New directions in four-dimensional mathematical visualization

Yan, Xiaoqi

2015

Yan, X. (2015). New directions in four-dimensional mathematical visualization. Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/65695>

<https://doi.org/10.32657/10356/65695>

NANYANG
TECHNOLOGICAL
UNIVERSITY

New Directions in Four-Dimensional Mathematical Visualization

A thesis submitted
for a degree of Doctor of Philosophy
by

YAN Xiaoqi

School of Computer Engineering
Nanyang Technological University

November 08, 2015

Abstract

Four-dimension visualization is an interesting topic as we live in a 3D space, we do not have the chance to directly “see,” manipulate or “touch” 4D objects. Four-dimensional visualization is an important topic since mathematicians have long been wondering how to visualize beautiful geometries such as knotted spheres, quaternions, and Calabi-Yau space cross-sections, while physicists have been wanting to visualize the hypersphere, which is Einstein’s universe. However, only computer graphics can be used to create 4D visualization tools which have the power to accurately represent objects for which physical models are difficult or impossible to build, and which have the ability to allow the user to interact with simulated worlds.

Without 4D visualization tools we cannot even plot a complex-valued function with only one independent variable. Graphics pioneers did great work on 4D visualization, but it is not enough: the major problems to be addressed in 4D visualization (how to present these models to the eyes and how to enhance the user’s intuitive experience of the abstract geometric world that we are trying to understand) have still not been completely solved. In more detail, projecting a 4D object onto 3D space is actually a dimension reduction. Certain geometrical information, such as symmetry and curvature, is unavoidably lost after the projection because we rely on the 3D projection view to explore the geometry that actually exists in four dimensions. Moreover, projecting the 4D surface onto a 3D space may lead to false occlusion. Although a 4D surface does not have any intersections in 4D space, projecting it onto a 3D space may cause intersecting (occlusion) lines. In addition, the user interaction is not intuitive or efficient, and the potential of improving accessibility of 4D visualization system for public users is virtually unexplored. This project aims to explore new research directions that address the above questions.

In particular, I have explored the following new research directions:

- Displaying 4D objects by new visualization methods, such as parallel coordinates;
- Designing more useful visual cues to enhance visualization;
- Developing new interaction methods for users to manipulate 4D objects;
- Performing evaluations from the aspect of human perception;
- And lastly, developing a set of practical applications that can be ported to mobile devices, thus serving as educational tools for audiences from the general public to learn and explore geometries defined in four-dimensional space.

In accordance with the above research directions, this thesis proposed a comprehensive 4D visualization framework. Instead of replacing the conventional 4D visualization framework, the novel techniques proposed in this work are well designed and embedded in a typical 4D visualization framework. Based on the framework, a GPU based interactive 4D visualization system is implemented, and its light-weight version application can be run on mobile devices. Our comprehensive system improves on the conventional system in terms of revealing clear geometric information, enhancing depth perception, providing more appropriate visual cues, supporting intuitive and efficient user controls, providing greater flexibility for the running platform, and improving accessibility for public users. In this work, we also provide several user scenarios to illustrate the usage of the system, and we perform user-studies to evaluate the system.

Acknowledgments

Above all, I would like to thank my supervisor Prof. Chi-Wing Fu for continuous support of my Ph.D. study and research. This thesis would not have been possible without the guidance and patience of him, not to mention his advice and unsurpassed knowledge of computer graphics and human computer interaction. I would also like to thank Prof. Andrew J. Hanson, an expert in the visualization of abstract concepts in mathematics and physics, exploitation of quaternions, and the design of user interfaces for visualization applications, for his precious advice on various projects. Besides the valuable research skills that I learnt from him, his personal fascination also affects me to be a kind and happy person. I would grateful acknowledge Prof. Goh Wooi Boon, an expert in human computer interaction, for his precious advice and unlimited encouragement on various projects.

In addition, I would like to acknowledge my colleagues, especially Mr. William Lai, Dr. Peng Song, Dr. Xiaopei Liu, Mr. Pradeep Kumar, Mr. Niu Junpeng, Mr. Wei Zeng, Mr. Fuwen Tan, Ms. Mengyao Zhao for their assistance and constant support on many research projects. Meanwhile, many thanks to my friends, Dr. Qian Sun, Dr. Min Meng, Dr. Xiaoqun Wu, Ms. Anran Wang, Dr. Long Zhang, Mr. Xiaoning Wang, Dr. Minqi Zhang, Dr. Xiang Ying, Dr. Dayong Wang, Dr. Jiazhi Xia, for their kind support and assistance.

Last but not least, I would like to thank my parents and my grandparents for their constant supporting and encouragement. Although I was away from home for this study, I always feel their care and love.

Contents

Abstract	i
Acknowledgments	iii
List of Figures	vii
List of Tables	xiii
1 Introduction	1
1.1 Background	1
1.2 Research Objectives	10
1.3 Research Contributions	12
1.4 Thesis Organization	15
2 Related Work	16
2.1 Four-dimensional geometry visualization	16
2.2 Visualizing the high-dimensional data	23
2.3 Illustrative Visualization	26
2.4 Emerging User Interaction Technologies	28
2.5 Summary	31
3 A Comprehensive Four-dimensional Mathematical Visualization Framework	32
3.1 Introduction	32
3.2 The Conventional 4D Imaging Pipeline	33
3.3 Comprehensive Four-dimensional Visualization Pipeline	34
3.4 4D Visualization Enhancement	36
3.5 Summary	39

4	Dual-view 4D Visualization	40
4.1	Introduction	41
4.2	Related Work about Parallel Coordinate	43
4.3	Overview	47
4.4	Geometric Meanings of Visual Signatures	49
4.5	Pre-processing	51
4.6	Interactive Visualization	53
4.7	Interactive Exploration	58
4.8	Implementation Detail	59
4.9	Summary	60
5	Enhancing the 4D visualization by Volumetric Halos	61
5.1	Introduction	62
5.2	Enhancing Visualization with Halos	65
5.3	Overview of Our Approach	66
5.4	The Illustrative Visualization Pipeline	68
5.5	Extensions	74
5.6	Implementation and Results	77
5.7	Summary	85
6	Exploring 4D Models by Multitouch Interface	86
6.1	Introduction	87
6.2	Multitouch Based Exploration Methods	89
6.3	Exploration of 4D Geometry.	92
6.4	4D Interface Design and Implementation	98
6.5	Strategies of the Interface Design	102
6.6	4D Learning Scenario	103
6.7	Remote Control.	115
6.8	Summary	115
7	User Evaluation	116
7.1	Introduction	116

7.2	Review of the 4D Visualization Framework	117
7.3	Understanding the Geometric Properties by Dual-view Visualization	119
7.4	Looking for False Intersections by Halo Illustration	120
7.5	Rolling a 4D Die by Multitouch Interface	121
7.6	Summary	125
8	Conclusion and Future Work	126
8.1	Conclusion	126
8.2	Limitation and Future Work	127
	References	130
	Publications	143

List of Figures

1.1	Typical 4D geometries: (a) A 4D-hypercube rendered with tube edges and transparent surfaces; this figure is courtesy of Hise [68]. (b) A 4D-Knotted sphere rendered with screen door effect; this figure is courtesy of Chu et al. [27]. (c) A 4D-Fermat surface with color coded patches; this figure is courtesy of Hanson et al. [62]. (d) A Klein bottle rendered with 4D depth coding and ribbon slicing; this figure is courtesy of Banks [7].	2
1.2	3D printed 4D models: (a)-(b) Two printed 3D projection of a 4D complex surface; model courtesy of Stewart Dickson, and this figure is courtesy of Hanson and Zhang [63]. (c) A printed 3D projection of Klein bottle with its two halves; model courtesy of Stewart Dickson [34].	4
1.3	4D visualization systems: (a) MeshView by Hanson et al. at Indiana University [60]; (b) Geomview by the Geometry Center of the University of Minnesota [124].	6
2.1	Sequence of rotating 4D hypercube, this figure is courtesy of Noll in 1967 [105].	17
2.2	The visualization result of two-dimensional surfaces in four-dimensional space from the work of Hoffmann and Zhou [70]: (a) Bracket viewed from $\theta = (0,0,0,45,60,0)$ degrees and $r = (0,0)$ and shaded by color scale representing w-values; (b) Offset curve of $y - x^2 = 0$ traced in 4-space with highlighted silhouette curves; (c) Intersection of cylinder and moving sphere with the same radius.	18

2.3	Illuminated the 4D embedded flat torus (top) and the 4D knotted sphere (bottom) by 4D light sources; top: torus is rotated in zw plane; bottom: from left to right: a large opacity value is used in the volume rendering so that the model appears like an opaque surface; a screen door effect is added by applying a checkerboard deletion pattern in the parametric space; last, the 4D lighting direction is animated, this figure is courtesy of Chu et al. [27].	19
2.4	The illumination result of a 4D A bumpy torus and a fur-textured vector field protruding from it, this figure is courtesy of Banks [8]: (a) The torus is rendered with illumination only; (b) The torus is rendered with illumination together with a self-shadow effect; (c) The torus is rendered with illumination, self-shadow effect, and attenuated lighting.	20
2.5	(a,b) Tightening a 4D knotted sphere by pulling on the 3D shadow. (c,d,e) Pushing on the knotted sphere to loosen it, this figure is courtesy of Zhang and Hanson [140].	21
2.6	Interactive visualization the 4D subset for the Julia set in nested coordinate system, this figure is courtesy of Miller and Gavosto in 2004 [102]: (a) View inside the immersive visualization probe view; (b) Nested-coordinate-system view.	22
2.7	A scatterplot matrix for 5-dimensional data of 400 automobiles, this figure is courtesy of Becker et al. [10].	24
2.8	Left: radial coordinates, this figure is courtesy of Hoffman [69]. Right: group of star glyphs, this figure is courtesy of Chambers et al. [85].	24
2.9	(a) Stick figure family; (b) 5D image data using stick figures; (c) Part of (b) in original size, this figure is courtesy of Chambers et al. [85].	25
2.10	Left: pixel-based visualization of 6-dimensional data, this figure is courtesy of Keim et al. [90]. Right: n -vision, this figure is courtesy of Feiner et al. [43]. . .	25
2.11	Volume-based illustrative rendering. Left: Volume-based illustrative rendering for a human head, yellow color indicates low distance to the vessels, this figure is courtesy of Rautek et al. [111]. Right: Illustrative annotations of direct volume rendered carp by VolumeShop, this figure is courtesy of Bruckner et al. [18].	27

2.12	From left to right: Wiimote, Kinect, multi-touch handheld devices: iPad.	28
3.1	Overview of the eight major steps in our visualization pipeline: 1) Surface construction to generate a set of grid points; 2) Blue-noise surface sampling to obtain a sample point set; 3) Interactive 4D transformation (rotation and projection); 4) Parallel-coordinates plots with visual signatures; 5) Illustrative visualization; 6) Enhanced visual signatures (e.g., emphasize high curvature areas); 7) 4D geometry manipulation; And 8) interactive brushing.	35
4.1	Left: discrete parallel coordinates. Right: continuous parallel coordinates. This figure is courtesy from Heinrich and Weiskopf [66].	44
4.2	Left: illustrative parallel coordinates, this figure is courtesy from McDonnell and Mueller [101]. Right: 3D parallel coordinates, this figure is courtesy from Johansson et al. [86].	45
4.3	A combination of parallel coordinates and star glyphs, this figure is courtesy from Johansson et al. [42].	46
4.4	Changes in parallel-coordinates plots upon basic transformations, as illustrated by a 2D ELLIPSE.	48
4.5	Example correspondences between geometrical properties and visual signatures in parallel-coordinates plots: two simple geometric models are used: a 2D ELLIPSE ((a) and (b)) and a 3D ELLIPTICAL CYLINDER ((c) and (d)). . . .	49
4.6	Comparing different sampling approaches to generate continuous parallel-coordinates plots on the 4D KNOTTED SPHERE model. All approaches use 3000 sample points in 4D, and blue-noise sampling produces the highest-quality visualization. . . .	50
4.7	Interactive parallel-coordinates visualization upon dynamic subspace rotation in 4-dimensions. The three 4D geometric models used here are TORUS, TORUS KNOT-LIKE, and FERMAT SURFACE (from left to right column).	52
4.8	Interactive parallel-coordinates visualization upon dynamic subspace rotation in N -dimensions of 6D VERONESE model.	55
4.9	Using <i>AND</i> and <i>OR</i> brushing modes on the same set of brush points. We highlight related surface regions in red.	56

4.10	Two example scenarios of depicting curvature (on the 3D projection view and the parallel-coordinates plots) and brushing the curvature feature lines in the plots. Left: PILLOW model and Right: KLEIN2 model.	58
4.11	Other than mean curvature (see Figure 4.10), we can also choose to show min or max curvature.	60
5.1	Surface patches at different 4D depths in w projected to 3D image: (a) Oblique 4D view projected along the direction $(x, y, z, w) = (1, 0, 0, 1)$, which shows the physical separation in w ; (b) 4D view along $(0, 0, 0, 1)$, producing 3D self-intersection, with no halo; (c) 3D self-intersecting view enhanced by depth-dependent 4D volumetric halo method.	63
5.2	Enhancing visualization with halos: (a) Emphasize the occlusions of geometries by halo technique, this figure is courtesy from Appel et al. [2]. (b) Visualize magnetic field data with halo enhanced rendering, this figure is courtesy from Schussman et al. [115]. (c) Enhance volume data visualization by halo effect, this figure is courtesy from Bruchner et al. [20]. (d) Rendering dense line data with depth-dependent halos, this figure is courtesy from Everts et al. [41].	65
5.3	Halo illustrations for surfaces in 4D: (a) the surfaces are first projected from 4D space to 3D screen space; (b) view-oriented thickening is then applied to generate tetrahedra for the thickened volumes; (c) we voxelize the tetrahedral volumes and fill a 3D depth buffer with 4D depth values; (d) the final halo illustrations are produced by rendering the surfaces with an occlusion test; (e) other visualization effects, such as heuristic lighting, can also be supported in our visualization pipeline.	66
5.4	Modeling halos from a triangle primitive: (a) the triangle is first transformed and projected into 3D screen space; (b) we then thicken the triangle by offsetting it to both positive and negative sides of $\vec{n}$; (c) we decompose the extruded volumes into tetrahedra; (d) the tetrahedra are voxelized by planar slicing; (e) the halo generation algorithm is applied and the volume slice is rendered.	70
5.5	Thickening process for curves in 4D: the line segment (left) is first thickened into a tube (middle), then corresponding pizza slices at the two ends of the tube are connected into triangular prisms, each of which is further decomposed into three tetrahedra (right).	75

5.6	Resolving face mismatches between neighboring triangular prisms: (a) a common type of face mismatch; (b) fixable by reversing the construction of the red line in the middle rectangle; (c) another type of face mismatch that is not fixable by (b); (d) fixable by adding new vertices and reconstructing the red prism with a new decomposition.	78
5.7	SPUN TREFOIL: (a) traditional rendering; (b) translucent rendering; (c) result from [27]; (d) our result.	79
5.8	QUADRATIC FERMAT: (a) volume rendered constant color effect and pinch points marked with white arrows; (b) texturing effect.	81
5.9	CALABI-YAU QUARTIC CROSS-SECTION: (a) volume rendering effect; (b) color-coding with 4D depth values.	82
5.10	The degree two $CP(2)$ PRODUCT POLYNOMIAL: cross-eyed stereo.	82
5.11	TWIST-SPUN-TREFOIL KNOT: cross-eyed stereo and halo boundaries highlighted in green.	83
5.12	Halo illustrations of 4D curves: (a) curve from planar slice of the SPUN TREFOIL (color-coded with red denoting 4D depth values nearest to the viewpoint); (b) curves on a degree two FERMAT surface (color-coded with red denoting the 3D depth values nearest to the viewpoint); (c) anaglyphic stereo view of (b), for use with red-blue glasses.	83
6.1	Manipulate 3D objects on a multi-touch screen by the screen-space formulation method on a terrain navigation system, from top to bottom: single touch point, two touch points, four touch points [112].	90
6.2	Left: a 3D transformation widgets for multi-touch interface [29]. Right: a multi-touch interface for 3D animation design [93].	90
6.3	Left: rotate a 3D virtual object by a mobile device [89]. Right: interact with fine-grained details of documents by a mobile device [97].	91
6.4	Green arrows are forward-facing normals. Left: 3D Back-face culling. Right: 4D Back-face culling.	94
6.5	(a) The geometry of the 3D Rolling Ball controller on a 2D screen. (b) The geometry of the 4D Rolling Ball controller in a 3D screen.	96
6.6	Multitouch gestures.	98

6.7	The construction of a hypercube or tesseract by building up from lower dimensions.	104
6.8	The basic geometry, visual presentation, and face-numbering rules of a right-handed 3D die.	106
6.9	The basic geometry and visual presentation of a 4D die.	108
6.10	Illustrative rolling actions of the 3D die and the 4D die.	109
6.11	Rotation sequence of the 4D die.	110
6.12	How our two-finger multitouch controls produce 4D rotations in the w_x , w_y , and w_z planes.	111
6.13	The three alternate sets of interlocking pairs of tori that can be seen while rotating the hypercube in each of the three planes w_x , w_y , and w_z . Each torus on the left forms a revolving belt surrounding its partner torus that threads itself through the center.	112
6.14	A 3D die shown as a 3D box with square “3” as the lid, analogous to a 4D die shown as a 4D box with cube “4” as the lid. The image of the 3D die is projected to 2D, and the lid lies on top of all the other 5 faces, while the image of the 4D die is projected to 3D, and the image of the cubic lid occludes the images of all the other seven cubes in the projection.	113
6.15	Rotating various 4D objects via remote display control.	114
7.1	Eight major steps in comprehensive 4D visualization framework.	117
7.2	Summary of the user study results. Mouse+keyboard: asymptotic performance (c) 5.96 ± 0.11 seconds per trial, exponential fit coefficient (b) 0.0696 ± 0.004 per trial. Multitouch: asymptotic performance (c) 3.82 ± 0.09 seconds per trial, exponential fit coefficient (b) 0.1088 ± 0.008 per trial.	124

List of Tables

5.1	Comparison of halo illustrations between 3D and 4D graphics.	67
5.2	Performance of our illustrative visualization sub-pipeline (FPS).	80
6.1	The three pairs of tori, each composed of two complementary sequences of four cubic hypercube faces, that appear as we mix each of the three Cartesian axes with the w -axis by a simple rotation.	112

Chapter 1

Introduction

1.1 Background

Mathematical visualization is the process of producing a graphical representation of abstract mathematical formulas and geometries. This research area mainly focuses on the representation of statistic mathematical models such as the torus, and the exploration of mathematical phenomena such as eversion of a sphere. Before the appearance of high-performance interactive computer graphics systems, this procedure could only be realized with hand-drawn illusions and diagrams. With the aid of a computer graphics system, the mathematical formulas and geometries can be visualized in a more intuitive and holistic manner.

As one of the branches of mathematical visualization, four-dimensional mathematical visualization has focused on providing a visualization approach in order to intuitively present four-dimensional geometries to the human eyes. Although our physical world is only three-dimensional (3D), in the virtual world of computer modeling we can construct any kind of higher-dimensional geometry that we are capable of imagining and writing down. In a 3D Cartesian coordinate system, there are three coordinate axes, X , Y , and Z , while the standard basis consists of three vectors $(1,0,0)$, $(0,1,0)$ and $(0,0,1)$. An arbitrary direction can be written uniquely as a linear combination of these three vectors. In the 4D space, there is an

Figure 1.1: Typical 4D geometries: (a) A 4D-hypercube rendered with tube edges and transparent surfaces; this figure is courtesy of Hise [68]. (b) A 4D-Knotted sphere rendered with screen door effect; this figure is courtesy of Chu et al. [27]. (c) A 4D-Fermat surface with color coded patches; this figure is courtesy of Hanson et al. [62]. (d) A Klein bottle rendered with 4D depth coding and ribbon slicing; this figure is courtesy of Banks [7].

additional axis, W , which means that the standard basis consists of four vectors $(1,0,0,0)$, $(0,1,0,0)$, $(0,0,1,0)$ and $(0,0,0,1)$. Similarly, an arbitrary 4D vector can be written uniquely as a linear combination of these four vectors.

However, constructing understandable and meaningful representations for 4D geometries can be extremely challenging due to the limitations of human spatial perception. The main goal of the research thesis is to explore the visualization techniques from a new direction and incorporate them into a 4D visualization framework so that this framework can be used to interactively construct meaningful representations for 4D geometries to enhance the human perception of 4D geometries.

Why Visualize the 4D World

Mathematicians have long been wondering how to visualize 4D geometry and even higher dimensional geometry since they always have to deal with the 4D dimensional models, e.g., 4D-hypercube (Figure 1.1a), 4D-Knotted sphere (Figure 1.1b), 4D-Fermat surface (Figure 1.1c), Klein bottle (Figure 1.1d), and projective plane embeddings (see, e.g., Hilbert and CohnVossen [67]). However, it is not easy to explore and understand such models without visualizing them.

It is already known that data plotting is a very useful way to understand a function. A real-valued function of one independent variable can be plotted in 2D; a real-valued function of two independent variables can be plotted in 3D. However, it is not straightforward to plot a complex function, even for the simplest function which has only one independent variable. A complex variable has two parts, the real part and the imaginary part, so we need a two-dimensional graph to draw even a single variable. This means that we need 4D visualization if we want to plot to understand a complex-valued function even with only one single complex argument. Moreover, to emphasize the importance of high-dimensional visualization, a symposium called Hypergraphics was held in 1978. This symposium brought together the representation technologies of high-dimensional geometries in different research areas, such as art, architecture, mathematics, and computer science. A large variety of representations of high-dimensional objects were published in the proceeding of this symposium.

In physics the space-time model is also four-dimensional. In this model, the three spatial dimensions together with the fourth dimension, which is generally taken to be time, form a space-time continuum. We can visualize it either by examining the sequence of 3D space or by treating time geometrically so that a time interval can be represented as the distance between two 3D points. Moreover, visualizing 4D geometry is an important step towards gaining intuition about important scientific concepts such as Einstein's theories of special and general relativity. More generally, all the 3D scalar function with three variables $w = f(x, y, z)$ e.g., the time-space model (w is the time), and the temperature distribution throughout the 3D space (w is the temperature), can all be treated as 4D data. The three variables x, y, z are three independent dimension, whereas w is the scalar value, it depends with x, y, z . The techniques of visualizing 4D geometry can also be adapted to such visualization areas. Hanson and Heng [58] successfully generalized a large families of 4D geometry visualization techniques to the 3D scalar function visualization area.

Figure 1.2: 3D printed 4D models: (a)-(b) Two printed 3D projection of a 4D complex surface; model courtesy of Stewart Dickson, and this figure is courtesy of Hanson and Zhang [63]. (c) A printed 3D projection of Klein bottle with its two halves; model courtesy of Stewart Dickson [34].

Other than academic usage, 4D visualization can also serve as a tool for the general public to understand the 4D world. For example, it can be used as a teaching tool in an educational scenario. It can also be used to produce educational films and videos, or as a model generator for 3D printers. For example, the 3D shadow of 4D geometry can be printed by a 3D printer. Thus 4D visualization can provide the general public with a promising solution for gaining physical tactile feelings of the 4D geometries in the 3D world. Figure 1.2 shows an early attempt of physically actual 3D printed 4D models. Figure 1.2(a)-(b) shows two different views of a 3D printed complex surface model; the model is courtesy of Stewart Dickson. Figure 1.2(c) shows a Klein bottle and its two halves produced by Stereolithograph.

As the first dimension beyond our physical three-dimensional space, the research work of interaction and visualization technologies of the fourth-dimension can serve as the start point of high-dimensional visualization and has the potential to be further generate to the higher dimensions. The major missions of 4D visualization are basically how to produce a tangible experience of 4D geometries and how to provide an intuitive yet geometrically correct visualization procedure to enhance the mental model of general users. In accordance with these

major missions, the research works of 4D visualization take two basic directions: graphical representation and interaction.

The research on graphical representation in the 4D visualization area has focused on the methods to adapt and tune a 3D graphics display system for the display of 4D geometries. Projecting high dimensional objects into 3D may cause them to lose some important geometric properties. Since we cannot directly interact with the geometries embedded in four dimensional space, visualizing and studying these geometries requires the projection to 3D space; thus, the visualization is indirect. Some geometry features such as self intersections and curvatures are difficult to distinguish and understand. Although we can interactively rotate the geometry and observe dynamic changes in the projected geometry, without visual cues the visualization is limited to the 3D projection view, and such a view is highly sensitive to the choice of viewing direction.

The second family of 4D visualization research works focuses on interaction. This category of research explores the potential of exploiting emerging interaction techniques to provide an interactive interface for users to understand 4D geometries by manipulating them through the visualization system. Compared with the 3D visualization scenario, in the 4D environment, a typical interaction operation such as rotation requires extra degrees of freedom. Intuitive manipulation of displayed objects that are projected from a 4D model space to a 3D display space requires six degrees of orientation control to start with. A mouse has two degrees of freedom, which means that it needs the assistance of keyboard and mouse buttons. However, this increases the difficulty of recalling operations and makes interaction less intuitive.

4D Mathematical Visualization Systems

This subsection first surveys various visualization systems designed by different research groups. The second part of this subsection summarizes the common steps required by the existing con-

Figure 1.3: 4D visualization systems: (a) MeshView by Hanson et al. at Indiana University [60]; (b) Geomview by the Geometry Center of the University of Minnesota [124].

ventional framework of 4D visualization systems and discusses the limitations and insufficiencies of these common steps.

A very early real-time 4D visualization system [6] was developed by Banchoff's research group. This visualization system projects 4D surfaces to 3D with a custom-built multiplier and renders the wire-frame of the projected surfaces by a vector graphics display.

At Indiana University, the MeshView [60] system developed by Ma and Hanson is an interactive 4D viewer, see Figure 1.3(a). This system supports real-time 4D rotation using a combination interface of mouse and keyboard. Using the 4D mesh data files as input, this system can interactively display large families of 4D surfaces, such as the $n = 4$ construction of complex functions. Another system [59, 57] developed by Hanson and Heng simulates a 4D lighting environment by the analogous of 3D lighting. This system supports 4D illumination of points, curves, surfaces, and volume embedded in a 4D space by inventing a thicken method to achieve the unique 4D normal required by the 4D illumination computation. The GL4D interactive visualization system [27] of Chu et al. is a GPU implementation of this 4D lighting system. This system distributes 4D ray trace computing and geometry rendering to a GPU in parallel to provide real-time visualization of 4D objects illuminated by 4D light sources.

Another interactive system for visualizing the two-manifold embedded in four-dimension is called Fourphront [7]. This visualization system run on Pixel-Planes 5, a massively parallel graphics engine. It was designed by Banks at the University of North Carolina(Chapel Hill). A set of visualization techniques such as depth cuing, double-side coloring, intersection highlighting, and user customized opacity are provided by this system. This system also provides interaction controls such as rotation and translation.

Geomview [124] was originally developed as 3D visualization software. It was developed in 1987 at the Geometry Center of the University of Minnesota by Levy, Munzner, and Phillips. Geomview was created with flexibility as the primary goal. It runs on a number of platforms and can connect with other software such as Mathematica and Maple. It not only supports the viewing of 3D surfaces, but also provides a large variety of modules such the display of curves, dual surface computation, and intersection computation. Later, an external program called 4Dview developed by Meyer was incorporated into Geomview. This external program provides users the capabilities of changing the 4D viewpoint and creating 4D slices. After that, 4D visualization capabilities became a standard part of Geomview, see Figure 1.3(b). NDView developed by Holt and Levy is another external module of Geomview and runs on Silicon Graphics workstations. NDView supports the visualization of 4D or even higher-dimensional surfaces. It also provides a controller for users to customize the projection subspaces.

