AIM

Association for Information and Image Management

1100 Wayne Avenue, Suite 1100
Silver Spring, Maryland 20910

301/587-8202

Centimeter

Inches

MANUFACTURED TO AIM STANDARDS
BY APPLIED IMAGE, INC.

1 of 1

Conf-940723--4

UCRL-JC-117450
PREPRINT

Tomographic Image Reconstruction and Rendering with Texture-Mapping Hardware

Stephen G. Azevedo
Brian K. Cabral
James Foran

This paper was prepared for submittal to the
SPIE - Mathematical Methods in Medical Imaging III Conference
San Diego, CA
July 24-29, 1994

July 1994

This is a preprint of a paper intended for publication in a journal or proceedings. Since changes may be made before publication, this preprint is made available with the understanding that it will not be cited or reproduced without the permission of the author.

MASTER

dj
DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

DISCLAIMER

This document was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor the University of California nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial products, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or the University of California. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or the University of California, and shall not be used for advertising or product endorsement purposes.

Tomographic image reconstruction and
rendering with texture-mapping hardware*

Stephen G. Azevedo

Lawrence Livermore National Laboratory
P.O. Box 808 - L-333, Livermore, CA 94551
(azevedo3@llnl.gov)

Brian K. Cabral
James Foran

Silicon Graphics Computer Systems
2011 N. Shoreline Blvd. - M-405, Mountain View, CA 94043-1389
(cabral@sgi.com) (foran@asd.sgi.com)

ABSTRACT

The image reconstruction problem, also known as the inverse Radon transform, for x-ray computed tomography (CT) is found in numerous applications in medicine and industry. The most common algorithm used in these cases is filtered backprojection (FBP), which, while a simple procedure, is time-consuming for large images on any type of computational engine. Specially-designed, dedicated parallel processors are commonly used in medical CT scanners, whose results are then passed to a graphics workstation for rendering and analysis. However, a fast direct FBP algorithm can be implemented on modern texture-mapping hardware in current high-end workstation platforms. This is done by casting the FBP algorithm as an image warping operation with summing. Texture-mapping hardware, such as that on the Silicon Graphics Reality Engine² (TM), shows around 600 times speedup of backprojection over a CPU-based implementation (a 100 Mhz R4400 in our case). This technique has the further advantages of flexibility and rapid programming. In addition, the same hardware can be used for both image reconstruction and for volumetric rendering. Our technique can also be used to accelerate iterative reconstruction algorithms. The hardware architecture also allows more complex operations than straight-ray backprojection if they are required, including fan-beam, cone-beam, and curved ray paths, with little or no speed penalties.

1. INTRODUCTION

The image reconstruction problem common to medical and industrial computed tomography (CT) is similar to the three-dimensional rendering problem. Because of this fact, modern graphics hardware for performing fast volume rendering is also well suited for CT data processing. Our interest in three-dimensional (3D) industrial CT¹, and the graphical rendering of the resultant volumes, has led us to collaborate on a fast solution to these related problems.

In CT, radiation (usually x-ray) penetrates an object and is attenuated by the internal structure. The transmitted/attenuated beam is detected after passing through the object (or human body) and digitally captured onto a computer. By rotating around the object, a series of two-dimensional (2D) projections of the 3D volume are produced. A schematic of this operation for a divergent radiation pattern, or *cone beam*, from the x-ray source point *S* to the detector plane is shown in Figure 1. Motion of the source and detector plane about the object is typically done on a circular trajectory. At one angular position, the discrete data points measured on the detector plane are collectively called a projection. All projections are then processed to reconstruct the 3D volume from its projections. This process is called tomographic reconstruction, and can be accomplished with a variety of methods.²⁻⁵ Once a volume is reconstructed, it can be visualized using volume rendering techniques.⁶⁻⁸

Traditionally, these two operations, reconstruction and rendering, have been considered separate independent algorithms. It can be shown, however, that they have the same mathematical and algorithmic form. In addition, since efficient algorithms for volume rendering have been demonstrated using hardware texture mapping and summing buffers, this same hardware can be used for CT reconstruction. That is, the filtered backprojection algorithm for CT reconstruction can be reformulated into an algorithm that uses texture mapping in combination with an accumulation or summing buffer.

