Presented at the SPIE Medical Imaging Conference, San Diego, Feb. 21-27, 1998

A Fast and Accurate Projection Algorithm for 3D Cone-Beam Reconstruction with the Algebraic Reconstruction Technique (ART)

Klaus Mueller[1,2], Roni Yagel[1], and John J. Wheller[2]

[1] Department of Computer and Information Science, The Ohio State University, Columbus, OH
[2] Department of Pediatrics, The Ohio State University, Columbus, OH

ABSTRACT

The prime motivation of this work is to devise a projection algorithm that makes the Algebraic Reconstruction Technique (ART) and related methods more efficient for routine clinical use without compromising their accuracy. While we focus mostly on a fast implementation of ART-type methods in the context of 3D cone-beam reconstruction, most of the material presented here is also applicable to speed up 2D slice reconstruction from fan-beam data.

In this paper, we utilize the concepts of the splatting algorithm, which is a well known and very efficient voxel-driven projection technique for parallel projection, and devise an extension for perspective cone-beam projection that is considerably more accurate than previously outlined extensions. Since this new voxel-driven splatting algorithm must make great sacrifices with regards to computational speed, we describe a new 3D ray-driven projector that uses similar concepts than the voxel-driven projector but is considerably faster, and, at the same time, also more accurate. We conclude that with the proposed fast projection algorithm the computational cost of cone-beam ART can be reduced significantly with the added benefit of slight gains in accuracy. A further conclusion of our studies is that for parallel-beam reconstruction, on the other hand, a simple voxel-driven splatting algorithm provides for more efficient projection.

Keywords: Algebraic Reconstruction Technique, ART, 3D cone-beam reconstruction, iterative reconstruction, reconstruction from projections, splatting, projection algorithm, forward projection, backward projection

1. INTRODUCTION

The field of 2D and 3D reconstruction methods can be roughly divided into two main categories. On one side there is the domain of direct methods that capitalize on the Fourier Slice Theorem [4], while on the other side lies the domain of iterative methods that seek to solve the reconstruction problem by solving a system of simultaneous linear equations. The most prominent member of the former group is the Filtered Backprojection (FBP) algorithm. Here, the reconstruction is achieved by filtering the projection images with a ramp filter in frequency space, and then backprojection the filtered projections onto a reconstruction grid [5]. The first and still most prevalent representative of the iterative methods is the Algebraic Reconstruction Technique (ART), attributed to Gordon et. al. [9], while another well-known ART-type method is Simultaneous ART (SART), proposed by Andersen and Kak [2]. In both methods, a reconstruction grid is iteratively updated by a projection-backprojection procedure until a convergence criterion is satisfied.

In clinical, slice-based CT, FBP is nowadays exclusively used, since it is non-iterative and therefore potentially faster than the iterative ART, at least in the case when the number of projections is identical. Note, however, that clinical CT scanners usually acquire more than 500 line projections per slice, which approximates the continuous form of the inverse Radon integral rather well. The true power of ART is revealed in cases where one does not have a large set of projections available, or when the projections are sparse or missing at certain orientations [1][13]. In contrast to FBP,

Further author information -
Klaus Mueller (correspondence): 1080 Carmack Rd, 270 Bevis Hall, Columbus, OH 43210; Email: klaus@chopin.bme.ohio-state.edu; WWW: http://chopin.bme.ohio-state.edu/~klaus/klaus.html; Phone: (614) 292-1555; Fax: (614) 292-7301
Roni Yagel: Email: yagel@cis.ohio-state.edu; WWW: http://www.cis.ohio-state.edu/~yagel
John J. Wheller: Email: Wheller.1@osu.edu

ART also allows the application of spatial and numerical constraints, based on *a-priori* information, during the reconstruction process, which potentially leads to a better definition and contrast of the reconstructed object [21]. Finally, noisy data are also handled better with ART-type methods [13][17]. More recently, the importance of ART was reinforced by Matej et. al. [17] who found, for the reconstruction from noisy PET data, that ART produces quantitatively better results than the more popular FBP and MLE (Maximum Likelihood Estimation) methods.

In the present work, we will focus on the 3D reconstruction from cone-beam data, as this is still an active area of research. However, much of our discussion is also valid for the 2D reconstruction from fan-beam data. Although a variety of general cone-beam algorithms based on FBP have been proposed [7][8][20], there are, as of today, no clinical 3D cone-beam scanners. The present literature for 3D cone-beam reconstruction with ART is mostly restricted to the reconstruction of high-contrast objects, such as encountered in computed angiography [18][21] (where we usually have a small number of projections) and reconstruction from PET [17] and SPECT [19] data (where the projections are usually noisy).

We thus see that there is a great potential for ART, especially in the growing domain of cone-beam applications. However, the slowness of ART is still an obstacle in its utility for real-life clinical application. A first step to make the iterative framework of ART faster and also more competitive with FBP for the general reconstruction case is the reduction of the overall cost required for an iteration.

