Simple and efficient raycasting on modern GPU's read-and-write memory for fast forward projections in iterative CBCT reconstruction

Jonas Dittmann*, Randolf Hanke*†
*Lehrstuhl für Röntgenmikroskopie, Universität Würzburg, Germany
†Fraunhofer EZRT, Fürth, Germany
jonas.dittmann@physik.uni-wuerzburg.de

***Abstract*—Forward projection of 3D voxel volumes ("X-ray transform") is one of the central and computation intensive tasks of all iterative tomographic reconstruction algorithms. It is typically implemented using ray driven algorithms such as the often cited Siddon's algorithm, traversing a voxel volume along connecting lines between X-ray source and detector.**

While the texture units of Graphical Processing Units (GPUs) dedicated to fast read-only random memory accesses have long been employed for tomographic reconstruction, their performance advantage cannot be fully utilized in iterative techniques which inherently require steady read-and-write memory accesses to the to-be-reconstructed volume.

With the objective of accelerating iterative cone beam computed tomography (CBCT) reconstruction methods operating solely on read-and-write GPU main memory (RAM), a branchless 3D generalization of Joseph's projection algorithm is presented that is both highly efficient on GPU RAM and easy to implement.

The presented raycasting algorithm is benchmarked on a recent consumer grade GPU and compared to a DDA algorithm (equivalent to Siddon's). It outperforms the latter both with respect to memory access rate (factor 3.5) as well as total run time both on GPU RAM and texture memory (factor 1.2). At about 600 (740) GB/s of memory access rate, it computes over 350 (450) projections of a 512^3 voxel volume per second on main memory (texture memory).

I. Introduction

Iterative tomographic reconstruction techniques can be roughly summarized as alternating simulation (i.e. cone beam projection), correction (backprojection) and possibly regularization (such as edge preserving denoising filters) steps. Although both projection and backprojection can be expressed as 3D image resampling operations well suited for acceleration through GPUs' texture units, the inherent requirement for steady updates to the intermediate reconstruction results (also by regularization procedures) make it preferable to store and manipulate it within the read-and-write memory of the GPU (also referred to as GPU RAM, GPU main memory or GPU global memory).

Subject of this article will be the forward simulation, i.e. the cone beam projection of a voxel volume and its efficient GPU implementation. The rich literature body on the subject of volume projection (not only tomography related) covers both implementation efficiency as well as precision of forward modeling, with varying definitions of what precise modeling implies. Regarding efficient implementation, the main factors are the complexity and amount of computations required to determine the to-be-sampled memory addresses as well as the corresponding interpolation weights for a given forward model. Regarding modeling of volume and X-rays, there exist mainly three notions of "exact": cubic (or rectangular) voxels pierced by pencil beams, cubic voxels pierced by beams with rectangular profile, and smoothly overlapping voxels modeled by radial basis functions of finite extent whose projections can be exactly described by isotropic footprints (which may further be convoluted with some beam profile).

The popular algorithm by Siddon [1] (also mentioned earlier by [2]) belongs to the first class and can be easily implemented in 2D and 3D using the Digital Differential Analyzer otherwise known from line drawing and general purpose ray tracing [3]. Extending it to model finite beam widths requires oversampling, i.e. simulation of many rays within a beam profile, making it computationally intensive. Direct strip integral modeling for cubic voxels is simple in 2D [4], yet rather involved in three dimensions [5]. Isotropic footprint methods (e.g. [6], [7]), although simplifying projection at first, introduce normalization issues as radial basis functions are never space-filling.

Methods such as Joseph's projector [8] or the recently popular Distance Driven method [9] prioritize sampling considerations over "exact" system modeling by some definition. While the Distance Driven method is particularly well suited for single threaded algorithms, Joseph's algorithm lends itself to massive parallelization. 3D extensions of it have e.g. been shown in [10] for single threaded use on CPU and by [11] for parallal GPU based volume projection utilizing stacks of 2D textures.

