Efficient Area-Based Ray Integration Using Summed Area Tables and Regression Models

Sungsoo Ha, Heyi Li, and Klaus Mueller

***Abstract*—The increasing popularity of iterative reconstruction algorithms has raised the attention onto how to build more accurate, realistic CT system models. In our work, we model the CT projectors based on volume integrals. The higher computational complexity in computing the exact volume integration is hidden by memory-efficient, fast, and accurate look-up tables. For further reductions we also derive a simple linear regression model from the table. We demonstrate our ideas with data obtained with a fan-beam flat-detector CT system. We observe speed-ups of up to 30% while keeping a higher or at least similar image quality than existing advanced CT system models.**

***Index Terms*—CT system matrix, forward projection, line integral model, area-based model, volume integral model**

I. Introduction

With the increasing popularity of iterative reconstruction algorithms in the field of CT medical imaging, modeling a realistic CT system in software is becoming more crucial than ever. The CT system model can be represented by a huge matrix, **W**, whose columns correspond to the voxels subject to reconstruction (an N×1 vector) and the rows are the projections (or line integrals, an M×1 vector) that are measured by CT scanner. Then, each element, w_{ij}, of the matrix indicates the contribution of a voxel j to a detector cell i, so called weight coefficient. This gives rise to the linear algebra equation:

$$\mathbf{W} \cdot \mathbf{X} = \mathbf{P}$$

where **X** is the unknown image and **P** are the observed projection data. The process of calculating the line integrals is known as the forward projection and its reverse model, generally defined as the transpose of the forward projection, is known as back projection. As the size of the CT system matrix is enormous, the coefficients are usually computed on-the-fly during forward- and back projection.

The most intuitive and simplest way to compute the coefficient, w_{ij}, would be the line integral. It computes the intersection length between the *j*-th voxel and the *i*-th ray [1][2]. Here, a ray is depicted by a zero width line that connects the X-ray (point) source and the center of the detector cell for the ray-driven approach (or the center of a voxel for the voxel-driven approach). This kind of CT system modeling has low computational complexity but it can suffer from under-sampling and aliasing [3].

The other approach, which is much closer to a real CT system and which overcomes the sampling problems is the volume integration based approach. In this model, a single ray can be depicted by a 3D polygon (or 2D polygon for fan-beam) that connects the X-ray source with a detector cell. However, computing the intersected volume is not a trivial task and incurs high computational complexity. This precludes its use in iterative reconstruction routines where many intersections need to be computed.

Two well reported approaches exist which approximate the intersection volume. The first approach is the distance-driven (DD) method [3] that 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 to be more accurate than the DD methods, while keeping similar computational cost.

Recently, Ha et al. [5] introduced a method that computes the intersection volume exactly, and they also proposed three strategies that can approximate the volume at good accuracy. The approximate approaches aimed to make the volume integration process suitable for GPU-acceleration [6][7], using their massively parallel architecture to overcome the high computational cost of the exact ray-voxel intersection.

In this paper, we chose to go a different route with better computational efficiency while maintaining high accuracy. Here we were inspired by an established technique in computer graphics and texture mapping, called summed-area tables [8]. Our proposed method precomputes sampled intersection volumes and stores them in a summed area table such that unknown samples can be mapped into the table and be calculated using simple bilinear interpolation.

Our second approach derives a simple linear regression model from the table. While this method reduces accuracy, it is significantly faster. However, interestingly we can show that the reduced accuracy tends to occur in rays less frequently encountered and therefore reconstruction quality does not seem to suffer. We tested both methods with fan-beam X-ray flat-detector CT data and observed a computational cost reduction of up to 30% with a loss in image quality.

The remainder of our paper is organized as follows. Section II discusses details, Section III presents results. Finally, Section IV presents conclusions.

