1990

Visualization of Surfaces in Four-Dimensional Space

Christoph M. Hoffmann
Purdue University, cmh@cs.purdue.edu

Jianhua Zhou

Report Number:
90-960

VISUALIZATION OF SURFACES IN FOUR-DIMENSIONAL SPACE

Christoph M. Hoffman
Jianhua Zhou

CSD TR-960
May 1990

Visualization of Surfaces in Four-Dimensional Space *

Christoph M. Hoffmann
Jianhua Zhou

Department of Computer Science
Purdue University
West Lafayette, IN 47907

Abstract

We discuss some issues of displaying two-dimensional surfaces in four-dimensional (4D) space, including the behavior of surface normals under projection, the silhouette points due to the projection, and methods for object orientation and projection center specification. We have implemented an interactive 4D display system on z-buffer based graphics workstations. Preliminary experiments show that such a 4D display system can give valuable insights into high-dimensional geometry. We present some pictures from the examples using high-dimensional geometry, offset curve tracing and collision detection problems, and explain some of the insights they convey.

1 Introduction

The geometry of high-dimensional space has shown to be quite useful in the area of CAGD and solid modeling. Applications include describing the motion of 3D objects, modeling solids with nonuniform material properties, and formulating constraints for offset surfaces and Voronoi surfaces [3, 8, 15]. For inspection and understanding of the properties related to geometry, pictures are very effective to provide intuitions. Unfortunately it is very hard, if not impossible, for us to visualize objects in high-dimensional space. Therefore, visualization of high-dimensional space by means of computer graphics, especially interactively, is a research topic that attracts growing attention [10, 15].

*Work supported in part by NSF grant CCR 86-19817 and DMC 88-07550 and by the office of Naval Research under contract N00014-90-J-1599.

Visualization of 4D space is a good starting point because not only is it relatively easy, but also many problems fit naturally into a 4D formulation. For example, describing 3D objects in motion, embedding 3D projective space into affine space, and analyzing plane curves with complex roots and/or coefficients. Since 4D space has many properties unfamiliar to us, its visualization is full of problems to be explored.

We briefly review some basic ideas useful in 4D visualization. A solid object in 4D space is of dimension four. Its boundary is composed of one or more hypersurfaces of dimension three. In nondegenerate cases, the intersection of two hypersurfaces is a surface of dimension two, and the intersection of three hypersurfaces is a curve of dimension one. In contrast to 3D space, two surfaces in 4D space generally intersect in a point instead of a curve. Since human beings have no sense of 4D space, we have to map the 4D solid from 4D space into 3D space. One method is to intersect the 4D object with a hypersurface (perhaps a hyperplane normal to one coordinate axis) and get an image in 3D space. In order to perceive the whole 4D object, a series of images with different positions of the intersecting hypersurface would be needed. Another method is to project the 4D object into 3D space, orthographically or perspectively. Again, a series of images with different directions of projection is needed. Both methods must be supplemented with one more intersection or projection step if we want to render the 3D image on a conventional 2D device such as a piece of paper or a computer screen. The intermediate step of mapping objects into 3D can be so implemented, or can be combined with the mapping to 2D into a single procedure.

Eckhart [4] proposes a method to project an 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. However, since this method only displays curves that are three dimensions lower than a 4D object, and since the viewer must gather information from different pictures, it may be very hard to interpret such pictures.

This paper concentrates on the interactive display of two dimensional surfaces in 4D space, thereafter referred to as 2-surfaces. It is a worthwhile job for several reasons:

1. Some problems are naturally formulated with 2-surfaces in 4D space.

2. 2-surface display is an important subtask in visualizing 4D objects.

3. Interactive speeds are possible because of the available hardware on graphics devices.

For example, consider the collision detection problem of two solids moving in 3D space. Their moving boundaries are hypersurfaces in 4D space. The intersection of the two hypersurfaces is generally a 2-surface in 4D space. Also, for any curve problem with three equations in four variables, we can think of two of the equations as the definition of a 2-surface, and consider the third equation as the constraint for a point of the 2-surface to be on the curve. Displaying the curve and the 2-surface simultaneously is usually easier to understand than displaying the curve alone.

2-surface display is an important subtask in the visualization of 4D solids. A 4D solid can be displayed by its boundaries which are hypersurfaces. The projection of a hypersurface into 3D space is a 3D volume. Basically we have three choices in displaying the 3D volume: volume rendering, surface rendering, and curve rendering (wire mesh). In volume rendering, each voxel[1] corresponds to several points on the hypersurface. It can be reduced into one point on the hypersurface if we use a hidden hypersurface removal technique, similar to the hidden surface removal technique used in 3D rendering. To obtain a nontrivial picture, we need a shading model in 4D space [15]. Although this is an interesting topic, we may want to eliminate the effect of 4D light while trying to understand the structure of hypersurfaces as a set of points. On the other hand, rendering a 4D solid by curves shows a wire mesh, which involves no shading model at all. It is frequently used for displaying surfaces in 3D space. However, it seems hard to interpret such pictures when they are used for displaying hypersurfaces of a 4D solid [13]. Rendering a 4D object via surface rendering shows a group of surfaces placed within a hypersurface, and serves as a bridge between volume rendering and curve rendering. Surface rendering involves only a 3D shading model [7]. Here, the role of the 2-surfaces for displaying a hypersurface is analogous to that of curves for displaying a surface in 3D space. Often we can gain valuable insights from this analogy.

Interactive display is crucial for 4D visualization. After projection from 4D to 3D, and finally to 2D, a significant amount of information is lost. To compensate, the viewer should be able to see a real time animation of the object in translation and rotation, controlled by, say, a mouse. Currently most graphics workstations are designed for 3D rendering, and use efficient techniques of surface display. Investigating of the relationship between surfaces before and after projection may lead to efficient algorithms for 4D visualization.

We are experimenting with an interactive 4D visualization system on conventional z-

[1] A voxel is a three-dimensional pixel

buffer based graphics workstations. The input 2-surfaces are first polygonalized by algebraic or space division methods [1]. The polygons in 4D space are projected into 3D space and then fed into the 3D graphics engine. The method of "polygonalization before projection" is more desirable than the method of "projection before polygonalization" for interactive display. Usually, polygonalization requires more computation. Therefore, polygonalizing the 2-surface as a preprocessing step means that a better response can be obtained when changing the projection parameters repeatedly. Moreover, polygonalization in 4D space can better account for the intrinsic geometric properties of the 2-surface. Some of these properties are distorted by the projection to 3D space. We have carefully designed a user interface which offers position and orientation control of objects, lights, and projection centers.

The paper is organized as follows. Section 2 deals with intrinsic geometric issues such as surface normals and silhouettes under projections. Section 3 discusses one aspect of the user interface, namely, how to extend the Euler angles into 4D space and use these angles uniformly for object orientation and projection direction specification. In section 4, several examples are discussed and illustrated with pictures. Section 5, finally, draws some conclusions from this work.

2 Surface Normals And Silhouettes Under Projection

We map a 2-surface in 4D first to a surface in 3D space. Then the surface is rendered on a 2D device by the standard 3D methods for shading and illumination. To determine the intensity of the light reflected by a surface in 3D space, it is necessary to find the normal of the surface at each point. However, the projections of the normals of a 2-surface in 4D do not necessarily coincide with the normal of the projected 2-surface. 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 normal space of the 2-surface before projection. The latter seems to be more efficient because the derivation of the equation of the projected 2-surface could be expensive [9].

In this section we discuss in detail how to calculate the normal of the projected 2-surface from the normals of the 2-surface before projection. Given a 2-surface and a projection, there are certain points at which the normal space does not determine the normal of the

projected 2-surface. We will show that they are exactly those points whose tangent space is projected into a space of lower dimension, and that they are silhouettes. We also discuss how to recognize silhouettes.

A 2-surface S in 4D space ($\mathcal{R}^4$) can be represented either in implicit form:

$$f_1(x, y, z, w) = 0$$
$$f_2(x, y, z, w) = 0$$

or in parametric form:

$$x = f_1(s, t) \quad z = f_3(s, t)$$
$$y = f_2(s, t) \quad w = f_4(s, t)$$

At a nonsingular point $\mathbf{p}$ on S, there is a unique normal space and a unique tangent space. The two spaces are orthogonal to each other. Since in $\mathcal{R}^4$ both normal space and tangent space of a 2-surface have dimension two, both are determined either by two linearly independent normal vectors $\mathbf{n_1}$ and $\mathbf{n_2}$, or by two linearly independent tangent vectors $\mathbf{t_1}$ and $\mathbf{t_2}$. For a 2-surface in implicit form, it is easy to find the two normal vectors at a nonsingular point p:

$$\mathbf{n_1} = \nabla f_1(\mathbf{p})$$
$$\mathbf{n_2} = \nabla f_2(\mathbf{p})$$

On the other hand, for a 2-surface in parametric form, it is easy to find the two tangent vectors at a point p if the corresponding parameters s and t are given:

$$\mathbf{t_1} = (\frac{\partial f_1}{\partial s}, \frac{\partial f_2}{\partial s}, \frac{\partial f_3}{\partial s}, \frac{\partial f_4}{\partial s})^T$$
$$\mathbf{t_2} = (\frac{\partial f_1}{\partial t}, \frac{\partial f_2}{\partial t}, \frac{\partial f_3}{\partial t}, \frac{\partial f_4}{\partial t})^T$$

A simple method can be used to find two tangent vectors $\mathbf{t_1}$ and $\mathbf{t_2}$ from two normal vectors $\mathbf{n_1}$ and $\mathbf{n_2}$ and vice versa. Let i, j, k, l be the base vectors of $\mathcal{R}^4$, and a, b, c be three vectors where $\mathbf{a} = (a_x, a_y, a_z, a_w)^T$ and so on. We define operation $\otimes$ as:

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

Obviously $\otimes(\mathbf{a},\mathbf{b},\mathbf{c})$ is orthogonal to the subspace span$(\mathbf{a},\mathbf{b},\mathbf{c})$ if $\mathbf{a},\mathbf{b},\mathbf{c}$ are linearly independent. 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

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

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

In most cases $\mathbf{a}$, $\mathbf{b}$ can be chosen from the base vectors, and doing so reduces the needed arithmetic operations. Sometimes $\mathbf{b}$ can be replaced by $\mathbf{t_1}$ to make $\mathbf{t_1}$ and $\mathbf{t_2}$ orthogonal to each other.

A projective mapping v from $\mathcal{R}^4$ to $\mathcal{R}^3$ can be written as

$$v(\mathbf{p}) = v(p_x,p_y,p_z,p_w) = (\frac{p_x}{1-rp_w},\frac{p_y}{1-rp_w},\frac{p_z}{1-rp_w})^T$$

where we assume the center of projection is at $1/r = (0,0,0,1/r)^T$. It is an orthographic projection for $r = 0$ and a perspective projection for $r > 0$. The projection center can be put into an arbitrary position if translation and rotation in $\mathcal{R}^4$ are done before applying v.

