Report Number: WUCS-88-01

1988-01-01

Evaluation of 3D Voxel Rendering Algorithms for Real-Time Interaction on a SIMD Graphics Processor

Authors: Don Schreiter and John B. Zimmerman

The display of three-dimensional medical data is becoming more common, but current hardware and image rendering algorithms do not generally allow real-time interaction with the image by the user. Real-time interactions, such as image rotation, utilize the motion processing capabilities of the human visual system, allowing a better understanding of the structures being imaged. Recent advances in general purpose graphics display equipment could make real-time interaction feasible in clinical setting. We have evaluated the capabilities of one type of advanced display architecture, the PIXAR Imaging Computer, for real-time interaction while displaying three-dimensional medical data as two-dimensional projections.
It was discovered during this investigation that most suitable algorithms for implementation were based on the rendering of voxel rather than surface data. Two voxel-based techniques, back-to-front and front-to-back rendering produced acceptable, but not real-time performance. The quality of the images produced was not high, but allowed the determination of an image orientation which could then be used by a later high-quality rendering technique.
Two conclusions were reached: first, the current performance of display hardware may allow... **Read complete abstract on page 2.**

Evaluation of 3D Voxel Rendering Algorithms for Real-Time Interaction on a SIMD Graphics Processor

Complete Abstract:

The display of three-dimensional medical data is becoming more common, but current hardware and image rendering algorithms do not generally allow real-time interaction with the image by the user. Real-time interactions, such as image rotation, utilize the motion processing capabilities of the human visual system, allowing a better understanding of the structures being imaged. Recent advances in general purpose graphics display equipment could make real-time interaction feasible in clinical setting. We have evaluated the capabilities of one type of advanced display architecture, the PIXAR Imaging Computer, for real-time interaction while displaying three-dimensional medical data as two-dimensional projections.

It was discovered during this investigation that most suitable algorithms for implementation were based on the rendering of voxel rather than surface data. Two voxel-based techniques, back-to-front and front-to-back rendering produced acceptable, but not real-time performance. The quality of the images produced was not high, but allowed the determination of an image orientation which could then be used by a later high-quality rendering technique.

Two conclusions were reached: first, the current performance of display hardware may allow acceptable interactive performance and produce high-quality images if a scheme of adaptive refinement is used wherein successively higher quality images are generated for the user. Second, the correct algorithm to use for fast rendering of volume data is highly dependent upon the architecture of the display processor, and in particular upon the ability of the processor to randomly access image data. If the processor is constrained to sequential or near sequential access to the voxel data, the choice of algorithms and the utilization of parallel processing is severely limited.

EVALUATION OF 3D VOXEL RENDERING ALGORITHMS FOR REAL-TIME INTERACTION ON A SIMD GRAPHICS PROCESSOR

Don Schreiter and John B. Zimmerman

WUCS-88-01

January 1988

Department of Computer Science
Washington University
Campus Box 1045
One Brookings Drive
Saint Louis, MO 63130-4899

To appear in SPIE Proceedings of Medical Imaging II, Vol. 914(II), January 31-February 5, 1988, Newport Beach, California.

Evaluation of 3D Voxel Rendering Algorithms for Real-Time Interaction on an SIMD Graphics Processor

Don Schreiter
John B. Zimmerman

Department of Computer Science and
Mallinckrodt Institute of Radiology
Washington University
St. Louis, Missouri 63130

ABSTRACT

The display of three-dimensional medical data is becoming more common, but current display hardware and image rendering algorithms do not generally allow real-time interaction with the image by the user. Real-time interactions, such as image rotation, utilize the motion processing capabilities of the human visual system, allowing a better understanding of the structures being imaged. Recent advances in general purpose graphics display equipment could make real-time interaction feasible in a clinical setting. We have evaluated the capabilities of one type of advanced display architecture, the PIXAR[1] Image Computer, for real-time interaction while displaying three-dimensional medical data as two-dimensional projections.