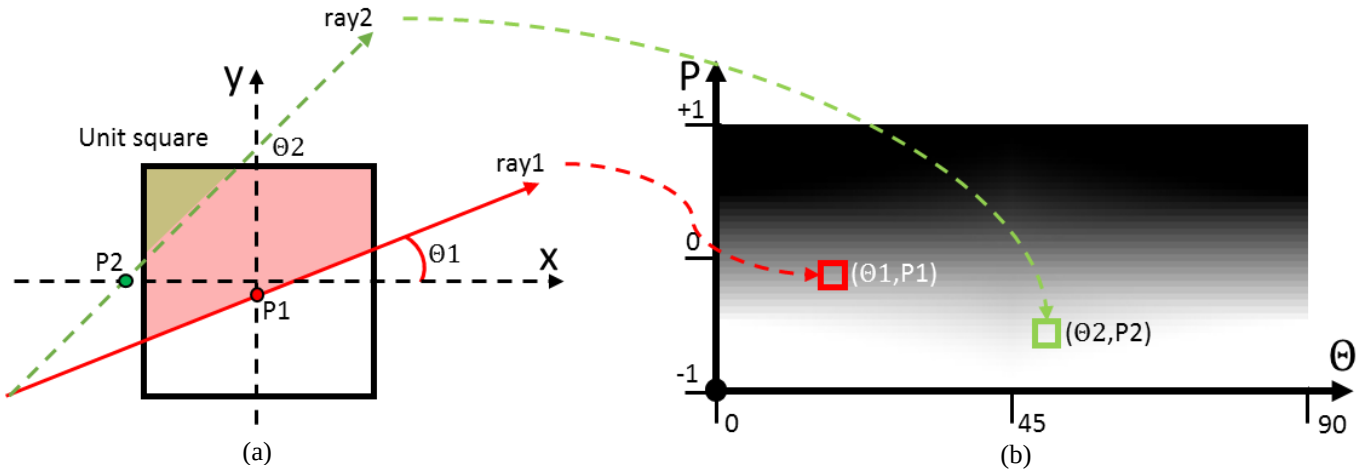

Figure 1. One side area look-up table. (a) Schematic view to compute intersection area between fan-beam and a unit pixel and (b) look-up table constructed with 1 degree and 0.05 mm step size resolution (99% accuracy).

II. Methodology

A. One-side Area Look Up Table (LUT)

We define a ray as a zero-width line that connects the X-ray point source to a point on a detector. Then, one side area is the area between a ray and unit square in one side either left or right. We parameterize the area by the ray incident angle, θ, which is measured from the x-axis to the ray, and an intersection point, P. The point is computed as an intersection point with the y-axis if the angle is less than or equal to 45 degrees. Otherwise it is computed with the x-axis to always have a single intersection point. The one-side area Look-Up Table (LUT) is a table that contains all possible one-side areas ranging from 0 to 360 degrees and -1 to 1 for the ray incident angle and intersection point, respectively. Then, given two parameters, θ and P, one side area is fetched from the LUT using bi-linear interpolation. The intersection area between a pixel and a fan-beam modeled by two rays can be efficiently computed by fetching two one side areas from the LUT and subtract one from the other. Figure 1 shows the schematic view of this with an example usages of the LUT.

Due to symmetry given the square area of a voxel, we only need to store from 0 to 90 degrees. We use step sizes of 1 degree and 0.05 mm to construct the LUT. The size of the LUT is, thus, 41×91 which takes up only about 15 K Bytes, which is trivial amount memory for most of modern computer. The accuracy of the LUT is measured by comparing it with analytical solutions of 1,000 random pairs of the two input parameters, θ and P. We find an accuracy of roughly 99% with a maximum error of 0.004.

B. One-side Area Regression Model

Figure 2 shows curves of the one side area LUT along the P-Area axis. Note that each curve corresponds to a column vector of the LUT constructed in previous section. We observe that all curves vary within the range of the curves extracted at 0 and 45 degrees. Approximation can be obtained by abstracting the curves with a piece-wise non-linear regression model. Here, we take a simple piece-wise linear regression model that can cause at most 0.1207 error (about 12% with 45° ray incident angle and $\pm$ 0.5 mm) to take advantage of low computational complexity as follows.

$$Area = \begin{cases} -P + 0.5 & , -0.5 < P < 0.5 \\ 0 & P \le -0.5 \\ 1 & P \ge 0.5 \end{cases}$$

We analyze the effect of the error by simulating a realistic set of forward projections over 360 degrees spaced by 1 degree. Note that in the area-based approach the forward and back projection are symmetric operators and so we only examine one. We draw an *error map* by computing the errors our regression model can make. In the LUT matrix view, we subtract the first column vector (0° ray incident angle) by the others. Then we measure the occurrence of each element in the LUT during projections, yielding a *frequency map*. This map represents how often each element is used during projections in a given CT system. The total number of fetches will be (# projections) × (# pixels) × (# detector cells affected by each pixel) × (at most 4 LUT elements for bilinear interpolation). The *average error* that can occur during projections is estimated by multiplying the error map and the frequency map. The total expected error is then computed by summing all values in the average error map. Figure 3 shows the three maps computed with the fan-beam flat-detector CT system and the average error is 1.0891 %.