In the following, a branchless 3D generalization of Joseph's algorithm will be presented that is simple to implement and highly efficient on modern GPUs' read-and-write main memory (as opposed to texture memory). This is of particular relevance for iterative reconstruction algorithms which steadily need to update the volume. Its performance both with respect to projection quality, GPU fill rate and total runtime will be compared to a GPU friendly DDA (equivalent to Siddon's algorithm) implementatation [12], which is of comparable complexity and popularity.

DOI: 10.12059/Fully3D.2017-11-3203040

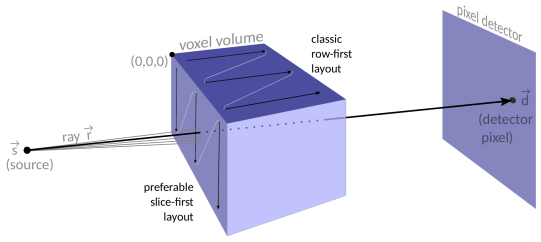


Figure 1. Illustration of the raycasting task: A ray $\vec{r} \propto \vec{d} - \vec{s}$ originating from source $\vec{s}$ traverses a voxel volume along its way to a detector pixel $\vec{d}$.

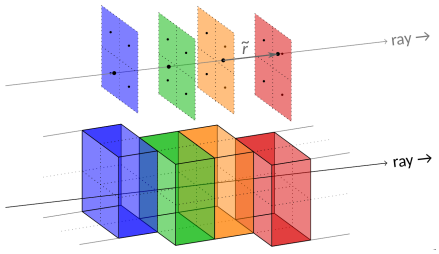


Figure 2. Illustration of a ray running through 4-voxel groups (bottom) or analogously piercing consecutive sampling planes aligned with the Cartesian voxel grid and oriented perpendicular to the ray's principal orientation ("driving axis"). Projection along a ray is done by accumulating bilinear interpolated samples from all intersected planes, weighted by the sampling distance $\|\tilde{r}\| \in (1, \sqrt{3})$.

II. Method: Generalized Joseph Projector

The basic problem is illustrated in Fig. 1. A ray $\vec{r}$ originating from a source position $\vec{s}$ traverses a voxel volume and hits a detector pixel at location $\vec{d}$. Along its intersection with the volume, the latter will be sampled in steps of $\tilde{r}$ (cf. Fig. 2). For convenience, the coordinate system is aligned with the voxel grid, i.e. the origin is at the corner of the voxel volume and all distance units will be expressed in terms of grid spacing.

A. Choice of sampling points

The sampling scheme will be based on the "driving axis" concept: All sampling points within the volume along the ray will be aligned to integer coordinates of one dedicated driving axis. The driving axis m is defined by the largest component of $\vec{r}$:

$$m = \underset{i}{\operatorname{argmax}}(|r_i|)\,. \tag{1}$$

The required increment $\tilde{r}$ between successive sampling points $\vec{p}^{(i)}$ is then:

$$\tilde{r} = \frac{\vec{r}}{r_m}\,. \tag{2}$$

The first possible sampling point is located at the plane perpendicular to the driving axis intersecting the coordinate origin (at which the voxel volume is bounded):

$$\begin{aligned} [\vec{s} + o\,\tilde{r}]_m &\stackrel{!}{=} 0 \\ \Rightarrow o &= -s_m \end{aligned} \tag{3}$$

where o is the distance between source and first sampling plane in units of the sampling increment $\tilde{r}$. o will thus be termed "sampling offset".

The volume can now be sampled at evenly spaced and driving-axis-aligned sampling points

$$\vec{p}^{(i)} = (\vec{s} + o\,\tilde{r}) + i\,\tilde{r} \tag{4}$$

for integer $i \in [0, i_{\max}]$, as illustrated in Fig. 2. $i_{\max}$ is defined by the extent of the voxel volume along the driving axis m.

These sampling points can readily be used for linear interpolated 3D texture sampling directly provided by GPUs, yielding a very brief and efficient projection algorithm.

