Graduate School Form 9
(Revised 8/89)

PURDUE UNIVERSITY
GRADUATE SCHOOL
Thesis Acceptance

This is to certify that the thesis prepared

By Jianhua Zhou

Entitled

Visualization of Four Dimensional Space

and Its Applications

Complies with University regulations and meets the standards of the Graduate School for originality and quality

For the degree of Doctor of Philosophy

Signed by the final examining committee:

Andrew M. Hanson, chair

Robert A. Lynch

Caroline Green

David C. Anderson

Approved by:

John R. Rice / William J. Gordon — Department Head — 25 November 1991 — Date

This thesis ☐ is ☒ is not to be regarded as confidential

Andrew M. Hanson

Major Professor

VISUALIZATION OF FOUR DIMENSIONAL SPACE AND ITS APPLICATIONS

A Thesis

Submitted to the Faculty

of

Purdue University

by

Jianhua Zhou

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

November 1991

To my parents,

Wenlong Zhou and Yuexian Chen

ACKNOWLEDGMENTS

I am most grateful to my advisor, Professor Christoph M. Hoffmann, for his constant encouragement and guidance, and for his reading the whole manuscript and making many valuable suggestions. He has significantly influenced my professional conduct and training. I want to particularly thank Professors David C. Anderson, Robert E. Lynch, and Concettina Guerra for serving on my thesis committee. I am also thankful to Professors Chanderjit Bajaj, Helmut Pottmann, Kokichi Sugihara, and George Vaněček, Jr. for discussions on CAGD and visualization. Although the Interactive 4D Visualization System was designed and programmed by myself, I cordially appreciate the kind help from William Bouma and Ravi Pradhan in making pictures and video tapes. My thanks are also due to all the friends who have helped and encouraged me during my stay at Purdue. Among them are Ching-Shoei Chiang, Jung-Hong Chuang, Neelam Jasuja, Honghai Shen, Jyh-Jong Tsay, Pamela Vermeer, and Jiaxun Yu.

My wife Yi Liu and daughter Vicky deserve special thanks for their love, support, and taking care of Karen who was born while I was writing this thesis. Such as they are, my accomplishments belong equally to them.

Finally, to my parents Wenlong Zhou and Yuexian Chen, who seemed to be proud of what I did, I would like to express my deepest gratitude. This thesis is dedicated to them.

This research has been supported in part by the National Science Foundation grant CCR 86-19817 and DMC 88-07550 and by the office of Naval Research under contract N00014-90-J-1599.

TABLE OF CONTENTS

Page

LIST OF FIGURES vi

ABSTRACT x

1. INTRODUCTION 1

1.1 Related Work 1
1.2 Proposed Method 3
1.3 Dimension Analogy 8
1.4 Thesis Organization 10

2. FUNDAMENTAL CONCEPTS 12

2.1 Orientation Specification 12
2.1.1 Euler Angles in n-space 13
2.1.2 Directions of Projection by Euler Angles 14
2.1.3 Quaternions 17
2.1.4 Directions of Projection by Quaternions 26
2.1.5 Relations between Euler Angles and Quaternions 28
2.1.6 Animation by Quaternions 30
2.2 Silhouettes and Envelopes 32
2.2.1 Silhouette Points 33
2.2.2 Envelopes 36
2.2.3 The Silhouette Surface of a Hypersurface in 4-space 37
2.2.4 The Normal of a Projected 2-Surface 38
2.3 Visibility 40

3. VISUAL PHENOMENA AND THEIR MEANING 44

3.1 Interpretation of Some Viewing Positions 44
3.1.1 Viewing Positions That Keep 2D Image Invariant 44
3.1.2 Viewing Positions with Special Effects 45
3.2 Interpretation of the 3D Images of Hypersurfaces 50

Page

3.2.1 Procedural Construction of the 3D Image of a Hypersurface . 50
3.2.2 Understanding by analogy 54
3.3 Observing the Curvature of a Hypersurface 59

4. SYSTEM ARCHITECTURE 68

4.1 System Overview 68
4.2 Polygonalization of Implicit 2-Surfaces in 4-Space 73
4.2.1 The Basic Algorithm and Data Structure 74
4.2.2 Newton Iteration for Point Refinement 77
4.2.3 Merging Polygons 82
4.3 Visibility Determination 84
4.3.1 Intersection of 2-surfaces in the Image Space 85
4.3.2 Determining the Partition Function on a 2-Surface 89
4.3.3 Sight Number Calculation and Propagation 94

5. APPLICATIONS 97

5.1 Understanding Differential Geometry 98
5.2 Collision Detection and Analysis 104
5.3 Scalar Fields in 3-Space 107
5.4 Generalized Offset Curves 109
5.4.1 Motivation 109
5.4.2 Definitions 115
5.4.3 Properties 116
5.4.4 Finding Optimal Q-offset Curves 128

6. CONCLUSION AND FUTURE WORK 131

6.1 Concepts and Intuitions 131
6.2 System and Applications 133
6.3 Toward the Fifth Dimension 134

BIBLIOGRAPHY 136

VITA 142

LIST OF FIGURES

Figure		Page
1.1	A torus projected into a plane and then projected into a line	8
2.1	Euler Angles: (a) $\theta_1, \theta_2, \theta_3$ in $\mathcal{R}^4$ (b) θ_4, θ_5 in $\mathcal{R}^3$ (c) θ_6 in $\mathcal{R}^2$	15
2.2	The body-fixed vector l and its projections in $\mathcal{R}^3$ and $\mathcal{R}^2$	16
2.3	Rotation of quaternions	21
2.4	The projections of the world base vectors i and l in the 3D image space .	27
2.5	Ambiguities in conversion to quaternions	31
2.6	Visibility	42
3.1	A square plate projected into a line. (a)viewing positions. (b)eye_3 at B, eye_2 at A. (c)eye_3 at A, eye_2 at B. (d)eye_3 at A, eye_2 at C. (e)eye_3 at D, eye_2 at E	45
3.2	Bracket viewed from $\theta = (0, 0, 0, 45, 60, 0)$ degrees	46
3.3	Bracket viewed from $\theta = (0, 0, 0, 45, 60, 0)$ but shaded by color scale representing w-values	46
3.4	Bracket viewed from $\theta = (45, 60, 90, 0, 0, 0)$; Positions of eye_4 and eye_3 are exchanged	47
3.5	Bracket cut by z-clipping plane to display isosurface w = constant . . .	47
3.6	The 3D image viewed from a different eye_3 direction: $\theta = (45, 60, 90, -75, 45, 0)$	48
3.7	The 3D image viewed from a different eye_3 direction: $\theta = (45, 60, 90, -75, 90, 0)$	48
3.8	Bracket viewed from $\theta = (45, 45, 45, 105, 90, 0)$. Only three surfaces in the grid with constant x, y and z values are displayed	51

Figure Page

3.9 Three isosurfaces on the hypersurface $x^2 + y^2 + z - w^2 = 0$ 51

3.10 The silhouette surface of the hypersurface 52

3.11 The boundary surface $z + 0.5 = 0$ of the hypersurface 52

3.12 The hypersurface viewed from $\theta = (45, 105, 78, 90, 81, 0)$ with visibility determination 53

3.13 The hypersurface viewed from $\theta = (45, 75, 102, 90, 81, 0)$ with visibility determination 53

3.14 Surface $x^2 + y - z^2 = 0$ with two different views 55

3.15 Hypersurface $x^2 + y^2 + z^2 - w^2 - 1 = 0$ viewed from $\theta = (-165, 45, 75, -165, 84, 0)$ 55

3.16 The dimension reduction of the silhouette surface in 3D image space . . 56

3.17 The same silhouette surface viewed from another direction 56

3.18 The hypersurface viewed from $\theta = (-165, 45, 15, -165, 84, 0)$ 57

3.19 Surface $x^2 + y^2 - z^2 - 1 = 0$ with three different views 57

3.20 Curvature of a Hypersurface 61

3.21 Hypersurface of Gauss Distribution 65

3.22 Hypersurface of Gauss Distribution Clipped by a Plane 65

4.1 System architecture 71

4.2 Point Refinement in 3-Space 78

4.3 Point Refinement in 4-Space 79

4.4 Silhouette Surface Before and After Visibility Determination 90

4.5 Intersection of Two Boundary Curves in Image Space 90

4.6 Intersection of Two Boundary Surfaces in Image Space 91

4.7 An Isosurface trimmed by the Silhouette Surface in Image Space 93

4.8 Function $\bar{f}(\lambda)$ near $\lambda = 0$ 96

Figure Page

5.1 Curves (a) $y - x^2 = 0$ and (b) $y^2 - x^3 = 0$ and their offset curves by 1 . 99

5.2 The Offset Curve (a) traced in $\mathcal{R}^4$ viewd from $\theta = (0, 0, 0, 0, 0, 0)$ 101

5.3 The Offset Curve (a) traced in $\mathcal{R}^4$ viewd from $\theta = (45, 105, 45, 75, 165, 0)$ 101

5.4 The Offset Curve (b) traced in $\mathcal{R}^4$ viewd from $\theta = (0, 0, 0, 0, 0, 0)$ 102

5.5 The Offset Curve (b) traced in $\mathcal{R}^4$ viewd from $\theta = (45, 40, 60, 105, 75, 0)$ 102

5.6 The Offset Curve (b) traced in $\mathcal{R}^3$ viewd from $\theta = (0, 0, 0, 0, 0, 0)$ 103

5.7 The Offset Curve (b) traced in $\mathcal{R}^3$ viewd from $\theta = (0, 0, 0, -40, 60, 0)$ . . 103

5.8 A cylinder and a sphere in motion 105

5.9 Intersection of a cylinder and a moving sphere with the same radius, viewed from $\theta = (0, 18, 9, 120, 75, 0)$ 106

5.10 The cylinder has a larger radius, viewed from $\theta = (0, 30, 105, -105, 30, 0)$ 106

5.11 The cylinder has a smaller radius, viewed from $\theta = (0, -15, 123, 120, 90, 0)$ 110

5.12 A snub dodecahedron 110

5.13 Contour and slicing surfaces show the structure of a virus 111

5.14 Contour and slicing surfaces within a subcube 111

5.15 The contour and slicing surfaces viewed from $\theta = (0, 0, 15, 315, 60, 0)$ . . 112

5.16 The contour and slicing surfaces viewed from $\theta = (0, 0, 90, 315, 90, 0)$ . . 112

5.17 A comparison between two cutters 113

5.18 Cutter moving direction 114

5.19 Q-offset curve of $(t, t^2), a = 0.25, b = 0.125, \theta = 30$ 117

5.20 Q-offset curve of $(t, t^2), a = 0.25, b = 0.086, \theta = 30$ 118

5.21 Q-offset curve of $(t, t^2), a = 0.25, b = 0.04, \theta = 30$ 119

5.22 Interference near a cusp 124

5.23 The hypersurface $\beta(t, b, \theta)$ 126

Figure Page

5.24 The curvature $\kappa(t, b, \theta)$ 126

5.25 Optimal Q-offset curve of $(t, t^2), a = 0.25$ 129

ABSTRACT

Zhou, Jianhua. Ph.D., Purdue University, December 1991. Visualization of Four Dimensional Space and Its Applications. Major Professor: Christoph M. Hoffmann.

In this thesis a method has been proposed to visualize curves, surfaces and hypersurfaces in four-dimensional space. Objects in 4-space are first projected into the 3D image space and further projected into the 2D image space. Four topics have been investigated: (1) Fundamental Concepts. (2) Visual Phenomena and Their Meaning. (3) System Architecture. (4) Applications.

The orientation of the 3D and 2D image spaces can be specified by a set of six Euler angles or a pair of quaternions. Since the the quaternion pairs representing 4D rotations are not unique, three useful forms have been discussed. Each form is suitable for rotation combination, for conversion between quaternions and Euler angles, or for user interface. The silhouette point of an m-surface with respect to a projection from n-space to l-space ($m \leq l < n$) has been defined. The close relationship between silhouette and envelope turned out to be useful in explaining some phenomena with a surface defined by the envelope theorem and in understanding the image of a hypersurface.

Several geometric properties and phenomena in 4-space have been demonstrated by computer generated pictures and animations. They include the ambiguity caused by projection, the degeneracy of the silhouette surface of a hypersurface, and the principal curvatures of a hypersurface.

The architecture of an interactive 4D visualization system has been presented. The system was built on z-buffer based graphics workstations. Algorithms and data structures for polygonalization, point refinement, merging polygons, and visibility determination in 4-space are discussed.

Finally several examples have been presented to illustrate the application of the visualization system: tool path generation for NC machines, collision detection and analysis for robot motion planning, and visualization of electron density data of a virus for molecular biology.

Supplementary media for this thesis, a video tape and a set of color slides, is available in the Film Library of Purdue University.

1. INTRODUCTION

High-dimensional space is playing an increasing role in computer-aided geometric design (CAGD) and solid modeling. Applications include describing the motion of 3D objects, modeling solids with nonuniform material properties, deriving spline curves uniformly, and formulating constraints for offset surfaces and Voronoi surfaces [Cam84, Hof89, Ros89, Sei90a]. Apart from the modeling aspects, it is important to develop visualization tools for high-dimensional space, for these tools can provide insights and methods for investigating geometric phenomena and dynamic systems [Ban86, KBBL86]. This thesis investigates this topic with a view towards CAGD and solid modeling, concentrating on four-dimensional space.

In this introductory chapter the previous related work is briefly reviewed. Then a proposed method is sketched. To facilitate the understanding and explanation of visualization of 4-space, an analogous example in 3-space is presented in the third section. The organization of the thesis is given in the last section of this chapter.

1.1 Related Work

High dimensional geometry and its visualization can be dated to the last century. In those books dedicated to geometry of four dimensions [For30, Man56, Cox47], concepts are often illustrated by figures from 4-space. A recent book by Banchoff [Ban90] contains rich materials about the history and approaches to visualization of four dimensional space. Here we review the various techniques for visualizing the high dimensional space by means of computer graphics.

In his book [Eck68] Eckhart proposed a method to project an 4D object into several planes that are orthogonal to different pairs of coordinate axes. The 2D images so obtained can be put together in a systematic way in analogy to the principal

views in traditional engineering drawings. For example, a plane is divided into four quadrants by X- and Y-axes. Each quadrant is considered as a 2D image plane in 4-space, namely, (x, y)-, (y, z)-, (z, w)-, or (w, x)-plane. This method can be used to project scattered data by computers [TT81], and it can be extended so that the 2D image planes are the support plane of the faces of a regular polytopes, not necessarily pairwise orthogonal. However, since this method only displays points or curves that are four or three dimensions lower than the dimension of the ambient space, and since the viewer must gather information from several different views, it may be very hard to interpret such pictures.

Noll introduced computer animation to visualize hyperobjects in his pioneering paper [Nol67]. With animation the viewer can connect mentally the consecutive pictures of slightly different views. At least it is easier than connecting several static pictures of quite different views as in the above method. Noll's movie also made use of stereoscope, trying to get the depth feeling of the image in 3-space. His hyperobjects were represented by discrete points connected by line segments (edges). It relied therefore on perspective projection to get the depth feeling in 4-space.

Banchoff and his colleagues used computer graphics and animation to investigate geometric phenomena related to complex function and dynamics systems [Ban86, KBBL86]. They displayed 2-surfaces in 4-space by two projections. 3D shading was added to the second projection. By using stereographic projection as the first projection, a hypersphere except for one point is mapped to the whole 3D space. The structure of the hypersphere is displayed via a set to flat tori which are the inverse images of circles on a sphere under the Hopf map [KL87].

The advantages of using two step projections was discussed by Rossignac [Ros89]. Several common CAD and solid modeling problems were reformulated in 4-space. He also proposed a method to display polytopes with light sources in 4-space. The visibility in 4-space was also mentioned in the context of projected polytopes. A different algorithm for visibility problem in general n-space was presented in [BS82]. It only considered the hidden lines of polytopes. As we discuss in this thesis, however, curved

surfaces require a more elaborated notion of visibility, and the simple techniques that suffice for polytopes must be significantly changed.

All the above methods are based on projection, and so preserve the properties studied in projective geometry. However, a picture so obtained by projection from 5 or higher dimensional space seems hardly comprehensible [Nol67]. There are a few methods that have been tried for higher dimensions. They are based on nonprojective maps. Inselberg proposed a method called the plane with parallel coordinates [Ins85, Ins90]. In the (X, Y)-plane n lines parallel to the Y-axis are drawn representing the n Cartesian coordinates of $\mathcal{R}^n$. The figure drawn is basically the image of objects under a nonprojective map from $\mathcal{R}^n$ to $\mathcal{R}^2$. Several nice dualities were found related to this map. Another method presented by Mihalisin is suitable for displaying multivariate functions [MGTS89, MTS91]. It can be understood as though the sampled function values are stored in an n-dimensional array and then displayed as a univariate function after enumerating the array elements linearly. It treats the high dimensional space inhomogeneously. For example, exchanging the role of x and y will drastically change the 2D graph.

In summary, projective methods are enhanced by animation, shading, and visibility determination, and so reveal more information about the 4D geometry; nonprojective methods have been proposed in an effort to handle much higher dimensions. There seems to be no one method suitable for *all* the problems, and there are many aspects in which existing methods can be improved. To the author's knowledge, the correlation between silhouette and the visibility of curved hypersurfaces have not been thoroughly discussed in the context of 4D visualization.

1.2 Proposed Method

Our method is based on the two-step projection approach. The objects in 4-space, i.e. curves, 2-surfaces and hypersurfaces, are first projected into 3-space and then further projected into 2-space. Since the second projection is quite familiar to us, we put emphasis on understanding the first projection and the combination of the two

projections. The central part of the work is dealing with 2-surfaces in 4-space. The projected 2-surface in 3-space is again a surface and can be shaded in a standard way. Curves in 4-space are displayed as curves on 2-surfaces, e.g. the silhouette curves of 2-surfaces. Hypersurfaces in 4-space are displayed through their silhouette surfaces, boundary surfaces, intersection surfaces, and isosurfaces that are all 2-surfaces. Even though we do not display the whole hypersurface, some of its geometric properties can be so recovered.

The method is basically an extension of certain 3D graphics techniques. The crucial difference is that when we look at the 2D images of 3D objects, we are *outside* the image space, but when we look at the 3D images of 4D objects, we are *inside* the image space. This fact causes difficulties in that we can look at all parts of the 2D image at the same time, but we cannot do this in a 3D image since, e.g. parts of it are occluded by other image objects. So we include the following functionalities:

- The orientation of the first and second projections is controlled and coordinated in a systematic way so that the image is more predictable.

- The silhouette points of 2- or 3-surfaces with respect to one or two projections, and their relationship to the envelopes, are discussed. Silhouette surfaces or curves turn out to be crucial in sketching the shape, determining visibility, and recognizing certain geometric properties of 2- or 3-surfaces in 4-space.

- The visibility with respect to both projections are explored. Ambiguities could be caused by the first and/or the second projection. An algorithm is presented for the visibility determination associated with the first projection.

These functionalities are further explained as follows.

Orientation. It is well known that the orientation of a rigid body in 4-space can be specified by six independent parameters. But how these parameters affect the images in 3-space and 2-space, i.e. in the 3D image space and the 2D image space, respectively, have not been well studied. Suppose an object is rotating in 4-space and

its 3D image is inspected. Since we are in the same space as the 3D image, in order to see it from all sides, the 3D image has to be further rotated. Do we need another set of three parameters to specify the orientation of the 3D image before it is projected into 2D image? The answer depends on how the image space is defined. To specify systematically the orientation of objects, of the image space, and of the projection, we use generalized Euler angles. The centers of the first and second projections are called eye_4 and eye_3, respectively. The 3D image space is defined to be orthogonal to the direction of eye_4, and the 2D image space is orthogonal to both the eyes' direction. They are arranged so that their orientation can be concisely expressed by a set of six Euler angles. Three of them are used for the control of eye_4, and two are for the control of eye_3. The last one would be used for control of eye_2, the center of the third projection, but it is not necessary for us since we live in 3-space. Interpretation of some useful angle choices will be given with examples. The generalized Euler angles have other advantages such as keeping the vertical directions, and being compatible to existing 3D graphics user interfaces. However, the Euler angles have some drawbacks such as the singularities encounted in converting matrices to Euler angles, and the difficulties in animation by interpolation of Euler angles [Sei90b].

Quaternions. To overcome the problems of Euler angles, many authors have advocated the use of quaternions for orientation specification, especially in 3D animations [Mar85, Sho85, Sei90b]. Likewise, the quaternions can be used for orientation specification in 4-space and 4D animations. The difference is that any 3D rotation is representable by *one quaternion* while any 4D rotation is representable by *a pair of quaternions.* To make use of quaternion for 4D visualization, we need to understand how a quaternion rotates in 4-space, a task that is much more confusing than the case of 3-space. For example, it is well known that a 3D rotation always has an axis and any vector parallel to this axis is invariant under the rotation. We will show that the 4D rotation by a nonreal quaternion has no fixed-point at all. The best we can do is to decompose the the rotation into two *orthogonal subrotations.* Moreover, the quaternion pair to represent a 4D rotation is not unique. We will discuss the three

most useful forms. The first is suitable for rotation combination and interpolation. The second is suitable for conversion between quaternions and matrices or between quaternions and Euler angles. The third one has a simpler interpretation, and so it is suitable for the user interface.

Silhouettes. When a surface in 3-space is projected into a plane, those points on it where the surface normal is orthogonal to the ray from eye to the point are called the *silhouette points.* The concept is extended to surfaces of dimension m in $\mathcal{R}^n$ with respect to a projection from $\mathcal{R}^n$ to $\mathcal{R}^l$ $(1 \leq m \leq l < n)$. Particularly, the silhouette points of a 3-surface (hypersurface) in $\mathcal{R}^4$ with respect to the first projection comprise a *silhouette surface* of the hypersurface. The silhouette points of a 2-surface in $\mathcal{R}^4$ with respect to the combined first and second projections comprise a *silhouette curve.* In modern 3D graphics workstations, the silhouette curve of a surface need not be constructed explicitly. When the polygons representing the surface are small enough, the picture with hidden surfaces removed shows the approximate silhouette curve by the discontinuity of shading. This technique cannot be extended into 4-space. If all the points on the hypersurface are shaded and projected into 3D image space, we get a 3D volume. Each point in the volume has its own color of different intensity according to some kind of shading model in 4-space. Such an "emitting" volume cannot be seen clearly due to the fact that we are in the same space as the 3D image. For the same reason showing a hypersurface by dense isosurfaces is unsuitable. A good method for visualizing the hypersurface is to construct the silhouette surface explicitly, and to display it with boundary surfaces, self-intersection surface (if any) and a *sparse* set of isosurfaces.

Envelopes. In 2-space, the envelope of a family of curves is another curve tangent to every curve in the family. The definition generalizes to arbitrary n-space straightforwardly. What is its use in visualization? We show that in some sense the envelope and silhouette are equivalent concepts. This link of silhouette and envelope provides several applications. First, in CAGD and solid modeling the *dimensionality paradigm* is suitable for representing exactly complex surfaces that satisfy prescribed

constraints, such as offset surfaces, blending surfaces, and equal-distance or Voronoi surfaces [Hof90, HV91]. The equations of constraint can be formulated by the envelope theorem. From another point of view, these surfaces are the sets of silhouette points of surfaces in higher dimension projected into lower dimension. Visualization helps explain some phenomena that may occur during the surface interrogation. Second, 4D visualization is subtle in that often the pictures look dazzling. Although the silhouette surface of a hypersurface can be generated by computer from its definition, it will be more comprehensible if the silhouette surface can also be constructed by the viewer in his/her mind. Envelopes offer a conceptual tool in such a construction. In fact, most of the pictures of the hypersurfaces presented in this thesis had been qualitatively predicted by the author before there were generated by the system. Third, the correlation between envelope and silhouette reveals the singularities in the intersection of surfaces on a hypersurface, necessitating some precautions in designing the algorithms for visibility determination.

Geometric Properties. Visualization has been frequently utilized for observing geometric properties, such as coincidence, collinearity, tangency, continuity, convexity, and curvature. These properties can also be observed in 4-space through 4D visualization. Among them, observing the curvature seems hardest. We will discuss how to infer the curvature of a hypersurface qualitatively through its 3D image. Again, the silhouette surface of the hypersurface plays an important role in curvature observation.

Visibility. Projection is a many-to-one map and causes ambiguity when the image is interpreted. An effective technique to eliminate the ambiguity is called the hidden surface removal, or the visibility determination. Visibility is a concept associated with a projection down one dimension. In the case of a projection composed of several steps, visibility associated with each step has to be considered. Another problem is the dimension of the potentially visible objects and the dimension of the potentially hiding objects. Here object refers to a surface of any dimension and its image under any projection. For example, a solid in 3-space, i.e. an object of dimension 3, can be

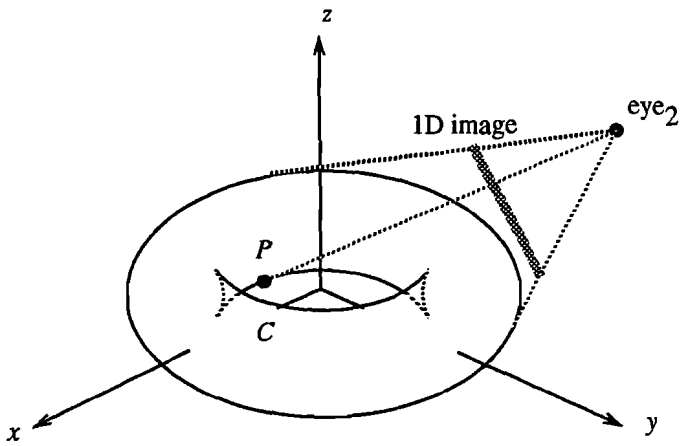

Figure 1.1 A torus projected into a plane and then projected into a line

the image of 3-surface in 4-space, or the image of 3- or 4-surface in 5-space, and so on. If these surfaces can be potentially hiding objects in 3-space, most 2-surfaces could be hidden. Our solution is that for a projection from $\mathcal{R}^n$ to $\mathcal{R}^{n-1}$, the hypersurface in $\mathcal{R}^n$ is the potentially hiding object. Its image in $\mathcal{R}^{n-1}$ is neither the potentially visible object nor the potentially hiding object. Applying this rule to 4D visualization, although in 3D image space a hypersurface is invisible, the 2-surfaces on it can be partially or totally hidden by the hypersurface.

1.3 Dimension Analogy

When we try to visualize 4-space, the situation is similar to the inhabitants of 2-space, called the flatland in [Abb63], who try to visualize 3-space. It will be helpful if we first investigate an analogous situation with reduced dimension, called the *dimension analogy*, and see what problems may happen. As an example let us investigate the silhouette with respect to the projections from 3-space to 1-space.

Consider a torus projected into a 2-space that contains the page on which Figure 1.1 is printed. The center of this projection, eye_3, is somewhere in the first octant

of the (x, y, z)-coordinate system above the page. Imagine now that we live in that 2-space and try to "see" the picture in a 1D projection. This is the second projection which produces a 1D image of the torus, on a line. The center of the second projection, called eye_2, is then confined to the plane orthogonal to the direction of eye_3, i.e. orthogonal to the vector from the origin to eye_3 in 3-space.

Figure 1.1 actually is not the projection of the whole torus but the projection of the *silhouette curve* of the torus with respect to the first projection. A point on a surface becomes a silhouette point if the tangent plane of the surface at that point is projected into a line. This happens when the ray from that point to eye_3 falls into the tangent plane. Clearly, when eye_3 is moving, the silhouette curve changes its shape and so does its projection. Now let us fix the eye_3 position and concentrate on this silhouette curve as an ordinary space curve C. A point on a curve becomes a *pinch point* if the tangent line of the curve at that point is projected into a point.[1] This happens when the ray from that point to eye_3 coincides with the tangent line. The four cusps in figure 1.1 are such points. They may cause trouble in determining the normal of the projected curve, which is necessary if shading is added to the second projection. The second projection may also introduce pinch points, for instance the point P on C in Figure 1.1. It appears as a discontinuity of shading in the 1D image. Note that the ray from P to eye_2 does not necessarily coincide with the tangent line of C at P in 3-space. Also note that the tangent plane of the torus at P is mapped to a point under the two projections. So it could be called a *doubly silhouette point* of the torus with respect to the two projections. The concepts of silhouette, doubly silhouette and pinch point have the common feature that at such points the tangent space of an object reduces in dimension under the projection.

[1]For the definition of pinch points on a 2-surface in 4-space see [Ban86].

1.4 Thesis Organization

In this thesis the proposed method is elaborated in both theoretical and practical aspects. Chapter 2 defines the fundamental concepts for our method of 4D visualization: the specification of the orientations of objects and projections by Euler angles and quaternions, the silhouette point of an m-surface and its relationship with envelopes, and the visibility determination for elimination of ambiguities caused by projections. All these concepts, except for the quaternions, are given in general form valid for arbitrary dimensions, as well as in special form suitable for 4D visualization.

In Chapter 3, Visual Phenomena and Their Meanings, several pictures generated by our interactive 4D visualization system are presented. We explain how to adjust Euler angles to achieve special visual effects, and how to construct the image of a hypersurface in one's mind so as to understand the computer generated pictures. The possible degeneracy of the silhouette surface is explained by the dimension analogy. Finally, a method for the curvature observation is derived and illustrated by examples.

Chapter 4 is devoted to the architecture of the interactive 4D visualization system. The input to the system can be the functions defining implicit or parametric 2-surfaces in 4-space, or discrete function values representing a 3D scalar field. Algorithms are presented for polygonization and visibility determination. The various singularities caused by the projection, and the desingularization techniques based on geometric intuition are also discussed. The discussion would be extremely complicated, if not impossible, without the pictures which are the result of 4D visualization. The design of the system architecture has met its original goal: interactivity, flexibility, and compatibility to 3D graphics.

The wide applicability of 4D visualization can only be sampled in Chapter 5. Examples include the offset curves displayed as the silhouette curve of a 2-surface with respect to the projection from 4-space to 2-space. Rotation in 4-space reveals that some apparent singularities do not really exist in 4-space. Collision of 3D objects can be detected and analyzed in (x, y, z, t)-space. The intersection of two moving surfaces

is thus a 2-surface in 4-space. Some precautions are drawn for the initial colliding point problem from observing the pictures. The 3D scalar field display is common in 3D graphics. Here it is put in 4-space and seen from a different view. The final example, tool path generation for 5-axes milling machines, discusses a problem in 5-space using some simplifications. Then the problem and its solution are visualized in 4-space.

Chapter 6 summarizes our results and presents some considerations for future work: toward the fifth dimension.

2. FUNDAMENTAL CONCEPTS

In this chapter we discuss some fundamental concepts in visualization of high dimensional space via projection: orientation specification, silhouettes and envelopes, and visibility. In each of the following sections, after the general form applicable to arbitrary n-dimensional space, a specific form for 4D visualization is presented.

2.1 Orientation Specification

To visualize objects in n-space, it is necessary to specify and adjust interactively the orientation of objects, projection directions, and light source directions. They can be uniformly considered as the orientations of rigid bodies. The orientation of a *body-fixed coordinate system* with respect to the *reference coordinate system* can be expressed by an $n \times n$ orthonormal matrix $A = (a_{ij})$ called the *direction cosine matrix*:

$$\mathbf{p} = A\mathbf{q}$$

where $\mathbf{p} = (p_1, \ldots, p_n)^T$ is a vector expressed in the reference coordinates, and $\mathbf{q} = (q_1, \ldots, q_n)^T$ is the same vector in the body-fixed coordinates. Among the $n \times n$ elements in A only $n(n-1)/2$ of them are independent. It will be convenient to specify the orientation by a set of $n(n-1)/2 + m$ parameters with m constraints, where m is a small number, say 0, 1 or 2. This problem is called the parameterization of the rotation group. In [Stu64] a number of methods for parameterizing the 3D rotations are reviewed. Not all of the methods can be generalized into arbitrary n dimensional spaces. In this section we discuss two methods: the generalized Euler angles for rotations in arbitrary n-space, and the quaternions for rotations in 3- and 4-space.

2.1.1 Euler Angles in n-space

The matrix A can be written as the product of $n(n-1)/2$ basic rotation matrices. The *basic rotation* is a one-parameter rotation within a plane spanned by two base vectors of the coordinate system. The basic rotations should be chosen systematically so that the geometric relationship is easy to explain and to remember. Some common choices in 3D are the Euler angles and Bryant angles [Wit77].[1]

In 3D kinematics, Euler angles specify the orientation of objects by three successive basic rotations:

$$\mathbf{p} = R^3_{xy}(\theta_1)R^3_{yz}(\theta_2)R^2_{xy}(\theta_3)\mathbf{q}$$

where $R^i_{xy}(\cdot)$ is the basic rotation matrix in the (x, y)-plane. The number superscript indicates the *rotation phases*, explained as follows.

We conceptualize 3D Euler angles as two separate rotation phases: In the first phase, specified by $R^3_{xy}(\theta_1)R^3_{yz}(\theta_2)$, the body-fixed z-axis is oriented into its final position in 3-space. In the second phase, a single rotation, $R^2_{xy}(\theta_3)$, brings the body-fixed x- and y-axes into their final positions within the the 2D subspace orthogonal to the (oriented) body-fixed z-axis.

It is clear that with this conceptualization Euler angles can be naturally extended into arbitrary n-space. The Euler angles in $(n + 1)$-space can be considered as n rotation phases:

1. The first phase orients the body-fixed x_{n+1}-axis by n basic rotations in the (x_1, x_2)-plane, (x_2, x_3)-plane, ..., (x_n, x_{n+1})-plane.

2. The remaining $n - 1$ phases orient the n-dimensional subspace orthogonal to the x_{n+1}-axes by the Euler angles in n-space.

In particular, the orientation of objects by Euler angles in 4-space is expressed as:

$$\mathbf{p} = R^4_{xy}(\theta_1)R^4_{yz}(\theta_2)R^4_{zw}(\theta_3)\ R^3_{xy}(\theta_4)R^3_{yz}(\theta_5)\ R^2_{xy}(\theta_6)\,\mathbf{q}$$

[1]These angles are all referred to as the Euler angles. Here we use the name convention in [Wit77].

Figure 2.1 shows the three rotation phases. The body-fixed axes after the i-th rotation are denoted by x_i, y_i, z_i, w_i.

To calculate Euler angles from the direction cosine matrix A we need the *inverse Euler angle formula.* Since the definition is recursive, it is not hard to derive the formula for arbitrary n-space as follows:

1. Calculate the angles $\theta_1, \ldots, \theta_{n-1}$ from the last column of A via

$$\theta_1 = -\arctan(\frac{a_{1n}}{a_{2n}}) \tag{2.1}$$

$$\theta_i = -\arctan(\frac{\sqrt{\sum_{j=1}^{i} a_{jn}^2}}{a_{i+1,n}}) \quad i = 2, \ldots, n-1 \tag{2.2}$$

2. Calculate the remaining angles recursively from the $n-1 \times n-1$ submatrix of

$$(R^n_{x_1x_2}(\theta_1) \cdots R^n_{x_{n-1}x_n}(\theta_{n-1}))^T A$$

Within each rotation phase only the angle of $R^i_{x_1x_2}$ has a full range of 2π. The others are restricted to a range of π so as to eliminate ambiguities. That is why there is no $\pm$ before the square root in (2.2).

The i-th column of the direction cosine matrix can be considered as the reference coordinates of the i-th base vector of the body-fixed coordinate system. From (2.1) and (2.2) we see that the angles in i-th rotation phase are the polar coordinates of the $(n-i+1)$-th base vector of the body-fixed coordinate system. Figure 2.2 shows the case of 4-space, where l is the fourth base vector (in the direction of w-axis) of the body-fixed coordinate system. The orthographic projection from $\mathcal{R}^4$ to $\mathcal{R}^i$ is denoted as π^i_4.

Note that numerical difficulties arise when both the numerator and the denominator in the argument of arctan are close to zero. This is a well-known drawback of Euler angles [Wit77].

2.1.2 Directions of Projection by Euler Angles

In 4-space, we assume a *world coordinate system* as the reference coordinate system. Each object is translated and rotated by specifying a transformation relating

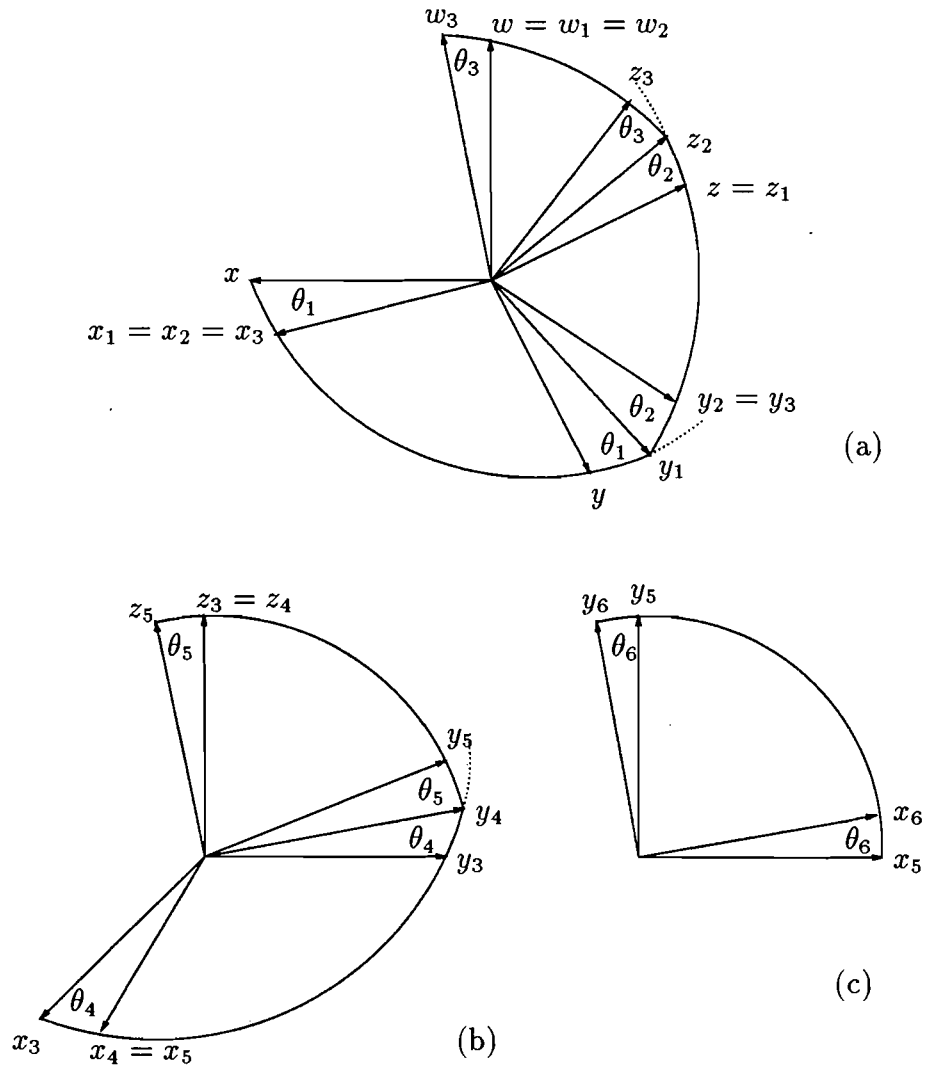

Figure 2.1 Euler Angles: (a) $\theta_1, \theta_2, \theta_3$ in $\mathcal{R}^4$ (b) θ_4, θ_5 in $\mathcal{R}^3$ (c) θ_6 in $\mathcal{R}^2$

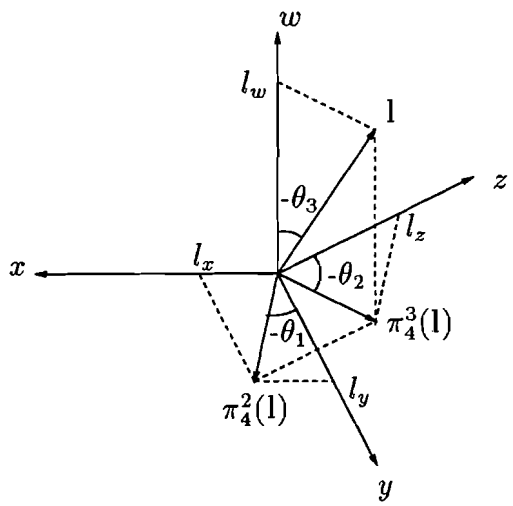

Figure 2.2 The body-fixed vector l and its projections in $\mathcal{R}^3$ and $\mathcal{R}^2$

its body-fixed coordinate system to the world coordinate system. The 4-space is first projected into the *3D image space* orthogonal to the first projection direction. Then, the 3D image space is projected into the *2D image space* orthogonal to both the first and second projection directions. The first and second centers of projection are called eye_4 and eye_3, respectively. They can be at finite or infinite distance from the origin. We require that the origins of the three coordinate systems of the 4D world, of the 3D image, and of the 2D image space coincide. Thus, the relationship between the coordinate systems can be expressed by pure rotations with only six independent parameters.