* Work performed under auspices of the U.S. Department of Energy by the Lawrence Livermore National Laboratory under contract No. W-7405-ENG-48.

Figure 1. Cone-beam computed tomography orientation and geometry. X-rays are generated by source S , penetrate the object and are detected by a 2D radiographic imager at the detector plane. The source and detector move along a circular trajectory.

There are three levels of complexity that we will identify in tomographic reconstruction; they are, in order of least to most complex, parallel-beam, fan-beam, and cone-beam problems. The latter is inherently a 3D problem, as is shown in Figure 1, and is analogous to *perspective* volume rendering where S is the viewing point. In all cases described here, we will assume a circular trajectory of S . The fan-beam problem is the 2D case of cone-beam, where the divergent beam is restricted to the x - y plane. Physically, the x-rays are collimated to lie in the plane of a 1D detector array ($z=0$), and that plane is aligned perpendicular to the rotational axis. A 3D volume is generated by acquiring multiple 2D planes independently.

The parallel-beam case can be thought of as identical to the fan-beam or cone-beam problems with S moved away from the origin to infinity. The mathematics is simplified and the reconstruction problem is also formulated in 2D. However, because the lines of radiation are parallel in all planes, the physical detection problem can be done in 3D without acquiring independent planes. The graphics analog to the parallel-beam problem, then, is *orthographic* volume rendering. Note that there is no graphical analog to the multi-plane fan-beam case.

For fan- and cone-beam problems, tomographers often distinguish between straight or curved detector planes (with equi-spaced or equi-angular detector elements).⁹ In the interest of simplicity, we will not make this distinction here. Our focus will be on straight (for fan-beam) or planar (for cone-beam) detectors. However, the mathematics and implementation for curved detectors would be similar.

We will develop the mathematical bases for the reconstruction and rendering operations and show how texture-mapping hardware can solve them both with significant improvements in speed (up to three orders of magnitude). We will also show how the volume rendering operation, as described in [7] and implemented in current workstation hardware, is a generalized form of the forward CT problem known as the Radon transform. This is an important result for CT systems designers because the same hardware can be used for different steps in the processing chain. Special-purpose hardwired backprojectors are no longer needed to accelerate the reconstruction, which is usually the slowest step. Results will be given for an example parallel-beam reconstruction problem.

2. PARALLEL-BEAM FORMULAS

The 3D cone-beam reconstruction and perspective rendering problems shown in Figure 1 are the most complex cases of volume processing. We will begin instead with a discussion of the simpler parallel or orthographic cases first, followed by

the cone-beam problem in the next section. The mathematics will be given without proof, but for a more rigorous developments, see [2] or [3].

3.1 2D Radon Transform

Consider a 2D object function of finite extent, $f(x,y)$, that represents the x-ray attenuation coefficient at each spatial position as shown in Figure 2. Then the 2D Radon transform of that function is defined as follows:

$$g(s,\theta) = \int_{-\infty-\infty}^{\infty} \int_{-\infty-\infty}^{\infty} f(x,y) \delta(x\cos\theta + y\sin\theta - s) dx dy \quad (1)$$

where s and θ define a line in the plane as shown in Figure 2, and δ is the Dirac delta function. The value of each position in Radon space is the integral of the values along the associated line in the spatial domain; i.e., it is a line integral. Equation (1) and Figure 2 show the Radon transform in its parallel-beam form. After some initial processing of input data, CT scanners produce discrete values of an object's Radon transform. The CT reconstruction problem, then, is to find the inverse of the Radon transform; that is, given many measurements of $g()$ for many projected lines around the object, find an estimate of the original cross-section function $f()$ in this slice.