B. Sampling and interpolation

Sampling from GPU main memory further requires explicit addressing of the 4-neighborhood $\{\vec{v}^{(i,1)}, \vec{v}^{(i,2)}, \vec{v}^{(i,3)}, \vec{v}^{(i,4)}\}$ of each sampling point among which interpolation is to be applied. By construction, every sampling point $\vec{p}^{(i)}$ as defined by Eqs. 1–4 has (in 3D) at most two non-integer components. These non-integer coordinates necessarily lie between two integer ones along their respective axes. For each sampling point, the group of four nearest neighbor voxels $\vec{v}^{(i,1-4)}$ (shown as $1 \times 2 \times 2$ boxes in Fig. 2) can thus be identified by regarding all combinations of floor and ceiling values of the two non-integer components of $\vec{p}^{(i)}$. By exploiting that floor and ceiling values of an integer will be identical, this can be conveniently formulated independent of the actual driving axis m:

$$\begin{aligned} \vec{v}^{(i,1)} &= (\lfloor p_1^{(i)} \rfloor, \lfloor p_2^{(i)} \rfloor, \lfloor p_3^{(i)} \rfloor) \\ \vec{v}^{(i,2)} &= (\lfloor p_1^{(i)} \rfloor, \lceil p_2^{(i)} \rceil, \lceil p_3^{(i)} \rceil) \\ \vec{v}^{(i,3)} &= (\lceil p_1^{(i)} \rceil, \lfloor p_2^{(i)} \rfloor, \lceil p_3^{(i)} \rceil) \\ \vec{v}^{(i,4)} &= (\lceil p_1^{(i)} \rceil, \lceil p_2^{(i)} \rceil, \lfloor p_3^{(i)} \rfloor) \end{aligned} \tag{5}$$

where $\lfloor\,\rfloor$ and $\lceil\,\rceil$ designate the floor and ceiling operators respectively.

Independent of $m \in \{1, 2, 3\}$, the vectors $\vec{v}^{(i,1-4)}$ define a group of at most four voxels in a plane perpendicular to the driving axis. Special cases arise when either of the non-m components of $\vec{p}^{(i)}$ is also integer, which leads to only one or two different $\vec{v}^{(i,1-4)}$. These special cases will be handled by the definition of the interpolation weights, which will evaluate to zero for duplicate $\vec{v}$.

Bilinear interpolation weights are based on the fractional parts in the $\vec{p}^{(i)}$ (corresponding to dimension-wise 1D distances to voxel grid points):

$$\begin{aligned} w_{\mathrm{cl},j}^{(i)} &= p_j^{(i)} - \left\lfloor p_j^{(i)} \right\rfloor \\ w_{\mathrm{fl},j}^{(i)} &= 1 - w_{\mathrm{cl},j}^{(i)} \end{aligned} \tag{6}$$

The driving axis components $w_{\mathrm{cl},m}^{(i)}$ and $w_{\mathrm{fl},m}^{(i)}$ are treated specially and forced to be 1:

$$w_{\mathrm{cl},m}^{(i)} := w_{\mathrm{fl},m}^{(i)} := 1 \tag{7}$$

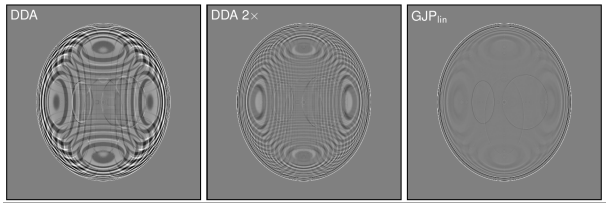
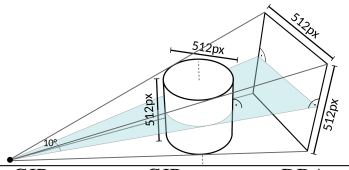


Figure 3. Differences between analytic reference projections (based on analytic integrals over the defining ellipses of the Shepp Logan phantom) and numeric projections based on a discretely sampled Shepp Logan phantom using different projection algorithms. From left to right: DDA (equivalent to Siddon's algorithm), 2×2 fold oversampled DDA, and the Generalized Joseph Projector GJP$_{\text{lin}}$.