We specify the projection directions in the same way as object orientation. A single set of Euler angles specifies the orientation of both eye_4 and eye_3 because we can think of the two lines from the origin to eye_4 and eye_3 as the w- and z-axes of a rigid object. Thus, the projection is determined by $\theta = (\theta_1, \ldots, \theta_6)$ and by the pair $r = (r_4, r_3)$ of the reciprocal distances of eye_4 and eye_3 from the origin. In figure 2.1, (x, y, z, w) is the world coordinate system. (x_3, y_3, z_3) is the 3D image coordinate system, and (x_5, y_5) is the 2D image coordinate system.

The position of eye_4 in the world coordinate system is controlled by $\theta_1, \theta_2, \theta_3$ and r_4. The position of eye_3 in the 3D image space is controlled by θ_4, θ_5 and r_3. Strictly speaking, The last angle θ_6 is useless unless we want to project the 2D image down to 1D image. It is included for compatibility with the existing 3D graphics, to specify the *twist* of the 2D image.

There is a nice property of Euler angles to specify the directions of projection, *keeping the vertical directions.* Look at Figure 2.1. The three axes w, w_3 and z_3 are coplanar. That means no matter how we rotate eye_4, the world w-axis is always kept vertical in the 3D image space. It is upward when $-\frac{\pi}{2} \leq \theta_3 \leq \frac{\pi}{2}$ and downward otherwise. Similarly, no matter how we rotate eye_3, the world w-axis is again kept vertical in the 2D image space, as long as $\theta_6 = 0$.

A drawback of Euler angles is that at certain critical values the inverse Euler angle formula encounters numerical difficulties. Fortunately, in a visualization system we mostly calculate the direction cosine matrix from Euler angles. The inverse Euler angle formula is used only in a few cases which will be discussed in Section 3.1.

Bryant angles can also be extended into n-space. For example, the orientation of objects by Bryant angles in 4-space is expressed as:

$$\mathbf{p} = R^4_{zw}(\theta_1) R^4_{yw}(\theta_2) R^4_{xw}(\theta_3)\ R^3_{yz}(\theta_4) R^3_{xz}(\theta_5)\ R^2_{xy}(\theta_6)\, \mathbf{q}$$

They also have the critical values, though different from those of Euler angles. For example, since $(0, \ldots, 0)$ is not a critical value, Bryant angles are more suitable to represent small variations in orientation. But this is not important in visualization. Moreover, since Bryant angles do not keep the vertical direction, they are harder to be grasped, and so less suitable in the user interface.

2.1.3 Quaternions

The quaternions, an algebra invented by Sir William Rowan Hamilton in 1843, were originally defined as the angular relations (quotient) between pairs of vectors in 3-space [Ham69]. Algebraically the quaternions are an extension to the complex

numbers [PS74]. Their geometric interpretation is equivalent to that of Euler parameters or Cayley-Klein parameters in mechanics [Wit77, Gol80]. The quaternions have recently been advocated by a number of authors for 3D computer graphics [BM90, Mar85, Sho85, Sei90b]. Mathematically, any rotation in 4-space can be represented by a pair of quaternions [Por81]. However, their use in 4D computer graphics has not been exploited.

A quaternion $q \in \mathcal{H}$ is defined as[2]

$$q = q_x i + q_y j + q_z k + q_w$$

where q_x, q_y, q_z, q_w are real numbers and i, j, k are symbols obeying the following product rules:

$$i^2 = j^2 = k^2 = -1, \quad ij = k, \quad jk = i, \quad ki = j$$

For example, the quaternion product of p and q is

$$\begin{aligned} pq &= (p_x i + p_y j + p_z k + p_w)(q_x i + q_y j + q_z k + q_w) \\ &= (p_w q_x - p_z q_y + p_y q_z + p_x q_w) i \\ &+ (p_z q_x + p_w q_y - p_x q_z + p_y q_w) j \\ &+ (-p_y q_x + p_x q_y + p_w q_z + p_z q_w) k \\ &+ (-p_x q_x - p_y q_y - p_z q_z + p_w q_q) \end{aligned} \tag{2.3}$$

Note that the quaternion product is associative but not commutative. We will discuss the quaternions from the viewpoint of geometry instead of algebra. It is convenient to identify the symbols i, j, k with the base vectors $\mathbf{i}, \mathbf{j}, \mathbf{k}$, respectively, and the real number 1 with the fourth base vector $\mathbf{l}$ of $\mathcal{R}^4$. Then a quaternion q can be written in matrix form:

$$\mathbf{q} = \begin{pmatrix} q_x \\ q_y \\ q_z \\ q_w \end{pmatrix} = \begin{pmatrix} V_q \\ S_q \end{pmatrix}$$

[2]Note that our convention is nonstandard. We put the real part q_w as the last component instead of the first in order to be consistent with the rest of the thesis.

where $V_q = (q_x, q_y, q_z)^T$ denotes the *vector part* and $S_q = q_w$ denotes the *scalar part* of q. A quaternion q is said to be *real* if $V_q = \mathbf{0}$, and to be *pure* if $S_q = 0$. Consider V_p and V_q as vectors in $\mathcal{R}^3$. Their dot product $V_p \cdot V_q$ is written in matrix form $V_p^T V_q$. Their cross product $V_p \times V_q$ is written in matrix form $X_p V_q$ where

$$X_p = \begin{pmatrix} 0 & -p_z & p_y \\ p_z & 0 & -p_x \\ -p_y & p_x & 0 \end{pmatrix}$$

Note that X_p is skew-symmetric and that

$$X_p X_q = V_q V_p^T - V_p^T V_q I \tag{2.4}$$

Now Equation (2.3) can be written in matrix form as

$$L_p \mathbf{q} = \begin{pmatrix} S_p I + X_p & V_p \\ -V_p^T & S_p \end{pmatrix} \begin{pmatrix} V_q \\ S_q \end{pmatrix} \tag{2.5}$$

or equivalently,

$$R_q \mathbf{p} = \begin{pmatrix} S_q I - X_q & V_q \\ -V_q^T & S_q \end{pmatrix} \begin{pmatrix} V_p \\ S_p \end{pmatrix} \tag{2.6}$$

The quaternion product is thus interpreted as the linear transformation of a vector in 4-space. The choice of (2.5) or (2.6) depends on whether p and q are interpreted as coordinates or as transformations, although as quaternions they are not distinguishable. Note that $L_p \neq R_p$ because of $pq \neq qp$. Since $(pq)r = p(qr)$ we have $R_r L_p \mathbf{q} = L_p R_r \mathbf{q}$. That means L_p and R_r commute. This can also be verified from their definitions and (2.4). For example, the matrix form of $p_1 p_2 q r_2 r_1$ can be $L_{p_1} L_{p_2} R_{r_1} R_{r_2} \mathbf{q}$ or $L_{p_1} R_{r_1} L_{p_2} R_{r_2} \mathbf{q}$, etc. Also note that $L_{p_1 p_2} = L_{p_1} L_{p_2}$ and $R_{r_2 r_1} = R_{r_1} R_{r_2}$.

The *conjugate* of $q = q_x i + q_y j + q_z k + q_w$ is $-q_x i - q_y j - q_z k + q_w$ denoted by $\bar{q}$, or in matrix form, $\bar{\mathbf{q}} = (-V_q, S_q)^T$. The *norm* of q is the real number $q\bar{q} = q_x^2 + q_y^2 + q_z^2 + q_w^2 = \mathbf{q}^T \mathbf{q}$. The set of all unit norm quaternions is a 3-sphere in $\mathcal{R}^4$ denoted by S^3. Let $|V_q| = \sqrt{V_q^T V_q}$. The *axis* of q is defined as $A_q = \frac{V_q}{|V_q|}$ if $|V_q| \neq 0$. Otherwise A_q is

undetermined. The *angle* of q is defined as $\alpha_q = \arctan \frac{|V_q|}{S_q}$. The meanings of these quantities in 4D rotation will be explained shortly. Note that $A_{\bar{q}} = -A_q$ and $\alpha_{\bar{q}} = \alpha_q$ according to the definitions.

Lemma 2.1 Let p be a unit quaternion.

(a) Both L_p and R_p are rotations in R^4.

(b) If $p \neq 1$, for all $\mathbf{q} \in \mathcal{R}^4$, $L_p\mathbf{q} \neq \mathbf{q}$ and $R_p\mathbf{q} \neq \mathbf{q}$.

Proof: (a) It is straightforward to calculate that $L_p^T L_p = R_p^T R_p = I$ and $\det(L_p) = \det(R_p) = 1$. (b) The eigenvalues of L_p and R_p are $S_p \pm \sqrt{-V_p^T V_p}$ and are not real if p is not real. When $p = -1$, all vectors in $\mathcal{R}^4$ are mapped to their opposites. □

The lemma above can also be proved with the quaternion algebra as shown in [Por81].

The property (b) in Lemma 2.1 cannot hold for 3D rotations because the 3×3 rotation matrix always has a real eigenvalue 1. We discuss the 4D rotations defined by L_p and R_p. Let $\mathbf{m}$ and $\mathbf{n}$ be two nonzero orthogonal vectors in $\mathcal{R}^4$. We denote a rotation in the plane span($\mathbf{m}$,$\mathbf{n}$) by an angle α as $\mathrm{Rot}(\mathbf{m}, \mathbf{n}, \alpha)$. The direction of the rotation is defined by requiring $\mathrm{Rot}(\mathbf{m}, \mathbf{n}, \frac{\pi}{2})\mathbf{m} = \mathbf{n}$. For notational convenience, we may put a vector in $\mathcal{R}^3$, say A_p, in place of $\mathbf{m}$ or $\mathbf{n}$. In such cases, the vector is assumed to be embedded in $\mathcal{R}^4$ with its fourth component being zero.

Theorem 2.1 Let p be a unit nonreal quaternion. The rotations L_p and R_p can each be decomposed into two rotations, one in the plane of span(1, A_p) and one in the plane orthogonal to the former. More specifically,

$$\begin{aligned} L_p &= \mathrm{Rot}(1, A_p, \alpha_p)\mathrm{Rot}(\mathbf{m}, \mathbf{n}, \alpha_p) \\ R_p &= \mathrm{Rot}(1, A_p, \alpha_p)\mathrm{Rot}(\mathbf{m}, \mathbf{n}, -\alpha_p) \end{aligned}$$

where $\mathbf{m}, \mathbf{n}$ and A_p in that order form a right-handed orthonormal frame in $\mathcal{R}^3$, a subspace orthogonal to 1 in $\mathcal{R}^4$.

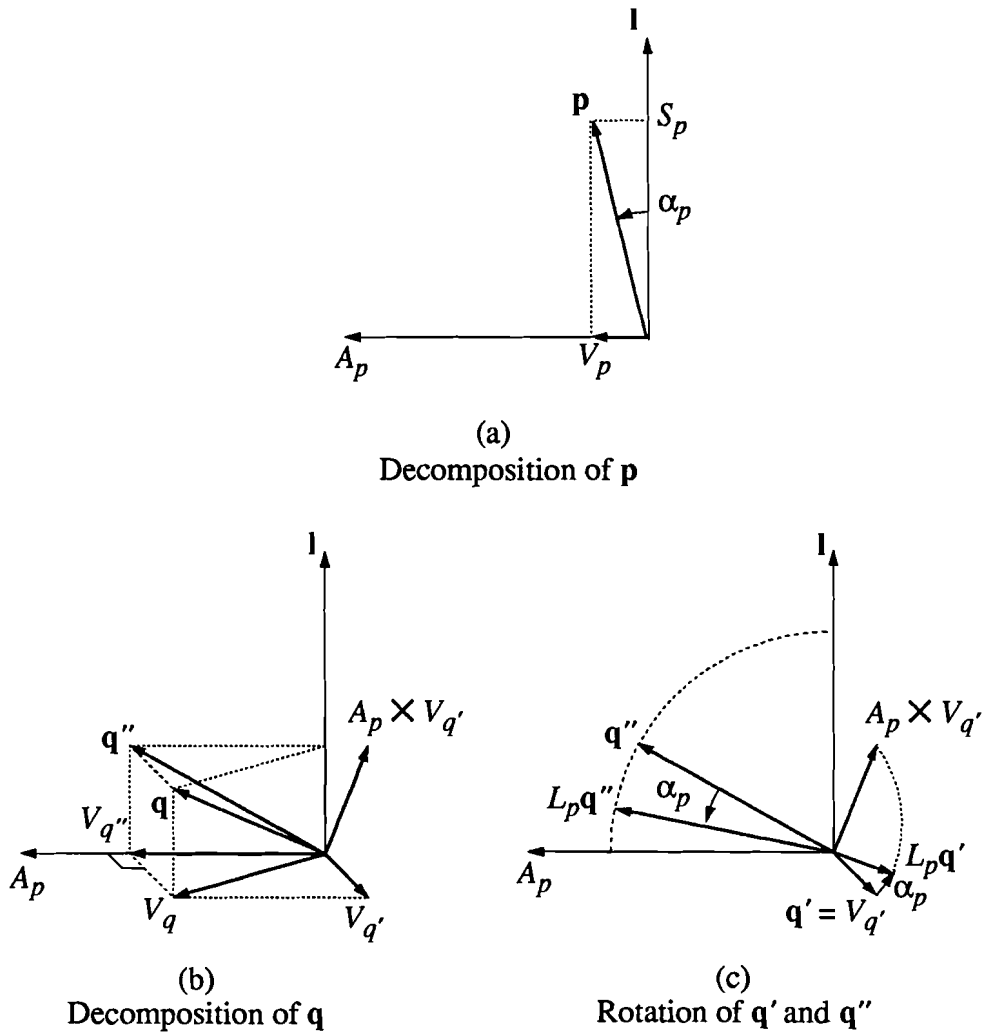

Figure 2.3 Rotation of quaternions

Proof: Since p is nonreal, V_p is a nonzero vector. It can be normalized into A_p; see Figure 2.3(a). Any $q \in \mathcal{H}$ can be decomposed into two parts $q = q' + q''$ such that

$$\begin{aligned} S_{q'} &= 0, & A_p \cdot V_{q'} = 0 \\ S_{q''} &= S_q, & A_p \times V_{q''} = 0 \end{aligned}$$

The vector parts of q' and q'' can be calculated from $V_{q''} = (A_p \cdot V_q)A_p$ and $V_{q'} = V_q - V_{q''}$; see Figure 2.3(b). Rotating the two parts separately we get

$$\begin{aligned} L_p\mathbf{q}' &= \begin{pmatrix} S_pI + X_p & V_p \\ -V_p^T & S_p \end{pmatrix} \begin{pmatrix} V_{q'} \\ 0 \end{pmatrix} \\ &= \begin{pmatrix} S_pV_{q'} + V_p \times V_{q'} \\ 0 \end{pmatrix} \\ &= \begin{pmatrix} \cos\alpha_p V_{q'} + \sin\alpha_p A_p \times V_{q'} \\ 0 \end{pmatrix}. \\ &= \mathrm{Rot}(V_{q'}, A_p \times V_{q'}, \alpha_p)\mathbf{q}' \\ &= \mathrm{Rot}(\mathbf{m}, \mathbf{n}, \alpha_p)\mathbf{q}' \\ L_p\mathbf{q}'' &= \begin{pmatrix} S_pI + X_p & V_p \\ -V_p^T & S_p \end{pmatrix} \begin{pmatrix} V_{q''} \\ S_{q''} \end{pmatrix} \\ &= \begin{pmatrix} S_pV_{q''} + S_{q''}V_p \\ -V_p^TV_{q''} + S_pS_{q''} \end{pmatrix} \\ &= \begin{pmatrix} (\cos\alpha_p|V_{q''}| + \sin\alpha_p S_{q''})A_p \\ -\sin\alpha_p|V_{q''}| + \cos\alpha_p S_{q''} \end{pmatrix} \\ &= \mathrm{Rot}(\mathbf{l}, A_p, \alpha_p)\mathbf{q}'' \end{aligned}$$

The rotations of $\mathbf{q}'$ and $\mathbf{q}''$ are shown in Figure 2.3(c). Combining them yields

$$\begin{aligned} L_p\mathbf{q} &= L_p\mathbf{q}' + L_p\mathbf{q}'' \\ &= \mathrm{Rot}(\mathbf{m}, \mathbf{n}, \alpha_p)\mathbf{q}' + \mathrm{Rot}(\mathbf{l}, A_p, \alpha_p)\mathbf{q}'' \\ &= \mathrm{Rot}(\mathbf{l}, A_p, \alpha_p)\mathrm{Rot}(\mathbf{m}, \mathbf{n}, \alpha_p)\mathbf{q}' + \mathrm{Rot}(\mathbf{l}, A_p, \alpha_p)\mathrm{Rot}(\mathbf{m}, \mathbf{n}, \alpha_p)\mathbf{q}'' \\ &= \mathrm{Rot}(\mathbf{l}, A_p, \alpha_p)\mathrm{Rot}(\mathbf{m}, \mathbf{n}, \alpha_p)\mathbf{q} \end{aligned}$$

The case of R_p can be proved similarly. □

This theorem described the rotation caused by *one* quaternion. That the two subrotations are orthogonal and have the same magnitude is not surprising because the quaternion contains no more information than a vector and a scalar. Since $V_{\bar{p}} = -V_p$ and $\alpha_{\bar{p}} = \alpha_p$, any combination of $L_p, L_{\bar{p}}, R_p, R_{\bar{p}}$ are expected to be a simple rotation. They can be worked out using Theorem 2.1 with little effort.

First, the rotations L_p and $L_{\bar{p}}$ cancel each other. This is obvious because $L_{\bar{p}} = L_p^T = L_p^{-1}$. It can also be verified by computing the quaternion product. When p is a unit quaternion, its conjugate equals to its inverse, i.e. $p\bar{p} = \bar{p}p = 1$.

Second, $L_pL_p = \text{Rot}(\mathbf{l}, A_p, 2\alpha_p)\text{Rot}(\mathbf{m}, \mathbf{n}, 2\alpha_p)$. This is due to the fact that the two subrotations are in orthogonal planes and therefore commute. The result can be extended to arbitrary $\lambda \in \mathcal{R}$:

$$(L_p)^\lambda = L_{p^\lambda} = \text{Rot}(\mathbf{l}, A_p, \lambda\alpha_p)\text{Rot}(\mathbf{m}, \mathbf{n}, \lambda\alpha_p) \tag{2.7}$$

This property is similar to that of complex numbers.

Third, the famous representation of 3D rotations is

$$L_pR_{\bar{p}} = \text{Rot}(\mathbf{m}, \mathbf{n}, 2\alpha_p) \tag{2.8}$$

This is because the rotation in span(1, V_p) is canceled while the rotation in the orthogonal plane is doubled.

Finally, another rotation that is less known is given by

$$L_pR_p = \text{Rot}(\mathbf{l}, A_p, 2\alpha_p) \tag{2.9}$$

Although this rotation is restricted to a plane, it is not in $\mathcal{R}^3$ and so has not been mentioned in most papers dealing with quaternions. We will show in the next subsection that it is useful for a user friendly interface of a 4D visualization system.

While a single quaternion can only represent certain simple 4D rotations and every 3D rotation, a pair of unit quaternions (p, r) is powerful enough to represent every rotation in $\mathcal{R}^4$. More specifically, we define the product of two pairs of quaternions

as a map $\cdot : \mathcal{H}^2 \times \mathcal{H}^2 \to \mathcal{H}^2$ defined by

$$(p_1, r_1) \cdot (p_2, r_2) = (p_1 p_2, r_1 r_2)$$

It is clear that the product so defined is associative. The identity in $\mathcal{H}^2$ is naturally $(1, 1)$. The inverse is defined as

$$(p, r)^{-1} = (p^{-1}, r^{-1})$$

Since we concentrate on unit quaternions, we define

$$\overline{(p, r)} = (\bar{p}, \bar{r})$$

Thus $(\mathcal{H}^2, \cdot)$ is a group and $((S^3)^2, \cdot)$ is a subgroup of it.

Theorem 2.2 The map $\rho_1 : S^3 \times S^3 \to SO(4)$ defined by $\rho_1(p, r) = L_p R_{\bar{r}}$ is a group surjection with kernel $(1, 1), (-1, -1)$.

A proof can be found in [Por81]. Briefly, that it is a group homomorphism is shown by

$$\begin{aligned} \rho_1((p_1, r_1) \cdot (p_2, r_2)) &= \rho_1(p_1 p_2, r_1 r_2) \\ &= L_{p_1 p_2} R_{\overline{r_1 r_2}} \\ &= L_{p_1} L_{p_2} R_{\bar{r}_1} R_{\bar{r}_2} \\ &= \rho_1(p_1, r_1) \rho_1(p_2, r_2) \end{aligned}$$

For any 4D rotation T, suppose T rotates the base vector $\mathbf{l}$ to $\mathbf{s}$, i.e. $T\mathbf{l} = \mathbf{s}$. Then $L_{\bar{s}} T$ will fix $\mathbf{l}$ and so it is a 3D rotation, expressible as $L_r R_{\bar{r}}$. Therefore $T = L_s L_r R_{\bar{r}}$. Writing $sr = p$, the rotation is then $L_p R_{\bar{r}}$.

We call the quaternion pair (p, r) with the map $\rho_1(p, r) = L_p R_{\bar{r}}$ *the first form* of 4D rotation representation. In this form, two or more 4D rotations can be efficiently combined into one. Hence it is suitable for the internal representation in a system.

The discussion above suggests that (s, r) can be used also for representations of 4D rotations. The map is defined as $\rho_2(s, r) = L_s L_r R_{\bar{r}}$. We called it *the second*

form which will be shown handy in conversion between quaternions and matrices or between quaternions and Euler angles. However, to make ρ_2 preserve the group structure, the product $\cdot : \mathcal{H}^2 \times \mathcal{H}^2 \rightarrow \mathcal{H}^2$ has to be redefined as[3]

$$(s_1, r_2) \cdot (s_2, r_2) = (s_1 r_1 s_2 \bar{r_1}, r_1 r_2) \tag{2.10}$$

It can be shown that the product so defined is also associative. Again $(1, 1)$ is the identity in $\mathcal{H}^2$. From it we define

$$(s, r)^{-1} = (r^{-1} s^{-1} r, r^{-1}) \tag{2.11}$$

$$\overline{(s, r)} = (\bar{r} \bar{s} r, \bar{r}) \tag{2.12}$$

So, under the redefined product the $(\mathcal{H}^2, \cdot)$ is also a group. The computation is more involved than for the first form. Therefore, it is better to convert to the first form before rotations are combined.

Since $L_p R_p$ has a clean interpretation, we want to define *the third form* (u, v) with the map $\rho_3(u, v) = L_u R_u L_v R_{\bar{v}}$. This representation is equivalent to the first and second forms, shown from the relations

$$\begin{aligned} p &= uv = sr \\ r &= \bar{u} v \\ s &= u^2 = p\bar{r} \\ u &= s^{1/2} = (p\bar{r})^{1/2} \\ v &= s^{1/2} r = (p\bar{r})^{1/2} r \end{aligned} \tag{2.13}$$

The product should also be redefined to make ρ_3 a homomorphism. This is omitted here. Like the second form, the third form is not well suited for rotation combination. But it is ideal for a user interface as will be discussed next.

The three forms are summarized in the following table.

[3]Strictly the different product definitions should be denoted by $\cdot_1$ and $\cdot_2$. We distinguish them instead through context.

Form	1st.	2nd.	3rd.
quaternion pair	(p, r)	(s, r)	(u, v)
quaternion expression	$pq\bar{r}$	$srq\bar{r}$	$uvq\bar{v}u$
matrix expression	$L_p R_{\bar{r}}$	$L_s L_r R_{\bar{r}}$	$L_u R_u L_v R_{\bar{v}}$
suitable for	rotation combination	conversion	user interface

2.1.4 Directions of Projection by Quaternions

Recall that the world coordinate system, the 3D image coordinate system and the 2D image coordinate system are related by a 4D rotation. By means of quaternions, the user interface can be designed in a fashion different from that using Euler angles.

Suppose the 4D rotation is expressed in the third form (u, v). Let $\mathbf{q}$ be a vector expressed in the world coordinate system, and $\mathbf{q}'$ be the same vector in the body-fixed coordinate system. They are related by

$$\mathbf{q} = L_u R_u L_v R_{\bar{v}} \mathbf{q}'$$

The 2D image space is defined as span of the body-fixed x- and y-axes. eye_4 is on the body-fixed w-axis and eye_3 is on the body-fixed z-axis. Since $L_v R_{\bar{v}}$ is a 3D rotation, it represents the relation between the 3D image and 2D image coordinate system. In other words, let

$$\mathbf{q} = L_u R_u \mathbf{q}''$$

Then a map $\varphi : \mathcal{R}^4 \to \mathcal{R}^3$ will project $\mathbf{q}''$ into the 3D image coordinate system as $\varphi(\mathbf{q}'')$. When eye_4 is at infinity, i.e. $r_4 = 0$, the projection is simply the drop of the fourth coordinate, called the natural projection π, written as $\pi(\mathbf{q}'') = V_{q''}$. Now the image of $\mathbf{q}$ in the 3D image coordinate system is

$$\pi(\mathbf{q}'') = V_{q''} = \pi(L_{\bar{u}} R_{\bar{u}} \mathbf{q}) = V_{\bar{u}q\bar{u}}$$

Defining $t = \bar{u}$ to simply the notation it becomes V_{tqt}.

We investigate how the axes of the world coordinate system move in the 3D image space in response to the rotation $L_t R_t$. According to Theorem 2.1 the rotation matrix

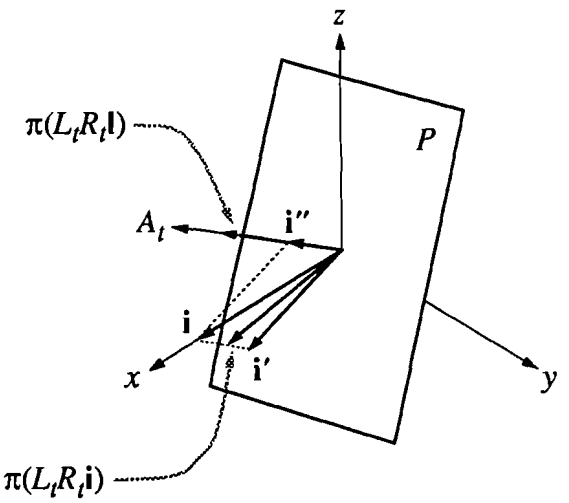

Figure 2.4 The projections of the world base vectors $\mathbf{i}$ and $\mathbf{l}$ in the 3D image space

is $L_tR_t = \mathrm{Rot}(\mathbf{l}, V_t, 2\alpha_t)$. When $t = 1$, $L_tR_t = I$, the world x-, y- and z-axes coincide with those of the 3D image axes. The world w-axis is invisible. When $t \neq 1$ and V_t is nonzero, it determines a direction A_t in the 3D image space. The plane orthogonal to A_t is denoted by P; see Figure 2.4. The image of world w-axis can be calculated from the base vector $\mathbf{l}$ as

$$\begin{aligned} \pi(L_tR_t\mathbf{l}) &= \pi(\mathrm{Rot}(\mathbf{l}, V_t, 2\alpha_t)\mathbf{l}) \\ &= \pi((\sin 2\alpha_t)A_t + (\cos 2\alpha_t)\mathbf{l}) \\ &= (\sin 2\alpha_t)A_t \end{aligned}$$

It is in the direction of A_t. That offers a convenient way to put the world w-axis in any desired direction in the 3D image space. The image of the world x-axis needs a little more effort. First the base vector $\mathbf{i}$ is decomposed into $\mathbf{i} = \mathbf{i}' + \mathbf{i}''$ with $\mathbf{i}'$ in the plane P and $\mathbf{i}''$ parallel to A_t. Then

$$\begin{aligned} \pi(L_tR_t\mathbf{i}) &= \pi(L_tR_t(\mathbf{i}' + \mathbf{i}'')) \\ &= \pi(\mathrm{Rot}(\mathbf{l}, V_t, 2\alpha_t)(\mathbf{i}' + \mathbf{i}'')) \\ &= \mathbf{i}' + (\cos 2\alpha_t)\mathbf{i}'' \end{aligned}$$

When $\alpha_t = 0$, the image of the world x-axis coincides with the 3D image x-axis. As α_t increases, the image of the world x-axis moves toward the plane P and falls into P when α_t reaches $\frac{\pi}{4}$. Meanwhile, the length of the image of the world w-axis increases from zero to its maximum. The behavior of the world y- and z-axes under this rotation is similar to that of the x-axis.

An interface based on this idea lets the user control two independent quaternions u and v. The positions of the axes of the world coordinate system in the 3D image space are controlled by u. The positions of the axes of the 3D image coordinate system in the 2D image space are controlled by v. Note that the rotations are explained in terms of the axes and angles of u and v. These concepts are comprehensible without the knowledge of quaternions.

2.1.5 Relations between Euler Angles and Quaternions

We have shown that 4D rotations can be represented by Euler angles, quaternions and matrices. User interfaces employing Euler angles and quaternions may coexist in a 4D visualization system. The internal representations are usually matrices, but they could be quaternions as advocated by some authors [Mar85, Sei90b]. It is therefore necessary to find ways for converting among these representations. For 3D rotations the conversion problem has been discussed in [She78, Sho85, Sei90b].

Given a pair of unit quaternions (p, r), the corresponding matrix form is simply obtained from $L_p R_{\bar{r}}$. Conversely, suppose that $A = (a_{ij}) = L_p R_{\bar{r}}$ is given, to find the corresponding (p, r) directly from A is possible but tedious. It is better found using the second form (s, r). Define $M = (m_{ij}) = L_r R_{\bar{r}}$. Since M is a 3D rotation matrix we have $m_{i4} = m_{4i} = 0$ for $i = 1, 2, 3$ and $m_{44} = 1$. Now from $A = L_s L_r R_{\bar{r}} = L_s M$ we immediately get $\mathbf{s} = (a_{41}, a_{42}, a_{43}, a_{44})^T$. Then M can be obtained by $L_{\bar{s}} A$. From the upper left 3×3 submatrix of M, r can be calculated by the methods described in [She78, Sho85, Sei90b]. Finally, $(p, r) = (sr, r)$.

Given Euler angles $\theta = (\theta_1, \ldots, \theta_6)$ a matrix can be calculated. Then the quaternions can be obtained by the above method. But it is more convenient to find the

quaternions directly from θ. Recall that the rotation by Euler angles is defined as

$$R^4_{xy}(\theta_1)R^4_{yz}(\theta_2)R^4_{zw}(\theta_3)\ R^3_{xy}(\theta_4)R^3_{yz}(\theta_5)\ R^2_{xy}(\theta_6)\,\mathbf{q}$$

According to Theorem 2.1, it can be written in quaternion form

$$r_1 r_2 r_3 r_4 r_5 r_6 q \bar{r_6}\bar{r_5}\bar{r_4}r_3\bar{r_2}\bar{r_1}$$

where

$$\begin{aligned}
\mathbf{r_1} &= (0,0,\sin\frac{\theta_1}{2},\cos\frac{\theta_1}{2})^T \\
\mathbf{r_2} &= (\sin\frac{\theta_2}{2},0,0,\cos\frac{\theta_2}{2})^T \\
\mathbf{r_3} &= (0,0,-\sin\frac{\theta_3}{2},\cos\frac{\theta_3}{2})^T \\
\mathbf{r_4} &= (0,0,\sin\frac{\theta_4}{2},\cos\frac{\theta_4}{2})^T \\
\mathbf{r_5} &= (\sin\frac{\theta_5}{2},0,0,\cos\frac{\theta_5}{2})^T \\
\mathbf{r_6} &= (0,0,\sin\frac{\theta_6}{2},\cos\frac{\theta_6}{2})^T
\end{aligned}$$

Rewrite it into $srq\bar{r}$ of the second form. Then r and s can be calculated by $r = r_1 r_2 \bar{r_3} r_4 r_5 r_6$ and $s = r_1 r_2 r_3^2 \bar{r_2}\bar{r_1}$. Note that $r_{123} = r_1 r_2 \bar{r_3}$ and $r_{456} = r_4 r_5 r_6$ have the same pattern.

$$\mathbf{r}_{123} = \begin{pmatrix} \sin\frac{\theta_2}{2}\cos\frac{-\theta_1+\theta_3}{2} \\ -\sin\frac{\theta_2}{2}\sin\frac{-\theta_1+\theta_3}{2} \\ \cos\frac{\theta_2}{2}\sin\frac{\theta_1+\theta_3}{2} \\ \cos\frac{\theta_2}{2}\cos\frac{\theta_1+\theta_3}{2} \end{pmatrix} \tag{2.14}$$

$$\mathbf{r}_{456} = \begin{pmatrix} \sin\frac{\theta_5}{2}\cos\frac{-\theta_4+\theta_6}{2} \\ -\sin\frac{\theta_5}{2}\sin\frac{-\theta_4+\theta_6}{2} \\ \cos\frac{\theta_5}{2}\sin\frac{\theta_4+\theta_6}{2} \\ \cos\frac{\theta_5}{2}\cos\frac{\theta_4+\theta_6}{2} \end{pmatrix} \tag{2.15}$$

$$\mathbf{r} = \begin{pmatrix} \sin\frac{\theta_2}{2}\cos\frac{\theta_5}{2}\cos\frac{-\theta_1+\theta_3+\theta_4+\theta_6}{2} + \cos\frac{\theta_2}{2}\sin\frac{\theta_5}{2}\cos\frac{\theta_1+\theta_3+\theta_4-\theta_6}{2} \\ -\sin\frac{\theta_2}{2}\cos\frac{\theta_5}{2}\sin\frac{-\theta_1+\theta_3+\theta_4+\theta_6}{2} + \cos\frac{\theta_2}{2}\sin\frac{\theta_5}{2}\sin\frac{\theta_1+\theta_3+\theta_4-\theta_6}{2} \\ \cos\frac{\theta_2}{2}\cos\frac{\theta_5}{2}\sin\frac{\theta_1+\theta_3+\theta_4+\theta_6}{2} + \sin\frac{\theta_2}{2}\sin\frac{\theta_5}{2}\sin\frac{-\theta_1+\theta_3+\theta_4-\theta_6}{2} \\ \cos\frac{\theta_2}{2}\cos\frac{\theta_5}{2}\cos\frac{\theta_1+\theta_3+\theta_4+\theta_6}{2} - \sin\frac{\theta_2}{2}\cos\frac{\theta_5}{2}\cos\frac{-\theta_1+\theta_3+\theta_4-\theta_6}{2} \end{pmatrix} \tag{2.16}$$

$$\mathbf{s} = \begin{pmatrix} -\sin\theta_1\sin\theta_2\sin\theta_3 \\ \cos\theta_1\sin\theta_2\sin\theta_3 \\ -\cos\theta_2\sin\theta_3 \\ \cos\theta_3 \end{pmatrix} \tag{2.17}$$

Conversely, given a pair of quaternions (p, r), to find the corresponding Euler angles we first get the second form (s, r). This step is useful because s is unrelated to $\theta_4, \theta_5, \theta_6$. Hence $\theta_1, \theta_2, \theta_3$ can be found from $\mathbf{s}$ by (2.17) and (2.2). Then r_{123} is formed by (2.14), from which r_{456} can be obtained by $\overline{r_{123}}r$. Finally, $\theta_4, \theta_5, \theta_6$ are found from $\mathbf{r}_{456}$ by (2.15).

2.1.6 Animation by Quaternions

In computer animation the position and orientation of objects and eyes are specified as functions of time t. They can be given explicitly in symbolic form, but more often are given as a set of function values at discrete time instances, and the intermediate values are obtained through interpolation. The interpolation of orientation can be done by the interpolation of matrices, of Euler angles, or of quaternions. It has been argued that quaternions are best for this purpose [Sho85, Sei90b].

Given two unit quaternions $q_1, q_2 \in S^3$, the *great arc* is defined as the curve on the intersection of S^3 and the plane passing through the two points and the origin. The spherical linear interpolation, abbreviated as slerp, from q_1 to q_2 with parameter $\lambda \in [0, 1]$ is defined as[Sho85]:

$$\text{slerp}(q_1, q_2; \lambda) = q_1(\bar{q}_1 q_2)^\lambda$$

or equivalently,

$$\text{slerp}(q_1, q_2; \lambda) = \frac{\sin(1-\lambda)\theta}{\sin\theta} q_1 + \frac{\sin\lambda\theta}{\sin\theta} q_2$$

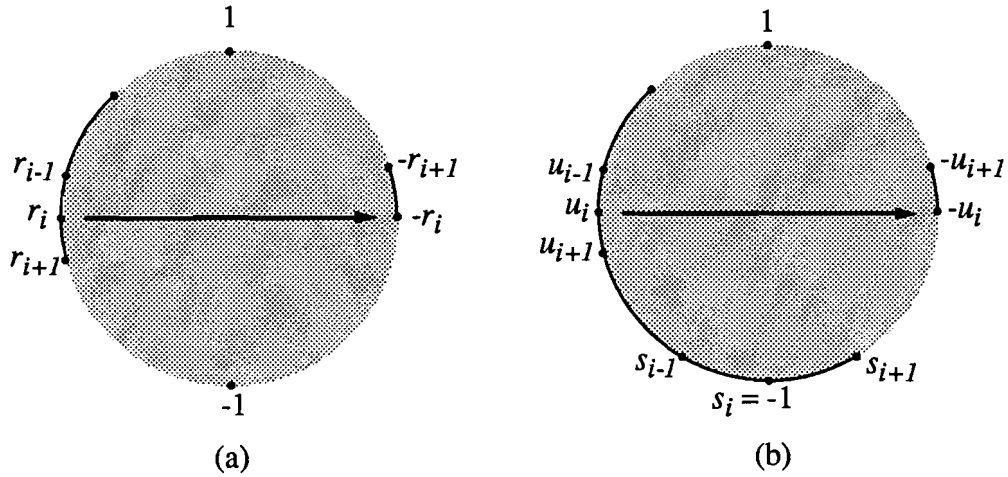

Figure 2.5 Ambiguities in conversion to quaternions

where $\mathbf{q}_1 \cdot \mathbf{q}_2 = \cos\theta$. Based on slerp, the spherical splines and Bézier curves and be constructed. The higher order continuity makes the animation smoother.

