FAST VOLUME RENDERING USING A SHEAR-WARP FACTORIZATION OF THE VIEWING TRANSFORMATION

Philippe G. Lacroute

Technical Report: CSL-TR-95-678

September 1995

This research has been supported by ARPA/ONR under contracts N00039-91-C-0138 and 175-6212-1, NSF under contract CCR-9157767, and grants from Software Publishing Corp., SoftImage, and the sponsoring companies of the Stanford Center for Integrated Systems.

FAST VOLUME RENDERING USING A SHEAR-WARP FACTORIZATION OF THE VIEWING TRANSFORMATION

Philippe G. Lacroute

Technical Report: CSL-TR-95-678

September 1995

Computer Systems Laboratory
Departments of Electrical Engineering and Computer Science
Stanford University
Stanford, CA 94305-4055
e-mail: pubs@shasta.stanford.edu

Abstract

Volume rendering is a technique for visualizing 3D arrays of sampled data. It has applications in areas such as medical imaging and scientific visualization, but its use has been limited by its high computational expense. Early implementations of volume rendering used brute-force techniques that require on the order of 100 seconds to render typical data sets on a workstation. Algorithms with optimizations that exploit coherence in the data have reduced rendering times to the range of ten seconds but are still not fast enough for interactive visualization applications. In this thesis we present a family of volume rendering algorithms that reduces rendering times to one second.

First we present a scanline-order volume rendering algorithm that exploits coherence in both the volume data and the image. We show that scanline-order algorithms are fundamentally more efficient than commonly-used ray casting algorithms because the latter must perform analytic geometry calculations (e.g. intersecting rays with axis-aligned boxes). The new scanline-order algorithm simply streams through the volume and the image in storage order. We describe variants of the algorithm for both parallel and perspective projections and a multiprocessor implementation that achieves frame rates of over 10 Hz.

Second we present a solution to a limitation of existing volume rendering algorithms that use coherence accelerations: they require an expensive preprocessing step every time the volume is classified (i.e. when opacities are assigned to the samples), thereby limiting the usefulness of the algorithms for interactive applications. We introduce a data structure for encoding spatial coherence in unclassified volumes. When combined with our rendering algorithm this data structure allows us to build a fully-interactive volume visualization system.

Key Words and Phrases: Volume Rendering, Scientific Visualization, Medical Imaging, Software Algorithms, Parallel Algorithms.

Acknowledgements

Many people have contributed to this work and provided support during my years at Stanford. First of all, I would like to express my sincerest gratitude to my advisor, Marc Levoy, who sparked my interest in computer graphics and provided encouragement at key times. Marc always stays involved with his students, whether during marathon brain storming sessions or on canoeing expeditions in the swamps of Florida. I would also like to thank Pat Hanrahan and Sandy Napel for serving on my reading committee and for their interest in my work. Their comments greatly improved this manuscript. I am also indebted to Mark Horowitz who provided guidance during my first few years at Stanford and who helped shape my interest in computer systems.

Next, I would like to thank my colleagues in the Stanford Graphics Group for providing a stimulating and fun research environment. In particular, I thank Brian Curless, Reid "Ankles" Gershbein, Craig Kolb, Matt Pharr, and especially Eric Veach for their comments on this manuscript. Maneesh Agrawala contributed to the implementation of the parallel volume renderer described in Chapter 6.

I would also like to thank the members of the DASH, FLASH and TORCH groups. In particular, J.P. Singh helped me to make sense of my parallel performance results, and Margaret Martonosi, Steve Woo, Mark Heinrich, and Dave Nakahira answered my countless questions about their cache simulators and the DASH multiprocessor.

Many thanks to Charlie Orgish, the only system administrator I know who lets the students mis-administer the machines and then is always on call to fix the problems. Thanks also to Sarah Beizer for cheerfully dealing with the University bureaucracy.

Silicon Graphics, Inc. provided time on the Challenge multiprocessor for the performance results in Chapter 6. The volume data sets used in this work were provided by Pat

Hanrahan, Sandy Napel, Siemens Medical Systems, Inc., and the North Carolina Memorial Hospital. My graduate work was supported by ARPA/ONR under contracts N00039-91-C-0138 and 175-6212-1, NSF under contract CCR-9157767, and grants from Software Publishing Corp., SoftImage, and the sponsoring companies of the Stanford Center for Integrated Systems.