Before we solve the inverse problem, let us first see how to discretize the Radon transform and implement it on computer. The approximate discrete Radon transform is

$$g(s_i, \theta_j) \approx \Delta l \sum_{m=1}^M f(x(l_m), y(l_m)) \quad (2)$$

where l is the line defined by the argument to the delta function in (1) and Δl is the sampling increment along that line. Interpolation of some kind will be necessary because the line will not generally cross pixels evenly. The (x,y) pixel location for the summation is parameterized by the line function. Notice, however, that the basic operation is very simple—summing points along a line—while the large numbers of points and the computation of each individual point location can be the time-consuming aspect of this formula.

Figure 2. Parallel-beam CT geometry that is represented by the 2D Radon transform.

3.2 2D Parallel-beam Reconstruction

By far the most common methods for CT image reconstruction, whether for parallel-beam, fan-beam, or cone-beam applications, are based on a filtered backprojection (FBP) algorithm.² This technique is attractive because it is based on an analytic formulation, it is noniterative, it provides some control over noise reduction, and it has a similar implementation to the forward problem. The analytic FBP inversion formula can be written in two parts as follows:

$$p(s, \theta) = g(s, \theta) * h(s) \quad [\text{Filter}] \quad (3)$$

$$f(x, y) = \int_0^\pi p(x \cos \theta + y \sin \theta, \theta) d\theta \quad [\text{Backprojection}] \quad (4)$$

where $*$ is the convolution operator. The filter, $h(s)$, is a one-dimensional fixed filter that can be rapidly implemented using fast Fourier transforms (FFT's) and can have various pass-band characteristics. We will not study filter implementations in this paper. Instead, we will focus on the backprojection operation, which is the most demanding part of the algorithm.

Like the Radon transform implementation, a discrete backprojection operator can be written as

$$f(x_i, y_j) = \frac{\pi}{N} \sum_{n=1}^N p(s_n, \theta_n) \quad (5)$$

where N is the number of projections and we are assuming uniformly-spaced projections over 180 degrees. The s_n term is an interpolated version of $(x_i \cos \theta_n + y_j \sin \theta_n)$. This is similar to the Radon transform formula in that it is a simple summation, and interpolation is a key component of this algorithm also.

3.3 Orthographic Volume Rendering

The discrete form of the Radon transform (2) can be generalized to handle all single scattering volume rendering techniques.^{6,7} This is done by multiplying the summand by a weighting term, c_m ,

$$v(s_i, \theta_j) = \Delta l \sum_{m=1}^M c_m f(x(l_m), y(l_m)). \quad (6)$$

Here s represents a pixel in a scan line and θ the viewing position about the volume, f , while v is the volume rendered image for a fixed z . Computing this for each scan line produces the entire rendered image. The variations between different volume rendering algorithms focuses on determining the weighting values and choosing an efficient mechanism for computing (6). A simple weighting scheme is to make c_m be proportional to l_m or l_m^2 . This technique is widely used in computer graphics and is more commonly known as depth queueing. It has been used for volume rendering. However, most volume renderers use a more complicated data-dependent weighting scheme, the most common of which is defined by the over operator described in [10] and used in volume rendering by [11]. In this case, c_m depends on previous c 's, so most volume rendering algorithms perform this calculation iteratively front to back, or vice versa, computing c_m implicitly as the value of v get summed. By looking at volume rendering as a generalized Radon transform, we keep the mathematical form of both the forward and inverse Radon transform the same, implying that the same basic hardware primitives can be used to solve either problem with equal ease.

3. DIVERGENT-BEAM FORMULAS

Before describing how one might create the above algorithms it is important to describe three other types of formulas. The first is 2D fan-beam reconstruction. The second is the 3D cone-beam reconstruction, which is a generalization of the fan beam. This is particularly interesting because it is isomorphic to the third formula—the perspective form of volume rendering.

