Comput. & Graphics Vol. 17, No. 1, pp. 31–37, 1993
Printed in Great Britain.

0097-8493/93 $6.00 + .00

PARALLEL VOLUME VISUALIZATION ON WORKSTATIONS

KWAN-LIU MA and JAMES S. PAINTER
Department of Computer Science, University of Utah, Salt Lake City, UT 84112

Abstract—This paper discusses the use of general-purpose graphics workstations for interactive high-resolution volume visualization. We survey previous research results in parallel volume rendering as well as commercial products that take advantage of parallel processing to make volume rendering a practical visualization method. Our focus is on developing distributed computation methods that can distribute the memory and computational demands of volume visualization across a network of general purpose workstations. We describe three distributed computation strategies based on ray-casting volume rendering that can be implemented on either shared-memory multiprocessor workstations or on a network of ordinary workstations. Multiple views of real-time feature extraction give tremendous insight to the volume data. Multiple variable visualization helps scientists to capture the interaction between important variables in a simulation. Divide-and-conquer rendering allows interactive high-resolution volume visualization of large data sets on a network of midrange workstations, even when the data set is too large for available memory on any single workstation. Several examples in medical imaging and computational fluid dynamics are shown illustrating the practicality of these methods.

1. INTRODUCTION

The advance of parallel architecture and software carries computational science into a new dimension in which scientists have been able to more accurately approximate and explore the world around us and thus derive many new scientific discoveries. Scientific visualization, the use of computer graphics to provide visual interpretations of simulations of physical phenomena or acquired data from scanning systems, often requires intensive computation and the management of enormous amount of data. Over the past few years, highly commercial and academic interests have led to the development of many new parallel architectures and software algorithms for data visualization [1, 2, 3].

Vector supercomputers like the Cray and distributed-memory supercomputers like the Connection Machine are expensive to maintain and run, and usually must be shared between many users. Other special-purpose architectures, developed in research environments, are also not generally available to scientists who may need to conduct visualization procedures daily in their laboratory. The increase in performance and decrease in price of general-purpose graphics workstation suggest that we could make good use of them for not only numerical simulations but also data visualization.

In this paper, we describe how general-purpose graphics workstations, particularly multiprocessor and networked workstations, can be used to visualize large or multiple data sets, and to do interactive feature extraction and viewing. We show examples in medical imaging and combustion simulations making use of IRIS 4D/240 GTX or multiple IBM RS/6000 Model 520/530 workstations to make the process of data visualization much more effective and efficient. The examples shown here use IVES (Interactive Volume Exploration System) [4, 5], an visualization system based on ray-casting volume rendering that we have developed for local scientists.

2. PARALLEL VOLUME VISUALIZATION

Most scientific and biomedical data sets are scalar or vector fields of three spatial dimensions known as volume data. Direct volume rendering, creating an image directly from volume data without constructing intermediate graphics primitives, has been shown to be a very effective method for visualizing scalar volume data [6, 7, 8]. However, volume rendering is computational expensive and the rendering time grows linearly with the size of the data set. As a result, many algorithms have developed which take advantage of the coherence in the data and the rendering process [4, 9]. Parallel volume rendering schemes have also been developed, using special-purpose hardware [10, 11] or supercomputers [12, 13].

The major algorithmic strategy for parallelizing volume rendering is the divide-and-conquer paradigm. The volume rendering problem can be divided either by data space subdivision (DSS) or by image space subdivision (ISS). While DSS assigns the computation associated with particular subvolumes to processors, ISS distributes the computation associated with particular portions of the image space. DSS is usually implemented on a distributed-memory parallel computing environment. On the other hand, ISS is most efficient on a shared-memory multiprocessor computer. Hybrid methods are also feasible.

Splatting is a data-space parallel volume rendering algorithm that maps the volume data onto the image plane. A full resolution rendering of a data set with 96 $\times$ 128 $\times$ 113 grid points takes about one minute on either a Sun TACC-1 or four Sun-4s [14]. As splatting is better suited for coarse to medium grain parallelism, a multi-pass shear decomposition algorithm has been implemented on the finely distributed-memory Connection Machine [12] to approximate real-time rotation of the volume data. The rendering uses parallel computation constructs offered by the Connection Machine

which allow the use of sophisticated shading models and still maintain high speed throughput. Multiple frames per second for data sets of size $64 \times 64 \times 64$ can be achieved on 64K-processor CM-2.