There is a rich set of research papers in the 4D visualization area apart from the above key works. This thesis conducts a more detailed survey of these on this in Chapter 2.

Conventional Directions and Challenges

The fourth dimension is beyond our three-dimensional spacial space; it is impossible to gain a mental model by directly “seeing” or “feeling” a physically “real” 4D object. Assisting users’ perception and helping users to achieve a mental model of a 4D world is the major task of a

4D visualization system. In more detail, the pioneer research work and visualization systems of 4D visualization can be categorized in two directions: creating a holistic visual experience of 4D geometries, and interacting with 4D geometries.

a) Creating a holistic visual experience of 4D geometries. The common step for the pioneer 4D visualization system is projection. This is due to the fact that to visualize N dimensions, one only needs an $(N-1)$ -dimensional retina. In 3D, our screen is 2D, if we want to visualize a 3D object, we have to project it to one lower dimension which is 2D. We regularly observe 2D representations of 3D objects on movie screens, cameras, and computer displays, and we instinctively consider those 2D images to represent real 3D objects. That is, we have become accustomed to the properties of *projections* of 3D objects onto real and simulated 2D images. For moving images, we perceptually exploit motion parallax to fill in the 3D world via 2D image sequences.

Similarly, in 4D, we do not have the option of directly interacting with 4D objects, so we also need a projection, but this time our screen is one dimension higher, which is 3D, when we want to visualize 4D objects, we have to project them onto a 3D volumetric screen, and then see them inside this *3D screen*. Projection in 3D worlds transforms a 3D object into a 2D object by either “throwing away” the z coordinate or dividing it by the depth to get perspective distortion. In a 4D world, the analogous process is to project a 4D object from 4D to 3D by throwing away the w coordinate, again with a possible perspective 4D depth division. Of course, we must accept the limitation that, unless we have a stereographic display, we have only a 2D screen from which to acquire a full 3D mental image. In general, this projection is a two-step procedure: a 4D to 3D projection and a 3D to 2D screen projection. However, this two-step projection will cause the following three problems:

#1: Dimension lost. The 4D to 3D projection is actually a dimension reduction, and this procedure can result in information being lost from the reduced dimension. To resolve this issue,

certain visual cues must be provided. One typical solution is illumination. In an environment with a 4D light source, the shading of the 4D object relies on the 4D normal, which is computed by using the 4D coordinates (before projection) of the geometries. This means the shading results will be affected by the reduced dimension; on the other hand, the reduced dimension can be reflected by the shading. Another solution is more straightforward. This is depth coloring; the geometries will be colored according to the reduced dimension.

#2: Geometrical information lost. Certain geometrical information such as symmetry and curvature is unavoidably lost after the projection because we rely on the 3D projection view to explore the geometry that actually exists in four dimensions. Moreover, the visualization of such geometrical information relies heavily on the choice of viewing direction for presenting the 4D geometries. To resolve this problem, this thesis exploits the parallel coordinate and provides dual-view visualization.

#3: Occlusion. Occlusions are caused by both 4D to 3D, and 3D to 2D projections. The occlusions caused by the 3D to 2D projection will affect our visualization of the internal structures of the geometries. This type of occlusion problem belongs to the 3D visualization and can be resolved by applying a transparent rendering to the geometries. Projecting the 4D surface to a 3D space may lead to the second type of occlusion. The 3D issue analogous to this is projecting a 3D trefoil into 2D. Although in 3D space, the trefoil does not have any intersections, projecting the 3D trefoil knot onto a 2D screen can still produce intersecting (occlusion) points. Although a 4D surface does not have any intersections in 4D space, projecting it to 3D may cause intersecting (occlusion) lines. This thesis provides a 4D halo technique to handle this problem.

b) Interacting with 4D geometries. Typical interactions in a 4D visualization system include: 4D rotation, scaling, translation, and 4D focal length control. 4D rotation is the most common interaction in visualization of a single 4D object; it helps users to gain a holistic perception of

the 4D geometry among different viewing directions. However, compared with 3D rotation, 4D rotation requires more degrees of freedom. In 3D, there are three independent rotational degrees of freedom, and a typical list of these would include rotations in the yz -plane, the zx -plane, and the xy -plane. In 4D, all the rotational degrees of freedom are given by 6 independent parameters, corresponding to the rotations in the planes labeled by wx , wy , wz , yz , zx , and xy . Researchers have been using a combination of mouse and keyboard's short keys to control these 6 degrees of freedom, see MeshView [60] as an example.

However, in terms of intuitiveness and efficiency, the mouse and keyboard's combined interface is not sufficient. Because a mouse has only a two degrees of freedom input, to control the six degrees of freedom of 4D rotation, users need to either use the mouse to switch between different states, or use the keyboard to assist with the mode selection. Another limitation is that the mouse and keyboard interface is not suitable for public usage of a 4D visualization system, e.g., using a 4D visualization system as an educational tool in a topology lecture. This research thesis proposes a multitouch interface with remote control capability to enhance the interaction.

1.2 Research Objectives

The problems to be solved are basically how to present 4D models to our eyes and how to enhance the user's intuitive experience of the abstract geometric world that we are trying to understand. Two basic elements form the foundation of the approaches available to us for exploring the fourth dimension: the first is the family of display methods adapted to represent 4D geometry in a conventional graphics environment that is highly-tuned for displaying three-dimensional objects. The second is the interactive interface used to manipulate the resulting displayed objects to enhance 4D understanding. The major goal of this research project is to find new directions for solving these two problems. In more detail, my research objectives are

to enhance 4D visualization by discovering new directions and building a comprehensive 4D visualization system to fulfill the following requirements:

- 1) *The projection view should be supplemented by other view.* Projecting 4D geometry to 3D space in a Cartesian coordinate system is an essential step to produce the visual perception of the 4D geometry. Nevertheless, this dimension-reduced projection can also lead to the loss of the geometric information and may affect the understanding of the geometry. Hence, the 4D visualization system should support other new visualization methods, e.g., parallel coordinates, to provide a complementary view to recover the lost information.
- 2) *Both 3D lighting and 4D lighting should be supported.* In 3D visualization, lighting is an important factor in enhancing depth perception and for producing a realistic visual feeling. In our 4D visualization system, 4D lighting is supported to reveal 4D depth perception. This system also supports 3D lighting as a comparison for the 4D lighting effect. Moreover, 3D lighting also serves as a choice for low performance machines.
- 3) *Appropriate visual cues should be exploited.* Visual cues can reveal the internal structure of 4D geometries as well as help users to observe the correct intersection and occlusion information when interacting with 4D geometries. After the projection the depth information should not be discarded. Our system should take advantage of it to provide visual cues such as 4D depth coloring, 4D halos, and intersection curves. These visual cues can enhance depth perception and enrich visualization.
- 4) *Support a multitouch interface.* The multitouch technique has become a part of our daily life in the last few decades. Compared with a mouse, a multitouch interface allows control with multiple degrees-of-freedom by simultaneous use of multiple fingers for input. This advantage makes the multitouch interface more suitable as a controller for a 4D visualization interface which requires a controller with multiple degrees of freedom. The mouse and keyboard interface should also be supported and incorporated with the multitouch controller.

5) *A mobile device application with a maximum subset of the above components.* Mobile devices are pocketable and their computing power is rapidly growing. This means they have the potential to become platforms for 4D visualization systems. In our system we explore and implement a maximum subset of the above components that can run on mobile devices, so that more general users can be provided with our educational tools.

1.3 Research Contributions

Since we are living in a 3D space, comprehending geometry in 4D or even higher dimensional space can be very challenging, especially for ordinary people. Inspired by the information and illustrative visualization technologies, as well as the emerging interaction technologies, this thesis enhances 4D visualization by providing a comprehensive 4D visualization framework with a dual-view visualization, novel visual cues; and enriches the 4D visualization interface by exploring a new interaction method. In more detail, this research thesis contributes to 4D visualization by exploring the following new directions:

A comprehensive 4D visualization framework. First of all, this research work explores new directions and techniques for forming a comprehensive 4D visualization framework. Instead of replacing the conventional 4D visualization framework (see Chapter 3 for detail), the novel techniques proposed in this work are well designed and embedded in a typical 4D visualization framework. Based on the framework, a GPU based interactive 4D visualization system is implemented, and its light-weight version application can be run on the mobile devices. Our comprehensive system improves on the conventional system in terms of revealing clear geometric information, enhancing depth perception, providing more appropriate visual cues, supporting intuitive and efficient user controls, providing greater flexibility for the running platform, and improving accessibility for public users. In this work, we also provide several

user scenarios to illustrate the usage of the system, and perform user-studies to evaluate the system. See Chapter 7 for more detail.

Efficient construction of continuous parallel-coordinates plots of 4D surfaces. Secondly, this thesis proposes a dual-view visualization technique, which takes advantage of parallel-coordinates plots to assist the projection view. This visualization technique proposes the use of blue-noise sampling to efficiently generate continuous parallel-coordinates plots for 4D geometries, so that the plots can be anti-aliased while being a continuous metaphor of the 4D geometry. See Chapter 4 for details. In such visualization, the visual signatures in parallel-coordinates plots could reveal interesting geometric structures such as rotational and reflection symmetry in the corresponding co-dimensions. A collection of visual signatures with corresponding geometric meanings are presented in Chapter 4. In order to achieve interactive visualizations, this technique is developed on GPU by OpenGL and GLSL. While we interactively transform (e.g., by a rotation) the input geometry in the 3D projection view, our GPU shader can render the continuous parallel-coordinates plots for the 4D geometry in real-time, and reveal its visual signature on the plots for interactive examination. This thesis also supports interactive brushing across two views. Users can also interactively brush both the 3D projection view and parallel-coordinates plots to highlight co-related geometric features. Furthermore, given its interactive performance, users can also incorporate assorted computational geometric properties such as curvature to refine and enhance the visualization.

Interactive exploration of 4D geometry with volumetric halos. Thirdly, this thesis presents a comprehensive volume-rendering-based approach to the creation of halo structures for arbitrary curves and surfaces embedded in 4D and projected to a 3D volume image. Early volume-based 4D visualization work [59] was limited by the complexity of processing 4D objects and rendering them into a 3D image which, in turn, had to be volume-rendered onto a 2D screen. In this work, we have extended the fast GPU-based methods of Chu et al. [27] to include volumetric

halos to support interactive exploration of 4D haloed occlusion and depth. Redundant information cues, including color-coding, heuristic lighting, texturing, and stereoscopic viewing, are supported to enhance the perception of spatial relations among projected objects. Tetrahedron mismatch is a challenging technical problem in the creation of the auxiliary halo 3-manifolds needed to implement our approach; in this work, novel algorithms are developed to resolve the computational geometry issues that occur in this process. The details of the technique will be presented in Chapter 5.

New interaction methods. Moreover, this research thesis also expends effort in creating a multi-touch interface to intuitively interact with 4D objects, basically through rotation, scaling, and translation plus other commonly-used desirable controls. Multi-touch methods can expand the user's degree of freedom significantly beyond the traditional two-degree-of-freedom desktop mouse while avoiding the complexity of special purpose devices embedded in 3D space. These extra degrees of freedom can be effectively utilized in applications requiring control of data regarding unconventional geometry, with 4D space being a good example. This research thesis exploits an idealized set of user controls to facilitate a quasi-physical experience of (simulated) 4D objects such as the hypercube or, equivalently, a rolling four-dimensional die via a computer graphics touch screen combined with minimally-complex mapping of these controls onto currently available iPhone multitouch gestures. We also illustrate the functionality of our interface in Chapter 6, with a detailed pedagogical example showing how one can interact with and learn about the detail and structure of a hypercube viewed as a back-face-culled eight-sided 4D die. As this 4D die rolls in the fourth dimension, it exposes numerous clear analogs to the familiar physical properties of 3D dice. Details of this work will be presented in Chapter 6.

1.4 Thesis Organization

The remainder of this thesis is organized as follows: Chapter 2 reviews the history of four-dimensional geometry visualization, and surveys related work in the information visualization area and illustrative visualization area, as well as interaction methods that can be used to manipulate high-dimensional data/geometry. Chapter 3 introduces the pipeline of our comprehensive 4D visualization system and overviews its novel directions, and the later chapters describe these new directions in detail. Chapter 4 presents the dual-view visualization and the details of methods that take continuous parallel-coordinates plots to help visualize 4D geometries. Chapter 5 describes how the illustrative visual cues are used to reduce the visual occlusion caused by the dimension-reduced projection and enhanced 4D visualization. Chapter 6 describes our proposed interactive environment to teach intuitive properties of 4D models by taking full advantage of the multi-touch capabilities typically available in tangible hand-held devices. Chapter 7 presents several user evaluations to explore different aspects of the usability of the 4D visualization system. Chapter 8 draws a conclusion and discusses future work.

Chapter 2

Related Work

Exploiting the capability of computer graphics to help people understand the 4D geometries is a challenging problem. It requires not only a thoroughly study of the pioneers' work, but also an exploring of other research directions. This chapter starts by first reviewing the history of 4D visualization. Secondly, the techniques of visualizing the high-dimensional data in the information visualization area are studied. Thirdly, this chapter surveys the research works of visual enhancement in illustrative visualization area. Lastly, this chapter surveys emerging user interaction technologies that could possibly be employed and adopted to provide and support the interaction needed for 4D visualization.

2.1 Four-dimensional geometry visualization

Using computer graphics and visualization methods, we can interactively explore geometry in four (or more) dimensions [1, 45, 67], even though such geometry cannot be directly constructed or observed in our own physical world. One of the earliest comprehensive approaches was set forward by Noll in 1967 [105], who examined the general problem of projecting N -dimensional objects to a graphics screen for examination. Various fundamental techniques were proposed in this work for extending our three-dimensional experience to the fourth spa-

Figure 2.1: Sequence of rotating 4D hypercube, this figure is courtesy of Noll in 1967 [105].

tial dimension. This work proposed the two basic techniques of visualizing the N -dimensional objects: Project the N -dimensional objects to 3D by perspective projection and orthogonal projection; Manipulate the N -dimensional objects by N -dimensional rotation. To demonstrate the proposed techniques of visualizing the N -dimensional objects, this work generated a movie of animated wireframe hypercube with stereoscopic views, see Figure 2.1. Basic graphics methods for representing 4D objects, initially as wire frames, were then developed for geometric objects of interest and converted into images and animated projections by numerous authors [5, 46, 6, 71]. The pioneering work of Banchoff (see, e.g., [5, 6]) focused on viewing well-known 4D mathematical objects using continuously rotating projections and was a fundamental influence. Later on, Feiner and Beshers [43] developed n -Vision testbed, allowing users to systematically explore nested subspaces in n -dimensional worlds. n -Vision is implemented using a hierarchy of nested boxes. These boxes serve as containers for presenting graphical output and capturing graphical input. A box's coordinate system represents a transformation relative to that of its parent. In n -Vision testbed, three types of interactions includes dipstick, waterline, magnifying box were supported. The dipstick is a small probe that the user may pick up and move within a surface. The waterline is a plane that is perpendicular to one of the axes in a world. The magnifying box is a 3D version of the familiar 2D “detail” window that

Figure 2.2: The visualization result of two-dimensional surfaces in four-dimensional space from the work of Hoffmann and Zhou [70]: (a) Bracket viewed from $\theta = (0, 0, 0, 45, 60, 0)$ degrees and $r = (0, 0)$ and shaded by color scale representing w -values; (b) Offset curve of $y - x^2 = 0$ traced in 4-space with highlighted silhouette curves; (c) Intersection of cylinder and moving sphere with the same radius.

provides a higher-resolution display of part of another window. This work also proposed an example of visualizing financial multi-variable functions by the n -Vision testbed.

Hoffmann and Zhou [70] proposed a framework for rendering two-dimensional surfaces in four-dimensional space using z-buffer graphics workstation. In this work, several techniques for visualizing the two-dimensional surfaces in four-dimensional space were proposed. These techniques including methods to specify the orientation of objects and of projection centers, to determine silhouette points of a 2-surface with respect to projections, and to calculate the normal of a projected 2-surface from its normal plane in 4-space. This work also applied their proposed techniques in visualizing offset curves and collision detection, see Figure 2.2. Hanson et al. proposed a set of mathematical and visualization approaches for illustrating concepts related to Fermats Last Theorem. These approaches include: Explicit parametric solutions of the equation; Cutaway surfaces; Surface evolution (animating the bounds of the parameters); Transparency; Rotations; Focal length; Model parameters; And compactification. These techniques were implemented in a 4D Meshview system [60]. This system can be used to display and perform mouse-driven interaction with surfaces embedded in 4D and projected

Figure 2.3: Illuminated the 4D embedded flat torus (top) and the 4D knotted sphere (bottom) by 4D light sources; top: torus is rotated in zw plane; bottom: from left to right: a large opacity value is used in the volume rendering so that the model appears like an opaque surface; a screen door effect is added by applying a checkerboard deletion pattern in the parametric space; last, the 4D lighting direction is animated, this figure is courtesy of Chu et al. [27].

to 3D with 3D lighting.

Some of researchers focused on shading and illuminating the 4D geometries. Early work including Steiner and Burton [120], and Carey et al. [22] was about extending the 3D lighting model and shading method into 4D. Bhaniramka et al. [13] proposed a fast Marching-Cubes-like algorithm to generate tetrahedral meshes for modeling isosurfaces over time-varying data, while Banks and Beason [9] developed an efficient approach to generate high-quality global-illumination effects on the 3-manifold in 4-dimensional space. The proposed approach includes a novel formulation of the light transport equation that can be applied to the graph of a scalar function in $n + 1$ -space to pre-compute illumination for level sets of the function in n -space and two steps: Flatten light transport from $n + 1$ -space to n -space; Sampling the flattened radiance on graphs in $n + 1$ -space. By taking the proposed illumination method, the isosurfaces can be globally illuminated in real time. Weiskopf et al. [132] revealed the geometric structure of Einstein's theories of special and general relativity using four dimensional ray tracing.

Hollasch [71] proposed using ray-tracing methods to solve hidden surface and shadowing of objects in four-dimensional space. This work generated the 3D ray-tracing algorithm and ap-

Figure 2.4: The illumination result of a 4D A bumpy torus and a fur-textured vector field protruding from it, this figure is courtesy of Banks [8]: (a) The torus is rendered with illumination only; (b) The torus is rendered with illumination together with a self-shadow effect; (c) The torus is rendered with illumination, self-shadow effect, and attenuated lighting.

plied it to the computation of the 4D illumination calculations. As a result, this work provided visualization examples include hyperspheres, tetrahedra and parallelepipeds rendered by using the 4D raytracer. A nice interactive example is the broad approach adopted by Banks for controlling displays of 4D geometry in the Pixel Planes environment [7], they also proposed various cues to carefully examine the geometry and depth information in the 4D to 3D projection, e.g., the distortion caused by perspective projection, parallax and stereopsis, focus dependent transparency. In order to reveal the inside structure of the projected 4D surfaces, this work employed the techniques such as ribbon, clip, transparency. Moreover, this work also proposed methods to analytically compute the silhouettes and intersections of the 3D projection of 4D surfaces. Banks [8] also suggested a general lighting model to illuminate k -manifolds in n -space with diffuse and specular reflections. In order to exploit human perceptual capabilities of 4D geometry and make 3D images, Hanson and Heng [59, 57] extended the three-dimensional illuminating method to four-dimensional illumination. In our three dimensional world, lighting is an important factor for humans to see and understand the world. Light reflecting from objects is detected by our eyes, then we have the light distribution, then our mind can deduce the shape of objects and perceive them. This also provided graphics researchers the idea of pre-

Figure 2.5: (a,b) Tightening a 4D knotted sphere by pulling on the 3D shadow. (c,d,e) Pushing on the knotted sphere to loosen it, this figure is courtesy of Zhang and Hanson [140].

senting the three-dimensional objects, they illuminated three-dimensional objects to creating two-dimensional images. Analogous to three-dimensional graphics, Hanson and Heng [59, 57] provided a set of methods for four-dimensional illumination including: (1) Replace the depth-buffered scan-line conversion approach used for ordinary 2D images of 3D objects by the analogous 4D approach, which uses scan-plane conversion and 4D depth-buffering of 4D objects to produce volume renderings. (2) Introduce a technique that allows points, curves, and surfaces in 4D to interact correctly with 4D lighting by thickening them symmetrically so they acquire unique 4D normal vectors. Recently, a 4D display system capable of interactively rendering the 4D lighting features described in [59, 57] has been implemented using GPU methods to overcome many performance obstacles [27]. They developed a GPU-based rasterization approach to efficiently render 3-manifolds into a volumetric screen buffer, see Figure 2.3. Banks [8] also proposed a light model for arbitrarily large dimensions. The results generated in this work also enhanced to produce global effects such as self shadow and attenuated lighting, see Figure 2.4. different with the work of Hanson and Heng [59, 57], this work proceeded directly from the geometry to the illumination solution, without regard to the participating dimensions.

Another stream of research concerns with the interface of 4D visualization system and how one can interact with 4D geometries. Duffin and Barrett [36] developed a user interface method for manipulating rotation and projection in n -space. Their technique provides n -dimensional rotation matrices solely from information about the visualized data coordinate system and its projection onto the viewing plane. In their approach, user performed the rotations by selecting

Figure 2.6: Interactive visualization the 4D subset for the Julia set in nested coordinate system, this figure is courtesy of Miller and Gavosto in 2004 [102]: (a) View inside the immersive visualization probe view; (b) Nested-coordinate-system view.

a data coordinate axis and dragging the projected end of the axis in the viewing plane. From the path traversed in the viewing plane, a sequence of n -dimensional rotation matrices is created. Miller and Gavosto [102] employed nested coordinate methods to develop a user interface tool for continuously exploring 4D subspaces in n -dimensional data. Their work demonstrated that the immersive visualization probe is effective for displaying continuous 4D subsets of n -dimensional data, they also provided the visualization scenario in which users can interactively adjust the hyperplanes to slice down through several dimensions, and illustrated how the combination of an embedded coordinate system and the resulting 4D slice representation contributes to an understanding of the underlying physical or mathematical process involved. For both computational and visual reasons, they sampled on a set of planes rather than a volumetric display for the final 4D slice. More recently, Zhang and Hanson [140] proposed a reduced-dimension approach by generating shadows of higher-dimensional objects and manipulating the objects via its shadow with haptics devices. By adding force-feedback to the concept of reduced-dimension object manipulation controllers, they created a unique way of building intuition about 3D knots and curves in a 2D touchable space; their approach is naturally extensible

to a full 3D haptic controller that provides touchable intuition and force feedback through simulated representations of shapes in a 4D world. In their work, they pointed out that the key to editing topological surfaces embedded in four dimensions is a proper collision handling mechanism that realizes material forces and prevents collisions from occurring in 4D. They proposed a mechanisms of 4D collision detection and avoidance, moreover, they also extend mass-spring system to 4D in order to simulate 4D Cloth-environment. Figure 2.5 shows the procedure of tightening a 4D knotted sphere.

2.2 Visualizing the high-dimensional data

High-dimensional data visualization is part of the information visualization field, where it focuses on dealing with the visual representations of data with multiple variants. This area typically has a long history and consists of a number of different visualization methods tailored for use in different scenarios. In this section, we first overview the visualization methods involved and then focus on those that we planned to use for visualizing/summarizing statistical structures in 4D geometry, e.g., parallel coordinates.

According to Ferreira et al. [44], there are four categories of techniques for visualizing multivariate or multi-dimensional data:

- Geometric-based methods, such as scatterplot [10] (see Figure 2.7), parallel coordinates [82, 83], radial coordinates visualization [69] (see Figure 2.8 left), etc. These techniques aim at helping users to explore the informative projections of multidimensional data sets;
- Icon-based methods, such as star glyph (see Figure 2.8 right), stick figures (see Figure 2.9), etc. This kind of techniques maps each multidimensional data point to an icon (or glyph) and the visual features of the icon depend on data values; this technique is not

Figure 2.7: A scatterplot matrix for 5-dimensional data of 400 automobiles, this figure is courtesy of Becker et al. [10].

Figure 2.8: Left: radial coordinates, this figure is courtesy of Hoffman [69]. Right: group of star glyphs, this figure is courtesy of Chambers et al. [85].

Figure 2.9: (a) Stick figure family; (b) 5D image data using stick figures; (c) Part of (b) in original size, this figure is courtesy of Chambers et al. [85].

Figure 2.10: Left: pixel-based visualization of 6-dimensional data, this figure is courtesy of Keim et al. [90]. Right: n -vision, this figure is courtesy of Feiner et al. [43].

suitable for visualizing the continuous geometry data since we have to design different icons for different pieces of independent data with different values [85] while the 4D geometric data is continuous in nature;

- Pixel-oriented methods. The idea here is to represent each attribute value by a pixel based on some color scale [90], see Figure 2.10 left;
- Hierarchical methods. These techniques subdivide the data space and present subspaces in a hierarchical fashion. Note that these techniques have already been used to visualize n -dimensional function, i.e., the n -vision technique developed by Feiner and Beshers [43], see Figure 2.10 right.

Among the above techniques, geometric projection is more suitable for visualizing continuous geometric data. The continuous version of scatterplot equals to that of the multi-view version of the conventional visualization method (the 4D to 3D projection). Moreover, the radial coordinates is similar to the parallel coordinates. The only difference between them is that in parallel coordinates plots, the coordinate axes are in parallel while in radial coordinates plots, the coordinate axes are radiated from a common center point. Note that I take parallel coordinates as the final choice in my visualization framework. Hence, I will focus more on surveying research work on parallel coordinates in Chapter 4.

2.3 Illustrative Visualization

While non-photorealistic rendering techniques [52, 134, 50] depict data in an expressive way, illustrative visualization [111, 33] focuses also on the use of visual emphasis to highlight relevant features of interest in the visualization. This is particularly useful when exploring massive or complicated data. Apart from halos, several illustrative visualization methods have been proposed. Diepstraten et al. [35] explored different approaches to generate cutaway illustrations at

Figure 2.11: Volume-based illustrative rendering. Left: Volume-based illustrative rendering for a human head, yellow color indicates low distance to the vessels, this figure is courtesy of Rautek et al. [111]. Right: Illustrative annotations of direct volume rendered carp by VolumeShop, this figure is courtesy of Bruckner et al. [18].

interactive speed with a small set of rules. Li et al. [96] further considered a geometry-aware approach supporting interactive authoring in cutaway illustrations of complex 3D models. Later, Bruckner and Gröller [19] developed a view-dependent approach to generate exploded views for volume visualization. Karpenko et al. [88] recently illustrated shape and internal structures in complicated mathematical surfaces using exploded view diagrams by partitioning the surfaces into parallel slices. Meanwhile, Hummel et al. [72] examined various non-photorealistic rendering techniques on integral surfaces to convey both shape and directional information, while Born et al. [15] proposed a novel way to render stream surfaces.