To achieve this goal, we must improve the speed of ART's projection engine, as most of the computational expense of ART is spent for projection and backprojection. It turns out that the computational cost of this projection engine is greatly affected by the perspective nature of the cone-beam projection. In the following sections, we will give a detailed description of two new, highly accurate 3D projection algorithms, one voxel-driven and one ray-driven, and analyze their efficiency in both the parallel-beam and cone-beam setting. Although other voxel-driven projectors [23] and ray-driven projectors [15] have been described, these algorithms are only efficient for the parallel-beam case. Furthermore, our voxel-driven perspective projection algorithm is considerably more accurate than the one described by Westover [23]. Our ray-driven algorithm, on the other hand, is a 3D extension of the 2D algorithm proposed by Hanson and Wecksung [10].

Thus the outline of this paper is as follows. Section 2 gives a short recap on the workings of ART-type algorithms and describes previous work. Section 3 then describes a voxel-driven projection algorithm for cone-beam that is more accurate for perspective projection than existing ones, but does not improve the state of the art in terms of speed. Section 4 gives a new ray-driven projection algorithms for cone-beam ART that fulfills both goals: accuracy and efficiency. Finally, Section 5 presents some timings and results obtained with our ART testbed software.

2. PRELIMINARIES AND PREVIOUS WORK

In this section, we will give a brief review of those aspects of ART that are relevant to the work presented here. While ART originally was proposed for the 2D case, the mathematical notation translates trivially to the 3D case.

As was mentioned before, ART poses the reconstruction problem as a system of linear equations:

$$\begin{aligned} w_{11}v_1 + w_{12}v_2 + w_{13}v_3 + \ldots + w_{1N}v_N &= p_1 \\ w_{21}v_1 + w_{22}v_2 + w_{23}v_3 + \ldots + w_{2N}v_N &= p_2 \\ &\ldots \\ w_{M1}v_1 + w_{M2}v_2 + v_{M3} + \ldots + w_{MN}v_N &= p_M \end{aligned} \tag{1}$$

Here, the v_j are the values of the reconstruction grid elements (called *voxels* from now on), the p_i are the values of the pixels in the acquired projection images, and the weight factors w_{ij} represent the amount of influence a voxel j has on a ray passing from the source through image pixel i.

Usually one reconstructs on a cubic voxel grid with a side length of n voxel, thus the total number of voxels $N=n^3$ and the number of image pixels $M=S\cdot n^2$, where S is the number of projections. Also, for a 3D single-orbit reconstruction we generally assume a spherical reconstruction region. In this case we have $N = (1/6)\pi n$ unknown voxel values and $M = (1/4)\pi n^2$ relevant pixels per image. For the equation system (1) to be determined, the number of projections $S=N/M$ has then to be $0.67n$. This means that for n=128, 86 projections images are required. However, it is not always the case that S=0.67n. Sometimes (1) is overdetermined, or, more often, it is underdetermined. In either case, the large magnitude of (1) does not allow its solution by matrix inversion or least-squares methods. In addition, usually noise as well as sampling errors in the ART implementation do not provide for a consistent equation system anyhow. Thus an iterative scheme proposed by Kaczmarz [12] is used. Starting

from an initial guess for the volume vector, $\boldsymbol{V}=\boldsymbol{V}^{(0)}$, we select at each iteration step k, $k>0$ one of the equations in (1), say the one for p_i. A value $p_i^{(k)}$ is measured which is the value of pixel i computed using the voxel values as provided by the present state of the vector $\boldsymbol{V}=\boldsymbol{V}^{(k)}$. A factor related to the difference of $p_i^{(k)}$ and p_i is then distributed back onto $\boldsymbol{V}^{(k)}$ which generates $\boldsymbol{V}^{(k+1)}$ such that if a $p_i^{(k+1)}$ were computed from $\boldsymbol{V}^{(k+1)}$, it would be closer to p_i than $p_i^{(k)}$. Thus, we can divide each grid update into three phases: a projection step, a correction factor computation, and a backprojection step.

The correction process for one element of $\boldsymbol{V}$, i.e. v_j, can be expressed by:

$$v_j^{(k+1)} = v_j^{(k)} + \lambda \frac{p_i - \sum_{n=1}^{N} w_{in} v_n^k}{\sum_{n=1}^{N} w_{in}^2} w_{ij} \tag{2}$$

where λ is the *relaxation coefficient* typically chosen within the interval (0.0,1.0], but usually much less than 1.0 to dampen correction overshoot. This procedure is repeated in an iterative fashion for all equations in (1).

