Eurographics Symposium on Parallel Graphics and Visualization (2020)
S. Frey, J. Huang, F. Sadlo (Editors)

Effective Parallelization Strategies for Scalable, High-Performance Iterative Reconstruction

Christiaan Gribble[1]

[1]Applied Technology Operation, SURVICE Engineering

Abstract

Iterative reconstruction techniques in X-ray computed tomography converge to a result by successively refining increasingly accurate estimates. Compared to alternative approaches, iterative reconstruction imposes significant computational demand but generally leads to higher reconstruction quality and is more robust to inherently imperfect scan data. We explore several strategies for exploiting parallelism in iterative reconstruction and evaluate their scalability and performance on modern workstation-class systems. Results show that scalable, high performance iterative reconstruction is possible with careful attention to the expression of parallelism in both the projection and backprojection phases of computation.

CCS Concepts

• ***Computing methodologies*** → ***Parallel algorithms;*** *Ray tracing;*

1. Introduction

X-ray computed tomography (XCT) is an important visualization technique in many medical and industrial imaging scenarios. Three-dimensional (3D) reconstructions of scanned artifacts are obtained by computing the distribution of X-ray attenuation coefficients using two-dimensional (2D) projections (i. e., images) produced by XCT scanners.

XCT data acquisition proceeds by rotating an emission source and a corresponding detector around the target object to record the pattern of X-ray emission in the field of view at many angles. In particular, an X-ray of initial intensity I_0 is emitted and passes through an object; the X-ray is attenuated according to the Beer-Lambert Law and measured as intensity I_1 at the detector:

$$I_1 = I_0 exp^{-\int_L \mu(x)dx},$$

where μ is the attenuation coefficient of the material at location x and L is the path of the X-ray through the object. The attenuation coefficient characterizes the rate at which X-rays are weakened by scattering or absorption as they propagate through the object. Intuitively, this coefficient is proportional to the material density at the corresponding position. Therefore, recovering the coefficients from projection intensities I_0 and I_1 is equivalent to computing the density distribution of the object.

Whereas analytic reconstruction techniques exploit properties of operations in the frequency domain to produce accurate reconstructions, iterative reconstruction techniques converge to a solution by successively refining increasingly accurate estimates. Iterative reconstruction algorithms impose significant computational demand: a single iteration of the reconstruction imposes one (forward) projection and one backprojection, as well as intermediate updates; these steps are in contrast to the single backprojection step in typical analytic methods. However, iterative algorithms generally lead to higher reconstruction quality (in terms of both contrast and resolution) and are more robust to noisy or sparse projection data. As such, there is renewed interest in iterative reconstruction, particularly with the introduction of widely available, relatively low-cost parallel computing platforms.

After a brief review the physical and mathematical basis of XCT, we explore several strategies for exploiting parallelism in iterative reconstruction and evaluate their scalability and performance on modern workstation-class systems. Results show that scalable, high-performance iterative reconstruction is possible with careful attention to the expression of parallelism in both the projection and backprojection phases of computation.

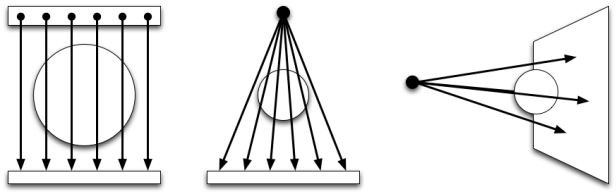
2. Background

XCT data acquisition consists of directing X-rays through a volume for several source-detector orientations and measuring the decrease in intensity along a series of linear paths as characterized by the Beer-Lambert Law. XCT reconstruction then proceeds by computing the distribution of X-ray attenuation within the volume from the scan data. We review the physical and mathematical basis of XCT before exploring several strategies for exploiting parallelism in iterative reconstruction in Section 3.

DOI: 10.2312/pgv.20201074

2.1. Data Acquisition

The development of XCT reconstruction techniques is directly related to the progression of scanning technologies. The earliest scanners used *parallel-beam* geometries (Figure 1a), but later improvements led to *fan-beam* geometries (Figure 1b). Reconstructions from projections obtained with these scanners are typically stacked 2D cross-sections, rather than full 3D volumes, and often lead to poor resolution along the axial direction. Modern scanners enable continuous data acquisition—and thereby improved image quality—by continuously rotating the gantry while simultaneously translating the target object, leading to a spiral or helical locus. Continued developments in such spiral or helical scanners include multi-slice (or multi-detector) devices. Here, multiple rows of detectors capture multiple cross-sections simultaneously. In contrast, modern *cone-beam* geometries comprise a large number of detector rows such that coverage in the axial direction is comparable to coverage across the transverse direction (Figure 1c).



(a) parallel-beam (b) fan-beam (c) cone-beam

Figure 1: Common XCT scanner geometries. *Early XCT scanners used parallel-beam geometries (a), but later improvements led to fan-beam geometries (b). Modern cone-beam geometries comprise an area detector (c).*

X-ray photons emitted from a radiation source naturally form a cone emanating from that source. Collimators are used to restrict the X-ray beam to a single line (parallel-beam) or set of lines (fan-beam) on the detector plane. In contrast, cone-beam geometries comprise area (2D) detectors, which reduce scan time and improve X-ray energy efficiency. For small cone angles, fan-beam reconstruction techniques are typically adapted for cone-beam geometries. However, for beams subtending large areas on the detector (typically ten degrees or more), reconstruction techniques designed explicitly for cone-beam geometries must be applied.

2.2. Reconstruction

A more explicit view of the XCT reconstruction process outlined in Section 1 is given by the following equation:

$$-\frac{I_1}{I_0} = \int_L \mu(x)dx.$$

In other words, given input and output X-ray intensities I_0 and I_1, respectively, the line integral of the object attenuation coefficients along the ray path can be determined. The task of XCT reconstruction is to calculate the attenuation coefficient function $\mu(x)$ at various locations x based on initial and measured intensities I_0 and I_1 for a collection of different source-detector orientations.

Radon Transform. Consider the 2D cross-section depicted in Figure 2. Let $f(x,y)$ be the target object; further, parameterize each

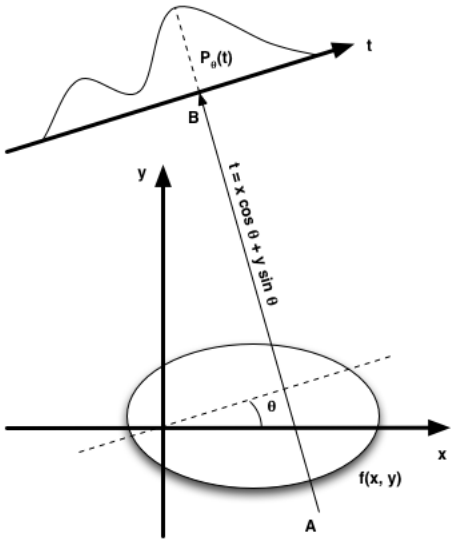


Figure 2: The Radon transform. *The line integral $P_\theta(t)$ is called the* Radon transform *of the function $f(x,y)$. Intuitively, $P_\theta(t)$ is the sum of values $f(x,y)$ at each point t along line AB. (Schematic after* Figure 3.1 *in* Principles of Computerized Tomographic Imaging *[KS88])*

line integral by (θ,t). Then, line AB in Figure 2 is:

$$t = x\cos\theta + y\sin\theta.$$

Using this relationship, define the line integral $P_\theta(t)$ as:

$$P_\theta(t) = \int_{l(\theta,t)} f(x,y)dl.$$

The function $P_\theta(t)$ is called the *Radon transform* of the function $f(x,y)$. Intuitively, $P_\theta(t)$ is simply the sum of values $f(x,y)$ at each point t along line AB, sometimes called the *ray-sum*.