Given a 2-surface S and its normal plane N at a point $\mathbf{p}$, by projection we get their images in $\mathcal{R}^3$ denoted by $v(S)$, $v(N)$, and $v(\mathbf{p})$ respectively. In general $v(N)$ is of dimension two and cannot be used as the normal of $v(S)$ at point $v(\mathbf{p})$. Fortunately tangency is a projectively invariant property (see, e.g. [16]). Hence, a natural method to calculate the normal of the projected surface is the following:

Step 1: From the two normals, find two tangent vectors $\mathbf{t_1}$ and $\mathbf{t_2}$ as explained above.

Step 2: Project the three points $\mathbf{p}$, $\mathbf{p}+\mathbf{t_1}$, $\mathbf{p}+\mathbf{t_2}$ into $\mathcal{R}^3$ with the mapping v.

Step 3: Use the cross product to find the normal $\bar{\mathbf{n}}$

$$\bar{\mathbf{n}} = [v(\mathbf{p}+\mathbf{t_1}) - v(\mathbf{p})] \times [v(\mathbf{p}+\mathbf{t_2}) - v(\mathbf{p})]$$

To derive a more efficient method, we need to know the relationship of the 2-surface normals before and after projection. Given a vector $\mathbf{a} = (a_x,a_y,a_z,z_w)^T$ in $\mathcal{R}^4$, we define its *natural projection* into $\mathcal{R}^3$ as $\pi_{\mathrm{XYZ}}(\mathbf{a}) = (a_x,a_y,a_z)^T$. Given a vector $\bar{\mathbf{a}} = (a_x,a_y,a_z)^T$ in $\mathcal{R}^3$, we define its *natural extrusion* into $\mathcal{R}^4$ as $\varepsilon_{\mathrm{W}}(\bar{\mathbf{a}}) = (a_x,a_y,a_z,0)^T$.

Lemma 1 Suppose that $\mathbf{t}_1,\mathbf{t}_2$ $(\mathbf{n}_1,\mathbf{n}_2)$ are two linearly independent tangent (normal) vectors at a nonsingular point $\mathbf{p}$ on a 2-surface S, that v is a projection with center $1/r$, and that $\mathbf{r} = 1/r - \mathbf{p}$ is the ray from the point $\mathbf{p}$ to the projection center. Then,

(a) if $\mathbf{a_i} = \varepsilon_W(v(\mathbf{p}+\mathbf{t_i}) - v(\mathbf{p}))$, then $\mathbf{a_i}, \mathbf{t_i}, \mathbf{r}$ are linearly dependent for $i = 1, 2$.

(b) if $\mathbf{n} = \otimes(\mathbf{t_1}, \mathbf{t_2}, \mathbf{r})$, then $\pi_{XYZ}(\mathbf{n})$ is the normal vector of $v(S)$ at point $v(\mathbf{p})$.

(c) if $\mathbf{n} = \alpha\mathbf{n_1} + \beta\mathbf{n_2}$ satisfying $\mathbf{n} \cdot \mathbf{r} = 0$, then $\pi_{XYZ}(\mathbf{n})$ is the normal vector of $v(S)$ at point $v(\mathbf{p})$.

Proof: (a) Since there is no difference for $i = 1, 2$ we drop the subscripts of $\mathbf{t_i}$ and $\mathbf{a_i}$ temporarily. The vector $\mathbf{a}$ can be written as:

$$\mathbf{a} = \begin{pmatrix} \frac{p_x+t_x}{1-r(p_w+t_w)} - \frac{p_x}{1-rp_w} \\ \frac{p_y+t_y}{1-r(p_w+t_w)} - \frac{p_y}{1-rp_w} \\ \frac{p_z+t_z}{1-r(p_w+t_w)} - \frac{p_z}{1-rp_w} \\ 0 \end{pmatrix}$$

Then

$$\alpha_1\mathbf{a} + \alpha_2\mathbf{t} + \alpha_3(\mathbf{l}/r - \mathbf{p}) = 0$$

where

$$\begin{aligned} \alpha_1 &= (1 - rp_w - rt_w)(1 - rp_w) \\ \alpha_2 &= -(1 - rp_w) \\ \alpha_3 &= rt_w \end{aligned}$$

Clearly α_2 is nonzero because otherwise $\mathbf{p}$ is mapped to infinity. Moreover, by choosing the length of $\mathbf{t}$ appropriately, α_1 is also nonzero. For orthographic projection, we can simply set $\mathbf{r} = \mathbf{l}$ and $\alpha_3 = t_w$, and the proposition is still true.

(b) Let $\mathbf{a_i}$ $(\mathbf{i} = 1, 2)$ be defined as in (a). Then,

$$\begin{aligned} \pi_{XYZ}(\mathbf{n}) \cdot [v(\mathbf{p}+\mathbf{t_i}) - v(\mathbf{p})] &= \otimes(\mathbf{t_1}, \mathbf{t_2}, \mathbf{r}) \cdot \mathbf{a_i} \\ &= \det([\mathbf{a_i}, \mathbf{t_1}, \mathbf{t_2}, \mathbf{r}]^T) \end{aligned}$$

The determinant is zero according to (a).

(c) $\mathbf{n} = \alpha\mathbf{n_1} + \beta\mathbf{n_2}$ guarantees that $\mathbf{n} \cdot \mathbf{t_1} = 0$ and $\mathbf{n} \cdot \mathbf{t_2} = 0$. Together with $\mathbf{n} \cdot \mathbf{r} = 0$ we know that $\mathbf{n}$ is in the same direction as $\otimes(\mathbf{t_1}, \mathbf{t_2}, \mathbf{r})$. By (b) the conclusion follows. □

When the normal vectors of a surface are directly available, the method described in part (c) of the lemma is quite efficient. This would be the case for definitions of the form

$f(x,y,z,w) = 0 \quad g(x,y,z,w)$. When the tangent vectors are directly available, the method described in part (b) can be used. This would apply for parametric surface definitions.

Lemma 1 cannot be applied to calculate the normal in all cases. If $\mathbf{r}$ lies in span$(\mathbf{t_1}, \mathbf{t_2})$, the calculated $\mathbf{n}$ will be a zero vector. This case cannot be avoided with a different choice of $\mathbf{t_1}, \mathbf{t_2}$ or $\mathbf{n_1}, \mathbf{n_2}$, and is a property of the projection.

We define *silhouette points* on a surface S with respect to a projection v as those non-singular points whose tangent space reduces dimension under the projection v.

Lemma 2 Suppose that $\mathbf{t_1}, \mathbf{t_2}$ are two linearly independent tangent vectors at point $\mathbf{p}$ on a surface, that v is a projection with center $1/r$, and that $\mathbf{r} = 1/r - \mathbf{p}$ is the ray from $\mathbf{p}$ to the projection center. Then $\mathbf{p}$ is a silhouette point with respect to v if and only if $\mathbf{t_1}, \mathbf{t_2}, \mathbf{r}$ are linearly dependent.

Proof: First we assume that $\mathbf{t_1}, \mathbf{t_2}, \mathbf{r}$ are linearly dependent, and so $\mathbf{r}$ can be expressed as linear combination of $\mathbf{t_1}, \mathbf{t_2}$. Let $\mathbf{a_i} \quad (i = 1,2)$ be the natural extrusion of the vector $v(\mathbf{p} + \mathbf{t_i}) - v(\mathbf{p})$. By Lemma 1 (a) $\mathbf{a_i}, \mathbf{t_i}, \mathbf{r}$ are linearly dependent. Then,

$$\begin{aligned} \mathbf{a_1} &= \alpha_1 \mathbf{t_1} + \alpha_2 \mathbf{t_2} \\ \mathbf{a_2} &= \beta_1 \mathbf{t_1} + \beta_2 \mathbf{t_2} \end{aligned}$$

Consider the fourth coordinates, a_{1w} and a_{2w}, of the above equations. Then $a_{1w} = a_{2w} = 0$ by definition. But t_{1w} and t_{2w} cannot be both zero because r_w is not zero. So

$$\begin{vmatrix} \alpha_1 & \alpha_2 \\ \beta_1 & \beta_2 \end{vmatrix} = 0$$

Hence $\mathbf{a_1}, \mathbf{a_2}$ are in the same direction and so are their natural projections. The direction will not change if we choose any other pair of vectors within the tangent plane, which means the tangent plane becomes a line under projection. So $\mathbf{p}$ is a silhouette point.

Now assume $\mathbf{t_1}, \mathbf{t_2}, \mathbf{r}$ are linearly independent. Again by Lemma 1 (a) we have

$$\begin{aligned} \mathbf{a_1} &= \alpha_1 \mathbf{t_1} + \alpha_2 \mathbf{r} \\ \mathbf{a_2} &= \beta_1 \mathbf{t_2} + \beta_2 \mathbf{r} \end{aligned}$$

Since both α_1 and β_1 are nonzero, $\mathbf{a_1}, \mathbf{a_2}$ cannot be linearly dependent. The tangent plane does not reduce dimension under projection. So $\mathbf{p}$ is not a silhouette point. □

The picture we actually see is two-dimensional. So we are more concerned with the silhouette points with respect to a projection from $\mathcal{R}^4$ to $\mathcal{R}^2$. Define v as:

$$v(\mathbf{p}) = v(p_x, p_y, p_z, p_w) = (\frac{p_x}{1 - r_1 p_w - r_2 p_x}, \frac{p_y}{1 - r_1 p_w - r_2 p_z})^T$$

where $\mathbf{l}/r_1$ and $\mathbf{k}/r_2$ are the two projection centers. Note that the projection order is irrelevant: We get the same result mapping first from $\mathcal{R}^4$ to the XYZ-hyperplane and then to the XY-plane, as we obtain by mapping first from $\mathcal{R}^4$ to the XYW-hyperplane and then to the XY-plane.

Lemma 3 Suppose that $\mathbf{t_1}, \mathbf{t_2}$ $(\mathbf{n_1}, \mathbf{n_2})$ are two linearly independent tangent (normal) vectors at a nonsingular point $\mathbf{p}$ on a 2-surface S, that v is a projection from $\mathcal{R}^4$ to $\mathcal{R}^2$ with the centers $\mathbf{l}/r_1$ and $\mathbf{k}/r_2$, and that $\mathbf{r_1} = \mathbf{l}/r_1 - \mathbf{p}$, $\mathbf{r_2} = \mathbf{k}/r_2 - \mathbf{p}$ are the two rays from the point $\mathbf{p}$ to the two projection centers. Then,

(a) if $\mathbf{a_i} = \varepsilon_{ZW}(v(\mathbf{p} + \mathbf{t_i}) - v(\mathbf{p}))$, then $\mathbf{a_i}, \mathbf{t_i}, \mathbf{r_1}, \mathbf{r_2}$ are linearly dependent for $i = 1, 2$.