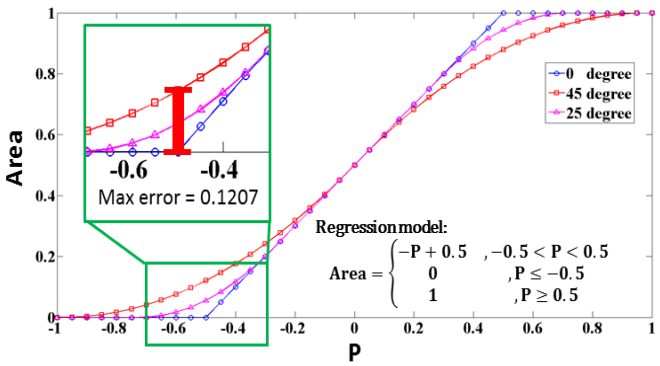

Figure 2. One side area piece-wise linear regression model

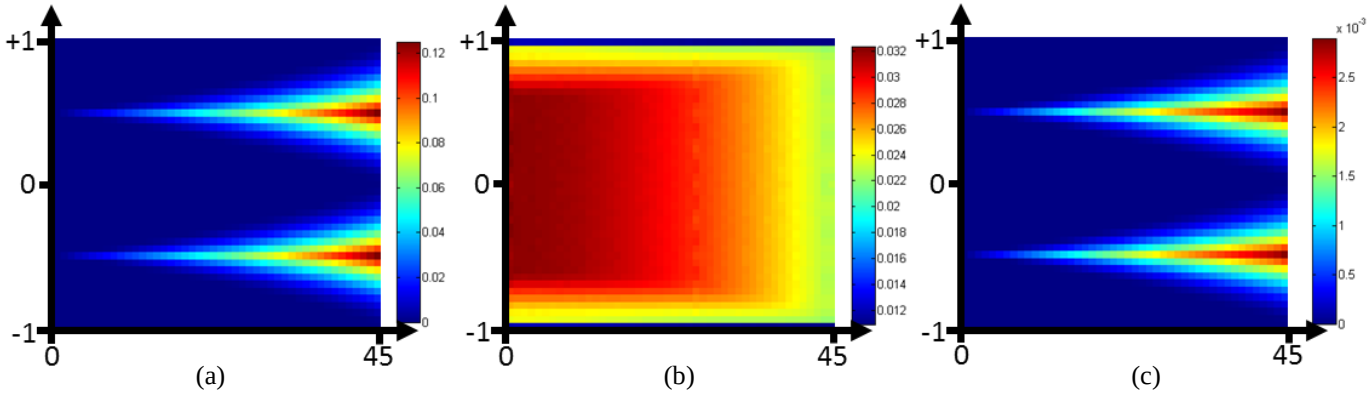

Figure 3. One side area piece-wise linear regression model. (a) error map, (b) frequency map and (c) average error map.

III. Results

We tested the two proposed CT system models with a fan-beam flat-detector X-ray CT system with a detector size of 1024 cells spaced by 0.384 mm. The source to detector distance is 1147.7 mm and the source to rotation center distance is 647.7 mm. The fan-angle is about 19.4 degrees. For the forward/back projection analysis, we used a 512×512 Shepp-Logan phantom data with 0.415 × 0.415 mm^2 pixel size. All experiments use 360 views uniformly distributed over 360 degrees and the forward/back projection operators are implemented using the pixel driven method. Note that all implementations are accelerated by an NVIDIA Tesla K40c.