In practice, $P_\theta(t)$ and $f(x,y)$ are functions of discrete variables—the variables θ, t, x, and y take on only a finite number of values. In XCT, each point of measurement on the detector, for each projection angle, is a bin, or ray-sum, and the elements of each 2D cross-section are pixels. The number of bins N_b equals the number of points of measurement N_s multiplied by the number of angles N_θ. The raw data captured by XCT is represented by an $N_s \times N_\theta$ image called a *sinogram*, as shown in Figure 3.

Values in a sinogram and the corresponding reconstructed slice can be considered as matrices or vectors. It can be shown (see, for example, Appendix 3 in Bruyant's article [Bru02]) that vector P is the matrix product of the forward projection operator A and vector f:

$$P = Af.$$

Thus, the value P_i for any bin i is a weighted sum of the n pixel values in the image:

$$P_i = a_{i1}f_1 + a_{i2}f_2 + \ldots + a_{in}f_n = \sum_{j=1}^{n} a_{ij}f_i.$$

This equation encodes a discrete formulation of the projection

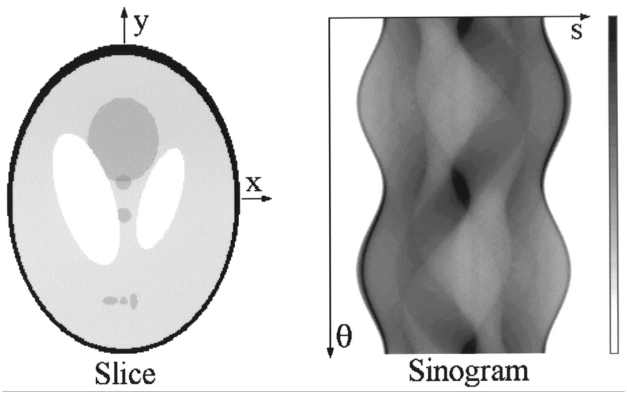


Figure 3: A sinogram. *Sinograms provide a visual representation of raw XCT data. Here, the left panel shows a 256×256 slice of the* Shepp-Logan phantom *[SL74], while the right panel depicts the corresponding sinogram, with 256 pixels per row (*s*) and 256 uniformly sampled angles (*θ*). (Sinogram image originally published in* JNM *by P. Bruyant [Bru02].* © *2002 SNMMI. Available via* http://jnm.snmjournals.org/content/43/10/1343.long.*)*

operation. Importantly, this equation also leads to a discrete form of the reconstruction process:

$$f = A^{-1}P.$$

This system expresses the reconstructed slice as a matrix product of the inverse projection operator A^{-1} and measured values P.

In theory, direct methods exist to solve the system $f = A^{-1}P$, but a direct approach is impractical for several reasons:

- A^{-1} may not exist.
- A^{-1} may not be unique.
- A^{-1} may be ill-conditioned.

In practice, A^{-1} is (at least) ill-conditioned, a result of the measurement noise inherent to XCT data acquisition. Moreover, matrix inversion is computationally expensive, even for relatively small matrices. As a result, other more efficient methods must be used to solve $f = A^{-1}P$.

Backprojection. The techniques necessary to overcome the drawbacks of a direct solution to $f = A^{-1}P$ rely on the *backprojection operator*,

$$b(x,y) = \int_0^{\pi} P(\theta,t)d\theta,$$

which represents the ray-sums of all rays passing through any point (x,y). As with the projection operator, $b(x,y)$ can be discretized, leading to:

$$b(x,y) = \sum_{k=1}^{n} P(\theta_k,t_k)\Delta\theta,$$

where n is the number of projections acquired over π radians, θ_k is the k^{th} angular position of the detector, t_k is the k^{th} location along the detector, and $\Delta\theta$ is the angular step between two successive projections.

Importantly, backprojection is not simply the inverse of forward projection: applying backprojection does not yield $f(x,y)$, but a blurred version of $f(x,y)$. As a result, practical methods strive to reduce or eliminate artifacts due to blur.

In particular, one solution to decrease blur first backprojects the measured data using a summation algorithm and then filters the result; this approach is therefore called *backprojection filtering*. An alternative approach reverses the order of backprojection and filtering, leading to the *filtered backprojection* (FBP) method. FBP is most commonly used in commercial XCT scanners, for which the Feldkamp-Davis-Kress (FDK) algorithm [FDK84] and its derivatives are the most popular.

Work to accelerate FDK-type FBP algorithms follows the historical trends common to high performance computing techniques applied over the past 30 years, including low-level algorithmic improvements for core operations; exploitation of parallelism via threaded or vector processing, distributed computing, and modern programmable graphics processing units (GPUs); and, dedicated hardware architectures or field programmable gate array implementations. Noteworthy examples from the literature include an early single-instruction, multiple-data (SIMD) implementation for CPUs [YNC01], a more recent CUDA implementation for NVIDIA GPUs [MMBL12], and a hybrid MPI/CUDA implementation for distributed memory systems [BAL*13]. Turbell [Tur01] provides an excellent summary of several FDK-type algorithms for FBP.

In contrast to these analytic methods, iterative reconstruction techniques converge to a solution by successively refining increasingly accurate estimates. The various iterative algorithms, which include families of both algebraic and statistical reconstruction techniques, are distinguished by the way in which differences between estimates and measurements are computed and applied to determine the estimate for the next iteration.

For example, the algebraic reconstruction technique (ART) [GBH70], is expressed by:

$$f_j^{(k+1)} = f_j^{(k)} + \frac{g_i - \sum_{j=1}^{N} f_{ji}^{(k)}}{N},$$

where $f_j^{(k)}$ and $f_j^{(k+1)}$ are the current and next estimates, respectively; g_i is the measured number of counts for ray i; and, for the k^{th} iteration, $\sum_{j=1}^{N} f_{ji}^{(k)}$ is the sum of counts in the N pixels along ray i. In this case, the next estimate is determined by adding a correction term, computed as the difference between the measured and estimated projections, to the current estimate.

