University of Utah
UNDERGRADUATE RESEARCH JOURNAL

ALGEBRAIC RECONSTRUCTION OPTIMIZATION THROUGH PRECOMPUTATION OF WEIGHTS MATRIX

Jared Green (Cem Yuksel)
Department of Engineering

ABSTRACT

There are several methods for volumetric scene analysis. This research focuses on real-time optimization of the ray-driven projector for forward and backward projections in the application of medical x-ray imaging. One of the current methods used by Computed Tomography systems to produce 3D medical images is the mathematical calculation of the line integral to simulate X-ray physics combined with Jacobi iterations, termed the Joseph method of discrete projection[1]. The proposed method is to expand upon the ray-driven model and precompute a sparse matrix of voxel to pixel relations. The transfer of a computational barrier to memory is only recently viable in real-time due to GPU hardware advances. The parallelization potential means that the mathematical operations can be precomputed and, with the utilization of modern GPU memory transfer protocols, cycled through for Jacobi iterations that rely on a sparse matrix of operations. Thus, the theoretical potential of the proposed method is twofold. First, the mathematical kernel operations become matrix operations which is already a deeply studied optimization problem, and second, the precomputation of angular information allows for more complex data compression. The second half of the research focuses on the extent to which algebraic convergence can improve by utilizing a high-resolution input matrix while still operating under the constraints of real-time. And with a final exploration of application to higher resolution reconstructions. This paper focuses on determining an optimal balance between algebraic reconstruction convergence, image quality, and time for a precomputed matrix.

INTRODUCTION

In real time application one of the most fundamental aspects of any algorithm is the time needed to first produce an initial output, and secondly, the time needed to complete the process. The ray-driven projector is currently used in medical image reconstruction as it incorporates localized memory access and a natural parallel structure, both of which improve the time efficiency of the resulting reconstruction. Applying the process to an iterative algorithm to generate a volume over sequential forward and back projections mandates even stricter time restraints for use in operating procedures. The minimization of delays in the operating room is critical to inform the surgeon of complications as soon as possible and maximize success rate. Thus, the necessity for optimization is instrumental to the development of iterative reconstruction in a medical setting.

BACKGROUND

The distance projector is a more recent rendition of the ray-driven projector [3]. The ray-driven projector initializes rays starting at the source position and projects through the volume using the projection matrix at that angle. For the forward projection this ray accumulates the voxel values

to the projection plane and for the back projection retraces the projection path to place voxel densities into the volume. This iterative process of forward and back projection results in approximate density placement based on the intensity of xRays in the initial projections [3]. The ray-driven method suffers from several key problems for image quality. Namely, the number of rays is directly correlated to precision of density placement and inversely correlated to the time needed to generate. That is, a correct scene reconstruction requires extensive computation and thus is not ideal for time limited operations. Secondly, the ray-driven method suffers from a loss of information from angled rays. This is a direct result of the reduction of rays for a time intensive process. As the rays originate from a source, by necessity the ray density decreases the closer to the projection plane, also relative to the distance of the source to the projection plane. In order to reduce this impact the operation of splatting was introduced. Splatting interpolates voxel values in an area of diminishing contribution around the projected ray [5]. However this adds even further computation to an already computational intensive problem. Distance driven projectors inherently produce a similar result through area calculations without adding to the algorithmic complexity. The splat for the distance driven projection occurs with the projection of the points to the projection plane thus expanding the influence of a single voxel across a region of the detector. A second solution to ray-driven problems is to project from the plane back to the source and thus concentrate rays in local areas. The second method gains the benefit of visibility to regional effects on the projection plane but fundamentally is just the inverse of the original and still suffers from similar, albeit inverted, problems. Thus, in 2002 Bruno De Man and Samit Basu introduced distance driven projections [3]. The distance driven projectors operate under similar conditions with the relation between volume and projection planes. Instead of tracing rays the distance driven method operates by projecting each z layer of voxels onto the projection plane and then calculating overlap and weights accordingly [3]. The mathematical procedure for the operation becomes independent of voxel location. However, for the forward projection in particular, the distance driven method suffers from the necessity of branching in the kernel due to each voxel affecting varying pixels of the projection plane. This branching operation reduces warp efficiency and ultimately slows down the entire process. To resolve this a branchless method was proposed by utilizing the Summed Area Table (SAT). This method precomputes the projections as an SAT in a cumulative summation and then uses the computation to efficiently calculate area overlap for region calculation. The distance driven projection to the plane is computed and the error result calculated from the SAT projection subtraction. The area value can then be calculated by subtraction operations. Through (bottom right) - (top left) - (the top right - top left) - (bottom left - top left) which simplifies to BR - TR - BL + TL [2]. Thus the area projection calculation can be done without looping over the influence region for weight calculation. More recent work has moved away from the optimization of the distance projector in favor of alternative approaches such as machine learning [6] and denoising techniques such as total variation [7]. Yet, the proposed work is still relevant as most approaches are still centered around the fundamentals of the forward and backward iterative method [7,8]. Notably, the distance driven SAT method still requires computation of the SAT at every iteration step.

