Investigating Multi-threaded SIMD for Helical CT Reconstruction on a CPU

Richard Sampson, Madison G. McGaffin, Thomas F. Wenisch, Jeffrey A. Fessler
Department of EECS, University of Michigan

***Abstract*—Iterative reconstruction for X-ray CT is computationally expensive, so it is desirable to examine acceleration methods such as algorithm design, software implementation, and computing hardware. This paper explores using single-instruction, multiple data (SIMD) operations on modern CPUs to accelerate projection and back-projection using the separable footprint (SF) method. Slightly modifying the axial footprint calculation facilitates SIMD implementation, providing up to 5× acceleration using 8-wide SIMD with Intel AVX2 instructions over multi-threading (MT) alone. Due to memory bandwidth constraints, overall speedup saturates at ≈55× faster than a single-thread, non-SIMD version (still 2× faster over MT with 72 threads). Despite the bandwidth limits, the MT+SIMD runtimes are competitive with corresponding GPU versions.**

I. Introduction

Model-based iterative reconstruction (MBIR) for X-ray CT has improved image quality and reduced X-ray dose compared to filtered back-projection [1]. However, MBIR's high computational requirements have led researchers to explore acceleration techniques to make it more practical for routine clinical use. Efforts to reduce computational requirements and speed convergence have shown great progress; nevertheless, the computation requirement still remains undesirably high.

One method for mitigating the high complexity is exploiting parallel computation. Previous work has achieved significant acceleration by using parallelism both in distributed systems in the cloud [2, 3] as well as locally on GPUs [4, 5]. However, these techniques are tuned to specific hardware platforms and can be difficult to adapt to new platforms. There has been less study of the enhanced capabilities of modern CPUs that support both higher thread counts and SIMD programming, allowing for even more parallelism on a single chip [6, 7]. SIMD instruction set extensions (e.g., Intel's AVX2) allow a single instruction to perform element-wise operations (e.g., 8 single-precision floating-point values) concurrently. The main challenge in exploiting SIMD lies in orchestrating memory layout, as the instructions are efficient only when accessing contiguous memory locations.

This work investigates using modern CPUs in MBIR for X-ray CT, focusing on the increased parallel performance enabled by the latest SIMD extensions. We describe reconstruction algorithm modifications that facilitate SIMD programming and examine the bandwidth limitations of combining SIMD with multi-threading. We also explore the performance of high parallelism on modern CPUs with and without SIMD in comparison to GPU-based reconstruction.

Supported in part by NIH grant U01 EB018753 and Intel equipment donations. {rsamp | mcgaffin | fessler | twenisch} @umich.edu

II. Methods

Consider the following MBIR problem for X-ray CT [1]:

$$\widehat{\mathbf{x}} = \arg\min_{\mathbf{x}\geq\mathbf{0}} \Psi(\mathbf{x}), \qquad \Psi(\mathbf{x}) = \frac{1}{2}\,||\mathbf{A}\mathbf{x}-\mathbf{y}||^2_{\mathbf{W}} + \mathsf{R}(\mathbf{x}), \quad (1)$$

with X-ray CT system matrix $\mathbf{A} \in \mathbb{R}^{M\times N}$, noisy data $\mathbf{y}$, diagonal matrix of statistical weights $\mathbf{W}$ and convex edge-preserving regularizer R. The large dimension of $\mathbf{x}$, the often nonquadratic regularizer, the nonnegativity constraint, and the space-varying nature of the Hessian of Ψ make (1) challenging.

This paper accelerates primal gradient-based methods, *e.g.* [8, 9]. These methods perform an update of the form:

$$\mathbf{x}^{(n+1)} = \left[\mathbf{x}^{(n)} - \left[\mathbf{D}^{(n)}\right]^{-1}\mathbf{g}^{(n)}\right]_+; \quad (2)$$

where $\mathbf{D}^{(n)}$ is a diagonal majorizer [8]; $\mathbf{g}^{(n)}$ approximates the gradient of Ψ in (1) at the current iterate, $\mathbf{x}^{(n)}$; and $[\cdot]_+$ enforces the nonnegativity constraint. Iterative algorithms like (2) that update all the voxels of $\mathbf{x}$ simultaneously can exploit the increasing parallelism in modern computing hardware.