It was discovered during this investigation that the most suitable algorithms for implementation were based on the rendering of voxel rather than surface data. Two voxel-based techniques, back-to-front and front-to-back rendering produced acceptable, but not real-time, performance. The quality of the images produced was not high, but allowed the determination of an image orientation which could then be used by a later, high-quality rendering technique.

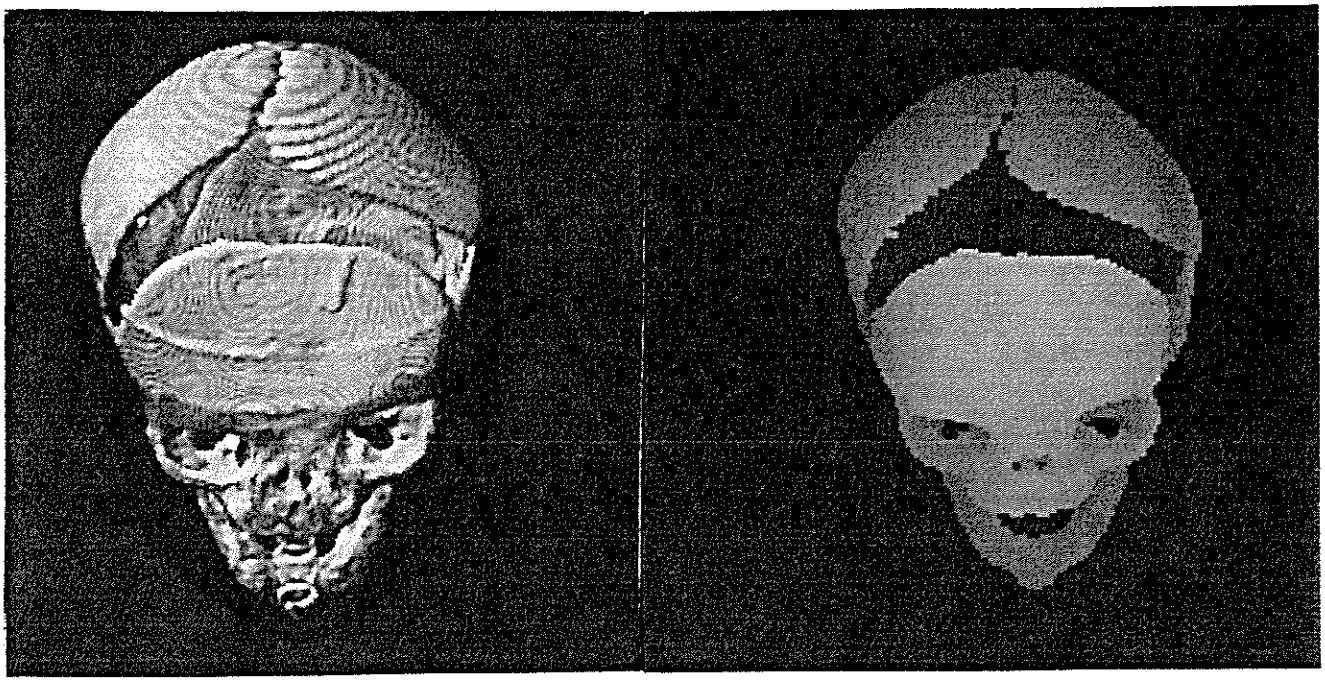
Two conclusions were reached: first, the current performance of display hardware may allow acceptable interactive performance and produce high-quality images if a scheme of adaptive refinement is used wherein succesively higher quality images are generated for the user. Second, the correct algorithm to use for fast rendering of volume data is highly dependent upon the architecture of the display processor, and in particular upon the ability of the processor to randomly access image data. If the processor is constrained to sequential or near-sequential access to the voxel data, the choice of algorithms and the utilization of parallel processing is severely limited.