Independent of the method, what the iterative process seeks to achieve is a relation between the x-ray projections and the volume over which the projections were generated. By forming a relation, the projections are iteratively compared against the relational error in order to converge to a solution. Thus, the goal of this paper is to propose that any method can be precomputed to form a relational matrix, or projection operator, which can then be applied to the projections. This matrix is by definition sparse as not every voxel interacts with every projection pixel. The relational distribution does vary between methods, the ray-driven method is notably sparser than the distance driven method due to the locality of rays versus the z level diffusion of the distance driven method. The generated projection operator can then be applied through matrix vector multiplication methods which have been intensively studied. Furthermore, the constructed matrix compression techniques are also critical to the application. The full matrix size of a projection operator matrix has a number of rows equal to the number of voxels in the desired output volume and a number of columns equal to the input projection size. For a volume of size 384x384x384 versus a projection size of 244x244 and a float representation the total size is approximately 13 terabytes. Subsequently, the application of such a matrix is infeasible in terms of storage and memory transfer time. Fortunately, the majority of such a matrix is composed of zero values. As noted, this is especially the case with the ray projection method as ray traversal through a volume leaves the vast majority of the volume irrelevant to the line integral. Thus, matrix compression techniques must be applied in order for the application of the projection operator to be tenable. Current matrix compression techniques include the Coordinate Format (COO), Compressed Sparse Row (CSR) and derived Compressed Sparse Column (CSC) implementations, ELLPACK, as well as more advanced compression algorithms. All of these compression algorithms only function with the assumption that the number of non-zero components in a matrix is significantly less than the total size of the matrix, on the order of 90 to 99 percent zero valued weights.

$$\begin{bmatrix} 1 & 5 & 7 & 0 & 0 \\ 0 & 3 & 4 & 9 & 0 \\ 0 & 0 & 6 & 2 & 4 \\ 0 & 0 & 0 & 3 & 1 \\ 0 & 0 & 0 & 7 & 5 \end{bmatrix}$$

COO Format

$$\begin{bmatrix} 1 & 5 & 7 & 3 & 4 & 9 & 6 & 2 & 4 & 3 & 1 & 7 & 5 \end{bmatrix}$$
$$\begin{bmatrix} 0 & 1 & 2 & 1 & 2 & 3 & 2 & 3 & 4 & 3 & 4 & 3 & 4 \end{bmatrix}$$
$$\begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 2 & 2 & 2 & 3 & 3 & 4 & 4 \end{bmatrix}$$

CSR Format

$$\begin{bmatrix} 1 & 5 & 7 & 3 & 4 & 9 & 6 & 2 & 4 & 3 & 1 & 7 & 5 \end{bmatrix}$$
$$\begin{bmatrix} 0 & 1 & 2 & 1 & 2 & 3 & 2 & 3 & 4 & 3 & 4 & 3 & 4 \end{bmatrix}$$
$$\begin{bmatrix} 0 & 3 & 6 & 9 & 11 \end{bmatrix}$$

Figure 1. Compression of an example matrix to COO and

Coordinate format is the simplest implementation of matrix compression with the direct saving of column, row, and value information. For every value in the input matrix the corresponding column and row is stored. Thus, the compression reduces a large sparse matrix from size m*n to (number of non-zero components)*(weight representation in bytes) + (number of non-zero components)*(2*column and row representation size in bytes). The CSR method takes the COO compression a step further. The non-zero values are stored in a list with the corresponding column index for CSR. The row information is compressed in a series of pointers indicating the start of each row. Thus the size is further compressed to (number of non-zero components)*(weight representation in bytes) + (number of non-zero components)*(column representation size in bytes) + n * (row representation in bytes).

