A Study of Volume Integration Models for Iterative Cone-Beam Computed Tomography

Sungsoo Ha, Ayush Kumar, and Klaus Mueller

***Abstract*— With the help of modern parallel computers, iterative reconstruction algorithms have become a feasible research topic in the field of CT. These types of algorithms can greatly benefit from an accurate, realistic CT system model. In our study, we model the CT projection as volume integrals and propose a set of methods that can compute the volume integrals either exactly or approximately. Our approximate volume integral methods have a much smaller complexity algorithmically than the exact method, but their accuracy is close to it. More importantly, the proposed approximate methods can be easily ported to modern parallel processors to utilize their massive computation powers.**

***Index Terms*—computed tomography, system matrix, forward projection, line integral model, volume integral model**

I. Introduction

With the increasing popularity of iterative reconstruction algorithm in the field of CT medical imaging, modeling realistic CT systems in software is becoming more crucial than ever. The CT model is usually described by a large matrix, **A**, whose columns are corresponding to voxels to be reconstructed (J×1 vector) and the rows are corresponding to the projections (or line integrals, I×1 vector) that are usually measured by the CT scanner. Then, each element, a_{ij}, of the matrix represents the contribution of a voxel *j* to a line integral *i*, the so called weight coefficient. This gives rise to the system equation:

$$\mathbf{AI} = \mathbf{J}$$

The process of calculating the line integrals is known as the forward model and its reverse model, generally defined as the transpose of the forward model, is known as the back-projection operation.

The most intuitive approach to describe the forward model would be the line integral model (LIM) where each weight coefficient, a_{ij}, is computed as the intersection length between th *j*-th voxel and the *i*-th ray. Usually, in this type of approaches [1][2], a single ray is described by a zero width line that connects the x-ray source and the center of the detector cell for the ray-driven approach (or the center of a voxel for the voxel-driven approach).

The other approach that is much closer to a real CT system is the volume integral model (VIM). In this model, a single ray can be described by a pyramid shaped 3D polytope that connects the x-ray source and the four sides of the rectangular shape of a detector cell. Then, the weight coefficient, a_{ij}, is the intersection volume between the ray and the cube shaped voxel. However, exactly computing the intersection volume (or arbitrary shape of convex polygon) is not a trivial problem.

There are two well reported approaches that approximate the intersection volume. The first approach is the distance-driven (DD) method [3] which computes the coefficient as the row or slab intersection length combined with the overlap coefficient. The overlap coefficient is computed based on the length or area of overlap between a voxel and a detector cell when they are mapped onto each other as seen by the source. The other approach is the separable footprint (SF) method [4] which approximates the voxel footprints as 2D separable functions. This approximation not only greatly simplifies the computation of the coefficient but has also been shown as more accurate than the DD methods, while keeping a similar computational cost.

Recently, Zhang et al. [5] presented a method that computes the exact intersection of the X-ray beam geometry with a voxel. However, they only consider 2D fan-beam reconstruction, deferring its extension to cone-beam to future work. The fan beam method itself is fairly complex, breaking the problem up into a set of special cases depending on the type of intersection.

We present a closed-form solution for exact fan-beam intersection which is fairly straightforward and does not require a subdivision into cases. Unfortunately this closed form does not generalize to cone-beam without a subdivision into simpler primitives. In fact, there is no reported closed-form solution for the volume of the generalized 3D polytope that arises from intersecting a pyramidal come-beam with a voxel. We choose a solution that subdivides the polytope into a set of tetrahedrons, but like other method of this kind, this bears significant overhead. For this reason we also propose three methods that approximate the 3D intersection volume by chopping a voxel into sub-voxels with known volume. Such solutions are quite amenable to parallelization on the GPU.

Next we first describe our volume integral methods and then evaluate their performances in terms of accuracy and speed.

II. Methodology

We propose four different schemes that compute the weight coefficients in a CT system as volume integrals. Figure 1 and 2 show schematic representations of the four methods in 2D (for simplicity) but all have been implemented and tested in 3D. Especially, the schemes shown in Figure 2 approximate (or indirectly) estimate a_{ij} in a fashion that lends itself well to the massive parallel computing power of modern GPUs. In the following subsections we describe all methods in closer detail.

Sungsoo Ha, Ayush Kumar, and Klaus Mueller are with the Visual Analytics and Imaging Lab, Computer Science Department, Stony Brook University, NT, and SUNY, Korea (e-mail: {sunha, ayush.kumar, mueller}@sunykorea.ac.kr)..

