

# **Lattice-based Immersive Visualization**

A Dissertation Presented  
by  
**Kaloian Petkov**

to  
The Graduate School  
in Partial Fulfillment of the  
Requirements  
for the Degree of  
**Doctor of Philosophy**  
in  
**Computer Science**  
Stony Brook University

**December 2013**

Copyright by  
Kaloian Petkov  
2013

**Stony Brook University**

The Graduate School

**Kaloian Petkov**

We, the dissertation committee for the above candidate for the

Doctor of Philosophy degree, hereby recommend

acceptance of this dissertation.

**Arie Kaufman – Dissertation Advisor  
Distinguished Professor and Chair, Computer Science Department**

**Xianfeng Gu - Chairperson of Defense  
Associate Professor, Computer Science Department**

**Klaus Mueller  
Professor, Computer Science Department**

**Amitabh Varshney  
Professor, Computer Science Department, University of Maryland**

This dissertation is accepted by the Graduate School

**Charles Taber  
Dean of the Graduate School**

Abstract of the Dissertation

**Lattice-based Immersive Visualization**

by

**Kaloian Petkov**

**Doctor of Philosophy**

in

**Computer Science**

Stony Brook University

**2013**

The sizes of data are increasing at a very rapid pace in many applications, including medical visualization, physical simulations and industrial scanning. Some of this growth is not only due to the development of high resolution medical and industrial scanners, but also due to the wide availability of high performance graphics processing units (GPUs) which allow for the interactive rendering of large datasets. However, the increase of problem sizes has generally outpaced the increase of onboard GPU memory. At the same time, the resolution of the traditional display systems has not improved significantly compared to the exponential growth of computing power. We have developed a comprehensive approach that tackles the efficiency of the data representation through lattice-based techniques, as well as the visualization capabilities for exploring that data. We have constructed the Immersive Cabin and the Reality Deck facilities, along with a set of visualization techniques, to address the challenge of growing data sizes.

In terms of sampling lattices, we have developed a Computational Fluid Dynamics (CFD) simulation framework based on the lattice Boltzmann method and using optimal sampling lattices. Our focus is specifically on the Face-centered Cubic lattice (FCC), which can achieve a stable simulation with only 13 lattice velocities while at the same time improving the sampling efficiency compared to using the traditional Cartesian grid. We demonstrate the resulting fD3Q13 LBM model for use in highly interactive smoke dispersion simulations. The simulation code is coupled with our visualization framework, which includes a high-performance volume renderer and support for virtual reality systems. The volume rendering is further enhanced with a novel LOD scheme for large volume data that allows for mixing the optimal sampling lattices in adjacent levels of the hierarchy with a computationally cheap indexing function.

We have developed a visualization framework for the Immersive Cabin which supports the traditional virtual reality components, such as distributed rendering, stereo, tracking, rigid-body physics and sound. The lattice-based visualization and simulation techniques, including the support for mixed-lattice hierarchies, are integrated in the framework and can be combined with the mesh rendering and the rigid-body physics simulation. Based on our experience in the Immersive Cabin, we have designed and constructed the Reality Deck, which is the world's first 1.5 gigapixel immersive display. The Reality Deck contains 416 high resolution LCD monitors in a 4-wall surround layout, and similarly to the Immersive Cabin, uses a unique automatic door that is also a display surface. The graphics are generated on an 18-node cluster with 24 displays connected to each node using the AMD Eyefinity technology. We have extended the Immersive Cabin visualization framework to support the Reality Deck and developed a new gigapixel image renderer targeted at scientific and immersive visualization.

We have developed a set of visualization techniques for the exploration of large and complex data in both of our facilities. Conformal Visualization is a novel retargeting approach for partially-enclosed VR environments, such as the Immersive Cabin and the Reality Deck, to allow for the complete visualization of the data even when display surfaces are missing. Our technique uses conformal mapping to ensure that shape is preserved locally under the transformation and we demonstrate its use for the visualization of both mesh and lattice data. In our Frameless Visualization technique, the traditional framebuffer is replaced with reconstruction from a stream of rendered samples. This approach allows smooth user interaction even when rendering on a gigapixel display, such as the Reality Deck, or when using computationally-expensive visualization algorithms. Our system generates low-latency image samples using an optimal sampling lattice in the 2D+time space, as well as importance-driven higher quality samples. Finally, we have developed the Infinite Canvas visualization technique for horizontally-enclosed visual environments. As the user moves through the physical space of the facility, the graphics outside of the field of view are updated to create the illusion of an infinite continuous canvas. The Infinite Canvas has been used for the visual exploration of gigapixel datasets that are an order of magnitude larger than the surface area of the Reality Deck, including very large image collections.

*To my parents.*



# Contents

<b>List of Figures</b>	<b>xi</b>
<b>List of Tables</b>	<b>xxi</b>
<b>Acknowledgements</b>	<b>xxiii</b>
<b>Publications</b>	<b>xxv</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Motivation . . . . .	1
1.2 Contributions . . . . .	3
<b>2 Background</b>	<b>5</b>
2.1 Sampling Lattices . . . . .	5
2.2 Lattice Reconstruction . . . . .	8
2.3 Multi-resolution Techniques for Volumetric Data . . . . .	9
2.4 The Lattice Boltzmann Method . . . . .	10
2.4.1 Definition . . . . .	10
2.4.2 Boundary Conditions . . . . .	12
2.5 Immersive Visualization . . . . .	13
2.6 Frameless Rendering . . . . .	15
<b>3 LBM Fluid Simulation and Visualization on Optimal Sampling Lattices</b>	<b>17</b>
3.1 Introduction . . . . .	17
3.2 The fD3Q13 Lattice . . . . .	19
3.3 Validation . . . . .	24
3.3.1 Poiseuille Flow . . . . .	24
3.3.2 Flow Past a Sphere . . . . .	26

## CONTENTS

---

3.4	LBM Implementation with CUDA . . . . .	29
3.4.1	Programming Model . . . . .	29
3.4.2	LBM with CUDA . . . . .	30
3.5	Smoke Simulation in Urban Environment . . . . .	33
3.5.1	Urban Modeling . . . . .	33
3.5.2	Modeling of Smoke in Urban Environment . . . . .	34
3.5.3	Rendering . . . . .	38
3.5.4	Results . . . . .	41
<b>4</b>	<b>Hierarchical Volume Rendering on Mixed Lattices</b>	<b>43</b>
4.1	Introduction . . . . .	43
4.2	Mixed-Lattice Hierarchy Construction . . . . .	44
4.3	Mixed-lattice Rendering . . . . .	46
4.4	Implementation and Results . . . . .	49
4.4.1	Reconstruction Filters . . . . .	49
4.4.2	Hierarchical Rendering . . . . .	53
<b>5</b>	<b>Immersive Environments</b>	<b>57</b>
5.1	The Immersive Cabin . . . . .	58
5.1.1	Equipment and Construction . . . . .	59
5.1.2	Rendering Framework . . . . .	61
5.1.3	Remote Visualization . . . . .	63
5.1.4	Applications . . . . .	66
5.2	The Reality Deck . . . . .	70
5.2.1	Equipment and Construction . . . . .	71
5.2.2	Rendering Framework . . . . .	78
5.2.3	Applications . . . . .	80
5.3	Case Studies in Immersive Visualization . . . . .	85
5.3.1	Case Study: World-wide data visualization . . . . .	85
5.3.2	Case Study: Fusion of GIS and procedural modeling for driving simulation in New York . . . . .	87
5.3.3	Case Study: Immersive Medical Visualization . . . . .	91

---

## CONTENTS

<b>6 Distributed Immersive Visualization</b>	<b>97</b>
6.1 Conformal Visualization . . . . .	98
6.1.1 Theoretical Background . . . . .	99
6.1.2 Implementation Details . . . . .	105
6.1.3 Visualization Techniques . . . . .	110
6.1.4 Results . . . . .	114
6.1.5 Evaluation . . . . .	124
6.2 Frameless Visualization . . . . .	132
6.2.1 Algorithm Overview . . . . .	133
6.2.2 Adaptive Sampling . . . . .	134
6.2.3 Implementation Details . . . . .	137
6.2.4 Results . . . . .	138
6.3 The Reality Deck Infinite Canvas . . . . .	146
6.3.1 Basic Formulation . . . . .	147
6.3.2 Spiral Navigation . . . . .	150
6.3.3 Multiuser Visualization . . . . .	152
6.3.4 Results . . . . .	153
<b>7 Conclusions</b>	<b>157</b>
7.1 Summary of Contributions . . . . .	157
7.2 Future Research . . . . .	158
<b>Bibliography</b>	<b>161</b>

## **CONTENTS**

---

# List of Figures

2.1	The BCC lattice can be constructed by (d) adding one lattice site at the center of every cell of the CC lattice (a), or by (e) interleaving two CC lattices. The FCC lattice can be constructed by (b) placing sites at the face centers of the CC unit cell, or by (c) interleaving four CC lattices. . . . .	6
2.2	Unit cell and Voronoi cell for the (a) FCC and (b) BCC lattices. Unit cells (illustrated with the black edges) are the cuboctahedron and the rhombic dodecahedron, respectively. The Voronoi cells (colored in red) are the rhombic dodecahedron and the truncated octahedron, respectively. . . . .	6
2.3	Curved boundary treatment. The filled circles represent the discretized geometric object. The clear circles represent fluid cells. The blue circle at position $x_w$ is the intersection point between the surface of the geometric object and a lattice link. . . . .	14
3.1	Hexagonal 2D lattice, D2Q7. The 6 nearest neighboring sites (green dots) of a site (red dots) form a regular hexagon. The Voronoi cell (blue) is also a regular hexagon. . . . .	19
3.2	A 3D BCC lattice (a) can be constructed by adding one lattice site at the center of every cell of the CC lattice. A 3D FCC lattice (b) can be constructed by placing sites at the centers of the square surfaces of every cell of the CC lattice. A 3D HCP lattice (c) can be constructed by layering 2D HCP lattices. In (c), the blue sites form the 2D HCP lattice, which is the first layer of the 3D HCP lattice. The green sites represent the second layer. The entire 3D HCP lattice contains alternating blue and green layers. . . . .	20
3.3	A $4^3$ CC lattice divided into two interleaving FCC lattices denoted by light and dark cells. .	21
3.4	Unit cell and Voronoi structures for the FCC lattice. . . . .	22
3.5	Unit cell and Voronoi structures for the BCC lattice. . . . .	23
3.6	Comparison of the velocity profiles for the Poiseuille Flow. The Cartesian and hexagonal lattices perform identically and both achieve the analytical result after 20,000 iterations. . . .	25

## LIST OF FIGURES

---

3.7 (a) Illustration of the Poiseuille Flow after 20,000 iterations. The simulations based on D2Q7 and D2Q9 lattices produce identical results. (b) and (c) Simulation results for flow past a circular boundary in 2D for the D2Q9 and D2Q7 lattices, respectively. (d) Relative scale used for visualization of the speed of the flows. . . . .	26
3.8 Comparison with the steady-state analytical approximation for the drag coefficient of a sphere moving through a viscous media over time. . . . .	28
3.9 Visualization of vortices in the flow behind the sphere at $Re = 80$ . $\lambda = \frac{d}{D} = \frac{15}{60}$ , where $d$ is the diameter of the sphere and $D$ is the diameter of the cylindrical boundary. . . . .	29
3.10 Performance scaling for CPU and GPU architectures. . . . .	32
3.11 Comparison of detail resolution under different smoke advection schemes. . . . .	37
3.12 Every slice (green lines) of the lighting volume (blue box) is perpendicular to the light direction. The lighting volume stores light intensity and density sampled from the density volume (black box). The light intensity of slice $i$ is calculated by attenuating slice $i - 1$ with the current opacity values. The ray is traced through the lighting volume. The entry point and exit point of the ray are on the bounding box of the density volume. . . . .	39
3.13 Snapshots of smoke simulation in the urban environment. . . . .	40
4.1 Visualization of the Voronoi cells in the (a) CC, (b) FCC and (c) BCC lattices for an identical number of voxels. The optimal (BCC) and near-optimal (FCC) sampling lattices allow for improved spatial resolution at the same voxel density compared to the traditional CC lattice. Compared to the traditional octree hierarchy illustrated in (d) our mixed lattice hierarchy (e) allows for smoother level-of-detail transitions. . . . .	44
4.2 Hierarchical domain refinement using the mixed CC (blue spheres), BCC (red spheres) and FCC (green spheres) lattices. The top layer shows the next CC level in the refinement. . . . .	45
4.3 Hierarchical storage assuming bricks with identical dimensions for the CC, FCC and BCC levels. Each lower resolution brick spans exactly 2 bricks from the higher resolution level in the hierarchy. . . . .	47
4.4 The Dragon dataset is represented by a $128^3$ distance field and we visualize the Voronoi cells of the underlying voxels. The hierarchies use $32^3$ bricks and a $16^3$ index. Compared to the octree hierarchy (a), both the CFB (b) and the CBF (c) hierarchies allow for smoother transitions between the levels and take advantage of the optimal and near-optimal sampling properties of the BCC and FCC lattices. Successive LOD levels are colored cyclically in red, green and yellow, starting with the finest resolution CC level. . . . .	48

---

## LIST OF FIGURES

4.5	The Marschner-Lobb (ML) function is sampled on a $40^3$ CC grid, a $50 \times 50 \times 25$ FCC grid and a $32 \times 64 \times 32$ BCC grid. We compare the different reconstruction filters implemented on the GPU in terms of their ability to resolve the thin structures in the ML dataset. . . . .	50
4.6	We compute the distance field for the Stanford Dragon mesh dataset over the CC, BCC and FCC lattices at similar voxel densities (equivalent to $64^3$ voxels). Each pair of images illustrates the Voronoi cells of the lattice positions on the left and the reconstructed isosurface on the right. . . . .	51
4.7	Filtered voxelization on the CBF hierarchy with 7 LODs. Here we illustrate the transition between two successive CC levels. . . . .	54
4.8	Filtered voxelization on a non-persistent CC hierarchy shown for comparison. Voxel density at each level is approximately equal to the density at the corresponding level in the mixed hierarchy. This hierarchy is not practical due to the non-uniform coverage by bricks of constant size. Also, compared to Figure 4.7, the advantage of optimal and near-optimal sampling levels is lost. . . . .	54
4.9	Hierarchical volume rendering of a large voxelized scene. (a) and (b) show high-quality direct volume rendering of the distance field. (c) illustrates the LOD selection with a 4-level CC hierarchy where brightness indicates sample density. (d) uses the CBF hierarchy with 12 levels where red, green and blue indicate the CC, BCC and FCC levels respectively and brightness denotes the sample density. . . . .	55
4.10	Volume rendering of the Skull dataset using (a) the CC octree with 4 levels, (b) the CBF hierarchy with 12 levels and the same LOD selection heuristic. . . . .	56
4.11	(a) The full resolution dataset is rendered from a distance, which results in aliasing artifacts. (b) The CBF hierarchy with an LOD selection allows for finely-adaptive antialiasing. . . . .	56
5.1	(a) The Immersive Cabin is a 5-sided CAVE with an automatic door and a high-end active stereo projector system. (b) The Reality Deck is a large 416 display installation with an immersive layout and an aggregate resolution in excess of 1.5 billion pixels. . . . .	57
5.2	Diagram for an installation of the Immersive Cabin. . . . .	60
5.3	Blue cloud represents the tracking coverage areas seen simultaneously by (left to right) 3, 6 and 8 cameras. The positions near the center of the IC that are seen by 6 cameras are optimal for head and gesture tracking since optical occlusions are minimized. . . . .	61
5.4	Volume rendering used for (a) medical visualization and (b) simulation data exploration. . .	63
5.5	The NVIDIA OptiX GPU raytracing is used in the Immersive Cabin for (a) complex reflections and shadows, and (b) global illumination. . . . .	64

## LIST OF FIGURES

---

5.6	Chess scene rendered with the different graphics pipelines. . . . .	64
5.7	Video streams generated by IC nodes are displayed on a tablet. (a) The UI is running natively on the tablet and controlling the IC; the video is streamed from the cluster head node. (b) and (c) The output of the forward-view node in the IC is streamed to the tablet; the UI is running on the cluster head node. . . . .	66
5.8	Exploration of a large urban environment. . . . .	67
5.9	Architectural rendering of (a) the AERTC and (b) the new Computer Science building, both at Stony Brook University. . . . .	68
5.10	Virtual Colonoscopy in the Immersive Cabin. . . . .	69
5.11	The Reality Deck uses 416 LCD panels to achieve resolution in excess of 1.5 billion pixels in an immersive 4-wall layout. . . . .	70
5.12	We evaluated the perceptual effects of the screen size and the bezel size for different LCD monitors in the Immersive Cabin using various datasets. The study included both 2D and stereoscopic rendering. Left wall in both images simulates a large-format Planar monitor (60" diameter, 3840 × 2160); right wall is the Samsung S27A850D with modified bezels (27" diameter, 2560 × 1440). . . . .	72
5.13	Each monitor is tested for image uniformity when displaying a full white and a full black signal. We also look for any significant color issues. (a) to (c) are the results for a rejected monitor (significant uniformity issues when showing a black image). (d) to (f) correspond to a reference monitor without any image quality issues. . . . .	73
5.14	Schematic drawings of (a) the mounting bracket attached to each monitor, (b) the mounting frame and (c) two complete columns of mounted monitors. . . . .	74
5.15	Our facility contains an automatic door composed of a 3 × 5 grid of monitors. There is a small handle used to open the door during power outages. Except for the handle and the exit sign, the door is visually indistinguishable from the rest of the display. . . . .	75
5.16	Photographs of the Reality Deck running GIS applications. (a) A gigapixel aerial photograph for New York City with 0.5m resolution is mapped on 5m elevation data and we provide an interface for 3D flight over the terrain. The entire dataset is shown in Figure 5.20. (b) High-resolution global elevation data is displayed with a relief shader, including a simple interactive threshold-based flooding simulation. The control interface is running on the PixelSense touch table. . . . .	76
5.17	The Reality Deck utilizes 24 infrared cameras for optical tracking. The positions and the orientations of the cameras are designed to maximize the effective tracking volume, especially near the displays. . . . .	77

---

## LIST OF FIGURES

5.18 The Reality Deck sound system uses a 24.4 surround speaker layout with 12 speakers below the displays, 12 speakers above and 4 large subwoofers at the corners. . . . .	78
5.19 Photograph of the front and parts of the side walls in our immersive display showing a gigapixel panoramic image of Dubai at full resolution. Both small-scale and large-scale features are visible without panning or zooming the data. The close-up images are obtain with macro photography simply by moving the camera near the screen. . . . .	81
5.20 Gigapixel aerial photograph of New York, obtained from ArcGIS's World Imagery map service, is combined with the elevation data from USGS. We use pre-compressed tile data to provide very fast texture uploads during interactive exploration in the Reality Deck (examples of using this data in the Reality Deck are shown in Figure 5.16a and Figure 5.28). . . . .	82
5.21 Photograph of the front and parts of the side walls in the Reality Deck showing a 6 gigapixel view of the Milky Way from NASA's Spitzer Space Telescope that has been wrapped around the display. . . . .	83
5.22 9 gigapixel view of the Milky Way from the VISTA survey telescope at the ESO Paranal Observatory. The image is captured directly from our visualization software and the close-up images show the highest resolution tile data. . . . .	84
5.23 Our software can render a scene with 40M triangles with interactive framerates at the full 1.5 billion pixel resolution of our immersive display. . . . .	84
5.24 Interactive visualization of world-wide GIS data in the Reality Deck. . . . .	87
5.25 Roads generated procedurally from the GIS street network and attributes at different levels of detail. . . . .	89
5.26 (a) Buildings generated procedurally on randomized lots within the GIS street network. (b) At the higher levels of detail more facade details are generated. (c) Buildings generated from accurate GIS footprints and heights, together with procedural photo-based facades. . . . .	90
5.27 Different views of the 3.3 mile Manhattan track. On the foreground in (a) are low-resolution landmark 3D models obtained from LIDAR data, which are used as a backdrop during the driving simulation. . . . .	91
5.28 (a),(b) Snapshots of the 3D model in the Immersive Cabin for visual validation with stereoscopic rendering. (c) Visualization of the entire model in the Reality Deck, including the two generated tracks at the maximum LOD, the backdrop LIDAR buildings and a gigapixel terrain from aerial photographs (the IC version uses high resolution but not gigapixel terrain texture). A version of the terrain model is also shown in Figure 5.20. . . . .	94

## LIST OF FIGURES

---

5.29 Exploration of (a) structure <i>Ikee</i> from the Protein Data Bank in (b) the Immersive Cabin and (c) the Reality Deck. Only the left-eye images from the stereoscopic pair in the IC are shown to improve the clarity of the image. . . . .	95
5.30 Visualization for Virtual Colonoscopy in (a) the Immersive Cabin and (b) the Reality Deck. The IC presents the traditional 3D navigation interface following with centerline and using volume rendering. The same interface is available in the RD, but (b) presents a novel gigapixel visualization where two scans of the entire colon of the patient are displayed at once. . . . .	95
5.31 Exploration of the Visible Human dataset in (a) the Immersive Cabin and (b) the Reality Deck. The high pixel density in the RD allows for a more natural visualization compared to the 12 dpi pixel density in the IC, which requires constant translation and zooming. . . . .	96
 6.1 Circle Packing Metric. . . . .	102
6.2 Riemann Mapping Algorithm . . . . .	104
6.3 Algorithm for conformal mapping between a 5-sided CAVE and a 6-sided CAVE. . . . .	106
6.4 Template meshes for a 5-sided CAVE. . . . .	107
6.5 Template meshes for a 3 screen target. The cut in (a) is suitable for the 3-sided CAVE. For an arrangement of flat-panel displays (c), the cut in (b) reduces the distortion effects. . . . .	108
6.6 Template meshes for a 4-sided CAVE. . . . .	109
6.7 Cubemaps for the $T_{ray}$ conformal transformation based on the visualization targets described in Section 6.1.2. The cubemap in figure (a) is the identity transformation. . . . .	110
6.8 Raytracing of a checkerboard sphere with conformal visualization on different display targets. The layout for (a)-(c) is in the standard vertical cross format and the color coding is defined for the original walls in the 6-sided CAVE configuration (blue for front/back, red for left/right, green for top/bottom). For (d), we show the output of all 3 displays side by side. . . . .	115
6.9 Navigation in the checkerboard tunnel with conformal visualization where the camera pans down in (a)-(d). Each triplet of images shows the original front view (lower-left), original top view (upper-left) and front view with conformal visualization (right). . . . .	116
6.10 Performance of the cubemap generation running on a single quad-core Intel Xeon E5620. . . . .	117
6.11 Visualizing the mesh model of a patient’s colon during Virtual Colonoscopy. The left pairs of images show the front and top views, and the image on the right shows the front view after conformal distortion. The shape of the polyp is preserved under the conformal geometry deformation. . . . .	118

---

## LIST OF FIGURES

6.12 Conformal visualization applied to the rasterization pipeline. The images show the original front view (lower-left), the original top view (upper-left) and the front view with conformal visualization (right) for the same camera position. Triangular patches are drawn in blue, tessellated triangles are in black. Our technique produces artifact-free results even for coarse scene geometries. . . . .	119
6.13 (a) Snapshot of Immersive Virtual Colonoscopy in the 5-sided CAVE. The shape of the polyp on the top view is preserved under the conformal visualization. Compared to the traditional volume rendering (b), the conformal visualization allows for a more thorough analysis of the dataset (c) and the entire surface of the colon is visible as the user negotiates a bend in the colon. . . . .	121
6.14 Snapshot of an architectural fly-through using conformal visualization expanded visual context during the navigation. . . . .	122
6.15 Conformal visualization results for a large tiled display. (a) The standard pinhole camera model with wide FOV leads to significant distortions near the periphery of the image. (b) In contrast, conformal visualization allows for 180° horizontal and vertical FOV while locally preserving the shapes in the data. . . . .	127
6.16 Conformal visualization results. (a) Each triplet of images shows screen captures of the original front view (lower-left), the original top view (upper-left) and the front view with conformal visualization (right). The photographs (b)-(d) illustrate views from inside the CAVE where the front wall is on the left side of the image. Missing visual information from (c) the top wall and (d) the top and back walls is presented on the available display surfaces. The 4-sided CAVE is simulated by disabling the back wall. . . . .	128
6.17 Conformal visualization results for the GPU raytracing pipeline with dynamic visibility manipulation. The standard conformal parameterization for the 5-sided CAVE is used, while the reference point for computing the GPU textures moves between the extreme points at the center of (b) the back wall and (c) the front wall. The recomputation executes at realtime speeds and allows for a context-preserving zoom operation (e.g., toward the dense cluster of nodes at the front wall). . . . .	129
6.18 High resolution stitched photographs from the four walls in the Reality Deck while running protein visualization. (a) shows normal rendering with OpenGL, (b) uses our conformal visualization technique to expand the field of view while preserving the shapes locally. . . .	130
6.19 Phantom dataset for the user study. The center-line for the generated data is based on a section of a patient's VC. . . . .	131
6.20 The rendering pipeline for our frameless visualization technique. . . . .	134

## LIST OF FIGURES

---

- 6.21 The volume rendering is performed onto 2D slices of the BCC lattice (lattice sites in red). During reconstruction, full images can be produced at arbitrary time steps (green layers) . . . . . 135
- 6.22 The analytic ML function (a) is rendered with raytracing using image-space lattice samples on (b) the CC grid and (c) the BCC grid with an equal number of samples. . . . . 139
- 6.23 Image reconstruction over time with the frameless visualization system. The low resolution initial image (a) is produced very rapidly with GPU rasterization over the BCC lattice samples. With additional adaptive samples as in (b), the global illumination effects are clearly distinguishable and the reconstruction converges on the GI solution in (c). . . . . 140
- 6.24 During interactive visualization, initial low resolution images reconstructed from the lattice samples (b) are significantly less noisy than images produced with the traditional frameless rendering (a) . . . . . 141
- 6.25 During dynamic scene changes, our system is biased toward the lattice reconstruction as shown in (b). The noise in the image is reduced compared to (a) the traditional frameless rendering. . . . . 141
- 6.26 Frameless visualization of the Visible Human dataset with Direct Volume Rendering on the GPU. (a) The initial image during interaction is generated from the lattice samples. (b) Adaptive samples are streamed to improve the resolution of the image. (c) A high-order filter and a smaller step size are used during the reconstruction of the volume data to generate high quality samples, which are streamed continuously at a slower rate. . . . . 142
- 6.27 Interactive frameless visualization of (a) the combustion simulation dataset and (b) the Visible Human volumetric dataset at the full display resolution. The backend sample rendering is configured respectively for GPU raytracing with 6 nodes / 6 GPUs and volume rendering with 12 nodes / 30 GPUs. . . . . 145
- 6.28 The full dataset (center strip) is three times longer than the surface of the immersive display. As the user rotates clockwise, different segments of the data are revealed. The angle of relative rotation is indicated next to the human figure and the cumulative angle of rotation is shown at the bottom left of each subfigure. The discontinuity where the canvas wraps over is always kept behind the user, which provides the perception of one long continuous surface. The numbers on the screens in each subfigure correspond to the numbers on the original strip of data. . . . . 148
- 6.29 The infinite canvas technique from Figure 6.28 is visualized using the NASA Milky Way dataset on a 3D model of the Reality Deck. The images for the simulated screens are captured directly from our visualization software. . . . . 149

---

## LIST OF FIGURES

6.30 The spiral navigation interface provides a quick overview of the entire dataset that the user can explore by moving inside the immersive display. A red flashing highlight is used to indicate the exact position in the full data. . . . .	150
6.31 Exploring the Milky Way data with the spiral navigation interface. The original image (bottom) is five times longer than the immersive display. The user can quickly overview the entire dataset by moving up and down through the spiral visualization using a gamepad or another interaction device. . . . .	151
6.32 Synthesized view of the multi-user exploration interface. The person closer to the screen is examining the beginning of the dataset, while the second person is using the spiral navigation interface to select point near the center of the dataset (the red highlight is at the central cycle in the spiral). . . . .	152
6.33 The 6GPixel infrared portrait of the inner Milky Way galaxy from the combined observations of the Galactic Legacy Infrared Mid-Plane Survey Extraordinaire (GLIMPSE) and MIPSGAL projects is shown on the immersive display using the navigation spiral. . . . .	154
6.34 Our art gallery dataset contains more than 70,000 painting which can be displayed one per screen (right) or in grid layouts depending on the image resolution (left) using the infinite canvas. . . . .	154
6.35 The entire 70,000 paintings art gallery is displayed using the spiral interface. There is sufficient resolution to visually identify clusters of paintings. . . . .	155
6.36 Two-user exploration of the Milky Way dataset in the immersive display. The photo shows the front-right corner of the facility. The first user is looking toward the front screen and the visualization is not affected by the actions of the second user who is primarily looking at the right screen when interacting with the data. . . . .	156

## **LIST OF FIGURES**

---

# List of Tables

3.1	Comparison of LBM performance on the D2Q7 and D2Q9 lattices in terms of Seconds Per Time Step (SPTS) and Lattice Updates Per Second (LUPS). Lattice sizes are $64 \times 64$ for the Cartesian lattices and $59 \times 68$ for the hexagonal lattice. Simulations are run on an Intel Core Duo 1.86GHz processor. . . . .	26
3.2	Comparison of LBM performance on different lattice types in terms of Seconds Per Time Step (SPTS) and Lattice Updates Per Second (LUPS). Lattice sizes are $64 \times 64 \times 64$ for the Cartesian lattices, $32 \times 64 \times 64$ for the FCC lattice and $45 \times 45 \times 91$ for the BCC lattice. Simulations are run on an Intel Core Duo 1.86GHz processor. . . . .	30
3.3	The effect of memory access optimizations on CUDA performance, expressed in MLUPS. . . . .	31
3.4	LBM Performance in MLUPS for different architectures and domain sizes. . . . .	32
3.5	Performance comparison for CUDA code running in single- and double-precision. Performance numbers are in MLUPS. . . . .	33
3.6	Comparison of runtime performance for simulations using D3Q19 SRT LBM, D3Q13 MRT LBM, and fD3Q13 MRT LBM. . . . .	41
4.1	Comparison of the performance characteristics of the NVIDIA Quadro FX 2500M (GPU1) and NVIDIA Quadro FX 5800 (GPU2) GPUs. . . . .	52
4.2	Relative performance of the different filters on the GPU. The baseline is the CC filter with non-native interpolation running on the NVIDIA Quadro FX 2500M (GPU1). . . . .	52
6.1	Relative performance of the different filters on the NVIDIA Geforce GTX 470, together with the cost in terms of Nearest Neighbor (NN) or trilinear (lin) texture fetches. The baseline is the performance of the cubic B-spline filter for CC data. . . . .	139
6.2	Sample throughput (samples/sec) and convergence times (s) for a 6-node GPU cluster and a single quad-GPU visualization node running 24 screens at $7960 \times 12088$ total resolution. . . . .	143



# Acknowledgements

This thesis is the culmination of a long but wonderful journey that would not have been possible without the help of others. I would like to take this opportunity to thank everyone who helped and supported me during this time.

I am immensely grateful to my family - my parents, my brother, my sister, and everyone else back home - for their love and encouragement throughout my life. Without their support, especially during the difficult times, this thesis simply would not exist.

I am truly grateful to my adviser, Prof. Arie Kaufman, for his guidance and inspiration over the past seven years, and for all the opportunities he has given me to pursue my interests. He has taught me so much and it has truly been a privilege to be his graduate student. I would also like to thank Profs. Klaus Mueller and Xianfeng David Gu for their invaluable help and support over the years. I have been very fortunate to have them on my thesis committee and I cannot imagine producing this work with anyone else. I am also very grateful to the last member of my committee, Prof. Amitabh Varshney, for helping me make this the best thesis that I can.

During my time at the Visualization lab, I have met so many incredible people that it is simply not possible to thank all of them individually. My friends and lab mates, in no particular order - Feng Qiu, Zhe Fan, Joseph Marino, Charilaos Papadopoulos, Ievgeniia Gutenko, Xin Zhao, Lei Wang, Krishna Chaitanya, Koosha Mirhosseini, Ji Hwan Park, Saad Nadeem, and anyone else I may have missed, I am grateful to all of them for their help and support, for our collaborations and for the ideas we have shared. My special thanks go to Ken Gladky, Brad Nelson and Bin Zhang, without whom none of the amazing systems we built would have been possible.

*THANK YOU!*

## **0. ACKNOWLEDGEMENTS**

---

# Publications

## Refereed Journal Publications

1. **K. Petkov**, and A. Kaufman. Lattice-based Frameless Volume Visualization. *IEEE Transactions on Visualization and Computer Graphics*. (submitted) 2013. (Section 6.2)
2. **K. Petkov**, A. Kaufman, and K. Mueller. Hierarchical Volume Rendering on Mixed Lattices. *IEEE Transactions on Visualization and Computer Graphics*. (submitted) 2013. (Chapter 4)
3. C. Papadopoulos, **K. Petkov**, A. Kaufman, and K. Mueller. The Reality Deck - Engineering and Research. *IEEE Computer Graphics and Applications*. (to be submitted) 2014. (Section 5.2)
4. **Petkov, K.**, Papadopoulos, C., Zhang, M., Kaufman, A., and Gu, X., Interactive Visibility Retargeting in VR using Conformal Visualization. *IEEE Transactions on Visualization and Computer Graphics*, 18(7):1027–1040. 2012. (Section 6.1)
5. **Petkov, K.**, Fan, Z., Qiu, F., Mueller, K., and Kaufman, A.E. Efficient LBM Flow Simulation on Face-Centered Cubic Lattices. *IEEE Transactions on Visualization and Computer Graphics*, 15(5):802–814. 2009. (Chapter 3)
6. Kaufman, A., Fan, Z., and **Petkov, K.** Implementing the Lattice Boltzmann Model on Commodity Graphics Hardware. *Journal of Statistical Mechanics: Theory and Experiment*, 2009(6):P06016. 2009. (Chapter 3)

## Refereed Conference Publications

7. I. Gutenko, X. Zhao, J. H. Park, **K. Petkov**, C. Papadopoulos, A. Kaufman, R. Cha. Remote volume rendering pipeline for mHealth applications. *SPIE Medical Imaging*. (to appear) 2014. (Section 5.1.3)
8. **K. Petkov**, C. Papadopoulos, A. Kaufman. Visual Exploration of the Infinite Canvas. *IEEE Virtual Reality*, pp. 11–14. 2013. (Section 6.3)
9. **K. Petkov**, C. Papadopoulos, M. Zhang, A. Kaufman, X.D. Gu. Conformal Visualization for Partially-Immersive Platforms. *IEEE Virtual Reality*, pp. 143–150, 2011. (Section 6.1)

## 0. PUBLICATIONS

---

10. F. Qiu, B. Zhang, **K. Petkov**, L. Chong, A. Kaufman, K. Mueller, and X.D. Gu. Enclosed Five-Wall Immersive Cabin. *ISVC '08: Proceedings of the 4th International Symposium on Advances in Visual Computing*, pp. 891-900. 2008. (Section 5.1)

## Other Publications

11. **K. Petkov**, A. Kaufman. Visualization Applications in the RealityDeck. *PacificVis*. 2013. Poster and abstract. (Section 5.2)
12. **K. Petkov**, A. Kaufman. Efficient Indexing for Hierarchical Mixed Lattices. *PacificVis*. 2012. Poster and abstract. (Chapter 4)
13. **K. Petkov**, and A. Kaufman. Advanced Visualization and Interactive Applications in the Immersive Cabin. *CEWIT International Conference*. 2010. (Section 5.1)
14. **K. Petkov**, C. Papadopoulos, and A. Kaufman. Immersive Virtual Colonoscopy. *CEWIT International Conference*. 2010. (Section 5.1)
15. C. Papadopoulos, **K. Petkov**, A. Kaufman and K. Mueller. Reality Deck - Immersive GigaPixel Display. *CEWIT International Conference*. 2010. (Section 5.2)
16. **K. Petkov** and A. Kaufman. Immersive Exploration of Large Datasets. *Center for Dynamic Data Analytics (CDDA) Planning Workshop*. 2010. Poster and abstract. (Section 5.2)
17. **K. Petkov** and A. Kaufman. Advanced Visualization Applications in the Immersive Cabin. *CEWIT International Conference*. 2009. Poster and abstract. (Section 5.1)

# 1

## Introduction

### 1.1 Motivation

The size of data, and especially volumetric data, is increasing at a very rapid pace in many applications, including medical visualization, physical simulations and industrial scanning. Some of this growth is not only due to the development of high resolution medical and industrial scanners, but also due to the availability of high performance graphics processing units (GPUs) which allow for the interactive rendering of large volumetric datasets. The GPU is a massively parallel computation device dedicated to graphics processing and much of the performance evolution over the last 10 years can be attributed to the desire for ever increasing realism in 3D computer games. At the same time, each new generation has become more programmable to the point where commodity graphics hardware can be used for general purpose computations in addition to accelerating the rasterization graphics pipeline.

However, the increase of problem sizes has generally outpaced the increase of onboard GPU memory. A number of techniques have been developed to tackle this challenge, including distributed GPU rendering and simulation [37], compression [77] and hierarchical representations [9]. Generally, the Cartesian grid is the predominant regular sampling domain in areas such as medicine, science and engineering due to its simplicity, efficient mapping to the CPU and GPU memories and well-understood sampling properties. Recently, the optimal sampling lattices have been proposed as a viable (and in fact desirable) alternative in different areas, from data sampling and reconstruction [33] to physical simulations [1, 84]. Through the Fourier sampling theorem, it can be shown that the Body-centered Cubic (BCC) lattice in the spatial domain results in the closest packing of spectra in the frequency domain, which in turn results in the sparsest arrangement of samples back in the spatial domain. The dual of the BCC is the Face-centered Cubic (FCC) lattice, which also provides a more efficient arrangement of samples than the cubic Cartesian (CC) lattice. In fact, the optimal sampling lattices allow for a reduction of sampling density by up to 14% in the 2D, 30%

## 1. INTRODUCTION

---

in the 3D and 50% in the 4D domains.

The use of more efficient sampling is one way to address the challenges of growing data sizes, but by itself, it is not enough. Volumetric data represents a discretized version of continuous objects and phenomena, where various factors limit the precision and the resolution. For example, when dealing with medical data, the resolution is determined by the capabilities of the medical scanner, while in Computational Fluid Dynamics (CFD), the resolution of the simulation domain may be determined by the available memory and the computational resources. However, the full resolution of the data is not necessary in areas of low frequency, and using a single fixed resolution can lead to sub-optimal representation of the underlying phenomenon. A number of multi-resolution techniques exist, such as octree domain refinement [74] and the flat brickling scheme [71]. The CC grid is used almost exclusively for multi-resolution volume rendering techniques and binary transitions on the CC exhibit persistency of sample position at neighboring levels of the hierarchy. This is a desirable property since it simplifies the communications between different levels of the hierarchy. However, the difference in sampling density between adjacent levels becomes very large, leading to visual artifacts during rendering. Although there are techniques that allow arbitrary resolution in the hierarchy, the relationships between the multiple levels of detail (LODs) of a single sample are lost. At the other end of the spectrum are the irregular grids [116], but their rendering is cumbersome because they lack regular structure that allows for efficient indexing. A mixed hierarchy that cycles through the CC, BCC and FCC lattices can be used to provide a more adaptive, more continuous and localized refinement compared to a CC-only hierarchy. The samples in such a structure would maintain the positional persistency while lowering the sampling density difference between adjacent levels of the hierarchy.

While the growth of computing capability and the development of new algorithms have allowed for increased problem sizes across many fields, the resolution of the traditional display devices has not improved significantly. We have developed a comprehensive approach that tackles the efficiency of the data representation through lattice-based techniques, as well as the visualization capabilities for exploring that data. One alternative to the workstation display is the class of immersive display devices that are often associated with Virtual Reality (VR). Head-mounted Displays (HMDs) and CAVEs immerse the user in a virtual environment, which has been shown to have a positive effect on certain tasks [92]. The depth perception allowed by these environments can be especially helpful when exploring data containing complex 3D structures, and especially volumetric data over large optimal sampling lattices. Traditional tiled displays are another alternative and generally provide significantly higher aggregate resolution (up to 300 megapixels) but with less immersion than CAVEs and HMDs. We have designed and constructed the Immersive Cabin and the Reality Deck facilities, along with a set of visualization techniques, to address the challenge of growing data sizes. The Reality Deck in particular combines the high resolution of the tiled displays with some of the immerge-

sive properties of the CAVE to create a unique large-data visualization environment with a state-of-the-art aggregate resolution of more than 1.5 gigapixels.

## 1.2 Contributions

### Simulation and rendering with optimal sampling lattices (Chapter 3)

We have developed a CFD simulation framework based on the lattice Boltzmann method (LBM) and using optimal sampling lattices. Our focus is specifically on the FCC lattice, which can achieve a stable simulation with only 13 lattice velocities while at the same time improving the sampling efficiency compared to using the Cartesian grid. We demonstrate the resulting fD3Q13 LBM model (13 lattice velocities with the FCC grid in 3D) for use in highly interactive smoke dispersion simulations. The simulation code is coupled with our visualization framework, which includes a high-performance volume renderer for data sampled on the CC, FCC and BCC lattices and support for virtual reality and immersive visualization systems.

### Mixed-lattice hierarchical structures (Chapter 4)

We have developed an LOD scheme for large volume data that allows for mixing the CC, FCC and BCC lattices in adjacent levels of the hierarchy. Our packing scheme stores the bricked data for the different lattice types in a unified cache on the GPU, allowing for more refined LOD transitions while the indexing cost is similar to that of the Cartesian grid octree.

### Immersive Environments (Chapter 5)

We have developed a visualization framework for the Immersive Cabin, which is a 5-sided CAVE with an automatic door. Our code supports the traditional virtual reality components, such as distributed rendering, stereo, tracking, and sound. The lattice-based visualization and simulation techniques, including the support for mixed-lattice hierarchies, are integrated in the framework and can be combined with the mesh rendering and the rigid-body physics simulation.

We have designed and constructed the Reality Deck, which is the world's first 1.5 gigapixel immersive display. The Reality Deck contains 416 high resolution LCD monitors in a 4-wall surround layout, and similarly to the Immersive Cabin, uses a unique automatic door that also holds a  $3 \times 5$  grid of monitors. The graphics are generated on an 18-node cluster with 24 displays connected to each node using the AMD Eyefinity technology. We have extended the Immersive Cabin visualization framework to support the Reality Deck and developed a new gigapixel image renderer targeted at scientific and immersive visualization.

## 1. INTRODUCTION

---

### Distributed Immersive Visualization Techniques (Chapter 6)

We have developed a new retargeting approach for partially-enclosed VR environments, such as the Immersive Cabin and the Reality Deck, to allow for the complete visualization of the data even if display surfaces are missing. Our technique uses conformal mapping to ensure that shape is preserved locally under the transformation and we demonstrate its use for the visualization of both mesh and lattice data.

We have developed a new frameless visualization technique in which the traditional framebuffer is replaced with reconstruction from a stream of rendered samples. This approach allows smooth user interaction even when rendering on a gigapixel display or when using computationally expensive visualization algorithms. Our system generates low-latency image samples using an optimal sampling lattice in the 2D+time space, as well as importance-driven higher quality samples.

We have developed the infinite canvas visualization technique for horizontally-enclosed visual environments. As the user moves through the physical space of the facility, the graphics outside of the field of view are continuously changed to create the illusion of an infinite continuous canvas. We also provide a spiral navigation interface, which presents a compressed view of large canvas segments. The infinite canvas has been used for the visual exploration of gigapixel datasets that are an order of magnitude larger than the surface area of the Reality Deck, as well as very large image collections.

# 2

## Background

### 2.1 Sampling Lattices

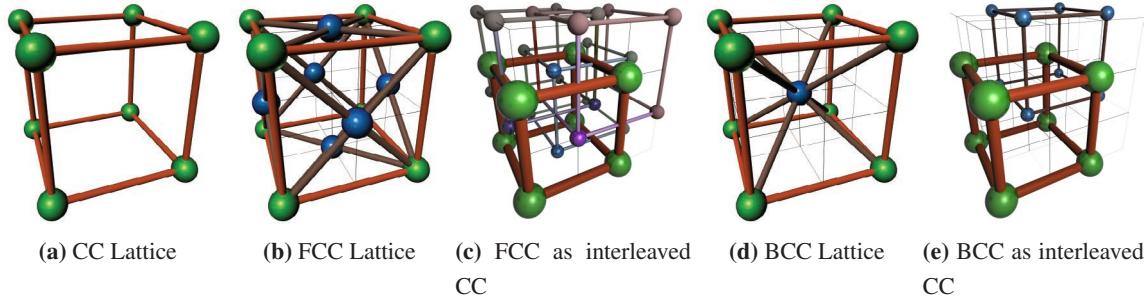
There are a number of definitions for the term *lattice* depending on the field of study, for example in physics, crystallography or electrical engineering. In mathematics, and depending on the sub-field, we have definitions as partially ordered sets and as algebraic structures. However, for the use of describing sampling grids in the fields of computer graphics and visualization, the geometric definition of the Bravais lattice is more suitable. The Bravais lattice is an infinite collection of discrete points that can be described by linear combinations over a set of basis vectors with integer coefficients. There are a total of 5 Bravais lattices in 2D and 14 in 3D. The rectangular in 2D and the Cartesian Cubic (CC) in 3D are the most often used sampling lattices, however our focus is also on the remaining cubic lattices in 3D - Body-Centered Cubic (BCC) and Face-Centered Cubic (FCC). We further consider the hexagonal lattice in 2D and the Hexagonal Close-Pack (HCP) in 3D, although the HCP is not a Bravais lattice and its complex layer structure is computationally more expensive to handle. For the remainder of the text, we will refer to these structures simply as *lattices*.

The Voronoi and unit cells of a lattice position are an important feature for the design of reconstruction filters for the different sampling grids. Both of these structures are constant across a given lattice, and therefore we can refer to a single Voronoi cell and unit cell. In 3D, the CC lattice is the most widely used regular arrangement with well-understood sampling properties and it is generated by the basis vectors  $(1, 0, 0)$ ,  $(0, 1, 0)$  and  $(0, 0, 1)$ . Its layout maps efficiently to the memory layouts of modern computer architectures in both 2D and 3D. However, one disadvantage of the CC lattice is that it provides a sub-optimal sampling of the domain compared to HCP, FCC and BCC.

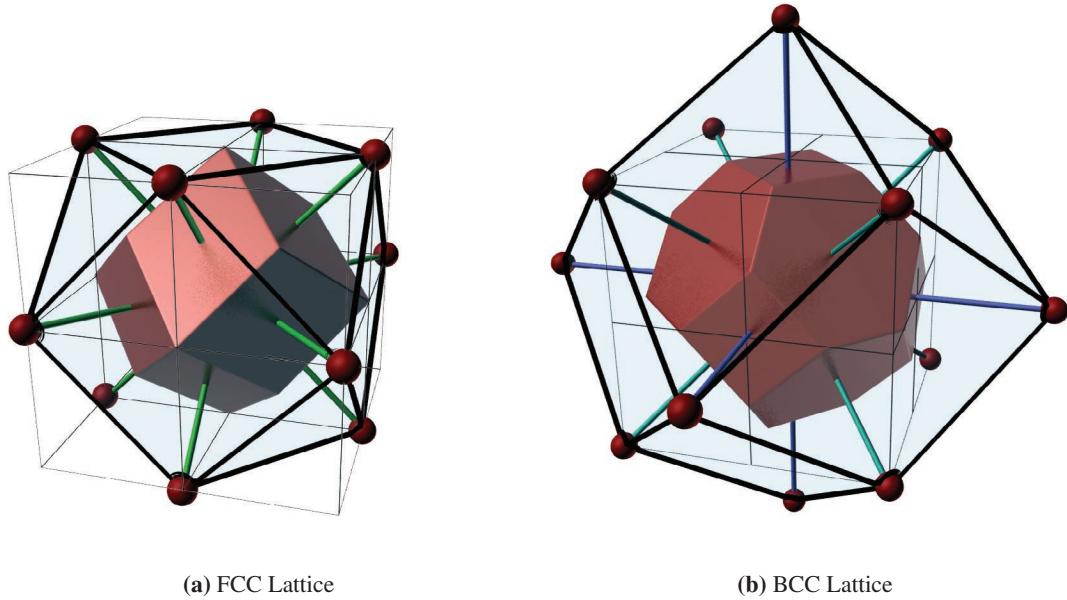
Figure 2.1 illustrates the construction of the three lattice types. The FCC lattice is non-uniquely defined by the generating vectors  $(1, 1, 0)$ ,  $(1, 0, 1)$  and  $(0, 1, 1)$ . A more intuitive construction starts with the 8

## 2. BACKGROUND

---



**Figure 2.1:** The BCC lattice can be constructed by (d) adding one lattice site at the center of every cell of the CC lattice (a), or by (e) interleaving two CC lattices. The FCC lattice can be constructed by (b) placing sites at the face centers of the CC unit cell, or by (c) interleaving four CC lattices.



**Figure 2.2:** Unit cell and Voronoi cell for the (a) FCC and (b) BCC lattices. Unit cells (illustrated with the black edges) are the cuboctahedron and the rhombic dodecahedron, respectively. The Voronoi cells (colored in red) are the rhombic dodecahedron and the truncated octahedron, respectively.

## 2.1 Sampling Lattices

---

sampling positions defined on the CC lattice and adds a new sampling position at the face centers of each CC unit cell face. This construction is illustrated in Figure 2.1b. The FCC lattice achieves the optimal sphere packing in 3D [20]. Another optimal sphere packing in 3D is the asymmetric HCP lattice, which can be constructed from interleaved layers of the 2D hexagonal lattice. However, the HCP is not a Bravais lattice and its complex layer structure is difficult to map to the CPU and GPU memory layouts. Similarly to the hexagonal lattice in 2D, the FCC is also the lattice with the maximum *kissing number*, which means that each lattice site has the maximum possible number of equidistant closest neighbors. For example, on the Cartesian grid, each grid point has 6 nearest neighbors at unit distance, 12 secondary neighbors at distance  $\sqrt{2}$  and 8 tertiary neighbors at distance  $\sqrt{3}$ , while the FCC has only 12 nearest neighbors. This maximal number of first-degree neighbors, also known as the *kissing number*, is related to the problem of packing spheres in 3D space. As shown in Figure 2.2a, the 12 neighbors of an FCC lattice site form a cuboctahedron, and the Voronoi cell is a rhombic dodecahedron.

The generating vectors for the BCC lattice are  $(1, 1, -1)$ ,  $(1, -1, 1)$  and  $(-1, 1, 1)$ . This construction is equivalent to adding an extra site at each cell of the CC lattice, or to interleaving 2 CC lattices with a  $(1, 1, 1)$  offset. In BCC, each site has 8 nearest neighbors and 6 secondary neighbors, and the unit cell is a rhombic dodecahedron, as shown in Figure 2.2b. Thus, BCC and FCC are duals of each other and the Voronoi cell in BCC is a truncated octahedron. The faces on the truncated octahedron are 8 regular hexagons (corresponding to the 8 nearest neighbors) and 6 squares (corresponding to the 6 secondary neighbors). BCC is the optimal lattice for sampling in 3D, since its dual in the Fourier domain (the FCC) achieves the tightest packing of spectra. Conversely, the dual of the FCC in the Fourier domain is the BCC, which gives a near-optimal sphere packing. Therefore, the FCC in the spatial domain exhibits near-optimal sampling properties. In fact, the BCC lattice allows for a reduction of sampling density by 29.3% in 3D compared to CC for a fixed signal frequency [106].

A number of lattice-based techniques have been developed in the fields of visualization and physical simulations. With the continuous improvements to GPU speeds, GPU memory sizes and bandwidths, techniques such as the volumetric global illumination over the FCC lattice [88] and the lattice Boltzmann method (LBM) [125] become applicable to a variety of interactive and realtime visualization tasks. In particular, LBM is a mesoscopic method that traces the collision and streaming of particle distributions along a lattice instead of tracing individual particles. Traditionally, LBM uses the CC lattice with different neighbor sets (e.g., 19 neighbors in D3Q19 LBM), but recently GPU implementations of the LBM have been developed over the optimal sampling lattice, including the D3bQ15 LBM for the BCC lattice [1]. Simple simulation domain refinements for LBM on the GPU have been demonstrated in the context of CC simulations [131], but not for the optimal sampling lattices.

## 2. BACKGROUND

---

### 2.2 Lattice Reconstruction

There is a very rich literature on signal processing and reconstruction techniques for the CC lattice; however, the same is not true for the BCC and FCC. A number of higher order filters have been proposed for rendering on the BCC and FCC lattices based on box splines [33, 36, 61]. Although these filters produce excellent reconstruction results, their applicability is limited due to the high computational cost even when a GPU-accelerated version is available for volume rendering. In practice, the linear filters defined over CC, BCC and FCC are often used for realtime visualization applications. For the Cartesian grid in particular, the native trilinear interpolation on the GPU provides the best performance.

The BCC is the optimal sampling lattice in 3D, and there exist efficient reconstruction algorithms for its grid structure. The unit cell of the BCC lattice is the rhombic dodecahedron, which is a zonohedron, and in a general position it covers exactly 3 nearest BCC sites in addition to the central site. The linear element for the unit cell is a function that evaluates to 1 at a lattice position and has a linear fall-off to 0 at the first degree nearest neighbor positions. For the unit cell of the BCC lattice, the linear element can be defined as the intersection of a set of half-spaces [34]:

$$L_{BCC}(x, y, z) = \frac{1}{2} \max(0, 2 - \max(|x| + |y|, |x| + |z|, |y| + |z|)). \quad (2.1)$$

where the point  $(x, y, z)$  is in lattice coordinates. Finkbeiner et al. [41, 42] present an efficient GPU implementation of the linear and quintic box splines for BCC, where the reconstructions exhibit  $C^0$  and  $C^2$  continuity respectively. The linear box spline is equivalent to BCC barycentric interpolation and can be implemented with only 4 nearest neighbor (NN) texture fetches. The quintic box spline has a larger support that covers 32 lattice positions and is significantly more expensive to compute.

The FCC lattice is significantly more difficult to render directly. Its linear element is defined similarly over the cuboctahedron [87]:

$$L_{FCC}(x, y, z) = \max\left(0, 1 - \max\left(|x|, |y|, |z|, \frac{1}{2}(|x| + |y| + |z|)\right)\right). \quad (2.2)$$

In a general position, up to 3 points from the 12 point neighborhood fall inside the unit cell of the FCC lattice, in addition to the central node. However, the cuboctahedron is not a zonohedron since it contains triangular faces. As a result we cannot construct a linear box spline element with a cuboctahedron support. This can be illustrated by translating the cuboctahedron by a permutation of  $(\pm 1, 0, 0)$ . The unit cell at those positions contains no lattice sites, and therefore all the weights defined by Equation 2.2 are 0. The unit cell can be extended to include the 6 neighbors along the major axes and becomes the truncated octahedron, which contains 6 square and 8 hexagonal faces, all of which exhibit point symmetry. The linear element can then be defined as:

$$L_{FCC}(x, y, z) = \frac{1}{2} \max \left( 0, 2 - \max \left( |x|, |y|, |z|, \frac{1}{3} (|x| + |y| + |z|) \right) \right). \quad (2.3)$$

For any given point in 3D, the extended support of the filter includes the nearest lattice position, its 12 nearest neighbors and up to 3 of the 6 secondary neighbors along the major axes. These 16 FCC points become the basis for a 6-directional  $C^1$  box spline [33]. Although this spline and its 9-directional extension ( $C^3$  reconstruction with 40 FCC points in the support) can produce high quality reconstructions, their evaluation is computationally expensive [61] and there is no known GPU implementation. Instead, the weights for the 16 FCC points inside the truncated octahedron can be evaluated using the linear element defined in Equation 2.3.

## 2.3 Multi-resolution Techniques for Volumetric Data

One of the early approaches for multi-resolution volume representation on the GPU was proposed by LaMar et al. [74] who use an octree domain refinement scheme. This hierarchical approach splits the original volume into a set of fixed-size bricks and a distance-based heuristic determines the span of each brick in the original volume. In contrast, later methods introduce a flat bricking schemes that reduces the size of the bricks in the coarser level of the hierarchy [71]. Crassin et al. [23] have proposed an efficient multi-resolution rendering framework in which the LOD selection is guided by the raycasting phase of the rendering. In terms of the optimal sampling lattices, Entezari et al. use the CC, FCC and BCC lattice for downsampling of volume data [35] for the purpose of building a multi-resolution representation with a fan-out of 2. However, their approach does not allow the different representations to be used in a single mixed-lattice hierarchy.

One of the challenges in rendering multi-resolution volumes is handling the boundary between bricks of different resolution. Ljung et al. [71] have proposed an interblock interpolation technique that allows smooth transition between bricks of arbitrary resolutions without duplicating the brick boundaries. A common approach on the CC lattice is in fact to duplicate boundary regions and Beyer et al. [9] have proposed a smooth interpolation filter with  $C^0$  continuity of the reconstruction across the resolution boundaries. In our approach, we enforce smooth transitions between the LODs, which cycle through the CC, BCC and FCC lattices, and the introduction of new sampling positions is guided by a set of simple rules. As a result, the boundary structures are well-defined, allowing for efficient interpolation. The mixed lattice representation can be used with both hierarchical and flat brickling schemes; however, the hierarchy approach simplifies the management of the index texture and the brick cache.

## 2. BACKGROUND

---

### 2.4 The Lattice Boltzmann Method

#### 2.4.1 Definition

The Lattice Boltzmann method (LBM) is a grid-based model that is used to compute macroscopic fluid behavior by simulating mesoscopic particle interactions [125]. The approach is similar to the previously employed lattice-gas cellular Automata (LGCA), which as the name suggests models fluid flows in a manner similar to a cellular automaton. In the lattice-based approach, we describe the mesoscopic behavior by particle collisions at lattice sites and propagations along lattice links. The LGCA method models single particle Boolean dynamics, which leads to statistical noise. The LBM overcomes this limitation by tracing particle density distributions and the average effect of particle collisions. In the limit, as the lattice pitch approaches 0, the system solves the Navier-Stokes differential equation for incompressible fluids.

In the LBM, we have a lattice, which discretizes the spatial domain into a regular arrangement of lattice sites  $\mathbf{x}$ . Also, particle velocities  $\mathbf{u}$  are represented by a finite set of lattice velocities denoted by  $\mathbf{c}_i$ . The continuous distribution function  $f(\mathbf{x}, \mathbf{u}, t)$  is therefore represented by a set of distributions  $f_i(\mathbf{x}, t)$ , which assign a particle density value to each of the discrete lattice velocities for a site  $\mathbf{x}$  at time  $t$ . For a given state of the particle densities, the next discrete state of the simulation is described by the discrete Lattice Boltzmann equation:

$$f_i(\mathbf{x} + \mathbf{c}_i \Delta t, t + \Delta t) - f_i(\mathbf{x}, t) = -\frac{1}{\tau} (f_i^{eq} - f_i). \quad (2.4)$$

In essence, this formula describes a simple collision and propagation rule. At each step of the simulation, and for each lattice site, we examine the particle distributions, calculate the distribution contributions from particle collision, and synchronously propagate the new densities along the lattice links. Note that here we employ the BGK (Bhatnagar-Gross-Krook) approximation to the collision integral, which describes how the particle densities are affected by the collision. The BGK approximation models collisions as a relaxation of momentum toward an equilibrium state defined by the distributions  $f_i^{eq}$  and conserves mass and momentum in the system. The constant  $\tau$  controls the relaxation rate and its value is derived from the kinematic viscosity of the fluid  $\nu$ :

$$\nu = \frac{1}{2} c_s^2 (2\tau - 1) \Rightarrow \tau = \frac{\nu}{c_s^2} + \frac{1}{2}. \quad (2.5)$$

In this equation,  $c_s$  is the lattice speed of sound. This value determines the highest possible speed for information propagation and is specific to the selected lattice structure. Negative values for the distributions  $f_i$  quickly lead to numeric instability and therefore stable simulations can be achieved only for a limited range of  $\tau$  values. At each step of the simulation, we can recover the macroscopic properties of the flow from the mesoscopic particle distributions at lattice sites:

$$\rho(\mathbf{x}, t) = \sum_i f_i(\mathbf{x}, t) \quad (2.6)$$

$$j(\mathbf{x}, t) = \rho(\mathbf{x}, t) \nu(\mathbf{x}, t) = \sum_i \mathbf{c}_i f_i(\mathbf{x}, t). \quad (2.7)$$

In these equations,  $\rho$  is the mass density at a specific site location  $\mathbf{x}$  and time  $t$ , and  $\mathbf{j}$  is the momentum density. We can then calculate the particle velocity:

$$\mathbf{u}(\mathbf{x}, t) = \frac{\mathbf{j}(\mathbf{x}, t)}{\rho(\mathbf{x}, t)} = \frac{1}{\rho(\mathbf{x}, t)} \sum_i \mathbf{c}_i f_i(\mathbf{x}, t). \quad (2.8)$$

The equilibrium state used in the BGK collision model depends only on the conserved quantities, namely mass and momentum, and is described by the following:

$$f_i^{eq} = W_i \rho \left( 1 + \frac{\mathbf{c}_i \cdot \mathbf{u}}{c_s^2} + \frac{(\mathbf{c}_i \cdot \mathbf{u})^2}{2c_s^4} - \frac{\mathbf{u} \cdot \mathbf{u}}{2c_s^2} \right). \quad (2.9)$$

Here,  $c_s$  is the speed of sound for the lattice,  $\mathbf{u}$  is the particle velocity calculated from Equation 2.8 and each lattice velocity  $\mathbf{c}_i$  has an associated weight  $W_i$ . These weights and the lattice speed of sound are specific to the selected lattice geometry. Their values are obtained from the non-vanishing elements of the even velocity moments up to fourth order:

$$\begin{aligned} \sum_i W_i &= \rho_0 \\ \sum_i W_i c_{i\alpha} c_{i\beta} &= \rho_0 c_s^2 \delta_{\alpha\beta} \\ \sum_i W_i c_{i\alpha} c_{i\beta} c_{i\gamma} c_{i\delta} &= \rho_0 c_s^4 (\delta_{\alpha\beta} \delta_{\gamma\delta} + \delta_{\alpha\gamma} \delta_{\beta\delta} + \delta_{\alpha\delta} \delta_{\beta\gamma}). \end{aligned} \quad (2.10)$$

The calculated weights  $W_i$  ensure that: (1) the mass density is positive, and (2) the lattice velocity moments up to fourth order are identical to the respective velocity moments over the Maxwell distribution [125]. Vanishing velocities have no influence on the isotropy of the lattice tensors. In these equations,  $\rho_0$  is the constant mass density for incompressible fluids, which is usually set to 1, and  $\delta_{ij}$  is the Kronecker delta symbol:

$$\delta_{ij} = \begin{cases} 1 & \text{if } i = j; \\ 0 & \text{if } i \neq j; \end{cases}. \quad (2.11)$$

Note that the odd velocity moments vanish and do not provide any additional constraints. Each of the Greek letters  $\alpha$ ,  $\beta$ ,  $\gamma$  and  $\delta$  corresponds to a Cartesian index of the lattice velocity, such as 1 or 2 in the

## 2. BACKGROUND

---

case of 2D LBM and 1, 2 or 3 in the case of LBM in 3D. Detailed derivations and analysis of the LBM on frequently used lattices can be found in the work of Wolf-Gladrow [125] and Chen and Doolen [17].