The ELLPACK (ELL) format is another method to compress sparse matrices [11]. Matrix data is aligned to a matrix consisting of all non-zero elements, with non-equal element rows padded. The corresponding column indices are aligned to a matrix of identical dimensions.

$$\begin{bmatrix} 1 & 5 & 7 & 0 & 0 \\ 0 & 3 & 4 & 9 & 0 \\ 0 & 0 & 6 & 2 & 4 \\ 0 & 0 & 0 & 3 & 1 \\ 0 & 0 & 0 & 7 & 5 \end{bmatrix}$$

Elements

$$\begin{bmatrix} 0 & 1 & 2 \\ 1 & 2 & 3 \\ 2 & 3 & 4 \\ 3 & 4 & * \\ 3 & 4 & * \end{bmatrix}$$

Column Indices

$$\begin{bmatrix} 1 & 5 & 7 \\ 3 & 4 & 9 \\ 6 & 2 & 4 \\ 3 & 1 & * \\ 7 & 5 & * \end{bmatrix}$$

Figure 2. Example matrix

There are many more compressed formats than the three presented in this paper. While more advanced compression techniques may offer additional advantages, compressed sparse row and column implementations were selected for both simplicity in implementation and alignment of data structures. The ray-driven method produces rows that have equivalent numbers of non-zero elements as each ray step along the ray originating at some pixel (u,v) fetches the equivalent number of elements. Therefore, there is no significant advantage to using a blocked compression method.

Secondly, for efficient utilization of the projection operator the optimization of vector multiplication must be introduced. The main strategy employed is the utilization of warp memory coalescing. With the utilization of CSR format for A and the corresponding CSC format for A^T, all operations occur on a cascading warp index architecture such that the memory access for a warp of 32 threads is coalesced along the given axis. Each thread block of 32 is given the task of computing the output value at some row index. Matrix vector multiplication can be represented with the following computation; each output row o[n] is the result of the summation of the input vector at row m multiplied by the corresponding matrix value as row m, column n; e.g.

$$o[n] = \sum_{i=0}^{n} M[n][m] * V[m]$$

This poses the unique GPU implementation problem of memory access. The most efficient utilization of memory read occurs along the warp such that each thread within a warp reads at an offset of continuous memory[9]. That way the instruction multiprocessor can efficiently access data for the entire warp. The warp coallescence pairs with the CSR compression since the number of elements per row remains constant and there is no multiprocessor delay due to branching. Finally, the kernel operation must perform a warp reduction for each of the individual summations along the subsections. This can be performed with the CUDA warp level instruction sets.

METHODS

Regardless of whether the method is ray-driven or distance driven, a projection is a summation for every pixel over a series of weights multiplied by corresponding voxels.

$$\sum_{0}^{i} w_x x_i = y_{uv}$$

The weights and corresponding voxel locations can be precomputed at known angles resulting in a sparse matrix that subsequent acquisitions of x-ray data utilize to quickly compute a projection. For this paper, only the ray-driven method was used to precompute a projection

