Fast Volume Visualization From Composited Two-Pass Affine Image Transformations

Wes Bethel, Vince Beckner, Terry Ligocki
Berkeley National Laboratory

1.0 Abstract

As computing capacity advances, scientific computations similarly increase in resolution producing more and more data. Increasing capacity and resolution exacerbate the "fire-hose of data" problem, elevating the issue of understanding and analyzing data from academic to urgent. Visualization, the process of creating images from abstract data suitable for interpretation by humans, has been successfully used for feature "triage" as well as for detailed inspection of features within data.

We present a method for visualizing large three-dimensional volumes of data in a parallel computing environment. Our method is an alternative "shear-warp-like" parallel volume rendering algorithm. The primary design goal motivating our work is the need to visualize large scientific datasets at interactive rates over a standard ethernet line to the desktop from a one of a variety of multiprocessor architectures. Previous work in parallel volume rendering has focused upon difficult-to-implement parallel versions of a particularly fast serial algorithm. In contrast, our work outlines the development of a serial volume visualization technique with the important property that a parallel implementation is both efficient and straightforward.

2.0 Introduction

Volume rendering is a visualization technique which has been successfully used for creating images of three-dimensional data. Volume visualization has the unique property that the entire the dataset is rendered *in toto*. In contrast, other visualization techniques extract a surface from the volume (isosurface) or take "slices" from a three-dimensional array of data. Over the years, numerous approaches have been evaluated for volume rendering. The various techniques bifurcate into either image-order and object-order. Image-order algorithms start from on-screen pixels and then integrate back into the data, computing the contribution from voxels to the screen pixel. Object order algorithms start from the volume data itself, and scan convert from voxels to pixels.

For software implementations, the current "recognized champion" for volume rendering is the Lacroute and Levoy "shear-warp" algorithm [1]. Frame rates of 1 per second using a desktop machine and data sizes of 256**3 were reported in 1994. Shear-warping makes use of both image-space and object-space coherence to achieve a high level of efficiency in software. A discussion of this algorithm, both in serial and parallel implementations, will serve as the jumping-off point for our work.

Our work closely parallels that of the earlier shear-warp implementations, but with some variations in parallelization strategy as well alterations in the fundamental shear-warp kernel. We had one primary goal in mind when beginning this work: delivery of direct volume rendered images of realistic-sized scientific data to the desktop at interactive frame rates over a standard ethernet connection from one of several different MP architectures.

3.0 The Serial Shear-Warp Formulation

The shear-warp method is based upon a factorization of the viewing and transformation matrices in such a way that slices of volume data are first sheared in three dimensions, parallel to the line of sight, followed by an image composition stage, then finally a single two-dimensional image warp (see Figure 1). The guiding observation is that parallel slices of volume data remain parallel regardless of orientation, and more importantly, remain parallel to each other with respect to the viewer. Note that this volume rendering technique works only upon structured meshes which are "well behaved," or, in other words, rectilinear.

4.0 The Parallel Shear-Warp

Lacroute [3] describes a parallel implementation of the shear-warp algorithm on SMP-class machines. The discussions of load balancing and algorithm partitioning are worthwhile for review, but the SMP architecture is unforutnately inadequate in terms of meeting one of our design goals: rendering of "very large" data. A distributed memory

architecture provides for higher memory capacities than are currently available on SMP architectures.

Amin et. al. [2] present a shear-warp parallelization for a distributed-memory architecture. They propose to partition work in "sheared volume" space as a compromise between a purely object-order or image-order decomposition. In the sheared-volume space, the partitioning plane is parallel to the line-of-sight. Each processor, or work unit, can "drive

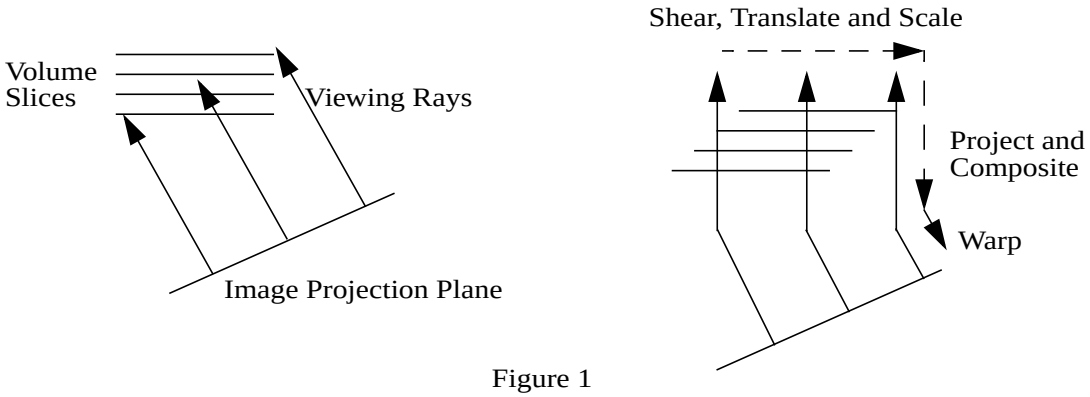


Figure 1

rays" through its assigned volume segment. The resulting image fragments are disjoint and can be independently warped, and since partial intermediate images are disjoint, the corresponding partial warped images are also disjoint. Because of this, no compositing is required across processor boundaries. The principal overhead for this partitioning scheme results from communication of volume data when the volume is sheared. They also note that their method works exclusively for parallel projections and preclassified volumes, as well as the fact that parallelizing the shear-warp factorization "presents considerable challenges."