Volume-based illustrative rendering approaches include the work of Bruckner and Gröller [18], who proposed the concept of direct volume illustration environments (see Figure 2.11Right), generating high-quality illustrations from volumetric data, and the work of Svakhine et al. [123], who studied traditional drawings of medical illustrations to create an illustration system designed for medical volume data. Rautek et al. [110] proposed using a semantic layer in transfer function design, allowing domain experts to map volumetric attributes to various illustrative visual styles (see Figure 2.11Left). Chen et al. [25] suggested a novel approach to illustrate

Figure 2.12: From left to right: Wiimote, Kinect, multi-touch handheld devices: iPad.

volumetric muscle models from DTI images and 2D examples, while Pelt et al. [127] presented a GPGPU framework for illustrative volume visualization that renders multiple volume features interactively in pen-and-ink style.

2.4 Emerging User Interaction Technologies

Some highly emerging user interaction technologies/devices include the followings: remote controller such as Wiimote, Kinect, multi-touch handheld/tablet devices (see Figure 2.12) and autostereoscopic displays with mid-air interaction. Among them, Wiimote and Kinect are all motion-based input devices; they provide multiple degrees-of-freedom interactions. Using such kind of devices for a long time can easily make our hands/arms fatigue. On the contrary, multi-touch devices are touch-based interfaces, thus users can rest their arms on a physical surface such as a table during their interactions. With multi-touch devices, we can also perform complex interactions with many degrees of freedom. In addition, multi-touch is also very popular in a way that it is available in most handheld devices nowadays. The major advantages of multi-touch interfaces that meet the requirements of 4D interaction are: 1) more degrees of freedom and 2) more efficient. So in the following, I will focus on reviewing research work and technologies on multi-touch interaction.

Different multitouch devices. There are four general approaches to construct the hardware system for sensing multi-touch interactions: capacitive touch-screen based on capacitive coupling effects, optical imaging touch-screen using computer-vision-based tracking, infrared touch-screen using vertical and horizontal infrared sensors , and FTIR touch-screen based on total internal reflection. Based on their advantages and disadvantages, they are used to construct different kinds of multi-touch devices. According to the physical size (or form factor), multi-touch has been integrated and practised in various hardware [21], as shown below:

- **Handheld device.** A number of popular handheld devices support multi-touch, e.g., Apple Inc.'s iPod touch, iPhone and iPad [74], Nokia Inc.'s X6 [78] and Microsoft Inc's Windows Phone 7 [77], etc. This kind of devices typically uses the capacitive touch-screen because it is responsive and inexpensive, and can provide continuously feedback. However, it cannot handle a stylus or a gloved hand, because the capacitive touch-screen method relies on the capacitive sensors so that it can only sense conductive material. Other than that, another unique feature about conventional handheld devices is that they usually support a collection of hardware features additional to multi-touch, e.g., camera, accelerometer, magnetometer, voice, etc.
- **Trackpad & Mouse.** Apple Inc. integrated a multi-touch trackpad on its laptops, so that users can use various multi-touch gestures in most of its applications. For desktop, Apple Inc. also developed a standalone trackpad called magic pad and a kind of multi-touch mouse called magic mouse [128]. Instead of directly manipulating on the screen, user can perform multi-touch on trackpad or magic mouse like using a conventional mouse; this approach can avoid the hand occlusion problem.
- **LCD displays.** Multi-touch-enabled LCD displays are also available on the market in recent years, e.g., the all-in-one LCDs from PQ Labs Inc. [94] can support 32 simultaneous touches; 3M Inc's capacitive touch screen can support 20 simultaneous touches [73],

while HP Inc.'s TouchSmart Tx2 can support 9 simultaneous touches [75]. Since common operating systems such as Windows 7 [76], Ubuntu 10.10 [79] and Mac OS X 10.5 [74] already integrated with multi-touch input, multi-touch hardware can cooperate with the software applications more easily and all these also help to promote multi-touch interactions to the general public users.

- **Tabletop.** When interacting with a vertical display, our hands and arms can get tired easily. Tabletop multi-touch devices provides an alternative: it's horizontal and user do not need to lift their arms during the interaction. In addition, users can also put object, such as mobile phones and digital cameras, on the table, and the table can recognize and communicate with these additional digital devices. One example is if the user put a Bluetooth-enabled phone on surface, surface can directly download photos or music from it; users can have tangible interaction with this surface through physical objects in addition to multi-touch.
- **Large displays.** This kind of multi-touch devices are generally used by research labs, museums and government organizations, so it often supports not only multi-touch but also multi-users. MultiTouch Ltd.'s MultiTouch Box is a kind of large-displays, it is a multi-user, scalable and versatile device. Han's FTIR approach [53] is often used in constructing these large-scale multi-touch displays nowadays.

Characteristics of multi-touch devices. There are several characteristics about multi-touch devices [21]: number of touch points, touch accuracy, response rate, resolution, and physical size. Number of touch points refers to the degrees of freedom of the multi-touch interaction, each touch point has 2 DOF. We can have a larger multi-touch gesture set if the device can support more touches points. Touch-screen with a higher accuracy and response rate can make the interaction more effective. Resolution is an important characteristic to multi-touch display, but not that important to a multi-touch trackpad.

2.5 Summary

This chapter summarizes the research work related to this research project. I first review the history of four-dimensional geometry visualization. Then, I summarize the related works in high-dimensional information visualization and illustrative visualization area, respectively. Lastly, I survey interaction methods that can possibly be used to manipulate high-dimensional data/geometry. After this chapter, I will introduce the comprehensive 4D visualization framework in Chapter 3.

Chapter 3

A Comprehensive Four-dimensional Mathematical Visualization Framework

Visualizing and understanding geometries in 4D or even higher dimensional space can be very difficult due to the fact that we are living in a 3D space. Exploiting computer graphics as a visualization tool is a promising solution. Pioneer 4D visualization systems have made a good start in presenting 4D geometries to the human eye. However, these systems still have some unsolved problems such as dimension lost, loss of geometrical information, occlusion problems, and interaction efficiency. In this chapter, we first discuss the conventional pipeline of 4D visualization systems, and then propose a more comprehensive four-dimensional mathematical visualization framework to solve existing problems of conventional 4D visualization pipeline and enhance the 4D visualization.

3.1 Introduction

The major research objective in the area of 4D visualization is to produce a graphic representation of 4D geometries correctly and properly so that they can be understood by humans. Existing 4D visualization research work has tried hard to achieve this goal. In these research works,

several 4D visualization systems have been developed and applied as research tools in mathematical research. However, the existing systems can still be improved in terms of dimension lost, loss of geometrical information, occlusion problems, and interaction efficiency. This thesis enhances 4D visualization by designing a comprehensive 4D visualization framework with a novel dual-view visualization, a rich set of visual cues, and an intuitive and efficient interface. In this chapter, we propose and introduce this comprehensive 4D visualization framework. Moreover, this chapter serves as an overview of the following chapters, enhances different aspects of the 4D visualization, and details the novel techniques used in the proposed framework. Instead of replacing the typical 4D visualization framework, the novel techniques proposed in this thesis are well designed and embedded in the conventional 4D visualization framework. Based on the framework, a GPU based interactive 4D visualization system is implemented. Moreover, we have designed and implemented the light version of our 4D visualization system on the Apple family of mobile devices. Our comprehensive system improves the conventional system in terms of revealing clear geometric information, enhancing depth perception, providing more appropriate visual cues, supporting intuitive and efficient user controls, improving the flexibility of the running platform, and improving accessibility for users.

This chapter is organized as follows: Section 3.2 summarize the visualization pipeline of the existing 4D visualization systems. Section 3.3 proposes and introduces the enhanced pipeline of our comprehensive 4D visualization system. Finally, we discuss how our framework enhance the existing 4D visualization system in different ways in Section 3.4, and we conclude the paper in Section 3.5.

3.2 The Conventional 4D Imaging Pipeline

Before describing the pipeline of our comprehensive four-dimensional mathematical visualization system, we first outline the basic 4D imaging procedure. In general, 4D imaging can be

understood as an exact parallel of 3D imaging with the following refinements:

- The image of a 4D geometric object is a voxel image in a 3D volume instead of a 2D pixel image. The representation of a 4D object is thus realized in a 3D image and needs to be rotatable in the 3D screen volume to see all its aspects when presented on a 2D display; the analogous rotation for a 3D object would be a spinning of its image in the 2D screen plane, which, in contrast, provides little or no useful information.
- Orthogonal projection is implemented by a 4×3 projection matrix eliminating a 4D directional line of sight instead of the familiar 3×2 projection matrix eliminating a 3D directional line of sight.
- Perspective projection is implemented by dividing by the 4D distance between a rendered element and the 4D viewpoint instead of dividing by the 3D distance to the 3D viewpoint.
- In 3D, points lying on coincident lines of sight result in curves that shadow segments of other curves that are more distant; curves that do not collide in 3D can appear to touch or cross each other at *points* in the 2D image. Non-colliding 4D surface patches can similarly result in illusory 3D *curves* where two surfaces appear to intersect, shadowing one another in the 3D projection, but never meeting in 4D.

3.3 Comprehensive Four-dimensional Visualization Pipeline

The input to our system is a set of equations (by a dynamically-linked plugin) that represent a 2D geometric surface in N -space, e.g., the 4D PILLOW model is:

$$\begin{cases} X_1(u, v) = \cos(u) \\ X_2(u, v) = \cos(v) \\ X_3(u, v) = \sin(u)\sin(v) \\ X_4(u, v) = \sin(u)\cos(v) \end{cases},$$

Figure 3.1: Overview of the eight major steps in our visualization pipeline: 1) Surface construction to generate a set of grid points; 2) Blue-noise surface sampling to obtain a sample point set; 3) Interactive 4D transformation (rotation and projection); 4) Parallel-coordinates plots with visual signatures; 5) Illustrative visualization; 6) Enhanced visual signatures (e.g., emphasize high curvature areas); 7) 4D geometry manipulation; And 8) interactive brushing.

where (u, v) denotes the parametric space and $X_i(u, v)$'s denote the geometric surface. Figure 7.1 presents an overview of our visualization pipeline by using this PILLOW model as a running example. Given the input equations, we have the following series of seven steps:

- **Pre-processing:** #1) *surface construction*: we construct from $X_i(u, v)$ a grid of 4D points on the given geometry; and #2) *blue-noise sampling*: additionally, we generate also a set of sample points on the surface later for constructing the parallel-coordinates plots.
- **Interactive Visualization:** #3) *4-dimensional transformation*: 4D rotation is next applied to transform the pre-computed grid vertices and sample points, followed by an 4-to-3D projection on the grid vertices; and #4) *parallel-coordinates plots*: next, we employ the transformed sample points to generate continuous parallel-coordinates plots for compiling the *visual signatures*. The details of the the above dual-view visualization structure and the construction of the parallel-coordinates plots are discussed in Chapter 4.
- **Visual Enhancement:** #5) *illustrative visualization*: Projecting 4D surfaces into 3D can easily produce occlusions. Although some surfaces are comfortably separated in actual 4D space, their projections may appear to penetrate each other in the 3D screen. The major task for this visual enhancement is to provide illustrative visualization emphasis

for 4D depth relations among apparently coinciding surface projections in the 3D screen. We thus generalized the halo technique, which is widely used in 3D visualization and providing 4D halo visualization. Other than the novel 4D halo visual enhancement, this framework is also inherent in existing 4D visualization systems and supports other visual enhancement such as 4D lighting, 3D lighting, and transparency rendering, etc. Please refer to Chapter 5 for the details of the 4D halo illustrative visualization and other visual ques. #6) *enhanced visual signature*: furthermore, we can compute and incorporate geometric properties such as curvature to further enrich the visualization. The high curvature area will be highlighted in both views. Please refer Chapter 4 for more details.

- **Interaction:** #7) *4D geometry manipulation*: we designed a set of multitouch gestures to allow users to intuitively and efficiently perform geometric manipulations such as rotation, translation, scale, and focal length changing. The details of the interface design and user scenarios can be found in Chapter 6 ; and #8) *interactive brushing*: lastly, we can also support interactive brushing on both the 3D projection view and parallel-coordinates plots for feature markup and exploration (see Chapter 4).

3.4 4D Visualization Enhancement

To provide a meaningful representation of 4D geometries so people can achieve correct perceptions of the 4D world, we explore the potential of comprehending the existing 4D visualization systems in several new directions. In more details, the above comprehensive framework improves the existing 4D visualization system and enhances the 4D visualization in the following four aspects:

Holistic Visual Representation. Having the proper visual perception of a geometry is indispensable for understanding the geometry. It requires the geometry to be holistically and

correctly presented. This is more challenging when we are visualizing geometries beyond three dimensions, e.g., 4D. The most direct and efficient method is to project the geometry to 3D in our familiar 3D Cartesian coordinate system. However, this method is not perfect, since the projection reduces dimensions and this will affect the holistic perception of the geometry. Hence, this thesis proposes a dual-view visualization technique, which takes advantage of parallel-coordinates plots to assist the projection view. This visualization technique uses blue-noise sampling to efficiently generate continuous parallel-coordinates plots for 4D geometries, so that the plots can be anti-aliased while being a continuous metaphor of the 4D geometry. By employing such dual-view visualization in our framework, the dimensions reduced in 3D projection views will appear in parallel-coordinates plots. Users can interact with the geometry in one of the view while visualizing the changes in both of the views to get a holistic visual perception of the geometry. Moreover, in such a visualization, some important geometric structures such as rotational and reflection symmetry, in the corresponding co-dimensions can be observed. See Chapter 4 for details.

Correct Visual Perception. As a significant depth cue, occlusion reveals information about the relative depth, proximity, and spatial arrangement of the objects in a given view. However, projecting the 4D object to 3D will also cause occlusion. This type of occlusion does not exist in actual 4D space, and is due to the dimension reduced projection. In our framework, we refine the occlusion representation by means of the halo cutaway technique, which is a schematically emphasized occlusion diagram for surfaces in 4D space projected to a 3D image. Moreover, our halo cutaway technique is designed as a GPU-based algorithm, so that it can be easily embedded in our 4D visualization system and can be used interactively. Assisted by the halo cutaway technique, our 4D visualization system can redress the visual perception of 4D surfaces by presenting them with correct occlusion information, and emphasizing 4D depth relations. As well as the halo technique, redundant information cues, including color-coding, heuristic lighting, texturing, and stereoscopic viewing, are supported in our framework

to enhance the perception of spatial relations among projected objects. Please refer to Chapter 5 for the technical details and for more information on how the 4D visual perception can be enhanced by our carefully designed visual cues.

Intuitive Interaction. Interaction is another important aspect of studying and getting a holistic mental picture of a geometry. In our framework, we enhanced the traditional mouse and keyboard interface by supporting the multitouch interface. Multitouch methods can remarkably expand the user's degree of freedom beyond the traditional two-degree-of-freedom provided by a desktop mouse while preventing the use of complex devices with special purposes embedded in 3D space. With our multitouch interface, users can either intuitively interact with 4D objects in the 3D projection view, basically through rotation, scaling, and translation plus other commonly-used desirable controls, or they can perform brush across the parallel-coordinate view and 3D projection view. In our visualization system, we implement the multitouch interface in a server and remote controller architecture. We can install a graphics program that displays the 4D object on another computer driving a projector screen or desktop display, and control that computer remotely from the handheld device. Examples of a general 4D display program being controlled remotely are shown in We then have *two families of remote control methods* that can be employed, either separately or together, on the remote display. The entire set of multi-touch controls, including one-finger 3D control, two-finger 4D control, perspective, zoom, resetting, and mode selection, can all be applied to the remote display using an extremely simple wireless connection between the two devices. We need only to provide the URL of the remote display to the handheld application, and can then have full 4D control of a lecture display, for example, while walking around the room. As well as the fully-functioning PC system with remote controller, we also customized our main visualization system and developed an application for iOS platforms called 4Dice. By using this application, users can manipulate a 4D die by using a full set of multitouch gestures on a pocketable Apple device.

Details of the interface design and how it can be used as an educational tool will be presented in Chapter 6.

User Evaluations. In order to evaluate the enhancements of our comprehensive 4D visualization framework compare to the existing framework, in this thesis, we conduct three user evaluations. With these evaluations, we study the enhancement of our system to the existing 4D visualization framework in the following different aspects: The first user evaluation is to evaluate whether our dual-view visualization can enhance users' perceptions of the geometric features of the 4D objects; The second user evaluation is to evaluate the illustrative effects of the halo visualization; The last one is to test whether our multitouch interface can improve the 4D exploration experience in terms of efficiency and intuitiveness. Please refer to Chapter 7 for the details of the setting and the results of the user evaluations.

3.5 Summary

Firstly, this chapter summarizes the conventional 4D visualization framework. Then, a comprehensive 4D visualization framework is proposed and the novel techniques to enhance the 4D visualization are overviewed. Lastly, the chapter discusses how our framework enhances the 4D visualization in different aspects.

Chapter 4

Dual-view 4D Visualization

Although we can interactively rotate a 3D projected high-dimensional geometry and observe its dynamic changes, this traditional visualization method is limited and highly sensitive to the choice of viewing direction. Parallel-coordinates plots supplement this visualization scenario by providing statistical analysis of the geometry for distinct pairs of co-dimensions. Such analysis results in visual signatures that embed geometric structures such as symmetry, and thus allows us to overview the status of the missing dimensions while exploring the projected geometry.

This chapter presents a blue-noise sampling approach for efficient construction of continuous parallel-coordinates plots of high-dimensional geometric surfaces defined by mathematical equations. We employ the parallel-coordinates plots with the embedded visual signatures to assist the interactive exploration of high-dimensional geometries, typically for 2-manifold embedded in 4-space. While we interactively explore the 3D projected geometry, we can observe dynamic changes on its visual signature. Various geometric properties can also be identified and visualized. Moreover, we can interactively brush the plots, and see their counterparts in the 3D projection. Assorted geometric properties such as curvature can further be used to enhance the visual signature.

4.1 Introduction

Existing data sets in high-dimensional data visualizations are usually represented as a set of spatial data points, where each dimension corresponds to a certain data attribute. These visualizations generally aim at overviewing the entire data, and revealing data correlation and structure across different attributes. High-dimensional geometry data is fundamentally very different because they are defined by mathematical functions, so the high-dimensional spatial points are connected as a continuous manifold with geometric properties. Even if we rotate the geometry in high-dimensional space, its overall geometric structures and properties remain invariant. However, rotating conventional point-based data may not be meaningful as its data dimensions could be unrelated.

To visualize high-dimensional geometries, a common approach is to first *project the geometry to lower dimensions*, e.g., to two or three dimensions, so that the geometry can be rendered on computer screens with the assistance of interactive 3D graphics. In general, such a projection is carried out with an orthographic projection, which selectively takes two or three dimensions out of the 4 dimensions, or a perspective projection extended to handle 4-to-3-D projection. In addition to projection, we can also transform (typically with an 4-D rotation) the geometry in its 4-dimensional space, so that we can *bring out different aspects* of the geometry into the view of its 3D projection. In this way, dynamic changes in its projected geometrical structure can be seen. But even with such interactive exploration, certain geometrical information is unavoidably lost after the projection because we rely on the 3D projection view to explore the geometry that actually resides in higher dimensions. Moreover, such visualization relies heavily on the choice of viewing direction for presenting the high-dimensional geometry.

In this chapter, our goal is *not* to replace the 3D projection view, and use solely parallel coordinates [82, 83] to visualize the high-dimensional geometries. Rather, we aim at maximizing

our capability in visualizing and exploring high-dimensional geometries by using the parallel-coordinates plots to supplement the 3D projection view. Typical geometric surfaces employed in this work are mostly 4D, but our method can be extended to handle surfaces in higher dimensions. In detail, we take a coordinated-view approach on the 3D projection view and the parallel-coordinates plots, where we *overview* the transformation procedure of the geometry with parallel-coordinates plots, and *examine* the detail with the projected geometry in the 3D projection view. Motivated by the recent works of Choi et al. [26] and Tricaud and Saadé [125] who considered visual signatures in parallel-coordinates plots for identifying patterns in network traffic, we aim to study the correspondence and meanings between visual signatures in parallel-coordinates plots and high-dimensional geometry. This work also proposes various novel ways of visualizing and exploring the two views (3D projection view and parallel-coordinates plots) together, so that they can serve not only as a *visualization tool*, but also as an *exploration tool*. The following summarizes the major contributions of this chapter:

- **Blue-noise sampling to generate continuous parallel-coordinates plots:** First, this chapter proposes to use blue-noise sampling to efficiently generate continuous parallel-coordinates plots for high-dimensional geometries, so that the plots can be anti-aliased while being a continuous metaphor of the high-dimensional geometry, see Section 4.5 for detail.
- **Interpreting visual signatures in parallel-coordinates plots:** Second, the visual signatures in parallel-coordinates plots could reveal interesting geometric structures such as rotational and reflection symmetry, in the corresponding co-dimensions. A collection of visual signatures with corresponding geometric meanings are presented in this work.
- **GPU-based interactive visualization:** While we interactively transform (e.g., by a rotation) the input geometry in the 3D projection view, our GPU shader can render the continuous parallel-coordinates plots in real-time, and reveal its visual signature on the plots for interactive examination.

- **Interactive brushing across views:** Lastly, we can also interactively brush both the 3D projection view and parallel-coordinates plots for highlighting co-related geometric features. Furthermore, given the interactive performance, we can also incorporate assorted computational geometric properties such as curvature to refine and enhance the visualization.

4.2 Related Work about Parallel Coordinate

The idea of probing high-dimensional functions with two-dimensional plots was invented as early as 1885; mathematician Maurice d’Ocagne proposed solving equations with 2D plots known as nomography [17]. In more recent decades, Inselberg et al. [82, 83] developed a comprehensive framework for parallel coordinates and employed this framework as a flexible representation to depict multi-dimensional data on two-dimensional space. In short, the dimensions in the given data are mapped to parallel axes, where each multi-dimensional data point are converted to line strips pegged on the parallel axes. In addition to the formulation, Inselberg et al. [83] also demonstrated parallel coordinates in a number of visualization applications, thus popularizing parallel coordinates, particularly for its use in visualizing multivariate and high-dimensional data. After that, Wegman [131, 103] extended the parallel-coordinates representation as a high-dimensional data analysis tool and proposed using averaged shifted histograms to produce parallel-coordinates density plots. More recently, motivated by the need to accommodate continuous data, Heinrich and Weiskopf [66] proposed continuous parallel coordinates for multi-variate data defined in a continuous domain, see Figure 4.1. They derived a mathematical model to map and sample continuous data and also a sampling method with Monte Carlo integration to produce parallel-coordinates plots. Based on this work, we also generate continuous parallel-coordinates plots in Chapter 4, but we use blue-noise sampling to improve the efficiency in parallel-coordinates plots for high-dimensional geometry,

Figure 4.1: Left: discrete parallel coordinates. Right: continuous parallel coordinates. This figure is courtesy from Heinrich and Weiskopf [66].

and we propose also a number of novel ways of visualizing and exploring high-dimensional geometries. Related works on parallel coordinates can be categorized as follows:

Cluttering reduction. With increase in data set size, direct plotting of all data items in parallel coordinates often results in cluttering. Fua et al. [49] proposed a hierarchical clustering approach on parallel coordinates; they rendered representative paths for groups of similar objects, and used color shading cues to indicate the spread of the object groups. Artero et al. [3] constructed frequency or density plots in high-dimensional data so that we can reduce visual clutter while filtering noise and highlighting interested data patterns. Novotny [106] proposed a visual abstraction approach, where colored clusters of data are plotted as regions instead of individual zig-zagging lines. Peng et al. [108] devised a formulation for visual clutter measurement and reduction, and quantitatively analyze visual cluttering in parallel-coordinates plots.

Visual enhancement. Graham and Kennedy [51] proposed using smoothly-graduating curves to replace zig-zagging lines in parallel-coordinates plots, aiming at discerning individual paths through knots. Johansson et al. [87] constructed clusters and applied high-precision textures in parallel-coordinates plots; transfer functions were also proposed to help highlighting different aspects of data clusters. Opposite to clustering, which focuses on frequent data pat-

Figure 4.2: Left: illustrative parallel coordinates, this figure is courtesy from McDonnell and Mueller [101]. Right: 3D parallel coordinates, this figure is courtesy from Johansson et al. [86].

terns, Novotny and Hauser [107] suggested a focus+context approach to detect and highlight outliers in parallel-coordinates plots. Zhou et al. [141] applied a visual clustering approach, where curved lines were used to form visual bundles on clusters in parallel-coordinates plots. McDonnell and Mueller [101] incorporated illustrative visualization techniques to parallel-coordinates visualizations, aiming at enhancing the large-scale structures with visual cues such as silhouettes, shadowing, and halos, see Figure 4.2 left. Wang et al. [129] applied illustrative visualization techniques to choose colors for enhancing the parallel-coordinates visualization. Targeting for small display devices, Shearer et al. [116] studied the use of animation to enhance the traditional parallel-coordinates plots so that we can highlight global trends in the data.

Integrating with other visual representations Siirtola [117] combined parallel coordinates with reorderable matrix, and studied the conceptual linkages between these two visual representations. Fanea et al. [42] integrated parallel coordinates with star glyphs by using the third dimension to connect them together, see Figure 4.3. Johansson et al. [86] extended parallel-coordinates plots to three dimensions so that we can simultaneously show correlations between a selected dimension with all the others in the three-dimensional space, see Figure 4.2 right. Recently, Yuan et al. [139] enriched the parallel-coordinates plots by incorporating scattered points from scatter plots into the space of parallel-coordinates plots.

Figure 4.3: A combination of parallel coordinates and star glyphs, this figure is courtesy from Johansson et al. [42].

Integrating with a variety of data types Wegenkittl et al. [130] proposed extruded parallel coordinates to visualize trajectories in high-dimensional dynamical systems. Berthold and Hall [12] extended parallel-coordinates plots to include fuzzy data in the form of fuzzy clusters. Bendix et al. [11] adopted the layout of parallel coordinates to create a new visualization method, namely parallel sets, for categorical data, while Blass et al. [14] extended parallel coordinates to interactive exploration of large-scale time-varying data sets, including also the contest data appeared in the conferences IEEE Visualization 2004 and 2008.

User interaction Wong and Bergeront [136] proposed a multiresolution brushing technique using wavelets, allowing efficient user controls to identify and highlight data subsets. Hauser et al. [65] later suggested angular brushing, which provided higher flexibility for selecting features across two data dimensions. Ellis et al. [40, 39] devised different methods to measure occlusion in parallel-coordinates plots and applied the magic lenses metaphors with random sampling to allow interactive exploration of cluttered regions in the plots. Siirtola [118] proposed two interaction techniques on manipulating parallel coordinates: polyline averaging for summarizing a set of polylines, and animated correlation coefficients between two neighboring dimensions to emphasize the correlation characteristics.

4.3 Overview

This section has two parts: First, we overview the related steps in our visualization framework. Then, we review some basic properties of parallel-coordinates plots.

Related Steps of Visualization Pipeline. Recall the visualization pipeline in Section 5.3, the input to our system is a set of equations (by a dynamically-linked plugin) that represent a 2D geometric surface in 4D space, Given the input equations, we have the following series of six related steps:

- **Pre-processing** (Section 4.5): #1) *surface construction*: we construct from $X_i(u, v)$ a grid of 4-dimensional points on the given geometry; and #2) *blue-noise sampling*: additionally, we generate also a set of sample points on the surface later for constructing the parallel-coordinates plots.
- **Interactive Visualization** (Section 4.6): #3) *4-dimensional transformation*: 4-D rotation is next applied to transform the pre-computed grid vertices and sample points, followed by an 4-to-3D projection on the grid vertices; and #4) *parallel-coordinates plots*: next, we employ the transformed sample points to generate continuous parallel-coordinates plots for compiling the *visual signatures*.
- **Interactive Exploration** (Section 4.7): #5) *enhanced visual signature*: furthermore, we may compute and incorporate geometric properties such as curvature to further enrich the visualization; and #6) *interactive brushing*: lastly, we can interactively brush both the 3D projection view and parallel-coordinates plots for feature markup and exploration.