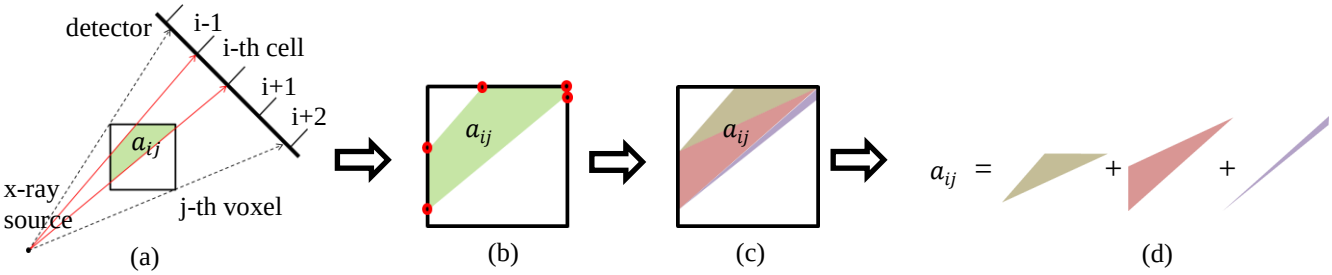


Figure 1. Exact Volume Integral Method. (a) Find detector cells that interact with voxel *j*, (b) Clip the voxel with the x-ray beam, (c) Tetrahedralize the intersected convex polygon, and (d) compute a_{ij} by summing all tetrahedrons.

A. *Exact Volume Integration Method*

The intersected volume can be computed in an exact manner as shown in Figure 1. To compute the intersected volume, a_{ij}, we first project a voxel, v_j, to a (flat) detector to find detector cells that interact with the voxel. For each cell, c_i, a pyramid-like shape of a beam can be designed with four-planes that connect the x-ray source to the four-edges of the cell. Then, the intersected volume between the voxel and the beam is equal to the volume of the voxel clipped by the four-planes. We used the Sutherland-Hodgman clipping algorithm [6] which works by extending each plane of the convex shape of the beam in turn and selecting only vertices from the subject polygon, v_j, that are on the visible side.

For the 2D (fan-beam) case, given the set of vertices returned from the clipping algorithm, there is a closed-form solution for finding the area of the resulting arbitrary convex polygon:

$$area = \left| \frac{(x_1y_2 - y_1x_2) + (x_2y_3 - y_2x_3) + \cdots + (x_ny_1 - y_nx_1)}{2} \right|$$

where x_m and y_m are the x- and y-coordinate of a vertex, and number of vertices in order, going either clockwise or counter-clockwise, staring at any vertex. However, there is no such closed-form solution the volume for a convex polygon (a polytope) in 3D. There are a number of ways to go about this, but decomposing the polytope into a number of tetrahedrons whose volume is equal to 1/6 of the absolute value of the triple product:

$$Volume = \frac{1}{6} \begin{vmatrix} u_1 & u_2 & u_3 \\ v_1 & v_2 & v_3 \\ w_1 & w_2 & w_3 \end{vmatrix}$$

Here, *u*, *v*, and *w* are the vectors between three vertices and a vertex, which can be chosen randomly out of four vertices in a tetrahedron. In our implementation, the tetrahedralization is realized using 3-D Delaunay triangulation approach [7]. Then, the exact intersected volume of a_{ij} is computed as the sum of all tetrahedrons within the polytope.

B. *Approximate Volume Integral Method*

We will now describe three strategies that approximate or indirectly estimate the intersected volume, a_{ij}, by dividing a voxel *j* into a number of small sub-voxels. The first two methods are to approximate the volume by counting the number of sub-voxels within a beam; while the third method will indirectly estimate a_{ij} by projecting all sub-voxels onto the detector. All of these methods are algorithmically much simpler than computing the exact volume and, more importantly, are targeted to utilize the parallel computation power of modern GPUs. Figure 2 shows a schematic representation of these methods and details of each method will be explained next.

1) *Riemann sum approach*

One of the simplest and most intuitive approaches to approximate the intersected volume, a_{ij}, would be counting the number of sub-voxels of known volume within the intersected volume. The approach is inspired by an approximation of the area underneath a curve, also known as a Riemann sum where the area is divided into a number of rectangles (or trapezoids) and approximated by the sum of all the rectangles. Similarly, a voxel, v_j, is first divided into N^3 of sub-voxels whose side length is 1/*N* of the voxel's side. Then, the intersected volume is approximated by counting the number of sub-voxels whose central points are inside the beam. Figure 2(a) shows a brief explanation of this approach with the case that *N* is 4 (in 2D for simplicity). In this example, the volume is approximated by $6 \cdot \delta$ where 6 is the number of rectangles whose central points (red dots) lie inside of the beam (green area) and δ is the area of a small rectangle.