The gradient-approximating term $\mathbf{g}^{(n)}$ is often computed with an ordered-subsets (OS) approximation:

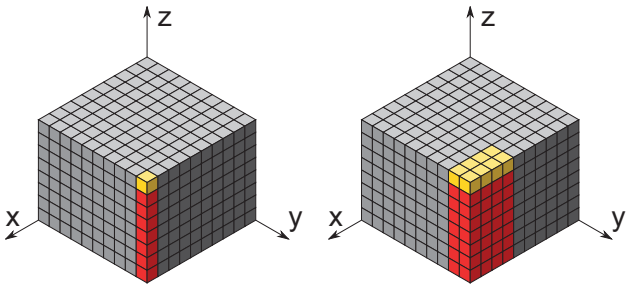
$$\begin{aligned}\mathbf{g}^{(n)} &= \nabla\mathsf{R}\left(\mathbf{x}^{(n)}\right) + \frac{N_{\text{view}}}{|\mathscr{S}_n|}\sum_{v\in\mathscr{S}_n}\mathbf{A}_v^\intercal\mathbf{W}_v\left(\mathbf{A}_v\mathbf{x}^{(n)}-\mathbf{y}_v\right)\\ &\approx \nabla\mathsf{R}\left(\mathbf{x}^{(n)}\right) + \mathbf{A}^\intercal\mathbf{W}\left(\mathbf{A}\mathbf{x}^{(n)}-\mathbf{y}\right) = \nabla\Psi(\mathbf{x}_n), \quad (3)\end{aligned}$$

where $\mathscr{S}_n$ is a subset of the views in the CT system matrix [8].

The most time-consuming step in the image update (2) is computing the data-fit part of the approximate gradient $\mathbf{g}^{(n)}$ (3). For example, for an 8-turn helical scan with 7,872 views and 12 subsets, each $\mathbf{g}^{(n)}$ requires 656 single-view projections and back-projections. These computations dominate the relatively inexpensive regularizer gradient computation. Thus, we focus on accelerating the projection and back-projection in (3).

A. Separable footprints CT system model

We consider the separable footprints (SF) CT system model [10]. The SF model is a "splatting" approach that implements the product $\mathbf{A}\mathbf{x}$ by superimposing the "footprints" of each voxel: $\mathbf{A}\mathbf{x} = \sum_{j=1}^{N}\mathbf{a}_j x_j$. Each 2D footprint is (ap-



(a) Multi-threaded (b) SIMD

Fig. 1: Data Structure Layout: Data varies fastest along vertical (z) axis. Yellow blocks are data accessed in a single memory operation, and red blocks are future accesses in the entire iteration of forward or back projection loop. (a) Data layout and access of original multi-threaded code. (b) SIMD layout and access. Data needed per access spans across multiple columns, which would require expensive gather operations. We eliminated strided accesses by adjusting the mapping of image coordinates to memory addresses to densely pack each cluster of values together in memory, resulting in sequential accesses.

proximated by) a separable product of two functions, and the elements of $\mathbf{a}_j$ use 2D integrals of this function:

$$a_{ij} = r_i v_j \left(\int_{s \in S_i} g_{ij}(s) \; ds \right) \left(\int_{t \in T_i} h_{ij}(t) \; dt \right), \qquad (4)$$

where r_i and v_j are ray and voxel weights, respectively. We use a trapezoidal function g_{ij} in the channel (transaxial) direction and a rectangular function h_{ij} in the row (axial) direction; this corresponds to the "SF-TR" approximation detailed in [10].

Our implementation of the SF system model for projecting into a single view v is represented mathematically as:

$$\mathbf{A}_v = \mathbf{R}_v \mathbf{S}_v \mathbf{T}_v \mathbf{V}_v, \qquad (5)$$

where $\mathbf{R}_v$ and $\mathbf{V}_v$ are diagonal matrices that apply weights to each ray and voxel, respectively. The most computationally expensive operations are the multiplications with the separable matrices $\mathbf{S}_v$ and $\mathbf{T}_v$ that implement the s and t integrals of (4), respectively. Conceptually, $\mathbf{T}_v \in \mathbb{R}^{N_t N_x N_y \times N}$ and $\mathbf{S}_v \in \mathbb{R}^{N_s N_t \times N_t N_x N_y}$, although our implementation does not store the intermediate $N_t N_x N_y$-element vector. Our single-view back-projection implementation follows the transpose of (5): $\mathbf{A}_v^\intercal = \mathbf{V}_v \mathbf{T}_v^\intercal \mathbf{S}_v^\intercal \mathbf{R}_v$.