While the LBM can model complex fluid systems with a set of simple collision and propagation rules, it suffers from a few limitations. It cannot easily handle the high velocity flows used in aerodynamic simulations, for example. The physical system under consideration is made dimensionless by the selection of a length scale and a time scale, and is further discretized using an appropriate lattice step and a time step. The lattice viscosity depends on the discretization parameters and restricts the range of flow velocities that lead to stable simulations. Decreasing the lattice spacing and the time step increases the viscosity used in the LBM and allows for stable simulations with larger physical velocities (see [67] for additional information). However, simulating high Reynolds number flows can be prohibitively expensive in terms of both memory requirements and computational time. Hou et al. [51] proposed a sub-grid model to deal with high Reynolds Number flows. Also, the single-relaxation-time constraint imposed by the BGK collision model leads to numerical instabilities when the simulation is coupled with heat transfer or body forces. The multiple-relaxation-time LBM (MRT-LBM) performs the collisions in moment space as opposed to the single-relaxation-time LBM (SRT-LBM) collision in the distributions space, and each moment is given an independent relaxation parameter [29]. After the relaxation, the inverse transform is used and the propagation rules are applied in phase space, or the space of the distributions. This approach exhibits improved numerical stability and has been coupled with heat transfer in the hybrid thermal Lattice Boltzmann Model (HTLBM) [66].

### 2.4.2 Boundary Conditions

In many simulations, the fluid interacts with objects whose boundaries may not follow the links in the discrete lattice structure, for example when the object is defined procedurally or through a finite mesh. Different schemes for handling boundary conditions can be implemented by simply manipulating the particle densities during the collision step. This simplicity is offset by the fact that any error in the scheme propagates throughout the simulation domain. The bounceback scheme is often used to implement no-slip boundary conditions [125]. The boundary is discretized over the lattice nodes and is assumed to intersect lattice links at the halfway point. Densities propagating along those links are simply moved to links in the opposite direction. This approach is very efficient in handling boundaries aligned with the lattice links, but leads to inaccuracies with curved boundaries. Different boundary conditions exist for LBM [73, 76, 114] including schemes that exhibit second order accuracy for the boundary representation of curved surfaces.

In the 2D case, we employ the scheme proposed by Bouzidi et al. [12] for the treatment of curved boundary conditions. The curved boundary is illustrated in Figure 2.3. The scheme combines the bounce-

back approach with linear interpolation along lattice links to achieve second-order accuracy of the boundary condition:

$$f_{i'}(\mathbf{x}_f, t+1) = \begin{cases} 2\Delta f_i^c(\mathbf{x}_f, t) + (1 - 2\Delta) f_i^c(\mathbf{x}_f - \mathbf{c}_i, t) & \text{if } \Delta < \frac{1}{2}; \\ \frac{1}{2\Delta} f_i^c(\mathbf{x}_f, t) + \frac{2\Delta-1}{2\Delta} f_i^c(\mathbf{x}_f, t) & \text{if } \Delta \geq \frac{1}{2}. \end{cases} \quad (2.12)$$

Here,  $f_i^c$  denotes the particle distribution along link  $i$  after the collision step, but before propagation, and  $\Delta = \frac{\|\mathbf{x}_f - \mathbf{x}_w\|}{\|\mathbf{x}_f - \mathbf{x}_b\|}$ . Our 3D simulations implement a different, computationally more expensive scheme based on linear extrapolation [76] with the following streaming rule:

$$\begin{aligned} f_{i'}(\mathbf{x}_f, t+1) &= (1 - \chi) f_i^c(\mathbf{x}_f, t) + \chi f_i^{feq}(\mathbf{x}_b, t) + 2W_i \rho \frac{\mathbf{c}_{i'} \cdot \mathbf{u}_w}{c_s^2} \\ f_i^{feq}(\mathbf{x}_b, t) &= W_i \rho(\mathbf{x}_f, t) \left[ 1 + \frac{\mathbf{c}_i \cdot \mathbf{u}_{bf}}{c_s^2} + \frac{(\mathbf{c}_i \cdot \mathbf{u}_f)^2}{2c_s^4} - \frac{\mathbf{u}_f \cdot \mathbf{u}_f}{2c_s^2} \right], \end{aligned} \quad (2.13)$$

where  $f_i^{feq}(\mathbf{x}_b, t)$  is the equilibrium distribution for the fictitious fluid node  $\mathbf{x}_b$  within the boundary of the object, weighted by the extrapolation (or interpolation) factor  $\chi$ . The fluid velocity at the wall position  $\mathbf{x}_w$  is denoted as  $\mathbf{u}_w$  and equals the velocity of the boundary object. The extrapolated fluid velocity  $\mathbf{u}_{bf}$  along lattice link  $\mathbf{c}_i$  is not uniquely defined. Mei et al. propose the following choices [76], which retain the second-order accuracy of the boundary condition treatment and improve the computational stability:

$$\begin{aligned} \mathbf{u}_{bf} &= \begin{cases} \frac{\Delta-1}{\Delta} \mathbf{u}_f + \frac{1}{\Delta} \mathbf{u}_w & \text{if } \Delta \geq \frac{1}{2}; \\ \mathbf{u}_f & \text{if } \Delta < \frac{1}{2}. \end{cases} \\ \chi &= \begin{cases} \frac{2\Delta-1}{\tau} & \text{if } \Delta \geq \frac{1}{2}; \\ \frac{2\Delta-1}{\tau-2} & \text{if } \Delta < \frac{1}{2}. \end{cases} \end{aligned} \quad (2.14)$$

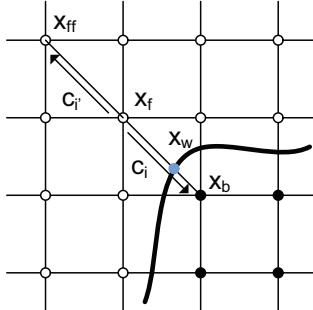
The value of  $\Delta$  is again defined as  $\Delta = \frac{\|\mathbf{x}_f - \mathbf{x}_w\|}{\|\mathbf{x}_f - \mathbf{x}_b\|}$ , which is illustrated in Figure 2.3 for the 2D case.

## 2.5 Immersive Visualization

Immersive visualization systems allow the user to explore data in novel ways that go beyond the standard 2D images on a workstation. This is particularly important when trying to analyze data sampled on a 3D lattice or the results of 3D LBM simulations where a flat 2D display presents a sub-optimal view into the volumetric data. The immersive platforms provide a superior depiction of the information via a significantly wider field of view and enhanced depth and shape perception. The first such environment, the CAVE [24], offers an immersive experience using back-projected images on 3 walls and front projection on the floor. Other display arrangements have been proposed, including the 5-sided Immersive Cabin (IC) [89] and the

## 2. BACKGROUND

---



**Figure 2.3:** Curved boundary treatment. The filled circles represent the discretized geometric object. The clear circles represent fluid cells. The blue circle at position  $x_w$  is the intersection point between the surface of the geometric object and a lattice link.

6-sided CAVE [47]. Compared to Head-Mounted Displays (HMD), these environments provide a more natural visualization and allow users to interact with visualization-augmented physical objects, following the paradigm of *Augmented Virtuality* [78], or more generally, *Mixed Reality*. However, building fully-immersive facilities is expensive and introduces many challenges in terms of head tracking, sound systems and even air circulation.

A variety of retargeting techniques can be applied for immersive visualization. Focus and Context (F+C) techniques such as magic lenses [85, 120, 127] are traditionally designed for a single projection surface and do not translate directly to immersive environments with multiple non-planar displays. Illustrative deformations for data exploration have been proposed [22] which could be used to warp data that lies around the CAVE volume, however they do not provide any guarantee in regard to shape preservation. The technique presented by Lorenz and Döllner [72] handles piecewise approximation of non-planar perspective projections on the GPU. Non-planar projections can be used to define a projection surface that *wraps* around the CAVE, including part of the ceiling, and thus recovering the non-visualized sections of the data. The technique can be applied in either image-space or geometry-space, however in the first case the sampling artifacts can impact visual quality significantly [110], while the geometry approach does not scale well with mesh density [72].

Virtual Colonoscopy (VC) has been established as a non-invasive alternative to traditional Optical Colonoscopy (OC) for cancer screening [49, 58]. A VC session involves the acquisition of Computed Tomography (CT) scans of the patient’s abdomen and the extraction and visualization of the colon surface via segmentation and volume rendering techniques. Traditional VC covers only about 91% of the colon surface after full navigation in both the antegrade and the retrograde directions [50] and the percentage is significantly lower for a single direction (about 75%). One shortcoming of existing partially-immersive display configurations for Immersive Virtual Colonoscopy (IVC) is that the missing display surfaces may hide a sig-

nificant amount of information. While this may be acceptable in certain applications, it becomes a critical issue for the exploration of medical data that must be handled by the visualization system.

## 2.6 Frameless Rendering

As the resolutions of both data and displays increase, a number of challenges arise related to building high performance visualization systems. One fundamental problem is that the majority of the existing techniques are frame-based - the system must prepare a complete grid of pixels which are then displayed to the user. In contrast, the frameless rendering introduced by Bishop et al. [11] implements a new method for generating images from stochastic distributions of samples as compared to using grids of pixels. The most notable advantage of this type of system is that the rendering is decoupled from the display and therefore it is not necessary to wait for a full frame to be completed before the results can be shown to the user. This is particularly effective when the rendering process is expensive, such as in the traditional CPU raytracing or in the volumetric rendering of lattice data. More recently, Dayal et al. introduced the concept of Adaptive Frameless Rendering [25] which uses adaptive spatial and temporal filters to improve the quality of the reconstructed images and to minimize the motion artifacts.

A number of researchers have developed sample reprojection methods which reposition samples from the previous frame as a way of reducing the computational workload. In the Render Cache [118, 119], samples are reprojected at each frame to account for the motion of the camera and the final image is reconstructed from those samples. There are distributed [6] and GPU implementations [113, 132] of the RenderCache, as well as other GPU-friendly techniques for efficiently reusing old samples [97, 98].

Others have implemented different techniques for decoupling the rendering from the display. Interruptible rendering [126] is a progressive rendering framework where a coarse framed image is continually refined, optimizing for both spatial and temporal error, before being sent to the display device. Hauswiesner et al. have introduced an image-based multi-frame rate rendering approach [46] where the workload is spread over multiple GPUs working asynchronously. This allows the main display to provide high frame rates and forward image warping is used to minimize artifacts due to the varying latency.

## **2. BACKGROUND**

---

# 3

## LBM Fluid Simulation and Visualization on Optimal Sampling Lattices

### 3.1 Introduction

The lattice Boltzmann method (LBM) for visual simulation of fluid flow generally employs cubic Cartesian (CC) lattices such as the D3Q13 and D3Q19 lattices for the particle transport (13 and 19 lattice velocities respectively over the Cartesian grid in 3D). However, the CC lattices lead to sub-optimal representation of the simulation space. We introduce the face-centered cubic (FCC) lattice with 13 lattice velocities, fD3Q13, for LBM simulations. Compared to the CC lattices, the fD3Q13 lattice creates a more isotropic sampling of the simulation domain and its single lattice speed (i.e., link length) simplifies the computations and data storage. Furthermore, the fD3Q13 lattice can be decomposed into two independent interleaved lattices, one of which can be discarded, which doubles the simulation speed. The resulting LBM simulation can be efficiently mapped to the GPU, further increasing the computational performance. We show the numerical advantages of the FCC lattice on channeled flow in 2D and the flow-past-a-sphere benchmark in 3D. In both cases, the comparison is against the corresponding CC lattices using the analytical solutions for the systems as well as velocity field visualizations. We also demonstrate the performance advantages of the fD3Q13 lattice for interactive simulation and rendering of hot smoke in an urban environment using thermal LBM.

When it comes to regular gridded sample arrangements in both space and time, the Cartesian lattice has been the *lingua franca* in science, medicine, and engineering. It is simple, straightforward to index, and easy to collect data on. However, recent years have seen an increasing move toward other regular lattices hailed to be more space efficient and thus more optimal in their distribution of samples. These proposals were based on the recognition that an optimal packing of these lattices in the frequency domain yields the sparsest sample arrangement in the spatial domain, assuming a near-spherical frequency spectrum. This finding

### **3. LBM FLUID SIMULATION AND VISUALIZATION ON OPTIMAL SAMPLING LATTICES**

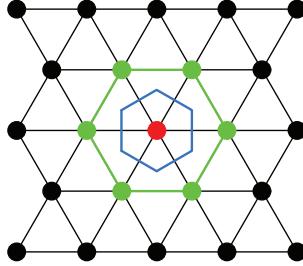
---

emerges from the application of the Fourier sampling theorem and also the theory of lattice reciprocity. The one lattice that achieves this sparse sampling is the BCC (Body Centered Cubic) lattice. It was elaborately shown that the BCC lattice affords a 30% reduction in the number of samples, or a 30% increase of resolution in the spatial domain, a number that grows to 50% when extended into 4D. The reciprocal lattice to the BCC is the FCC lattice, which is the lattice that exhibits the tightest packing. However, this tight packing has other implications as well. It also produces the most isotropic sample distribution, yet not the sparsest. This property, while often neglected, is very desirable in any type of communication scenarios and also for digital object representation. It affords the best optimal orientation independency in the coverage of space and time.

We claim that the FCC lattice is the perfect medium to conduct simulations that use communications over lattice links to propagate energy within the simulation medium. At the same time we also claim that this lattice is ideal to model interactions of the flow phenomena with scene objects, since, afforded by this isotropy, this lattice is most insensitive to how scene objects came to rest with respect to the orientation and organization of the lattice. Interestingly enough, it is this isotropy that subsequently affords a more efficient communication in these simulations which leads to the possibility to dispense with some of the lattice links without harm, and thus save on costly communication computations. We employ the FCC lattice in the context of flow simulation and visualization via the Lattice Boltzmann method which has become a popular method for physically-based simulations.

In recent years, the importance of physically-based simulations has been growing in fields such as computer games, movie productions and product design testing, where there is a need for computer generated realism at interactive rates. Simulations based on the finite element method are expensive and unsuitable for interactive application while lattice-based simulations have emerged as an efficient and easy to implement alternative. In particular, the LBM is a powerful technique that can be used to model complex fluid flows [125]. It uses a simple set of collision and propagation rules that are applied independently at a mesoscopic level on particle distributions in the lattice. These local distributions can then be used to model the global behavior of the fluid flow, solving the Navier-Stokes equations in the incompressible limit.

We adopt the notation  $DmQn$  used by physicists to describe lattices, where  $m$  represents the number of spatial dimension and  $n$  is the number of velocity links at each lattice site, including a zero-velocity link. Traditionally, LBM simulations have been performed on the CC lattices where the D3Q19 lattice provides a good balance between stability and computational/memory cost. Its properties and limitations are well known [125] and its cubic grid structure maps conveniently to memory layouts on both CPU and GPU architectures. At the same time, its high number of links and multiple link speeds introduce additional complexity in the LBM simulation, increase communication overhead across the links, and therefore decrease performance. Based on the arguments of isotropy raised above, we propose the use of the fD3Q13 lattice for highly efficient LBM simulations.



**Figure 3.1:** Hexagonal 2D lattice, D2Q7. The 6 nearest neighboring sites (green dots) of a site (red dots) form a regular hexagon. The Voronoi cell (blue) is also a regular hexagon.

The low number of single-speed links and the efficient node indexing on the fD3Q13 lattice greatly simplify and accelerate the simulation. The LBM is already amenable to GPU acceleration because of the highly local computations and the use of the fD3Q13 lattice further increases the speed of GPU-based implementations.

## 3.2 The fD3Q13 Lattice

In LBM, both space and velocity are discretized [125]. It is expected that the spatial discretization is smoothed on large enough scales compared to the grid size. However, it is not obvious whether the coarse angular discretization of the velocity space on the lattice links correctly models the macroscopic fluid behavior. It has been proven that the LBM recovers the Navier-Stokes equations at the incompressible limits through Chapman-Enskog expansion [125]. In this multi-scale analysis, sufficient lattice symmetry is necessary to guarantee the isotropy of 2<sup>nd</sup> and 4<sup>th</sup> rank lattice tensors. The generalized lattice tensor is defined by:

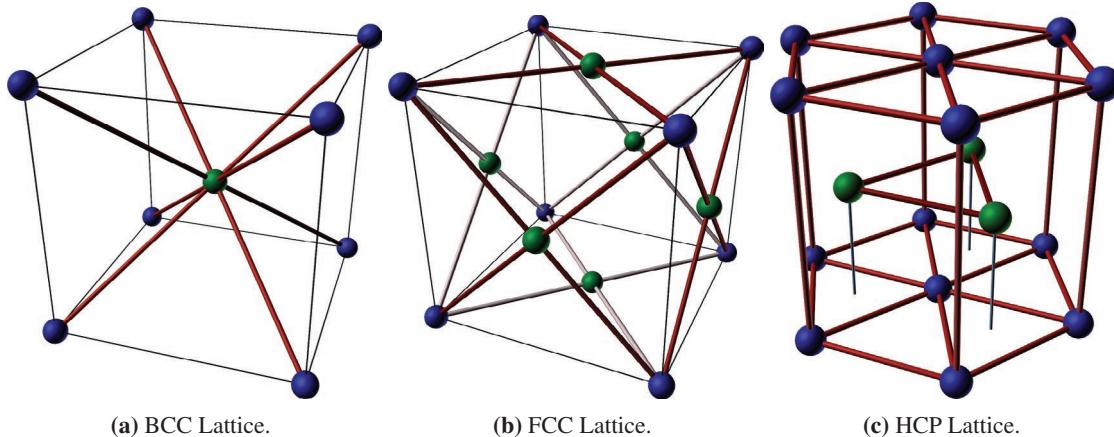
$$G_{\alpha_1 \alpha_2 \dots \alpha_n} = \sum_i W_i c_{i\alpha_1} c_{i\alpha_2} \dots c_{i\alpha_n}. \quad (3.1)$$

Here,  $c_{i\alpha_j}$  is the  $\alpha_j$ -th Cartesian component of the lattice velocity  $\mathbf{c}_i$ . From this formulation, it follows directly that velocities with the same vector length have the same weights. This can also be proven by solving Equation 2.10. In single speed models such as D2Q7 on the hexagonal lattice and D3Q13 on the Cartesian cubic (CC) lattice, all the lattice links have the same length (1 and  $\sqrt{2}$  respectively), and therefore all non-zero velocities have the same weight.

The D2Q7 LBM is built on the hexagonal lattice, also called the triangular lattice. As shown in Figure 3.1, every lattice site in the hexagonal lattice is linked to 6 nearest neighboring sites. The 6 neighbors form a regular hexagon, and the Voronoi cell is also a regular hexagon. It has been proven that the hexagonal lattice is the optimal sphere packing in 2D [20]. Moreover, the hexagonal lattice has the maximum kissing

### 3. LBM FLUID SIMULATION AND VISUALIZATION ON OPTIMAL SAMPLING LATTICES

---

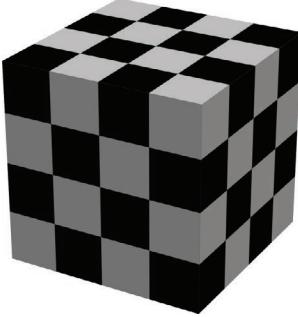


**Figure 3.2:** A 3D BCC lattice (a) can be constructed by adding one lattice site at the center of every cell of the CC lattice. A 3D FCC lattice (b) can be constructed by placing sites at the centers of the square surfaces of every cell of the CC lattice. A 3D HCP lattice (c) can be constructed by layering 2D HCP lattices. In (c), the blue sites form the 2D HCP lattice, which is the first layer of the 3D HCP lattice. The green sites represent the second layer. The entire 3D HCP lattice contains alternating blue and green layers.

number (or the number of nearest neighbors) in 2D, which means it has the maximum number of velocity vectors of all possible single speed 2D LBM. It was used in the first LGCA model that has been proven to recover the Navier-Stokes equation [43]. In comparison, the D2Q9 LBM is built on the Cartesian lattice, where every lattice site is connected to 4 nearest neighbors and 4 secondary neighbors. Because of the different link lengths (1 for nearest neighbors and  $\sqrt{2}$  for secondary neighbors), the weights  $W_i$  used for the calculation of equilibrium distributions in Equation 2.9 are also different.

In 3D, many lattices can be considered for LBM, such as the CC, BCC, FCC, and HCP lattices (Figure 3.2 shows the construction for the latter three, see also Section 2.1). Traditionally, the D3Q13, D3Q15 and D3Q19 LBM are the most frequently used models, all based on the CC lattice. The difference is the choice of lattice links. In the D3Q13 lattice, links connect only secondary neighbors. In the D3Q15 lattice, sites are connected to primary and tertiary neighbors and in the D3Q19 lattice, they are connected to primary and secondary neighbors. The CC lattice is prevailing because of its simplicity and because researchers have extensive knowledge of its properties. The FCC and BCC lattices have not been fully explored as the underlying lattice structures for LBM simulations.

In the D3Q13 LBM, it is observed that the lattice sites can be divided into two distinct half-sets [28]. One set contains all the sites where the sum of index components is even, which we call even sites. The complementary set contains all the odd sites and there does not exist any link between an odd site and an even site. The partitioning is illustrated in Figure 3.3. Based on this observation, it is clear that an LBM simulation over a D3Q13 lattice can be decomposed into two independent simulations, each executed on an



**Figure 3.3:** A  $4^3$  CC lattice divided into two interleaving FCC lattices denoted by light and dark cells.

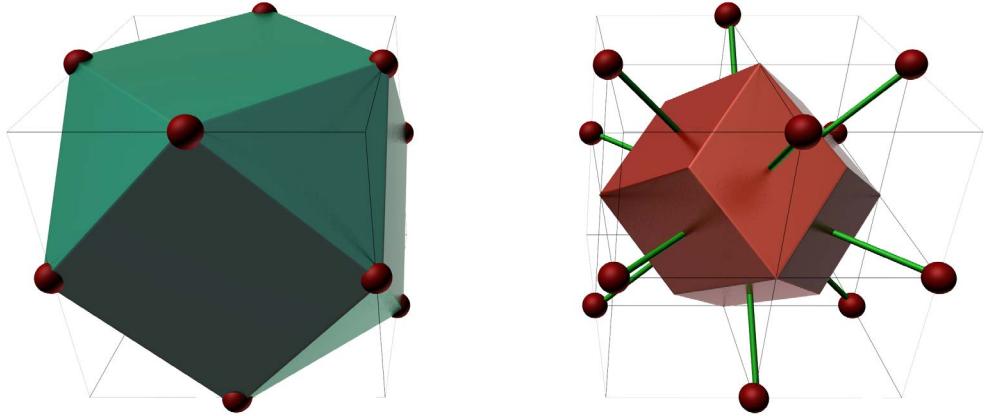
FCC lattice. In other words, for a given simulation configuration (same parameters, initial and boundary conditions), a simulation on a half-set of D3Q13 produces identical results on the corresponding nodes of the full D3Q13 simulation. The reason is that during the streaming step the physical properties of one site are calculated only from linked neighboring sites, and the collision step is a local computation at each lattice site. This feature is very important in the sense that we do not need to prove the validity of the LBM simulation on an FCC lattice. The D3Q13 LBM has been proven to recover the Navier-Stokes equation in the incompressible limit, which automatically leads to the conclusion that the LBM on an FCC lattice also recovers the Navier-Stokes equation. Furthermore, LBM is a mesoscopic method and the macroscopic physical values of the fluid field are statistically calculated from the particle distribution. Therefore, the LBM on the FCC lattice (the fD3Q13 LBM) can recover the macroscopic flow field with half the number of sites of the D3Q13 LBM, which means that the simulation speed can be doubled with little to no overhead. The possibility of using half the nodes of D3Q13 for LBM simulation has been presented by D’Humières et al. [28] and we propose an efficient implementation that maps naturally to GPU architectures.

An FCC lattice achieves the optimal sphere packing in 3D (another optimal sphere packing in 3D is the asymmetric HCP) [20]. It is also the lattice with maximum kissing number, similarly to the hexagonal lattice in 2D. Among all possible single speed LBMs in 3D, it is also the one with best angular discretization granularity of the velocity space. As shown in Figure 3.4, the 12 neighbors of an FCC lattice site form a cuboctahedron, and the Voronoi cell is a rhombic dodecahedron. Every lattice link (velocity vector) of the site intersects one rhombus on the Voronoi cell at the rhombus center.

Both the FCC and the HCP structures can be constructed by layering 2D hexagonal lattices. Given a hexagonal lattice  $M$  on the plane  $z = 0$ , the HCP lattice can be constructed as  $\{M + 2i(0, 0, \frac{\sqrt{6}}{3})|i \in \mathbb{Z}\} \cup \{M + (2i + 1)(\frac{1}{2}, \frac{\sqrt{3}}{6}, \frac{\sqrt{6}}{3})|i \in \mathbb{Z}\}$  while the FCC is defined as  $\{M + i(\frac{1}{2}, \frac{\sqrt{3}}{6}, \frac{\sqrt{6}}{3})|i \in \mathbb{Z}\}$ . In HCP, each lattice site also has 12 nearest neighbors, which is the maximum possible value in 3D and the lattice is an optimal sphere packing. However, for any link with direction  $d = (d_x, d_y, d_z)$  such that  $d_z \neq 0$ , the link

### 3. LBM FLUID SIMULATION AND VISUALIZATION ON OPTIMAL SAMPLING LATTICES

---



(a) FCC unit cell (cuboctahedron)

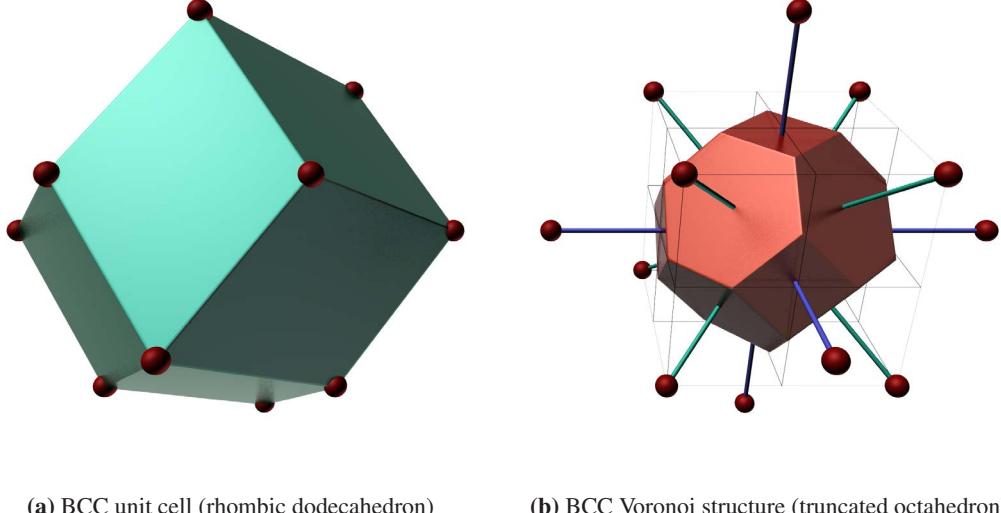
(b) FCC Voronoi structure (rhombic dodecahedron)

**Figure 3.4:** Unit cell and Voronoi structures for the FCC lattice.

with direction  $-d$  does not exist. Consequently, the HCP lattice does not satisfy the isotropy constraints and cannot be used in LBM simulations.

The BCC lattice can be constructed by adding an extra site at each cell of the CC lattice. BCC is the optimal lattice for sampling in 3D, but is not necessarily the best choice for LBM simulation. In BCC, each site has 8 nearest neighbors and 6 secondary neighbors, and the unit cell is a rhombic dodecahedron as shown in Figure 3.5. Thus, BCC and FCC are duals of each other and the Voronoi cell in BCC is a truncated octahedron. The faces on the truncated octahedron are 8 regular hexagons (corresponding to 8 nearest neighbors) and 6 squares (corresponding to 6 secondary neighbors). The distance from the site at the Voronoi cell center to the faces is not uniform, while in FCC this distance is uniform. As a result, the BCC lattice is less isotropic than the FCC. Furthermore, the FCC lattice is capable of simulating the flow field with half the number of CC lattice sites, while the BCC reduces the site count by about 30% [1]. Therefore, the FCC is our lattice of choice for LBM and we call the resulting simulation model fD3Q13 LBM.

In any simulation of a practical fluid field problem, the geometric objects of boundary conditions must be discretized on the underlying lattice, which inevitably introduces some discretization error. Both the hexagonal lattice in 2D and the FCC lattice in 3D are better than CC lattices in representing geometric objects because of their better isotropy. Consider the vertices of the Voronoi cell, in which the distance to sites is a local maximum: these are called *holes* [20]. If the distance from a point to the nearest site is the absolute maximum, it is called a deep hole, otherwise it is shallow. The maximum distance is also called the covering radius. Suppose the distance between two neighboring sites is 2, the covering radius is  $\frac{2\sqrt{3}}{3} \approx 1.15$  for hexagonal lattice and  $\sqrt{2} \approx 1.41$  for the FCC lattice, while the covering radius for the CC lattice is  $\sqrt{2} \approx 1.41$  in 2D and  $\sqrt{3} \approx 1.73$  in 3D. Intuitively, the smaller covering radius leads to less noisy and



**Figure 3.5:** Unit cell and Voronoi structures for the BCC lattice.

more isotropic representation of the discrete geometric object's surface. In Section 2.4.2, we have discussed the curved boundary treatment used in our simulation which is second order accurate. The precision of the extrapolation depends on the lattice link length. In hexagonal and FCC lattices, the link length is uniform thus the extrapolation precision does not vary with the link direction.

A hexagonal lattice in 2D is compactly stored in a 2D array of dimension  $(n_x, n_y)$  in row major order. For the  $i$ -th lattice site on row  $j$ , its lattice coordinates are defined as  $(i, j)$  and it is the  $j \times n_x + i$ -th element in the array. The coordinates in Cartesian space are computed by the following:

$$(x, y) = \left( i + \frac{1}{2} (j \bmod 2), \frac{\sqrt{3}}{2} j \right). \quad (3.2)$$

Here, the mod 2 operation can be implemented with a bitwise AND instruction. The 6 neighboring lattice sites are enumerated in counterclockwise order:  $(i+1, j)$ ,  $(i+j \bmod 2, j+1)$ ,  $(i+1+j \bmod 2, j+1)$ ,  $(i-1, j)$ ,  $(i+1+j \bmod 2, j-1)$ ,  $(i+j \bmod 2, j-1)$ . There are many ways to generate the lattice sites for the FCC lattice. We adopt a checkerboard generation pattern illustrated in Figure 3.3:

$$D_3 = \{(i, j, k) \in Z^3 : (i + j + k) \bmod 2 = 0\}. \quad (3.3)$$

The pattern can be viewed as removing half of the lattice sites with odd index sums from the D3Q13 lattice. Similarly to the 2D hexagonal lattice, the FCC lattice is stored in a 3D array of dimension  $(n_x, n_y, n_z)$  in row-major format. Compared to the D3Q13 LBM of size  $(n'_x, n'_y, n'_z)$ , each row of the lattice contains  $n_x = n'_x/2$  lattice sites and the other two dimensions do not change:  $n_y = n'_y$ ,  $n_z = n'_z$ . The index of a lattice site in the fD3Q13 LBM is defined to be the same as in the D3Q13 LBM. The lattice site  $(i, j, k)$

### 3. LBM FLUID SIMULATION AND VISUALIZATION ON OPTIMAL SAMPLING LATTICES

---

is the  $(i/2 + j * n_x + k * n_x * n_y)$ -th element in the array. The index of the  $t$ -th lattice site in the array is described by the following:

$$\begin{aligned} k &= t/(n_x \cdot n_y) \\ j &= (t - k \cdot n_x \cdot n_y)/n_x \\ i &= (t - k \cdot n_x \cdot n_y - j \cdot n_x) \cdot 2 + (j + k) \bmod 2. \end{aligned} \quad (3.4)$$

With this scheme, the 12 neighbors of a lattice site  $(i, j, k)$  can be enumerated as  $(i \pm 1, j \pm 1, k)$ ,  $(i, j \pm 1, k \pm 1)$ , and  $(i \pm 1, j, k \pm 1)$ .

## 3.3 Validation

As described in Section 3.2, the D3Q13 LBM simulation can be decoupled into two independent LBM simulations on fD3Q13 lattices. The two fD3Q13 lattices are defined over the nodes with even and odd sums of index components respectively. Therefore, the validity of the fD3Q13 LBM is proven by the validity of the D3Q13 LBM. We further compare the performance on the Cartesian and hexagonal lattices in 2D and the CC, FCC and BCC lattices in 3D. The results of interest include stability of the simulation, convergence speed and accuracy with respect to the known steady-state analytical solution or approximation.

### 3.3.1 Poiseuille Flow

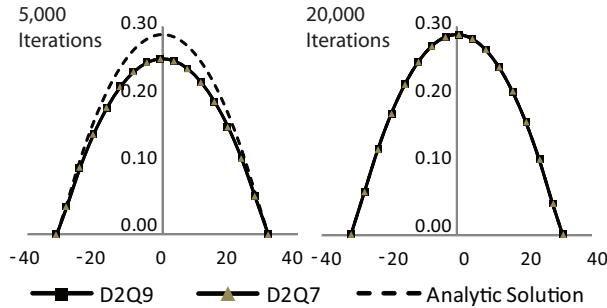
We investigate the performance of the hexagonal D2Q7 and Cartesian D2Q9 lattices in an LBM simulation of the steady Poiseuille channel flow. For a channel with width  $2L$  that supports a steady flow driven by a constant force  $F$  along the  $x$ -axis, the velocity is governed by the simplified Navier-Stokes equation [125]:

$$\nu \frac{d^2 \mathbf{u}}{dy^2} + F = 0, \quad (3.5)$$

where  $\nu$  is the fluid viscosity, and the boundaries along the  $x$ -axis at  $y = -L$  and  $y = L$  enforce a no-slip boundary condition. Then, the steady-state solution for the velocity is:

$$\mathbf{u}(y) = \frac{F}{2\nu} (L^2 - y^2). \quad (3.6)$$

Our first experiment employs LBM to simulate the Poiseuille flow using both the D2Q9 and the hexagonal D2Q7 lattices. The simulation setup consists of solid walls across the top and bottom row of lattice sites and a periodic boundary condition along the  $x$  axis. The boundary condition on the walls reduces to the simple bounce-back scheme since the boundary is aligned with the lattice links for both the D2Q7 and



**Figure 3.6:** Comparison of the velocity profiles for the Poiseuille Flow. The Cartesian and hexagonal lattices perform identically and both achieve the analytical result after 20,000 iterations.

the D2Q9 lattices. The flow is initially at rest and is accelerated by a constant force parallel to the  $x$ -axis, using the microscopic forcing method [125]. With this method, a force contribution is added to the velocity moment of the equilibrium distribution of each lattice node at each time step. Both simulations use approximately the same number of lattice sites and our hypothesis is that the hexagonal lattice performs no worse than the cubic lattice in terms of convergence behavior and ability to reach the steady-state velocity profile given by the analytical solution.

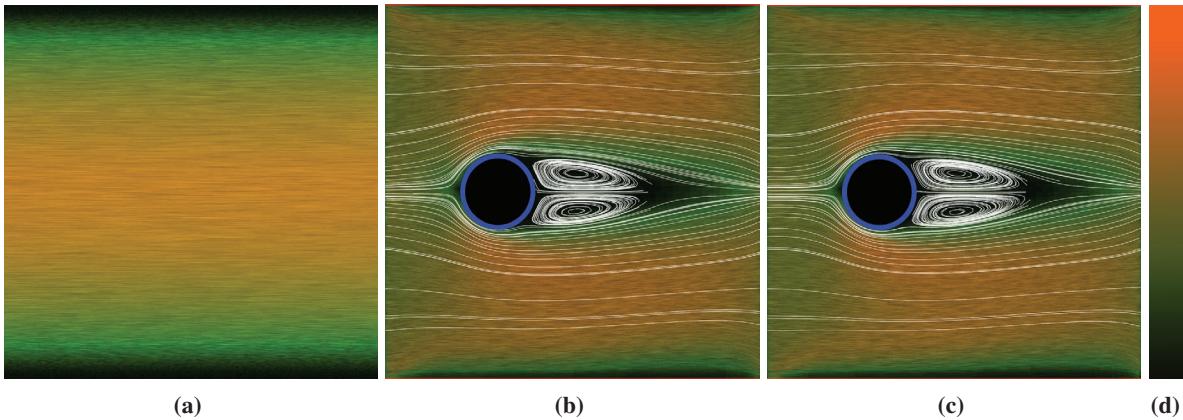
In our experiments of Poiseuille flow, the D2Q9 domain contains  $63 \times 63$  nodes with a channel width  $2L = 62$ . The viscosity is  $\nu = 0.1667$  and the driving force is set to  $F = 0.0001$ . The equivalent D2Q7 domain contains  $59 \times 68$  nodes and the LBM parameters are scaled appropriately. The initial fluid density is set to  $\rho = 1$  and link densities are in equilibrium. The force is applied at each simulation step, slowly accelerating the fluid. As can be seen in Figure 3.6, after 20,000 iterations, the simulations have reached a steady state and the velocities agree with the analytical velocity profile described by Equation 3.6. Figure 3.7a shows the velocity field rendered with the Line Integral Convolution (LIC) method [15].

In addition to the simple Poiseuille Flow, we have also implemented a simulation of a flow around a circular boundary. The boundary condition in this case uses a combination of bounce-back rules with first order linear interpolation, which is able to handle curved boundaries [12]. We describe this technique in Section 2.4.2. Instead of a constant driving force, we now employ inlet and outlet boundary conditions along the left and right walls of the simulation domain, respectively. The inlet and outlet are implemented as Dirichlet boundary conditions where the velocity moment along the boundary is reset to the initial value at each simulation step. Our investigation focuses on the ability of both lattices to resolve the vortex structures that form near the circular boundary.

For the simulation with circular boundary, we place a disc of diameter  $d = 0.2$  at position  $(0.3, 0.5)$ . The disc parameters are normalized to the dimensions of the simulation domain. The viscosity is changed to  $\nu = 0.04$  and the inlet velocity is set to  $u = 0.1$ . Figure 3.7 shows the resulting steady-state flows

### 3. LBM FLUID SIMULATION AND VISUALIZATION ON OPTIMAL SAMPLING LATTICES

---



**Figure 3.7:** (a) Illustration of the Poiseuille Flow after 20,000 iterations. The simulations based on D2Q7 and D2Q9 lattices produce identical results. (b) and (c) Simulation results for flow past a circular boundary in 2D for the D2Q9 and D2Q7 lattices, respectively. (d) Relative scale used for visualization of the speed of the flows.

**Table 3.1:** Comparison of LBM performance on the D2Q7 and D2Q9 lattices in terms of Seconds Per Time Step (SPTS) and Lattice Updates Per Second (LUPS). Lattice sizes are  $64 \times 64$  for the Cartesian lattices and  $59 \times 68$  for the hexagonal lattice. Simulations are run on an Intel Core Duo 1.86GHz processor.

Lattice Type	Nodes	SPTS (ms)	LUPS (u/s)
D2Q7	4012	2.172	1,847,145
D2Q9	4096	3.376	1,213,270

using a GPU-accelerated version of the LIC method with additional streamline rendering to enhance the presentation of the vortices. The hexagonal lattice produces flow results comparable to the Cartesian lattice while improving the speed of the simulation, as shown in Table 3.1, due to the decreased use of memory and computational resources.

#### 3.3.2 Flow Past a Sphere

We investigate the performance of fD3Q13 LBM in simulating a flow moving past a sphere, which is a well-studied problem [39, 59, 60, 123]. It is one of the few setups in 3D for which there is a known analytical approximation to the dimensionless drag coefficient if the simulation domain is an infinitely long circular pipe with a finite diameter  $D$  [39]. For this simulation, the Reynolds number is defined as  $Re = \frac{U_0 d}{\nu}$  where  $U_0$  is the steady speed of the sphere moving against a fluid with viscosity  $\nu$ . The diameter of the sphere is  $d$ .  $c'_d = \frac{24}{Re}$  is the drag coefficient for Stokes's Law region ( $Re \leq 1$ ). We use the analytical approximation developed by Wham et al. [124], which accounts for the effect of the wall on a moving sphere for flows with  $Re \in [1, 100]$ . The drag coefficient  $c_d$  is then defined by:

$$\frac{c_d}{c'_d} = \left(1 + 0.03708 Re^{1.514 - 0.1016 \ln Re}\right) \times \frac{1 - 0.75857 \lambda^5}{1 - K\lambda + 2.0865\lambda^3 - 1.7068\lambda^5 + 0.72603\lambda^6}, \quad (3.7)$$

where  $\lambda = \frac{d}{D}$  is the ratio of diameters for the sphere and the cylindrical domain and  $K = 0.6628 + 1.458e^{-0.05635Re}$ . The drag force acting on the sphere is:

$$F_d = c_d \frac{\rho U_0^2}{2} \pi \frac{d^2}{4}. \quad (3.8)$$

In our simulation, we compute the force that a density of particle exerts on the boundary by considering the change of momentum along a link that intersects that boundary. Suppose we have a link  $m$  that originates from a lattice site at position  $\mathbf{x}_f$  toward a position  $\mathbf{x}_b$  inside the boundary. The link has index  $i$  in the lattice structure, velocity  $\mathbf{c}_i$  and link  $i'$  is the opposite in the lattice. Then the force  $F_m$  exerted on the boundary by momentum exchange on link  $m$  is:

$$F_m = -c_i (f_i(x_f, t + \Delta t) + f_{i'}(x_f, t)). \quad (3.9)$$

The total force acting on the boundary can be computed by integrating the force over the surface, which in our discrete case corresponds to finding the sum of the individual force contributions from each link  $m$  that intersects the boundary:

$$F = V \sum_m F_m \quad (3.10)$$

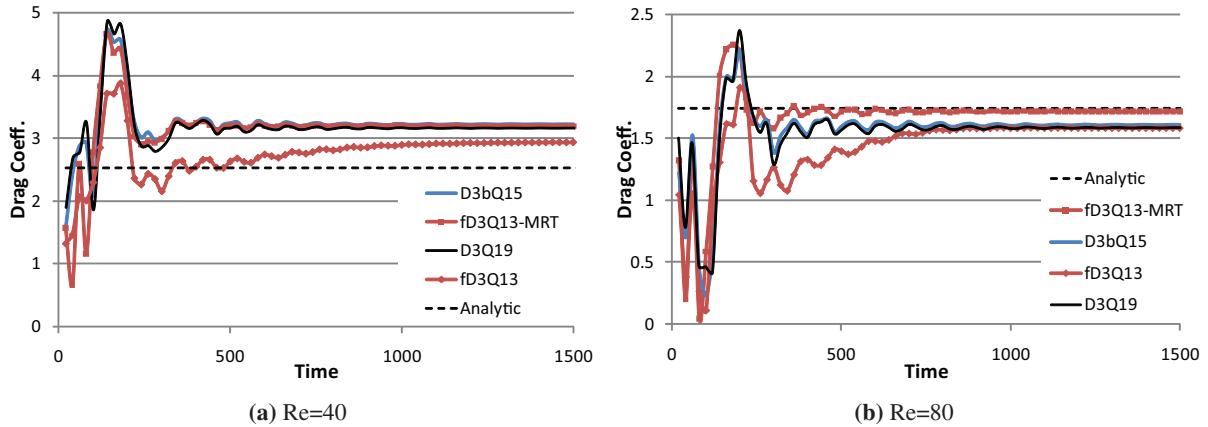
Note that in Equations 3.9 and 3.10 we assume that each site represents a unit of the simulation domain. Therefore, to make the results consistent across different simulations, the force must be scaled by the volume  $V$  of the unit cell in the lattice.

We have performed LBM simulations based on the Cartesian lattices (D3Q13, D3Q15, D3Q19, D3Q27), the BCC lattice (D3bQ15) [1] and the FCC lattice (fD3Q13). The cubic and BCC lattices use the single-relaxation-time model, while the FCC lattice employs the multiple-relaxation-time model (MRT-LBM) [29] to improve numerical stability. The additional computational overhead of MRT-LBM is offset by the performance advantage of using only half of the simulation nodes compared to the Cartesian lattices. This allows us to achieve numerical stability comparable to D3Q19 at reduced computational cost.

We model a channel of finite length  $L = 64$  enclosed by a solid cylindrical boundary with a cross section diameter  $D = 60$ . Inlet and outlet boundary conditions are applied along the left and right simulation boundaries, respectively. Additional velocity boundary conditions are enforced at the fluid interface on the pipe boundary and the sphere boundary through the  $u_w$  term in Equation 2.13. The velocity of the sphere with diameter  $d = 15$  is 0 and the velocity at the pipe boundary is  $U_0$ . The viscosity is set to  $\nu = 0.0375$ .

### 3. LBM FLUID SIMULATION AND VISUALIZATION ON OPTIMAL SAMPLING LATTICES

---

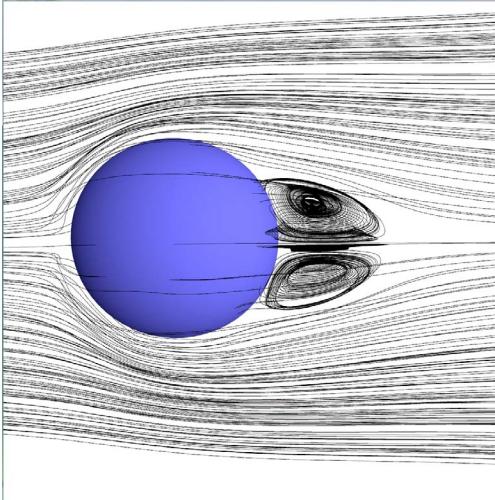


**Figure 3.8:** Comparison with the steady-state analytical approximation for the drag coefficient of a sphere moving through a viscous media over time.

This setup effectively models the steady movement of a sphere through a viscous media in an infinitely long pipe. We use Equation 3.10 to calculate the total drag force acting on the sphere for varying Reynolds numbers. The dimensionless drag coefficient computed from Equation 3.8 is then compared to the analytical approximation for the system, given by Equation 3.7.

Figure 3.8 shows the convergence toward the steady-state analytical approximation to the dimensionless drag coefficient. The *D3Q15* and *D3Q27* simulations yield results virtually equivalent to the *D3Q19* simulation and are omitted from the graph. The simulation based on the BCC lattice also exhibits similar convergence characteristics with 30% less samples. The FCC lattice further reduces the number of samples and the *fd3Q13-MRT* simulation shows marginally faster convergence rates. It can also be expected that the MRT model allows for greater numerical stability when  $Re > 80$ . The performance results for our LBM implementations with the different lattice types are shown in Table 3.2.

Figure 3.9 uses a dense set of streamlines to visualize the vortices formed behind the sphere. The visualization component creates streamlines in a single slice parallel to the  $xz$ -plane. The slice is slightly away from the center line of the cylindrical domain in order to capture the three dimensional nature of the velocity field around the sphere. Additional streamline seeds are placed behind the sphere to enhance the rendering of the vortices.



**Figure 3.9:** Visualization of vortices in the flow behind the sphere at  $Re = 80$ .  $\lambda = \frac{d}{D} = \frac{15}{60}$ , where  $d$  is the diameter of the sphere and  $D$  is the diameter of the cylindrical boundary.

## 3.4 LBM Implementation with CUDA

### 3.4.1 Programming Model

General programming for a GPU architecture requires intimate knowledge of the graphics pipeline and associated graphics primitives, which hinders the efforts of developers who lack this background. NVIDIA CUDA (Compute Unified Device Architecture) is a framework that abstracts supported GPUs using the standard C language with extensions for stream computing. These extensions allow for example the definition of device vs. host routines and the execution of computational kernels in parallel. The framework also includes optimized FFT and BLAS implementations.

The GPU is a compute device that (1) is a co-processor to the CPU or host, (2) has its own onboard device memory, and (3) can execute many threads in parallel. Data-parallel algorithms are defined in the form of kernels, which run in parallel over the threads. The GPU threads are very lightweight compared to CPU threads with very little creation overhead, since the GPU must keep 100s of threads in flight to efficiently utilize its available computational resources. Data is transferred between the host memory and the device memory through a set of highly optimized function calls and the framework supports a number of different memory layouts. For example, data can be copied to 'linear global memory', which supports read-write access, but the memory access is not cached. In contrast, read-only 'texture memory' supports linear data interpolation and may offer higher bandwidth due to caching.

At the lowest level, kernel executions in CUDA are grouped into *thread blocks*, in which the individual threads can synchronize their execution using a sync barrier and share data through very fast per-block shared

### 3. LBM FLUID SIMULATION AND VISUALIZATION ON OPTIMAL SAMPLING LATTICES

---

**Table 3.2:** Comparison of LBM performance on different lattice types in terms of Seconds Per Time Step (SPTS) and Lattice Updates Per Second (LUPS). Lattice sizes are  $64 \times 64 \times 64$  for the Cartesian lattices,  $32 \times 64 \times 64$  for the FCC lattice and  $45 \times 45 \times 91$  for the BCC lattice. Simulations are run on an Intel Core Duo 1.86GHz processor.

Lattice Type	Nodes	SPTS (s)	LUPS (u/s)
<b>fD3Q13</b>	<b>131,072</b>	<b>0.20892</b>	<b>627,379</b>
D3Q13	262,144	0.41114	637,603
D3Q15	262,144	0.44122	594,134
<b>D3bQ15</b>	<b>184,275</b>	<b>0.38581</b>	<b>477,631</b>
D3Q19	262,144	0.54297	482,796
D3Q27	262,144	0.70180	373,531
<b>fD3Q13-MRT</b>	<b>131,072</b>	<b>0.43914</b>	<b>298,474</b>
D3Q15-MRT	262,144	1.04203	251,570
D3Q19-MRT	262,144	1.44079	181,945

memory. The size of the thread blocks can be defined with 1D, 2D or 3D vectors depending on application requirements and each thread is assigned a thread ID that can be queried at run-time. At a higher level, individual thread blocks are grouped into grids, whose size can again be defined by a multi-dimensional vector. This hierarchical organization can greatly simplify the processing of multi-dimensional data. More information, including code examples, is available in the NVIDIA CUDA Programming Guide [81].

#### 3.4.2 LBM with CUDA

Our CUDA implementation of LBM is built as an extension to the CPU-based framework. The simulation involves the lid-driven cavity flow problem on the D3Q19 LBM. During initialization, memory is allocated on the GPU and all LBM control structures are copied to constant memory. The host is also responsible for initializing the geometry array and the two distributions arrays. Thread blocks are defined so that the threads in each block access elements along a single axis-aligned line of the simulation domain. Such an organization allows for efficient reading and writing of the density values [109]. The computation grid is defined using the remaining two dimensions of the domain.

Our CUDA kernel combines the collision and propagation steps of the LBM and its implementation is a straightforward extension of our CPU framework. Most of the code is reused from the CPU kernel and the resulting application achieves performance that is an order of magnitude higher than our CPU application. The main performance bottleneck is the streaming step, during which densities are copied to neighboring lattice sites. Because of the memory layout and grid configuration, all memory writes with offsets  $(0, 0, 0)$ ,  $(0, 0, \pm 1)$ ,  $(0, \pm 1, 0)$ , and  $(0, \pm 1, \pm 1)$  are coalesced and executed very efficiently. However, the remaining 10 propagations are offset along the  $x$  direction, which forces writes to misaligned memory locations within

### 3.4 LBM Implementation with CUDA

---

**Table 3.3:** The effect of memory access optimizations on CUDA performance, expressed in MLUPS.

CUDA Device	Un-optimized				Optimized			
	$32^3$	$64^3$	$96^3$	$128^3$	$32^3$	$64^3$	$96^3$	$128^3$
NVIDIA Quadro FX 4600 (CUDA)	28	33	32	20	101	144	127	134
NVIDIA Tesla C1060 (CUDA)	59	64	60	46	232	240	278	233

a single thread block. These non-coalesced memory transactions may reduce memory bandwidth by an order of magnitude. The hardware capabilities of a CUDA device are described by compute capability versions, which are listed in the NVIDIA CUDA Programming Guide [81]. In general, devices with compute capability 1.0 and 1.1 can coalesce memory transactions only when threads within a block access memory words sequentially.

We employ shared memory during the propagation step, which achieves coalesced memory writes for all propagation operations and greatly improves the computational throughput. Each thread copies the critical densities to the appropriate offset in the allocated shared memory instead of the global memory. After the synchronization barrier, each thread within the block writes to global memory at sequentially increasing memory offsets, which achieves coalescing of the memory writes.

As an additional optimization, we allocate single float variables instead of float[19] arrays in the CUDA kernel, which ensures that the variables are kept in registers instead of spilling into local memory. This optimization may not be feasible for more complex simulations or larger domain sizes. Table 3.3 compares the performance of our straightforward implementation and the optimized version.

We have compared the performance of our Zippy-based implementation of LBM against the new code based on CUDA. In both cases we have implemented the SRT-LBM on the D3Q19 Cartesian lattice and the applications do not perform any data collection or rendering of the resulting velocity fields. We have also compared the performance against a CPU implementation of LBM in C++, which performs a similar computational load without any particular optimizations to the computation kernel or the memory layout. We have implemented multi-threading using OpenMP directives on the main computational loop.

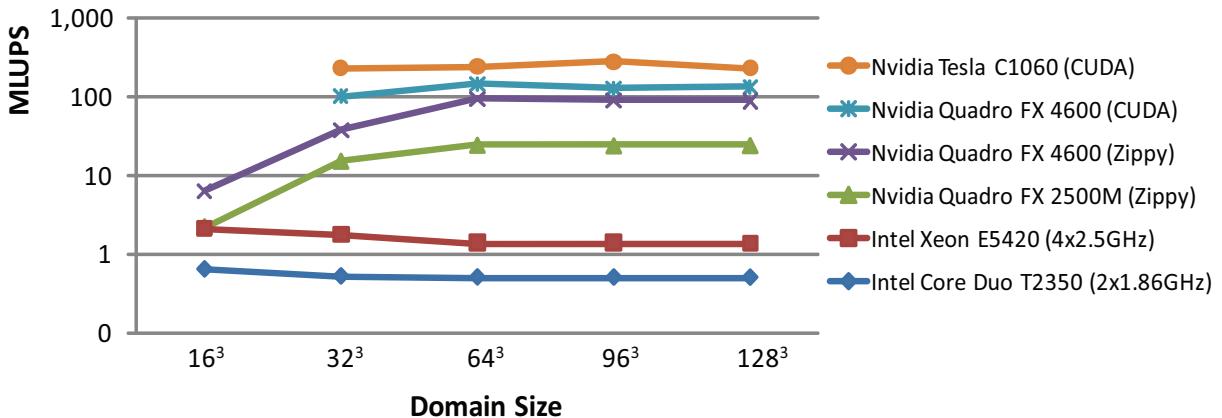
Our benchmarks were conducted across dual- and quad-core CPU systems and across three generations of graphics processors. The hardware includes an Intel Core Duo 1.86Ghz machine with an NVIDIA Quadro FX 2500M GPU (230 GFLOPs theoretical peak performance) and an Intel Xeon Quad-core 2.5Ghz workstation with an NVIDIA Quadro FX 4600 and an NVIDIA Tesla C1060 board (40 GFLOPs, 518 GFLOPs and 933 GFLOPs theoretical peak performance respectively). Table 3.4 lists the performance of our implementations in MLUPS (Mega Lattice site Updates per Second).

It is clear that a CUDA implementation of LBM provides an increase in performance over an implementation that relies on the graphics pipeline. In particular, performance for relatively small problem sizes

### 3. LBM FLUID SIMULATION AND VISUALIZATION ON OPTIMAL SAMPLING LATTICES

**Table 3.4:** LBM Performance in MLUPS for different architectures and domain sizes.

Architecture	Domain Size				
	$16^3$	$32^3$	$64^3$	$96^3$	$128^3$
<i>Intel Core Duo 1.86GHz</i>	0.6	0.5	0.5	0.5	0.5
<i>Intel Xeon E5420</i>	2.0	1.7	1.4	1.4	1.4
<i>NVIDIA Quadro FX 2500M (Zippy)</i>	2.2	15	25	24	24
<i>NVIDIA Quadro FX 4600 (Zippy)</i>	6.4	37	94	91	89
<i>NVIDIA Quadro FX 4600 (CUDA)</i>	—	101	144	127	134
<i>NVIDIA Tesla C1060 (CUDA)</i>	—	232	240	278	233



**Figure 3.10:** Performance scaling for CPU and GPU architectures.

is improved dramatically. Nevertheless, efficient use of the GPU requires a sufficient number of threads per block and at least 100 blocks per grid [81]. This is in contrast to CPUs, where the increased domain size lowers the effectiveness of memory caching and thus lowers performance. Another difference is that the GPU is generally a memory-bound architecture, both in terms of memory bandwidth and memory size. Obtaining high memory throughput generally involves the use of the fast shared memory, which is a limited resource per multiprocessor. Additional memory constraints are placed on the number of available registers per block and the amount of global memory. The limited scalability of GPUs also has broad implications for the use of GPU clusters. Even when using CUDA as the back-end computation interface, the limited range of efficient problem sizes translates into poor scalability for fixed-size problems. On the other hand, GPU clusters can exhibit close to linear performance increase when the problem size grows with the introduction of new GPU nodes.

One feature of particular interest for scientific computing is the support for extended precision opera-

**Table 3.5:** Performance comparison for CUDA code running in single- and double-precision. Performance numbers are in MLUPS.

CUDA Device	single-precision		double-precision	
	64 <sup>3</sup>	96 <sup>3</sup>	64 <sup>3</sup>	96 <sup>3</sup>
<i>Nvidia Quadro FX 4600 (CUDA)</i>	144	127	32.9	33.0
<i>Nvidia Tesla C1060 (CUDA)</i>	240	278	140	121

tions under CUDA. In particular, the compute capability 1.3 profile for CUDA hardware introduces support for 64-bit precision computations [81] at the cost of reduced peak throughput. For compute-bound kernels, the stated peak performance of the NVIDIA Tesla C1060 for double-precision computations is 1/12th of the peak single-precision performance. On the other hand, for bandwidth-bound code, such as our optimized LBM kernel, the performance decreases by approximately a factor of 2 to 121 MLUPS for the 96<sup>3</sup> domain (see Table 3.5). In our experience, the single-precision arithmetic of the GPU architecture provides sufficiently accurate and stable results in the LBM for interactive applications.

## 3.5 Smoke Simulation in Urban Environment

We demonstrate the high performance of LBM based on the fD3Q13 lattice in a simulation of smoke propagation in an urban environment. A GPU-accelerated LBM simulation coupled with an OpenGL renderer allows for interactive exploration of a city environment encoded by a hierarchical definition and a texture library.

### 3.5.1 Urban Modeling

Our model captures intrinsic features of an urban environment using a hierarchical, grammar-based representation. At the highest level, there are three basic primitives - buildings, sidewalks and roads. We take advantage of the regular structure found in metropolitan areas by modeling buildings as rectangular boxes and the roads and sidewalks as layers of polygons. This simplicity of representation allows for efficient storage of large data sets and rapid rendering. Additional details are inserted into the scene by rendering their respective sub-hierarchies to textures, which are then mapped onto the primitives defined at the higher levels. For example, our facade grammar defines features such as brick type, storefront, window and balcony structure, etc. A library of textures is then used to create a novel appearance or model the facade based on real-world images.

We use a prototype OpenGL-based renderer with render-to-texture capabilities and support for shaders written with NVIDIA's CgFX language. At run time, we traverse the grammar file and render the elements of

### 3. LBM FLUID SIMULATION AND VISUALIZATION ON OPTIMAL SAMPLING LATTICES

---

each facade to a texture. All textures are kept locally on the GPU and texture resolution is based on available memory. Certain elements in the facade grammar have associated CgFX shaders, which are executed during the rendering stages. For this project, we have implemented a simple reflection and refraction effect, which is used to model the appearance of glass. It is associated with windows and glass sections on doors and storefronts.

#### 3.5.2 Modeling of Smoke in Urban Environment

We use the MRT-LBM [29] to simulate the air flow in the urban environments which achieves better numerical stability compared with the commonly-used SRT-LBM [107, 122]. The idea of the MRT-LBM is to transform the particle distributions from phase space (i.e., the space of the distributions  $f_i$ ) to the space of hydrodynamic moments (i.e., density, momentum, energy, etc.) and to perform the collision step in the moment space. As in the SRT-LBM model, the effect of collisions is approximated by a relaxation toward an equilibrium state, but in the moment space each moment is allowed to relax individually. The transformations between the phase space and the moment space moments are given by:

$$\begin{aligned} |f\rangle &= (f_0, f_1, \dots, f_n)^T, \\ |m\rangle &= (m_0, m_1, \dots, m_n)^T, \\ |m\rangle = M|f\rangle, \quad |f\rangle &= M^{-1}|m\rangle. \end{aligned} \tag{3.11}$$

Here, the Dirac notation  $|.\rangle$  is used to denote column vectors,  $T$  denotes the transpose,  $n$  is the number of the distributions ( $n=13$  for the fD3Q13 LBM), and  $M$  is a  $n \times n$  transformation matrix. In MRT-LBM, the collision step becomes:

$$|f(\mathbf{x}, t + \Delta t)\rangle - |f(\mathbf{x}, t)\rangle = -M^{-1}S[|m(\mathbf{x}, t)\rangle - |m^{eq}(\mathbf{r}, t)\rangle] \tag{3.12}$$

where the components of the vector  $|m^{eq}\rangle$  are the local equilibrium values of the moments. The matrix  $S$  in the collision equation is a diagonal matrix  $S \equiv diag(s_0, s_1, \dots, s_n)$  whose elements are the relaxation rates. Their values are directly related to the kinematic shear and bulk viscosities [29].

The improved stability of the MRT-LBM allows thermal effects to be coupled in, resulting in the hybrid thermal LBM (HTLBM) [66]. Ye et al. [130] have used the HTLBM to model the weakly compressible thermal flow for shimmering effects. We adopt it for smoke simulation in an urban environments. To capture thermal effects, temperature is coupled to the MRT-LBM through the energy moment, meaning that the energy equilibrium is modified during the calculation of equilibrium distributions at each time step.