As with analytic techniques, work to accelerate iterative algorithms follows historical trends in high performance computing. From low-level algorithmic improvements to exploitation of parallelism at many levels, optimization of iterative algorithms has improved performance such that these techniques are very nearly ready for practical application in clinical and industrial imaging scenarios. Noteworthy examples from the literature include an early SIMD implementation of ray-driven forward projection for CPUs [CSB*99] and a CUDA implementation for NVIDIA GPUs [CCHW11]. Other work examines the use of accelerated *ray-tracing methods* (RTM) for both projection and backprojection on contemporary GPUs [NJL13, NL15], while Flores et al. [FVM*13]

explore a linear system formulation in CUDA. The parallelization strategies we explore in Section 3 leverage RTM for both projection and backprojection. Though we explore thread-level parallelism for multicore CPUs in this work, these strategies can be applied on a per-lane, rather than per-thread, basis to leverage vector processing on modern CPUs and GPUs.

Treibig et al. [THH*13] investigate the impact of low-level algorithmic optimizations, parallelization, SIMD vectorization, and microarchitectural features for backprojection performance on contemporary multicore CPUs. Vanhove et al. [VVHV13] present a similar study in which contemporary GPU implementations of several iterative reconstruction algorithms are compared to determine the fastest alternative with favorable spatial-resolution/noise tradeoffs. These early studies provide a useful point of reference for understanding iterative reconstruction performance within the evolution of modern CPUs and GPUs; our scalability study adds another data point by providing results for modern multicore CPUs (with possible extension to modern GPUs, if desired).

Molina et al. [dMSGB*18] evaluate large volume (1024^3 voxels) reconstruction problems using ray-driven projection and voxel-driven backprojection, demonstrating up to 48× improvement over an OpenMP implementation for modern CPUs. Biguri et al. [BLB*19] introduce a partitioning strategy for both projection and backprojection that enables efficient single-CPU/multi-GPU execution with arbitrarily-large volumes within the Tomographic Iterative GPU-based Reconstruction (TIGRE) Toolbox [BDHS16], a collection of methods for XCT reconstruction in MATLAB using CUDA. In this work, we evaluate performance with reconstruction volumes up to 800^3 voxels on several different (single-node, shared-memory) multicore CPU systems; however, these strategies can be combined with other work- or data-distribution algorithms to implement iterative reconstructive across multiple nodes in a distributed-memory system, if desired.

For additional information concerning the role of parallelism and iterative algorithms in contemporary XCT reconstruction, the reader is referred to excellent review articles by Ni et al. [NLHW06] and by Beister et al. [BKK12], which explore these topics in more detail.

3. Parallelization Strategies

Iterative reconstruction algorithms have high computational demand but generally lead to higher reconstruction quality, are more robust to noisy or sparse projection data, and—as we show here–can be explicitly parallelized to scale gracefully. Our fastest iterative reconstruction approaches also offer high performance: reconstructions involving 300 million or more ray-sums complete in seconds or minutes using modern workstation-class systems, thereby allowing users to quickly identify features of interest, ultimately reducing time-to-insight across various imaging scenarios.

We are motivated by an industrial imaging application designed to support near real-time decision-making in non-destructive testing and dimensional metrology. Key system parameters and performance targets for this application are outlined in Table 1. These goals serve as the primary metrics in our performance study and will ultimately guide deployment decisions regarding the imaging and visualization pipeline in our target application. In particular, any XCT reconstruction technique intended to inform decisions in our target application must process about 80–160 projections in two minutes (or about 0.6–1.3 projections per second) to satisfy our near real-time performance constraint.

Parameter	Goal
Reconstruction interval	2 minutes
Projections per iteration	8–16 images
Maximum iterations	10 iterations
Spatial extent	1.2 m^3
Feature size	1:400 (3 mm^3)
Image resolution	2 MP (1600×1200 pixels)

Table 1: Key performance and fidelity goals. *We are motivated by an industrial imaging application designed to support near real-time decision-making in non-destructive testing and dimensional metrology. The values here outline the key performance and fidelity goals for our target application.*

We seek to capitalize on the computing resources afforded by widely available, relatively low-cost parallel computing platforms for our target application; as such, we design and implement several forward projection (FP) and backprojection (BP) variants to support iterative reconstruction, and we evaluate their scalability and performance on modern workstation-class systems.

To begin, we observe that ray-sum computations in the FP operation are trivially parallel, as the corresponding computations do not update reconstruction volume voxels, but only per-pixel ray-sums; as a result, rays through each detector pixel (s,t) can be propagated simultaneously without use of low-level synchronization primitives or significant concern for high-level task/thread assignments.

In contrast, the BP operator updates each reconstruction volume voxel (x,y,z) through which rays corresponding to each detector pixel (s,t) pass. As a result, BP is not trivially parallel, and care must be taken when attempting to exploit parallelism during BP.

A straightforward approach to parallel BP uses low-level synchronization primitives, e. g., mutexes, to serialize updates to data shared among threads. Though correct, contention for shared data may limit the effectiveness of this approach in practice.

An alternative parallel BP approach instead assigns tasks, i. e., rays through detector pixels (s,t), to simultaneously executing threads such that no two rays within the same group pass through the same voxel (x,y,z). In particular, the *pixel spacing work group assignment* approach determines this conflict-free task/thread mapping by computing the maximum distance between any two rays that pass through the same voxel (x,y,z) across the entire detector plane. BP tasks are then assigned to threads such that the distance between rays, $d(s,t)$, is greater than the maximum distance for the corresponding axis, $d_{max}(s,t)$, as depicted in Figure 4.

Another approach to determine conflict-free task/thread mappings leverages voxel projection. In this case, the corners of each reconstruction volume voxel (x,y,z) are projected to the detector plane, and the maximum projection extents are tracked across all

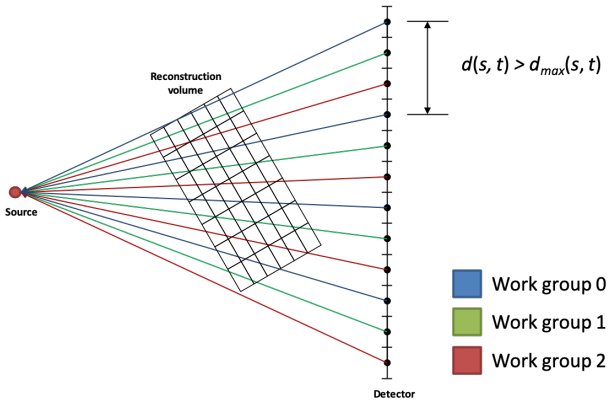


Figure 4: Backprojection with pixel spacing work group assignment. *We implement conflict-free parallel BP without low-level synchronization by assigning BP tasks to threads such that no two rays within the group pass through the same voxel. Here, the pixel spacing used in work group assignment, $d(s,t)$, is greater than the maximum distance on the detector plane between any two rays that pass through the same voxel, $d_{max}(s,t)$.*

voxels to determine the pixel spacing distance, $d(s,t)$. BP tasks are then assigned to threads in the manner described above.