The ray and voxel scaling operations $\mathbf{R}_v$ and $\mathbf{V}_v$ are trivial to parallelize with SIMD, so we focus on the more difficult channel and row operators. The next few sections describe accelerating the projector; the back-projector is similar.

B. Existing implementation

We modify an existing projector that implements $\mathbf{A}_v \mathbf{x}$ as:

$$\mathbf{A}_v \mathbf{x} = \mathbf{R}_v \sum_{xy} \mathbf{S}_{v,xy} \mathbf{T}_{v,xy} \mathbf{V}_{v,xy} \mathbf{x}_{xy}. \qquad (6)$$

This is, the algorithm loops over each axial xy-column and applies the volume weights for that column ($\mathbf{V}_{v,xy}$), applies the footprints along the axial t/z direction ($\mathbf{T}_{v,xy}$), applies the footprints along the transaxial s direction ($\mathbf{S}_{v,xy}$), accumulating into a buffer where it applies the ray weights $\mathbf{R}_v$. Fig. 1(a) illustrates this behavior: the algorithm serially processes each red-colored xy-column of $\mathbf{x}$. We obtain parallelism across CPU cores by processing different views on each core.

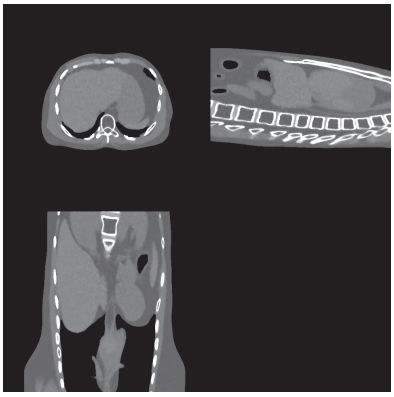


Fig. 2: Central slices of the reconstructed XCAT phantom with the conventional separable footprints system model. Images displayed on a $[800, 1200]$ modified Hounsfeld unit scale where air is 0 HU.

Fig. 3: Difference along central slices between the two reconstructed images, displayed on a [-20, 20] modified Hounsfeld unit window.

C. Modifying SF to suit SIMD

SIMD instructions require that an identical sequence of element-wise operations be performed on the vector operands of each instruction. A naive SIMD approach might process multiple elements of $\mathbf{x}_{xy}$ simultaneously. However, the axial footprint operation $\mathbf{T}_{v,xy}$ is heterogeneous within each xy-column, because the axial footprint of each successive voxel intersects a varying number of detector cells in a cone-beam CT geometry. Hence, a small loop with a trip count varying in z is needed to calculate the contribution of each voxel to each cell, thwarting SIMD efficiency.

To improve SIMD efficiency, we perform SIMD operations over a rectangular region of eight adjacent columns (as in Fig. 1(b)), projecting an xy-patch of 8 voxels with identical z. Our intuition is that the axial footprints of neighboring voxels in a small xy patch are all very similar, enabling an efficient SIMD loop. Whereas the original SF method approximates the axial footprint using the centers of the top and bottom faces of each *voxel*, for our SIMD investigation we approximate the axial footprint using the centers of the top and bottom faces of each *patch*. Section III reports the impact of this approximation.

Although this SIMD approach eliminates control flow divergence, it creates a new challenge. The conventional image volume memory layout for SF has z varying fastest, for which each SIMD instruction would require voxel values that are scattered in memory. Although supported by many SIMD instruction sets, gather-type memory operations that can load non-sequentially located data are highly inefficient. Instead, we transform the memory layout to interleave the eight voxels in each patch (shown in yellow in Fig. 1(b)) consecutively in memory before advancing to the next z coordinate, allowing a regular SIMD load operation to retrieve all eight values.