The method can be extended to the interpolation of 4D rotations. Given a pair of quaternions, no matter in which form, the two components are independent from each other. Therefore, they can be interpolated separately. However, there are several problems worth discussing.

First, it is well known that the quaternion representation of 3D rotation is not unique. The same statement holds for 4D rotation. Mathematically speaking, the maps from $(S^3)^2$ to $SO(4)$ are not bijections. It can be proven that ρ_1 and ρ_2 are 2-to-1 maps while ρ_3 is a 4-to-1 map. To verify this, it is easy to check that

$$\begin{aligned}
\rho_1(p,r) &= \rho_1(-p,-r) \\
\rho_2(s,r) &= \rho_2(s,-r) \\
\rho_3(u,v) &= \rho_3(-u,-v) = \rho_3(-u,v) = \rho_3(u,-v)
\end{aligned}$$

The geometric interpretation is that, by Theorem 2.1, the double appearances of r, u and v or their conjugates cause the angles to be doubled. In practice, we need to find

the sources of ambiguity in the conversion from the matrices to quaternions. When converting matrix A to the second form (s, r) the quaternion s is uniquely determined by the fourth column of A. In the calculation r from $M = L_{\bar{s}}A$ there is a square root of a real number. A suggestion by [She78] is to choose $S_r > 0$. However, such a choice will cause a jump during animation when $S_r \approx 0$; see Figure 2.5(a). A practical solution by [Sho85] is to make sure that r at adjacent time instances are close to each other. Converting the second form to the first form encounters no trouble at all. A simple quaternion product $p = sr$ completes the task. The conversion from the second form to the third form involves a *square root of a quaternion.* From $u^2 = s$ we get $u = \pm s^{1/2}$. Theoretically we could choose the positive root, called the *principal value* in [Ham69], as we did in (2.13). But this also causes a jump during animation when $s \approx -1$ as shown in Figure 2.5(b). Worse than that, the quaternion square root near -1 is ill behaved because the axis is indeterminate. Consequently, in practice the conversion to the third form should be avoided. The animation by interpolating (u, v) can be done only if the third form is directly available from, say, the user interface based on it.

Another question is whether the interpolations by the three forms are equivalent. More precisely, let $\rho_1(p_i, r_i) = \rho_2(s_i, r_i) = \rho_3(u_i, v_i)$ for $i = 1, 2$, and let $p_{12}(\lambda) = \text{slerp}(p_1, p_2; \lambda)$, etc., we want to know whether $\rho_1(p_{12}(\lambda), r_{12}(\lambda)) = \rho_2(s_{12}(\lambda), r_{12}(\lambda)) = \rho_3(u_{12}(\lambda), v_{12}(\lambda))$ for all $\lambda \in [0, 1]$. The answer is no because in general $\text{slerp}(s_1, s_2, \lambda)\text{slerp}(r_1, r_2, \lambda) \neq \text{slerp}(s_1 r_1, s_2 r_2, \lambda)$, and so on. However, the difference is slight if the successive quaternion pairs in the sequence are reasonably close.

2.2 Silhouettes and Envelopes

The concept of silhouette is important in 4D visualization for the following reasons: First, from the silhouette surfaces or curves we can infer the shape and some geometric properties of the 3-surfaces or 2-surfaces in 4-space. Second, a 3-surface in 4-space is better displayed by the silhouette surfaces and other 2-surfaces on it than by a

shaded volume. So, we wish to generate the silhouette explicitly. Third, the silhouette surfaces determine how the visibility of points on a hypersurface in 4-space changes.

The concept of *envelope* is closely related to the silhouette, and turns out to be very useful in explaining the 3D image of the hypersurfaces in 4-space.

2.2.1 Silhouette Points

We use the term m-surface $\mathcal{M}$ as an m-dimensional manifold in n-dimensional Euclidean space $\mathcal{R}^n$ $(n \geq m)$. Particularly, a 1-surface is also called a curve; an n-surface in $\mathcal{R}^{n+1}$ is also called a hypersurface; a 2-surface in $\mathcal{R}^3$ is also called a surface.

We define *silhouette points* of an m-surface $\mathcal{M} \subset \mathcal{R}^n$ with respect to a projection $\varphi_n^l : \mathcal{R}^n \rightarrow \mathcal{R}^l$ $(m \leq l < n)$ as those regular points $\mathbf{p}$ on $\mathcal{M}$ such that the tangent space of $\mathcal{M}$ at $\mathbf{p}$ reduces its dimension under the projection φ_n^l. In the following we always assume that:

1. $\varphi_n^l : \mathcal{R}^n \rightarrow \mathcal{R}^l$ is a projection with the centers on the x_n-,..., x_{l+1}-axes at finite or infinite distances from the origin, specified by their reciprocal distances $r_n, \ldots, r_{l+1}$. Thus

$$\varphi_n^l(x_1, \ldots, x_n) = \frac{(x_1, \ldots, x_l)}{1 - \sum_{i=l+1}^{n} r_i x_i}$$

   When all the centers of projection are at infinity, φ_n^l is an orthographic projection and is denoted π_n^l.

2. $\mathbf{p}$ is a regular point on an m-surface $\mathcal{M}$. The point $\mathbf{p}$ is not mapped to infinity under the projection. $\mathbf{t}_1, \ldots, \mathbf{t}_m$ are m linearly independent tangent vectors of $\mathcal{M}$ at $\mathbf{p}$. $\mathbf{n}_1, \ldots, \mathbf{n}_{n-m}$ are $n - m$ linearly independent normal vectors of $\mathcal{M}$ at $\mathbf{p}$.

3. $\mathbf{r}_{l+1}, \ldots, \mathbf{r}_n$ are vectors in the direction from $\mathbf{p}$ to the centers of projection on the x_{l+1}-,..., x_n-axis, respectively. If the center of projection on the x_k-axis $(l+1 \leq k \leq n)$ is at infinity, then $\mathbf{r}_k$ is the k-th base vector $\mathbf{e}_k$. Otherwise,

$\mathbf{r}_k = \frac{1}{r_k}\mathbf{e}_k - \mathbf{p}$. Let $\mathbf{m}_1, \ldots, \mathbf{m}_l$ be l linearly independent vectors in the subspace orthogonal to span$(\mathbf{r}_{l+1}, \ldots, \mathbf{r}_n)$.

A differentiable mapping $\varphi\colon \mathcal{R}^n \to \mathcal{R}^l$ will induce two linear transformations (see e.g. [AM63]):

$$\begin{aligned} \varphi_*(\text{tangent vector to } \gamma) &= \text{tangent vector to } \varphi \circ \gamma \\ \varphi^*(\text{normal vector to } f) &= \text{normal vector to } f \circ \varphi \end{aligned}$$

where $\gamma\colon \mathcal{R} \to \mathcal{R}^n$ is a curve in the domain of φ, and $f\colon \mathcal{R}^l \to \mathcal{R}$ is a function on the range of φ. The matrix forms of the two linear transformations φ_* and φ^* are the Jacobian matrix $J(\varphi)$ and its transpose $J(\varphi)^T$, respectively.

Lemma 2.2 The null space of φ_*, viz. $\varphi_*^{-1}(\mathbf{0})$, is span$(\mathbf{r}_{l+1}, \ldots, \mathbf{r}_n)$.

Proof: Let $\mu = 1 - \sum_{i=l+1}^{n} r_i x_i$. The Jacobian matrix $J(\varphi)$ is

$$J(\varphi) = \begin{pmatrix} \frac{1}{\mu} & 0 & \cdots & 0 & \frac{r_{l+1}x_1}{\mu^2} & \cdots & \frac{r_n x_1}{\mu^2} \\ 0 & \frac{1}{\mu} & \cdots & 0 & \frac{r_{l+1}x_2}{\mu^2} & \cdots & \frac{r_n x_2}{\mu^2} \\ \vdots & & \ddots & \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & \frac{1}{\mu} & \frac{r_{l+1}x_l}{\mu^2} & \cdots & \frac{r_n x_l}{\mu^2} \end{pmatrix}$$

It is easy to verify that $J(\varphi)\mathbf{r}_k = \mathbf{0}$ for $k = l+1, \ldots, n$. The null space of $J(\varphi)$ must be span$(\mathbf{r}_{l+1}, \ldots, \mathbf{r}_n)$ if we can prove that its dimension is at most $n - l$.

Let $(\mathbf{e}_1, \ldots, \mathbf{e}_n)$ be a vector base of $\mathcal{R}^n$. We know that any nonzero vector in span$(\mathbf{e}_1, \ldots, \mathbf{e}_l)$ will not be projected into a zero vector. If the null space has dimension greater than $n - l$, its intersection with span$(\mathbf{e}_1, \ldots, \mathbf{e}_l)$ must have dimension greater than 1. Then any nonzero vector in the intersection contradicts the definition of the two sets. □

Theorem 2.3 $\mathbf{p}$ is a silhouette point with respect to φ_n^l if and only if $\mathbf{t}_1, \ldots, \mathbf{t}_m, \mathbf{r}_{l+1}, \ldots, \mathbf{r}_n$ are linearly dependent.

Proof: First we assume that $\mathbf{t}_1, \ldots, \mathbf{t}_m, \mathbf{r}_{l+1}, \ldots, \mathbf{r}_n$ are linearly dependent, and so $\alpha_1\mathbf{t}_1 + \cdots + \alpha_m\mathbf{t}_m + \beta_{l+1}\mathbf{r}_{l+1} + \cdots + \beta_n\mathbf{r}_n = \mathbf{0}$ where at least one of $\alpha_1, \ldots, \alpha_m$ is not zero. Applying the linear transformation φ_* and using lemma 2.2, we get $\alpha_1\varphi_*(\mathbf{t}_1) + \cdots + \alpha_m\varphi_*(\mathbf{t}_m) = \mathbf{0}$. That means the tangent space of $\mathcal{M}$ reduces its dimension under the projection, and so $\mathbf{p}$ is a silhouette point. Conversely, assume that $\mathbf{p}$ is a silhouette point. Then the vectors $\varphi_*(\mathbf{t}_1), \ldots, \varphi_*(\mathbf{t}_m)$ are in the tangent space of dimension less than m. So we have $\alpha_1\varphi_*(\mathbf{t}_1) + \cdots + \alpha_m\varphi_*(\mathbf{t}_m) = \mathbf{0}$ with $\alpha_1 \cdots \alpha_m \neq 0$. Again by Lemma 2.2 we get $\alpha_1\mathbf{t}_1 + \cdots + \alpha_m\mathbf{t}_m + \mathbf{r} = \mathbf{0}$ where $\mathbf{r}$ is any vector in $\text{span}(\mathbf{r}_{l+1}, \ldots, \mathbf{r}_n)$, i.e. $\mathbf{t}_1, \ldots, \mathbf{t}_m, \mathbf{r}_{l+1}, \ldots, \mathbf{r}_n$ are linearly dependent. □

Corollary 2.1 When $l = m$, the condition in Theorem 2.3 is equivalent to that $\mathbf{m}_1, \ldots, \mathbf{m}_l, \mathbf{n}_1, \ldots, \mathbf{n}_{n-m}$ are linearly dependent.

Proof: It is sufficient to prove that $\mathbf{t}_1, \ldots, \mathbf{t}_m, \mathbf{r}_{l+1}, \ldots, \mathbf{r}_n$ are linearly independent if and only if $\mathbf{m}_1, \ldots, \mathbf{m}_l, \mathbf{n}_1, \ldots, \mathbf{n}_{n-m}$ are linearly independent. Assuming that $\mathbf{t}_1, \ldots, \mathbf{t}_m, \mathbf{r}_{l+1}, \ldots, \mathbf{r}_n$ are linearly independent, they form a base of $\mathcal{R}^n$. Then $\mathbf{n}_1, \ldots, \mathbf{n}_{n-m}$ are linear combinations of $\mathbf{r}_{l+1}, \ldots, \mathbf{r}_n$, and $\mathbf{m}_1, \ldots, \mathbf{m}_l$ are linear combinations of $\mathbf{t}_1, \ldots, \mathbf{t}_m$. Therefore, $\mathbf{m}_1, \ldots, \mathbf{m}_l, \mathbf{n}_1, \ldots, \mathbf{n}_{n-m}$ must be linearly independent. The converse direction is symmetric. □

For example, a regular point $\mathbf{p}$ on a 2-surface is a silhouette point with respect to φ_4^3 if $\mathbf{t}_1, \mathbf{t}_2, \mathbf{r}_4$ are linearly dependent, which is equivalent to $\mathbf{n}_1 \cdot \mathbf{r}_4 = 0$ and $\mathbf{n}_2 \cdot \mathbf{r}_4 = 0$. Adjoined to the two equations defining the 2-surface, the solution is usually a 0-dimensional set. At such a silhouette point, the normal of the projected 2-surface cannot be determined, and this has to be taken into account if shading is to be added to the projection φ_3^2. A regular point $\mathbf{p}$ on a 2-surface is a silhouette point with respect to φ_4^2 if $\det(\mathbf{t}_1, \mathbf{t}_2, \mathbf{r}_3, \mathbf{r}_4) = 0$, or equivalently, $\det(\mathbf{n}_1, \mathbf{n}_2, \mathbf{m}_1, \mathbf{m}_2) = 0$. The solution is usually a 1-dimensional set called the *silhouette curve* of the 2-surface. A regular point $\mathbf{p}$ on a hypersurface in $\mathcal{R}^4$ is a silhouette point with respect to φ_4^3 if $\det(\mathbf{t}_1, \mathbf{t}_2, \mathbf{t}_3, \mathbf{r}_4) = 0$, or equivalently, $\det(\mathbf{n}_1, \mathbf{m}_1, \mathbf{m}_2, \mathbf{m}_3) = 0$. Adjoined to the equation defining the hypersurface, the solution is usually a 2-surface called the *silhouette surface* of the hypersurface.

2.2.2 Envelopes

Given a family of hypersurfaces in $\mathcal{R}^n$, then a hypersurface $\mathcal{M}$ is called the *envelope* of the family of hypersurfaces if: (a) At every point $\mathcal{M}$ is tangent to some hypersurfaces of the family; and (b) $\mathcal{M}$ touches all the hypersurfaces of the family.

A family of hypersurfaces in $\mathcal{R}^n$ can be expressed in implicit form as

$$\begin{aligned} f_1(x_1,\ldots,x_n,u_1,\ldots,u_m) &= 0 \\ f_2(u_1,\ldots,u_m) &= 0 \\ &\vdots \\ f_m(u_1,\ldots,u_m) &= 0 \end{aligned} \tag{2.18}$$

or in parametric form as

$$\begin{aligned} x_1 &= s_1(t_1,\ldots,t_{n-1},v) \\ &\vdots \\ x_n &= s_n(t_1,\ldots,t_{n-1},v) \end{aligned} \tag{2.19}$$

where $u_1,\ldots,u_m$ and v are the parameters of the family, and $t_1,\ldots,t_{n-1}$ are the parameters of the hypersurfaces. The envelope of a family of hypersurfaces can be computed by the envelope theorem [Spi79]:

Theorem 2.4

(a) Suppose that a family of hypersurfaces is defined by (2.18). Then every point of the envelope satisfies (2.18) adjoined by:

$$\det(\nabla_u f_1,\ldots,\nabla_u f_m) = 0 \tag{2.20}$$

where

$$\nabla_u f = (\frac{\partial f}{\partial u_1},\ldots,\frac{\partial f}{\partial u_m})^T$$

(b) Suppose that a family of hypersurfaces is defined by (2.19). Then every point of the envelope satisfies (2.19) adjoined by:

$$\det(\nabla s_1,\ldots,\nabla s_n) = 0$$

Notice that the hypersurface obtained by eliminating the parameters $u_1, \ldots, u_m$ from the (2.18) and (2.20) is called the *discriminant hypersurface* that consists of the envelope and the locus of all singular points on the hypersurfaces of the family. Because singular points cannot be silhouette points by definition, from Corollary 2.1 and Theorem 2.4(a) we have:

Corollary 2.2 The envelope of a family of hypersurfaces in $\mathcal{R}^l$ is the image of the silhouette of a l-surface in $\mathcal{R}^{l+m}$ with respect to the orthographic projection π^l_{l+m}.

The *offset curves* or *offset surfaces* can be formulated by the envelope theorem. The envelope can be traced numerically in $\mathcal{R}^l$ or in $\mathcal{R}^{l+m}$. The former corresponds to tracing the silhouette in image space and the latter to tracing the silhouette in object space. The latter is more stable since the projection will cause apparent cusps, self-intersections of the silhouette. This issue will be discussed in Chapter 5.

2.2.3 The Silhouette Surface of a Hypersurface in 4-space

Let $\mathcal{M}$ be a hypersurface in $\mathcal{R}^4$ in parametric form given by

$$(x_1, \ldots, x_4) = s(t_1, t_2, t_3)$$

and $\varphi : \mathcal{R}^4 \rightarrow \mathcal{R}^3$ be a projection. Assuming that $\mathbf{p} = s(t_1, t_2, t_3)$ is a regular point, the Jacobian matrix $J(s)$ has rank 3, representing three linearly independent tangent vectors. By definition $\mathbf{p}$ is a silhouette point with respect to φ if and only if $\det(J(\varphi)J(s)) = 0$. On the other hand, the hypersurface can also be considered as a family of 2-surfaces by setting one of the parameters, say t_3, as the parameter u of the family, written as $s(t_1, t_2, u)$. After projection, it becomes a family of 2-surfaces in $\mathcal{R}^3$, $\varphi \circ s(t_1, t_2, u)$. By Theorem 2.4(b), the envelope of the family of the projected 2-surfaces satisfies $\det(J(\varphi \circ s)) = \det(J(\varphi)J(s)) = 0$. Thus we get the following corollary:

Corollary 2.3 The silhouette surface (in image space) of a hypersurface with respect to a projection from $\mathcal{R}^4$ to $\mathcal{R}^3$ can be obtained from the envelope of the family of the projected isoparametric 2-surfaces on the hypersurface.

The corollary above is useful in understanding the 3D image of hypersurfaces. Using it we can draw qualitatively the silhouette surface of a hypersurface in image space without calculation, even without knowing the position of the projection center. Examples will be given in Chapter 3.

2.2.4 The Normal of a Projected 2-Surface

In 3D graphics, illumination and shading of surfaces is computed from the surface normal and the light directions. Since a 2-surface in 4-space has two independent normal directions, the generalization of 3D illumination models to 4-space is more complicated than merely illuminating the 3D image of the 2-surface after the first projection step by standard methods. Furthermore, the critical problem in 4D visualization is to gain insight into the properties of the first projection step, from 4-space to 3-space. Therefore, we obtain maximum information about the shape of the 3D image when shading in 3-space, and can concentrate on understanding the first projection step.

One way to find the normal of the projected 2-surface is to calculate it from the equation representing the projected 2-surface. Another way is to calculate the normal directly from the tangent or normal plane of the 2-surface before projection. The latter is usually more efficient because the construction of the equation of the projected 2-surface could be expensive [Hof90].

If the 2-surface is in parametric form and so the tangent vectors are directly available, it is easy to calculate the normal vector $\bar{\mathbf{n}}$ of the projected 2-surface by first transforming the tangent vectors and then applying the cross product:

$$\bar{\mathbf{n}} = \varphi^3_{4*}(\mathbf{t_1}) \times \varphi^3_{4*}(\mathbf{t_2})$$

If the 2-surface is in implicit form and so the normal vectors are directly available, we can first use the cross product in $\mathcal{R}^4$ to find the tangent vectors and then follow the same procedure as that for parametric 2-surfaces.

Let $\mathbf{i},\mathbf{j},\mathbf{k},\mathbf{l}$ be the base vectors of $\mathcal{R}^4$, and $\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$ be three vectors where $\mathbf{a} = (a_x, a_y, a_z, a_w)^T$ and so on. The cross product $\otimes$ is defined as:

$$\otimes(\mathbf{a},\mathbf{b},\mathbf{c}) = \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} & \mathbf{l} \\ a_x & a_y & a_z & a_w \\ b_x & b_y & b_z & b_w \\ c_x & c_y & c_z & c_w \end{vmatrix}$$

From linear algebra we know that $\otimes(\mathbf{a},\mathbf{b},\mathbf{c})$ is orthogonal to the subspace span$(\mathbf{a},\mathbf{b},\mathbf{c})$ if $\mathbf{a},\mathbf{b},\mathbf{c}$ are linearly independent. From two normal vectors $\mathbf{n_1}$ and $\mathbf{n_2}$ we can find two tangent vectors $\mathbf{t_1}$ and $\mathbf{t_2}$, and vice versa, as follows. Given $\mathbf{n_1}$ and $\mathbf{n_2}$, choose any two vectors $\mathbf{a}$ and $\mathbf{b}$ such that $\mathbf{n_1},\mathbf{n_2},\mathbf{a},\mathbf{b}$ are linearly independent. A base of the tangent space is then

$$\mathbf{t_1} = \otimes(\mathbf{n_1},\mathbf{n_2},\mathbf{a}), \qquad \mathbf{t_2} = \otimes(\mathbf{n_1},\mathbf{n_2},\mathbf{b})$$

A more efficient way is to find the normal of the project 2-surface directly from the two normal vectors without calculating the tangent vectors first.

Theorem 2.5 Suppose that $\mathbf{p}$ is a regular point on a 2-surface, and is a nonsilhouette point with respect to φ_4^3. Let $\mathbf{n} = \alpha\mathbf{n_1} + \beta\mathbf{n_2}$ satisfy $\mathbf{n}\cdot\mathbf{r_1} = 0$. Then $\pi_4^3(\mathbf{n})$ is the normal vector of the projected 2-surface at the point $\varphi_4^3(\mathbf{p})$.

Proof: Assume that $\bar{\mathbf{n}}$ is the normal vector of the projected 2-surface at the point $\varphi_4^3(\mathbf{p})$. From the Jacobian matrix $J(\varphi_4^3)$ we know that $\pi_4^3(\varphi_4^{3*}(\bar{\mathbf{n}}))$ is parallel to $\bar{\mathbf{n}}$. It suffices to show that $\mathbf{n}$ as defined above is parallel to $\varphi_4^{3*}(\bar{\mathbf{n}})$ in $\mathcal{R}^4$. The vector $\bar{\mathbf{n}}$ satisfies

$$\bar{\mathbf{n}}\cdot\varphi_{4*}^3(\mathbf{t_1}) = 0, \qquad \bar{\mathbf{n}}\cdot\varphi_{4*}^3(\mathbf{t_2}) = 0, \qquad \bar{\mathbf{n}}\cdot\varphi_{4*}^3(\mathbf{r_1}) = 0$$

The last equation is actually satisfied when $\bar{\mathbf{n}}$ is any vector in $\mathcal{R}^3$. These three equations are equivalent to

$$\varphi_4^{3*}(\bar{\mathbf{n}})\cdot\mathbf{t_1} = 0, \qquad \varphi_4^{3*}(\bar{\mathbf{n}})\cdot\mathbf{t_2} = 0, \qquad \varphi_4^{3*}(\bar{\mathbf{n}})\cdot\mathbf{r_1} = 0$$

Since $\mathbf{p}$ is a nonsilhouette point, $\mathbf{t_1}, \mathbf{t_2}, \mathbf{r_1}$ are linearly independent by Theorem 2.3. Hence $\mathbf{n}$ is parallel to $\varphi_4^{3*}(\bar{\mathbf{n}})$. □

Theorem 2.5 can also be applied to calculate the normal vector of a projected silhouette surface of a hypersurface. The silhouette surface S of a hypersurface f with respect to φ_4^3 is a 2-surface in 4-space:

$$f(x_1, \ldots, x_4) = 0 \tag{2.21}$$

$$\nabla f(x_1, \ldots, x_4) \cdot \mathbf{r} = 0 \tag{2.22}$$

As a 2-surface, S has its own silhouette points with respect to φ_4^3. According to Corollary 2.1, they should satisfy:

$$\mathbf{n_1} \cdot \mathbf{r} = \nabla f(x_1, \ldots, x_4) \cdot \mathbf{r} = 0 \tag{2.23}$$

$$\mathbf{n_2} \cdot \mathbf{r} = \nabla(\nabla f(x_1, \ldots, x_4) \cdot \mathbf{r}) = 0 \tag{2.24}$$

Note that (2.23) is redundant. Therefore, (2.24) determines on S a curve C where the projected S is singular. Except for the points on C, by Theorem 2.5, the normal to the projected silhouette surface is $\pi_4^3(\alpha\mathbf{n_1} + \beta\mathbf{n_2}) = \pi_4^3(\alpha\mathbf{n_1})$ because β is obviously zero. We state this result as the following corollary:

Corollary 2.4 Let S be the silhouette surface of a hypersurface $f = 0$ with respect to φ_4^3. Suppose that $\mathbf{p}$ is a regular point on S, but not a point satisfying (2.24). Then $\pi_4^3(\nabla f)$ is the normal vector of $\varphi_4^3(S)$ at the point $\varphi_4^3(\mathbf{p})$.

2.3 Visibility

Theoretically we can project objects in arbitrary n-space into 2-space and produce their 2D images. But the mapping is not one-to-one. A point in the image space can have infinitely many points as its preimage. To resolve this ambiguity, the concept of visibility has to be introduced.

The visibility associated with the projection from 3-space to 2-space is directly taken from our experience of seeing the real world. For the higher dimensional space,

however, visibility is only a mathematical definition. There are many ways to extend the visibility into high dimensional space. To choose a suitable one, several factors have to be considered:

1. The definition of visibility can be extended from φ_3^2 to arbitrary φ_n^{n-1}. Taking a point in the image space, its preimage points lie in a line in the object space. The visibility information is obtained from the order of those points in the line. For the projection φ_n^{n-2}, however, the preimage points lie in a plane, with no natural total order. The visibility has to be considered step by step, first with respect to φ_n^{n-1}, then to φ_{n-1}^{n-2}, and so on.

2. The dimension of the potentially visible objects with respect to φ_n^{n-1} could be restricted to $n-1$ and n as advocated by [BS82]. But in some applications, lower dimensional objects such as curves and 2-surfaces need to be displayed as well. On the other hand, an object of dimension n will definitely reduce dimension during projection. Its image is indistinguishable from the image of its bounding $(n-1)$-surface. Therefore, the dimensions of potentially visible objects range from 0 up to $n-1$.

3. The dimension of the potentially hiding objects with respect to φ_n^{n-1} could be $n-1$ and n. Again, a point hidden by an object of dimension n must also be hidden by its boundary $(n-1)$-surfaces. Therefore, the dimension of potentially hiding objects is limited to $n-1$.

4. It will be helpful if we can display transparent objects, which are actually images produced by previous projections. Therefore, we choose the definition of *quantitative visibility* [SSS74, EC90, Wal90]. The visibility of a point is quantified as the number of hiding objects intersecting the line segment from this point to the center of projection.

In the following the term object can also refer to the image of an object produced by the previous projections. Consider the projection φ_n^{n-1} with the center $\mathbf{c}$. A point

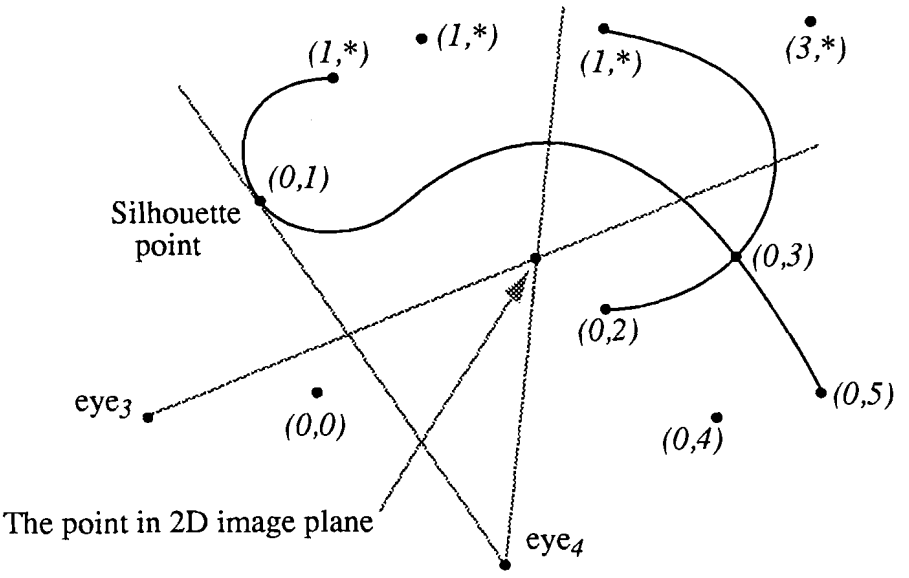

Figure 2.6 Visibility

p on an object of any dimension less than n has a *sight number* which is the number of intersections of the ray from **p** to **c** (excluding **p**) with objects of dimension $n-1$. The total visibility information of the point is expressed as a list of sight numbers.

Visualization of 4-space involves only two projections, For the first projection we remove those points with nonzero sight numbers. For the second projection we use the sight numbers to show transparency. If all the objects in $\mathcal{R}^4$ are 2-surfaces or curves, then every object is visible with respect to the first projection. Hypersurfaces in $\mathcal{R}^4$ will make points on other objects invisible, but after the first projection, their images are totally transparent. The shapes of the hypersurfaces are shown mainly by their silhouette surfaces, boundary surfaces, intersection surfaces, and isoparametric surfaces, that are all considered as objects of dimension 2.

Suppose that all objects in $\mathcal{R}^4$ are contained in a hypercube that contains neither eye_4 nor eye_3. A point in the 2D image space, together with eye_4 and eye_3, determines a plane. The plane will intersect in general with a 2-surface in a point and with a hypersurface in a curve. All of the intersection points and curves will be projected to the same point in the 2D image space. Fig. 2.6 shows an example. Each point

is associated with a pair (a, b), where a is the sight number with respect to the first projection, b is the sight number with respect to the second projection if $a = 0$. When $a > 0$, the point will be removed after the first projection, and so b is undefined as denoted by an asterisk.

The explanation above suggests a ray tracing method for 4D visualization. Instead of tracing a line segment, we can construct a plane and calculate its intersection with objects. Instead of finding the roots of a single equation, we need to solve a system of equations, which could be overdetermined in the case of curve-plane intersection in $\mathcal{R}^4$. Presently, this method is not suitable for interactive display.

Other methods are based on the following theorem that is an extension to those presented in [EC90, Wal90].

Theorem 2.6 In the 3D image space, the silhouette, self-intersection, and boundary surfaces of a hypersurface partition the hypersurface into regions such that all points in the same region have the same sight number.

The silhouette, self-intersection, and boundary surfaces are called the *active surfaces*. Consider a point moving on a hypersurface. Its sight number changes when its image, in 3D image space, passes through an active surface. The change of sight number can be calculated by several methods, such as the differential method described in [Wal90], or the propagation rules described in [EC90]. Once we find the sight number of a point, it is propagated into the whole region without any further calculation. The implementation will be discussed in Chapter 4.

3. VISUAL PHENOMENA AND THEIR MEANING

One of the major tasks of 4D visualization is to interpret the 3D images of objects in 4-space. Through the visual phenomena explained in this chapter, we want to show that the 3D images of the objects in 4-space can expose some geometric properties in 4-space. We will explain how to choose viewing positions, how to interpret the 3D images of hypersurfaces, and how to observe the curvature of a hypersurface in 4-space.

3.1 Interpretation of Some Viewing Positions

3.1.1 Viewing Positions That Keep 2D Image Invariant

Let L be the line in $\mathcal{R}^4$ passing through the two centers of projection eye_4 and eye_3. If we fix the 2D image space and let eye_4 and eye_3 move on the line L, the basic 2D image, i.e. the image without considering visibility, will not change. This phenomenon can be observed by setting $\theta_4 = \theta_5 = \theta_6 = 0$ and rotating in (z, w)-plane, i.e. changing θ_3. Meanwhile, the distances of eye_4 and eye_3 have to be modified to keep them on the fixed line L, according to:

$$\begin{aligned} r_3 &= a\cos\theta_3 + b\sin\theta_3 \\ r_4 &= -a\sin\theta_3 + b\cos\theta_3 \end{aligned}$$

Particularly, if θ_3 changes from 0 to $\pm\frac{\pi}{2}$, correspondingly (r_4, r_3) changes from (b, a) to $(\mp a, \pm b)$, so that the positions of eye_4 and eye_3 are *exchanged.*

For nonzero $\theta_4, \theta_5, \theta_6$, the positions of eye_4 and eye_3 can be exchanged as follows. Suppose the first picture is obtained by $\theta = (\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5, \alpha_6)$ and $r = (b, a)$. To

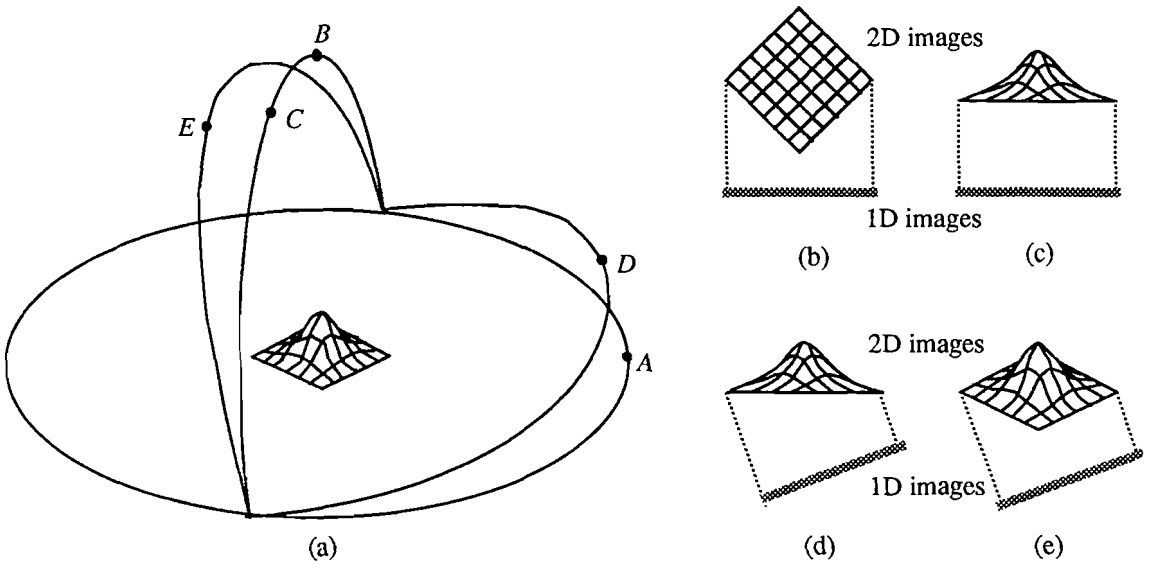

Figure 3.1 A square plate projected into a line. (a)viewing positions. (b)eye$_3$ at B, eye$_2$ at A. (c)eye$_3$ at A, eye$_2$ at B. (d)eye$_3$ at A, eye$_2$ at C. (e)eye$_3$ at D, eye$_2$ at E

obtain the second picture we can set the rotation matrix as:

$$R^4_{xy}(\alpha_1)R^4_{yz}(\alpha_2)R^4_{zw}(\alpha_3)R^3_{xy}(\alpha_4)R^3_{yz}(\alpha_5)R^2_{xy}(\alpha_6)R^4_{zw}(\pm\frac{\pi}{2})$$

and set $r = (\mp a, \pm b)$. We can use the inverse Euler angle formula (2.1) and (2.2) to find the Euler angles corresponding to the above rotation matrix.

3.1.2 Viewing Positions with Special Effects

Consider a bracket of nonuniform material density that is divided into cubic elements by a grid of planes x = constant, y = constant, z = constant; where the w-value represents the density. The "planes" in the grid are actually 2-surfaces in 4-space. In order to facilitate understanding the different situations, Figure 3.1 shows a companion example of a surface in 3-space that is projected into a line. In this companion example, a square plate of nonuniform material density is shown, where the z-value represents density.

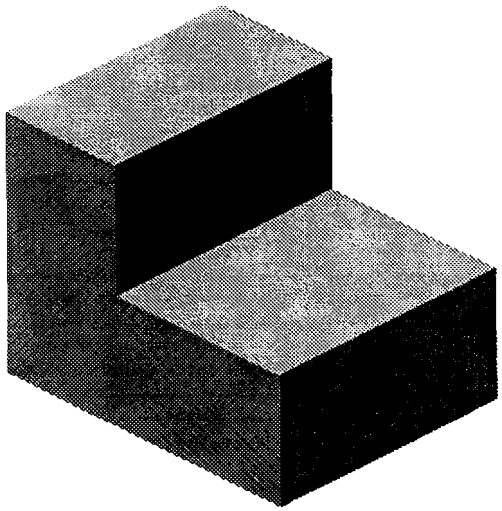

Figure 3.2 Bracket viewed from $\theta = (0, 0, 0, 45, 60, 0)$ degrees

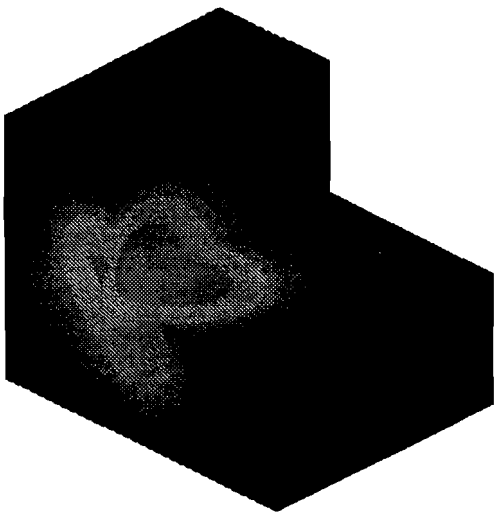

Figure 3.3 Bracket viewed from $\theta = (0, 0, 0, 45, 60, 0)$ but shaded by color scale representing w-values

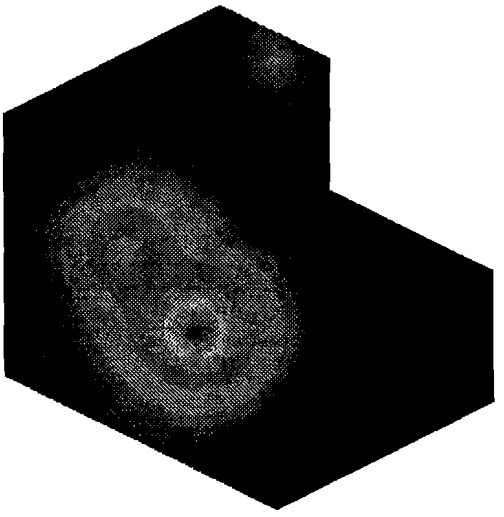

Figure 3.4 Bracket viewed from $\theta = (45, 60, 90, 0, 0, 0)$; Positions of eye_4 and eye_3 are exchanged

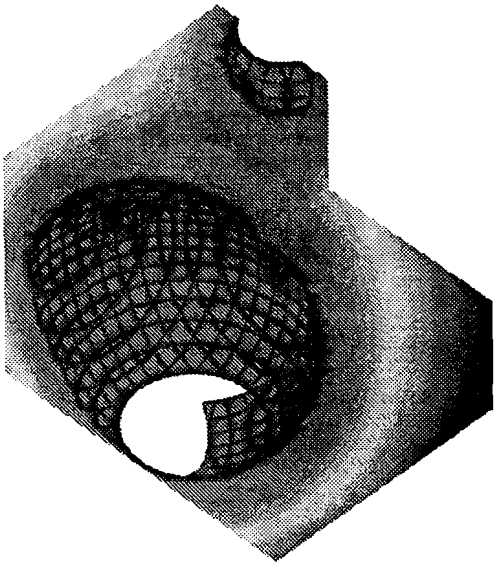

Figure 3.5 Bracket cut by z-clipping plane to display isosurface w = constant

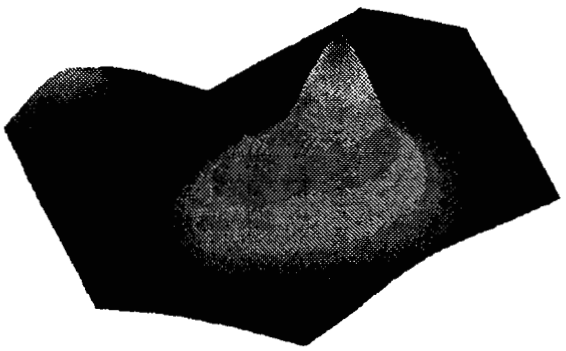

Figure 3.6 The 3D image viewed from a different eye_3 direction: $\theta = (45, 60, 90, -75, 45, 0)$

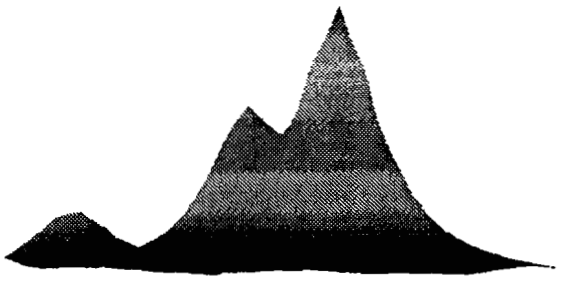

Figure 3.7 The 3D image viewed from a different eye_3 direction: $\theta = (45, 60, 90, -75, 90, 0)$

1. By ignoring the w-values, a normal 3D image is obtained: Set $r = (0, a)$ and $\theta = (0, 0, 0, \alpha_1, \alpha_2, 0)$. By varying α_1, α_2 and a we obtain the usual 2D pictures of the 3D object (Figure 3.2). Note that it is still possible to see the w-value on the boundary of the 3D object using a color scale (Figure 3.3).[1] Compare Figure 3.1(b). Moreover, the same effect can be achieved by setting $\theta = (\alpha_1, \alpha_2, 0, 0, 0, 0)$ and $r = (0, a)$. This is possible because $R^4_{zw}(0)$ is the identity matrix and $R^4_{xy}(\alpha_1)R^4_{yz}(\alpha_2) = R^3_{xy}(\alpha_1)R^3_{yz}(\alpha_2)$.