An example of a hybrid method is the Pixel-Planes 5 project. Pixel-Planes 5 [11, 15] is a heterogeneous multiprocessor graphics system capable of performing ray-casting volume rendering. The hardware consists of up to 32 graphics processors (40 MHz Intel i860 microprocessors), up to 16 rendering units, and a conventional 1280×1024-pixel frame buffer, interconnected by a five gigabit ring network. The volume data set is distributed among the rendering units. Each graphics processor is assigned a subimage, performs corresponding rays sampling and requests needed voxel values from the rendering units. A sequence of high resolution images can be generated at one frame per second. Most of the above methods either use special-purpose hardware or sacrifice image quality to achieve interactive to near real-time rendering. We instead utilize multiple workstations to accomplish interactive high-resolution volume visualization.

3. IVES

Our work in parallel volume visualization is built on a locally developed interactive volume exploration system, IVES. IVES, based on an image-order front-to-back ray-traced volume rendering algorithm [9], implements a set of real-time interaction techniques that have been developed to permit exploration of a volume data set. Within the limitation of a static viewpoint, the user is able to interactively alter the position and shape of an area of interest, and modify local viewing parameters. A run length encoded cache of volume rendering samples provides the means to rerender the volume at interactive rates. The user locates and plants "seeds" in areas of interest through the use of data slicing and isosurface techniques. Image processing techniques applied to volumes (*i.e.*, volume processing), can then automatically form regions of interest that in turn modify the rendering parameters. This "region growing" of "seedlings" incrementally alters the image in real-time providing further visual cues concerning the contents of the data. These tools allow interactive exploration of internal structures in the data that may be obscured by other imaging algorithms. Currently IVES has been implemented using the IRIS Graphics Library (GL), which is also supported on the IBM RS/6000 workstations.

4. PROBLEMS WITH DATA VISUALIZATION

It is clear that for data visualization, interaction is important and motion is even more powerful. Most existing parallel volume rendering architectures and algorithms have been designed to achieve highly interactive, or if possible real-time, visualization. One problem with present general-purpose workstations is the inability in processing speed to support real-time high-resolution volume visualization activities. Image resolution can be sacrificed to speed up volume rendering, however this gives the user a poor quality result that may not have sufficient detail for interpretation. IVES solves this problem partially by providing fast rerendering of static views and powerful feature extraction facilities.

Another major problem for data visualization on workstations is the high memory demands of the volume visualization algorithms. Often it is not possible to hold the entire data set in memory at one time. For example, a typical three dimensional combustion simulation produces three components of velocity and a dozen of other scalars like temperature of some chemical concentrations. As a result, a few hundred megabytes of data is fairly typical. Moreover, a single scalar data set may take over 100 megabytes of storage in an applications like medical imaging. Further, for interactive volume rendering, we need not only store the volume data to be visualized but also values such as the surface normals, attribute maps, *etc.* Efficient algorithms, which trade space for time, require even more memory capacity to cache reusable results. Our workstations are equipped with 16–128 megabytes of main memory. Thus the use of a single workstation for interactive visualization often forces us to reduce the data set or to process at most two to three scalars at a time. Data reduction may cause serious artifacts in the resulting images and wash out many details important to the scientists. The inability to view multiple related data sets simultaneously prevents scientists from capturing important interactions between certain elements in the simulation.

There are many other important issues with data visualization that we have not addressed here, such as data formats, user interface, *etc.* Since our main focus in this paper is distributed data visualization on workstations, we waive the discussion of those other issues.