### 3.5 Smoke Simulation in Urban Environment

---

The row of the transformation matrix  $M$  relevant to the computation of the energy moment is constructed as follows:

$$M_4 = \left\langle \frac{13}{2} \mathbf{c}_i \cdot \mathbf{c}_i - 12 \right\rangle = (-12, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1), \quad (3.13)$$

where  $\mathbf{c}_i$  is a lattice velocity and  $\langle . \rangle$  denotes a row vector. The energy moment is computed through Equation 3.12 and the equilibrium energy moment for the HTLBM model is defined by:

$$m_4^{eq} = e^{eq} n_1 (c_{s0}^2 - n_2) \rho + n_3 (n_4 - \gamma) \mathbf{u} \cdot \mathbf{u} + q_1 T. \quad (3.14)$$

The new variables in Equation 3.14 are the temperature  $T$ , its constant coupling coefficient  $q_1$ , and the specific heat  $\gamma$ . The parameters  $n_1$  to  $n_4$  are constants and their values are determined by the linear stability analysis. The constant  $c_{s0}$  is the isothermal speed of sound, which is specific to the chosen lattice velocity set. Heat transfer is modeled separately with a standard advection-diffusion equation:

$$\partial_t T + \mathbf{u} \cdot \nabla T = \kappa \Delta T + q_2 (\gamma - 1) c_{s0}^2 \nabla \cdot \mathbf{u}. \quad (3.15)$$

The parameter  $\kappa$  is the thermal diffusivity of the fluid and  $q_2$  is another constant coupling coefficient. The advection computation executes after the streaming step in LBM and modifies the temperature used in the calculation of the energy moment in the following simulation step. Finally, the evolution of the smoke density volumetric dataset is also modeled by an advection-diffusion equation, which is computed by a back-tracing algorithm [103]. We use the fluid velocity computed at each lattice site to determine a sampling position for the back-tracing density value. The monotonic cubic interpolation [38] is used to compute densities at off-grid sampling positions and we have also implemented the lower-order trilinear interpolation for cases that require higher performance at the expense of visual fidelity. The advection-diffusion process for the smoke density  $\rho_s$  is described by the following:

$$\partial_t \rho_s + \mathbf{u} \cdot \nabla \rho_s = 0. \quad (3.16)$$

Here, the flow velocity  $\mathbf{u}$  is obtained from the HTLBM. Based on the smoke density, a gravity force is applied to the HTLBM:

$$f_{body} = -\alpha \rho_s y + \beta (T - T_0) \hat{y} \quad (3.17)$$

In this equation,  $\hat{y}$  is the unit upward vector,  $\alpha$  and  $\beta$  are the coefficients of gravity and buoyancy, and  $T_0$  is the temperature of the surrounding air. The force is applied through the velocity moment of the equilibrium

### 3. LBM FLUID SIMULATION AND VISUALIZATION ON OPTIMAL SAMPLING LATTICES

---

distribution, calculated in Equation 3.12. More details are available in the paper on HTLBM [66] and in the work of Zhao et al. [130].

The advection for the smoke density scalar field  $\rho_s$  is computed with a back-tracing algorithm [103] using the semi-Lagrangian method. We use the fluid velocity computed at each lattice site to determine a sampling position for the back-tracing density value. The monotonic cubic interpolation [38] can be used to compute densities at off-grid sampling positions. We have also implemented lower-order interpolation schemes that can be executed efficiently on the GPU for cases that require higher performance at the expense of visual fidelity.

The semi-Lagrangian advection approach leads to excessive smoothing of the resulting scalar field, which in our case translates into loss of detail in the resulting smoke images. We employ the higher order MacCormack advection scheme [94], which consists of a forward advection, a backward advection, and an error correction. Given an advection scheme  $A$ , such as the semi-Lagrangian described above,  $A^R$  refers to an advection run with reversed velocity. The scheme can be described by the following:

$$\begin{aligned}\hat{\rho}_s^{n+1} &= A(\rho_s^n) \\ \hat{\rho}_s^n &= A^R(\hat{\rho}_s^{n+1}) \\ \rho_s^{n+1} &= \hat{\rho}_s^{n+1} + \frac{1}{2}(\rho_s^n - \hat{\rho}_s^n).\end{aligned}\tag{3.18}$$

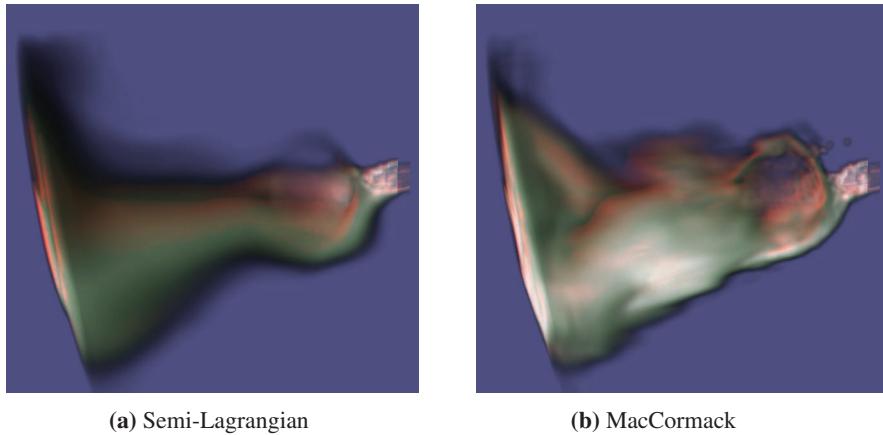
Here,  $\rho_s^n$  refers to the current scalar field,  $\rho_s^{n+1}$  is the final advected field, and the scheme requires the use of two temporary fields  $\hat{\rho}_s^n$  and  $\hat{\rho}_s^{n+1}$  to store the intermediate results. Unlike the semi-Lagrangian, this scheme is not unconditionally stable. Therefore, we clamp the smoke density to the range of densities defined by the samples that were used for the initial advection step. Fig 3.11 illustrates the difference in detail resolution between the semi-Lagrangian and the MacCormack schemes.

In general, computational fluid dynamics methods, and specifically LBM, may lead to excessive non-physical numeric dissipation, which leads to viscous flows that cannot represent the details of highly turbulent phenomena. Vorticity confinement reintroduces small-scale features to the flow in a physically-based fashion [38]. We have implemented this approach in our simulation framework. Vorticity is defined as the curl of the velocity field:

$$\omega = \nabla \times \mathbf{u}.\tag{3.19}$$

The vorticity confinement force is then defined over the discrete simulation domain as:

$$F_v c = \epsilon N \times \omega,\tag{3.20}$$



**Figure 3.11:** Comparison of detail resolution under different smoke advection schemes.

where  $N$  is the normalized vorticity gradient  $\eta = \nabla|\omega|$  and  $\epsilon$  is a parameter which determines the amount of small-scale detail that is added to the velocity field. This force adds small-scale rolling features that are propagated along the flow. As with the temperature advection, the finite-difference operators are defined over a stencil using the full velocity set of the simulation lattice in order to improve numerical accuracy. Another possible approach for suppressing the energy dissipation is the wavelet turbulence technique [62], which adds turbulence features at different scales in the flow and can be implemented as a post-processing step on the GPU [40].

Unlike Zhao et al. [130] who have implemented the D3Q13 HTLBM on the CC grid on the GPU, we implemented the FCC HTLBM on the GPU. Half of storage consumption and almost half of the computations are therefore saved. As described in Section 3.2, the FCC lattice can be represented as a 3D array and the indices of lattice sites can be calculated with Equation 3.4. Therefore, a natural choice is to represent the lattice data on the GPU as a set of 3D textures. The thirteen particle distributions  $f_i$  are stored in the color channels of four 3D textures, the elements of which have 4, 4, 4, and 1 components, respectively. The post-collision particle distributions  $f_i(\tilde{\mathbf{x}}, t^+)$ , moments  $m_i$ , and equilibrium moments  $m_i^{eq}$  are also stored in this way. The flow velocity  $\mathbf{u}$  and density  $\rho$ , and the temperature and smoke density are stored in two additional 3D textures.

Each simulation step includes a series of computation sub-steps, such as computing the macroscopic flow velocity and density, computing the equilibrium moments, performing particle collision and streaming, applying buoyancy and gravity body forces, and advecting the density and the temperature fields with the LBM flow field. The sub-steps involve only data from the local neighbors of a site in the FCC lattice and the operations are explicitly parallel. They are implemented as computation kernels in OpenGL fragment programs, each of which operates on the lattice in SIMD fashion and updates the corresponding lattice data.

### **3. LBM FLUID SIMULATION AND VISUALIZATION ON OPTIMAL SAMPLING LATTICES**

---

The fragment program samples the lattice data from the 3D textures, computes the new lattice data, and renders slices of the output back into the 3D textures.

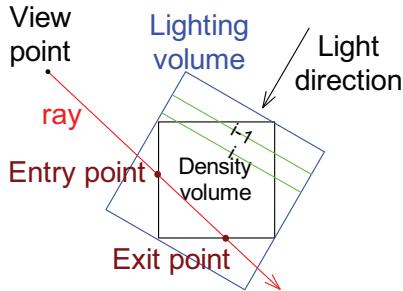
To handle the boundary conditions of objects in the flow, we calculate the intersection points at which the lattice links are cut by the object surfaces and use this information to locally modify the particle distributions. In preprocessing, the intersection points of the object surface and lattice links are computed and the related boundary information is stored as a vertex buffer object. During simulation, we trigger the boundary condition computation kernel on the intersected links by rendering this vertex buffer object to the frame-buffer. This operation performs the calculations described in Equation 2.13 on the particle distributions stored in textures on the GPU.

#### **3.5.3 Rendering**

The fD3Q13 LBM simulation produces the smoke density volume on an FCC lattice. Qiu et al. [88] have proposed a framework for volumetric global illumination on the FCC lattice volume, which is capable of capturing multiple scattering. However, although this method is much faster than conventional methods such as radiosity and photon mapping, it does not achieve real time performance. Because the simulation is performed on the GPU at interactive speed with results residing on the local video memory, we would like to also use the GPU for rendering the smoke in the urban environment. A major difficulty of rendering FCC volumes is that a computationally intensive filter such as Gaussian filter is necessary for reconstruction [88], which is slow even on the GPU. Entezari has proposed box splines for reconstruction of data sampled on the FCC lattice [33], however we are not aware of any GPU implementations. Therefore, in this application we first convert the FCC lattice volume to a rectilinear volume. This allows us to take advantage of the texturing unit in current GPUs for efficient trilinear interpolation. Note that the index of a lattice site in the fD3Q13 LBM is defined to be the same as in the D3Q13 LBM. For a sampling point in the regular volume with index  $(i, j, k)$ , if  $(i + j + k) \bmod 2 = 0$ , the density value is copied from the fD3Q13 LBM lattice. Otherwise, the density value is interpolated from neighboring even points:

$$d(i, j, k) = \frac{1}{6} (d(i \pm 1, j, k) + d(i, j \pm 1, k) + d(i, j, k \pm 1)). \quad (3.21)$$

We use ray-tracing with single scattering for volume rendering. The illumination  $I(P, \omega')$  arriving at point  $P$  for a directional light source in the direction  $-\omega'$  is calculated and stored in a lighting volume. As shown in Figure 3.12, the lighting volume is the minimal enclosing box of the density volume such that slices of the lighting volume are perpendicular to the light direction. The light intensity  $L(p_i)$  at point  $p_i$  on slice  $i$  is computed by attenuating  $L(p_{i-1})$  on slice  $i - 1$  with the dispersion opacity  $\tau(p_{i-1})$ . In our application, the objects cast shadows on the volume. Because the objects are opaque,  $L(p)$  is 0 if  $p$  is occluded by some



**Figure 3.12:** Every slice (green lines) of the lighting volume (blue box) is perpendicular to the light direction. The lighting volume stores light intensity and density sampled from the density volume (black box). The light intensity of slice  $i$  is calculated by attenuating slice  $i - 1$  with the current opacity values. The ray is traced through the lighting volume. The entry point and exit point of the ray are on the bounding box of the density volume.

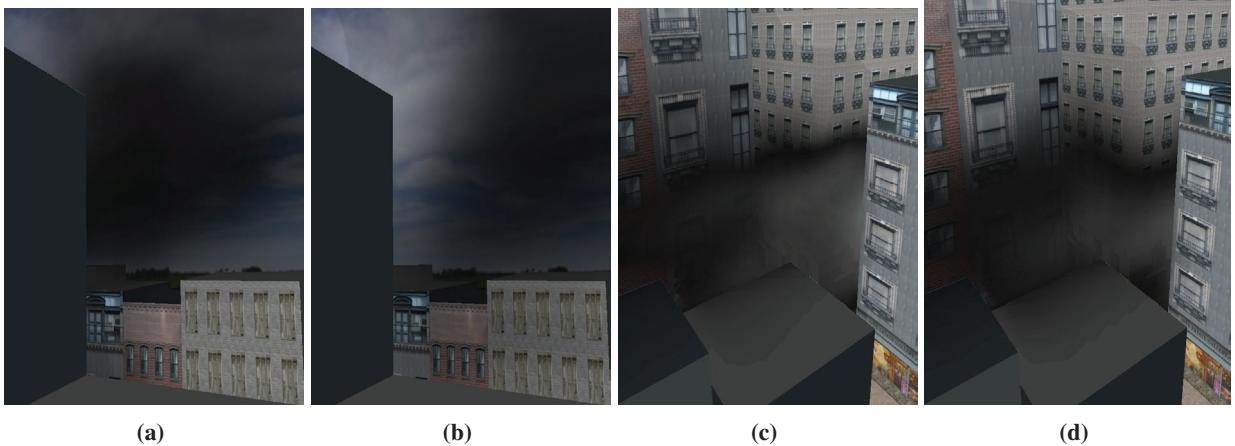
objects from the light source. This can be accomplished by calculating the shadow volume of the geometric mesh for the light source and using it in the fragment program to modulate the light intensity.

One approach to calculating the lighting volume involves a multi-pass algorithm using the OpenGL pipeline. In a preprocessing pass, the polygonal mesh of the buildings is rendered and only the surface depth is stored in the depth buffer. This pass can be executed very efficiently since modern GPUs double their effective pixel fill-rate when writing only to a depth buffer. In each of the following passes, a single slice of the lighting volume is calculated and stored in a 3D texture. For each fragment, we compare the depth with the surface depth so that occluded fragments have a zero lighting value. Due to hardware and OpenGL limitations, the fragment program cannot simultaneously read and write the 3D texture of the lighting volume. Therefore, in each pass, we sample the light intensity of the previous slice from the 3D texture and the result is used to compute the intensities of the current slice. After this operation, the current slice is copied from the frame buffer to the 3D texture. This algorithm can be implemented using only the fixed-function OpenGL pipeline [7], and can therefore run on 3D hardware without shading capabilities. Our initial implementation uses fragment shaders, but suffers from some inherent limitations of the algorithm nevertheless. During a single pass, every slice of the light volume is rendered to the frame buffer, copied to the 3D texture and sampled in the fragment program. A major performance limitation is that the OpenGL pipeline is stalled at the end of each rendering pass while waiting for the texture copy. This memory transfer also consumes the limited memory bandwidth of the GPU.

To address the performance issues, we have implemented a new method for calculating the lighting volume with NVIDIA's CUDA toolkit. CUDA is a C language environment for writing applications that execute in parallel on the processing units of supported NVIDIA graphics hardware [81]. The main advantage is that it alleviates some of the limitations of the graphics pipeline in respect to general purpose computations. In our application, it allows for memory scatter operations, or the ability of a computation

### 3. LBM FLUID SIMULATION AND VISUALIZATION ON OPTIMAL SAMPLING LATTICES

---



**Figure 3.13:** Snapshots of smoke simulation in the urban environment.

kernel to write to multiple memory addresses in a single thread. This is the exact feature needed to compute the lighting volume efficiently. For a volume of resolution  $x \times y \times z$ , we call a CUDA kernel with  $x \times y$  threads with each thread processing one lighting ray. The kernel samples  $z$  voxels on the path of each ray using a loop and stores the resulting values in the lighting volume.

With a single kernel call we obtain the entire lighting volume, which is much more efficient than the multi-pass OpenGL approach and relaxes the memory bandwidth requirements. Currently, CUDA cannot share buffers with OpenGL as they live in different contexts and therefore we need to copy the resulting volume data from the CUDA memory to an OpenGL pixel buffer object and then to a 3D texture. This process is much faster than the memory copy needed in the multi-pass algorithm since it only involves high-bandwidth GPU memory and does not use the comparatively slower PCI-Express or AGP buses for the data transfer.

Previous approaches trace rays in both the density volume and the lighting volume. Each ray needs to maintain the current sampling position and ray direction in both volumes, and at each sampling point, the program performs two trilinear interpolations, which is inefficient. We propose a method of tracing rays in a single volume only. During the computation of the lighting volume, the CUDA kernel samples the density volume with trilinear interpolation. For each point  $p$  in the lighting volume, in addition to the light intensity  $L(p)$  we also store the interpolated density  $s(p)$ . This reduces the number of texture sampling operation needed during the ray-casting pass.

Krüger and Westermann [64] have implemented ray-casting for volume rendering on the GPU with empty space skipping and early ray termination. However, empty space skipping requires an additional data structure (such as min-max octree) to be calculated and stored on the GPU. Our simulation generates different density volumes at each time step and recalculating the data structure for empty space skipping has

**Table 3.6:** Comparison of runtime performance for simulations using D3Q19 SRT LBM, D3Q13 MRT LBM, and fD3Q13 MRT LBM.

Simulation Type	Performance (Frames/second)
fD3Q13 MRT LBM	187
D3Q13 MRT LBM	108
D3Q19 SRT LBM	72

a high cost compared to the rendering of a single frame. Also, because of the phenomena we model, the dispersion volume is highly transparent and the opacity of most rays is not saturated. As a result, early ray termination is not efficient in this case. Therefore, our implementation is a single-pass ray-casting algorithm without support for empty space skipping or early ray termination.

### 3.5.4 Results

Figure 3.13 shows the result of our smoke simulation with a region in New York City, from 6th to 7th Avenue and from 57th to 59th Street. This urban model has 6 blocks and tens of buildings. An east wind blows through the urban environment and the hot smoke is released from an upwind position. Its motion is affected by the wind, buoyancy force due to the temperature, gravity due to the smoke density, and the geometry of the buildings.

The platform we use is a PC with a Geforce 8800 GTX graphics card of 768MB memory and two Xeon 3.6GHz CPUs (our CPU code is single-threaded). In Table 3.6, we compare three different implementations on the GPU: D3Q19 SRT LBM, D3Q13 MRT LBM, and our fD3Q13 MRT LBM. The simulation resolution for D3Q19 and D3Q13 is  $128 \times 64 \times 64$ . The simulation resolution for fD3Q13 is  $64 \times 64 \times 64$ , as only half of the lattice sites are needed for the FCC lattice. Performance does not increase linearly with the number on lattice sites due to fixed overhead associated with texture and data management on the GPU. Nevertheless, the fD3Q13 LBM is shown to be much more efficient than simulations using the two other lattice types. The major reason for this performance improvement is that fD3Q13 computation requires less memory access operations, while on the GPU the memory bandwidth is the dominant factor for computational performance. With the fD3Q13 MRT LBM, our simulation runs at 187 FPS. With the GPU rendering incorporated into the system, the simulation runs at 66 FPS. This allows for real time navigation in the urban environment.

### **3. LBM FLUID SIMULATION AND VISUALIZATION ON OPTIMAL SAMPLING LATTICES**

---

# 4

## Hierarchical Volume Rendering on Mixed Lattices

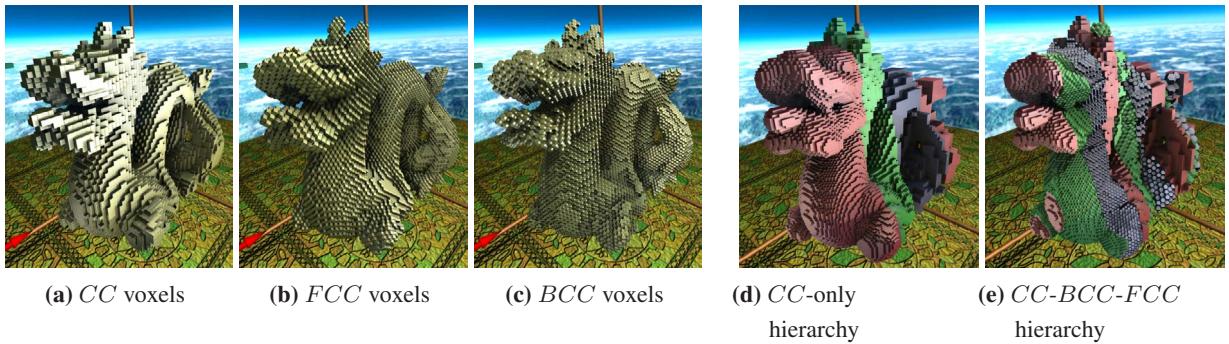
### 4.1 Introduction

Volume rendering generally operates on data that is sampled over the cubic Cartesian (CC) lattice. The properties of the CC lattice are well-known, and while its structure can be mapped efficiently to current GPUs, it exhibits sub-optimal sampling properties. Optimal sampling lattices, such as the body-centered cubic (BCC) and face-centered cubic (FCC) offer a more efficient representation of the 3D domain, which results in increased image quality over the CC lattice at similar voxel densities. We propose a mixed hierarchical domain representation that employs the CC, BCC and FCC lattices at successive layers in the hierarchy. Our approach takes advantage of the indexing and sampling properties of each lattice type and reduces the fan-out to 2, and at the same time it employs a simple reconstruction scheme that can be implemented efficiently on the GPU. Our mixed lattice rendering framework is evaluated with synthetic data and with complex scenes voxelized directly onto the mixed hierarchy. The results show that compared to CC hierarchies, we achieve smoother level transitions for improved image quality. At the same time, simpler inter-level filtering is used to improve the rendering performance.

The CC grid is used almost exclusively for multi-resolution volume rendering techniques. Binary transitions on the CC are the basis for octree subdivision schemes and they exhibit persistency of sample position at neighboring levels of the hierarchy. This is a desirable property since it simplifies the communications between different levels of the hierarchy. However, in the case of the CC lattice, the resolution difference becomes very large with a fan-out of 8 (i.e., a single level in the hierarchy contains 8 times more samples than the nearest lower resolution level). Although there are techniques that allow for arbitrary resolutions in the hierarchy, the relationships between the multiple levels of detail (LODs) of a single sample are lost. We

## 4. HIERARCHICAL VOLUME RENDERING ON MIXED LATTICES

---



**Figure 4.1:** Visualization of the Voronoi cells in the (a) CC, (b) FCC and (c) BCC lattices for an identical number of voxels. The optimal (BCC) and near-optimal (FCC) sampling lattices allow for improved spatial resolution at the same voxel density compared to the traditional CC lattice. Compared to the traditional octree hierarchy illustrated in (d) our mixed lattice hierarchy (e) allows for smoother level-of-detail transitions.

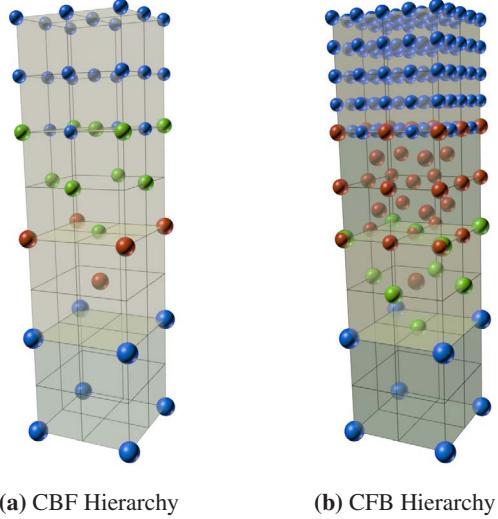
propose a mixed hierarchy for volume visualization that cycles through the CC, BCC and FCC lattices. The advantages of this multi-resolution representation are:

- Persistency of sample positions in the hierarchy, while allowing for a low fan-out.
- Fine control over the LODs, allowing for smooth transition and more adaptive antialiasing compared to binary hierarchies over the CC lattice.
- Higher rendering quality due to the use of optimal sampling lattices for intermediate LOD levels.

Our technique is also driven by the desire to devise a grid structure that provides a more adaptive, more continuous and localized refinement so that spatial detail can be better reproduced and the resolution more finely and locally controlled. Irregular grids perform well in this regard [116], but their rendering is cumbersome because they lack the regular structure which would allow for more efficient indexing. Our framework combines the advantages of both approaches - it adapts well to detail and at the same it has regular structure that allows for implicit addressing.

### 4.2 Mixed-Lattice Hierarchy Construction

Hierarchical volume rendering is generally performed on the CC lattice through a bricking scheme. Each brick has a fixed resolution that represents portions of the domain at different levels of detail. The octree subdivision of the volume has a fan-out of 8, which introduces a significant increase in spatial resolution at each step and leads to visual artifacts at the boundaries between bricks of different resolution during rendering.



**Figure 4.2:** Hierarchical domain refinement using the mixed CC (blue spheres), BCC (red spheres) and FCC (green spheres) lattices. The top layer shows the next CC level in the refinement.

Two different non-congruent lattice refinements have been proposed with certain unique properties [35, 53]. The  $CC \rightarrow BCC \rightarrow FCC$  (CBF) refinement is shown in Figure 4.2a and the  $CC \rightarrow FCC \rightarrow BCC$  (CFB) refinement is illustrated in Figure 4.2b. The CFB hierarchy supports a persistent refinement, in which each lattice site at the coarser resolution has a corresponding lattice site in the finer resolution. However, this is not true for the CBF structure. The additional point at the center of the BCC cell is not present after the refinement to the FCC lattice, but is reintroduced at the next CC level. This lack of persistence does not affect applications such as volume rendering that rely on a discrete reconstruction filter, but it requires special treatment when the hierarchy is used to transport data in lattice-based simulations. For example, natural phenomena simulations based on the lattice Boltzmann method (LBM) [125] can take advantage of the finer refinement afforded by the persistent CFB hierarchy without resampling the simulation domain at intermediate levels of detail. Here, intermediate CC, FCC and BCC levels in the hierarchy are constructed by resampling the closest higher resolution CC level. As a result, the point reintroduced at the CC level in the CBF hierarchy is an accurate representation of its phantom counterpart at the lower resolution FCC level.

The CFB hierarchy can be constructed by taking alternating layers from the CBF. Essentially, there is an implicit transition through an intermediate CC level. In terms of sampling density, a transition between two levels of CC with density 1 and 64 contains a single intermediate level in a hierarchy based solely on CC lattices, 2 intermediate levels in the CFB hierarchy (there is an implicit transition through a CC level) and 5 intermediate levels in the CBF hierarchy. Therefore, the CBF hierarchy offers the advantage of a more continuous domain representation. The goal is to achieve an adaptive refinement scheme that offers a

## 4. HIERARCHICAL VOLUME RENDERING ON MIXED LATTICES

---

well-defined structure, can support efficient signal reconstruction, and has a fine refinement step. In contrast, tetrahedral meshes can support arbitrarily fine refinement with well-behaved primitives [65], however the unstructured grid is expensive to render even with GPU-assisted visibility sorting [16] and dynamic adaptive refinement requires an expensive remeshing step.

In practice, we can combine the CC-only, the CFB and the CBF hierarchies in an adaptive manner by choosing the hierarchy type during the refinement of every CC brick in the data structure and storing the decision in the indexing structure. In order to simplify the index, we require two consecutive CBF refinements to be performed so that all three choices reach the same sampling density.

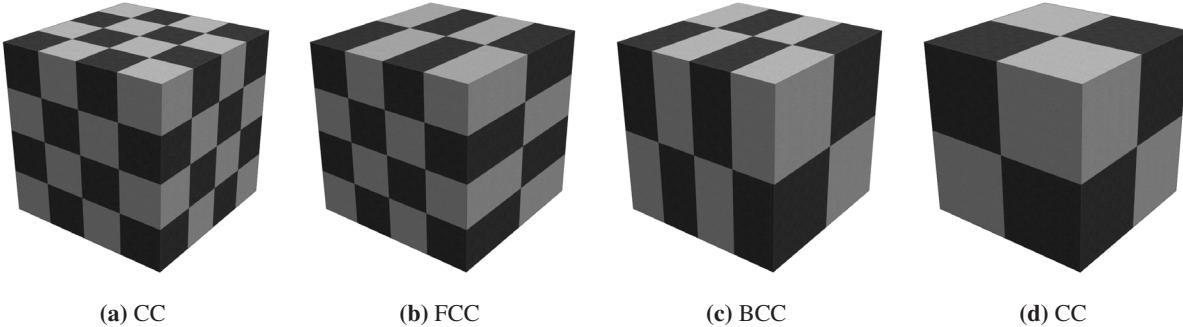
### 4.3 Mixed-lattice Rendering

The volume domain is split into bricks of a fixed size that represent varying regions depending on their level in the hierarchy and the lattice type. The coarsest level of the hierarchy is always defined over the CC grid in order to simplify the indexing and traversal stages. The index texture is also defined over CC and stores the offset of a brick in the brick cache, together with a single scale value which is used to compute the step size for raycasting and to determine the lattice type. Since bricks of fixed voxel dimensions span different physical spaces based on the lattice type, we construct the hierarchy in such a way that each brick spans exactly two bricks from the previous level. This is implemented in the indexing function for the lattice positions, where slices of the lattice are packed along the  $z$ -axis and the  $y$ -axis for the FCC and the BCC lattices respectively.

The FCC lattice can be described as a checkerboard arrangement of the volume domain (e.g., the light cells in Figure 4.3a) and we pack two FCC layers along the  $z$ -axis into a single Cartesian layer. As a result, a single FCC brick stored in a brick cache with Cartesian layout covers the same physical volume as 2 CC bricks. This packing is illustrated in Figure 4.3a and Figure 4.3b. For example, the volume covered by two  $64^3$  CC bricks in a  $64 \times 64 \times 128$  arrangement can be represented by a single  $64^3$  FCC brick.

For the BCC bricks, we choose a storage scheme based on Cartesian layers along the  $y$ -axis where every odd slice is offset by  $(0.5, 0, 0.5)$ . The dimensions of each slice are half of the original resolution of the CC brick and as a result, each BCC brick covers exactly 2 FCC bricks or 4 CC bricks (Figure 4.3b and 4.3c). Similarly to the FCC case, a single  $64^3$  BCC brick covers the same physical space as 4 CC bricks in a  $64 \times 128 \times 128$  arrangement. Finally, the lower resolution CC brick in Figure 4.3d covers the 2 BCC bricks and the cycle continues until the entire domain is represented by a single brick.

For a given brick size, the brick hierarchy is constructed by splitting the original volume into the level 0 bricks in the hierarchy and combining  $2 \times 2 \times 2$  sets of bricks and the adjacent boundary voxels to produce the higher levels. For convenience, we generate the next FCC, BCC and lower resolution CC bricks by



**Figure 4.3:** Hierarchical storage assuming bricks with identical dimensions for the CC, FCC and BCC levels. Each lower resolution brick spans exactly 2 bricks from the higher resolution level in the hierarchy.

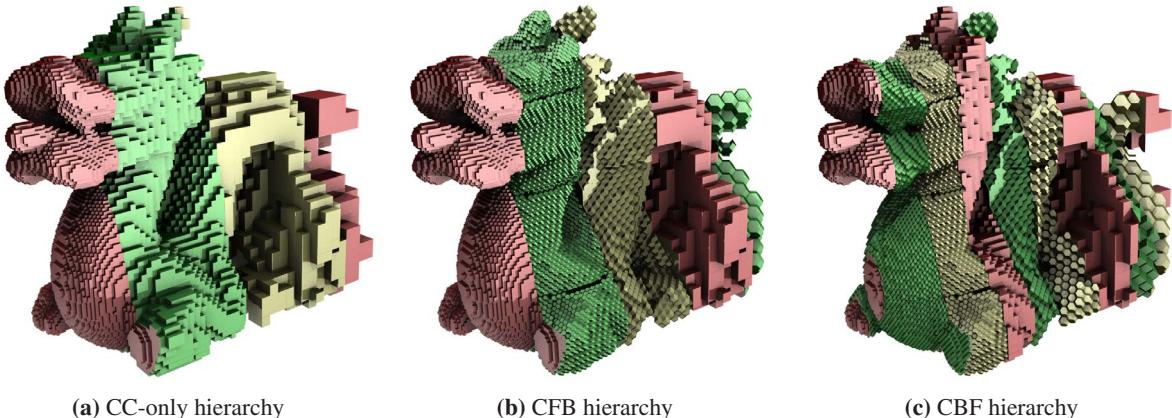
downsampling directly from the high resolution CC bricks. If brick overlapping is specified during the construction stage, the boundary information of each brick is updated with voxels from the 26 neighbors for each level of the hierarchy.

Our LOD selection is guided by a distance-based heuristic [74], which computes the distance from the camera to the centers of each brick and determines the diameter of the brick’s bounding sphere projection onto the image plane. A threshold is then applied to determine whether the current brick needs to be replaced in the index by its children in order to maintain consistent visual acuity. The index texture and the brick cache are maintained dynamically, similarly to the approach described by Beyer et al. [9]. The default dimensions of the index match the size of the finest resolution brick layer in the hierarchy or the user can specify a multiple of that size. High resolution index textures allow for finer refinements in smaller sections of the volume; however, the cost of managing the index increases proportionally. At the basic level, our framework allows the user to specify a LOD level for each entry in the index. We traverse the hierarchy to find the appropriate brick that covers the current index position and store its offset in the brick cache texture and a scaling factor. Since a given index entry can refer to a part of a brick, we also store the coverage of the full brick in terms of the number of index entries. This is performed mostly for convenience since it implicitly encodes the information about the brick’s lattice type and simplifies the computations in the GPU shader.

Additional indexing features are built on top of this base functionality. In our current implementation, the index is rebuilt asynchronously in a separate thread from the main rendering loop. The code can refine specific blocks of indices based on the distance from the camera or some manually specified criteria or heuristic. We also enforce the consistent cyclic structures of the CBF and the CFB hierarchies and only allow hierarchy type changes after the CC levels. This index maintenance is more involved than in the

## 4. HIERARCHICAL VOLUME RENDERING ON MIXED LATTICES

---



**Figure 4.4:** The Dragon dataset is represented by a  $128^3$  distance field and we visualize the Voronoi cells of the underlying voxels. The hierarchies use  $32^3$  bricks and a  $16^3$  index. Compared to the octree hierarchy (a), both the CFB (b) and the CBF (c) hierarchies allow for smoother transitions between the levels and take advantage of the optimal and near-optimal sampling properties of the BCC and FCC lattices. Successive LOD levels are colored cyclically in red, green and yellow, starting with the finest resolution CC level.

traditional CC-only octree scheme; however, in practice the additional workload has negligible effect on the perceived performance of the system.

The index is sampled during ray integration on the GPU with a nearest neighbor filter to determine the current sampling position from the brick cache and to select the appropriate filter for the current lattice type. Figure 4.4 illustrates the rendering of the same fully-bricked volume data with different index textures, allowing for dynamic selection between the CC-only, the CFB and the CBF hierarchies. For illustrative purposes, the LOD selection uses distance along the  $x$ -axis only. The rendering of the isosurface is performed using the corresponding nearest neighbor filters for each lattice type to illustrate the mixing of the different voxel unit cells.

One of the main challenges in rendering the hierarchical representations is the handling of the boundary between two grids of different resolution. In our case, the fully-bricked volume data is defined over the *CBF* hierarchy, and we enforce single-level transitions between adjacent bricks during the construction of the index. As a result, transitions between resolution levels are well defined and all possible interfaces are illustrated in Figure 4.2a. Each brick includes a ghost layer of cells to allow for reconstruction filters that sample the first-ring neighbors and also to allow for interlevel interpolation. In the CBF hierarchy, higher resolution layers only add sample points to lower-resolution levels (the one exception in the *BCC*  $\rightarrow$  *FCC* transition does not affect the boundary region), the higher resolution level interpolate directly from the data in its ghost layer. In the opposite direction, high resolution positions that are not in the lower resolution lattice can still be sampled. If we allow only smooth level transitions, each level has information about the

topology of its neighbors and can compute the coordinates for an NN texture fetch. This sample is then weighted into the reconstruction of the boundary in the direction from lower resolution to higher resolution data. In certain cases, the fine refinement in the CBF hierarchy may allow for this step to be skipped with only minimal visual artifacts, for example in areas of low gradient for volume data or small curvature for isosurfaces.

## 4.4 Implementation and Results

### 4.4.1 Reconstruction Filters

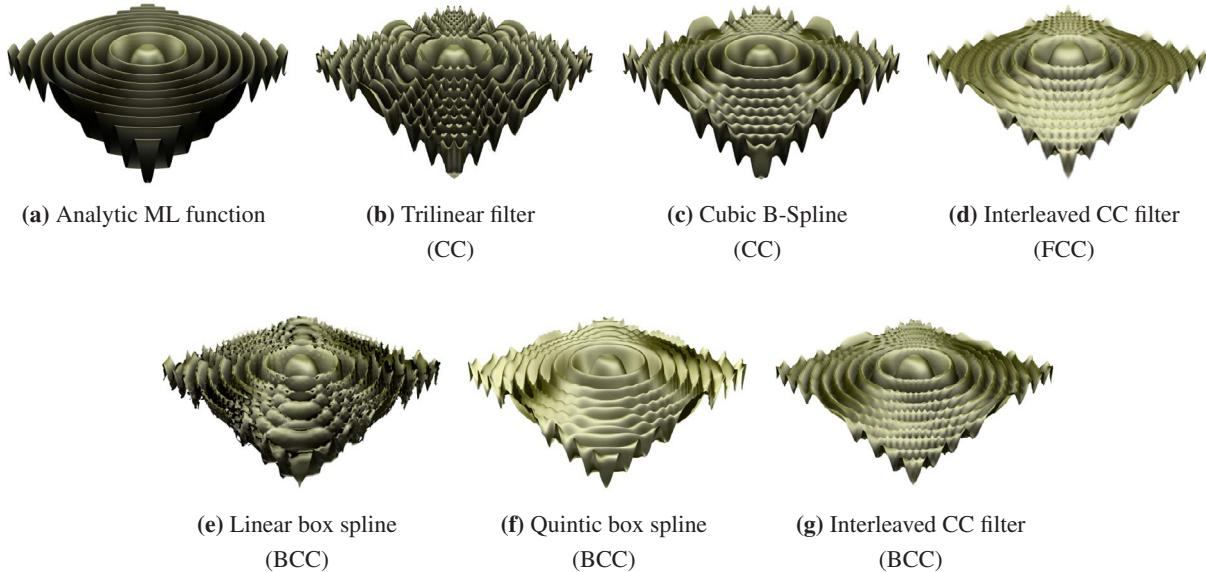
We implement reconstruction filters for the CC, BCC and FCC lattices that are suitable for applications requiring interactive to real-time performance. GPUs natively support trilinear filtering for 3D textures for both 16bit/channel and 32bit/channel data. Older hardware such as the NVIDIA G7x supports only nearest neighbor texture fetches for 32bit data, in which case we perform the trilinear interpolation manually in the raycasting fragment shader. The cost of the filter rises significantly, from 1 linear texture fetch to 8 nearest neighbor (NN) texture fetches and 3 lerp instructions. We also implement a cubic B-spline filter for CC data, which can be evaluated efficiently when trilinear interpolation is available on the GPU [95]. In this approach, the offset of a sampling point from a lattice site is used to compute 8 offsets and weights, which allow for the evaluation of the cubic B-spline with only 8 linear texture fetches. The offsets and weights can be read from textures or computed directly, depending on the performance characteristics of the hardware.

Figure 4.5 illustrates the reconstruction quality of the filters using the Marschner-Lobb (ML) function [75] sampled over the three lattice types. The ML function is initially sampled on a  $40^3$  CC grid and the resolution for the other sampling lattices can be computed based on the volumes of their respective unit cells. For example, the FCC and BCC achieve comparable reconstruction performance to the CC using 71% and 77% the number of samples respectively. However, since our approach involves a brickling scheme with a constant brick size, we are more interested in the comparison between lattices using an approximately equal number of voxels in their 3D domains.

Both the BCC and FCC filters reduce the oscillations along the concentric peaks that are present in the CC reconstruction as shown in Figure 4.5g and 4.5d. Arguably, the results are perceptually better even at a comparable number of samples. The ML function exhibits thin features that are parallel to a major axis in CC, and these features are more difficult to resolve on lattices that do not have a generating vector in that particular direction. All isosurfaces in these figures are computed directly on the GPU using similar parameter ranges to those used in rendering the other non-analytic examples. As a result, small artifacts are visible along the concentric peaks due to the limited precision of the GPU compared to CPU raytracing implementations.

## 4. HIERARCHICAL VOLUME RENDERING ON MIXED LATTICES

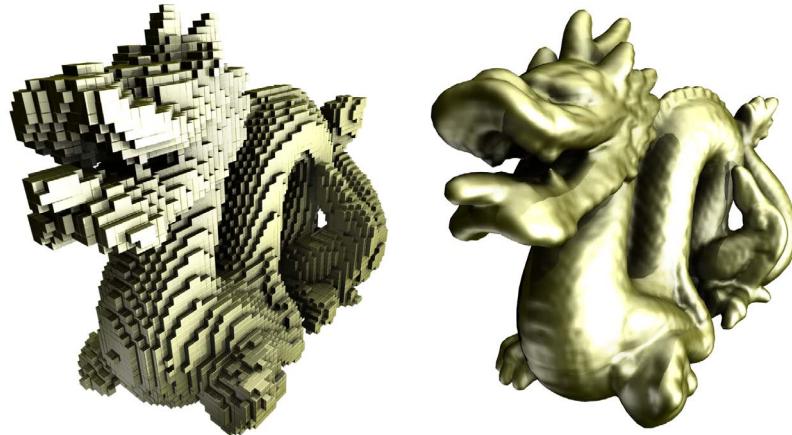
---



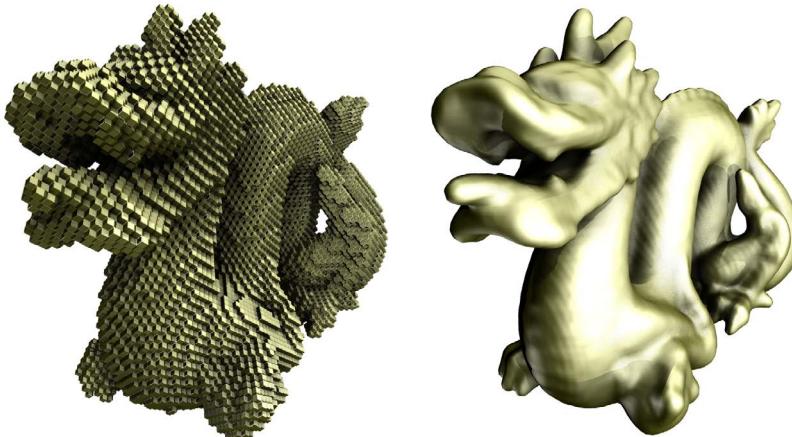
**Figure 4.5:** The Marschner-Lobb (ML) function is sampled on a  $40^3$  CC grid, a  $50 \times 50 \times 25$  FCC grid and a  $32 \times 64 \times 32$  BCC grid. We compare the different reconstruction filters implemented on the GPU in terms of their ability to resolve the thin structures in the ML dataset.

It is not immediately obvious that any reconstruction advantages on analytic data translate into improved quality on real-world data (i.e., not axis-aligned). Figure 4.6 shows the reconstruction of the Stanford Dragon from the distance field obtained over the different lattice types. We first use the nearest neighbor filters, which illustrates the different arrangement of the samples in CC, BCC and FCC. All three datasets contain approximately the same number of voxels and both BCC and FCC show marginal improvement in resolving curved surfaces. However, the differences are significantly less pronounced than in the synthetic ML dataset. The advantage of the optimal and near-optimal sampling lattices is that they preserve comparable detail to the CC lattice at approximately 70% of the voxel count. This feature will become more important when we discuss the construction of the mixed lattice hierarchy in the following sections.

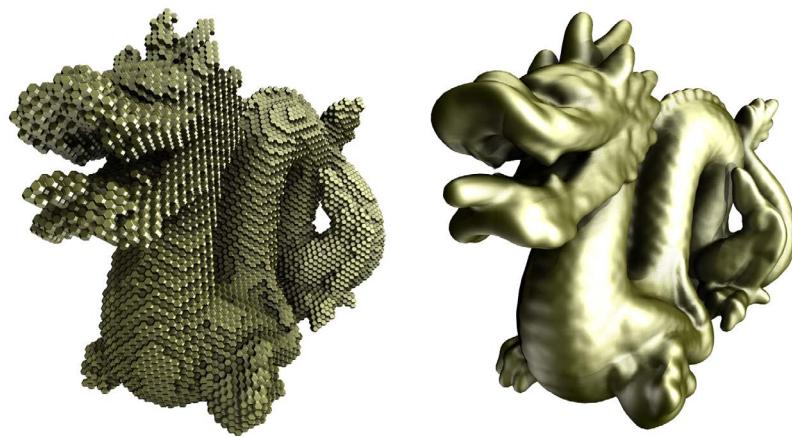
We evaluate the performance of the different filters on datasets with comparable voxel counts. The resolution of the rendered image is  $512 \times 512$  and all performance and quality optimizations are disabled in the shader (e.g., early ray termination, pre-integrated transfer function, etc.) to isolate the reconstruction performance. Our benchmarks are run on the NVIDIA Quadro FX 2500M, which is a DX9-class GPU with 24 fragment shaders and 8 vertex shaders. The NVIDIA Quadro FX 5800 is a newer generation GPU with 240 unified shaders. The theoretical performance capabilities of the two GPUs are summarized in Table 4.1. The reconstruction filters are implemented in the Cg language using version 3.0 of the NVIDIA Cg Toolkit. All shaders are compiled with the latest shader profiles supported by the hardware and the performance



(a) CC reconstruction



(b) FCC reconstruction



(c) BCC reconstruction

**Figure 4.6:** We compute the distance field for the Stanford Dragon mesh dataset over the CC, BCC and FCC lattices at similar voxel densities (equivalent to  $64^3$  voxels). Each pair of images illustrates the Voronoi cells of the lattice positions on the left and the reconstructed isosurface on the right.

## 4. HIERARCHICAL VOLUME RENDERING ON MIXED LATTICES

---

**Table 4.1:** Comparison of the performance characteristics of the NVIDIA Quadro FX 2500M (GPU1) and NVIDIA Quadro FX 5800 (GPU2) GPUs.

	Shaders	GFLOPs	Memory	Bandwidth
GPU1	24f+8v	230	512 MB	38.4 GB/s
GPU2	240u	933	4096 MB	102 GB/s

**Table 4.2:** Relative performance of the different filters on the GPU. The baseline is the CC filter with non-native interpolation running on the NVIDIA Quadro FX 2500M (GPU1).

Filter	GPU1	GPU2	speedup
CC linear (1 lin)	2.80	30.30	10.8
CC linear (8 NN)	1.00	5.86	5.9
CC cubic (8 lin)	0.70	5.88	8.4
<b>BCC (4 NN)</b>	<b>0.77</b>	<b>10.32</b>	<b>13.4</b>
FCC linear (4 lin)	2.13	11.18	5.2
FCC linear (16 NN)	0.15	1.87	12.5

numbers are presented in Table 6.1. The results for the newer NVIDIA Geforce GTX 470 are presented in Section 6.2.4.

For the cubic CC filter, the Quadro 2500M uses pre-computed filter weights (computing them in the shader is 61% more expensive), while the Quadro 5800 computes the weights directly (fetching the pre-computed values is 44% more expensive). This general trend also applies to the remainder of the benchmarks, with the newer architecture allowing for larger gains when the filter employs denser computations. This is particularly advantageous for reconstruction on the BCC and FCC lattices which require more complicated computation and branching than the other filters (e.g., sorting of 4 scalars in the shader).

For higher performance, an alternative approach involves reconstruction of the missing sampling points along the major axes in the FCC lattice by simple linear interpolation of the 6 samples along the major axes, followed by trilinear or barycentric interpolation on the resulting CC lattice. Although the native linear interpolation of the GPU can be used to achieve high reconstruction performance, the volume data has to be pre-processed and stored as a CC volume, which negates the advantage of the original optimal sampling lattice. Another algorithm proposed by Qiao et al. [87] decomposes the FCC lattice into 4 interleaved CC lattices (see Figure 2.1c) and stores them in the RGBA channels of a 3D texture. During rendering, the average of 4 tri-linear interpolations is used to reconstruct the volume. Since the GPU hardware is used for the texture fetches, evaluation is very fast, achieving performance comparable to the linear filter on the BCC lattice (Table 6.1 lines 4 and 5). However, the resulting kernel has a very wide support of up to 32 neighboring points, smooths the data, and may produce artifacts in regions of high gradient due to the averaging

operation. This excessive smoothing is illustrated in Figure 4.5d, where the peaks and troughs in the ML function are significantly compressed in relation to the corresponding structures in the CC reconstruction with the cubic B-Spline filter (Figure 4.5c). In our framework, the interleaved CC reconstruction filters for BCC and FCC are implemented by manually computing the trilinear interpolation over the interleaved CC lattices, while preserving the original data storage scheme in the brick cache.

### 4.4.2 Hierarchical Rendering

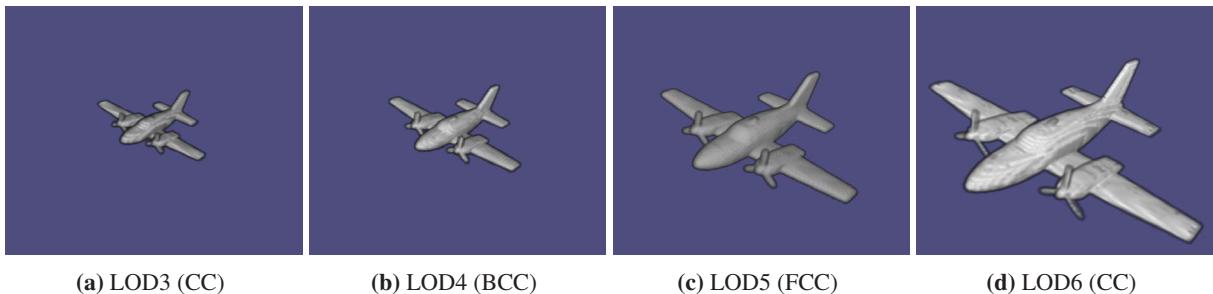
Voxelization of large mesh scenes is one of the sources of volume data used to evaluate the reconstruction filters since the differences are more clearly visible compared to density volumes. This voxel representation is used during direct volume rendering and also defines the boundary conditions for physically-based natural phenomena simulations. For the CC lattice, binary voxelization is performed on the GPU [32]. In order to support all three lattice types with a single voxelization technique, we store the geometry of the scene in a kd-tree and then obtain a solid binary voxelization using ray-tracing with parity-checks to distinguish positions inside and outside the scene geometry. The construction of the kd-tree uses the surface area heuristic (SAH) [117], which improves query performance by up to a factor of 2 compared to the spatial median heuristic. For filtered voxelization, we compute the distance field near the geometry using a nearest-neighbor query on the kd-tree. The computed distance is to the true nearest neighbor on the geometry primitives. The density of a voxel is then linearly proportional to the clamped value of the distance field [115]. The voxelization is performed directly on the individual levels of the hierarchy, mimicking our proposed use of the mixed lattice hierarchy as a simulation domain for the lattice Boltzmann method (LBM).

Using optimal sampling lattices for intermediate layers improves the rendering quality and reduces the perceived difference in sampling densities between adjacent regions in the data. Figure 4.7 shows the results of hierarchical rendering using the CBF hierarchy with a distance-based LOD selection. For Figure 4.8, we build a hierarchy based on the CC lattice alone, where each level contains approximately the same number of voxels as the corresponding level in the mixed lattice. In terms of rendering quality, the BCC reconstruction exhibits less artifacts than the equivalent CC reconstruction. The FCC reconstruction is also subjectively better, however it is not clear how much is due to the near-optimal sampling properties of FCC or the smoothing property of the filter. Barycentric FCC interpolation will provide a more fair comparison, however an efficient GPU implementation remains a significant challenge.

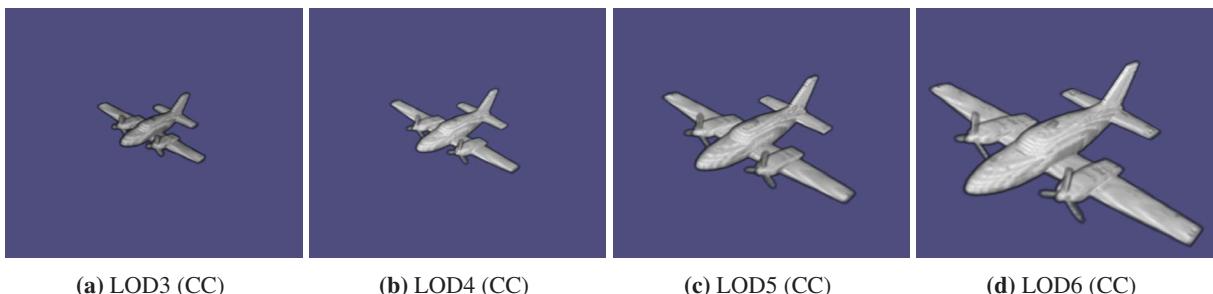
For the visualization of traditional volume data, such as medical scans or the results of lattice-based CFD simulations (see for example Chapter 3), our mixed lattice brickling scheme is implemented as an additional transparent step to the volume processing pipeline. We build the hierarchy based on the resampling framework presented in Section 4.2 when a new volume is loaded and serialize all the data to a binary file to speed up future access operations. The sampling position calculation in our single-pass raycasting implementation

#### 4. HIERARCHICAL VOLUME RENDERING ON MIXED LATTICES

---



**Figure 4.7:** Filtered voxelization on the CBF hierarchy with 7 LODs. Here we illustrate the transition between two successive CC levels.

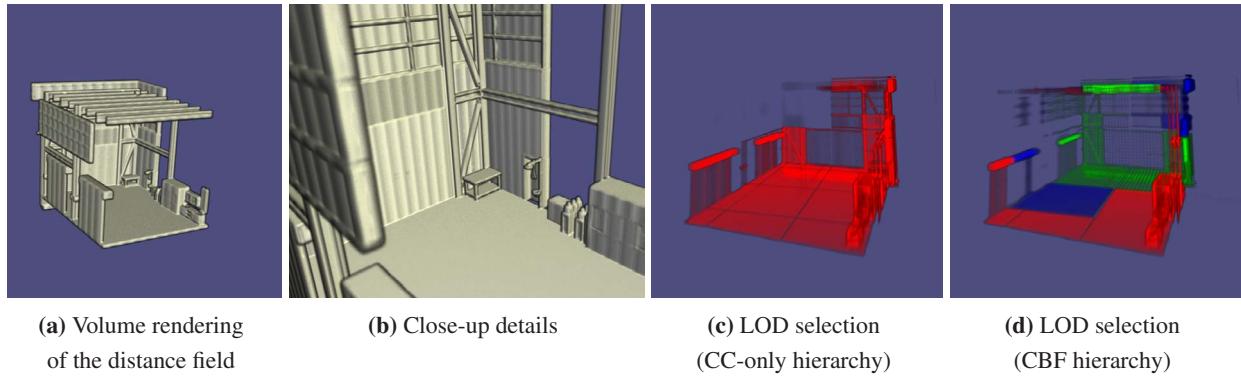


**Figure 4.8:** Filtered voxelization on a non-persistent CC hierarchy shown for comparison. Voxel density at each level is approximately equal to the density at the corresponding level in the mixed hierarchy. This hierarchy is not practical due to the non-uniform coverage by bricks of constant size. Also, compared to Figure 4.7, the advantage of optimal and near-optimal sampling levels is lost.

is augmented to support the indexing texture and the sampling function selects the correct filter based on the lattice type of the current brick. All the different filters discussed so far are available to the user based on image quality and performance requirements. The rest of our pipeline remains unchanged, including the support for different mesh- and volume-based rendering effects, stereoscopic rendering and distributed visualization environments.

Figure 4.9 illustrates the hierarchical rendering of large real-world datasets. The architectural scenes are of particular importance for the natural phenomena simulation, since any advantage of the mixed lattice hierarchy directly translates into better boundary conditions and improved rendering of the simulation results in the form of smoke or fluid interfaces. In Figure 4.11 we examine the anti-aliasing properties of the CBF hierarchy for the rendering of complex architectural surfaces. The image in Figure 4.11a uses only the original CC data, leading to aliasing artifacts as the distance to the camera increases. In comparison, the CBF hierarchy allows for smooth transitions to filtered lower resolution representations of the data, such as the final BCC hierarchy level shown in Figure 4.11b. Figure 4.9 and Figure 4.10 further show snapshots during interaction with both voxelized and medical data, where the indexing texture contains references to

#### 4.4 Implementation and Results

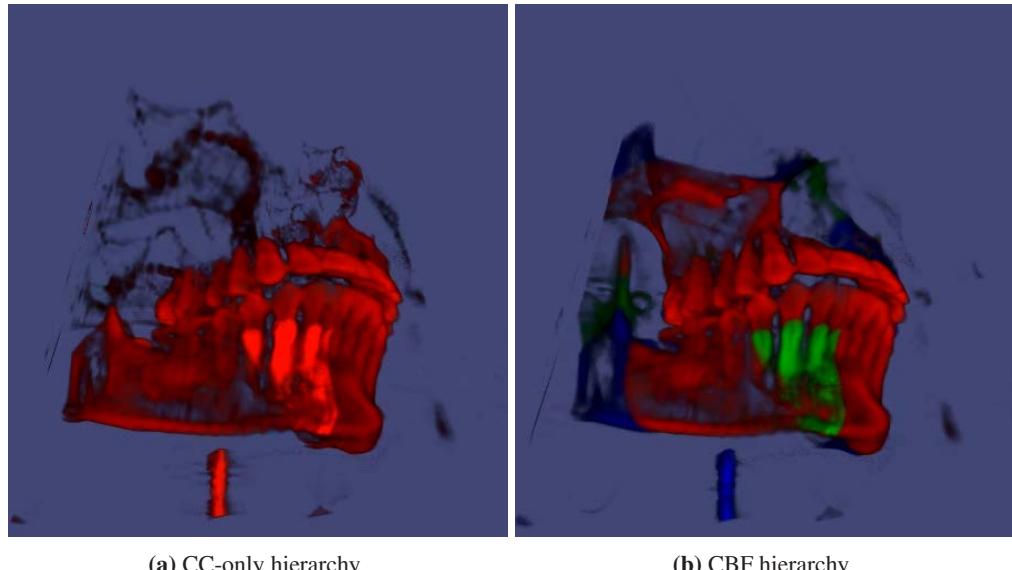


**Figure 4.9:** Hierarchical volume rendering of a large voxelized scene. (a) and (b) show high-quality direct volume rendering of the distance field. (c) illustrates the LOD selection with a 4-level CC hierarchy where brightness indicates sample density. (d) uses the CBF hierarchy with 12 levels where red, green and blue indicate the CC, BCC and FCC levels respectively and brightness denotes the sample density.

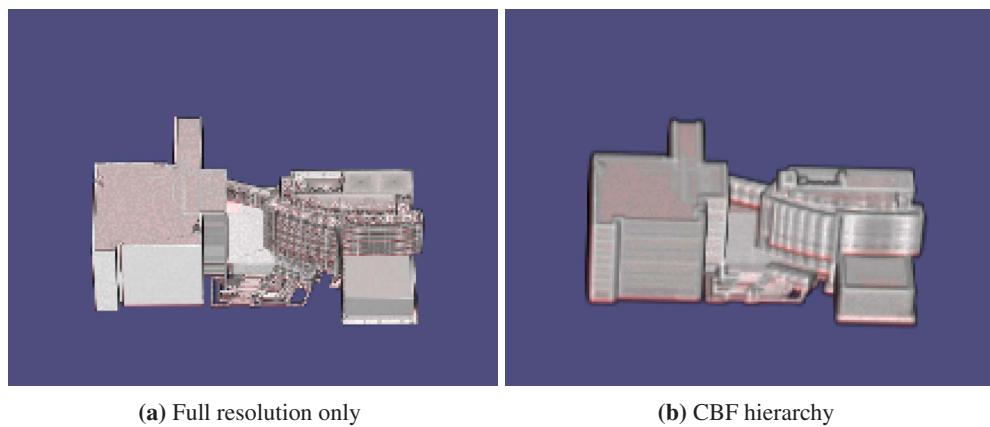
bricks of different hierarchy levels and lattice types. In both cases, the color encodes the lattice type of the current brick (red - CC, green - BCC, blue - FCC), and the increasing color intensity denotes a higher hierarchy level.

#### 4. HIERARCHICAL VOLUME RENDERING ON MIXED LATTICES

---



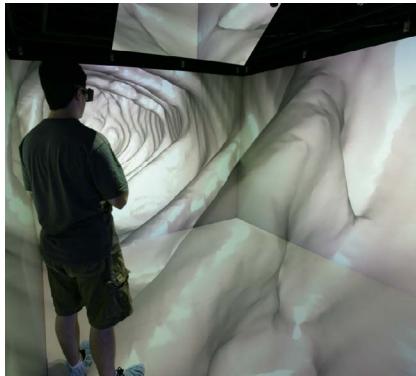
**Figure 4.10:** Volume rendering of the Skull dataset using (a) the CC octree with 4 levels, (b) the CBF hierarchy with 12 levels and the same LOD selection heuristic.



**Figure 4.11:** (a) The full resolution dataset is rendered from a distance, which results in aliasing artifacts. (b) The CBF hierarchy with an LOD selection allows for finely-adaptive antialiasing.

# 5

## Immersive Environments



(a) The Immersive Cabin



(b) The Reality Deck

**Figure 5.1:** (a) The Immersive Cabin is a 5-sided CAVE with an automatic door and a high-end active stereo projector system. (b) The Reality Deck is a large 416 display installation with an immersive layout and an aggregate resolution in excess of 1.5 billion pixels.

## 5. IMMERSIVE ENVIRONMENTS

---

### 5.1 The Immersive Cabin

We present a visualization and interaction framework for the Immersive Cabin (IC), which is a 5-wall CAVE-like immersive 3D environment. The back-end rendering is provided by a cluster of 5 workstations with 2 NVIDIA Quadro GPUs each. We utilize a pair of projectors for each wall and an external LCD shutter system that is synchronized with active stereo glasses. Each workstation produces a synchronized pair of stereoscopic images for a single wall in the IC. If only one GPU is dedicated to rendering, the second one is utilized as a computational device using C/C++ and the NVIDIA CUDA extensions. In addition to the depth cues and surround immersion from the visualization system, we use wireless optical head and hand tracking to further enhance the data exploration capabilities. Combined with a range of interaction and navigation tools, our system can support a variety of interactive applications, including architectural and automotive pre-visualization, urban planning, medical imaging, and simulation and rendering of physical phenomena.