Instead of updating the volume on a ray-basis, SART [2] corrects the volume on an image-basis. This was shown to significantly reduce the noise artifacts that were observed with ray-iterative ART. The projection step of SART performs a summed volume rendering [11] of the reconstruction grid, then subtracts the rendered image from the acquired projection image, normalizes the result, and backprojects the image in an inverse volume rendering process. More formally, the SART correction equation is as follows:

$$v_j^{(k+1)} = v_j^{(k)} + \lambda \frac{\sum_{p_i \in P_\varphi} \left(\frac{p_i - \sum_{n=1}^{N} w_{in} v_n^{(k)}}{\sum_{n=1}^{N} w_{in}} \right) w_{ij}}{\sum_{p_i \in P_\varphi} w_{ij}} \tag{3}$$

In this equation, the correction term for voxel j depends on a weighted average of all rays of a projection $P\varphi$ that traverse the voxel j. (Here, φ denotes the orientation angle at which the projection was taken.)

The sum terms in the nominators of (2) and (3) require us to compute the integration of a ray across the volume. The integration process can be performed by using raycasting, i.e., sampling the volume at equidistant locations with an interpolation kernel h and accumulating the interpolated values. Since accurate integration requires many sampling points, this is very time consuming. A better way that allows a more efficient and more accurate evaluation of the ray integral was proposed by a number of authors [10][14][15][23]. It consists of reordering the ray integral so that each voxel's contribution to the integral can be viewed isolated from the other voxels. In this alternative volume decomposition, the interpolation kernel is placed at the voxel locations, and the volume grid is viewed as a field of overlapping interpolation kernels which, as an ensemble, make up the continuous object representation. A voxel j's contribution is then given by $v_j \cdot \int h(s)ds$, where s follows the integration of the interpolation kernel in the direction of the ray. Here, $\int h(s)ds$ represents a voxel weight factor in (1), (2) and (2). If the viewing direction is constant for all voxels or if the interpolation kernel is radially symmetric, we can pre-integrate $\int h(s)ds$, often analytically, into a lookup-table (also called the *kernel footprint*). We can then map all voxel footprints to the screen, scaled by the voxel value, where they accumulate into a projection image, as is done in the splatting approach of Westover [23]. Alternatively, one can use rays to intersect the lookup tables in volume space, again scale the indexed value by the voxel value, and accumulate the density integrals ray by ray [10][14][15]. The former approach we call voxel-driven splatting, which produces an image or an image region at a time and makes only sense in conjunction with an image-based correction algorithm, such as SART. The second approach is called ray-driven splatting and can be used with either ART or SART. Note, that the lookup-tables are also used to retrieve all other weight terms in (2) and (3). Moreover, backprojection is performed in a similar way than forward projection, only that here the voxels receive (corrective) energy, scaled by their weight factors, instead of emitting it.

The splatting approach has two advantages: (i) The ray integrals are calculated very accurately, since each footprint table entry

can be integrated analytically or with good quadrature, and (ii) the complexity for interpolation is reduced from O(n^3) in volume space to O(n^2) in image space. Fast incremental algorithms can then be used to index the footprint tables in volume space (in ray-driven splatting) or image space (in voxel-driven splatting).

While the concept of representing a volume by a field of interpolation kernels and pre-integrating a 3D kernel into a 2D footprint is common to all existing splatting implementations, the strategy chosen to map the footprint table onto the screen (in the voxel-driven approach) or to map the rays into the footprint table (in the ray-driven approach) varies. The mapping task is facilitated since we only use spherically symmetric kernels and cubic grids, which yields a circular footprint. For voxel-driven splatting, both Westover [23] and Matej and Lewitt [15] simply map the circular footprint to the projection screen for one voxel and use incremental shifts for the remaining voxels at that projection angle. This, however, is only correct for parallel projections, since in perspective projection the elliptical shape and size of the footprint is different for every voxel (More detail is given in Section 3). In the case of ray-driven splatting we again assume a spherically symmetric interpolation kernel. In one approach, Matej and Lewitt [15] decompose the voxel grid into a set of 2D slices. Here the orientation of the slices is that orientation most parallel to the image plane. Recall that a footprint is the pre-integrated kernel function in the direction of a ray, thus a footprint is not necessarily planar with the slice planes. The authors project this footprint function onto a slice plane, giving rise to an elliptical footprint. Since in parallel projection all rays for a given projection angle have the same angle with the volume slices, this re-mapped elliptical footprint can be used for all slices and all rays that are spawned for a given projection orientation. Simple incremental algorithms can be designed to trace a ray across the volume slices, computing all indices into the elliptical footprints that are intersected. However, for perspective projection, every ray has a different orientation, necessitating a footprint re-mapping for every ray, which is inefficient both to compute on the fly and to store. A more appropriate approach was outlined for the 2D case by Hanson and Wecksung [10]. These authors model a 2D ray as an implicit line equation. If one runs a line parallel to the ray across the center of a given voxel, then the offset difference of the equations of these two lines yield the perpendicular distance of the ray to the voxel center, which then can be used to index a 1D footprint table. Our ray-driven approach is a 3D extension of this algorithm, optimized for speed, that enables the efficient use of the same footprint table for all projection rays everywhere in the volume.