Finally, we observe that, for any particular source-detector orientation, the set of voxels generating maximum distances will necessarily incorporate at least one voxel along an edge of the reconstruction volume. As such, potentially significant computational efficiency is gained by projecting only edge voxels, rather than every voxel in the reconstruction volume, to compute these distances.

Thus, by carefully assigning BP tasks to simultaneously executing threads such that no two rays within the group pass through the same voxel, we can implement conflict-free parallel BP without low-level synchronization mechanisms.

We exploit these observations in the implementation of the prototype XCT reconstruction system described below.

3.1. Implementation

We implement the FP and BP phases of a prototype XCT reconstruction system based on ART [GBH70] using thread-level parallelism expressed with OpenMP [DM98, Ope15]. ART has an intuitive formulation that computes ray-sums directly, and the algorithms serves as the basis of more advanced algebraic and statistical iterative reconstruction techniques—our prototype system can thus be extended for more advanced applications, if necessary.

Likewise, OpenMP provides an open, portable, and widely supported mechanism for expressing parallelism at different granularities; we leverage thread-level parallelism in the prototype implementation described here, but other approaches, e. g., vector processing via SIMD operations, or even a combined vectorized/threaded approach, could be used to exploit the observations described above.

We summarize each of the FP and BP implementations below. To support reproducibility and further experimentation, we also provide the full source code for our prototype XCT system online; please see the supplemental materials accompanying this paper for more information about accessing the code.

Serial FP (sFP). The *sFP* approach employs a simple computation scheme in which ray-sums are computed by serially propagating rays corresponding to each detector pixel (s,t) through the reconstruction volume.

This scheme does not leverage parallelism in any way, and represents the baseline FP metric for scaling performance.

Parallel FP (pFP). As noted, ray-sum computations in FP are trivially parallel, so the *pFP* scheme leverages the OpenMP *parallel for* construct to exploit pixel-level parallelism over the ray-sums corresponding to each detector pixel (s,t).

Our current implementation simply assigns FP tasks to threads in a straightforward nested-loop order (inner loop over s, outer loop over t). While more sophisticated task/thread mappings using a traversal order based on image tiles or z-order curves, for example, might provide better *pFP* performance, FP is not the computational bottleneck in reconstruction, so exploration of alternative task/thread mappings in *pFP* is left as future work.

pFP with per-voxel mutex pixel spacing prepass (pvmFP). This method computes the distances for pixel spacing work group assignment used in subsequent BP operations, $d(s,t)$ and $d_{max}(s,t)$, by tracking the minimum and maximum detector pixel coordinates (s,t) of rays passing through each voxel during FP. This operation imposes an update, so per-voxel mutexes are used to ensure correctness.

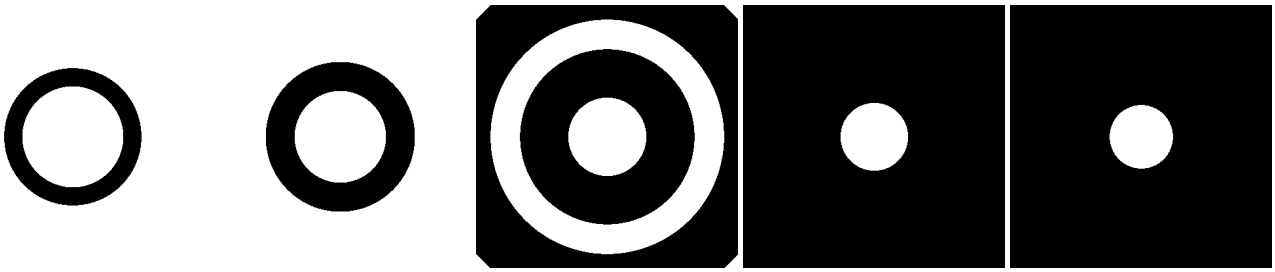
This scheme leverages the OpenMP *parallel for* construct to exploit pixel-level parallelism during FP, uses low-level synchronization primitives (per-voxel mutexes) to ensure correct updates, and supports the parallel BP schemes that build on the conflict-free pixel spacing work group assignment concept.

pFP with voxel projection pixel spacing prepass (vpaFP). This approach determines conflict-free task/thread mappings for subsequent BP operations using voxel projection: here, the corners of each reconstruction volume voxel (x,y,z) are projected to the detector plane, and the maximum projection extents are tracked across all voxels. BP tasks are then assigned to threads in the manner described above.

Again, this tracking operation itself is an update, but a lightweight per-voxel data structure can be used to exploit loop-level parallelism over the voxels (x,y,z) during projection. Determining the final maximum distance requires reduction over the per-voxel extents, which is relatively inexpensive.

This scheme leverages the OpenMP *parallel for* construct to exploit loop-level parallelism both during FP and during voxel projection, reduces per-voxel extents to the minimum/maximum values using a simple serial loop over all voxels, and supports the parallel BP schemes that build on the conflict-free work group assignment concept.

pFP with edge-only voxel projection pixel spacing prepass (vpeFP). As noted above, the set of voxels generating maximum



(a) z-axis slices through the *sinewave* phantom dataset

(b) Simulated 2 MP XCT *sinewave* projections

Figure 5: The *sinewave* XCT phantom. *Several reconstruction volume resolutions, from 100^3 to 800^3 voxels, are used to test the scalability and performance of our parallelization strategies. The images in this figure depict slices through the z-axis (a) and simulated XCT projections with a 2 MP detector (b) using the 400^3-voxel* sinewave *phantom.*

distances will necessarily incorporate at least one voxel along an edge of the reconstruction volume, so significant computational efficiency is gained by projecting only edge voxels to compute these distances. In this approach, the corners of only edge voxels (x,y,z) are projected to the detector plane, and the maximum projection extents are tracked across all active voxels. BP tasks are then assigned to threads in the manner described above.

This scheme leverages the OpenMP *parallel for* construct to exploit loop-level parallelism both during FP and during voxel projection, projects only edge voxels for efficiency, reduces per-voxel extents to the minimum/maximum values using a simple serial loop over all voxels, and supports the parallel BP schemes that build on the conflict-free work group assignment concept.

We note that even though per-voxel extents corresponding to interior voxels are null in this scheme, the reduction operation is so inexpensive that the implementation complexity necessary to minimize the set of per-voxel extents stored and compared during reduction does not present a reasonable tradeoff. As such, we store null extents and reduce across all voxels, as in the *project-all-voxels* approach, despite computing valid extents for only edge voxels.

Serial BP (sBP). The *sBP* approach employs a simple computation scheme in which reconstruction volume values are computed by serially backpropagating ray-sum values for each voxel (x,y,z) through which each ray passes.

This scheme does not leverage parallelism in any way, and represents the baseline BP metric for scaling performance.

Parallel BP with per-voxel mutexes (pvmBP). In this approach, per-voxel mutexes ensure correct update operations by serializing access to the per-voxel ray-sum values: rays corresponding to each detector pixel (s,t) must first acquire the lock for each voxel (x,y,z) through which they pass, then update the voxel ray-sum value, and finally release the lock.