The size of data has grown in recent years to the point where traditional techniques for its visualization and analysis are not sufficient. It is not uncommon to be presented with multi-gigabyte volumetric datasets from fluid dynamics simulations, high resolution CT scans, or data structures so complex that projections onto 2D display are ineffective. A number of techniques have been developed for the visualization of large-scale complex data. Facilities such as the CAVE [24] and devices such as Head Mounted Displays (HMDs) provide a much larger field of view into a virtual environment and further improve the perception of spatial relationships between data by utilizing stereoscopic rendering. Despite their advantages, both solutions present a number of challenges that ultimately limit their appeal and traction outside the visualization field. HMDs are usually bulky and heavily wired, and although they present a visually unobstructed view into the data, their usage is associated with eye strain, headaches and other medical issues. CAVEs on the other hand offer a more natural visualization environment that is amenable to augmented reality applications. However, building fully enclosed facilities remains a challenge and many installations have a limited number of display surfaces. Although fully enclosed CAVEs exist [47], they present a number of engineering challenges that result in significant cost increases.

We have proposed and constructed the IC [89] as a robust and affordable design for a fully enclosed visualization environment. The IC consists of 4 projection walls and a floor and the design is compatible with multiple projectors per surface and the new 120Hz stereo projectors. Our design utilizes an automatic door to provide access into the IC, which compares favorably cost-wise with facilities that use heavy duty machinery to move the back wall. Our facility has been used for a number of projects and we present a summary of the advances in terms of the software framework, the IC instruments and some of the applications.

### 5.1.1 Equipment and Construction

The IC is a compact, fully enclosed visualization environment with 5 projection surfaces and an automatic door as part of the back wall. Each projection node is equipped with the following:

- Dual Hitachi SXGA+ projectors
- Beacon SX+ shutter and synchronization
- IR emitter

The layout of the installation is illustrated in Figure 5.2 and additional details about the construction are available in the paper by Qiu et al. [89]. The shutters in front of the projectors are connected to the Beacon system and operate at 120Hz. The IR emitter is also connected to the Beacon system and allows for synchronization with the LCD shutter glasses. The 5 Beacon systems are daisy-chained so that all the shutters in the IC are synchronized. Each pair of projectors is driven by a single workstation with two NVIDIA Quadro FX 4600 boards. Although perfect synchronization between the GPUs and the projectors is possible with the use of NVIDIA G-Sync boards, our setup can operate with software-based framelocking. The G-Sync boards are mandatory for installations with active stereo but without an external shutter system, for example when using native 120Hz projectors for stereoscopic rendering.

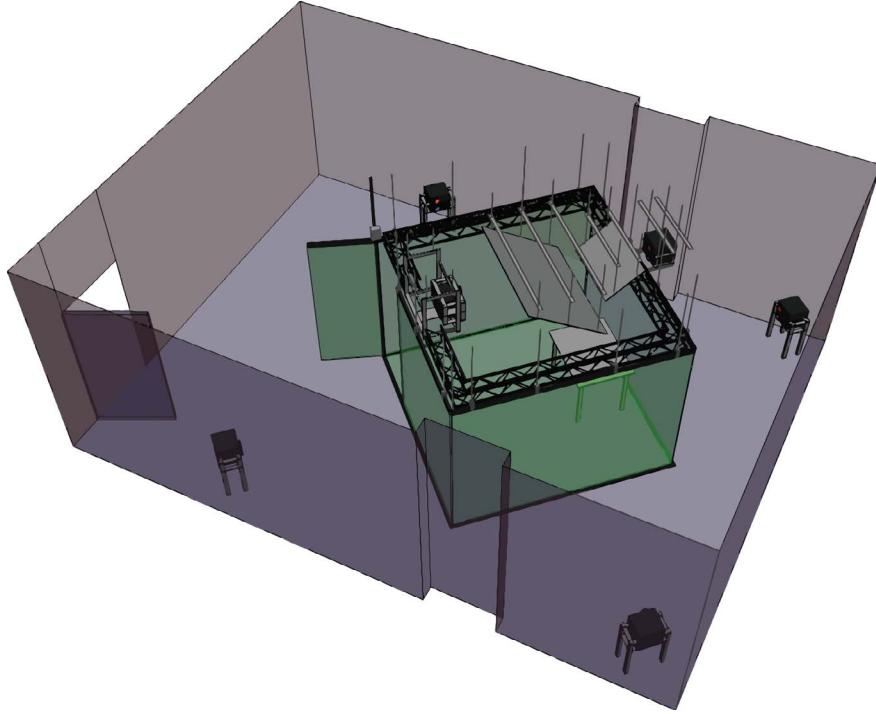
Our cluster contains a single head node which is responsible for managing all the machines and processing all user interactions. In addition, we have a 32-node cluster with dual CPUs at each node and an NVIDIA Quadro FX 4500 graphics board. These machines are not used for direct output to the projection surfaces in the IC and are instead dedicated to off-line rendering and natural phenomena simulations.

The IC utilizes an infrared tracking system from NaturalPoint that contains 8 IR cameras with wide-angle lenses, 2 synchronization hubs and the ARENA motion capture software. The cameras are mounted along the top edge of the IC walls and provide sufficient coverage for head and gesture tracking throughout most of the interior space. The synchronization hubs are connected to the sync signal of the Beacon shutter system so that the pulsed IR illumination from the cameras does not interfere with the IR signal for the shutter glasses. We use between 3 and 5 markers to define rigid bodies for the tracking software and special care has to be taken so that the marker arrangements are sufficiently different in the topological sense. For the LCD shutter glasses we use a 5 tape marker pattern that reduces the occurrence of optical occlusions, while for other objects, 3 markers may be sufficient. In the case of hand gloves, protruding spherical markers are used as occlusions prevent the efficient use of flat tape markers.

Figure 5.3 illustrates the coverage areas for tracking that we achieve in the IC. The blue cloud represents the area that is seen by a specified number of cameras and we provide the results for coverage by 3, 6 and 8 cameras. A point needs to be seen by at least 3 cameras in order to obtain its position in 3D space and

## 5. IMMERSIVE ENVIRONMENTS

---

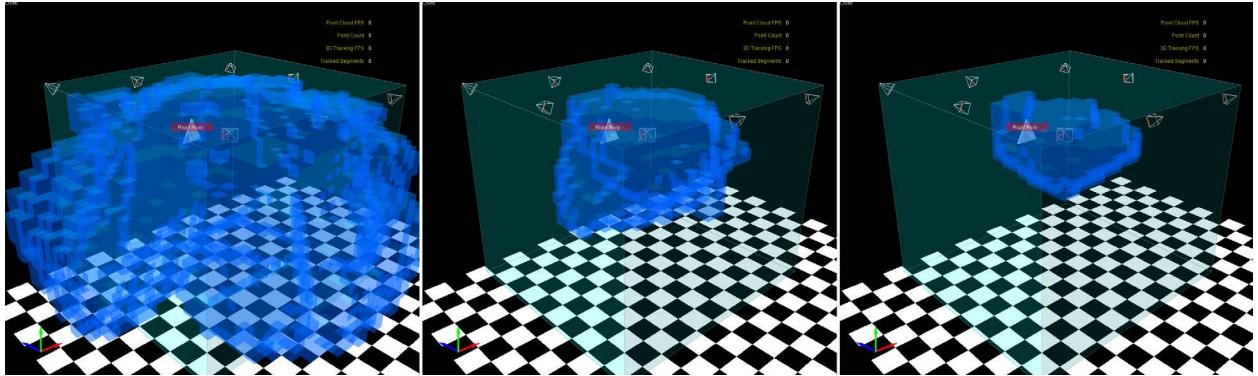


**Figure 5.2:** Diagram for an installation of the Immersive Cabin.

a rigid body can contain between 3 and 6 points. Although most of the volume in the IC is covered by at least 3 cameras, the illustration in Figure 5.3 does not account for optical occlusions which are a significant problem. Rather, we have optimized the camera placement for optimal tracking near the center of the IC and toward the front screen.

Our system supports standard input devices such as keyboards and mice, although they are not particularly suitable for the exploration of virtual environments. We primarily rely on wireless gamepads as the main navigation tool and the framework currently supports the Logitech Rumblepad 2 and the Xbox 360 wireless controllers. In our experiments, the 3DConnexion wireless SpaceMouse has provided superior performance for medical visualization and architectural applications as it combines 6 degrees of freedom in a very intuitive and compact device.

The IR tracking system is deployed to provide a different modality of user interactions. Head tracking is used to interact with the visual system directly by allowing the user unobstructed movement inside the volume of the IC. The position and orientation of the user are both taken into account when rendering the virtual world, which allows, for example, peeking around a corner or simulating an object that is inside the IC. The input data from the various devices is optionally modified using the head orientation from the tracking system to provide a more natural user experience. The orientation is used to correct the direction of the virtual translation movements so that users can steer by rotating and tilting their head. In addition,



**Figure 5.3:** Blue cloud represents the tracking coverage areas seen simultaneously by (left to right) 3, 6 and 8 cameras. The positions near the center of the IC that are seen by 6 cameras are optimal for head and gesture tracking since optical occlusions are minimized.

the tracking system allows the development of a more intuitive user interface that relies on natural gestures instead of tracked devices.

### 5.1.2 Rendering Framework

Designing a flexible framework for distributed GPU rendering is generally a challenging task. Compared to a traditional 3D engine, our system needs to support a network of workstations with single or multiple rendering pipes, multiple display surfaces per node and stereo rendering. Support for more traditional display layouts, such as multiple angled monitors on a desk, is essential for debugging and application development. The Chromium library [52] provides support for rendering on a cluster of nodes by directly manipulating the stream of rendering commands issued by a graphics API such as OpenGL. A set of stream filters are used at each rendering node to create a parallel graphics system in which the visualization application is not necessarily aware of the underlying rendering platform. While this approach greatly simplifies the process of creating Virtual Reality applications, network bandwidth becomes a major bottleneck with modern applications that may utilize hundreds of high resolution textures, advanced shaders and multiple rendering passes per frame [102]. CGLX [30] follows a similar approach by providing a replacement for the OpenGL library that abstracts the visualization surfaces. VRJuggler [10] and Equalizer [31] on the other hand operate at a higher level using a runtime library to manage multiple instances of the application on the cluster computers. With this approach, only the user input is transmitted over the network which results in significant bandwidth savings. On the other hand, the visualization framework is less transparent to the programmer and may require significant changes to existing code.

Our framework for distributed rendering is based on a master-slave model similar to the architecture of VRJuggler, in which instances of the application are executed on each of the rendering nodes. The

## 5. IMMERSIVE ENVIRONMENTS

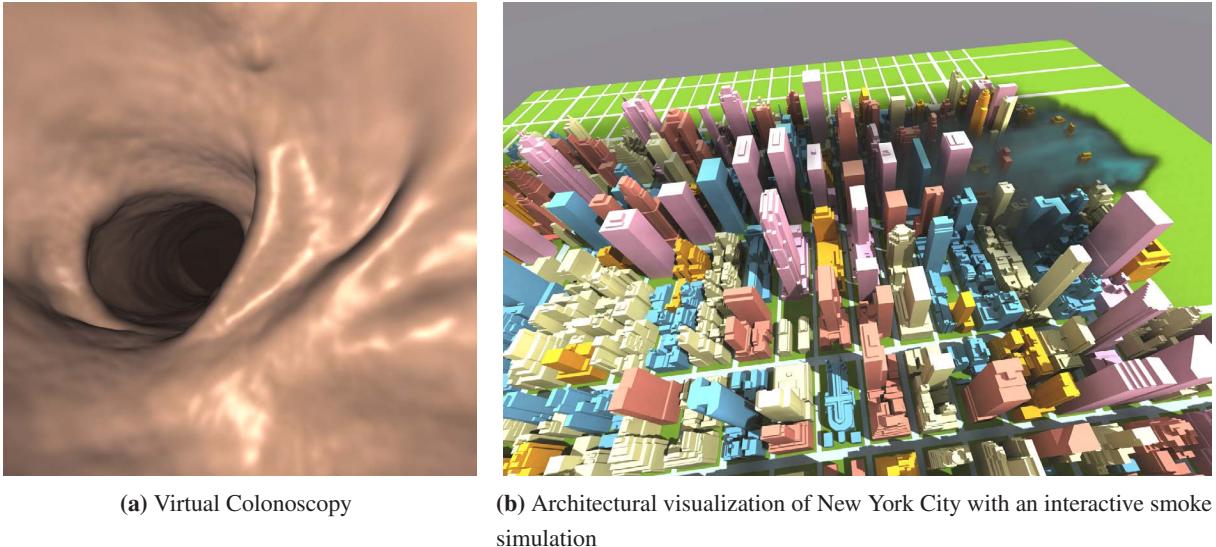
---

user interacts directly with the head node of the cluster, which is responsible for processing user input, tracking data and physics interactions. The head node also transmits updates for dynamic scene elements (e.g., cameras and objects under the control of the physics engine). For clusters with genlock/framelock capabilities, rendering synchronization between the cluster nodes is handled in hardware; otherwise our framework falls back to a software implementation of framelocking based on the head node’s message passing capabilities.

The main rendering capabilities of our framework are based on the NVIDIA SceniX library [80]. SceniX provides a robust implementation of the scene graph with a plug-in mechanism for loading and saving of scene data and related file formats. Our assets are primarily stored in the COLLADA [4] and Autodesk FBX formats, although we have developed custom plug-ins for a number of data formats used in other scene graph implementations and our own internal research. The rendering system uses OpenGL 2.1, the NVIDIA Cg and the GLSL languages for writing shaders, and supports modern GPU features such as geometry shaders and GPU-based tessellation.

Our volume rendering module is tightly integrated with the scene graph. We have implemented a GPU-based single-pass ray-casting approach [44] with iso-surface extraction, a Cook-Terrance light transport model and distance-based light attenuation. The code is implemented in the NVIDIA Cg language and the resulting shader is bound to standard primitives in the scene graph. A 3D box is sufficient to initialize the starting and ending positions for the volume integration pass so that the final images represent a valid stereoscopic pair. In addition, we intersect the box with the near clipping plane of the scene’s camera so that we have valid starting points for user positions inside the volume data. This is essential for medical applications that involve navigating inside an organ. The ray’s ending positions are also modified by the scene geometry so that we can support the rendering of environmental effects such as smoke around an airplane. As can be seen in Figure 5.4, the smoke correctly interacts with the solid object in the scene.

The standard OpenGL rendering can produce high quality images at real-time rates, however it is not particularly suitable for certain visual effects that are related to the transport of light in the scene. In particular, sharp specular reflections and refractions, accurate shadows and global illumination effects are difficult to produce. Such effects are typically associated with high quality off-line rendering and ray-tracing. A number of GPU-based ray-tracing algorithms have been proposed that can achieve interactive frame-rates even for large geometric models [70, 82, 86, 133]. Our raytracing module is based on NVIDIA OptiX [82] and it is tightly coupled into the scene graph. Since both raytracing and OpenGL rendering traverse the same underlying scene representation, using OptiX as a renderer requires only minor changes to the user interface. Implementing advanced effects such as global illumination, requires writing additional OptiX programs in the CUDA language. The performance is significantly lower than with OpenGL, however we can achieve interactive rendering for scenes with low to moderate complexity (see Figure 5.5 for examples).



**Figure 5.4:** Volume rendering used for (a) medical visualization and (b) simulation data exploration.

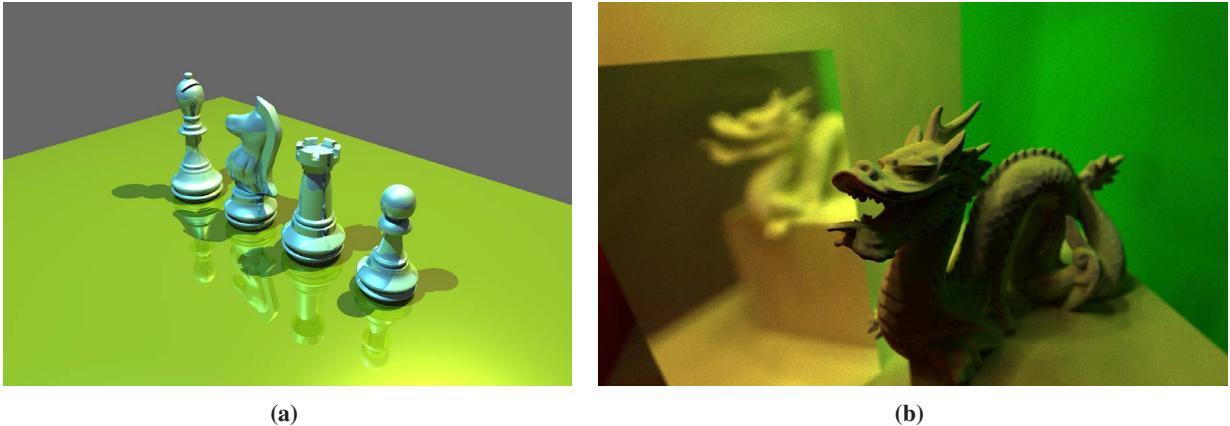
Even though the OptiX rendering can produce high quality images at interactive frame rates, the quality of the output is reduced compared to traditional Reyes rendering systems [21]. We have implemented a translation from the SceniX scene graph to a Renderman-compatible interface. This module has been tested with Pixar’s Renderman Pro Server and the open-source Pixie renderer. Our second implementation uses the now deprecated NVIDIA Gelato renderer, which is based on a shading language similar to Renderman, but uses NVIDIA Quadro-based GPUs to accelerate certain computations. The comparison between the rendered outputs is presented in Figure 5.6. In terms of deployment to the IC, off-line rendering is currently executed individually on each rendering node and for each display surface. Although this allows for immediate visual feedback as parts of the images are rendered, it is highly inefficient. A future direction would be to dedicated nodes from the Visualization Cluster for back-end rendering only and process all surfaces at the same time.

#### 5.1.3 Remote Visualization

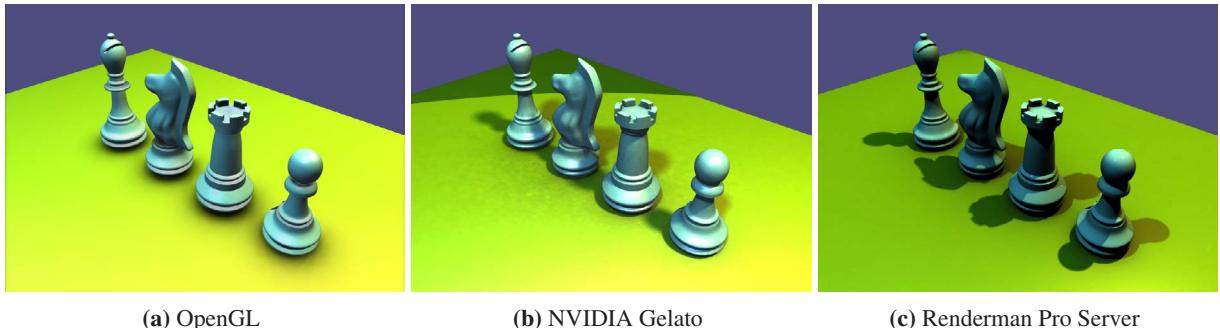
The rendering features described so far primarily deal with generating images at the local node that is also driving the display device. While this may be sufficient for many tasks that require high levels of interactivity (and therefore very low latency), we have also implemented a number of remote rendering features that can leverage larger computational resources. One such example is the use of cloud-based rendering services (e.g., NVIDIA Iray cloud rendering with RealityServer) where high-quality images are generated in the data center, streamed over a network connection and the user controls the visualization parameters on the local device. Traditionally, remote rendering services (and particularly those targeted at interactive gaming)

## 5. IMMERSIVE ENVIRONMENTS

---



**Figure 5.5:** The NVIDIA OptiX GPU raytracing is used in the Immersive Cabin for (a) complex reflections and shadows, and (b) global illumination.



**Figure 5.6:** Chess scene rendered with the different graphics pipelines.

deployed custom hardware and software to encode the images into a video stream, which increases the complexity of the system and introduces additional latency at the user interface. Recent graphics hardware have started to include on-board high quality video encoders, which eliminates a major source of latency.

We leverage video encoding and decoding in the H.264 format to provide remote visualization support in our framework, both for streaming image data into the IC and for live monitoring of the visualization results from outside the facility. The codec was chosen because of the wide compatibility with decoding devices, from GPUs to tablets and cell phones, and the availability of low-latency realtime encoding software and hardware. We implement the following encoding libraries:

**x264:** open-source library for encoding video streams in the H.264 format. It includes a special mode optimized for zero-latency encoding. One disadvantage is that, as it is CPU-based, rendered images need to be streamed from GPU memory to CPU memory. For high resolution streams, this can have a significant impact on the latency.

## 5.1 The Immersive Cabin

---

**NVCUVENC:** a CUDA-based video encoder for NVIDIA GPUs. This encoder is suitable for realtime applications as it can access the rendered images without any additional memory transfers and only the compressed bitstream needs to be transferred to the CPU. The impact on the visualization performance is minimal.

**NVENC:** a new dedicated video encoder in NVIDIA Kepler GPUs. This encoder is designed for low latency realtime applications. Furthermore, it produces significantly higher quality video streams than NVCUVENC at similar bitrates. The disadvantage is that it is only available on the latest generation of NVIDIA GPUs and its use on non-workstation GPUs (e.g., Geforce) requires a separate license.

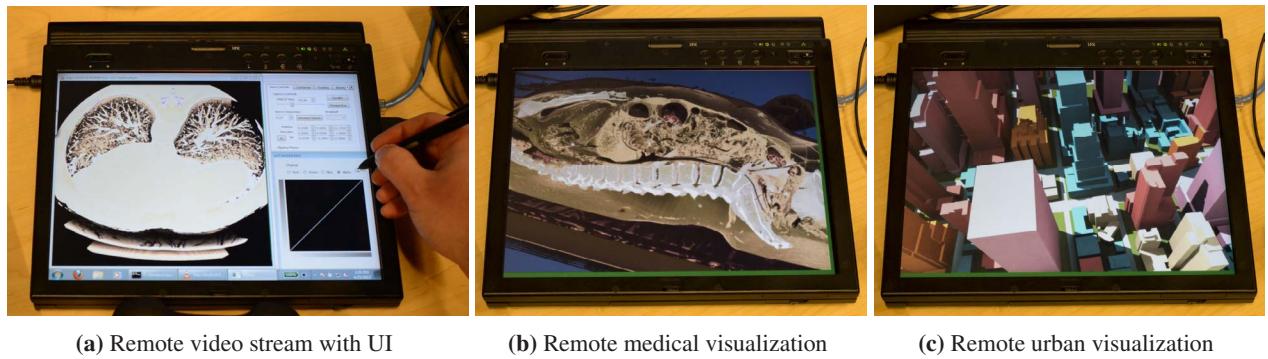
The NVENC encoder is of particular interest in our application since it supports a number of features dedicated to real-time streaming. These include support for specifying areas of interest, dynamic resolutions and invalidation of bad reference in the video stream due to missing network packets. Also, it supports encoding of data in the YV12 4:4:4 format where the color information is not downsampled, in contrast to many other GPU-based encoders which only support YV12 4:2:0 input data. The latest generation of AMD and Intel GPUs also include hardware support for H.264 encoding and in the future we plan to implement the corresponding APIs as well.

The encoding is implemented in the optional compositing stage of our visualization framework, which resolves the multi-sampled render targets. In addition to the RGB color output, we also generate a secondary output directly in the YUV color space expected by the encoder (usually in the YV12 planar layout and optionally with chroma downsampling). If compositing is disabled, the encoding requires copying the RGB data from the rendered buffer, which is less efficient. The encoded video stream is then sent to our custom streaming server, which uses the LIVE555 library to implement the Real-Time Streaming Protocol (RTSP). One advantage of this approach is that multiple clients can view the streaming session with standard video players, which opens new possibilities related to education and collaboration. We also plan to develop a custom streaming protocol that takes advantage of the new reliability features in the NVENC encoder. Specifically, two-way communication would allow clients to notify the encoder of network errors, so that future frames are encoded without references to the corrupted frames. At this point, RTSP provides resiliency to network transmission errors at the cost of increased latency.

Once the video stream reaches the clients, it can be viewed with a simple video player or directly in our software where the decoded video stream replaces the OpenGL input to the compositing stage. This allows for example a thin client, such as a tablet, to present the user with the control interface to the Immersive Cabin where the actual 3D rendering for the UI is performed on one of the cluster nodes. The decoding uses the LibVLC library and can be hardware-accelerated through the DXVA API in certain software configurations (e.g., when outputting to a dedicated window without any further processing). We also implemented video decoders using GPU-specific APIs on supported hardware:

## 5. IMMERSIVE ENVIRONMENTS

---



(a) Remote video stream with UI      (b) Remote medical visualization      (c) Remote urban visualization

**Figure 5.7:** Video streams generated by IC nodes are displayed on a tablet. (a) The UI is running natively on the tablet and controlling the IC; the video is streamed from the cluster head node. (b) and (c) The output of the forward-view node in the IC is streamed to the tablet; the UI is running on the cluster head node.

**Intel Media SDK** interfaces with the Intel GPUs available in x86 tablets, such as the Microsoft Surface. The SDK provides hardware-accelerated H.264 decoding and integrates with OpenCL to provide additional processing of the video stream on the client. One major disadvantage is that this SDK is a hybrid system with major computation load on CPU.

**NVCUVID** library for accessing the video decoding functionality of NVIDIA GPUs. Depending on the GPU capabilities, the decoding may be implemented in the CUDA language, or using power-efficient on-board hardware. The resulting images can be further processed in CUDA or combined with client-based rendering in OpenGL.

Figure 5.7 demonstrates the use of the video streaming component in our framework to integrate a low-power tablet into the IC. In Figure 5.7a, the entire UI is running on the tablet with only the rendered image being streamed from the head node. The advantage of this approach over using VNC or similar remote desktop software is that we have the flexibility of handling tablet-specific features such as touch input and pressure sensitivity. In Figure 5.7b and 5.7c, the tablet is only used to monitor the output to one of the walls in the IC. The data is sufficiently complex that it cannot be rendered natively on the device. Additional applications are possible when the tablet is tracked within the IC and we use the remote rendering to display additional layers of data over what the user sees in the environment.

### 5.1.4 Applications

The highly immersive and interactive nature of visualizations in the IC lends itself to applications that deal with large volumes of data. In particular, the planning and design of large urban environments involves visualizations at multiple scales of detail, from large scale overviews of the entire city to small scale details at the street level. The IC allows very intuitive transitions between the levels, while the immersive visualization



**Figure 5.8:** Exploration of a large urban environment.

help the user retain the context of the overall model. In addition, real-time plume dispersion models [90] can take advantage of the powerful GPU cluster of the IC and our volume rendering framework to provide interactive simulations and training for national security and natural disaster scenarios. Figure 5.4b demonstrates volumetric rendering of the dispersion simulation results together with the corresponding mesh model in our visualization framework. Figure 5.8 illustrates the exploration of a large urban environment designed for the IC. The user has an overview of the entire city together with details at the middle and small scales. This model renders at interactive frame rates on the IC’s GPU cluster.

Architectural flythroughs are a very natural application for IC and take advantage of the extended depth perception and the immersion over 2D workstation displays. We primarily utilize the OpenGL renderer to provide more fluid animations, although for certain scenes with complex geometry the OptiX renderer may provide competitive performance. Our framework provides tools such as custom clipping planes, transparency shaders, high-quality materials and environmental effects, which are particularly useful for rendering detailed building exteriors and interiors. Head tracking is fully supported, although it is generally not used when the models are examined by more than one person or when the rendering performance is below 15 frames per second. Figure 5.9 shows snapshots from the exploration of two buildings at the Stony Brook University campus - the Advanced Energy Research and Technology Center (AERTC) and the new Computer Science building. In both cases, the visualization in the Immersive Cabin was available to the architects and the engineers at least 6 months before the construction began.

Our volume rendering module is primarily designed to support medical visualization applications such as Virtual Colonoscopy [49, 58] in immersive environments. The rendering framework can achieve interactive performance in the IC for  $512 \times 512 \times 451$  16-bit CT scan of a patient’s abdomen. Since the examination in VC consists of traversing the colon along the centerline, we rely heavily on optimizations such as empty space skipping and early ray termination, while additional computational resources are dedicated to high quality lighting and shading at the iso-surface representing the internal colon wall. We have also developed

## 5. IMMERSIVE ENVIRONMENTS

---



(a) AERTC



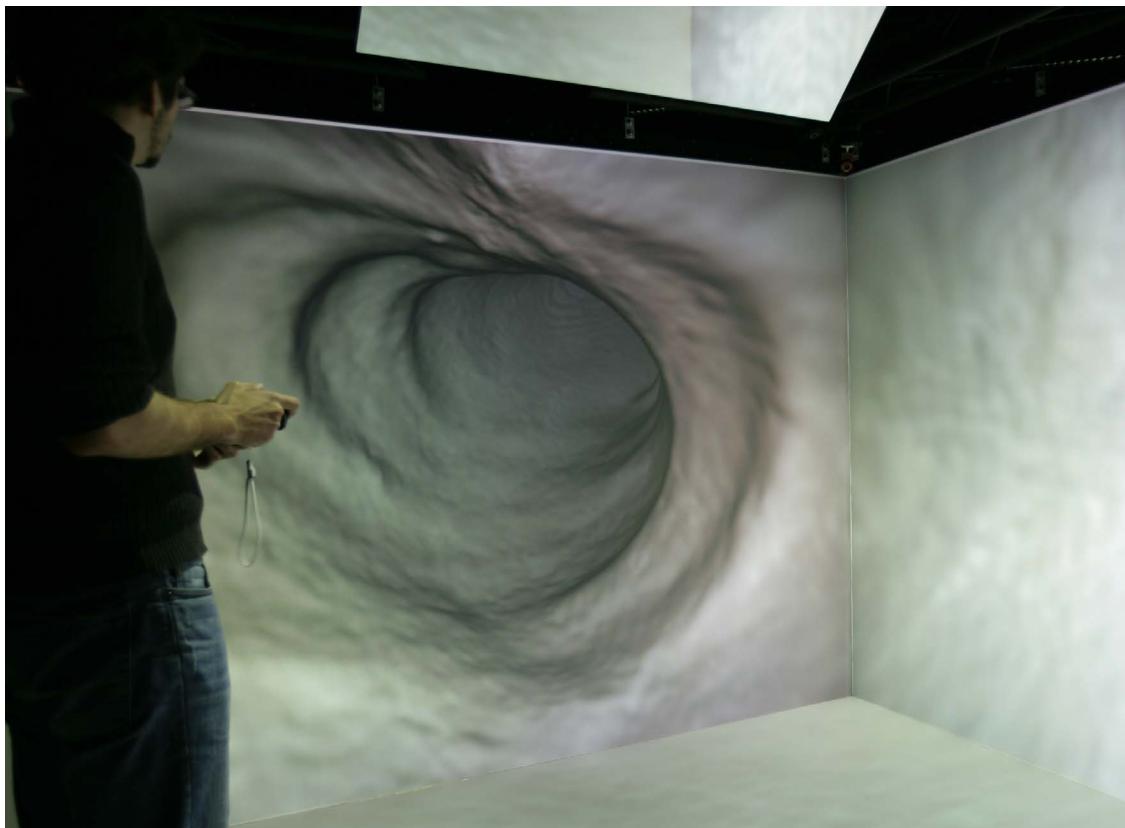
(b) New Computer Science Building

**Figure 5.9:** Architectural rendering of (a) the AERTC and (b) the new Computer Science building, both at Stony Brook University.

a conformal visualization that maps visual information from the missing ceiling in the IC to the side walls. The conformal property ensures that angles are preserved locally, and as a result shapes are also preserved locally. This is particularly advantageous for Virtual Colonoscopy since the user can examine the entire surface of the colon in a partially immersive environment such as the IC, and polyps retain their circular shape even though their size on the screen may be reduced. Conformal visualization is described in detail in Section 6.1.

## **5.1 The Immersive Cabin**

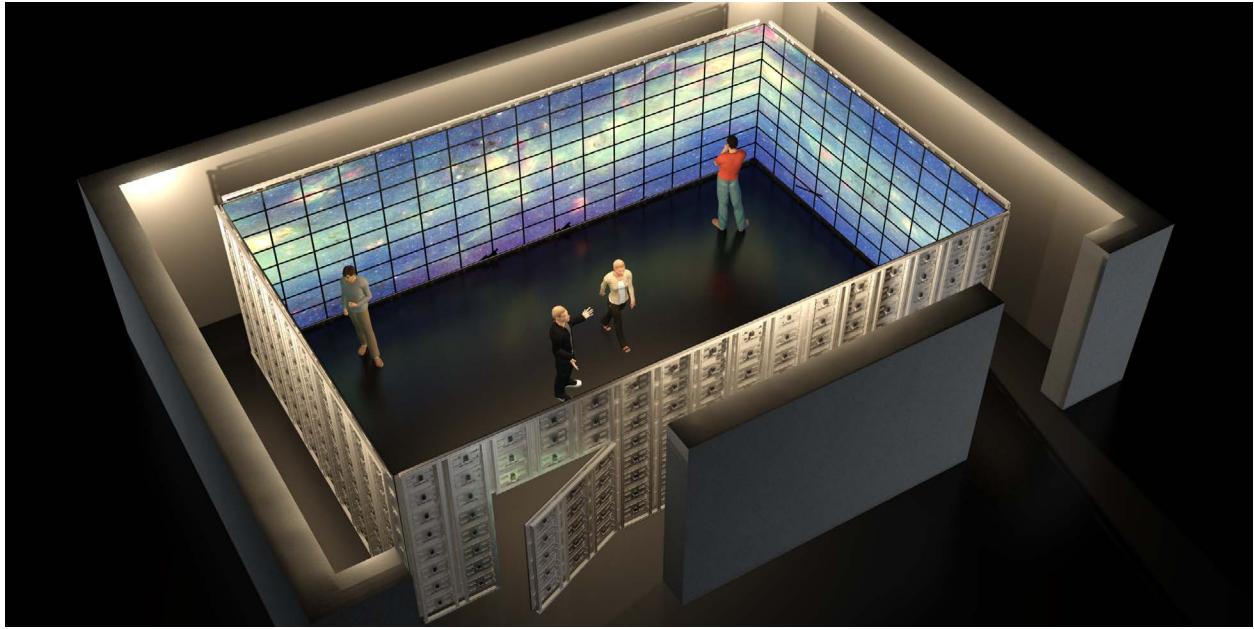
---



**Figure 5.10:** Virtual Colonoscopy in the Immersive Cabin.

## 5. IMMERSIVE ENVIRONMENTS

---



**Figure 5.11:** The Reality Deck uses 416 LCD panels to achieve resolution in excess of 1.5 billion pixels in an immersive 4-wall layout.

## 5.2 The Reality Deck

One of the main challenges in using the Immersive Cabin (see Section 5.1) for the visualization of big data is the very low pixel density. While the immersion and the depth perception can be helpful when dealing with 3D structures, it only provides a total resolution of approximately 7 megapixels. Even the highest-end CAVEs are limited to about 90 megapixels [26]. At the same time, gigapixel photography has become easily accessible due to advancements in cameras and computing power, resulting in many photographs that exceed 50 gigapixels in size. The panoramic photographs are particularly suitable for display in immersive visualization facilities, however, the low pixel density of the CAVE does not allow for fine details in the image to be resolved without panning and zooming. The size of data that needs to be visualized is growing in many other fields as well, from manufacturing and architectural design, to supercomputing CFD simulations, medical visualization and web image collections.

We have designed and built the Reality Deck as an answer to the challenge of big data visualization. The facility retains the immersive display surface layout of the 4-sided CAVE, but uses a large number of tiled high-resolution LCD monitors to provide an aggregate resolution in excess of 1.5 gigapixels. The work area is increased significantly from the  $100\text{ ft}^2$  in the CAVE to approximately  $675\text{ ft}^2$ , allowing for more collaboration between users of the facility. Compared to traditional tiled displays, our design increases the available resolution by a factor of more than 5 [27, 112], while offering a fully enclosed visualization space

with a more manageable work area (compared to the dimension requirements of a 1.5 gigapixel display with a single planar wall). The Reality Deck is driven by a compact 20-node cluster with 80 GPUs that uses an extended version of our visualization framework from the Immersive Cabin. As a result, many of the applications described in Section 5.1.4 are available in the Reality Deck without any modification, and we have developed additional ones that specifically take advantage of the increased resolution and pixel density.

### 5.2.1 Equipment and Construction

The Reality Deck is housed in a large  $40' \times 30'$  lab where the 416 LCD monitors are mounted in a 4-wall arrangement. The total work space enclosed by the monitors is  $32.8' \times 20.5'$ . Figure 5.1b and Figure 5.11 illustrate the general layout of the facility and all 3D models are rendered at the correct scale based on the engineering schematics. The large front and back walls are composed of a  $16 \times 8$  grid of monitors and the side walls are a  $10 \times 8$  grid.

### Display Design

Similarly to other large tiled displays, the visualization aspects of our Reality Deck are provided by LCD monitors and the choice of those monitors was in fact the most critical design decision. The goal of the Reality Deck was to be the first gigapixel display in the world and we formulated the following set of requirement for the monitors, ordered by importance:

**Resolution target:** Based on the available space, the monitors need to provide  $2560 \times 1600$  resolution for a 30" display,  $1920 \times 1080$  for 23" or  $5120 \times 3200$  for 60".

**Bezel size:** Sub 5-mm bezel is ideal, however the size should not exceed be 8mm for a 23" display and 15mm for a 30" display.

**Display size:** Larger monitors (e.g., 30" and above) are preferable as long as they can deliver the required pixel density.

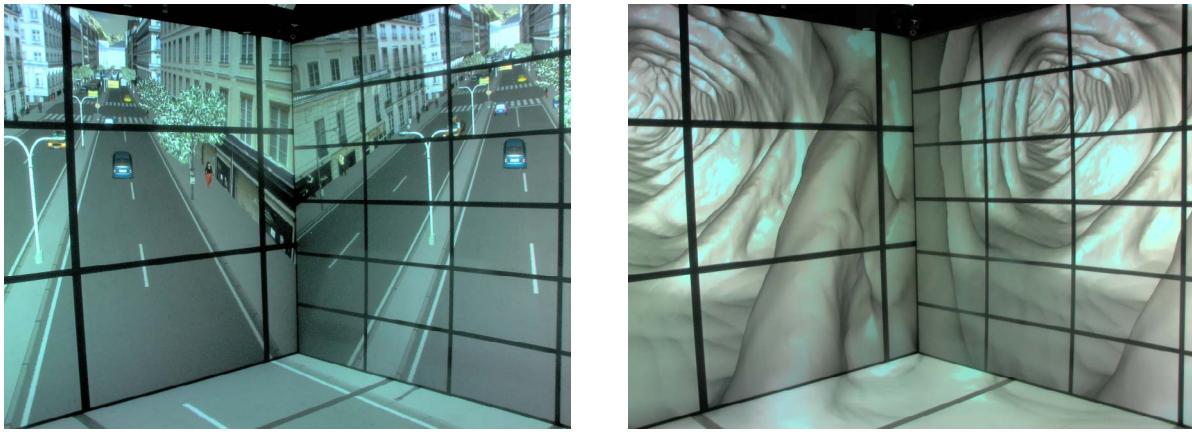
**Image quality:** The monitors should use high quality, professional-grade panels with good contrast, backlight uniformity and viewing angles.

**Stereo support:** Stereo is a very desirable feature, but not at the cost of image quality or significantly reduced pixel density.

Based on these characteristics, we evaluated a number of different displays that covered panel sizes from 23" to 60", the IPS, PVA and PLS panel technologies and various bezel sizes. We simulated the tiled displays in the Immersive Cabin for both 2D and stereoscopic rendering (see Figure 5.12) in order to analyze the perceptual effects of the bezels on different visualization tasks. The results of an informal user study were then used as a contributing factor in the display selection process. While no mass-produced display on the market as of 2013 satisfies all five requirements, the Samsung S27A850D provides a good balance.

## 5. IMMERSIVE ENVIRONMENTS

---

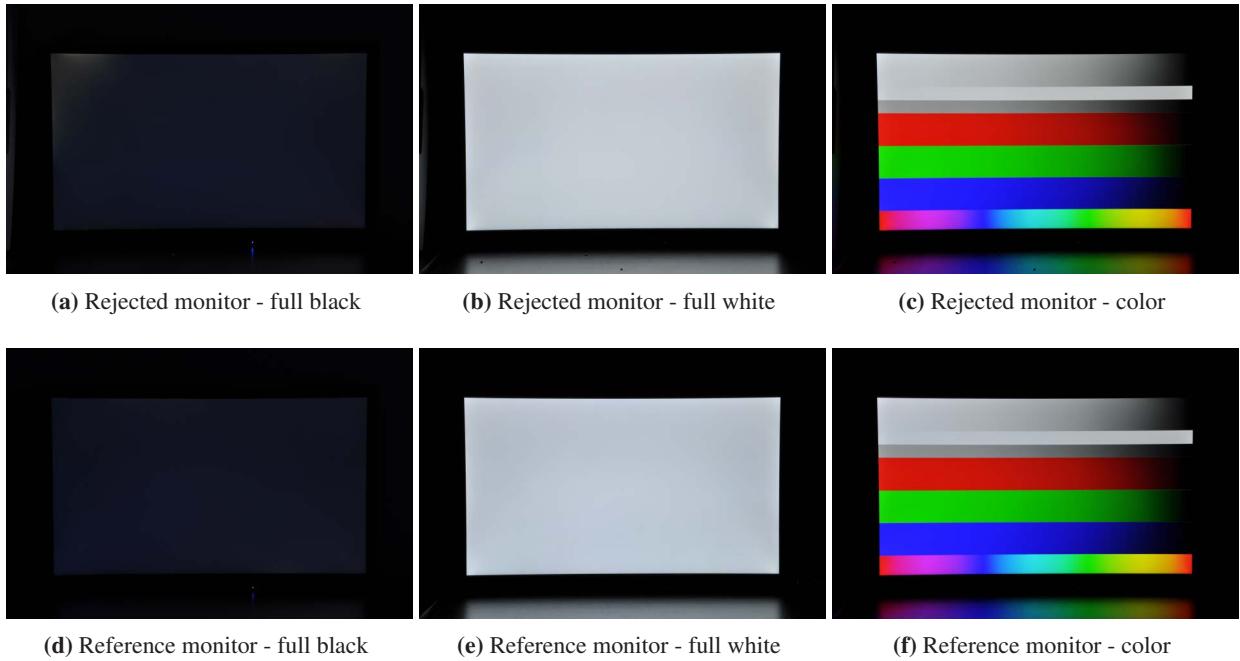


**Figure 5.12:** We evaluated the perceptual effects of the screen size and the bezel size for different LCD monitors in the Immersive Cabin using various datasets. The study included both 2D and stereoscopic rendering. Left wall in both images simulates a large-format Planar monitor (60" diameter, 3840 × 2160); right wall is the Samsung S27A850D with modified bezels (27" diameter, 2560 × 1440).

It is a professional 27" PLS panel with 2560x1440 resolution and excellent viewing angles, contrast and color saturation (comparable to many P-IPS and S-IPS panels, but not the highest-end 10bit IPS panels). In comparison to other high-end monitors with CCFL backlights, the S27A850D uses a white LED backlight, which significantly reduces the weight and the power requirements (46W for the S27A850D compared to 134W for the Dell U2711). The wide-gamut monitors with RGB LEDs that we evaluated did not bring sufficient benefits to our target applications to justify the significant increase in cost. Finally, while the original plastic bezel is relatively large, the lightweight monitor can be easily modified with a custom mounting solution that reduces the bezel from 20mm (30mm on the bottom edge) to approximately 14mm on all sides.

Each monitor was subjected to an image quality test and rated on a scale from 1.0 to 3.0, where 1.0 corresponds to a display with significant uniformity or color issues and 3.0 is a display with no visible problems. We also identified a small number of reference displays that were labeled as 3.0+. We were primarily interested in image uniformity when displaying a full white and a full black signal, as well as identifying any issues with the color reproduction. Three photographs were taken of each monitor from a fixed camera position and with a standard set of camera and monitor settings. Figure 5.13 shows the results for a rejected monitor rated at 1.0 and a reference 3.0+ monitor. Based on the stacks of images and the ratings, a total of 98 from the first batch of 441 monitors were replaced. After testing the second batch, we selected the best 416 displays (rated at 2.0 and above), as well as a set of spare units, for modification and use in the Reality Deck.

The use of lightweight monitors allowed us to design custom mounting brackets (see Figure 5.14a) and a



**Figure 5.13:** Each monitor is tested for image uniformity when displaying a full white and a full black signal. We also look for any significant color issues. (a) to (c) are the results for a rejected monitor (significant uniformity issues when showing a black image). (d) to (f) correspond to a reference monitor without any image quality issues.

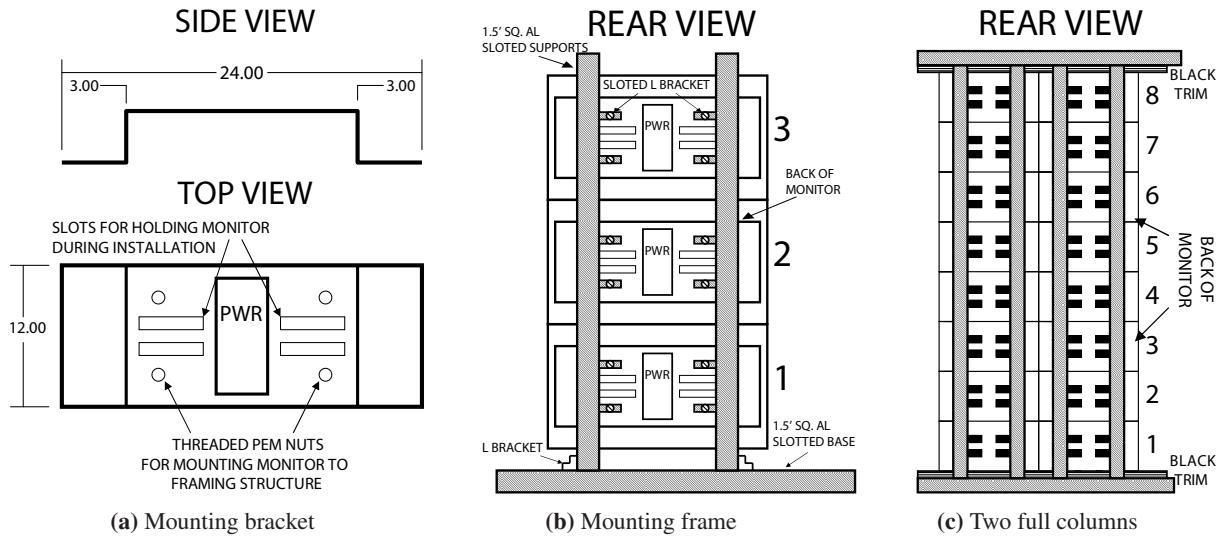
simple aluminum frame (see Figure 5.14b and Figure 5.14c) so that individual monitors can be aligned with sub-millimeter accuracy and can also be replaced by a single person. The plastic cover of the S27A850D houses the circuit board, the user controls and the power supply, which are moved to the brackets during the modification. The door to the facility is a section of the frame that is mounted on a hinge and holds a  $3 \times 5$  grid of monitors. It is power operated but can also be opened manually in case of an emergency. When the door is closed, it is completely flush with the rest of the wall and it is visually indistinguishable from the other displays, as shown in Figure 5.15.

### GPU Cluster

We designed a number of GPU clusters with varying specifications in regards to the number of monitors that can be connected to each GPU and the number of GPUs in each node. In terms of GPU performance, the most critical aspect is having sufficient GPU memory for the large framebuffers when driving six 2560x1440 displays out of each single GPU, where the OpenGL buffers and a couple of multisampled fullscreen render targets can easily require 2GB or more of memory, before the application-specific data is even loaded. Therefore, the minimum memory size we considered is 4GB per GPU. While at the time 6GB GPUs were

## 5. IMMERSIVE ENVIRONMENTS

---



**Figure 5.14:** Schematic drawings of (a) the mounting bracket attached to each monitor, (b) the mounting frame and (c) two complete columns of mounted monitors.

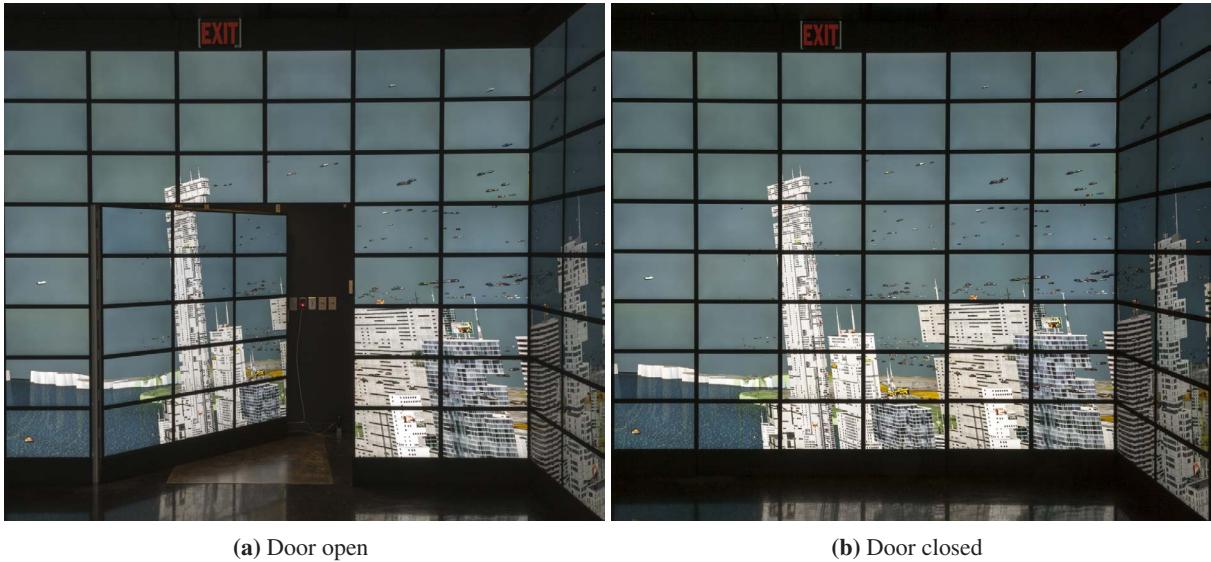
not practical (the only such GPU, the NVIDIA Quadro 6000 could only drive 2 displays), the larger memory sizes of newer GPUs will allow for the visualization of more complex datasets in future Reality Deck installations.

During our evaluation we determined that the most cost-effective solution would provide sufficient performance for many of the target applications and therefore our cluster was designed around the AMD FirePro V9800 GPU with Eyefinity technology:

- 20 node GPU cluster using Exxact Spectrum FXR410-192R
- Dual Intel Xeon E5645 (6-core, 12-thread), 48GB RAM per node
- Quad AMD FirePro V9800 (6 DisplayPort outputs, 4GB RAM) and AMD FirePro S400
- FDR Infiniband (54Gbit/s)
- 240 CPU cores total - 2.3 TFLOPs peak performance, 1.2 TB distributed memory
- 80 GPUs total - 220 TFLOPs peak performance, 320 GB distributed memory

The GPU cluster is housed in a machine room adjacent to the Reality Deck lab and we use 50-100ft fiber optic DisplayPort cables for the connections to the monitors. The AMD Eyefinity technology is used to abstract  $3 \times 2$  groups of displays as a single bezel-compensated frame buffer. In order to avoid having a single GPU controlling monitors on adjacent walls, two of the corners are configured as  $1 \times 4$  groups.

We evaluated the low-level rendering performance with the SPECviewperf 11 benchmark, which measures the OpenGL rendering performance on traces from a variety of 3D workstation applications. The ob-



**Figure 5.15:** Our facility contains an automatic door composed of a  $3 \times 5$  grid of monitors. There is a small handle used to open the door during power outages. Except for the handle and the exit sign, the door is visually indistinguishable from the rest of the display.

served performance drop when moving from a single WQXGA display to the full 6-display Eyefinity group in the Reality Deck was between 3% and 20%, depending on the application. Moving to 8x multisampling resulted in an additional 24% to 40% performance loss, while the visualization remained interactive even at the increased image quality and resolution. For shader-bound workloads (e.g., volume rendering, 3D scenes with complex effects, etc.), the performance scales linearly as we move from 1 to 6 displays per GPU. Given the high resolution per GPU, some of the traditional rendering algorithms (e.g., volume rendering and GPU raytracing) need to be redesigned for interactive performance in the Reality Deck with the current generation of GPUs. Others such as virtual/gigapixel texturing can still run at 60Hz provided that there is sufficient memory, memory transfers are managed asynchronously and no fullscreen render targets or compositing at the OS level are used. Figure 5.16a shows an example of gigapixel image exploration in the Reality Deck. The details for our implementation of virtual texturing in the Reality Deck are presented in Section 5.2.2. We expect that newer generations of GPUs will lift some of these performance restrictions in the future.

We have developed software for configuring the Reality Deck monitors (e.g., enabling Eyefinity, bezel compensation and hardware synchronization, changing display brightness and color settings, etc.) based on the AMD Display Library (ADL) SDK. The computers in the cluster are controlled remotely via IPMI. The cluster controller is a separate high performance machine which is used to operate the visualization software and the additional components of the Reality Deck, such as the tracking and sound systems. Remote access to the controller with full support for OpenGL and USB peripherals is available through the HP Remote

## 5. IMMERSIVE ENVIRONMENTS

---



(a) Exploration of a gigapixel aerial photograph of New York City.

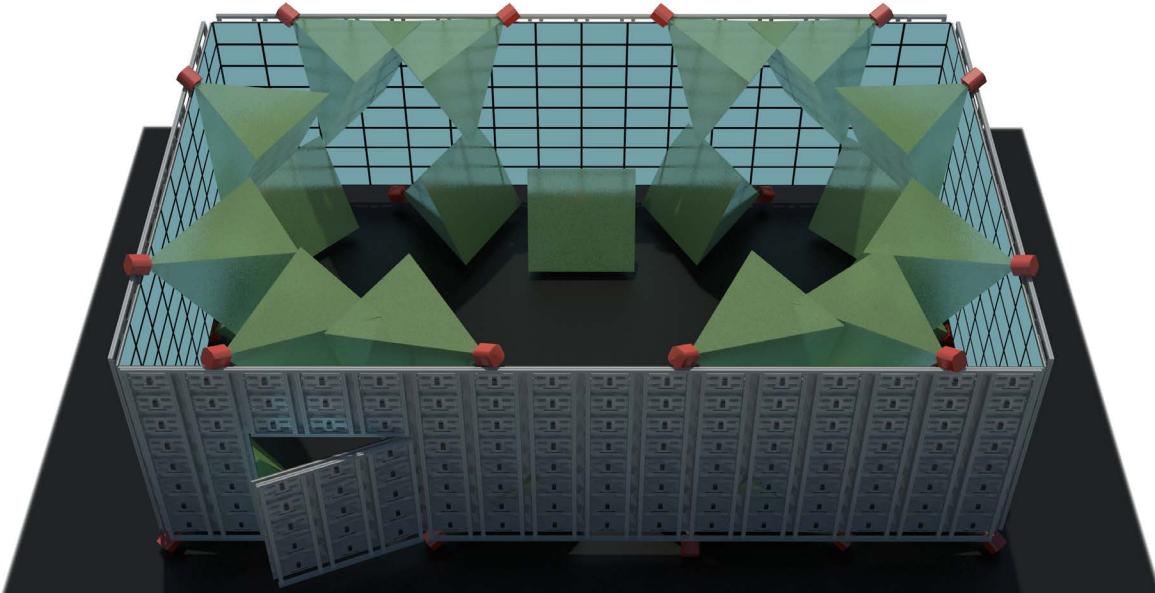
(b) The control interface for the Reality Deck.

**Figure 5.16:** Photographs of the Reality Deck running GIS applications. (a) A gigapixel aerial photograph for New York City with 0.5m resolution is mapped on 5m elevation data and we provide an interface for 3D flight over the terrain. The entire dataset is shown in Figure 5.20. (b) High-resolution global elevation data is displayed with a relief shader, including a simple interactive threshold-based flooding simulation. The control interface is running on the PixelSense touch table.

Graphics Software on laptops, a wireless tablet and a Microsoft PixelSense touch-table that is installed in the middle of the Reality Deck (see Figure 5.16b). In the future, we plan to support the native touch and image-sensing interfaces of the PixelSense table directly in our visualization software.

### Tracking System

The Reality Deck provides unique opportunities for developing user interfaces for interacting with high resolution displays and large data. One promising area of research is the gesture-based interaction where the user's body becomes the exploration tool. The size of the facility and the requirements for precise tracking present a number of unique challenges. We have evaluated a number of tracking methods and technologies, and ultimately decided on a cost-effective 24-camera optical tracking system from OptiTrack. The system is based on the S250e camera, which uses a  $832 \times 832$  sensor running at up to 250fps and has a 56°FOV. The cameras are connected over gigabit LAN and the tracking setup can be expanded with up to 72 additional cameras. We have evaluated a number of layouts for the cameras that provide different tracking volumes and resilience to optical occlusions. Figure 5.17 illustrates a layout that extends the tracking volume up to the surface of the displays at the cost of increased occlusions when small passive markers are used.



**Figure 5.17:** The Reality Deck utilizes 24 infrared cameras for optical tracking. The positions and the orientations of the cameras are designed to maximize the effective tracking volume, especially near the displays.

### Sound System

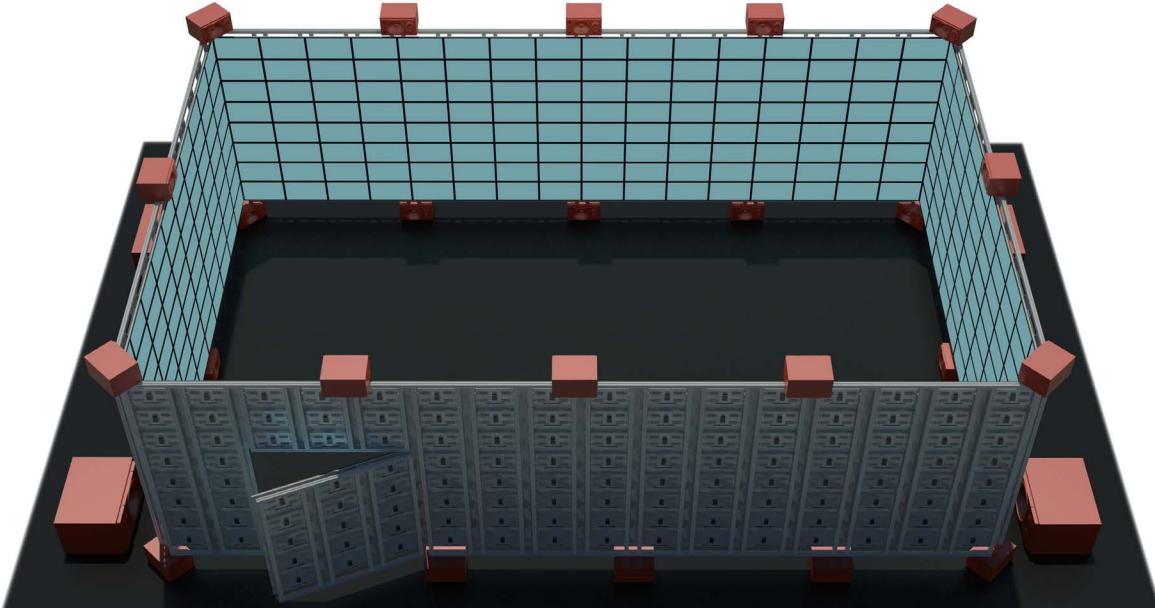
We have designed and deployed a high quality 24 channel surround sound system. In comparison to the distributed systems used in many virtual reality installations, we opted for a simpler configuration based around a studio-grade audio interface with the following components:

- 1× *MOTU PCIe-424 card* - computer interface, DSP
- 1× *MOTU 2408mk3* - 8 analog, 16 digital IO channels
- 1× *MOTU 24i/o* - 24 analog IO channels
- 3× *Alesis DEQ830* - 8-channel equalizers
- 24× *Genelec 6010A* - 24W speakers
- 4× *JBL LSR4312SP* - 450W subwoofers
- 12× *ART CLEANBoxPro* - balanced to unbalanced converters

The base PCIe interface contains a mixing DSP and can be expanded with up to 4 interfaces to provide at most 96 independent inputs and 96 outputs. We use the Motu 24i/o to drive the 24 speakers (Genelec 6010A). The outputs are also looped back into the interface so that we can create the mixes for the subwoofers. The MOTU 2408mk3 provides 8 additional channels, 4 of which are used for the subwoofers, and a number of digital IO channels. The analog outputs for the speakers are fed through three Alesis DEQ830 equalizers,

## 5. IMMERSIVE ENVIRONMENTS

---



**Figure 5.18:** The Reality Deck sound system uses a 24.4 surround speaker layout with 12 speakers below the displays, 12 speakers above and 4 large subwoofers at the corners.

which allow us to compensate for the auditory characteristics of the Reality Deck lab, and also to filter out the low frequency component of the signal. One limitation of the Genelec 6010A is that unlike the larger speakers in the series, they only accept unbalanced analog inputs. Therefore, we have installed 12 ART CLEANBoxPro converters near the speakers that take the balanced output from the equalizers and produce the required unbalanced signal. With this setup, we keep the balanced signal over the long cable runs (50-100ft), which minimizes the signal interference. The layout of the speakers is illustrated in Figure 5.18.

The main software API for accessing the audio interface is the Steinberg ASIO driver, which is supported by the majority of digital audio workstation (DAW) software and some middleware (e.g., FMOD). However, the existing libraries often offer very limited support for positional audio and may not even work with a large number of output channels. Our visualization framework uses the FMOD library to provide 7.1 surround sound in the Immersive Cabin, but we have also added support for OpenAL, which is an API for working with positional sound that is conceptually similar to OpenGL. We chose the Rapture3D driver, which implements OpenAL on top of the ASIO drivers and supports ambisonic rendering with a larger number of channels at realtime speeds.

### 5.2.2 Rendering Framework

The visualization software for the Reality Deck is based on our previous framework for the Immersive Cabin (see Section 5.1.2) and supports many of the same features. Primarily, we have extended the SceniX inte-

gration with support for tiled planar viewports, optimized the synchronization primitives to work with the significantly larger GPU cluster and its Infiniband network, and added support for anaglyph stereoscopic rendering, among other optimizations. Conversely, these enhancements are now also available in the Immersive Cabin and any other display device we support, including workstations and HMDs.

