

Volume Reconstruction and Parallel Rendering Algorithms: A Comparative Analysis

by
Ulrich Neumann

A dissertation submitted to the faculty of The University of North Carolina at Chapel Hill in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Computer Science

Chapel Hill
1993

Approved by:

Advisor: Henry Fuchs

Reader: Turner Whitted

Reader: Stephen M. Pizer

Reader: Jan F. Prins

© 1993
Ulrich Neumann
ALL RIGHTS RESERVED

ULRICH NEUMANN. Volume Reconstruction and Parallel Rendering Algorithms: A Comparative Analysis. (Under the direction of Henry Fuchs.)

Abstract

This work focuses on two issues of concern to designers and implementers of volume-rendering applications - finding the most efficient rendering method that provides the best image possible, and efficiently parallelizing the computation on multicomputers to render images as quickly as possible.

Three volume rendering methods: ray casting, splatting, and volume shearing, are compared with respect to their reconstruction accuracy and computational expense. The computational expense of the rendering methods is modeled and measured on several workstations. The most-efficient rendering method of the three is found to be splatting. Three reconstruction-filter kernels are evaluated for their accuracy. Two error measurement methods are used. The first is image-based and uses a heuristic metric for measuring the difference between rendered images and reference images. The second method is analytical and uses a scale-space measure of feature size to compute an error bound as a function of feature size and sampling density. Of the three filter kernels tested, the separable cubic filter is found to produce the most-accurate reconstruction of the volume function.

Parallel volume-rendering algorithms may be classified and described by the partitioning of tasks and data within the system. A taxonomy of algorithms is presented and the members are analyzed in terms of their communication requirements. Three optimal algorithm-classes are revealed: image partitions with both static and dynamic data distributions, and object partitions with contiguous dynamic block data distributions. Through analysis and experimental tests, a 2D mesh network-topology is shown to be sufficient for scalable performance with an *object-partition* algorithm when the image size is kept constant. Furthermore, the network-channel bandwidth-requirement actually decreases as the problem is scaled to a larger system and volume data size.

Implementations on Pixel-Planes 5 and the Touchstone Delta demonstrate and verify the scalability of object partitions. As part of these implementations, a new load balancing approach is demonstrated for object-partition algorithms.

Acknowledgments

Many thanks to my committee members and especially to Henry Fuchs for acting as my advisor and supporting me through this effort. Their suggestions were helpful in raising the quality of this work in many ways. Any remaining flaws are certainly my responsibility.

Professor Steve Taylor and Mike Palmer at Caltech supported my efforts by acting as hosts for my visit to Caltech and supplying me with information and access to the Touchstone Delta. David Ellsworth, a kindred spirit at UNC who also visited Caltech at the same time, was helpful during many discussions and fruitful late-night coding sessions (when the Delta was up and available). A special thanks goes to Terry Yoo who coordinated the Caltech visit as the UNC site-coordinator of the Graphics and Visualization Science and Technology Center in addition to helping me with countless other details.

Andre State, Qin Fang, and Tim Cullip are fellow-volume renderers who helped keep my thinking straight by participating in numerous discussions about volume rendering and parallel algorithms. Andre State also deserves special thanks for his help with the VVEVOL tests. Tim Cullip must be recognized and thanked for continuously raising the standard with respect to parallel-rendering rates.

Last, but not by any means least, I thank my wife Patricia. Over the years she has mastered the art of balancing encouragement and occasional whip-cracking. Her strength and faith always provided support along this journey. This work is dedicated to her patience and perseverance.

Financial support for this work has come from the Pixel-Planes 5 and VistaNET grants.

Table of Contents

1. Introduction	1
1.1. Volume Rendering Model	2
1.2. Volume Rendering Algorithms	4
1.3. Multicomputer Architectures	6
1.4. Parallel Algorithms	8
1.5. Thesis and Contributions	10
2. Previous Work	12
2.1. Uniprocessor Methods	12
2.2. Parallel Algorithms	19
2.3. Reconstruction Methods	23
3. Image-Based Reconstruction Error Comparison	26
3.1. Feature Scale	26
3.2. Test Data	27
3.3. Image Comparison Metric	31
3.4. Rendering Methods	32
3.5. Experimental Results	33
4. Analytical Reconstruction-Error Comparison	47
4.1. Pyramid Filter	47
4.2. Gaussian Filter	50
4.3. Separable Cubic Filter	54
4.4. Filter Comparison	56
5. Rendering Cost Comparison	60
5.1. Ray Casting with a Pyramid Filter	61
5.2. Splatting with a Gaussian Filter	63
5.3. Volume Shearing with a Separable Cubic Filter	65
5.4. Comparison and Discussion	66
6. Parallel Volume-Rendering Algorithms	70
6.1. Taxonomy	70
6.2. Image Partitions	72
6.2. Object Partitions	77