By applying our SIMD optimization to both forward and back-projection, we change the coordinate-to-memory address mapping throughout the CT code (i.e., there is no need to reorganize image layout during execution). Regularization requires gathers from disparate memory locations regardless of the data layout; we simply adjust the memory address calculations for the modified layout.

III. Results

A. Effect of footprint approximation

We performed an XCAT [11] simulation to validate that the axial footprint approximation that we introduced to facilitate SIMD-friendly control flow does not cause the reconstructed images to deviate significantly from those reconstructed using the original SF system model (which itself is also based on an approximation). Recall that for SIMD we approximate the axial footprint of neighboring voxels in a 2-by-4 patch with the axial footprint corresponding to the patch center. We compared results from the SIMD reconstruction to reconstruction using the original SF algorithm.

We reconstructed a 512^3 simulated scan of an XCAT phantom [11] using a detector with 888 channels, 64 rows and 8 helix turns of 984 views each. The edge-preserving regularizer R penalized the differences between each pixel and its 26 3D neighbors using the Fair potential function,

$$\psi(d) = \delta^2 \left(\left| \frac{d}{\delta} \right| - \log\left(1 + \left| \frac{d}{\delta} \right| \right) \right), \qquad (7)$$

with $\delta = 10$ HU. Fig. 2 shows orthogonal slices from a converged solution to the MBIR problem (1).

Fig. 3 shows the difference maps between the two reconstruction methods. The reconstructed images differ slightly with 1.5 HU root mean squared difference. We believe this difference is comparable to other approximation errors incurred by the SF system model and does not significantly degrade the quality of the reconstruction.

B. SIMD acceleration

We evaluated the acceleration provided by both multi-threading (MT) and by SIMD over a single-thread, non-SIMD implementation on a dual-socket Xeon 2699 system with a total of 72 logical CPU cores (36 physical). Our results compare average runtimes of computing forward and back projections for 572 views with a $528 \times 496 \times 768$-voxel volume with the same detector geometry. We averaged 25 runs each, varying the CPU thread count from 1 to 72. For SIMD we used 8-element AVX2 floating point operations as described in Section II.

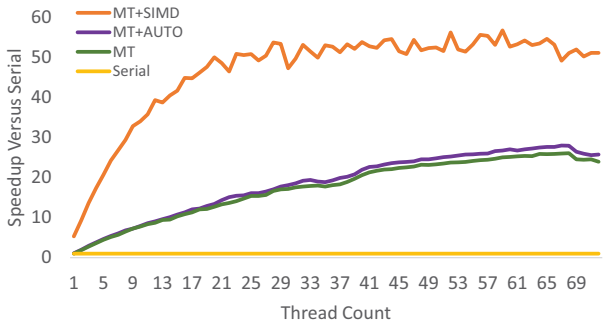


Fig. 4: Average speedup over single-thread, non-SIMD: Speedup of data-fit gradient computation for our multi-threaded SIMD (MT+SIMD) version, an auto-vectorized multi-threaded (MT+AUTO) version, and conventional non-SIMD multi-threaded (MT) version, versus a single-threaded, non-SIMD version (Serial). Times averaged over 25 runs.

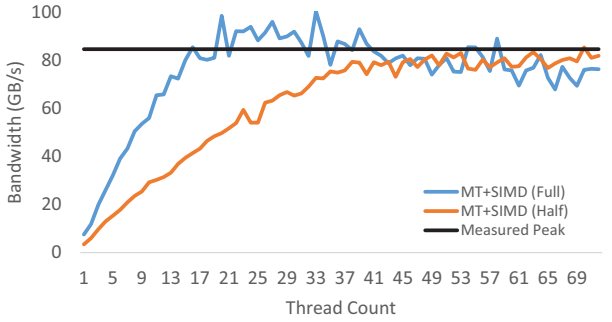


Fig. 5: Average Memory Bandwidth Consumption: Calculated bandwidth consumption for the $\mathbf{T}_v^{\mathsf{T}}$ operation in back-projection with multithreaded SIMD for 1-72 threads, each averaged over 25 runs, compared with measured peak bandwidth of the system. "Full" denotes 32-bit single-precision data, and "Half" is emulated 16-bit precision by reading/writing half of the data. Peak bandwidth measured using STREAM triad benchmark[12] with 72 threads.