2) *Recursive sub-division approach*

In this approach, instead of having a fixed number of sub-voxels per voxel as in the Riemann sum approach, we recursively divide it into *N* sub-voxels when the current cube is intersecting with the beam. Suppose that each time a voxel is divided into *N* sub-voxels. Each sub-voxel will be evaluated if it intersects with a beam or not. If it intersects, the sub-voxel will be divided into another *N* sub-voxels. This process will be recursively enacted until a pre-defined number of sub-divisions has been reached. The final volume is computes by counting the number of the smallest size of sub-voxels that pass the intersection test. This process is described in Figure 2(b) for the 2D case when *N* is 2 and there are two sub-divisions. In this example, the volume is approximated as $9 \cdot \delta$ where 9 is the number of rectangles at the finest levels that are determined as the one having overlap region between the beam (green area) and tangent circle to the rectangle, and δ is area of a rectangle at the finest level.

The intersection test is performed by setting the criterion of non-intersection cases. There are four cases when a voxel (or a sub-voxel) does not overlap with a beam, designed by four planes with inward direction of normal vector.

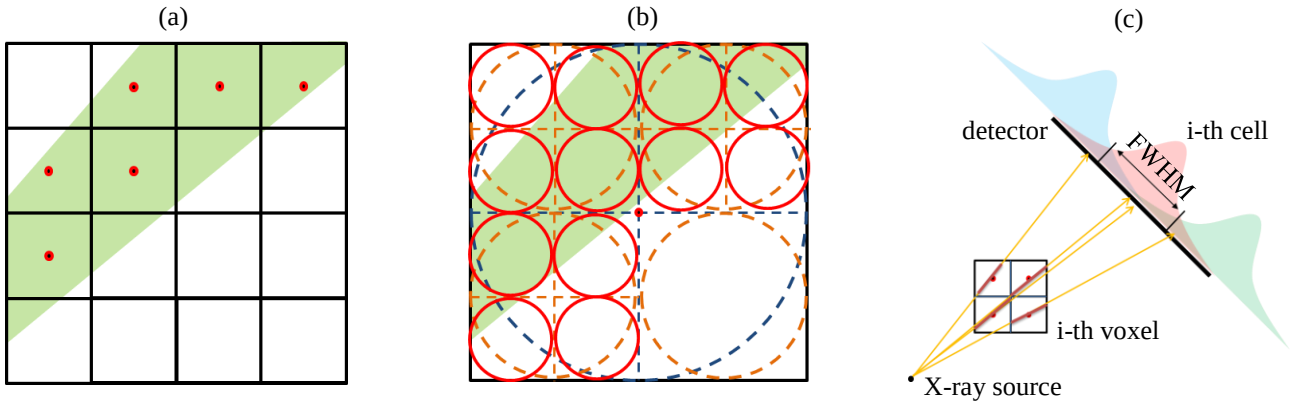


Figure 2. Approximate Volume Integral Methods. (a) Riemann sum approach, (b) Recursive sub-division approach, and (c) Multi-projection approach.

1. If d_u (d_d) is negative distance and its magnitude is larger than the half side of the cube, the beam is passing below (above) of the cube.
2. If d_l (d_r) is negative distance and its magnitude is larger than the half side of the cube, the beam is passing right (left) of the cube.

Here d_x is a signed distance from the center of the sub-voxel to a plane, x, and the subscripts $\{u, d, l, r\}$ are short-hand for the location of a plane as $\{up, down, left, right\}$. If any geometry configuration between a sub-voxel and a beam violates one of the non-overlap criteria, we perform another sub-division for the (sub-)voxel or count the voxel as part of the intersected volume, if it reaches the pre-defined number of sub-divisions.