A. Forward projection

In area-based CT system models, forward- and back-projection operators are symmetric. We measure the accuracy of our regression model for forward projection by comparing it with the LUT-based approach. We define the normalized root mean square error (NRMS) as

$$e(\phi) = \frac{1}{N}\sum_{i}^{N} \left\| F_{\phi}^{LUT}(i) - F_{\phi}^{REG}(i) \right\|_2$$

where $F_{\phi}(i)$ returns normalized forward projected value at a detector cell i at a projection angle, ϕ, such that

$$F_{\phi}(i) = \frac{\sum_{j=1}^{M} f(j) \cdot w_{ij}}{\sum_{j=1}^{M} w_{ij}}.$$

Here, $f(j)$ is a j-th pixel value, and N and M represent the number of detector cells and pixels. The w_{ij} is the contribution of a pixel j to a detector cell i that computes either by look-up table or regression model and superscript, *LUT* and *REG*, used to indicate them, respectively. Figure 4 shows the NRMS measurement over 360 views. As the same pattern is repeated every 90 degrees, we only show 0° to 90° projection angles. As the projection angle increases, there are more chances that a ray can have the incident angles, θ, around 45°, and it reaches the maximum occurrence around 30° and 64° projection angles as the combination effect with fan-angles (19.4°). At this time the error is about 0.002106 (about 2% error). After the peak point, the error goes down as the occurrence reduction of erroneous ray incident angles.

B. Within iterative CT reconstruction

The proposed methods were plugged into a simultaneous algebraic reconstruction technique (SART) framework that updates each projection at a time.

$$f_j^{k+1} = f_j^k + \lambda \cdot \frac{\sum_i \left[\frac{p_i - \sum_{i=1}^{N} w_{ij} \cdot f_j^k}{\sum_{i=1}^{N} w_{ij}} \right] w_{ij}}{\sum_i w_{ij}}$$

where f_j^k is reconstructed j-th pixel at k-th iteration and p_i is i-th projection data. The constant factor, λ, is update step size. Using the SART, we measure the performance in terms of time and reconstruction quality. We used Shepp-Logan

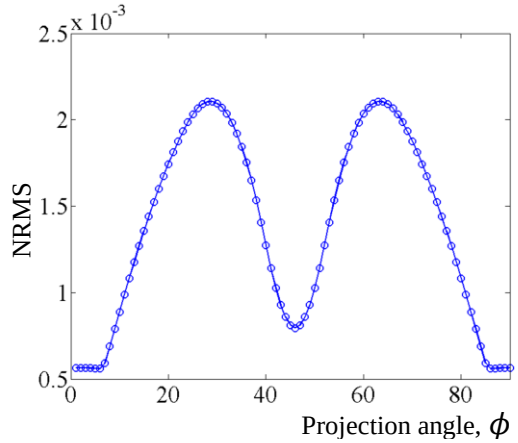

Figure 4. NRMS error comparison between look-up table and regression model approach as forward projectors.

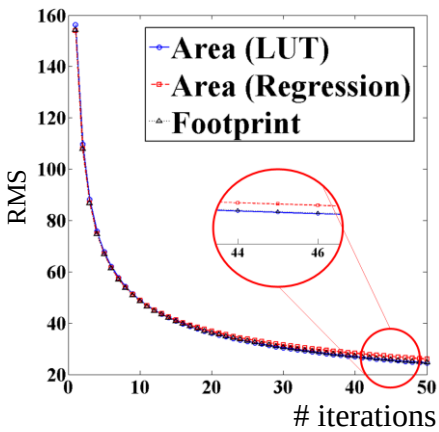

Figure 5. Convergence comparisons among three different projectors, LUT-, Regression model- and Footprint-based.