5. MULTIPLE-VIEW RENDERING

As mentioned above, real-time rotation of volume data, using direct volume visualization, on general-purpose workstations is still not generally achievable. IVES is a single-view visualization system that is very effective in capturing features in the volume data. The use of multiple workstations or a multiprocessor workstation like the IRIS 4D GTX or VGX allows the user to watch multiple views of a particular structure in the data. Figure 1 displays four different views of the vascular structure within the brain of a patient suffering from an aneurysm. This data set was acquired from Magnetic Resonance Angiography (MRA), in which the focus of attention is on exploring the vascular structure within the brain or other regions of the body. MRA techniques are used to diagnose malformations and aneurysm within the brain's blood supply, and to plan surgical and catheterization procedures. The intricate nature of the vascular structure as well as the somewhat noisy data capture require the ability to focus attention on specific vessels as potential anomalies are discovered.

All four renderings use one single large seed in the center of the volume to capture most of the vessels while eliminating the vessels at the outer edges which complicate and obscure the interior. Multiple views in this case help the physicians identify certain structures

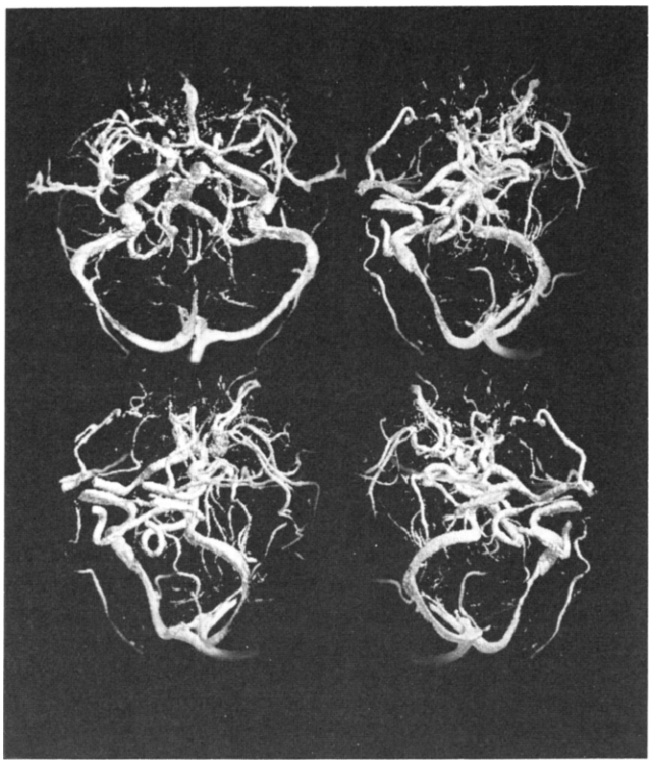

Fig. 1. Multiple views of vascular structures.

and anomalies. IVES gives rapid re-rendering by caching ray samples for a particular view, and thus allows interactive feature extraction. The real-time feature extraction available through volume seedlings[5] is even more powerful when multiple views are simultaneously available.

Volume seedling is in essence a feature extraction technique for scalar field data. Current research in progress extends IVES to handle vector field data[16], which further illustrates the effectiveness of multiple views displayed concurrently. Figure 2 shows the visualization of the data from the simulation of a full-

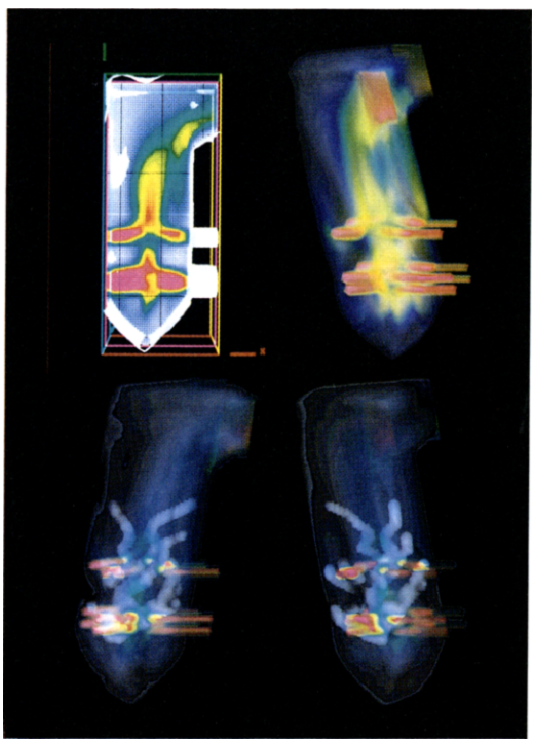

Fig. 2. Multiple views of flow velocity.