4.1 2D Fan-beam Reconstruction

The 2D fan-beam transform, we will call $g_f(s, \beta)$, is only different from the parallel-beam case in the direction of the x-ray paths. Indeed, there is a direct geometrical mapping between the parallel and fan-beam formulas; i.e., each line-integral of a fan beam corresponds to an equivalent line-integral in the parallel beam at some angle. A filtered backprojection algorithm can then be derived using the appropriate geometry and Jacobian. It generally looks like the following:

$$p_f(s, \beta) = [w(s) g_f(s, \beta)] * h(s) \quad [\text{Filter}] \quad (7)$$

$$f(x, y) = \int_0^{2\pi} \left(\frac{1}{r^2} \right) p_f(t, \beta) d\beta \quad [\text{Backprojection}] \quad (8)$$

where t is a geometric parameter defining the line, and r is the distance to the source along the central ray for point (x, y) in the object. The r weighting is an inverse projective distance-squared normalization term and will be explained later. The discretized fan-beam reconstruction formula is analogous then to the parallel-beam case, with appropriate weightings and geometry.

4.2 X-ray Transform

The more interesting problem for both volume rendering and reconstruction occurs if the sensor or detector array is a two-dimensional plane as in Figure 1. For volume rendering, this is a more natural approach since the output of any volume renderer is necessarily a two-dimensional array of pixels. For reconstruction, it means that data can be gathered at a much higher rate and geometric magnification can be used to obtain higher spatial resolution.

The cone-beam forward equation is often called the x-ray transform¹² and is written as follows:

$$g_c(s, t, \beta) = \int_S^{A(s, t)} f(x(l), y(l), z(l)) dl \quad (9)$$

where S is the source point, $A(s, t)$ is a point in the detector plane (see Figure 1), and l is the incremental distance along the line from S to A . Here again we are dealing with line-integrals through the volume and assuming circular source trajectories. Note that the x-ray transform is *not* a 3D Radon transform, which involves planar integrals and is not discussed here.

4.3 Cone-beam Reconstruction

The last ten years has seen a wealth of research on the cone-beam problem and reconstruction methods. For a single circular trajectory, as we have stated the problem above, there is insufficient information to correctly reconstruct the entire volume (except for the central $z=0$ -plane, which is the only plane completely characterized). Therefore, several methods have been developed that produce approximate results, such as Feldkamp's⁴ and Grangeat's⁵. Both of these are analytic methods that have backprojection elements. We will develop our algorithms for Feldkamp's method because it is the simpler of the two and it matches more closely the rendering problem. However, because Grangeat's is more accurate for larger cone angles, it will be of interest to implement it in the future.

The Feldkamp cone-beam reconstruction can be thought of as an extension of fan-beam filtered backprojection into 3D:

$$p_c(s, t, \beta) = [w(s, t) g_c(s, t, \beta)] * h(s, t) \quad [\text{Filter}] \quad (10)$$

$$f(x, y, z) = \int_0^{2\pi} \left(\frac{1}{r^2} \right) p_c(s, t, \beta) d\beta. \quad [\text{Backprojection}] \quad (11)$$

As with the fan beam, the projections are first weighted and filtered; this time with 2D weightings and filters. After convolution, the filtered projection data are used to calculate the three-dimensional volume data in Cartesian coordinates by another weighted backprojection. The weighting factor is determined by inspecting the relationship of the three-dimensional voxel grid (x, y, z) to the coordinate system of an individual projection. This coordinate system is composed of the s and t indexes of the normalized projection data set and the perpendicular coordinate, r , which represents the direction from the source

to the detector array. These three form a projective coordinate system in that rays emanating from the source to the detector are considered to be at a constant (s, t) position.