This scheme leverages the OpenMP *parallel for* construct to exploit pixel-level parallelism during BP but uses low-level synchronization primitives (per-voxel mutexes) to ensure correct updates for per-voxel ray-sums.

Parallel BP with pixel spacing work group assignment (psBP). This approach assumes the maximum distance between any two rays that pass through the same reconstruction voxel (x,y,z) across the entire detector plane, $d_{max}(s,t)$, is available prior to work group assignment, and is thus paired with the parallel FP pixel spacing prepass techniques described above. BP tasks are then assigned to threads such that the pixel spacing distance, $d(s,t)$, is greater than $d_{max}(s,t)$.

This scheme leverages the OpenMP *parallel for* construct to exploit pixel-level parallelism during BP and employs pixel spacing work group assignment to ensure correct updates for per-voxel ray-sums.

3.2. Performance

Using the implementations described above, we execute an initial performance study to explore the impact of thread count and reconstruction volume resolution on performance.

Experimental Setup. We use a synthetic XCT phantom dataset, *sinewave*, for this study. Figure 5a depicts several slices through the *sinewave* dataset; in these images, the volume comprises a 1.2 m^3 spatial extent with 400^3 voxels, corresponding to a 3 mm^3 feature size. Similarly, Figure 5b depicts several simulated XCT projections of the *sinewave* dataset; here, the projections capture the 400^3-voxel volume using a 2 MP, or 1600×1200 pixel, detector.

Results are obtained on several systems with various hardware configurations:

- *Test Platform #0* (*TP0*)—a Debian 8.11 system with two Intel Xeon E5-2699 v3 2.30 GHz processors (36 cores, 72 hardware threads), 64 GB of RAM, and the *GNU Compiler Collection* (GCC) 7.3.0.
- *Test Platform #1* (*TP1*)—an Ubuntu 18.04 system with two Intel Core i7-7820X 3.60 GHz processors (8 cores, 16 hardware threads), 64 GB of RAM, and GCC 7.4.0.
- *Test Platform #2* (*TP2*)—an Ubuntu 16.04 system with two Intel Core i7-7800X 3.50 GHz processors (6 cores, 12 hardware threads), 64 GB of RAM, and GCC 5.4.0.

We note that *TP0* provides more but lower frequency cores, whereas *TP1* and *TP2* provide fewer but higher frequency cores.

We scale the system from one to the maximum number of hardware threads for each of the following $FP+BP$ combinations:

- $pFP+sBP$
- $pFP+pvmBP$
- $pvmFP+psBP$
- $vpaFP+psBP$
- $vpeFP+psBP$

For each $FP+BP$ combination and thread count, we test reconstruction volumes comprising a 1.2 m^3 spatial extent at the following spatial resolutions:

- 100^3 voxels; feature size, 12.2 mm^3
- 200^3 voxels; feature size, 6.1 mm^3
- 400^3 voxels; feature size, 3.0 mm^3
- 800^3 voxels; feature size, 1.5 mm^3

For each experiment in this study, we use a 2 MP detector, 16 simulated projections in a full-ring configuration at 22.5° increments, and 10 ART iterations.

Results. For brevity, we report results for only *TP0* here; however, the supplemental data accompanying this paper includes scaling results for *TP1* and *TP2* as well. Generally speaking, we observe the same trends on these platforms as for *TP0*.

In particular, we report performance in projections/second across each $FP+BP$ combination and thread count for each reconstruction volume resolution in Figures 6–9. In these figures, *serial* indicates performance for the baseline serial combination, $sFP+sBP$.

As seen in Figures 6 and 7, most parallelization strategies provide at least some improvement over the serial case for 9–72 threads. In particular, strategies employing per-voxel mutexes ($pFP+pvmBP$, $pvmFP+psBP$) outperform the serial case by a factor of 1.2–4.3×, with $pFP+pvmBP$ performing as the better of the two. Interestingly, projecting all voxels ($vpaFP+psBP$) does not perform well, barely exceeding (100^3 voxels) or underperforming (200^3 voxels) the serial baseline. The sheer number of voxels makes this approach unacceptably expensive, even for these relatively low-resolution reconstruction volumes. In contrast, edge-only voxel projection ($vpeFP+psBP$) provides the best absolute performance for 100^3- and 200^3-voxel reconstruction volumes (very nearly 4 projections/seconds in each case), outperforms the serial baseline as much as 6×, and provides an additional 1.2–1.4× performance over the next-best strategy ($pFP+pvmBP$).

The data in Figure 8 also show some improvement for the 400^3-voxel reconstruction volume over the serial case for most strategies when using 9–72 threads. As before, projecting all voxels ($vpaFP+psBP$) underperforms the serial case, though this time by as much as 30%. With this approach, projection time is directly proportional to the total number of voxels, so the already-expensive voxel projection step simply becomes even more costly with this higher-resolution reconstruction volume. And as before, edge-only voxel project performs well, outperforming the serial baseline by more than 6×. Interestingly, strategies employing per-voxel mutexes ($pFP+pvmBP$, $pvmFP+psBP$) become more attractive in this case. For a given detector size and volume extent, the relative projected size of each voxel decreases as the number of voxels increases, which in turn reduces contention for the corresponding per-voxel mutexes across rays. In fact, $pFP+pvmBP$ provides the best absolute performance—about 1.5 projections/second—for the 400^3-voxel reconstruction volume.

Performance trends are similar for the 800^3-voxel reconstruction volume, as shown Figure 9: most strategies provide at least some improvement over the serial baseline (1.5–9.9×), and projecting all voxels ($vpaFP+psBP$) simply does not scale. Moreover, these results indicate that $pFP+pvmBP$ performs best for high-resolution reconstruction volumes: as in the 400^3-voxel case, for a given detector size and volume extent, the relative projected size of each voxel decreases as the number of voxels increases, which once again reduces contention for the corresponding per-voxel mutexes across rays. Here, too, $pFP+pvmBP$ provides the best absolute performance (about 0.78 projections/second), outperforms the serial baseline by almost 10×, and provides an additional 1.5× performance over the next-best strategy ($vpeFP+psBP$).

We note that absolute performance begins to lag behind the target performance goals with 800^3-voxel reconstruction volumes on *TP0* for all but $pFP+pvmBP$ with 36 or more threads. (Similarly, for 800^3-voxel reconstruction volumes, neither *TP1* nor *TP2* provide performance sufficient for our goals with any strategy at any thread count.) Nevertheless, $pFP+pvmBP$ scales well with thread count, so a machine supporting more hardware threads could be used to further reduce processing time for 800^3-voxel reconstruction volumes. We thus anticipate that additional computing resources will enable reconstruction performance satisfying our near real-time constraint with higher resolution reconstruction volumes, should the requirement for 1.5 mm^3 (or smaller) features arise in future imaging scenarios.