scale utility boiler using IVES. These power generation systems typically stand 300 feet high, 30 feet wide and generate 500 mega-watts of electricity. The upper-left picture is a combination of isosurface and two-dimensional color contour rendering. The upper-right picture is an ordinary ray-casting volume rendering of the velocity magnitude. The lower-left and right pictures give two views of the vector field. A "seed" has been planted in the volume and is grown into a seedling based on the direction and magnitude of the velocity at each voxel. The seedling grows from the seed as though an opaque dye was being emitted from the seed point in the fluid. The seedling growth occurs dynamically and can be viewed in real time. The rate and direction of the seedling growth can provides many cues about the flow field, particularly when several views are available simultaneously. The ability to watch flow movement in real-time is extremely powerful and far less expensive than experimental flow visualization in laboratories.

Because of this real-time activity, workstations connected with an Ethernet network do not provide sufficient bandwidth to update multiple view in real time. A shared-memory multiprocessor such as the IRIS 4D/240 GTX is more appropriate for the task. The above multiple-view rendering has been implemented in the IRIX parallel programming primitives, which is a subset of the Sequent parallel programming primitives[17]. IRIX supports calls for creating processes that can execute in parallel, synchronization primitives such as locks and semaphores, and shared memory allocation routines. As IVES requires extra memory space for caching ray samples, the number of views can be rendered and displayed simultaneously is limited by available memory.

6. MULTIPLE-VARIABLE VISUALIZATION

As previously discussed, the ability to view multiple related data sets simultaneously allows scientists to relate important interactions in the simulation. For example, a simulation of a pilot utility boiler conducted by a local scientist produced 35 scalar data sets. To hold all the data sets in main memory for concurrent viewing is difficult. What we can do is to distribute the 35 scalar data sets to multiple networked workstations. One workstation with high-resolution graphics support is used as the host, and displays the main user interface and desired images. Remotely computed images are sent back to the host for display. Figure 3 shows four images each of which describes a scalar from the simulation of the industrial broiler. The upper-left one displays the wall structure, the upper-right the temperature, the lower-left CO concentration and the lower-right CO_2 concentration in the furnace. While examining these four pictures, the user can submit another job that renders a different combination of scalars. The level of interactivity depends on the processing power of each workstation and the image quality desired. Usually a 640×640-pixel image of reasonable resolution, casting one ray for every other pixel, takes about two minutes on an IBM RS/6000 Model 530. A smaller image can be rendered within seconds.

7. DIVIDE-AND-CONQUER VOLUME RENDERING

Volume data sets can be quite large, often too large for a single workstation to hold in memory at once. Such data sets can still be rendered by a divide-and-conquer algorithm: divide the data up into smaller subvolumes, render them separately, and combine the resulting images. If multiple workstations are available,

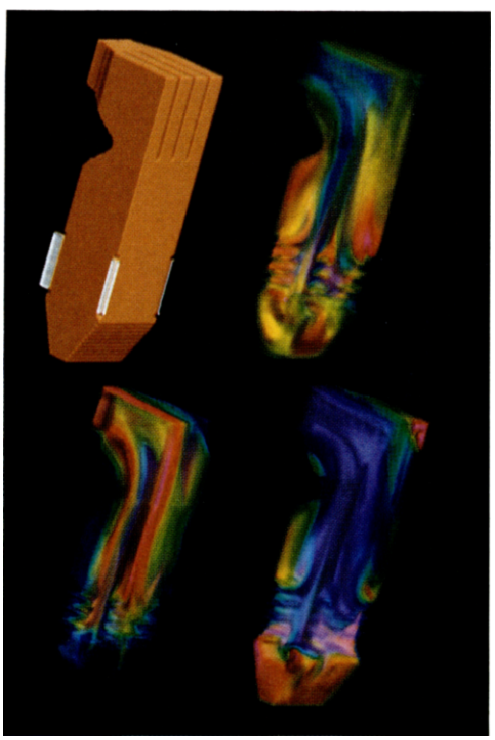

Fig. 3. Visualization of multiple scalars in a combustion simulation.