2. Assume that $\theta = (\alpha_1, \alpha_2, \pm\frac{\pi}{2}, 0, 0, 0)$ and $r = (\mp a, 0)$. Then we obtain the same basic 2D image as before because of the exchanged eye_4 and eye_3 positions. However, the w-value displayed through color is the maximum, or the minimum,[2] of all w-values on a line through the bracket from eye_4. The situation is analogous to looking at a mountain from atop (Figure 3.4). The base of the mountain is the projected bracket shape, and the height is the w-value. Compare Figure 3.1(c).

3. Once we have generated the picture of (2), an isosurface (w=constant) can be obtained by z-clipping (Figure 3.5). The isosurface is not displayed explicitly but implied by the curves that are the intersection of the isosurface with the grid surfaces. These intersection curves would have the same shape if they were displayed in Figure 3.3.

4. The 3D image implied by the picture of (2) can also be seen from other directions, with $\theta = (\alpha_1, \alpha_2, \pm\frac{\pi}{2}, \beta_1, \beta_2, 0)$ and $r = (\mp a, b)$. Using the analogy of the mountain, we fix the base and height of the mountain (corresponding to the first projection) but view it from a different direction. Figures 3.6 and 3.7 show the cases $\beta_2 = \frac{\pi}{4}$ and $\beta_2 = \frac{\pi}{2}$. Compare Figure 3.1(d).

[1] Here the pictures are shown in monochrome to comply with the university thesis format. For color pictures see [HZ91].

[2] The maximum and minimum are approximate if $a \neq 0$.

5. Because θ_6 only rotates the 2D image on the screen, we have the most general case with $\theta = (\alpha_1, \alpha_2, \alpha_3, \beta_1, \beta_2, 0)$ and $r = (a, b)$. When α_3 is varied from 0 to $\pm\frac{\pi}{2}$, intermediate w-values are seen. The situation is analogous to viewing the mountain from different perspectives, except that now the shape of the mountain changes as well, because of the 4D motion. In Figure 3.8 only three surfaces in the grid, namely $x = \text{constant}_1$, $y = \text{constant}_2$, and $z = \text{constant}_3$ are displyed. Compare Figure 3.1(e).

3.2 Interpretation of the 3D Images of Hypersurfaces

3.2.1 Procedural Construction of the 3D Image of a Hypersurface

To understand the computer-generated pictures of hypersurfaces, we describe how to construct in one's mind the 3D image of a hypersurface. The procedure is explained with an example: the hypersurface in $\mathcal{R}^4$ defined by $x^2 + y^2 + z - w^2 = 0$. One of its parametric forms is

$$\begin{aligned} x &= r \sin s \\ y &= r \cos s \\ z &= t^2 - r^2 \\ w &= t \end{aligned}$$

1. Select a point as the origin of the 3D image space. From the origin draw four incoplanar vectors as the 3D images of the four base vectors of the world coordinate system. As explained in Section 2.1, the upward direction is the projected world w-axis.

2. Fix one of the parameters of the hypersurface and draw a set of isosurfaces. This is essentially the same way as we draw a 2-surface in 3-space except that there are four base vectors and they no longer form an orthogonal system. In Figure 3.9 three isosurfaces, $w = -0.5$, $w = 0$, and $w = 0.5$ are drawn.

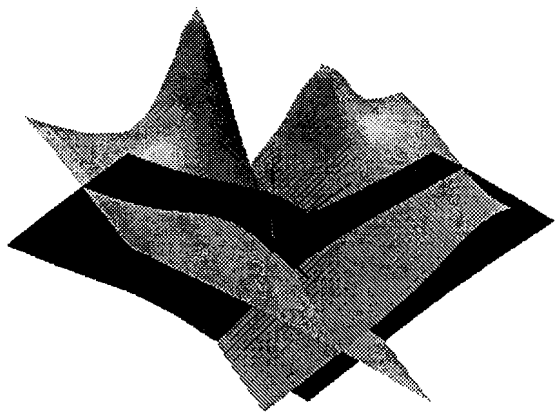

Figure 3.8 Bracket viewed from $\theta = (45, 45, 45, 105, 90, 0)$. Only three surfaces in the grid with constant x, y and z values are displayed

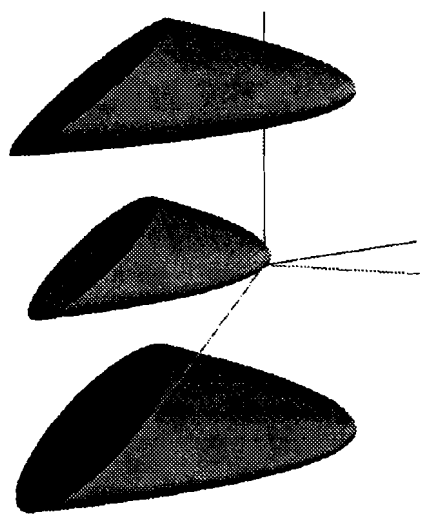

Figure 3.9 Three isosurfaces on the hypersurface $x^2 + y^2 + z - w^2 = 0$

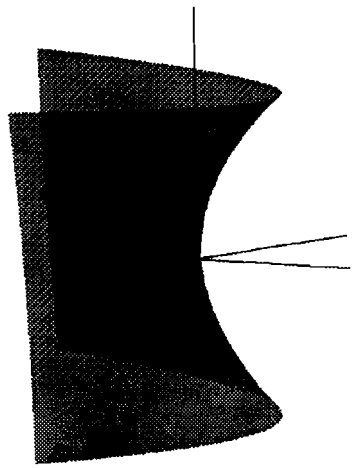

Figure 3.10 The silhouette surface of the hypersurface

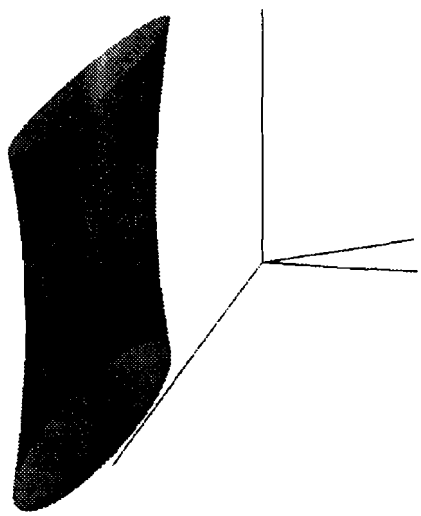

Figure 3.11 The boundary surface $z + 0.5 = 0$ of the hypersurface

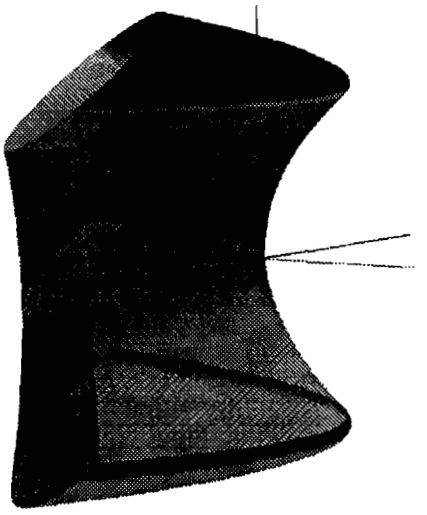

Figure 3.12 The hypersurface viewed from $\theta = (45, 105, 78, 90, 81, 0)$ with visibility determination

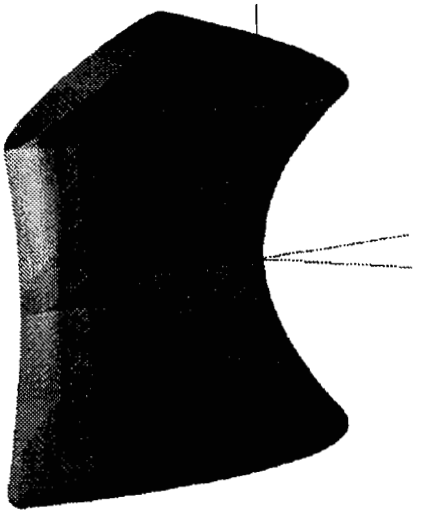

Figure 3.13 The hypersurface viewed from $\theta = (45, 75, 102, 90, 81, 0)$ with visibility determination

3. According to Corollary 2.3, the silhouette surface in the image space is the envelope of the family of the isosurfaces. So, we can construct the the silhouette surface, as shown in Figure 3.10, following the shape of the isosurfaces.

4. Add boundary surfaces, if necessary. In Figure 3.11 a boundary surface $z = -0.5$ is shown. The isosurfaces $w = -0.5$ and $w = 0.5$ also serve as boundary surfaces. The boundary surfaces and silhouette surface enclose a volume in 3D image space. This volume is the 3D image of the portion of the hypersurface we are displaying.

5. Trim the 2-surfaces to show the visibility. In Figure 3.9, as the w value changes from -0.5 to 0.5, the isosurface sweeps a volume. Within the volume some points are swept twice by the isosurface at different w values. The ambiguity can be resolved by making a choice at those points, either the first sweep hides the second sweep, or vice versa. The result of the two choices are shown in Figures 3.12 and 3.13. They correspond to two different viewing directions.

3.2.2 Understanding by analogy

Another way to understand the 3D image is to create an analogous example in 3-space and project it into 2-space. Figure 3.14(a) and (b) show a surface $x^2 + y - z^2 = 0$ from two different views. The silhouette curve, isocurves, and boundary curves are the counterparts to the silhouette surface, isosurfaces, and boundary surfaces in Figure 3.12 and 3.13.

Let us examine another hypersurface $x^2 + y^2 + z^2 - w^2 - 1 = 0$. Figure 3.15 shows its silhouette surface, five isosurfaces with constant w values of $-1, -0.5, 0, 0.5, 1$. Two of them also serve as the boundary surfaces. A curve on the hypersurface is also shown, with the parametric form

$$\begin{aligned} x &= \sqrt{t^2+1}\cos^2 t \\ y &= \sqrt{t^2+1}\cos t \sin t \\ z &= \sqrt{t^2+1}\sin t \end{aligned}$$

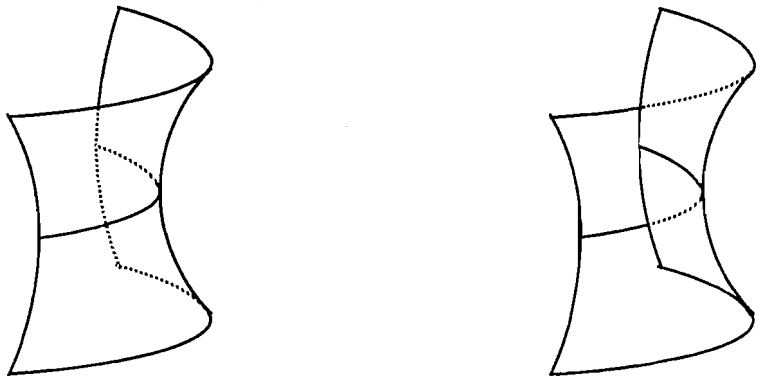

Figure 3.14 Surface $x^2 + y - z^2 = 0$ with two different views

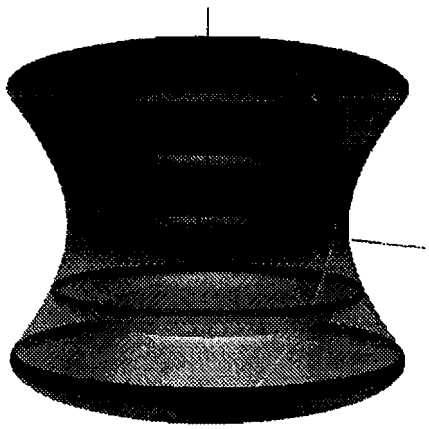

Figure 3.15 Hypersurface $x^2 + y^2 + z^2 - w^2 - 1 = 0$ viewed from $\theta = (-165, 45, 75, -165, 84, 0)$

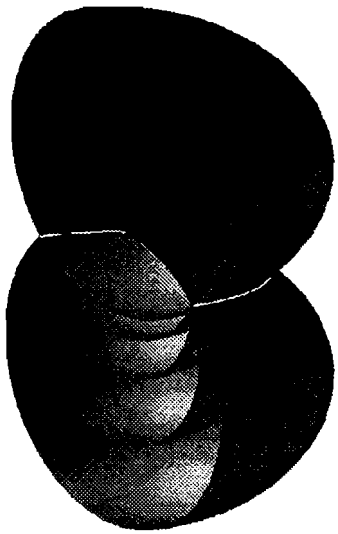

Figure 3.16 The dimension reduction of the silhouette surface in 3D image space

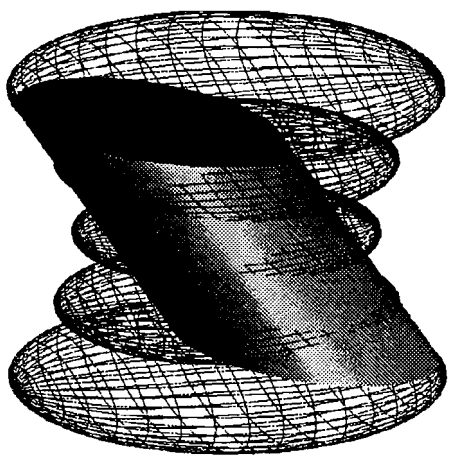

Figure 3.17 The same silhouette surface viewed from another direction

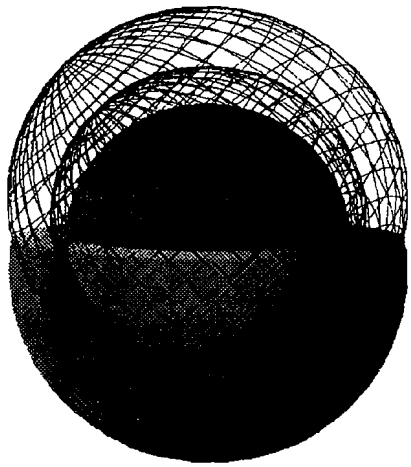

Figure 3.18 The hypersurface viewed from $\theta = (-165, 45, 15, -165, 84, 0)$

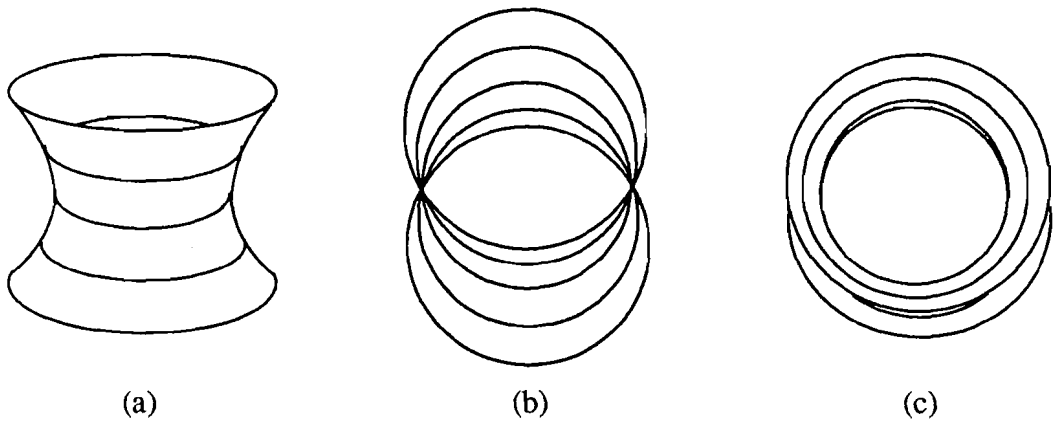

Figure 3.19 Surface $x^2 + y^2 - z^2 - 1 = 0$ with three different views

$$w = t$$

The analogous example is $x^2 + y^2 - z^2 - 1 = 0$ as shown in Figure 3.19(a). From the comparison it is easy to understand why in the 3D image space part of the isosurface $w = 0.5$ appears to be inside the boundary surface $w = 1$.

The interpretation by analogy has some limitations: The curve in Figure 3.15 has no counterpart in the 3D example. This is because the curve is only partly visible. According to its dimension, its counterpart should be a point, but a point cannot be only partly visible. The limitation illustrates the fact that 4D visualization really has some phenomena that have no analogy in 3D visualization.

Another phenomenon we observe is the *dimension reduction of the silhouette surfaces in image space.* In Figure 3.15, the hypersurface is viewed from $\theta_3 = \frac{5}{12}\pi$. As eye_4 rotates in the (z,w)-plane towards $\theta_3 = \frac{\pi}{4}$, the silhouette surface shrinks. It is reduced to a circle at $\theta_3 = \frac{\pi}{4}$. In the 3D image space it is still the envelope of the isosurfaces as shown in Figure 3.16. But in 4-space it is a cylinder swept by the circle in the direction towards eye_4. This can be seen by fixing the silhouette surface while rotating eye_4; see Figure 3.17. To understand this phenomenon, a similar situation is shown in Figure 3.19(b). When eye_3 is at an angle of $\frac{\pi}{4}$ with the (x,y)-plane, the silhouette curves of the surface $x^2 + y^2 - z^2 - 1 = 0$ are reduced to two points. In 3-space they are actually two lines on the surface.

As eye_4 rotates from $\theta_3 > \frac{\pi}{4}$ to $\theta_3 < \frac{\pi}{4}$, the 3D image of the silhouette surface changes from hyperboloid to ellipsoid. In addition, the volume that is the image of the hypersurface changes from "inside" the silhouette surface to "outside" the silhouette surface. See Figure 3.18 where eye_4 is at $\theta_3 = \frac{\pi}{12}$. The analogous case is shown in Figure 3.19(c).

These observations indicate that except for a few critical directions, the shape of silhouette surface is insensitive to small variations of the eye_4's orientation. This property might be useful for matching hypersurfaces in 4-space. Suppose one hypersurface is constructed from experimental data, and another hypersurface is accurately predicted by theory. Then their silhouette surfaces and the volume of the projected

hypersurfaces are expected to be similar even if their orientations do not perfectly coincide.

3.3 Observing the Curvature of a Hypersurface

Graphical methods have been used successfully in analyzing the intrinsic shape and curvature properties of surfaces in 3-space [Far87]. We discuss how to infer the curvature properties of a hypersurface in 4-space from the image of its silhouette surface. Notice that given a regular point on a hypersurface, it is always possible to make it a silhouette point by adjusting the projection direction.

Let $\mathcal{M}$ be a hypersurface in $\mathcal{R}^4$, and N be a unit normal vector field on $\mathcal{M}$. For each point $\mathbf{p}$ of $\mathcal{M}$, the *shape operator* of $\mathcal{M}$ at $\mathbf{p}$ is a linear operator $S_p\colon T_p(\mathcal{M}) \rightarrow T_p(\mathcal{M})$ given by

$$S_p(\mathbf{v}) = - \bigtriangledown_v N$$

It is a symmetric operator because it can be shown that $S_p(\mathbf{v}) \cdot \mathbf{w} = S_p(\mathbf{w}) \cdot \mathbf{v}$ for any pair of tangent vectors $\mathbf{v}$ and $\mathbf{w}$ at $\mathbf{p}$ [O'N66].

Let α be a curve on $\mathcal{M}$ with $\alpha(0) = \mathbf{p}$ and $\alpha'(0) = \mathbf{v}$. By Meusnier's theorem [dC76],

$$S_p(\mathbf{v}) \cdot \mathbf{v} = \alpha''(0) \cdot N(\mathbf{p})$$

The normal curvature of $\mathcal{M}$ in the direction of a unit vector $\mathbf{u}$ is defined as

$$k(\mathbf{u}) = S_p(\mathbf{u}) \cdot \mathbf{u}$$

The extreme values of the normal curvature $k(\mathbf{u})$ of $\mathcal{M}$ at $\mathbf{p}$ are the *principal curvatures* of $\mathcal{M}$ at $\mathbf{p}$, and are denoted by k_1, k_2, k_3. The directions in which these extreme values occur are called *principal directions* of $\mathcal{M}$ at $\mathbf{p}$. Unit vectors in these directions are called the *principal vectors* of $\mathcal{M}$ at $\mathbf{p}$ [O'N66, Tho79].

Lemma 3.1 The principal curvatures of $\mathcal{M}$ at $\mathbf{p}$ are the eigenvalues of the shape operator S_p and the principal vectors of $\mathcal{M}$ at $\mathbf{p}$ are the eigenvectors of S_p.

S_p is positive definite if $k_1, k_2, k_3 > 0$, negative definite if $k_1, k_2, k_3 < 0$, and indefinite otherwise. If the principal curvatures are distinct, then the principal directions are orthogonal. If they are not distinct, say $k_1 = k_2 \neq k_3$, then the principal directions corresponding to k_1, k_2 can be chosen arbitrarily in a plane orthogonal to the third principal direction.

Theorem 3.1 Let $\mathbf{p}$ be a silhouette point of a hypersurface $\mathcal{M}$ in $\mathcal{R}^4$ with respect to the projection $\varphi\colon \mathcal{R}^4 \to \mathcal{R}^3$. Consider the image of $\mathcal{M}$ in the neighborhood $\mathcal{D}$ of $\mathbf{p}$. The images of the silhouette surface and the tangent hyperplane are denoted by $\varphi(S)$ and $\varphi(T)$, respectively.

(a) If $\varphi(S)$ is a plane and $\varphi(\mathcal{D})$ is contained in $\varphi(S)$, then $k_1 = k_2 = k_3 = 0$.

(b) If $\varphi(S)$ is a plane and $\varphi(\mathcal{D})$ is on one side of $\varphi(S)$, then $k_1 = k_2 = 0$ and $k_3 \neq 0$.

(c) If $\varphi(S)$ is a cylindrical surface, $\varphi(S)$ and $\varphi(\mathcal{D})$ are on the same side of $\varphi(T)$, then $k_1 = 0$ and $k_2 k_3 > 0$.

(d) If $\varphi(S)$ is a cylindrical surface and is on one side of $\varphi(T)$, $\varphi(\mathcal{D})$ is on both sides of $\varphi(T)$, then $k_1 = 0$ and $k_2 k_3 < 0$.

(e) If $\varphi(S)$ is an elliptic surface with positive Gaussian curvature, $\varphi(S)$ and $\varphi(\mathcal{D})$ are on the same side of $\varphi(T)$, then $k_1 k_2 k_3 \neq 0$ and all have the same sign.

(f) If $\varphi(S)$ is an elliptic surface with positive Gaussian curvature, $\varphi(\mathcal{D})$ is on both sides of $\varphi(T)$, then $k_{min} < 0 < k_{max}$.

(g) If $\varphi(S)$ is a surface similar to a hyperbolic paraboloid with negative Gaussian curvature, then $k_{min} < 0 < k_{max}$.

The cases (a) to (g) are shown in Figure 3.20. In the pictures (a) and (b) a planar $\varphi(S)$ is drawn. An isosurface is drawn in picture (b) to show that $\varphi(\mathcal{D})$ is on one side of $\varphi(S)$. The isosurface is trimmed due to the visibility change at the silhouette surface S. In pictures (c) and (d) a cylindrical $\varphi(S)$ is drawn. An isosurface shows on which side is $\varphi(\mathcal{D})$. Although $\varphi(T)$ is not drawn in the pictures, it can be inferred from the shape of $\varphi(S)$ and the position of p. In pictures (e) and (f) an elliptic $\varphi(S)$

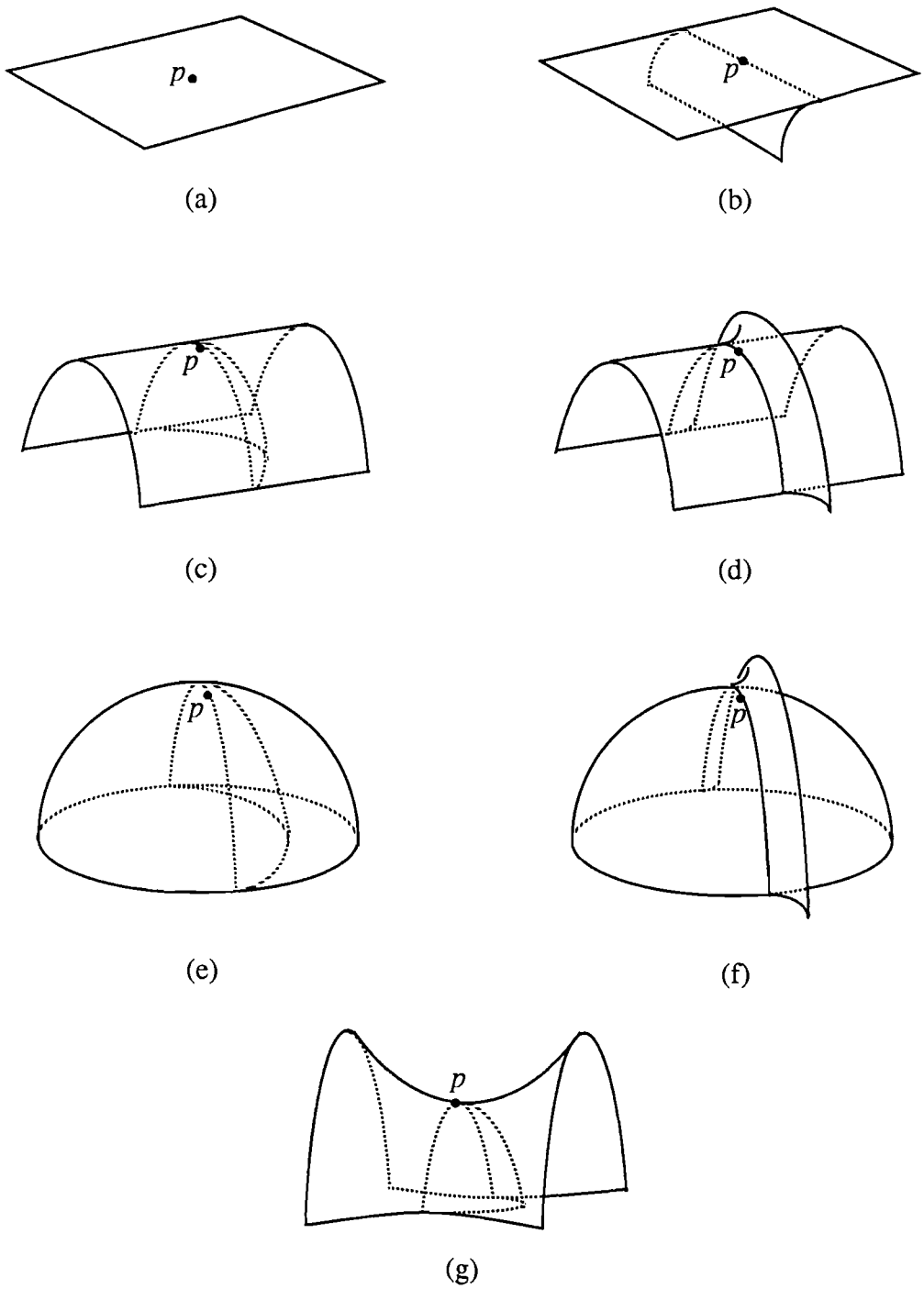

Figure 3.20 Curvature of a Hypersurface

and an isosurface are drawn. Picture (g) is the only case when $\varphi(S)$ is similar to a hyperbolic paraboloid. If in picture (g) $\varphi(\mathcal{D})$ is on the other side of $\varphi(S)$, it still represents the same case.

To prove Theorem 3.1, we use two lemmas. Recall that an active surface is a silhouette surface, a boundary surface, or a self intersection surface of a hypersurface.

Lemma 3.2 Suppose that a curve $\bar{\alpha}$ in the volume of $\varphi(\mathcal{M})$ does not cross any projected active surfaces. Then there is a curve α on $\mathcal{M}$ such that $\bar{\alpha} = \varphi(\alpha)$.

Proof: Assume that the hypersurface $\mathcal{M}$ is expressed in implicit form $f(x_1, x_2, x_3, x_4) = 0$ and the curve $\bar{\alpha}$ is expressed in parametric form $(\bar{\alpha}_1(t), \bar{\alpha}_2(t), \bar{\alpha}_3(t))$. Then we can consider t as known and solve the system of four equations in four variables: $f(x_1, x_2, x_3, x_4) = 0$ and $\varphi(x_1, x_2, x_3, x_4) = (\bar{\alpha}_1(t), \bar{\alpha}_2(t), \bar{\alpha}_3(t))$. There must be at least one solution $(x_1, x_2, x_3, x_4) = (\alpha_1(t), \alpha_2(t), \alpha_3(t), \alpha_4(t))$ that is continuous in t because $\bar{\alpha}$ does not cross any projected active surfaces. □

Lemma 3.3 Suppose that the curve α on a hypersurface $\mathcal{M}$ passes through a silhouette point $\mathbf{p}$ with respect to a projection φ. Let $\alpha(0) = \mathbf{p}$. Then $\alpha''(0) \cdot \mathbf{n}$ is proportional to $(\varphi \circ \alpha)''(0) \cdot \bar{\mathbf{n}}$, where $\mathbf{n}$ is the normal vector of $\mathcal{M}$ at $\mathbf{p}$ and $\bar{\mathbf{n}}$ is the normal vector of $\varphi(S)$ at $\varphi(\mathbf{p})$.

Proof: The curve $\alpha = (\alpha_1, \ldots, \alpha_4)$ is projected into $\mathcal{R}^3$ as

$$\varphi(\alpha) = \frac{(\alpha_1, \alpha_2, \alpha_3)}{1 - r\alpha_4}$$

Its first and second derivatives are:

$$\begin{aligned}
(\varphi \circ \alpha)' &= \frac{(\alpha_1', \alpha_2', \alpha_3')}{1 - r\alpha_4} + \frac{r\alpha_4'(\alpha_1, \alpha_2, \alpha_3)}{(1 - r\alpha_4)^2} \\
(\varphi \circ \alpha)'' &= \frac{(\alpha_1'', \alpha_2'', \alpha_3'')}{1 - r\alpha_4} + \frac{2r\alpha_4'(\alpha_1', \alpha_2', \alpha_3')}{(1 - r\alpha_4)^2} + \\
&\quad \frac{r\alpha_4''(\alpha_1, \alpha_2, \alpha_3)}{(1 - r\alpha_4)^2} + \frac{2r^2(\alpha_4')^2(\alpha_1, \alpha_2, \alpha_3)}{(1 - r\alpha_4)^3}
\end{aligned}$$

By Corollary 2.4 the normal vector $\bar{\mathbf{n}}$ of $\varphi(S)$ at $\varphi(\mathbf{p})$ is parallel to $\pi(\mathbf{n}) = (n_1, n_2, n_3)$. We can choose it as $\bar{\mathbf{n}}$ and get

$$\begin{aligned}(\varphi\circ\alpha)''(0)\cdot\bar{\mathbf{n}} &= \frac{\sum_{i=1}^{3}\alpha_i''(0)n_i}{1-r\alpha_4(0)} + \frac{2r\alpha_4'\sum_{i=1}^{3}\alpha_i'(0)n_i}{(1-r\alpha_4(0))^2} + \\ &\quad \frac{r\alpha_4''\sum_{i=1}^{3}\alpha_i(0)n_i}{(1-r\alpha_4(0))^2} + \frac{2r^2(\alpha_4')^2\sum_{i=1}^{3}\alpha_i(0)n_i}{(1-r\alpha_4(0))^3} \qquad (3.1)\end{aligned}$$

The ray from $\mathbf{p} = (\alpha_1(0), \ldots, \alpha_4(0))$ to eye_4 is $\mathbf{r} = (-\alpha_1(0), -\alpha_2(0), -\alpha_3(0), \frac{1}{r} - \alpha_4(0))$. By Theorem 2.3 we have $\mathbf{r}\cdot\mathbf{n} = 0$, i.e.

$$\sum_{i=1}^{3}\alpha_i(0)n_i = (\frac{1}{r} - \alpha_4(0))n_4 \qquad (3.2)$$

Also we know that $\alpha'(0)\cdot\mathbf{n} = 0$, i.e.

$$\sum_{i=1}^{3}\alpha_i'(0)n_i = -\alpha_4'(0)n_4 \qquad (3.3)$$

Substituting (3.2) and (3.3) into (3.1) yields

$$(\varphi\circ\alpha)''(0)\cdot\bar{\mathbf{n}} = \frac{\alpha''(0)\cdot\mathbf{n}}{1-r\alpha_4(0)}$$

□

Now we prove Theorem 3.1. From Lemma 3.2 we can choose an arbitrary curve in $\varphi(\mathcal{M})$ as $\varphi(\alpha)$. Let $\varphi(\alpha)$ pass through the point $\varphi(\mathbf{p})$ and assume that α passes through $\mathbf{p}$. By lemma 3.3 we know that $\alpha''(0)\cdot\mathbf{n}$ is proportional to $(\varphi\circ\alpha)''(0)\cdot\bar{\mathbf{n}}$. We cannot estimate quantitatively the normal curvature of $\mathcal{M}$ in the direction of $\alpha'(0)$ from $(\varphi\circ\alpha)''(0)\cdot\bar{\mathbf{n}}$ because we do not know if α is a unit speed curve. Nevertheless, we can obtain information about the sign of the normal curvature of $\mathcal{M}$ at $\mathbf{p}$.

(i) If $(\varphi\circ\alpha)''(0)\cdot\bar{\mathbf{n}} \neq 0$, then at least one of the principal curvatures of $\mathcal{M}$ is nonzero. If $(\varphi\circ\alpha_1)''(0)\cdot\bar{\mathbf{n}}$ and $(\varphi\circ\alpha_2)''(0)\cdot\bar{\mathbf{n}}$ are nonzero and have the opposite sign, then $\mathcal{M}$ has two principal curvatures of opposite sign.

(ii) In the neighborhood of $\varphi(\mathbf{p})$ if $\varphi(\alpha)$ is contained in $\varphi(T)\cap\varphi(S)$ where $\varphi(T)$ and $\varphi(S)$ coincide or tangent to each other, then one of the principal curvature of $\mathcal{M}$ is zero. To see this, notice that all the points in $S\cap\alpha$ share the same tangent hyperplane

T. Now α is contained in T and hence $\nabla_{\alpha'} N = \mathbf{0}$, or equivalently, $S_p(\alpha') = \mathbf{0}$. That means one of the eigenvalue of the shape operator S_p is zero.

Now we analyze the cases (a) to (g):

(a) Since $\varphi(\mathcal{D}) \subset \varphi(S) = \varphi(T)$, we know that $\mathcal{D}$ is part of a hyperplane, and so $k_1 = k_2 = k_3 = 0$.

(b) We choose two curves, $\varphi(\alpha_1)$ and $\varphi(\alpha_2)$ in $\varphi(S)$, intersecting transversely at the point $\varphi(\mathbf{p})$. From (ii) above, and that $\text{span}(\alpha_1', \alpha_2')$ has dimension two, we obtain that S_p has two zero eigenvalues. We choose one more curve $\varphi(\alpha_3)$ with nonzero curvature at $\varphi(\mathbf{p})$. From (i) above, the third eigenvalue is nonzero.

(c) The fact that $\varphi(S) \cap \varphi(T)$ is of dimension one implies that there is only one zero eigenvalue. Any other curve $\varphi(\alpha)$ in $\varphi(\mathcal{D})$, intersecting $\varphi(T)$ only at $\varphi(\mathbf{p})$ will have the same sign of $(\varphi \circ \alpha)''(0) \cdot \bar{\mathbf{n}}$. From Lemma 3.3 we obtain that $k_2 k_3 > 0$.

(d) Like (c) we have $k_1 = 0$. But now we can find two curves $\varphi(\alpha)$ in $\varphi(\mathcal{M})$ with opposite signs of $(\varphi \circ \alpha)''(0) \cdot \bar{\mathbf{n}}$. From (i) above we obtain that $k_2 k_3 < 0$.

(e) Any curve $\varphi(\alpha)$ in $\mathcal{D}$ must bend in the same way at $\varphi(\mathbf{p})$, which implies k_1, k_2, k_3 have the same sign.

(f) We can choose two curves $\varphi(\alpha)$ in $\varphi(\mathcal{D})$ with opposite signs of $(\varphi \circ \alpha)''(0) \cdot \bar{\mathbf{n}}$. Since we have no information about the third eigenvalue, we conclude $k_{min} < 0 < k_{max}$.

(g) No matter on which side of $\varphi(S)$ is $\varphi(\mathcal{M})$, we can always choose two curves $\varphi(\alpha)$ in $\varphi(\mathcal{D})$ with opposite signs of $(\varphi \circ \alpha)''(0) \cdot \bar{\mathbf{n}}$. Hence $k_{min} < 0 < k_{max}$. □

Take the hypersurface $x^2 + y^2 + z^2 - w^2 - 1 = 0$ as an example. No matter how we rotate eye_4, except for $\theta_3 = \pm\frac{\pi}{4}$, the 3D image of the hypersurface is similar to either Figures 3.15 or Figure 3.18. The images fall into the cases (g) and (f), respectively. We conclude that every point on the hypersurface has both positive and negative principal curvatures. This can be verified by calculation from the hypersurface definition.

