Some techniques for visualizing surfaces in four-dimensional space*

Christoph M Hoffmann and Jianhua Zhou

The issues of visualizing two-dimensional surfaces in four-dimensional space are discussed, including methods to specify the orientation of objects and of projection centres, to determine silhouette points of a 2-surface with respect to projections and to calculate the normal of a projected 2-surface from its normal plane in 4-space. We have implemented an interactive 4D display system on a z-buffer graphics workstation. Preliminary experiments show that such a 4D display system can give valuable insights into high-dimensional geometry. Some examples are presented illustrating nonuniform material property display, offset curve geometry and collision detection.

solid modelling, visualization, 4-space

High-dimensional space is playing an increasing role in computer-aided design and solid modelling. Applications include describing the motion of 3D objects, modelling solids with nonuniform material properties, deriving spline curves uniformly, and formulating constraints for offset surfaces and Voronoi surfaces[5,7,10,11]. Apart from the modelling aspects, it is important to develop visualization aids for high-dimensional space. We have begun to investigate this topic with a view towards CAD and solid modelling, concentrating on four-dimensional space. Visualization of high-dimensional space has also successfully provided insights and methods for investigating geometric phenomena and dynamic systems[4,9].

We consider 2D surfaces in 4-space. Objects in 4-space could have dimension 0, 1, 2, 3, or 4, so dealing only with 2-surfaces is a restriction. However, it is not a severe one: for example, one could draw a 1-surface, i.e., a curve, in 4-space in the natural way. But displaying instead a tubular 2-surface 'wrapped around' the 1-surface provides better shape cues. Similarly, one could render volumetrically semi-transparent 3- and 4-surfaces. But just as the shape of a 2-surface in 3-space can be grasped from a rendered net of curves in the 2-surface, so 3- and 4-surfaces in 4-space can be understood from nets of 2-surfaces in them. Rendering 2-surfaces in 4-space is thus a quite powerful tool.

Graphics workstations have built-in hardware to render polygons in 3-space at interactive speeds. This custom hardware cannot be reprogrammed to deal with 4-space directly, and so 4D visualization would be much slower. Nevertheless, much efficiency is retained by separating the projection into two stages[10]. First, project the 2-surface into 3-space, then project its image onto the screen. As long as the first projection remains from a fixed view point, the second projection can be left to the hardware and is therefore as efficient as 3D graphics. The two projections are defined by two movable 'eyes', called eye_4 and eye_3. Here eye_4 controls the first projection, into 3-space. Object and view orientation are controlled by generalized Euler angles, as sketched in the second section, and are specified by users during animation. Note that the authors restrict attention to pure rotation. The object can be translated in 4-space during the animation. This section also discusses how properties of the 4D geometry can be recovered from certain projection strategies.

Given a 2-surface in 4-space, we first approximate it by a mesh of polygons. Since a 2-surface in 4-space has two independent normal directions, we do the rendering after the first projection has mapped the mesh into 3-space, where also vertex normals are added. Instead of computing the normals from the projected data in 3-space, in the third section a method for calculating them from the original 2-surface is given, thereby gaining some accuracy and speed. The fourth section discusses our implementation and its efficiency. Several examples illustrating the use of the system are given in the fifth section.

Projecting down two dimensions obliterates information; eye_4 might see some interesting features not seen by eye_3. In particular, silhouette points, at which the line of sight grazes the object, provide crucial shape information. Thus, we have ways of computing analytically silhouettes with respect to the projections and then render them as curves in the 2-surface so that they can be viewed from different directions. This computation is also explained in the third section.

Department of Computer Science, Purdue University, West Lafayette, IN 47907, USA
Paper received: 4 May 1990. Revised: 29 October 1990

*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.