a parallel algorithm results by distributing the subproblems to different machines. The memory demands on the workstations are modest since each workstation need only hold a subset of the total data set. This approach can be used to render high resolution data sets in an environment with many midrange workstations (*e.g.*, equipped with 16 megabytes memory) on a local area network. Many scientific and engineering computing environments have an abundance of such workstations that could be harnessed for volume rendering provided that the memory usage on each machine is reasonable.

7.1. *Data subdivision*

The divide-and-conquer algorithm requires that we partition the input data into subvolumes. There are many ways to partition the data; the only requirement is that an unambiguous front-to-back ordering can be determined for the subvolumes to establish the required order for compositing subimages. Ideally each subvolume will require about the same amount of computation. Further, if the viewpoint is known, we should subdivide in a manner that minimizes the overlap between the images resulting from the subvolumes. This will reduce the cost of merging since compositing need only be applied where subimages overlap.

The simplest method is to partition the volume along planes parallel to the coordinate planes of the data. If the viewpoint is fixed and known when partitioning the data, the coordinate plane most nearly orthogonal to the view direction can be determined and the data can be subdivided into "slices" orthogonal to this plane. When orthogonal projection is used, this will tend to produce subimages with small overlap. If the viewpoint is not known, or if perspective projection is used, it is better to partition the volume equally along *all* coordinate planes. This can be accomplished using a k-D tree structure[18], with alternating binary subdivision of the coordinate planes at each level in the tree as indicated in Fig. 4. Front-to-back image compositing order can be determined hierarchically by a recursive traversal of the k-D tree structure, visiting the "front" child before the "back" child. This is similar to well-known front-to-back traversals of BSP-trees[19, 20] and octrees[21, 22]. In addition, the hierarchical structure provides a natural way to accomplish the compositing in parallel: Sibling nodes in the tree may be processed concurrently.

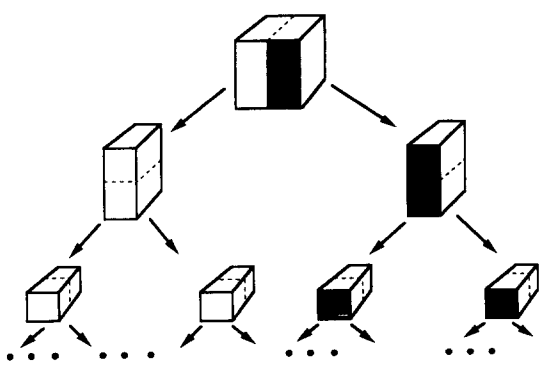

Fig. 4. K-D tree subdivision of a data volume.

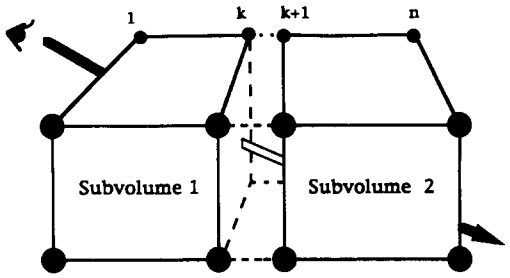

Fig. 5. Volume boundary replication.

As shown in Fig. 5, when a volume of grid points (voxels) is evenly subdivided into, for example two subvolumes, each subvolume may contain half of the total grid points. Note that each voxel is located at a corner of the grid. Consequently, those ray samples that lie between the cut boundary region (the dotted region) are lost. If the view vector is parallel to the cut plane, a black strip will appear at each cut boundary in the composited image. Therefore, we need to duplicate one layer of the boundary grid at each subvolume so the composited ray-casting image does not drop out features originally in the volume. For the case shown in Fig. 5, one possible arrangement is that Subvolume 1 includes layer *1* to layer k and Subvolume 2 includes layer k to layer n; that is, in S_2, layer k is replicated.