Finally, the discrete backprojection formula can be written as follows:

$$f(x_i, y_j, z_k) \approx \frac{2\pi}{N} \sum_{n=1}^N p_c(s_n, t_n, \beta_n). \quad (12)$$

4.4 Perspective Volume Rendering

The x-ray transform can be further generalized to the three-dimensional perspective rendering problem. Like the cone-beam problem, one has to take into account both the perspective weighting due to the off center traversal of $[x(l), y(l), z(l)]$ and the $1/r^2$ volumetric differential term. With this in mind equation (9) can be rewritten as:

$$v_c(s, t, \beta) = \frac{d}{\sqrt{s^2 + t^2 + d^2}} \int_s^{A(s, t)} \left(\frac{1}{r^2} \right) f(x(l), y(l), z(l)) dl \quad (13)$$

where d is the center to eye-point distance, $c(l)$ is the continuous form of the weighting term introduced in Section 3.3, and $1/r^2$ is the same projective weighting as described above. The v_c function is the image plane for a projective (s, t, r) view from any arbitrary direction. The perspective weighting term is necessary because the sensor/image plane resides in a Cartesian coordinate system. Since the projective weighting term is constant with respect to s and t , it can be performed as a post multiply on the projection v_c and is given by:

$$v_c(s, t, \beta) \approx \frac{d}{\sqrt{s^2 + t^2 + d^2}} \Delta l \sum_{m=1}^M \left(\frac{1}{r^2} \right) c_m f(x_m, y_m, z_m) \quad (14)$$

Notice how the mathematical form of equations (12) and (14) are the same—that of a weighted sum. In particular, both sums are being weighted by a projection term. This similarity means that both techniques are ideally suited for implementation on high performance true three-dimensional texture rendering hardware. We will now show that the difficulty of this problem merits hardware acceleration.

4. COMPLEXITY

For a volume of N-cubed voxels, the order of complexity of both the Radon transform and the filtered backprojection algorithms with linear interpolation is $O(N^4)$. This is generally true for parallel-, fan-, or cone-beam systems, except for multiplicative factors that increase with problem complexity. Advances in detector technology and digital acquisition have been such that the speed of measurement has always been greater than the capability of general-purpose computers to process the data. The computational complexity of the discrete filtered backprojection is sufficiently large that most CT scanners include specialized hardware to perform this task. Two trends are rapidly changing medical and industrial CT imaging. One is that increasing pressure to reduce costs is driving a transition from special purpose hardware to general purpose "off the shelf" hardware. The second is that technological advances are increasing the quantity and quality of the data that can be collected for use in diagnosis.

These trends call for a computer system architecture that addresses data visualization and reconstruction in a general purpose way. This is not, however, the same as saying that the problem must be solved by a Von Neuman type processor of sufficient SPEC-mark rating. Although that may someday be possible it will be quite far in the future. Furthermore, it will always be true that if sufficient volume is available to drive economics, a dedicated processor will outperform and/or cost less than a general purpose processor which can accomplish the same function.

A typical modern industrial CT scanner has a 2D detector with on the order of 512x512 detectors and is capable of taking 30 projections (radiographs) per second. The goal is to have the computational speed match the data acquisition speed, which for this system is ~200 Mpixels/sec. Consider the backprojection operation only; i.e., assume that the projections are already

efficiently convolved with an FFT-based procedure. The usual reconstructed volume has the same number of voxels as detectors in each dimension, which is 512 in our case. To create a 512-cubed volume, the contribution to each voxel of each convolved projection must be calculated and summed. It is this operation that is the most costly in the reconstruction process. For each projection, each voxel's projection onto the detector data set must be calculated. The data must then be resampled (usually using linear interpolation for 1D or bilinear interpolation for 2D) and summed to accumulate the total for that voxel. For the problem above, these operations will need to be performed 200 million times per second. Each operation requires several machine-level operations as well as memory fetches out of a 128 Mvoxel memory.

Present day CPUs are nowhere near being able to attain these performance requirements. However, a general purpose imaging and graphics architecture, which is available today, can implement these algorithms in the one to two second range. This type of architecture is truly general purpose, because it can be used in a wide variety of situations permitting economic benefit of the economy of scale required to amortize continuing evolutionary development and practical manufacturability. The same architecture can be economically used for visual simulation applications as well as photogrammetry, reconnaissance imaging, and advanced design. The Reality Engine² architecture is such an architecture. It comprises a large source image memory connected to the frame buffer through a high performance resampling engine. All of this hardware is under the direct control of a geometry engine subsystem providing floating point capacity of 1.2 GFLOPS.