The challenge of parallelizing a perspective projection version of the shear-warp lies in the fact that the perspective foreshortening effect induces non-uniform per-slice voxel scaling. This has the effect of disrupting the object-order partitioning of the parallel implementation described in [2].

5.0 Compositions of Two-Pass Affine Warps

With the existing shear-warp work in mind, we now describe our effort. One departure from the classical shear-composite-warp formulation is to perform the shear and warp in one step, then perform the composition.

With this approach, we can implement a highly-efficient two-pass image warping kernel of the form described by Smith and Catmull in 1980 [4]. Such a warping kernel has been shown to be highly efficient, with at least one vendor implementing this image manipulation tool in hardware[1]. An additional benefit of performing the shear and warp prior to the composition phase is that perspective projections can be performed "for free." The non-uniform voxel scaling is trivially accomodated using the image warping algorithm in the pre-composition phase. Using the two-pass transformation, there is no benefit gained from separating the shear from the warp. They can be performed at once for less than the cost of performing them in separate stages.

5.1 The Two-Pass Kernel

The two-pass transformation was originally used for performing object-order texture mapping operations in early systems. The general motivation for its use follows from the decomposition of a two-dimensional filtering operation into two serial and separable (and simpler) filters.

For the general warp of a bilinear patch, we start with a patch which is parameterized over two variables and warp it into a new spatial location (Figure 2):

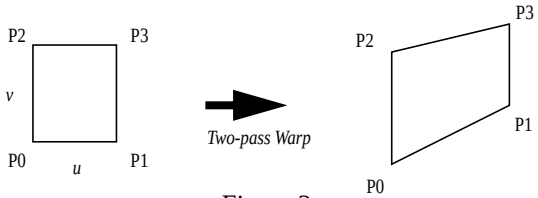


Figure 2

The general representation for the bilinear patch has the representation $x(u, v) = \begin{bmatrix} u & 1 \end{bmatrix} \begin{bmatrix} a00 & a01 \\ a10 & a11 \end{bmatrix} \begin{bmatrix} v \\ 1 \end{bmatrix}$

where a00=(x3-x2)-(x1-x0), a01=x1-x0, a10=(x2-x0), and a11=x0. There are similar formulations for the *y* (and *z*) components of a bilinear patch. For our purposes, since we are only warping one two-dimensional image into another two-dimensional image in the same plane, we can effectively ignore the z-coordinate in the

1. "System for Spatially Transforming Images," Philip P. Bennett and Steve A. Gabriel, U.S. Patent Number 4,472,732. Assigned to Ampex Corporation.

general two-pass affine transformation. The general warp transformation is then represented as:

$$\begin{bmatrix} uv & u & v & 1 \end{bmatrix} \begin{bmatrix} a00 & b00 \\ a01 & b01 \\ a10 & b10 \\ a11 & b11 \end{bmatrix} \quad (1)$$

This formulation is separable into disjoint *x*- and *y*-component passes, such that we first warp horizontal then vertical scanlines.

Two primary benefits are realized from the two-pass formulation. First, a two-dimensional filter can be replaced by two serial one-dimensional filters. Second, a broader range of image transformations can be applied than those described in Lacroute and Levoy. Most notably, we get perspective projection “for free.” A third benefit will be discussed when we discuss parallelization.

5.2 Performance Analysis of Serial Implementation

The detailed analysis of the serial version of the two-pass compositing volume renderer has been removed from the abbreviated paper. A synopsis of the analysis is as follows. The aggregate cost of the algorithm is a function of:

- The cost of performing the two-pass transformation upon a single slice of volume data, and
- The cost of compositing the resulting images together.

The cost of the two-pass operation is a function of both the size of the slice of volume data, as well as the size of the image into which the volume data is “warped.” The analysis captures this relationship. The cost of compositing is a function of the size of the final warped image (only). When the final image is smaller than the size of a volume data slice, the cost of compositing is negligible compared to the cost of the two-pass transformation.

We present a table showing the cost in MFLOPS of the two-pass transformation, the image composition, and then the entire volume rendering using varying parameters for input volume data size and final image size.

From these tables, we can make the following observations:

- The “brute force” two-pass formulation of the shear warp will render small datasets at a reasonable rate of speed on a desktop workstation.
- The theoretical cost of the “brute force” two-pass formulation is comparable to the original shear-warp algorithm. Consider the performance of the highly-optimized shear warp algorithm when processing extremely noisy data, i.e., data which has very few zero-valued pixels and very short voxel runs.
- Increasing the size of the final image has a substantial effect upon cost.
- For the larger volumes, reasonable rendering times are beyond the reach of any conventional or foreseeable workstation. The only hope for near-interactive rendering rates is to make use of a large MPP system.