7.2. *Subvolume rendering*

Note that we use ray-casting volume rendering. Each workstation can perform raytracing independent of other workstations. There is no data communication required during the subvolume rendering. All subvolumes are rendered using an identical view position and only rays within the region covering the corresponding subvolume are cast and sampled. Since we sample along each ray at a predetermined interval, consistent sampling locations must be ensured for all subvolumes so we can reconstruct the original volume. As shown in Fig. 6, for example, the location of the first sample $S_{2,1}$ on the ray shown in Subvolume 2 should be calculated correctly so that the distance between $S_{2,1}$ and $S_{1,n}$ is equivalent to the predetermined interval. Otherwise, small features in the data might be enhanced in an erroneous way.

7.3. *Image merging*

In order to apply the divide-and-conquer algorithm we must develop a method for merging the images

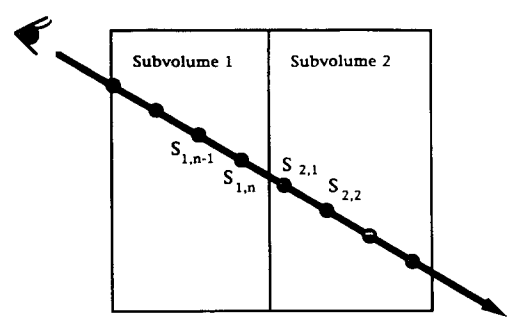

Fig. 6. Correct ray sampling.

resulting from the subvolumes. In order to correctly merge subimages into the final total image, we need to store not only the color at each pixel but also the accumulated opacity there. The rule for merging subimages is based on the **under** compositing operator of [23]. **Under** operator is associative. To prove that the composited samples along a single ray onto a total volume is equivalent to the merging of multiple rays onto subvolumes, let's consider the following case. If we have n samples along a single ray, which are

$$C_1\alpha_1,\ C_2\alpha_2,\ C_3\alpha_3,\ \ldots,\ C_{n-1}\alpha_{n-1},\ C_n\alpha_n,$$

then the accumulated color of the ray C is

$$C_1\alpha_1 + (1-\alpha_1)[C_2\alpha_2 + (1-\alpha_2) \times [C_3\alpha_3 + \cdots C_n\alpha_n];$$

that is,

$$C = \sum_{i=1}^{n} C_i\alpha_i \prod_{j=1}^{i-1}(1-\alpha_j).$$

Suppose we subdivide the total volume evenly into two subvolumes. The color of the ray segment in the front subvolume is

$$C_f = \sum_{i=1}^{n/2} C_i\alpha_i \prod_{j=1}^{i-1}(1-\alpha_j),$$

and the color of the ray segment in the back subvolume is

$$C_b = \sum_{i=n/2+1}^{n} C_i\alpha_i \prod_{j=n/2+1}^{i-1}(1-\alpha_j).$$

According to the **under** operation,

$$C = C_f + (1-\alpha_f)C_b$$

where

$$\alpha_f = \sum_{i=1}^{n/2} \alpha_i \prod_{j=1}^{i-1}(1-\alpha_j).$$

Therefore, we can derive

$$\begin{aligned} C &= C_f + (1-\alpha_f)C_b \\ &= C_f + \Big(\prod_{j=1}^{n/2}(1-\alpha_j)\Big)C_b \\ &= C_f + \Big(\prod_{j=1}^{n/2}(1-\alpha_j)\Big)\sum_{i=n/2+1}^{n} C_i\alpha_i \prod_{j=n/2+1}^{i-1}(1-\alpha_j) \\ &= \sum_{i=1}^{n/2} C_i\alpha_i \prod_{j=1}^{i-1}(1-\alpha_j) + \sum_{i=n/2+1}^{n} C_i\alpha_i \prod_{j=1}^{i-1}(1-\alpha_j) \\ &= \sum_{i=1}^{n} C_i\alpha_i \prod_{j=1}^{i-1}(1-\alpha_j) \end{aligned}$$

in which

$$\begin{aligned} (1-\alpha_f) &= 1 - \sum_{i=1}^{n/2} \alpha_i \prod_{j=1}^{i-1}(1-\alpha_j) \\ &= (1-\alpha_1)\Big(1 - \sum_{i=2}^{n/2} \alpha_i \prod_{j=2}^{i-1}(1-\alpha_j)\Big) \\ &= (1-\alpha_1)(1-\alpha_2)\Big(1 - \sum_{i=3}^{n/2} \alpha_i \prod_{j=3}^{i-1}(1-\alpha_j)\Big) \end{aligned}$$