5. IMPLEMENTATIONS

Due to the mathematical simplicity and inherent parallelism of the parallel-beam reconstruction algorithms, fast hardware solutions to these problems have been proposed and implemented.¹³⁻¹⁶ They can be based on systolic architectures with fast interconnects. Divergent-beam methods are more complex, both in computational and communication requirements, so the implementation must be done on more advanced hardware. In this section, we will describe such hardware, first by discussing texture-mapping hardware for fast perspective rendering, followed by the use of the same hardware for cone-beam reconstruction. The parallel-beam and orthographic methods are simplifications of these.

6.1 Texture-map based Rendering

Texture mapping is the process of taking an image (the texture), warping it into the shape of a surface with known weights, mapping it onto that surface, and blending it with other associated functions (e.g., lighting). It is used extensively and effectively for graphical magic in the entertainment industry. However, these operations clearly have application for other mathematical formulations also. Volume rendering is one such application.

Most volume rendering algorithms are in one of two types: either backwards projecting or forwards projecting. The backwards projecting, or ray tracing, approach solves equation (14) by iterating over each s and t and summing along the ray. A forwards projecting technique loops over all values of f , in back to front order, finding the appropriate s and t into which to sum. Our approach is a hybrid of these two and operates as follows:

For each voxel (x_m, y_m, z_m) starting at the image plane and going some fixed distance L ,
 Intersect a slicing plane at l_m parallel to v_c and trilinearly interpolate f
 Blend (weighted sum) the texture-mapped slice into the framebuffer, v_c
 Attenuate v_c by $\frac{d}{\sqrt{s^2+t^2+d^2}}$

It is the blending operation that is optimized so effectively by fast texture-mapping hardware. The final attenuation is necessary, as was mentioned above, to handle the path length difference for off-center pixels.

6.2 Texture-map based Reconstruction

Here again we focus on the backprojection component of reconstruction. Initially consider the parallel-beam reconstruction of Section 3.2. The discrete form of backprojection in (5) is accomplished by smearing p back across object space (x, y) , at an angle θ . One way to implement this is directly from (5):

```

p ← Filtered projections
For all pixels (xi,yj)
    f(xi,yj) ← 0
    For all angles θn
        Find interpolated p
        f(xi,yj) ← f(xi,yj) + p(sn,θn)
    f(xi,yj) ← f(xi,yj) * (2π/N)

```

Each pixel is independently calculated from the contribution of all projections. By viewing the heart of backprojection as a resampling problem, one can see how to recast this process into texture mapping. If we treat all p functions at each slice z as a single two-dimensional texture and reverse the order of the loops in the above algorithm, the backprojection algorithm becomes a texture mapping around a circle rotated by θ_n followed by an accumulation.

```

p ← Filtered projection
For all angles θn
    Rotate the entire projection by θn
    Render a texture-mapped circle with the texture p(sn,θn)
    Accumulate into the frame buffer
Scale and return the Accumulation

```

The fan beam case is implemented using a similar technique. The major differences are in the weighting and filtering, the $1/r^2$ attenuation term, and the divergent-beam geometry itself. It turns out these differences will not adversely affect the performance the texture-mapping implementation. This is because we can use the z-depth cueing to handle the $1/r^2$ behavior at no extra rendering time cost, and because the triangle shape of the fan is no harder to render than the circle used for the parallel case. Given these changes the above algorithm becomes:

```

p ← Filtered projection
For all angles βn
    Rotate the entire projection by βn
    Setup the depth queueing for the divergent beam
    Render a texture-mapped fan with the texture p(sn,βn)
    Accumulate into the frame buffer
Scale and return the Accumulation

```

The cone-beam algorithm will have a similar form. In all cases above, both texture mapping and frame-buffer accumulation is hardware accelerated on existing state-of-the-art graphics systems. These new algorithms are significantly faster than previous algorithms as we shall see in the next section. Indeed, they rival specialized hardware designed explicitly to perform these tasks.