Furthermore, when considering results for *TP1* and *TP2*, we see that processor frequency also impacts overall performance. For example, the best performing strategy for 100^3-voxel reconstruction volumes on *TP0* ($vpeFP+psBP$ with 36 threads) actually underperforms the same strategy with only 16 threads on *TP1*: 3.85 projections/second vs. 4.04 projections/second. This result suggests that the necessarily serial operations along any one ray—e.g., ray-sum computations in forward projection—benefit from lower per-operation cost on higher frequency cores. Though this result does not hold for higher-resolution volumes (200^3 or more voxels) on our test platforms, fewer but higher frequency cores may nevertheless provide better absolute performance for imaging scenarios involving low-resolution reconstruction volumes.

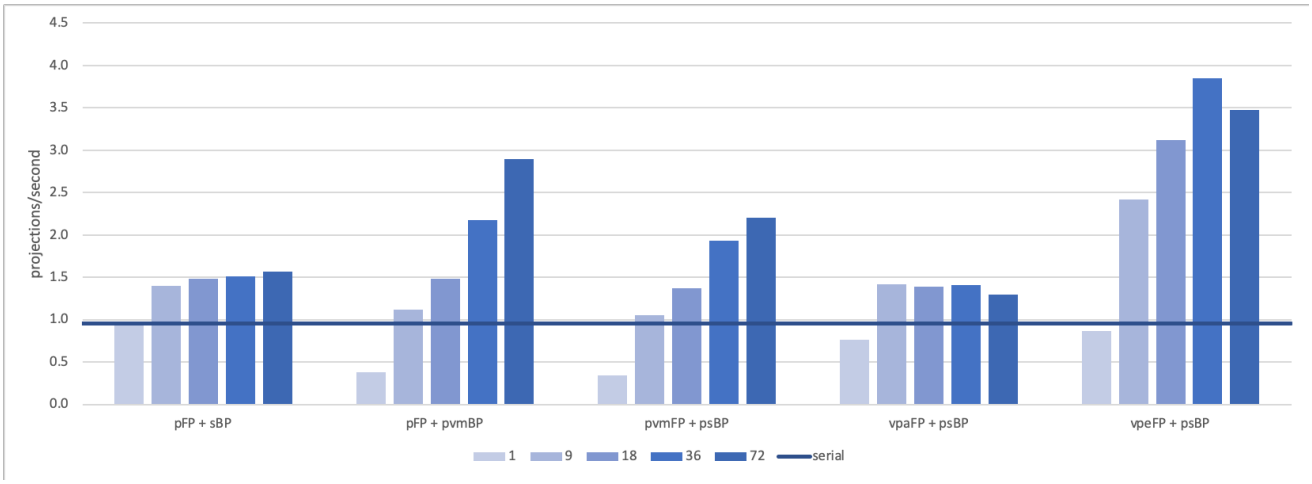


Figure 6: Reconstruction performance with 100^3-voxel reconstruction volume. *All five parallelization strategies outperform the serial baseline when using 9–72 threads, with* vpeFP + psBP *performing best overall and* pFP + pvmBP *showing promise.*

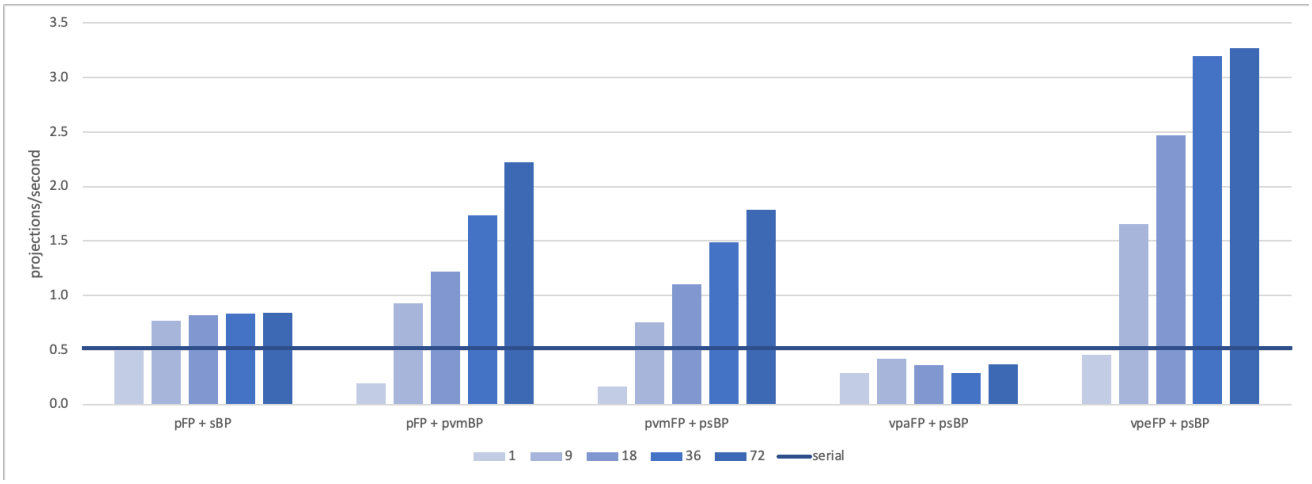


Figure 7: Reconstruction performance with 200^3-voxel reconstruction volume. *As in the 100^3-voxel case,* vpeFP + psBP *performs best overall, while projecting all voxels (*vpaFP + psBP*) is simply too costly for even this relatively low-resolution reconstruction volume.* pFP + pvmBP *also shows promise for this 200^3-voxel reconstruction volume.*

Finally, we observe that several parallelization strategies already meet or exceed the target performance goals for 400^3-voxel reconstruction volumes on all three of our test platforms—in some cases, with fewer-than-all threads. Here, the results in Figure 8 show that *pFP + pvmBP*, *pvmFP + psBP*, and *vpeFP + psBP* process at least 0.6 projections/second for a 1.2 m^3 volume with a 3 mm^3 feature size on *TP0*, while similar results are also observed for *TP1* and *TP2* in the supplemental data.

The results of this study demonstrate excellent scalability through the maximum number on each test platform, particularly with the *pFP + pvmBP* strategy. The best-performing combination depends not only on reconstruction volume resolution and thread count, but also on processor frequency, so the characteristics of any particular imaging scenario can be used to guide the appropriate choice at runtime.

4. Conclusion and Future Work

The high computational demand of iterative reconstruction techniques represents a key opportunity for the application of modern parallel computing hardware and software to XCT reconstruction. In this work, we design and implement several parallel iterative reconstruction strategies and evaluate their performance on modern workstations. The methodology development strategy we employ quickly uncovers effective parallel computing techniques that are used to accelerate iterative reconstruction on these platforms.