(b) $\mathbf{p}$ is a silhouette point with respect to v if and only if $\mathbf{t_1}, \mathbf{t_2}, \mathbf{r_1}, \mathbf{r_2}$ are linearly dependent.

(c) let $\mathbf{m_1}, \mathbf{m_2}$ be two linearly independent vectors in the plane orthogonal to the plane span$(\mathbf{r_1}, \mathbf{r_2})$, then $\mathbf{p}$ is a silhouette point if and only if $\mathbf{n_1}, \mathbf{n_2}, \mathbf{m_1}, \mathbf{m_2}$ are linearly dependent.

Proof: (a) Since there is no difference for $i = 1, 2$ we drop the subscripts of $\mathbf{t_i}$ and $\mathbf{a_i}$ temporarily.

$$\mathbf{a} = \begin{pmatrix} \frac{p_x + t_x}{1 - r_1(p_w + t_w) - r_2(p_z + t_z)} - \frac{p_x}{1 - r_1 p_w - r_2 p_z} \\ \frac{p_y + t_y}{1 - r_1(p_w + t_w) - r_2(p_z + t_z)} - \frac{p_y}{1 - r_1 p_w - r_2 p_z} \\ 0 \\ 0 \end{pmatrix}$$

Then,

$$\alpha_1 \mathbf{a} + \alpha_2 \mathbf{t} + \alpha_3 \mathbf{r_1} + \alpha_4 \mathbf{r_2} = 0$$

where

$$\begin{aligned} \alpha_1 &= [1 - r_1(p_w + t_w) - r_2(p_z + t_z)](1 - r_1 p_w - r_2 p_z) \\ \alpha_2 &= -(1 - r_1 p_w - r_2 p_z) \end{aligned}$$

$$\begin{aligned}\alpha_3 &= r_1 t_w + r_1 r_2 (t_z p_w - t_w p_z) \\ \alpha_4 &= r_2 t_z - r_1 r_2 (t_z p_w - t_w p_z)\end{aligned}$$

Clearly, α_2 is nonzero because otherwise $\mathbf{p}$ is projected to infinity. Moreover, by choosing the length of $\mathbf{t}$ appropriately, α_1 is also nonzero.

(b) First assume that $\mathbf{t_1}, \mathbf{t_2}, \mathbf{r_1}, \mathbf{r_2}$ are linearly dependent. Since span$(\mathbf{t_1}, \mathbf{t_2})$ is of dimension two, we can find another vector $\mathbf{b}$ such that $\mathbf{r_1}, \mathbf{r_2}$ each is a linear combination of $\mathbf{t_1}, \mathbf{t_2}, \mathbf{b}$. By (a) we have

$$\begin{aligned}\mathbf{a_1} &= \alpha_1 \mathbf{t_1} + \alpha_2 \mathbf{t_2} + \alpha_3 \mathbf{b} \\ \mathbf{a_2} &= \beta_1 \mathbf{t_1} + \beta_2 \mathbf{t_2} + \beta_3 \mathbf{b}\end{aligned}$$

That means $\mathbf{a_1}, \mathbf{a_2}$ lie in the intersection of span$(\mathbf{t_1}, \mathbf{t_2}, \mathbf{b})$ and span$(\mathbf{i}, \mathbf{j})$. This intersection must be of dimension lower than two because otherwise span$(\mathbf{t_1}, \mathbf{t_2}, \mathbf{b})$ is equal to span$(\mathbf{i}, \mathbf{j}, \mathbf{c})$ for some vector $\mathbf{c}$, which causes $\mathbf{r_1}, \mathbf{r_2}$ to be parallel within ZW-plane, and so $\mathbf{p}$ is mapped into infinity. Consequently $\mathbf{p}$ is a silhouette point.

Now assume that $\mathbf{t_1}, \mathbf{t_2}, \mathbf{r_1}, \mathbf{r_2}$ are linearly independent. Again by (a) we have

$$\begin{aligned}\mathbf{a_1} &= \alpha_1 \mathbf{t_1} + \alpha_2 \mathbf{r_1} + \alpha_3 \mathbf{r_2} \\ \mathbf{a_2} &= \beta_1 \mathbf{t_2} + \beta_2 \mathbf{r_1} + \beta_3 \mathbf{r_2}\end{aligned}$$

Since both α_1 and β_1 are nonzero, $\mathbf{a_1}, \mathbf{a_2}$ must be linearly independent, and so are their natural projections into $\mathcal{R}^2$. Therefore $\mathbf{p}$ is not a silhouette point.

(c) It is sufficient to prove that $\mathbf{t_1}, \mathbf{t_2}, \mathbf{r_1}, \mathbf{r_2}$ are linearly independent if and only if $\mathbf{n_1}, \mathbf{n_2}, \mathbf{m_1}, \mathbf{m_2}$ are linearly independent. Assuming that $\mathbf{t_1}, \mathbf{t_2}, \mathbf{r_1}, \mathbf{r_2}$ are linearly independent, they form a base of $\mathcal{R}^4$. Then both $\mathbf{n_1}$ and $\mathbf{n_2}$ are linear combinations of $\mathbf{r_1}, \mathbf{r_2}$, and both $\mathbf{m_1}$ and $\mathbf{m_2}$ are linear combinations of $\mathbf{t_1}, \mathbf{t_2}$. Therefore $\mathbf{n_1}, \mathbf{n_2}, \mathbf{m_1}, \mathbf{m_2}$ must be linearly independent. The converse direction is symmetric. □

We will explain the application of the above lemma by the examples in Section 4.

3 Orientation Specification

3.1 Object orientation specification

Rigid body rotation in 4D space can be expressed by an orthonormal 4×4 matrix A called the *direction cosine matrix*:

$$\mathbf{p} = A\mathbf{p_1}$$

In the equation, $\mathbf{p} = (x, y, z, w)^T$ is a vector expressed in the world coordinate system and $\mathbf{p_1} = (x_1, y_1, z_1, w_1)^T$ is the same vector expressed in the body-fixed coordinate system. Among the sixteen elements in the direction cosine matrix only six are independent. It will be convenient to specify the orientation of an object by six independent parameters. One way is to write the direction cosine matrix as a product of six basic rotation matrices. A *basic rotation* is a one-parameter rotation within a plane spanned by two coordinate axes. For example,

$$R_{\mathrm{YZ}}(\theta) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

is a rotation within the plane span(j,k) whose normal plane is span(i,l). The six basic rotations should be chosen systematically such that the geometric relationship is easy to explain and easy to remember. We have extended into 4D the Euler angles commonly used in 3D rigid-body kinematics [17].

3D Euler angles specify the orientation of objects as three successive rotations, first about the z-axis, then about the x-axis, and finally about the z-axis again. Note that these axes are body-fixed. If the 3D Euler angles are $\theta_1, \theta_2, \theta_3$, the relationship between the world coordinates $\mathbf{p}$ and the body-fixed coordinates $\mathbf{p_3}$ is given by:

$$\mathbf{p} = R_{\mathrm{XY}}(\theta_1) R_{\mathrm{YZ}}(\theta_2) R_{\mathrm{XY}}(\theta_3) \mathbf{p_3}$$

We conceptualize Euler angles as two separate rotation phases: In the first phase, specified by $R_{\mathrm{XY}}(\theta_1)R_{\mathrm{YZ}}(\theta_2)$, the body-fixed z-axis is oriented in $\mathcal{R}^3$ and put into its final position. In the second phase, a single rotation in $\mathcal{R}^2$ orthogonal to the already oriented body-fixed z-axis brings the entire object into its final orientation. Thus the second phase is expressed as $R_{\mathrm{XY}}(\theta_3)$.

To clearly distinguish the two phases, we add a number subscript to the rotations, indicating the subspace that is rotated. Thus the coordinate relationship equation by 3D Euler angles can be rewritten as:

$$\mathbf{p} = R_{\mathrm{XY3}}(\theta_1)R_{\mathrm{YZ3}}(\theta_2)R_{\mathrm{XY2}}(\theta_3)\mathbf{p_3}$$

Note that conceptually R_{XY3} is a rotation in $\mathcal{R}^3$ and R_{XY2} is a rotation in $\mathcal{R}^2$, but their matrix forms are the same if both are written as an $n \times n$ matrix, for $n > 2$.

We extend Euler angles into $\mathcal{R}^4$ as follows:

1. Orient the body-fixed w-axis of the object by three rotations in $\mathcal{R}^4$:

$$R_{\mathrm{XY4}}(\theta_1)R_{\mathrm{YZ4}}(\theta_2)R_{\mathrm{ZW4}}(\theta_3)$$

2. Orient the body-fixed z-axis of the object in the 3D subspace orthogonal to the already oriented w-axis, by two rotations in $\mathcal{R}^3$: $R_{\mathrm{XY3}}(\theta_4)R_{\mathrm{YZ3}}(\theta_5)$.

3. Orient the remaining body-fixed x- and y-axes in the 2D subspace orthogonal to the already oriented z- and w-axes, by a single rotation in $\mathcal{R}^2$: $R_{\mathrm{XY2}}(\theta_6)$.

It is clear that with this conceptualization Euler angle specification can be naturally extended to any dimension. The Euler angle specification in dimension $n+1$ is done inductively by:

1. Orienting the new x_{n+1}-axis using n angles.

2. Orienting the n-dimensional subspace orthogonal to the x_{n+1}-axis with the n-dimensional Euler angle specification.

Note that the two dimensional Euler angle is simply a single rotation within a plane.

In 4D space, the relationship between the world coordinates $\mathbf{p}$ and the body-fixed coordinates $\mathbf{p_6}$ is:

$$\mathbf{p} = R_{\mathrm{XY4}}(\theta_1)R_{\mathrm{YZ4}}(\theta_2)R_{\mathrm{ZW4}}(\theta_3)R_{\mathrm{XY3}}(\theta_4)R_{\mathrm{YZ3}}(\theta_5)R_{\mathrm{XY2}}(\theta_6)\mathbf{p_6}$$

The rotations caused by each phase are shown in Figure 1. In the following equations $\mathbf{p_i} = (x_i, y_i, z_i, w_i)^T$ is the same vector expressed in the i-th coordinates system, that coincides with the body-fixed coordinate system at the end of the i-th rotation. Particularly, $\mathbf{p_3}, \mathbf{p_5}, \mathbf{p_6}$ are the coordinates at the end of the three rotation phases, respectively.