Figures 3.21 and 3.22 show another example, the Gauss normal distribution in $\mathcal{R}^3$. The hypersurface is defined as

$$w - ae^{-\frac{x^2+y^2+z^2}{2}} = 0$$

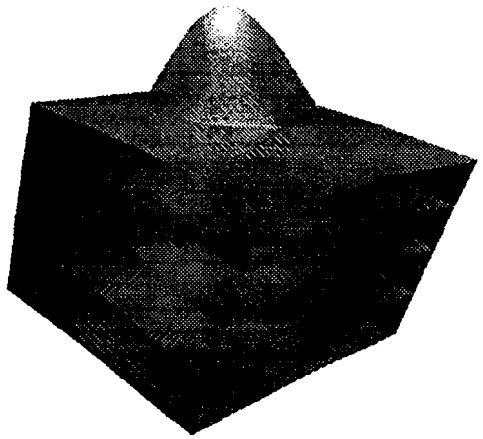

Figure 3.21 Hypersurface of Gauss Distribution

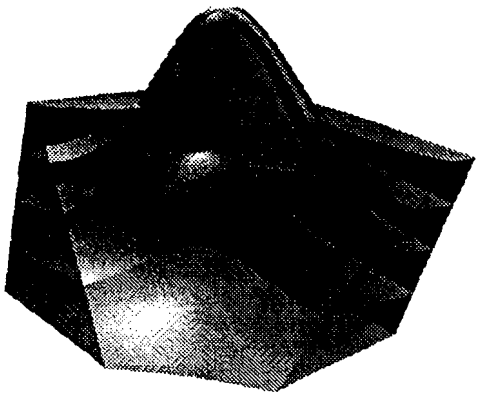

Figure 3.22 Hypersurface of Gauss Distribution Clipped by a Plane

where a is a constant. The volume displayed is $-2 \leq x, y, z, \leq 2$. Five isosurfaces with $z =$ constants of $-2, -1, 0, 1, 2$ and the boundary surfaces with $x, y =$ constants of $-2, 2$ are shown. Notice that if eye_4 is at infinity on the world w-axis, the picture is simply a box with several planes inside it. Even the isosurfaces $w =$ constant are trivial concentric spheres. In Figures 3.21 and 3.22 the viewing direction in Euler angles is $\theta = (0, \frac{\pi}{12}, \frac{\pi}{4}, \frac{3\pi}{4}, \frac{5\pi}{12}, 0)$. The silhouette surface appears due to the inclination of eye_4. The top of the bell-shaped silhouette surface is the part of the hypersurface near $p = (0, 0, 0, a)$. It is the case (e) of Theorem 3.1. The principal curvatures satisfy $k_1, k_2, k_3 > 0$ or $k_1, k_2, k_3 < 0$ depending on the choice of the normal vector of the hypersurface. The case (g) of Theorem 3.1 can be observed near the rim of the bell-shaped silhouette surface. The principal curvatures at those points satisfy $k_1 < 0 < k_3$. Notice that the hypersurface is mapped to both sides of the silhouette surface, but in $\mathcal{R}^4$ only those points whose images are inside the bell can be in the neighborhood of a silhouette point. This can be seen from the trim of the isosurface $z = 2$ at the top of the cube. The isosurface is occluded by the points on and inside the bell, but they are distant in $\mathcal{R}^4$. Theorem 3.1 only observes the neighborhood of a silhouette point in $\mathcal{R}^4$.

In [NFHL91] the curvature of a trivariate function was calculated from the definition and then displayed using a color scale. The curvature was defined as $K = k_1 k_2 k_3$, an extension to the Gaussian curvature. It seems hard to use a scalar quantity to represent the curvature of a 3-dimensional hypersurface. For example, when $k_1 k_2 k_3 < 0$ it could be the case $k_1 < 0,\ k_2 < 0,\ k_3 < 0$ as shown in Figure 3.20(e), or the case $k_1 < 0,\ k_2 > 0,\ k_3 > 0$ as shown in Figure 3.20(f) and (g). Our method of displaying curvature is more intuitive in that the curvature is inferred from the hypersurface's image rather than by a calculation of a scalar value from its equation.

The idea of our method is adjusting the viewing direction so that at the point the normal of the hypersurface is orthogonal to the viewing direction. An alternate method would be adjusting the viewing direction so that at the point the normal of the hypersurface is parallel to the viewing direction. Then the curvature can be

observed from the 2-surface that is the intersection of the hypersurface and its tangent hyperplane. This method is an extension to the Dupin curve or Dupin indicatrix of a surface in $\mathcal{R}^3$ [dC76, O'N66]. We think that further investigation of this method will produce useful and interesting results.

4. SYSTEM ARCHITECTURE

We have implemented an Interactive 4D Visualization System, named IView, on a conventional z-buffer based 3D graphics workstation. In this chapter some related algorithms, data structures, and efficiency considerations are discussed. After an overview of the system in the first section, two major issues, namely polygonalization and visibility determination, are presented in successive sections.

4.1 System Overview

There are several goals determining the design of the system:

1. Interactive response. If the frame modification does not involve a 4D operation, the response should be as fast as in ordinary 3D graphics. For example, when eye_3 rotates or the z-clipping plane moves, a scene of about five thousand polygons takes a few tenths of second to update.[1] The response time to an eye_4 motion is about two or three times that of an eye_3 motion. According to our experience, to examine a 4D object, after one motion of eye_4, many motion steps of eye_3 are necessary to understand fully the shape of the 3D images. For complicated 4D operations like silhouette surface generation plus 4D visibility determination, as shown by the hypersurface examples in Chapter 3, the response time is still within a tolerable range up to fifteen seconds.

2. Flexibility. All 2-surfaces and curves are considered independent objects, and so can be translated and rotated. They can be grouped into arbitrary levels. They can also be shaded by different methods, e.g. wire mesh, opaque or transparent surface, depth-cueing, or shaded by 4D lighting. The visibility mode can be

[1] Experiments were conducted on a Silicon Graphics Inc. Personal Iris 4D/35.

controlled independently of the shading method. Currently the system supports three visibility modes: eye_4-visible, eye_3-visible, and invisible modes. The eye_3-visible mode shows the conventional hidden surface removal in 3D image space. The eye_4-visible mode shows the visibility associated with both the first and the second projections. This mode is useful to display a hypersurface in 4-space. A hypersurface is considered a special group consisting of silhouette surface, boundary surfaces, self-intersection surfaces, and isosurfaces. When shown in eye_4-visible mode, all the 2-surfaces on the hypersurface are trimmed according to the visibility associated with the first projection. The generation of the 2-surfaces on a hypersurface can be controlled independently. For example, it is possible to fix a silhouette surface while rotate eye_4 to show the phenomenon in Figure 3.17. A similar example, fixing the silhouette curve of a 2-surface with respect to the projection from 4-space to 2-space while rotate both eye_4 and eye_3, will be given in Chapter 5. It is also very useful to turn off some of the objects by making them invisible. Note that objects behind eye_4 are typically culled by the projection algorithm.

3. Compatibility with 3D Graphics. As mentioned in Chapter 2, the three Euler angles controlling eye_3 are just the ordinary azimuthal, incidence and twist angles in 3D graphics, and the three Euler angles controlling eye_4 are their natural extension. The 3D image of 4D objects can be shaded according to a 3D lighting model. In our system, objects in 3-space can be displayed in two ways. By setting $w = 0$ the objects exist in the subspace spanned by the world x-, y-, and z-axes. When eye_4 rotates, the shapes of these objects could be changed. Another way is to put those objects into 3D image space. Then eye_4 motion has no effect on them at all. This is not only compatible with existing 3D graphics, but also useful for understanding 4D objects. For example, we can display several 4D objects together with a 3D cube. While eye_4 is moving, the 3D images of 4D objects change, but the 3D cube remains unchanged, giving clues how the 3D images are changed in response to the eye_4 motion.

The system architecture is shown in Figure 4.1.

The objects in this system are classified into the following types:

1. CURVE3 and CURVE4. These are parametric curves in 3D image space and 4D world space, respectively. They are transformed into types PT3 and PT4, a list of points in 3-space and 4-space, respectively.

2. SURF4P and SURF4I. These are parametric and implicit 2-surfaces defined in 4-space. The functions defining the surfaces may be arbitrary C^1 continuous functions. After polygonization, they become types POLY4T and POLY4N, a list of polygons, each vertex attached with two tangent vectors or two normal vectors, respectively. If the 2-surface is defined by a complicated procedure, it can be polygonalized by a separated program and sent to the system as type POLY4T or POLY4N. Another type, POLY4, without tangent or normal vectors, can be used to define polytops in 4-space.

3. SURF3P and SURF3I. These are parametric and implicit 2-surfaces defined in 3D image space. They are polygonized into type POLY3N, i.e., 3D polygons with a normal vector at each vertex. The type of 3D polygons without normal vector is POLY3.

4. GROUP. Objects of all types can be grouped together. For example, a set of isosurfaces, or a 2-surface with silhouette curves. All objects in the system are thus organized into a tree structure. The effect of moving the aim point could be achieved by translating the root object.

5. HYPER. This is the type to represent a hypersurface in 4-space. Similar to the type GROUP, it contains as members the silhouette surface, self-intersection surfaces, boundary surfaces, isosurfaces and curves on a hypersurface. When displayed in eye_4-visible mode, it is used for trimming the 2-surfaces and curves on the hypersurface according to the definition of 4D visibility. How the member

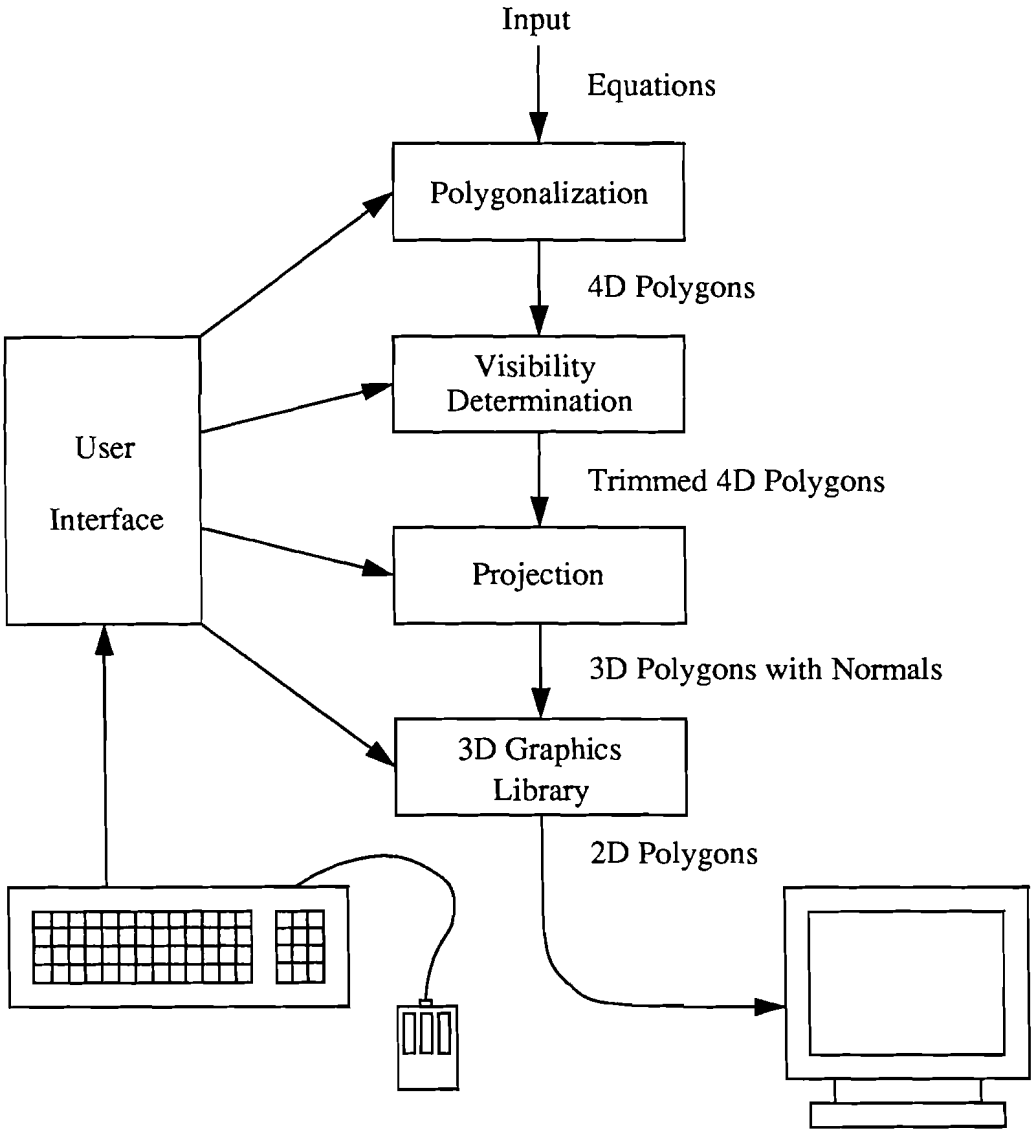

Figure 4.1 System architecture

surfaces and curves are to be generated is not automatic in order to increase flexibility.

The top level algorithms are the following:

1. Polygonalization. Polygonalizing parametric surfaces and 3D implicit surfaces has been discussed extensively in the literature, e.g. [Blo88, HW90, LC87, RHD89, Vla90]. So, we will concentrate on the polygonalization of 2-surfaces in 4-space. We adapt the algorithm presented by Allgower [AG87] with some special considerations for 4D visualization. After the polygons are generated, Newton iteration is applied to bring the points onto the 2-surface. Finally, tiny polygons or those of poor aspect ratios are merged with adjacent polygons to reduce the total number of polygons.

2. Visibility determination. First, generate all the intersection curves between those pairs of 2-surfaces, in which at least one of them is an active surface, in 3D image space. The intersections tend to be singular, e.g. intersecting tangentially, or several intersection curves meeting at a common point. This was predicted by the discussion in Chapter 2 and was illustrated by the pictures in Chapter 3. The intersection curves divide the 2-surfaces into regions. Then, for each region choose one point to calculate the sight number and propagate it across the whole region. Invisible parts of polygons are trimmed away.

3. Projection and Shading. To make full use of the 3D graphics capacity, the coordinates and normals of polygon vertices in 3D image space are calculated and stored in memory. They are updated only when the objects are translated or rotated in 4-space, or when the position of eye_4 is modified. The wire mesh, flat, and Gouraud shading can be performed by the 3D graphics library. But other shading methods have to be implemented by the system. For examples, When using the depth cueing in 4-space, the color and intensity at a point is determined by the distance from that point to eye_4. When shaded by 4D lights,

the intensity at a point is determined by the dot product of the hypersurface normal and the vector from that point to eye_4.

4.2 Polygonalization of Implicit 2-Surfaces in 4-Space

An implicit 2-surfaces in 4-space is defined by two equations in four variables: $f(x,y,z,w) = 0,\ g(x,y,z,w) = 0$. Although all 2-surfaces have to be projected into 3D image space before they can be displayed, we prefer polygonalization in 4-space for the following reasons. First, polygonalization usually requires more computation than projection. Therefore, polygonalizing the 2-surface as a preprocessing step means that a better response can be obtained when changing the projection parameters repeatedly. Most of the 2-surfaces need only to be polygonalized once. Second, the silhouette surface is determined by the eye_4 position. It has to be repolygonalized whenever eye_4 moves. However, we may wish to keep its polygonalized form in 4-space in order to show a "fixed" silhouette surface during eye_4 rotation. Third, the visibility calculation must be based on the positions of polygons in 4-space, similar to the situation in 3D graphics. Finally, polygonalization in 4-space can better account for the intrinsic geometric properties of the 2-surface. Some of these properties are distorted by the projection to 3-space.

If f and g are restricted to polynomials, algebraic methods such as resultants could be used to eliminate one variable, followed by a parameterization in 3-space. It is also possible to parameterize certain algebraic hypersurfaces in 4-space and then display a set of isosurfaces [Baj90], but it seems hard to directly parameterize a 2-surface in 4-space.

If f and g are general functions, or even defined by discrete function values on grid points, direct polygonalization in 4-space is more appropriate. A few methods have been proposed to polygonalize implicit 2-surface in high dimensional space. Categorized by the space the algorithm subdivides, they are: Allgower's simplicial continuation method [AS85, AG87] which subdivides the object space, Rheinboldt's moving-frame method [Rhe87] which subdivides the tangent space of the object, and

Chuang's method based on the local degree 2 approximation [Chu90] which subdivides the image space. Rheinboldt and Chuang's methods require tangent or normal vectors while in Allgower's algorithm the first order partial derivatives are used only for an optional point refinement step. It is possible to apply Allgower's algorithm without the point refinement step, and this is useful for visualizing functions defined by the values on grid points. Another advantage of Allgower's algorithm is that by triangulating the object space it can handle the case where the projection introduces apparent singularities (see the discussion in Chapter 2), and the case where the 2-surface is closed in the object space. The disadvantage of the algorithm is that its complexity is exponential in the dimension of the space, but this is not an issue here since the dimension is fixed at 4.

The simplicial continuation method described in [AG87] has been tailored to a version dealing with 2-surfaces in 3-space [AG90]. In the following three subsections we will discuss some considerations for applying the algorithm to 2-surfaces in 4-space: the basic algorithm and data structure, the Newton iteration for point refinement, and the merging of polygons to reduce the total number of polygons.

4.2.1 The Basic Algorithm and Data Structure

The following definitions are briefly cited from [AG87] with a restriction in 4-space.

The *Freudenthal Triangulation* of $\mathcal{R}^4$ is a set of 4-simplices $\sigma = [v_0, \ldots, v_4]$ where $[\cdot]$ denotes the convex-hull, and the vertices v_i satisfying

$$\begin{aligned} v_0 &= \delta z \\ v_i &= v_{i-1} + \delta e_{\pi(i)} \qquad i = 1, \ldots, 4 \end{aligned}$$

In the definition above z is an integer vector in $\mathcal{R}^4$, π is a permutation of $(1, \ldots, 4)$, $(e_1, \ldots, e_4)$ is the base of $\mathcal{R}^4$, and δ is the uniform mesh size. The triangulation is a subdivision of the grid in $\mathcal{R}^4$ with size δ. Each hypercube in the grid contains $4! = 24$ simplices. Each 4-simplex contains C_5^{i+1} i-faces, also called vertices, edges, faces, and

facets, respectively. Two adjacent simplices share 4 vertices, 6 edges, 4 faces and 1 facet. The number of simplices containing a common face is either 4 or 6.

The *pivoting of a vertex* v_i *across the facet* i is given by $\tilde{v}_i = v_{i-} - v_i + v_{i+}$ where $i+ = (i+1) \bmod 5$ and $i- = (i+4) \bmod 5$. By this operation a simplex $\sigma = [v_0, \ldots, v_i, \ldots, v_4]$ is pivoted to $\tilde{\sigma} = [v_0, \ldots, \tilde{v}_i, \ldots, v_4]$.

Let $\tau = [v_0, v_1, v_2]$ be a face, and let $H : \mathcal{R}^4 \rightarrow \mathcal{R}^2$ define the 2-surface $H(x) = (f(x), g(x))^T = 0$. Then τ is *completely labeled* if the labeling matrix

$$\Lambda = \begin{pmatrix} 1 & 1 & 1 \\ H(v_0) & H(v_1) & H(v_2) \end{pmatrix}$$

has a lexicographically positive inverse, which means that in each row of Λ^{-1} the leading nonzero element is positive. When τ is completely labeled, the first column of Λ^{-1}, written as $(\lambda_0, \lambda_1, \lambda_2)^T$, gives an approximate intersection point $x = \lambda_0 v_0 + \lambda_1 v_1 + \lambda_2 v_2$ of the 2-surface with the face. A simplex, a facet, or a face is called *transverse* if it is or contains a completely labeled face.

Algorithm 4.1 Simplicial Continuation Method

input: A 2-surface defined by $H : \mathcal{R}^4 \rightarrow \mathcal{R}^2$, and a regular point x on the 2-surface. Also a domain $D \subset \mathcal{R}^4$ and a mesh size δ.

output: A list of polygons that is the piecewise linear approximation of H.

(1) σ_0 = The first transverse simplex containing the seed x;
(2) $L = \text{list}(\sigma_0)$;
(3) $L_c = L$; /* L_c is a pointer */
(4) while $L_c \neq$ nil do
(5) σ = head of L_c;
(6) L_c = rest of L_c;
(7) Step in σ along the chain of transverse faces;
(8) Construct the polygon, i.e. the linear approximation of H in σ;
(9) for each transverse facet τ of σ do

(10) $\tilde{\sigma}$ = pivot σ across τ;

(11) if $\tilde{\sigma} \subset D$ and $\tilde{\sigma} \notin L$ then

(12) append $\tilde{\sigma}$ to the end of L;

(13) endfor

(14) endwhile

The data structures involved in the algorithm are simplex, face, and vertex. A facet is represented equivalently by the missing vertex. Moreover, we need to represent polygons and their vertices also. To avoid confusion, a polygon vertex is called a *point* since initially it is a point on a face of a simplex. There is a one-to-one correspondence between a simplex and a polygon, and between a transverse face and a point. The adjacency information between simplices and transverse faces also represent the adjacency information between polygons and points. Therefore we have three types of data structures: VERTEX, FACE/POINT, and SIMPLEX/POLYGON. In a VERTEX the function values of f and g at the vertex are stored to avoid repeated evaluations. A FACE/POINT contains the 3D and 4D coordinates and normals, and a set of pointers to all the adjacent simplices/polygons. In SIMPLEX/POLYGON there is a set of pointers to all the adjacent faces/points.

The information about faces and vertices could be stored locally in the simplex, or globally in a table, say a hash table, to make sure there is only one copy in the memory. The latter is preferable for efficiency, robustness, and convenience when constructing connected polygons.

Given an n-simplex in $\mathcal{R}^n$, $\sigma = [v_0, \ldots, v_n]$, its i-face ($i = 0, \ldots, n$) is $\tau = [u_0, \ldots, u_i]$ where $\{u_0, \ldots, u_i\} \subset \{v_0, \ldots, v_n\}$. To address the face τ without referring to the simplex σ, its *code* is defined as

$$C_\tau = \sum_{j=0}^{i} z_j = \frac{i}{\delta} \text{barycenter of } \tau$$

where z_j is the integer vector such that $u_j = \delta z_j$. To recover the vertices of an i-face from its code, the following equations could be used.

$$z_0 = C_\tau/(i+1)$$

$$z_j = z_{j-1} + \sum_{C_\tau[k] \bmod (i+1) = i-j+1} e_k \quad j = 1, \ldots, i$$

A special case is when $i = n$, and recovers the simplex σ, as discussed in [AG87]. Given a lower dimensional face τ in a simplex σ, it is impossible to identify the original simplex σ from the code C_τ of τ. This represents the global nature of C_τ. The face τ is no longer associated with a particular simplex.

We use as example the silhouette surface shown in Figure 3.15, and compare the global versus the local storage strategies.

	global	local
Total number of simplices constructed	3834	3834
Total number of transverse faces evaluated	2918	5590
Total number of vertices evaluated	1164	4086
Singular Λ matrices encountered	35	52
Time in seconds	5	40

4.2.2 Newton Iteration for Point Refinement

After a transverse face is found, an initial estimate of the intersection point of the 2-surface and the face can be obtained via $x^0 = \lambda_0 v_0 + \lambda_1 v_1 + \lambda_2 v_2$. Then Newton iteration could be used to refine the point onto the 2-surface if the first order partial derivatives are available. In polygonalization of a 2-surface in 3-space, such refinement is usually restricted to the transverse edge because the 2-surface intersects all the transversal edges by the mean value theorem; see Figure 4.2. Such a restricted refinement method has the advantage that each refined polygon is still within the simplex originally containing it. But this is no longer the case for polygonalizing 2-surface in 4-space. Now a transverse face need not have intersection point with the 2-surface, as illustrated by the following example.

Example 4.1 A silhouette surface is defined by

$$f(x, y, z, w) = x^2 + y^2 + z^2 - w^2 - 1 = 0$$

$$g(x, y, z, w) = -0.48296x + 0.48296y - 0.68301z + 0.25882w = 0$$

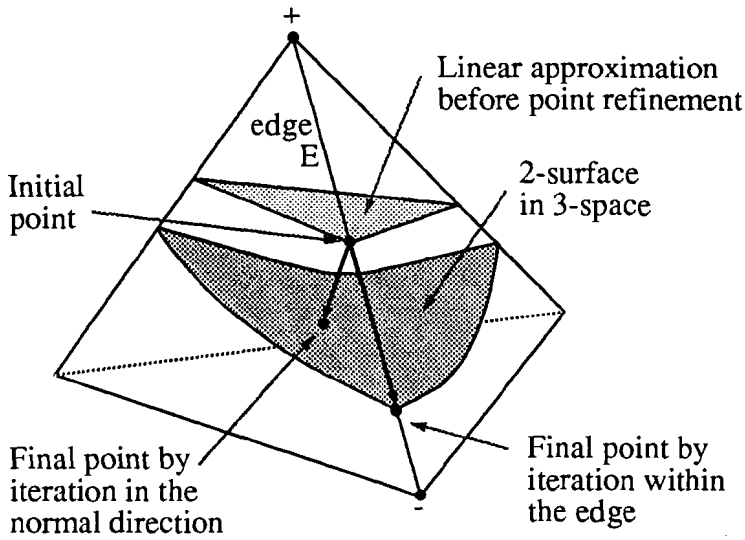

Figure 4.2 Point Refinement in 3-Space

where f is a hypersurface and g is a condition for the silhouette surface when eye_4 is at $\theta = (\pi/4, \pi/4, 7\pi/12)$ and $r_4 = 0$. The faces we are going to examine is $\tau = [v_0, v_1, v_2]$, where $v_0 = (0.75, 0.75, 0, -0.25)^T$, $v_1 = (0.75, 1, 0.25, -0.25)^T$, $v_2 = (0.5, 0.75, 0, -0.25)^T$. A point p on the face in barycentric coordinates is $p = \sum_{i=0}^{2} \alpha_i v_i$ with $\sum_{i=0}^{2} \alpha_i = 1$. The functions f and g substituted by (α_0, α_1) yields

$$\begin{aligned} \bar{f}(\alpha_0, \alpha_1) &= 0.1875\alpha_1^2 + 0.125\alpha_0\alpha_1 + 0.625\alpha_1 + 0.0625\alpha_0^2 + 0.25\alpha_0 - 0.25 \\ \bar{g}(\alpha_0, \alpha_1) &= -0.17075\alpha_1 - 0.12074\alpha_0 + 0.056036 \end{aligned}$$

The two solutions are $(\alpha_0 = -0.10091, \alpha_1 = 0.39952)$ and $(\alpha_0 = 3.60716, \alpha_1 = -2.22249)$. Both are outside the face τ. But the face is transverse since the matrix Λ^{-1} has a positive first column $(0.063497, 0.283271, 0.653232)^T$.

Three point refinement methods based on Newton iteration are considered. We call them *iteration in the support plane, iteration within the face*, and *iteration in the normal plane*, as shown in Figure 4.3

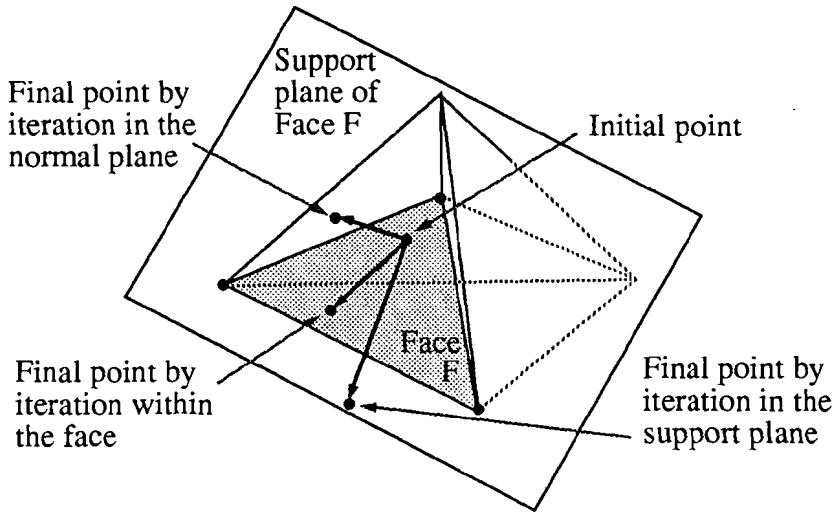

Figure 4.3 Point Refinement in 4-Space

Let τ be a face and P be its support plane, viz. $\tau \subset P$. The method of iteration in the support plane is just the Newton iteration of the point in P without checking the boundary of τ:

$$\alpha^{k+1} = \alpha^k - (J(\bar{H})(\alpha^k))^{-1}\bar{H}(\alpha^k)$$

where $\alpha = (\alpha_0, \alpha_1)^T$, $\bar{H}(\alpha) = (\bar{f}(\alpha), \bar{g}(\alpha))^T$, and

$$\begin{aligned} J(\bar{H})(\alpha) &= \begin{pmatrix} \bar{f}_{\alpha_0} & \bar{f}_{\alpha_1} \\ \bar{g}_{\alpha_0} & \bar{g}_{\alpha_1} \end{pmatrix} \\ &= \begin{pmatrix} \nabla f^T(\alpha_0 v_0 + \alpha_1 v_1 + \alpha_2 v_2) \\ \nabla g^T(\alpha_0 v_0 + \alpha_1 v_1 + \alpha_2 v_2) \end{pmatrix} \begin{pmatrix} v_0 - v_2 & v_1 - v_2 \end{pmatrix} \end{aligned} \tag{4.1}$$

Applying this method to an initial estimated point inside the face τ, the iteration could converge to a point outside τ as in Example 4.1. Experiments with this method showed that about one percent of the refinement converged to points outside the face. Moreover, a few points were far away from the face. In that case, the 2-surface looks smooth except for a few bumps. Worse, if the 3D graphics engine cannot handle nonconvex polygons, cracks may appear in the projected 2-surface.

Another drawback is that the refined polygons can cross the boundaries of simplices. In visibility determination or rendering by ray tracing, the algorithms may use the structure of the space division. The existence of crossing polygons will make those algorithms much more complicated.

To confine the final polygons within the originally enclosing simplex, the method of iteration within the face could be applied. The process does not find an actual zero of $\bar{H}$ but finds an optimal approximation defined by[2]

$$\begin{aligned} \min \quad & \bar{h}(\alpha) \\ \text{s.t.} \quad & \alpha \in \Omega \end{aligned}$$

where

$$\begin{aligned} \bar{h}(\alpha) &= \|\bar{H}(\alpha)\|^2 \\ &= \bar{f}^2(\alpha) + \bar{g}^2(\alpha) \\ \Omega &= \{\alpha \mid (0 \le \alpha_0 \le 1) \wedge (0 \le \alpha_1 \le 1) \wedge (0 \le \alpha_0 + \alpha_1 \le 1)\} \end{aligned}$$

This is a constrained nonlinear programming problem and many methods are available. For example, the steepest descent method can be used when only the first order partial derivatives are available. At each iteration step, a linear search is conducted in the direction of the negative gradient of $\bar{h}$:

$$\begin{aligned} d^k &= -\frac{1}{2} \bigtriangledown \bar{h}(\alpha^k) \\ &= -(J(\bar{H})(\alpha^k))^T \bar{H}(\alpha^k) \qquad (4.2) \\ \alpha^{k+1} &= \alpha^k + \mu^k d^k \\ \text{s.t.} \quad \bar{h}(\alpha^k + \mu^k d^k) &= \min\{\bar{h}(\alpha^k + \mu d^k) | \mu > 0 \wedge \alpha^k + \mu d^k \in \Omega\} \end{aligned}$$

When α^k is at the boundary edge of Ω and the d^k calculated by Equation 4.2 is toward the outside, the d^k should be replaced by its projection on the edge. If the the length of the projection is zero, a local optimal point is found.

[2]The abbreviation s.t. stands for "subject to".

The steepest descent method converges only linearly. Since ninety-nine percent of the points are really on the face, the method wastes time. Therefore, we consider a modification of the iteration in the support plane.

$$\begin{aligned} d^k &= -(J(\bar{H})(\alpha^k))^{-1}\bar{H}(\alpha^k) \\ \alpha^{k+1} &= \alpha^k + \mu^k d^k \\ \text{s.t.} \quad \mu^k &= \min\{\mu \mid (\mu > 0 \wedge \alpha^k + \mu d^k \in \partial\Omega) \vee (\mu = 1)\} \end{aligned}$$

When α^k is at $\partial\Omega$ and d^k is toward outside, the iteration stops since there is no criterion of optimization. For those points that are really on the faces and the initial points are sufficiently close to them, this method converges at the same rate as the method of iteration in the support plane. For those points outside the face, however, the iteration is stopped and the points stay at the edges of the faces and do not cross the enclosing simplices. The overall speed is found to be almost the same as that of iteration in the support plane. However, this method has the following problem. The points in the iteration $\alpha^0, \alpha^1, \ldots$ may go in and go out of the face several times. When simply terminating the iteration the convergent point may be lost. Therefore, the visual effect of this method may be better or worse than that of iteration in the support plane depending on the surface geometry.

If crossing polygon is allowed, a better method is the iteration in the normal plane of the 2-surface. Let $p = (x, y, z, w)^T$ and p^k be a point close to the 2-surface $H(p) = (f(p), g(p))^T = 0$. Then $\bigtriangledown f(p^k)$ and $\bigtriangledown g(p^k)$ are two approximate normal vectors of $H(p) = 0$ passing through p^k. Let $\alpha = (\alpha_0, \alpha_1)^T$. A point in this approximate normal plane is $p^k + \alpha_0 \bigtriangledown f(p^k) + \alpha_1 \bigtriangledown g(p^k) = p^k + J(H)^T(p^k)\alpha$. Define $\hat{H}(\alpha) = H(p^k + \alpha_0 \bigtriangledown f(p^k) + \alpha_1 \bigtriangledown g(p^k))$. Expanding near $\alpha = 0$ yields

$$\hat{H}(\alpha) = \hat{H}(0) + J(\hat{H})(0)\alpha + o(\alpha)$$

By setting $\hat{H}(\alpha) = 0$ and rewriting $J(\hat{H})(0)$ in terms of $J(H)$ the following iteration is obtained:

$$p^{k+1} = p^k - J(H)[J(H)^T J(H)]^{-1}(p^k)H(p^k)$$

The same result can also be derived by the least-squares method. Based on this geometric explanation we make two observations. First, since in each step of the iteration the refinement of the point is in the direction within the normal plane, faster convergence is expected than in case of iteration in the support plane. Second, in most situations the normal plane is different from the face support plane, so this method generates much more crossing polygons than iteration in the support plane. Visually this method delivers the best pictures among the three methods. The system lets the user choose one of these iteration methods. Note that in consequence the structure of the space subdivision cannot be assumed by any algorithm applied after the refinement stage.

4.2.3 Merging Polygons

The polygons generated by the simplex method contain some tiny and skinny polygons. Merging them with adjacent polygons could be done before or after the point refinement stage. The former could make use of the underlying space subdivision structure, but the curvature information is not available since the points are not yet on the 2-surface. The latter is independent of how the polygons are generated, and so it can also be applied to polygons of parametric 2-surfaces. Moreover, from the normals at the refined points, the flatness of the 2-surface can be estimated, a more important criterion than the size of polygons. The system offers an optional operation to merge polygons after the point refinement stage.

The operation to merge polygons can be done in 4-space or in the 3D image space. Since only the 3D images are seen, We thought it is better to merge polygons according to the normals of the projected 2-surface. However, the operation performed in 4-space has its advantages. All the 2-surfaces except the silhouette surface are independent of eye_4 position. Therefore, the merging of 4D polygons needs to be done only once. The saved final version of 4D polygons could be projected into 3D image space repeatedly. Our experiments comparing the operations done in 4-space

versus 3D image space showed little difference in the visual effect and the number of polygons merged.

Tiny or skinny polygons have at least one short edge, so the basic operation is to merge two vertices of a short edge. The two vertices v_1 and v_2 of an edge can be merged into one if the following condition is satisfied.

$$(\|v_1 - v_2\| < \varepsilon) \wedge (1 - N_1(v_1) \cdot N_1(v_2) < \eta) \wedge (1 - N_2(v_1) \cdot N_2(v_2) < \eta) \qquad (4.3)$$

where ε and η are user defined tolerances, and N_1 and N_2 are two unit normal vector fields of the 2-surface. This condition is conservative in that the coincidence of two normal planes does not require the coincidence of the normal vectors. The meaning of (4.3) is that each hypersurface in the definition of the 2-surface has almost coincident normal vectors at v_1 and v_2. For parametric 2-surfaces, the N_1 and N_2 in (4.3) are substituted by T_1 and T_2. That is, we base merging on the tangent plane instead of the normal plane. We do not calculate the normal vectors of a parametric 2-surface from the tangent vectors since the the calculated normal vectors do not bear their own meaning.

It has been proved that each polygon generated by the simplex method has at most 5 vertices, and that each vertex is shared by at most 6 polygons [AS85, AG87]. It is useful for data structure and algorithm design if these limits are kept when vertices are merged. Suppose v_1 and v_2 are merged into v. All the polygons adjacent to v_1 or v_2 will not increase their number of sides. But $\deg(v)$, i.e., the number of edges incident to the new vertex v, could be larger than $\deg(v_1)$ or $\deg(v_2)$. So, another condition for merging two vertices is

$$\deg(v) = \deg(v_1) + \deg(v_2) - \mathrm{tri}(v_1, v_2) - 2 \leq 6 \qquad (4.4)$$

where $\mathrm{tri}(v_1, v_2)$ is the number of triangles (polygons with 4 or more sides are excluded) adjacent to the edge (v_1, v_2). The new v could be one of v_1 or v_2 provided that a boundary vertex is never removed.

Algorithm 4.2 Merging Polygons

input: A list of polygons in 4-space, each vertex is attached with two normal vectors or two tangent vectors.

output: A list of polygons with the total number possibly reduced.

```
(1)   for each edge (v1, v2) in the polygon list do
(2)       if Conditions (4.3) and (4.4) are true and
(3)           v1 or v2 is not a boundary vertex then
(4)               merge(v1, v2);
(5)               remove triangles containing edge (v1, v2)
(6)       endif
(7)   endfor
```

The algorithm can be applied repeatedly until the number of polygons cannot be further reduced. The system lets the user apply this operation and inspect the results interactively. Usually one application is enough. For example, applying Algorithm 4.2 once with $\varepsilon = 0.01$ and $\eta = 0.0001$ to the 2-surface defined in Example 4.1 reduces the number of polygons from 3834 to 3110 without any noticeable change in the 3D image. With the same ε and η the minimum number of polygons, 3046, is reached by repeating the algorithm four times.