In particular, results show that several parallelization strategies

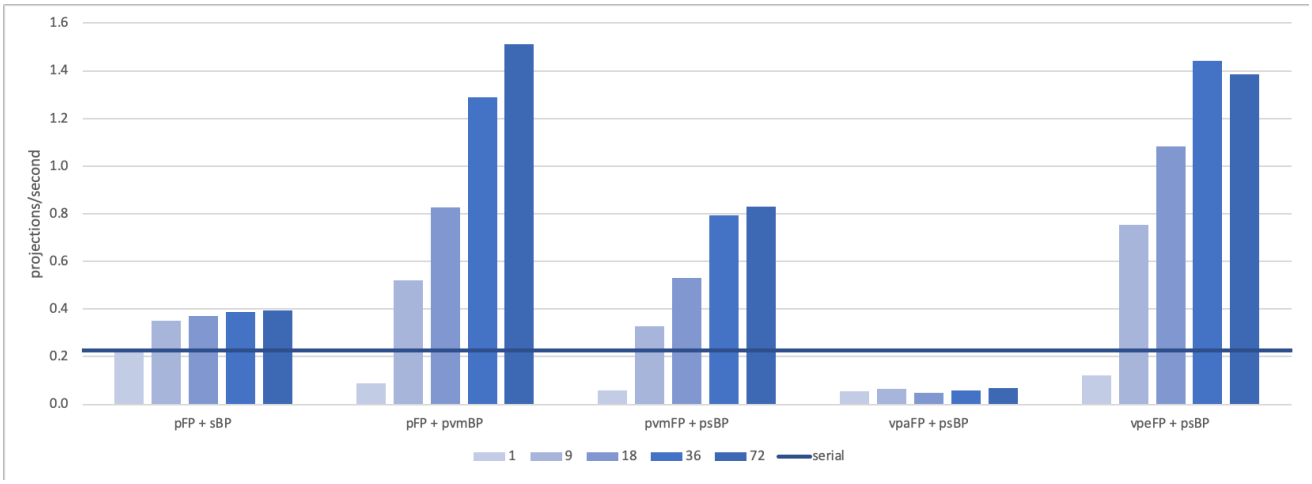


Figure 8: Reconstruction performance with 400^3-voxel reconstruction volume. *Here, projecting all voxels (*vpaFP + psBP*) continues to perform poorly, while* pFP + pvmBP *becomes more attractive, outperforming* vpeFP + psBS *by about 5–10%.*

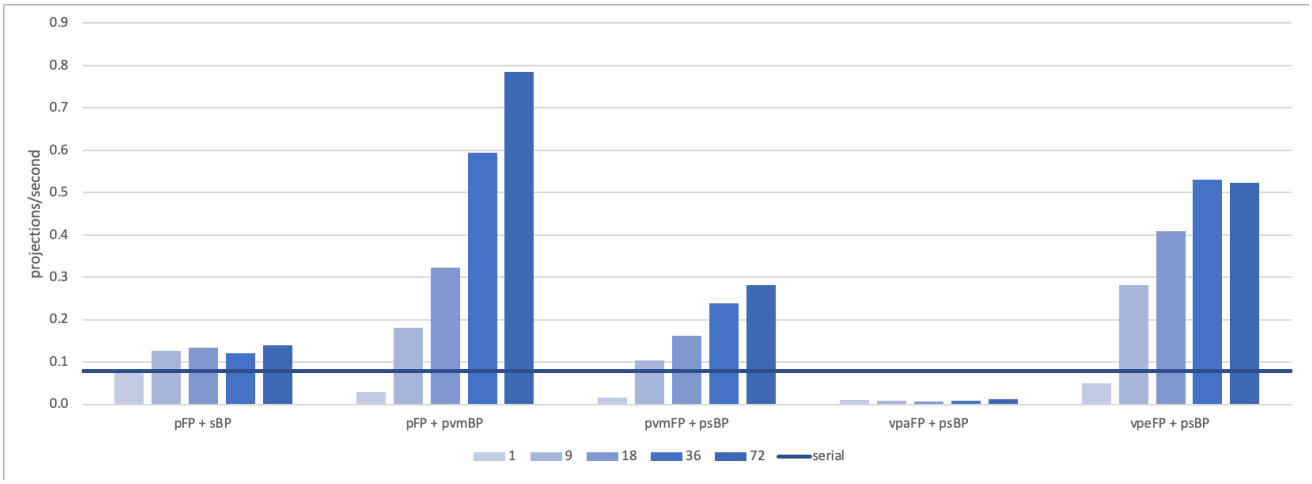


Figure 9: Reconstruction performance with 800^3-voxel reconstruction volume. *As with the lower-resolution reconstruction volumes, projecting all voxels (*vpaFP + psBP*) simply does not scale. However,* pFP + pvmBP *remains viable in situations involving high-resolution reconstruction volumes, achieving nearly 0.8 projections/second with 72 threads.*

process at least 80 high-resolution projections in less than two minutes for 1.2 m^3 volumes with 3 mm^3 feature size. These results demonstrate the feasibility of an affordable, scalable, and accelerated visualization pipeline to guide decision-making in near real-time for our target application.

There are many practical implications for medical and industrial imaging if the computational challenges facing iterative reconstruction techniques are overcome. For example, robustness to sparse projection data provides the opportunity to use fewer acquisition steps in our target application (where data capture is costly in both time and resources) while keeping overall reconstruction quality high. Likewise, accurate reconstruction with sparse data could reduce scan times in medical settings by requiring fewer angular samples to accurately image patients. Similarly, these techniques promise to reduce patient exposure to potentially harmful X-ray radiation by permitting accurate reconstructions from noisy data acquired by low-dose CT scanners.

We plan to continue development of the iterative reconstruction engine for the prototype XCT system described here. The family of algebraic reconstruction techniques, in particular, provides an opportunity to apply modern, high performance ray tracing techniques to XCT reconstruction. Use of ray tracing in this context—specifically the computation of radiological paths for forward projection—stems from early work by Joseph [Jos83] and Siddon [Sid85] that initiated a line of inquiry to optimize ray/grid traversal [JSS*98, HLY99, ZR03] in XCT reconstruction. In fact, the work to accelerate ray/grid traversal in this context parallels ray tracing research in computer graphics over the past 30 years.

However, it is unclear that additional optimizations from computer graphics—for example, packet-based ray tracing—have been adapted for XCT reconstruction. The application of advanced techniques from computer graphics—for example, coherent grid traversal [WIK*06]—to XCT reconstruction represents one avenue of future work.

More immediately, additional performance improvements may be possible with vector processing by applying the parallelization strategies outlined here on a per-lane, rather than per-thread, basis. We plan to explore a combined vectorized/threaded implementation—parallel concepts both expressible in OpenMP (version 4.0 or better)—in our XCT reconstruction prototype. Similarly, a vectorized implementation in CUDA, for example, could be used to exploit our parallelization strategies for GPU-accelerated iterative reconstruction.

Acknowledgments

We gratefully acknowledge the contributions of Jefferson Amstutz (Intel Corporation), Beto Castelo (Chatham Financial), and Chris Peitsch (Johns Hopkins University Applied Physics Laboratory) to this work.