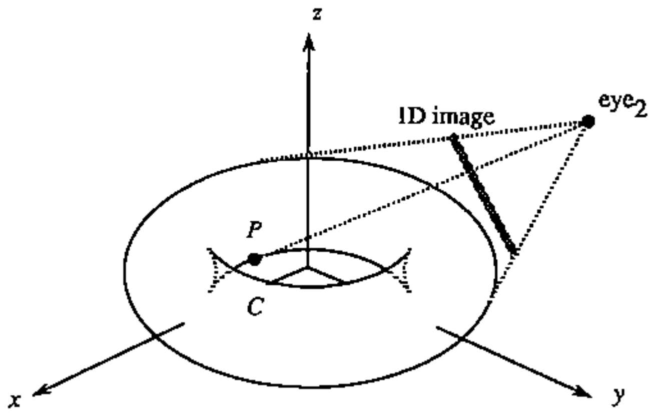


Figure 1. Torus projected into a plane and then into a line

A silhouette with respect to two projections is more complex than a silhouette with respect to one. As an analogy from 3-space, consider a torus projected into a 2-space that contains the page on which Figure 1 is printed. The centre of this projection, eye$_3$, is somewhere in the first octant of the (x, y, z)-coordinate system and not shown. 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 centre 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 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 the projection centre falls into the tangent plane. Clearly, when eye$_3$ is moving, the silhouette curve changes its shape and so does its projection. Now let's 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*. This happens when the ray from that point to eye$_3$ coincides with the tangent line. The four cusps in Figure 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. 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 its dimension under the projection.

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

ORIENTATION SPECIFICATION

In this section we focus on the orientation problem for displaying 2-surfaces in 4-space. We want to orient the two projection centres so that the 3D and 2D images of the 2-surface can be controlled independently. We also discuss some specific orientation choices and what information they reveal about the 4D geometry.

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

Euler angles, common in 3D kinematics[14], specify the orientation of objects by three successive rotations. Generalized to 4-space, Euler angles orient objects by six successive rotations. Denoting the six angles as θ_1, ..., θ_6, the coordinates are related via

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

where $R^{i}_{xy}(\cdot)$ are basic rotation matrices in the (x, y)-plane, etc., and $\mathbf{p}$ and $\mathbf{q}$ are the coordinates of a point in the world and in the body-fixed coordinate system, respectively. The rotations are grouped into three phases, as indicated by the common superscripts. In the first phase with superscript 4, the body-fixed w-axis is oriented into its final position in 4-space. In the second phase with superscript 3, the body-fixed z-axis is oriented into its final position within the 3D subspace orthogonal to the (oriented) body-fixed w-axis. In the third phase with superscript 2, the body-fixed x- and y-axes are oriented within the 2D subspace orthogonal to the other two axes. Note that the angles θ_1, θ_2, θ_3 constitute polar coordinates in 4-space.

The authors 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 $r = (r_1, r_2)$ of reciprocal distances of eye$_4$ and eye$_3$.

We explain how to interpret images for certain positions of eye$_4$ and eye$_3$. 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 2 shows a companion

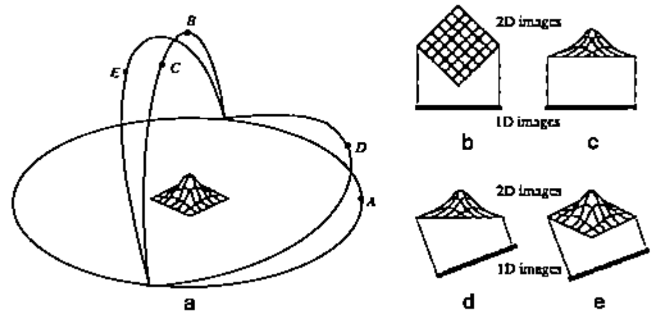


Figure 2. Square plate projected into a line. (a) Viewing positions. (b) Eye$_3$ at B, eye$_2$ at A. (c) Eye$_3$ at A, eye$_2$ at B. (d) Eye$_3$ at A, eye$_2$ at C. (e) Eye$_3$ at D, eye$_2$ at E.

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.

1. By ignoring the w-values, a normal 3D image is to be 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 (Colour Plate 1(a)). Note that it is still possible to see the w-value on the boundary of the 3D object using a colour scale (Plate 1(b)). Moreover, the same effect can be achieved by setting $\theta = (\alpha_1, \alpha_2, 0, 0, 0, 0)$ and $r = (0, a)$. Compare Figure 2(b).
2. Assume that $\theta = (\alpha_1, \alpha_2, \pm\pi/2, 0, 0, 0)$ 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 colour is the maximum, or the minimum,[†] of all w-values on a line through the bracket from eye$_4$. The situation is analogous to looking at a mountain from atop (Colour Plate 1(c)). The base of the mountain is the projected bracket shape, and the height is the w-value. Compare Figure 2(c).
3. Once we have generated the picture of (2), the isosurface (w = constant) can be obtained by z-clipping (Plate1(d)). 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 Colour Plate 1(b).
4. The 3D image implied by the picture of (2) can also be seen from other directions, with $\theta = (\alpha_1, \alpha_2, \pm\pi/2, \beta_1, \beta_2, 0)$ 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. Colour Plates 1(e) and 1(f) show the cases $\beta_2 = \pi/4$ and $\beta_2 = \pi/2$. Compare Figure 2(d).
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 $\pm\pi/2$, 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 Colour Plate 1(g) only three surfaces in the grid, namely $x = \text{constant}_1$, $y = \text{constant}_2$, and $z = \text{constant}_3$, are displayed. Compare Figure 2(e).