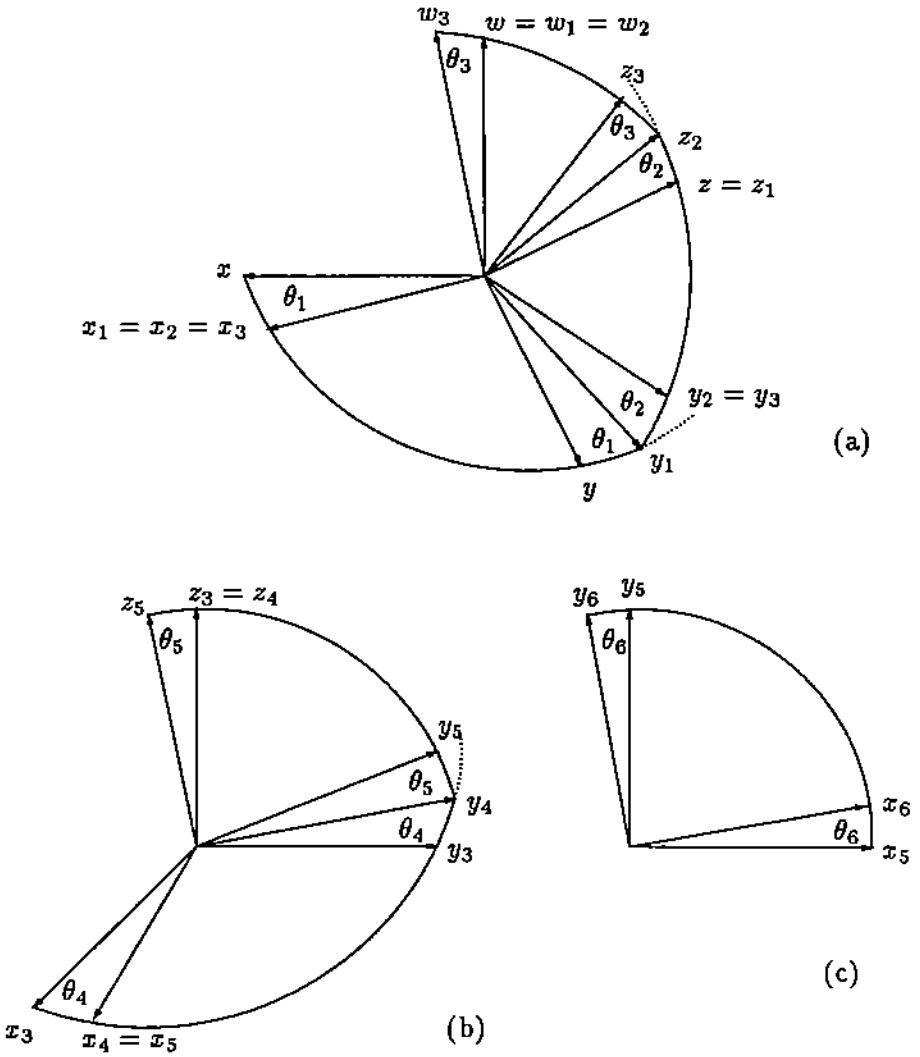


Figure 1: Euler Angles: (a) $\theta_1, \theta_2, \theta_3$ in $\mathcal{R}^4$ (b) θ_4, θ_5 in $\mathcal{R}^3$ (c) θ_6 in $\mathcal{R}^2$

$$\begin{pmatrix} x \\ y \\ z \\ w \end{pmatrix} = R_{XY4}(\theta_1)R_{YZ4}(\theta_2)R_{ZW4}(\theta_3)\begin{pmatrix} x_3 \\ y_3 \\ z_3 \\ w_3 \end{pmatrix} = \begin{pmatrix} c_1 & -s_1c_2 & s_1s_2c_3 & -s_1s_2s_3 \\ s_1 & c_1c_2 & -c_1s_2c_3 & c_1s_2s_3 \\ 0 & s_2 & c_2c_3 & -c_2s_3 \\ 0 & 0 & s_3 & c_3 \end{pmatrix}\begin{pmatrix} x_3 \\ y_3 \\ z_3 \\ w_3 \end{pmatrix}$$

$$\begin{pmatrix} x_3 \\ y_3 \\ z_3 \\ w_3 \end{pmatrix} = R_{XY3}(\theta_4)R_{YZ3}(\theta_5)\begin{pmatrix} x_5 \\ y_5 \\ z_5 \\ w_5 \end{pmatrix} = \begin{pmatrix} c_4 & -s_4c_5 & s_4s_5 & 0 \\ s_4 & c_4c_5 & -c_4s_5 & 0 \\ 0 & s_5 & c_5 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} x_5 \\ y_5 \\ z_5 \\ w_5 \end{pmatrix}$$

$$\begin{pmatrix} x_5 \\ y_5 \\ z_5 \\ w_5 \end{pmatrix} = R_{XY2}(\theta_6)\begin{pmatrix} x_6 \\ y_6 \\ z_6 \\ w_6 \end{pmatrix} = \begin{pmatrix} c_6 & -s_6 & 0 & 0 \\ s_6 & c_6 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} x_6 \\ y_6 \\ z_6 \\ w_6 \end{pmatrix}$$

The direction cosine matrix in $\mathcal{R}^4$ can also be considered as the world coordinates of the four base vectors i, j, k, l of the body-fixed coordinate system at the end of the rotation:

$$A = \begin{pmatrix} i_x & j_x & k_x & l_x \\ i_y & j_y & k_y & l_y \\ i_z & j_z & k_z & l_z \\ i_w & j_w & k_w & l_w \end{pmatrix}$$

It is easy to verify that the corresponding Euler angles can be found as follows:

Step 1: Calculate $\theta_1, \theta_2, \theta_3$ from the fourth column via

$$\begin{aligned} \theta_1 &= -\arctan(\frac{l_x}{l_y}) \\ \theta_2 &= -\arctan(\frac{\sqrt{l_x^2 + l_y^2}}{l_z}) \\ \theta_3 &= -\arctan(\frac{\sqrt{l_x^2 + l_y^2 + l_z^2}}{l_w}) \end{aligned}$$

Step 2: Left-multiply the third column $(k_x, k_y, k_z, k_w)^T$ by $R_{ZW4}(-\theta_3)R_{YZ4}(-\theta_2)R_{XY4}(-\theta_1)$ to transform it into $(k_{x_3}, k_{y_3}, k_{z_3}, 0)^T$, i.e. the body-fixed vector k expressed in the third coordinate system. Then calculate θ_4, θ_5 via

$$\theta_4 = -\arctan(\frac{k_{x_3}}{k_{y_3}})$$

$$\theta_5 \quad = \quad -\arctan(\frac{\sqrt{k_{x_3}^2 + k_{y_3}^2}}{k_{z_3}})$$

Step 3: Left-multiply the second column $(j_x, j_y, j_z, j_w)^T$ by

$$R_{\mathrm{YZ3}}(-\theta_5)R_{\mathrm{XY3}}(-\theta_4)R_{\mathrm{ZW4}}(-\theta_3)R_{\mathrm{YZ4}}(-\theta_2)R_{\mathrm{XY4}}(-\theta_1)$$

to transform it into $(j_{x_5}, j_{y_5}, 0, 0)^T$, i.e. the body-fixed vector j expressed in the fifth coordinate system. Then calculate θ_6 via

$$\theta_6 = -\arctan(\frac{j_{x_5}}{j_{y_5}})$$

Note that steps 2 and 3 can be combined into one by 3D inverse Euler angle formulas [17]. We stress again the similarity between the rotation chains in $\mathcal{R}^4, \mathcal{R}^3, \mathcal{R}^2$. Within the rotation chain in $\mathcal{R}^i \quad (i = 4, 3, 2)$, only the angle of $R_{\mathrm{XY}i}$ has a full range of 2π, others are restricted in a range of π to eliminate ambiguities. That is why there is no $\pm$ before the square roots. Figure 2 shows the body-fixed vector l and its projections in $\mathcal{R}^3$ and $\mathcal{R}^2$. The angles $\theta_1, \theta_2, \theta_3$ can be considered as polar coordinates extended in to $\mathcal{R}^4$.

Note that numerical difficulties arise when both the numerator and the denominator in the argument of arctan are close to zero. This can happen when at least one of the angles $\theta_2, \theta_3, \theta_5$ is close to $k\pi \quad (k = 0, \pm 1, \ldots)$, and is a well-known drawback of Euler angles [17].

3.2 Projection Centers Specification

We specify the projection centers in the same way as object orientation. Consider the screen for display as a rigid body and attach to it a coordinate system (x_6, y_6, z_6, w_6) in such a way that the first projection center, called the *sensor*, is along the direction of w_6-axis, and the second projection center, called the *eye*, is along the direction of z_6-axis. Using Euler angles, the sensor's direction is determined by $\theta_1, \theta_2, \theta_3$. In effect, these angles are polar coordinates extended in $\mathcal{R}^4$. The eye's direction is determined by θ_4, θ_5, which are the usual polar coordinates in 3D, also called the azimuth and incidence angles. The *twist angle*, θ_6, only affects the orientation of the final 2D picture in the projection plane. In addition to these angles, the user interface offers other parameters such as the reciprocal of the distance from the sensor or the eye to the origin, the field-of-view angle, the distance

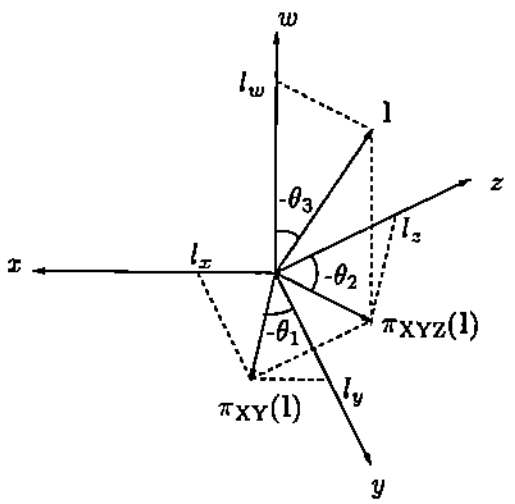


Figure 2: Body-fixed vector l and its projections in $\mathcal{R}^3$ and $\mathcal{R}^2$

and the pitch of the front clipping plane. All these quantities are with respect to the (x_6, y_6, z_6, w_6)-coordinate system.