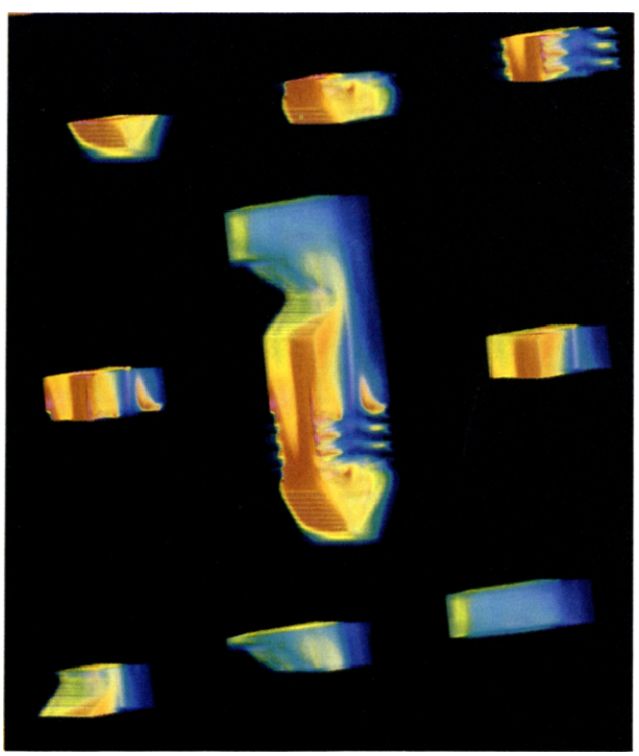

Fig. 7. Subimages and their composite.

$$= \prod_{j=1}^{n/2}(1 - \alpha_j)(1 - 0)$$

$$= \prod_{j=1}^{n/2}(1 - \alpha_j)$$

Figure 7 shows images for eight subvolumes and their composited volume in the middle. The composited image should be identical to the image that is rendered directly from the total volume if numerical errors can be avoided, for example by using fixed point arithmetic. According to our experimental results, optimal linear speedup can be easily obtained on networked workstations since the rendering of a smaller volume can better take advantages of the locality of memory access; in addition, the time needed to composite subimages can be negligible since a carefully coded compositing operation takes in milliseconds compared with seconds to minutes needed for the rendering of all the subimages.

This divide-and-conquer scheme is best suited for mid and large-grain parallelism. For massively parallel computers, which may have many thousands of processors, the image compositing time may become significant. A set of workstations connected with Ethernet or token ring network are best suited for processing large data sets that a single workstation cannot handle efficiently. If a fiber optic network is available, then even real-time rendering can be achieved on small to medium scale data sets.

8. CONCLUSIONS

We have described methods that utilize general-purpose graphics workstations to do interactive, high-resolution volume visualization. We have demonstrated that the memory and computational demands can be distributed across multiple workstations, allowing a network of workstations to be harnessed for high performance, high resolution volume visualization of large data sets. We have shown examples applying these methods and have demonstrated their effectiveness and practicality. The ability to perform feature extraction in an interactive manner offered by IVES makes these multiprocessing schemes even more powerful and attractive. Currently we are developing graphics user interface for managing distributed volume rendering on networked workstations. We are also investigating the practicality of applying these methods to other volume visualization algorithms.

Acknowledgments—This work has been supported in part by NSF/ACERC and an IBM grant for Scientific Visualization. The Medical Imaging Laboratory at the University of Utah provided the MRA data set. Dr. Philip Smith at ACERC provided the furnace data set. Elena Driskill implemented an early version of the data slicer used in IVES. Mihir Mehta integrated an isosurface program into the data slicer. Special thanks go to Dr. Michael Cohen for his guidance during the development of IVES.