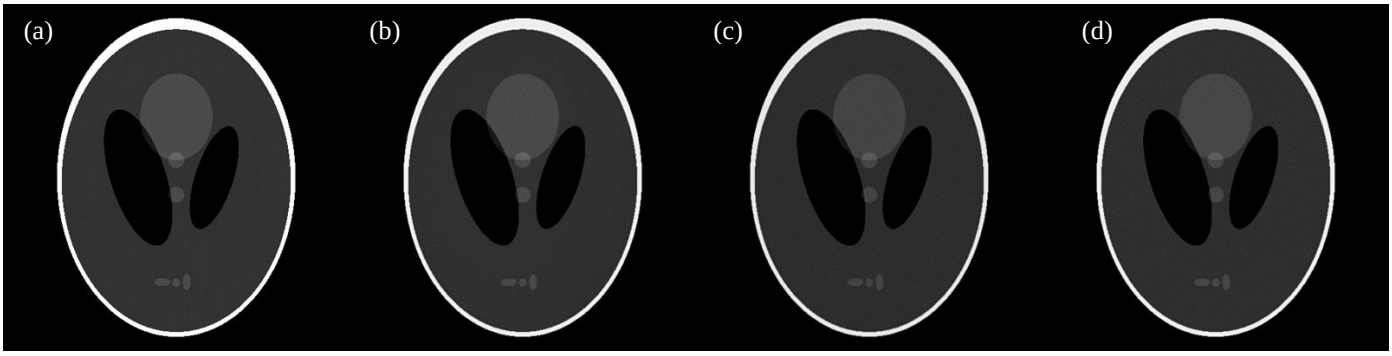

Figure 6. Visual comparisons. (a) Shepp-Logan phantom, (b) Footprint, (c) LUT and (d) Regression

phantom data for this, and the projection data were simulated using the LUT based approach as it has about 99% of accuracy to compute the intersection area compared to an analytical solution. Finally, we compared our methods with the state-of-the-art footprint-based approach [4]. In the following, the experimental results were obtained by running 200 iterations with 0.01 update step size.

Convergence: We measure the convergence rate among three different projection methods. Two are the proposed ones and the other is the separable footprint. To maintain fairness among different approaches, we define the root mean square (RMS) errors as follows:

$$RMS = \sum_{i=1}^{M} \left\| \hat{f}(i) - f_{sart}^{norm} \right\|_2$$

where $\hat{f}$ is the reference phantom data we used to generate projection data and f_{sart}^{norm} is reconstructed image normalized by the maximum value. Figure 5 shows the result. All methods have similar convergence rate and converge to similar solution. (at 45 iterations, 25.58, 25.72 and 27.05 for LUT, Regression and Footprint, respectively).

Visual Assessment: Figure 6 presents visual comparisons of reconstructed images after 200 iterations. As all methods are converged to a similar solution, the visual look is also similar to each other.

Time Performance: The SART consists of 4 kernels, Forward Projection (FP), Correct, Back Projection (BP) and Update. We measure the running time of each kernel. There are 360 projections and we run it 200 times. The average time is presented in Table 1. In our implementation, both projection operators were implemented based on the pixel-driven method. As the result, BP is faster than FP in all methods because it does not require atomic operations; while it is necessary for the FP to avoid race condition. For the FP, with LUT, we can achieve 10% of speed-up due to the reduced computational complexity and Regression can improve the time performance another 15%.as it does not require memory fetching operations for the LUT. For similar reasons, in the BP, the proposed methods are faster by about 15% and 30% for LUT and Regression, respectively.

TABLE I. TIME PERFORMANCE COMPARISON [MILLISECOND]

	FP	Correct	BP	Update
Footprint	0.215		0.170	
LUT	0.192	0.025	0.147	0.05
Regression	0.161		0.121	

IV. CONCLUSION

We described and studied two area-based methods to construct a more realistic and accurate CT system model. We find that both methods can outperform the separable footprint method in terms of time complexity without a loss in reconstruction image quality. In future work we would like to investigate this finding further in the context of cone-beam and helical CT.

ACKNOWLEDGMENT

This research was supported by the MSIP (Ministry of Science, ICT and Future Planning), Korea, under the "ICT Consilience Creative Program" (IITP-2015-R0346-15-1007) supervised by the IITP (Institute for Information & communications Technology Promotion). We also thank Medtronic, Inc. for their continued support.

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. Ha, A. Kumar, and K. Mueller, "A Study of Volume Integration Models for Iterative Cone-Beam Computed Tomography," Fully3D Image Reconstruction in Radiology and Nuclear Medicine, June 2015.

[6] F. Xu, K. Mueller, "Accelerating popular tomographic reconstruction algorithms on commodity PC graphics hardware," IEEE Trans. on Nuclear Science, 52(3):654-663, 2005

[7] F. Xu, K. Mueller, "Real-Time 3D Computed Tomographic Reconstruction Using Commodity Graphics Hardware," Physics in Medicine and Biology, 52:3405-3419, 2007.

[8] C. Franklin, "Summed-area tables for texture mapping," ACM SIGGRAPH computer graphics 18(3), 207-212, 1984