[†] Approximately if $a \neq 0$.

SILHOUETTES AND SURFACE NORMALS UNDER PROJECTION

In this section the authors discuss two related issues: silhouette points and the normals of 2-surfaces under projection.

Silhouette points

Recall from the first section that silhouette points are characteristic image features that provide important shape information. In some applications one would like to examine the silhouette curve belonging to a particular projection from a different view. Silhouettes should also be taken into account when calculating the normal of the projected 2-surfaces. Thus we give methods for determining the silhouettes explicitly, without inferring them from a projected image.

We define silhouette points of a 2-surface with respect to a projection as those nonsingular points on it whose tangent plane is projected into a line or a point. In the following we always assume that:

1. ϕ_1: $\mathscr{R}^4 \rightarrow \mathscr{R}^3$ is the first projection with the centre on the w-axis. ϕ_2: $\mathscr{R}^4 \rightarrow \mathscr{R}^2$ is the combined projection with the two centres on the w- and z-axes.
2. $\mathbf{p}$ is a nonsingular point on a 2-surface S, and is not mapped to infinity under the projection. $\mathbf{t}_1$, $\mathbf{t}_2$ ($\mathbf{n}_1$, $\mathbf{n}_2$) are two linearly independent tangent (normal) vectors of S at $\mathbf{p}$.
3. $\mathbf{r}_1$ is a vector in the direction from $\mathbf{p}$ to the first projection centre, $\mathbf{r}_2$ is a vector in the direction from $\mathbf{p}$ to the second projection centre.

A differentiable mapping $\phi: \mathscr{R}^n \rightarrow \mathscr{R}^m$ will induce two linear transformations (see e.g.[2]):

$$\phi_{\bullet}\ (\text{tangent vector to } \gamma) = \text{tangent vector to } \phi \circ \gamma$$
$$\phi^{\bullet}\ (\text{normal vector to f}) = \text{normal vector to f} \circ \phi$$

where $\gamma: \mathscr{R} \rightarrow \mathscr{R}^n$ is a curve in the domain of ϕ, and $f: \mathscr{R}^m \rightarrow \mathscr{R}$ is a function on the range of ϕ. The matrix forms of the two linear transformations $\phi_{\bullet}$ and $\phi^{\bullet}$ are the Jacobian matrix $J(\phi)$ and its transpose $J(\phi)^T$, respectively.

Lemma 1

(a) The null space of $\phi_{1\bullet}$, namely $\phi_{1\bullet}^{-1}(\mathbf{0})$, is span($\mathbf{r}_1$).
(b) The null space of $\phi_{2\bullet}$, namely $\phi_{2\bullet}^{-1}(\mathbf{0})$, is span ($\mathbf{r}_1$, $\mathbf{r}_2$).

This can be verified by means of the Jacobian matrices.

Proposition 1

(a) $\mathbf{p}$ is a silhouette point with respect to ϕ_1 if and only if $\mathbf{t}_1$, $\mathbf{t}_2$, $\mathbf{r}_1$ are linearly dependent.

(b) $\mathbf{p}$ is a silhouette point with respect to ϕ_2 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 with respect to ϕ_2 if and only if $\mathbf{n}_1$, $\mathbf{n}_2$, $\mathbf{m}_1$, $\mathbf{m}_2$ are linearly dependent.

Proof: (a) and (b) are similar and so only the proof of (b) is given as follows. First we assume that $\mathbf{t}_1$, $\mathbf{t}_2$, $\mathbf{r}_1$, $\mathbf{r}_2$ are linearly dependent, and so $\alpha_1\mathbf{t}_1 + \alpha_2\mathbf{t}_2 + \alpha_3\mathbf{r}_1 + \alpha_4\mathbf{r}_2 = \mathbf{0}$ where at least one of α_1 and α_2 is not zero. Applying the linear transformation ϕ_{2*} we get $\alpha_1\phi_{2*}(\mathbf{t}_1) + \alpha_2\phi_{2*}(\mathbf{t}_2) = \mathbf{0}$. That means the tangent plane is mapped onto a line which may degenerate into a point, and so $\mathbf{p}$ is a silhouette point. Conversely, assume that $\mathbf{p}$ is a silhouette point. Then the two vectors $\phi_{2*}(\mathbf{t}_1)$ and $\phi_{2*}(\mathbf{t}_2)$ are either zero or parallel. In either case we have $\alpha_1\phi_{2*}(\mathbf{t}_1) + \alpha_2\phi_{2*}(\mathbf{t}_2) = \mathbf{0}$ with $\alpha_1\alpha_2 \neq 0$. By Lemma 1(b) we get $\alpha_1\mathbf{t}_1 + \alpha_2\mathbf{t}_2 + \mathbf{r} = \mathbf{0}$ where $\mathbf{r}$ is any vector in span $(\mathbf{r}_1, \mathbf{r}_2)$, i.e. $\mathbf{t}_1$, $\mathbf{t}_2$, $\mathbf{r}_1$, $\mathbf{r}_2$ are linearly dependent.

