Image-Based Rendering of Surfaces from Volume Data

Baoquan Chen[†], Arie Kaufman[††], and Qingyu Tang[††]

[†] Computer Science & Engineering Department
University of Minnesota at Twin Cities
4-192 EE/CSci Building, 200 Union St. SE, Minneapolis, MN 55455
baoquan@cs.umn.edu
[††] Center for Visual Computing and Computer Science Department
State University of New York at Stony Brook
Stony Brook, NY 11794-4400
ari@cs.sunysb.edu

Abstract. We present an image-based rendering technique to accelerate rendering of surfaces from volume data. We cache the fully volume rendered image (called *keyview*) and use it to generate novel views without ray-casting every pixel. This is achieved by first constructing an underlying surface model of the volume and then texture mapping the keyview onto the geometry. When the novel view moves slightly away from the keyview, most of the original visible regions in the keyview are still visible in the novel view. Therefore, we only need to cast rays for pixels in the newly visible regions, which usually occupy only a small portion of the whole image, resulting in a substantial speedup. We have applied our technique to a virtual colonoscopy system and have obtained an interactive navigation speed through a 512^3 size patient colon. Our experiments demonstrate an average of an order of magnitude speedup over that of traditional volume rendering, compromising very little on image quality.

Keywords: image-based rendering, volume rendering.

1 INTRODUCTION

Major efforts have been dedicated to accelerating volume rendering, in both software and hardware approaches. One hardware approach is to leverage existing hardware, especially 3D texture mapping hardware, to accelerate volume rendering [3, 8, 29]. However, 3D texture mapping hardware is not scalable and does not support interactive classification. Another hardware approach is to utilize special-purpose hardware, such as VolumePro [24], for volume rendering. VolumePro can render a 256^3 volume at 30 frames per second. However, the performance of VolumePro is still challenged by large data sets. For example, for a 512^3 data, only 3-4 frames per second rendering performance can be theoretically obtained. Furthermore, the current VolumePro supports only parallel projection. Many applications, for example the virtual colonoscopy system in biomedical application, require perspective projection. Parallel projection of a straight tube appears as a ring, which prevents us from examining the interior walls of the colon. With some software design, VolumePro could be utilized to support perspective projection, but at great cost to the rendering speed as well as image quality [16]. A typical

K. Mueller et al. (eds.), *Volume Graphics 2001*

data size in a virtual colonoscopy system is 512^3; a straightforward implementation using VolumePro delivers a much lower frame rate than the interaction requirement when perspective rendering is supported.

The software approach includes early ray termination [19], distance encoding, presence acceleration [1, 6], ray coherence [14], and shear warping [17]. These methods still cannot meet the interactive speed requirement unless they are parallelized on powerful machines [17, 23].

Recently, image-based rendering techniques have been developed to model complex scenes using images or sprites instead of complex geometries, and to render directly from images. One can pre-compute a group of *keyviews* (or *reference images*) so that any novel view can be generated from one (or a subset) of these keyviews. This is similar to sampling the full plenoptic function [21], resulting in 3D, 4D or even 5D image sprites [12, 20]. These methods usually need a very dense view sampling (thousands of images), which requires a large amount of memory storage. Dally et al. [9] use a delta tree to represent all the keyviews.

Even though most of these methods were developed for accelerating polygon rendering, their concepts can be applied to accelerating volume rendering. Choi and Shin [5] have applied Dally et al.'s method to render the fully opaque surface of the volume using a quadtree structure instead. Their method supports only parallel projection. Even though tens of thousands of reference images are pre-computed, holes may still appear in the generated image. The nearest neighbor (zero-order) interpolation has been used for efficiency, but at the cost of degrading the image quality. Generally, a traditional IBR method is inappropriate for volume rendering because whenever the transfer function changes, it is required to regenerate the whole set of reference images, resulting in a huge amount of computation and storage. Gudmundsson and Randen [13] have proposed to utilize the coherence between neighboring views and to incrementally generate the novel view. This incremental approach avoids storing thousands of images. Similar to the above technique, it supports only parallel projection and the rendering of the fully opaque surface. In addition, it delivers a degraded image quality as only a zero-order interpolation is used; similarly, holes may appear in the generated image. These disadvantages have prevented these methods from a broad usage in volume rendering. Yagel et al. [30] have extended Gudmundsson and Randen's technique to support flexible volume rendering. Rather than using the incremental method to directly generate the final image, they use it to only generate the "C-buffer," storing the object-space coordinates of the first non-empty voxel visible from every pixel. They further use thus generated "C-buffer" to accelerate the ray-casting operation by "space-leaping" (skipping the empty space). Full rays are still required to cast into the data. Currently, space-leaping is considered a standard optimization technique for ray-casting, together with several other techniques, such as early ray termination. The technique to be discussed in this paper will be compared against the existing optimized ray-casting.

Most recently, more research has been conducted to further accelerate volume rendering using the image-based approach. Brady et al. [2] have proposed a two-phase approach. This method divides the viewing frustum into regions (or slabs) based on the Euclidean distance from the eyepoint. The algorithm then ray-casts each region separately. The images created from the regions are subsequently texture mapped onto