1 INTRODUCTION

As medical imaging devices have become more powerful, it has become possible to acquire detailed three-dimensional (3D) information about the structure of the body. Such information is difficult to interpret as two-dimensional (2D) slices; thus, considerable interest has been aroused in techniques for rendering this data as 2D projections of surfaces or volumes which will allow for easier interpretation. A number of commercial systems have become available for doing such renderings. These systems typically allow the user to define tissue types or structure surfaces within the data, remove obscuring structures, and select a viewing position or positions relative to the image. High-quality 2D images are then generated by the system. Such images have proven useful in some cases for the diagnosis of difficult osteopathic pathology [1]. However, the amount of computation required for the production of high-quality images which will allow effective diagnosis is large, particularly if a cinè loop of the data from different viewpoints is desired [2]. Such computations typically must proceed off-line and if an incorrect view is specified, the images must be recomputed.

Recently, new display architectures have appeared which use very fast processors arranged in parallel to permit rapid computation of such images. The PIXAR Image Computer is an example of such a machine; it uses a set of four processors arranged in a single-instruction, multiple-data (SIMD) configuration and is capable

[1]PIXAR is a trademark of PIXAR Incorporated



(a) (b)

Figure 1: Renderings of Voxel images. Depicted spatial resolution is 224 × 224. (a) ChapVolumes (b) fast front-to-back (FTB).

of a peak speed of 40 million instructions per second (MIPS). The PIXAR-developed volumetric rendering software, ChapVolumes [3], produces high quality images which retain the information content of the original slice data (Figure 1a); however, the production of each individual image (a few minutes per image) is too slow to be used effectively in an interactive manner.

The purpose of the current work is to investigate algorithms for rendering 3D volume data as swiftly as possible with the intent of allowing the user real-time or near real-time selection of the viewing position. We are willing to sacrifice some image quality for speed (Figure 1b), because our implementation is meant to be a front end for rendering systems producing higher quality images, such as ChapVolumes. We envision physicians setting up the views, cuts, and motions they desire using our interactive system and then producing final images in batch mode using higher quality rendering systems. The PIXAR Image Computer was used as a testbed for these studies.

An extensive study of volumetric rendering algorithms has been conducted by Reynolds [4]; this work provided a starting point for our evaluation. A number of these algorithms were implemented on the PIXAR Image Computer and a Sun Microsystems workstation. Our eventual goal is to apply adaptive refinement rendering ideas, similar to those presented by Bergman *et.al.* [5] to the problem of volumetric rendering. The first step was to pick the proper high speed interactive algorithm for the first stage of the adaptive refinement system which compliments our PIXAR hardware and software.

2 PIXAR SIMD ARCHITECTURE

The PIXAR Image Computer consists of two parts which are important for this discussion [6]: a four processor SIMD computing unit, called the Chap, and a large (16 to 64 million 12-bit words) frame buffer memory. The memory is organized in a hierarchical fashion with slower speeds for the memory systems with more capacity.

The Chap provides the opportunity for pipelined operations within each of the four processing units. All of these characteristics affect the execution efficiency of the PIXAR image computer for our volume rendering algorithms. We have not experimented extensively with the pipelining capabilities of the processors, so we will not report further on pipeline effects.

The SIMD architecture of the Chap was originally designed so that each processor operates on a single component of a four-component, 48-bit, red, green, blue, and transparency (RGBA) pixel. This arrangement is very efficient for calculating full-color graphic images; however, most volumetric medical data consists of single-component (grey-scale) pixels. The Chap can also be used to access a single component (*e.g.*, red) of four different pixels, but this locality of reference in both operating modes makes the exploitation of parallelism difficult. However, some of the algorithms we have examined can be broken into a number of sufficiently similar parts to use the SIMD architecture effectively.