4 Examples

4.1 Hypersphere

The unit hypersphere S^3 in $\mathcal{R}^4$ has the equation:

$$x^2 + y^2 + z^2 + w^2 - 1 = 0$$

If the hypersphere is projected into $\mathcal{R}^3$, we get an ellipsoid, which contains little information unless there is a 4D shading model resulting in different intensities at different interior points of the ellipsoid. A more convenient way to visualize the hypersphere is to mark or color certain areas on it in analogy to drawing longitude and latitude circles on a sphere S^2. We can obtain the "longitude" and "latitude" on a hypersphere by the Hopf map [11, 14].

$$\begin{aligned} h : S^3 &\longrightarrow S^2 \\ (x, y, z, w) &\longmapsto (X, Y, Z) \end{aligned}$$

where

$$X = 2(xy + zw)$$

$$
\begin{aligned}
Y &= 2(xw - yz) \\
Z &= (x^2 + z^2) - (y^2 + w^2)
\end{aligned}
$$

For example, given a latitude $Z = a$ on S^2, its inverse image on S^3 is a 2-torus, $S^1 \times S^1$:

$$
\begin{aligned}
x^2 + z^2 - \frac{1+a}{2} &= 0 \\
y^2 + w^2 - \frac{1-a}{2} &= 0
\end{aligned}
$$

In Figures 6–8, the inverse images of the latitudes $Z = -\frac{\sqrt{2}}{2}$, $Z = 0$, $Z = \frac{\sqrt{2}}{2}$ are represented by the green, gray and red surfaces respectively. The inverse image of the south and north poles, $Z = -1$ and $Z = 1$ are represented by magenta and blue curves. We also display x, y, z, w axes as straight lines of red, orange, green, and blue, respectively. The w-axis is now invisible because the sensor is positioned on it. The hypersphere is translated by 1 along the positive w-axis so that when the sensor's distance is 2, the hypersphere is mapped to the whole $\mathcal{R}^3$. Figures 6–8 show three snap shots of an animation where the sensor is moving from infinity towards the hypersphere. The moments shown are when the sensor's distance is infinity, 3, and 2, respectively. As the sensor approaches, the circumference of the green 2-surface enlarges significantly while the red one enlarges only moderately. This animation can be understood by comparing it with an animation in 3D: Look at Figure 6 and consider the surfaces as objects in 3D space. Imagine that your eye is moving on the y-axis from infinity towards the origin, then you will see a similar animation but happens in 3D space. Figure 8 also shows how, by using the tilted front clipping plane, some hidden details can be revealed. See also [11] for a visualization of 4D rotation.

4.2 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 [5, 8] as a set of equations:

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

where

$$
\nabla_{uv} g = (\frac{\partial g}{\partial u}, \frac{\partial g}{\partial v})^T
$$

$$\bar{\mathbf{t}} \;=\; (\frac{\partial f}{\partial v}, -\frac{\partial f}{\partial u})^T$$

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

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

Note that the condition C' is equivalent to $\frac{\partial h}{\partial t}=0$. If the greatest common divisor $\phi(t)=$ GCD$(u'(t),v'(t))$ is not a constant, the condition C' can be further simplified as [5]:

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

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 [5], or using Gröbner basis [8]. The offset curve can also be traced numerically in $\mathcal{R}^4$ or $\mathcal{R}^3$ by the method described in [2].

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 XY-plane (see Figure 3). 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 4).

We tried to explain these phenomena by means of 4D surface visualization. The equations $g=0, \quad f=0$ are two hypersurfaces in XYUV-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 Lemma 3 (c) it is a silhouette point with respect to an

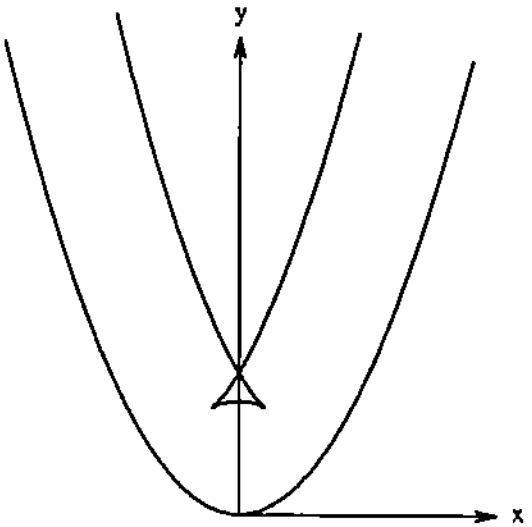


Figure 3: Curve $y - x^2 = 0$ and its offset curve by 1

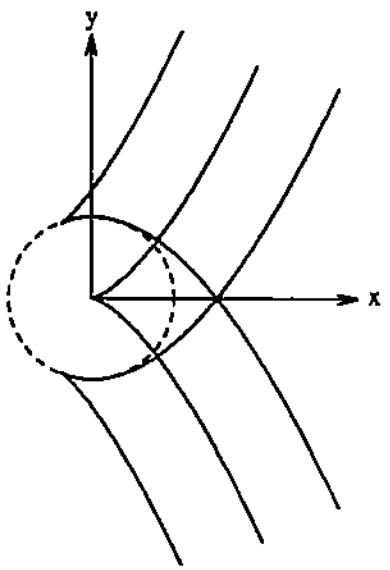


Figure 4: Curve $y^2 - x^3 = 0$ and its offset curve by 1

orthographic projection with two centers along u- and v-axes. The silhouette points form a curve on the pipe-like surface S in $\mathcal{R}^4$. In Figure 9 and 10 we show the 2-surface S and the silhouette curve corresponding to the offset curve in Figure 3. In $\mathcal{R}^4$ the curve is smooth without cusps or self-intersections as we can see in Figures 10 from 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 they are not silhouettes. They are exactly the additional points described in the second phenomenon. In Figure 11 and 12 the 2-surface S and the silhouette curve corresponding to the offset curve in Figure 4 are shown in different viewing directions. The silhouette curves are still smooth without cusps or self-intersections. But the 2-surface is not a smooth "pipe". The singular points form a circle corresponding to the dashed circle in Figure 4.

If the curve f has a parametric form, the offset curve can be traced in XYT-space. The two equations $h = 0, \quad \frac{\partial h}{\partial t} = 0$ are two surfaces and their intersection is a curve. Note that $\frac{\partial h}{\partial t} = 0$ is equivalent to $\nabla h \cdot \mathbf{k} = 0$. It also means that the intersection curve is the silhouette on the surface h with respect to a 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 ∇h is always a nonzero vector. The dashed circle in Figure 2 is actually another branch of silhouette curve as shown in Figures 13 and 14.

If the greatest common divisor $\phi(t)$ is not a constant, the condition $\nabla h \cdot \mathbf{k} = 0$ 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 pipe-like surface $h = 0$ with the planes $t = t_i$ perpendicular to the t-axis, where t_i's are the roots of $\phi(t)$. The other factor is the same as condition C'', and represents the silhouette curve branches corresponding to the offset curve.

4.3 Collision Detection

Cameron [3] proposes an algorithm to detect collisions based on 4D model. The basic idea is that "if an object can be represented by a set-combination in a CSG scheme. and the primitive objects can be extruded (into 4D) in this scheme, then the extrusion of the object is the set-combination of the extrusions of the primitives". The extruded object means the

object in motion considered in XYZT-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. In 3D the primitives can be the half space to one side of an oriented surface. The extrusion of the surface is a hypersurface in 4D space.

We discuss some points in the design of an algorithm to test whether two hypersurfaces intersect. The hypersurfaces are

$$f(x, y, z, t) = 0, \quad g(x, y, z, t) = 0$$

The intersection is a 2-surface in $\mathcal{R}^4$ and can be examined with our visualization system.

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. We consider the 2-surface that is the intersection of the two hypersurfaces:

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

Note that as the radii and speeds change, the topology of the resulting intersection 2-surface can be quite different. Figure 15 shows the case $r_c = r_s = 1$, $v_c = 0$, $v_s = 1$. The green wire mesh represents the the cylinder at rest, i.e. intersection of the cylinder with the hyperplane $t = 0$. Since the sensor's position is just a little off the t axis, the intersection 2-surface resembles the sweep of the intersection curve in $\mathcal{R}^3$. Figure 16 shows the case $r_c = 1$, $r_s = 0.7$, $v_c = 0.2$, $v_s = 1$. Since the radius of the sphere is smaller, the intersection 2-surface has two separate components. Figure 17 shows the case $r_c = 1$, $r_s = 1.2$, $v_c = 0$, $v_s = 1$. The sphere has a larger radius. Although the 2-surface is connected, 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.

One advantage of using 4D geometry is that the intersection is nonempty as long as there is a collision. The commonly used method to test intersection in $\mathcal{R}^3$ at a series of sampling times may not be able to detect the following cases:

1. The interpenetration begins and ends between two consecutive sampling times.

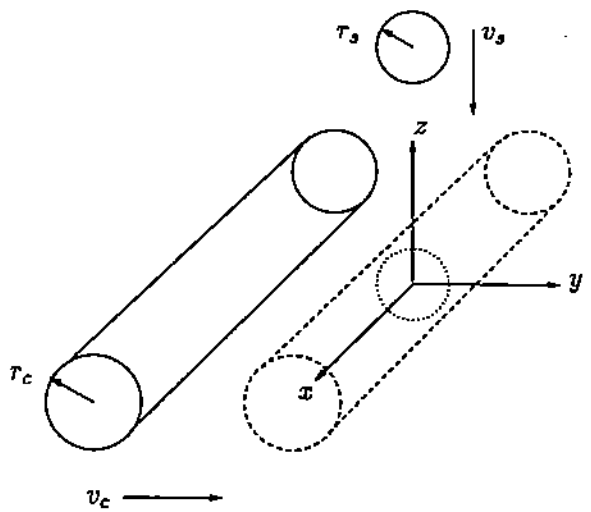


Figure 5: A cylinder and a sphere in motion

2. If in $\mathcal{R}^3$ the algorithm does only boundary intersection test but not containment test, an interpenetration that lasts longer than one sampling period may also be overlooked.

Often we only need to find the *initial colliding point*, i.e. the point in the intersection 2-surface with the smallest value of t. Since we do not consider "patches" or "trimmed surface", a necessary condition for initial colliding point is:

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

Together with $f = 0$, $g = 0$ it represents a zero-dimensional point set in $\mathcal{R}^4$. If f, g are polynomials, this set can be solved by algebraic methods, e.g. using the Gröbner Basis techniques [8]. When the object motion includes a rotation, f and g are no longer algebraic hypersurfaces. Such a set of equations often is hard to solve. One possible method is to relax the condition C as:

$$C': \qquad \frac{\partial f}{\partial x}\frac{\partial g}{\partial y} - \frac{\partial f}{\partial y}\frac{\partial g}{\partial x} = 0$$

Together with $f = 0$, $g = 0$ it represents a curve in $\mathcal{R}^4$ and can be traced numerically. The point on the curve with the smallest t is the initial colliding point.

The condition C' can be rewritten as:

$$\det(\mathbf{l}, \mathbf{k}, \nabla f, \nabla g) = 0$$

According to Lemma 3 (c), this is the silhouette curve on the 2-surface with respect to a orthographic projection along x- and y-axes. It also works if $\mathbf{k}$ is replaced by any nonzero linear combination of $\mathbf{i}, \mathbf{j}, \mathbf{k}$, corresponding to a projection in other directions. It is important to note the following phenomena when tracing the silhouette curve:

1. The 2-surface may have several separate components. Then the silhouette curve must have several separate branches as seen in Figure 16.

2. If the 2-surface is connected, it is nevertheless possible that the silhouette curve has several separate branches. See also Figure 17.

Finding the initial colliding point can also be considered as a constrained nonlinear programming problem optimizing t (see, e.g. [12]). Investigating the topology of the 2-surface by means of 4D visualization may help to design faithful algorithms.

5 Conclusion

We have discussed two aspects of visualization of 2-surfaces in 4D space: first, some geometric issues related to developing an interactive 4D visualization system, and second, some examples illustrating the use of such a system.