At the native resolution of our Reality Deck ( $133,120 \times 11,520$ ), the visualization of high-resolution data presents a number of unique challenges. Each GPU in the cluster renders a large  $7680 \times 2880$  viewport and the traditional texture mapping technique cannot provide sufficient amounts of data to take advantage of the high resolution. Since graphics hardware currently does not support texture memory virtualization, we have implemented out-of-core texturing that is similar to sparse virtual texturing [5] and clipmaps [3]. In our system, texture data is stored in tiles of a fixed size, which can be loaded from external memory or streamed over the network. We manage a large tile cache and an index in GPU memory, which are used to provide texture indirection. Then, we use a pre-rendering visibility pass to determine the page IDs of all tiles that are currently active in the GPU’s viewport. The tile data is then streamed asynchronously to the tile cache on the GPU and the index texture is also updated. Each processed tile is also kept in a large cache in system memory. Tiles streamed from web services are further stored in a local persistent cache on the hard drive of each rendering node. Our virtual texturing system is fully integrated with the SceniX scene graph and our material system, and can be applied to arbitrary meshes in the scene. Our virtual texturing system supports a variety of data formats:

- 8-bit texture data with optional runtime compression
- 8-bit pre-compressed texture data
- 16-bit/32-bit floating point uncompressed texture data
- 16-bit integer/32-bit floating point elevation data

The 8-bit formats are suitable for general visualization tasks, such as exploring gigapixel photographs and aerial maps. In most cases, the texture data is decompressed from JPEG tiles and then pre-processed to create the correct boundary pixels for accurate texture filtering near the edges of the tiles. We can optionally compress the final tiles to the DXT1 format before uploading to the GPU cache. The compression currently executes asynchronously in multiple threads on the CPU, although in the future we plan to implement a GPU-based compressor. The floating point formats are particularly important for the visualization of simulation and medical data, and are combined with HDR rendering techniques at the compositing stage in our framework. We also provide a simple slider for selecting discrete segments of the dynamic range.

The performance of our tile streaming is evaluated with a single virtual texture composed of 60,384 tiles with  $520 \times 520$  resolution each that have been pre-compressed in the DXT1 format. Our system can stream at approx. 2700 tiles/sec peak rate (equivalent to 730 MPixel/sec) to each GPU in the quad-GPU rendering

## 5. IMMERSIVE ENVIRONMENTS

---

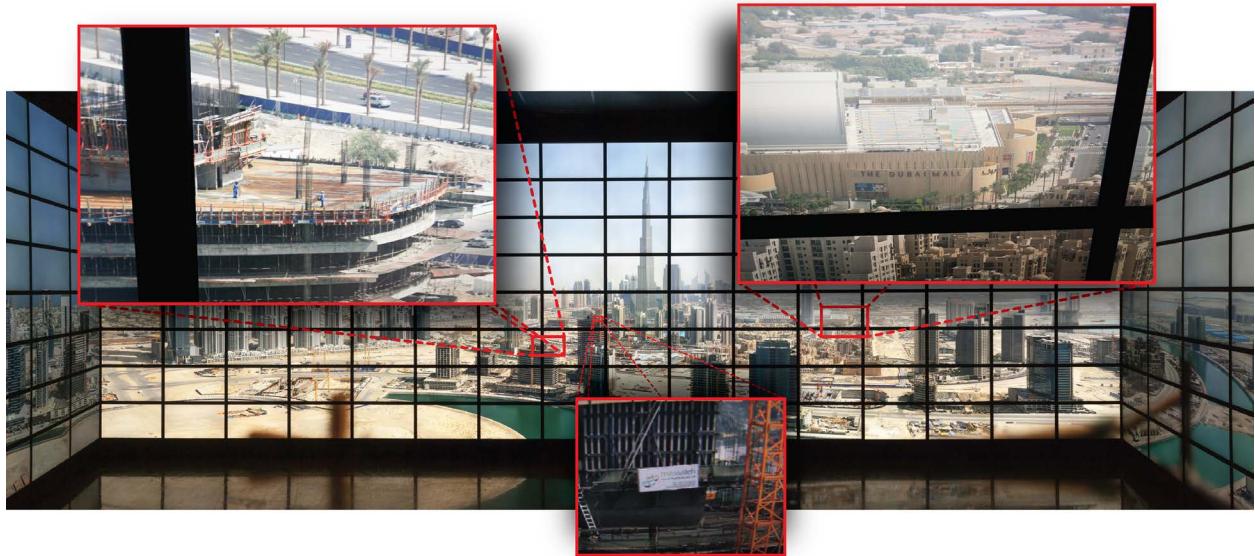
nodes for tiles that are already loaded in system memory. For visualization at 60Hz, we limit the number of tiles uploaded per frame to 32, although the number can be an order of magnitude lower for uncached data residing on the local HDD or if runtime compression is used.

A major bottleneck of the straightforward implementation is the copying of the visibility data back to the CPU for computing the page IDs. We have implemented a fully asynchronous rendering pipeline where the uploading and downloading to GPU memory is performed in separate OpenGL contexts and we use synchronization barriers to notify the analysis thread when the visibility data has been downloaded. One advantage of this approach is that it does not require any GPGPU capabilities and can run on OpenGL 2.1 hardware. However, the analysis is time consuming and may require many rendering passes to complete, thereby delaying the process of refreshing the texture cache. We have also implemented the analysis in OpenCL, where the page IDs are computed directly on the GPU, the resulting array is compacted, and only a list of the unique IDs needs to be transmitted back to the CPU. The implementation is similar to the technique by Hollemeersch et al. [48] and the main advantage is that the analysis can run for multiple virtual textures at each frame with only a minimal performance impact.

We have also implemented support for high resolution video (up to a single 8K stream or two 4K streams per GPUs) that is fully integrated with the scene graph. The video stream is decoded at runtime with LibVLC and converted to a planar YV12 4:2:0 format, which stores the luminance at full resolution and the chromaticity is downsampled by 2 in each dimension, resulting in a 12 bits/pixel storage requirement. The decompression to RGB is done on the GPU with a pixel shader and we use a tricubic B-spline filter for upsampling. In the scene graph, the video node is an extension of the texture class with background threads for decoding and uploading to the GPU, and can be applied to arbitrary mesh geometries through the material system. Also, depending on the video resolution, the encoding complexity and the available decoding resources, multiple video streams can be active at the same time and our visualization framework performs the necessary synchronization to display the video across the entire display. Alternatively, each GPU can display independent video streams.

### 5.2.3 Applications

The immersive layout of the Reality Deck is particularly suitable for the interactive exploration of panoramic gigapixel photographs. In our system, 2D data is rendered under perspective projection onto a 3D canvas that matches the topology of the original photograph. The virtual canvas is then moved away from the clipping planes defined by the surfaces of the display, so that a head-tracked user can naturally look around the bezels of the displays during movement within the large working space [2]. Figure 5.19 shows an example of using our system for the visualization of a 45 gigapixel panoramic photograph of Dubai. The close-up images



**Figure 5.19:** Photograph of the front and parts of the side walls in our immersive display showing a gigapixel panoramic image of Dubai at full resolution. Both small-scale and large-scale features are visible without panning or zooming the data. The close-up images are obtain with macro photography simply by moving the camera near the screen.

demonstrate the amount of detail that is available for a stationary image without losing the overall context of the visualization.

When working with gigapixel photographs, the tiled data is generally stored with JPEG compression, which minimizes the storage requirements. However, the runtime compression to the DXT format introduces a delay while the tiles are compressed in multiple background threads. While this latency may be acceptable for general exploration, it significantly affects the interaction quality during virtual flythroughs, for example over high resolution terrain data. One such case is illustrated in Figure 5.20 where we have created a 3D terrain mesh from multiple elevation sources (10m dataset obtained from the USGS and 3m data sampled from high resolution contour maps for the areas near New York City). A high resolution aerial photograph (0.5m resolution) is then mapped onto the terrain mesh. The tiles for the aerial map were pre-compressed and the throughput of the upload to the GPU memory is then limited by the speed of the local hard drive or the network and not by the compression performance.

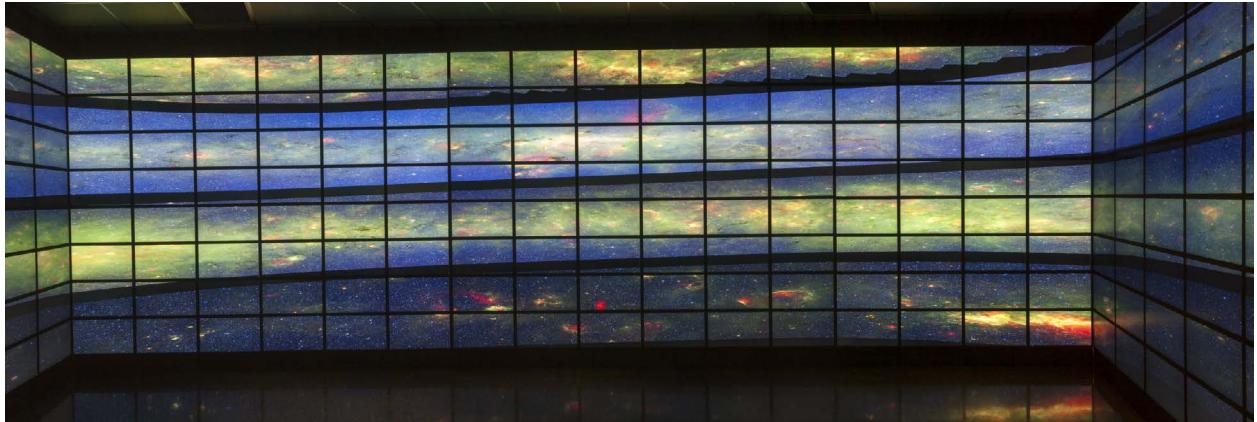
We also demonstrate the visualization capabilities of the Reality Deck with the 6 gigapixel infrared portrait of the inner Milky Way galaxy from the NASA's Spitzer Space Telescope (see Figure 5.21), which combines observations of the Galactic Legacy Infrared Mid-Plane Survey Extraordinaire (GLIMPSE) and MIPSGAL projects [8, 19]. A different view of the Milky Way is provided by the VISTA survey telescope at ESO's Paranal Observatory (see Figure 5.22). In both cases, we use the original 8-bit tile data without

## 5. IMMERSIVE ENVIRONMENTS

---



**Figure 5.20:** Gigapixel aerial photograph of New York, obtained from ArcGIS's World Imagery map service, is combined with the elevation data from USGS. We use pre-compressed tile data to provide very fast texture uploads during interactive exploration in the Reality Deck (examples of using this data in the Reality Deck are shown in Figure 5.16a and Figure 5.28).



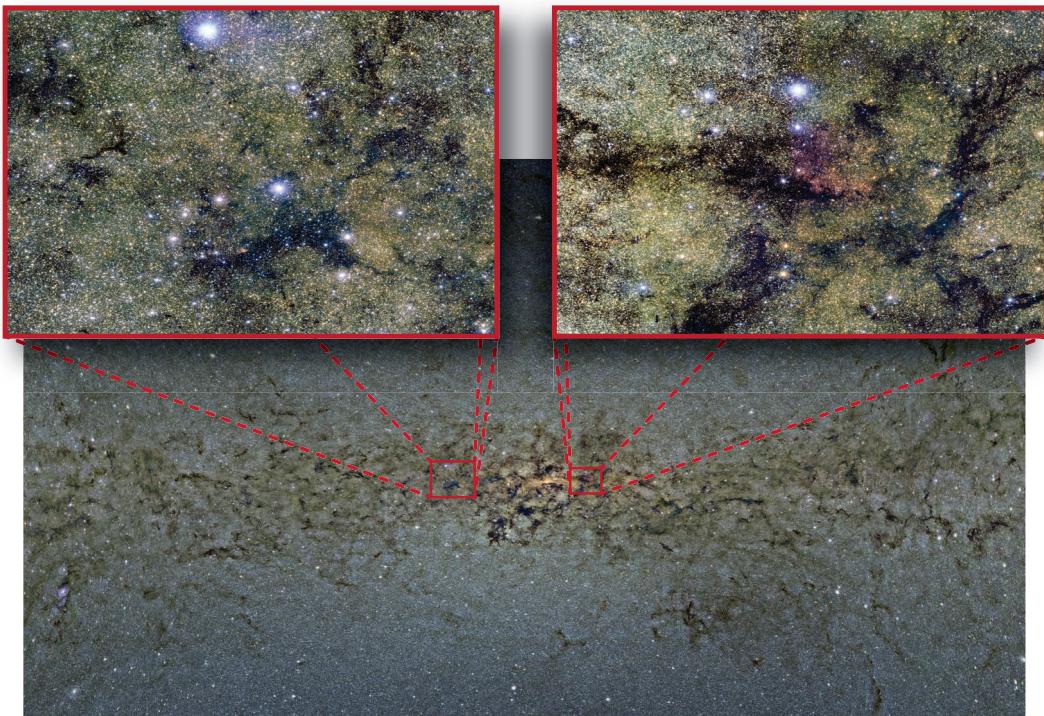
**Figure 5.21:** Photograph of the front and parts of the side walls in the Reality Deck showing a 6 gigapixel view of the Milky Way from NASA's Spitzer Space Telescope that has been wrapped around the display.

any additional processing and without the runtime compression, both of which can degrade subtle details in the image.

The architectural visualization applications from the Immersive Cabin (see Section 5.1.4) are also available in the Reality Deck. Although the stereoscopic rendering, the floor projection and the uninterrupted images are lost, the significantly higher resolution and pixel density are especially beneficial for very complex models. Also, in cases where the color perception is not particularly important, the optimized anaglyph rendering can still provide passive stereoscopic rendering. Figure 5.23 presents a snapshot from the interactive exploration of a complex scene composed of 40M polygons. The rendering is performed at the full 1.5 billion pixel resolution of the Reality Deck. The mesh rendering is also used for high resolution molecular visualization. More advanced examples of rendering techniques and applications in the Reality Deck are described in Chapter 6.

## 5. IMMERSIVE ENVIRONMENTS

---



**Figure 5.22:** 9 gigapixel view of the Milky Way from the VISTA survey telescope at the ESO Paranal Observatory. The image is captured directly from our visualization software and the close-up images show the highest resolution tile data.



**Figure 5.23:** Our software can render a scene with 40M triangles with interactive framerates at the full 1.5 billion pixel resolution of our immersive display.

## 5.3 Case Studies in Immersive Visualization

### 5.3.1 Case Study: World-wide data visualization

The visualization of GIS data is a natural application for the Reality Deck as it contains visual information at multiple scales. The user can take advantage of the high pixel density and observe smaller-scale details by simply walking closer to the display and larger-scale features by taking a few steps back. With the door closed, the facility also encloses its users in the visual information, providing opportunities for novel visual interfaces. We support the following visualization modes:

- **Planar View:** Even though the Reality Deck is enclosed, we treat it as a large planar display with 104:9 aspect ratio or four independent planar displays with 32:9 and 20:9 aspect ratios. This configuration can be used for 2D interfaces and dense 2D data, and less effectively for exploration of 3D data.
- **3D View:** We use perspective projections to explore 3D scenes, similar to our Immersive Cabin. While not as effective for visualization of 3D structures as when using stereoscopic projectors without bezels, the motion parallax and very high pixel density are particularly suitable for global high resolution GIS, among other applications.
- **Redirected View:** The display is treated as a planar visualization surface, but we use the optical tracking system during user interaction. The idea is inspired by the family of redirection techniques in the field of Virtual Reality. Our specific technique is described in Section 6.3 where we use the tracking data to simulate the exploration of an infinitely long visualization surface.

Our software uses a variety of ways to display GIS mapping data by manipulating the underlying 3D mesh. For example, we can use a rectangle as the mesh with the 'Planar View' or a cylindrical mesh under perspective projection and head tracking in the '3D View'. Both cases take advantage of the full Reality Deck and are effective for analyzing regional data. Global data is more challenging since the aspect ratio of our display is larger than the ones used in the traditional cartographic projections. While we can use panning and zooming in the 'Planar View', we also use a two-sided sphere mesh in the '3D View'. This provides a familiar 3D planetary visualization with support for flythroughs. When the virtual camera is inside the sphere, we take advantage of the enclosed display surface to visualize an entire horizontal strip of the planet and we use the traditional interaction devices to move along the surface. This is the visualization method used to generate the images in Figure 5.24.

Our system supports a number of sources for GIS data. In addition to standalone mapping areas, we have assembled a high resolution global elevation map that combines the 90m global coverage with 30m coverage for Europe and the US from NASA's Shuttle Radar Topography Mission (SRTM). Results are presented in

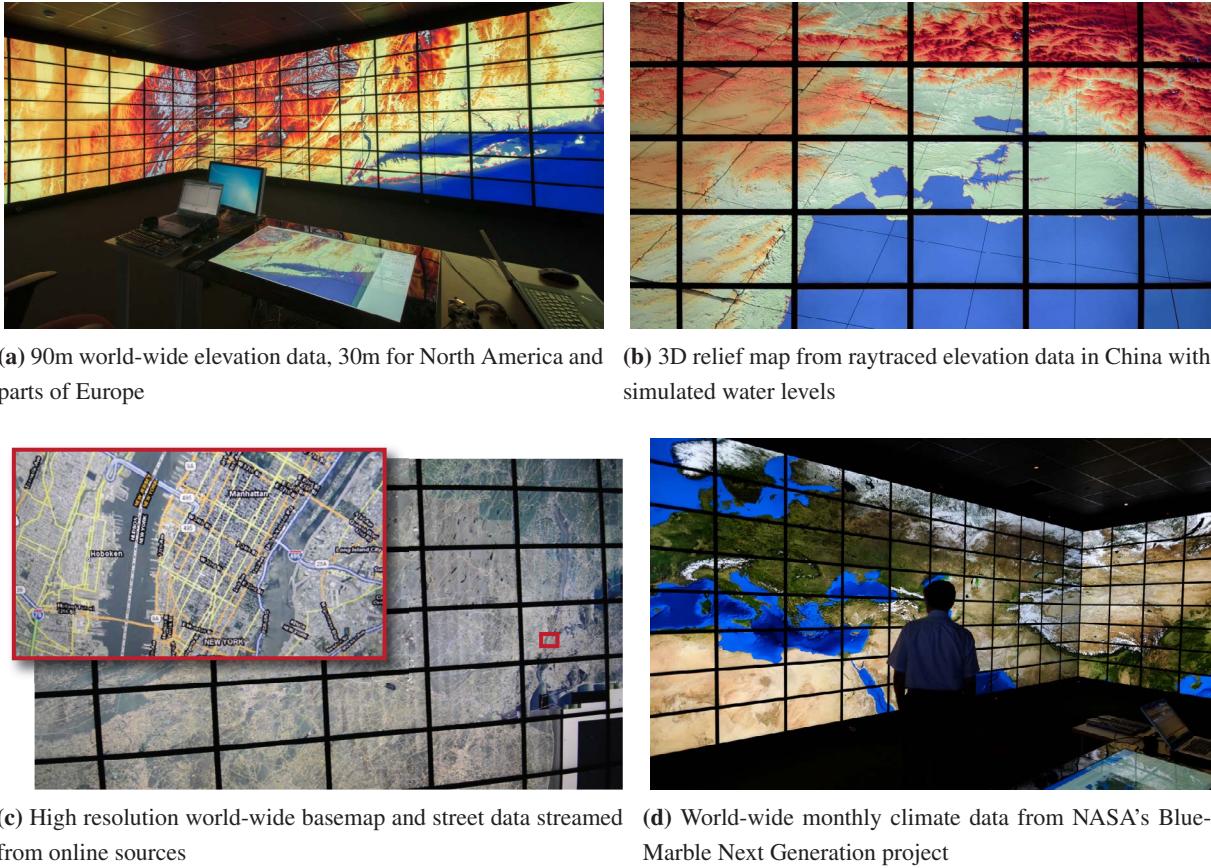
## 5. IMMERSIVE ENVIRONMENTS

---

Figure 5.24a and Figure 5.24b. This is implemented in our software as multiple separate virtual textures that are combined using logic operators during rendering. Certain areas of interest are further enhanced with the 10m elevation data from USGS when such data is available. Other sources can be used as well, such as high resolution contour maps and ocean bathymetry, which can be resampled and combined with the 30m elevation or loaded separately. Such integration is presented in Section 5.3.2 where we use high resolution contour maps for the New York area together with bathymetry for the Long Island Sound. Unlike the terrain in Figure 5.20 which uses an adaptively tessellated mesh, we visualize the global elevation data directly from the tiled data using a GPU shader (see Figure 5.24a). Normals for the lighting are computed directly from the elevation data and we optionally use shader-based heightfield raytracing for a more complex 3D relief effect (see Figure 5.24b). We have expanded the system presented in Section 5.2.2 to use a texture array for the tiles on the GPU, which allows for more high resolution tiles to be kept in GPU memory compared to using a single  $16k \times 16k$  2D texture. While this reduces the paging in all of our virtual texturing applications for GPUs with large memories, it is particularly important for computational algorithms using the data, such as the heightfield raytracing.

In addition to the local pre-processed data, we have implemented tile sources that interface with web services, such as those provided by ESRI ArcGIS and OpenStreetMaps (see Figure 5.24c). With the online sources, we use additional caching levels for the raw downloaded tiles (before compositing), including a persistent local cache at each rendering node. Occasionally, we consolidate the 72 individual persistent cache and redeploy them to the nodes in order to minimize the traffic to off-site web services. Finally, the full integration of the virtual texturing with the SceniX scene-graph library allows the terrain and maps to be combined with 3D GIS data, such as buildings reconstructed from LiDAR point clouds and building footprints, or detailed landmark 3D models (e.g., Figure 5.28c).

Some datasets require custom visualization or interaction tools for effective exploration. One example is the climate data from NASA's BlueMarble Next Generation project, which contains ice and vegetation coverage for the entire planet, optionally with bathymetry and relief shading. There is a separate tile set for each month of 2004. We combine all the tiles in a single virtual texture and use the moving window implementation from the Infinite Canvas (see Section 6.3) to select a specific month. We also implemented additional logic in the visibility analysis rendering pass to start prefetching tiles for the adjacent months even though those tiles are not currently active. With these modifications, we can support quick toggling between months for visual differential analysis or using a slider for a smoother transition. An example of visualizing this data in the Reality Deck is presented in Figure 5.24d.



**Figure 5.24:** Interactive visualization of world-wide GIS data in the Reality Deck.

### 5.3.2 Case Study: Fusion of GIS and procedural modeling for driving simulation in New York

The GIS data we have considered so far is global in nature and may not be sufficient for the detailed analysis of a particular area. There are also a multitude of geolocated datasets that deal with human activity, such as transportation networks, building footprints, land zoning and data from the social sciences. Here we consider the generation of 3D models from the GIS data for driving simulation in the area of New York City. Many of the traditional sources of GIS data do not provide sufficient resolution at the ground level and therefore we combine GIS with procedural modeling of street-level details. Both the Immersive Cabin and the Reality Deck were used as visualization platforms during the design of the model and for interactive visual validation of the results.

The most critical aspect in modeling the terrain of New York is obtaining a high resolution Digital Elevation Model (DEM). We have evaluated a number of publicly available sources and have identified the DEMs provided by the United States Geological Survey (USGS) as the most suitable source of data. In

## 5. IMMERSIVE ENVIRONMENTS

---

contrast to the global SRTM data, the USGS DEMs are available in 1 arc-second (approx. 30m) resolution and 1/3 arc-second (approx. 10m) resolution, and the quality for our target area was significantly higher. 3m resolution DEMs are available for parts of the country, including parts of Long Island, however, we found those files to contain a number of errors and missing coverage. In our initial experiments we decided to focus on the following area:

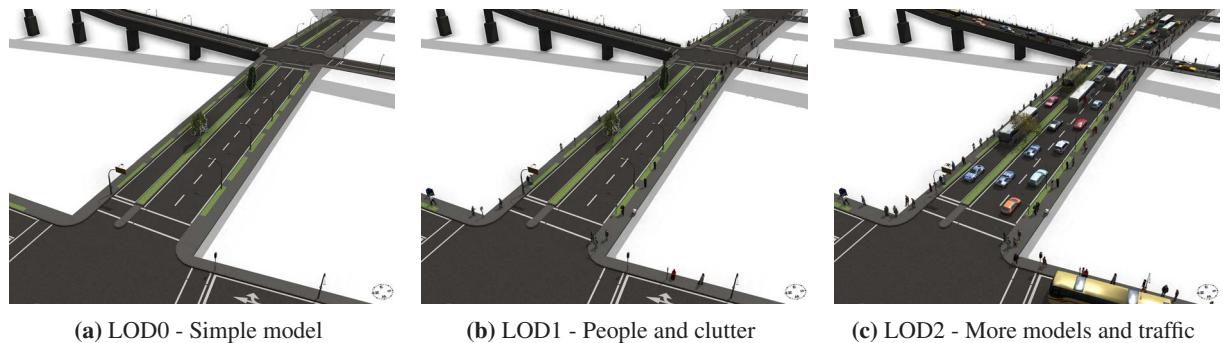
- Latitude: [40°31'59.8794", 41°0'0"]
- Longitude: [-73°24'0", -74°9'0"]
- Resolution: 8739 × 7153 (10m per pixel)

While 10m horizontal resolution is sufficient for resolving the general shape of the terrain, it is generally not sufficient for the highly detailed ground-based simulations that we want to create. It is unlikely that higher resolution DEMs will be available in the foreseeable future, however there exist very detailed topographic maps that can be used to augment the existing 10m DEMs. Such maps contain detailed contours for a specific set of elevation levels. While the elevation resolution is limited depending on the number of contour curves, the curves themselves can be accurate down to sub-1m resolution and provide a better representation of artificial terrain features. The contours we obtained were for a part of New York. The curves are discretized into segments and the areas between adjacent curves are triangulated. Once we have this surface description of the terrain, we can sample the surface at arbitrary grid position and therefore obtain a higher resolution elevation model. For the following results, the contour map is sampled at 3m horizontal resolution and merged with the original 10m DEM (also resampled to 3m) to obtain the final land elevation data.

The elevation data obtained from the USGS does not contain any bathymetry data, or underwater topography. While the depth of the ocean floor is not critical to this project, it is nevertheless needed when modeling bridges and waterfront streets. We use data provided by the Coastal and Marine Geology Program at USGS for the Long Island Sound estuary. 5m resolution bathymetry contours are available for the entire target area, while a 1m dataset is available for the actual estuary between Connecticut and Long Island. The procedure for integrating the contour data into the elevation map is similar to the land case.

We have prepared a number of overlay maps that are registered to the DEMs, including a number of non-realistic maps:

- Atlas relief maps (3m, 10m)
- USGS scanned topographic map (2m, 10m)
- OpenStreetMap (0.5m, 2m)
- Aerial imagery (0.5m, 1m, 2m, 10m)



**Figure 5.25:** Roads generated procedurally from the GIS street network and attributes at different levels of detail.

- High resolution topography (0.5m, 2m, 10m)

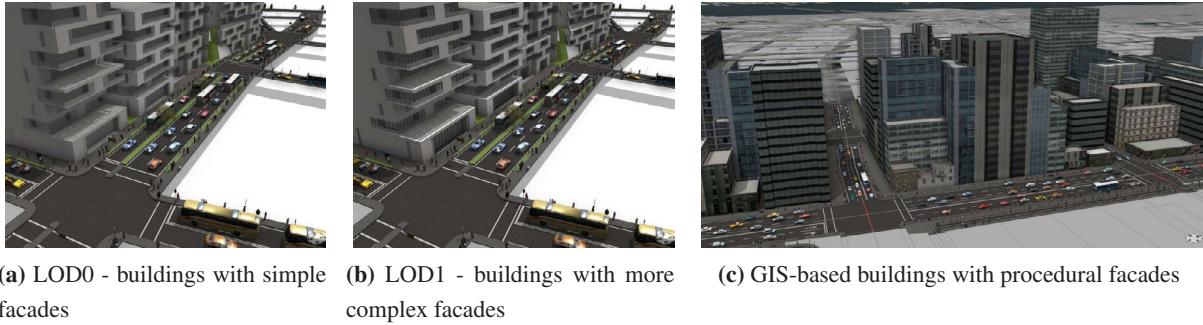
The goal is for these maps is to serve as a reference during the modeling of the street network, especially in very complex situations, such as the entrance and exit ramps of major highways. The high resolution aerial imagery (see Figure 5.20) can additionally be used during the modeling of landmark buildings. Many of these maps are not suitable for traditional visualization applications. For the selected area, 5m horizontal resolution reaches the texture size limit of all GPUs available on the market and visualizing the higher resolution maps requires support for virtual texturing. For example, the 0.5m aerial imagery is a  $139,810 \times 114,428$  map and contains 16 gigapixels, or about 50GB of image data.

It is possible to model a small section of the NY road network based on the reference maps we have prepared. However, this approach does not scale beyond a handful of medium-length streets as the task of manually tracing each street becomes unmanageable. The procedural rules in CityEngine are defined through a CGA shape grammar, which can be parametrized to generate a variety of roads, building shapes and building facades. The creation of the virtual city model starts with the definition of the street network for the target area. While CityEngine can generate a variety of random street networks that approximate real cities, it can also use import real street networks. We evaluated a number of GIS data sources, for example ArcGIS and OpenStreetMap, but chose the street centerline data from the NYC Open Data project. The major advantage compared to other sources is that the street widths are available, allowing for a more realistic model of the area. The major disadvantage, which is also shared with the other GIS sources, is that the graph is 2D only and does not capture 3D road features such as ramps, tunnels and overpasses. As a result, it required a very significant cleaning effort to produce a usable street network in CityEngine.

The street network is assigned a grammar file that generates a variety of road shapes at different levels of detail. Figure 5.25 shows a sample road generated with the different grammar parameters. In general, we use LOD0 and LOD1 for the large urban areas. LOD2 is used in the examples below, however the static car models are not used in the actual simulator, which provides its own dynamic traffic system.

## 5. IMMERSIVE ENVIRONMENTS

---



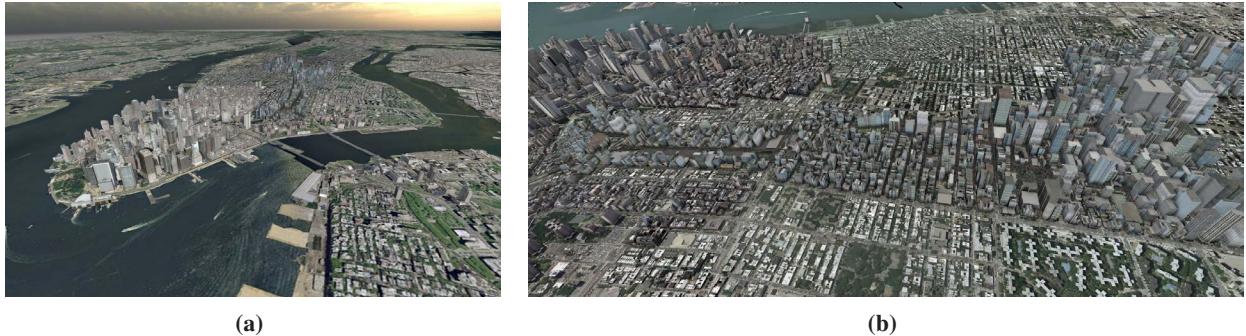
**Figure 5.26:** (a) Buildings generated procedurally on randomized lots within the GIS street network. (b) At the higher levels of detail more facade details are generated. (c) Buildings generated from accurate GIS footprints and heights, together with procedural photo-based facades.

We have created a grammar file that creates interesting building shapes without using any additional GIS data. The street network defines a set of enclosed blocks which are recursively tessellated and each sub-block is used to grow a building and optionally a garden. A number of different layouts/presets are available and the parameters are randomized so that we do not have repeating structures. The facades are available in different levels of detail, but in general all versions have high geometric details and are suitable for a driving simulator (see Figure 5.26a and Figure 5.26b).

The GIS data from NYC Open Data includes accurate building footprints for the entire area. We have created a second building grammar that uses the footprints to create virtual buildings that generally match the overall shape of the real-world buildings. For the Manhattan area, we have created a map containing many of the building heights, which together with the outlines result in relatively realistic building geometries. The building elevation for other areas is not available so we randomize that parameter based on an overall inspection in Google Earth. The facades for the GIS building grammar use photographs of urban centers to approximate the feel of New York. We have also obtained a large collection of photographs from NYC, which could be used in the future to create more realistic facades. Figure 5.26c shows an example area in Manhattan that uses the GIS building grammar.

Finally, we have obtained a 3D model for Manhattan that is based on aerial photographs and LIDAR data. This results in very accurate and realistic models, however the resolution is not sufficient for the extreme close ups during a driving simulation. The resolution is too low in both the textures and the geometry. However, this 3D model can be used as a very effective backdrop during simulations farther away from Manhattan. Figure 5.27a shows an image from that data. This model can be added during the processing in 3D Studio Max together with the sky, the lighting and any minor edits to the final model that may be needed. It can be seen in the foreground of Figure 5.27a.

We chose two tracks that demonstrate different capabilities of the content creation pipeline. The first



**Figure 5.27:** Different views of the 3.3 mile Manhattan track. On the foreground in (a) are low-resolution landmark 3D models obtained from LIDAR data, which are used as a backdrop during the driving simulation.

is a 3.3 mile route through Manhattan (see Figure 5.27a). The street network is relatively regular and easy to process, except for the southern end with the bridge exits. Since the GIS data is 2D, the bridges were created manually. Note that while the GIS data is 2D, we project it onto the terrain model so that we have accurate elevations for the streets. Certain complex intersections were simplified so that the streets can be generated in CityEngine, but overall, the final network is close to the real-world network and the streets have the correct number of lanes and medians (those were edited manually based on observations in Google Earth). Since the area is relatively small, we have added a large number of props to the scene, including urban clutter and human models.

The other route covers a much larger area and a 13 mile track from JFK through Queens and toward Manhattan (see Figure 5.28c). This route was very challenging due to the sheer size and the low quality of the GIS data. The required manual cleaning was extensive, but the final street network generates a 3D model that matches well with the real-world data (again, road parameters were tweaked during observations with Google Earth, particularly for the number of lanes on the streets, which are not available from the GIS data). We chose to use the GIS building grammar with randomized building heights. Figure 5.28 shows snapshots of the visual validation of the final models in the Immersive Cabin and the Reality Deck. The IC visualization was primarily used to verify the placement of the models in 3D and for a ground-level flythrough with stereoscopic rendering (see Figure 5.28a and Figure 5.28b). The RD visualization uses our highest resolution 0.5m gigapixel maps to verify the correspondence with the generated models from an overhead view where the high pixel density was most effective (see Figure 5.28c).

### 5.3.3 Case Study: Immersive Medical Visualization

The immersive visualization in the Immersive Cabin and the Reality Deck provides a unique exploration experience for 3D lattice data, whether that is the LBM simulations in Chapter 3, physical samples scanned

## 5. IMMERSIVE ENVIRONMENTS

---

with Computed Microtomography (CMT), or non-invasive medical CT or MRI scans. The common thread in all these types of data is that they contain intricate 3D structures that can be studied effectively using stereoscopic rendering and motion parallax. In many cases, virtual flight through the data is also important, for example, moving through the colon during examination with Virtual Colonoscopy, following the porous structures of a soil sample in a CMT scan, or visualizing a 3D dispersion simulation together with the underlying urban mesh model. Finally, the sizes of data in all of these areas are growing faster than the resolutions of traditional display devices, and high resolution facilities such as the Reality Deck provide the means of managing that growth in terms of building effective visualization systems. In the following, we focus specifically on examples of immersive medical visualization.

Our first example is a protein visualization system used to study the effects of stereoscopy and body motion within an immersive visualization environment. Our data is from the Protein Data Bank (PDB) and we use the structural information in the PDB files to synthesize 3D meshes that can be displayed directly in our visualization facilities, either with the OpenGL or the interactive raytracing pipelines. Figure 5.29 contains snapshots from exploration sessions. For the IC, the user controls the system with a gamepad or the 3DConnexion SpaceMouse. The head orientation component from the tracking data is used to control the direction of the virtual translation, while the head position is used to generate the off-axis projection matrices. The latter allows the user to physically move within the visualization, effectively simulating molecules within the physical space of the IC. In our experience over many demonstrations of the systems (and an informal user study), even this limited body interaction significantly improves the immersion in the data and allows for more effective understanding of its 3D structure. The Reality Deck is less effective for 3D visualization because of the 2D displays and the bezels. However, the larger physical space allows for more walking interaction through the tracking system and in fact with careful scaling, much of the protein data can be explored without any controllers by simply moving within the space. Motion parallax with head tracking remains effective and in fact it minimizes the effect of the screen bezels [2]. The results generalize to other dataset which contain intricate 3D structures together with large amounts of connectivity information, such as in 3D brain tractography. We plan to explore that particular application in detail.

Virtual Colonoscopy (VC) has been established as a non-invasive alternative to traditional Optical Colonoscopy (OC) for cancer screening [49, 58] with improved coverage of the colon surface [50]. Our visualization support a number of immersive rendering techniques related to VC in both the IC and the RD. While Figure 5.1a is generated with the mesh rendering pipeline for the colon surface, we also support direct volume rendering of the CT data over all the major lattice types (see Figure 5.30a for the volume rendering result in the IC). A major limitation for clinical deployment is that in partially immersive environments such as the IC and the RD, significant areas of the colon surface may be skipped due the missing ceiling in the

IC and the missing floor and ceiling in the RD. In Section 6.1 we describe a novel visualization technique based on conformal mapping to address this limitation.

The 3D visualization is less effective in the Reality Deck. We have developed a technique for gigapixel colon rendering where the entire colon is first flattened to a rectangle and we use virtual cameras along the colon centerline to generate a gigapixel image with volume rendering. The resulting image is then displayed across the Reality Deck with our virtual texturing framework, taking advantage of the fully enclosed display when the door is closed. In Figure 5.30b we show two different complete scans for a single patient (in supine and prone position) which have been registered with quasi-conformal mapping [129]. The advantage is that a radiologist can examine the entire colon at once. This visualization can also be integrated with more traditional 3D views overlaid near suspicious areas, for example.

Finally, we demonstrate a more general volume rendering example with the Visible Human dataset, which is a  $512 \times 512 \times 1877$  scalar dataset with 16 bit precision. The visualization in the IC leverages the tracking system to allow for a more natural exploration through movement, in addition to using a gamepad or 3DConnexion SpaceMouse for the navigation. As can be seen in Figure 5.31a, the 12 dpi pixel density is a problematic with high frequency transfer functions or when the user moves closer to the display. In contrast, the Reality Deck has sufficient resolution to display virtually all the interesting structures in the data (see Figure 5.31b).

The resolution of the Reality Deck introduces a new set of problems when expensive visualization techniques such as volume rendering and raytracing are used. The problem is that the traditional double-buffered rendering requires that a complete frame is rendered before it is sent to the display. As the resolution per GPU in the RD is  $7960 \times 3022$ , it may take up to 10 sec. to generate a full resolution image with volume rendering, during which time the user interface is completely frozen. The example in Figure 5.31b was generated at  $1990 \times 754$  resolution per GPU, which allows for interactive exploration. In Section 6.2 we describe a progressive rendering framework based on frameless rendering and BCC lattice reconstruction, which can be used for full-resolution volume rendering in the Reality Deck.

## 5. IMMERSIVE ENVIRONMENTS

---



(a) Immersive Cabin: Caton Ave. near Prospect Park Parade Ground

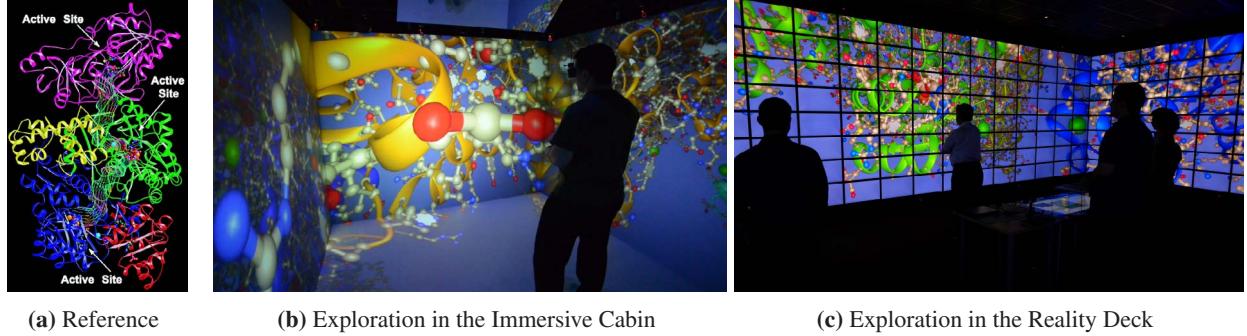


(b) Immersive Cabin: E. Broadway and Forsyth Street

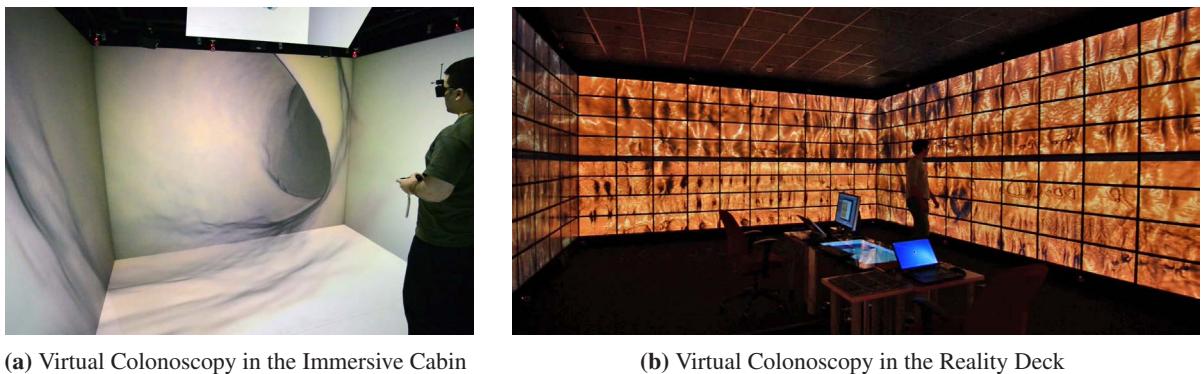


(c) Reality Deck: JFK and Manhattan tracks, LIDAR buildings and gigapixel terrain

**Figure 5.28:** (a),(b) Snapshots of the 3D model in the Immersive Cabin for visual validation with stereoscopic rendering. (c) Visualization of the entire model in the Reality Deck, including the two generated tracks at the maximum LOD, the backdrop LIDAR buildings and a gigapixel terrain from aerial photographs (the IC version uses high resolution but not gigapixel terrain texture). A version of the terrain model is also shown in Figure 5.20.



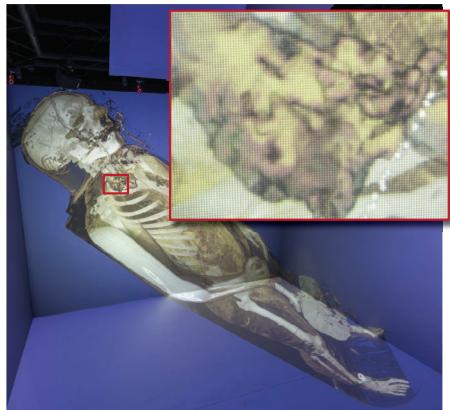
**Figure 5.29:** Exploration of (a) structure *1kee* from the Protein Data Bank in (b) the Immersive Cabin and (c) the Reality Deck. Only the left-eye images from the stereoscopic pair in the IC are shown to improve the clarity of the image.



**Figure 5.30:** Visualization for Virtual Colonoscopy in (a) the Immersive Cabin and (b) the Reality Deck. The IC presents the traditional 3D navigation interface following with centerline and using volume rendering. The same interface is available in the RD, but (b) presents a novel gigapixel visualization where two scans of the entire colon of the patient are displayed at once.

## 5. IMMERSIVE ENVIRONMENTS

---



(a) Exploration in the Immersive Cabin



(b) Exploration in the Reality Deck

**Figure 5.31:** Exploration of the Visible Human dataset in (a) the Immersive Cabin and (b) the Reality Deck. The high pixel density in the RD allows for a more natural visualization compared to the 12 dpi pixel density in the IC, which requires constant translation and zooming.

# 6

## Distributed Immersive Visualization

## **6. DISTRIBUTED IMMERSIVE VISUALIZATION**

---

### **6.1 Conformal Visualization**

In Virtual Reality, immersive systems such as the CAVE are an important tool for the collaborative exploration of large 3D data. Unlike head-mounted displays, these systems are often only partially immersive due to space, access or cost constraints. The resulting loss of visual information becomes a major obstacle for critical tasks that need to utilize the users' entire field of vision. We have developed a conformal visualization technique that establishes a conformal mapping between the full 360° field of view and the display geometry of a given visualization system. The mapping is provably angle-preserving and has the desirable property of preserving shapes locally, which is important for identifying shape-based features in the visual data. We apply the conformal visualization to both forward and backward rendering pipelines in a variety of retargeting scenarios, including CAVEs and angled arrangements of flat panel displays. In contrast to image-based retargeting approaches, our technique constructs accurate stereoscopic images that are free of resampling artifacts. Our user study shows that on the visual polyp detection task in Immersive Virtual Colonoscopy, conformal visualization leads to improved sensitivity at comparable examination times against the traditional rendering approach. We also develop a novel user interface based on the interactive recreation of the conformal mapping and the real-time regeneration of the view direction correspondence.

A number of visualization technologies have been developed for the immersive exploration of complex large-scale data. Prime examples are the CAVE [24] and Head-Mounted Displays (HMD) which provide a much larger field of view into a virtual environment compared to traditional desktop systems, and also use stereoscopic pairs of images to improve the perception of spatial relationships. While HMDs allow for arbitrary views in the virtual world, they are usually bulky, wired and easily lead to eye fatigue. At the same time, CAVEs provide a much more natural visualization without the need for a virtual avatar. Building a fully enclosed CAVE, however, remains a difficult task. Although such installations exist [47], they present an engineering challenge in terms of cost, facility access, as well as head and gesture tracking.

*Immersion* in the virtual data is a function of different factors, such as sensory perceptions, interaction techniques and realism of the visualization. We focus on *visual immersion*, defined by the field of view coverage afforded by the physical visualization platform, as well as the coverage of the virtual scene provided by the visualization software. The disadvantage of partially-immersive environments, such as CAVEs with at least one missing display surface, is that important visual information may be lost. While many applications may tolerate one or more missing projection screens, this partial loss of visual context adversely affects the general navigation capabilities of the user and becomes a critical issue in the exploration of medical data.

We have developed a visualization approach that utilizes conformal mapping to modify scene geometries or viewing rays at runtime, depending on the rendering modality. As a result, the full virtual environment can

be displayed on a partially-immersive visualization platform, for example a 5-sided CAVE, without the artifacts typically associated with image-based retargeting approaches. In mathematics, the conformal map is an angle preserving function that describes a mapping between two Riemannian surfaces [108]. Intuitively, it allows us to map the geometry of the fully-immersive 6-sided CAVE to an arbitrary configuration of display surfaces that is topologically equivalent to a disk, such as a 5-sided CAVE or a non-planar arrangement of flat panel displays. This mapping is then used to transform the viewing directions during rendering with ray tracing, for example. The main advantage of using a conformal map to define the transformation is the guarantee that shapes are preserved locally even though distances are not. This is particularly beneficial for the exploration of medical data, such as in Virtual Colonoscopy (VC) where potentially cancerous polyps are detected by the radiologist based on their shape.

The following are the specific contributions of our work:

- We develop a conformal visualization technique that establishes a shape-preserving transformation between the  $360^\circ$  field of view and an arbitrary display configuration. We demonstrate applications to visibility retargeting for CAVEs with 3, 4 and 5 display surfaces, as well as for angled arrangements of flat panel displays.
- The conformal transformation of the viewing directions is applied during rendering with rasterization, direct volume rendering and ray-tracing. Compared to the traditional image-based retargeting approaches, our technique does not introduce resampling artifacts and constructs accurate stereoscopic image pairs for all three rendering modalities using a unified transformation definition.
- Our system computes the conformal mapping at interactive speeds and regenerates the viewing ray correspondence on the GPU at real-time speeds allowing for a novel user interface for the dynamic manipulation of the visibility. The performance penalty for applying the conformal transformation during rendering is approximately 1% for ray tracing and volume rendering.
- Our approach is applied to the visualization of both mesh and volume data under 3 different retargeting scenarios. Our user study shows an improvement in the task of visual polyp detection in phantom colonoscopy datasets during Immersive Virtual Colonoscopy.

### 6.1.1 Theoretical Background

Our conformal visualization system uses the discrete Ricci flow [56] to compute the mapping between the full  $360^\circ$  spherical field of view and the partially-immersive platforms. The main advantage is that the local shape of the projected scene geometry and the features in volume data are preserved under the conformal transformation. The Ricci flow method has found a broad range of applications in the graphics and

## 6. DISTRIBUTED IMMERSIVE VISUALIZATION

---

visualization fields, including optimal surface parameterization [121, 128], shape analysis [57] and surface matching [69].

Least Square Conformal Mapping (LSCM) is an alternative heuristic technique for computing a conformal mapping [68] and it is also used in a variety of visualization techniques. For example, Raskar et al. have used the LSCM to display the output of an ad-hoc cluster of projectors onto an arbitrary set of surfaces [91]. Unlike Ricci flow, which automatically produces seam-free global conformal parameterizations, LSCM and other computational approaches often require special algorithms or heuristic inputs when dealing with certain topologies [56]. The main disadvantage of LSCM is that it cannot control the boundary of the region. In our case, we require a mapping between the visibility boundary of the CAVE and the unit circle. Therefore, LSCM is not a viable technique for our application. Furthermore, unlike the approach by Springborn et al. [101], the Ricci flow is a stable algorithm with a theoretical assurance of convergence.

We start by briefly describing the theoretical foundations of our conformal visualization approach. The overall goal is to develop a 1-to-1 mapping between two surfaces that represent the source and target visibility geometries. In both cases, the geometry is constructed as the section of the field of view that a visualization system can cover from a certain viewpoint. For our application, the source geometry is most often defined as the complete unit sphere around the viewpoint.

According to the Riemann mapping theorem, simply connected surfaces with a single boundary can be mapped onto the planar disk, such that the mapping is *angle-preserving*. Locally, the mapping is only a scaling and therefore it is also shape-preserving. This is advantageous in our rendering technique as it maps the viewing directions from the 6 original projections in the fully-enclosed CAVE to the 5-sided CAVE while preserving the local shapes. Our computational algorithm is based on the surface Ricci flow theorem [18].

### Conformal Mapping

Let  $S_1$  and  $S_2$  be two surfaces with Riemannian metrics  $\mathbf{g}_1$  and  $\mathbf{g}_2$ , and let  $\phi : (S_1, \mathbf{g}_1) \rightarrow (S_2, \mathbf{g}_2)$  be a homeomorphism between them. We say  $\phi$  is *conformal*, if the pull back metric tensor induced by  $\phi$  on the source differs from the original metric by a scalar:

$$\phi^* \mathbf{g}_2 = e^{2\lambda} \mathbf{g}_1,$$

where  $\lambda : S_1 \rightarrow \mathbb{R}$ , is a function. The following Riemann mapping theorem plays a fundamental role in the current work:

**Theorem 6.1.1 (Riemann Mapping)** *Suppose a surface  $S$  is simply connected with a Riemannian metric. There exist conformal mappings  $\phi : S \rightarrow \mathbb{D}$ , where  $D$  is the unit disk on the complex plane and all such mappings differ by a Möbius transformation.*

A Möbius transformation of a point  $z$  on the complex plane is given by

$$z \rightarrow e^{i\theta} \frac{z - z_0}{1 - \bar{z}_0 z},$$

where  $\theta$  and  $z_0 \in \mathbb{C}$  are constants.

In order to compute the mapping  $\phi$ , one can compute the pull back metric first, which can be achieved by the surface Ricci flow.

## Ricci Flow

A surface Ricci flow is the process used to deform the Riemannian metric of the surface. The deformation is proportional to the Gaussian curvature so that the curvature evolves in a manner similar to heat diffusion. In mathematics, it is a powerful tool for finding a Riemannian metric satisfying the prescribed Gaussian curvature. Chow and Luo have described the theoretic foundation for the discrete Ricci flow on surfaces [18], and Jin et al. have developed an efficient computational algorithm [56].

Let  $\Sigma = (V, E, F)$  be a triangular mesh embedded in  $\mathbb{R}^3$ , where  $V$ ,  $E$  and  $F$  are respectively the sets of vertices, edges, and faces. A *discrete Riemannian metric* on  $\Sigma$  is a piecewise constant metric with cone singularities at the vertices. The edge lengths are sufficient to define a discrete Riemannian metric,

$$l : E \rightarrow \mathbb{R}^+, \quad (6.1)$$

as long as, for each face  $f_{ijk} = [v_i, v_j, v_k]$ ,  $f_{ijk} \in F$ , the edge lengths satisfy the triangle inequality:  $l_{ij} + l_{jk} > l_{ki}$ .

For simplicity, we use  $e_i$  to denote the edge against the vertex  $v_i$ , namely  $e_i = [v_j, v_k]$ , and  $l_i$  is the edge length of  $e_i$  in triangle  $f_{ijk}$ . The cosine laws are given by:

$$l_k^2 = l_i^2 + l_j^2 - 2l_i l_j \cos \theta_k \quad (6.2)$$

The *discrete Gaussian curvature*  $K_i$  on a vertex  $v_i \in \Sigma$  can be computed as the angle deficit,

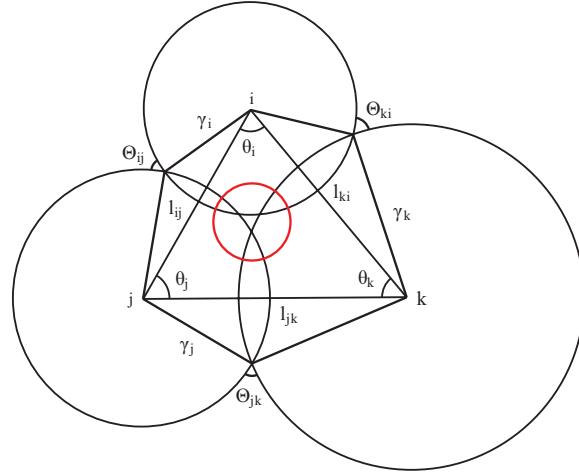
$$K_i = \begin{cases} 2\pi - \sum_{f_{ijk} \in F} \theta_i^{jk}, & v_i \notin \partial\Sigma \\ \pi - \sum_{f_{ijk} \in F} \theta_i^{jk}, & v_i \in \partial\Sigma \end{cases} \quad (6.3)$$

where  $\theta_i^{jk}$  represents the corner angle attached to vertex  $v_i$  in all the face  $f_{ijk} \in F$  that share vertex  $v_i$ , and  $\partial\Sigma$  represents the boundary of the mesh  $\Sigma$ .

The circle packing metric (see Figure 6.1) was introduced to approximate the conformal deformation of metrics [104, 108]. The function  $\Gamma : V \rightarrow \mathbb{R}^+$  assigns a radius  $\gamma_i$  to each vertex  $v_i \in V$ . Similarly, the weight function  $\Phi : E \rightarrow [0, \frac{\pi}{2}]$  associates the acute angle  $\Theta_{ij}$  with each edge  $e_{ij} \in E$ , where  $\Theta_{ij}$  is defined

## 6. DISTRIBUTED IMMERSIVE VISUALIZATION

---



**Figure 6.1:** Circle Packing Metric.

as the angle of intersection of the circles at vertices  $v_i$  and  $v_j$ . The circle packing metric for  $\Sigma$  is the pair  $(\Gamma, \Phi)$ .

Let  $u : V \rightarrow \mathbb{R}$  be the *discrete conformal factor*, which measures the local area distortion, where  $u_i = \log \gamma_i$  for each vertex  $v_i \in V$ . Then, the discrete Ricci flow is described by the following:

$$\frac{du_i(t)}{dt} = (\bar{K}_i - K_i), \quad (6.4)$$

where  $\bar{K}_i$  is the prescribed curvature at vertex  $v_i$ . The discrete Ricci flow can also be formulated in the variational setting, namely, it is the negative gradient flow of some special energy form:

$$f(\mathbf{u}) = \int_{\mathbf{u}_0}^{\mathbf{u}} \sum_{i=1}^n (\bar{K}_i - K_i) du_i, \quad (6.5)$$

where  $\mathbf{u}_0$  is an arbitrary initial metric. The integration above is well-defined, and it is called the Ricci energy. Then, the discrete Ricci flow is the negative gradient flow of the discrete Ricci energy and the discrete metric which induces  $\bar{\mathbf{k}} = (\bar{K}_1, \bar{K}_2, \dots, \bar{K}_n)^T$  is the minimizer of that energy.

Computing the desired metric with prescribed curvature  $\bar{\mathbf{k}}$  is equivalent to minimizing the discrete Ricci energy, which is strictly convex (namely, its Hessian is positive definite). The global minimum exists uniquely, corresponding to the metric  $\bar{\mathbf{u}}$ , which induces  $\bar{\mathbf{k}}$ . The discrete Ricci flow converges to this global minimum [18] and the global conformal parameterization for  $\Sigma$  can be computed in an automated and robust fashion.

---

**Algorithm 1** Discrete Ricci Flow

**Require:** Triangular mesh  $\Sigma$ , target curvature for each vertex  $\bar{K}_i$ , error threshold  $\epsilon$ .

**Ensure:** Discrete metric (edge lengths) satisfying the target curvature.

```

1:  $\mathbf{u} = [u_i], \mathbf{v} = [v_i]$  for  $\mathbf{u}, \mathbf{v} \leftarrow 0$ 
2: while true do
3:   Compute edge length  $l_{ij}$  for edge  $[v_i, v_j]$ :  


$$l_{ij} = e^{u_i} + e^{u_j} + 2 \cos \phi_{ij} e^{u_i + u_j}$$

4:   Compute the corner angle  $\theta_i^{jk}$  in triangle  $[v_i, v_j, v_k]$ :  


$$\theta_i^{jk} = \cos^{-1} \frac{l_{ij}^2 + l_{ki}^2 - l_{jk}^2}{2l_{ij}l_{ki}}$$

5:   Compute the curvature  $K_i$  at  $v_i$ :  


$$K_i = \begin{cases} 2\pi - \sum_{f_{ijk} \in F} \theta_i^{jk}, & v_i \notin \partial\Sigma \\ \pi - \sum_{f_{ijk} \in F} \theta_i^{jk}, & v_i \in \partial\Sigma \end{cases} \quad \mathbf{K} = [K_i]$$

6:   if  $\max |\bar{K}_i - K_i| < \epsilon$  then
7:     return the discrete metric  $l_{ij}$ 
8:   end if
9:   Update  $\mathbf{u}$ :  

Compute the Hessian Matrix  $H$ ,  $H_{ij} = \frac{\partial K_i}{\partial u_j}$   


$$\mathbf{u} \leftarrow \mathbf{u} - H^{-1}(\bar{\mathbf{K}} - \mathbf{K})$$

10: end while

```

---

### Discrete Ricci flow

Suppose  $\Sigma$  is a triangle mesh embedded in  $\mathbb{R}^3$ . We associate each vertex  $v_i$  with a circle  $(v_i, \gamma_i)$  where  $\gamma_i$  equals the minimal length of any edge in the immediate neighborhood of  $v_i$ . Then we compute the intersection angle  $\Theta_{ij}$  such that the circle packing metric is as close to the induced Euclidean metric as possible.

We compute the curvature at each vertex  $v_i$  and adjust the conformal factor  $u_i$  in proportion to the difference between the target curvature  $\bar{K}_i$  and the current curvature  $K_i$ . Then, we update the metric, recompute the curvature, and repeat this procedure until the difference between the target curvature and the current curvature is less than the given threshold. Algorithm 1 summarizes the computational steps and more details can be found in the work of Jin et al. [56].

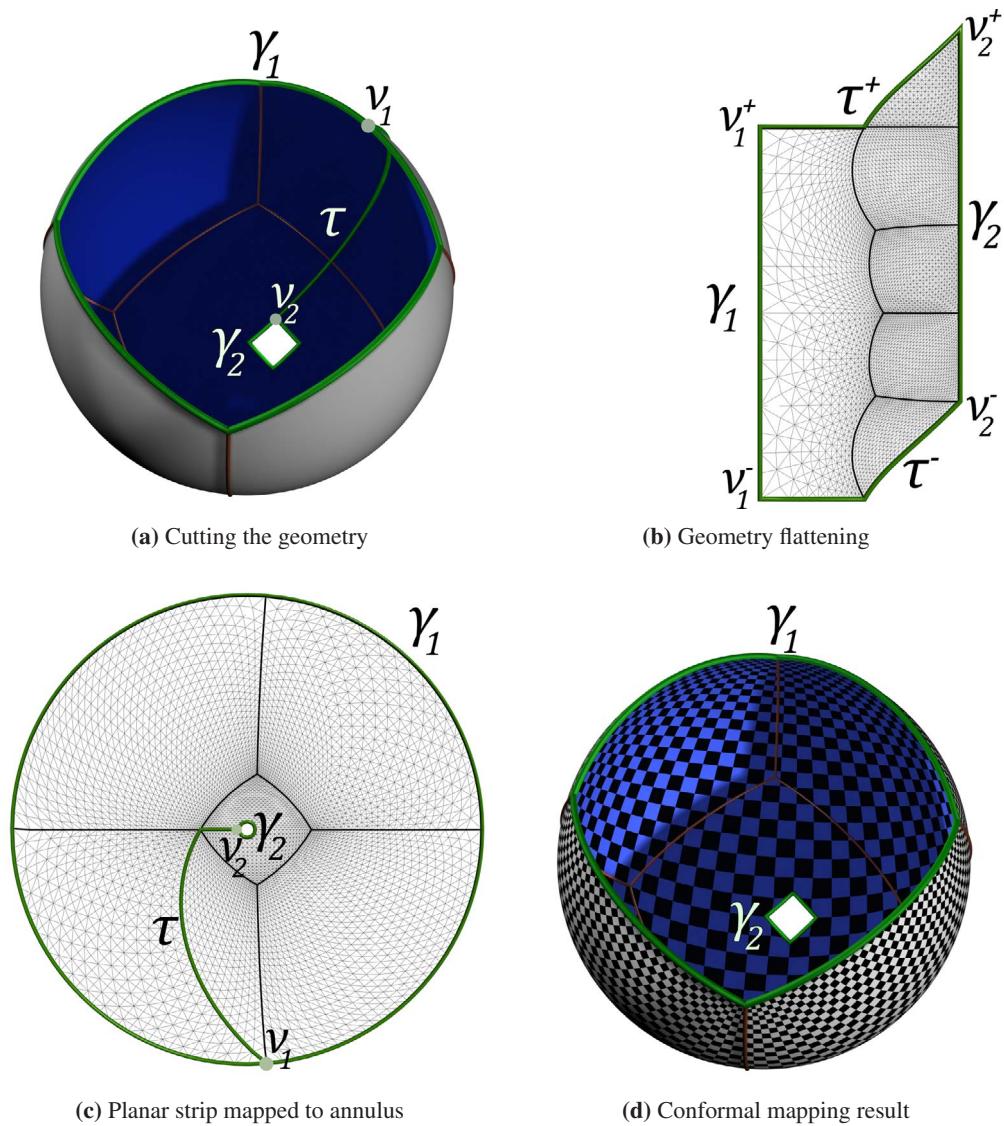
### Riemann Mapping

Figure 6.2 illustrates the algorithm for computing the Riemann mapping. We remove a face from the mesh  $\Sigma$  to convert it to a topological cylinder (Figure 6.2a), resulting in the creation of a new boundary  $\gamma_2$ . The target curvature for both the interior and the boundary vertices is set to zero. The Ricci flow described in

## 6. DISTRIBUTED IMMERSIVE VISUALIZATION

---

Algorithm 1 produces a flat cylinder that is periodically embedded in the complex plane (Figure 6.2b). Each period of the embedding is a rectangle and the original boundaries  $\gamma_1$  along the cut face and  $\gamma_2$  are aligned with the imaginary axis in the complex plane. The cylinder is then mapped to the unit disk with the hole in the center of the image by the exponential map  $e^z$  (Figure 6.2c). Figure 6.2d illustrates the mapping by texture mapping a checkerboard pattern back on the cut mesh. As a final step, the removed face is inserted back into the mesh, yielding the conformal mapping on the original mesh.