The PIXAR Image Computer has a three-tier hierarchical memory organization. The lowest level is the frame buffer memory which is organized as RGBA pixels. This memory is addressed through a programmable memory controller and is most efficiently accessed in large blocks because of the memory controller overhead and the use of memory interleaving. Memory words can be fetched efficiently either as RGBA pixels or as four like components (*e.g.*, four red components) of adjacent pixels. The sustained rate of access is about 7.5 megapixels (MP)/second.

Each processor in the Chap contains 16 kilowords (KW) of scratch pad memory which can contain pixel data. This memory forms the middle tier of the hierarchy and can be accessed randomly and quickly in pixel or component form. The access time is usually 4 clock cycles (about 340 nsec).

The top of the memory hierarchy consists of 30 data registers, 14 memory pointer registers and 14 index registers. Each processor has its own data registers, but the pointer and index registers are shared by all processors. Data can be accessed from the registers in 1 clock cycle. A central control unit contains 16 thousand 96-bit words of instruction memory and directs the sequencing of the four processors.

Efficient algorithms for the PIXAR require memory accesses, especially at the frame buffer level, to be large contiguous blocks of pixels or components. The various volume rendering algorithms considered have different memory access requirements and this affects their performance on the PIXAR.

3 COMPARISON OF RENDERING ALGORITHMS

There are two basic methods for rendering volumetric data: methods which fit surfaces to the data and then render these surface models using standard computer graphics techniques and methods which render images directly from the volumetric data using volumetric visualization techniques.

Algorithms have been developed for interactive surface rendering from volume data [4, pp. 72–112], [7], but these have major drawbacks when used for fast rendering. Producing the surface models requires extensive preprocessing which, for good surface detection, often involves human intervention. In this process, much of the information contained in the original volume data is lost. If a change is made in tissue classification or cut plane, the surfaces must be reconstructed from the original volume data. These problems caused us to reject surface rendering for our purposes.

Rendering 3D images directly from volumetric data using methods such as those developed by PIXAR eliminates the extensive preprocessing time needed to build surface models and preserves all the information contained in the data. The PIXAR volumetric rendering method considers each slice of the volume to be a transparent colored filter. These filters are rotated and projected into a 2D image on the screen. The density, color, and refractivity of each voxel in a slice is controlled by user-defined tissue classification tables based on the grey-scale voxel intensities. Care is taken to introduce as few artifacts as possible and to preserve the original data by careful filtering and interpolation during the rendering.

The major advantages of this method are its high-quality images and preservation of volume information. Figure 1a shows a volumetric rendering of an infant's skull taken from CT data. However each image takes a relatively long time to produce, usually several minutes, so the technique is not suitable for interactive use. This method also needs large amounts of memory for the volume slices. Building classification tables is made tedious

by the long waits to check the resulting image after modifying the tables. A method is needed which combines minimal preprocessing with a fast rendering capability and which is compatible with high quality volumetric techniques and the PIXAR hardware architecture.

3.1 Fast Rendering Algorithms

3D Primitive Methods. We considered three algorithms which use 3D primitives (cubes) as their basic objects for manipulation, as described by Reynolds [4, pp. 113–146], Tuy and Tuy [8], and Frieder *et.al.* [9]. These methods retain the grey-scale slice data, so changing the classification tables is simple and display shading based on voxel value as well as distance is possible. They also require no preprocessing to change cut planes or classification levels, so these attributes can be changed interactively.

On the negative side, because these 3D algorithms retain the grey-scale data, they require large amounts of slice storage in the frame buffer memory. There is also a limitation on the scale factors allowed during viewing due to the direct object space to image space projections used in most of these algorithms. Finally, the 3D methods are slower than the 1D primitive methods mentioned below; however, because some of the 3D methods are more amenable to parallel execution than the 1D techniques, this may not be a serious difficulty.