Basic Properties of Parallel-coordinates Plots. Parallel-coordinates plot is generally a statistical tool that indicates data distribution over a set of mutually-orthogonal dimensions. Each region in the plot summarizes the data distribution over a corresponding pair of co-dimensions.

Concerning the plotting of continuous geometry, we first present the following basic properties about parallel-coordinates plots.

First of all, we review the point-line duality property of parallel-coordinates plots in N -D space: A point P between axes X_1 and X_2 in a parallel-coordinates plot corresponds to an $(N - 2)$ sub-space in the N -D geometry space. This subspace intersects the $X_1 - X_2$ plane in the geometry space exactly along a certain (dual) line related to point P . This dual line represents all x_1 -

Figure 4.4: Changes in parallel-coordinates plots upon basic transformations, as illustrated by a 2D ELLIPSE.

Figure 4.5: Example correspondences between geometrical properties and visual signatures in parallel-coordinates plots: two simple geometric models are used: a 2D ELLIPSE ((a) and (b)) and a 3D ELLIPTICAL CYLINDER ((c) and (d)).

x_2 coordinates combinations that intersect at P in the parallel-coordinates plot. Hence, we can compute the amount of intersection between the high-dimensional geometry and the dual line of P in the geometry space, and obtain the (pixel) density value at P in the continuous parallel-coordinates plot. Based on this duality property, we could also mark (or brush) on the parallel-coordinates plot, and see the corresponding areas on the geometry in the 3D projection view. Second, the range covered by the geometry in each dimension can be clearly observed in each axis of the plot, see the changes in range when we rotate the 2D ELLIPSE in Figure 4.4(a). Lastly, translating the geometry produces a vertical shift or shearing in the plots, see the following three cases shown alongside in Figure 4.4(b):

- Translating the geometry along X_1 -axis results in a shearing of the plot at the X_1 dimension (but X_2 axis is invariant);
- Translating along $X_1=X_2$ results in a vertical shift; and
- An arbitrary translational direction results in a shearing along both X_1 and X_2 .

4.4 Geometric Meanings of Visual Signatures

This section presents our study on the corresponding implication between geometrical properties and visual signatures in the continuous parallel-coordinates visualizations. Representative visual signatures generated in the plots for a high-dimensional geometry can indicate sum-

Figure 4.6: Comparing different sampling approaches to generate continuous parallel-coordinates plots on the 4D KNOTTED SPHERE model. All approaches use 3000 sample points in 4D, and blue-noise sampling produces the highest-quality visualization.

marized geometry information. In particular, the following visual signatures implications are identified:

- (i) **Vertical symmetry about a horizontal line in the plot** (see Figure 4.5(a)). In this situation, for every data point, say (x_1, y_1) , on the geometry, there must be another data point at $(-x_1, -y_1)$, so that their dual lines always produce a symmetric accumulated density plot about the horizontal line. Hence, the projection of the given geometry in the corresponding 2D subspace (the two axes) must be 180° -rotational symmetric about the origin. For the 2D ELLIPSE shown in Figure 4.5(a), its parallel-coordinates plots always exhibit vertical symmetry no matter how we rotate it about the origin, so it must exhibit 180° -rotational symmetry in the corresponding 2D subspace.
- (ii) **Horizontal symmetry** (see Figure 4.5(b)). Horizontal symmetry between a pair of axes, say X_1 and X_2 , in the parallel-coordinates plot refers to the situation that the continuous density plot between X_1 and X_2 is symmetric about a vertical line midway between the two axes. So the density value of every pair of pixels correspondingly on the opposite side of the vertical line should be the same. Hence, if we have a data point at (x_1, y_1) on the geometry (in dimensions X_1 and X_2), there must be another data point at (y_1, x_1) on the geometry, so that their dual lines produce such a symmetric pattern. Therefore, the geometry must be a *mirror-reflection* of itself about the diagonal line (or the sub-

space it subtends), i.e., $X_1=X_2$. For the 2D ELLIPSE shown in Figure 4.5(b), when we interactively rotate it to a certain orientation, we can see a horizontal symmetric pattern, showing that the geometry is mirror-reflected in the corresponding dimensions.

- (iii) **Symmetry between two neighboring regions in the plot** (see Figure 4.5(c)). The patterns described above can be further generalized for mirror reflection between neighboring regions in the plots. Taking a 3D ELLIPTICAL CYLINDER as an example, see Figure 4.5(c), we observe a horizontal symmetric pattern between regions X_1-X_2 and X_2-X_3 , indicating that the geometry is *mirror-reflected* about the subplane $X_1=X_3$.

The above signature implications are basic hints when examining higher-dimensional geometries, while shifted or sheared vertical symmetry signatures could happen due to a combination of vertical symmetry and geometry translation. As shown in Figure 4.5(d), if we translate the geometry along $X_1=X_2$, the symmetry pattern between axes X_1 and X_2 shifts upward correspondingly.

4.5 Pre-processing

Surface Construction. The first pre-processing step is to construct the surface of the 4-D geometry given its mathematical equations. In detail, we take the user-specified numerical range for parameters (u, v) , and pre-compute a 2D grid of vertices on the geometry surface in the 4-D object space. These vertices will later be transformed and rendered as a quadrilateral mesh for visualization.

Sample Points Generation. Second, we pre-compute a set of sample points on the geometry surface in the 4-D object space for generating the continuous parallel-coordinates plots. As described in [66], the construction of continuous parallel-coordinates plots is a sampling problem [137, 30], and random sampling strategies can be applied to integrate the continuous data domain. In general, we could have different sampling approaches, see Figure 4.6:

Figure 4.7: Interactive parallel-coordinates visualization upon dynamic subspace rotation in 4-dimensions. The three 4D geometric models used here are TORUS, TORUS KNOT-LIKE, and FERMAT SURFACE (from left to right column).

- **Grid sampling.** One straightforward sampling approach is just to take the grid points on the constructed surface as the sample points. However, since the sample points are not uniformly distributed on the geometry surface, it could heavily bias the plotting results, making it invalid, see Figure 4.6.
- **Random sampling.** To achieve more uniform sampling, we could generate sample points by stratified sampling. However, if there are not enough sample points, aliasing problem could occur, see Figure 4.6 (middle).
- **Blue-noise sampling.** To leverage the aliasing problem while minimizing the number of sample points for constructing continuous parallel-coordinates plots, we propose to use blue-noise sampling, which can generate sample points that are stochastically uniform and not too close to one another. Figure 4.6 shows a comparison: blue-noise sampling is able to produce the highest-quality continuous parallel-coordinates plot given the same number of sample points.

There are many different methods for blue-noise sampling, e.g., the classical dart throwing algorithm [104] and adaptive dart throwing [100]. Among them, we choose the recent capacity-

constrained Voronoi tessellation (CCVT) method by Balzer et al. [4] since we can iteratively generate high-quality blue-noise samples without needing to specify the number of iterations as in conventional Lloyd’s relaxation [98]. This feature is critical to our problem because tuning the number of iterations requires the knowledge of the distribution quality in the high dimensional space, but it is not feasible for us to directly observe such distribution quality.

To apply the CCVT method, we first use stratified sampling [126] to generate a very large initial random point set on the geometry surface, and employ direct distances in the 4-dimensional object space to approximate the surface geodesics. This approximation can still produce high-quality sample points since the 3000 sample points generated on the high-dimensional surface are sufficiently dense for covering the surfaces we employed in this work. Note also that when generating blue-noise samples on geometric surfaces, we may adaptively sample the surface according to the geometric features [109]. To do so, we may take a density function adaptation approach that constructs a density function on the surface, e.g., by curvature, to modulate the sampling process, see [4, 95].

4.6 Interactive Visualization

After the two pre-processing steps, see again Figure 7.1, our visualization interface provides the following two interactive views:

View #1: 3D Projection. First, we have a standard 3D projection view, where three out of the 4 dimensions can be selected to be shown in the view. Moreover, we can also select two or three out of 4-D for an interactive 2D/3D subspace rotation using the rolling ball rotation algorithm; such a rotation can incrementally update an 4-D rotational matrix, which is an 4-by-4 orthogonal matrix. In detail, the accumulated 4-D rotation matrix (followed by a translation) is first applied to transform the surface grid points and the blue-noise sample points. Then, we

apply an 4-to-3-D orthographic projection to transform the grid points to the space of the 3D projection view for visualization, see Figure 7.1(top row).

This 3D projection view is the main interactive window for rotating the high-dimensional geometry. By default, the orthographic projection takes the first three dimensions out of 4 with the axes icon (on the bottom left of the view) showing the axes projection. Later, we will incorporate this view with the parallel-coordinates plots to further support interactive exploration.

View #2: Parallel-Coordinates Plots. Besides the 3D projection view, we have another view to show a continuous parallel-coordinates plot. This view involves the following computation: 1) First, we transform the blue-noise sample points from their object space via the 4-by-4 rotation matrix; 2) Then, between each pair of neighboring axes in the parallel-coordinates plot, we develop a GPU fragment shader to compute the density of each pixel by summing up the contributions of each transformed sample point, see Section 5.6 for detail; and 3) Lastly, we use a simple transfer function to color each pixel on the continuous parallel-coordinates plot based on its density value. We also employ Gaussian filtering in the shader to further alleviate the aliasing.

Visualization Scenarios Since we can generate the continuous parallel-coordinates plot interactively, we can perform interactive visualization on a variety of high-dimensional geometries, see Figure 4.7 and 4.8. These geometric models are rotated interactively by the accumulated 4-D rotation in the projection view, and during the interaction, we can see dynamically-changing parallel-coordinates plots of the transformed surfaces.

The geometric models shown in Figure 4.7 are 2-manifolds embedded in 4D, so there are six pairs of co-dimensions altogether. Hence, we can exhibit all six pairs by having two parallel-coordinates plots with axes: $X_1-X_2-X_3-X_4$ and $X_2-X_4-X_1-X_3$. These two plots together compile the *visual signature*. Similarly, our dual-view visulization can be extend to even higher

dimension, e.g., 6D. Take 6D VERONESE model as an example, we can present all the 15 pairs of co-dimensions for the 6D VERONESE model by using three series of parallel-coordinates plots, see Figure 4.8. Note the symmetric patterns exhibit over various pairs of co-dimensions in the visual signatures. As described in Section 4.4, these visual signatures give hints about the geometric properties of the high-dimensional geometry. Lastly, though the above axes arrangement scheme considers all pairs of co-dimensions, it cannot show all possible ways of putting “three” axes consecutively for region-to-region comparison, e.g., putting X_2 in-between X_1 and X_4 in a plot. Hence, our interface further allows interactive swapping and re-ordering of axes by mouse click and drag. In the following, we describe the visualization of the four models shown in Figure 4.7 and Figure 4.8:

1) Torus. The TORUS model shown in Figure 4.7(left) is being rotated about subplane X_2-X_3 , but from the 3D projection view, the rotation axes cannot be observed easily from the axes

Figure 4.8: Interactive parallel-coordinates visualization upon dynamic subspace rotation in N -dimensions of 6D VERONESE model.

Figure 4.9: Using *AND* and *OR* brushing modes on the same set of brush points. We highlight related surface regions in red.

icon, which is compromised but tedious. One may be able to observe that the length of the red axis (X_1) is reducing while the length of the yellow axis (X_4) is increasing with the other two axes being fixed. Having the parallel-coordinates plots, it could be observed that the unchanged axes are X_2 and X_3 , which trivially implies a rotation about subplane X_2-X_3 . This example also demonstrates a visualization aid on one of the geometric properties, i.e., symmetry. During the rotation, we can see that all views of the TORUS model are having a vertical symmetry signature. Recalling signature implication 1 in Section 4.4, this directly implies that the high-dimensional geometry being plotted is 180° -rotational symmetric about the origin for all axes, i.e., symmetric about all axes. Furthermore, the first and last views both exhibit horizontal symmetry signature. Signature implication 2 in Section 4.4 explains this phenomena to be mirror-reflection about all diagonal lines of the axis pairs, which also concludes the complete symmetry property of TORUS.

2) Torus Knot-like. We rotate the TORUS KNOT-LIKE model shown in Figure 4.7(middle) about subplane X_1-X_3 . It is explicit that the X_1-X_3 coordinate pair region in the $X_2-X_4-X_1-X_3$

ordered parallel-coordinates plots remains unchanged, no matter what the rotation angle is. Moreover, the X_2-X_4 coordinate pair region of the other ordered plots remains vertical symmetric during the entire rotation. This implies that the geometry is 180° -rotational symmetric about X_2 and X_4 axes (signature implication 1 in Section 4.4). Despite the fact that this geometry is complicated, we can still quickly visualize and conclude the corresponding symmetry features from the parallel-coordinates plots. By just examining the projection view of the TORUS KNOT-LIKE, it is difficult to notice that this geometry has symmetric property about X_4 , while its respective parallel-coordinates plots can explicitly conclude this.

3) Fermat surface. We rotate the FERMAT SURFACE shown in Figure 4.7(right) about subplane X_1-X_2 . From the parallel-coordinates plots, it could be observed that the X_1-X_2 coordinate pair region of the $X_1-X_2-X_3-X_4$ plots remains unchanged throughout the rotation. Similar to the TORUS model, the FERMAT SURFACE always exhibit vertical symmetry signature during the rotation in both ordered plots. This reveals that the FERMAT SURFACE is completely symmetric based on signature implication 1.

4) Veronese. We rotate the 6D VERONESE model shown in Figure 4.8 *within* subplane X_2-X_6 . Hence, we could observe from the plots that only the regions that involve X_2/X_6 are changing; other regions, e.g., X_3-X_4 and X_4-X_5 , remain unchanged. Moreover, region X_4-X_5 in the $X_1-X_2-X_3-X_4-X_5-X_6$ plots is vertical symmetric. This implies that the geometry is 180° -rotational symmetric about X_4 and X_5 axes based on signature implication 1. Furthermore, we could observe that there are symmetry between X_5-X_1 and X_1-X_4 regions in the $X_3-X_6-X_2-X_5-X_1-X_4$ plots (the rightmost portion of the plots); this implies that the 6D VERONESE model is *mirror-reflective* about subplane $X_5=X_4$ based on signature implication 3 in Section 4.4.

a larger area of markup on the geometry surface in the 3D projection view, and by this mode, we can highlight all geometric points that pass through a certain data range.

Figure 4.9 shows the effect of these two brushing modes with the same set of brush points on the parallel-coordinates plot of PILLOW. During the interactive brushing, corresponding surface regions are highlighted in the 3D projection view and the intensity of highlight depends on the proximity in the intersection computation.

Enhanced Visual Signature. To enrich the visual exploration, we could also compute geometric properties such as curvature over the parametric surface in N -D, and employ them to illustrate both the surface in the 3D projection view and the visual signature in the parallel-coordinates plots. As shown in Figure 4.10, we use red colors to indicate high curvature regions on the surface in the 3D projection view and mark the sample points that are close to the high-curvature regions as solid green lines in the parallel-coordinates plots together. This helps enhance the visual signatures. Furthermore, we can also interactively brush the plots (see last row in Figure 4.10) to reveal correspondence between surface and plots. Other than mean curvature, we can also select to show min or max curvature, see Figure 4.11. Here we use cyan and pink lines instead of green in case of max and min curvature, respectively.

4.8 Implementation Detail

We developed our visualizations by OpenGL and GLSL. After the pre-processing steps, the grid vertices and sample points are stored as 2D textures on the GPU memory, so that we can efficiently retrieve and transform them on the GPU. The parallel-coordinates plots are generated by fragment shaders, where we compute the distance from each pixel in the plot to every dual line represented by the blue-noise sample points. If a distance is below a user threshold (which is a GLSL uniform variable), we Gaussian-filter the distance and accumulate it into the density value (initialized as zero) of the pixel. Lastly, we apply a simple transfer

Figure 4.11: Other than mean curvature (see Figure 4.10), we can also choose to show min or max curvature.

function at the end of the fragment shader to map the accumulated density value (together with the curvature value from the closest high-curvature dual line w.r.t. this pixel, if curvature mode is enabled) to compute the final color of the pixel in the plot.

4.9 Summary

This chapter presents methods to compile, interpret, and explore visual signatures in continuous parallel-coordinates plots of high-dimensional geometries. First, we propose and show how blue-noise sampling can improve the efficiency of generating continuous parallel-coordinates plots. Then, we discuss various geometric patterns that could be identified in the continuous parallel-coordinates plots. Furthermore, we demonstrate novel ways of interactive visualizations and explorations, including interactive brushing and enhanced visual signatures. We implement our method on the GPU and real-time performance can be achieved. Lastly, several visualization examples and a preliminary study are also presented.

Chapter 5

Enhancing the 4D visualization by Volumetric Halos

Halos have been employed as a compelling illustrative hint in many applications to promote depth perception and to emphasize occlusion effects among projected objects. We generalize the application of halo methods from the widely-used domain of 2D projections of 3D objects to the domain of 3D projections of 4D objects. Since 4D imaging involves a projection from 4D geometry (such as a surface with 4D vertices) to a 3D image, projection typically produces intersecting surfaces, and thus occlusion phenomena result in apparent curves in 3D space. Adding volumetric halos to the surfaces then gives useful information about the spatial relations of intersecting surfaces (just as halos do in 3D-to-2D projections), and allows a more accurate perception of the geometry. A typical application is knotted spheres embedded in 4D, and the volumetric halos perform the same function as traditional knot diagrams do in 2D drawings of 3D knotted curves. To achieve real-time updating of the halo-enhanced image when the geometry is interactively rotated in 4D, we design a series of GPU-based algorithms to reduce the memory load and to accelerate the computation. Furthermore, various enhancements are supplied in the visualization interface, including color-coding, heuristic lighting, texturing, and stereoscopic viewing.

5.1 Introduction

Occlusion is an important depth cue, providing information about relative depth, proximity, and spatial arrangement of the objects in a given view. A refinement of occlusion is the *crossing diagram* of a set of curves embedded in 3D space, an illustrative technique that adds schematic emphasis to the 2D image by cutting away small neighborhoods of each occluded segment; viewer perception of the 3D features in the 2D image is often significantly enhanced by this method. In this chapter, we explore the generalization of 3D crossing diagrams to produce schematically emphasized occlusion diagrams for surfaces in 4D space projected to a 3D image. There are many methods to produce occlusion diagrams of 3D curves as well as 4D surfaces. We confine our attention here to the *halo method* [2], which requires volume rendering methods to produce the desired crossing diagrams of 4D surfaces projected to a 3D volume. Just as ordinary knots are a common useful domain for crossing diagrams in 2D images of 3D objects, knotted spheres [56, 31, 61, 27, 81] are a classic example of a 4D visualization problem that profits from the exploitation of crossing diagrams in the rendering process. In addition to explaining how to generate and render 4D halos, we also show how this potentially slow and difficult rendering process can be made interactive by exploiting high-performance GPU methods, thus facilitating intuitive user exploration of the fourth dimension.

Illustrative rendering techniques such as halos and image sharpening can significantly improve the perception of depth and occlusion. Typical halo methods [2, 84] render halos around lines or object silhouettes, emphasizing the occlusion among projected objects on the image screen. Similarly, we can employ any available depth information to sharpen and emphasize the properties of an object in its rendered image [99], and to locally enhance the image color and contrast around object silhouettes. Thus we can both emphasize foreground objects to make them more recognizable, and reveal the spatial relations among projected objects. These rendering techniques are also commonly used by artists to strengthen object perception in their drawings, especially for illustrative figures in a technical context.

Figure 5.1: Surface patches at different 4D depths in w projected to 3D image: (a) Oblique 4D view projected along the direction $(x, y, z, w) = (1, 0, 0, 1)$, which shows the physical separation in w ; (b) 4D view along $(0, 0, 0, 1)$, producing 3D self-intersection, with no halo; (c) 3D self-intersecting view enhanced by depth-dependent 4D volumetric halo method.

Our Approach. 3D geometry can be visualized by projecting and rendering to a 2D image plane, while 4D geometry typically requires projecting and rendering to a 3D image volume. The result may require further processing techniques like volume rendering to produce a 2D screen image. During the 4D to 3D projection, 4D depth information should not be discarded, but should be retained for the same reasons we keep 3D depth information while projecting 3D information to 2D. Apparent surface intersections in the 3D volume image are normally projection artifacts and not intrinsically intersecting geometric structures in the original 4D data; illustrative enhancement of these false intersections should be just as useful as haloing in 2D images [41].

Advances in graphics hardware capabilities have provided us with the ability to replace offline animation by 4D algorithms that achieve previously impractical real-time performance (see,

for example, Chu et al. [27]). This work similarly employs GPU-based shader programs to support interactive halo rendering for 4D scenes with significant occlusions in the projected 3D volume image.

Following the spirit of [2, 20, 41], we design and implement halo-based illustrative visualization methods to annotate the rendering of 4D geometry such as curves and surfaces after projecting them from 4D to 3D. Figure 5.1 presents an elementary example of the desired results for five 4D colored surface patches whose 3D projections intersect the white patch, but that have different 4D depths relative to the white patch. In addition to the basic halo-based crossing diagram cutaways shown in the Figure, we will be able to include additional perceptual cues such as motion parallax (thank to fast GPU rendering), color-coding, heuristic lighting, texturing, and stereoscopic viewing.

Contributions of This Chapter.

- We present a comprehensive volume-rendering-based approach to the creation of halo structures for arbitrary curves and surfaces embedded in 4D and projected to a 3D volume image.
- Early volume-based 4D visualization work [59] was limited by the complexity of processing 4D objects and rendering them into a 3D image, that, in turn, had to be volume-rendered into a 2D screen. We have extended the fast GPU-based methods of Chu et al. [27] to include volumetric halos to support interactive exploration of 4D haloed occlusion and depth.
- Redundant information cues, including color-coding, heuristic lighting, texturing, and stereoscopic viewing, are supported to enhance the perception of spatial relations among projected objects.

Figure 5.2: Enhancing visualization with halos: (a) Emphasize the occlusions of geometries by halo technique, this figure is courtesy from Appel et al. [2]. (b) Visualize magnetic field data with halo enhanced rendering, this figure is courtesy from Schussman et al. [115]. (c) Enhance volume data visualization by halo effect, this figure is courtesy from Bruchner et al. [20]. (d) Rendering dense line data with depth-dependent halos, this figure is courtesy from Everts et al. [41].

- Tetrahedron mismatch is a challenging technical problem in the creation of the auxiliary halo 3-manifolds needed to implement our approach; we develop novel algorithms to resolve the computational geometry issues that occur in this process.

5.2 Enhancing Visualization with Halos

The utilization of halo methods in computer graphics (other than knot diagrams) goes back at least to the late 1970's when Appel et al. [2] proposed using halos in computer graphics rendering to emphasize the occlusions between lines and surfaces in various geometries (see Figure 5.2a). After that, Interrante and Grosch [84] introduced several novel halo-based techniques to illustrate 3D flow using volume line integral convolution to convey directional information. They defined a continuous 3D visibility-impeding halo region that fully enclosed the streamlines in the 3D texture. Schussman et al. [115] proposed various perceptually-effective techniques to visualize magnetic field data from the DIII-D Tokamak (see Figure 5.2b), while illumination and haloing are carefully integrated together to enhance the visual perception. Ebert and Rheingans [37] combined the strengths of direct volume rendering with the expressive power of non-photorealistic rendering to create volume-based feature halos. Svakhine

Figure 5.3: Halo illustrations for surfaces in 4D: (a) the surfaces are first projected from 4D space to 3D screen space; (b) view-oriented thickening is then applied to generate tetrahedra for the thickened volumes; (c) we voxelize the tetrahedral volumes and fill a 3D depth buffer with 4D depth values; (d) the final halo illustrations are produced by rendering the surfaces with an occlusion test; (e) other visualization effects, such as heuristic lighting, can also be supported in our visualization pipeline.

and Ebert [122] later developed an efficient implementation on programmable graphics hardware for feature halos. Luft et al. [99] developed a halo-like approach to enhance the perceptual quality of images by using depth information, and Bruckner and Gröller [20] proposed a wide variety of GPU-based halo effects, such as emissive and occlusive halos, to enhance the depiction of volumetric data (see Figure 5.2c). More recently, Everts et al. [41] suggested depth-dependent halos for effective illustrations of dense line data, such as DTI fiber tracks (see Figure 5.2d).

5.3 Overview of Our Approach

Projecting 3D curves into a 2D screen can easily produce occlusions, which are pixel locations where multiple curve segments lie on a single line of sight, and thus share a single pixel in the projection. While a depth buffer ensures the correct choice of the nearest pixel, distinguishing the relative depth between neighboring pixels remains a problem. This can be remedied by employing halo methods: gaps are introduced in screen space, enhancing and accentuating the rendered segments nearer to the viewer (see the illustration).

Table 5.1: Comparison of halo illustrations between 3D and 4D graphics.

Space	Geometry	Screen	Halos	Primitives
3D	0D/1D manifolds	2D pixels	2D	triangles
4D	1D/2D manifolds	3D voxels	3D	tetrahedra

The situation in 4D space for surfaces projected into a 3D screen is exactly parallel to the 3D context. Two surfaces may appear to penetrate each other in the 3D screen, and yet be quite comfortably separated in actual 4D space.

The major task in this chapter is to generalize the halo procedure, providing

illustrative visualization emphasis for 4D depth relations among apparently coinciding surface projections in the 3D screen. Table 5.1 compares the halo illustration paradigm for 3D and 4D graphics. Triangles are used as halo primitives in the 2D screen for 3D graphics, and tetrahedra are required to model halos in the 3D screen for 4D graphics.

Figure 7.1 outlines the volumetric halo process. We start with two surfaces that are separated in 4D space but appear to intersect each other after the 4D to 3D projection (see Figure 7.1(a)). To determine the halo regions, we first apply a thickening process to the projected surfaces to extend them into a 3D volume (see Figure 7.1(b)). We note that our thickening process is very different from that of Hanson and Heng [57], designed to support 4D lighting. Our thickening is done in 3D screen space rather than in 4D space, is view-dependent, and more efficient for haloing, though it supports only a heuristic variant of rigorous 4D lighting. Hence, we need to perform thickening in real-time for each change in viewpoint or projection. After thickening, we can construct tetrahedra as needed to populate the space occupied by the halos and to fill each voxel in a 3D depth buffer with its 4D depth value during the volume rasterization process (see Figure 7.1(c)). Finally, we can render the projected surfaces in the 3D screen space with a proper occlusion test against the 3D depth buffer, thereby producing volumetric halos in the visualization (see Figure 7.1(d)). Various visualization effects (see Figure 7.1(e)) are also supported to further enrich the illustration of 4D geometry.

5.4 The Illustrative Visualization Pipeline

The illustrative visualization belongs to the fifth step in the comprehensive 4D visualization framework. It makes full use of modern graphics hardware to achieve real-time and high-quality halo illustrations. In this section, we detail each major component of the sub-pipeline about illustrative visualization, and use surfaces (2-manifolds) in 4D as a running example.