**Figure 6.2:** Riemann Mapping Algorithm

## Practical Application

The previous sections introduce the theoretical foundation of the Discrete Ricci flow algorithm, which is the base algorithm in our conformal visualization technique. Using the 5-sided CAVE as an example, we consider all the possible view directions mapped onto a unit sphere for a reference position at the center of the environment. This sphere is cut at the position that corresponds to the center of the missing surface (Figure 6.3c) and towards the four corners. The top edges of the CAVE are also mapped to the sphere (Figure 6.3a) and they define the visibility boundary  $\partial\Sigma$  for the central CAVE position. We apply the Discrete Ricci flow to compute the conformal maps for both spheres, which are then aligned to provide a 1-to-1 mapping between the two surfaces (see Figure 6.3b and Figure 6.3d). Using this mapping, the full set of viewing directions defined over the 6-sided CAVE is then projected onto the geometry corresponding to the 5-sided CAVE. We then encode the viewing directions into a cube map which can be sampled efficiently during rendering. Although our rendering framework can utilize conformal maps produced with any computational algorithms, the automation, robustness and efficiency of the Ricci flow provide an advantage for quickly generating the mappings between the different display topologies.

### 6.1.2 Implementation Details

Our conformal visualization utilizes an efficient pipeline for generating the conformal mapping at interactive speeds. The input is a user-specified rendering target, such as an  $n$ -sided CAVE or an arrangement of displays, as well as a set of parameters that control the accuracy of the generated maps.

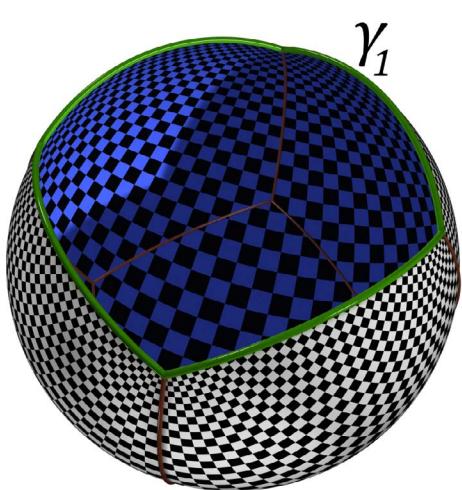
#### Mesh Templates

Our mesh processing toolkit utilizes a half-edge data structure to explicitly represent the mesh connectivity information. We start by generating the templates for the source and target visibility meshes. Although the conformal mapping is performed over the visibility spheres, the geometries at this stage are simple cubes in order to facilitate a more intuitive definition of the visibility boundaries and the cuts. Figure 6.4 and 6.6 illustrate the templates for the 5-sided and the 4-sided CAVE respectively. In both cases, the templates are parameterized with a reference point whose projection on the walls (shown as green spheres) define the intersection points of the cut.

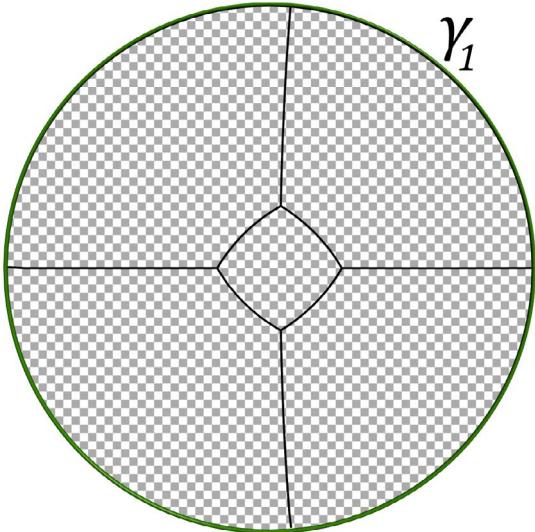
A similar template can be defined for the 3-sided CAVE as well (Figure 6.5a). However, because of the increased length of the cut, mapping to an arrangement of displays is not practical as the conformal visualization would introduce significant distortion artifacts. For a target mesh similar to Figure 6.5c, we map only a hemisphere of the original viewing directions (Figure 6.5b).

## 6. DISTRIBUTED IMMERSIVE VISUALIZATION

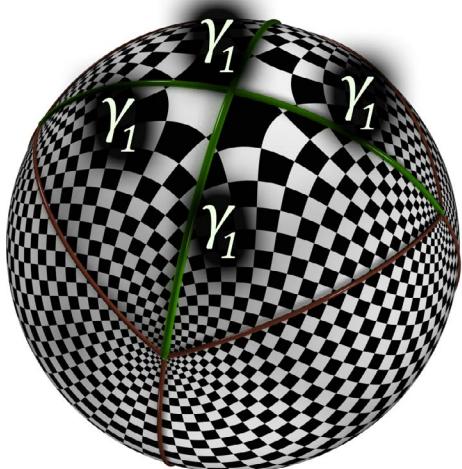
---



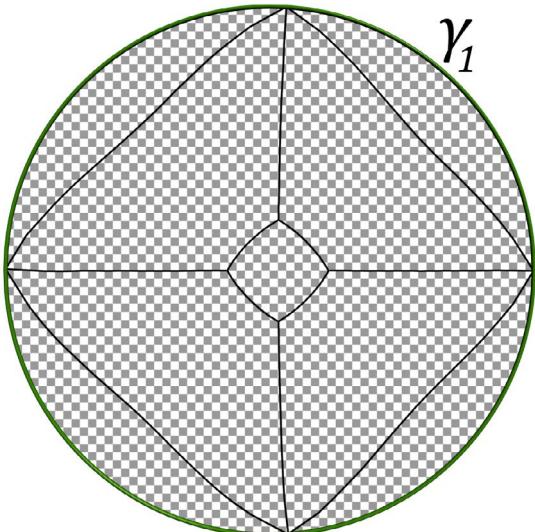
(a) 5-sided CAVE mapped to the sphere



(b) Conformal mapping of (a) to the unit disk

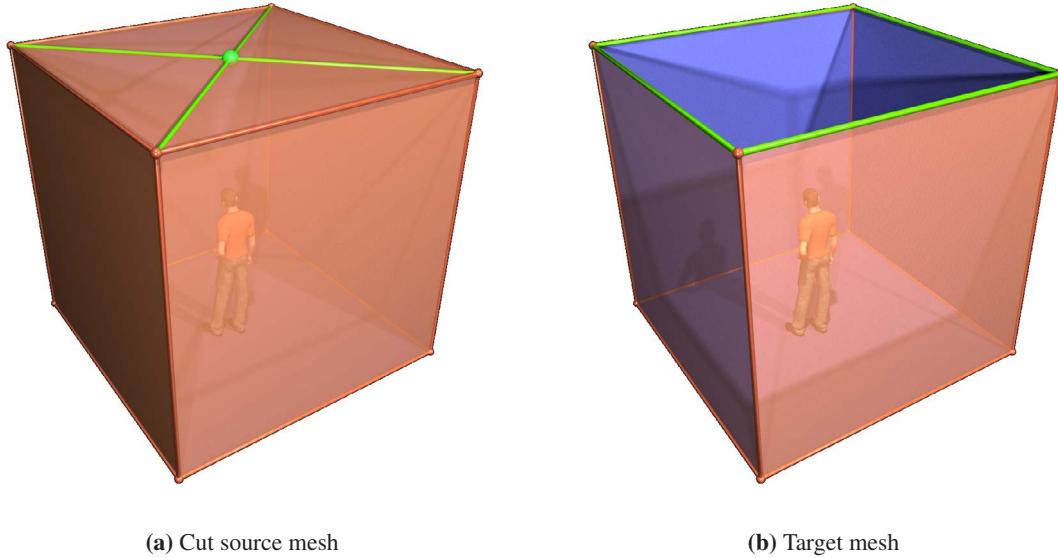


(c) 6-sided CAVE mapped to the sphere



(d) Conformal mapping of (c) to the unit disk

**Figure 6.3:** Algorithm for conformal mapping between a 5-sided CAVE and a 6-sided CAVE.



**Figure 6.4:** Template meshes for a 5-sided CAVE.

### Mesh Processing

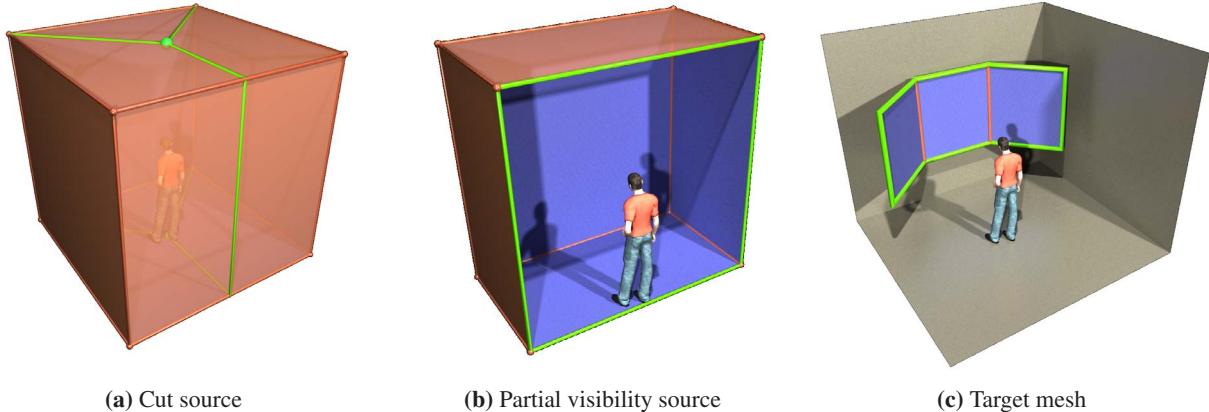
The source and target meshes are processed independently and in parallel to obtain the conformal mapping. The first step is to triangulate and refine the template to the desired granularity so that the computations can be performed with sufficient accuracy. We achieve this by an edge split operation where a new vertex is iteratively introduced at the mid-point of each edge, doubling the number of triangles. The process is performed until the edge length falls below a user-specified threshold. In our experiments, edge sizes corresponding to 20cm in the real-world allow for sufficiently accurate conformal maps to be generated at interactive speeds, while granularity below 5cm yields only marginal improvements. In addition to the performance constraints, the edge threshold also depends on the resolution of target displays.

At this point, each mesh is finely tessellated and contains a single closed boundary. Next, we remove a triangle so that the resulting mesh can be mapped to the complex plane (see Section 6.1.1). We select the triangle that is at the center of the mesh, or farthest from the original boundary. The same triangle is removed from both the source and target meshes. We also store a 3-vertex correspondence between the meshes, which is used to align the planar projections of the conformal maps. These vertices are selected from among the common vertices on the original closed boundaries. We then map the cube geometry to a sphere by the *direction map*

$$p \rightarrow \frac{p - c}{|p - c|},$$

## 6. DISTRIBUTED IMMERSIVE VISUALIZATION

---



**Figure 6.5:** Template meshes for a 3 screen target. The cut in (a) is suitable for the 3-sided CAVE. For an arrangement of flat-panel displays (c), the cut in (b) reduces the distortion effects.

where  $p$  is a point on the cube and  $c$  is the center of the cube.

The following step computes the shortest path  $\tau$  along the edges of the mesh between the original boundary  $\gamma_1$  and the newly created boundary  $\gamma_2$  at the center of the mesh. Since the embedding in the complex plane is periodic, the exact shape of  $\tau$  is not important; however, for consistency, we enforce that the same cut is made on both the source and the target meshes.

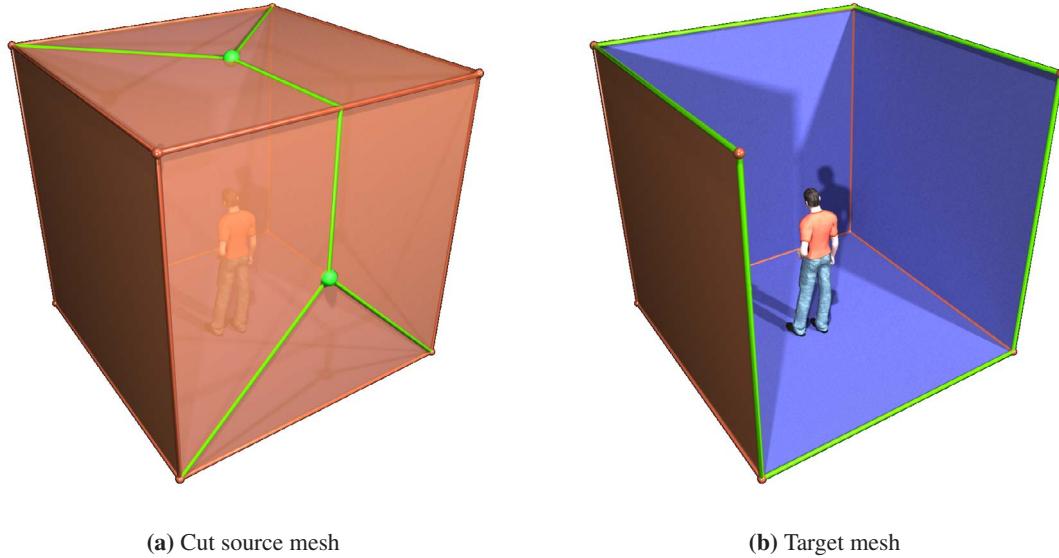
Next, the discrete Ricci flow algorithm presented in Section 6.1.1 is used to compute the conformal mapping to the unit disc. This mapping is stored as UV coordinates at the vertices of the two meshes. Figure 6.3 illustrates the results of this step. The Möbius transformation is then used to align the two conformal maps based on the vertex correspondence stored earlier. We use special Möbius transformations to map the point triplets to  $1, i$  and  $-1$  on the unit circle, which aligns the corresponding markers. Suppose  $\{p, q, r\}$  are three markers on the unit circle and

$$\eta_1(z) = \frac{z - p}{z - q} \times \frac{r - q}{r - p}$$

maps the markers to  $\{0, \infty, -1\}$ . Let

$$\eta_2(z) = \frac{1 + i}{2} \times \frac{z - 1}{z - i},$$

then  $\eta_2^{-1} \circ \eta_1$  is the desired Möbius transformation, which maps  $\{p, q, r\}$  to  $\{1, i, -1\}$ . Figure 6.3b and Figure 6.3d show the result after the alignment.



**Figure 6.6:** Template meshes for a 4-sided CAVE.

### Cubemap Generation

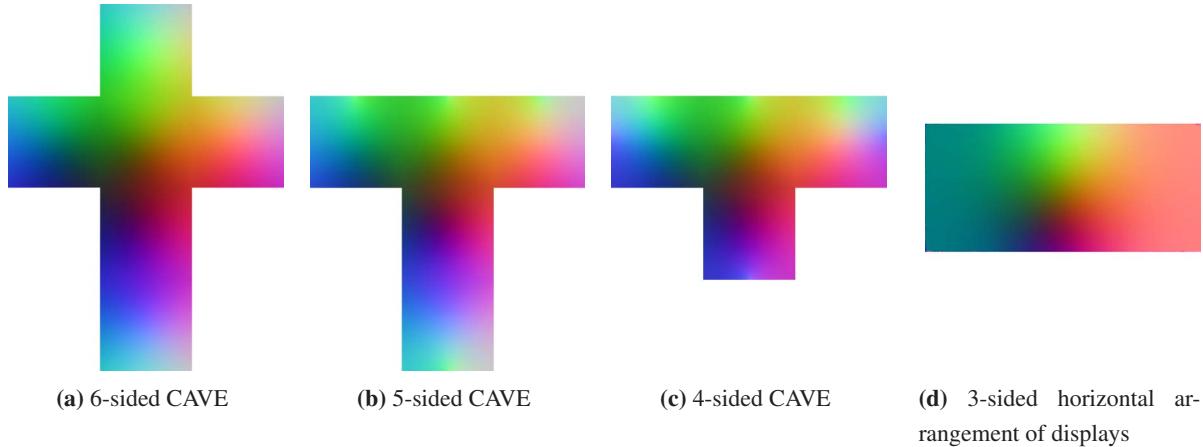
After the mesh processing, both the source and the target meshes contain aligned conformal mappings to the 2D complex plane stored in the UV channel at each vertex. All further processing is parallelized efficiently on the GPU using shaders developed in the NVIDIA Cg language.

In the next step, we utilize a vertex shader that computes the viewing direction through each vertex in the source mesh and stores the result in the vertex color attribute. The shader also flattens the mesh to the unit disc by copying the UV coordinates to the vertex position attribute. In the pixel shader we simply render the interpolated vertex colors to a high resolution floating-point RGB texture. This texture corresponds the circular map in Figure 6.3d where each pixel encodes a viewing direction. The reference point for computing these directions is specified by the user and is not necessarily the same point used to define the cut on the mesh template.

At the final stage, we map the circular texture onto the target mesh and use a pixel shader to render the view directions onto the faces of a cubemap. This cubemap is a discretization of a conformal transformation  $T_{ray}$  that maps the viewing directions of the target visual environment to the corresponding directions on the full visibility sphere. The cubemaps for the three visualization targets from Section 6.1.2 are shown in Figure 6.7. By reversing the source and the target meshes, we also compute the  $T_{mesh}$  transformation from the set of all viewing directions to the visibility defined by the boundaries of the target mesh. The direction vectors for both the circular maps and the cubemaps are stored in RGB textures with 32-bit per-

## 6. DISTRIBUTED IMMERSIVE VISUALIZATION

---



**Figure 6.7:** Cubemaps for the  $T_{ray}$  conformal transformation based on the visualization targets described in Section 6.1.2. The cubemap in figure (a) is the identity transformation.

channel precision. We set the resolutions to  $4096 \times 4096$  and  $1024 \times 1024$  respectively, although smaller textures can be used with minimal loss of directional accuracy if GPU resources are limited. However, if the rendering is performed on older GPU hardware that does not support bilinear filtering of 32-bit floating point textures, the largest supported resolution should be used with nearest-neighbor sampling. Alternatively, the precision can be reduced to 16 bits per channel, although the resulting loss of accuracy in the direction vectors may lead to objectionable rendering artifacts.

### 6.1.3 Visualization Techniques

We have implemented the system described in Section 6.1.2 as a library that can be integrated into existing visualization applications. The cubemap generator class contains a minimal interface for updating parameters such as edge threshold and reference cut points, as well as for notifying the main application when the cubemaps are updated. The  $T_{ray}$  and  $T_{mesh}$  transformations that we compute allow us to support both forward and backward rendering pipelines. We demonstrate applications to rasterization, single-pass raycasting for Direct Volume Rendering (DVR) and real-time raytracing on the GPU. We have also developed a dynamic visibility manipulation technique based on a reference position in a tracked VR environment.

#### Rasterization

We first apply the conformal visualization to a rasterization pipeline. An efficient vertex-based transformation is used for rendering well-tessellated geometry with OpenGL, similar to the approach by Spindler et al. [100]. Intuitively, every vertex in the scene is transformed so that triangles that are projected on the top

screen in the 6-sided CAVE configuration are instead projected on the 4 side screens in the 5-sided CAVE configuration. This transformation is defined by the following:

$$\begin{aligned} \mathbf{r}_c &= (M_{wc}^{-1})^T \cdot \text{norm} (\mathbf{p}_w - M_{wc}^{-1} \cdot [0, 0, 0, 1]^T) \\ \mathbf{p}_w &\rightarrow M_{wc}^{-1} \cdot (T_{geom}(\mathbf{r}_c) \cdot |M_{wc} \cdot \mathbf{p}_w|) \end{aligned} \quad (6.6)$$

where  $\mathbf{p}_w$  is the vertex position in world-space,  $\mathbf{r}_c$  is the normalized view direction in CAVE space and  $M_{wc}$  is the world-space to CAVE-space transformation matrix. In our rendering framework, the head node for the visualization cluster emits the camera information to all the rendering clients and the view matrix  $V$  associated with that camera is the world-space to CAVE-space transformation matrix. Each visualization node then computes the final view and projection matrices based on the target projection surface (e.g., front, left, etc.). The geometry transformation is performed in a custom vertex shader that is bound to every primitive in the scene.

The simple vertex shader approach suffers from a number of shortcomings. In particular, the requirement for well-tessellated meshes limits the applicability of our technique for many large real-world datasets. We take advantage of the new hardware tessellation features available in DirectX 11-compatible GPUs to implement adaptive tessellation in our rendering pipeline. Another issue is that the conformal map for the forward rendering pipeline is not continuous by design. In certain cases, scene elements spanning the template cuts (see for example Figure 6.4a) result in rendering artifacts that cannot be mitigated in the vertex shader. Instead, we move the computation of Equation 6.6 to the geometry shader, which is available on DirectX 10-compatible GPUs and operates on the mesh primitives instead of the individual vertices.

The tessellation stage of the hardware pipeline operates on a new patch primitive and in a pre-processing step, we convert all triangles in the scene to triangular patches. In the Tessellation Control (TC) shader, we compute the edge and inner tessellation levels based on the size of the screen-space projections of the patches and the edges after applying the conformal map to the patch vertices. In the Tessellation Evaluation (TE) shader, new vertex attributes are computed using barycentric interpolation. Finally, a geometry shader operates on each triangle produced by the tessellation stage. We apply Equation 6.6 to each vertex and detect triangles that span a discontinuity in the conformal map by a threshold on the area and the maximum edge length of the modified triangles. Detected triangles are simply not passed on to the rasterization stage. The fragment shader is not modified from our original implementation.

Compared to the vertex shader implementation, the improved image quality of using the adaptive tessellation incurs a more significant decrease in rendering performance on current generation GPUs. The relative speed reduction is less pronounced for 3D engines that already use GPU tessellation for effects such as displacement mapping or employ high-resolution meshes.

## 6. DISTRIBUTED IMMERSIVE VISUALIZATION

---

### Direct Volume Rendering

The conformal transformation is applied in a similar fashion to volume rendering. Our visualization algorithm is based on single-pass ray-casting over 3D textures with support for advanced lighting and shadowing, as well as pre-integrated transfer functions [44]. Our framework integrates volume rendering tightly into the scene graph and we render out the volume-space positions of the front and back faces of the volume bounding box, modified by the depth of other scene geometry. One possible approach for incorporating the conformal map is to tessellate the bounding box and apply the  $T_{geom}$  transformation as described in Section 6.1.3. However, our target application is the exploration of the virtual colonoscopy data, in which case the camera is often within the volume and the starting positions of the rays are defined on the near clipping plane. It is more accurate to transform the positions on the near clipping plane and the back face of the bounding volume to world-space and then to apply the following transformation:

$$\begin{aligned} \mathbf{r}_c &= (M_{wc}^{-1})^T \cdot \text{norm}(\mathbf{p}_w - M_{wc}^{-1} \cdot [0, 0, 0, 1]^T) \\ \mathbf{p}_w &\rightarrow M_{wc}^{-1} \cdot (T_{ray}(\mathbf{r}_c) \cdot |M_{wc} \cdot \mathbf{p}_w|) \end{aligned} \quad (6.7)$$

where, again,  $\mathbf{p}_w$  is the vertex position in world-space,  $\mathbf{r}_c$  is the normalized view direction in CAVE space and  $M_{wc}$  is the world-space to CAVE-space transformation matrix. The viewing vector is then constructed from the modified starting and ending positions. Note that this transformation is very similar to Equation 6.6. The difference is that since this is a backward rendering pipeline, we are transforming the view directions directly and the  $T_{ray}$  cubemap is used. Our framework varies the step size for ray integration based on the voxel projection size on the image plane, similar to the differential sampling approach introduced by Knoll et al. [63]. We extend the idea further by taking into account the distortion profile of the conformal map along the CAVE  $y$ -axis and lowering the sampling density in areas of larger compressive distortions. Elaborate acceleration structures for empty space skipping can further improve the rendering speed, however for the particular task of navigating the colon dataset, such techniques are not necessary since most of the time the user is close to the colon wall. Instead we employ early ray termination [79] to stop the ray integration after finding an iso-surface or accumulating density above a user-specified threshold. We also use stochastic jittering of the starting positions of the rays [93] to reduce sampling artifacts and a binary search for the iso-surface [96] to improve the reconstruction quality. During interaction with the system, we generally sample the volume data using a trilinear filter with a reduced step size in order to improve the response time. However, for a static camera position we render the images using higher order filtering on the GPU [45, 95] for both the density data and the on-the-fly gradient computations.

### GPU Raytracing

A major limitation of the geometry transformation approach is that Equation 6.6 needs to be computed for every vertex in the scene, including vertices generated by the tessellation shaders. As a result, the rendering performance scales poorly as the data sizes increase. Furthermore, while GPU adaptive tessellation and rejection of triangles spanning map discontinuities solves many of the rendering quality issues for non-uniformly tessellated meshes, those approaches affect the rendering performance and are more complex to implement. In contrast, the conformal visualization for a backward rendering pipeline transforms the viewing directions as opposed to the scene geometry itself. As a result, mesh tessellation is not an issue, the conformal map does not contain discontinuities and the scene shaders do not need to be modified.

A number of GPU-based ray-tracing algorithms have been proposed that can achieve interactive frame-rates even for large geometric models [70, 86, 133]. We integrate a ray-tracing renderer with our scene graph based on the NVIDIA OptiX engine. OptiX accelerates ray-tracing on the GPU by defining ray-generation, ray-scene intersections and shading programs in the CUDA language which access traditional acceleration structures that are also stored on the GPU. The programs are then intelligently scheduled on all CUDA-enabled GPUs in the system. Its features are comparable to other real-time ray-tracing approaches [86, 133]. For our conformal visualization, the ray transformation is applied at the ray-generation level, which is separate from the scene-graph and therefore much simpler to re-implement. The computation is also simplified:

$$\mathbf{d}_w \rightarrow M_{wc}^T \cdot (T_{ray} ((M_{wc}^{-1})^T \cdot \mathbf{d}_w)) . \quad (6.8)$$

Similarly to the volume rendering approach, we transform the world-space ray direction  $\mathbf{d}_w$  to CAVE-space and fetch a new ray direction from the conformal map. CUDA (and by extension OptiX) does not currently support cube textures natively, therefore indexing the conformal map is implemented in the ray-generation program. We also augment the conformal map generation stage of our pipeline to copy the pixel data to a 2D texture in the vertical cross format.

### Dynamic Visibility Manipulation

Our conformal visualization pipeline generates the ray transformations  $T_{ray}$  and  $T_{geom}$  based on a reference point inside the target mesh. In a tracked VR environment, the reference point is generally the head position in order to allow for more accurate visibility retargeting as the user moves through the environment. For example, as the user approaches a corner of the 5-sided CAVE, the visual information from the missing top projection would be presented mainly onto the 3 remaining visible surfaces. However, recomputing the mesh cut and the conformal mapping is an expensive operation that approaches interactive performance

## 6. DISTRIBUTED IMMERSIVE VISUALIZATION

---

only for coarse tessellations of the source and target templates. In contrast, the GPU processing can be performed very efficiently even for large texture sizes on mid-range GPUs, allowing for realtime changes to the position of the reference point.

In our application, we decouple the head-tracking for rendering and computing the mesh templates from the tracking of the reference point for the viewing rays computations. This provides a novel user interface for controlling the conformal transformations  $T_{ray}$  and  $T_{geom}$  without recomputing the conformal maps. For a given conformal parameterization of the source and target meshes, moving the reference point changes the effective frustum of viewing rays that is mapped to each target display surface. As a result, zooming into the visual data is achieved by moving the reference point away from the given display surface. However, the interface presented to the user is actually the inverse as it is more natural to move a tracked device toward and area of interest. The conformal mapping established between the source and the target meshes ensures that all viewing directions are mapped onto the existing display surfaces, thereby maintaining the visual context around the focus point.

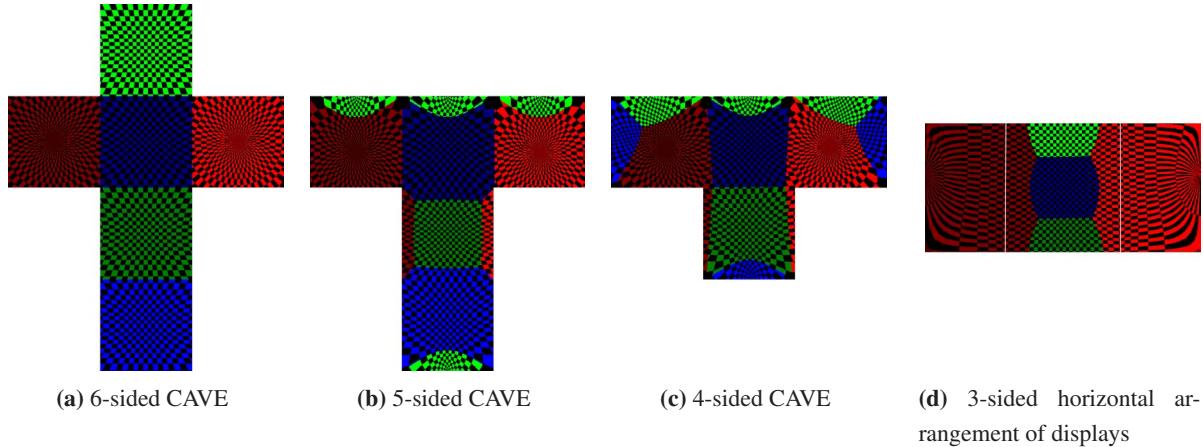
### 6.1.4 Results

#### Visualizing the Conformal Transformation

We first demonstrate the visual properties of our technique on a unit sphere with a checkerboard texture. In Figure 6.8, the virtual camera is positioned at the center of the sphere so that the two poles of the texture are shown on the left and the right displays. Figure 6.8b shows the results of our conformal visualization and demonstrates how the visual information originally shown on the top screen is instead rendered on the left, front, right and back screens. In Figure 6.8c we apply the  $T_{ray}$  transformation computed for the 4-sided CAVE mesh target (see Figure 6.6b). The result is that compared to the 5-sided case, the visual information from the back screen is pushed toward the bottom and the side screens.

We also visualize the conformal transformation for the scenario presented in Figure 6.5 where the 180° field of view is mapped to an arrangement of workstation monitors. The monitor setup is modeled after the Samsung MD230X6 array of  $3 \times 2$  high resolution thin bezel monitors. This represents a challenge for the conformal mapping since with an aspect ratio of 2.66 : 1, the boundaries of the source and target mesh are significantly different. Nevertheless, our approach produces an angle-preserving conformal transformation that enhances the field of view of the user.

User interactions with the virtual scenes often involve navigation in 3D space. We demonstrate this aspect of our visualization technique with a checkerboard tunnel. Figure 6.9 illustrates the camera panning down during the navigation with the 5-sided CAVE rendering target. Note that compared to the straightforward front projection, the conformal rendering preserves the context of the tunnel's direction even when the



**Figure 6.8:** Raytracing of a checkerboard sphere with conformal visualization on different display targets. The layout for (a)-(c) is in the standard vertical cross format and the color coding is defined for the original walls in the 6-sided CAVE configuration (blue for front/back, red for left/right, green for top/bottom). For (d), we show the output of all 3 displays side by side.

camera is pointed downward as in Figure 6.9d. The visual information corresponding to the original front projection is presented with minimal distortion, as also shown in Figure 6.8b.

## Performance

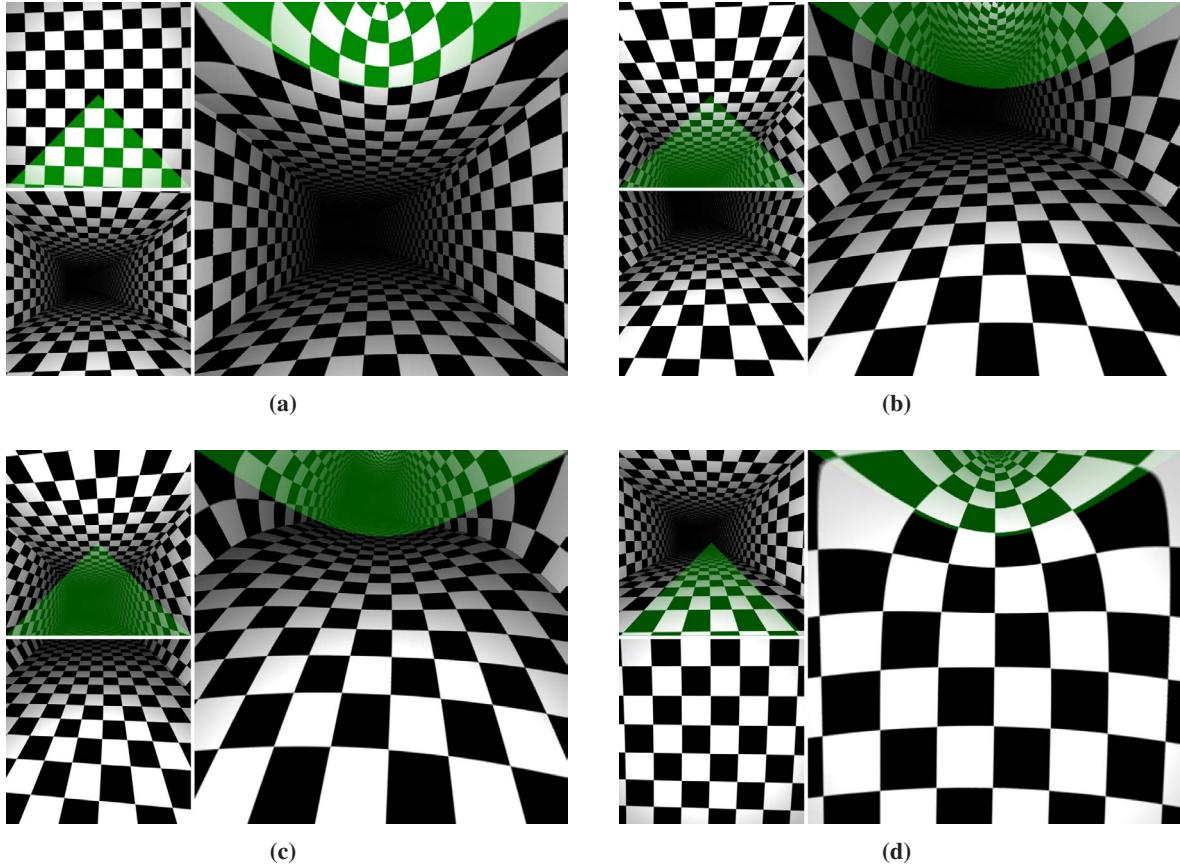
The pipeline described in Section 6.1.2 is designed to execute most efficiently on a system with a modern multi-core CPU and at least one mid- to high-end GPU. The mesh processing for the source and the target templates is performed in parallel and we also utilize the PARDISO sparse linear solver from the Intel Math Kernel Library during the computation of the discrete Ricci flow. We measured the performance of a variety of systems with 2, 4 and 8 CPU cores and found that quad-core CPUs strike the best balance between cost and performance. Although dual quad-core CPUs show an improvement with the PARDISO solver, the overall computation time is dominated by the mesh processing. In our current implementation, this is for the most part a serial operation that does not benefit from more than 2 CPU cores.

We also compare the performance of the GPU stage of the cubemap generation pipeline for a variety of hardware. Since our shaders are relatively simple, the only GPU requirement is the support for bilinear 32-bit floating point texture filtering and the appropriate texture sizes for the target display resolution. The oldest GPU we have tested is the NVIDIA Quadro 2500M, which can produce  $4096 \times 4096$  textures for the circular map and a  $1024 \times 1024$  cubemap texture at up to 20 iterations per second.

Our primary testing machine contains a single Intel Xeon E5620 which is a quad-core CPU running at 2.4GHz and a single NVIDIA Geforce GTX 480. Figure 6.10 summarizes the performance for a range of edge length thresholds that are reasonable for real-world applications. The threshold values are defined

## 6. DISTRIBUTED IMMERSIVE VISUALIZATION

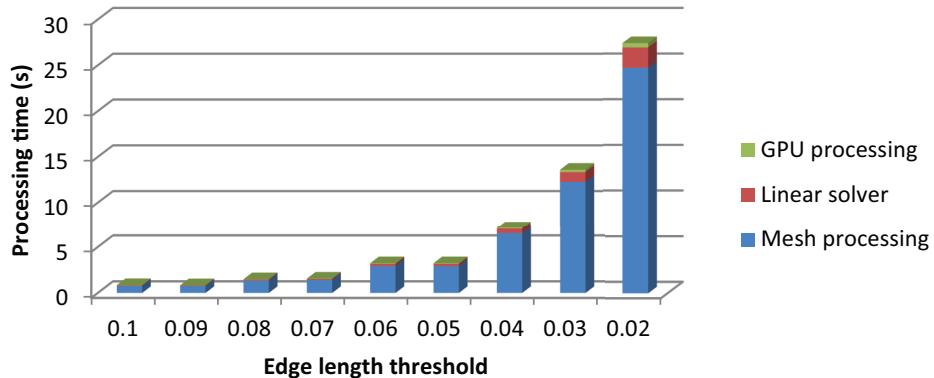
---



**Figure 6.9:** Navigation in the checkerboard tunnel with conformal visualization where the camera pans down in (a)-(d). Each triplet of images shows the original front view (lower-left), original top view (upper-left) and front view with conformal visualization (right).

for the normalized vertex positions of the source and target meshes. At the 0.1 threshold, our conformal generation executes up to 2 iterations per second, allowing for interactive updates of the conformal maps. We expect that the most current CPU architecture running at higher clock speeds will double the performance of our current implementation. For higher quality images with a static conformal map, the 0.05 thresholds produces sufficient conformal map resolution for a  $1400 \times 1050$  rendering target.

The GPU stage as used for the dynamic visibility manipulation can currently execute at minimum 60 iterations per second, although this is not indicative of real-world performance. As described in Section 6.1.3, the GPU raytracing requires that the cube texture be copied to a 2D format, which is an expensive operation that disrupts the GPU pipeline. The time for the copying operation is included in Figure 6.10. The performance may increase significantly when using the latest version of OptiX, which supports cubemap textures natively. In practice, the time for generating the conformal maps on the GPU is negligible for both volume rendering and ray-tracing.



**Figure 6.10:** Performance of the cubemap generation running on a single quad-core Intel Xeon E5620.

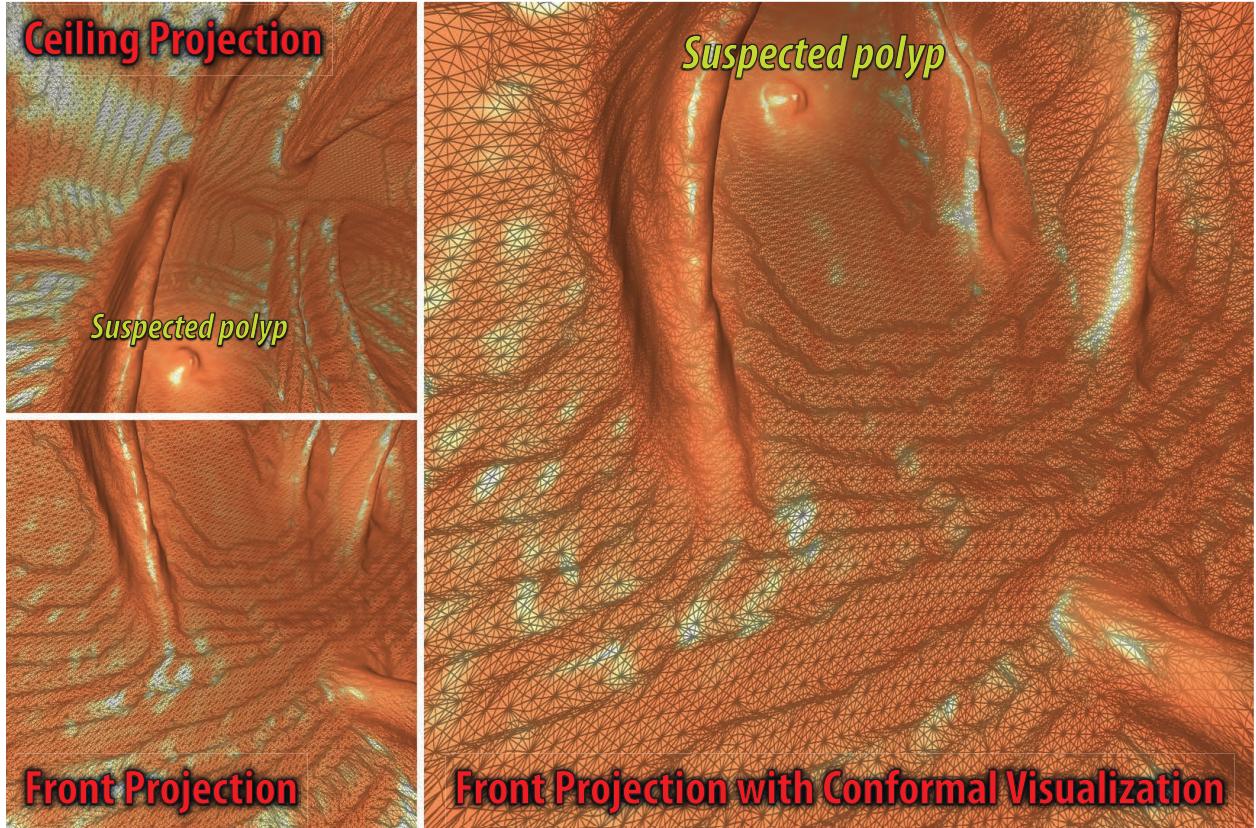
We first evaluate the geometry-based conformal distortion on a surface model from a virtual colonoscopy dataset. The triangular mesh is uniformly well-tessellated and as a result, transforming the individual vertices as described in Section 6.1.3 yields acceptable rendering results. Figure 6.11 demonstrates the conformal distortion near a known polyp. The two images on the left illustrate the original front and top projections in the CAVE. Since our visualization platform is not fully enclosed, the polyp would not be visible. With conformal visualization, the polyp is displayed on the front screen in the CAVE and its shape is preserved under the conformal mapping. Other sections of the missing viewing space would similarly be transformed to the side and the back screens. The mesh contains 4M vertices and the rendering performance decreases by approximately 1% when the distortion is applied on modern GPUs with unified shader units (e.g., NVIDIA GeForce 8 series and above). The performance hit is significantly larger on older hardware with dedicated vertex shaders (e.g., NVIDIA GeForce 7 series and below). Also, such GPUs have very limited texturing capability in the vertex processing pipeline, and as a result vertex shaders may even be emulated on the CPU, leading to non-interactive frame rates.

## Applications

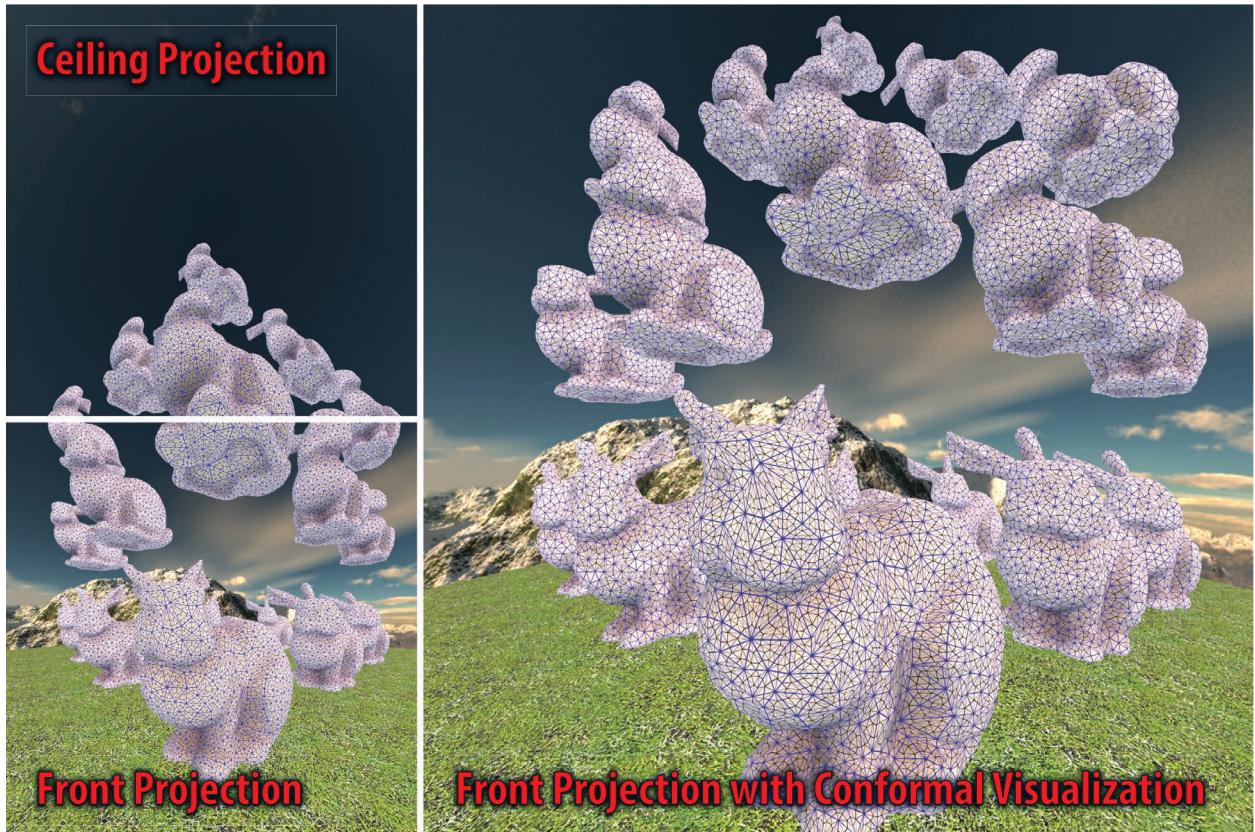
We first evaluate the geometry-based conformal visualization technique on a course mesh scene. Our rendering system uses OpenGL rasterization and we have augmented the material system with the shaders described in Section 6.1.3. The other significant change is that triangle meshes in the scene are converted to the new OpenGL patch primitive with 3 control points per patch that match the original vertices of the triangles. The edge tessellation factors are computed based on a user specified edge length threshold so that silhouette edges in the scene appear smooth with the conformal visualization. The test scene used to generate the results in Figure 6.12 is composed of 19 instances of the Stanford Bunny dataset where each instance contains 2,500 triangles. Compared to the traditional pinhole projection camera, the conformal

## 6. DISTRIBUTED IMMERSIVE VISUALIZATION

---



**Figure 6.11:** Visualizing the mesh model of a patient's colon during Virtual Colonoscopy. The left pairs of images show the front and top views, and the image on the right shows the front view after conformal distortion. The shape of the polyp is preserved under the conformal geometry deformation.



**Figure 6.12:** Conformal visualization applied to the rasterization pipeline. The images show the original front view (lower-left), the original top view (upper-left) and the front view with conformal visualization (right) for the same camera position. Triangular patches are drawn in blue, tessellated triangles are in black. Our technique produces artifact-free results even for coarse scene geometries.

visualization significantly expands the user’s field of view and allows the visual context of the scene to be maintained throughout the navigation. The angle-preserving property of the conformal maps ensures that the overall shape of the meshes is recognizable even through geometries directly above the user are scaled down, as illustrated in Figure 6.8b.

The performance of the conformal visualization scales almost linearly with the number of triangles in the scene after the tessellation. Depending on the scene complexity, its initial tessellation, the edge length threshold and the position of the user in the scene, we have observed reductions in the framerate of up to 50%. For the example in Figure 6.12, the observed performance was reduced by approximately 15% after enabling the conformal visualization.

Our system integrates full support for the rendering of volumetric data with particular focus on medical visualization such as IVC. The pair of images on the left in Figure 6.13a are the original perspective projections for the front and top screens in a fully-enclosed CAVE. The image on the right is the result of the

## 6. DISTRIBUTED IMMERSIVE VISUALIZATION

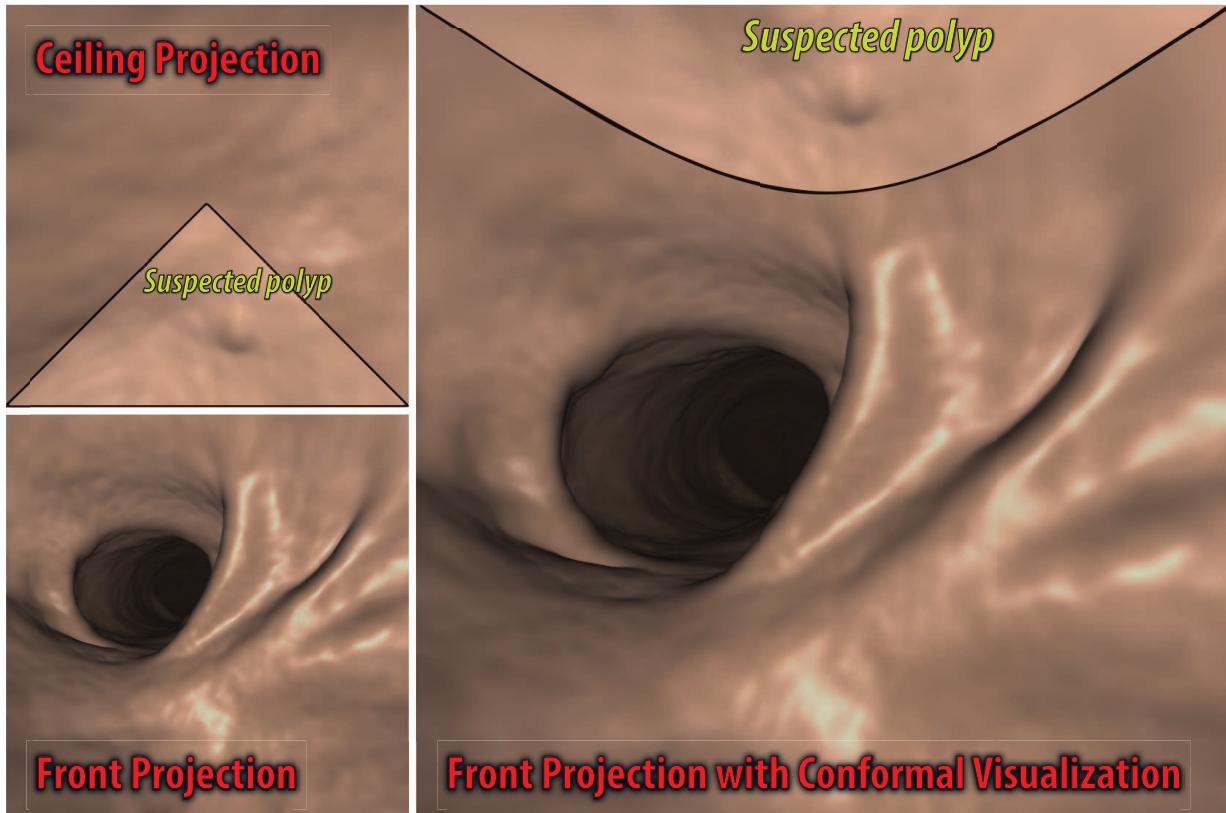
---

conformal visualization described in Section 6.1.3 for the front screen in the 5-sided CAVE. Note that unlike in image-based retargeting approaches, the final result is rendered directly, and the original images are included here only for demonstration purposes. The source of the volume data is a  $512 \times 512 \times 451$  16-bit CT scan of a patient’s abdomen after digital cleansing of the colon. A suspicious area is visible on the top screen, which would normally be missed in the CAVE due to the angle of the approach and the lack of a top screen. After the conformal transformation of the viewing rays, this area is projected onto the front CAVE screen and the shape of the polyp on the colon wall is preserved. Although conformal mapping is not distance-preserving and sizes are scaled, we provide tools to measure the actual distance in voxel-space. Since the conformal distortion map is computed in a pre-processing step, the cost of the initial transform of the ray direction is negligible compared to the ray integration. In practice, the performance drop is measurable but it is less than 1% on average.

Interactive ray-tracing is supported in our visualization software as a full replacement for the OpenGL renderer. As described in Section 6.1.3, the application of the conformal map is simplified in terms of both the computation and the integration with the existing scenes. The main benefit is that the rendering performance scales with the size of the viewport as opposed to the scene complexity. The ray-tracing approach also makes no assumptions regarding the tessellation of the data and provides high-quality conformal visualization results even in areas of low polygonal density.

Figure 6.14 compares the projection images of the front and top screens (left pairs of images) to the conformally distorted front view. As before, the original images are only used for illustrative purposes and the final result is rendered directly. The performance of the renderer scales with the size of the viewport as opposed to the scene complexity in the rasterization pipeline. Although our colon mesh contains geometry with uniformly-high triangle density, the same is not true for the architectural scene. The ray-tracing approach makes no assumptions regarding the tessellation of the data and provides high-quality conformal visualization results even in areas of low polygonal density. While the performance for the architectural scene is lower than with the rasterization pipeline, it allows for interactive frame-rates and special effects such as accurate glass reflections and shadows. Again, the cost of applying the transformation map is negligible compared to the overall rendering and its effect on the frame-rate is below 1%.

The conformal visualization preserves shapes locally, however scenes with expansive straight edges present certain challenges. In particular, straight edges projected onto the missing ceiling surface are bent toward the top center of the front, side and back screens. The Ricci flow allows for direct control of the boundary during the mapping with the result that the distortion is minimized for areas of the scenes that were projected on the original 5 surfaces. Figure 6.14 illustrates this - the lower railings and the column are visually similar to the original projections onto the front screen. The goal of conformal visualization in this application is to present additional visual context for the navigation while preserving the relationships



(a) Virtual Colonoscopy navigation with conformal visualization



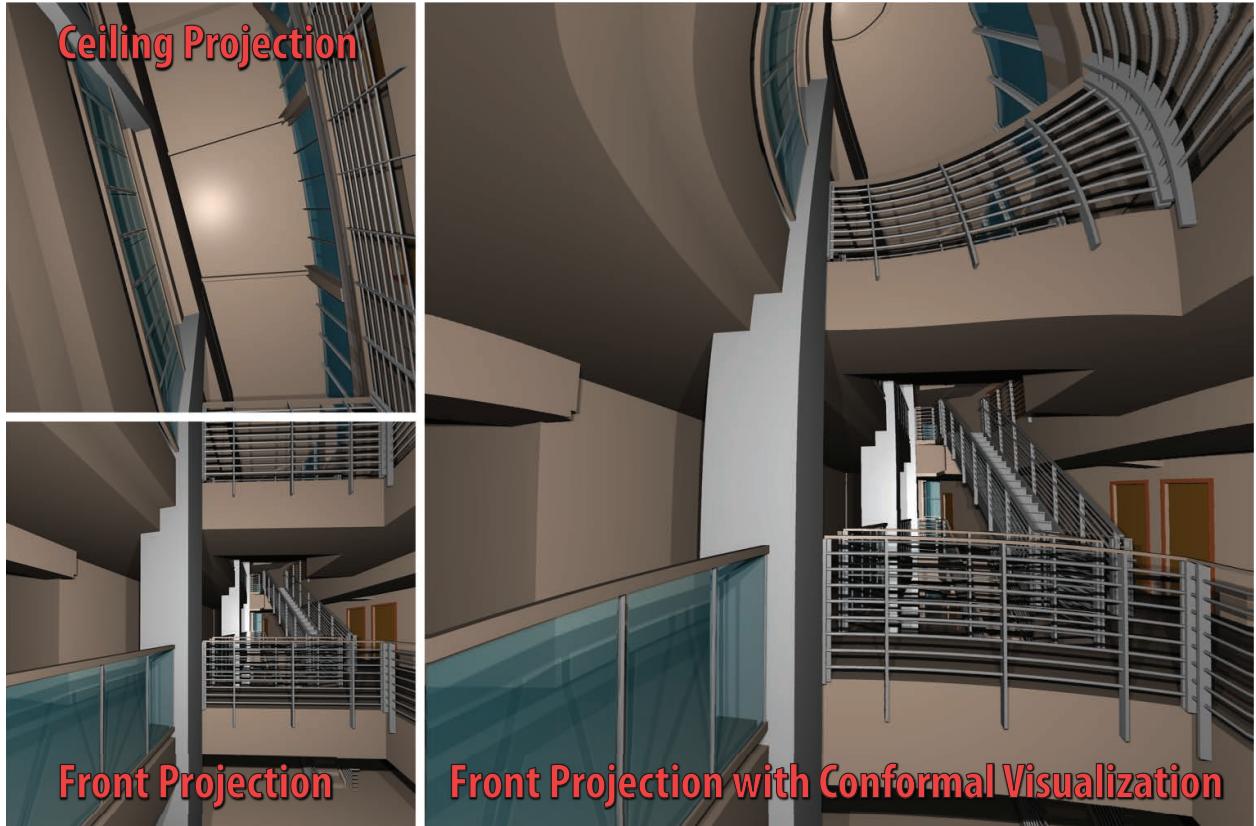
(b) 5-sided CAVE without conformal visualization

(c) 5-sided CAVE with conformal visualization

**Figure 6.13:** (a) Snapshot of Immersive Virtual Colonoscopy in the 5-sided CAVE. The shape of the polyp on the top view is preserved under the conformal visualization. Compared to the traditional volume rendering (b), the conformal visualization allows for a more thorough analysis of the dataset (c) and the entire surface of the colon is visible as the user negotiates a bend in the colon.

## **6. DISTRIBUTED IMMERSIVE VISUALIZATION**

---



**Figure 6.14:** Snapshot of an architectural fly-through using conformal visualization expanded visual context during the navigation.

## 6.1 Conformal Visualization

---

between local structures. We provide a simple control over the pre-generated conformal maps that allows the user to specify how much of the ceiling projection is visualized onto the screens. The resulting choice is between visual distortions and visual coverage, and the selection can be optimized for the particular application or dataset. This control is implemented in the ray-generation kernel and can be modified at runtime without re-computing the conformal map or the cube texture. For illustrative purposes, Figure 6.14 uses the full conformal map.

Our GPU raytracing pipeline is used in visualization experiments that involve large connected data structures. The conformal visualization is particularly suitable for scenes containing a significant number of elements when the exact shape of the connections is not relevant to understanding the data, but the shape of the elements may be. The expanded field of view and the resulting visual context are particularly beneficial for exploration tasks in information visualization applications. Figure 6.16 presents our test scene, which contains an abstract binary tree with 10,000 data elements and color-coded connections. Note that with conformal visualization for the 5-sided CAVE the missing visual information from the top screen is displayed near the 4 top edges of the CAVE. We further simulate a 4-sided CAVE by disabling the rendering to the back wall and applying the transformation depicted in Figure 6.8c. As the checkerboard rendering shows, the missing information from the back wall is presented primarily on the back edges of the side screens, while the top is distributed over the front and side screens. In both cases, the visual transformation is angle-preserving and the nodes in the graph retain their shape.

We also render the scene with the mesh template described in Figure 6.5c. Compared to the pinhole camera rendering in Figure 6.15a, the conformal visualization in Figure 6.15b increases the field of view while preserving the local shapes of the objects in the scene. Our current mesh template utilizes only a hemisphere of the user's original field of view, however, that can be increased by augmenting the tiled array with additional screens on the top and the bottom. The resulting boundary of the screen geometry would then more closely resemble the original visibility boundary, which in turn would provide more uniform conformal visualization results.

One notable feature of the graph dataset presented in Figure 6.16 and Figure 6.15 is that it contains a large number of dense node clusters, which are difficult to explore using the traditional navigation paradigms. In Figure 6.17 we demonstrate our dynamic visibility manipulation technique for a stationary position of the user in both the physical space of the 5-sided CAVE and the virtual space of the 3D scene. The user manipulates the reference point used to generate the GPU textures for conformal visualization as described in Section 6.1.3 in order to focus on the cluster at the back wall (Figure 6.17b) and the distant clusters displayed on the front wall (Figure 6.17c) while maintaining the visual context. In both cases, the reference point was moved to the extreme positions on the back and the front walls respectively. Figure 6.17a shows the default conformal visualization result where the reference point matches the user's viewing position. As

## 6. DISTRIBUTED IMMERSIVE VISUALIZATION

---

discussed in Section 6.1.4, the visibility manipulation is implemented entirely on the GPU, and as a result, the zooming interface appears very smooth to the user.

Finally, we construct the transformation maps for use in the Reality Deck. Compared to the Immersive Cabin, both the physical space and the resolution are increased significantly, and both are floor and the ceiling are missing as display surfaces. To accommodate the increased resolution, we have scaled the circular map to  $8192 \times 8192$  and the cubemap textures to  $4096 \times 4096$ . These are in fact the largest maps we can handle on current GPUs due to the large memory requirements when generating 32-bit floating point data. The threshold for the source and target mesh refinement is set to 0.01, and due to the increased computation cost, the final cubemaps are pre-generated and stored as OpenEXR image files. We adapt the target mesh template from Figure 6.5c to the rectangular fully-enclosed layout of the Reality Deck by extending the screens toward the back-right corner. The source layout is similar to the one for the 5-sided Immersive Cabin (see Figure 6.4a) with additional cuts on the floor and across the back-right vertical edge.

In Figure 6.27, we present the results of using our technique with the protein visualization in the Reality Deck (that application is described in Section 5.3.3). We use OpenGL to render the mesh representations of the molecular surfaces and the meshes are of sufficient quality so that the GPU-based tessellation is not needed. With the traditional projection in Figure 6.18a, the user misses significant portions of the data during the interactive exploration because of the absent display surfaces. The problem is particularly pronounced in the Reality Deck, where the floor and the ceiling are in fact the largest flat surfaces. In contrast, the conformal visualization in Figure 6.18b expands the field of view while preserving the shapes locally. Similarly to the graph data, molecular structures remain recognizable with only changes to the size of the elements near the top and bottom edges of the display wall.