The first two 3D primitive methods considered were ray tracing and recursive back-to-front rendering. Ray tracing has the advantage of no scale limits because samples are taken in all image pixels. Recursive back-to-front rendering is particularly simple and easily implemented in hardware [4, pp. 193–224]. We rejected both of these methods for implementation on our SIMD hardware because they require memory accesses organized by cubic volumes in object space; the PIXAR memory is most easily and efficiently accessed in linear segments. They also show a slower execution time on uniprocessor machines than the nonrecursive back-to-front algorithm described below.

We chose to implement the third algorithm considered, nonrecursive back-to-front rendering (BTF). This method depends on the fact that a correct back to front ordering of the voxels in a volume can be obtained by simply traversing slices, lines and voxels in forward or reverse order depending on the viewpoint. Since the voxels are processed in back to front order, the projections of the closer voxels will replace the more distance voxel projections as the image is rendered, giving a correct distance image. Each voxel location is transformed from object space to image space using a standard rotational transformation. Because we are dealing with discrete integer voxel locations, the transformation process can be implemented using table lookup methods which greatly decrease the time required for the voxel transformations.

The BTF method accesses memory line by line within a slice and each slice, line, and voxel is examined only once. This memory access pattern fits our hierarchical memory structure well. BTF is simple to implement and shows promise for parallel execution because each voxel transform is independent. The algorithm also showed the highest speed of all of the 3D primitive methods in a uniprocessor implementation.

1D Primitive Methods. The other algorithm that was implemented uses 1D primitives [4, pp. 147–176]. In this method, the grey-scale volume data is converted, using thresholding, to a run-length encoded binary representation of non-empty segments in each line of voxels. These segments are examined in front to back order using the starting points and increments for slices, lines, and segments that are calculated using the desired viewpoint. As voxels are projected to image space using the table lookup method, a linked-list data structure for each image line is used to keep track of the image pixels which have not been filled. This structure allows voxel segments which project to previously written image pixels to be ignored.

The front-to-back, FTB, method is fast because during examination of the data only visible voxels are fully processed and these voxels are processed as segments, not individually. These two characteristics give the 1D front-to-back method the fastest uniprocessor performance of the algorithms considered and greatly reduce the amount of data storage required for the volume data. The FTB algorithm accesses memory in a manner similar to the BTF method, so it fits the PIXAR memory arrangement well.

Unfortunately, these advantages are obtained by introducing several disadvantages. Like the 3D primitive methods, the direct projection used here limits the allowable scaling of the images. This method also requires some preprocessing to build the binary volume, which slows down the overall execution speed and makes it difficult to re-threshold quickly. No preprocessing is needed to change cut planes, however, so this can still be

done interactively. Because a binary volume is used, grey-scale based shading of the data is impossible. This limits the flexibility of the algorithm for adaptive refinement rendering. Finally, the merge step used to combine object segments with image segments makes parallel execution of the algorithm difficult, especially on an SIMD processor.

4 IMPLEMENTATION RESULTS

To compare the performance of the voxel rendering algorithms, two methods were encoded on both the Chap processor of the PIXAR Image Computer and a uniprocessor workstation (Sun Microsystems 3/180). Only integer arithmetic was used. The algorithms implemented were nonrecursive back-to-front (BTF) rendering and front-to-back (FTB) rendering. Three volume images (pelvis, skull, and spine) taken from CT images were rendered at various spatial resolutions. The original data were slices of 256 × 256 pixels; the slices were interpolated so that the voxels were cubical at the desired resolution. For comparison purposes, the same images were rendered using a beta test version of the voxel rendering routines from the ChapVolumes software supplied to us by Philips Medical Systems and Cemax Incorporated. It is anticipated that the times required for rendering will be somewhat less with the production version of this software. Tissue classifications were chosen to show the surface of the bone for the FTB and BTF algorithms; the ChapVolumes renderings also depicted the soft tissue.