This work is funded in part by Intel Corporation and by a US Air Force Small Business Innovation Research Program grant.

References

[BAL*13] BLAS J., ABELLA M., LIRIA E., ISAILA F., CARRETERO J., DESCO M.: Parallel implementation of X-ray tomography reconstruction algorithm based on MPI and CUDA. In *Proceedings of the 20th European MPI Users' Group Meeting* (September 2013). 3

[BDHS16] BIGURI A., DOSANJH M., HANCOCK S., SOLEIMANI M.: TIGRE: a MATLAB-GPU toolbox for CBCT image reconstruction. *Biomedical Physics and Engineering Express 2* (2016). 4

[BKK12] BEISTER M., KOLDITZ D., KALENDER W. A.: Iterative reconstruction methods in X-ray CT. *Physica Medica 28*, 2 (2012), 94–108. 4

[BLB*19] BIGURI A., LINDROOS R., BRYLL R., TOWSYFYAN H., DEYHLE H., BOARDMAN R., MAVROGORDATO M., DOSANJH M., HANCOCK S., BLUMENSATH T.: Arbitrarily large iterative tomographic reconstruction on multiple GPUs using the TIGRE toolbox. *arXiv preprint arXiv:1905.03748v1 [cs.DC]* (2019). 4

[Bru02] BRUYANT P. P.: Analytic and iterative reconstruction algorithms in SPECT. *Journal of Nuclear Medicine 43*, 10 (2002), 1343–1358. 2, 3

[CCHW11] CHOU C.-Y., CHUO Y.-Y., HUNG Y., WANG W.: A fast forward projection using multithreads for multirays on GPUs in medical image reconstruction. *Medical Physics 38*, 7 (2011), 4052–4065. 3

[CSB*99] CHRISTAENS M., SUTTER B. D., BOSSCHERE K. D., CAMPENHOUT J. V., LEMAHIEU I.: A fast, cache-aware algorithm for the calculation of radiological paths exploiting subword parallelism. *Journal of Systems Architecture 45*, 10 (1999), 781–790. 3

[DM98] DAGUM L., MENON R.: OpenMP: an industry standard API for shared-memory programming. *IEEE Computational Science and Engineering 5*, 1 (1998), 46–55. 5

[dMSGB*18] DE MOLINA C., SERRANO E., GARCIA-BLAS J., CARRETERO J., DESCO M., ABELLA M.: GPU-accelerated iterative reconstruction for limited-data tomography in CBCT systems. *BMC Bioinformatics 19* (2018), 171. 4

[FDK84] FELDKAMP L. A., DAVIS L. C., KRESS J. W.: Practical cone-beam algorithm. *Journal of the Optical Society of America A 1*, 6 (1984), 612–619. 3

[FVM*13] FLORES L., VIDAL V., MAYO P., RODENAS F., VERDU G.: CT image reconstruction based on GPUs. *Procedia Computer Science 18* (2013), 1412–1420. 3

[GBH70] GORDON R., BENDER R., HERMAN G. T.: Algebraic reconstruction techniques (ART) for three-dimensional electron microscopy and X-ray photography. *Journal of Theoretical Biology 29*, 3 (1970), 471–481. 3, 5

[HLY99] HAN G., LIANG Z., YOU J.: A fast ray-tracing technique for TCT and ECT studies. In *IEEE Nuclear Science Symposium* (October 1999), pp. 1515–1518. 9

[Jos83] JOSEPH P. M.: An improved algorithm for reprojecting rays through pixel images. *IEEE Transactions on Medical Imaging MI-1*, 1 (1983), 192–196. 9

[JSS*98] JACOBS F., SUNDERMANN E., SUTTER B. D., CHRISTIAENS M., LEMAHIEU I.: A fast algorithm to calculate the exact radiological path through a pixel or voxel space. *CIT. Journal of computing and information technology 6*, 1 (1998), 89–94. 9

[KS88] KAK A. C., SLANEY M.: *Principles of Computerized Tomographic Imaging*. IEEE Press, New York, NY, 1988. 2

[MMBL12] MUKHERJEE S., MOORE N., BROCK J., LEESER M.: CUDA and OpenCL implementations of 3D CT reconstruction for biomedical imaging. In *High Performance Extreme Computing* (September 2012), pp. 1–6. 3

[NJL13] NGUYEN V.-G., JEONG J., LEE S.-J.: GPU-accelerated iterative 3D CT reconstruction using exact ray-tracing method for both projection and backprojection. In *IEEE Nuclear Science Symposium and Medical Imaging Conference* (October-November 2013), pp. 1–4. 3

[NL15] NGUYEN V.-G., LEE S.-J.: Parallelizing a matched pair of ray-tracing projector and backprojector for iterative cone-beam CT reconstruction. *IEEE Transactions on Nuclear Science 62*, 1 (2015), 171–181. 3

[NLHW06] NI J., LI X., HE T., WANG G.: Review of parallel computing techniques for computed tomography image reconstruction. *Current Medical Imaging Reviews 2*, 4 (2006), 405–414. 4

[Ope15] OPENMP ARCHITECTURE REVIEW BOARD: OpenMP Application Program Interface Version 4.5, November 2015. URL: https://www.openmp.org/wp-content/uploads/openmp-4.5.pdf. 5

[Sid85] SIDDON R. L.: Fast calculation of the exact radiological path for a three-dimensional CT array. *Medical Physics 12*, 2 (1985), 252–255. 9

[SL74] SHEPP L., LOGAN B. F.: The Fourier reconstruction of a head section. *IEEE Transactions on Nuclear Science NS-21*, 3 (1974), 21–43. 3

[THH*13] TREIBIG J., HAGER G., HOFMANN H., HORNEGGER J., WELLEIN G.: Pushing the limits for medical image reconstruction on recent standard multicore processors. *International Journal of High Performance Computing Applications 27*, 2 (2013), 162–177. 4

[Tur01] TURBELL H.: *Cone-Beam Reconstruction Using Filtered Backprojection*. PhD thesis, Linköpings University, 2001. 3

[VVHV13] VANHOVE C., VANDEGHINSTE B., HOLEN R. V., VANDENBERGHE S.: Performance evaluation of GPU implementations of four different iterative reconstruction algorithms for micro-computer tomography. *Journal of Nuclear Medicine 54*, 2 (2013). 4

[WIK*06] WALD I., IZE T., KENSLER A., KNOLL A., PARKER S. G.: Ray tracing animated scenes using coherent grid traversal. *ACM Transactions on Graphics 25*, 3 (July 2006), 485–493. 10

[YNC01] YU R., NING R., CHEN B.: High speed cone beam reconstruction on PC. *Proceedings of SPIE 4322 Medical Imaging 2001*, Image Processing (2001), 964–973. 3

[ZR03] ZHAO H., READER A. J.: Fast ray-tracing technique to calculate line integral paths in voxel arrays. In *IEEE Nuclear Science Symposium* (November 2003), pp. 2808–2812. 9