Finally, I would like to thank my parents, Bernard and Ronni Lacroute, who have provided love, encouragement, and home-cooked meals whenever my fridge was empty. I dedicate this dissertation to them.

Contents

Acknowledgements **iii**

1 Introduction **1**

1.1 Volume Rendering . . . 3

1.1.1 The Volume Rendering Equation . . . 3

1.1.2 The Volumetric Compositing Approximation . . . 8

1.1.3 Data Representation and Sampling . . . 11

1.1.4 The Visualization Process . . . 13

1.2 A New Family of Fast Volume Rendering Algorithms . . . 15

1.3 Organization . . . 16

2 Prior Work **18**

2.1 Volume Rendering Algorithms . . . 18

2.1.1 Ray Casting . . . 19

2.1.2 Splatting . . . 20

2.1.3 Cell Projection . . . 21

2.1.4 Multipass Resampling . . . 22

2.2 Acceleration Techniques . . . 24

2.2.1 Spatial Data Structures . . . 24

2.2.2 Early Ray Termination . . . 26

2.3 Chapter Summary . . . 28

3 The Shear-Warp Factorization **29**

3.1 An Overview of the Factorization . . . 30

3.2 The Affine Factorization . . . 33
3.3 The Perspective Factorization . . . 35
3.4 Properties of the Factorization . . . 39
3.5 Existing Shear-Warp Algorithms . . . 40
3.6 Chapter Summary . . . 43

4 Three Fast Volume Rendering Algorithms **44**
4.1 Parallel Projection Rendering Algorithm . . . 44
4.1.1 Overview of the Parallel Projection Algorithm . . . 44
4.1.2 The Run-Length Encoded Volume . . . 47
4.1.3 The Run-Length Encoded Image . . . 49
4.1.4 Resampling the Volume . . . 53
4.1.5 Warping the Intermediate Image . . . 54
4.1.6 Opacity Correction . . . 54
4.1.7 Implementation of the Parallel Projection Algorithm . . . 56
4.2 Perspective Projection Rendering Algorithm . . . 60
4.2.1 Overview of the Perspective Projection Algorithm . . . 61
4.2.2 Resampling the Volume . . . 62
4.2.3 Opacity Correction . . . 62
4.2.4 Implementation of the Perspective Projection Algorithm . . . 63
4.2.5 Limitations of the Perspective Projection Algorithm . . . 66
4.3 Fast Classification Algorithm . . . 68
4.3.1 Overview of the Fast Classification Algorithm . . . 69
4.3.2 The Min-Max Octree . . . 71
4.3.3 The Summed-Area Table . . . 74
4.3.4 Implementation of the Fast Classification Algorithm . . . 76
4.3.5 Limitations of the Fast Classification Algorithm . . . 79
4.4 Switching Between Modes . . . 81
4.5 Chapter Summary . . . 83

5 Performance Analysis **84**
5.1 Performance of the Shear-Warp Algorithms . . . 86

5.1.1 Speed and Memory Performance 86
5.1.2 Image Quality 95
5.2 Comparison of Coherence-Accelerated Volume Rendering Algorithms . . 98
5.2.1 Asymptotic Complexity 99
5.2.2 Experimental Methodology 103
5.2.3 Comparison of Speedups from Algorithmic Optimizations 105
5.2.4 Costs of Coherence Accelerations 110
5.2.5 Memory Overhead 114
5.2.6 Analysis of the Shear-Warp Coherence Data Structures 116
5.3 Low-Coherence Volumes 119
5.3.1 Categories of Volume Data 119
5.3.2 Voxel Throughput 121
5.3.3 The Impact of Coherence on Rendering Time 123
5.3.4 The Role of Coherence in Visualization 126
5.4 Chapter Summary 127

