A Survey of Parallel Volume Rendering

Claudio T. Silva
Information Visualization Research Department
AT&T Labs-Research
http://www.research.att.com/~csilva

Overview

• Volume Rendering Introduction
• Regular-Grid Techniques
• Irregular-Grid Techniques
• Rendering Systems
• Final Remarks

Introduction to Volume Rendering

Optical Models

$$I(s+x) = I(s) + g(s)\Delta x - I(s)\Omega\Delta s$$
$$I(s) = \int_0^x g(x)\Delta y \, dy$$

Compositing

Back-to-front

$$I_{11} = \alpha_1 I_1 + (1-\alpha_1)\alpha_2 I_2$$
$$I_{12} = \alpha_2 I_2 + (1-\alpha_1)\alpha_3 I_3$$
$$\alpha_{11} = \alpha_1 + (1-\alpha_1)\alpha_2$$
$$\alpha_{12} = \alpha_2 + (1-\alpha_1)\alpha_3$$

Back-to-front

$$I_{11} = \alpha_1 I_1 + (1-\alpha_1)\alpha_2 I_2$$
$$I_{12} = \alpha_2 I_2 + (1-\alpha_1)\alpha_3 I_3 + (1-\alpha_2)(1-\alpha_3)\alpha_4 I_4$$
$$\alpha_{11} = \alpha_1 + (1-\alpha_1)\alpha_2$$
$$\alpha_{12} = \alpha_2 + (1-\alpha_1)\alpha_3 + (1-\alpha_2)(1-\alpha_3)\alpha_4$$
Over-operator

\[ \alpha_i \oplus \alpha_i = \alpha_i + (1 - \alpha_i) \alpha_i, \]

and then

\[ C_{i,j} = C_j \oplus C_j \oplus C_i \]

\[ C_{i,j} = \alpha_j \oplus \alpha_i + \oplus \alpha_i \]

One more thing...

\[ (\alpha_i \oplus \alpha_j) + \oplus \alpha_i = \alpha_j \oplus (\alpha_i + \oplus \alpha_i) \]

Regular-Grid Volume Rendering

- Shared-Memory Ray Casting
- Distributed-Memory Ray Casting
- Shared-Memory Shear-Warp

Parallel Ray Casting

Idea: Assign rays to separate processors

Issues:

- Load Balancing

Parallel Ray Casting

Idea: Assign rays to separate processors

Issues:

- Load Balancing
Load Balancing

- Static versus Dynamic
- Data locality and cache coherence
- Scalability

Nieh and Levoy 92

- Ray casting on the Stanford DASH
- Screen is sub-divided into titles
- Titles assigned dynamically
- Maximum speedup 40 (out of 48 processors)
- Best frame rate: 3 Hz (speedup of 33 -- adap. rendering)

Parker, Shirley, Livnat, Hansen, Sloan 98

- Parallel ray casting of iso-surfaces
- Tile-based screen load balancing
- Care to avoid cache trashing
- Dynamic scheme with variable chunks
- For 900M dataset, speedups of 102, 113 on 128 proc. SGI

Other Work

- Palmer-et-al, PRS ‘97
- Law and Yagel, Graphics Interface ‘96
- Mackerras and Corrie, PRS ‘93

Regular-Grid Volume Rendering

- Shared-Memory Ray Casting
- Distributed-Memory Ray Casting
- Shared-Memory Shear-Warp
Sorting problem

(1) Sampling the volume
(2) Compositing the samples

Important: Compositing is associative

Issues

- Data replication
- Communication overhead
- Load balancing

Resampling

(1) Align the rays
(2) Then composite

Works well on SIMD machines
see [Schroeder and Salem91]

Ma, Painter, Hansen, Krogh 93

(1) The data set partitioning
based on a K-d tree
(2) The image compositing
based on a recursive
image-subdivision scheme

Ma, Painter, Hansen, Krogh 93

(1) The data set partitioning
based on a K-d tree
(2) The image compositing
based on a recursive
image-subdivision scheme
(1) The data set partitioning based on a K-d tree
(2) The image compositing based on a recursive image-subdivision scheme

Each processor renders each piece independently which can be composited together since the data is partitioned into convex, non-overlapping regions of 3-space.

- Compositing uses all processors at all times
- $O(\log p)$ compositing
- Renders “empty regions”
  - See Karia 94 for better load balancing

Any work being performed by processors associated with empty blocks is wasted.
Load balancing -- Silva 96

Content-Based Load Balancing (BSP-tree based)

- Always subdivide along the largest "axis"
- Always subdivide the region with the plane that cuts the number of full sub-cubes in half

Silva and Kaufman 94

- P0 → P1 → ... → Pn

(1) Subdivide volume into slabs (2) Each processor computes an image request and sends its sub-image to the next (3) Last processor will have the final image

Alternative Arch.

Ramakrishnan and Silva 99

Optimal Processor Allocation for BSP-tree compositing

- Rendering cluster
- Sub-images
- Compositing cluster

Other Techniques

- Wittenbrink and Somani, PRS 93
- Hsu, PRS 93
- Westermann, IPPS 95
- ...

Regular-Grid Volume Rendering

- Shared-Memory Ray Casting
- Distributed-Memory Ray Casting
- Shared-Memory Shear-Warp
Shear-Warp Volume Rendering

Lacroute 1996 -- Similar to Nieh and Levoy, but using groups of scanlines instead of square tiles

Irregular-Grid Volume Rendering Ma and Crockett 97

- Distributed-Memory Technique
  - Round-robin of the cell primitives
- Two phases:
  - Resampling
  - Compositing
- Key optimization:
  - Use k-d tree sub-partitioning to bound the size of ray segments

Irregular-Grid Volume Rendering Ma and Crockett 97

Viewing direction

Irregular-Grid Volume Rendering Ma and Crockett 97

Viewing direction

Irregular-Grid Volume Rendering Ma and Crockett 97

Viewing direction

Parallel Volume Rendering Systems

- VFleet
- PVR
VFleet

- Developed by Joel Welling at PSC
- Support distributed memory machines
- GUI needs Motif/UNIX
- Supports multiple volumes
- Flexible transfer function
- Source and binaries available from Welling’s web site

PVR

Silva, Kaufman, Pavlakos 96

System components

- pvrsh
- client.edu
- pvrd.sunmos
- pvr.mpi
- vista9.watson.ibm.com
- pvrd.sandia.gov
- caster.sandia.gov
- pvrd_mpi
- caster_mpi

Simple interactive

```bash
pvr_session $s
$p set -partsz 1,1,8
$p open2 vista9
$p set -partsz 1,1,8 /* Need to configure */
$p set -dataset neghip.slc -cluster 4 -group 0,0
$p render rotation 0,1,0 0,10:30
$p close2
```

PVR Summary

- Runs under Intel NX and MPI
- Tcl/Tk-based command language
- Tk/Tk GUI front-end
- Supports image-space, object-space, and time-space parallelism
- Volume Rendering (currently only regular grids)
- Polygon Rendering (using Mesa)
- General compositing model and for sort-last architectures
- Over 20K lines of C code (not including GUI)

Challenges/Opportunities Ahead

- Hardware-Assisted Rendering Techniques
- Large- and high-resolution displays
- Out-of-core rendering
- Time-critical rendering