5.3 Parallel Implementation

An object-order decomposition is used in the parallelization of the serial two-pass algorithm. Each PE will operate upon some number of volume data slices, or slice clusters. For slice within each cluster, the PE will perform the two-pass transformation upon a single slice of volume data, then composite the resulting image into an accumulation image. When all such slices within the chunk are processed, at each PE, all the resulting images are composited together in a tree-like reconstruction (Figure 3).

Volume data decomposed into “slice clusters”. Each slice of each cluster is warped and then composited together. Finally, the intermediate images are composited together to form a final image.

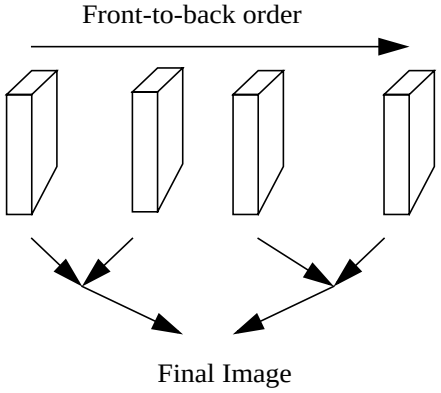


Figure 3

Like all volume rendering algorithms, composition [5] must occur in front-to-back or back-to-front order. Similar to the work described in [1], we also require three copies of data, one for each of the principle viewing axes.

In parallel projections, the time required to render each slice cluster is nearly the same (assuming the same num-

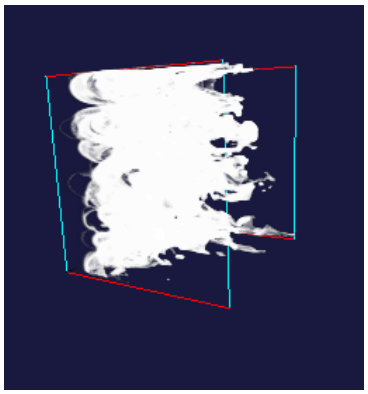
ber of slices per cluster). In perspective projections, using an identical number of slices per cluster induces a load imbalance. This is due to the fact that slice clusters closer to the viewer result in images which cover more screen pixesl than those further away. The two-pass image transformation kernel is run time is indeed sensitive to final image size. Thus, future work on load balancing will take into account the perspective foreshortening effect, and will compute the number of slices per cluster in such a way as to achieve nearly equal shear-warp time per slice. The present implementation assigns a nearly equal number of slices per cluster, and in the perspective projection case, we have observed an average 75% load balancing level.

5.4 Results

The table below shows various statistics for our implemtation as run on a 256x256x256 dataset (we're presently working on a 1024x1024x1024 dataset), and rendered at 256x256. Each row represents statistics collected for a given number of processors. The statistics show elapsed time in milliseconds for the shear-warp and image composition within a particular slice cluster (SW) and the elapsed time for the image composition across all the processors.: average run time across all PE's for cost of computing a finished image for that slice cluster (SW); the average cost of performing the collective image composition (Comm). An interesting observation is that

N PE's	SW	Comm
4	2497	247
8	1390	301
16	816	213
32	542	167

on the Cray T3E-900, that as more PE's become involved in communication, that the internal communication network becomes more efficient. We expect that this trend will "bottom out" as the number of PE's increases. The performance of the two-pass kernel scales nicely with increasing numbers of PE's.



Shock Wave Propogation

Figure 4

6.0 Conclusions

The serial implementation of our volume renderer, based upon compositions of two-pass affine transformations, is not as "fast" as the classical shear-warp algorithm with all its optimizations. On the other hand, what we sacrifice in the serial implementation, we gain in the potential to interactively visualize large datasets which will not fit into primary memory on workstations or SMP's. Further optimizations which takes advantage of voxel runs, or skipping over transparent voxels, will contribute to an overall improvement in algorithmic performance.

7.0 Bibliography

1. "Fast Volume Rendering Using a Shear-Warp Factorization of the Viewing Transformation," Computer Graphics 28(3), Proceedings of Siggraph 1994.

2. "Fast Volume Rendering Using an Efficient, Scalable Parallel Formulation of the Shear-Warp Algorithm," Minesh B. Amin, Anath Grama, Vineet Singh. 1995 Parallel Rendering Symposium, ACM Siggraph, October 1995.

3. "Real-Time Volume Rendering on Shared Memory Multiprocessors Using the Shear-Warp Factorization," Philippe Lacroute, Proceedings of the 1995 Parallel Rendering Symposium.

4. "3-D Transformations of Images in Scanline Order," Ed Catmull and Alvy Ray Smith, Computer Graphics 12(3), Proceedings of Siggraph 1980.

5. "Compositing Digital Images," Thomas Porter and Tom Duff, Computer Graphics 18(3), Proceedings of Siggraph 1984.