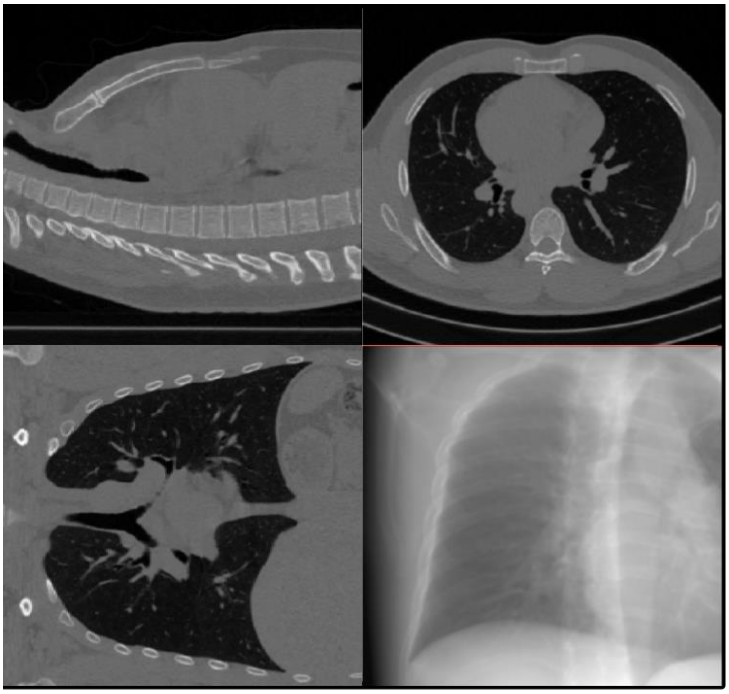

Figure 3. Volume Slices of CT (Top Left, Top Right, Bottom Left) used as input for simulated projections (Bottom Right)

operator. The ray-driven method was selected as it accentuates the benefit of sparse matrix multiplication. While both are viable constructs, the distance-driven method increases the density of voxel pixel relations and so would require further investigation in compression techniques to apply. The infrastructure was to first precompute the matrix and then parallelize the CUDA computation by performing the warp level computation reduction as explained above. The full projection operator matrix was computed, and CSR compression was applied. The computed matrix was loaded into global GPU memory and applied to input projections. The projector match was verified versus a constant value input projection stack and subsequently from simulated CT scans. The simulation CT scans were generated by applying the Joseph method of projection to CT computed volume (Figure 3).

The second stage of the investigation concerned the application of multi-iteration projection operators. The precomputation of the first stage can be expanded to include the voxel-pixel relations of the nth iteration by the computed formula and applied in the same manner as a single iteration, $x_n=Mg$.

$$M = \sum_{f=1}^{n} C_f \lambda^f A^T (AA^T)^{f-1}$$

To generate the precomputed weights, the multi-iteration matrix generation was generalized into a series of column vector multiplications. That is, to achieve a matrix generation, the column multiplication is applied recursively. The base case is A^T, and each subsequent formula index is either a CSR application or a CSC application versus the recursively generated column. A column at iteration *i* is computed from the stored projection operator matrix multiplied by the previous column for some formula F consisting of a series of A^TAA^TA operations.

$$M\_i = F_i M_{i-1}$$

Thus, all matrix operations can be reduced to a series of matrix-vector operations and optimized. Furthermore, the space complexity requires only the matrix A and a column vector in memory at any given time. The main benefit of precomputing multiple iterations is from subsequent compression methods that can be applied not only along with data correlations from the ray projection derivation but also along the iteration dimension to maximize the efficiency of data utilization. Thus for some given application of $x_n=Mg$, the relative data complexity can be dramatically reduced from the previous single iteration projection operator A. The compression

technique applied was a thresholding of the matrix M. The weights below the threshold value are binned by angle index and applied as a secondary step in the iteration process. The matrix multiplication then becomes the multiplication of the critical components versus the projection data plus the binned information versus the binned projection data.

$$x_n = M_{threshold} g + V_{binned} g_{binned}$$

The majority of information in subsequent iterations is composed of small negative weights versus angular peaks (Figure 4). The angular peaks are mainly composed of the initial AT information and refined by subsequent iteration reductions from other ray-driven data. Thus, by thresholding the matrix, the density information is preserved, and the application of a compressed binned weight versus projection data reintroduces the remaining subtraction components. This is critical for empty space density placement as the projection data will subtract the additional information versus the initial estimate of density from the precomputed geometric projection operator.