(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 $\mathscr{R}^4$. 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. □

Thus the silhouette point of S with respect to ϕ_1 is determined by $\mathbf{r}_1 \cdot \mathbf{n}_1 = 0$, $\mathbf{r}_1 \cdot \mathbf{n}_2 = 0$. Together with the equations for S, the solution is usually a 0-dimensional set. The silhouette point of S with respect to ϕ_2 is determined by $\det(\mathbf{t}_1, \mathbf{t}_2, \mathbf{r}_1, \mathbf{r}_2) = 0$, or equivalently by $\det(\mathbf{n}_1, \mathbf{n}_2, \mathbf{m}_1, \mathbf{m}_2) = 0$. Here, the solution is usually a 1-dimensional set called the silhouette curve.

Surface normals

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 illuminating the 3D image of the 2-surface after the first projection step, using 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 derivation of the equation of the projected 2-surface could be expensive[8].

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:

$$\bar{\mathbf{n}} = \phi_{1*}(\mathbf{t}_1) \times \phi_{1*}(\mathbf{t}_2)$$

If the 2-surface is in implicit form and so the normal vectors are directly available, we can first use the cross product in $\mathscr{R}^4$ 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 $\mathscr{R}^4$, 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 base vectors $\mathbf{n}_1$ and $\mathbf{n}_2$ we can find two tangent base 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

$$\mathbf{t}_1 = \otimes(\mathbf{n}_1, \mathbf{n}_2, \mathbf{a}), \quad \mathbf{t}_2 = \otimes(\mathbf{n}_1, \mathbf{n}_2, \mathbf{b})$$

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. Define π: $\mathscr{R}^4 \rightarrow \mathscr{R}^3$ as the natural projection given by $\pi(x, y, z, w) = (x, y, z)$.

Proposition 2

Suppose that $\mathbf{p}$ is a nonsingular, nonsilhouette point of S with respect to ϕ_1. Let $\mathbf{n} = \alpha\mathbf{n}_1 + \beta\mathbf{n}_2$ satisfy $\mathbf{n} \cdot \mathbf{r}_1 = 0$. Then $\pi(\mathbf{n})$ is the normal vector of the projected 2-surface at point $\phi_1(\mathbf{p})$.

Proof: Assume that $\bar{\mathbf{n}}$ is the normal vector of the projected 2-surface at the point $\phi_1(\mathbf{p})$. From the Jacobian matrix $J(\phi_1)$ we know that $\pi(\phi_1^*(\bar{\mathbf{n}}))$ is parallel to $\bar{\mathbf{n}}$. It suffices to show that $\mathbf{n}$ as defined above is parallel to $\phi_1^*(\bar{\mathbf{n}})$ in $\mathscr{R}^4$. The vector $\bar{\mathbf{n}}$ satisfies

$$\bar{\mathbf{n}} \cdot \phi_{1*}(\mathbf{t}_1) = 0, \quad \bar{\mathbf{n}} \cdot \phi_{1*}(\mathbf{t}_2) = 0, \quad \bar{\mathbf{n}} \cdot \phi_{1*}(\mathbf{r}_1) = 0$$

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

$$\phi_1^*(\bar{\mathbf{n}}) \cdot \mathbf{t}_1 = 0, \quad \phi_1^*(\bar{\mathbf{n}}) \cdot \mathbf{t}_2 = 0, \quad \phi_1^*(\bar{\mathbf{n}}) \cdot \mathbf{r}_1 = 0$$

Since $\mathbf{p}$ is a nonsilhouette point, $\mathbf{t}_1$, $\mathbf{t}_2$, $\mathbf{r}_1$ are linearly independent by Proposition 1(a). Hence $\mathbf{n}$ is parallel to $\phi_1^*(\bar{\mathbf{n}})$. □

IMPLEMENTATION AND EFFICIENCY CONSIDERATIONS

We have implemented an interactive system version on Silicon Graphics Personal IRIS workstations. Assuming that a 2-surface has been preprocessed into 1200 4D polygons, we achieve the following performance. If eye$_3$, the light sources, and the z-clipping plane are moved, the screen can be updated at a rate of approximately 10 frames per second. If eye$_4$ is moved, however, the first projection stage must be recomputed and the screen update rate is only about 4 frames per second.