4D Geometry Generation

Our *vertex program* is designed to generate the 4D geometry directly on the GPU. Similar to the case of mathematical surfaces in 3D, we can employ uv -parametrized equations to define surfaces in 4D. In practice, we program these equations in the *vertex program*, thus allowing us to send only a small number of parameters to the GPU to construct the surface geometry; the GPU can generate the necessary geometry data internally. Furthermore, instead of uniformly sampling the uv space, we adaptively sample and construct the underlying surface according to its local curvature and pre-compute two lists of u and v coordinates (with their connections to form triangles) so that the *vertex program* can help compute the corresponding 4D vertex positions. In this way, we can better balance the visualization quality and performance.

For each 4D vertex position on the surface, we pre-compute a pair of 4D tangent vectors, say $\vec{t}_1$ and $\vec{t}_2$, and project them to the 3D image. Their cross-product produces a normal direction that we can use for thickening as well as heuristic lighting. We also store the u and v values for texturing purposes. Finally, all these per-vertex data attributes are stored using a vertex buffer object (VBO) to reduce the burden of transferring data between the CPU and the GPU. Note that we need to perform the computation in this stage only once unless the 3D geometry changes.

Transformation and Projection

The bare computational framework for rendering 4D geometry is based on a 3D (volumetric) image buffer and a 3D depth buffer. Just as 3D rotations produce different 2D screen images of a 3D scene, 4D rotations of a 4D scene produce different 3D volume images. Note that while 3D rotations have 3 degrees of freedom (e.g., rotations in the planes yz , zx , and xy), 4D rotations have 6 degrees of freedom (e.g., rotations in the planes wx , wy , wz , yz , zx , and xy). Moreover, the 4D projection can be orthographic or perspective just as in 3D. The 4D transformations and projections are per-vertex operations (see Figure 5.4(a)) implemented with a specific *vertex program* that computes the operations and stores the results along with 4D depth information.

Thickening Mechanism

To construct a model for the halo regions, we thicken the projected surface and create the 3D generalization of the standard 2D wing-like extension. Since the rendering primitives are now triangle patches rather than line segments, the thickened geometry is a thin triangular prism corresponding to each triangle of the projected surface in the 3D screen.

Mathematically, we can derive the thickening direction $\vec{n}$ by taking a 3D cross product of the two projected surface tangents $\vec{t}_1$ and $\vec{t}_2$ after applying any transformations. There are no surviving 4D components at this point, so

$$\vec{n} = \vec{t}_{1,xyz} \times \vec{t}_{2,xyz} . \quad (\text{Eq. 5.1})$$

Note that the thickening direction $\vec{n}$ is dependent on the 4D viewing projection because the 3D projections of $\vec{t}_1$ and $\vec{t}_2$ change with the 4D rotation during interactive exploration.

During the thickening process, we create two vertex instances for each vertex on the original projected surface: positive or negative, according to whether it lies on positive or negative side

Figure 5.4: Modeling halos from a triangle primitive: (a) the triangle is first transformed and projected into 3D screen space; (b) we then thicken the triangle by offsetting it to both positive and negative sides of $\vec{n}$; (c) we decompose the extruded volumes into tetrahedra; (d) the tetrahedra are voxelized by planar slicing; (e) the halo generation algorithm is applied and the volume slice is rendered.

of $\vec{n}$ (see Figure 5.4(b)). Hence, we can form two offset surfaces for each group of vertex instances over the projected surface. After that, we attach a per-vertex parameter t that takes a value of 1 or 0 for vertex instances on positive or negative offset surface, respectively. Now, we can also define an interactively-controllable parameter ρ to adjust the overall size of the halos:

$$\vec{v}_{new} = \vec{v}_{orig} + (2t - 1) \rho \vec{n},$$

where $\vec{v}_{orig}$ and $\vec{v}_{new}$ are the original and extruded coordinates on the projected and offset surfaces, respectively.

Modeling Halos with Tetrahedra

After thickening, we next construct tetrahedra to model the volumetric space occupied by the halos. Instead of creating two separate layers of tetrahedra, we bypass including the projected surface in the middle and directly connect the two offset surfaces, reducing the number of tetrahedra needed in the pipeline. Note that by interpolating the t parameter, we can determine the offset distance from the projected surface to any point inside a halo volume.

We join each pair of corresponding triangles on the two offset surfaces into a triangular prism, and then decompose each prism into three tetrahedra (see Figure 5.4(c)). Altogether, there are eight possible ways of decomposing the faces of a triangular prism into triangles, only six of them leading to valid tetrahedra; applying a given decomposition to all prisms without checking neighboring faces will typically produce face mismatches similar to the T-join problem in triangular meshes (see Figure 5.6). We show how to resolve mismatched faces in subsection 5.6.

We define a dedicated *geometry program* in the GPU to perform the thickening and tetrahedralization. The input to this program is a single triangle with three ordered vertices transformed and projected by the *vertex program* of subsection 5.4. The *geometry program* ultimately outputs the geometry as a list of three tetrahedra for each input triangle. These tetrahedra can either be sent directly to the next stage of our visualization pipeline, or pre-stored in another *VBO* before the next step. Having another *VBO* as a temporary buffer can avoid unnecessarily re-generating the tetrahedra for different planar slices when rendering the halos in the next stage, thus improving the overall performance.

Depth-dependent Volumetric Halos

Our volumetric halos adopt the depth-dependent property from Everts et al. [41] so that we can vary the gap size according to the difference of 4D depth values between the projected surfaces.

Figure 5.1 illustrates how this idea is generalized in our volumetric halos. Given the white surface at $w = 0$, and five colored surfaces at varying w depths, the projection from an oblique view shows no intersections (see Figure 5.1(a)); changing the projection as in (b) and rendering without halos produces apparent intersections and gives no hint of the true geometry. By adding volumetric halos (see Figure 5.1(c)), we can generate gaps around surface intersections according to their relative 4D depth. Hence the spatial relations in actual 4D space can be highlighted, e.g., we can see that the red surface is nearest to the 4D viewer, while the blue surface is the farthest from the 4D viewer.

To produce this depth-dependent effect with volumetric halos, we first need to properly compute a 4D depth value for every voxel inside the halo volumes, i.e., inside the tetrahedra we generated previously. This is analogous to the case of depth-dependent halos in 3D graphics, where we increase the 3D depth values across the winged region around a 1D line on the 2D screen. By using this depth increment strategy, we can fill the depth values for regions occupied by the halos, and use these extrapolated 4D depth values to eliminate any object fragment that is shadowed by the halos. We increase the 4D depth value of each voxel (within the halo volumes) according to its offset distance from the projected surface. Given a point p with a 4D depth value of d_{orig} on the projected surface, we use the following formula to compute the new 4D depth value d_{new} at any point extruded along $\vec{n}$ from p :

$$d_{new} = d_{orig} + \max(0, |2t - 1| - \delta t) \delta d, \quad (\text{Eq. 5.2})$$

where δd controls the drop-off rate of the depth value, and δt controls the size of the middle region having the same constant 4D depth value as the point p .

Producing the Halo Visualization

Conceptually, we can rasterize the tetrahedra one by one into the 3D screen (see Figure 7.1), and perform a z -depth-test by making use of the associated 3D depth buffer with the 4D depth

values from the halo volumes. Hence, we can eliminate any surface fragment that is shadowed by the halos. Volume rendering methods are applied in the final step to composite and display the 3D volume image on a standard 2D screen.

We can in fact avoid using the additional 3D depth buffer to reduce the memory overhead. We employ a multi-slice approach to generate and render halos in our visualization pipeline by composing the 2D slice images from back to front to produce the overall volume rendering. To improve the performance, before we render the slices, we first pre-compute the z ranges of groups of tetrahedra in the 3D screen space so that during the multi-slice processing, we can quickly determine if a tetrahedron intersects the current slice being considered; thus we can quickly prune away a large number of non-intersecting tetrahedra.

Figure 5.4(d) illustrates the slicing process, which is performed on each tetrahedron by a *geometry program* (different from the one for thickening). Given a slicing plane and a tetrahedron, we first compute the points where the tetrahedron edges intersect the slicing plane. Since our slicing planes are all perpendicular to the z -axis in the 3D screen space, this step can be quickly done by computing the differences of the z components of all tetrahedron vertices and checking the signs. After we identify the intersecting edges (with opposite signs for the two end points), we can interpolate various per-vertex parameters along the related edges, including the 4D depth value and the t value. Then, we can output a triangle strip from the *geometry program* representing the intersection footprint, which could either be a quad or a triangle. Note that during the thickening process, the vertex instances on the two offset surfaces take the same 4D depth values as their original counterparts on the projected surface. Hence, interpolating the 4D depth values is equivalent to interpolating them over the projected surface, giving a proper 4D depth value to be used as d_{orig} in Equation (Eq. 5.2).

After applying the *geometry program*, we interpolate the per-vertex attributes over the footprint triangle in the rasterization, so that each pixel (voxel) fragment receives its own piece of

interpolated data that is to be further processed by a *fragment program*. By then, we can compute Equation (Eq. 5.2) in the *fragment program* and fill a 2D depth buffer properly with the 4D depth values corresponding to the halo volumes on the slice being considered. By checking the interpolated t value at the pixel fragment (its proximity to 0.5), we can then properly render the projected surface with halos, and generate a slice image with anti-aliasing (see Figure 5.4(e)). At the end, we can progressively compose the slice images to produce the overall volume rendering.

5.5 Extensions

Halos on Curves in 4D (1-Manifolds)

Our thickening mechanism for 2-manifolds in 4-space can be extended to handle 4D embedded curves, i.e., 1-manifolds. In general, there are few applications for halos of 4D curves, since this is the analog of having a *point* shadow a curve in a 3D world. However, it turns out that the intersection of a plane with a halo-enhanced self-intersecting surface produces self-shadowing curves that occur naturally in the study of the structure of knotted spheres. So even though the smallest deformation of the curve on the surface moves away from the region in which curve-curve halos occur, a curve lying *exactly* in a slicing plane can exhibit informative halo structure.

We begin by projecting the curve to 3D and thickening each line segment of the curve into a tube as shown in Figure 5.5. Next, we compute the tangent vector to the projected curve at each vertex and determine the null space of the tangent by applying the Gram-Schmidt process. The result is a pair of 3D vectors normal to the curve, say $\vec{n}_1$ and $\vec{n}_2$, at each vertex. Since $\vec{n}_1$ and $\vec{n}_2$ are both perpendicular to the 4D viewing vector, which is assumed to be $(0, 0, 0, 1)$, the w component of $\vec{n}_1$ and $\vec{n}_2$ must vanish. We can thus again perform the thickening process,

Figure 5.5: Thickening process for curves in 4D: the line segment (left) is first thickened into a tube (middle), then corresponding pizza slices at the two ends of the tube are connected into triangular prisms, each of which is further decomposed into three tetrahedra (right).

this time for a 4D curve, in the 3D screen space. We orient a disk with radius ρ at each vertex by aligning it with $\vec{n}_1$ and $\vec{n}_2$. Then, we connect neighboring disks and generate a triangular prism for each pizza slice on the disk (see Figure 5.5 (middle)). Since these prisms are connected cyclically, we can avoid the face mismatching problem by matching the triangle patterns between each pair of neighboring prisms during the decomposition (see Figure 5.5 (right)). Finally, we take the tetrahedra into the pipeline by attaching a t value from 0.0 at the curve itself to 1.0 on the boundary tube.

Other Illustrative Features

We can enrich the halo visualization by incorporating additional illustrative features into the pipeline. These features are computed in the same *fragment program* that computes the 4D depth values; we can render the projected surface directly by checking if a fragment's t value is close to 0.5 (see subsection 5.6 for details). For each slice, the projected surface and halo occlusion computation are rendered in a single pass supported by 2D depth buffering.

Supported enhancements include the following:

- **Heuristic lighting.** Rigorous 4D lighting can be accomplished using the methods of [59, 56, 31, 27], which involves full computation of 4D normals or sophisticated limits thereof. In this application, we focus instead on heuristic illustrative lighting, and use a simplified approximation that incorporates *only* the projected 3D components of the 4D normals, yielding what might be called a “sketch” of the lighting features. The resulting illustrative representation provides a surprisingly useful emphasis of salient properties of the surface.
- **Texturing.** Texture coordinates can be propagated and interpolated through the pipeline, and applied via standard texturing methods to produce a fragment color on the projected surface.
- **Translucency.** Illustrative translucent effects can be incorporated into the *fragment program* to reveal the internal structure of the 3D volume image of the 4D geometry. This see-through effect is useful when there are massive intersections in a complicated surface.
- **Color-coding.** Depth values in 3D or 4D can be encoded using local surface color maps. Depth encoding in the example images is produced by linearly interpolating from red to green for small to large depth values.
- **Stereoscopic viewing.** Because our results are 3D volume images, a 2D screen image does not provide complete information about the structure of the projection from 4D except through 3D motion parallax. A better perceptual representation of the depth structure is provided by stereoscopic images of the full volume, including the haloed occlusion features. We support stereoscopic imaging using cross-eyed pairs, wall-eyed pairs, and red(L):blue(R) anaglyphic imaging.
- **Emphasizing the halo boundaries.** By checking the t value in the fragment shader, we can determine the boundaries of the halos in the resulting 3D volume images, and then

draw these linear outlines to emphasize the halo illustrations.

5.6 Implementation and Results

Implementation Issues

Our visualization pipeline is implemented using OpenGL and GLSL. In addition to exploiting the shader units, including vertex, geometry and fragment shaders, we employ the following techniques to support efficient computation in the graphics hardware.

- **Vertex buffer objects.** To reduce geometry transfer between the CPU and GPU, and avoid data re-computation in the pipeline, VBOs are used to cache the 4D geometry data in the GPU. We also store the z ranges of groups of tetrahedra so that only the tetrahedra that overlap with the z value of the current slice are recalled and assembled in the *geometry program* during the multi-slice processing, thereby reducing the intensive computation on tetrahedron-slice intersections.
- **Decomposing triangular prisms.** As mentioned earlier in subsection 5.4, if we randomly decompose all the triangular prisms, we will likely encounter face mismatches between some neighboring prisms; typically, there are two types of face mismatches (see Figure 5.6(a) and 5.6(c)). The challenge here is that each triangular prism may contact as many as three other prisms constructed between the two offset surfaces. Hence, when we try to match the triangle pattern between a certain pair of prisms, we need to keep the matched triangle patterns for the other four faces on these two prisms. Otherwise, we may create new mismatches somewhere else when we fix the face mismatch between the current prism pair.

In the first case shown in Figure 5.6(a), we can fix the face mismatch by simply changing the way we decompose the red prism (see Figure 5.6(b)). The outer faces on the red prism

Figure 5.6: Resolving face mismatches between neighboring triangular prisms: (a) a common type of face mismatch; (b) fixable by reversing the construction of the red line in the middle rectangle; (c) another type of face mismatch that is not fixable by (b); (d) fixable by adding new vertices and reconstructing the red prism with a new decomposition.

can remain unchanged but the triangle pattern on the shared face can be flipped; this is a common occurrence. In the second case shown in Figure 5.6(c), it is more complicated because there are no ways of decomposing the red prism that can allow us to resolve the face mismatch while keeping the triangle patterns on the outer faces. To resolve this case, we need to introduce new vertices to subdivide the red prism into three smaller prisms (see Figure 5.6(d)) so that we can keep the desired triangle patterns on the outer faces that are originally on the red prism. Note that the second case can potentially increase the tetrahedron count in the halo modeling; to help avoid this, we use a breadth-first approach to decide the ways we decompose the prisms, rather than randomizing the

Figure 5.7: SPUN TREFOIL: (a) traditional rendering; (b) translucent rendering; (c) result from [27]; (d) our result.

decompositions and then fixing them with the two methods. We first randomly pick a prism as the seed and decompose it into tetrahedra; then we visit its neighboring prisms in a breadth-first manner, trying to apply method 1 to match the triangle patterns between all visited prisms and applying method 2 only when method 1 fails.

- **Rendering the projected surface.** To render the projected surface, we can check the pixel fragment's t value and render the polygonal level set at $t = 0.5$, which is the original projected surface. To achieve a soft halo transition effect, we can adjust the fragment color and transparency according to the distance from the projected surface:

$$t' = |2t - 1| \quad (\text{Eq. 5.3})$$

$$\alpha = \begin{cases} 1.0 & \text{if } t' \leq t_L \\ (t_H - t') / (t_H - t_L) & \text{if } t_L < t' \leq t_H \\ 0.0 & \text{otherwise,} \end{cases} \quad (\text{Eq. 5.4})$$

where t_H and t_L are tuning parameters that control the width of volumetric halos; they are set to be 0.5 and 0.1 for all the running examples in the chapter, respectively.

Performance

To evaluate the performance of our illustrative visualization sub-pipeline, we tested it with three 4D geometric models (QUADRATIC FERMAT, SPUN TREFOIL and TWIST-SPUN-TREFOIL KNOT) on a desktop PC with Inter (R) Core (TM) i7 CPU 960 3.20GHz, with 4.096GB memory, and a *Geforce GTX580* graphics card with 1536MB memory.

Table 5.2: Performance of our illustrative visualization sub-pipeline (FPS).

Model / Num. Slices	64	128	256	512
QUADRATIC FERMAT	69.97	57.20	43.73	25.64
SPUN TREFOIL	69.16	55.31	42.67	24.30
TWIST-SPUN-TREFOIL KNOT	65.45	49.35	35.32	20.98

Table 5.2 shows the performance results in frames-per-second (FPS). “*Num. Slices*” indicates the number of slicing planes in the multi-slice processing, which is an important factor affecting both the overall performance and image quality; we usually need around 256 slicing planes to achieve reasonable quality for most results presented here. In addition, the size of the rendering viewport also influences the performance of our visualization pipeline, although it is not as significant as the slice number. Here the viewport size is set to be 512×512 for all the test cases. It is clear from the table that we still achieve real-time performance even with a large number of slicing planes.

Results

In this subsection, we demonstrate and explain the results of halo visualization with various 4D geometric models:

- SPUN TREFOIL. Figure 5.7(a) shows the result of rendering the spun trefoil without haloed occlusion cutaways; the translucent effects added in Figure 5.7(b) do not provide much help on showing the spatial relations. Figure 5.7(c) is an expensive fully 4D-illuminated volume rendering with 4D occlusions but no expansion of the occlusions using halos. The volumetric halos in Figure 5.7(d) emphasize the 4D occlusion in the same manner as gaps in a 3D knot-diagram visualization, carving out holes in surface parts that are shadowed by other parts in the projection.

Figure 5.8: QUADRATIC FERMAT: (a) volume rendered constant color effect and pinch points marked with white arrows; (b) texturing effect.

- **QUADRATIC FERMAT.** Figure 5.8 shows a 2nd-order FERMAT surface given by $(z_1)^2 + (z_2)^2 = 1$ in $CP(2)$. There are two pinch points (see the white arrows) at which haloed cutaways begin, starting with zero 4D depth difference and monotonically increasing away from the pinch point; this is a particularly clear illustration of the value of having the 4D depth of the halo increasing with the distance from the occlusion curve. Figure 5.8(a) volume-renders the surface, highlighting the high density regions, and Figure 5.8(b) shows a texturing effect with patterns varying across the two components of the surface
- **CALABI-YAU QUARTIC CROSS-SECTION.** This manifold, the K3 cross-section, is a relatively more complicated 4th-order manifold in $CP(2)$, with $4^2 = 16$ surface patches and numerous intersections in the projected image. Figure 5.9 shows our illustrative visualization results with volumetric halos. Figure 5.9(a) shows the depth effect of the volume rendering method, while Figure 5.9(b) uses color-coding (interpolating from red to green for small to large 4D depth values) to provide additional 4D relative depth

Figure 5.9: CALABI-YAU QUARTIC CROSS-SECTION: (a) volume rendering effect; (b) color-coding with 4D depth values.

Figure 5.10: The degree two $CP(2)$ PRODUCT POLYNOMIAL: cross-eyed stereo.

Figure 5.11: TWIST-SPUN-TREFOIL KNOT: cross-eyed stereo and halo boundaries highlighted in green.

Figure 5.12: Halo illustrations of 4D curves: (a) curve from planar slice of the SPUN TREFOIL (color-coded with red denoting 4D depth values nearest to the viewpoint); (b) curves on a degree two FERMAT surface (color-coded with red denoting the 3D depth values nearest to the viewpoint); (c) anaglyphic stereo view of (b), for use with red-blue glasses.

information.

- **PRODUCT POLYNOMIAL.** Figure 5.10 shows the degree two product polynomial surface in $CP(2)$ described locally by the complex equation $(z_1)^2(z_2)^2 = 1$. The stereoscopic viewing provides better perception about the spatial relations among different surface patches in the projected volume image, and the depth-dependent volumetric halos further show the occlusion in the original 4D space.
- **TWIST-SPUN-TREFOIL KNOT.** Figure 5.11 illustrates another complicated 4D model, based on adding a twist to the trefoil knot used to generate the topological sphere. Curiously, this surface is *not* a knot — unlike the spun trefoil, this can be deformed smoothly to an ordinary sphere. Even when transparency is combined with the halo cutaways to show the internal structure of the geometry, it is still quite hard to get a clear mental map of the model due to the complexity of the internal intersections. A cross-eyed stereo representation with annotated cutaway curves is shown to assist in extracting the structure.
- **4D CURVES.** Taking a planar slice through the spun trefoil and the degree 2 Fermat surface produces specific situations with self-occluding curves; rendering these curves as in Figure 5.12 with the curve-specific haloed occlusion technique helps annotate and clarify the structure of the underlying surfaces. Note in Figure 5.12(b) and 5.12(c) that the 2D image of the Fermat surface slices shows both a pair of haloed intersections (having the same position in the 3D projection) and a pair of non-haloed intersections (having distinct, non-intersecting 3D geometry, as can be seen in the anaglyphic stereoscopic image). Note that we use a red-blue filter here with the red filter for left eye, and that the 3D depth is color-coded in Figure 5.12(b) so that the depths can be checked using multiple cues.

5.7 Summary

In this chapter, we generalize the halo visualization method from 3D graphics to 4D graphics. We develop the notion of illustrative volumetric halos with the goal of more clearly revealing the 4D occlusions of geometric objects that are artifacts of their projections from 4D to a 3D volume image. We design and implement halo-based occlusion-enhancement methods, and employ GPU-based techniques enabling real-time performance to enhance the viewer's interactive exploration experience. Options for supplementary illustrative visualization elements such as color-coding, texture, and heuristic lighting are included in the system to produce as many redundant cues as possible. Stereoscopic viewing support is supplied to assist in understanding the full structure of the 3D volume imaging geometry.

Chapter 6

Exploring 4D Models by Multitouch Interface

Multitouch technology has recently been exploited in a variety of environments to provide innovative modes of user interaction. Multitouch methods can expand the user degrees of freedom significantly beyond the traditional two-degree-of-freedom desktop mouse while avoiding the complexity of special purpose devices embedded in 3D space. These extra degrees of freedom can be effectively utilized in applications requiring control of data having unconventional geometry, with 4D space being a compelling example. Intuitive manipulation of displayed objects that are projected from a 4D model space to a 3D display space requires six degrees of orientation control to start with, plus other desirable controls such as scaling, focal length adjustment, and translation. In this chapter, we study, implement, and discuss the problem of creating an interactive system to teach intuitive properties of 4D models by taking full advantage of the multi-touch technology typically available in tangible hand-held environments. A specific implementation is presented for the iPhone family of platforms.

6.1 Introduction

Four-dimensional objects is a challenge that can be effectively met using interactive computer graphics methods. Even though our physical world is only three-dimensional, in the virtual world of computer modeling we can construct any kind of higher-dimensional geometry that we are capable of imagining and writing down. The problem to be solved is how to present these models to the eye and how to enhance the user's intuitive experience of the abstract geometric world that we are trying to understand. Two basic elements form the foundation of the approaches available to us for exploring the fourth dimension: the first is the family of display methods adapted to represent 4D geometry in a conventional graphics environment highly tuned for displaying three-dimensional objects, and the second is the interactive interface used to manipulate the resulting display to enhance 4D understanding. In this chapter, we focus our attention on this second question, particularly the 4D exploitation of multitouch and accelerometer interfaces available for applications on many handheld devices, and the mechanisms by which such interfaces can significantly contribute to the user's experience of 4D manipulation.

Multitouch is now becoming part of everyday computing technology, evidenced by its emergence in a wide variety of computing devices, including full-size flat-screen displays for desktop and laptop computers, built-in and detached touchpads, and of course small and medium-sized displays on popular handheld mobile devices. The general advantages of multitouch technology include the following:

- First, multitouch is a highly intuitive and physically natural technique, allowing direct manipulation of graphical objects on the screen as demonstrated by a wide range of multitouch applications. In particular, Kin et al. [92] showed that input via direct-touch can be about twice as fast as mouse-based input.

- Multitouch has a high interface-adoption success rate. Users apparently can easily learn and master multitouch-based user interface controls due to their intuitive nature.
- Compared to specialized interaction devices such as the space mouse and 3D wands, multitouch interfaces are much more familiar and widely adopted by the general public, and thus are especially suitable for educational environments. We have conducted user studies comparing existing mouse-based 4D-control interfaces with the multitouch-based system described here, and found significant improvements both in the rate of learning for 4D navigation tasks and in experience effects on accuracy of prediction tasks.
- Since multitouch allows simultaneous use of multiple fingers for input, we can potentially control more degrees-of-freedom in the user interaction, which is an advantage for complex control environments such as the 4D worlds of specific interest to us.
- Finally, multi-finger gestures (possibly combined with accelerometer-based input) can provide extra one-handed degrees of freedom, allowing us to circumvent two-handed methods such as the use of modifier keys to switch control contexts.

These unique characteristics motivate us to explore the use of multitouch methods as a natural way to interact with the extra degrees of freedom characterizing 4D geometry. In this chapter, we first discuss the mathematical foundations for exploring and manipulating 4D geometry. Then, we relate these methods to the necessary user controls and present our novel interface design supporting 4D exploration using multitouch gestures. We also show how 3D accelerometer input can effectively control all six 4D rotational degrees of freedom. A number of unique implementation issues and useful specialized techniques are discussed as well.

Our main contribution is the exploitation of an idealized set of user controls to facilitate a quasi-physical experience of (simulated) 4D objects such as the hypercube or, equivalently, a rolling four-dimensional die. This is accomplished via a computer graphics touch screen combined with what we argue is a minimal-complexity mapping of these controls onto the

available iPhone multitouch gestures. There are a large number of mathematical structures embedded in 4D that are of pedagogical interest and that are well-suited for exploration with our interface: these include knotted spheres, Calabi_yau space cross-sections, quaterions, the three-sphere, $SU(2)$ group actions, and projective plane embeddings (see, e.g., Hilbert and CohnVossen [67]). However, most would agree that the single object that is most familiar by name to the general reader is the hypercube or tesseract.

We therefore choose to illustrate the functionality of our interface with a detailed pedagogical example showing how one can interact with and learn the detail and structure of a hypercube viewed as a back-face-culled eight-sided 4D die; as this 4D die rolls in the fourth dimension, it exposes numerous clear analogs to familiar physical properties of 3D dice. We include a user study illustrating the effectiveness of our chosen design for exploring and learning the 4D properties of the hypercube.

6.2 Multitouch Based Exploration Methods

A wide range of research work has investigated multi-touch interaction methods and gestures for exploring 3D scenes. This include work based on large multi-touch displays and tables. Frank et al. [47] analyzed the advantages and disadvantages of multi-touch devices for object manipulation. Chang et al. [23] designed a two-handed touch-based interface for creating virtual origami. Reisman et al. [112] proposed the screen-space formulation method to directly manipulate 3D objects on a multi-touch screen (see Figure 6.1), while Hancock et al. [54] proposed a force-based interaction system, called sticky tools, allowing users to manipulate objects in a 3D scene via a multi-touch table. Wobbrock et al. [135] suggested allowing users to define their own multi-touch gestures on the computing surface and presented a corresponding user study.

