Chapter 9

Conclusions

9.1 Final Summary

Many volume rendering algorithms have been proposed in the last decade, yet volume rendering has not become a widely-used visualization technique because of the lack of interactive systems. By a system we mean an algorithm combined with the hardware it runs on. Existing algorithms are too slow to provide interactive response on a workstation, and multiprocessors with sufficient computational power are too expensive for general-purpose use. Interactive rendering rates are essential because the process of visualizing volume data requires experimentation.

This thesis describes a family of new volume rendering algorithms that delivers interactive rendering rates on current-generation workstations. The algorithms improve on existing methods by combining the advantages of two classes of algorithms: image-order ray casting algorithms and object-order splatting algorithms. Like other object-order algorithms, the new shear-warp algorithms stream through the volume data in storage order so they incur little overhead due to addressing arithmetic and data structure traversal. In contrast, these overheads constitute a substantial fraction of rendering time in image-order coherence-accelerated algorithms.

The shear-warp algorithms also benefit from synchronized scanline-order access to both the volume data and the image, making it possible to use coherence in the image to implement early ray termination. Most previous uses of early ray termination have been restricted to ray casting algorithms. Furthermore, implementing efficient, high-quality resampling filters in the shear-warp algorithms is as straightforward as in image-order algorithms.

The technique that allows us to traverse both the volume and the image in storage order is the shear-warp factorization: a factorization of the viewing matrix into a 3D shear, a projection into a distorted 2D intermediate image, and a 2D warp that transforms the intermediate image into the correct final image. We showed how to use the factorization and coherence data structures based on run-length encoding to develop a parallel-projection volume rendering algorithm for classified volumes. We then showed how to extend the factorization to perspective projections and developed a fast perspective volume rendering algorithm. Finally, we developed coherence data structures based on a min-max octree and a summed-area table that, when combined with either rendering algorithm, allows interactive classification of volume data. These three algorithms complement each other by addressing different phases of the visualization process.

Performance tests show that on current desktop workstations the new parallel projection algorithm can render 256^3 medical data sets in less than one second and 128^3 data sets at over 5 frames per second. These speeds are over five times faster than previously reported results for a ray caster with an octree. The fast classification algorithm can classify and render a 128^3 volume in under a second. None of these algorithms use any specialized hardware so the rendering rates will continue to improve as faster workstations appear on the market, and the algorithms are portable to any general-purpose machine. Rendering speed no longer limits the usefulness of volume rendering.

We have also parallelized the rendering algorithms for shared-memory multiprocessors. On a 16-processor SGI Challenge the parallel projection algorithm renders 256^3 data sets at over 10 frames per second and 128^3 data sets at over 25 frames per second. Inexpensive small- to medium-scale shared-memory multiprocessors based on fast microprocessors are likely to become common in the coming years, and the multiprocessor shear-warp volume rendering algorithm is well-suited to this type of architecture.

Finally, we have demonstrated that the different shear-warp algorithms can be integrated into a practical interactive volume rendering system. We have presented an efficient method for precomputing the necessary data structures when switching between the rendering algorithms, and we have presented extensions for rapidly rendering a variety of visual cues:

depth cueing, shadows, clipping planes, and lookup-table based shading functions. These visual cues help to produce informative, easily-interpreted visualizations.

The rendering algorithms presented in this thesis have several limitations. Most important is that the speed of the algorithms depends on the degree of coherence in the volume. Classified volumes containing large transparent regions or large opaque regions result in fast rendering rates. The algorithms exhibit slower performance if a substantial fraction of the volume consists of low-opacity voxels. However, such data sets typically result in fuzzy, indistinct images that do not convey useful information. Even if the original volume contains little apparent coherence, in our experience properly-classified volumes expose inherent structure in the volume. Thus substantial performance gains can be achieved by exploiting coherence.