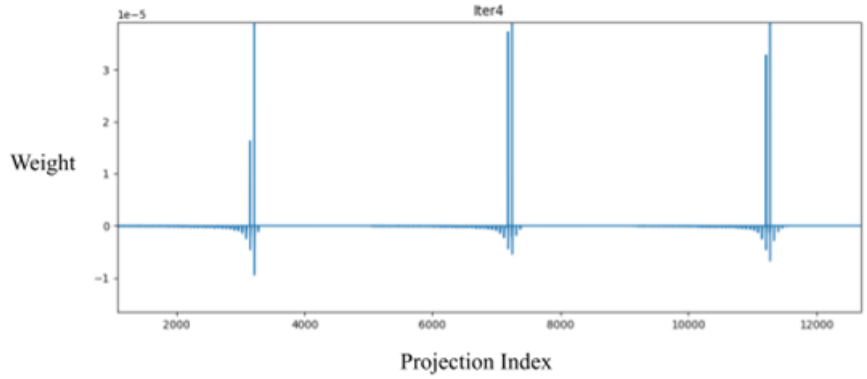

Figure 4. Matrix weight values at 1D panel pixel indices

The third expansion addressed in this paper is the application of the acceleration to meaningful volumes through a multi-resolution approach. While GPU resources limited the matrix computation capability to relatively small volumes in stage 2, stage 3 performed an upscale and seed application to volumes with the following application of the in-line approach. The upscaling was performed in a trilinear manner and introduced to the Jacobi iterations.

RESULTS

The testing was done utilizing Nvidia's visual profiler for speed optimizations, with the base case being the original distance-driven projector method and the second case being the attempted optimization using precomputation. The system configuration was an Intel i9-10900X, 64 GB DDR4 RAM, Ubuntu 18.04.5 LTS with kernel version 4.15.0-112-generic, CUDA 11.4 with Nvidia driver 470.57.02, and compiled with distributed nvcc version 11.4. The weights, read, and write indices were stored in an initial kernel precomputation stage, with the forward projector kernel computation analyzed afterward. The forward projector was run 10,000 times for both the attempted optimization and the original, producing the same results for a volume consisting of ones for a single angle as shown by Figure 5 and secondly to the simulated CT projections. The resulting projection was matched for both methods. However, the timing showed that the

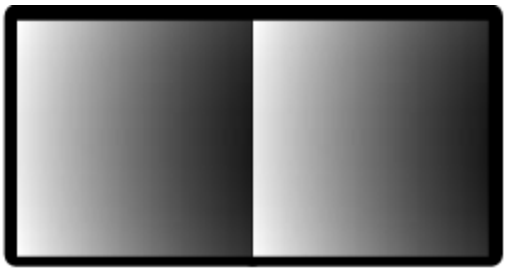

Figure 5. Projections using the original Distance Driven method (left) and the precomputed matrix (right) at a single angle of 30 degrees

attempted optimizations slowed the kernel down from approximately 138 to 245 nanoseconds. Thus, the access time for reading in pre-calculated values and performing the atomic addition onto the projection plane ultimately occupied more time than performing the mathematical operations. Memory consumption versus volume computation size was an additional problem.

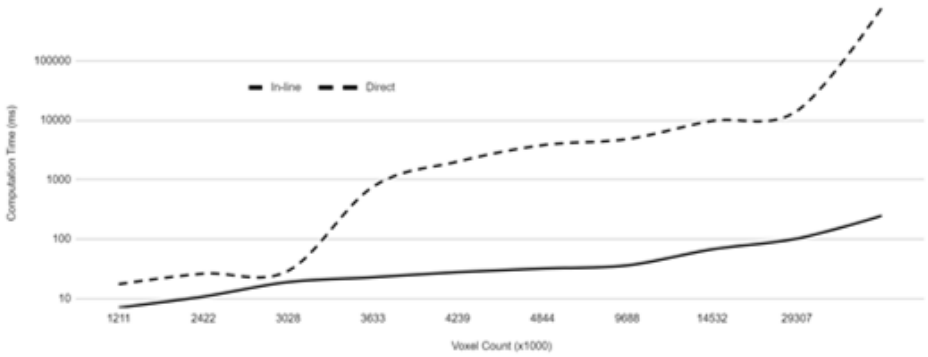

Figure 6. Computation time (ms) for single forward projection utilizing the direct matrix technique versus in-line ray projection