Figure 6.1: Manipulate 3D objects on a multi-touch screen by the screen-space formulation method on a terrain navigation system, from top to bottom: single touch point, two touch points, four touch points [112].

Figure 6.2: Left: a 3D transformation widgets for multi-touch interface [29]. Right: a multi-touch interface for 3D animation design [93].

Figure 6.3: Left: rotate a 3D virtual object by a mobile device [89]. Right: interact with fine-grained details of documents by a mobile device [97].

Rivière et al. [32] developed a cube-shaped device with a multi-touch surface on each face, allowing 3D subspace manipulation corresponding to the cube face being touched. Recently, Fu et al. [48] proposed a set of multi-touch gestures for exploring large-scale scenes, while Yu et al. [138] proposed putting a context frame around a central visualization, so that touching different parts of the frame would support different forms of interaction. More recently, Sun et al. [121] developed a multi-touch sketching interface for efficient design of 2D vector graphics. Kin et al. [93] introduced the multi-touch technique to the design of 3D animation scenes, see Figure 6.2 right. Cohé et al. [29] designed a 2-fingers controlled widget for 3D transformation control, see Figure 6.2 left. Song et al. [119] developed a multi-touch system for the visualization and exploration of the volume data. They also provided a fast contour drawing technique for visual annotations.

Multitouch interactions on mobile devices. Other recent research work focused on exploring innovative interaction on multi-touch mobile devices. Roth et al. [114] developed the “Bezel Swipe” interaction technique, allowing users to efficiently perform cut, copy, and paste actions. Kim et al. [91] proposed a technique using an iPhone/iPod Touch to navigate in virtual environments. Liao et al. [97] suggested using both camera image input and touch input on smartphones to support a variety of gestures for interacting with fine-grained details of doc-

uments, see Figure 6.3 right. Wilson and Sarin [133] applied a vision-based front-projection multi-touch system and Bluetooth connection to support intuitive sharing of multimedia data between a smartphone and a multi-touch table. Hardy et al. [64] provided a technique allowing a mobile phone user to select & pick a picture from multi-touch displays, and vice versa. Katzakis and Hori [89] proposed a method for directly mapping the orientation information of mobile device to a 3D object shown on a computer screen, thus enabling users to use the mobile device as a tangible rotation controller, see Figure 6.3 left. Echtler et al. [38] tracked a mobile phone placed on a multi-touch table using its shadow to support interactions between mobile phones and board games shown on a multi-touch table.

6.3 Exploration of 4D Geometry.

Geometry of 4D Worlds

Those of us who dabble in the fourth dimension are often asked questions like “Is the fourth dimension real?” or “4D can’t be like our ordinary world, so it has to be time, right?” The proper answer is that once a piece of mathematics is inside the computer, it is as real as any other piece of mathematics inside the computer: thus, if for one kind of geometric data we represent the vertex points as 3D vectors, and for another kind we use 4D vectors, and for yet another we use 26D vectors, each is just as “real” as any other so far as the computing framework is concerned. The connection with reality is defined by the interface between the mathematics inside the computer and the outside world that can be sensed by the human observer. *That* connection is defined by our chosen transformation from the mathematical 4D models to the 2D computer screen (3D stereographic displays are also possible, but we will assume for the sake of our general arguments that we have a 2D screen on a handheld device).

4D Rotations

2D and 3D Rotations. Rigid rotations in any Euclidean dimension are classifiable in terms of the number of independent planes in which a rotation can occur. Thus in 2D, there is only one available rotation by an angle θ about the origin of the xy -plane given by

$$\begin{vmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{vmatrix}.$$

In 3D, there are three independent rotational degrees of freedom, and a typical list of these would include rotations in the yz -plane, the zx -plane, and the xy -plane. By a lucky coincidence that occurs *only* in 3D, there is a theorem of Euler showing that any arbitrary 3D rotation matrix has exactly one unique fixed direction, the real eigenvector of the matrix; thus it is almost universal practice to think of the three rotation planes instead as rotations about the fixed *axes*, the x -axis, the y -axis, and the z -axis, respectively. The rotation matrix spinning about the z -axis, for example, is just the 2D rotation matrix above extended to a 3×3 matrix padded with zeroes except for a one in the $(3,3)$ position; this obviously leaves the z -axis unchanged for any θ :

$$\begin{vmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{vmatrix}.$$

The remaining two 3D rotations correspond to obvious permutations of the rows and columns of the z -axis case.

Six 4D Rotation Parameters. Pursuing this idea of rotations in planes defined by pairs of orthogonal Euclidean coordinate axes, we can quickly convince ourselves that in 4D, all the rotational degrees of freedom are given by *six* independent parameters, corresponding to the rotations in the planes labeled by wx , wy , wz , yz , zx , and xy . There are many ways to obtain 4D rotation matrices, but the six-plane labeling is sufficient for our purposes. The three new matrices (that we adjoin to the three 3D xyz matrices) effectively tilt the w -axis into the positive

direction of the x -, y -, and z -axes when we move a 3D control parameter in those directions.

The w x 4D rotation matrix is

$$R(wx) = \begin{pmatrix} \cos \theta & 0 & 0 & \sin \theta \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -\sin \theta & 0 & 0 & \cos \theta \end{pmatrix},$$

and the other two are permutations consistent with the stated sign convention. We can see that multiplying $R(wx)$ into a pure w column-vector $(0,0,0,w)^t$ produces a small positive x component ($w \sin \theta$) for small θ , and multiplying a pure x column-vector $(x,0,0,0)^t$ gives a small negative w component ($-x \sin \theta$), as we require.

Figure 6.4: Green arrows are forward-facing normals. Left: 3D Back-face culling. Right: 4D Back-face culling.

4D Back-Face Culling

Comprehensive simulation of four-dimensional lighting and shading (see, e.g., [120, 59, 28]) automatically removes the 3D parts of the scene that cannot reflect light to the 4D viewer. In 3D graphics systems such as OpenGL, the invisible polygons with normals facing away from the viewer can be removed by enabling back-face culling. This mechanism alone can ensure correct visibility if the object we are rendering is a solid convex polyhedron. For more complicated scenes, depth buffering is also required. In four dimensions almost all graphics depictions of which we are aware neglect the 4D equivalent of back-face culling when constructing the

projection of an object to 3D; we now outline the general *4D back-face culling* process for 4D renderings.

The Culling Mechanism. We will assume that the 4D objects we can deal with are constructed from solid convex polyhedra all of whose surface triangles are shared with exactly one other convex polyhedron, analogous to a closed 3D surface made of triangles. Just as in 3D, we must have some kind of orientation to assign a 4D normal vector perpendicular to all vectors lying in the polyhedron. Assume we are given three ordered, non-degenerate vectors within the polyhedron, typically three edges computed from known vertex positions: $\vec{a} = \vec{x}_1 - \vec{x}_0$, $\vec{b} = \vec{x}_2 - \vec{x}_0$, and $\vec{c} = \vec{x}_3 - \vec{x}_0$. Then the *signed* 4-vector direction normal to the given polyhedral face is the 4D cross-product

$$\vec{n} = \det \begin{vmatrix} a_x & b_x & c_x & \hat{x} \\ a_y & b_y & c_y & \hat{y} \\ a_z & b_z & c_z & \hat{z} \\ a_w & b_w & c_w & \hat{w} \end{vmatrix} \quad (\text{Eq. 6.1})$$

normalized as usual to $\hat{n} = \vec{n}/|\vec{n}|$. There is some confusion in the early literature about the sign of this cross-product, and the given convention is consistent with the requirement that the cross-product of the x, y, z unit vectors produces a vector in the positive w direction. The final step is to pick a 4D eye-point $\vec{P}$, and to remove from the rendering all polyhedral faces for which

$$(\vec{P} - \vec{x}_0) \cdot \hat{n} \leq 0 ,$$

where $\vec{x}_0$ is a representative point lying in the polyhedral face (see Figure 6.4). We remark that the orientation of a 4D, possibly nonconvex, polyhedron projected into the 3D screen can be checked in the *3D screen*, without needing the full calculation given above, using an exact analogy to the standard 2D-projection check: $\sum_{i=0, n-1} (u_i v_{i+1} - u_{i+1} v_i) > 0$ where u and v are 2D coordinates of the projected polygon. We simply add the signed 3D projected-tetrahedron volumes, equivalent to computing only the w -component of the cross-product, yielding this

test:

$$\sum_{i=0, n-1} \det \begin{vmatrix} \vec{u}_i & \vec{u}_{\text{mod}(i+1, n)} & \vec{u}_{\text{mod}(i+2, n)} \end{vmatrix} > 0, \quad (\text{Eq. 6.2})$$

where $\vec{u}_i$ are the projected coordinates of the polyhedron in the 3D image. The result for a typical example such as the hypercube, described in detail below, is that half *or more* of the polyhedra that would appear in a wire-frame rendering *disappear* in any given culled rendering, reappearing in an appropriate sequence when 4D rotations are applied to the object. That is, just as only 1, 2, or 3 square faces of a 3D die can be seen simultaneously, but never more, only 1, 2, 3, or 4 cubic faces of a 4D die can be seen simultaneously, but never more.

The result for a typical example such as the hypercube, described in detail below, is that half of the polyhedra that would appear in a wire-frame rendering disappear in any given rendering, reappearing in an appropriate sequence as 4D rotations are applied to the object.

Figure 6.5: (a) The geometry of the 3D Rolling Ball controller on a 2D screen. (b) The geometry of the 4D Rolling Ball controller in a 3D screen.

The 4D Rolling Ball

A particularly interesting control mode for 4D rotations is inspired by a 3D method that is known by many names, but that we call the “Rolling Ball” [55]. In 3D, the Rolling Ball implements the physical process of putting a ball on a table, placing one’s flattened horizontal palm on the top of the ball, and rotating the ball incrementally by moving the hand to the right for a

positive y -axis (zx -plane) rotation, and towards the user for a positive x -axis (yz -plane) rotation (see Figure 6.5a). Moving the hand in clockwise horizontal circles produces counterclockwise rotations in the xy -plane (about the z -axis). Clearly a mouse-based implementation of this algorithm permits rotations in all three planes, thus achieving *any* 3D orientation with a finite sequence of 2D motions.

4D Ball Control. Four dimensions is peculiarly available to interactive exploration because the entire space of possible 4D orientations can be explored with an incremental 3D vector. That is, the precise analog of the 2D vector that controls the 3D Rolling Ball orientation is a 3D vector that controls the 4D Rolling Ball. Now imagine a 4D object projected to 3D space along the w -axis, which is “pointing out at you” and therefore invisible just like the 3D z -axis was. The remaining three axes (x, y, z) are *visible* in the projection from the 4D world to a 3D image if the projection is along the w -axis. Thus the perfect orientation exploration procedure is to specify a 3D drag vector $\vec{dx} = (dx, dy, dz)$ that *tilts* the w -axis slightly in the plane defined by the direction $\vec{dx}$ and the w -axis itself (see Figure 6.5b). This action hides an old piece of geometry on the side toward which the $\vec{dx}$ -vector is moving, and makes a new piece of geometry appear on the opposite side of the 3D image. This is exactly analogous to the 3D Rolling Ball, except that we must think of a 3D projection screen as our canvas instead of a 2D projection screen.

The 4D Rolling Ball matrix is a generalization of the 3D Rolling Ball matrix, and it takes the form:

$$\begin{vmatrix} 1 + (c-1)n_x^2 & (c-1)n_x n_y & (c-1)n_x n_z & n_x s \\ (c-1)n_x n_y & 1 + (c-1)n_y^2 & (c-1)n_y n_z & n_y s \\ (c-1)n_x n_z & (c-1)n_y n_z & 1 + (c-1)n_z^2 & n_z s \\ -n_x s & -n_y s & -n_z s & c \end{vmatrix}.$$

Here we take $r = |\vec{dx}|$ as the fundamental control vector, and $c = \cos \theta$, $s = \sin \theta$ with $\theta = \arctan(r/R)$ for some scale R . From these we define the normalized control vector $\hat{n} = (n_x, n_y, n_z) =$

Figure 6.6: Multitouch gestures.

$\vec{dx}/r$. An exercise for the reader is to verify that this formula can be derived from a sequence of 4D-plane rotations of the form

$$R_{4Droll} = R^{-1}(yz) \cdot R^{-1}(xy) \cdot R(\theta, wx) \cdot R(xy) \cdot R(yz) .$$

The main task of the interactive system, then, is to create natural ways with existing controllers to access these degrees of freedom; that is of course our main objective in this chapter.

Finding All the 4D Degrees of Freedom. 4D orientations have *six* degrees of freedom. How can our three degree-of-freedom 4D Rolling Ball, which apparently only rotates in the three planes (wx, wy, wz) , generate the other three degrees of freedom? In fact, any *pair* of directions such as (x, y) will produce a rotation in the *plane of the pair* if we move the controller in *circles* in that plane. Thus, since our three degrees of freedom (x, y, z) correspond to three *pairs* (yz, zx, xy) , we actually can produce, incrementally, rotations in all six planes (wx, wy, wz, yz, zx, xy) , and therefore any possible 4D orientation can be reached.

6.4 4D Interface Design and Implementation

Our basic objective is to design a fluid multitouch interface that will support an interactive 4D tutorial activity. To accomplish this, we need to decide what functions we absolutely must support at the top level of multiple-finger touching and dragging; we need to choose which actions will be available without switching modes, and which can be relegated to alternate

modes, menus, and sliders. Our basic vocabulary of multitouch gestures will be chosen from the list in Figure 6.6.

3D Screen Control

We begin by noting that we must have 3D control of the projected image, which in a default parallel projection simply chops off the 4th coordinate and displays the object geometry in terms of the first three coordinates. Thus *rotating in a 3D subspace of 4D is the same as rotating the 3D projected image*. That is useful, because we can simply choose to rotate in 3D with a one-finger drag in the same general fashion as dozens of existing touch-based (and mouse-based) 3D geometry systems (see Figure 6.5a). Our choice is the context-free 3D Rolling Ball [55], though there are other adequate context-dependent interfaces such as the Virtual Sphere [24] that also produce all three 3D rotation directions from a 2D single-finger or mouse drag.

4D Control Design

That leaves us with the other three 4D rotational degrees of freedom, which we can characterize as rotations in the wx -plane, the wy -plane, and the wz -plane. Since any wx rotation changes the x -component in the projected image and the hidden w -component according to

$$\begin{aligned}x_{\text{new}} &= x \cos \theta + w \sin \theta \\w_{\text{new}} &= -x \sin \theta + w \cos \theta ,\end{aligned}$$

we see that x and w continually exchange places with one another as θ increases monotonically from its initial value of zero. The same is true for wy rotations, and thus we can cleanly control any combination of wx and wy rotations with a *two-finger drag* on the screen's natural xy coordinate system.

Remaining is the question of what to do about the wz coordinate rotations needed to fill out the six-parameter rotation space. This is solvable by noting the natural correspondence between “spinning” the screen around its center and familiar actions such as turning a drill brace and bit or a screwdriver; such motions naturally have one fixed center and a moving curved path around the fixed center. A *two-finger action with one finger fixed* is a perfect intuitive realization of this type of “ z -direction screw motion,” and that is our preferred design choice. It is a two-finger action, just like the wx and wy motions, but multitouch software can easily separate it from a both-fingers-moving motion in the gesture analysis.

Just as for any type of 3D rotation, we must pick a particular center for the 4D rotations. Considering the most essential features of 4D intuition that we wish to deal with, we are going to assume that in general we only need to work with single objects having a natural center of mass for rotations at the screen center, and thus we relegate 4D translations, if we should need them, to auxiliary modes that do not need to be major part of the main interactive interface.

Body-fixed and Screen-fixed Rotations

After some experimentation with the 4D rotation interface, one discovers the fact, obvious *a posteriori*, that the 3D projected image gets in its own way for the potentially very informative wz rotations. There is an easy way around that, a technique that forms an essential part of the 4D learning interface in its own right. What we do is to provide a button that toggles between these two modes:

- **Screen-fixed Mode.** The coordinate system in which the finger-drags control wx , wy , and wz rotations is oriented to the user coordinate system (screen/eye space), with x and y being the screen plane and z the direction out of the screen towards the user.
- **Body-fixed Mode.** The coordinate system in which the finger-drags control wx , wy , and wz rotations take effect is *fluid*, and is computed relative to the original 3D frame

(object space) of the 4D object. Thus one can get a *from-the-outside* viewpoint of the effect of a 4D rotation on the object, as though looking at the projection volume from an oblique viewpoint instead of straight down into the projection volume. This is especially effective for wz rotations, which can be turned completely sideways to see exactly what is happening from an arbitrary viewpoint.

4D Perspective, Screen Space Adjustment, and Reset

The final basic controls that we chose to have immediately available to the user via multitouch provide a handful of frequently used adjustments to the view that facilitate the exploration. These are

- **4D Perspective.** *Three-finger up-down dragging* changes the 4D perspective transformation from orthogonal (pushing away, to the top of the screen) to high-perspective distortion (dragging closer to the bottom of the screen). Note that the most obvious linear interpolation of the 4D focal length results in a difficult-to-control variation of the focal length, it changes too fast when it is small and too slow as it goes to orthogonal. This will make user have a jumpy feeling while they are controlling the 4D focal length. So it is desirable to apply non-linear resizing and interpolation smoothing, such as a hyperbolic tangent approach to the endpoint, to get an aesthetic result.
- **Screen Space Scaling.** If one wants to examine some object portion more carefully, it helps to have a rescaling operation centered around the object center. Two-fingers-moving-apart or two-fingers-moving together (*pinch*), allowing for a noise threshold to avoid confusion with two-finger drag, is used to zoom in or out of the current object view.
- **Screen Space Translation.** While we do not typically need 4D translations, it is sometimes useful to move the object around in the screen plane. We employ our last available

multitouch freedom, the *four-finger drag*, to accomplish this occasionally used functionality.

- **Reset.** The reset function for 4D views in fact must be done in two parts: sometimes we just want to set the “3D projected image” orientation back to the canonical position relative to the user coordinate system, and we may independently want to reset the 4D orientation matrix itself back to the identity. We use a *double-tap* to reset the 3D orientation and a *triple-tap* to reset the 4D orientation.

This set of actions exhausts the multitouch controls given in Figure 6.6.

6.5 Strategies of the Interface Design

To complete the set of capabilities needed to support 4D exploration and spatial intuition development, we found experimentally that several additional controls, modes, and adjustments were necessary. While these controls are to some extent a matter of taste, we found this to be a fairly complete set of controls to facilitate 4D learning.

Pedagogical Controls

We have included in our interface a small family of additional controls specifically for pedagogical purposes, of which we note particularly the following:

- **3D Die/4D Die Switch.** Our main teaching tool is a 4D hypercube that is back-face culled to simulate the main features of an actual 4D die viewed with 4D optics. However, many of the simplest features do not immediately reveal themselves to the viewer until the exactly analogous 3D features are presented first. Thus we provide a mode switch that changes between a 3D die with 3 opposite pairs of square faces, opposite numbers summing to 7, and a 4D die with 4 opposite pairs of cubic-volume faces, opposite numbers summing to 9 (see Figure 6.4).

- **Wire-frame Mode.** We can switch (in 3D or 4D) between a labeled solid-face mode and a wire-frame mode with no numerical face labels.
- **Back-face Cull/No-Cull Mode.** While one of our major features is support for 4D back-face culling, we find that users also want to be able to check on the standard wire-frame, non-back-face-culled representation that shows all the hidden cubic faces as well. A mode switch toggles between the two depictions, and, in wireframe mode, the technically invisible edges are shown in a distinct color (see Figure 6.4 (right)). This seems to be a very useful pedagogical feature.
- **Tracking Speed Slider.** The optimal tracking speed for exploration turns out (unexpectedly) to be a very personal interaction tuning parameter. To account for a wide variety of personal variation in the optimal desired response speed, we provide a slider with a customizable setting.

6.6 4D Learning Scenario

We conclude with a description of a detailed learning scenario that uses a back-face culled hypercube as its principal pedagogical tool. This scenario implements our goal of developing both an intellectual and a kinesthetic feel for what a 4D world is like. Each of our family of multitouch controls and mode settings then plays a specific role in learning to understand and predict the behavior of the image on the screen.

About The Hypercube

The hypercube, also called by other names such as the tesseract, is an ideal subject for 4D tutoring because it consists of very simple parts that naturally delineate each of the four orthogonal dimensions of 4D space. In addition, it is a familiar object and many people have at least a vague idea of what it looks like.

Figure 6.7: The construction of a hypercube or tesseract by building up from lower dimensions.

Making a Hypercube. The hypercube is a member of a family of figures created by drawing lines between the corresponding points of two copies of the same figure one dimension lower. In Figure 6.7, we illustrate the classic construction of a hypercube by sweeping a line into a 2D square, the square into a 3D cube, and finally using 8 lines to connect the corners of two copies of the 3D cube to form a 4D hypercube. The first version of this final view is an orthogonal projection, and the second is a perspective projection, in which the more distant edges of the hypercube's structure appear smaller, exactly analogous to ties on a railroad track shrinking in the distance. The last image on the right is the back-face culled hypercube image that is an essential focus of our presentation.

We observe that the 1D cube has two points that bound its interior, one at either end of the single x -axis; the 2D square's interior is bounded by two pairs of lines, one at each end of the x -axis, and one at each end of the y -axis. The cube has three pairs of squares bounding a 3D volume centered around the origin. We thus can see that the hypercube's four pairs of cubes, with one cube at each end of the x -, y -, z -, and w -axes, must be the boundary of a 4D solid volume if we could perceive such a thing directly. That is, at the center of this 4D volume, a 4D ant could walk in any of four directions, and would have to move a considerable distance before it finally bumped into one of the eight boundary cubes that we usually associate with a hypercube.

Hypercubes in Literature and Art. The idea of the hypercube as a somewhat mystical object has appeared in many places. One well-known science fiction story, —*And He Built a Crooked House*— published by Robert Heinlein in 1941, uses the hypercube as an essential plot element. A radical architect builds a house based on a 3D unfolding of the faces of a hypercube, giving a strange configuration of eight cubical rooms; the 3D analogy is the unfolding of a paper cube’s six square to form a flat cross-shaped object. Nothing is amiss in the story until a strange earthquake occurs, collapsing the house into the fourth dimension, so all that can be seen from the real world is a single cube, and yet that cube connects correctly to all the other (now outwardly-invisible) cubes as it would in four dimensions. Finally, after another earthquake, the house disappears from the 3D world altogether. The artist Salvador Dalí used a figure essentially identical in shape to the description of Heinlein’s 3D architecture in his surreal painting of the crucifixion (*Corpus Hypercubus*), showing Christ crucified on an arrangement of the eight cubic surfaces of an unfolded hypercube, essentially a doubled solid cross.

Hypercubes in Electronic Media. Creating pictures as well as animations of hypercubes has occupied artists, students, hobbyists, scientists, amateurs, and professionals since the earliest times of electronic media. Thomas Banchoff and his collaborators at Brown University made some of the first documented computer animations of 4D objects, and those that have followed are almost too numerous to count. A search for “hypercube” on YouTube results in 1,250 hits! Our own YouTube contribution to this genre is *4Dice*, <http://www.youtube.com/watch?v=fx7eh17YvMY>; it is the first and possibly only animated hypercube presentation of which we are aware that includes back-face culling, simulating the view that would be seen by an actual 4D viewer under correct 4D illumination models.

Geometric Exploration of a Hypercube

We now outline some scenarios that exploit our control methods to teach basic features of 4D geometry.

Understanding the Ordinary Cube as a 3D Die.

We begin with a study of the 3D rolling die, as shown in Figure 6.8:

Figure 6.8: The basic geometry, visual presentation, and face-numbering rules of a right-handed 3D die.

- **Six Faces with Numbers.** A standard right-handed 3D die is a cube with numbers in pairs on square faces at opposite ends of the $\pm x$, $\pm y$, and $\pm z$ axes. We label the faces on the positive x -, y -, and z -axes with 1, 2, and 3, respectively, and then require the sum of numbers on opposite faces to be *seven*; hence the face labels on the negative x -, y -, and z -axes become 6, 5, and 4, respectively.
- **See How They Roll around Y.** The initial default display shows only the number “3” labeling the positive z -axis. When we drag one finger to the left, the die spins about the negative y -axis, as though we were rolling a ball lying on a table; the “3” label shrinks and the “1” label of the positive x -axis comes into view (see Figure 6.10). After 90 degrees, we see only the “1”; as we continue past 90 degrees, the “4” label of the negative z -axis comes into view, then the “6” of the negative x -axis, and finally we come full circle back

to the “3” label of the positive z -axis. We have made a full circle through the labels of the square 3D die faces in the zx -plane.

- **See How They Roll around X.** Likewise, when we drag the 3D die with one finger moving downward on the screen, the die spins about the positive x -axis and the “2” label of the positive y -axis comes into view. Continuing to rotate by pulling in the downward direction, we pass from “3” to “2” and then we see the “4” label of the negative z -axis *again* — this time we have arrived at the back face of the die coming from a different direction than before. If we continue pulling down, we finally arrive at the “5” labeling our last unseen face, the negative y -axis, and then return to “3”.
- **Opposite numbers sum to 7.** Each time we go from the positive end of a given axis to its negative end, the sum of the opposite-end labels is *seven*, a fact well-known to gamblers.
- **$x : y$ Circles Spin in z .** By moving the single dragging finger in a small clockwise (counterclockwise) circle, we can spin the die counterclockwise (clockwise) about the z -axis, the direction perpendicular to the screen for the user. If the “3” label is shown, we can see the “1”, “2”, “6”, and “5” labels peeking around the edges in sequence as we trace a counterclockwise circle. The “4” label remains hidden on the far side, opposite our main “3” label. Starting with any other face showing, we get the appropriate sequence of four neighboring labels peeking around the edges.

Understanding the Hypercube as a 4D Die

There is nothing new or difficult about the scenario we just described for exploring the geometry of 3D die on a computer screen using a labeled 3D cube, and applying rotations that roll the die with a single-finger drag. Our main 4D pedagogical device now becomes to *repeat the 3D process* by exact analogy, and to try to make the strangeness of 4D geometry as familiar as 3D geometry.

Figure 6.9: The basic geometry and visual presentation of a 4D die.

First, we need to establish the following mental picture, schematized in Figure 6.9, relating a 4D die to a 3D die:

- **There are 4 axes instead of 3.** Mentally label the axes as x , y , z , and add a fourth axis, w . That means there are 4 pairs of labels, one pair for the two ends of each axis.
- **The sum of the opposite labels is 9 instead of 7.** A 4D gambler would be very familiar with the rule that the opposite ends of the x -axis are labeled by “1” and “8”, the y -axis by “2” and “7” the z -axis by “3” and “6”, and the w -axis by “4” and “5.”
- **The geometric objects at the ends of the axes are *cubes*, instead of *squares*.** At the ends of the three axes of a 3D cube, we have square 2D faces. In 4D, we go up one dimension for everything, so the “face” objects at the ends of the four axes of a hypercube are 3D *cubes*. That is, hypercube “faces” are *cubic volumes* instead of *square surfaces*.
- **Draw the hypercube’s cubic face-volumes as wire-frames.** Since the hypercube “faces” are technically solid cube volumes, we need to “look inside them” to see the numbering because numbers label the inside of the cube rather than its surface (analogous to the case of 3D die, where the circular dot labels are placed in the middle of the square faces). Therefore we draw these solid objects as *wire-frame* cubes and use small solid spheres floating *inside* the wire-frame cubes as the number labels.

