Massive Dataset Visualization

Aiichiro Nakano

Collaboratory for Advanced Computing & Simulations
Dept. of Computer Science, Dept. of Physics & Astronomy,
Dept. of Chemical Engineering & Materials Science
Department of Biological Sciences
University of Southern California

Email: anakano@usc.edu
Immersive & Interactive Visualization

Billion-atom walkthrough

Parallel & distributed Atomviewer

Graphics server

Reduced data

User position

PC cluster

ImmersaDesk

Space-time multiresolution algorithm
Wavelet load-balancing
Fractal data compression

ImmersaDesk R2 virtual environment

real-time walkthrough

Octree visibility culling
Preprocessing
Multiresolution rendering

Fiber network

Linux cluster

GRID of supercomputers

Parallel & distributed Atomsviewer
Locality in Data Compression

Massive data transfer via wide area network: 75GB/step of data for 1.5 billion-atom MD! → Compressed software pipeline

Scalable encoding:
• Store relative positions on spacefilling curve: $O(N \log N) \rightarrow O(N)$

Result:
• Data size, 50 Bytes/atom → 6 Bytes/atom
Data Compression for Scalable I/O

Challenge: Massive data transfer via OC-3 (155 Mbps)
75 GB/frame of data for a 1.5-billion-atom MD!

Scalable encoding:
• Spacefilling curve based on octree index

3D → list map preserves spatial proximity
Spacefilling-Curve Data Compression

Algorithm:

1. Sort particles along the spacefilling curve
2. Store relative positions: $O(N \log N) \rightarrow O(N)$
   - Adaptive variable-length encoding to handle outliers
   - User-controlled error bound

Result:

• An order-of-magnitude reduction of I/O size: 50 $\rightarrow$ 6 Bytes/atom
Data Locality in Visualization

- Octree-based fast view-frustum culling
- Probabilistic occlusion culling
- Parallel/distributed processing

• Interactive visualization of a billion-atom dataset in immersive environment
Hierarchical Abstraction

- Larger clusters for longer distances
- Recursively subdivide the 3D space to form an octree
Visibility Culling

View frustum culling

Viewpoint

Remove atoms outside of the view frustum

Higher Depth

Occlusion culling

Remove atoms hidden by other atoms
Octree-based View-Frustum Culling

- Use the octree data structure to efficiently select only visible atoms

- Complexity
  Insertion into octree: $O(N)$
  Data extraction: $O(\log N)$
Probabilistic Occlusion Culling

- Remove atoms that are occluded by other atoms closer to the viewer.
- Regions farther away from the viewer is more likely to be occluded than one in front of the viewer.
- Draw fewer atoms per region as the distance of a region from the viewer increases: visibility value $v(x)$ for region $x$.
- Recurrence along the view line:
  $$v_x = \begin{cases} 
  1 & x = 0 \\
  f(D_x, v_x') & \text{else} \\
  \end{cases}$$
  where $D_x = \text{density of region}$.
- Run-time adaptation:
  $$v'_x = f(\text{user speed}) \times v_x$$
Results of Probabilistic Occlusion Culling

Original

Probabilistic

Difference

68% fewer objects

3× frame rate
Multiresolution Culling & Rendering

- **Per-octree node operations:**
  - Frustum culling
  - Probabilistic occlusion culling

- **Per-atom operations**
  - Multiple levels-of-detail
  - Occlusion culling (per-object, per-octree node)

Outflow pathways of optic nerves from the retina of a rabbit eye
(Experimental data by C. Burgoyne & R. Beuerman, LSU Eye Center)
Distributed Architecture

- DISTRIBUTED ARCHITECTURE
- OCTREE BASED DATA EXTRACTION MODULE
- PROBABALISTIC OCCLUSION CULLING MODULE
- RENDERING & VISUALIZATION MODULE
- RENDERING SYSTEM
- PER-ATOM OCCLUDER
- USER POSITION
- NEAR COMPLETE LIST OF VIEWABLE ATOMS
- OCTREE BASED DATA EXTRACTION MODULE
- PROBABALISTIC OCCLUSION CULLING MODULE
- REGIONS OF INTEREST
- TCP/IP SOCKET
Parallel Octree Extraction