4.3 Visibility Determination

As discussed in Chapter 2, the visibility to eye_4 needs to be done only if there are hypersurfaces. In our system a hypersurface is represented by an object of type HYPER that is a group of 2-surfaces and curves on that hypersurface. The group contains active surfaces including silhouette surfaces, boundary surfaces, self-intersection surfaces, and inactive surfaces and curves, including isosurfaces. An algorithm based on Theorem 2.6 is as follows:

Algorithm 4.3 Visibility Determination of a Hypersurface to eye_4

input: The equations defining implicitly the hypersurface and its silhouetter surface, boundary surfaces, and self-intersections (if they exist). The piecewise linear approximations of all the 2-surfaces and curves on the hypersurface, in the form of a list of connected polygons, or a list of connected line segments. Besides, the projection φ from 4-space to the 3D image space, and the position of eye_4 in the local coordinate system.

output: The polygons or line segments in the piecewise linear approximation of the 2-surfaces and curves having been trimmed according to the visibility to eye_4.

```
(1)   for each active surface S do
(2)       for each 2-surface or curve R do
(3)           construct the intersection of φ(S) and φ(R), put on R
(4)       endfor
(5)   endfor
(6)   for each 2-surface or curve R do
(7)       for each region of R divided by the intersection constructed in (3) do
(8)           choose a point in the region and calculate sight number
(9)           propagate the sight number to the whole region
(10)      endfor
(11)  endfor
```

In the following two subsections step (3) of Algorithm 4.3 is elaborated for the general case, the special case when S is a silhouette surface, and the special case when S is a boundary surface. The last subsection deals with steps (8) and (9) of the algorithm.

4.3.1 Intersection of 2-surfaces in the Image Space

Given two 2-surfaces R and S in $\mathcal{R}^4$, after projection their images $\varphi(R)$ and $\varphi(S)$ are surfaces in $\mathcal{R}^3$. The intersection of two surface in $\mathcal{R}^3$ has been studied for a

long time. The problem can be solved by algebraic, analytic, and numerical methods [PG86, CK87, OR87, BHHL88]. Here it is discussed in the context of 4D visualization. We distinguish three basic methods: (1) Both 2-surfaces are in symbolic form. (2) Both are piecewise linear. (3) One is in symbolic form and one is piecewise linear.

We outline the first two methods and concentrate on the third method in greater detail because in our experiments we have found that the third method is usually superior.

The first method uses the symbolic definition of the 2-surfaces. For a pair of parametric 2-surfaces $s_1, s_2 : \mathcal{R}^2 \rightarrow \mathcal{R}^4$, the intersection of them in the image space is expressed as three equations in four variables:

$$\varphi \circ s_1(u_1, v_1) = \varphi \circ s_2(u_2, v_2)$$

To represent self-intersections, the above equations should be adjoined by $(u_1 - u_2)^2 + (v_1 - v_2)^2 \neq 0$, which is equivalent to an additional equation with an additional variable:

$$\lambda[(u_1 - u_2)^2 + (v_1 - v_2)^2] - 1 = 0$$

For a pair of implicit 2-surfaces $f_1 = 0 \cap g_1 = 0$ and $f_2 = 0 \cap g_2 = 0$ where $f_1, f_2, g_1, g_2 : \mathcal{R}^4 \rightarrow \mathcal{R}$, the intersection in the image space is expressed as seven equations in eight variables:

$$f_1(x_1, y_1, z_1, w_1) = 0 \quad f_2(x_2, y_2, z_2, w_2) = 0$$
$$g_1(x_1, y_1, z_1, w_1) = 0 \quad g_2(x_2, y_2, z_2, w_2) = 0$$
$$\varphi(x_1, y_1, z_1, w_1) = \varphi(x_2, y_2, z_2, w_2)$$

The self-intersection problem can be tackled similarly. In contrast to the 3D case, one parametric 2-surface and one implicit 2-surface do not yield the simplest case. It still needs five equations in six variables:

$$f_1(x_1, y_1, z_1, w_1) = 0 \quad g_1(x_1, y_1, z_1, w_1) = 0$$
$$\varphi(x_1, y_1, z_1, w_1) = \varphi \circ s_1(u_1, v_1)$$

Any variable elimination methods could be used to solve the system of equations and obtain an intersection curve. But usually it is nontrivial to solve, especially when they are not restricted to polynomials. The curve can also be traced numerically in high dimensional space [BHHL88]. After the intersection curve is generated it should be mapped onto the piecewise linear approximation of the 2-surfaces. That is, the curve should have points exactly on the the edges of those polygons which are divided by the curve. This step can be done by slightly moving the points on the curve.

Since the 2-surfaces have already been polygonalized, an alternate method is to project the polygons into 3D image space and then find the piecewise linear intersection curve from the two sets of polygons. Starting at two intersecting polygons, one from each 2-surface, the adjacent polygons are searched and a new pair of intersecting polygons are formed. In this fashion the pair of polygons marches until a loop is detected or the boundary is reached, generating a piecewise linear curve exactly on the two sets of polygons.

This method works well only if the two 2-surfaces intersect transversally in the 3D image space. However, Corollary 2.3 tells that in the image space the isosurfaces intersect the silhouette surface tangentially. In such a case intersecting the piecewise linear approximation of the 2-surfaces is difficult. Instead of a smooth intersection curve, the method generates a lot of small loops scattered near the true intersection curve. Visibility displayed by trimming the polygons based on these small loops will be incorrect. Therefore, a third method that operates in 4-space is considered.

In the third method the active 2-surface S is given in the form of two equations $f = 0 \cap g = 0$ where f is the definition of the hypersurface. Another 2-surface R is given the piecewise linear approximation, which is readily available. We distinguish the intersection of $\varphi(S)$ and $\varphi(R)$ into *true* and *apparent* intersections. The true intersection is the projection of the intersection of S and R in 4-space. The apparent intersection is an intersection of $\varphi(S)$ and $\varphi(R)$ in the 3D image space to which there is no corresponding intersection in 4-space. Apparent intersection is an artifice of the projection.

In order to determine apparent intersections, we introduce partition functions h that map points of R to real numbers as follows. Let p be a point on R and q be the position of eye_4. Recall that S is defined by $f = 0 \cap g = 0$. Then a partition function is the function

$$h(p) = f((1 - \lambda_0)p + \lambda_0 q)$$

where λ_0 is the solution of $g((1 - \lambda)p + \lambda q)) = 0$. Note that the solutions λ_0 correspond to the points in which the line $\overline{pq}$ intersects the hypersurface g. Clearly if h changes sign as p ranges over a curve on R, then this curve passes a true or apparent intersection. We will discuss how to represent h after describing the algorithm for generating the intersection of $\varphi(S)$ and $\varphi(R)$ as follows:

Algorithm 4.4 Generating Intersection Curve on a 2-surface

input: A list of connected polygons which are the piecewise linear approximation of a 2-surface R in $\mathcal{R}^4$. A partition function $h : R \rightarrow \mathcal{R}$.

output: A list of connected points which is the piecewise linear approximation of the curve on R defined by $h = 0$.

(1) σ_0 = the first polygon with a transverse edge $\tau_0 = (v_0, v_1)$, i.e. $h(v_0)h(v_1) < 0$
(2) find the second transverse edge τ_1 of σ_0
(3) find zeros of h on τ_0 and τ_1 and construct a line segment connecting them
(4) $\sigma = \sigma_0$, $\tau = \tau_0$
(5) while τ is not a boundary edge of R and
(6) (σ = pivoting σ across τ) has not been visited do
(7) find the second transverse edge τ_1 of σ and construct a line segment
(8) $\tau = \tau_1$
(9) endwhile
(10) $\sigma = \sigma_0$, $\tau = \tau_1$
(11) redo steps (5) to (9)

When the intersection curve has more than one branch, the algorithm has to be applied to each of the branches.

4.3.2 Determining the Partition Function on a 2-Surface

When S is a silhouette surface and R is an isosurface, their images in 3D image space intersect tangentially. Their intersection constructed in 3D image space will often be incorrect due to roundoff errors. Working in 4-space using Algorithm 4.4, the partition function h can be chosen as the second equation for S, i.e. $h = \nabla f \cdot \mathbf{r}_4$. In $\mathcal{R}^4$ the hypersurface $h = 0$ intersects with R transversally, and so no numerical problems occur. The resulting trimming curve is smooth as shown in Figure 3.12 and 3.13.

For the silhouette surface there is another special problem that needs to be considered. Recall the discussion following Corollary 2.1 in Chapter 2. A 2-surface $f = 0 \cap g = 0$ usually has a 0-dimensional silhouette point set with respect to the first projection φ, determined by the equations $\nabla f \cdot \mathbf{r}_4 = 0$ and $\nabla g \cdot \mathbf{r}_4 = 0$. If the 2-surface is a silhouette surface, however, one of the equations is redundant and the zero set is a 1-dimensional curve. This curve signals visibility changes on the silhouette surface, just like a self-intersection curve would. To find the curve, Algorithm 4.4 can be applied. The 2-surface R being trimmed is the silhouetter surface and the partition function is $h = \nabla g \cdot \mathbf{r}_4 = \nabla(\nabla f \cdot \mathbf{r}_4) \cdot \mathbf{r}_4$. As example consider the Gauss distribution hypersurface shown in Figure 3.21 and 3.22. The silhouette surface is redrawn in Figure 4.4.

In the case that $\varphi(S)$ and $\varphi(R)$ have an apparent intersection curve, the definition of the partition function h is more complicated. The method is better understood by a 3D analogy. Figure 4.5 shows a surface $S: \ f(x,y,z) = x^2 + y - z^2 = 0$ with its boundary curves C_1 and C_2. Curve C_1 is in the plane $B_1: \ g_1(x,y,z) = y + 0.5 = 0$, and C_2 is in the plane $B_2: \ g_2(x,y,z) = z - 0.5 = 0$. Points a and b denote the true intersection of C_1 and C_2 in $\mathcal{R}^3$. Point c denotes their apparent intersection, in the 2D image space. In $\mathcal{R}^3$ the point c is actually on C_2, and another point c', exactly behind c in Figure 4.5(a), is on C_1 as can be seen in a different direction (b). As the point p moves on C_2, it occlude a corresponding point p' on the plane B_1, so forming a curve C_2' in B_1. When p passes through c, the function value $f(p')$ will pass through

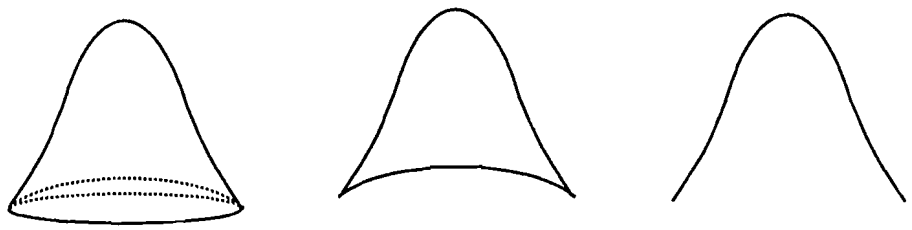

Figure 4.4 Silhouette Surface Before and After Visibility Determination

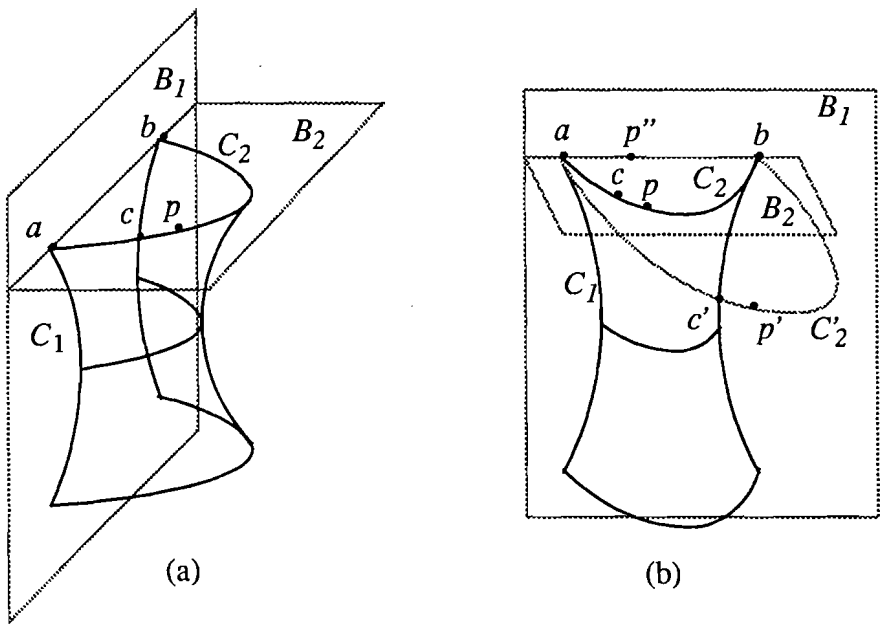

Figure 4.5 Intersection of Two Boundary Curves in Image Space

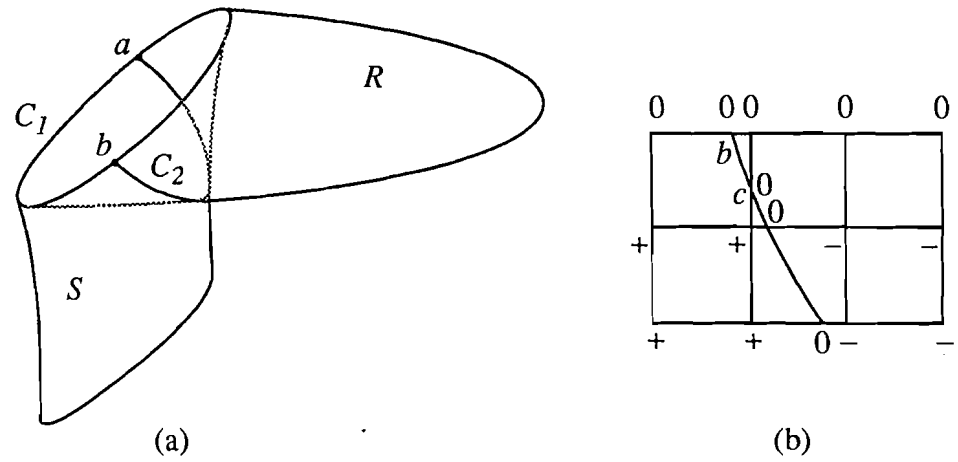

Figure 4.6 Intersection of Two Boundary Surfaces in Image Space

zero. Therefore the partition function on C_2 can be defined as $h(p) = f(p')$ such that $g(p') = 0$ and that p, p' and eye_3 are collinear. When g is linear, this constraint is easily solved and expressed as a linear function $p' = l(p)$. Hence $h(p) = f(l(p))$. The zero point of h can be the true intersection such as a and b, or the apparent intersection such as c. These 0-dimensional zero points are separated on C_2.

The method can be extended into $\mathcal{R}^4$. Again we use the hypersurface in Figure 3.12 and 3.13 as example. The hypersurface is $f(x, y, z, w) = x^2 + y^2 + z - w^2 = 0$. The active 2-surface S is defined as $f = 0$ and $g_1(x, y, z, w) = z + 0.5$ and the 2-surface R being trimmed is defined as $f = 0$ and $g_2(x, y, z, w) = w - 0.5$. The 2-surfaces S and R are redrawn in Figure 4.6. A linear function l maps a point p on R to a point p' on the hyperplane $g_1 = 0$. The partition function is then $h(p) = f(l(p))$. The zero points of h are 1-dimensional curves that could be true intersections of S and R in $\mathcal{R}^4$ shown as C_1 in Figure 4.6(a), or apparent intersections shown as C_2. The two curves C_1 and C_2 intersects at points a and b. They are the singular points of h and will cause numerical difficulties in Algorithm 4.4. Figure 4.6(b) shows some polygons near the intersection point b. The signs of h at each vertex of the polygons are marked.

Point b is on an edge where h is constant and equals to zero. Point c is on an edge with one zero and one nonzero h value at the two vertices, but the vertex with zero h value is not the desired point. Therefore, desingularization has to be considered when an edge has at least one vertex with zero h value.

There are several algebraic or analytic desingularization techniques [BHHL88, Hof89, OR87]. We use a technique based on geometric intuition. The basic idea is to divide h by a function h' which has zero value at the true intersection points of S and R, but has nonzero value at their apparent intersection points. The construction of h' is explained by the 3D analogy in Figure 4.5. The point p on curve C_2 is orthographically projected to plane B_1. Its image p'' on B_1 is called the *foot point* of p. Clearly there is another linear map l' such that $p'' = l'(p)$. We claim that the two linear maps l and l' cannot be the same. Suppose $l(p) \equiv l'(p)$, the projection by eye_3 must also be orthogonal to B_1. In such a case, the two curves C_1 and C_2 cannot have apparent intersection, and so Algorithm 4.4 need not be applied. Using the two maps l and l', the partition function on C_2 can be defined as $\frac{f(l(p))}{f(l'(p))}$, although it is not necessary because the true and apparent intersection points are separated.

The technique above extended into $\mathcal{R}^4$ works as follows. The partition function in Algorithm 4.4 is still $f(l(p))$. But steps (3), (4), (7) and (8) are modified. Whenever there is an edge with at least one zero function value, the desingularization procedure is called. At each vertex v of the edge, the partition function value is calculated via

$$\lim_{p \to v} \frac{f(l(p))}{f(l'(p))}$$

At a singular point it is an indeterminate form and l'Hôpital's Rule can be applied. The modified algorithm works well as shown in Figure 3.12. The visual effect is better than that produced by intersecting polygons in 3D image space.

When S has a nonlinear g, the map from R to hypersurface $g = 0$ cannot always be expressed explicitly. Nevertheless, the point p' can still be calculated numerically. Let p be the point on R and q be the position of eye_4. Then p' can be found by an iterative search for $g(p') = 0$ on the line $L: \; p' = \lambda p + (1 - \lambda)q$, say, using the secant

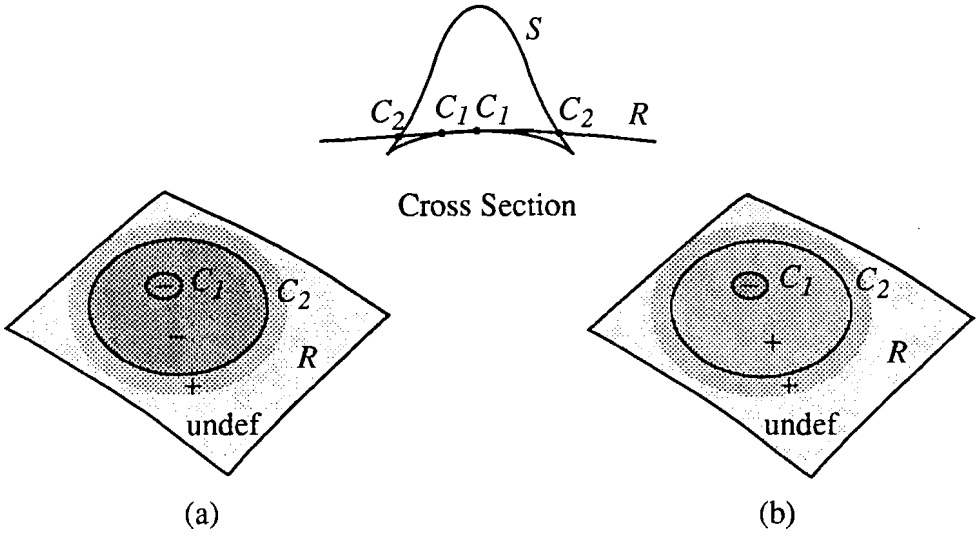

Figure 4.7 An Isosurface trimmed by the Silhouette Surface in Image Space

method. The partition function is defined as $h(p) = f(p')$. There are two problems to be addressed.

First, the line L may have no intersection with the hypersurface $g = 0$. In this case p' is undefined, and so is $h(p)$. In Algorithm 4.4 an edge is then considered as not transverse when at least one of its vertex has an undefined value of h. To make the algorithm work correctly the polygonalization of R should be fine enough so that the actual transversal edges do not fall into the region where h is undefined.

Second, the line L may have more than one intersection point with the hypersurface $g = 0$. To make h a continuous function over R, the parameter λ found by the iterative search should also be continuous over R. This can achieved by the homotopy continuation method [All90]. Suppose point p_1 has a known λ_1, to find the unknown λ_2 of point p_2 a curve smoothly connecting p_1 and p_2 is formed conceptually. By moving along the curve from p_1 to p_2, the result λ is smoothly changed from λ_1 at p_1 to λ_2 at p_2. In our algorithm λ is found by iteration. If the polygonization is fine enough, the initial value of λ can be taken from an adjacent vertex with known λ value. In this way the calculation of λ is propagated from a point to the whole R.

As an example see the hypersurface of Gauss distribution in Figure 3.21. The isosurface R with constant z value $z = 2$ is trimmed by the silhouette surface $S : f = 0 \cap g = 0$ where both f and g are nonlinear. The isosurface is redrawn in Figure 4.7. In the 3D image space R intersects with S on two circular curves C_1 and C_2. Curve C_1 is the true intersection of R and S while C_2 is their apparent intersection. The ray from a point p on R to eye_4 can intersect with $g = 0$ in 0, 1 or 2 points. Figure 4.7(a) and (b) show the sign of the $f(p')$ when p' is chosen on the front and back portions of $g = 0$, respectively. In the 3D image space the hypersurface $g = 0$ is a solid cylinder with its axis parallel to the projected world w-axis. Its intersection with $\varphi(R)$ is the region with defined partition function values.

4.3.3 Sight Number Calculation and Propagation

The trimming curves divide a 2-surface into several regions. For each region a point p is selected. To calculate the sight number at p, a ray from p to eye_4 is formed. It is then clipped by the domain of hypersurface being displayed. The domain is usually a hyperparallelogram. The number of intersection points of this clipped ray with the hypersurface is the desired sight number at p. In our display scheme, only those points with zero sight number are visible. That means there is no transparent hypersurface in $\mathcal{R}^4$. So the task is to find if there is any intersection, instead of how many intersections. Note that the second sight number must be calculated quantitatively since we need to display transparent surfaces in the 3D image space. This is a typical 3D graphics problem that we will not discuss.

Let p and q be the two end points of the line segment and $f = 0$ be the hypersurface. We want to find if the function $\bar{f}(\lambda) = f(p+\lambda(q-p))$ has zero points $\lambda \in (0,1)$. The behavior of the function $\bar{f}$ at $\lambda = 0$ is summarized in the following lemma.

Lemma 4.1 Suppose that p is a point on the hypersurface $\mathcal{M} : \ f(x,y,z,w) = 0$, and that the line segment $p + \lambda(q-p)$ is in the direction from p to the center of the projection $\varphi : \mathcal{R}^4 \to \mathcal{R}^3$. Then the function $\bar{f}(\lambda) = f(p + \lambda(q-p))$ has 0 as its

(a) zero point for all $p \in \mathcal{M}$;

(b) at least double zero point for all $p \in S$, the silhouette surface of $\mathcal{M}$ with respect to φ;

(c) at least triple zero point for all $p \in C$, the silhouette curve of S with respect to φ.

Proof:

(a) trivial.

(b) $\bar{f}'(0) = \bigtriangledown f(p) \cdot (q - p)$. It is zero if p is a silhouette point of $\mathcal{M}$ with respect to φ according to Theorem 2.3. Therefore 0 is at least a double zero point of $\bar{f}$.

(c) Let $g(p) = \bigtriangledown f(p) \cdot (q - p)$. The silhouette surface S can be defined as $f = 0 \cap g = 0$.

$$\begin{aligned} \bar{f}''(\lambda) &= \frac{d}{d\lambda}[\bar{f}'(\lambda)] \\ &= \frac{d}{d\lambda} g(p + \lambda(q - p)) \\ &= \bigtriangledown g(p + \lambda(q - p)) \cdot (q - p) \\ \bar{f}''(0) &= \bigtriangledown g(p) \cdot (q - p) \end{aligned}$$

By Theorem 2.3 $\bar{f}''(0) = 0$ if p is a silhouette point of S with respect to φ. □

Using this lemma we can analyze how the local behavior of $\bar{f}$ affects the sight number. Assume that p is on a 2-surface other than the silhouette surface. If p is in the vicinity of the true intersection curve with the silhouette surface, then $|\bar{f}'(0)|$ is almost 0, and the sign of its numerical value cannot be trusted. Figure 4.8(a) and (b) show two possible situations with $|\bar{f}'(0)| \approx 0$ and $\bar{f}''(0) \gg 0$. The sight number for (a) is greater than that for (b) by 1. But from the numerical values at point p the two situations cannot be distinguished. Therefore, the sight number should be calculated at a point away from the intersection with the silhouette surface, and then propagated to the entire region including those points close to the intersection curve. A criterion for selecting the point is thus $|\bar{f}'(0)| > \varepsilon_1$. Once we have eliminated the possibility that a zero point is very close to but not exactly at $\lambda = 0$, the sight number can be found by numerical methods such as linear search or interval subdivision.

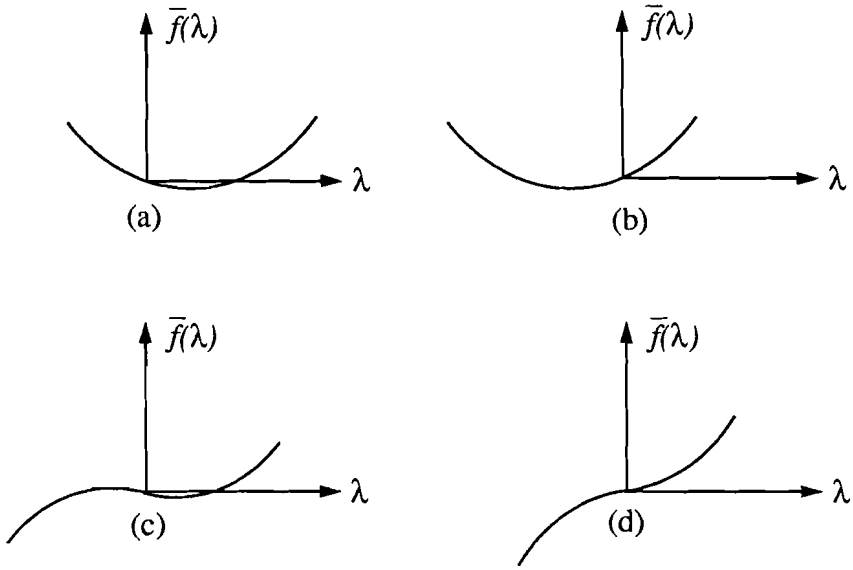

Figure 4.8 Function $\bar{f}(\lambda)$ near $\lambda = 0$

Assume that p is on a silhouette surface. By Lemma 4.1, $\bar{f}'(0)$ is always zero. Although the numerical value of $\bar{f}'(0)$ can be of small magnitude, the ambiguity shown in Figure 4.8(a) and (b) cannot happen for silhouette surface. This is because by the definition of silhouette surface the situation (a) can be eliminated. If p is close to the silhouette curve, however, ambiguities may occur as shown in Figure 4.8(c) and (d), where $\bar{f}'(0) \approx 0$, $\bar{f}''(0) \approx 0$ and $\bar{f}'''(0) \gg 0$. A Similarly, a criterion for selecting the point is that $|\bar{f}''(0)| > \varepsilon_2$. No extra calculation of the second order derivatives is needed, because $\bar{f}''(0) = \nabla g(p) \cdot (q - p)$. That is the partition function values used to construction the silhouette curve.

5. APPLICATIONS

Visualization of high dimensional space has found wide spread applications in science and engineering, as listed in Chapter 1. The visualization of four dimensional space is more suitable for the problems naturally defined in 4-space, such as 3D objects in motion, and 3D scalar fields. For a problem defined in arbitrary n-dimensional space, to get intuition by visualization, the usual way is to examine the problem of reduced dimension in 3-space or 2-space. But sometimes when the problem is reduced to 3-space or 2-space it becomes a trivial one. Then the visualization of 4-space can serves as a bridge from the "trivial cases" to the "nontrivial case". Such an example already appeared in Chapter 4, where we discussed and displayed the singularity on the intersection curve in image space of two adjacent boundary surfaces of a hypersurface in 4-space. This kind of singularity disappears when the dimension is reduced.

Several examples will be presented in the following sections that illustrate the applications of our 4D visualization system. The first section is about understanding differential geometry by two examples of planar offset curves. Although offset curve is a problem in 2-space, by the envelope theorem it can be defined in higher dimensional space. Some comparisons will be drawn between 2-, 3- and 4-spaces. The second section shows the intersection 2-surface of two moving objects in 3-space. Although the moving objects are simple, their intersection surface in 4-space is surprisingly complicated. A 3D scalar field, the electron density field of a virus, is shown in the third section. By putting it in 4-space, the concepts of contouring and slicing are uniformized. The 3D scalar field can be manipulated in several ways by simply moving eye_4. The final example is the tool path generation for 5-axes milling machines. The original problem is defined in 5-space, and can be treated by algorithms working in

5-space. We put emphasis on intuitions by visualization. The problem is simplified and illustrated in 4-space.

5.1 Understanding Differential Geometry

In Chapter 2 the relationship between silhouette and envelope was discussed. In this section it is explained and illustrated by the example of offset curves.

Given a curve $f(x,y) = 0$ in $\mathcal{R}^2$, its offset curve by distance $r > 0$ can be formulated by the envelope method [FN90, Hof89] as a set of equations:

$$\begin{aligned} g: \quad (x-u)^2 + (y-v)^2 - r^2 &= 0 \\ f(u,v) &= 0 \\ C: \quad \nabla_{uv} g \cdot \bar{\mathrm{t}} &= 0 \end{aligned}$$

where

$$\begin{aligned} \nabla_{uv} g &= (\frac{\partial g}{\partial u}, \frac{\partial g}{\partial v})^T \\ \bar{\mathrm{t}} &= (\frac{\partial f}{\partial v}, -\frac{\partial f}{\partial u})^T \end{aligned}$$

If the parametric form of the curve f is available, the set of equations can be simplified as:

$$\begin{aligned} h: \quad (x-u(t))^2 + (y-v(t))^2 - r^2 &= 0 \\ C': \quad (x-u(t))u'(t) + (y-v(t))v'(t) &= 0 \end{aligned}$$

Note that the condition C' is equivalent to $\frac{\partial h}{\partial t} = 0$. If the greatest common divisor $\phi(t) = \mathrm{GCD}(u'(t), v'(t))$ is not a constant, the condition C' can be further simplified as [FN90]:

$$C'': \quad (x-u(t))p(t) + (y-v(t))q(t) = 0$$

where

$$p(t) = \frac{u'(t)}{\phi(t)}, \quad q(t) = \frac{v'(t)}{\phi(t)}$$

An implicit equation for the offset curve can be determined by the resultant method [FN90], or using Gröbner bases [Hof89]. The offset curve can also be traced numerically in $\mathcal{R}^4$ or $\mathcal{R}^3$ by the method described in [BHHL88].

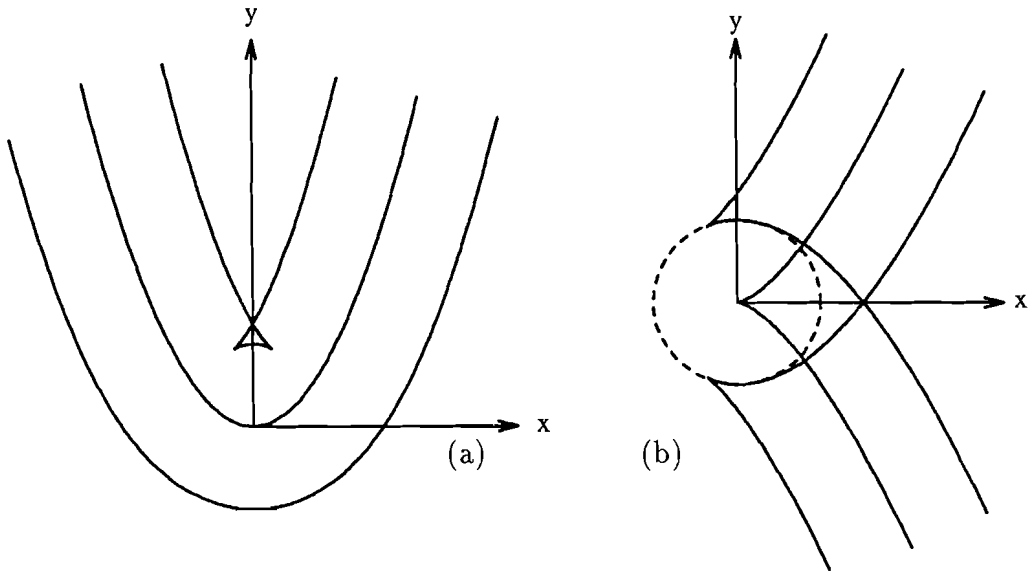

Figure 5.1 Curves (a) $y - x^2 = 0$ and (b) $y^2 - x^3 = 0$ and their offset curves by 1

It is important to note the following points about the envelope method for formulating offsets:

1. The offset curve may have cusps and/or self-intersections in (x, y)-plane (see Figure 5.1(a)). But the singularities often disappear when the curve is traced in higher dimensional space.

2. The equations may describe additional points which have a distance r from the singular points on the curve f (see Figure 5.1(b)).

We now explain these phenomena by means of 2-surface visualization. The equations $g = 0$ and $f = 0$ are two 3-surfaces in (x, y, u, v)-space and their intersection S is a 2-surface. Moreover, at the point $\mathbf{p} = (x, y, u, v)$ on S, the two normals are:

$$\begin{aligned} \mathbf{n_1} = \nabla g(\mathbf{p}) &= (2(x-u), 2(y-v), -2(x-u), -2(y-v))^T \\ \mathbf{n_2} = \nabla f(\mathbf{p}) &= (0, 0, \frac{\partial f}{\partial u}, \frac{\partial f}{\partial v})^T \end{aligned}$$

They are linearly independent as long as $\mathbf{n_2}$ is a nonzero vector since $(x - u)$ and $(y - v)$ cannot be both zero. The condition C can be rewritten as $\det(\mathbf{i}, \mathbf{j}, \mathbf{n_1}, \mathbf{n_2})$

$= 0$. If $\mathbf{p}$ is a nonsingular point on S, by Corollary 2.1 it is a silhouette point with respect to an orthographic projection with two centers along the u- and v-axes. The silhouette points form a curve on the tubular surface S in $\mathcal{R}^4$. In Figure 5.2 and 5.3 we show the 2-surface S and the silhouette curve corresponding to the offset curve in Figure 5.1(a). In $\mathcal{R}^4$ the curve is smooth without cusps or self-intersections as we can see in Figure 5.3 from a different viewing direction.

On the other hand, if $\mathbf{p}$ is a singular point, then $\mathbf{n_1}$ and $\mathbf{n_2}$ are linearly dependent, and so $\mathbf{n_2}$ must be a zero vector. Surely condition C is satisfied, but according to our definition $\mathbf{p}$ cannot be a silhouette point. These singular points are exactly the additional points described above as the second phenomenon. In Figure 5.4 and 5.5 the 2-surface S and the silhouette curve corresponding to the offset curve in Figure 5.1(b) are shown from different viewing directions. The silhouette curves are still smooth without cusps or self-intersections. But the 2-surface is not a smooth tube. The singular points form a circle corresponding to the dashed circle in Figure 5.1(b).

If the curve f has a parametric form, the offset curve can be traced in (x, y, t)-space. The two equations $h = 0$ and $\frac{\partial h}{\partial t} = 0$ are two surfaces and their intersection is a curve. Note that in this case $\frac{\partial h}{\partial t} = 0$ is equivalent to $\bigtriangledown h \cdot \mathbf{k} = 0$. This means that the intersection curve is the silhouette on the surface h with respect to an orthographic projection along the t-axis. But the surface $h = 0$ is smooth without any singular points because $\frac{\partial h}{\partial x}$ and $\frac{\partial h}{\partial y}$ cannot be zero simultaneously and so $\bigtriangledown h$ is always a nonzero vector. The dashed circle in Figure 5.1(b) is actually another branch of the silhouette curve as shown in Figure 5.6 and 5.7.

If the greatest common divisor $\phi(t)$ is not a constant, the condition $\bigtriangledown h \cdot \mathbf{k} = 0$ is equivalent to:

$$\phi(t)[(x - u(t))p(t) + (y - v(t))q(t)] = 0$$

The factor $\phi(t) = 0$ represents those silhouette curve branches that are circles resulting from intersecting the tubular surface $h = 0$ with the planes $t = t_i$ perpendicular to the t-axis, where t_i's are the zeros of $\phi(t)$. The other factor is the same as condition C'', and represents the silhouette curve branches corresponding to the offset curve.

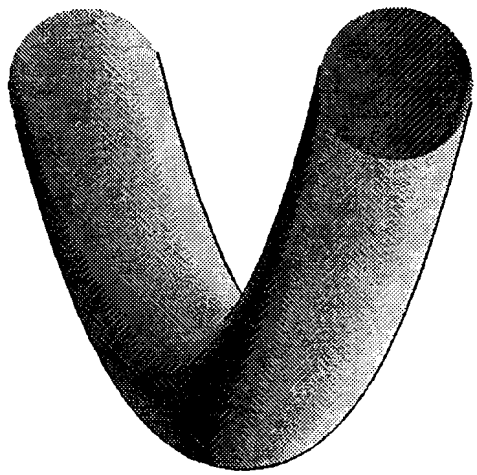

Figure 5.2 The Offset Curve (a) traced in $\mathcal{R}^4$ viewd from $\theta = (0, 0, 0, 0, 0, 0)$

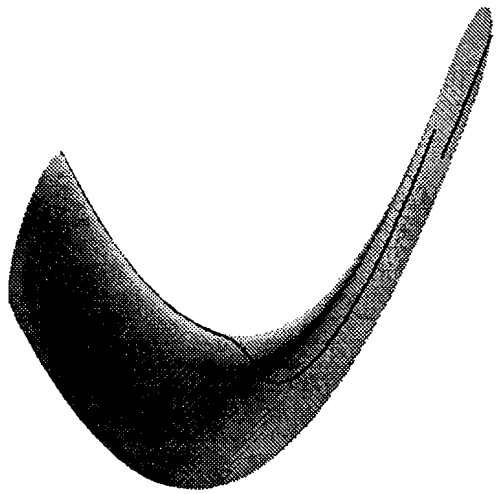

Figure 5.3 The Offset Curve (a) traced in $\mathcal{R}^4$ viewd from $\theta = (45, 105, 45, 75, 165, 0)$

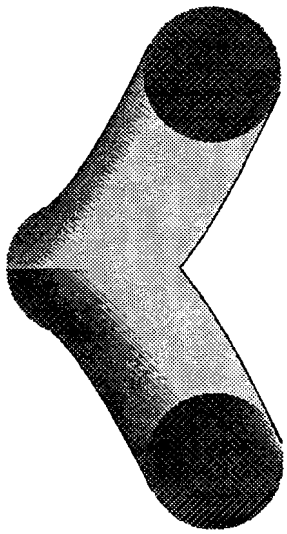

Figure 5.4 The Offset Curve (b) traced in $\mathcal{R}^4$ viewd from $\theta = (0, 0, 0, 0, 0, 0)$

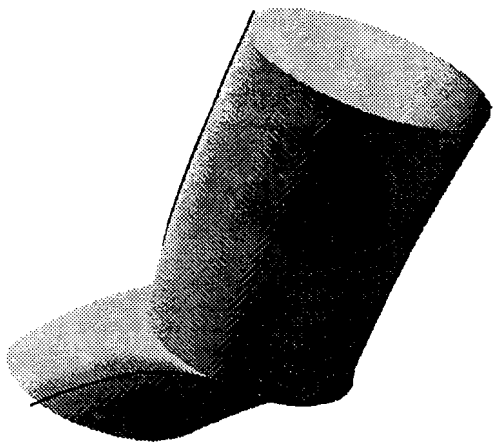

Figure 5.5 The Offset Curve (b) traced in $\mathcal{R}^4$ viewd from $\theta = (45, 40, 60, 105, 75, 0)$

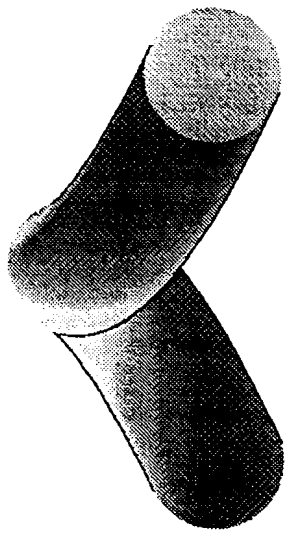

Figure 5.6 The Offset Curve (b) traced in $\mathcal{R}^3$ viewd from $\theta = (0.0, 0, 0, 0, 0, 0)$

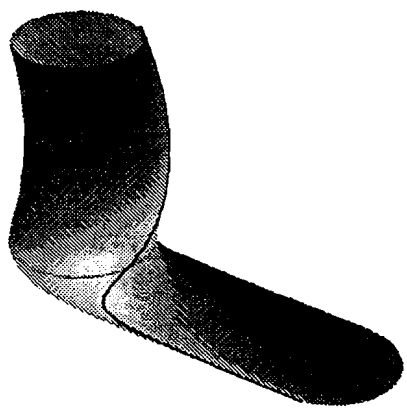

Figure 5.7 The Offset Curve (b) traced in $\mathcal{R}^3$ viewd from $\theta = (0, 0.0, -40, 60.0)$

5.2 Collision Detection and Analysis

The advantage of using 4D geometry to deal with the collision detection problem has been explained in [Cam84, Ros89]. The basic idea expressed by Cameron is that "if an object can be represented by a set-combination in a CSG scheme, and the primitive objects can be extruded (into 4-space) in this scheme, then the extrusion of the object is the set-combination of the extrusions of the primitives" [Cam84]. The extruded object means the object in motion considered in (x, y, z, t)-space. Two moving objects collide if and only if the intersection of their extrusions is nonempty. The problem is then reduced to testing the intersection of each pair of primitive extrusions.