3) Multi-projection approach

This approach is a bit different from the other approaches in that it does not compute the intersected volume between a voxel and a beam. Instead, the central points of all N^3 sub-voxels obtained as in Riemann sum approach are projected onto the detector. Then, the weight coefficient is computed as follows:

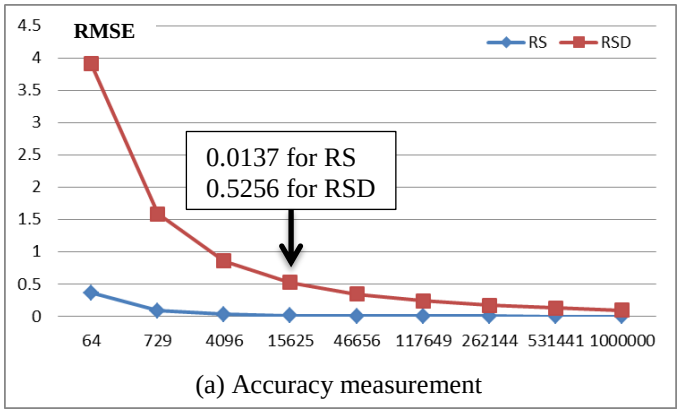
$$a_{ij} = \frac{1}{N^3} \sum_k l_k \times G(p_k; 0, \sigma^2)$$

where k is an index of a central point of a sub-voxel, l_k is the intersection length between the k-th sub-voxel and a zero-width ray passing through its central point, p_k is the projection location on a (flat) detector of the k-th central point and $G(\cdot)$ is a cell sensitivity kernel described by a zero mean Gaussian distribution. In our implementation, the variance of the cell sensitivity kernel is determined to have the side length of a cell as its FWHM. Also, the density value at each sub-voxel is obtained by using (tri-) linear interpolation and the projected values are scattered over nearby cells according to their sensitivities. Figure 2(c) shows an illustration of this approach for the 2D case.

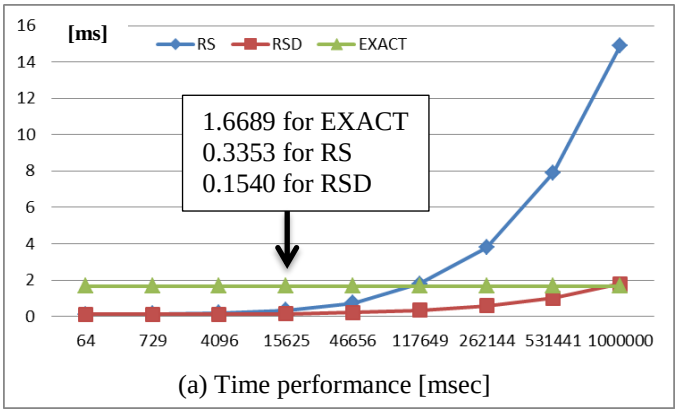
III. Results

We tested the proposed volume integration methods (VIMs) with a voxel-driven forward projection of a 8×256×16 sized uniform slab with 1×1×1 mm voxels, imaged at 45 degrees in conventional cone-beam CT geometry. The distances from the source to rotation axis and to the center of a flat detector were set to 400 mm and 500 mm, respectively. The detector has a size of 512×512 (1×1 mm bins). The projection image obtained using exact VIM is used as a reference for all other methods.

We first tested the Riemann sum (RS) and recursive sub-division (RSD) approaches by varying the number of sub-voxels. We varied the number of sub-voxels in RS by matching the total number of sub-voxels at the finest levels in RSD, to enable a fair comparison of accuracy and time performance between them. The RMSE metric is used to measure the accuracy of both approximate VIMs as follows:



(a) Accuracy measurement



(a) Time performance [msec]

Figure 3. Accuracy and time performance of approximate volume integral methods. (a) Accuracy and (b) Time performance. Note that x-axis in both measurements is the number sub-voxels in a voxel.

$$RMSE = \frac{1}{C}\sum_{i=0}^{I}\sqrt{\left(p_i^{exact} - p_i^{appro}\right)^2}$$

where I is total number of line integrals in a view, p_i is projected value at detector cell *i*, and the super-script denotes exact and approximate VIMs. The normalization factor C is set to the number of non-zero projection values in exact VIM.

Figure 3 shows the accuracy and time performance comparisons among RS, RSD and exact VIMs. As the RS method checks all sub-voxels in a voxel, it shows better accuracy than RSD; while RSD shows better time performance than RS because it quickly skips a portion of a voxel in which a beam does not intersect. Note that the time performance shown in Figure 3(b) is the average time to compute the exact or approximate intersected volume of a voxel.

We also visually inspect and compare the projection images obtained from the exact and the two approximate VIMs (RS and RSD) as shown in Figure 4. After having 15625 ($= 5^{3*2}$) sub-cubes in total for each voxel, both approximate VIMs results in almost similar visual appearance as exact method even though RSD has much higher RMS error. In addition, RSD shows about 2 times faster than RS and 10 times faster than exact methods.