Figure 6.10: Illustrative rolling actions of the 3D die and the 4D die.

Exercise: The Back-Face Culled Hypercube as a 4D Die

Pursuing the parallels between the 3D die and the 4D die, we now study the nature of a rolling 4D die, illustrated schematically in Figure 6.9. Our rolling 4D die has the following features that we can recognize as closely analogous to the familiar 3D case:

- See How They Roll.** The initial default display shows only the number “4” labeling the positive w -axis. Rolling the projected image with a single-finger drag produces a rigid 3D motion of this single cube, and, as before, moving one finger in a circle spins the image in the virtual 3D screen around the screen z -axis. When we drag *two* fingers to the left, the die spins in the wx -plane, shrinking the “4” and exposing the “1” tagging the positive x -axis (see Figure 6.10); dragging two fingers towards the bottom of the screen causes a rotation in the wy -plane, exposing instead the “2” label of the positive y -axis. With a two-finger/one-fixed spin in wz , the number “3” labeling the positive z -axis appears right in front of us on top of the “4” label; this is best seen using the body-fixed mode supporting oblique viewing relative to the hand motion. Continuing the wz rotation, the

“4” label shrinks and is replaced by the “3” label. As the “3” label shrinks away, we see the “5” label of the negative w -axis, and then the “6” of the negative z -axis. Finally we come full circle back to the “4” label on the positive w -axis. We have now made a full circle through the labels 4-3-5-6 of the cubic 4D die faces as we rotate in the wz -plane.

- **See How They Roll in WX and WY.** When we drag the 4D die with two fingers moving to the left on the screen, the die spins in the wx -plane, passing from the “4” label to the “1” label. Continuing to rotate by pulling in leftward, we go further to “5,” (negative w -axis) *again* — coming from a different direction than before. If we continue pulling left, we arrive at “8,” and then return back to “4.” Pulling downward produces the analogous 4-2-5-7 sequence induced by wy -plane rotations.

Figure 6.11: Rotation sequence of the 4D die.

- $x : y$ **Circles Spin in z .** There exists a *separate* effect that parallels the z -axis rotation induced by a 3D/one-finger rotating drag. This is the z -axis rotation induced by moving *two* fingers in a small clockwise circle; we can make the 4D die spin counterclockwise

Figure 6.12: How our two-finger multitouch controls produce 4D rotations in the w_x , w_y , and w_z planes.

in its 3D image about the z -axis (the direction perpendicular to the screen for the user); counterclockwise circles produce clockwise z rotation. If the “4” label is showing, we can see the “1”, “2”, “8”, and “7” labels peeking around the edges in sequence as we trace a counterclockwise circle. The “5” label remains hidden on the far side of w , opposite our main “4” label. Starting with any other face showing, we get the appropriate sequence of four neighboring labels peeking around the edges.

With a two-finger/one-fixed clockwise “into-the-screen” spin in w_z , the number “3” labeling the positive z -axis appears right in front of us on top of the “4” label; this is best seen using the body-fixed mode supporting oblique viewing relative to the hand motion. Continuing the w_z rotation, the “4” label shrinks and is replaced by the “3” label. As the “3” label shrinks away, we see the “5” label of the negative w -axis, and then the “6” of the negative z -axis. Finally we come full circle back to the “4” label on the positive

Figure 6.13: The three alternate sets of interlocking pairs of tori that can be seen while rotating the hypercube in each of the three planes wx , wy , and wz . Each torus on the left forms a revolving belt surrounding its partner torus that threads itself through the center.

$wx-yz$	4-1-5-8	2-3-7-6	$wy-zx$	4-2-5-7	3-1-6-8	$wz-xy$	4-3-5-6	1-2-8-7
---------	---------	---------	---------	---------	---------	---------	---------	---------

Table 6.1: The three pairs of tori, each composed of two complementary sequences of four cubic hypercube faces, that appear as we mix each of the three Cartesian axes with the w -axis by a simple rotation.

w -axis. We have now made a full circle through the labels 4:3:5:6 of the cubic 4D die faces as we rotate in the wz -plane. The resulting cyclic face sequences are illustrated in the bottom row of Figure 6.11.

The correspondences between each multitouch 4D action and our chosen screen-centered Cartesian coordinates are shown in Figure 6.12.

Exercise: Topology of 4D Sphere

Now we have learned to control the rolling of the 4D die, and to understand the sequence of hyperfaces as we roll in each of the three independent directions: wz -rolling shows “4-3-5-6,” wx -rolling shows “4-1-5-8,” and wy -rolling shows “4-2-5-7.” As a final challenge to our ability to envision the hypercube, we turn on the perspective projection, which makes more distant faces smaller, and makes the hypercube’s cubic faces transparent, so we can see through to the parts facing away from us.

The geometric facts are these: the eight cubes bounding the hypercube’s 4D-solid interior also constitute one of the simplest representations of the generalization of the sphere to four dimensions. By convention, the usual sphere — the surface of the Earth for example — is

called a “two-sphere” or S^2 because its surface is two-dimensional, even though it is embedded in 3D and surrounds a solid 3D ball; therefore, the 4D extension is called a “three sphere” or S^3 . While this three-sphere is itself a three-dimensional space, it completely encloses a solid 4D ball. Just as a cube is a simplified schematic version of the spherical Earth S^2 , a hypercube is a simplified schematic version of the hypersphere S^3 .

Figure 6.14: A 3D die shown as a 3D box with square “3” as the lid, analogous to a 4D die shown as a 4D box with cube “4” as the lid. The image of the 3D die is projected to 2D, and the lid lies on top of all the other 5 faces, while the image of the 4D die is projected to 3D, and the image of the cubic lid occludes the images of all the other seven cubes in the projection.

Mathematically, S^3 can be written down as a pair of solid tori that wrap around each other in a very special way to make the continuous, boundary-less S^3 . Depending on the projection from 4D to 3D, we can see one or another of these tori particularly well. In fact, there are *three* different ways we can see S^3 as a pair of tori embedded in the hypercube. Consider as a starting scenario one set of 4-die actions, say the *wz*-rolling sequence “4-3-5-6.” We know there are *eight* hyperfaces in total, so we are missing precisely “1-2-8-7” (see also Table 6.1). Each of these forms a circular sequence of cubes, which, if we think about it, is exactly what a very simple torus must be — we break up the two smooth circular paths on the familiar torus into squares, and we see four cubes, as shown in Figure 6.13. If we look at the non-back-face-culled hypercube rendered with perspective, we see this first torus (e.g., “1-2-8-7”) plainly surrounding the central *z*-axis. In the middle, we see the four cubes of the central second torus,

e.g., the “4-3-5-6” sequence. However, the central sequence of four has three that are easy to see, and a fourth that is not so easy to see: it is the “lid” on the box containing the solid 4D ball! In projection, it completely covers the other 7 cubes; this is familiar, since if we project a 3D cube to 2D, the face “3” will occlude the projection of all “1-2-4-5-6” faces, forming the top of the box on the solid cube. In Figure 6.14, we illustrate both the 3D and 4D versions of this box-like geometry, with square “3” forming the lid of the 3D box, and the cube “4” forming the lid of the 4D box.

What is special about the 4D case is that *the lid joins the other three interior cubes to form its own torus*. In fact, each sequence of 4 cubes seen when rolling in w_x , w_y , or w_z forms its own torus, and is complemented by a second independent torus. Table 6.1 enumerates these tori with their matching cube labels, and Figure 6.13 shows how they appear in the interactive 4D die rolling scenario: as we roll the 4D die in any one direction, one of the tori simply spins around its central ring, staying in the same place, while the other torus moves so each of the four cubes in sequence becomes the “lid” like the cube “4” depicted in Figure 6.14.

Figure 6.15: Rotating various 4D objects via remote display control.

6.7 Remote Control.

The wireless internet capabilities of a typical handheld multi-touch device lead us to one final interesting application: we can install the graphics program that displays the 4D object on another computer driving a projector screen or desktop display, and control that computer remotely from the handheld device. Examples of a 4D display program being controlled remotely are shown in Figure 6.15.

6.8 Summary

Multitouch interfaces can provide an excellent environment for exploring and building intuition about geometric concepts. The main idea presented here is that the degrees of freedom of many current multitouch interfaces are uniquely well-suited to controlling 4D worlds with intuitive gestures. This is the first interface that exploits multitouch gestures for seamless exploration of all six 4D rotational degrees of freedom, thanks in part to the mathematical serendipity of the 4D Rolling Ball. With the addition of novel interface features, such as support for 4D back-face culling, pedagogical exploration and visualization of the properties of 4D objects such as the hypercube become as natural as we can make them.

Chapter 7

User Evaluation

This research thesis provides a comprehensive 4D visualization framework to improve the conventional 4D visualization system and enhance the human perception of 4D geometry in different aspects. In previous chapters, we gave the technical detail of our proposed framework and discussed how the comprehensive framework enhances the existing 4D visualization. In this chapter, we evaluate the comprehensive 4D visualization framework from the human perceptual point of view.

7.1 Introduction

To explore the potential of our comprehensive 4D visualization framework, we conducted three user evaluations. These three user evaluations test the enhancement of our framework comparing to the conventional visualization method in three aspects: the first user evaluation aims to explore whether the dual-view visualization can improve user's understanding of the geometry properties; the second user evaluation is to investigate whether the illustrative halo visualization can provide users with correct depth perception; and the last user-study assesses whether our multitouch interface can provide a more intuitive and efficient interaction experience to

Figure 7.1: Eight major steps in comprehensive 4D visualization framework.

users compared to a mouse and keyboard interface. The remaining sections detail the settings and results of the user evaluations.

7.2 Review of the 4D Visualization Framework

Before the description of the user evaluations, we first review our comprehensive 4D visualization framework briefly in this section. There are eight major steps in our visualization pipeline, see Figure 7.1 again: 1) Surface construction to generate a set of grid points; 2) Blue-noise surface sampling to obtain a sample point set; 3) Interactive 4D transformation (rotation and projection); 4) Parallel-coordinates plots with visual signatures; 5) Illustrative visualization; 6) Enhanced visual signatures (e.g., to emphasize high curvature areas); 7) 4D geometry manipulation; 8) interactive brushing. Compared with the conventional 4D visualization system, our comprehensive framework improves 4D visualization by means of the following three new features:

Dual-view Visualization. We introduce the dual-view visualization technique in Chapter 4. In the dual-view visualization technique, a parallel-coordinates plots view cooperates the 3D projection view and these two views complement each other. In the preprocessing step, we take blue-noise sampling to generate high-quality sample 4D points which can be used for the efficient

construction of continuous parallel-coordinates plots. By employing such a dual-view visualization into our framework, the information loss problem caused by the dimension-reduced 4D-3D projection has been reduced. In this dual-view visualization, users can also explore the 4D geometries in both of the views. In addition to the basic interactions, users can also brush in either of the views to emphasize the visualization of the corresponding geometry in certain regions. During the interaction, the geometries of the two views change simultaneously to form a visual signature such as rotational and reflection symmetry. Users can achieve a holistic visual perception of the geometry while observing the geometric structures in both of the views.

Halo Illustration. In Chapter 5, we schematically emphasized the occlusion diagram for the 3D image of the 4D surfaces by providing the halo cutaway technique. Without our halo visualization, the false interaction may impair the users' understanding of the abstract geometry. Moreover, in order to provide our halo visualization with interactive performance and allow it to work as an online component in our system, we designed our halo cutaway technique as a GPU-based algorithm. Assisted by the halo cutaway technique, our 4D visualization system can redress the visual perception of 4D surfaces by presenting them with correct occlusion information and emphasized 4D depth relations. Apart from the halo technique, our system also supports other information cues, such as color-coding, heuristic lighting, texturing, and stereoscopic viewing. The visual cues work together with the main visualization to enhance the spatial perception of the 4D object.

Multitouch Interaction. The traditional control interface of 4D visualization systems is the mouse and keyboard. We enhanced this in Chapter 6 by providing a carefully designed multitouch interface. The multitouch interface takes multiple touch points as input. By appropriate mapping, these inputs can expand the degrees of freedom of the controls compared with the traditional two-degrees-of-freedom desktop mouse. With our multitouch interface, users

can either intuitively interact with 4D objects in a 3D projection view, basically through rotation, scaling, and translation plus other commonly-used desirable controls, or they can perform brushes across the parallel-coordinate view and 3D projection view. In our visualization system, the multitouch interface will be installed in a mobile phone and serves as a remote controller, while the main visualization program can be installed in a computer with a large screen or a projector. This servers and remote controller architecture make our system perform better as an education tool in public lectures.

7.3 Understanding the Geometric Properties by Dual-view Visualization

We conducted a preliminary study to evaluate our dual-view visualization with 12 participants: 9 males and 3 females; age from 24 to 32. These participants were randomly divided into two equal-sized groups. None of them had knowledge of the 4D mathematical equations employed in this study, and none of them knew about parallel coordinates, nor high-dimensional geometries and their properties. This preliminary study has the following two phases:

- *Learning phase (7-15 min.):* In this phase, we first presented the concept of 2-manifolds embedded in 4D space, and showed interactive visualization of some objects by traditional 4-to-3D projection. Then, we gave a tutorial only to the 2nd group (extra 8 min.): the basic knowledge of parallel-coordinates plots with the aid of Section 5.3 and the geometric meaning of visual signatures with the aid of Section 4.4.
- *Task phase (15 min.):* In this phase, we tested the participants' understanding of the 4D models shown in Figure 4.7. We set the 1st group as the control group, where the participants explored the models only with the 3D projection view via an interactive subspace rotation. In contrast, the participants in the 2nd group can see also the parallel-coordinates view. We prepared seven multiple choice questions about symmetry and

visual signatures of the three 4D models. The participants from both groups can interactively explore the models while answering the questions. For example, one of the questions is: *“From 1c (the figure label), the geometry exhibits mirror-reflection about the diagonal of which axis [X1/X2/X3/X4]. There may be more than one correct answer(s).”* The participants scored 1 point for the question *only* if he/she chose all the correct answers; so the full mark of the quiz is 7 points. Moreover, they had to complete the quiz in 15 min.

After checking the participants’ answers, we found that the mean score and the related standard error of the 2nd group (with parallel-coordinates view) are 6.33 and 0.21, respectively, while those of the 1st group (without parallel-coordinates view) are 3.83 and 0.40, respectively. Since the 2nd group performs better (with a small deviation), this suggests that the parallel-coordinates plots and the related visual signatures can likely help the participants understand the geometric properties of the 4D geometries.

7.4 Looking for False Intersections by Halo Illustration

To evaluate the illustrative effects of our halo visualization, we conducted an informal evaluation using the results presented in Section 5. In summary, ten participants were involved; among them, we had six males and four females with ages ranging from 20 to 35. During the evaluation, we first introduced the basic knowledge of 2D halos in 3D graphics to the participants. Then, we presented to them the mechanism of our volumetric halos for illustrating 4D geometry by analogy to halo-enhanced lines in 3D graphics.

After that, we showed four groups of static images to the participants, each group including three images with different visualization effects (traditional transparent rendering, color-coding (without halos) and our halo illustration). So altogether, each participant received twelve static images with four of them containing our volumetric halos. Then we asked the participants

to locate all the false intersections (intersections that only introduced during the 4D to 3D projection, but not in actual 4D space) in the test images. Once the false intersections were identified, the participants could then tell which parts were in the front or at the back.

After the evaluation, we found that in the traditional transparent rendering, even the participants saw a lot of intersections in projected surface, they were not sure whether these intersections existed in the 4D space. It might be possible for the participants to locate the false intersections when interacting with the 4D model, but it was still difficult and needed a long time. It became better when color-coding results were involved since the participants could perceive certain occlusion by mapping the color to depth information. However, with our volumetric halos, the participants could quickly find out all the false intersections because gaps were introduced to promote the occlusion and halos could be easily located. In addition, we also let the participants interact with 4D geometry using our visualization pipeline. They were all impressed with the effect that the volumetric halos changed (became smaller or disappeared) when the 4D viewport was transformed.

7.5 Rolling a 4D Die by Multitouch Interface

In order to test our qualitative impression that the multitouch interface presented here provides an improved 4D exploration experience, we performed a user study comparing the 4D navigation performance of the multitouch interface on an iPad with a mature mouse+keyboard interface. The server visualization system is our 4D visualization system for both of the interface. We adjusted both the performance parameters and the screen size to match as closely as possible, with similar required mouse-drag distances vs finger-drag distances, and identical physical window sizes.

Participants. This is a between-subject experiment, where the 20 recruited participants are randomly and equally divided into two groups: G1 (multitouch): 7 males, 3 females, ages between 24 and 30 (mean 26.7), and G2 (mouse+keyboard): 5 males, 5 females, ages between 22 and 29 (mean 26.2). All participants were at least somewhat familiar with both mouse+keyboard and multitouch interaction methods, and none of them had used our multitouch system before the study. Among them, 2 had some basic knowledge of 4D (tested by asking them to draw a hypercube) and they were assigned separately into G1 and G2.

The Procedure. Each participant went through the following three phases:

- *Learning phase (~20 minutes):* In this phase, we first gave a tutorial to the participant based on the material presented in this paper. First, we presented the geometry of a hypercube as a generalization of a 3D cube, and discussed the characteristics of a 3D die and a 4D die. Then, we gave an interactive demonstration of the interaction controls (either multitouch or mouse+keyboard) and how they could be used to manipulate a cube represented as a 3D die and a hypercube represented as a 4D die.
- *Practice phase (5 minutes):* After that, the participants were given 5 minutes to practice rotating the 4D die using mouse+keyboard or multitouch, as appropriate. In the first two phases, the initial view presented the front face of the 4D die labeled “5.”
- *Task phase (~60 minutes):* Next, we entered the task phase with the following task sheet:

This is a test of your ability to rotate a 4D die to find a particular view with the interface control. It has 140 trials altogether, and in each trial, you will see a 4D die with a random number facing you, and a goal number displayed at the top of the screen. Your task is to apply the interaction controls you just learned to quickly rotate the 4D die so that you can clearly see the dots in the face with the goal number. When you find the goal number, immediately press

the done button and then press the start button when you are ready to go on to the next trial. If the number you found is incorrect, an error message will be displayed and you have to continue the rotation to look for the goal number.

When you are ready to start these trials, press the start button displayed below.

To make up the 140 trials, we first created 28 trials with the initial state face numbers from “1” to “4,” and the target goal numbers from the seven remaining numbers correspondingly; this process is repeated five times and the resulting 140 cases were presented in a random order to the participants. Before each trial, the participants had to press a start button to begin, and they were allowed to press the done button at will. The testing environment detects whether the goal number has been reached by checking the angular difference (in degrees) between the goal hyperface’s normal direction and the view direction of the 4D projection, accepting matches below a threshold angle, typically 45° .

Results and analysis. Our testing interfaces recorded the starting and ending time (i.e., time taken) of each trial, the accuracy (the angular difference between the final view and the goal description), the initial and goal numbers of each trial, and the number of times that the error message was shown.

To compare the data from the two interfaces, we first applied a moving window average of size 15 to the raw data to compensate for the noise in the time data. We then computed the average time taken and the standard error in the times for the ten participants in each trial, repeating the process for each group. This yielded the smoothed sampled values T_k^1 and T_k^2 for G1 and G2, respectively, for $k \in [1, 140]$. To further analyze the asymptotic performance and the learning rate, we performed a least-squares fit on the mean curves (T_k^1 and T_k^2) using the following exponential model:

$$T = a e^{-b k} + c .$$

Figure 7.2: Summary of the user study results. Mouse+keyboard: asymptotic performance (c) 5.96 ± 0.11 seconds per trial, exponential fit coefficient (b) 0.0696 ± 0.004 per trial. Multi-touch: asymptotic performance (c) 3.82 ± 0.09 seconds per trial, exponential fit coefficient (b) 0.1088 ± 0.008 per trial.

Figure 7.2 shows the results with standard errors displayed as error bars. Note that we plot only the first 100 of the 140 trials, as the curves beyond that are flat, with negligible changes in asymptotic performance.

We also compared the average accuracy (quantified by the angular differences between the ideal goal and the user-declared final view) for the two interfaces, with the result that the multitouch interface offers accuracy comparable to that of the mouse+keyboard interface. We averaged the accuracy values over the 140 trials for each participant, and then performed a matched-paired t -test on the two sets of ten averaged accuracy values (per participant) from G1 and G2. The mean values (and standard deviations) for the accuracy of the multitouch and mouse+keyboard interfaces are found to be 14.58 (12.14) and 13.76 (11.13) degrees, respectively. The t -value is found to be 0.2861 with 9 degrees of freedom (DOF), indicating no significant difference between the two sets of accuracy values.

7.6 Summary

In this chapter, we performed three user evaluations to explore the our visualization framework in different aspects. The first evaluation shows our dual-view visualization can help users effectively differentiate the geometric properties. The second user evaluation demonstrates that our halo visualization can enhance user's depth perception of 4D geometry, while the last user evaluation suggests our multitouch interfaces can offer a more efficient and intuitive 4D exploration experience.

Chapter 8

Conclusion and Future Work

8.1 Conclusion

So far, this thesis has presented a comprehensive 4D visualization framework which enhances 4D mathematical visualization in various new directions. First, we discussed existing problems in various aspects of 4D visualization. Then, we proposed a comprehensive 4D visualization framework with a set of novel techniques (new research directions), which include 1) new visualization methods, 2) new interaction methods, 3) enhancing the current visualization system by using novel visual cues, 4) the evaluations from the aspect of human perception.

Interactive Dual-view Visualization. In detail, we employed new visualization methods such as continuous parallel-coordinates plots to supplement conventional 4D visualization. First, we proposed the method of using blue-noise sampling to sample the high-dimensional geometries for efficient construction of continuous parallel-coordinates plots. In addition, we also discussed the geometric features which can be observed in the patterns shown in continuous parallel-coordinates plots. We also proposed new ways for interactively visualizing and exploring four dimensional surfaces by using parallel-coordinates plots: viewing the dynamic pattern changes in the parallel-coordinates plots by rotating the geometry in the 3D projection view, brushing the parallel-coordinates plots, and visualizing the corresponding surface regions in a

3D projection view. In addition, geometric properties, such as curvature, have been employed to enhance the visualization.

Enhancing the 4D visualization by Volumetric Halos. We generalized the halo visualization method from 3D graphics to 4D graphics. We developed the notion of illustrative volumetric halos with the goal of more clearly revealing the 4D occlusions of geometric objects that are artifacts of their projections from 4D to a 3D volume image. We designed and implemented halo-based occlusion-enhancement methods, and employed GPU-based techniques enabling real-time performance to enhance the viewer's interactive exploration experience. Options for supplementary illustrative visualization elements such as color-coding, texture, and heuristic lighting are included in the system to produce as many redundant cues as possible. Stereoscopic viewing support is supplied to assist in understanding the full structure of the 3D volume imaging geometry.

A Multitouch Interface for 4D Visualization. Moreover, we proposed a novel design that first employs and merges multi-touch to develop an interactive interface that matches the exact requirements of 4D exploration with a new set of gestures capable of delivering the best ever pedagogical examples shown of the hypercube as a 4D die, and also as a 3-sphere. In addition, we demonstrated a detailed pedagogical example showing how the features of the proposed interface can help with learning the details and structure of a hypercube viewed as an eightsided 4D die, rolling in the fourth dimension, exposing numerous clear analogs to a familiar 3D die via our interactive interface. We also conducted a user evaluation to evaluate the efficiency of the proposed interface.

8.2 Limitation and Future Work

In my research work, I provided novel techniques for analyzing the 4D geometries, enhancing the 4D depth perception, and interacting with the 4D geometries. However, my research work

still have some limitations. In the rest of this section, I list the limitation, and propose future work to overcome the limitations:

- **The potential of extend the developed techniques to visualize the 3-manifold embedded in 4D is uncertain.** In the current stage, my research work is focused on the 2-manifold embedded in 4D. Whether the proposed techniques can be extended to 3-manifold still have to be explored. To overcome this limitation, I plan to start from extend the current 4D visualization pipeline for the 3-manifold. After that, the problem of using dual-view visualization system for analyzing the 3-manifold should be studied. And lastly, I will also explore the potential of enhancing the 4D depth perception of 3-manifold by using halo technique.
- **The area of none-rigid 4D geometries is unexplored.** Some problems such as visualizing the mathematic procedure like untying the twisted spun trefoil knot are related to the none-rigid 4D geometries. In my future work, I would like to create 4D deformation system by using Free-form deformation technique. Free-form deformation (FFD) is a technique for manipulating any shape in a free-form manner. It is flexible and interactive. Thus, it is the method I would like to try to port to 4D. In this work, dealing with the interaction is a challenge. Botsch et.al. [16] proposed a framework for users to customize a deformation region and a handle region. By using their framework, users can quickly and flexibly deform a 3D model. Similar to Botsch et.al., I will also try to design a more efficient and more flexible interface, e.g., users can define a control region in which the geometry will not deform but can only translate, and a frozen region, in which the geometry within the control region cannot move at all. The geometry in between these two regions belongs to the deformable region. More than that, I will try to employ interaction methods such as multi-touch and haptics. These devices have more degree-of-freedom. Which may help to simplify the interaction.

- **The user group of our system is small.** The end users of our systems are mathematicians and the students in the university. The users are supposed to have some visualization and mathematic background. Whether general public can understand the 4D geometries is still a unknown. To overcome this, a formal userstudy involves participants without visualization and mathematic background should be conducted first. And then, according to userstudy results, I will design and develop an interactive educational tool, and study improve of participants understanding by another userstudy. Lastly, I plan to develop some smaller stand-alone applications. So that the general public can have more chance to access the 4D mathematical models. A knotted-sphere untying system will be the first application that I am going to develop. Smooth curves can always be untied in 3D without self intersection, but they are knotted in 3D as if they are closed curves. Surfaces can be knotted in 4D. However, some surfaces can be unknotted although they appear to be knotted [62]. Twist spun trefoil knot (a kind of knotted-sphere) is this kind of knot. They can be “untied” in principle by a series of deformations developed by Dennis Roseman [80, 113], during which the surface does not produce self-intersections.

References

- [1] E. A. Abbott. *Flatland*. Dover Publications, Inc., 1952.
- [2] A. Appel, F.J. Rohlf, and A. Stein. The haloed line effect for hidden line elimination. In *SIGGRAPH 1979*, pages 151–157, 1979.
- [3] Almir Olivette Artero, Maria Cristina Ferreira de Oliveira, and Haim Levkowitz. Uncovering clusters in crowded parallel coordinates visualizations. In *IEEE Symp. on Info. Vis.*, pages 81–88, 2004.
- [4] Michael Balzer, Thomas Schlomer, and Oliver Deussen. Capacity-constrained point distributions: A variant of Lloyd’s method. *ACM Trans. on Graphics (SIGGRAPH)*, 28(3):86:1–86:8, 2009.
- [5] T. F. Banchoff. Visualizing two-dimensional phenomena in four-dimensional space: A computer graphics approach. In E. Wegman and D. Priest, editors, *Statistical Image Processing and Computer Graphics*, pages 187–202. Marcel Dekker, Inc., New York, 1986.
- [6] T. F. Banchoff. Beyond the third dimension: Geometry, computer graphics, and higher dimensions. *Scientific American Library*, 1990.
- [7] David C. Banks. Interactive manipulation and display of two-dimensional surfaces in four-dimensional space. In *Symposium on Interactive 3D Graphics*, pages 197–207, 1992.
- [8] David C. Banks. Illumination in diverse codimensions. In *SIGGRAPH 1994*, pages 327–334, 1994.