REFERENCES

1. P. Dew, R. Earnshaw, and T. Heywood, (Eds.) *Parallel Processing for Computer Vision and Display.* Addison Wesley, Reading, MA (1989).
2. J. Foley, A. V. Dam, S. K. Feiner, and J. F. Hughes, *Computer Graphics: Principles and Practice.* Addison Wesley, Reading, MA (1990).
3. S. Green, *Parallel Processing for Computer Graphics.* MIT Press, Cambridge, MA (1991).
4. K.-L. Ma, M. F. Cohen, and J. Painter, Volume seeds: A volume exploration technique. *J. Visual. Comp. Anim.* **2**(4), 135–140 (1991).
5. M. F. Cohen, J. S. Painter, M. Mehta, and K.-L. Ma, Volume seedlings. In *Proceedings of the ACM Interactive 3D Graphics* (March 1992).
6. R. A. Drebin, L. Carpenter, and P. Hanrahan, Volume rendering. *Computer Graphics (Proceedings of SIGGRAPH 1988)* **22**(4), 65–74, (1988).
7. M. Levoy, Display of surfaces from volume data. *IEEE Computer Graphics and Applications,* 29–37 (May 1988).
8. C. Upson and K. M., V-buffer: Visible volume rendering. *Computer Graphics (Proceedings of SIGGRAPH 1988)* **22**(4), 59–64 (1988).
9. M. Levoy, Efficient ray tracing of volume data. *ACM Transactions on Graphics* **9**(3), 245–261 (1990).
10. A. Kaufman and R. Bakalash, Memory and processing architecture for 3D voxel-based imagery. *IEEE Computer Graphics and Applications* **8**(6), 10–23 (1988).
11. H. Fuchs, J. Poulton, J. Eyles, T. Greer, J. Goldfeather, D. Ellsworth, S. Molnar, G. Turk, B. Tebbs, and L. Israel, Pixel-planes 5: A heterogeneous multiprocessor graphics system using processor-enhanced memories. *Computer Graphics (SIGGRAPH '89 Proceedings)* **23**(3), 111–120 (1989).
12. P. Schröder and J. B. Salem, Fast rotation of volume data on data parallel architectures. In *Proceedings of Visualization '91,* 50–57 (October 1991).
13. T. T. Elvins and D. Nadeau, NetV: An experimental network-based volume visualization system. In *Proceedings of Visualization '91,* 239–245 (October 1991).
14. L. Westover, Interactive volume rendering. In *Proceedings of Chapel Hill Workshop on Volume Visualization,* 9–16 (May 1989).
15. T. Yoo. U. Neumann, H. Fuchs, S. Pizer, T. Cullip, J. Rhoades, and R. Whitaker, Achieving direct volume visualization with interactive semantic region selection. In *Proceedings of Visualization '91,* 58–68 (October 1991).
16. K.-L. Ma and P. J. Smith, Virtual Smoke: An interactive 3d flow visualization technique. In *Proceedings of Visualization* '92, 46–53 (October 1992).
17. A. Osterhaug, *Guide to Parallel Programming on Sequent Computer Systems.* Sequent Computer Systems, Inc. (1986).
18. J. Bentley, Multidimensional binary search trees used for associative searching. *Commun. ACM* **18**(8), 509–517, (1975).
19. H. Fuchs, Z. M. Kedem, and B. F. Naylor, On visible surface generation by a priori tree structures. In *Proceedings of SIGGRAPH 1980,* 58–67 (1980).
20. H. Fuchs, G. Abram, and E. D. Grant, Near real-time shade display of rigid objects. In *Proceedings of SIGGRAPH 1983,* 65–72 (1983).
21. L. Doctor and J. Torborg, Display techniques for octree-encoded objects. *IEEE Comput. Graphics and Appl.* **1,** 29–38, (July 1981).
22. D. Meagher, Geometric modeling using octree encoding. *Comput. Graphics Image Process. (USA)* **19,** 129–147 (June 1982).
23. T. Porter and T. Duff, Compositing digital images. *Computer Graphics (Proceedings of SIGGRAPH 1984)* **18**(3), 253–259 (1984).