We have presented a method suitable for implementation on conventional z-buffer based graphics workstations. The input 2-surfaces are polygonalized in 4D space. The 4D polygons, with each vertex attached to two linearly independent normal vectors, are then projected into 3D space and fed into the 3D graphics engine. The normal of the projected surface can be calculated efficiently by Lemma 1. The main problem in devising a user interface is how to specify the orientation of objects and the directions of projection. They can be done uniformly by the Euler angles when extended properly into the 4D space.

In Lemma 2 and 3 we have also discussed the conditions for a point on the 2-surface to be a silhouette point with respect to a projection. This concept plays an important role in the explanation of our examples. Offset curves, when traced in 4D space, can be considered as silhouette curves of a pipe-like 2-surface. Silhouette curves also roughly describe the shape of the 2-surface that is the intersection of two moving surfaces in 3D space. We pointed out some phenomena to be noticed in designing faithful algorithms for tracing offset curves and searching for initial colliding points. Our experience convinced us that such a 4D visualization system is useful for understanding four dimensional geometry.

Some problems remain to be solved. For example, the quality of the pictures may not be good enough because of the resolution of the polygonalization. When the sensor or the eye is too close to the 2-surface, this problem becomes especially acute, since the nearby polygons are seen much larger than those further away.

References

[1] Allgower, E. L. and Gnutzmann S., "An Algorithm for Piecewise Linear Approximation of Implicitly Defined Two-dimensional Surfaces", *SIAM J. Numerical Analysis*, Vol. 24, No. 2, pp. 452–469, 1987

[2] Bajaj, C., Hoffmann, C., Hopcroft, J. and Lynch, R., "Tracing Surface Intersections", *Computer Aided Geometric Design, 5*, pp. 285–307, 1988

[3] Cameron, S. A., Modelling Solids in Motion, Ph.D. Thesis, University of Edinburgh, 1984

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

[5] Farouki, R. T. and Neff, C. A., "Some Analytic and Algebraic Properties of Plane Offset Curves", Rept. RC 14364, IBM Yorktown Heights, 1989

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

[7] Glassner, A. S., An Introduction to Ray Tracing, Academic Press, 1989

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

[9] Hoffmann, C. M., "A Dimensionality Paradigm for Surface Interrogations", *Computer Aided Geometric Design*, to appear, 1990

[10] Koçak, H., Bisshopp, F., Banchoff, T., and Laidlaw, D., "Topology and Mechanics with Computer Graphics: Linear Hamiltonian Systems in Four Dimensions", *Advances in Applied Mathematics*, Vol. 7, pp. 282–308, 1986

[11] Koçak, H. and Laidlaw, D., "Computer Graphics and the Geometry of S^3", The Mathematical Intelligence, Vol. 9, No. 1, pp. 8–10, 1987

[12] Luenberger, D. G., Linear and Nonlinear Programming, Addison-Wesley, 1984

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

[14] Pinkall, U., "Hopf tori in S^3", *Inventiones Mathematicae*, Vol. 81, pp. 379–386, 1985

[15] Rossignac, J. R., "Considerations on the Interactive Rendering of Four-Dimensional Volumes", *CH Volume Visualization Workshop*, pp. 67–76, 1989

[16] Semple, J. G. and Kneebone, G. T., Algebraic Projective Geometry, The Clarendon Press, Oxford, 1952

[17] Wittenburg, J., Dynamics of systems of rigid bodies, B. G. Teubner Stuttgart, 1977