3D objects are bounded by surfaces in 3-space. The extrusion of such a surface is a hypersurface in 4-space. The intersection of two hypersurfaces is a 2-surface $S : f(x, y, z, t) = 0 \cap g(x, y, z, t) = 0$, and can be examined by our system. When the 2-surface is nonempty, in applications such as physical objects simulation or robotics motion planning it is necessary to find the *initial colliding point*, i.e. the point $\mathbf{p}$ on the 2-surface with the smallest value of t. Assuming $\mathbf{p}$ is a nonsingular point of the 2-surface S, the natural projections of the two normals at $\mathbf{p}$ into (x, y, z)-subspace are parallel.

$$C: \qquad \frac{\partial f/\partial x}{\partial g/\partial x} = \frac{\partial f/\partial y}{\partial g/\partial y} = \frac{\partial f/\partial z}{\partial g/\partial z}$$

Together with $f = 0$ and $g = 0$ this condition determines a zero-dimensional solution set on S. To solve the nonlinear equations describing these points, one may relax the condition and use numerical methods such as curve tracing. Condition C can be rewritten as:

$$C' : (\forall \mathbf{n} = (n_x, n_y, n_z, n_t)^T) \begin{vmatrix} 0 & 0 & 0 & 1 \\ n_x & n_y & n_z & n_t \\ \partial f/\partial x & \partial f/\partial y & \partial f/\partial z & \partial f/\partial t \\ \partial g/\partial x & \partial g/\partial y & \partial g/\partial z & \partial g/\partial t \end{vmatrix} = 0$$

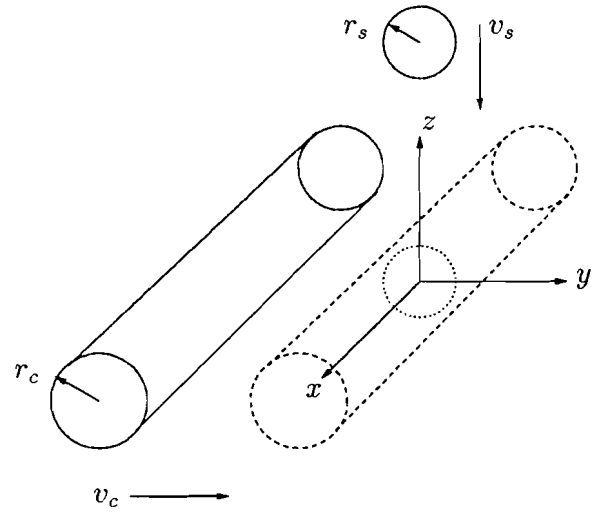

Figure 5.8 A cylinder and a sphere in motion

By Corollary 2.1, **p** must be on the silhouette curve of the 2-surface with respect to any orthographic projection with two centers both inside (x, y, z)-subspace. The point on the silhouette curve with the smallest t is the initial colliding point.

Consider a cylinder of radius r_c about the x-axis moving in the positive y-direction at a constant speed v_c, and a sphere of radius r_s moving in the negative z-direction at a constant speed v_s. At the time $t = 0$ both are at the origin as shown by the dashed cylinder and sphere in Figure 5.8. Their extrusions into 4-space are the two hypersurfaces defined as:

$$\begin{aligned} (y - v_c t)^2 + z^2 - r_c^2 &= 0 \\ x^2 + y^2 + (z + v_s t)^2 - r_s^2 &= 0 \end{aligned}$$

The intersection 2-surfaces in different cases are displayed in Figure 5.9 through 5.11. The curves on the 2-surface are the silhouette curve branches with respect to the orthographic projection with two centers along the x-, y- or z-axes but now seen from different viewpoints.

Figure 5.9 shows the case where the cylinder and the sphere have the same radius, $r_c = r_s = 1$, $v_c = 0$, $v_s = 1$. Since eye_4's position is just a little off the the t-axis,

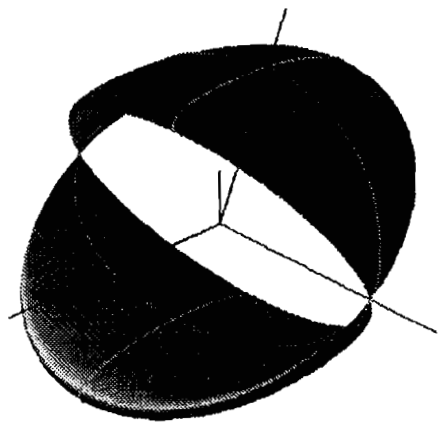

Figure 5.9 Intersection of a cylinder and a moving sphere with the same radius, viewed from $\theta = (0, 18, 9, 120, 75, 0)$

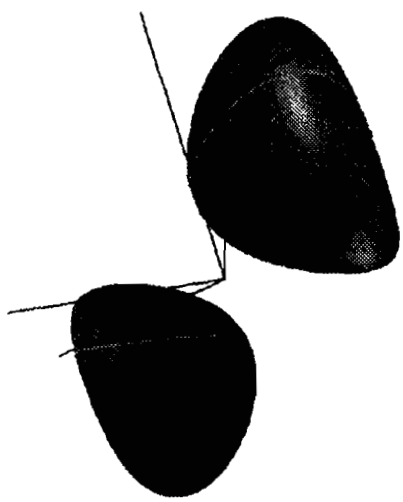

Figure 5.10 The cylinder has a larger radius, viewed from $\theta = (0, 30, 105, -105, 30, 0)$

the 2-surface resembles the sweep of the intersecting curve in (x, y, z)-subspace. The two singular points are due to the fact $r_c = r_s$, and will be present no matter how eye_4 rotates in 4-space. Figure 5.10 shows the case where the cylinder has larger radius, $r_c = 1$, $r_s = 0.7$, $v_c = 0.2$, $v_s = 1$. Note that the 2-surface has two separate components. This is because at $t = 0$ the sphere is totally inside the cylinder. If the curve tracing algorithm starts at a point on the component with larger t values, it will end up with an incorrect answer. Figure 5.11 shows the case where the sphere has larger radius, $r_c = 1$, $r_s = 1.2$, $v_c = 0$, $v_s = 1$. Note that although the 2-surface is connected, the silhouette curve branches can be separated. It has a "hole" due to the fact that at a certain time period the sphere and the cylinder intersect in a curve with more than one branch.

Even for such simple objects as cylinder and sphere, the intersection 2-surface in 4-space could be fairly complicated. When numerical method is used to search for the initial colliding point, the phenomena mentioned above have to be considered.

5.3 Scalar Fields in 3-Space

A 3D scalar field is a function defined on a subset of $\mathcal{R}^3$: $w = f(x, y, z)$. Displaying 3D scalar field is very common in science and engineering, such as the material density of a nonuniform solid, the temperature or pressure distribution in the atmosphere, and the electron density data of virus. The basic techniques used for displaying 3D scalar fields are volume rendering [Sab88, ST90, UK88], contouring [GN89, LC87], and slicing [SK90].

Put into 4-space, the 3D scalar field is a hypersurface defined by $w - f(x, y, z) = 0$. Conceptually there is no difference between contouring and slicing.[1] Both of them are considered as a 2-surface obtained by intersecting the hypersurface with a hyperplane. If the concept is extended further, the hyperplane can be replaced by curved hypersurfaces. Moreover, by rotation in 4-space, the extreme value of f and

[1] Note that the implementation could be different so that the most efficient algorithm can be used for each special case.

their distributions can be displayed by their relative spatial positions, in addition to the conventional color scales. When eye_4's direction is not orthogonal to the (x, y, z)-subspace, or its distance to the origin is finite, the shape of the 3D image is different from what we are used to see. Since this is an interactive system, we usually start at a normal view of the 3D scalar field, i.e., with eye_4 at infinity along the direction of w-axis. Then eye_4 moves in a series steps of small rotation and/or zooming. The animation helps keeping track of how the 3D image changes its shape in response to the eye_4 motion. The ideas are explained by an example of the electron density data of virus.

Some research in microbiology processes electron density field in complex molecules such as virus. The virus is first crystallized, then measurements are taken, for example, at CHESS (Cornell High-Energy Synchroton Source), followed by a Fourier analysis that quantifies the densities. The resulting data are essentially a scalar field in 3-space. In this example the graphics data is obtained by interpolating the electron density values that have been measured at the 8 million nodes of a $200 \times 200 \times 200$ grid and ranging from -14161 to 17880. Figure 5.13 shows the contour surfaces $w = 12000$ and $w = -12000$, and the slicing surface $x^2 + y^2 + z^2 = 1$. The actual radius of the sphere is 120Å. The contour surfaces look like dust particles floating around the sphere. On a computer screen the contour surfaces of level 12000 are painted in orange and those of level -12000 are painted in blue. In monochrome pictures they can be distinguished by the size. Those of level 12000 are larger than those of level -12000. This is because of the density distribution of this virus, which will be discussed shortly. Another way to distinguish contour surfaces of different levels is to move eye_4 from infinity toward the origin. As the reciprocal distance of eye_4 to the origin is changing from 0 to positive, the contour surfaces of positive level will move away from the origin while those of negative level will move towards the origin. The picture also shows a symmetric structure of the virus. The contour surfaces are clustered around the 60 vertices of a snub dodecahedron (see Figure 5.12). The snub dodecahedron is an Archimedean polyhedron, which means that every face

is a regular polygon, though the faces are not all of the same kind [CR61]. In the case of the snub dodecahedron, there are 12 pentagonal faces and 80 triangular faces.

Figure 5.14 shows the contour surfaces of level 12000 and the same spherical slicing surface within a subcube, covering four triangular faces of the snub dodecahedron. By rotating eye_3 it can be seen that the slicing surface passes through all of the six contour surfaces. The density value on the slicing surface can be shown in color scale, and the locations of the contour surfaces match the areas where the density values are high. Here we use the 4D rotation to show their relationship. When eye_4 rotates in the (z, w)-plane the vertical direction becomes the extra density axis, and so the density values are seen from the height of points lifted from the original slicing surface. See Figure 5.15. Notice that the contour surfaces stick to the slicing surface at their original locations. If eye_4 rotates by $\frac{\pi}{2}$ so that it is inside the (x, y, z)-subspace, a side view is obtained. See Figure 5.16. Now we can see the distribution of the 3D scalar field on the slicing surface. It is clearly observed that most density values are in the range of -6000 to 6000. There are six positive peak values (two of them are close to each other) and twelve negative peak values. The positive peaks are higher in the density direction and larger in the (x, y, z)-subspace than the negative peaks. This explains why there are more but smaller contour surfaces of level -12000 than those of level 12000 in Figure 5.13. It is also clear that we cannot find a negative value generating contour surfaces that match in number and size the level 12000 contour surfaces.

5.4 Generalized Offset Curves

5.4.1 Motivation

When machining curved surface by CAM system, ball-mills and end-mills can be used.[2] A comparison can be found in [VQ89]. Using an end-mill has the advantage

[2]They are also called ball-ended mills and flat-ended mills, respectively. Besides, there are toroidal cutters, also known as filleted mills and corner-radius mill.

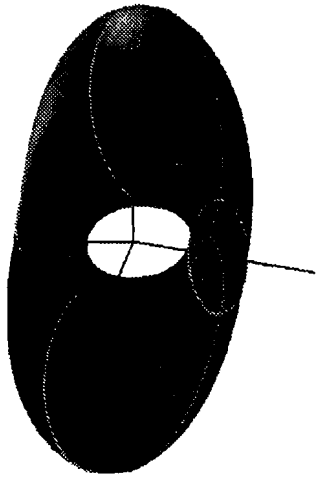

Figure 5.11 The cylinder has a smaller radius, viewed from $\theta = (0, -15, 123, 120, 90, 0)$

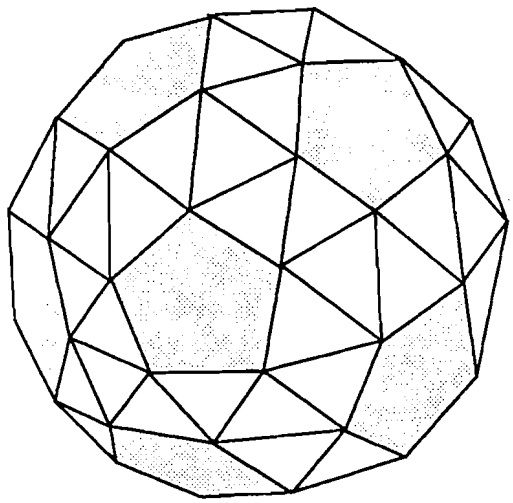

Figure 5.12 A snub dodecahedron

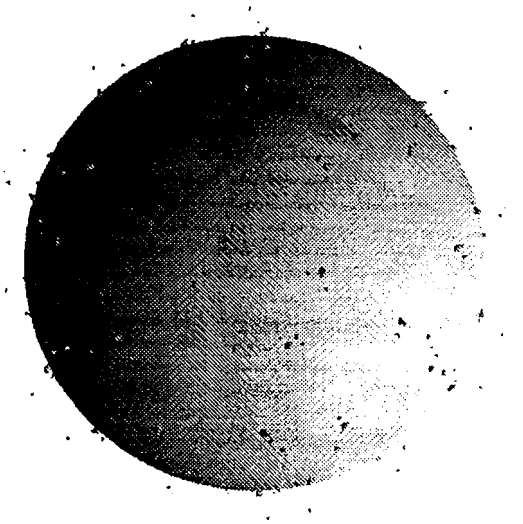

Figure 5.13 Contour and slicing surfaces show the structure of a virus

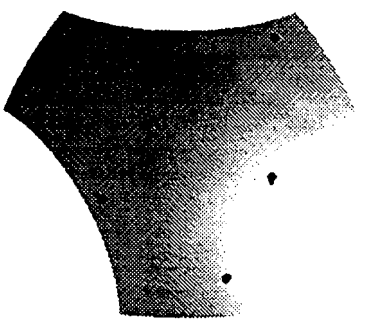

Figure 5.14 Contour and slicing surfaces within a subcube

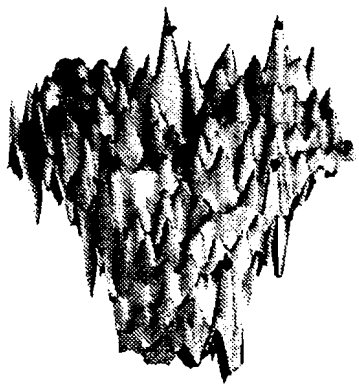

Figure 5.15 The contour and slicing surfaces viewed from $\theta = (0, 0, 15, 315, 60, 0)$

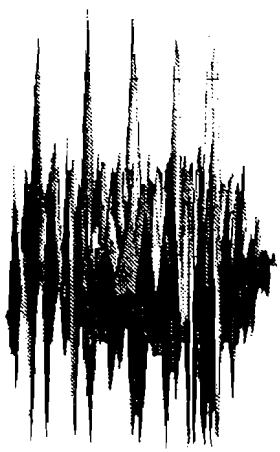

Figure 5.16 The contour and slicing surfaces viewed from $\theta = (0, 0, 90, 315, 90, 0)$

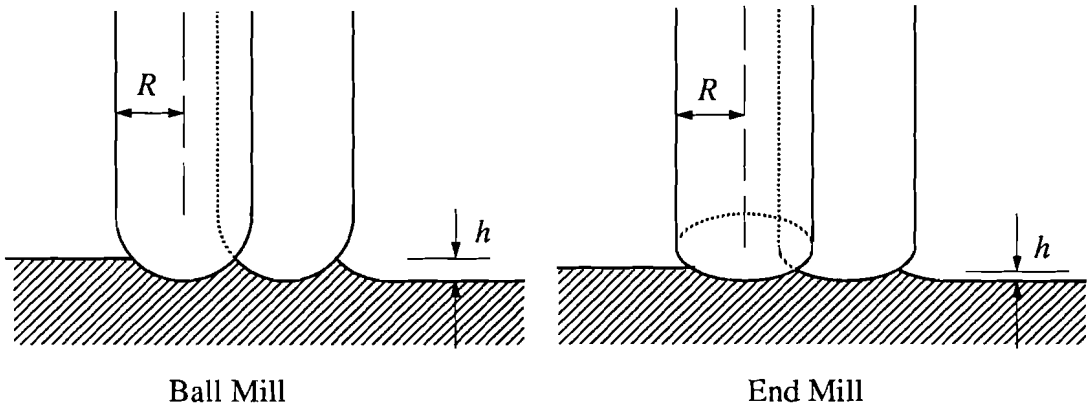

Figure 5.17 A comparison between two cutters

of smaller surface roughness and faster cutting speed. See Figure 5.17. The ball-mill has a low speed cutting region at its bottom, and the processed surface has a greater cusp height h. However, the control of end-mill is much more complicated than that of ball-mill. The tool path of a ball-mill is on the offset surface, and so it is controlled by three coordinates (x, y, z) on a 3-axis machine. To control an end-mill, in addition to the position coordinates, two angles are necessary for the orientation of the tool. So the end-mill is usually controlled by a 5-axis machine. Theoretically the five independent parameters form a 5-space and the possible motion of an end-mill is on a manifold in 5-space, called the *generalized offset surface.* But so far it has not been discussed in the literature in depth. A typical way to investigate a problem in high dimensional space is to study its dimension-reduced version first. Farouki [Far86] gave an example in 3-space with the restriction that the orientation of the tool is fixed. With some different restrictions, we reduce the problem into a *generalized offset curve* in 4-space, and visualize it by our system. The definitions and properties of the generalized offset curve will be discussed in the following subsections. Here we first explain and justify the restrictions.

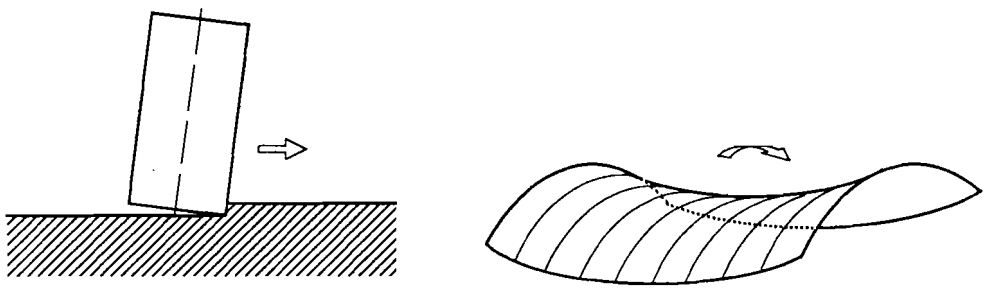

Figure 5.18 Cutter moving direction

In [Mar87] it was shown that to obtain the widest machining strips, the contact point of the cutter on the surface should move along a curvature line with minimum normal curvature. See Figure 5.18. Here the outward normal of the surface is used. So a negative curvature means convexity with respect to the cutter. When the minimum normal curvature of the surface is always nonpositive, or the radius of the end-mill is small enough, it is possible for the end-mill to follow the surface exactly along its moving direction. In such cases, more attention is paid on the plane orthogonal to the moving direction.

The projection of the cutting edge of an end-mill in the plane orthogonal to the moving direction is an ellipse with major radius a, minor radius b and rotated by an angle θ. The relationship between (a, b, θ) and the orientation of the end-mill depends on how the coordinate systems are specified and rotated. In any case, a is always equal to the radius R of the end-mill. Sasaki [Sas90] proposed an algorithm to fit the ellipse according to the curvature or the rate of change of the curvature. However, the *interference problem* needs to be solved.

The term interference refers to the phenomenon when the surface is overcut by the tool. In the case of the ball-mill, the interference can be detected from the self-intersection of the offset surface by analytical or algebraic methods [AU90, FN90]. Then the offset surface must be trimmed. However, the resulting tool path will cause an undercut. If the undercut is not tolerable, a subsequent process by a smaller radius ball-mill is necessary [CJ89]. On the other hand, the effective curvature of an end-mill can be adjusted during the process to match the curvature of the surface. With two more degrees of freedom, the interference is very hard to calculate. It is often detected by simulation [Sas90, MYGP90]. In this research the generalized offset curve is used for matching, in the plane orthogonal to the moving direction, the effective curvature of the projected cutting edge with the curvature of the intersection curve of the surface and the plane while avoiding interference.

5.4.2 Definitions

An ellipse in $\mathcal{R}^2$ is expressed as $x^TQx = 1$ where Q is a symmetric positive definite matrix. The two eigenvalues of Q are denoted by $\frac{1}{a^2}$ and $\frac{1}{b^2}$ satisfying $0 < \frac{1}{a^2} \leq \frac{1}{b^2}$.

The *rotation operator by 90 degree with respect to* Q is a matrix J defined by

$$J = abR(\frac{\pi}{2})Q$$

where $R(\cdot)$ is a matrix representing rotation in $\mathcal{R}^2$.

It is easy to verify that $x^TQJx = 0$, $J^TQJ = Q$, and $JJx = -x$.

Let $\alpha(t) = (\alpha_1(t), \alpha_2(t))^T$ be a regular curve $I \rightarrow \mathcal{R}^2$, called the *generator curve.* The *Q-offset curve* of α is a curve β defined by

$$\beta(t) = \alpha(t) + \frac{J\alpha'(t)}{\|\alpha'(t)\|_Q}$$

where

$$\|\alpha'(t)\|_Q = \sqrt{\alpha'(t)^TQ\alpha'(t)}$$

Each generator curve has two Q-offset curves. Together they satisfy the envelope equations:

$$(\beta - \alpha(t))^TQ(\beta - \alpha(t)) \;\; = \;\; 1$$

$$(\beta - \alpha(t))^T Q \alpha'(t) \;=\; 0$$

The *Q-curvature* of curve α is defined as

$$\kappa_Q^\alpha(t) = \frac{\alpha''(t)^T Q J \alpha'(t)}{\|\alpha'(t)\|_Q^3}$$

The superscript α will be dropped if it is clear from the context.

An *interference point* is defined as $\tau \in I$ such that

$$\exists t \in I, \;\; \|\beta(\tau) - \alpha(t)\|_Q < 1$$

An *interference segment* is defined as the open interval (τ_1, τ_2) of interference points, where τ_1 and τ_2 are not interference points. They are called the *endpoints of the interference segment.*

When the generator curve $\alpha(t)$ at τ satisfies $\kappa_Q(\tau) = 1$ and $\kappa_Q'(\tau) \neq 0$, the point on the Q-offset curve, $\beta(\tau)$, is a *cusp.* When $\kappa_Q(\tau) = 1$, $\kappa_Q'(\tau) = 0$ and $\kappa_Q''(\tau) \neq 0$, $\beta(\tau)$ is an *extraordinary point.* When $\kappa_Q'(\tau) = \kappa_Q''(\tau) = 0$, These definitions can be extended using higher order derivatives. The situations are essentially the same, but they will not be further addressed to simplify discussion. When the Q-offset curve satisfies $\beta(\tau_1) = \beta(\tau_2)$ and $\tau_1 \neq \tau_2$, $\beta(\tau_1) = \beta(\tau_2)$ is a *self-intersection point.*

In Figures 5.19 through 5.21, three Q-offset curves of (t, t^2) with different b values are shown together with their Q-curvatures. An extraordinary point occurs when $b = 0.086$ as shown in Figure 5.20. When $b < 0.086$ there is a self-intersection point as shown in Figure 5.21.

5.4.3 Properties

In the following discussion of some properties of Q-offset curves, a few conditions are always assumed. The regular generator curve α has no self-intersection points. The Q-offset curve to be considered is totally on one side of α. There is no global interference such as that caused by the narrow neck of a bottle-shaped surface. For NC tool path generation, these conditions are usually satisfied.

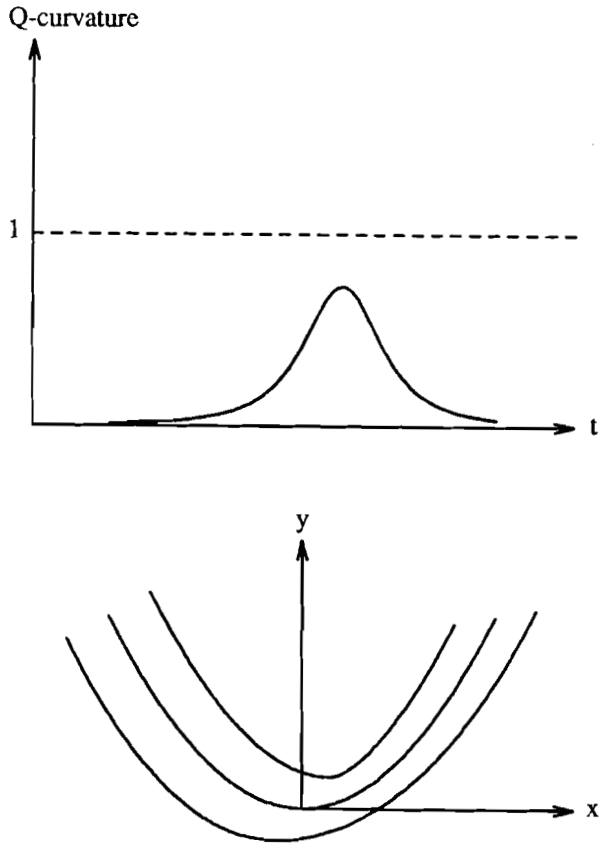

Figure 5.19 Q-offset curve of $(t, t^2), a = 0.25, b = 0.125, \theta = 30$

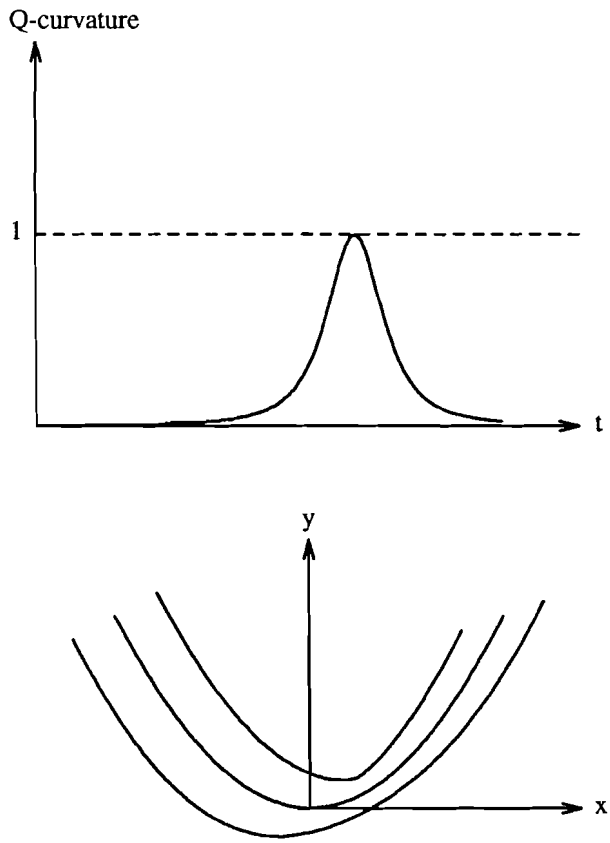

Figure 5.20 Q-offset curve of $(t, t^2), a = 0.25, b = 0.086, \theta = 30$

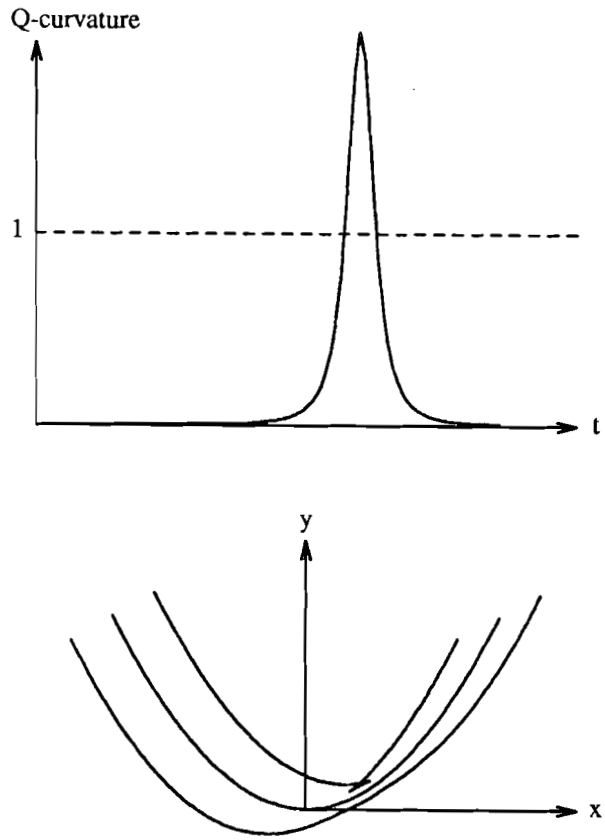

Figure 5.21 Q-offset curve of $(t, t^2), a = 0.25, b = 0.04, \theta = 30$

We first discuss the Q-offset curve with a fixed Q. When Q is the identity matrix, the results are the same as those of ordinary offset curve. We prove them for an arbitrary Q in order to discuss the *family of Q-offset curves* with b and θ as the parameters of the family.

Lemma 5.1 The tangent vectors and the Q-curvatures of the generator $\alpha(t)$ and Q-offset $\beta(t)$ are related as:

$$\begin{aligned} \beta'(t) &= (1 - \kappa_Q^\alpha(t))\alpha'(t) \\ \kappa_Q^\beta(t) &= \frac{\kappa_Q^\alpha(t)}{|1 - \kappa_Q^\alpha(t)|} \end{aligned}$$

Proof: Recall the definition of $\beta(t)$:

$$\beta(t) = \alpha(t) + \frac{J\alpha'(t)}{\|\alpha'(t)\|_Q}$$

Differentiating $\beta(t)$ yields:

$$\beta'(t) = \alpha'(t) + \frac{J\alpha''(t)}{\|\alpha'(t)\|_Q} - \frac{J\alpha'(t)(\alpha''(t)^T Q\alpha'(t))}{\|\alpha'(t)\|_Q^3}$$

$\beta'(t)$ is parallel to $\alpha'(t)$ because:

$$\beta'(t)^T QJ\alpha'(t) = \alpha'(t)^T QJ\alpha'(t) + \frac{\alpha''(t)^T J^T QJ\alpha'(t)}{\|\alpha'(t)\|_Q} - \frac{\|J\alpha'(t)\|_Q^2(\alpha''(t)^T Q\alpha'(t))}{\|\alpha'(t)\|_Q^3} = 0$$

Therefore, the tangent vector of β is:

$$\begin{aligned} \beta'(t) &= \beta'(t)^T Q\alpha'(t)\frac{\alpha'(t)}{\|\alpha'(t)\|_Q^2} \\ &= (\|\alpha'(t)\|_Q^2 + \frac{\alpha''(t)^T J^T Q\alpha'(t)}{\|\alpha'(t)\|_Q} - \frac{\alpha'(t)^T J^T Q\alpha'(t)(\alpha''(t)^T Q\alpha'(t))}{\|\alpha'(t)\|_Q^3})\frac{\alpha'(t)}{\|\alpha'(t)\|_Q^2} \\ &= (1 - \kappa_Q^\alpha(t))\alpha'(t) \end{aligned}$$

Differentiating again we get:

$$\beta''(t) = (1 - \kappa_Q^\alpha(t))\alpha''(t) - \kappa_Q^\alpha{}'(t)\alpha'(t)$$

Therefore, the Q-curvature of $\beta(t)$ is:

$$\begin{aligned}
\kappa_Q^\beta(t) &= \frac{\beta''(t)^T Q J \beta'(t)}{\|\beta'(t)\|_Q^3} \\
&= \frac{(1-\kappa_Q^\alpha(t))^2 \alpha''(t)^T Q J \alpha'(t) - \kappa_Q^\alpha{}'(t)(1-\kappa_Q^\alpha(t))\alpha'(t)^T Q J \alpha'(t)}{|1-\kappa_Q^\alpha(t)|^3 \|\alpha'(t)\|_Q^3} \\
&= \frac{\kappa_Q^\alpha(t)}{|1-\kappa_Q^\alpha(t)|}
\end{aligned}$$

□

Lemma 5.2 Let $\kappa_Q(\tau)$ be the Q-curvature of $\alpha(t)$ at point τ.

(a) if $\kappa_Q(\tau) > 1$, or $\kappa_Q(\tau) = 1$ and $\kappa_Q'(\tau) \neq 0$, then for any $\epsilon > 0$

$$\exists t \in (\tau - \epsilon, \tau + \epsilon), \ \ \|\alpha(t) - \beta(\tau)\|_Q < 1$$

(b) if $\kappa_Q(\tau) < 1$, or $\kappa_Q(\tau) = 1$ and $\kappa_Q'(\tau) = 0$ and $\kappa_Q''(\tau) < 0$, then there exists $\epsilon > 0$ such that

$$\forall t \in (\tau - \epsilon, \tau + \epsilon), \ \ \|\alpha(t) - \beta(\tau)\|_Q \geq 1$$

Proof: The derivatives of κ_Q can be expressed as:

$$\begin{aligned}
\kappa_Q'(t) &= \frac{\alpha'''(t)^T Q J \alpha'(t)}{\|\alpha'(t)\|_Q^3} - 3\frac{\alpha''(t)^T Q \alpha'(t)}{\|\alpha'(t)\|_Q^2}\kappa_Q(t) \\
\kappa_Q''(t) &= \frac{\alpha^{(4)}(t)^T Q J \alpha'(t)}{\|\alpha'(t)\|_Q^3} - 4\frac{\alpha'''(t)^T Q \alpha'(t)}{\|\alpha'(t)\|_Q^2}\kappa_Q(t) - \\
&\quad 3\frac{\alpha''(t)^T Q \alpha''(t)}{\|\alpha'(t)\|_Q^2}\kappa_Q(t) - 5\frac{\alpha''(t)^T Q \alpha'(t)}{\|\alpha'(t)\|_Q^2}\kappa_Q'(t)
\end{aligned}$$

Define function $d(t)$ and calculate its derivatives:

$$\begin{aligned}
d(t) &= \frac{1}{2}\|\alpha(t) - \beta(\tau)\|_Q^2 \\
d'(t) &= \alpha'(t)^T Q(\alpha(t) - \beta(\tau)) \\
d''(t) &= \alpha''(t)^T Q(\alpha(t) - \beta(\tau)) + \|\alpha'(t)\|_Q^2 \\
d'''(t) &= \alpha'''(t)^T Q(\alpha(t) - \beta(\tau)) + 3\alpha''(t)^T Q \alpha'(t) \\
d^{(4)}(t) &= \alpha^{(4)}(t)^T Q(\alpha(t) - \beta(\tau)) + 4\alpha'''(t)^T Q \alpha'(t) + 3\|\alpha''(t)\|_Q^2
\end{aligned}$$

Evaluation at $t = \tau$ yields

$$\begin{aligned} d(\tau) &= \frac{1}{2} \\ d'(\tau) &= 0 \\ d''(\tau) &= \|\alpha'(\tau)\|_Q^2(1 - \kappa_Q(\tau)) \end{aligned}$$

When $\kappa_Q(\tau) < 1$, $d(t)$ reaches a local minimum at $t = \tau$. When $\kappa_Q(\tau) > 1$, $d(t)$ reaches a local maximum at $t = \tau$. When $\kappa_Q(\tau) = 1$ we get $d''(\tau) = 0$ and

$$d'''(\tau) = -\|\alpha'(\tau)\|_Q^2 \kappa_Q'(\tau)$$

Then $\kappa_Q'(\tau) \neq 0$ implies $d'''(\tau) \neq 0$, which means that there exists a t close to τ such that $\|\alpha(t) - \beta(\tau)\|_Q < 1$. Finally, when $\kappa_Q(\tau) = 1$ and $\kappa_Q'(\tau) = 0$, we have $d''(\tau) = d'''(\tau) = 0$ and

$$d^{(4)}(\tau) = -\|\alpha'(\tau)\|_Q^2 \kappa_Q''(\tau)$$

Then $\kappa_Q''(\tau) < 0$ implies $d^{(4)}(\tau) > 0$, which means that $d(t)$ reaches a local minimum at $t = \tau$. □

Lemma 5.3 If $\beta(\tau_1) = \beta(\tau_2)$ is a self-intersection point of the Q-offset curve $\beta(t)$ such that $\beta'(\tau_1)$ and $\beta'(\tau_2)$ are not parallel, then for any $\epsilon > 0$ there is an interference point in $(\tau_1 - \epsilon, \tau_1 + \epsilon)$ and in $(\tau_2 - \epsilon, \tau_2 + \epsilon)$.

Proof: We calculate the first order approximation of $\beta(t)$ in the neighborhood of τ_1:

$$\begin{aligned} \beta(t) &= \beta(\tau_1) + \beta'(\tau_1)(t - \tau_1) + o(t - \tau_1) \\ &= \beta(\tau_2) + \beta'(\tau_1)(t - \tau_1) + o(t - \tau_1) \\ &= \alpha(\tau_2) + \frac{J\alpha'(\tau_2)}{\|\alpha'(\tau_2)\|_Q} + \beta'(\tau_1)(t - \tau_1) + o(t - \tau_1) \end{aligned}$$

Now the distance from $\beta(t)$ to $\alpha(\tau_2)$ can be expressed as:

$$\begin{aligned} d(t) &= \|\beta(t) - \alpha(\tau_2)\|_Q^2 \\ &= \frac{\|J\alpha'(\tau_2)\|_Q^2}{\|\alpha'(\tau_2)\|_Q^2} + \frac{\beta'(\tau_1)^T Q J\alpha'(\tau_2)}{\|\alpha'(\tau_2)\|_Q}(t - \tau_1) + \|\beta'(\tau_1)\|_Q^2(t - \tau_1)^2 + o((t - \tau_1)^2) \end{aligned}$$

Clearly $d(\tau_1) = 1$ and, by Lemma 5.1, $d'(\tau_1) \neq 0$. Hence it is always possible to find a t close to τ_1 such that t is an interference point. The analysis for τ_2 is similar. □

Lemma 5.4 Suppose that the endpoints of I are not interference points. If τ is an endpoint of an interference segment, then $\beta(\tau)$ is a self-intersection point, and $\kappa_Q(\tau) < 1$.