It should be mentioned that the choice of the interpolation kernel h varies in the various ART implementations. We will be using a kernel based on the Bessel-Kaiser window, as proposed by Matej and Lewitt [15]. Multidimensional Bessel-Kaiser functions have many desirable properties, such as fast decay for frequencies past the Nyquist rate and radial symmetry. It can also be tuned so that the kernel's frequency spectrum is at a minimum at multiples of the sampling frequency, where the signal's aliases are largest.

3. AN ACCURATE VOXEL-DRIVEN SPLATTING ALGORITHM FOR CONE-BEAM ART

Let us first introduce some terminology. As suggested by Crawfis and Max [6], we can think of the interpolation kernel footprint as a polygon with a superimposed texture map that is placed in object (volume) space. Here, the texture map is given by the projected kernel function, i.e. the array of line integrals. For the remainder of our discussion we will refer to the footprint in object space as the *footprint polygon*, while the projection of the footprint polygon onto the image plane will be called the *footprint image*. Recall that splatting accumulates the same value in a pixel on the image plane as a ray would accumulate when traversing the volume. Thus, when projecting the footprint polygon to obtain the line integral for the pixel in the footprint image we must ensure that we position the footprint polygon orthogonal to the direction of the sight-ray in object space. The line integrals are retrieved from the footprint table by indexing it at the ray-footprint polygon intersection point. Thus, for splatting to be accurate, the 2D footprint must be mapped to the pixel as if the ray emanating from the pixel had traversed it at a perpendicular angle. Only then does the looked-up pre-integrated integral match the true kernel integration of the ray. Westover's perspective extension to voxel-driven splatting violates this condition at three instances: (i) He does not align the footprint polygon perpendicularly to the voxel center ray when calculating the projected screen extent. Instead he aligns it parallel to the screen and stretches it according to the perspective viewing transform. (ii) When mapping the footprint to the screen pixels he uses a linear transform instead of a perspective one. (iii) The footprint polygon is not rotated for every mapped pixel such that the corresponding pixel ray traverses it at a perpendicular angle. While the error for the last approximation is rather small, the error ratio of the computed footprint table index vs. the correct index in the first two approximations can be up to 1.15 for a 30° cone half angle.

Consider now Fig. 1, where we illustrate a new and accurate solution for perspective voxel-driven splatting. For simplicity of drawing, we show the 2D case only. Note that the coordinate system is fixed at the eye point. To splat a voxel $v_{x,y,z}$, it is first rotated about the volume center such that the volume is aligned with the projection plane. Then the footprint polygon is placed orthogonal to the vector starting at the eye and going through the center of $v_{x,y,z}$. Note that this yields an accurate line integral

only for the center ray, all other rays traverse the voxel kernel function at a slightly different orientation than given by the placement of the 2D (1D in Fig. 1) footprint polygon in object space. Thus the first error in Westover's approximation still survives. This error, however, can be shown to be less than 0.01, even for voxels close to the source.

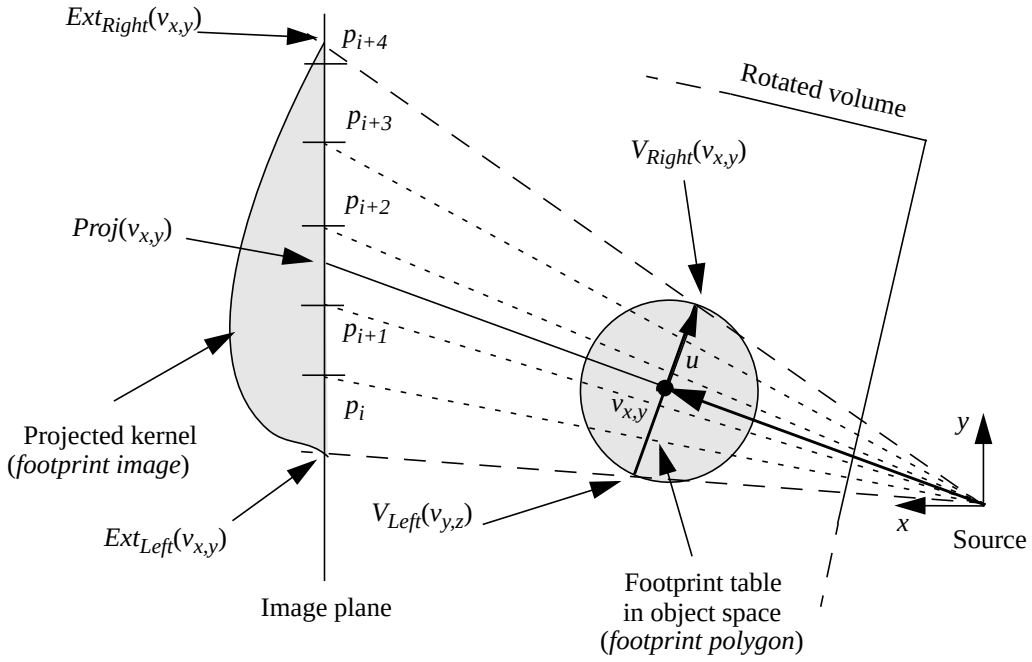


Fig. 1. Perspective voxel-driven splatting: First, the footprint polygon of voxel $v_{x,y}$ is mapped onto the image plane, then the affected image pixels p are mapped back onto the footprint table.