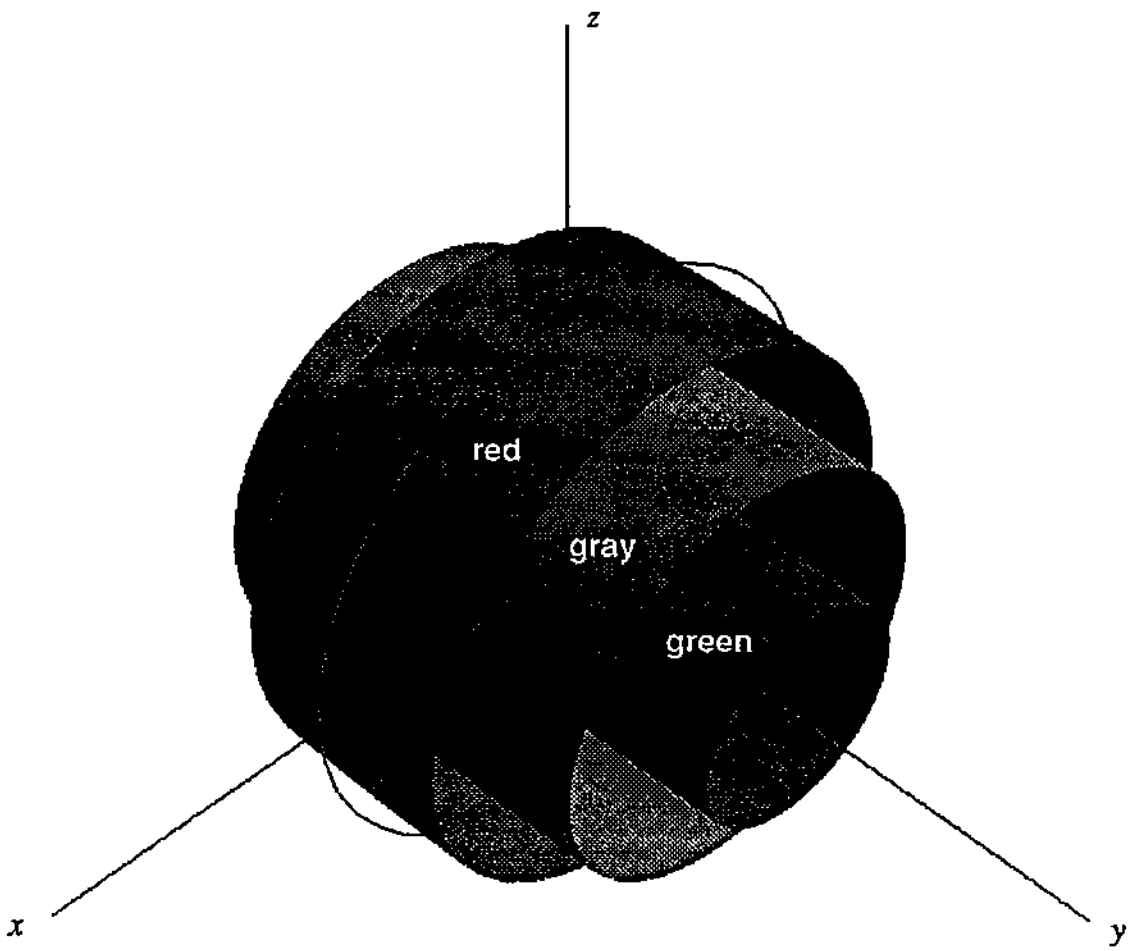


Figure 6 Hypersphere viewing from the sensor at infinity

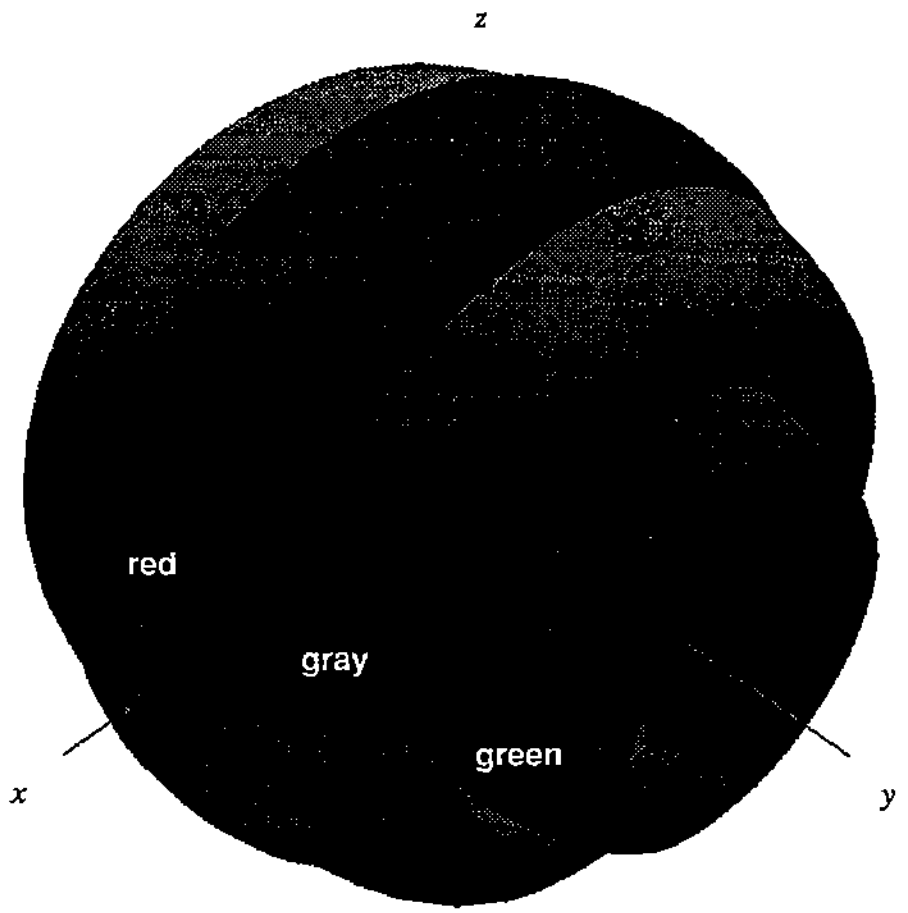


Figure 7 Hypersphere viewing from the sensor at distance 3

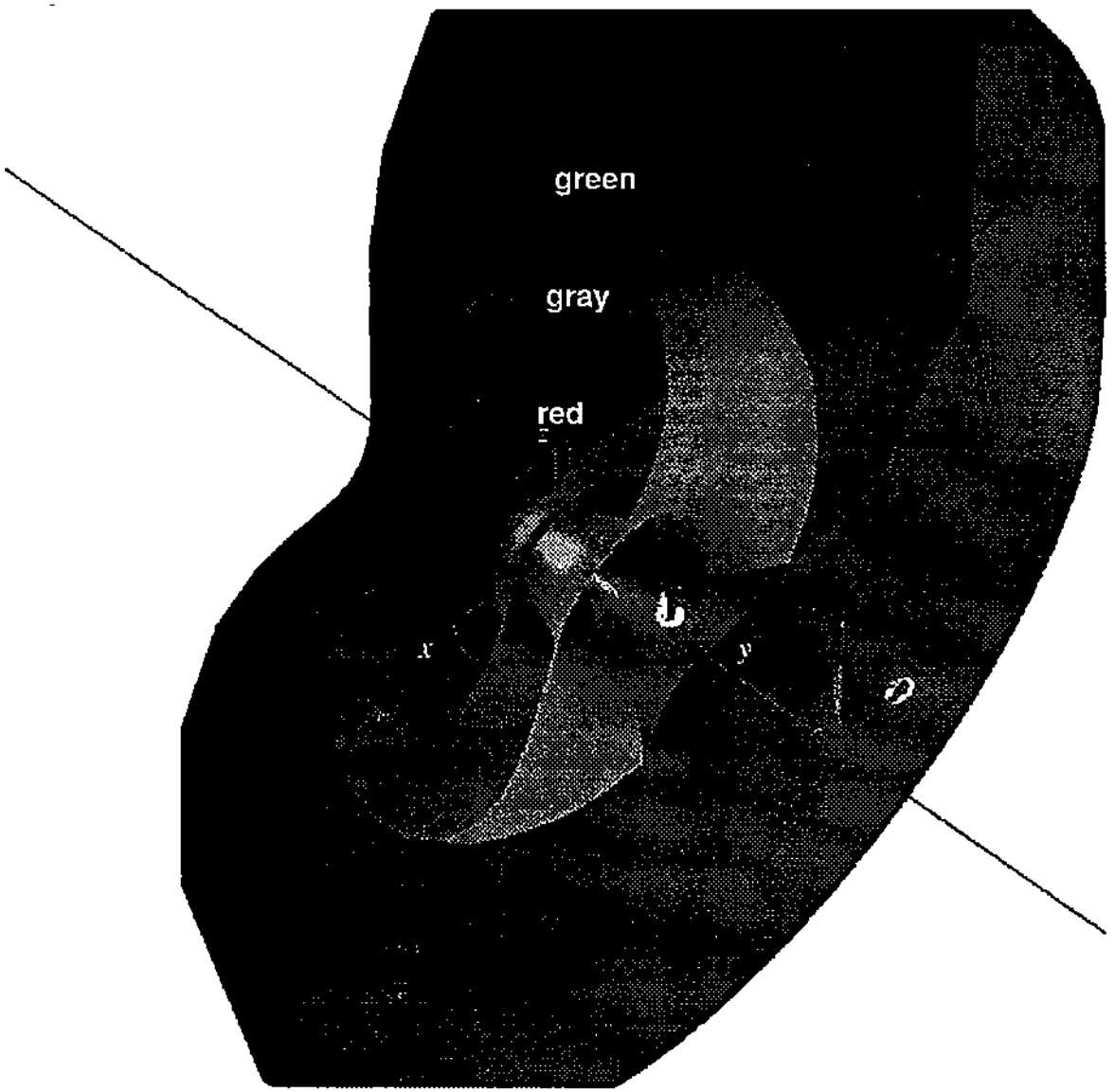


Figure 8 Hypersphere viewing from the sensor at distance 2

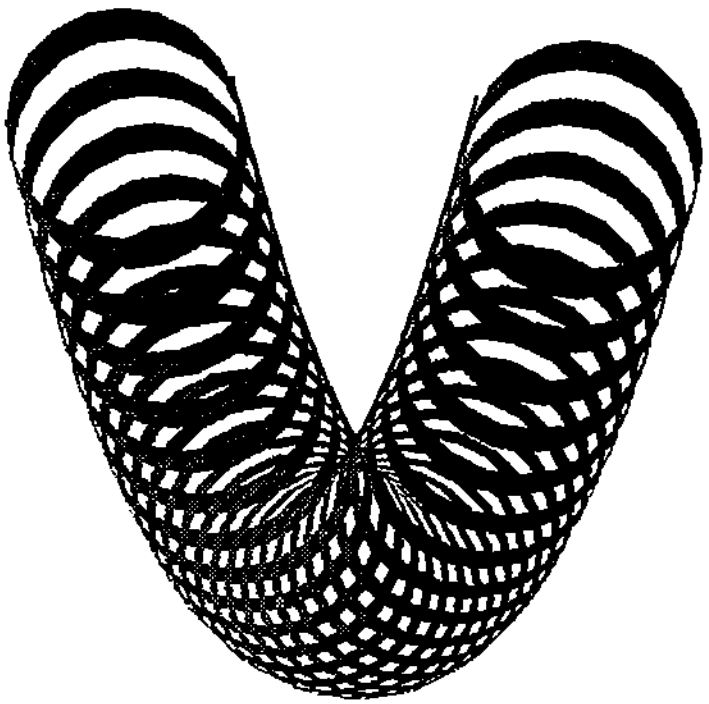


Figure 9 Offset curve of Figure 3 traced in 4D, viewing orthographically

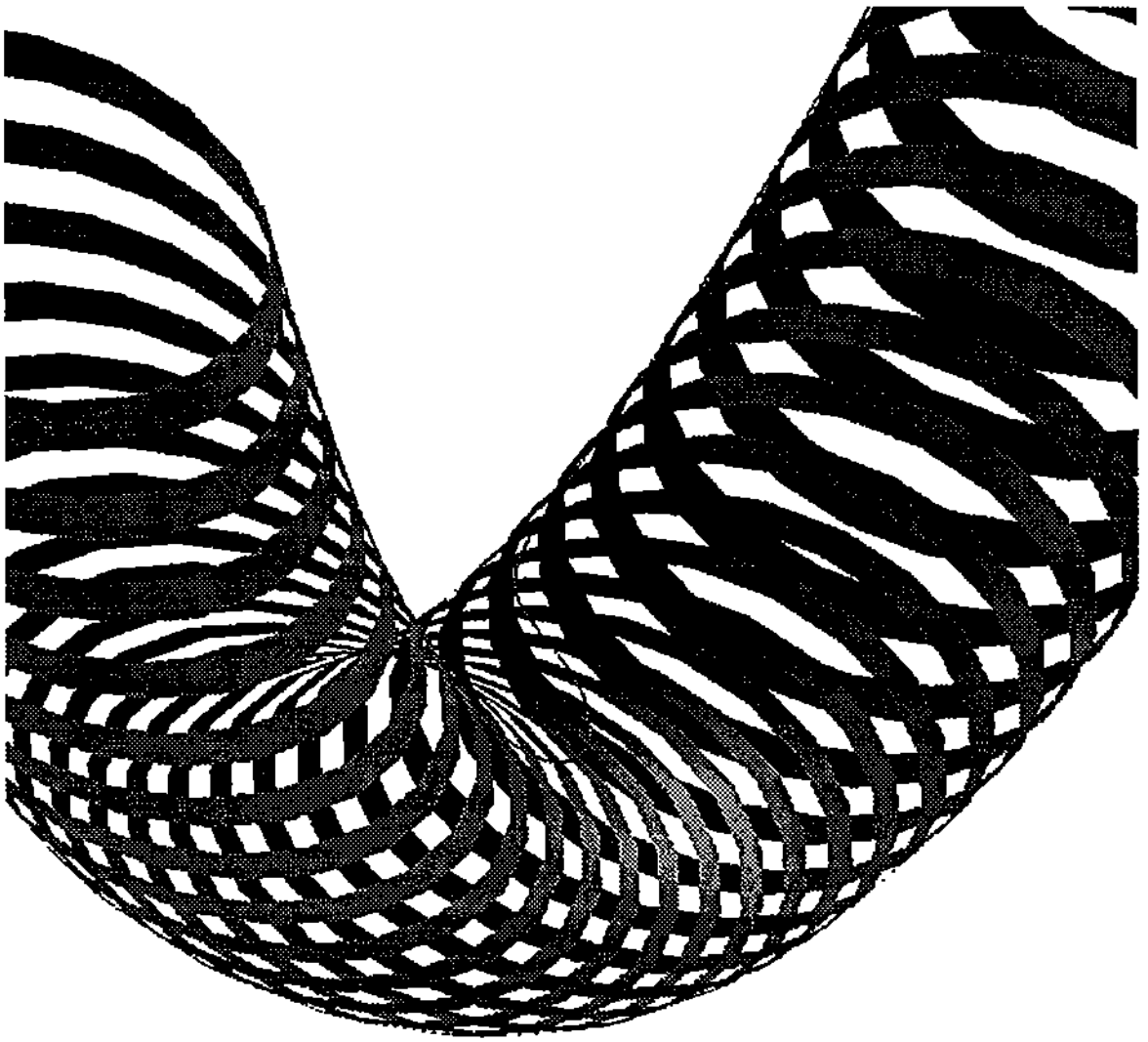
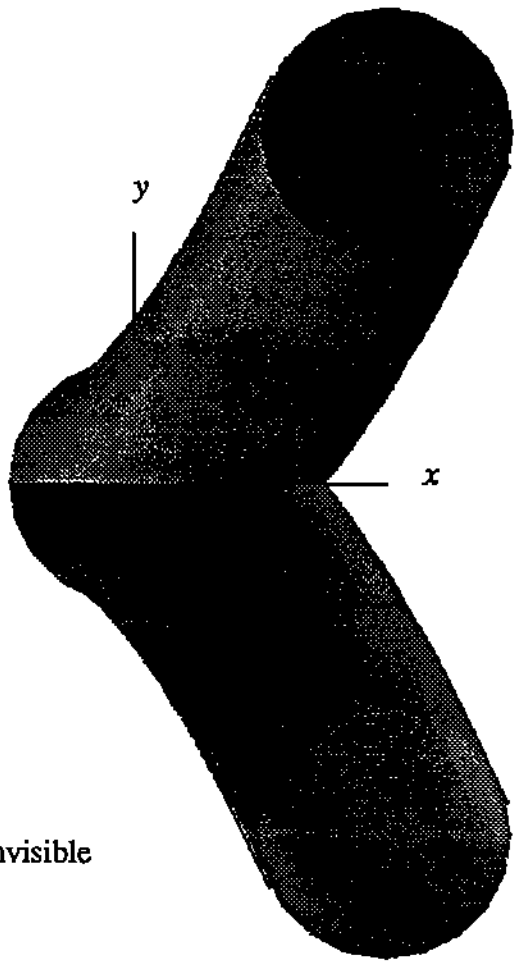
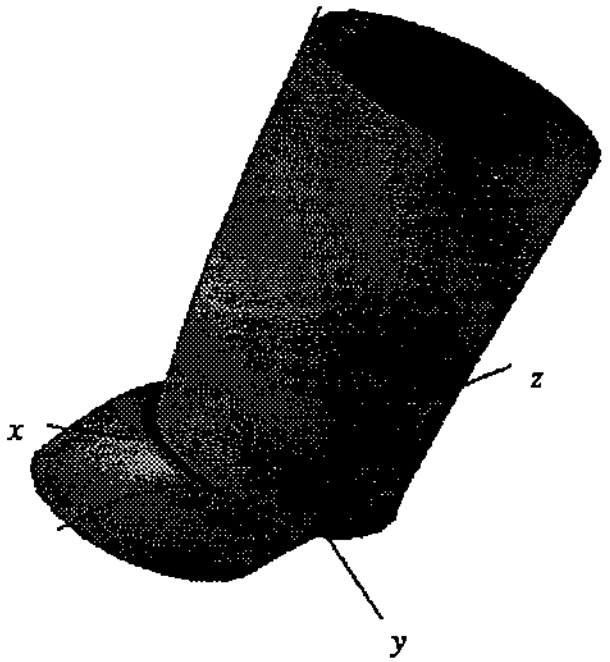