6. RESULTS

Initial performance studies were done on 2D image reconstructions only. A sample problem was developed, and the images reconstructed on different computing platforms. All texture-mapped timing measurements were performed on a four Raster Manager (RM) Reality Engine² Onyx with a 150 MHz R4400 CPU made by Silicon Graphics, Inc. (SGI). Both the reconstruction and volume rendering algorithms show a linear speed improvement with the number of RM's, so removing half the RM's doubles the processing time. The CPU-based algorithm, for comparison purposes, was performed on an SGI Indigo2.

Figure 3 shows the Radon transform domain, or parallel sinogram, of the commonly used Shepp-Logan phantom¹⁷. The angle of projection is along the vertical axis. There are 512 samples in each of 804 rows (angles) in this sinogram, representing 180 degrees of coverage. The resultant 512x512 reconstruction by filtered backprojection is shown in Figure 4. This data is first filtered by row, then backprojected. Only parallel-beam backprojection has been implemented and tested on the RM's at the current time. The filtering part of the algorithm is currently done on the main CPU, but future implementations will have an RM-base filter procedure. Backprojection was implemented in 12-bits to take advantage of the frame memory configuration in the RM, but this causes no significant error in the reconstruction.

This algorithm has shown roughly 600 times speed improvement over a CPU-based floating-point backprojection algorithm for the same image. The time to reconstruct on a workstation in floating point is 20+ minutes, while the RM implementation performs the same task in 2.1 seconds. An interesting side benefit of this technique is that the reconstructed 2D image is generated in the frame-buffer memory so that the intermediate results are visible on the screen. The image is formed, projection by projection, directly on the display. The final image is visible at the end of the computation without an additional display operation.

For 3D volume rendering, we have implemented both the perspective and orthographic volume rendering algorithms. These algorithms can render a 512x512x64 8-bit volume into a 512x512 window in 0.1 seconds. This time includes performing the trilinear interpolation implied by the arbitrary slicing of the volume by the sampling plane. It also includes on-the-fly density to RGB α color look-up transfer functions. Since the planar sampling is arbitrary and oblique with respect to the overall volume, it is trivial to provide for arbitrary half-plane slicing as seen in Figure 5. When coupled with arbitrary hardware clipping planes, as provided by the graphics hardware, it is possible to do both volume rendering and slicing at the same time with no performance degradation. The Reality Engine² supports up to six concurrent arbitrary clipping planes (in addition to the six viewing frustum clipping planes).

The next steps in our development will involve the fan-beam and the full three-dimensional reconstruction problems, and the incorporation of iterative procedures. With our current experience and preliminary testing, the fan beam case shows similar speed enhancements. Since Feldkamp's cone-beam algorithm is merely an extension of the 2D case, we expect to see the same level of performance increase over uniprocessor implementations. Other more accurate cone-beam methods such as Grangeat's, will also be greatly enhanced.

7. DISCUSSION

Modern texture-mapping hardware has capabilities far beyond the graphics and rendering functions for which it was originally intended. The image reconstruction problem in CT is one that maps directly onto the hardware with a dramatic speed improvement—around three orders of magnitude over a current (uniprocessor) workstation implementations. While hardware backprojectors are usually designed for this purpose, we have shown that a well-designed graphics processor with texture-mapping hardware can produce similar speed increases without the difficulties of a hardwired or special-purpose design. The fact that the same processor is then used for 3D volume rendering of the reconstructed volume makes this solution especially attractive.

There are many other volume-processing algorithms that could benefit from the architecture that was developed for texture mapping, and we intend to explore some of these in the future. These include backpropagation for diffraction tomography¹⁸ and domain iterative reconstruction for limited-data tomography³. A domain iterative approach iterates over a cycle of the forward and inverse Radon transform, applying constraints in each domain, to converge on an optimal solution. It is not commonly used because of the large computational demands caused by such techniques. However, since both the forward and inverse steps can be accelerated with a single piece of hardware, such domain-iterative solutions now become practical.

8. ACKNOWLEDGMENTS

Many thanks to Ms. Jane DeAnda for her help in preparing this manuscript.