The shear-warp algorithms also restrict the choice of resampling filter since 3D filter kernels cannot be supported. In practice the algorithms produce high-quality images provided the volume data is alias-free, so image quality is not a major limitation. The fast classification algorithm also places restrictions on the classification function. The algorithm supports multi-feature classification based on user-specified transfer functions.

9.2 Future Directions for Performance Improvements

Our performance measurements lead to two important observations that impact future work on high-performance volume visualization algorithms. First, we have found that object-order scanline-based algorithms are inherently more efficient than image-order algorithms. Streaming through the volume in storage order will always be faster than traversing the volume in an arbitrary view-dependent direction, and we have shown that in coherence-accelerated algorithms the performance difference between these two alternatives is significant. For this reason scanline-order algorithms are likely to dominate volume rendering in the future.

Scanline-order algorithms can yield performance gains in other volume-processing operations as well. For example, one application is to combine a series of range images (acquired with a laser range scanner) into a 3D volumetric model called an *occupancy grid*. One existing method operates by shooting rays from the range image pixels into a 3D grid and storing occupancy information at the lattice points [Connolly 1984]. However, a scanline-order algorithm based on the shear-warp factorization would be faster than a ray casting algorithm.

Even for a volume sampled on a non-rectangular grid, such as a curvilinear grid, we speculate that the fastest rendering algorithm may be an object-order algorithm that first resamples the volume to a rectangular grid and then uses a scanline-order rendering algorithm.

The second important observation is that there is no single performance bottleneck remaining in the shear-warp volume rendering algorithm. The rendering time is divided roughly equally between essential computation (shading, resampling and compositing voxels), overhead due to the coherence data structures and control logic, and memory overhead. It will therefore be difficult to further improve the algorithm. Remembering Amdahl's Law, we observe that optimizations that improve on any one of these areas will not have a major impact on overall execution time. Optimizations that reduce the cost per voxel, more efficient coherence data structures, and faster memory systems will not individually reduce rendering time by more than about 10-20%.

The most likely approach for improving performance is to trade off image quality for speed by further reducing the number of voxels processed, thereby potentially reducing all three cost categories. Techniques such as adaptive sampling [Levoy 1990a] and subsampling the volume [Laur & Hanrahan 1991] have been proposed and could be applied to the shear-warp algorithm. Approaches such as these or techniques based on rendering compressed volume data [Ning & Hesselink 1993, Muraki 1994] may also be effective ways of handling very large data sets, an important topic not addressed in this thesis.

9.3 Hardware Support for Volume Rendering

Our performance results also have implications for hardware designers. Specialized instruction sets for graphics appear not to be useful for volume rendering. This class of instructions is designed for accelerating inner loops that operate on pixels. The instructions perform operations in parallel on several pixels or color channels packed into one machine word. The processor includes special hardware to perform the packing and unpacking operations, and to perform arithmetic on the independent components in parallel. Such instructions may be useful for brute-force resampling and compositing loops, but we have shown that only a small fraction of the time in coherence-accelerated volume rendering algorithms is spent processing voxel data. By Amdahl's Law the overall speedup achievable with graphics instructions is negligible.

In a broader context, we speculate that many graphics algorithms can be accelerated by using high-level algorithmic optimizations and data structures that have the side effect of eliminating opportunities to use special-purpose graphics instructions. Unless the graphics instructions provide better speedups, the software solution is the preferred alternative.

High-performance hardware-accelerated volume rendering has recently become feasible with the advent of 3D texture mapping hardware in high-end graphics workstations [Cabral et al. 1994]. A volume can be treated as a 3D solid texture. To render the volume an application generates a sequence of polygons that slice through the volume and are parallel to the plane of the image. The graphics hardware then shades the polygons using the volume as a texture and composites the polygons together. Classification can be performed in hardware by mapping the voxel values through a lookup table containing the opacity transfer function.

