

Large-scale Shear Warp 3D Volume Rendering

Ho-Seong Kim¹ and Chang-Sung Jeong^{*}

¹School of Visual Information Processing, Korea University

^{*}School of Electrical Engineering, Korea University
Seoul, South Korea

[e-mail: rlaghtmd11@korea.ac.kr, csjeong@korea.ac.kr]

^{*}Corresponding author: Chang-sung Jeong

Abstract

This paper presents a parallel algorithm for large-scale shear warp 3D volume rendering on distributed computing environment. The algorithm achieves efficient memory consumption and speed up by parallelizing the run-length encoding for efficient volume rendering and direct accesses to the divided volume data. Specifically, our algorithm shows that distributed computing with multiple processors is very crucial for rendering very large data sets.

Keywords: Parallel processing, Shear warp volume rendering, volume data, distributed computing

1. Introduction

Nowadays, Rendering technology has been utilized in many fields such as medical field, game, Smart phone. The shear warp volume rendering technique was widely used. However, the process is complex and computation intensive work for large-scale volume rendering.

The shear warp volume rendering had studied in order to image quality and improvement of processing time such as volume memory, run-length-encoding method. However, these methods are not suitable for large-scale volume rendering.

This paper presents a parallel algorithm for large-scale shear warp 3D volume rendering on distributed computing environment. The algorithm achieves efficient memory consumption and speed up by parallelizing the run-length encoding for efficient volume rendering and direct accesses to the divided

volume data. Specifically, our algorithm shows that distributed computing with multiple processors is very crucial for rendering very large data sets.

The outline of our paper is as follows : In section 2, we describe related work about volume rendering. In section 3 and 4, we present two shear warp algorithms without compressed opaque voxel and with run-length-encoded volume respectively. Finally, we give a conclusion in section 5.

2. Related Work

One of the most volume rendering techniques is direct volume rendering, which calculates the color of screen pixels by processing voxels directly. Ray-casting[1] and splatting[2] are based on this method. However, direct volume rendering results are slow speed. The direct volume rendering is required traversing the entire volume data whenever the viewing

direction changes. The direct volume rendering method is limited in terms of speed if the size of volume data becomes large.

When cross sectional images of an object placed in 3D space are continuously shot at a fixed interval using CT or MRI, a set of 3D data representing the internal and external shapes of the object is created. Such 3D data are called volume data, and the pixel which composes the cross al images is called voxel. Volume rendering[2, 3] is a technique for for visualizing volume data on the screen. It can be classified into two basic techniques : surface -fitting volume rendering and direct volume rendering. The surface primitives are then displayed by means of APIs, such as OpenGL and DirectX.

In contrast to suface-fitting volume rendering, direct volume rendering calculates the colors of screen pixels by processing voxels directly.

The direct volume rendering is more widely used as it creates images with high-quality. Major research areas on direct volume rendering can be clssified into two fields : acceleration techniques for rendering speed, and rendering techniques for high quality image. The direct volume rendering is limited in terms of speed, if the size of volume data becomes large.

3. Shear Warp Algorithm without Compressed Opaque Voxel

We decrease memory consumption and preprocessing time for shear warp rendering algorithm as follws:

The volume data is divided into 3 pieces, and then each of data is computed on distributed computing environment. We do not create compressed opaque voxel from divided volume data for clear image, and access directly to the volume memory where the initially loaded volume data is stored in order to get voxels needed for bilinear-interpolation.

Therefore, we can get clear results very fast time with decrease in the memory consumption and preprocessing time.

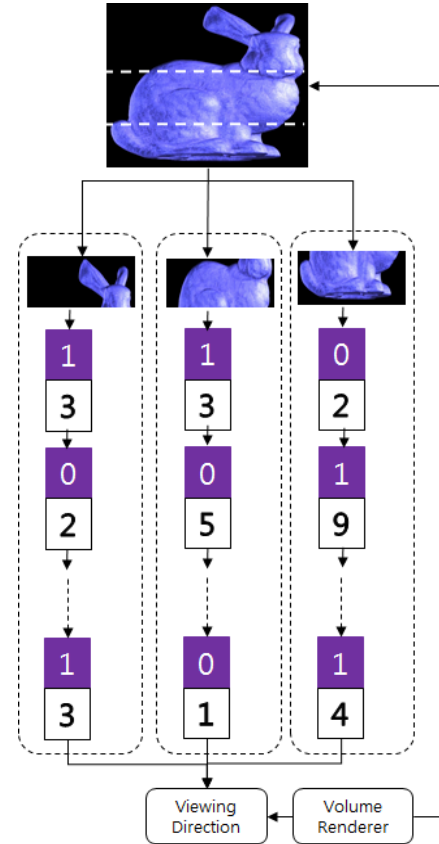


Fig. 1. Overview of shear warp algorithm with divided volume data

4. Shear warp algorithm with run length encoded volume data

In this section, we propose a parallel algorithm for increasing rendering speed by directly using perimetric pixels on distributed computing environment instead of using the bilinear-interpolation as preprocessing of shear-warp algorithm. When bilinear-interpolation is not used, it can achieves more efficient memory usage, and reduce the processing time.

Generally, shear-warp algorithm executes bilinear-interpolation as a preprocessing. However, it takes long time to get results. Instead, we divide volume data, and create run-length-encoded volume for each axis x, y, z. The volume data can be divided into N sub-volume(N : Maximum number of distributed computing element) shows in Fig. 2.

After the process, we create image for each sub-volume by using run-length-encoded volume, and then merge them. The merged images are shown to the client by using non photorealistic rendering.

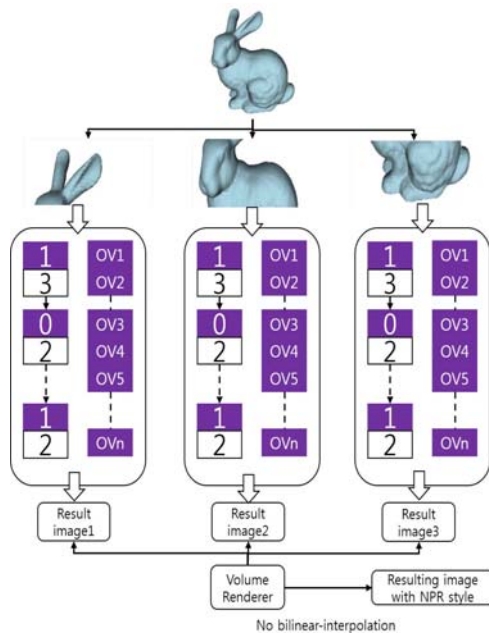


Fig. 2. Overview of shear warp algorithm with one run-length-encoded volume on parallel computing environment

4. Conclusions

In this paper we have presented parallel algorithms for large scale shear warp 3D volume rendering on distributed computing environment. By dividing the volume data and processing on parallel computing environment, we can reduce the loading time, while reducing preprocessing time and memory consumption by using each improved shear-warp algorithm.

Our improved shear-warp algorithm is expected to be easily parallelized and suitable on grid computing and cloud computing.

References

- [1] K.-Y. Choi, S.-U. Jo, and C.-S. Jeong, "New Optimization Techniques for Shear-Warp Volume Rendering," in *Future Information Technology*, ed: Springer, 2011, pp. 415-422.
- [2] Drebin R, Carpenter L and Hanrahan P., "Volume Rendering", *Computer Graphics 1988*; 22, pp.65-74.
- [3] Levoy M., "Display of surfaces from volume data", *IEEE Computer Graphics & Applications 1988*, 8(3), pp.29-37.