The coefficients of the footprint polygon's plane equation are given by the normalized center ray (the vector *source*-$v_{x,y,z}$). From this equation we compute two orthogonal vectors *u* and *w* on the plane (only *u* is shown in Fig. 1). Here, *u* and *w* are chosen such that they project onto the two major axes of the image. Using *u* and *w*, we can compute the spatial *x,y,z* positions of the four footprint polygon vertices in object space ($V_{Right}(v_{x,y})$ and $V_{Left}(v_{x,y})$ in the 2D case depicted in Fig. 1). These four vertices are perspectively projected onto the image plane. This yields the rectangular extent of the footprint image, aligned with the image axes ($Ext_{Right}(v_{x,y})$ and $Ext_{Left}(v_{x,y})$ in the 2D case). By expressing the intersections of the pixel rays with the footprint polygon in a parametric fashion, we can then set up an incremental scheme to relate the image pixels within the footprint image with the texture map entries of the footprint table.

The computational effort to map a footprint polygon onto the screen and to set up the incremental mapping of the pixels into the footprint table is quite large: Almost 100 multiplications, additions, and divisions, and two square root operations are necessary. No incremental scheme can be used to accelerate the mapping of neighboring grid voxels. The high cost is amplified by the fact that the expensive mapping has to be done at $O(N)=O(n^3)$. And indeed, in our implementation, perspective projection was more than twice as expensive than parallel projection.

4. A FAST AND ACCURATE RAY-DRIVEN SPLATTING ALGORITHM FOR CONE-BEAM ART

We saw in the previous section that perspective voxel-driven splatting can be made accurate, however, the expense of perspective voxel-driven splatting seems prohibitive for use in cone-beam reconstruction. In this section we take advantage of the fact that, in contrast to voxel-driven approaches, ray-driven methods are generally not sensitive to the non-linearity of the perspective viewing transform. It can thus be expected that ray-driven splatting is more advantageous to use in the perspective cone-beam situation. The new ray-driven approach is in some respect a 3D extension to the 2D algorithm sketched by Hanson and Wecksung [10] and will work both for ART and SART.

In ray-driven splatting, voxel contributions no longer accumulate on the image plane for all pixels simultaneously. In contrast, each pixel accumulates its raysums separately, which makes it also more suitable for ART than voxel-driven splatting. Our algorithm proceeds as follows. The volume is divided into 2D slices formed by the planes most parallel to the image plane. When a sight-ray is shot into the 3D field of interpolation kernels, it stops at each slice and determines the range of voxel ker-

nels within the slice that are traversed by the ray. This is shown in Fig. 2 for the 2D case: The ray originating at pixel p_i pierces the volume slice located at x_s at $y=y(i,x_s)$. The voxel kernels within the slice x_s that are intersected by the ray are given by the interval [Ceil($y_{Left}(i,x_s)$), Floor($y_{Right}(i,x_s)$)]. We compute $y_{Right}(i,x_s)$ as:

$$y_{Right}(i, x_s) = y(i, x_s) + \frac{extent_{kernel}}{\cos(\alpha)} \tag{4}$$

where α is the inclination of the ray. The computation for $y_{Left}(i,x_s)$ is analogous. After determining the active voxel interval $[y_{Left}(i,x_s), y_{Right}(i,x_s)]$ we must compute the indices into the voxel footprint table. This can be efficiently implemented by realizing that the index into the footprint table of a grid voxel v located at coordinates (y_v,x_v) is given by the distance dr of the two parallel lines (planes in 3D) that traverse v's centerpoint and the slice intersection point of the ray at $y(i,x_s)$, respectively (see Fig. 2b). One finds:

$$dr = a \cdot x_s + b \cdot y(i, x_s) - a \cdot x_s - b \cdot y_v = (b \cdot (y(i, x_s) - y_v)) \tag{5}$$

where a and b are the coefficients of the implicit line equation $a \cdot x_s + b \cdot y(i, x_s) = 0$ of the ray and are also given by the components of the (normalized) ray vector. Maintaining the variables $y_{Left}(i,x)$, $y_{Right}(i,x)$, and dr along a ray can all be done using incremental additions. f