6 A Multiprocessor Volume Rendering Algorithm 128
6.1 Multiprocessor Rendering Algorithm 130
6.1.1 Image and Object Partitions 131
6.1.2 Task Shape 132
6.1.3 Load Balancing 133
6.1.4 Data Distribution 134
6.1.5 Overall Algorithm 134
6.2 Implementation 136
6.2.1 Hardware Architectures 136
6.2.2 Software Implementation 137
6.3 Results 138
6.3.1 Rendering Rates 138
6.3.2 Performance Limits on the Challenge Multiprocessor 141
6.3.3 Performance Limits on the DASH Multiprocessor 144
6.3.4 Memory Performance 146

6.3.5 Load Balancing . . . 148
6.3.6 Related Work . . . 149
6.4 Chapter Summary . . . 150

7 Extensions **152**
7.1 Flexible Shading With Lookup Tables . . . 153
7.1.1 Shade Trees and Lookup Tables . . . 153
7.1.2 Implementation of Shading Functions . . . 155
7.1.3 General Shade Trees . . . 160
7.2 Fast Depth Cueing . . . 160
7.2.1 Depth Cueing . . . 160
7.2.2 Factoring the Depth Cueing Function . . . 162
7.2.3 Implementation of Fast Depth Cueing . . . 164
7.3 Rendering Shadows with a 2D Shadow Buffer . . . 165
7.3.1 Algorithms for Rendering Shadows . . . 165
7.3.2 Implementation of the Shadow Rendering Algorithm . . . 167
7.3.3 Performance of the Shadow Rendering Algorithm . . . 169
7.4 Rendering Mixtures of Volume Data and Polygons . . . 173
7.5 Clipping Planes . . . 175
7.6 Chapter Summary . . . 176

8 VolPack: A Volume Rendering Library **177**
8.1 System Architecture . . . 178
8.2 Volume Representation . . . 180
8.3 Classification and Shading Functions . . . 181
8.4 Viewing Model . . . 182
8.5 Functionality Provided by VolPack . . . 182
8.6 Chapter Summary . . . 183

9 Conclusions **185**
9.1 Final Summary . . . 185
9.2 Future Directions for Performance Improvements . . . 187

9.3 Hardware Support for Volume Rendering 188
9.4 Interactive Volume Rendering 190

A Mathematics of the Shear-Warp Factorization **192**
A.1 Coordinate Systems and Definitions 192
A.2 The Affine Factorization 194
A.2.1 Finding the Principle Viewing Axis 194
A.2.2 Transformation to Standard Object Coordinates 196
A.2.3 The Shear and Warp Factors 197
A.2.4 Projection to the Intermediate Image 199
A.2.5 The Complete Affine Factorization 201
A.3 The Perspective Factorization 202
A.3.1 Finding the Principle Viewing Axis 202
A.3.2 Transformation to Standard Object Coordinates 204
A.3.3 The Shear and Warp Factors 205
A.3.4 Projection to the Intermediate Image 207
A.3.5 The Complete Perspective Factorization 209

Bibliography **211**

List of Tables

5.1 Characteristics of the data sets used for performance testing. 86
5.2 Rendering parameters used for performance testing. 88
5.3 Rendering time and memory usage results. 88
5.4 Timing results for the 2D warp. 89
5.5 Speedup of the shear-warp algorithm relative to a ray caster. 110
5.6 Timing breakdowns for coherence-accelerated rendering algorithms. . . . 111

6.1 Characteristics of the multiprocessors used for performance testing. 136
6.2 Rendering rates and tasks sizes on the SGI Challenge. 140
6.3 Rendering rates and task sizes on the Stanford DASH Multiprocessor. . . 140
6.4 Comparison of load balancing algorithms. 149

7.1 Performance results for the shading algorithm. 159
7.2 Performance results for fast depth cueing. 164

List of Figures

1.1 Example of a volume rendering. 2
1.2 A physical model for volume rendering. 4
1.3 A simplified model for volume rendering. 7
1.4 Examples of sampling grids in 2D. 11
1.5 The visualization process. 14

3.1 Sheared object space for a parallel projection. 30
3.2 Sheared object space for a perspective projection. 31
3.3 The brute-force shear-warp algorithm. 32
3.4 Determining the affine shear coefficients. 34
3.5 The perspective shear transformation. 36
3.6 The perspective scale transformation. 38
3.7 Resampling a translated voxel slice. 40
3.8 Ray casting with the template optimization. 42