The execution time of the ChapVolumes rendering method consists of two parts: preprocessing and image rendering. The preprocessing time includes reformatting the slice data into a usable form on the host (Sun) computer, loading the reformatted slices into the PIXAR frame buffer memory, and interpolating new slices between the existing slices to form cubic voxels using the Chap processor. After the slices are interpolated, the actual image formation process begins. Several images in an animated sequence can be made without repeating the preprocessing steps. This method is highly optimized for the PIXAR SIMD architecture and uses all 4 processors in most phases of execution.

The BTF algorithm uses the same preprocessing routines as ChapVolumes; thus, the input data format and the time for preprocessing are identical for BTF and ChapVolumes. During BTF image formation, voxel coordinate transforms are performed in parallel using 4 processors, one each for neighboring voxels on a line. Because of the general nature of the 3D transformation, it cannot be guaranteed that the 4 voxels being processed will be adjacent in the frame buffer memory after transformation; thus, each value must be written into the memory independently, rather than in parallel. The distance of each voxel from the viewing plane, z, is used to determine voxel visibility and to shade the pixel intensity in the 2D image. Pixels which are farther from the viewpoint have a lower intensity; this distance shading gives the illusion of three-dimensionality to the image.

The FTB algorithm requires the same preprocessing as the ChapVolumes and BTF algorithms, plus an encoding of the slice data into a binary volume. The images are encoded on the PIXAR to take advantage of its approximately fivefold increase in speed over the host. No attempt has been made to use parallelism during encoding, although this should be possible. We have not been able to use the SIMD parallelism of the PIXAR during the merge step in the FTB algorithm because during the merge step, voxel segments are not independent. All of the speed advantage of the FTB method over the BTF method on the PIXAR derives from the efficiency of the underlying algorithm. However, lines in object and image space are independent, so possible parallelism exists if different processors can execute independently; thus, a multiple-instruction, multiple-data stream (MIMD) architecture could be used to good effect with this algorithm. The output image in the FTB method is distance shaded as with the BTF rendering.

Storage Requirements. Table 1 shows the storage requirements for the unencoded skull data used by the ChapVolumes and BTF algorithms compared to the binary run-length encoded skull data used by the FTB algorithm for each of three output resolutions. The unencoded data is from 10 to 20 times as large as the coded data, depending on volume size and the details of the slice data. Table 2 shows the variation in coded volume size with data set slice complexity for each of the three example volume images. It can be seen that, as expected, the spine data requires considerably more storage than the other two. The reduction in volume size realized in the FTB encoding is important because it saves disk space, disk access time, frame buffer space, and frame buffer access time. For example, ChapVolumes and BTF can not be used for volumes larger than $224 \times 224 \times 224$ given our 4 MP of frame buffer memory, but FTB can be used for 512 × 512 × 512 volumes without difficulty.

Encoding Method	128 × 128 × 110	192 × 192 × 165	224 × 224 × 192
Unencoded (ChapVolumes and BTF)	1760	5940	9408
FTB encoded	169	353	465

Table 1: Storage required for the unencoded and encoded skull data at three different spatial resolutions. All values are in 1024 byte blocks.

Data Set	256 × 256 × 220
pelvis	497
skull	585
spine	953

Table 2: Storage required for FTB encoded data for three different volume images. Sizes are given in 1024 byte blocks. The size varies depending on the complexity of the object surfaces.

Preprocessing Times. Table 3 compares the preprocessing times for the ChapVolumes, BTF, and FTB methods using the volume image of the skull at three different output image resolutions. The times for ChapVolumes and BTF are identical because the same program is used for volume preparation. Although more processing is done to encode the FTB volume than the others, two factors make FTB encoding slightly faster for this data set. First, the encoding is done on the PIXAR at 10 MIPS, so the encoding time is small. Second, the time required to write the larger volume data to disk for ChapVolumes and BTF more than makes up for the encoding time used for FTB. The figures for FTB will vary with dataset, but by no more than about factor of two. Very little optimization of the preprocessing has been done, so large improvements in efficiency are probably possible.