### 6.1.5 Evaluation

Our conformal visualization technique is evaluated informally in the 5-sided CAVE for the visual detection of polyps in phantom colon datasets. The evaluation focuses on the core retargeting properties of the technique, and therefore we use static mesh templates for a user position at the center of the CAVE without the dynamic visibility manipulation. In particular, we are interested in how the additional visual information affects the users' navigation experience and their accuracy in identifying the simulated polyps. The phantom data are generated based on sections of the colon centerline from a real patient who had undergone Virtual Colonoscopy and each volumetric scene contains between 25 and 29 synthetic hemispherical polyps.

There are 4 datasets similar to the one in Figure 6.19 with between 25 and 29 hemispherical polyps each. A cylinder is extruded along a section of the centerline and a large number of hemispheres are placed along the inside wall of the cylinder at arbitrary locations. The user control scheme in the CAVE uses an analog gamepad with the following mapping: view control with the left analog nub, forward/backward

## 6.1 Conformal Visualization

---

motion with the right analog numb, camera roll with the shoulder pads. The users are given time to study the navigation scheme for both regular and conformal visualization, after which they are asked to find as many of the simulated polyps as possible. The initial camera roll is selected randomly at the beginning of each experiment and the visualization modality order is pre-selected for each user. The examiner records the number of detected polyps, the number of false positives and the time for each fly-through.

Our informal study involved 12 participants and on average, the examination time was consistent between the regular and conformal visualization. The conformal visualization improved the non-biased detection sensitivity for polyps from 91% to 93% overall with individual differences in the range  $[-2\%, +10\%]$ . On average this translates to the detection of up to 0.9 additional polyps over the baseline per datasets with 29 polyps. The false positive rate for both visualization modalities was on average 0.1%. During each evaluation, the examiner monitored the visualization on a separate computer and marked polyps that were projected only onto the missing ceiling surface. If we isolate only those polyps from our results, the detection rates were virtually identical to the average for the datasets examined with conformal visualization, compared to 0% for the regular volume rendering.

In our follow-up questionnaire, the users indicated that stereoscopic vision remained effective in all sections of the display surfaces, including the areas at the interfaces with the missing display surface where the visual information is compressed. They also developed natural strategies for utilizing the additional visual information with minimal input from the examiner (the users were told during the short training session that the visualization at the top edge was generated from data directly above them). Almost all the users indicated that the additional information helped them navigate bends in the colon that point upward. A significant number of users developed the habit of rotating the virtual camera in order to examine polyps they had spotted in the compressed areas of the visualization. We received similar comments from a professional radiologist who informally reviewed our system with real colon data. Our preliminary experiments indicate that the dynamic visibility manipulation can reduce the need for camera rotation by allowing the user to temporarily zoom into area of interest while navigating the dataset.

Although we saw slight improvements in the detection rates on average, the results were more dramatic if we consider polyps that would only appear on the missing ceiling. In a large number of such cases, the user failed to detect the polyp under the regular rendering pipeline. Furthermore, there was general disorientation at times during the navigation when the colon bend pointed towards the ceiling. This effect was most pronounced for people who rated their familiarity with VR at 'low' and 'none', and who had little experience with interactive games. On the other hand, under conformal visualization, the detection rates for ceiling polyps were close to the averages for the entire datasets with below 0.3 false positives per navigation run.

## **6. DISTRIBUTED IMMERSIVE VISUALIZATION**

---

Another interesting observation is that many users naturally developed strategies for utilizing the additional visual information from the conformal visualization, even with little input from the examiner. One strategy involved using the peripheral vision to monitor the top of the front and side displays for polyps, but to point the virtual camera so as to examine the suspicious area without the distortion. Many users also learned to use the conformal visualization for navigation, especially when faced with an upward colon bend. This generally allowed for smoother navigation. The findings were similar to comments from a professional radiologist who informally reviewed our system with real colon data. We only performed a small-scale casual study for the architectural application. A 50% to 60% threshold for ceiling contribution seemed to work well and the users developed similar navigation and panning techniques to the VC study.



(a) No retargeting

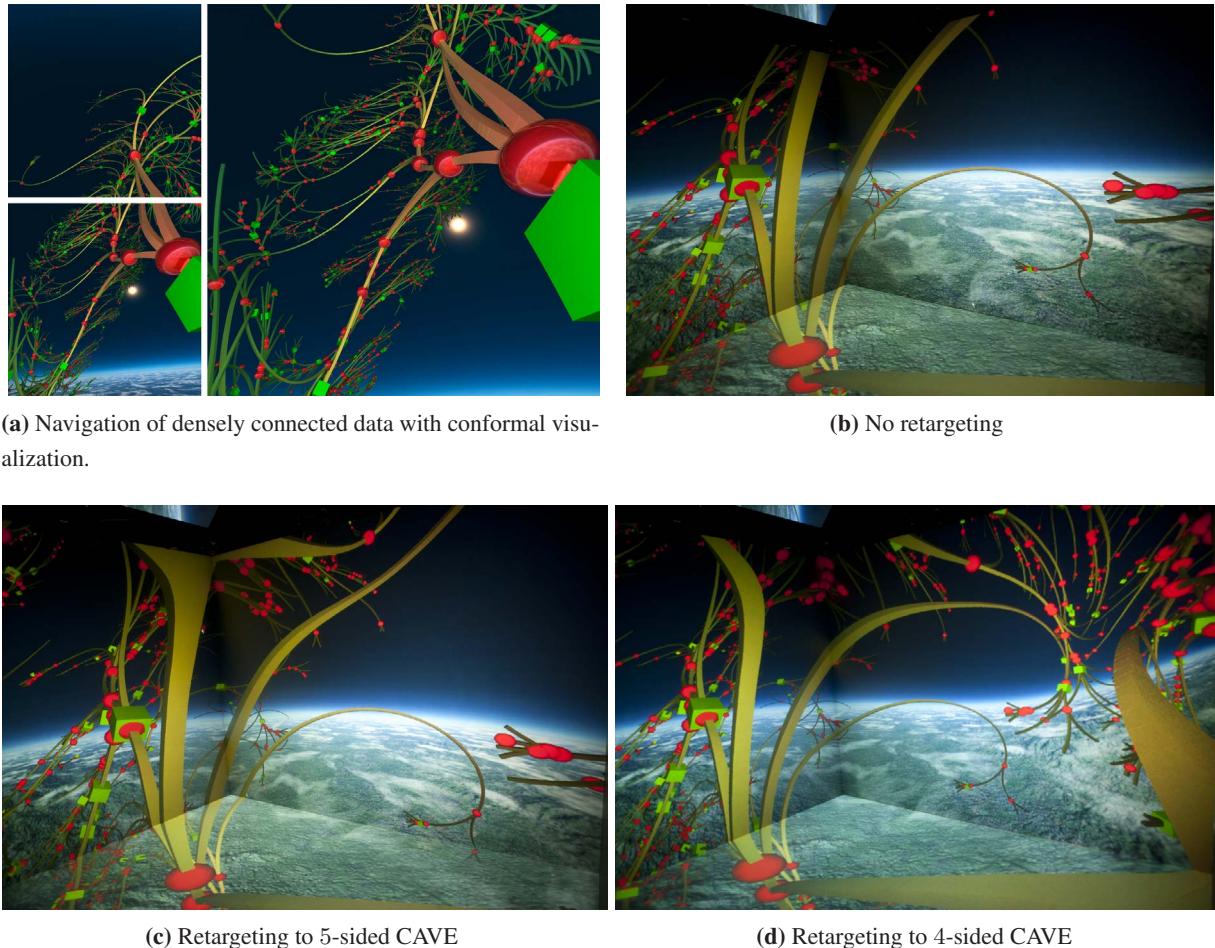


(b) With conformal visualization

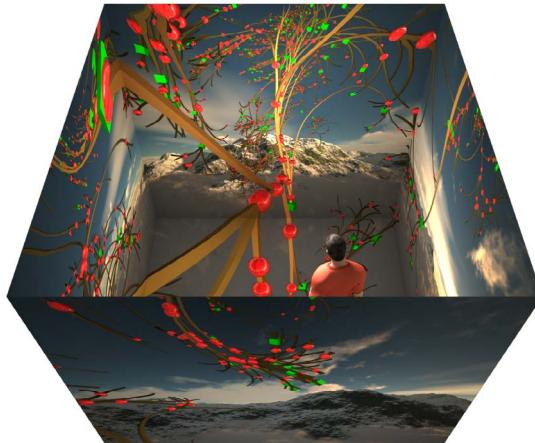
**Figure 6.15:** Conformal visualization results for a large tiled display. (a) The standard pinhole camera model with wide FOV leads to significant distortions near the periphery of the image. (b) In contrast, conformal visualization allows for 180° horizontal and vertical FOV while locally preserving the shapes in the data.

## 6. DISTRIBUTED IMMERSIVE VISUALIZATION

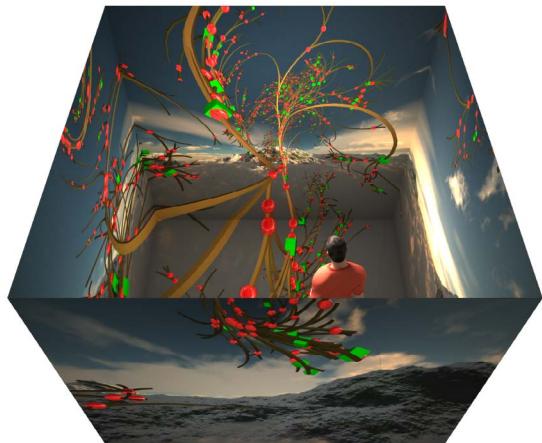
---



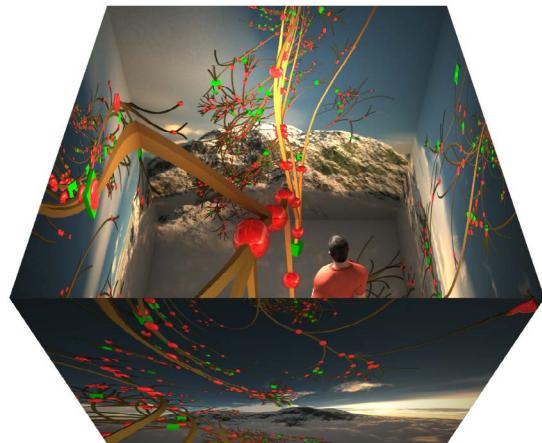
**Figure 6.16:** Conformal visualization results. (a) Each triplet of images shows screen captures of the original front view (lower-left), the original top view (upper-left) and the front view with conformal visualization (right). The photographs (b)-(d) illustrate views from inside the CAVE where the front wall is on the left side of the image. Missing visual information from (c) the top wall and (d) the top and back walls is presented on the available display surfaces. The 4-sided CAVE is simulated by disabling the back wall.



(a) Reference point at the center of the CAVE



(b) Reference point at the back wall

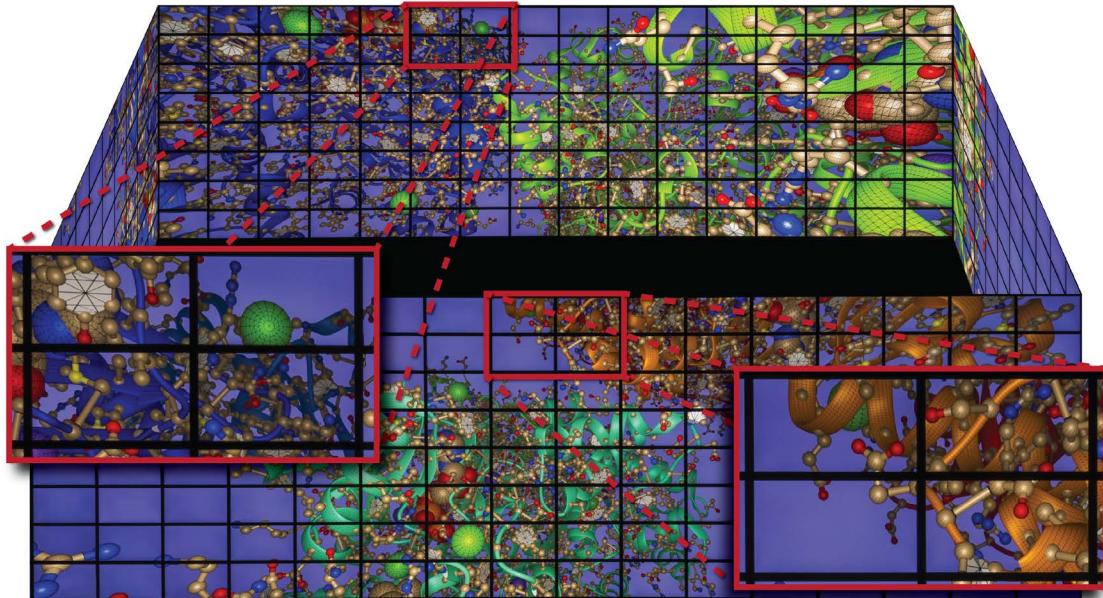


(c) Reference point at the front wall

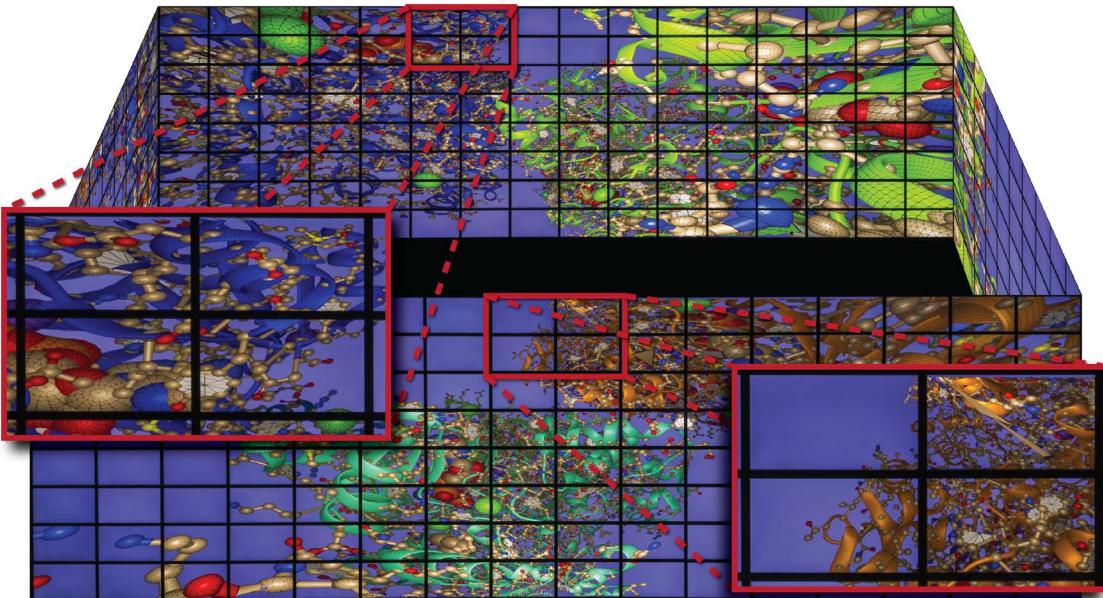
**Figure 6.17:** Conformal visualization results for the GPU raytracing pipeline with dynamic visibility manipulation. The standard conformal parameterization for the 5-sided CAVE is used, while the reference point for computing the GPU textures moves between the extreme points at the center of (b) the back wall and (c) the front wall. The recomputation executes at realtime speeds and allows for a context-preserving zoom operation (e.g., toward the dense cluster of nodes at the front wall).

## 6. DISTRIBUTED IMMERSIVE VISUALIZATION

---

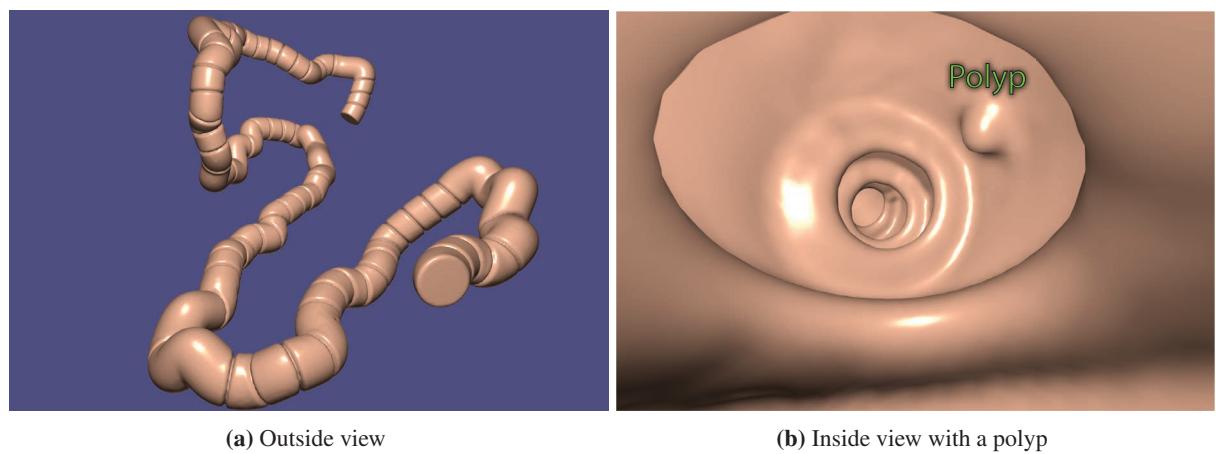


(a) Normal OpenGL rendering



(b) OpenGL conformal visualization

**Figure 6.18:** High resolution stitched photographs from the four walls in the Reality Deck while running protein visualization. (a) shows normal rendering with OpenGL, (b) uses our conformal visualization technique to expand the field of view while preserving the shapes locally.



**Figure 6.19:** Phantom dataset for the user study. The center-line for the generated data is based on a section of a patient's VC.

## **6. DISTRIBUTED IMMERSIVE VISUALIZATION**

---

### **6.2 Frameless Visualization**

We have developed a novel visualization system based on the reconstruction of high resolution and high framerate images from a multi-tiered stream of samples that are rendered framelessly. This decoupling of the rendering system from the display system is particularly suitable when dealing with very high resolution displays or expensive rendering algorithms, where the latency of generating complete frames may be prohibitively high for interactive applications. In contrast to the traditional frameless rendering techniques, we generate the lowest latency samples on the optimal sampling lattice in the 3D domain. This approach avoids many of the artifacts associated with existing sample caching and reprojection methods during interaction that may not be acceptable in many visualization applications. Advanced visualization effects are generated remotely and streamed into the reconstruction system using tiered samples with varying latencies and quality levels. We demonstrate the use of our visualization system for the exploration of volumetric data at stable guaranteed frame rates on high resolution displays, including a 470 megapixel section of the Reality Deck.

Modern advances in LCD technology have led to the proliferation of affordable high density displays in a variety of applications. While 4MP and even higher resolution display have been available on the market, the high initial cost has limited the use of such devices. On the other hand, 3MP displays have recently appeared on affordable tablet computers and it is expected that similar high resolution displays will be available on other computing platforms as well. Another contributing factor is the growth of the GPU technology. We now have single devices that can drive up to six displays, or up to 24MP resolutions, which significantly reduces the costs associated with building large tiled displays. However, implementing interactive visualization and graphics systems for such high resolution displays remains a significant challenge.

A major obstacle for visualizing large amounts of data on very high resolution displays is the traditional notion of *frames*. A large majority of applications use some type of a computation device, such as the GPU, that produces a set of pixels over a regular 2D grid, which is then sent to the display device using a double buffering scheme. As the resolution requirements increase, the latency associated with the image generation increases as well, which affects the usability of the system. Frameless rendering has been proposed as an alternative approach [11], where the computations of the pixels are decoupled from the display system. As rendered samples are streamed in, the system updates some internal data structure that can then be used to approximate the full resolution images at very high frame rates. Although many of the derived approaches address the problem of temporal coherency, objectionable visual artifacts can still occur during interaction with the system, which may not be acceptable depending on the visualization application.

Our frameless visualization system introduces the notion of *lattice samples* which are used to augment the frameless *adaptive samples*. Slices of lattice samples are generated with low latency, using for examples

a second GPU in the system, and stored in an optimal sampling grid. The reconstruction process then samples this grid together with the streaming adaptive samples to generate the final output. We demonstrate that using the BCC grid for the spatial and temporal upsampling increases the image quality for high resolution visualization compared to using the same number of lattice samples on the CC grid, or the same number of adaptive samples. Our system is particularly suitable for use with distributed ray-based image generation methods and we provide examples for the frameless visualization of medical volume data and the interactive exploration of iso-surfaces from a combustion simulation. The following are the specific contributions of our work:

- A novel visualization technique that augments the traditional frameless rendering with optimal sampling lattices for rapid image reconstruction at very high resolutions and with reduced visual artifacts.
- Unified handling of streaming image samples generated with different latencies and at different quality levels in the same sampling domain.
- Demonstration of our approach for the visualization of large volume data at stable interactive frame rates on a 470MP tiled display, driven by a hybrid GPU cluster. This includes both direct volume rendering and raytracing of isosurfaces.

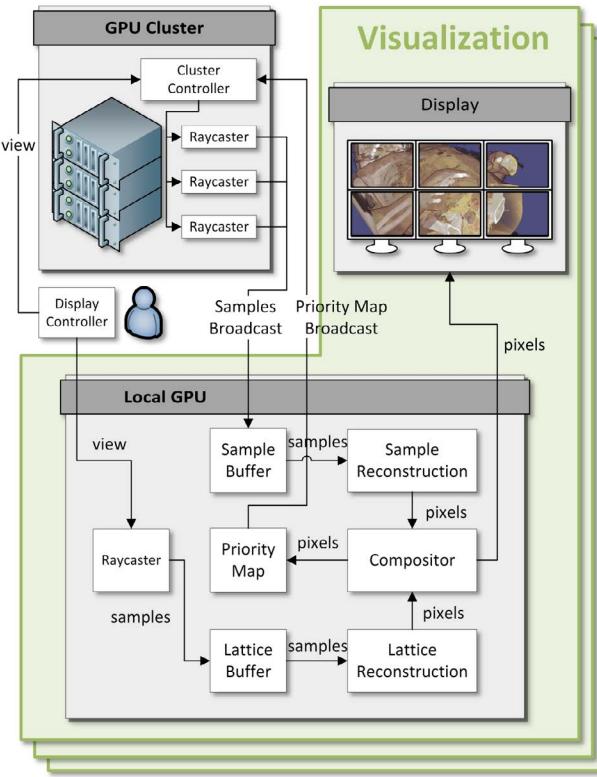
### 6.2.1 Algorithm Overview

Our system is designed to decouple the volume rendering from the display (see Figure 6.20). The main building block is a distributed single-pass ray-casting volume renderer running on a GPU cluster that can generate image samples both on regular grids and for stochastic ray directions. At the level of a single local GPU attached to a grid of displays, raycasting is used to generate color samples on a lattice where each slice is lower resolution than the target display. At the same time, lattice reconstruction is run asynchronously to generate a low resolution temporally upsampled preview image. This image is also used during the creation of the priority map that guides the rendering of unstructured samples on dedicated cluster nodes without attached displays. Depending on the configuration, the lattice samples can be generated remotely (e.g. a secondary GPU, or a dedicated node) as long as the latency of the transfer can be kept low.

The second component of the system is responsible for generating a stream of rendered samples, which are combined with the locally generated lattice samples to progressively create the final full resolution composite. For clarity, the illustration in Figure 6.20 uses a GPU cluster as the sample source, however, depending on the display configuration, the source can be an auxiliary GPU in the same system or even a

## 6. DISTRIBUTED IMMERSIVE VISUALIZATION

---



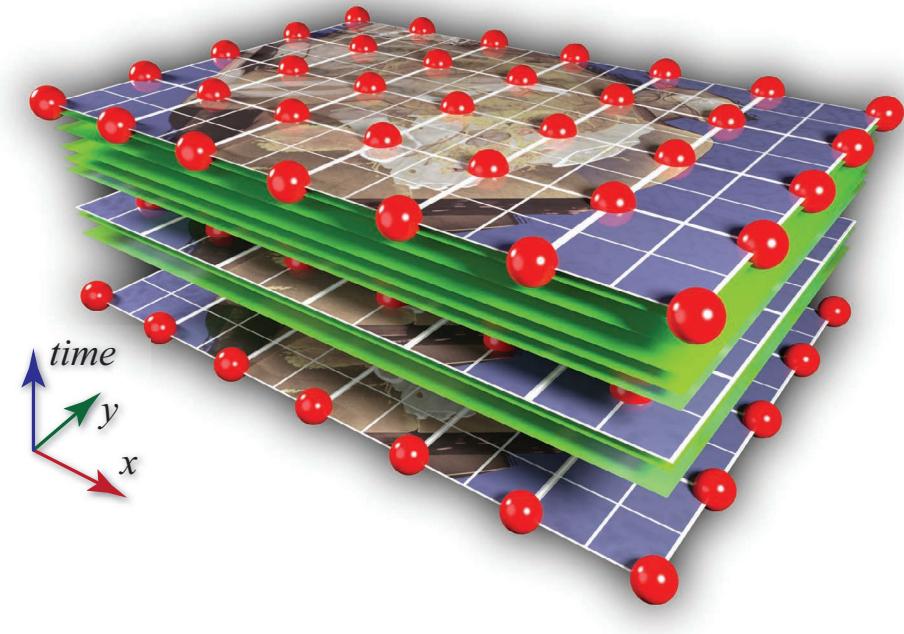
**Figure 6.20:** The rendering pipeline for our frameless visualization technique.

cloud-based service. Visualization parameters such as viewing direction and transfer functions are synchronized across all renderers. The resulting sample stream is broadcast across the network and each visualization node stores samples that belong to the local viewport in a buffer that matches the full resolution of the attached display.

The compositing step takes into account a number of attributes associated with each sample during image reconstruction. The most important attribute is the age as it determines which samples can be used during interactive exploration of the data. Depending on the proximity of the rendering source to the local GPU, the latency can vary widely. The lattice samples are generated locally and directly into the lattice buffer on the GPU, and can therefore be used during scene changes. On the other hand, samples generated remotely can have very high latencies depending on the network configuration and the rendering parameters, and can be used primarily to improve the resolution and image quality of static scenes.

### 6.2.2 Adaptive Sampling

In the traditional frameless rendering approach, samples are streamed at arbitrary times and spatial and temporal filters are used to reconstruct the actual images to be displayed. This reconstruction executes



**Figure 6.21:** The volume rendering is performed onto 2D slices of the BCC lattice (lattice sites in red). During reconstruction, full images can be produced at arbitrary time steps (green layers).

at fixed time intervals and ideally matches the refresh rate of the output display device. Our frameless visualization system incorporates a fixed number of samples that are arranged on a regular grid, and more specifically, on the optimal sampling lattice in 3D (BCC). The properties of the BCC lattice are presented in Section 2.1. The 2D slices reconstructed from that grid are used to guide both the generation of the adaptive streaming samples and subsequently the reconstruction of the final images. Figure 6.21 illustrates the image data stored as layers of the BCC data, with arbitrary intermediate frames (time steps) generated through 3D lattice reconstruction (shown in green).

Each image generator in the system receives lattice samples at regular intervals and stores them in a 3D texture on the GPU. These samples are generated using low-latency rendering techniques and may even be produced locally and not streamed over a network connection. The image generator uses spatial and temporal upsampling through the lattice reconstruction filter (see Section 2.2) to generate a low resolution approximation to the final result. At the same time, the most current image result is used to generate a local priority map, which is broadcast to the sample generators (see Figure 6.20). This priority map assigns weights to different sections of the display based on the color and luminosity gradients, as well as sensor data, such as position and orientation of the user from a head tracker.

In comparison to existing high-quality frameless rendering approaches [25], which build acceleration

## 6. DISTRIBUTED IMMERSIVE VISUALIZATION

---

structures on the CPU, we use a hierarchical stack of sample caches. The highest resolution cache is viewport-sized and the lowest resolution is approximately  $2\times$  the resolution of the lattice cache. Our system is designed to support frameless visualization applications on very high resolution display devices where tens of millions of sample points may be required. We store the following attributes for each sample (note that for the lattice samples, only the color attribute is needed):

**Position:** 2D floating point position of the sample within the global viewport.

**Color:** The color of the new sample is blended with the existing sample based on the age and the priority.

**Sample Age:** Used for blending with the existing samples. For static images, samples are accumulated. For dynamic images, samples are weighted according to the age during blending.

**Sample Priority:** Determined by the quality of the rendering. High quality samples override lower quality ones.

Our system keeps a synchronized clock across all sample and image generators in the cluster. We first filter out incoming samples based on their position within the local viewport of the image generator and the current timecode. If the timer is running and the sample is too old, that is, the sample time is not within a given factor of the times covered by the lattice samples, it is discarded. We stop the timer when the user interface is idle and we want to accumulate samples for antialiasing. Otherwise, the sample is blended with the current value in the cache. The blending value is scaled by the timecode of the old sample so that its contribution is 0 if the timecode is no longer covered by the lattice samples. Also, if a new sample quality value is greater than the value in the cache, the old sample is replaced. Once the cache is updated, the sample attributes are copied to GPU memory. Each sample is accumulated into all the levels of the hierarchy, allowing for rapid reconstruction of low resolution approximations in sparsely sampled areas.

We designed our CPU sample cache with the goals of maximizing the number of samples available to the reconstruction shader and minimizing the number of samples that need to be streamed to the GPU. The latter is particularly important since the GPU is not optimized for transferring small amounts of information over its external bus. As new samples are stored in the cache, we update a set of buffers on the GPU asynchronously to the main rendering loop. These buffers are then copied to the sample cache texture on the GPU before rendering. We start by reconstructing the color from the lattice data. We then adaptively determine the appropriate filter size by the resolution of the lattice grid and a k-nearest neighbor search in the sample cache when the cache hierarchy uses a single level. With a stack of caches at decreasing resolution, the algorithm traverses the hierarchy to find the level at which the cache has any accumulated samples, which is significantly faster. During the summation step, the weights for the samples from the cache and the computed lattice color are weighted by the age and the priority attributes to produce the final pixel value.

### 6.2.3 Implementation Details

Our frameless visualization system is developed on top of our distributed rendering framework, which supports a number of ray-based visualization techniques for both volumetric and mesh data. The system is currently implemented entirely within the rendering pipeline of the GPU, although a GPGPU implementation in NVIDIA CUDA or OpenCL may improve the rendering performance. One of the most critical aspects of our system is the efficient streaming of the sample attributes to the GPU. We have implemented an asynchronous architecture where a single CPU thread manages the OpenGL context and all the rendering commands; two threads manage the processing and the streaming of the lattice samples and the adaptive samples. The volume data processing and packing is implemented using OpenMP and may spawn additional threads based on the available resources. Finally, the network management is performed in a separate thread as well.

The lattice data is continuously processed, re-packed and uploaded to the GPU using two Pixel Buffer Objects (PBOs) as the slices are received from the networking thread. On modern OpenGL implementations, access to the GPU memory uses DMA and the thread can start processing the next slice as soon as the memory copy command is issued. Once the data is uploaded to GPU memory, the rendering thread copies it to a texture so that it can be sampled in a Cg shader. We immediately switch to the second PBO so that the processing thread can start the next upload.

The streaming of the adaptive samples uses a set of 2D textures on the image generator per cache hierarchy level to store the samples and their attributes. The sample processing is backed by a CPU queue and a collection of PBOs. If the sample generators are providing structured samples (e.g., small tiles or scanlines), we can optimize for that by batching the memory copy operations. However, in the general case, we copy the updated samples individually from the queue to the PBO as soon as possible. Before each render operation, the samples are copied from the PBOs to the textures.

One limitation of our system is that the memory transfer is currently not overlapped with the OpenGL rendering. High-end NVIDIA Quadro GPUs with dual Copy Engines support such concurrent execution, however that requires multiple OpenGL contexts and complex synchronization. Our preliminary results show that the streaming performance on the Quadro is measurably better even if the Copy Engines are not used directly. Alternatively, the rendering pipeline can be adapted to use the CUDA or OpenCL languages, where concurrent memory operations and kernel execution are possible even on the consumer-grade GPUs. New technologies such as faster buses, unified memory spaces and CPU/GPU hybrids can lead to increased performance in the future as well.

The lattice reconstruction filter requires that at least one slice of lattice samples should be available in advance before the reconstruction can begin. In the shader, we estimate the time until the next slice is available based on the previous elapsed times and move the sampling position across the  $z$ -axis accordingly.

## 6. DISTRIBUTED IMMERSIVE VISUALIZATION

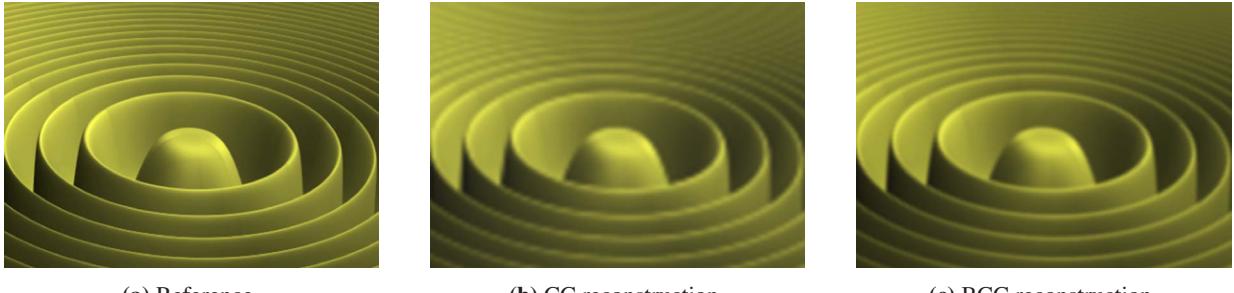
---

In practice, this works well and underestimation errors generally lead to a very small pause. However, the more serious consequence is on the latency of the system. Some of it is hidden by the latency associated with streaming samples from external sample generators and over a network. We also minimize the latency as much as possible by generating the lattice samples locally, for example using a secondary GPU and performing the memory transfer directly between the GPUs. In our experience, this latency, which is also present in some related reprojection methods, is acceptable for interactive visualization, but may not be acceptable for general realtime graphics applications.

### 6.2.4 Results

We first evaluate the sample streaming performance of our system. With the PBO-based implementation described in Sec. 6.2.3, the image generator can receive up to 50  $1024 \times 1024$  slices of lattice samples with four 32bit color channels per second. This configuration is used when the lattice samples are generated on a remote machine or a secondary GPU. However, for very high resolution displays, we can render the lattice samples directly into the lattice cache locally on the image generator. This is in fact the configuration we used in our experiments with the 470MP section of the Reality Deck. The performance when lattice and adaptive samples are used together is evaluated for a single 1.5MP viewport with a single level of the sample cache hierarchy, running on one node of the Reality Deck. The image reconstruction pipeline is running at 60Hz and processes 20 lattice slices and 2M individual samples per second. Note that while the lattice samples contain only the color attribute, all four attributes are streamed for the adaptive samples. The number of slices per second and the number of adaptive samples can be balanced depending on the requirements of the visualization application.

Our lattice-based rendering framework is used for both the frameless visualization and for volume rendering. We evaluate the lattice sampling component of our system using the BCC and the CC grids. We employ a GPU-based interactive raytracer to render the analytic Marschner-Lobb (ML) function [75]. In each case, an identical number of image samples are used per slice. The reconstruction starts after two slices have been stored in the cache and two new slices have arrived. For the following comparison, we implemented a cubic B-spline filter for CC data, which can be evaluated efficiently when trilinear interpolation is available on the GPU [95]. In this approach, the offset of a sampling point from a lattice site is used to compute 8 offsets and weights, which allow for the evaluation of the cubic B-spline with only 8 trilinear texture fetches. The BCC reconstruction uses the quintic box spline, which is a higher-order filter and is implemented on the GPU using 32 nearest-neighbor texture fetches [42]. As illustrated in Figure 6.22, the rendering quality with the BCC lattice is improved compared to the CC lattice. Specifically, the BCC reconstruction avoids the aliasing artifacts that can be seen near the top of Figure 6.22b while using the same number of image samples. Although not shown here, the perceptual image quality is also higher compared



**Figure 6.22:** The analytic ML function (a) is rendered with raytracing using image-space lattice samples on (b) the CC grid and (c) the BCC grid with an equal number of samples.

**Table 6.1:** Relative performance of the different filters on the NVIDIA Geforce GTX 470, together with the cost in terms of Nearest Neighbor (NN) or trilinear (lin) texture fetches. The baseline is the performance of the cubic B-spline filter for CC data.

Reconstruction Filter	Relative Performance
CC linear (1 lin)	2.74
CC cubic (8 lin)	1.00
<b>BCC linear (4 NN)</b>	<b>1.29</b>
<b>BCC quintic (32 NN)</b>	<b>0.31</b>
<b>FCC interleaved (4 lin)</b>	<b>1.37</b>
<b>FCC linear (16 NN)</b>	<b>0.64</b>

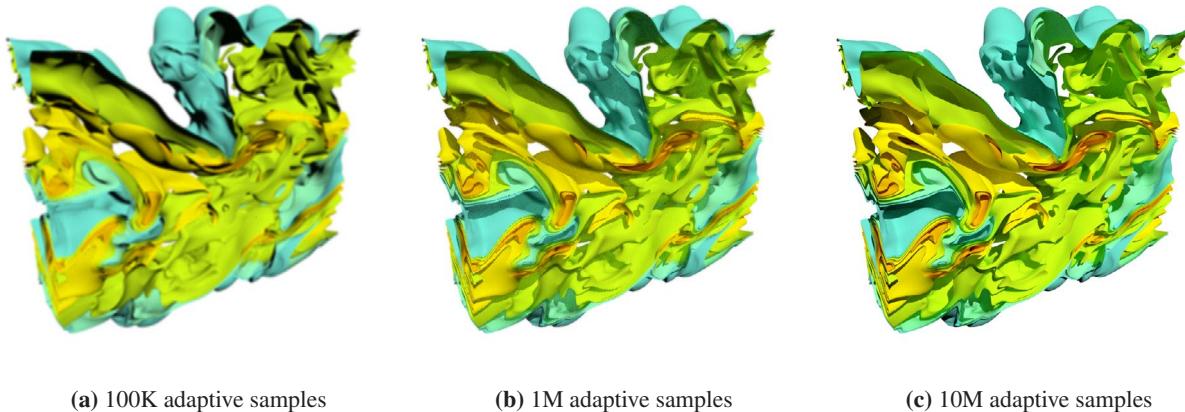
to the corresponding lower-order filters (trilinear for CC and the linear box spline for BCC), albeit at the expense of some overall smoothing.

We evaluate the performance of the different filters on datasets with comparable voxel counts. Our benchmark is run on the NVIDIA Geforce GTX 470. The reconstruction filters are implemented in the Cg language using version 3.0 of the NVIDIA Cg Toolkit. All shaders are compiled with the latest shader profiles supported by the hardware and the performance numbers are presented in Table 6.1. In our frameless visualization system, we perform a significantly smaller number of sampling operations compared to the ray casting in volume rendering, and even the slowest filter (quintic box spline on BCC) offers sufficient performance on modern GPUs.

Next, we employ our system for the visualization of complex iso-surfaces from a time-varying combustion simulation obtained through the SciDAC Institute for Ultra-Scale Visualization (480 × 720 × 120 resolution, 122 time steps). It is difficult to distinguish the intricate spatial relationships in that data without complex lighting computations. The lattice samples are rendered with OpenGL and per-pixel shading, but without any advanced light transport computations. Although the image quality is not sufficient for a visual

## 6. DISTRIBUTED IMMERSIVE VISUALIZATION

---



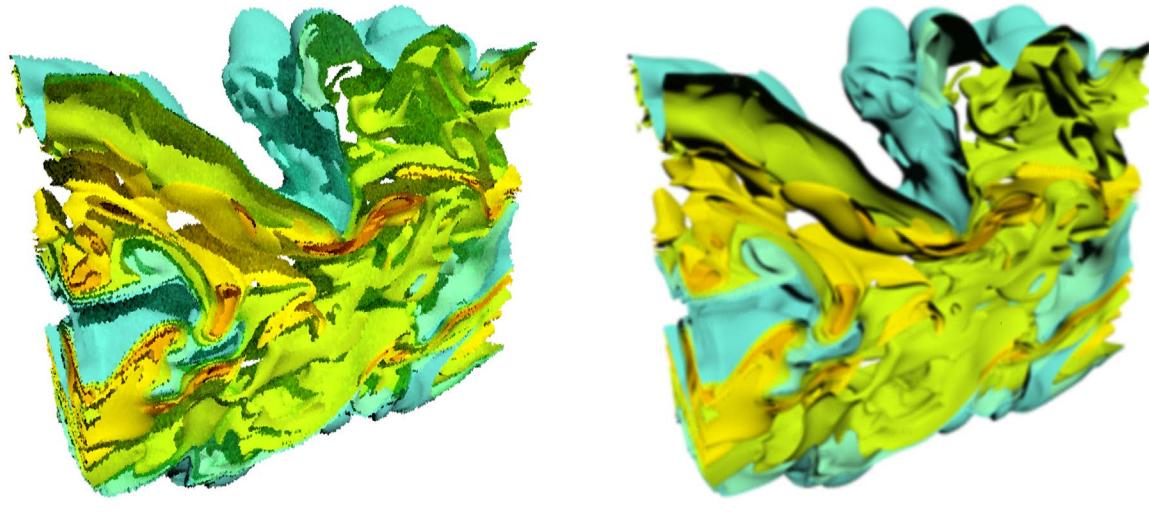
**Figure 6.23:** Image reconstruction over time with the frameless visualization system. The low resolution initial image (a) is produced very rapidly with GPU rasterization over the BCC lattice samples. With additional adaptive samples as in (b), the global illumination effects are clearly distinguishable and the reconstruction converges on the GI solution in (c).

analysis, this pipeline is very fast and we use it to render the slices of lattice samples directly on the GPU. For the adaptive samples, we use a more advanced GPU renderer that adds raytraced ambient occlusions to the image. The visualization of thin structures is improved significantly, especially in areas where multiple iso-surfaces are close together (e.g., lower left part of Figure 6.23c). Finally, the slowest GPU renderer generates samples with Global Illumination (GI) effects such as subtle color bleeding and soft shadows to improve the understanding of spatial relationships in the data.

Figure 6.23 illustrates the output of our frameless visualization system when using samples generated with all three methods. During user interaction and scene changes, the lowest latency samples are used to generate a low resolution representation of the data. As adaptive samples are continually streamed in, the resolution is improved and later converges to the output of the highest quality renderer. For static images, any additional samples are accumulated for supersampling. In our experiments, the result in 6.23b is obtained in under 1s and 6.23c is obtained in less than 8s.

In Figure 6.24 we focus on the initial images produced with our frameless visualization. We compare the cases of not using any lattice samples (Figure 6.24a) and running the full lattice-based pipeline (Figure 6.24b). Both images are taken from the same point in the running timeline and use approximately the same total number of samples. In the second case, the image is free from distracting visual artifacts and for static cameras, both converge towards the GI solution at the same speed.

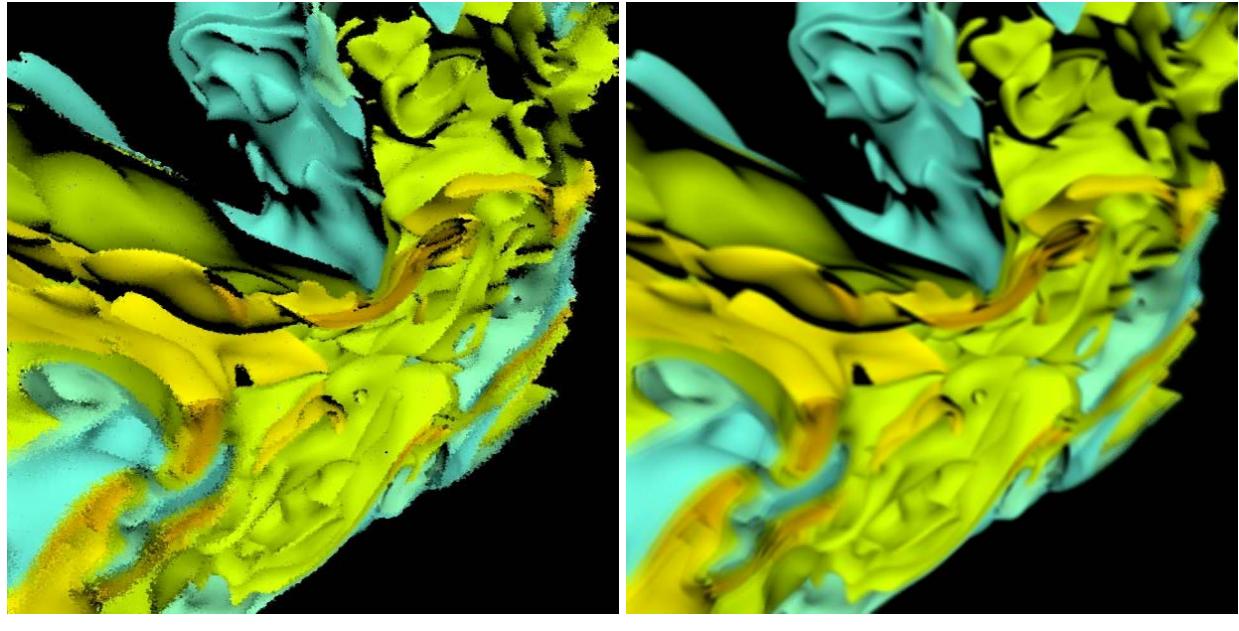
In dynamic scenarios where the user controls the camera, or scene elements change, the reconstruction is biased toward the temporally upsampled images from the lattice samples. We further restrict the allowable age of the samples so that adaptive samples are purged from the cache at the same time that slices are



(a) Frameless rendering

(b) Lattice reconstruction

**Figure 6.24:** During interactive visualization, initial low resolution images reconstructed from the lattice samples (b) are significantly less noisy than images produced with the traditional frameless rendering (a)



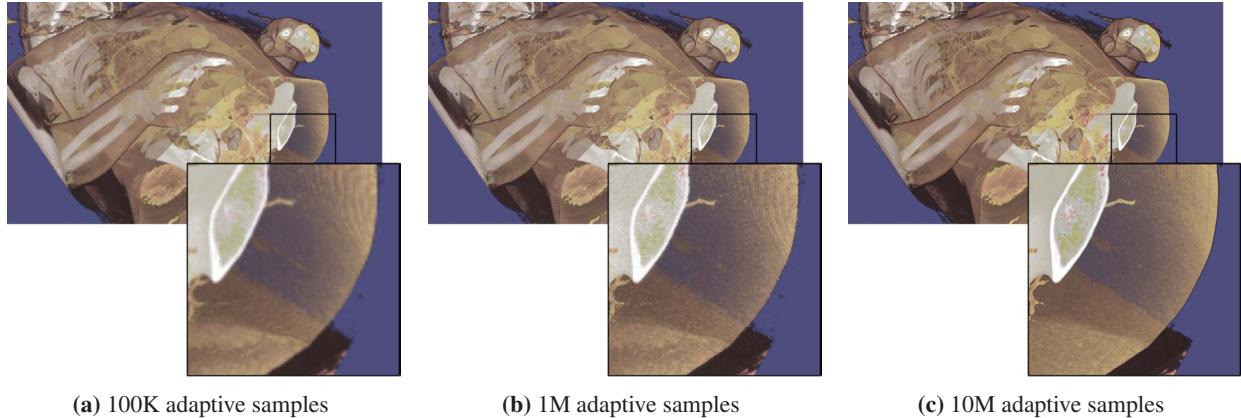
(a) Frameless rendering

(b) Lattice reconstruction

**Figure 6.25:** During dynamic scene changes, our system is biased toward the lattice reconstruction as shown in (b). The noise in the image is reduced compared to (a) the traditional frameless rendering.

## 6. DISTRIBUTED IMMERSIVE VISUALIZATION

---



**Figure 6.26:** Frameless visualization of the Visible Human dataset with Direct Volume Rendering on the GPU. (a) The initial image during interaction is generated from the lattice samples. (b) Adaptive samples are streamed to improve the resolution of the image. (c) A high-order filter and a smaller step size are used during the reconstruction of the volume data to generate high quality samples, which are streamed continuously at a slower rate.

replaced in the lattice cache. In Figure 6.25 we demonstrate the visual effect of rotating the camera when using adaptive samples alone and in combination with the lattice reconstruction. In the latter case, the noise associated with moving images in the traditional frameless rendering is reduced substantially.

Our last example uses the Visible Human dataset ( $512 \times 512 \times 2048$  resolution) and our direct volume rendering pipeline. The frameless visualization configuration is similar to the one used for the combustion data - the lattice samples are used to generate a low resolution noise free images, which is then augmented with adaptive samples from two types of sample generator. In this case, all sample generator types use the same rendering algorithm but different settings. The highest quality adaptive samples (HighQ) are generated with a higher-order reconstruction filter for the volume data and a step size of  $0.5 \cdot voxelWidth$ . The lower latency adaptive samples (LowQ) are produced at the same settings as the lattice samples (trilinear reconstruction, larger step size) and are used to rapidly improve the resolution of the image during interactive exploration and transfer function changes. Figure 6.26 shows the results obtained at three different times during the rendering of the Visible Human dataset.

We have tested our frameless visualization technique on a single display node of the Reality Deck, which comprises an 80MP tiled display. This is currently the highest resolution that can be driven from a single computer at 60Hz display refresh rate. If 30Hz is acceptable, 4K displays can also be driven from a single DisplayPort connector for a total 200MP resolution (4K 60Hz currently requires the use of a DisplayPort MST hub, which halves the number of monitors that can be connected to the GPU). To summarize the relevant specifications, we use 24 monitors at  $2560 \times 1440$  resolution connected to four AMD FirePro V9800 GPUs in a quad-GPU workstation. The monitors are configured in four bezel-compensated Eyefinity groups and each GPU renders to a single  $7960 \times 3022$  frame buffer. The sample generation is performed

**Table 6.2:** Sample throughput (samples/sec) and convergence times (s) for a 6-node GPU cluster and a single quad-GPU visualization node running 24 screens at  $7960 \times 12088$  total resolution.

Sample Type	Throughput	1 screen	24 screens
Adaptive LowQ	8.93M	0.8s	21s
Adaptive HighQ	1.25M	6s	160s

on the 6-node GPU cluster for the Immersive Cabin in which each node contains a single NVIDIA Quadro K5000 GPU. The two clusters are currently connected via a 1Gb connection. Table 6.2 summarizes the sample throughput for our cluster and the convergence times when updating a single screen and all 24 displays. Note that the 6-node cluster fully saturates the 1Gb network when the LowQ adaptive samples are used exclusively. In the future, we plan to extend our Infiniband network so that a larger GPU clusters can be used to generate full resolution images more quickly.

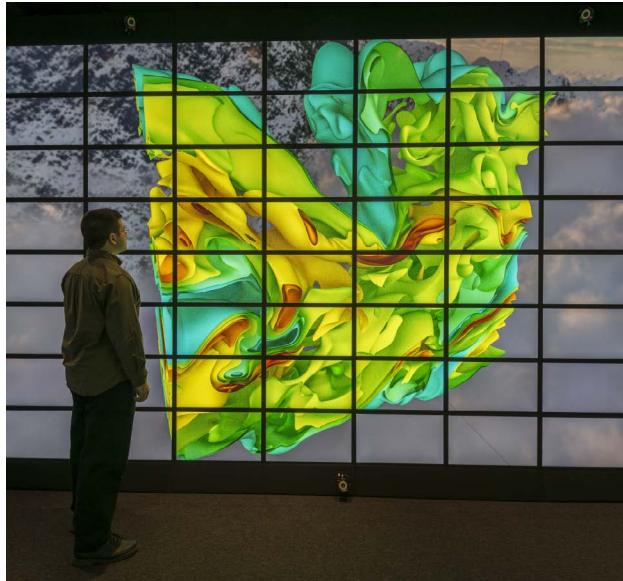
We also compared our frameless visualization against traditional volume rendering running on the 80MP display. We used settings similar to the ones for the LowQ adaptive samples for a total render time of approximately 10s per full resolution frame, during which time the user interface is frozen. In contrast, while our system requires 21s to converge at the same setting, the interface updates at stable 5 frames per second and functions such as camera and transfer function changes remain responsive. Initial experiments indicate that newer GPUs (e.g., AMD FirePro W9000) are up to twice as fast for frameless visualization at such high resolutions, primarily due to the significantly higher memory bandwidth. Our system can optionally use the Reality Deck’s 24 camera optical tracking system to determine the position and orientation of the user in relation to the screen. The tracking data is used to modify the priority map based on the distance to the screens and the screen coverage within the user’s field of view. Our initial experiments with this modified priority map indicate that approximately 10M adaptive samples may be sufficient to saturate the human eye (depending on the user’s visual acuity) which our system can deliver in under 10 seconds, even when the high quality samples are used.

Finally, Figure 6.27 presents the results obtained in the Reality Deck for both interactive GPU raytracing and volume rendering. The raytracing is configured to run the sample generators on the 6-node NVIDIA GPU cluster from the Immersive Cabin and produces results similar to the image in Figure 6.23c for the combustion dataset. The image reconstruction is running on one of the side walls of the Reality Deck, which is a  $10 \times 8$  grid of monitors with 300MP resolution. The initial low resolution result is generated very quickly while the user can use interactive navigation devices and the result in Figure 6.27a is obtained after 5 minutes of sample accumulation. When path tracing is disabled, the full resolution visualization of the isosurfaces is obtained in less than a minute.

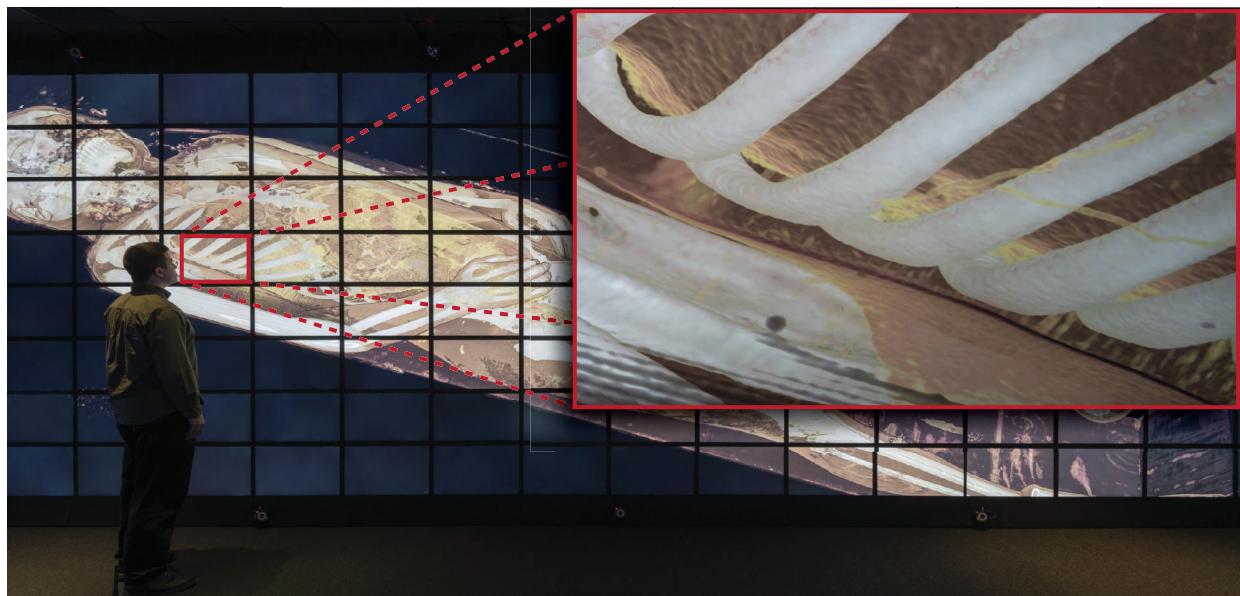
## **6. DISTRIBUTED IMMERSIVE VISUALIZATION**

---

The volume rendering in the Reality Deck is configured to use a much large heterogeneous GPU cluster for the sample generators. The image reconstruction is running on the front wall of the Reality Deck, which is a  $16 \times 8$  grid of displays with a 470MP total resolution, driven by 6 cluster nodes and 22 GPUs. The backend is running on 6 nodes from the Reality Deck cluster with 24 AMD FirePro V9800 GPUs and the 6 nodes from the Immersive Cabin cluster with 6 NVIDIA Quadro K5000 GPUs. The volume rendering is configured to produce the HighQ adaptive samples, which use both high-order filtering for the volume reconstruction and a small step size for the raycasting. These are currently the highest quality settings available in our visualization framework, producing the full resolution result depicted in Figure 6.27b in about a minute and with immediate visual feedback to user interaction with the virtual camera and the transfer function.



(a) Combustion Simulation



(b) Visible Human

**Figure 6.27:** Interactive frameless visualization of (a) the combustion simulation dataset and (b) the Visible Human volumetric dataset at the full display resolution. The backend sample rendering is configured respectively for GPU raytracing with 6 nodes / 6 GPUs and volume rendering with 12 nodes / 30 GPUs.

## **6. DISTRIBUTED IMMERSIVE VISUALIZATION**

---

### **6.3 The Reality Deck Infinite Canvas**

Immersion is a significant factor in virtual environments and it has been shown to positively affect various tasks such as visual searching [92]. The growth of data sizes in both science and industry generally outpaces advancements in display technology, as well as our ability to explore the data efficiently. We present the infinite canvas technique for managing the visualization of very large 2D datasets in immersive displays. Our method is not limited to any single display type and can be used with Head-Mounted Displays, CAVEs, as well as large tiled displays with immersive layouts. The goal is to provide the user with a very natural navigation interface based on simply walking within the display, thereby improving the immersion in the data.

We demonstrate the infinite canvas in the world's first 1.5 billion pixel tiled immersive display. Our system relies on the optical tracking of the user within the confines of the display. As the user walks along the display wall or turns in place, we manipulate the visual data outside of the field of view in order to simulate a single continuous surface that is many times larger than the actual display surface. In effect, the immersive visualization facility becomes a moving window into an *infinite canvas*. Since walking can be tiring, our technique includes a visual summarization of the infinite canvas in a navigation spiral. The user can move vertically through the spiral and thereby jump between points on the canvas by using any 3d navigation device, such as a gamepad or a wand.

Locomotion is one of the quintessential aspects of interaction with an immersive visualization [13] and a vast number of modalities exist for manipulating the position of a viewpoint within the virtual space. Natural walking has been shown to provide a greater sense of presence, contrary to techniques that do not utilize natural movements [111]. The primary difficulty associated with walking as a locomotion metaphor within an immersive visual environment is mapping the physical space to a much larger virtual space. Many techniques such as omnidirectional treadmills, when combined with tracking systems [55] and the popular walking-in-place modality [99], increase immersion but do not provide the feeling of physical displacement that one expects when walking.

A promising solution to this challenge is the *redirection* family of techniques, which manipulate the user's trajectory within the virtual environment, making it dissimilar to the physical path the user is traversing. Techniques such as Redirected Free Exploration with Distractors [83] combine subtle and overt reorientation of the user in order to maintain the position within the physical constraints of the tracked space. Methods that manipulate translation augment the user's movement towards the direction of intended travel [54]. Other techniques such as ArchExplore [14] integrate multiple redirection paradigms.

Our infinite canvas is inspired by the redirection techniques. Under the conventional formulation for redirection, a large 3D virtual scene is mapped to a smaller physical workspace. In our method, a large,

and theoretically unbounded, virtual canvas is mapped to a physical viewport into that canvas created by a fully-enclosed display. Similar to other redirection methods, this viewport is manipulated to mitigate the effect of the user approaching the physical boundary, which in our case corresponds to performing a full trip around the walls of the facility and ending back at the starting point. This manipulation occurs continuously in a manner invisible to the observer. Thus, under the Suma et al. taxonomy [105], our technique can be categorized as *continuous subtle repositioning*.

We introduce the concept of visualizing an arbitrarily long strip of high resolution data, hence the term *infinite canvas*, onto a fully enclosed immersive display together with a very natural navigation interface. In particular, the enclosed nature of the display allows us to define a closed curvilinear virtual surface that fully immerses the user in the data. Sections of the original strip are then mapped to the virtual surface and the entire strip can be visualized based on the input from the user. The main navigation tool is simply walking along the display surface and we have developed a technique that dynamically changes the mapping based on the tracked position and orientation. Therefore, a user walking along the edge of the display perceives a single continuous surface even as multiple revolutions are made within the physical space in both directions, essentially exploring an infinite horizontal canvas. In the following, we present the basic formulation of our technique, as well as an extension for accelerated navigation of very large datasets with a spiral visual interface.

