is [-500, 800]HU.
To suppress the truncation artifact in iterative reconstruction algorithms, the reconstructed image should cover the compact support of the object. Therefore, for real clinical datasets, a 768x768x512 volumetric image is reconstructed to cover a region of 750x750x500mm³. The pixel size is the same as in analytical reconstructions. Twenty subsets and FISTA are also used to accelerate the convergence of SART.

The intermediate transverse images during iterations (5, 15, 25 and 35 iterations) are displayed in Figure 44. Different rows show different step sizes for gradient descent based TV regularization. If the step size is too small, the regularization term is almost invalid. With the increase of iterations, the artifacts become more and more severe. On the other hand, if the step size is too large, the reconstructed images are prone to be piecewise blocky. Both the number of iterations and the step size for TV regularization have to be carefully chosen. In this experiment, 15~25 iterations are sufficient to reconstruct a decent image. After 15 iterations, the volumetric images are already updated 300 times. From the sagittal and coronal planes, we can see that the distortions caused by respiratory motion can be suppressed with IR algorithms. However, there still exist dark-and-bright artifacts resulting from the data truncation.

As we know, a larger number of OS reduces the computational burden for P/BP in one inner iteration. However, the total reconstruction time will increase. This is both because of the data transfer among host/device memories and because that the GPU speedup will be insignificant when the computing scale is not large enough to fully utilize all cores and bandwidth. Table XVI lists the computational costs in one iteration for three GPUs including the P/BP procedures.

Because the dimension of the volumetric image along patient bed direction is octuple of the number of detector slice, the backprojection task with branchless DD model is much more time-consuming. When the integral image is backprojected to voxels, most of the tapered areas between the source and the detector do not cover the voxels to be backprojected in each GPU thread. However, in GPU-based branchless DD backprojection implementation, all views have to
be accessed, and four texels on each view will be fetched which is futile. We can see that with 3 GPUs, the P/BP times cannot be one-third. For example, while it takes 20.6s for one iteration for 12,590 views (see Table XVII), the total computational cost will be about 10 minutes assuming 20 iterations and 20 subsets.

Table XVII. The total and kernel projection/backprojection times in triple GPUs with three different clinical datasets.

<table>
<thead>
<tr>
<th></th>
<th>8139 views</th>
<th>12590 views</th>
<th>6420 views</th>
</tr>
</thead>
<tbody>
<tr>
<td>Projection</td>
<td>2.41s</td>
<td>3.88s</td>
<td>2.08s</td>
</tr>
<tr>
<td>Projection Kernel</td>
<td>1.64s</td>
<td>2.59s</td>
<td>1.28s</td>
</tr>
<tr>
<td>Backprojection</td>
<td>10.94s</td>
<td>15.50s</td>
<td>8.52s</td>
</tr>
<tr>
<td>Backprojection kernel</td>
<td>8.69s</td>
<td>13.56s</td>
<td>7.76s</td>
</tr>
<tr>
<td>One iteration</td>
<td>14.17s</td>
<td>20.06s</td>
<td>12.32s</td>
</tr>
</tbody>
</table>

4.4. Discussion and conclusion

In this paper, we evaluated the performances of GPU-based analytical reconstruction algorithm (CB-FBP) and IR algorithm (TV-OSSART) for obese patient imaging with data truncation. The analytical reconstruction algorithm is faster, and the reconstructed images do not have to cover the entire compact support of the object. When the data truncation is moderate, CB-FBP can maintain high-frequency information. Its GPU-based implementation can speedup more than 300 folds compared to the single thread CPU implementation. In the near future, more efforts will be made to make it faster. We can see that because the reweighted projection data is too huge to store in device memory to perform the FFT completely, the filtering has to be carried out by dividing the projection data into several subsets and all subsets are filtered in order. According to the documentation of NVIDIA cuFFT library that the FFT can be finished in an off-board fashion, further potential speedup can be achieved. The backprojection can also be optimized to reduce the reconstruction time. However, for real clinical datasets, we still cannot conquer the respiration motion artifacts and dislocations. Developing analytical reconstruction algorithms that can
eliminate the respiration motion artifact is still an opening problem. The raw datasets provided in this paper can serve as a platform to evaluate the future new algorithms for artifact reduction.