	GJP$_{\text{lin}}$	GJP$_{\text{hw}}$	DDA	DDA 2×2
GPU RAM	2.70 ms 609 GB/s	—	5.20 ms 104 GB/s	6.74 ms 323GBs
Texture	2.40 ms 686 GB/s	2.23 ms 740 GB/s	3.46 ms 157 GB/s	7.65 ms 285 GB/s

Table I
BENCHMARK RESULTS ON AN NVIDIA GTX 1080 GPU FOR 10° CONE BEAM PROJECTIONS

so that 4 weights $w^{(i,1-4)}$ corresponding to the sampled voxel grid locations $\vec{v}^{(i,1-4)}$ can be defined:

$$\begin{aligned} w^{(i,1)} &= w^{(i)}_{\text{fl},1}\, w^{(i)}_{\text{fl},2}\, w^{(i)}_{\text{fl},3} \\ w^{(i,2)} &= w^{(i)}_{\text{fl},1}\, w^{(i)}_{\text{cl},2}\, w^{(i)}_{\text{cl},3} \\ w^{(i,3)} &= w^{(i)}_{\text{cl},1}\, w^{(i)}_{\text{fl},2}\, w^{(i)}_{\text{cl},3} \\ w^{(i,4)} &= w^{(i)}_{\text{cl},1}\, w^{(i)}_{\text{cl},2}\, w^{(i)}_{\text{fl},3} \end{aligned} \tag{8}$$

The integral or sum over all sampling points, i.e. the projection, must finally be weighted by the sampling interval $\|\tilde{r}\|$:

$$\text{projection}(\vec{s},\vec{d}) = \sum_{i=0}^{i_{\max}} \sum_{k=1}^{4} \|\tilde{r}\|\ \text{voxelVolume}\Big[\vec{v}^{(i,k)}\Big]\ w^{(i,k)}$$

III. RESULTS:

A. Projection quality

The proposed Generalized Joseph Projector is tested on a Shepp Logan phantom. The performance with respect to (implicit) system modeling is demonstrated on cone beam projections of a three dimensional Shepp Logan phantom. Analytic projections by means of exact line integrals through the ellipses defining the phantom serve as reference. In order to account for the finite beam extent, an array of 8×8 analytic pencil beam integrals is averaged for each detector pixel. In total 803 ($\approx \frac{\pi}{2}512$) projections onto a 512^2 pixel detector from different orientations covering the full angular range of 360° are computed for a 10° cone beam geometry. For comparison with numeric projections, the phantom is rendered on a 512^3 voxel grid using 5^3 fold oversampling for edge anti-aliasing. Fig. 3 shows deviations between numeric projections from a finitely sampled voxel volume by means of different algorithms from the reference analytic projections.

B. Projection speed

Run time performance is evaluated for projections of a cylindric volume within a cubic bounding box of 512^3 voxels onto a 512^2 pixel detector (as depicted above Table I) on a recent Nvidia GTX 1080 GPU. The volume is stored in 32bit floating point format in either main- or texture memory of the graphics processing unit. As typical for computed tomography setups, projections are performed for a multitude of source and detector orientations over the full angular range of 360° on a circular trajectory around the volume center. The rotational axis is aligned parallel to the fastest index of the memory layout (as also proposed in [13] for a Siddon-type algorithm). Table I lists average runtimes for each projection for different algorithms and memory choices. The table also displays the corresponding rates at which memory is accessed.