4.1 Run-length encoding the intermediate image. 45
4.2 Resampling and compositing run-length encoded scanlines. 46
4.3 Run-length encoding the volume. 48
4.4 A tree data structure to represent an opaque pixel run. 50
4.5 The path compression optimization. 51
4.6 Opacity correction. 55
4.7 Plot of the opacity correction function. 56
4.8 Pseudo-code for the parallel projection algorithm. 57
4.9 Sampling the volume in the perspective projection algorithm. 61

4.10 Pseudo-code for the perspective projection algorithm. 64
4.11 Splitting the volume to support a wide field of view. 66
4.12 Visiblity errors in the perspective rendering algorithm. 68
4.13 Goal of the fast classification algorithm. 69
4.14 Data structures for the fast classification algorithm. 70
4.15 Diagram of a min-max octree. 72
4.16 Diagram of a 2D summed-area table. 74
4.17 Pseudo-code for the fast classification algorithm. 77
4.18 A family of volume rendering algorithms. 82

5.1 Volume rendering of an MR scan of a brain. 87
5.2 Volume rendering of a CT scan of a head. 87
5.3 Timing results for a rotation sequence. 90
5.4 Execution time for each stage of rendering. 92
5.5 Color volume rendering of a CT scan of a head. 93
5.6 Color volume rendering of a CT scan of an engine block. 93
5.7 Color volume rendering of a CT scan of a human abdomen. 94
5.8 Perspective volume rendering of a CT scan of an engine block. 94
5.9 Image quality comparison of the shear-warp algorithm with a ray caster. . 95
5.10 Image quality comparison of a bilinear filter with a trilinear filter. 96
5.11 Timing breakdowns without coherence optimizations. 106
5.12 Timing breakdowns with coherence optimizations. 108
5.13 Comparison of timing breakdowns for several data sets. 109
5.14 Pseudo-code for tracing a ray through an octree. 112
5.15 Memory overhead without coherence optimizations. 115
5.16 Memory overhead with coherence optimizations. 115
5.17 Impact of individual coherence optimizations. 117
5.18 Categories of volume data. 119
5.19 Results of the voxel throughput experiment. 122
5.20 Test volumes for the voxel coherence experiment. 123
5.21 Results of the voxel coherence experiment. 124

6.1 Pseudo-code for the multiprocessor algorithm. 135
6.2 Block diagram of a 2 x 2 DASH system. 137
6.3 Test data sets for the multiprocessor algorithm. 139
6.4 Interactive user interface for the multiprocessor algorithm. 139
6.5 Timing results for a rotation sequence on the DASH multiprocessor. . . . 141
6.6 Speedup curve on the Challenge multiprocessor. 142
6.7 Overhead vs. task size on the Challenge multiprocessor. 143
6.8 Overhead vs. number of processors on the Challenge multiprocessor. . . . 143
6.9 Speedup curve on the DASH multiprocessor. 145
6.10 Overhead vs. number of processors on the DASH multiprocessor. 146

7.1 An example shade tree. 155
7.2 Quantizing normal vectors. 158
7.3 Volume rendering with depth cueing. 161
7.4 Factoring the depth cueing function. 162
7.5 Diagram of Grant's shadow mask sweep algorithm. 166
7.6 Pseudo-code for the shadow rendering algorithm. 167
7.7 Volume rendering with the shadow rendering algorithm. 170
7.8 Limitations on the sampling rate in the shadow rendering algorithm. . . . 171
7.9 Aliasing in the shadow buffer. 172
7.10 Rendering mixtures of volume data and polygons. 174

8.1 Schematic diagram of a VolPack rendering context. 179

A.1 Coordinate systems used in the derivation of the shear-warp factorization. 193
A.2 Determining the affine shear coefficients. 198
A.3 Definition of the intermediate image coordinate system. 200
A.4 The shear transformation in the perspective shear-warp factorization. . . . 205
A.5 The scale transformation in the perspective shear-warp factorization. . . . 206