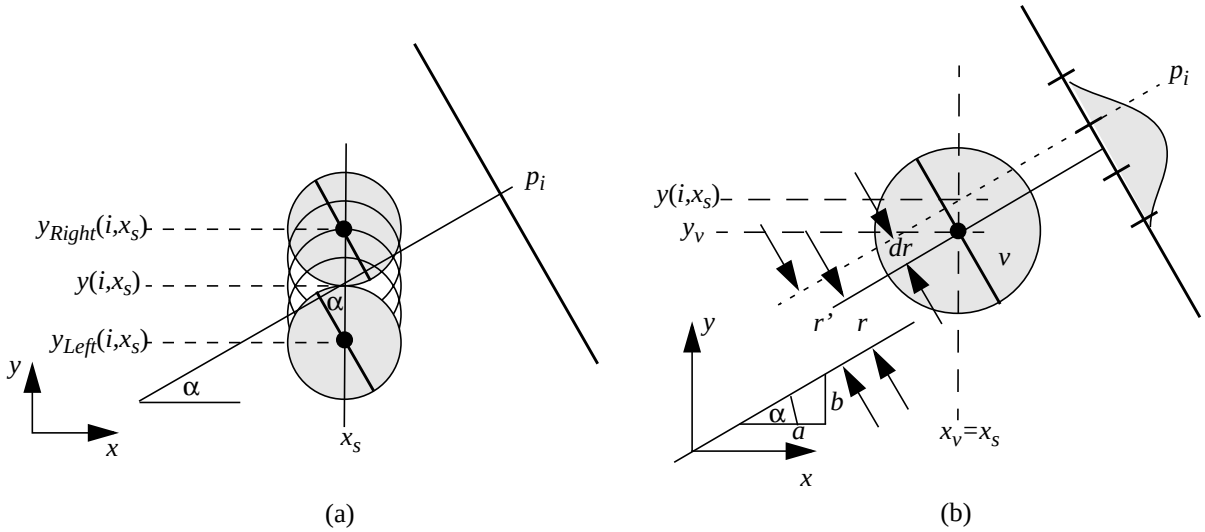


Fig. 2. Ray-driven splatting: (a) Determining the range of voxels within a given compositing plane that are traversed by a ray originating at pixel p_i. (b) Computing the index dr into the footprint table.

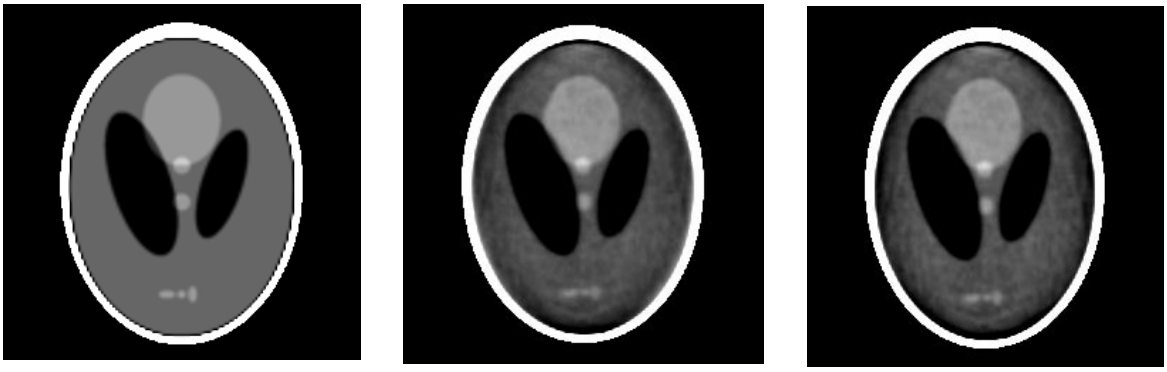
For the 3D case, we need to replace the linear ray by two planes. A 3D ray is defined by the intersection of two orthogonal planes cutting through the voxel field. The normal for one plane is computed as the cross product of the ray and one of the image plane axis vectors. The normal of the second plane is computed as the cross product of the ray and the normal of the first plane. Thus, the two planes are orthogonal to each other and are also orthogonal to the voxel footprint polygons. Thus the ray pierces the footprint polygon in a perpendicular fashion, as required. Intersecting the horizontal plane with a footprint polygon and using plane equations in the spirit of (5) results in the horizontal row index dr_{row} into the footprint table, while the intersection with the vertical plane yields the vertical column index dr_{col}. Using these two indices, the value of the ray integral can be retrieved from the 2D footprint table.

There are now three nested loops: The most outer loop sets up a new ray to pierce the volume, the next inner loop advances the ray across the volume slice by slice and determines the set of voxels traversed per slice, and finally, the most inner loop retrieves the voxel contributions from the footprint tables. For perspective projection, the plane equations have to computed for every ray. This amounts to about 50 extra additions, multiplications, and divisions, and three square roots per pixel. The cost of advancing a ray across the volume and determining the footprint entries is comparable to the cost of rotating a kernel and splatting it onto the image plane in the orthographic voxel-driven approach. The ray-driven approach changes the splatting algorithm from voxel order to pixel order. Thus, the most outer loop is of $O(n^2)$. This has the advantage that the complexity of any

extra work that has to be done for perspective projection (e.g. recomputing the two planes that define the ray in 3D) is roughly one order of magnitude less than in voxel-driven splatting. Note also that ray-driven splatting does not introduce inaccuracies. As a matter of fact, it prevents the indexing errors in the voxel-driven approach by design.