Fig. 4 shows that our SIMD implementation (MT+SIMD) provides significant additional speedup over the multi-threaded baseline (MT), achieving over 5× speedup for lower thread counts. We also include the speedup achieved by automatic SIMD vectorization of the MT baseline using Intel's icc compiler (MT+AUTO), which provides minimal gains as it cannot perform the proposed algorithmic modifications and layout transformations. However, the results also show that the MT+SIMD performance saturates at roughly 25 threads, limiting any further speedup beyond 50-55x over the non-parallelized reconstruction. Nevertheless, MT+SIMD provides at least 2x speedup over the MT baseline for all thread counts.

C. Memory Bandwidth

The MT+SIMD performance saturates around 25 threads because it exhausts the available memory bandwidth in the Xeon 2699. Various phases of the CT reconstruction algorithm are memory intensive, and the concurrent accesses from many threads overwhelm the capability of the memory subsystem.

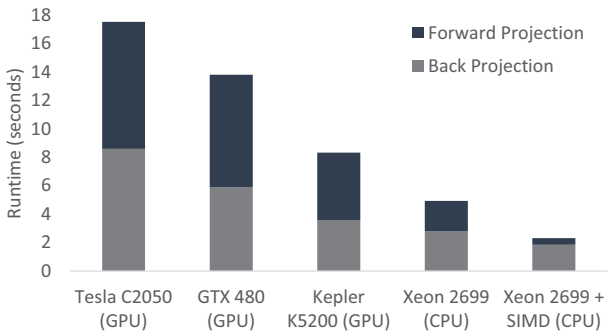


Fig. 6: Subset Gradient Runtime Comparison: Comparison of forward and back-projection runtimes (572 views) on various GPU generations and on the dual-socket Xeon 2699. Xeon MT and MT+SIMD reconstructions use all 72 logical cores. SIMD implementation uses 8-wide floating-point AVX2 instructions.

In particular, we found that the $\mathbf{T}_v^\intercal$ step of the back-projection, which performs an N-voxel read-modify-write operation, saturates available memory bandwidth with roughly 20 threads.

Fig. 5 illustrates average memory bandwidth consumption versus thread count. We estimated average memory bandwidth by precisely measuring the runtime of the $\mathbf{T}_v^\intercal$ step in each thread individually across 25 runs, then averaging across threads and runs. We then divide the total data read and written in each phase by the average runtime. The black line (Measured Peak) indicates the hardware's peak sustainable memory bandwidth, measured with the STREAM triad benchmark [12]. The average bandwidth consumption of our approach (MT+SIMD Full) matches the measured peak around 20 threads, and more threads do not improve performance.

Our bandwidth measurements imply that our multi-threaded SIMD aglorithm allows compute performance to greatly outstrip memory system performance on the Xeon 2699. Higher speedups could be obtained by using hardware with more memory bandwidth (*e.g.*, more DD4 memory channels or higher-bandwidth GDDR5 memory), or the same performance could be achieved at lower cost with a Xeon server with fewer cores. Alternatively, memory bandwidth can be reduced by storing the image more compactly in a half-precision format, still yielding acceptable reconstruction quality [13]. Fig. 5 (MT+SIMD Half) illustrates memory bandwidth scaling when we emulate half-precision format. Bandwidth of the $\mathbf{T}_v^\intercal$ step again saturates at the measured bandwidth peak, but with 40 threads instead of 20.

D. CPU vs GPU Comparison

Finally, we compare our MT+SIMD performance to prior SF results achieved with GPUs [5, 14]. Fig. 6 contrasts SF forward and back-projection on three GPU generations and our 72-thread MT and MT+SIMD performance on the Xeon 2699. The Xeon's high thread count allows the MT and MT+SIMD implementations to be faster than even the high-end K5200 GPU. The comparison also reveals the disparity in forward and back-projection runtimes for MT+SIMD that arise because back-projection incurs more memory traffic and saturates available bandwidth at a lower thread count. Future research should focus on memory bandwidth reduction (*e.g.*, via half-precision formats) to fully realize the remaining untapped speedup potential of SIMD.