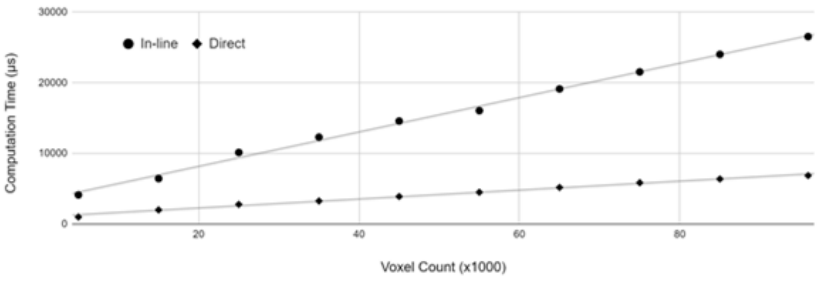

Figure 7. Computation time for 40 iterations of direct matrix method versus in-line ray projection method

Figure 6 displays the computation time of a single iteration for the direct matrix operator method versus the in-line ray projector. The application of the full matrix performs worse than the in-line method for all experimental resolutions. Furthermore, as seen in figure 6, there are significant time computation increases in two regions for the direct matrix method. The first region is due to the exceeding of GPU cache streamline capabilities and the subsequent use of global GPU memory. The second dramatic increase is with the introduction of unified CUDA memory, as the global memory limit was exceeded for the utilized GPU architecture. The total size of the uncompressed projection operation matrix became untenable for rapid memory utilization. Thus, a directed approach of compression was required with a further compression and time optimization of utilizing multiple iterations in a single step. With the introduction of matrix binning, the overall time to convergence was reduced versus the in-line method (Figure 7) while maintaining image quality. The relative convergence criteria also remained consistent with both solutions converging to approximately the same point (Figure 8).

For the third stage of the project, the volumes provided by the in-line method were applied to a higher resolution Jacobi iteration as a seed. The volumes were first upsampled through the use of CUDA texture resources and then applied as an input for the traditional Jacobi iteration algorithm. This resulted in a relative improvement in the starting point for the seeded method. Both converged approximately to the same point in the error projections. However, the seeded input reached both a lower threshold and similar final convergence in less time (Figure 9).

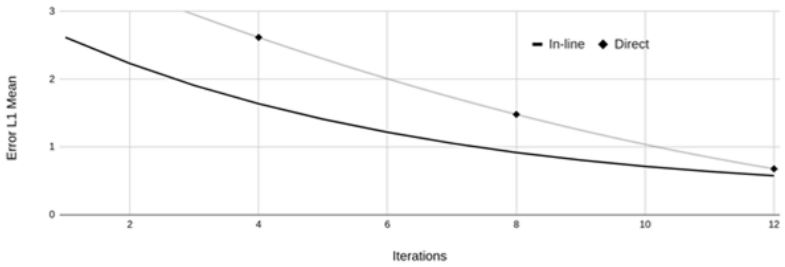

Figure 8. Convergence of error projections for direct matrix method versus in-line ray projector

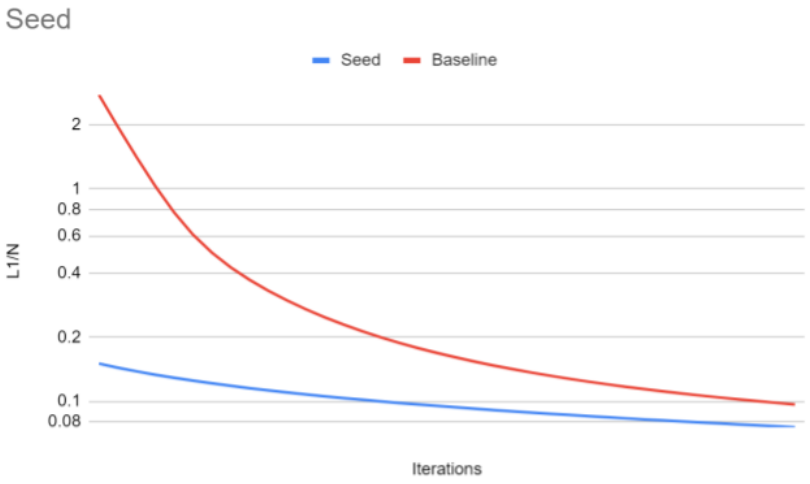

Figure 9. Convergence of Seeded Jacobi iterations from upsampled lower resolution matrix application versus no seed