Rendering Performance. Table 4 shows the rendering times of the three algorithms for the skull image. The improvement between BTF on the Sun and on the PIXAR is substantial because BTF on the PIXAR uses 4 way parallel execution and efficient access to the frame buffer memory. We estimate two-thirds of the improvement to be from parallel execution on the 40 MIPS Chap as compared with the 2 MIPS Sun and the remaining one-third of the improvement to be from efficient frame buffer access on the PIXAR relative to the Sun frame buffer access.

The improvement from FTB on the Sun to FTB on the PIXAR is much smaller than seen in the BTF implementations because 1) no use is made of the parallel capabilities of the Chap by FTB on the PIXAR and 2) the encoded data used for FTB is small enough to fit in RAM on the Sun, so the frame buffer to Sun channel is not a bottleneck for host execution. All of the improvement from the host to the PIXAR for FTB can be attributed to the 10 MIPS single channel speed of the Chap compared to the 2 MIPS speed of the Sun.

The most important point shown in Table 4 is the marked improvement in execution speed with the more efficient FTB algorithm. FTB execution on the Sun is faster than BTF on the PIXAR even though the PIXAR is at least 20 times faster than the Sun, because the FTB algorithm is much more efficient than the BTF method. As is usually the case, algorithm choice is more important than machine speed.

Finally, Table 5 shows the variation in FTB execution time with three different data sets. The time is dependent on the complexity of the coded volume. Reynolds states that the execution time is $O(BM^2)$, where M is the image size and B is $\min(C, M)$ with C the minimum number of convex parts of the object being imaged [4, pp. 67]. This compares to times of $O(M^3)$ for the other algorithms.

5 DISCUSSION AND CONCLUSIONS

The purpose of this research was to investigate the feasibility of interactive image generation on an SIMD processor. The results indicate that this goal is within reach; currently we can generate 2 to 6 images per second at a resolution adequate for determining acceptable viewpoints and cinè sequences for use in high-

Processing Method	128 × 128 × 110	192 × 192 × 165	224 × 224 × 192
ChapVolumes	83.6	114	134
BTF	83.6	114	134
FTB	82.8	113	132

Table 3: Preprocessing times for ChapVolumes, BTF, and FTB for the skull data set and three output image resolutions. All times are in seconds of program execution (operating system overhead has been removed).

Processing Method	128 × 128 × 110	192 × 192 × 165	224 × 224 × 192
BTF Sun	1035	3378	5259
ChapVolumes	856	2613	4031
BTF PIXAR	34.9	112	174
FTB Sun	29.4	65.6	90.6
FTB PIXAR	4.2	10.9	15.2

Table 4: Processing times for various PIXAR and Sun algorithms using the skull data set. Times are given in seconds for a 24 image animation sequence of a 360° rotation about the x axis.

Processor	pelvis	skull	spine
Sun	91.7	110	150
PIXAR	17.9	20.1	28.8

Table 5: FTB comparison times for three 256 × 256 × 220 volume images.

quality rendering. This speed approaches that necessary for comfortable interactive manipulation (at least 15 images/second). The sacrifice in image quality necessary to achieve this performance does not appear to be detrimental to the diagnostic performance of observers when combined with a back-end, high-quality rendering system. However, further work, including observer performance studies, is necessary to confirm this result.

It is clear that the exploitation of parallel computer architectures is a necessary step in learning to manage the large quantities of image data producable by modern imaging modalities. Unfortunately, previously developed algorithms do not alway map well onto parallel architectures. The crucial step in developing an acceptable display system is to match the algorithm with the characteristics of the hardware.

In selecting an appropriate algorithm, our work has led us to the following conclusions: if you need the greatest possible speed and you have a uniprocessor, SIMD machine with a few processors, or an MIMD machine, the FTB algorithm will give you maximum speed, but you will have only binary data for display and must re-encode to change classification.