Lastly, we tested the multi-projection method with the same cone-beam CT geometry and uniform slab. We note that this method, in fact, uses a line integral model, unlike the other methods, but it uses a higher level of subdivision than conventional such methods. Figure 4 at the bottom shows a projection of the slab using this method for the case where the number of sub-voxels id 15,625 (row 4 in Figure 4). The quality is quite similar but the projection is 1-2 orders of magnitudes faster due to the very simple projective mapping.

IV. Conclusion

In this paper, we have studied four volume integral methods (VIMs) to construct a more realistic and accurate CT system model. The exact VIM has provided a gold standard reference for developing the approximate methods. The approximate VIMs show very close accuracy to the exact one while having superior time performance (at most 10 times faster at visually acceptable level). Furthermore, the proposed approximate VIMs can be easily imported to the parallel processors like multi-CPU processors or GPU threads that would be quite complicated for exact methods. Projecting the sub-voxels centers to the projection plane and distributing their contribution turned out to give similar results but a much higher speed. In future work we would like to investigate this further. Finally, we also plan to reconstruct a full CT data set with the approximate VIMs and evaluate their performance.

Acknowledgments

This research was partially supported by NSF grant IIS-11732 and the MSIP(Ministry of Science, ICT and Future Planning), Korea, under the 'IT Consilience Creative Program (ITCCP)' (NIPA-2013-H0203-13-1001) supervised by NIPA (National IT Industry Promotion Agency).

References

[1] R.L. Siddon, "Fast calculation of the exact radiological path for a three-dimensional CT array," Med. Phys. 12(2):252-255, 1985.

[2] P.M. Joseph, "An improved algorithm for reprojecting rays through pixel images." IEEE Trans Med Imaging 1(3):192-196, 1983.

[3] B. De Man and S. Basu, "Distance-driven projection and backprojection in three dimensions," Phys. Med. Biol. 49(11), 2463-2475, 2004.

[4] Y. Long, J. A. Fessler, and J. M. Balter, "3D forward and back-projection for x-ray CT using separable footprints," IEEE Trans. Med. Imaging 29(11), 1839-1850, 2010.

[5] S. Zhang, D. Zhang, H. Gong, O. Ghasemalizadeh, G. Wang, and G. Cao, "Fast and accurate computation of system matrix for area integral model-based algebraic reconstruction technique," Optical Engineering 53(11), 113101, 2014

[6] I. E. Sutherland and G. W. Hodgman, "Reentrant Polygon Clipping," Communications of the ACM, 17(1), 32-42, 1974

[7] D. T. Lee and B. J. Schachter, "Two Algorithms for Constructing a Delaunay Triangulation," Int. J. Computer Information Sci., 9, 219-242, 1980

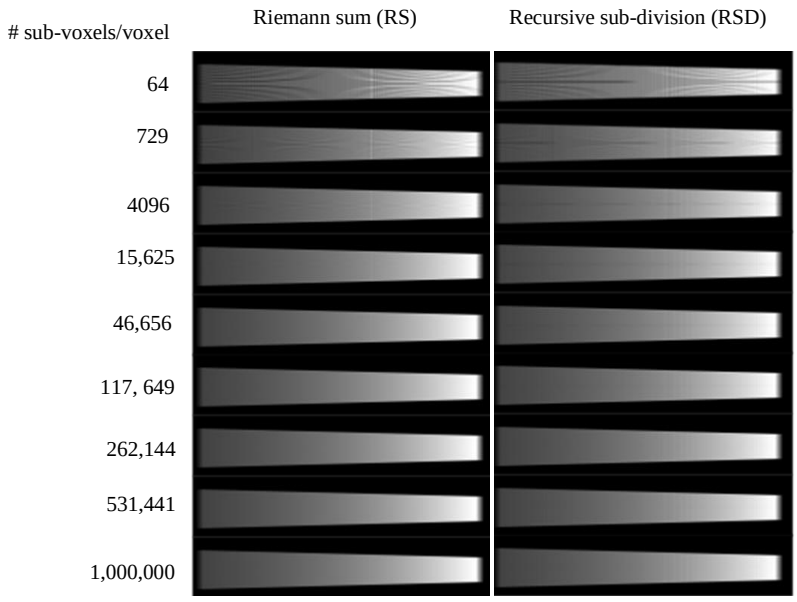


Figure 4. Visual inspection of projection images obtained from the exact and the three approximate VIMs. For RS and RDS, the number of sub-cubes in approximate methods is increasing from top to bottom as x-axis in Figure 3.