Joint EUROGRAPHICS - IEEE TCVG Symposium on Visualization (2002)
D. Ebert, P. Brunet, I. Navazo (Editors)

Parallel and Out-of-core View-dependent Isocontour Visualization Using Random Data Distribution

Xiaoyu Zhang, Chandrajit Bajaj, Vijaya Ramachandran

Department of Computer Sciences, The University of Texas at Austin, Austin, TX 78712

Abstract

In this paper we describe a parallel and out-of-core view-dependent isocontour visualization algorithm that efficiently extracts and renders the visible portions of an isosurface from large datasets. The algorithm first creates an occlusion map using ray-casting and nearest neighbors. With the occlusion map constructed, the visible portion of the isosurface is extracted and rendered. All steps are in a single pass with minimal communication overhead. The overall workload is well balanced among parallel processors using random data distribution. Volumetric datasets are statically partitioned onto the local disks of each processor and loaded only when necessary. This out-of-core feature allows it to handle scalably large datasets. We additionally demonstrate significant speedup of the view-dependent isocontour visualization on a commodity off-the-shelf PC cluster.

1. Introduction

Today tomographic imaging and computer simulations are increasingly generating large datasets that are to be effectively visualized for better understanding of the underlying scientific information. Isocontour visualization is a popular interactive and exploratory visualization technique used to determine and browse regions of interest within volumetric imaging data, and validate the results of computer simulations. Interactive isocontour visualization extracts multiple 2-dimensional surfaces satisfying $F(\mathbf{x}) = const$ from a given scalar field $F(\mathbf{x})$, $\mathbf{x} \in \mathbf{R}^3$, and renders them at interactive frame rates ($30HZ$).

Related Work: As the size of the input data increases, isocontouring algorithms necessarily need to be executed out-of-core and/or on parallel machines for both efficiency and data accessibility. Hansen and Hinker[16] describe parallel methods for isosurface extraction on SIMD machines. Ellsiepen [11] describes a parallel isosurfacing method for FEM data by dynamically distributing working blocks to a number of connected workstations. Shen et al. [31] implement a parallel algorithm by partitioning load in the span space. Parker et al. [27] present a parallel isosurface rendering algorithm using ray-tracing. Chiang and Silva [6, 8] give an implementation of out-of-core isocontouring using the I/O optimal external interval tree on a single processor. Bajaj et al. [2] use range partition to reduce the size of data that are loaded for given isocontour queries and balance the load within a range partition. More recently Zhang et al. [35] propose a scalable isosurface visualization framework for massive datasets on commodity off-the-shelf clusters by combining the parallel and out-of-core isocontouring techniques. Chiang et al. [7] also try to combine parallel and out-of-core techniques for isosurface and volume rendering for unstructured grids.

An isocontour visualization is not complete without isosurfaces being rendered and displayed to the user. Rendering is a critical and indispensable part of visualization. In the case of isosurface rendering, the triangles of an isosurfaces are projected from the $3D$ object space to the $2D$ image space by graphics hardware. Efficient rendering is of special importance when the user wants to observe an isosurface from different viewpoints after the surface is extracted. Large isosurfaces extracted from large datasets often need to be rendered with parallel graphics pipes for interactivity.

Udeshi and Hansen [33] employ the multi-pipes of a SGI Onyx2 reality monster to render large isosurfaces in a sort-last fashion [25]. They use a binary-swap method [24] to efficiently composite the images in $log_2(P)$ steps, where P is the number of rendering processors. With the fast advances of PC graphics hardware, it is now possible for a single PC graphics card to render millions of triangles at the same or even better rate than some custom graphics boards. Combining the power of these commodity graphics hardware will