Proof: First, we eliminate several possibilities.

An extraordinary point cannot be an endpoint of an interference segment. By Lemma 5.2, $\forall t \in (\tau - \epsilon, \tau + \epsilon)\ \|\beta(\tau) - \alpha(t)\|_Q \geq 1$. If $\exists t \in I\ \|\beta(\tau) - \alpha(t)\|_Q < 1$ then τ is an interference point but not the endpoint of an interference segment.

A cusp point is itself an interference point by Lemma 5.2, and so cannot be the endpoint of an interference segment. The situation is subtle and deserves detailed analysis. See Figure 5.22. Suppose $\kappa_Q(\tau) = 1$ and $\kappa'_Q(\tau) > 0$. We can find $\tau_1 > \tau$ such that $\|\beta(\tau) - \alpha(\tau_1)\| < 1$. Let E be an ellipse defined by $\|x - \alpha(\tau_1)\| = 1$. There must be an intersection of β and E at $\beta(\tau_2)$ where $\tau_2 < \tau$. If τ_1 is an endpoint of I, then τ_2 is an endpoint of the interference segment. This case is eliminated by the assumption of the lemma. If t goes further beyond τ_1, the endpoint τ_3 of the interference segment must satisfy $\tau_3 < \tau_2$. Similar analysis can be done for $\kappa'_Q(\tau) < 0$. Consequently, at the endpoint τ of an interference segment $\kappa(\tau) < 1$.

Now assume τ is an endpoint of an interference segment. From the discussion above and Lemma 5.1 there exists an interval $(\tau - \epsilon, \tau + \epsilon)$ on which β is regular. Let τ_1 be an interference point in $(\tau - \epsilon, \tau + \epsilon)$. There must be a $\tau'_1 \notin (\tau - \epsilon, \tau + \epsilon)$ such that $\|\beta(\tau_1) - \alpha(\tau'_1)\|_Q < 1$. Since τ_1 can be arbitrarily close to τ, there must be a $\tau' \notin (\tau - \epsilon, \tau + \epsilon)$ at which $\|\beta(\tau) - \alpha(\tau')\|_Q = 1$. The point $\beta(\tau)$ is on the curve segment $\beta(t)\ t \in (\tau' - \epsilon', \tau' + \epsilon')$ if we can prove that $\|\beta(\tau) - \alpha(t)\|_Q$ achieves a local minimum at $t = \tau'$. This is obvious because otherwise τ is an interference point which is a contradiction. □

The lemmas above all deal with the Q-offset curve with a fixed Q. The matrix Q depends on parameters a, b and θ. Since a is fixed because of the fixed radius of the tool, we consider a familiar of Q-offset curves β with parameters b and θ, written as $(x_1, x_2) = \beta(t, b, \theta)$.

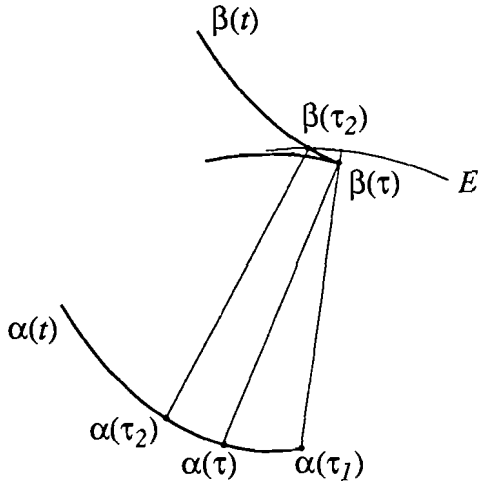

Figure 5.22 Interference near a cusp

Lemma 5.5 The family of Q-offset curves forms a hypersurface in (x_1, x_2, b, θ)-space. The self-intersection points compose a 2-surface on the hypersurface.

Proof: The family of Q-offset curves can be written as $(x_1, x_2) = \beta(t, b, \theta)$. The map $\mathbf{x} : (t, b, \theta) \mapsto (\beta(t, b, \theta), b, \theta)$ defines a (parametric) 3-surface if it is differentiable [dC76]. It suffices to calculate the partial derivatives of β.

$$\begin{aligned}
\frac{\partial \beta}{\partial t} &= (1 - \kappa_Q(t))\alpha'(t) \\
\frac{\partial \beta}{\partial b} &= \frac{\frac{\partial J}{\partial b}\alpha'(t)}{\|\alpha'(t)\|_Q} - \frac{J\alpha'(t)(\alpha'(t)^T \frac{\partial Q}{\partial b}\alpha'(t))}{2\|\alpha'(t)\|_Q^3} \\
\frac{\partial \beta}{\partial \theta} &= \frac{\frac{\partial J}{\partial \theta}\alpha'(t)}{\|\alpha'(t)\|_Q} - \frac{J\alpha'(t)(\alpha'(t)^T \frac{\partial Q}{\partial \theta}\alpha'(t))}{2\|\alpha'(t)\|_Q^3}
\end{aligned}$$

Note that the denominator will never vanish and so $\mathbf{x}$ is differentiable everywhere in its domain. But the Jacobian matrix of $\mathbf{x}$ reduces its rank at the point (t, b, θ) satisfying $\kappa_Q(t) = 1$. These are the singular points of the 3-surface.

The self-intersection points satisfy the following equations:

$$f(t_1, t_2, b, \theta) = \beta(t_1, b, \theta) - \beta(t_2, b, \theta) = 0$$

Suppose that $p_0 = (\tau_1, \tau_2, b_0, \theta_0)$ is a self-intersection point, and that $\frac{\partial \beta}{\partial t}(\tau_1, b_0, \theta_0)$ and $\frac{\partial \beta}{\partial t}(\tau_2, b_0, \theta_0)$ are not parallel, then

$$\det(\frac{\partial f}{\partial t_1}, \frac{\partial f}{\partial t_2})_{p_0} = \det(\frac{\partial \beta}{\partial t}(\tau_1, b_0, \theta_0), -\frac{\partial \beta}{\partial t}(\tau_2, b_0, \theta_0)) \neq 0$$

By the implicit function theorem of advanced calculus, in the neighborhood of p_0 there is a differentiable function η such that $(t_1, t_2) = \eta(b, \theta)$ and $f(\eta(b, \theta), b, \theta) = 0$. Now the set of self-intersection points in the neighborhood of p_0 can be expressed as $\beta(\eta_1(b, \theta), b, \theta)$ that is a 2-dimensional patch on the 3-surface. Other patches can be constructed similarly. Together they compose a 2-surface. □

Lemma 5.6 The family of Q-curvatures, written as $\kappa(t, b, \theta)$, is differentiable to order m if α is differentiable to order $m + 2$.

Proof: This can be easily verified by the partial derivatives.

$$\begin{aligned}
\kappa(t, b, \theta) &= \frac{\alpha''(t)^T Q J \alpha'(t)}{\|\alpha'(t)\|_Q^3} = \frac{\alpha''(t)^T R(\frac{\pi}{2}) \alpha'(t)}{ab\|\alpha'(t)\|_Q^3} \\
\frac{\partial \kappa}{\partial t} &= \frac{\alpha'''(t)^T R(\frac{\pi}{2}) \alpha'(t)}{ab\|\alpha'(t)\|_Q^3} - \frac{3\alpha''(t)^T Q \alpha'(t)}{\|\alpha'(t)\|_Q^2} \kappa(t, b, \theta) \\
\frac{\partial \kappa}{\partial b} &= (-\frac{1}{b} - \frac{3\alpha'(t)^T \frac{\partial Q}{\partial b} \alpha'(t)}{2\|\alpha'(t)\|_Q^2}) \kappa(t, b, \theta) \\
\frac{\partial \kappa}{\partial \theta} &= -\frac{3\alpha'(t)^T \frac{\partial Q}{\partial \theta} \alpha'(t)}{2\|\alpha'(t)\|_Q^2} \kappa(t, b, \theta)
\end{aligned}$$

Continuing the differentiation shows that the denominators are in the form $a^i b^j \|\alpha'(t)\|_Q^k$, and so they will never be zero. The matrix Q is also differentiable to any order. □

As a simple example, let $\alpha(t) = (t, t^2)$. The hypersurface $\beta(t, b, \theta)$ in (x_1, x_2, b, θ)-space is shown in Figure 5.23. The hypersurface $\kappa(t, b, \theta)$ in (t, b, θ, κ)-space is shown in Figure 5.24. The parameter a is fixed at 0.25. The domain shown is $-1 \leq t \leq 1$, $0.025 \leq b \leq 0.25$, $-\frac{\pi}{4} \leq \theta \leq \frac{\pi}{4}$. The isosurfaces on these hypersurfaces have the constant values $\theta = -\frac{\pi}{4}, -\frac{\pi}{8}, 0, \frac{\pi}{8}, \frac{\pi}{4}$. The curves on the hypersurfaces will be explained later.

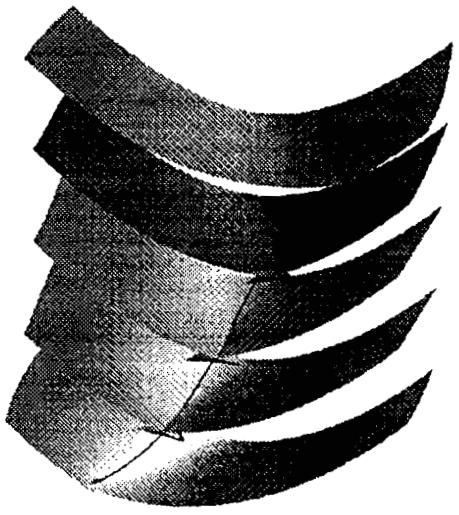

Figure 5.23 The hypersurface $\beta(t, b, 0)$

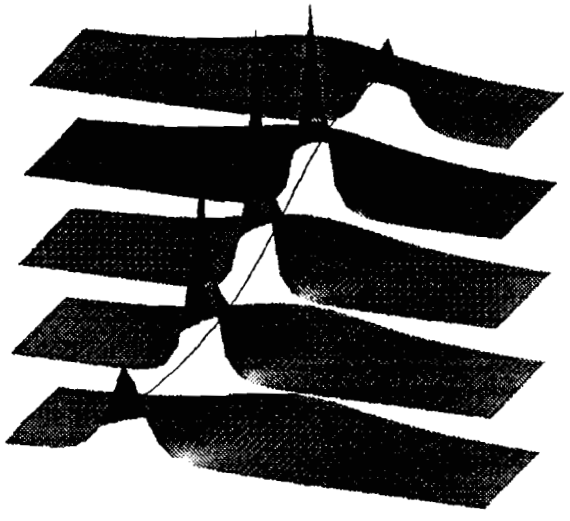

Figure 5.24 The curvature $\kappa(t, b, 0)$

Now we discuss the interference related to the hypersurface $\beta(t, b, \theta)$. A point (t, b, θ) is said to be an interference point if and only if t is an interference point of the Q-offset $\beta(t)$ where Q is determined by b and θ.

Theorem 5.1 Suppose that the hypersurface $\beta(t, b, \theta)$ is defined on a convex domain $D \subset \mathcal{R}^3$, and that for any pair of (b, θ) the two endpoints of the domain, $t_{\min}(b, \theta)$ and $t_{\max}(b, \theta)$, are not interference points. If $\beta(t_1, b_1, \theta_1)$ is an interference point while $\beta(t_2, b_2, \theta_2)$ is not, then any curve segment on the hypersurface $\beta(t, b, \theta)$ and connecting them will pass through a self-intersection point or an extraordinary point.

Proof: Assume that the curve segment connecting the two points is $\beta(t, b(t), \theta(t))$ $t_1 \leq t \leq t_2$ with $b(t_1) = b_1$, $b(t_2) = t_2$ and $\theta(t_1) = \theta_1$, $\theta(t_2) = \theta_2$. From the continuity of the hypersurface and the curve, it is possible to find a τ such that $\forall t < \tau$ $(t, b(t), \theta(t))$ is an interference point while $(\tau, b(\tau), \theta(\tau))$ is not. Consider the Q-offset curve in the neighborhood of τ where Q is fixed by $b(\tau)$ and $\theta(\tau)$, namely, the curve $\beta(t, b(\tau), \theta(\tau))$. There are two cases. In the first case, those points with $t < \tau$ are interference point while those points with $t > \tau$ are not. By Lemma 5.4 $\beta(\tau, b(\tau), \theta(\tau)$ is a self-intersection point. In the second case, both sides of τ are not interference point, we claim that $\beta(\tau, b(\tau), \theta(\tau))$ is an extraordinary point. If $\kappa(\tau, b(\tau), \theta(\tau) < 1$, by Lemma 5.2 and 5.6 there is a neighborhood of $(\tau, b(\tau), \theta(\tau))$ containing no interference point. This contradicts the definition of τ. Hence $\kappa(\tau, b(\tau), \theta(\tau)) = 1$. The cusp can be ruled out because it is an interference point. The remaining case is an extraordinary point. It is possible because slightly perturbing b and θ will cause interference. □

When b and θ can vary, the situation is more complicated than the Q-offset curve of a fixed Q. This theorem states that if a curve on the hypersurface $\beta(t, b, \theta)$ never touches a self-intersection point or an extraordinary point, interference can be avoided.

5.4.4 Finding Optimal Q-offset Curves

There are infinitely many interference free curves on the hypersurface if its domain is suitably large. We establish a criterion for choosing one curve among them. The *optimal Q-offset curve* is defined as a curve on the hypersurface $\beta(t, b, \theta)$, denoted by $\beta(t, b(t), \theta(t))$ such that t is never an interference point, and that

$$\int_I (1 - \kappa(t, b(t), \theta(t)))w(t)dt \tag{5.1}$$

reaches its minimum, where $w(t)$ is a positive weight function.

The optimal Q-offset curve represents the most efficient and interference free tool path. We need to find b and θ of the ellipse as functions of t such that interference never happens. Besides, the curvature of the ellipse and the curvature of the generator curve match as close as possible at the contact point. This is equivalent to that the Q-curvature of the generator is close to 1.

It is not practicable, and perhaps impossible, to find the explicit form of $b(t)$ and $\theta(t)$ for an optimal Q-offset curve. Therefore, a numerical method will be applied. Consequently we may find a solution which is only locally optimal.

When the curve $\alpha(t)$ has only one peak value of $\kappa_Q(t)$ for any Q in the domain, or the peaks of $\kappa_Q(t)$ are separated enough so that their interactions can be ignored, a method to find a locally optimal curve works as follows.

For each high Q-curvature region, the curve follows the extraordinary points defined by

$$\kappa(t, b(t), \theta(t)) = 1 \qquad \frac{\partial \kappa}{\partial t}(t, b(t), \theta(t)) = 0$$

Since the only peak value of $\kappa(t, b, \theta)$ is kept as 1, it is interference free and its contribution to the integration of (5.1) is 0.

Figure 5.23 shows such a curve segment. It consists of the extraordinary points of $\beta(t, b, \theta)$. The curvature $\kappa(t, b, \theta)$ is shown in Figure 5.24. The fact that $\frac{\partial \kappa}{\partial t}(t, b(t), \theta(t)) = 0$ can be observed at the intersection of the curve and the 2-surfaces in the hypersurface $\kappa(t, b, \theta)$. The projection of the optimal curve segment in $\mathcal{R}^2$ is

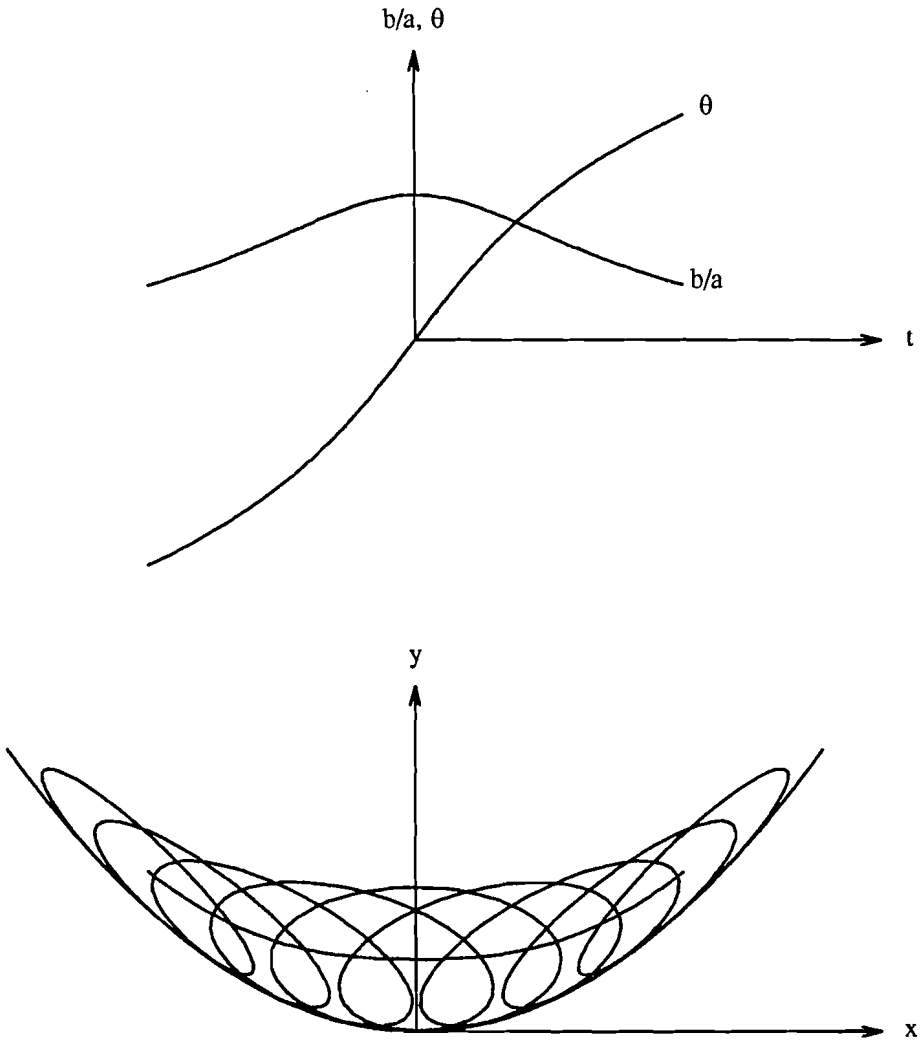

Figure 5.25 Optimal Q-offset curve of $(t, t^2), a = 0.25$

shown in Figure 5.25, together with the corresponding ellipses. The orientation and minor radius of the ellipse change according to the shape of the generator curve.

Figure 5.23 also shows that the curve segment ends at the boundary of the domain. This can happen when the orientation of the ellipse θ fails to match the slope of the generator curve, or the minor radius b causes the ellipse not to match the curvature of the generator curve. In such cases the parameter b and θ could be fixed to their values at the intersection of the curve with the domain boundary.

6. CONCLUSION AND FUTURE WORK

In this thesis a method has been proposed for visualizing 1-, 2-, and 3-surfaces in 4-space. The research has been done on a conceptual level as well as validated experimentally. The work is summarized in the following two sections. The last section discusses some considerations on visualization of 5-space.

6.1 Concepts and Intuitions

Orientation. The generalized Euler angles have been shown to be suitable for specifying the orientation of objects and the projections. The two centers of projections, eye_4 and eye_3, are controlled by one set of Euler angles. The arrangement guarantees that (a) eye_3 is inside the subspace, called the 3D image space, orthogonal to eye_4's direction; (b) the projected world w-axis can be simply kept vertical in the 2D image space which is always orthogonal to both the eyes' directions; (c) the control of eye_3 is consistent with 3D computer graphics. The use of quaternions can overcome the difficulties with Euler angles. Although the theory of 4D rotation can be built by quaternion algebra, we prefer a geometric interpretation for the application purpose. We have shown that the 4D rotation can be decomposed into two orthogonal subrotations. Since the quaternion pairs representing 4D rotations are not unique, some useful forms have been discussed: (a) the first form $\rho_1(p,r) = L_p R_{\bar{r}}$ is suitable for rotation combination and interpolation; (b) the second form $\rho_2(s,r) = L_s L_r R_{\bar{r}}$ is suitable for conversion between quaternions and matrices or Euler angles; (c) the third form $\rho_3(u,v) = L_u R_u L_v R_{\bar{v}}$ is suitable for user interface.

Silhouette and Envelopes. The silhouette point of an m-surface with respect to a projection φ_n^l from $\mathcal{R}^n$ to $\mathcal{R}^l$ has been defined. Some special forms, such as the

silhouette surface of a hypersurface with respect to φ_4^3 and the silhouette curve of a 2-surface with respect to φ_4^2 are discussed in detail. The relationship between silhouette and envelope finds its applications in (a) the explanation of some phenomena in the interrogation of a constraint surface defined by the envelope theorem; (b) the construction of an image of hypersurface in mind; (c) the design of algorithms for visibility determination. In the definition of envelope, the locus of singular points is excluded. Correspondingly, the singular points on an m-surface are excluded as proper silhouette points. The set of singular points on a hypersurface can be called the singular surface or the discontinuity surface. They might be visualized by shading in 4-space or could be explicitly constructed and then shaded in 3D image space. The behavior of the discontinuity surface under projection and the relationship with the discriminant hypersurface needs to be investigated.

Visibility. It has been shown that visibility determination is necessary to eliminate the ambiguities caused by projection. Figure 3.12 and 3.13 are a pair of pictures of the same hypersurface from different viewing directions. They will be identical if the visibility in 4-space is ignored. The definition of visibility has been carefully considered so that (a) curves, 2- and 3-surfaces can be displayed altogether; (b) a 3-surface can hide other surfaces in 4-space but not in 3-space; (c) the visibility is quantified for displaying transparency. Although the definition of visibility is valid for spaces of any dimensions, the practical limit seems to be 5 or 6.

Geometric Properties and Phenomena. Several geometric properties and phenomena in 4-space have been demonstrated by computer generated pictures and animation. They include the ambiguity caused by projection, the dimension reduction (degeneracy) of the silhouette surface, and the principal curvatures of hypersurfaces. The emphasis is put on observation instead of calculation. Currently the curvature observation depends on silhouette surfaces. For those points not on a silhouette surface, one way is to adjust the viewing direction to make them silhouette points. Other

ways might be using a generalization of Dupin's indicatrix, or a generalization of lighting models in high dimensional space. Another challenging problem is to visualize the C^0, C^1, C^2 continuities of patches or hyperpatches in 4 or higher space.

6.2 System and Applications

Polygonalization. We have adapted Allgower's algorithm [AG87] for polygonalizing implicit 2-surface in 4-space. Three types of Newton iteration for point refinement have been considered. An algorithm for merging polygons has been presented. The methods chosen for point refinement and polygon merging affect the available methods for the following stages of visibility determination and rendering. Currently the speed of polygonalization are sufficient for interactive display. Since the number of simplices grows exponentially with the dimension, it might be very slow if extended into 5 or higher dimensions. Polygonalization for visualization is different than for other purposes. For example, the polygons generated need not be connected if the visibility determination algorithm does not depend on it. Special efficient algorithms can be expected.

Visibility Determination. An algorithm for visibility determination in 4-space has been presented. The basic operation is to find the intersection curves of pairs of 2-surfaces in 3D image space. But the nature of projection caused several kinds of singularities. So the intersection is actually done in 4-space. We discussed the special cases when one of the intersecting 2-surface is a silhouette surface or a boundary surface. Several desingularization techniques are found based on geometric intuitions. The special cases when one of the intersecting 2-surfaces is a self-intersection surface of the hypersurface, or an intersection surface of two hypersurfaces have not been studied in detail. Also, there is room left for the improvement of the efficiency and robustness of the algorithms.

Applications. Four examples have been presented to illustrate the use of 4D visualization: understanding differential geometry, collision detection and analysis, 3D scalar field display, and tool path generation for 5-axes milling machines. It should

be pointed out that the emphasis is put on explaining the problems and techniques through 4D visualization tools rather than on a complete analysis of each problem. The solution to each of the problems deserves a separate thesis. There are other research topics of CAGD and solid modeling where 4D visualization might be useful. An example is the skeleton in 3-space. As defined in [HV91], the *interior skeleton* of a 3D solid is the locus of the centers of all inscribed maximal spheres. To make it an informationally-complete representation, each point on the skeleton should be associated with its distance from the boundary of the 3D object. So the skeleton is composed of 2-surfaces in 4-space. Moreover, it is the singular surface of a hypersurface that is the *trimmed cyclographic map* of the boundary surfaces of the 3D solid. When eye_4 is off the w-axis (the distance axis) by more than $\frac{\pi}{4}$, part of the skeleton will be occluded by the hypersurface as can be inferred from the pictures in [HV91] by the dimension analogy. Visibility in 4-space is therefore necessary.

6.3 Toward the Fifth Dimension

The definitions defined in Chapter 2 all have a general form that is valid for spaces of any dimensions. Theoretically, it is possible to project objects in any high dimensional space down to the final 2D image space. But it is extremely hard to interpret the pictures so generated. Let us explain how we can interpret the pictures from 5D visualization.

It is helpful to think of the 5D world space, and the 4D, 3D and 2D image spaces, as related by a 5D rotation. The rotation can be specified by ten parameters such as the Euler angles. The recursive definition of Euler angles also suggests how these image spaces are related. The generalization of quaternion seems much harder.[1] The group of 2×2 quaternion matrices is a double cover of the rotation group in $\mathcal{R}^5$. The ten independent real parameters of the 5D rotation can be arranged in this way: four parameters form a unit vector in 5-space called the axis of the rotation, and the

[1] Next to the quaternions is the algebra of octonions, called the Cayley algebra, which is nonassociative [Por81].

remaining six parameters, a pair of unit quaternions, form the 4D rotation orthogonal to the axis.

Displaying 1-, 2-, 3-surfaces in 5-space will not be too much different from displaying them in 4-space. We only need to consider the distortion introduced by the additional projection from 5-space to 4-space. The most difficult task will be displaying the 4-surface, i.e. the hypersurface in 5-space. The 4-surface is displayed in 4D image space via its silhouette 3-surfaces, boundary 3-surfaces, and so on. The visibility determined by eye_5 will cause these 3-surfaces trimmed. Therefore, in the 4D image space there should be several trimmed 3-surfaces. The techniques presented in this thesis can be extended to the simultaneous display of several trimmed 3-surfaces. After further research and training, we might be able to recognize these trimmed 3-surfaces from their 3D and 2D images. Finally, putting together the information revealed by these trimmed 3-surfaces, the hypersurface in 5-space could be visualized.

BIBLIOGRAPHY

BIBLIOGRAPHY

[Abb63] E. A. Abbott. *Flatland - A Romance of Many Dimensions.* Barnes & Noble Books, Inc., 1963. (First Edition 1884).

[AG87] E. L. Allgower and S. Gnutzmann. An algorithm for piecewise linear approximation of implicitly defined two-dimensional surfaces. *SIAM Journal on Numerical Analysis*, 24(2):452–469, 1987.

[AG90] E. L. Allgower and S. Gnutzmann. Polygonal meshes for implicitly defined surfaces. Colorado State University, 1990.

[All90] E. L. Allgower. *Numerical Continuation Methods: an Introduction.* Springer-Verlag, 1990.

[AM63] L. Auslander and R. E. MacKenzie. *Introduction to Differentiable Manifolds.* MaGraw-Hill Book Company, Inc., 1963.

[AS85] E. L. Allgower and P. H. Schmidt. An algorithm for piecewise linear approximation of implicitly defined manifold. *SIAM Journal on Numerical Analysis*, 22(2):322–346, 1985.

[AU90] S. Aomura and T. Uehara. Self-intersection of an offset surface. *Computer Aided Design*, 22(7):417–422, 1990.

[Baj90] C. L. Bajaj. Rational hypersurface display. *Computer Graphics*, 24(2):117–127, 1990.

[Ban86] T. F. Banchoff. Visualizing two-dimensional phenomena in four-dimensional space: A computer graphics approach. In E. J. Wegman and D. J. DePriest, editors, *Statistical Image Processing and Graphics*, pages 187–202. M. Dekker, New York, 1986.

[Ban90] T. F. Banchoff. *Beyond the Third Dimension: Geometry, Computer Graphics, and Higher Dimensions.* Scientific American Library, 1990.

[BHHL88] C. Bajaj, C. Hoffmann, J. Hopcroft, and R. Lynch. Tracing surface intersections. *Computer Aided Geometric Design*, 5:285–307, 1988.

[Blo88] J. Bloomenthal. Polygonization of implicit surfaces. *Computer Aided Geometric Design*, 5:341–355, 1988.

[BM90] S. B. M. Bell and D. C. Mason. Tesseral quaternions for the octtree. *The Computer Journal*, 33(5):386–397, 1990.

[BS82] R. P. Burton and D. R. Smith. A hidden-line algorithm for hyperspace. *SIAM Journal on Computing*, 11(1):71–80, 1982.

[Cam84] S. A. Cameron. *Modelling Solids in Motion.* PhD thesis, University of Edinburgh, 1984.

[Chu90] J. H. Chuang. *Surface Approximations in Geometric Modeling.* PhD thesis, Purdue University, 1990.

[CJS9] B. K. Choi and C. S. Jun. Ball-end cutter interference avoidance in NC machining of sculptured surfaces. *Computer Aided Design*, 21(6):371–378, 1989.

[CK87] V. Chandru and B. S. Kochar. Analytic techniques for geometric intersection problems. In G. E. Farin, editor, *Geometric Modeling: Algorithms and New Trends.* SIAM Publications, 1987.

[Cox47] H. S. M. Coxeter. *Regular Polytopes.* Pitman Publishing Corporation, 1947.

[CR61] H. M. Cundy and A. P. Rollett. *Mathematical Models.* Oxford University Press, 1961. (First edition 1951).

[dC76] Manfredo P. do Carmo. *Differential Geometry of Curves and Surfaces.* Prentice-Hall, Inc., 1976.

[EC90] G. Elber and E. Cohen. Hidden curve removal for free form surfaces. *Computer Graphics*, 24(4):95–104, 1990.

[Eck68] L. Eckhart. *Four-dimensional space (in German, 1929).* Indiana University Press, 1968. English translation by A. L. Bigelow and S. M. Slaby.

[Far86] R. T. Farouki. The approximation of non-degenerated offset surfaces. *Computer Aided Geometric Design*, 3:15–43, 1986.

[Far87] R. T. Farouki. Graphical methods for surface differential geometry. In R. R. Martin, editor, *The Mathematics of Surfaces II*, pages 363–385. Clarendon Press, Oxford, 1987.

[FN90] R. T. Farouki and C. A. Neff. Algebraic properties of plane offset curves. *Computer Aided Geometric Design*, 7:101–127, 1990.

[For30] A. R. Forsyth. *Geometry of Four dimensions.* The University Press, Cambridge, 1930.

[GN89] R. S. Gallagher and J. C. Nagtegaal. An efficient 3-D visualization technique for finite element models and other coarse volumes. *Computer Graphics*, 23(3):185–194, 1989.

[Gol80] Herbert Goldstein. *Classical Mechanics (Second Edition).* Addison-Wesley Publishing Company, Inc., 1980.

[Ham69] William R. Hamilton. *Elements of Quaternions (Third Edition).* Chelsea Publishing Company, 1969. (First Edition 1866, Ed. by Charles J. Joly).

[Hof89] C. M. Hoffmann. *Geometric and Solid Modeling: An Introduction.* Morgan Kaufmann Publishers, Inc., 1989.

[Hof90] C. M. Hoffmann. A dimensionality paradigm for surface interrogations. *Computer Aided Geometric Design*, 7:517–532, 1990.

[HV91] C. M. Hoffmann and G. Vaněček, Jr. Fundamental techniques for geometric and solid modeling. In C. T. Leondes, editor, *Advances in Control and Dynamics.* Academic Press, 1991. to appear.

[HW90] M. Hall and J. Warren. Adaptive polygonalization of implicitly defined surfaces. *IEEE Computer Graphics & Applications*, pages 33–42, November 1990.

[HZ91] C. M. Hoffmann and J. Zhou. Some techniques for visualizing surfaces in four-dimensional space. *Computer Aided Design*, 23(1):83–91, 1991.

[Ins85] A. Inselberg. The plane with parallel coordinates. *The Visual Computer*, 1(1):69–91, 1985.

[Ins90] A. Inselberg. Parallel coordinates: A tool for visualizing multi-dimensional geometry. In *Proceedings of the First IEEE Conference on Visualization*, pages 361–378, San Francisco, Calfornia, October 1990.

[KBBL86] H. Koçak, F. Bisshopp, T. Banchoff, and D. Laidlaw. Topology and mechanics with computer graphics: Linear hamiltonian systems in four dimensions. *Advances in Applied Mathematics*, 7:282–308, 1986.

[KL87] H. Koçak and D. Laidlaw. Computer graphics and the geometry of S^3. *The Mathematical Intelligence*, 9(1):8–10, 1987.

[LC87] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. *Computer Graphics*, 21(4):163–169, 1987.

[Man56] Henry P. Manning. *Geometry of Four Dimensions.* Dover Publications, Inc., 1956. (First Edition 1914).

[Mar85] R. R. Martin. Rotation by quaternions. *Mathematical Spectrum*, pages 42–48, March 1985.

[Mar87] Krzysztof Marciniak. Influence of surface shape on admissible tool positions in 5-axis face milling. *Computer Aided Design*, 19(5):233–236, 1987.

[MGTS89] T. Mihalisin, E. Gawlinski, J. Timlin, and J. Schwegler. Multidimensional graphing in two-dimensional space. *Computers in Physics*, pages 32–39, November 1989.

[MTS91] T. Mihalisin, J. Timlin, and J. Schwegler. Visualization multivariate functions, data, and distributions. *IEEE Computer Graphics & Applications*, pages 28–35, May 1991.

[MYGP90] E. McLellan, G. M. Young, R. J. Goult, and M. J. Pratt. Interference checking in the 5-axis machining of parametric surfaces. Technical report, Department of Applied Computing and Mathematics, Cranfield Institute of Technology, Cranfield Bedford, MK43 0AL, England, 1990.

[NFHL91] G. M. Nielson, T. A. Foley, B. Hamann, and D. Lane. Visualizing and modeling scattered multivariate data. *IEEE Computer Graphics & Applications*, pages 47–55, May 1991.

[Nol67] A. M. Noll. A computer technique for displaying n-dimensional hyperobjects. *Communications of the ACM*, 10:469–473, 1967.

[O'N66] Barrett O'Neill. *Elementary Differential Geometry.* Academic Press, 1966.

[ORS87] J. C. Owen and A. P. Rockwood. Intersection of general implicit surface. In G. E. Farin, editor, *Geometric Modeling: Algorithms and New Trends*, pages 335–345. SIAM Publications, 1987.

[PGS86] M. J. Pratt and A. D. Geisow. Surface/surface intersection problems. In J. Gregory, editor, *The Mathematics of Surfaces*, pages 117–142. Oxford University Press, 1986.

[Por81] Ian R. Porteous. *Topological Geometry (second edition).* Cambridge University Press, 1981.

[PS74] G. Pickert and H.-G. Steiner. Complex numbers and quaternions. In H. Behnke, F. Bachmann, K. Fladt, and W. Süss, editors, *Foundations of Mathematics The Real Number System and Algebra*, volume 1, pages 456–482. The MIT Press, 1974. (in German 1962, translated by S. H. Gould).

[RHD89] A. Rockwood, K. Heaton, and T. Davis. Real-time rendering of trimmed surfaces. *Computer Graphics*, 23(3):107–116, 1989.

[Rhe87] W. C. Rheinboldt. On a moving-frame algorithm and the triangulation of equilibrium manifolds. In R. Seydel T. Küpper and H. Troger, editors, *Bifurcation: Analysis, Algorithms, Application.* Brikhäser Verlag Basel, 1987.

[Ros89] J. R. Rossignac. Considerations on the interactive rendering of four-dimensional volumes. In *Proceedings of Volume Visualization Workshop*, pages 67–76, Chapel Hill, NC, May 1989.

[Sab88] P. Sabella. A rendering algorithm for visualizing 3D scalar fields. *Computer Graphics*, 22(4):51–58, 1988.

[Sas90] Nobuo Sasaki. Five axis control of machining freeform surfaces. School of Mechanical Engineering, Purdue University, 1990.

[Sei90a] H. P. Seidel. Geometric constructions and knot insertion for β-splines of arbitrary degree. Technical report, Department of Computer Science, University of Waterloo, 1990.

[Sei90b] H. P. Seidel. Quaternionen in der computergraphik und robotik. *Informationstechnik*, it-32:260–275, 1990.

[She78] S. W. Shepperd. Quaternion from rotation matrix. *J. Guidance and Control*, 1(3):223-224, 1978.

[Sho85] Ken Shoemake. Animating rotation with quaternion curves. *Computer Graphics*, 19(3):245–254, 1985.

[SK90] D. Speray and S. Kennon. Volume probes: Interactive data exploration on arbitrary grids. *Computer Graphics*, 24(5):5–12, 1990.

[Spi79] Michael Spivak. *A Comprehensive Introduction to Differential Geometry.* Publish or Perish, Inc., 1979.

[SSS74] I. E. Sutherland, R. F. Sproull, and R. A. Schumacker. A characterization of ten hidden-surface algorithms. *Computing Surveys*, 6(1):1–55, 1974.

[ST90] P. Shirley and A. Tuchman. A polygonal approximation to direct scalar volume rendering. *Computer Graphics*, 24(5):63–70, 1990.

[Stu64] John Stuelpnagel. On the parametrization of the three-dimensional rotation group. *SIAM Review*, 6(4):422–430, 1964.

[Tho79] J. A. Thorpe. *Elementary Topics in Differential Geometry.* Springer-Verlag New York Inc., 1979.

[TT81] P. A. Tukey and J. W. Tukey. Graphical display of data sets in 3 or more dimensions. In V. Barnett, editor, *Interpreting Multivariate Data.* John Wiley & Sons, 1981.

[UK88] C. Upson and Michael Keeler. V-buffer: Visible volume rendering. *Computer Graphics*, 22(4):59–64, 1988.

[Vla90] V. Vlassopoulos. Adaptive polygonization of parametric surfaces. *The Visual Computer*, 6:291–298, 1990.

[VQ89] G. W. Vickers and K. W. Quan. Ball-mills versus end-mills for curved surface machining. *Trans. ASME, Journal of Engineering for Industry*, 111(1):22–26, 1989.

[Wal90] R. Walter. Visibility of surfaces via differential geometry. *Computer Aided Geometric Design*, 7:353–373, 1990.

[Wit77] J. Wittenburg. *Dynamics of Systems of Rigid Bodies.* B. G. Teubner Stuttgart, 1977.

VITA

VITA

Jianhua Zhou was born in Shanghai, China on November 2, 1952. He received the B.S. and M.S. degrees in computer engineering from Shanghai University of Technology, China, in 1982 and 1984, respectively. Then he worked with the Artificial Intelligence Laboratory in Shanghai, China for two years. He continued his education at Purdue University in West Lafayette, Indiana, where he received the M.S. degree in computer science in 1988. While at Purdue, he was supported as a graduate instructor and as a research assistant under the supervision of Professor Christoph M. Hoffmann.

His research interests include computer graphics, scientific visualization, geometric and solid modeling, simulation, computer aided design and computer aided manufacturing. He is a member of ACM and SIGGRAPH, and a member of the Honor Society of PHI KAPPA PHI.