IV. DISCUSSION AND CONCLUSION

The problem of driving axis based forward projection is expressed through a parametric vector formulation, which allows a concise representation of the required sampling points. The generally required case differentiation for the three possible driving axes is avoided by suiting evaluation of the voxel grid points that are to be sampled as well as the corresponding interpolation weights. The simple and branchless computations involved can be very efficiently handled by modern graphics processing units. The driving axis approach, ensuring grid-aligned steps through the voxel volume, ensures coherent memory accesses by adjacent rays that are traced simultaneously. The particular choice of memory layout ensures that for all projection angles there are always stacks of adjacent rays (parallel threads) simultaneously accessing continuous memory segments. This allows for extremely high projection rates on the GPU main memory, which in contrast to texture memory is not optimized for random accesses. For efficient tomographic reconstruction, the generalized Joseph projector can be combined with a voxel driven backprojection algorithm. The linear interpolation kernel used may further be exchanged by smoother options similar to [14].

The comparison with a GPU optimized version the classic Siddon or DDA algorithm shows an overall runtime advantage of GJP over DDA in terms of absolute runtime despite its more than 3 times higher amount of total memory accesses. Good utilization of GPU memory caches allows to exceed the nominal memory transfer rate of the employed GPU, indicating very high fill rates of its resources.

REFERENCES

[1] R. L. Siddon, "Fast calculation of the exact radiological path for a three-dimensional CT array," *Medical Physics*, vol. 12, no. 2, pp. 252–255, 1985.

[2] R. H. Huesman, G. T. Gullberg, W. L. Greenberg, and T. F. Budinger, *RECLBL Library Users Manual – Donner Algorithms for Reconstruction Tomography*, 1977.

[3] J. Amanatides and A. Woo, "A Fast Voxel Traversal Algorithm for Ray Tracing," *Eurogrpahics*, vol. 87, no. 3, p. 10, 1987. [Online]. Available: http://www.cse.chalmers.se/edu/year/2015/course/TDA361/grid.pdf

[4] S. B. Lo, "Strip and Line Path Integrals with a Square Pixel Matrix: A Unified Theory for Computational CT Projections," *IEEE Transactions on Medical Imaging*, vol. 7, no. 4, pp. 355–363, 1988.

[5] W. Yao and K. Leszczynski, "Analytically derived weighting factors for transmission tomography cone beam projections," *Physics in Medicine and Biology*, vol. 54, no. 3, pp. 13–533, 2009.

[6] K. M. Hanson and G. W. Wecksung, "Local basis-function approach to computed tomography," *Applied Optics*, vol. 24, no. 23, pp. 4028–4039, 1985.

[7] R. M. Lewitt, "Alternatives to voxels for image representation in iterative roconstruction algorithms," *Physics in Medicine and Biology*, vol. 37, no. 3, pp. 705–716, 1992.

[8] P. M. Joseph, "An Improved Algorithms for Reprojecting Rays Through Pixel Images," *IEEE Transactions on Medical Imaging*, vol. MI-1, no. 3, pp. 192–196, 1982.

[9] B. De Man and S. Basu, "Distance-driven projection and backprojection in three dimensions," *Physics in Medicine and Biology*, vol. 49, no. 11, pp. 2463–2475, 2004.

[10] C. Schretter, "A Fast Tube-of-Response Raytracer," *Medical Physics*, vol. 33, no. 12, pp. 4744–4748, 2006.

[11] F. Xu and K. Mueller, "Accelerating Popular Tomographic Reconstruction Algorithms on Commodity PC Graphics Hardware," *IEEE Transactions on Nuclear Science*, vol. 52, no. 3, pp. 654–663, 2005.

[12] K. Xiao, D. Z. Chen, X. S. Hu, and B. Zhou, "Efficient implementation of the 3D-DDA ray traversal algorithm on GPU and its application in radiation dose calculation ," *Medical Physics*, vol. 39, no. 12, pp. 7619–7625, December 2012.

[13] W. M. Thompson and W. R. B. Lionheart, "GPU Accelerated Structure-Exploiting Matched Forward and Back Projection for Algebraic Iterative Cone Beam CT Reconstruction," *The Third International Conference on Image Formation in X-Ray Computed Tomography*, pp. 355–358, June 2014.

[14] J. Sunnegårdh and P. Danielsson, "A New Anti-Aliased Projection Operator for Iterative CT Reconstruction," 2007.