7. Parallel Algorithm Performance	80
7.1. Network Performance Model	80
7.2. Image-Partition Redistribution Costs	83
7.3. Object-Partition Redistribution Costs	90
8. Parallel Implementations	95
8.1. System Overview	95
8.2. Mesh Redistribution-Time	96
8.3. Touchstone Delta Implementation	97
8.4. Pixel-Planes 5 Implementation	100
9. Summary, Conclusions, and Future Work	102
9.1. Summary and Conclusions	102
9.2. Future Work	103
10. References	105

List of Figures

1.1	Embedded image and object lattices	5
2.1	Ray casting	13
2.2	Splat kernels	16
2.3	Volume shearing	18
2.4	Signal processing in volume rendering	24
3.1	Multiscale operator response function	27
3.2	<i>Points</i> test data generator coordinates	28
3.3	<i>Mixed</i> test data generator coordinates	29
3.4	Test data sets viewed down the k-axis	29
3.5	Prealiasing error $P(\sigma)$	30
3.6	Specifying a view (adapted from Rogers and Adams pp. 56)	32
3.7	Reference images	33
3.8	Ray casting with pyramid filter reconstruction	34
3.9	Ray casting with pyramid filter reconstruction	35
3.10	Inner Product of references with test images made with pyramid filter reconstruction. Horizontal axis is ray z-step size (data taken for step size = 1, 2, 4, 6, and 8)	36
3.11	Reference animation of <i>points</i>	37
3.12	<i>Points</i> animation using pyramid filter reconstruction	37
3.13	Splatting with Gaussian filter reconstruction	38
3.14	Splatting with Gaussian filter reconstruction	39
3.15	Inner Product of references with test images made with a Gaussian filter Horizontal axis is Gaussian kernel σ (data taken for $\sigma = 0.4, 0.5, 0.6, 0.7, 0.8,$ and 0.9)	40
3.16	<i>Points</i> animation rendered by splatting with Gaussian filter $\sigma = 0.5$	41
3.17	Volume shearing with separable cubic filter reconstruction	42
3.18	Volume shearing with separable cubic filter reconstruction	43
3.19	Inner Product of references with test images made with separable cubic filter. Horizontal axis is number of pixels per sample points (data taken at $1 \times 1, 2 \times 2,$ and 4×4)	44
3.20	Animation sequence made with Catmull-Rom cubic reconstruction filter	45
3.21	Inner product comparison	45

4.1	Linear reconstruction of 1D Gaussian	47
4.2	$G(x)$, $R(x)$, and $Err(x)$ as a function of x_0	48
4.3	Pyramid filter error as a function of feature size	49
4.4	$G(x)$, $R(x)$, and $Err(x)$ with ϵ as a parameter	51
4.5	$NerrID$ for Gaussian kernel $\upsilon = 0.45, 0.5, 0.55,$ and 0.6	52
4.6	Gaussian filter error as a function of feature size	52
4.7	Normalized peak-to-peak ripple amplitude	53
4.8	$G(x)$, $R(x)$, and $Err(x)$ as a function of ϵ	55
4.9	Cubic filter error as a function of feature size	56
4.10	$Nerr(\sigma)$ for one, two, and three dimension filters	57
4.11	Isosurface renderings of <i>mixed</i> data	58
4.12	Isosurface renderings CT data	59
5.1	Rendering times for $64 \times 64 \times 64$ image lattice	67
5.2	Rendering times for $128 \times 128 \times 64$ image lattice	67
6.1	Full taxonomy of parallel volume-rendering algorithms	71
6.2	Load balancing with dynamic contiguous slabs	73
6.3	Dynamic contiguous shaft load balancing on scan lines	73
6.4	Image-partition task distributions	74
6.5	Image-partition data distributions	76
6.6	Image-partition options	76
6.7	Object-partition task options	78
6.8	Object-partition image options	79
6.9	Object-partition options	79
7.1	Average latency vs. normalized throughput (adapted from [Ngai89])	81
7.2	Redistribution with slab image-lattice distribution	86
7.3	Redistribution with shaft image-lattice distribution	86
7.4	Redistribution with block image-lattice distribution	87
7.5	Redistribution with slab object-lattice distribution	90
7.6	Redistribution with shaft object-lattice distribution	91
7.7	Redistribution with block object-lattice distribution	91
8.1	Redistribution times measured on the Touchstone Delta (in seconds)	97
8.2	Redistribution sizes for test on the Touchstone Delta	97
8.3	Isosurface rendering of <i>mixed</i> data set	98

List of Abbreviations

A	Alpha (or Opacity)
ALU	Arithmetic Logic Unit
B	Blue
CAT	Computer-Aided Tomography
G	Green
I/O	Input / Output
K	$2^{10} = 1024$
M	$2^{20} = 1048576$
MIMD	Multiple Instruction Multiple Data
MRI	Magnetic Resonance Imaging
R	Red
RISC	Reduced Instruction Set Computer
SIMD	Single Instruction Multiple Data
VLSI	Very Large Scale Integration
1D	One-Dimensional
2D	Two-Dimensional
3D	Three-Dimensional