Using this algorithm, current-generation high-end graphics workstations can achieve sub-second rendering rates and can support interactive classification. However the hardware is expensive and cannot support sophisticated shading or classification. The only type of shading available is a mapping from the original scalar value to a color (implemented via a hardware lookup table). The Phong illumination model, which requires the gradient of the scalar value, cannot be implemented. Assuming that the cost of the hardware drops in the future, this lack of flexibility is the chief short-coming of 3D texture mapping hardware used for volume rendering. Nevertheless, for some specialized applications supported by the hardware this approach may be an effective solution.

However, a second argument against the widespread use of brute-force graphics hardware for volume rendering is the large performance difference between brute-force and optimized algorithms. For example, compared to the brute-force ray caster, the optimized shear-warp algorithm is 25-50x faster (as shown in Chapter 5). Hardware solutions require large amounts of hardware just to match this speedup. This situation is different than for the case of polygon rendering hardware since the speedups achievable with software optimizations are not as high. Polygon rendering algorithms that compute visibility prior to rasterization (such as Watkins' scanline algorithm and Warnock's area-subdivision algorithm [Foley et al. 1990, Ch. 15]) can reduce the cost of rasterization, but even if we ignore the costs of computing visibility the maximum possible speedup equals the depth complexity of the scene. A rule of thumb for the average depth complexity in typical polygonal scenes is three to four [Molnar et al. 1994], so the speedup necessary to make hardware solutions attractive is much lower for polygons than it is for volumes. Thus even though polygon rendering hardware is becoming widely available, it is less likely that hardware for volume rendering will become cost effective.

Special-purpose hardware implementations of a specific algorithm will always be faster than a software-only implementation, but fully-custom hardware is too expensive for general use. We believe the most cost-effective platform for volume rendering consists of simple, fast, general-purpose microprocessors combined with optimized software algorithms. Inexpensive small-scale shared-memory multiprocessors built from general-purpose microprocessors will continue this trend.

9.4 Interactive Volume Rendering

A fast volume rendering system is the first step towards the widespread use of volume rendering. The next step is to create effective user interfaces that provide control over the many parameters required by the rendering algorithm without overwhelming the user. Choosing classification functions that expose the information in a volume and tuning shading parameters to highlight that information are complicated tasks.

In the past, much research has been devoted to the automatic choice of parameters. In particular, many algorithms have been proposed for automatic classification by segmenting volume data (e.g. distinguishing individual organs and tissue types in a medical data set). With a slow rendering system it is not practical to require a user to choose parameters interactively, so an automatic algorithm is attractive. Unfortunately, with the exception of methods for certain restricted applications (such as classification of high-contrast structures in CT data), automatic algorithms are not robust and do not produce reliable results.

Fast volume rendering enables a new approach to processing and visualizing volume data. Instead of using complicated, unreliable batch algorithms, an application can provide a set of simple controls that can be adjusted by the user. As long as the user can immediately see the result while adjusting a control, the feedback loop created by the rendering system and the user creates a much more robust and stable system than an automatic batch algorithm.

As an example, medical CT data is often classified by specifying a “window”: a target data value and a width indicating a tolerance around the target value. The target value can be adjusted to select a tissue type with a particular CT value, and the window width excludes tissues with different values. A user interface with two controls tied to the window value and width, combined with a fast volume rendering system, would allow a user to rapidly select tissues of interest.

A fast volume rendering system also enables new applications. Morphing of 3D volumetric models is now computationally feasible [Lerios et al. 1995]. 3D morphs are superior to traditional 2D morphs because occlusions and lighting changes can be accurately rendered in the intermediate frames, but a fast rendering system is necessary to make the technique practical. Another new application is simplification of complex geometry via 3D scan conversion into a volumetric model, which is a useful method for manipulating and visualizing complex scenes [Wang & Kaufman 1994]. In the near future, real-time visualization in the operating room using open-magnet MR scanners and fast volume rendering may become practical.

In this thesis we have provided appropriate algorithms for the rendering system. The next step is to create simple yet powerful user interfaces that harness the power of volume rendering.