Different from the CB-FBP, TV-OS-SART is more time-consuming which requires multiple P/BPs to reconstruct optimal results. From the time bar analysis, we can see that there is no overlap between the host/device memory copy and the GPU computing. This means they are in a synchronous manner. We will work on an asynchronous implementation, and all the image/projection data updates in IR can be completed in GPU without multiple data transfers between host and device memories. Although the data transfer time could be ignored compared to the computational time in both P/BP when the data scale is large, the data transfer still occupies considerable time if the OS technique is employed. This is because the P/BP scale becomes smaller, especially when these small scaled P/BP procedures are performed multiple times in iterations. Regularized by the TV term, a satisfying image can be reconstructed from much fewer projection data compared to analytical reconstruction algorithms. The parameter to balance the data fidelity and regularization term has to be carefully tuned. Usually, the optimal parameter is determined by different image contents and noise levels. However, it is still an open problem to select the parameter automatically.

In summary, we implemented and evaluated both analytic and iterative algorithms using GPU for data truncation problem of obese patients. While the approximate analytic reconstruction algorithm outperforms the iterative reconstruction algorithm regarding computational cost, the iterative algorithm outperforms the analytic algorithm in terms image quality especially when the projection data is suffered from patient respiration motion. To reconstruct a 512×512×512 volumetric image from 12,590 views, satisfied image quality can be achieved within 10 minutes by the GPU-based iterative algorithm. Raw datasets of an obese patient are provided to peers for future evaluation of algorithms for image reconstruction and artifact reduction.
References


K. Mueller, R. Yagel, and J. J. Wheller, *A Fast and Accurate Projection Algorithm for 3D Cone-Beam Reconstruction with the Algebraic Reconstruction Technique (ART)*.


K. Mueller, F. Xu, and N. Neophytou, “Why do commodity graphics hardware boards (GPUs) work so well for acceleration of computed tomography?,” in *SPIE Electronic Imaging, 2007*, vol. 6498, p. 64980N.


R. Liu, L. Fu, B. De Man, and H. Yu, "GPU-based Branchless Distance-Driven Projection and Backprojection," *TCI*.

Curriculum Vitae

Educational background

- 2012-2016, Ph.D.: Biomedical Engineering, Wake Forest University
  - Dissertation: Software and Hardware Acceleration for Interior Tomography
- 2009-2012, Master: Computer Science, Chinese Academy of Sciences
  - Thesis: The Design and Implementation of a GPU-Based X-Ray Phase-Contrast Imaging Simulation System
- 2005-2009, Bachelor: Computer Science, South China Normal University

Academic awards

- Student Travel Award of 13th Fully 3D Image Reconstruction in Radiology and Nuclear Medicine 2015

Memberships

- IEEE Student Member
- SPIE Student Member
- BMES Student Member

Publications

- R. Liu, L. Fu, B. De Man, H. Yu, “GPU-based Branchless Distance-Driven Projection and Backprojection” (submitted to IEEE Transaction on Computational Imaging)


• M. Wang, Y. Zhang, R. Liu, S. Guo, H. Yu, “An Adaptive Reconstruction Algorithm for Spectral CT Regularized by a Reference Image” accepted by Physics of Medicine and Biology

• R. Liu, H. Yu, “GPU-based Fast Implementation for Interior Tomography” BMES 2013 annual meeting. (Poster)

• R. Liu, H. Yu, “GPU-based implementation for interior tomography,” 3rd CT meeting 2014


• R. Liu, H. Yu, “CUDA based Spectral CT simulation,” BMES 2015 annual conference (Poster)

• R. Liu, L. Fu, B. D. Man, H. Yu, “GPU Acceleration of Branchless Distance Driven Projection and Backprojection,” 4TH CT Meeting 2016


Experiences

Research Assistant
University of Massachusetts Lowell – Lowell, MA
• Developing the cross-platform high performance medical imaging software and evaluating its performance in different practical patient imaging problems.
• Developing the dictionary learning and deep learning based denoising algorithms to improve the image quality from highly noisy samplings.
• Proposed and implemented a 150-folds faster cross-platform imaging software library in GPU.
• Proposed a novel non-iterative image restoration algorithm on Amazon EC2 cloud computing platform.
• Implemented a highly parallelizable image reconstruction algorithm by interpolating the helically scanned data into decoupled fan-beam data on GPU with dictionary learning denoising method.
• Attended the Low Dose CT Grand Challenge competition.
• Developed GPU based multi-spectral medical imaging simulation platform.

Intern
GE Global Research – Niskayuna, NY
• Proposed three x-ray imaging models for medical imaging.
• Implemented 180-folds faster GPU medical imaging library with both MATLAB and C++ interfaces.
• Proposed novel data structure to improve the accuracy of medical imaging model in GPU texture memory and applying for an intelligent patent.
• Designed four test cases to evaluate the performance of the software library.
Intern
MathWorks – Natick, MA

08/2016 to 11/2016

• Debugger supporter for Simulink.

• Developed new features for Simulink.