- Individual copies of the octree with each node
- Spherical extraction by the use of shells of equal volume
- Load balancing due to the equal use of each processor for extraction
Latency Hiding

- Individual modules are multithreaded to reduce network or module latency
- Minimize latency due to inter-modular dependencies by overlapping the inter-module communication and module computation

- Instantaneously trained neural network (CC4 [Tang & Kak, CSSP’98]) predicts the user’s next position [Liu et al., PDPTA’02]
Parallel & Distributed Atomviewer

Real-time walkthrough for a billion atoms on an SGI Onyx2 (2 × MIPS R10K, 4GB RAM) connected to a PC cluster (4 × 800MHz P3)

IEEE Virtual Reality Best Paper
Parallel Rendering

ParaViz: A Spatially Decomposed Parallel Visualization Algorithm Using Hierarchical Visibility Ordering

Cheng Zhang¹, Scott Callaghan², Thomas Jordan², Rajiv K. Kalia¹,
Aiichiro Nakano¹*, Priya Vashishta¹

- Parallel rendering of spatially distributed data: hybrid sort-first/sort-last
- Scalable depth buffer by domain-level distributed visibility ordering
- On-the-fly visualization of parallel simulation without data migration
- Parallel efficiency 0.98 on 1,024 processors for 16.8 million-atom molecular-dynamics simulation
Atomsvviewer Code

- **Programming language**
  - C++

- **Graphics**
  - OpenGL
  - CAVE Library (optional)

- **Platforms**
  - Windows
  - Macintosh OS X
  - SGI Irix
Scalable and portable visualization of large atomistic datasets

Ashish Sharma *, Rajiv K. Kalia, Aiichiro Nakano, Priya Vashishta

Collaboratory for Advanced Computing and Simulations, Department of Computer Science, Department of Physics & Astronomy, Department of Materials Science & Engineering, University of Southern California, Los Angeles, CA 90089-0424, USA

Received 15 June 2004; accepted 8 July 2004
Available online 16 September 2004

Abstract

A scalable and portable code named Atomsvviewer has been developed to interactively visualize a large atomistic dataset consisting of up to a billion atoms. The code uses a hierarchical view frustum-culling algorithm based on the octree data structure to efficiently remove atoms outside of the user's field-of-view. Probabilistic and depth-based occlusion-culling algorithms then select atoms, which have a high probability of being visible. Finally, a multiresolution algorithm is used to render the selected subset of visible atoms at varying levels of detail. Atomsvviewer is written in C++ and OpenGL, and it has been tested on a number of architectures including Windows, Macintosh, and SGI. Atomsvviewer has been used to visualize tens of millions of atoms on a standard desktop computer and, in its parallel version, up to a billion atoms.

Program summary

Title of program: Atomsvviewer
Catalogue identifier: ADUM
Program summary URL: http://cpc.cs.qub.ac.uk/samaries/ADUM
Program obtainable from: CPC Program Library, Queen's University of Belfast, N. Ireland
Computer for which the program is designed and others on which it has been tested: 2.4 GHz Pentium 4/Xeon processor, professional graphics card; Apple G4 (867 MHz)/G5, professional graphics card
Operating systems under which the program has been tested: Windows 2000/XP, Mac OS 10.2/10.3, SGI IRIX 6.5
Programming languages used: C++, C and OpenGL
Memory required to execute with typical data: 1 gigabyte of RAM
High-speed storage required: 90 gigabytes
No. of lines in the distributed program including test data, etc.: 550241
No. of bytes in the distributed program including test data, etc.: 6258245
Number of bits in a word: Arbitrary

* This paper and its associated computer program are available via the Computer Physics Communications homepage on ScienceDirect (http://www.sciencedirect.com/science/journal/00104655).
* Corresponding author.
E-mail address: nakano@usc.edu (A. Nakano).

http://www.cpc.cs.qub.ac.uk/cpc