CONCLUSIONS

The direct application of the full projection operator matrix was proven through memory model predictions and application to be slower than the traditional Jacobi iteration method. The direct in-line method of a single projection operator, even with CSR compression and optimized matrix-vector multiplications, did not achieve faster results than the traditional method. This required the investigation of further mathematical compression techniques to overcome the method access issues. The binning of small weights resulted in a significant reduction in overall matrix size with minimal impact on image quality and convergence. The reduction in matrix size offered a reduction in computation time, especially when applied to a multi-iteration model. Finally, the seed method shows promise for the direct application of the in-line method to production systems. Further investigation is still possible in the utilization of the in-line method to larger volumes with more sophisticated compression techniques. Further work would need to

improve upon the matrix generation efficiency or have access to more GPU infrastructure as the predicted time to compute a single projection operator at 384x384x384 volumes, and a four-step multi-iteration would take approximately three weeks with the GPU resources utilized in this paper.

REFERENCES

[1] P. M. Joseph. An Improved Algorithm for Reprojecting Rays through Pixel Images. IEEE Transactions on Medical Imaging. 1982 Nov; 1(3), pp. 192-196. doi: 10.1109/TMI.1982.4307572.

[2] Liu, Rui & Fu, Lin & De Man, Bruno. (2017). GPU-Based Branchless Distance-Driven Projection and Backprojection. IEEE Transactions on Computational Imaging. PP. 1-1. 10.1109/TCI.2017.2675705.

[3] B. De Man and S. Basu, "Distance-driven projection and backprojection," *2002 IEEE Nuclear Science Symposium Conference Record*, Norfolk, VA, USA, 2002, pp. 1477-1480 vol.3, doi: 10.1109/NSSMIC.2002.1239600.

[4] Du, Y., Yu, G., Xiang, X. *et al.* GPU accelerated voxel-driven forward projection for iterative reconstruction of cone-beam CT. *BioMed Eng OnLine* **16,** 2 (2017). https://doi.org/10.1186/s12938-016-0293-8

[5] M. Zwicker, H. Pfister, J. van Baar and M. Gross, "EWA splatting," in *IEEE Transactions on Visualization and Computer Graphics*, vol. 8, no. 3, pp. 223-238, July-Sept. 2002, doi: 10.1109/TVCG.2002.1021576. http://hhoppe.com/recursive.pdf

[6] Chen, Gaoyu, et al. "AirNet: Fused analytical and iterative reconstruction with deep neural network regularization for sparse-data CT." *Medical physics* 47.7 (2020): 2916-2930.

[7] Zhang, Zheng, et al. "Directional-tv algorithm for image reconstruction from limited-angular-range data." *Medical Image Analysis* (2021): 102030.

[8] Geyer, Lucas L., et al. "State of the art: iterative CT reconstruction techniques." *Radiology* 276.2 (2015): 339-357.

[9] Bakhoda, Ali, et al. "Analyzing CUDA workloads using a detailed GPU simulator." *2009 IEEE International Symposium on Performance Analysis of Systems and Software*. IEEE, 2009.

[10] Harun, H. H., et al. "The influence of iterative reconstruction level on image quality and radiation dose in CT pulmonary angiography examinations." *Radiation Physics and Chemistry* 178 (2021): 108989.

[11] Grimes, Roger G., David R. Kincaid, and David M. Young. *ITPACK 2.0 user's guide*. Center for Numerical Analysis, Univ., 1979.

[12] K. Choo, W. Panlener and B. Jang, "Understanding and Optimizing GPU Cache Memory Performance for Compute Workloads," 2014 IEEE 13th International Symposium on Parallel and Distributed Computing, Marseille, France, 2014, pp. 189-196, doi: 10.1109/ISPDC.2014.29.

[13] Al-Kharusi, Ibrahim, and David W. Walker. "Locality properties of 3D data orderings with application to parallel molecular dynamics simulations." *The International Journal of High Performance Computing Applications* 33.5 (2019): 998-1018.

[14] Bell, Nathan, and Michael Garland. *Efficient sparse matrix-vector multiplication on CUDA*. Vol. 2. No. 5. Nvidia Technical Report NVR-2008-004, Nvidia Corporation, 2008.