### 6.3.1 Basic Formulation

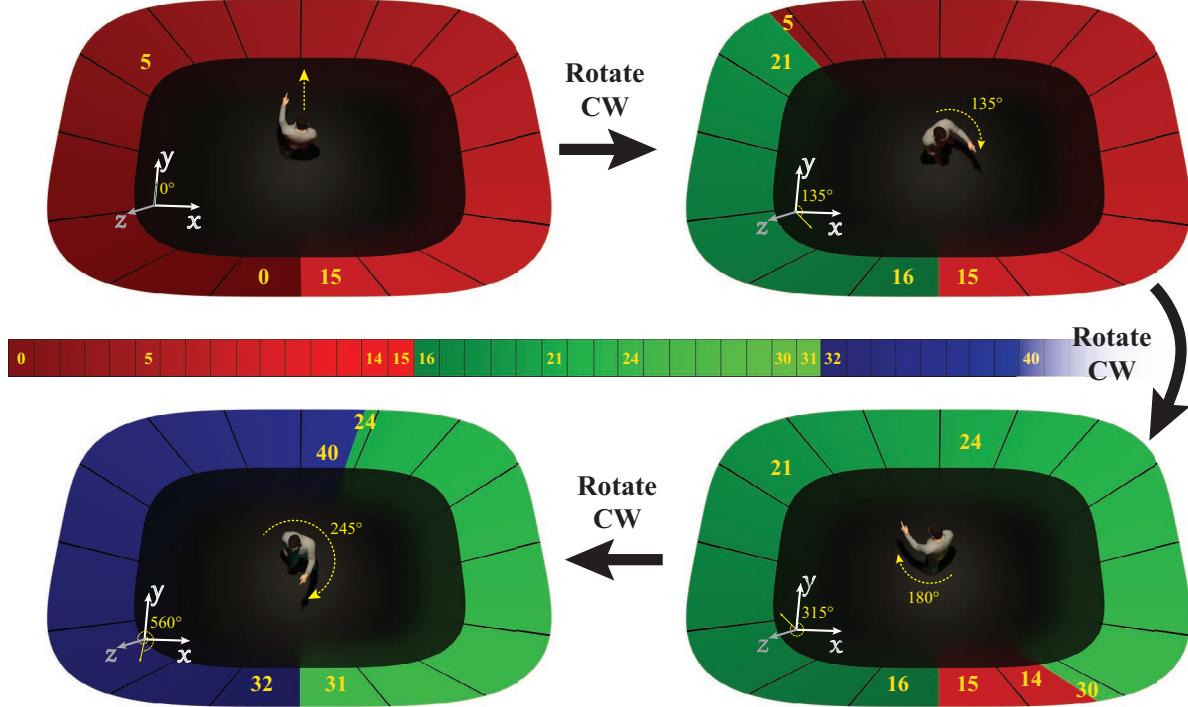
In our technique, 2D data is rendered under perspective projection onto a 3D surface that matches the topology of the display surfaces. The most intuitive canvas shape for creating the infinite canvas is the cylinder and it is in fact our choice when the display device does not possess fixed screens (e.g., HMDs), or when the screens are approximately square (e.g., fully-enclosed, 5 or 6-sided CAVEs). The fixed curvature of the cylinder allows for certain operations on the canvas, such as rotation of the geometry, without destroying the illusion of smoothness. However, the cylinder is not appropriate for displays with varying aspect ratios as that leads to significant loss of visual information for the screens with lower aspect ratios under perspective projection. Instead, we procedurally generate a canvas whose horizontal profile is a rectangle with rounded edges. We use the following parametric specification for the curve with  $\theta \in [0, 2\pi]$ :

$$\begin{aligned} x(\theta) &= a \cdot |\cos\theta|^{\frac{2}{n}} \operatorname{sign}(\cos\theta) \\ y(\theta) &= b \cdot |\sin\theta|^{\frac{2}{n}} \operatorname{sign}(\sin\theta) \end{aligned} \tag{6.9}$$

where for example,  $a = 1.6$  and  $b = 1.0$  are used to create a virtual canvas for a display facility with a 16 : 10 floor aspect ratio. The corner curvature is determined by  $n$  and it is application-specific. However,

## 6. DISTRIBUTED IMMERSIVE VISUALIZATION

---



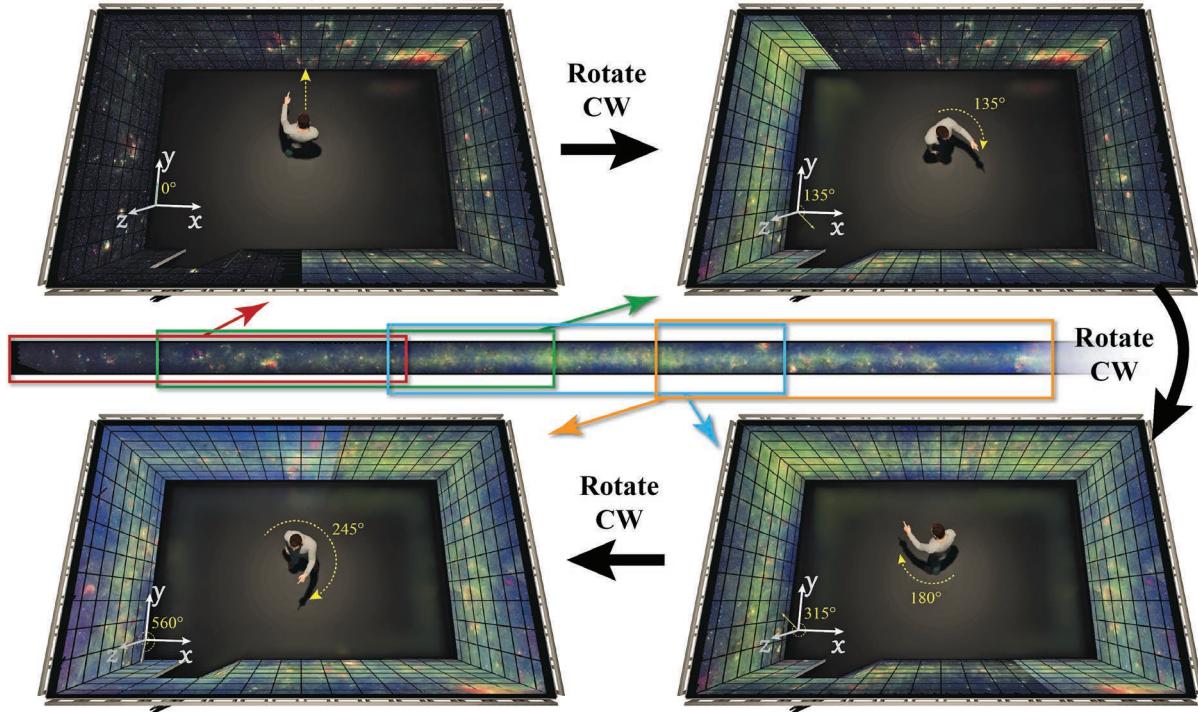
**Figure 6.28:** The full dataset (center strip) is three times longer than the surface of the immersive display. As the user rotates clockwise, different segments of the data are revealed. The angle of relative rotation is indicated next to the human figure and the cumulative angle of rotation is shown at the bottom left of each subfigure. The discontinuity where the canvas wraps over is always kept behind the user, which provides the perception of one long continuous surface. The numbers on the screens in each subfigure correspond to the numbers on the original strip of data.

we have determined experimentally that  $n = 4$  is sufficient so that the illusion of a smooth continuous surface is achieved under perspective projection with minimal loss of visual information at the corners of the physical display.

We have developed a natural navigation interface where the user interacts with the visualization by walking and turning inside the enclosed physical space of the display. We obtain both the position  $\mathbf{p}$  and the orientation  $\mathbf{d}$  from a tracking system and the yaw angle is integrated to obtain the unbounded cumulative angle of rotation  $\sigma$  with respect to a reference orientation vector. The vector  $\mathbf{p} + \mathbf{d}$  is then intersected with the geometry of the virtual surface to obtain the point  $\mathbf{p}_{\text{back}}$  on the canvas that is directly behind the user and corresponds to the cumulative angle  $\sigma_{\text{start}}$ , where:

$$\sigma_{\text{start}} = 2\pi \left\lfloor \frac{\sigma}{2\pi} \right\rfloor - \arctan \left( \frac{p_{\text{back},x} - p_x}{p_{\text{back},y} - p_y} \right) \quad (6.10)$$

Then, the data in the region defined on the horizontal axis by  $a \cdot [\sigma_{\text{start}}/2\pi, \sigma_{\text{start}}/2\pi + 1]$  is mapped to the geometry of the canvas starting and ending at  $\mathbf{p}_{\text{back}}$  in the clockwise direction. The scaling factor  $a$  is



**Figure 6.29:** The infinite canvas technique from Figure 6.28 is visualized using the NASA Milky Way dataset on a 3D model of the Reality Deck. The images for the simulated screens are captured directly from our visualization software.

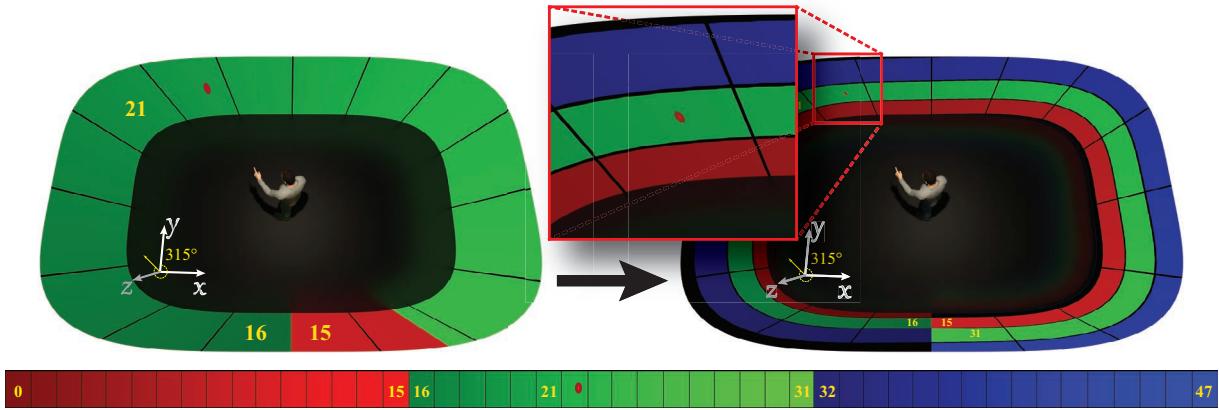
the ratio between the circumference of the horizontal profile described in Equation 6.9 and the width of the virtual canvas. For example, the canvas illustrated in Figure 6.28 is three times longer than the immersive display and therefore setting  $a = 1/3$  normalizes the coordinates that are used to fetch data from the data storage. Figure 6.29 illustrates the technique with a real dataset displayed on the 3D model of the Reality Deck.

Our navigation interface supports the following fundamental operations: (1) *Walking along the display*, (2) *Turning in place*, and (3) *Walking across the facility*. As the user walks along the display, the point  $\mathbf{p}_{\text{back}}$  continually moves based on the tracking data, which allows for the exploration of the horizontal axis of the infinite canvas. We map clockwise motion in the physical space to essentially going *forward* along the data and counterclockwise motion to going *backwards*. The same effect can be achieved for a stationary user, who can rotate their head clockwise or counterclockwise in order to quickly examine large sections of the infinite canvas without having to walk a long distance. Finally, the user can combine these interactions to skip between sections of the canvas by rotating and walking across the floor to the point on the canvas that they wish to examine.

Figure 6.28 illustrates the infinite canvas effect through four snapshots during the rotation of the user.

## 6. DISTRIBUTED IMMERSIVE VISUALIZATION

---



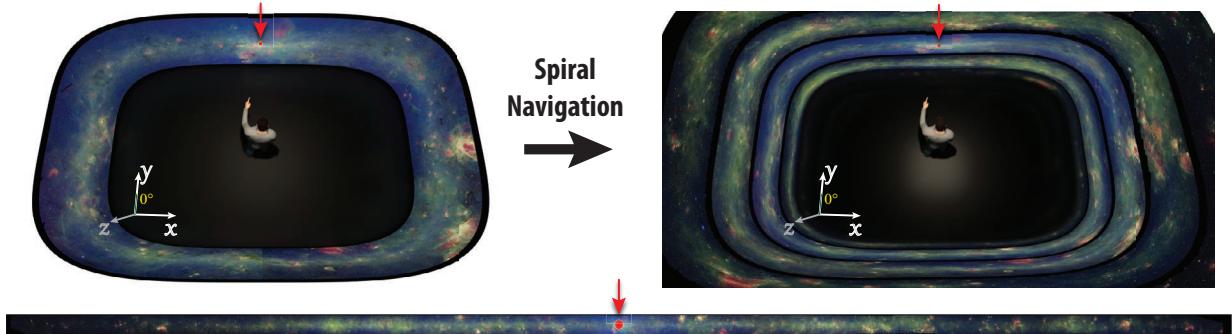
**Figure 6.30:** The spiral navigation interface provides a quick overview of the entire dataset that the user can explore by moving inside the immersive display. A red flashing highlight is used to indicate the exact position in the full data.

For simplicity, the example considers only the case of rotation, however, the same principles apply when translation is used as well. In the initial state (top-left), the immersive display shows the full red section from the data (segments 0 to 15). As the user rotates clockwise to 135 the wrap-around point  $p_{back}$  moves so that we bring into view the continuous collection of segments 16 through 21. As the user continues the rotation, more of the second section of the data, labeled in green, is shown. Finally, as the cumulative angle of rotation increases beyond 360 we start to display segments from the last data section, labeled in blue. The user can go back through the data by then rotating in the counterclockwise direction. The illusion of a single continuous surface is achieved by never displaying the discontinuity at  $p_{back}$  in the field of vision of the observer.

### 6.3.2 Spiral Navigation

Based on the definition in Section 6.3.1, our system can handle virtual canvases of arbitrary sizes. One of the challenges is that as the data sizes increase, the use of body motion for navigation in a large physical space becomes impractical. We solve this by providing the user with an overview of the entire virtual canvas in a spiral interface that is shown on the immersive display, can be triggered at any time, and it has a clear mapping with the original visualization.

Once the spiral interface is triggered, our system provides a smooth transition between the infinite canvas and the spiral visualization. This aspect is critical as it allows the user to easily identify the cycle of the spiral to which the current canvas is mapped. The animation is performed in two distinct steps. During the first phase, we smoothly scale down the canvas vertically based on how many cycle we want to visualize while the horizontal mapping is kept consistent. At the same time, we apply an angle-dependent vertical offset



**Figure 6.31:** Exploring the Milky Way data with the spiral navigation interface. The original image (bottom) is five times longer than the immersive display. The user can quickly overview the entire dataset by moving up and down through the spiral visualization using a gamepad or another interaction device.

$$y_{offset} = a \cdot \frac{\theta}{2\pi} \quad (6.11)$$

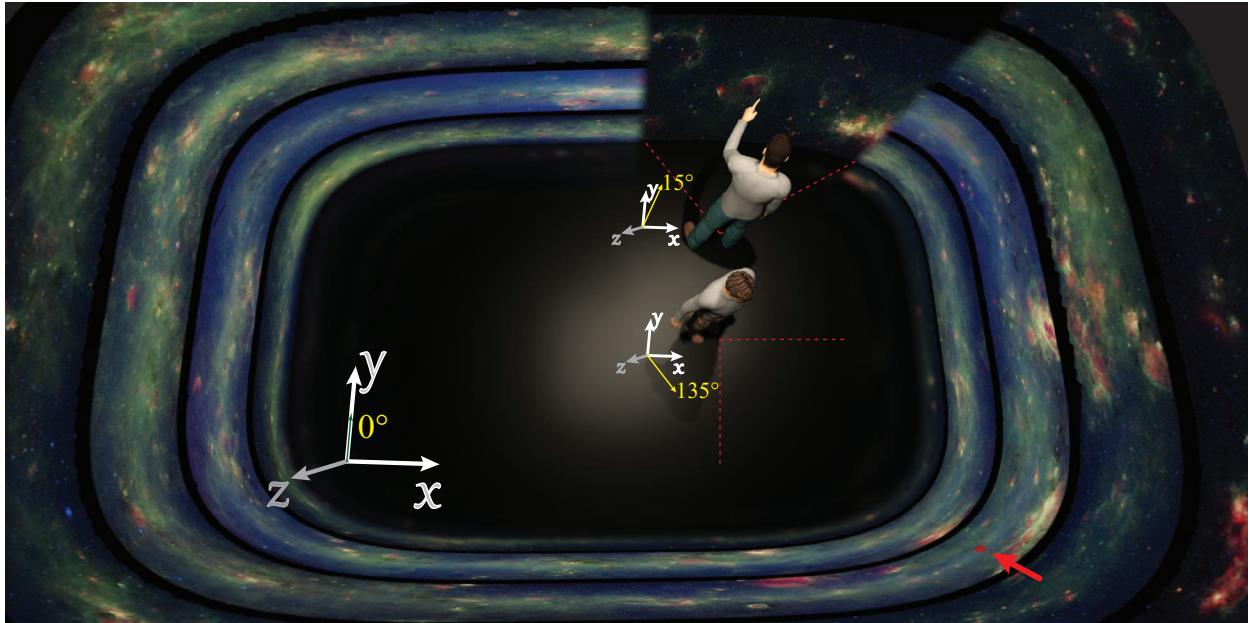
where  $a$  is the height of each strip in the spiral and  $\theta$  is the absolute angle of the image element with respect to a reference angle. The end result is that the original canvas is mapped to a single cycle on the spiral. In the second step, we fade in the full visualization of the spiral. Whenever the spiral is shown, a blinking circular highlight indicates the current position within the space of the canvas. As the user rotates, the highlight follows the view direction. When the spiral is turned off, the animation is run in reverse - the spiral visualization fades to black except for the currently selected cycle, which is then smoothly expanded to the original visualization. This principle is illustrated in Figure 6.30. The left-hand side shows the user exploring the center of the data. The spiral on the right shows the complete dataset, which the user can explore by moving within the immersive display and by moving vertically in the virtual 3D space using an interaction device, such as a gamepad. Figure 6.31 shows the case when the full dataset is seven times longer than the immersive display and therefore requires seven cycles in the spiral.

One challenge with this approach is that as the data size increases, the scaling required to fit the entire data becomes too large, thereby reducing the quality of the overview. In our experiments, we found that for image data, we can show at most three cycles of the spiral at the same time. For data in which the aspect ratio does not change the comprehension of the structure (e.g., graph or connectivity information), the limit is application-specific and may be higher. In the case of large data, we provide a secondary and more traditional interface for exploring the spiral using a gamepad or any similar input device to move vertically in 3D space through the spiral.

We have also implemented a spiral interface that preserves the aspect ratio of the data and the entire spiral is fitted to the same visible surface used to display the Infinite Canvas. While the mapping between the original visualization and the spiral navigation is less clear, the results are improved significantly for

## 6. DISTRIBUTED IMMERSIVE VISUALIZATION

---



**Figure 6.32:** Synthesized view of the multi-user exploration interface. The person closer to the screen is examining the beginning of the dataset, while the second person is using the spiral navigation interface to select point near the center of the dataset (the red highlight is at the central cycle in the spiral).

image data. We still provide a smooth animation to help the users identify where the current data is placed in the spiral. The difference is that we use the tracking data to rotate the spiral visualization so that the original view and the current position in the spiral are matched.

### 6.3.3 Multiuser Visualization

The physical size, the layout and the gigapixel resolution provided by our facility are uniquely ideal among the immersive and partially immersive displays (e.g., CAVEs and HMDs) for collaboration tasks and multi-user visualization. In particular, our infinite canvas implementation supports multiple tracked users who can explore the data and trigger the navigation spiral independently on the same shared surface. Each user is provided with a 90° horizontal field of view and a full vertical field of view. During rendering, we associate each pixel with a user based on the containment test for that user's viewing frustum, where pixels near the center of the frustum are assigned a higher weight. We further weigh by distance from the screen so that users examining the details in the image are not disturbed by users farther back who need an overview of the data on the same section of the display. Figure 6.32 illustrates the case of two user sharing the immersive display. The scene is artificially synthesized for clarity, however the actual image mapped on the surface is captured directly from our application running on the GPU cluster.

### 6.3.4 Results

We use the Reality Deck visualization framework presented in Section 5.2.2 as the basis for developing the Infinite Canvas. As a result, the technique can also be used in the Immersive Cabin and with HMDs, although the lower pixel density of these devices limits the resolution of the data that can be explored effectively. All the concepts presented in Section 6.3.1, 6.3.2 and 6.3.3 are implemented entirely on the GPU with vertex and pixel shaders in the GLSL language and a minimal interface to the underlying visualization framework. In particular, we create the canvas geometry based on Equation 6.9 and manipulate its texture coordinates in the shaders during rendering to achieve the infinite canvas and the navigation spiral effects. The resulting system is largely independent from the underlying visualization framework, which simplifies the integration with other systems. The original  $U$  texture coordinate range  $[0, 1]$  is mapped to  $a \cdot [\sigma_{start}/2\pi, \sigma_{start}/2\pi + 1]$ , where  $\sigma_{start}$  is computed as described in Section 6.3.1. The navigation spiral is implemented in a similar fashion. We compute Equation 6.11 in the pixel shader and then compute the correct texture coordinates for each pixel so that we sample the entire data and show the result in a spiral. If we need to stretch the canvas vertically in order to accommodate large datasets, the change to the canvas geometry is made in the vertex shader during rendering.

We have tested our Infinite Canvas technique in the *Reality Deck* which is presented in Section 5.2. To summarize the relevant details, it is a large fully-enclosed visualization facility composed of 416 high-resolution tiled LCD monitors in a 4-wall arrangement, a GPU cluster with 20 nodes, and an optical tracking system with 24 IR cameras. The door to the facility is a  $3 \times 5$  segment of the wall and it is visually indistinguishable from the remainder of the tiled display. The display is the first in the world to offer an aggregate resolution in excess of 1.5 billion pixels and it is particularly suitable for the visualization of high resolution scientific data. The unique feature of the Reality Deck related to the implementation of the Infinite Canvas is that the users are fully enclosed horizontally in the visualization when the door is closed, allowing for uninterrupted navigation through the virtual canvas surface.

We evaluate the infinite canvas visualization technique with a 6 gigapixel infrared portrait of the inner Milky Way galaxy from the NASA's Spitzer Space Telescope (see Figures 6.36a and 6.33), which combines observations of the Galactic Legacy Infrared Mid-Plane Survey Extraordinaire (GLIMPSE) and MIPSGAL projects [8, 19]. Compared to the computational cost of the gigapixel renderer, both the infinite canvas and the navigation spiral have a negligible impact on the system performance.

The second dataset is a collection of more than 70,000 paintings from the 14th century up to modern times (see Figure 6.34). The tiles in our gigapixel implementation are generated dynamically from the image files and any associated metadata. The canvas for this dataset is significantly larger and spans an area that is approximately 50 times larger than the area of the Reality Deck when the individual images are displayed at approximately full resolution. While we can project the grid of paintings directly onto the curved 3D mesh,

## 6. DISTRIBUTED IMMERSIVE VISUALIZATION

---



**Figure 6.33:** The 6GPixel infrared portrait of the inner Milky Way galaxy from the combined observations of the Galactic Legacy Infrared Mid-Plane Survey Extraordinaire (GLIMPSE) and MIPSGAL projects is shown on the immersive display using the navigation spiral.

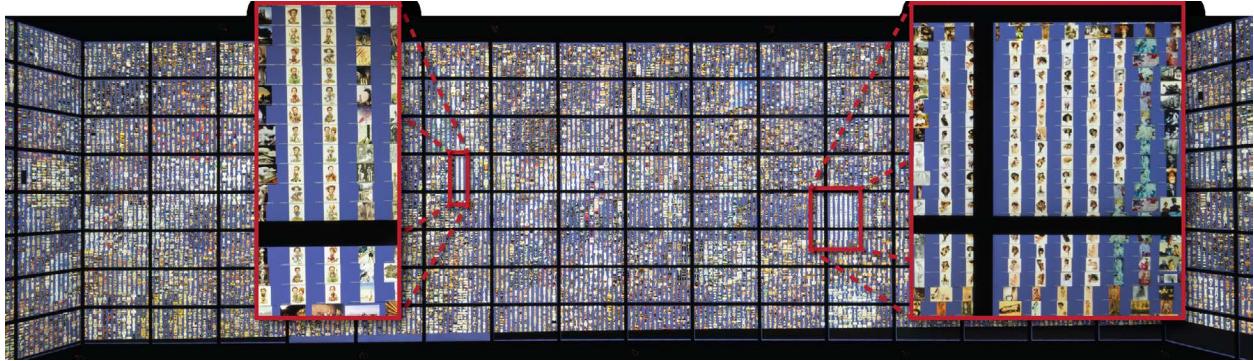


**Figure 6.34:** Our art gallery dataset contains more than 70,000 painting which can be displayed one per screen (right) or in grid layouts depending on the image resolution (left) using the infinite canvas.

similar to how we handle the Milky Way data, it is more effective to take advantage of the physical bezels as image separators. We instead dynamically generate a collection of rectangles that cover the visible surfaces in the Reality Deck and the Infinite Canvas is mapped to this mesh without any further modifications. As demonstrated in Figure 6.34, we can display a single high resolution painting per screen, or various layouts that are suitable for lower resolution images. The spiral navigation uses the aspect-preserving version of the algorithm and we can display the entire collection at once resulting in approximately  $13 \times 13$  images per screen. There is sufficient resolution so that the user can perceive some features of the paintings, as well as clusters in the data (see Figure 6.35).

Figure 6.36 illustrates the multiple user visualization with the Infinite Canvas, as rendered on the actual immersive display. In the initial stage (see Figure 6.36a), both users are exploring the same section of the

### 6.3 The Reality Deck Infinite Canvas



**Figure 6.35:** The entire 70,000 paintings art gallery is displayed using the spiral interface. There is sufficient resolution to visually identify clusters of paintings.

Milky Way dataset. The positions are similar to the ones in Figure 6.32 except that the second user is looking at the right wall and not at the back wall. In Figure 6.36b, the navigation spiral is triggered without affecting the view of the first user. In the third step (see Figure 6.36c), the spiral navigation interface is disabled and the second person continues to explore a different section of the canvas than the first.

We performed a small and informal study of the usability of our system. The users were free to walk inside the facility and to trigger the navigation spiral at will. We were interested in how natural the user interface felt and the results were generally positive - our users never saw the image discontinuity at point  $P_{back}$ , were often not aware that they had walked a complete circle around the facility and liked the simplicity of the interaction. The spiral navigation was also useful when searching for a specific feature in the data. In the future, we plan to conduct a more comprehensive user study of both the infinite canvas and the usability aspects of large-scale, immersive and high resolution displays. This is particularly important as building such installations is becoming easier and cheaper due to advances in systems, GPUs and display technologies.

## 6. DISTRIBUTED IMMERSIVE VISUALIZATION

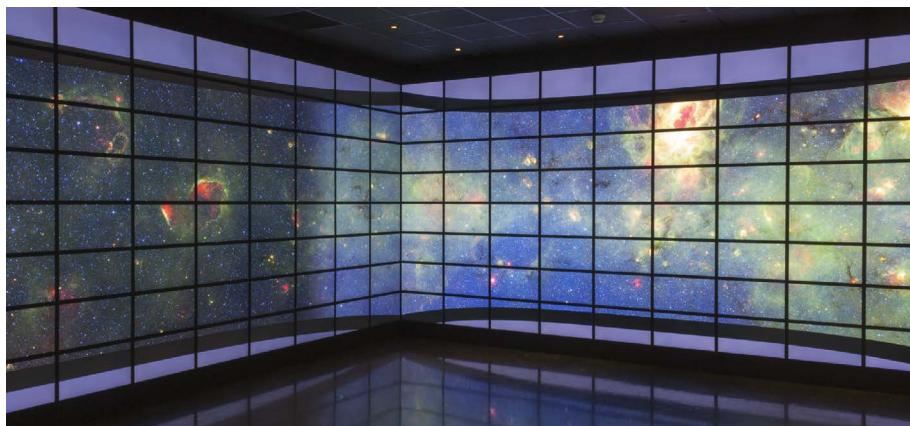
---



(a) Both users explore the same section of the Infinite Canvas.



(b) The second user triggers the Spiral Navigation and selects the third spiral cycle.



(c) The second user disables the spiral and continues to explore a different section of the infinite canvas.

**Figure 6.36:** Two-user exploration of the Milky Way dataset in the immersive display. The photo shows the front-right corner of the facility. The first user is looking toward the front screen and the visualization is not affected by the actions of the second user who is primarily looking at the right screen when interacting with the data.

# 7

## Conclusions

### 7.1 Summary of Contributions

In this work, several techniques have been presented that take advantage of the optimal sampling lattices for both natural phenomena simulation and for volume rendering. The LBM has been used for realtime plume dispersion in urban environments based on the fD3Q13 model over the FCC lattice. Our CFD code supports a variety of models (e.g., MRT-LBM and Hybrid Thermal LBM), higher order boundary conditions, the CC, BCC and FCC lattices with different sets of neighbor connectivity, and GPU acceleration. The simulation code is further integrated into a visualization framework that can provide immediate visual feedback and analysis of the simulation data.

The component of the visualization framework dealing with volumetric data further supports our hierarchical multiresolution volume representation and in particular the rendering of the mixed-lattice hierarchies. The use of the CBF and the CFB refinements provides smoother transition between adjacent levels of detail compared to the CC-only refinements. Furthermore, our code uses a unified storage scheme for all the lattice types, which simplifies the integration with volume rendering systems.

Section 5.1 presents a visualization frameworks for immersive visualization systems, and the Immersive Cabin in particular, that has been designed from the ground up for immersive rendering of volumetric data. While the system is sufficiently general and handles all of the traditional virtual reality tasks, the focus has been on visualizing mesh and volumetric data together for lattice-based natural phenomena visual simulations and for high quality medical visualization.

During the course of this work, we have designed and constructed the state-of-the-art Reality Deck facility, which is the first tiled display in the world to provide more than 1.5 gigapixel resolution with a horizontally enclosed immersive layout. The Immersive Cabin visualization software has been extended to support the Reality Deck, together with rendering features that take advantage of the gigapixel resolution.

## 7. CONCLUSIONS

---

Both of our facilities are running a unified version of the visualization framework, and some of the case studies in Section 5.3 leverage that integration. The frameless visualization technique in Section 6.2 also takes advantage of the unified GPU clusters, where the Immersive Cabin machines are used as sample generators for visualization in the Reality Deck.

This thesis further presents three novel techniques for immersive visualization:

**Conformal Visualization** uses angle-preserving conformal maps to create a correspondence between the complete set of viewing directions and the viewing directions possible in partially immersive environments such as the 5-sided Immersive Cabin and the 4-sided Reality Deck. The technique is particularly suitable for use with raytracing rendering pipelines, including our lattice-based volume rendering, as well as realtime GPU-based raytracing. The technique is also applicable to the rasterization graphics pipeline, although the scalability issues with very large meshes present certain challenges. The technique is demonstrated in both the Immersive Cabin and the Reality Deck.

**Frameless Visualization** is a technique designed to decouple the display from the rendering system by constructing the final image from a continuous stream of rendered samples rather than from a rendered frame. Compared to the traditional frameless rendering, our technique is designed to support very high resolutions, such as the 1.5 gigapixels in the Reality Deck, for volume rendering and raytracing applications.

The **Infinite Canvas** is a visualization technique for horizontally enclosed immersive facilities that is based on the VR concept of *redirection*. As the user moves through the tracked physical space, the visualization outside of the field of view is changed in a way that creates the illusion of a single infinitely long canvas, on which different types of data can be displayed. The technique is demonstrated in the Reality Deck for the visual exploration of a NASA dataset of the Milky Way and for very large image collections.

## 7.2 Future Research

### Short-term

This thesis presents a comprehensive framework for lattice-based simulation and visualization, however there are a number of interesting challenges that warrant further investigation. The LBM handles complex boundary conditions through the angular discretization of the lattice velocities and in fact the simplicity of applying the boundary conditions during the propagation step is a major advantage compared to some other simulation techniques. Moving boundaries can be handled on the GPU by using a binary voxelization at each simulation step, although this is not sufficient for higher-order boundary condition. Instead, we can use the realtime GPU raytracing in our framework to compute accurate intersections between the lattice links and the dynamic scene geometries. The same raytracing engine can then be used during visualization for both the scene geometry and the volume data, resulting in a much more tightly integrated system. It is also

interesting to explore the use of the mixed-lattice hierarchy for the LBM simulation itself and not just for volume rendering, as well as extending the framework to support modern GPU clusters in order to handle significantly larger problem sizes.

There are several interesting directions for future research based on the immersive visualization work in this thesis. One is the development and rigorous evaluation of user interfaces for interaction with large high resolution displays, such as the Reality Deck. In particular, we are interested in developing natural interaction techniques based on tracking data at different scales, such as full-body, hands and head, and fingers. Gigapixel displays allow for additional interaction as simply moving closer to the display can reveal a significant amount of new information. Positional sound cues can also play an important role in displays that surround the user and can be considered as part of the user interface.

The display bezels are an unfortunate limitation of current high resolution LCDs, however, they can be used in the user interface as natural separators of the data. We have demonstrated this with the Infinite Canvas exploration of image collections, but it should be handled in a more comprehensive and general way. For example, parallel coordinates visualizations of multidimensional data on the Infinite Canvas can place each axis along the vertical screen bezels, or graph layout algorithms can use the bezels for clustering of data points.

Another research direction is related to the handling of very large streaming data as part of the high resolution immersive visualization framework. Many of the large data examples in this thesis use pre-processed datasets, however, we are at a point where monolithic gigapixel cameras and high resolution CT scanners are becoming a reality and we need novel ways for the realtime processing, streaming and visualization of their data in high resolution facilities such as the Reality Deck.

### Long-term

This thesis and the aforementioned short-term research directions address some of the issues related to handling large data, from efficient sampling to building large immersive displays. However, over the long term, the challenges of big data remain as both the size and the complexity of the data increase. On the visualization side, the costs of all types of immersive displays are declining steadily while the quality increases. Together with the resurgence of technologies such as the head-mounted display, virtual reality has an important future role in the analysis of big data, in addition of course to the more traditional entertainment applications that benefit from the continual increases in computational power. In particular, interfaces that combine visualization with sound, physical touch and movement in the least obtrusive way can open new natural interaction opportunities.

A related challenge is the representation of data on these new types of displays for efficient analysis. While there are vast amounts of knowledge about the 2D displays, there are significantly fewer techniques

## **7. CONCLUSIONS**

---

for 3D, volumetric, holographic, and other exotic types of displays. Optimal sampling lattices can play an important for example in volumetric displays, where the memory savings in the higher dimensions (e.g. 3D+time) can be significant. Techniques such as the frameless visualization presented in this thesis can generalize to volumetric displays, however more comprehensive research would be required for rendering on such devices.

# Bibliography

- [1] U. R. Alim, A. Entezari, and T. Möller. The Lattice-Boltzmann method on optimal sampling lattices. *IEEE Transactions on Visualization and Computer Graphics*, 15(4):630–641, 2009. [1](#), [7](#), [22](#), [27](#)
- [2] R. A. d. Almeida, C. Pillias, E. Pietriga, and P. Cubaud. Looking behind bezels: french windows for wall displays. *International Working Conference on Advanced Visual Interfaces*, pages 124–131, 2012. [80](#), [92](#)
- [3] A. Asirvatham and H. Hoppe. *Terrain rendering using GPU-based geometry clipmaps*, chapter 2, pages 27–46. Addison-Wesley Professional, 2005. [79](#)
- [4] M. Barnes. Collada. *ACM SIGGRAPH Courses*, page 8, 2006. [62](#)
- [5] S. Barrett. Sparse virtual textures. *Game Developer Conference*, 2008. [79](#)
- [6] N. Beck and A. Hinkenjann. DiReC: distributing the render cache to PC-clusters for interactive environments. *ACM Symposium on Virtual Reality Software and Technology*, pages 159–162, 2005. [15](#)
- [7] U. Behrens and R. Ratering. Adding shadows to a texture-based volume renderer. *IEEE Symposium on Volume Visualization*, pages 39–46, 1998. [39](#)
- [8] R. A. Benjamin, E. Churchwell, B. L. Babler, T. M. Bania, D. P. Clemens, M. Cohen, J. M. Dickey, R. Indebetouw, J. M. Jackson, H. A. Kobulnicky, A. Lazarian, A. P. Marston, J. S. Mathis, M. R. Meade, S. Seager, S. R. Stolovy, C. Watson, B. A. Whitney, M. J. Wolff, and M. G. Wolfire. GLIMPSE. I. an SIRTF legacy project to map the inner galaxy. *Publications of the Astronomical Society of the Pacific*, 115(810):953–964, 2003. [81](#), [153](#)
- [9] J. Beyer, M. Hadwiger, T. Möller, and L. Fritz. Smooth mixed-resolution GPU volume rendering. *Volume Graphics*, pages 163–170, 2008. [1](#), [9](#), [47](#)

## BIBLIOGRAPHY

---

- [10] A. Bierbaum, C. Just, P. Hartling, K. Meinert, A. Baker, and C. Cruz-Neira. VR Juggler: A virtual platform for virtual reality application development. *IEEE Virtual Reality Conference*, pages 89–96, 2001. [61](#)
- [11] G. Bishop, H. Fuchs, L. McMillan, and E. J. S. Zagier. Frameless rendering: Double buffering considered harmful. *SIGGRAPH*, pages 175–176, 1994. [15](#), [132](#)
- [12] M. Bouzidi, M. Firdaouss, and P. Lallemand. Momentum transfer of a Boltzmann-lattice fluid with boundaries. *Physics of Fluids*, 13(11):3452–3459, 2001. [12](#), [25](#)
- [13] D. A. Bowman, E. Kruijff, J. J. LaViola, and I. Poupyrev. *3D User Interfaces: Theory and Practice*. Addison Wesley Longman Publishing Co., Inc., 2004. [146](#)
- [14] G. Bruder, F. Steinicke, and K. H. Hinrichs. Arch-explore: A natural user interface for immersive architectural walkthroughs. *IEEE Symposium on 3D User Interfaces*, pages 75–82, 2009. [146](#)
- [15] B. Cabral and L. C. Leedom. Imaging vector fields using line integral convolution. *SIGGRAPH*, pages 263–270, 1993. [25](#)
- [16] S. P. Callahan, M. Ikits, J. L. Comba, and C. T. Silva. Hardware-assisted visibility sorting for unstructured volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 11(3):285–295, 2005. [46](#)
- [17] S. Chen and G. D. Doolen. Lattice Boltzmann method for fluid flows. *Annual Review of Fluid Mechanics*, 30:329–364, 1998. [12](#)
- [18] B. Chow. The ricci flow on the 2-sphere. *Journal of Differential Geometry*, 2(33):325–334, 1991. [100](#), [101](#), [102](#)
- [19] E. Churchwell, B. L. Babler, M. R. Meade, B. A. Whitney, R. Benjamin, R. Indebetouw, C. Cyganowski, T. P. Robitaille, M. Povich, C. Watson, and S. Bracker. The Spitzer/GLIMPSE surveys: A new view of the milky way. *Publications of the Astronomical Society of the Pacific*, 121(877):213–230, 2009. [81](#), [153](#)
- [20] J. H. Conway, N. J. A. Sloane, and E. Bannai. *Sphere-Packings, Lattices, and Groups*. Springer-Verlag New York, Inc., 1987. [7](#), [19](#), [21](#), [22](#)
- [21] R. L. Cook, L. Carpenter, and E. Catmull. The reyes image rendering architecture. *Computer Graphics*, 21(4):95–102, 1987. [63](#)

## BIBLIOGRAPHY

---

- [22] C. Correa, D. Silver, and M. Chen. Illustrative deformation for data exploration. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1320–1327, 2007. [14](#)
- [23] C. Crassin, F. Neyret, S. Lefebvre, and E. Eisemann. Gigavoxels: Ray-guided streaming for efficient and detailed voxel rendering. *ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (I3D)*, pages 15–22, 2009. [9](#)
- [24] C. Cruz-Neira, D. J. Sandin, T. A. DeFanti, R. V. Kenyon, and J. C. Hart. The cave: Audio visual experience automatic virtual environment. *Communications of the ACM*, 35(6):64–72, 1992. [13](#), [58](#), [98](#)
- [25] A. Dayal, C. Woolley, B. Watson, and D. Luebke. Adaptive frameless rendering. *Eurographics Symposium on Rendering*, pages 265–275, 2005. [15](#), [135](#)
- [26] T. DeFanti, D. Acevedo, R. Ainsworth, M. Brown, S. Cutchin, G. Dawe, K.-U. Doerr, A. Johnson, C. Knox, R. Kooima, F. Kuester, J. Leigh, L. Long, P. Otto, V. Petrovic, K. Ponto, A. Prudhomme, R. Rao, L. Renambot, D. Sandin, J. Schulze, L. Smarr, M. Srinivasan, P. Weber, and G. Wickham. The future of the CAVE. *Central European Journal of Engineering*, 1(1):16–37, 2011. [70](#)
- [27] T. A. DeFanti, J. Leigh, L. Renambot, B. Jeong, A. Verlo, L. Long, M. Brown, D. J. Sandin, V. Vishwanath, Q. Liu, M. J. Katz, P. Papadopoulos, J. P. Keefe, G. R. Hidley, G. L. Dawe, I. Kaufman, B. Glogowski, K.-U. Doerr, R. Singh, J. Girado, J. P. Schulze, F. Kuester, and L. Smarr. The optiportal, a scalable visualization, storage, and computing interface device for the optiputer. *Future Generation Computer Systems*, 25(2):114–123, 2009. [70](#)
- [28] D. D’Humières, M. Bouzidi, and P. Lallemand. Thirteen-velocity three-dimensional lattice Boltzmann model. *Physical Review E*, 63:066702, 2001. [20](#), [21](#)
- [29] D. D’Humières, I. Ginzburg, M. Krafczyk, P. Lallemand, and L.-S. Luo. Multiple-relaxation-time lattice Boltzmann models in three dimensions. *Royal Society of London Philosophical Transactions Series A* 360, 1792:437–451, 2002. [12](#), [27](#), [34](#)
- [30] K.-U. Doerr and F. Kuester. CGLX: A scalable, high-performance visualization framework for networked display environments. *IEEE Transactions on Visualization and Computer Graphics*, 17(3):320–332, 2010. [61](#)
- [31] S. Eilemann, M. Makhinya, and R. Pajarola. Equalizer: A scalable parallel rendering framework. *IEEE Transactions on Visualization and Computer Graphics*, 15(3):436–452, 2009. [61](#)

## BIBLIOGRAPHY

---

- [32] E. Eisemann and X. Décoret. Single-pass GPU solid voxelization for real-time applications. *Proceedings of Graphics Interface 2008*, pages 73–80, 2008. [53](#)
- [33] A. Entezari. *Optimal Sampling Lattices and Trivariate Box Splines*. PhD thesis, Simon Fraser University, 2007. [1](#), [8](#), [9](#), [38](#)
- [34] A. Entezari, R. Dyer, and T. Möller. Linear and cubic box splines for the body centered cubic lattice. *IEEE Visualization*, pages 11–18, 2004. [8](#)
- [35] A. Entezari, T. Meng, S. Bergner, and T. Möller. A granular three dimensional multiresolution transform. *Eurographics/IEEE-VGTC Symposium on Visualization*, pages 267–274, 2006. [9](#), [45](#)
- [36] A. Entezari, D. Van De Ville, and T. Möller. Practical box splines for reconstruction on the body centered cubic lattice. *IEEE Transactions on Visualization and Computer Graphics*, 14(2):313–328, 2008. [8](#)
- [37] Z. Fan, F. Qiu, and A. E. Kaufman. Zippy: A framework for computation and visualization on a GPU cluster. *Computer Graphics Forum*, 27(10):341–350, 2008. [1](#)
- [38] R. Fedkiw, J. Stam, and H. W. Jensen. Visual simulation of smoke. *SIGGRAPH*, pages 15–22, 2001. [35](#), [36](#)
- [39] Z.-G. Feng and E. E. Michaelides. Hydrodynamic force on spheres in cylindrical and prismatic enclosures. *International Journal of Multiphase Flow*, 28(3):479–496, 2002. [26](#)
- [40] B. Fierz. Real-time fluid simulations with wavelet turbulence. Master’s thesis, ETH Zrich, 2008. [37](#)
- [41] B. Finkbeiner, U. R. Alim, D. V. D. Ville, and T. Möller. High-quality volumetric reconstruction on optimal lattices for computed tomography. *Computer Graphics Forum*, 28(3):1023–1030, 2009. [8](#)
- [42] B. Finkbeiner, A. Entezari, D. V. D. Ville, and T. Möller. Efficient volume rendering on the body centered cubic lattice using box splines. *Computers & Graphics*, 34(4):409–423, 2010. [8](#), [138](#)
- [43] U. Frisch, B. Hasslacher, and Y. Pomeau. Lattice-gas automata for the navier-stokes equation. *Physical Review Letter*, 56(14):1505–1508, 1986. [20](#)
- [44] M. Hadwiger, J. M. Kniss, C. Rezk-salama, and D. Weiskopf. *Real-time Volume Graphics*. A K Peters, 2006. [62](#), [112](#)
- [45] M. Hadwiger, C. Sigg, H. Scharsach, K. Bhler, and M. Gross. Real-time ray-casting and advanced shading of discrete isosurfaces. *Computer Graphics Forum*, 24(3):303–312, 2005. [112](#)

## BIBLIOGRAPHY

---

- [46] S. Hauswiesner, D. Kalkofen, and D. Schmalstieg. Multi-frame rate volume rendering. *Eurographics Symposium on Parallel Graphics and Visualization*, pages 19–26, 2010. [15](#)
- [47] A. Hogue, M. Robinson, M. R. Jenkin, and R. S. Allison. A vision-based head tracking system for fully immersive displays. *Proceedings of the Workshop on Virtual Environments*, pages 179–187, 2003. [14](#), [58](#), [98](#)
- [48] C.-F. Hollemeersch, B. Pieters, P. Lambert, and R. V. d. Walle. *Accelerating Virtual Texturing Using CUDA*, pages 623–642. A K Peters, 2010. [80](#)
- [49] L. Hong, S. Muraki, A. Kaufman, D. Bartz, and T. He. Virtual voyage: interactive navigation in the human colon. *SIGGRAPH*, pages 27–34, 1997. [14](#), [67](#), [92](#)
- [50] W. Hong, J. Wang, F. Qiu, A. Kaufman, and J. Anderson. Colonoscopy simulation. *Proc. SPIE Medical Imaging*, 6511, 2007. [14](#), [92](#)
- [51] S. Hou, J. D. Sterling, S. Chen, and G. D. Doolen. A lattice Boltzmann subgrid model for high Reynolds number flows. *Fields Institute Communications*, 6:151–166, 1996. [12](#)
- [52] G. Humphreys, M. Houston, R. Ng, R. Frank, S. Ahern, P. Kirchner, and J. T. Klosowski. Chromium: A stream processing framework for interactive rendering on clusters. *ACM Transactions on Graphics*, 21(3):693–702, 2002. [61](#)
- [53] J. Inoue and A. J. Stewart. Multiresolution sphere packing tree: A hierarchical multiresolution 3d data structure. *ACM Symposium on Solid and Physical Modeling*, pages 367–373, 2008. [45](#)
- [54] V. Interrante, B. Ries, and L. Anderson. Seven league boots: A new metaphor for augmented locomotion through moderately large scale immersive virtual environments. *IEEE Symposium on 3D User Interfaces*, pages 167–170, 2007. [146](#)
- [55] H. Iwata and Y. Yoshida. Path reproduction tests using a torus treadmill. *Presence: Teleoperators and Virtual Environments*, 8(6):587–597, 1999. [146](#)
- [56] M. Jin, J. Kim, F. Luo, and X. Gu. Discrete surface ricci flow. *IEEE Transactions on Visualization and Computer Graphics*, 14(5):1030–1043, 2008. [99](#), [100](#), [101](#), [103](#)
- [57] M. Jin, W. Zeng, F. Luo, and X. Gu. Computing Teichmüller shape space. *IEEE Transactions on Visualization and Computer Graphics*, 15(3):504–517, 2009. [100](#)
- [58] C. D. Johnson and A. H. Dachman. CT colonography: The next colon screening examination? *Radiology*, 216(2):331–341, 2000. [14](#), [67](#), [92](#)

## BIBLIOGRAPHY

---

- [59] T. A. Johnson and V. C. Patel. Flow past a sphere up to a Reynolds number of 300. *Journal of Fluid Mechanics*, 378:19–70, 1999. [26](#)
- [60] D. Kim, H. Choi, and H. Choi. Characteristics of laminar flow past a sphere in uniform shear. *Physics of Fluids*, 17(10):103602, 2005. [26](#)
- [61] M. Kim, A. Entezari, and J. Peters. Box spline reconstruction on the face-centered cubic lattice. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1523–1530, 2008. [8](#), [9](#)
- [62] T. Kim, N. Thürey, D. James, and M. Gross. Wavelet turbulence for fluid simulation. *SIGGRAPH*, pages 1–6, 2008. [37](#)
- [63] A. Knoll, Y. Hijazi, R. Westerteiger, M. Schott, C. Hansen, and H. Hagen. Volume ray casting with peak finding and differential sampling. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1571–1578, 2009. [112](#)
- [64] J. Krüger and R. Westermann. Acceleration techniques for GPU-based volume rendering. *IEEE Visualization*, pages 38–45, 2003. [40](#)
- [65] F. Labelle and J. R. Shewchuk. Isosurface stuffing: fast tetrahedral meshes with good dihedral angles. *ACM Transactions on Graphics*, 26(3):57, 2007. [46](#)
- [66] P. Lallemand and L.-S. Luo. Theory of the lattice Boltzmann method: Acoustic and thermal properties in two and three dimensions. *Physical Review E*, 68(3):036706, 2003. [12](#), [34](#), [36](#)
- [67] J. Latt. Choice of units in lattice Boltzmann simulations. Technical report, LBMethod.org, 2008. [12](#)
- [68] B. Lévy, S. Petitjean, N. Ray, and J. Maillot. Least squares conformal maps for automatic texture atlas generation. *ACM Transactions on Graphics*, 21(3):162–170, 2002. [100](#)
- [69] X. Li, X. Gu, and H. Qin. Surface matching using consistent pants decomposition. *ACM Solid and Physical Modeling Symposium*, pages 125–136, 2008. [100](#)
- [70] B. Liu, L.-Y. Wei, X. Yang, Y.-Q. Xu, and B. Guo. Nonlinear beam tracing on a GPU. Technical Report MSR-TR-2007-34, Microsoft Research, 2007. [62](#), [113](#)
- [71] P. Ljung, C. Lundström, and A. Ynnerman. Multiresolution interblock interpolation in direct volume rendering. *Eurographics*, pages 259–266, 2006. [2](#), [9](#)
- [72] H. Lorenz and J. Döllner. Real-time piecewise perspective projections. *International Conference on Computer Graphics Theory and Applications (GRAPP)*, pages 147–155, 2009. [14](#)

---

## BIBLIOGRAPHY

- [73] R. S. Maier, R. S. Bernard, and D. W. Grunau. Boundary conditions for the lattice Boltzmann method. *Physics of Fluids*, 8(7):1788–1801, 1996. [12](#)
- [74] E. C. L. Mar, B. Hamann, and K. I. Joy. Multiresolution techniques for interactive texture-based volume visualization. *IEEE Visualization*, pages 355–362, 1999. [2](#), [9](#), [47](#)
- [75] S. R. Marschner and R. J. Lobb. An evaluation of reconstruction filters for volume rendering. *IEEE Visualization*, pages 100–107, 1994. [49](#), [138](#)
- [76] R. Mei, W. Shyy, D. Yu, and L.-S. Luo. Lattice Boltzmann method for 3-D flows with curved boundary. *J. Comput. Phys.*, 161(2):680–699, 2000. 350006. [12](#), [13](#)
- [77] J. Mensmann, T. Ropinski, and K. H. Hinrichs. A GPU-supported lossless compression scheme for rendering time-varying volume data. *IEEE/EG International Symposium on Volume Graphics*, pages 109–116, 2010. [1](#)
- [78] P. Milgram and F. Kishino. Taxonomy of mixed reality visual displays. *IEICE Transactions on Information and Systems*, E77-D(12):1321–1329, 1994. [14](#)
- [79] A. Neubauer, S. Wolfsberger, M.-T. Forster, L. Mroz, R. Wegenkittl, and K. Buhler. Steps - an application for simulation of transsphenoidal endonasal pituitary surgery. *IEEE Visualization*, pages 513–520, 2004. [112](#)
- [80] NVIDIA. NVIDIA SceniX Application Acceleration Engine, 2012.  
<http://www.nvidia.com/object/scenix.html>. [62](#)
- [81] NVIDIA. Compute Unified Device Architecture Programming Guide, 2013. Version 5.  
<https://developer.nvidia.com/category/zone/cuda-zone>. [30](#), [31](#), [32](#), [33](#), [39](#)
- [82] S. G. Parker, J. Bigler, A. Dietrich, H. Friedrich, J. Hoberock, D. Luebke, D. McAllister, M. McGuire, K. Morley, A. Robison, and M. Stich. Optix: A general purpose ray tracing engine. *ACM Transactions on Graphics*, 29(4):1–13, 2010. [62](#)
- [83] T. C. Peck, H. Fuchs, and M. C. Whitton. An evaluation of navigational ability comparing redirected free exploration with distractors to walking-in-place and joystick locomotion interfaces. *IEEE Virtual Reality*, pages 55–62, 2011. [146](#)
- [84] K. Petkov, Z. Fan, F. Qiu, K. Mueller, and A. E. Kaufman. Efficient LBM flow simulation on face-centered cubic lattices. *IEEE Transactions on Visualization and Computer Graphics*, 15(5):802–814, 2009. [1](#)

## BIBLIOGRAPHY

---

- [85] E. Pietriga, O. Bau, and C. Appert. Representation-independent in-place magnification with sigma lenses. *IEEE Transactions on Visualization and Computer Graphics*, 16(3):455–467, 2010. [14](#)
- [86] S. Popov, J. Günther, H.-P. Seidel, and P. Slusallek. Stackless kd-tree traversal for high performance GPU ray tracing. *Computer Graphics Forum*, 26(3):415–424, 2007. [62](#), [113](#)
- [87] W. Qiao, D. S. Ebert, A. Entezari, M. Korkusinski, and G. Klimeck. Volqd: Direct volume rendering of multi-million atom quantum dot simulations. *IEEE Visualization*, pages 319–326, 2005. [8](#), [52](#)
- [88] F. Qiu, F. Xu, Z. Fan, N. Neophytos, A. Kaufman, and K. Mueller. Lattice-based volumetric global illumination. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1576–1583, 2007. [7](#), [38](#)
- [89] F. Qiu, B. Zhang, K. Petkov, L. Chong, A. Kaufman, K. Mueller, and X. D. Gu. Enclosed five-wall immersive cabin. *Proceedings of the 4th International Symposium on Advances in Visual Computing (ISVC)*, pages 891–900, 2008. [13](#), [58](#), [59](#)
- [90] F. Qiu, Y. Zhao, Z. Fan, X. Wei, H. Lorenz, J. Wang, S. Yoakum-Stover, A. Kaufman, and K. Mueller. Dispersion simulation and visualization for urban security. *IEEE Visualization*, pages 553–560, 2004. [67](#)
- [91] R. Raskar, J. v. Baar, P. Beardsley, T. Willwacher, S. Rao, and C. Forlines. iLamps: geometrically aware and self-configuring projectors. *ACM SIGGRAPH Courses*, 2005. [100](#)
- [92] G. Robertson, M. Czerwinski, and M. v. Dantzig. Immersion in desktop virtual reality. *Proceedings of the 10th Annual ACM Symposium on User Interface Software and Technology*, pages 11–19, 1997. [2](#), [146](#)
- [93] H. Scharsach, M. Hadwiger, A. Neubauer, S. Wolfsberger, and K. Bhler. Perspective isosurface and direct volume rendering for virtual endoscopy applications. *IEEE-VGTC Symposium on Visualization*, pages 315–322, 2006. [112](#)
- [94] A. Selle, R. Fedkiw, B. Kim, Y. Liu, and J. Rossignac. An unconditionally stable MacCormack method. *J. Sci. Comput.*, 35(2-3):350–371, 2008. [36](#)
- [95] C. Sigg and M. Hadwiger. *Fast Third-Order Texture Filtering*, chapter 20, pages 313–330. Addison-Wesley, 2004. [49](#), [112](#), [138](#)
- [96] C. Sigg, M. Hadwiger, M. Gross, and K. Bhler. Real-time high-quality rendering of isosurfaces. Technical report, VRVis Research Center,ETH Zrich, 2004. [112](#)

## BIBLIOGRAPHY

---

- [97] M. Simmons. *Tapestry: An Efficient Mesh-based Display Representation for Interactive Rendering*. PhD thesis, University of California at Berkeley, 2001. [15](#)
- [98] P. Sitthi-amorn, J. Lawrence, L. Yang, P. V. Sander, and D. Nehab. An improved shading cache for modern GPUs. *ACM SIGGRAPH/EUROGRAPHICS Symposium on Graphics Hardware*, pages 95–101, 2008. [15](#)
- [99] M. Slater, M. Usoh, and A. Steed. Taking steps: The influence of a walking technique on presence in virtual reality. *ACM Transaction on Computer-Human Interaction*, 2(3):201–219, 1995. [146](#)
- [100] M. Spindler, M. Bubke, T. Germer, and T. Strothotte. Camera textures. *Proceedings of the 4th International Conference on Computer Graphics and Interactive Techniques in Australasia and Southeast Asia (GRAPHITE)*, pages 295–302, 2006. [110](#)
- [101] B. Springborn, P. Schröder, and U. Pinkall. Conformal equivalence of triangle meshes. *ACM Transactions on Graphics*, 27(3):1–11, 2008. [100](#)
- [102] O. G. Staadt, J. Walker, C. Nuber, and B. Hamann. A survey and performance analysis of software platforms for interactive cluster-based multi-screen rendering. *Workshop on Virtual Environments*, pages 261–270, 2003. [61](#)
- [103] J. Stam. Stable fluids. *SIGGRAPH*, pages 121–128, 1999. [35](#), [36](#)
- [104] K. Stephenson. *Introduction To Circle Packing*. Cambridge University Press, 2005. [101](#)
- [105] E. A. Suma, G. Bruder, F. Steinicke, D. M. Krum, and M. Bolas. A taxonomy for deploying redirection techniques in immersive virtual environments. *IEEE Virtual Reality Workshops*, pages 43–46, 2012. [147](#)
- [106] T. Theußl, T. Möller, and M. E. Gröller. Optimal regular volume sampling. *IEEE Visualization*, pages 91–98, 2001. [7](#)
- [107] N. Thuerey and U. Ruede. Free surface Lattice-Boltzmann fluid simulations with and without level sets. *Vision, Modeling, and Visualization Conference*, pages 199–207, 2004. [34](#)
- [108] W. P. Thurston. Geometry and topology of three-manifolds. *Princeton Lecture Notes*, 1976. [99](#), [101](#)
- [109] J. Toelke and M. Krafczyk. Towards three-dimensional teraflop CFD computing on a desktop PC using graphics hardware. Technical report, Institute for Computational Modeling in Civil Engineering, TU Braunschweig, 2008. [30](#)

## BIBLIOGRAPHY

---

- [110] M. Trapp and J. Döllner. Generalization of single-center projections using projection tile screens. *Communications in Computer and Information Science*, 24:55–69, 2009. [14](#)
- [111] M. Usoh, K. Arthur, M. C. Whitton, R. Bastos, A. Steed, M. Slater, and J. Frederick P. Brooks. Walking > walking-in-place > flying, in virtual environments. *SIGGRAPH*, pages 359–364, 1999. [146](#)
- [112] University of Texas "Stallion" Powerwall. <http://www.tacc.utexas.edu/resources/visualization/>. [70](#)
- [113] E. Velázquez-Armendáriz, E. Lee, B. Walter, and K. Bala. Implementing the render cache and the edge-and-point image on graphics hardware. *Graphics Interface*, pages 211–217, 2006. [15](#)
- [114] R. Verberg and A. J. C. Ladd. Lattice-Boltzmann model with sub-grid-scale boundary conditions. *Physical Review Letters*, 84(10):2148–2151, 2000. [12](#)
- [115] M. Šrámek and A. Kaufman. Object voxelization by filtering. *IEEE Symposium on Volume Visualization*, pages 111–118, 1998. [53](#)
- [116] E. Vuçini, T. Möller, and M. E. Gröller. Efficient reconstruction from non-uniform point sets. *The Visual Computer*, 24(7):555–563, 2008. [2](#), [44](#)
- [117] I. Wald and V. Havran. On building fast kd-trees for ray tracing, and on doing that in  $O(N \log N)$ . *IEEE Symposium on Interactive Ray Tracing*, pages 61–69, 2006. [53](#)
- [118] B. Walter, G. Drettakis, and D. P. Greenberg. Enhancing and optimizing the render cache. *Eurographics Workshop on Rendering*, pages 37–42, 2002. [15](#)
- [119] B. Walter, G. Drettakis, and S. Parker. Interactive rendering using the render cache. *Rendering Techniques*, 10:235–246, 1999. [15](#)
- [120] L. Wang, Y. Zhao, K. Mueller, and A. Kaufman. The magic volume lens: An interactive focus + context technique for volume rendering. *IEEE Visualization*, pages 367–374, 2005. [14](#)
- [121] Y. Wang, X. Yin, J. Zhang, X. Gu, T. F. Chan, P. M. Thompson, and S.-T. Yau. Brain mapping with the ricci flow conformal parameterization and multivariate statistics on deformation tensors. *2nd MICCAI Workshop on Mathematical Foundations of Computational Anatomy*, pages 36–47, 2008. [100](#)
- [122] X. Wei, Y. Zhao, Z. Fan, W. Li, F. Qiu, S. Yoakum-Stover, and A. Kaufman. Lattice-based flow field modeling. *IEEE Transactions on Visualization and Computer Graphics*, 10(6):719–729, 2004. [34](#)

## BIBLIOGRAPHY

---

- [123] S.-B. Wen and C.-L. Lai. Theoretical analysis of flow passing a single sphere moving in a micro-tube. *Proceedings of the Royal Society. Mathematical, Physical and Engineering Sciences*, 459(2030):495–526, 2003. [26](#)
- [124] R. M. Wham, O. A. Basaran, and C. H. Byers. Wall effects on flow past solid spheres at finite Reynolds number. *Industrial & Engineering Chemistry Research*, 35(3):864–874, 1996. [26](#)
- [125] D. A. Wolf-Gladrow. *Lattice-Gas Cellular Automata and Lattice Boltzmann Models: An Introduction*. Lecture Notes in Mathematics. Springer-Verlag, 1st edition, 2000. [7](#), [10](#), [11](#), [12](#), [18](#), [19](#), [24](#), [25](#), [45](#)
- [126] C. Woolley, D. Luebke, B. Watson, and A. Dayal. Interruptible rendering. *Symposium on Interactive 3D Graphics*, pages 143–151, 2003. [15](#)
- [127] Y. Yang, J. X. Chen, and M. Beheshti. Nonlinear perspective projections and magic lenses: 3D view deformation. *Computer Graphics*, 25(1):76–84, 2005. [14](#)
- [128] Y.-L. Yang, J. Kim, F. Luo, S.-M. Hu, and X. Gu. Optimal surface parameterization using inverse curvature map. *IEEE Transactions on Visualization and Computer Graphics*, 14(5):1054–1066, 2008. [100](#)
- [129] W. Zeng, J. Marino, K. C. Gurijala, X. Gu, and A. Kaufman. Supine and prone colon registration using quasi-conformal mapping. *Transactions on Visualization and Computer Graphics*, 16(6):1348–1357, 2010. [93](#)
- [130] Y. Zhao, Y. Han, Z. Fan, F. Qiu, Y.-C. Kuo, A. Kaufman, and K. Mueller. Visual simulation of heat shimmering and mirage. *IEEE Transactions on Visualization and Computer Graphics*, 13(1):179–189, 2007. [34](#), [36](#), [37](#)
- [131] Y. Zhao, F. Qiu, Z. Fan, and A. Kaufman. Flow simulation with locally-refined LBM. *ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, pages 181–188, 2007. [7](#)
- [132] T. Zhu, R. Wang, and D. Luebke. A GPU-accelerated render cache. *Pacific Graphics*, 2005. [15](#)
- [133] M. Zlatuska and V. Havran. Ray tracing on a GPU with CUDA – comparative study of three algorithms. *Journal of WSCG*, 18(1-3):69–75, 2010. [62](#), [113](#)