9. REFERENCES

1. H. E. Martz, S. G. Azevedo, D. J. Schneberk, G. P. Roberson. Computed tomography, LLNL Report UCRL-ID 114541, reprinted from *Engineering Research, Development and Technology. Thrust Area Report FY92*, UCRL 53868-92, March, 1993.
2. G. T. Herman. *Image Reconstruction from Projections: The Fundamentals of Computerized Tomography*. Academic Press, 1980.
3. S. G. Azevedo. Model-based Computed Tomography for Nondestructive Evaluation, Ph.D. thesis, U. C. Davis, 1991.

4. L. A. Feldkamp, L. C. Davis, and J. W. Kress. Practical cone-beam algorithm. *Journal of the Optical Society of America*, 1:612-619, June 1984.
5. P. Grangeat. Mathematical framework of cone beam 3D reconstruction via the first derivative of the Radon transform. In G. T. Herman, A. K. Louis, and F. Natterer, editors, *Mathematical Methods in Tomography*, Lecture Notes in Mathematics 1497, p66-97, Springer Verlag, Berlin, 1991.
6. J. Kajiya and B. Von Herzen, Ray Tracing Volume Densities. *Computer Graphics*. 18(3):165-174, July 1984.
7. J. Kajiya, The Rendering Equation, *Computer Graphics*, 20(4):143-150, Aug. 1986.
8. P. A. Sabella, Rendering Algorithm for Visualizing 3D Scalar Fields. *Computer Graphics*. 22(4): 51-58, Aug. 1988.
9. A. C. Kak and M. Slaney. *Principles of Computerized Tomographic Imaging*. IEEE Press, New York, 1988.
10. T. Porter and T. Duff. Compositing Digital Images. *Computer Graphics*. 18(3):253-259, July 1984.
11. B. Drebin, L. Carpenter, and P. Hanrahan. Volume Rendering. *Computer Graphics*. 22(4):65-74, Aug. 1988.
12. D. C. Solmon, The X-ray Transform. *J. Math. Anal. & Appl.*, 56:61-83, 1976.
13. J. L. C. Sanz, E. B. Hinkle, and A. K. Jain. *Radon and Projection Transform-Based Computer Vision*. Springer-Verlag, Berlin, 1988.
14. A. J. De Groot, S. G. Azevedo, D. J. Schneberk, E. M. Johansson, and S. R. Parker. A systolic array for efficient execution of the Radon and inverse Radon transforms. *Proc. SPIE, Advanced Algorithms and Architectures for Signal Processing III*, 975, August 1988.
15. S. G. Azevedo, J. M. Brase, H. E. Martz, A. K. Jain, K. W. Current, and P. J. Hurst. A Radon Transform Computer for Multidimensional Signal Processing. In *Proceedings of ICASSP, Glasgow, Scotland*, p1457-1459, IEEE, May 1989.
16. I. Agi, P. Hurst, W. Current, E. Shieh, S. Azevedo, and G. Ford. A VLSI architecture for high-speed image reconstruction: consideration for a fixed-point architecture. In *SPIE Electronic Imaging*, 1990. Presented at SPIE Symposium on Electronic Imaging, Feb. 1990, Santa Clara, CA.
17. L. A. Shepp and B. F. Logan. The Fourier reconstruction of a head section. *IEEE Trans. Nuclear Science*, NS-21:21-43, 1974.
18. A. J. Devaney. A filtered backpropagation algorithm for diffraction tomography. *Ultrasonic Imaging*, 4:336-350, 1982.

Figure 3. Parallel sinogram, or Radon transform, of the Shepp-Logan phantom. There are 512 line-integral samples in the horizontal direction for each of 804 projection angles in the vertical direction (180 degrees of rotation).

Figure 4. Reconstruction of the Shepp-Logan phantom onto a 512 by 512 grid using texture-mapping hardware.

Figure 5. Volume-rendered CT data with half-plane slicing at one location.

DATE
FILMED

8 / 26 / 94

END