REFERENCES

- [9] D.C. Banks and K. Beason. Decoupling illumination from isosurface generation using 4D light transport. *IEEE TVCG (IEEE Visualization)*, 15(6):1595–1602, 2009.
- [10] Richard A. Becker and William S. Cleveland. Brushing scatterplots. *Technometrics*, 29(2):127–142, 1987.
- [11] Fabian Bendix, Robert Kosara, and Helwig Hauser. Parallel sets: Visual analysis of categorical data. In *IEEE Symp. on Info. Vis.*, pages 133–140, 2005.
- [12] Michael R. Berthold and Lawrence O. Hall. Visualizing fuzzy points in parallel coordinates. Technical report, University of California at Berkeley, Berkeley, CA, USA, 1999.
- [13] Praveen Bhaniramka, Rephael Wenger, and Roger Crawfis. Isosurfacing in higher dimensions. In *Proceedings of IEEE Visualization 2000*, pages 267–273, 2000.
- [14] J. Blaas, C.P. Botha, and F.H. Post. Extensions of parallel coordinates for interactive exploration of large multi-timepoint data sets. *IEEE TVCG (Proceedings of IEEE Visualization)*, 14(6):1436–1451, 2008.
- [15] Silvia Born, Alexander Wiebel, Jan Friedrich, Geric Scheuermann, and Dirk Bartz. Illustrative stream surfaces. *IEEE TVCG (Proceedings of IEEE Visualization)*, 16(6):1329–1338, 2010.
- [16] Mario Botsch and Leif Kobbelt. An intuitive framework for real-time freeform modeling. *ACM Trans. Graph.*, 23(3):630–634, 2004.
- [17] Selig Brodetsky. *A first course in nomography*. London, G. Bell and Sons, 1920.
- [18] S. Bruckner and M. Eduard Gröller. Volumeshop: an interactive system for direct volume illustration. In *Proceedings of IEEE Visualization 2005*, pages 671–678, 2005.
- [19] Stefan Bruckner and M. Eduard Gröller. Exploded views for volume data. *IEEE TVCG (Proceedings of IEEE Visualization)*, 12(5):1077–1084, 2006.
- [20] Stefan Bruckner and M. Eduard Gröller. Enhancing depth-perception with flexible volumetric halos. *IEEE TVCG (Proceedings of IEEE Visualization)*, 13(6):1344–1351, 2007.

REFERENCES

- [21] B. Buxton. Multitouch systems that I have known and loved. Technical report, Microsoft Research, 2007. <http://www.billbuxton.com/multitouchOverview.html>.
- [22] S. A. Carey, R. P. Burton, and D. M. Campbell. Shades of a higher dimension. *Computer Graphics World*, pages 93–94, October 1987.
- [23] Samuel Hsiao-Heng Chang, Lachlan Stuart, Beryl Plimmer, and Burkhard Wünsche. Origami simulator: a multi-touch experience. *CHI '09: Proceedings of the 27th international conference extended abstracts on Human factors in computing systems*, pages 3889–3894, 2009.
- [24] Michael Chen, S. Joy Mountford, and Abigail Sellen. A study in interactive 3D rotation using 2D control devices. In *SIGGRAPH 1988*, volume 22, pages 121–130, 1988.
- [25] Wei Chen, Zhicheng Yan, Song Zhang, John Allen Crow, David S. Ebert, Ronald M. McLaughlin, Katie B. Mullins, Robert Cooper, Zi'ang Ding, and Jun Liao. Volume illustration of muscle from diffusion tensor images. *IEEE TVCG (Proceedings of IEEE Visualization)*, 15(6):1425–1432, 2009.
- [26] Hyunsang Choi, Heejo Lee, and Hyogon Kim. Fast detection and visualization of network attacks on parallel coordinates. *Computers & Security*, 28(5):276–288, 2009.
- [27] A. Chu, C.-W. Fu, P.-A. Heng, and A. J. Hanson. GL4D: A GPU-based architecture for interactive 4D visualization. *IEEE TVCG (Proceedings of IEEE Visualization)*, 15(6):1587–1594, 2009.
- [28] Alan Chu, Chi-Wing Fu, Andrew J. Hanson, and Pheng-Ann Heng. GL4D: A GPU-based Architecture for Interactive 4D Visualization. *IEEE TVCG*, 15(6):1587–1594, 2009.
- [29] Aurélie Cohé, Fabrice Dècle, and Martin Hachet. tBox: a 3D transformation widget designed for touch-screens. In *CHI*, pages 3005–3008, 2011.
- [30] Robert L. Cook. Stochastic sampling in computer graphics. *ACM Trans. on Graphics*, 5(1):51–72, 1986.

REFERENCES

- [31] R. A. Cross and A. J. Hanson. Virtual reality performance for virtual geometry. In *Proceedings of IEEE Visualization 1994*, pages 156–163, 1994.
- [32] Jean-Baptiste de la Rivière, Cédric Kervégant, Emmanuel Orvain, and Nicolas Dittlo. CubTile: a multi-touch cubic interface. *VRST '08: Proceedings of the 2008 ACM symposium on Virtual reality software and technology*, pages 69–72, 2008.
- [33] José Díaz, Eva Monclús, Isabel Navazo, and Pere-Pau Vázquez. Adaptive cross-sections of anatomical models. *Comp. Graphics Forum*, 31(7-2):2155–2164, 2012.
- [34] Stewart Dickson. Right-hand half klein bottle 9.85 in. tall. http://www.ifp.illinois.edu/~sdickson/Tactile_Math.html.
- [35] J. Diepstraten, D. Weiskopf, and T. Ertl. Interactive cutaway illustrations. *Comp. Graphics Forum*, 22(3):523–532, 2003.
- [36] Kirk L. Duffin and William A. Barrett. Spiders: a new user interface for rotation and visualization of n-dimensional point sets. In *Proceedings of IEEE Visualization 1994*, pages 205–211, 1994.
- [37] David S. Ebert and Penny Rheingans. Volume illustration: non-photorealistic rendering of volume models. In *Proceedings of IEEE Visualization 2000*, pages 195–202, 2000.
- [38] Florian Echtler, Simon Nestler, Andreas Dippon, and Gudrun Klinker. Supporting casual interactions between board games on public tabletop displays and mobile devices. *Personal Ubiquitous Comput.*, 13:609–617, November 2009.
- [39] G. Ellis and A. Dix. Enabling automatic clutter reduction in parallel coordinate plots. *IEEE TVCG (Info. Vis.)*, 12(5):717–724, 2006.
- [40] Geoffrey Ellis, Enrico Bertini, and Alan J. Dix. The sampling lens: making sense of saturated visualisations. In *CHI Extended Abstracts*, pages 1351–1354, 2005.
- [41] Maarten H. Everts, Henk Bekker, Jos B. T. M. Roerdink, and Tobias Isenberg. Depth-dependent halos: Illustrative rendering of dense line data. *IEEE TVCG (Proceedings of IEEE Visualization)*, 15(6):1299–1306, 2009.

- [42] Elena Fanea, Sheelagh Carpendale, and Tobias Isenberg. An interactive 3D integration of parallel coordinates and star glyphs. In *IEEE Symp. on Info. Vis.*, pages 149–156, 2005.
- [43] S. Feiner and C. Beshers. Worlds within worlds: Metaphors for exploring n-dimensional virtual worlds. In *Proceedings of UIST 1990*, pages 76–83, 1990.
- [44] M.C. Ferreira de Oliveira and H. Levkowitz. From visual data exploration to visual data mining: a survey. *Visualization and Computer Graphics, IEEE Transactions on*, 9(3):378 – 394, 2003.
- [45] A. R. Forsyth. *Geometry of Four Dimensions*. Cambridge University Press, 1930.
- [46] George K. Francis. *A Topological Picturebook*. Springer Verlag, 1987.
- [47] Steinicke Frank, Hinrichs Klaus Hinrichs, Johannes Schöning, and Antonio Krüger. Multi-touching 3D data: Towards direct interaction in stereoscopic display environments coupled with mobile devices. In *AVI '08*, pages 46–49, 2008.
- [48] Chi-Wing Fu, Wooi-Boon Goh, and Allen Junxiang Ng. Multi-touch techniques for exploring large-scale 3D astrophysical simulations. *CHI '10: Proceedings of the 28th international conference on Human factors in computing systems*, pages 2213–2222, 2010.
- [49] Ying-Huey Fua, Matthew O. Ward, and Elke A. Rundensteiner. Hierarchical parallel coordinates for exploration of large datasets. In *IEEE Visualization*, pages 43–50, 1999.
- [50] Amy Gooch, Bruce Gooch, Peter Shirley, and Elaine Cohen. A non-photorealistic lighting model for automatic technical illustration. In *SIGGRAPH 1998*, pages 447–452, 1998.
- [51] M. Graham and J. Kennedy. Using curves to enhance parallel coordinate visualisations. In *International Conference on Information Visualization*, pages 10–16, 2003.
- [52] Paul Haeberli. Paint by numbers: abstract image representations. In *SIGGRAPH 1990*, pages 207–214, 1990.
- [53] Jefferson Y. Han. Low-Cost multi-touch sensing through frustrated total internal reflection. In *UIST '05*, pages 115–118, 2005.

REFERENCES

- [54] Mark Hancock, Thomas ten Cate, and Sheelagh Carpendale. Sticky tools: full 6DOF force-based interaction for multi-touch tables. In *ITS '09*, pages 133–140, 2009.
- [55] A. J. Hanson. The rolling ball. In David Kirk, editor, *Graphics Gems III*, pages 51–60. Academic Press, Cambridge, MA, 1992.
- [56] A. J. Hanson and R. A. Cross. Interactive visualization methods for four dimensions. In *Proceedings of IEEE Visualization 1993*, pages 196–203, 1993.
- [57] A. J. Hanson and P. A. Heng. Visualizing the fourth dimension using geometry and light. In *Proceedings of IEEE Visualization 1991*, pages 321–328, 1991.
- [58] A. J. Hanson and P. A. Heng. Four-dimensional views of 3D scalar fields. In *Proceedings of IEEE Visualization 1992*, pages 84–91, 1992.
- [59] A. J. Hanson and P. A. Heng. Illuminating the fourth dimension. *IEEE Computer Graphics and Applications*, 12(4):54–62, 1992.
- [60] A. J. Hanson, K. Ishkov, and J. Ma. Meshview. A portable 4D geometry viewer written in OpenGL/Motif, available by anonymous ftp from ftp.cs.indiana.edu:pub/hanson.
- [61] Andrew J. Hanson. knot⁴. *SIGGRAPH Video Review*, 93(1), 1993. 3:29 minute video animation.
- [62] Andrew J. Hanson, Tamara Munzner, and George Francis. Interactive methods for visualizable geometry. *Computer*, 27(7):73–83, July 1994.
- [63] Andrew J. Hanson and Hui Zhang. Multimodal exploration of the fourth dimension. In *Proceedings of IEEE Visualization 2005*, pages 263–270, 2005.
- [64] Robert Hardy and Enrico Rukzio. Touch & interact: touch-based interaction of mobile phones with displays. In *MobileHCI '08*, pages 245–254, 2008.
- [65] Helwig Hauser, Florian Ledermann, and Helmut Doleisch. Angular brushing of extended parallel coordinates. In *IEEE Symp. on Info. Vis.*, pages 127–130, 2002.
- [66] Julian Heinrich and Daniel Weiskopf. Continuous parallel coordinates. *IEEE TVCG (Proceedings of IEEE Visualization)*, 15(6):1531–1538, 2009.

REFERENCES

- [67] D. Hilbert and S. Cohn-Vossen. *Geometry and the Imagination*. Chelsea, New York, 1952.
- [68] Jason Hise. Projection of a rotating tesseract, 2007. <http://en.wikipedia.org/wiki/Hypercube#mediaviewer/File:8-cell.gif>.
- [69] Patrick Edward Hoffman. *Table visualizations: a formal model and its applications*. PhD thesis, 2000.
- [70] Christoph M. Hoffmann and Jianhua Zhou. Some techniques for visualizing surfaces in four-dimensional space. *Computer Aided Design*, 23(1):83–91, 1991.
- [71] S. R. Hollasch. Four-space visualization of 4D objects, 1991. Master thesis, Arizona State University.
- [72] Mathias Hummel, Christoph Garth, Bernd Hamann, Hans Hagen, and Kenneth I. Joy. IRIS: Illustrative rendering for integral surfaces. *IEEE TVCG (Proceedings of IEEE Visualization)*, 16(6):1319–1328, 2010.
- [73] 3M inc. 3M Multi-touch Display M2256PW. <http://www.3m.com/>.
- [74] Apple Inc. iPod touch, iPad, iPhone, Magic Mouse Magic Tracpad. <http://www.apple.com>.
- [75] HP Inc. HP Inc. Touchsmart Tx2. <http://www.hp.com/united-states/campaigns/touchsmart/TouchSmart-Tx2>.
- [76] Microsoft Inc. Microsoft Inc.'s Windows 7. <http://windows.microsoft.com/windows-7>.
- [77] Microsoft Inc. Windows Phone 7. <http://www.microsoft.com/windowsphone>.
- [78] Nokia Inc. X6. http://en.wikipedia.org/wiki/Nokia_X6.
- [79] Ubuntu Inc. Ubuntu 10.10. <http://www.ubuntu.com/>.
- [80] Ayumu Inoue. A natural ch-diagram of the n-twist spun trefoil knot. Technical report, Tokyo Institute of Technology, 2008.

REFERENCES

- [81] Ayumu Inoue. A symmetric motion picture of the twist-spun trefoil. *Experimental Mathematics*, 22(1):15–25, 2013.
- [82] Alfred Inselberg. The plane with parallel coordinates. *The Visual Computer*, 1(4):69–91, 1985.
- [83] Alfred Inselberg and Bernard Dimsdale. Parallel coordinates: a tool for visualizing multi-dimensional geometry. In *IEEE Visualization*, pages 361–378, 1990.
- [84] V. Interrante and C. Grosch. Visualizing 3D flow. *IEEE Computer Graphics and Applications*, 18(4):49–53, 1998.
- [85] B. Kleiner J. M. Chambers, W. S. Cleveland and P. A. Tukey. *Graphical Methods for Data Analysis*. Wadsworth Press, Belmont, 1983.
- [86] Jimmy Johansson, Matthew Cooper, and Mikael Jern. 3-dimensional display for clustered multi-relational parallel coordinates. In *International Conference on Information Visualization*, pages 188–193, 2005.
- [87] Jimmy Johansson, Patric Ljung, Mikael Jern, and Matthew Cooper. Revealing structure within clustered parallel coordinates displays. In *IEEE Symp. on Info. Vis.*, pages 125–132, 2005.
- [88] Olga A. Karpenko, Wilmot Li, Niloy J. Mitra, and Maneesh Agrawala. Exploded view diagrams of mathematical surfaces. *IEEE TVCG (Proceedings of IEEE Visualization)*, 16(6):1311–1318, 2010.
- [89] Nicholas Katzakis and Masahiro Hori. Mobile phones as 3-DOF controllers: A comparative study. In *Dependable, Autonomic and Secure Computing, IEEE International Symposium on*, pages 345–349, 2009.
- [90] D.A. Keim. Visual techniques for exploring databases. In *Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining Tutorial Program*, pages T6 – 1, 1977.
- [91] Ji-Sun Kim, Denis Gračanin, Krešimir Matković, and Francis Quek. iPhone/iPod touch as input devices for navigation in immersive virtual environments. In *Virtual Reality Conference, IEEE*, pages 261–262, 2009.

- [92] K. Kin, M. Agrawala, and T. DeRose. Determining the benefits of direct-touch, bi-manual, and multifinger input on a multitouch workstation. In *Proceedings of Graphics Interface 2009*, pages 119–124, 2009.
- [93] Kenrick Kin, Tom Miller, Björn Bollensdorff, Tony DeRose, Björn Hartmann, and Maneesh Agrawala. Eden: a professional multitouch tool for constructing virtual organic environments. In *CHI '11*, pages 1343–1352, 2011.
- [94] PQ Labs. PQ Labs G Plus. <http://multi-touch-screen.com>.
- [95] Hongwei Li, Diego Nehab, Li-Yi Wei, Pedro Sander, and Chi-Wing Fu. Fast capacity constrained Voronoi tessellation. In *Symp. on Interactive 3D Graphics and Games*, Feb. 2010.
- [96] Wilmot Li, Lincoln Ritter, Maneesh Agrawala, Brian Curless, and David Salesin. Interactive cutaway illustrations of complex 3D models. *SIGGRAPH 2007*, 26(3), 2007.
- [97] Chunyuan Liao, Qiong Liu, Bee Liew, and Lynn Wilcox. PACER: fine-grained interactive paper via camera-touch hybrid gestures on a cell phone. In *CHI '10*, pages 2441–2450, 2010.
- [98] S. Lloyd. An optimization approach to relaxation labeling algorithms. *Image and Vision Computing*, 1(2):85–91, 1983.
- [99] Thomas Luft, Carsten Colditz, and Oliver Deussen. Image enhancement by unsharp masking the depth buffer. In *SIGGRAPH 2006*, pages 1206–1213, 2006.
- [100] Michael McCool and Eugene Fiume. Hierarchical poisson disk sampling distributions. In *Graphics interface*, pages 94–105, 1992.
- [101] Kevin T. McDonnell and Klaus Mueller. Illustrative parallel coordinates. *Comp. Graphics Forum*, 27(3):1031–1038, 2008.
- [102] James R. Miller and Estela A. Gavosto. The immersive visualization probe for exploring N-dimensional spaces. *IEEE Computer Graphics and Applications*, 24(1), 2004.
- [103] John J. Miller and Edward J. Wegman. Construction of line densities for parallel coordinate plots. In *Computing and Graphics in Statistics*, pages 107–123, 1991.

REFERENCES

- [104] Don P. Mitchell. Generating antialiased images at low sampling densities. *ACM SIGGRAPH*, 21(4):65–72, 1987.
- [105] A. Michael Noll. A computer technique for displaying N-dimensional hyperobjects. *Communication ACM*, 10(8):469–473, 1967.
- [106] Matej Novotny. Visually effective information visualization of large data. In *Central European Seminar on Computer Graphics*, pages 41–48, 2004.
- [107] Matej Novotny and Helwig Hauser. Outlier-preserving focus+context visualization in parallel coordinates. *IEEE TVCG*, 12(5):893–900, 2006.
- [108] Wei Peng, Matthew O. Ward, and Elke A. Rundensteiner. Clutter reduction in multi-dimensional data visualization using dimension reordering. In *IEEE Symp. on Info. Vis.*, pages 89–96, 2004.
- [109] Jean-Luc Peyrot, Frédéric Payan, and Marc Antonini. Feature-preserving direct blue noise sampling for surface meshes. In *Eurographics*, pages 9–12, 2013.
- [110] Peter Rautek, Stefan Bruckner, and M. Eduard Gröller. Semantic layers for illustrative volume rendering. *Proceedings of IEEE Visualization 2007*, 13(6):1336–1343, 2007.
- [111] Peter Rautek, Stefan Bruckner, M. Eduard Gröller, and Ivan Viola. Illustrative visualization: new technology or useless tautology? *SIGGRAPH Computer Graphics (course note)*, 42(3):1–8, 2008.
- [112] Jason L. Reisman, Philip L. Davidson, and Jefferson Y. Han. A screen-space formulation for 2D and 3D direct manipulation. In *CHI '09*, pages 69–78, 2009.
- [113] Dennis Roseman. Reidemeister-type moves for surfaces in four space. *Banach Center Publications*, 42:348–380, 1998.
- [114] Volker Roth and Thea Turner. Bezel swipe: conflict-free scrolling and multiple selection on mobile touch screen devices. In *CHI '09*, pages 1523–1526, 2009.
- [115] Greg Schussman, Kwan-Liu Ma, David Schissel, and Todd Evans. Visualizing DIII-D tokamak magnetic field lines. In *Proceedings of IEEE Visualization 2000*, pages 501–504, 2000.

- [116] James Shearer, Michael Ogawa, Kwan-Liu Ma, and Toby Kohlenberg. Pixelplexing: Gaining display resolution through time. In *IEEE Pacific Visualization Symp.*, pages 159–166, 2008.
- [117] H. Siirtola. Combining parallel coordinates with the reorderable matrix. In *Coordinated and Multiple Views in Exploratory Visualization*, pages 63–74, 2003.
- [118] Harri Siirtola. Direct manipulation of parallel coordinates. In *CHI (extended abstracts)*, pages 119–120, 2000.
- [119] Peng Song, Woon Boon Goh, Chi-Wing Fu, Qiang Meng, and Pheng-Ann Heng. WYSIWYF: exploring and annotating volume data with a tangible handheld device. In *Proceedings of the 2011 annual conference on Human factors in computing systems*, CHI '11, pages 1333–1342, 2011.
- [120] K. V. Steiner and R. P. Burton. Hidden volumes: The 4th dimension. *Computer Graphics World*, pages 71–74, February 1987.
- [121] Qian Sun, Chi-Wing Fu, and Ying He. An interactive multi-touch sketching interface for diffusion curves. In *Proceedings of the 2011 annual conference on Human factors in computing systems*, CHI '11, pages 1611–1614, 2011.
- [122] Nikolai A. Svakhine and David S. Ebert. Interactive volume illustration and feature halos. In *Proceedings of the 11th Pacific Conference on Computer Graphics and Applications*, pages 347–354, 2003.
- [123] Nikolai A. Svakhine, David S. Ebert, and D. Stedney. Illustration motifs for effective medical volume illustration. *IEEE Computer Graphics and Applications*, 25(3):31–39, 2005.
- [124] The Geometry Center of the University of Minnesota. Geomview, 1987. <http://www.geomview.org/>.
- [125] Sebastien Tricaud and Philippe Saadé. Applied parallel coordinates for logs and network traffic attack analysis. *Journal in Computer Virology*, 6(1):1–29, 2010.
- [126] Greg Turk. Generating random points in triangles. pages 24–28, 1990.

REFERENCES

- [127] Roy van Pelt, Anna Vilanova, and Huub van de Wetering. Illustrative volume visualization using GPU-based particle systems. *IEEE TVCG*, 16(4):571–582, 2010.
- [128] Nicolas Villar, Shahram Izadi, Dan Rosenfeld, Hrvoje Benko, John Helmes, Jonathan Westhues, Steve Hodges, Eyal Ofek, Alex Butler, Xiang Cao, and Billy Chen. Mouse 2.0: multi-touch meets the mouse. In *UIST '09*, pages 33–42, 2009.
- [129] Lujin Wang, J. Giesen, K.T. McDonnell, P. Zolliker, and K. Mueller. Color design for illustrative visualization. *Proceedings of IEEE Visualization 2008*, 14(6):1739–1754, 2008.
- [130] Rainer Wegenkittl, Helwig Löffelmann, and Eduard Gröller. Visualizing the behavior of higher dimensional dynamical systems. In *IEEE Visualization*, pages 119–125, 1997.
- [131] Edward J. Wegman. Hyperdimensional data analysis using parallel coordinates. *Journal of the American Statistical Association*, 85(411):664–675, 1990.
- [132] D. Weiskopf, M. Borchers, T. Ertl, M. Falk, O. Fechtig, R. Frank, F. Grave, A. King, U. Kraus, T. Muller, H.-P. Nollert, I. R. Mendez, H. Ruder, T. Schafhitzel, S. Schar, C. Zahn, and M. Zatloukal. Explanatory and illustrative visualization of special and general relativity. *IEEE TVCG*, 12(4):522–534, 2006.
- [133] Andrew D. Wilson and Raman Sarin. Bluetable: connecting wireless mobile devices on interactive surfaces using vision-based handshaking. In *GI '07*, pages 119–125, 2007.
- [134] Georges Winkenbach and David H. Salesin. Computer-generated pen-and-ink illustration. In *SIGGRAPH 1994*, pages 91–100, 1994.
- [135] Jacob O. Wobbrock, Meredith Ringel Morris, and Andrew D. Wilson. User-defined gestures for surface computing. In *CHI '09*, pages 1083–1092, 2009.
- [136] Pak Chung Wong and R. Daniel Bergeron. Multiresolution multidimensional wavelet brushing. In *IEEE Visualization*, pages 141–148, 1996.
- [137] John I. Jr. Yellott. Spectral consequences of photoreceptor sampling in the rhesus retina. *Science*, 221(4608):382–385, 1983.

REFERENCES

- [138] Lingyun Yu, Pjotr Svetachov, Petra Isenberg, Maarten H. Everts, and Tobias Isenberg. FI3D: Direct-touch interaction for the exploration of 3D scientific visualization spaces. *IEEE TVCG*, 16(6):1613–1622, 2010.
- [139] Xiaoru Yuan, Peihong Guo, He xiao, Hong Zhou, and Huamin Qu. Scattering points in parallel coordinates. *IEEE TVCG*, 15(6):1001–1008, 2009.
- [140] H. Zhang and A. J. Hanson. Shadow-driven 4D haptic visualization. In *Proceedings of IEEE Visualization 2007*, volume 13, pages 1688–1695, 2007.
- [141] Hong Zhou, Xiaoru Yuan, Huamin Qu, Weiwei Cui, and Baoquan Chen. Visual clustering in parallel coordinates. *Comp. Graphics Forum*, 27(3):1047–1054, 2008.

Publications

Published:

- **Xiaoqi Yan**, Chi-Fu Lai, and Chi-Wing Fu. Visual Signature of High-Dimensional Geometry in Parallel Coordinates. IEEE Pacific Visualization Symposium (PacificVis), 65-72, 2014.
- Nicolas Mellado, Peng Song, **Xiaoqi Yan**, Chi-Wing Fu, and Niloy J. Mitra. Computational Design and Construction of Notch-free Reciprocal Frame Structures. Advances in Architectural Geometry, 2014.
- **Xiaoqi Yan**, Peng Song, Chi-Wing Fu, Wooi Boon Goh, and Kwan-Liu Ma. Exploring Volume Visualization with Whole-hand Multitouch Gestures. Pacific Graphics, 2013.
- Weiming Wang, **Xiaoqi Yan**, Chi-Wing Fu, Andrew J. Hanson, and Pheng-Ann Heng. Interactive Exploration of 4D Geometry with Volumetric Halos. Pacific Graphics, 2013.
- **Xiaoqi Yan**, Andrew J. Hanson, and Chi-Wing Fu. Educational software tool for 4D visualization: 4Dice, available at Apple Store, 2011.
- **Xiaoqi Yan**, Chi-Wing Fu, and Andrew J. Hanson. Multitouching the Fourth Dimension. IEEE Computer, 80-88, 2012.
- Weiming Wang, **Xiaoqi Yan**, Xie Yongming, Jing Qin, Wai-Man Pang, and Pheng-Ann Heng. A Physically-Based Modeling and Simulation Framework for Facial Animation. *Proceedings of the 2009 Fifth International Conference on Image and Graphics*, 521-526, 2009.

Submitted:

REFERENCES

- Chi-Fu Lai, Sai-Kit Yeung, **Xiaoqi Yan**, Chi-Wing Fu, and Chi-Keung Tang. 3-D Navigation on Impossible Figures via Dynamic Terrain Generation. Submitted to IEEE Transactions on Visualization and Computer Graphics.