I/O-Conscious Volume Rendering

Chuan-Kai Yang and Tzi-cker Chiueh

Department of Computer Science, State University of New York at Stony Brook, Stony Brook, NY 11794-4400, USA
emails: {ckyang, chiueh} @cs.sunysb.edu

Abstract. Most existing volume rendering algorithms assume that data sets are memory-resident and thus ignore the performance overhead of disk I/O. While this assumption may be true for high-performance graphics machines, it does not hold for most desktop personal workstations. To minimize the end-to-end volume rendering time, this work re-examines implementation strategies of the ray casting algorithm, taking into account both computation and I/O overheads. Specifically, we developed a data-driven execution model for ray casting that achieves the maximum overlap between rendering computation and disk I/O. Together with other performance optimizations, on a 300-MHz Pentium-II machine, without directional shading, our implementation is able to render a 128x128 grey-scale image from a 128x128x128 data set with an average end-to-end delay of 1 second, which is very close to the memory-resident rendering time. With a little modification, this work can also be extended to do out-of-core visualization as well.

1 Introduction

Despite the fact that volumetric data sets are inherently huge, most previous ray casting algorithms research reported performance numbers, assuming that data sets are entire memory-resident. This assumption is not valid when individual data sets are too large to fit into main memory (out-of-core rendering), or when users need to browse or explore a large number of data sets. Such assumptions tend not to hold especially on personal workstations, where volume visualization technology is gradually gaining grounds.

The motivation of this work is to develop a high-performance volume rendering system on commodity PCs without special hardware support, with a focus on reducing the end-to-end rendering delay, including the disk overhead of bringing the data sets in and out of the host memory. The key technique to minimize the performance impacts of disk I/O is to overlap disk operations with rendering computation so that the disk I/O time is masked as much as possible. To achieve this goal, a volumetric data set is decomposed into blocks, which are stored on disks and accessed as indivisible units. As data blocks are retrieved from disks, rendering computation on those blocks that are brought in earlier proceeds simultaneously. In this execution model, the minimum total rendering time for a disk-resident data set is the sum of the rendering time when the data set is entirely memory-resident, and the time required to fetch the first data block.

Surprisingly, the above overlapping execution model is difficult to get right in practice. This paper presents one such optimal execution model: data-driven block-based
volume rendering, which hides most of the disk I/O delay while at the same time ensures that a data block is completed exercised once it is brought into memory from the disk. The bottom-line result is that on a 300-MHz Pentium-II machine, without directional shading, this implementation strategy is able to complete the task of rendering a 128x128x128 data set into a 128x128 image in 1 second on the average, including the disk I/O time.

The rest of this paper is organized as follows. Section 2 reviews previous volume rendering work that paid attention to disk I/O issues. Section 3 describes the design dimensions of I/O-conscious volume rendering algorithms, and their associated performance tradeoffs. Section 4 proposes a simple extension of this work to do out-of-core visualization as well. Section 5 shows the results of a detailed performance evaluation of the prototype implementation, which is built on top of a Pentium-II machine running Linux. Section 6 concludes this paper with a summary of the major research results. Due to space limitation, we have omitted some details. Please refer to full paper at http://www.ecs1.cs.sunysb.edu/tr/TR89.pdf.

2 Related Work

The main focus of this work is to reduce the disk I/O performance overhead in volume rendering computation, particularly ray casting algorithms. Out-of-core rendering refers to the case where the rendering machine's physical memory can not hold the entire data set and thus need to perform disk I/O during the rendering process. Cox [4], [3] studied this problem by examining the performance impacts of the operating system interfaces on the disk I/O cost, as well as related file cache management issues. In contrast, our work attempts to use algorithm-specific prefetching to ensure that the data blocks could be brought in before they are needed. The proposed prefetching mechanism is closely tied with the rendering computation, and is completely algorithm-specific. This tightly integrated approach also sets itself apart from other more general-purpose disk prefetching research, as done in [6], [8], [9,1] and [7]. Another way to reduce the performance overhead due to disk I/O is to use compression to cut down the I/O traffic volume, as done in [10], [5], [2] and [11]. Our work assumes that the ray casting algorithm is more computation-intensive than I/O-intensive, and therefore spending additional decompression computation or restricting the data viewing scope to lower disk traffic is not considered a desirable tradeoff. Rather, we focus on how to mask the disk I/O delay.

3 I/O-Conscious Ray Casting Algorithm

3.1 Optimization for Memory-Resident Ray Casting Algorithm

To reduce the end-to-end volume rendering time, the performance of the ray casting algorithm when the data set is completely memory-resident should be optimized to the extent possible. We have added the following performance optimizations to arrive at a high-quality and high-performance ray caster, as the baseline case.