If you want grey-scale voxel display or have an SIMD machine with many processors, then the BTF algorithm is more appropriate. This method allows interactive classification changes and shading algorithms which take into account voxel grey value as well as distance. With a sufficient number of processors, a parallel implementation of this method should be usable interactively and can be developed easily on an SIMD machine.

Future Work. Several avenues for further work present themselves from these results. We are continuing to search for techniques which will allow application of adaptive refinement rendering to volumes. At this point, we have fast algorithms for initial images and a final stage, high quality, method. These methods can be integrated into a system which continually improves the image quality as the user pauses in interaction using techniques such as image space gradient shading [4,10] and image space anti-aliasing [11]. Further research is needed to provide similar capabilities for grey-scale and color images.

As the next generation of hardware becomes available, we hope to implement some of the algorithms investigated in this work on an MIMD machine, specifically an AT&T Pixel Machine. We expect to achieve true real time performance using this hardware and either the BTF or FTB algorithms appropriately modified to fit the Pixel Machine memory architecture.

Finally, the greatest current limitation of all of these techniques is their use of simple grey-level thresholding for image classification. Thresholding is successful in differentiating tissue types such as bone, muscle, and fat, but does not allow the easy visualization of organs and soft tissues of similar density. Better classification algorithms are needed to extract this information from the volume images and depict it accurately. At that point, it will be possible to evaluate the utility of three-dimensional volume imaging for medical diagnosis.

6 ACKNOWLEDGEMENTS

The authors gratefully acknowledge an equipment grant from Philips Medical Systems, Inc. in support of this research. CT image data was supplied by Michael Vannier. We thank Carolyn Offutt and David Beecher for useful discussions and Carolyn Brown for assistance with the poster layout.

7 REFERENCES

[1] E. K. Fishman, B. Drebin, D. Magid, W. W. Scott, D. R. Ney, A. F. Brooker, L. H. Riley, J. A. Stville, E. A. Zerhouni, and S. S. Siegelman, "Volumetric Rendering Techniques - Application for 3-Dimensional Imaging of the Hip," *Radiology*, 163 (3), 737–738 (1987).

[2] M. Flynn, R. Matteson, D. Dickie, J. W. Keyes, Jr., and F. Bookstein, "Requirements for the Display and Analysis of Three-dimensional Medical Image Data," *Picture Archiving & Communication Systems for Medical Applications*, 418, 213–223, SPIE (1983).

[3] PIXAR, *ChapVolumes Volume Rendering Package Technical Summary*, PIXAR Incorporated, San Raphel, California (1987).

[4] R. A. Reynolds, *Fast Methods for 3D Display of Medical Objects*, Ph.D dissertation, University of Pennsylvania, Philadelphia, Pennsylvania (1985).

[5] L. Bergman, H. Fuchs, and E. Grant, "Image Rendering by Adaptive Refinement," *Computer Graphics*, 20(4), 29–37 (1986).

[6] PIXAR, *Hardware Overview*, PIXAR Incorporated, San Raphel, California (1986).

[7] H. Fuchs, G. D. Abram, and E. D. Grant, "Near Real-Time Shaded Display of Rigid Objects," *Computer Graphics*, 17(3), 65–72 (1983).

[8] H. K. Tuy and L. T. Tuy, "Direct 2-D Display of 3-D Objects," *IEEE Computer Graphics and Applications*, 4(10), 29–33 (1984).

[9] G. Frieder, D. Gordon, and R. A. Reynolds, "Back to Front Display of Voxel-Based Objects," *IEEE Computer Graphics and Applications*, 5(1), 52–60 (1985).

[10] R. A. Reynolds, "Image Space Shading of 3-Dimensional Objects," *Computer Vision, Graphics, and Image Processing*, 29, 361–376 (1985).

[11] J. Bloomenthal, "Edge Inference with Applications to Antialiasing," *Computer Graphics*, 17(3), 157–162 (1983).