5. RESULTS

Fig. 3a shows a slice of the 3D extension of the Shepp-Logan phantom [22] (see [3] for details). Fig. 3b shows the same slice reconstructed from 20˚ cone-beam data with ART using the ray-driven splatting algorithm presented in the previous section. Fig. 3c shows the same slice reconstructed with SART, also using ray-driven splatting.



(a) (b) (c)

Fig. 3. (a) A slice from the 3D extension of the Shepp-Logan phantom [22], as described in [3], (b) the same slice reconstructed with ART and (c) reconstructed with SART. All reconstructions were performed for a cone angle γ=20˚, the volume was initialized to $\boldsymbol{V^{(0)}=0}$, S=80 projections, and reconstruction was performed for 3 iterations on a 128^3 grid. The relaxation coefficient for the ART reconstructions was λ=0.08, for SART λ=0.3.

Table 1 lists the run-times for SART and ART for both parallel-beam and cone-beam reconstruction. The runtimes were obtained on an SGI Indigo^2 workstation and refer to a reconstruction on a 128^3 grid, based on 80 projections with a cone angle of 40˚. In this table we see that for parallel-beam reconstruction with SART the voxel-driven approach is about 33% faster than the ray-driven approach. Hence, it is more advantageous in the parallel-beam case to perform the grid projection in object-order (i.e. to map the footprint polygons onto the screen) than to perform the projection in image-order (i.e. traverse the array of footprint polygons by the pixel rays). The computational savings in the voxel-driven algorithm for parallel-beam projection come from the fact that here the footprint-screen mapping is much simpler than the mapping described in Section 3, i.e. the mapping does not undergo perspective distortion.

In cone-beam reconstruction, however, the situation is reversed in favor of the ray-driven projector. Here, the speedup for using the ray-driven projector over the voxel-driven projector in SART is about 2.4. For ART, the use of the image-based voxel-driven splatting algorithm is not practical anyhow.

In Table 1 we also observe that SART is about 13% slower than ART. This is rooted in the fact that SART requires additional CPU cycles for averaging the ray contributions for each voxel and also incurs overhead to handle the temporary accumulation volume data.

6. CONCLUSIONS

The prime motivation of this work was to devise techniques that make ART and SART more efficient for routine clinical use, while not compromising their accuracy. Since the projection algorithm represents the main source of computations, we mainly focused on this portion of the algebraic algorithms in this paper. First, we described a cone-beam extension to Westover's voxel-driven parallel-beam splatting algorithm [23]. This new extension removes almost all inaccuracies of previously outlined extensions of that sort. Then, we analyzed existing ray-driven projectors in terms of their suitability for perspective cone-beam reconstruction. It was found that generally a ray-driven algorithm is far more suitable for the perspective cone-beam projection case than a voxel-driven splatting algorithm. It was also found that most of the existing ray-driven algorithm were not applicable for the special needs of cone-beam reconstruction. We then extended a conceptually existing 2D ray-driven splatting algorithm into

Method	Beam geometry	Splatting method	T_{corr} (sec)	T_{iter} (hrs)	T_{3iter} (hrs)
SART	parallel	voxel-driven	35.3	0.78	2.35
SART	parallel	ray-driven	47.1	1.04	3.14
SART	cone	voxel-driven	144.9	3.22	9.66
SART	cone	ray-driven	60.9	1.35	4.05
ART	cone	ray-driven	54.2	1.20	3.60

Table 1. Run-times for SART, using both voxel-driven and ray-driven splatting, and for ART using ray-driven splatting (voxel-driven splatting is not applicable for ART). (T_{corr}: time for one grid correction step, consisting of one projection and one backprojection, T_{iter}: time for 1 iteration (assuming 80 projection images, a 128^3 grid, and a cone angle of 40° in the cone-beam reconstruction), T_{3iter}:time for 3 iterations, the minimum time to obtain a reconstruction of good quality. Timings were obtained on a SGI Iris Indigo^2 with a 200MHz RS4000 CPU.)

3D and optimized it for speed and accuracy. In addition to its benefits in speed, this new ray-driven 3D splatting algorithm has none of the inaccuracies of Westover's perspective extension to splatting. Timing experiments revealed that while ray-driven splatting is considerably more efficient than voxel-driven splatting for cone-beam reconstruction, this situation is reversed for the reconstruction from parallel-beam data where a simple, parallel-beam voxel-driven splatting algorithm provides for a more efficient projection. These same relationships also exist in the context of fan-beam vs. parallel-beam 2D reconstruction.