Figure 10 Offset curve of Figure 3 traced in 4D, viewing perspectively



z- and w-axes are invisible

Figure 11 Offset curve of Figure 4 traced in 4D, viewing orthographically



w-axis is hidden by the 2-surface

Figure 12 Offset curve of Figure 4 traced in 4D, viewing perspectively

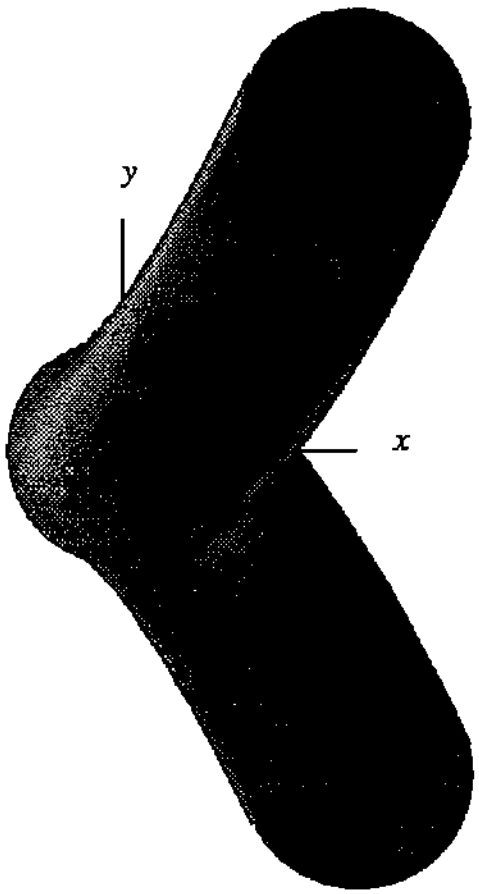


Figure 13 Offset curve of Figure 4 traced in 3D, viewing orthographically

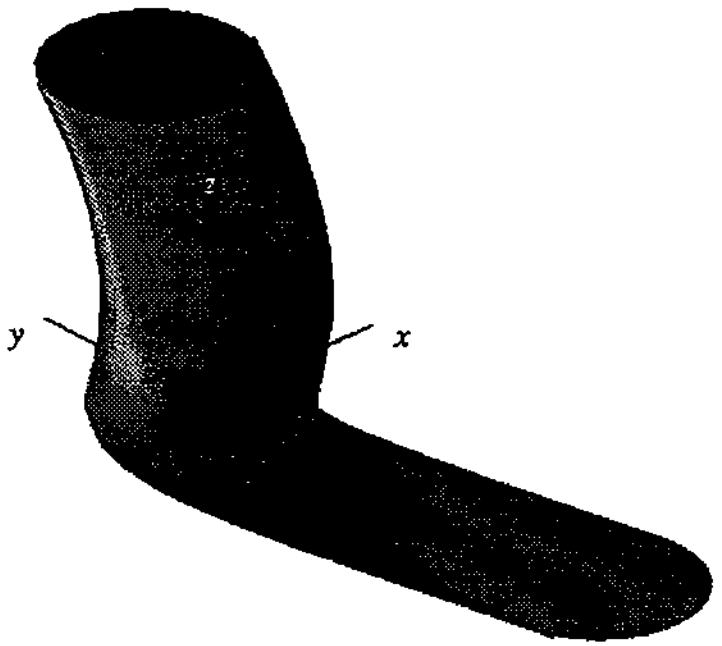


Figure 14 Offset curve of Figure 4 traced in 3D, viewing perspectively

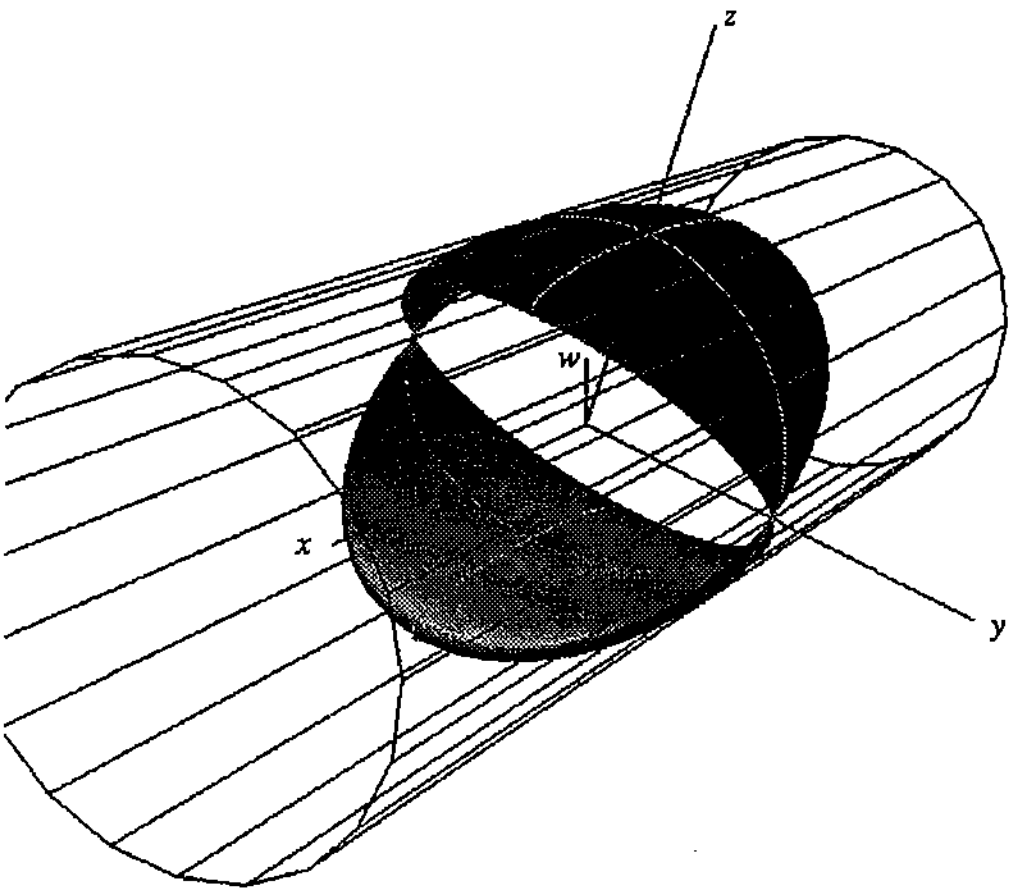


Figure 15 Intersection of a cylinder and a moving sphere with the same radius

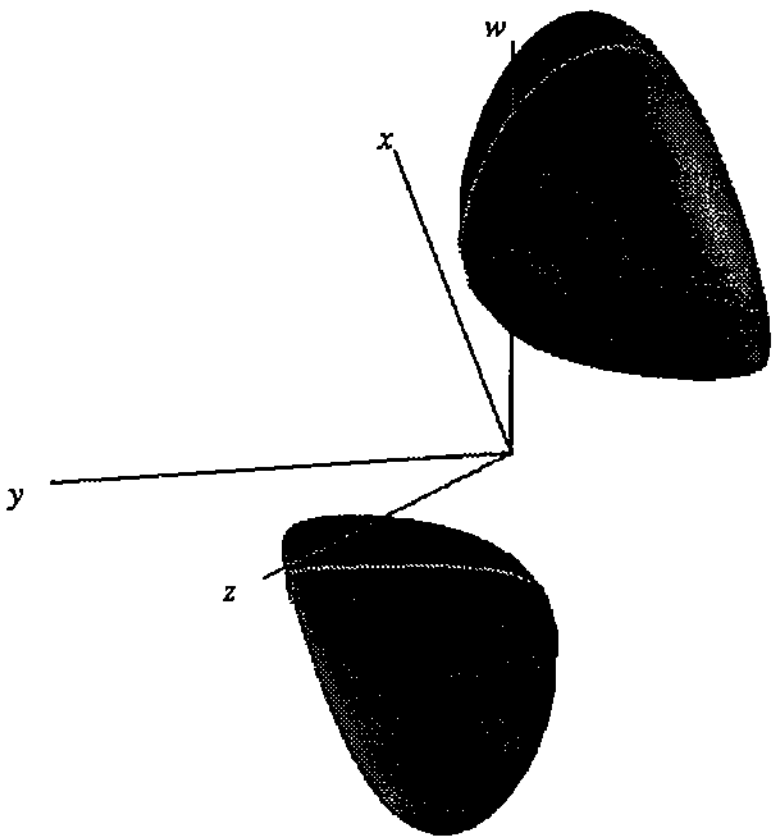


Figure 16 Intersection of a moving cylinder and a moving sphere with smaller radius

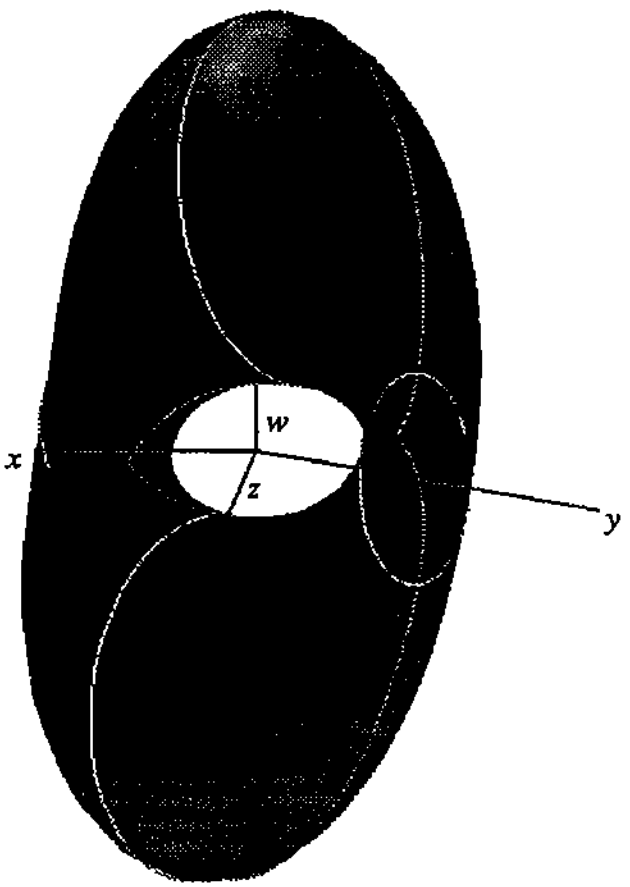


Figure 17 Intersection of a cylinder and a moving sphere with larger radius