IV. Summary and Conclusions

While iterative X-ray CT reconstruction provides excellent image quality, it still remains computationally expensive. Most prior work has focused on GPU and distributed computing to overcome this cost. This work examined the high thread count and SIMD support of modern CPUs. Our results show that with slight changes to the data mapping and a small approximation of the axial footprint, multi-threaded SIMD provides up to 55× speedup over a non-parallel implementation. We also showed that SIMD can provide up to 5× improvement over multi-threading alone, especially for lower thread counts; however, this improvement becomes limited by the memory bandwidth due to such high parallelism. Despite the bandwidth restrictions, multi-threaded SIMD performance was as good or better than a high-end GPU solution, so future work on overcoming the bandwidth limitations could provide even further improvement.

References

[1] J-B. Thibault, K. Sauer, C. Bouman, and J. Hsieh. A three-dimensional statistical approach to improved image quality for multi-slice helical CT. *Med. Phys.*, 34(11):4526–44, November 2007.

[2] J. M. Rosen, J. Wu, T. F. Wenisch, and J. A. Fessler. Iterative helical CT reconstruction in the cloud for ten dollars in five minutes. In *Proc. Intl. Mtg. on Fully 3D Image Recon. in Rad. and Nuc. Med*, pages 241–4, 2013.

[3] D. Kim and J. A. Fessler. Distributed block-separable ordered subsets for helical X-ray CT image reconstruction. In *Proc. Intl. Mtg. on Fully 3D Image Recon. in Rad. and Nuc. Med*, pages 138–41, 2015.

[4] M. Kachelrieß, M. Knaup, and O. Bockenbach. Hyperfast parallel-beam and cone-beam backprojection using the cell general purpose hardware. *Med. Phys.*, 34(4):1474–86, April 2007.

[5] M. McGaffin and J. A. Fessler. Alternating dual updates algorithm for X-ray CT reconstruction on the GPU. *IEEE Trans. Computational Imaging*, 1(3):186–99, September 2015.

[6] H. Scherl, M. Kowarschik, H. G. Hofmann, B. Keck, and J. Hornegger. Evaluation of state-of-the-art hardware architectures for fast cone-beam CT reconstruction. *Parallel Computing*, 38(3):111–24, March 2012.

[7] J. Treibig, G. Hager, H. G. Hofmann, J. Hornegger, and G. Wellein. Pushing the limits for medical image reconstruction on recent standard multicore processors. *Int. J. High Perf. Comp. Appl.*, 27(2):162–77, May 2013.

[8] H. Erdoğan and J. A. Fessler. Ordered subsets algorithms for transmission tomography. *Phys. Med. Biol.*, 44(11):2835–51, November 1999.

[9] D. Kim, S. Ramani, and J. A. Fessler. Combining ordered subsets and momentum for accelerated X-ray CT image reconstruction. *IEEE Trans. Med. Imag.*, 34(1):167–78, January 2015.

[10] Y. Long, J. A. Fessler, and J. M. Balter. 3D forward and back-projection for X-ray CT using separable footprints. *IEEE Trans. Med. Imag.*, 29(11):1839–50, November 2010.

[11] W. P. Segars, M. Mahesh, T. J. Beck, E. C. Frey, and B. M. W. Tsui. Realistic CT simulation using the 4D XCAT phantom. *Med. Phys.*, 35(8):3800–8, August 2008.

[12] J. D. McCalpin. Memory bandwidth and machine balance in current high performance computers. *IEEE Comp. Soc. Tech. Comm. on Comp. Arch. (TCCA) Newsletter*, pages 19–25, December 1995.

[13] C. Maaß, M. Baer, and M. Kachelrieß. CT image reconstruction with half precision floating-point values. *Med. Phys.*, 38(s1):S95–105, 2011.

[14] M. Wu and J. A. Fessler. GPU acceleration of 3D forward and backward projection using separable footprints for X-ray CT image reconstruction. In *Proc. Intl. Mtg. on Fully 3D Image Recon. in Rad. and Nuc. Med*, pages 56–9, 2011.