We should also mention that the projection methods outlined in this paper for cubic grids also fully extend to the dodecahedral or body-centered grids that were proposed by [16]. These grids were shown to reduce the number of voxels to be processed by about 30%. Since the dodecahedral grids are really just a stack of interleaved square grids, the incremental grid traversal algorithms have to be modified only slightly.

7. ACKNOWLEDGEMENTS

The authors would like to thank GE for supporting this project under CCH grant #950925 and OSU grant #732025.

8. REFERENCES

[1] A. H. Andersen, "Algebraic Reconstruction in CT from Limited Views," *IEEE Trans. Med. Img.*, vol. 8, no.1, pp. 50-55, 1989.

[2] A.H. Andersen and A.C. Kak, "Simultaneous Algebraic Reconstruction Technique (SART): a superior implementation of the ART algorithm," *Ultrason. Img.*, vol. 6, pp. 81-94, 1984.

[3] C. Axelson, "Direct Fourier methods in 3D reconstruction from cone-beam data," *Thesis*, Linkoping University, 1994.

[4] R.N. Bracewell, *The Fourier Transform and Its Applications*, 2nd ed., McGraw-Hill, Inc., 1986.

[5] Z.H. Cho, J.P. Jones, and M. Singh, *Foundations of Medical Imaging*, Wiley, 1993.

[6] R. Crawfis and N. Max, "Texture splats for 3D scalar and vector field visualization," *Visualization'93*, pp. 261-266, 1993.

[7] L.A. Feldkamp, L.C. Davis, and J.W. Kress, "Practical cone beam algorithm," *J. Opt. Soc. Am.*, pp. 612-619, 1984.

[8] P. Grangeat, "Mathematical framework of cone-beam 3D reconstraution via the first derivative of the Radon transform," *Proc. Mathematical Models in Tomography (Oberwohlfach, 1990), Springer Lecture Notes in Mathematics, 1497*, pp. 66-97.

[9] R. Gordon, R. Bender, and G.T. Herman, "Algebraic reconstruction techniques (ART) for three-dimensional electron microscopy and X-ray photography," *J. Theoretical Biology*, vol. 29, pp. 471-481, 1970.

[10] K.M. Hanson and G.W. Wecksung, "Local basis-function approach to computed tomography," *Applied Optics*, Vol. 24, No. 23, 1985.

[11] A. Kaufman, Ed., *Volume Visualization*, IEEE Press 1991.

[12] S. Kaczmarz, "Angenäherte Auflösung von Systemen linearer Gleichungen," *Bull. Int. Acad. Pol. Sci. Lett.*, A, vol. 35, pp. 335-357, 1937.

[13] A.C. Kak and M. Slaney, *Principles of Computerized Tomographic Imaging*. IEEE Press, 1988.

[14] R.M. Lewitt, "Alternatives to voxels for image representation in iterative reconstruction algorithms," *Phys. Med. Biol.*, vol. 37, no. 3, pp. 705-715, 1992.

[15] S. Matej and R.M. Lewitt, "Practical considerations for 3-D image reconstruction using spherically symmetric volume elements," *IEEE Trans. Med. Img.*, vol. 15, no. 1, pp. 68-78, 1996.

[16] S. Matej and R.M. Lewitt, "Efficient 3D grids for image reconstruction using spherically-symmetric volume elements," *IEEE Trans. on Nucl. Sci.*, vol. 42, no 4, pp 1361-1370, 1995.

[17] S. Matej, G.T. Herman, T.K. Narayan, S.S. Furuie, R.M. Lewitt, and P.E. Kinahan, "Evaluation of task-oriented performance of several fully 3D PET reconstruction algorithms," *Phys. Med. Bio*l, Vol. 39, pp. 355-367, 1994.

[18] R. Ning and S.J. Rooker, "Image intensifier-based volume angiography imaging system: work in progress," *Proc. SPIE, vol. 1896, Medical Imaging 1993: Physics of Medical Imaging*, pp. 145-155, 1993.

[19] D. Ros, C. Falcon, I Juvells, and J. Pavia, "The influence of a relaxation parameter on SPECT iterative reconstruction algorithms," *Phys. Med. Biol.*, no. 41, pp. 925-937, 1996.

[20] B. Smith, "Image reconstruction from cone-beam projections: necessary and sufficient conditions and reconstruction methods," *IEEE Trans. Med. Img.*, vol. 4, no. 1, pp. 14-25, 1985.

[21] D. Saint-Felix, Y. Trousset, C. Picard, C. Ponchut, R. Romeas, A. Rougee, "In vivo evaluation of a new system for 3D computerized angiography," *Phys. Med. Bio*l, Vol. 39, pp. 583-595, 1994.

[22] L.A. Shepp and B.F. Logan, "The Fourier reconstruction of a head section," *IEEE Trans. Nucl. Sci.*, vol. NS-21, pp. 21-43, 1974.

[23] L. Westover, "Footprint evaluation for volume rendering," *Computer Graphics (SIGGRAPH)* , vol. 24, no. 4, pp. 367-376, 1990.