The implementation is as follows. The input 2-surfaces are first polygonalized by algebraic or space subdivision methods (for example see[1]). The polygons in 4-space are projected into 3-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 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.

Each vertex of the polygons in 4-space is associated with two normal vectors or two tangent vectors, depending on the original specification of the 2-surface and the chosen method of polygonalization. The first projection of polygons and the calculation of normal vectors are done by software. The result is saved and recalculated only when necessary. For instance, when the user moves eye$_3$, light sources or the z-clipping planes, only the second projection and the colouring are affected and are handled efficiently by hardware. In this way, a better average response is achieved because usually after one motion of eye$_4$ several adjustments of eye$_3$ are needed to make a full inspection of the 3D image.

Hidden surface removal and z-clipping are almost standard techniques for 3D graphics. These techniques are not hard to extend to 4D graphics. However, since there is no hardware support, implementation by software seems too slow to be acceptable for interactive display. If all objects to be displayed are 2-surfaces it is not necessary to implement the 4D counterpart of hidden surface removal. Likewise, the 4D counterpart of z-clipping is also avoided for efficiency consideration. It can be simulated in part by exchanging eye$_4$ and eye$_3$ and using z-clipping. See the second section, above.

EXAMPLES

Examples illustrating some of the applications of our 4D visualization system are given. The authors have already discussed viewing a bracket with nonuniform material properties in Section 2. We add to this application an interpretation of the envelope theorem from differential geometry and a brief sketch of collision detection and analysis.

Offset curves

Given a curve $f(x, y)=0$ in $\mathscr{R}^2$, its offset curve by distance $\mathbf{r}>0$ can be formulated by the envelope method[6,7] as a set of equations:

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

where

$$\nabla_{uv} g=\left(\frac{\partial g}{\partial u}, \frac{\partial g}{\partial v}\right)^T$$
$$\mathbf{t}=\left(\frac{\partial f}{\partial v}, -\frac{\partial f}{\partial u}\right)^T$$

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 $\partial h/\partial t=0$. 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[6]:

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

where

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

An implicit equation for the offset curve can be determined by the resultant method[6], or using Gröbner bases[7]. The offset curve can also be traced numerically in $\mathscr{R}^4$ or $\mathscr{R}^3$ by the method described in[3].

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 the (x, y)-plane (see Figure 3(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 3(b)).

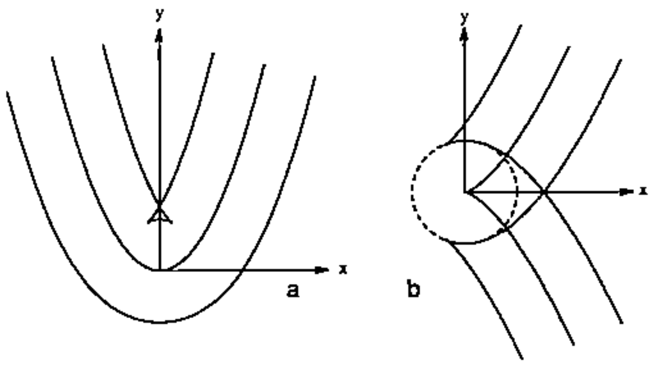


Figure 3. Curves (a) $y-x^2=0$ and (b) $y^2-x^3=0$ and their offset curves by 1

It has been attempted to 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:

$$\mathbf{n}_1 = \nabla g(\mathbf{p}) = (2(x-u), 2(y-v), -2(x-u), -2(y-v))^T$$

$$\mathbf{n}_2 = \nabla f(\mathbf{p}) = \left(0, 0, \frac{\partial f}{\partial u}, \frac{\partial f}{\partial v}\right)^T$$

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 Proposition 1 (c) it is a silhouette point with respect to an orthographic projection with two centres along the u- and v-axes. The silhouette points form a curve on the tubular surface S in $\mathscr{R}^4$. In Colour Plates 2(a) and 2(b) we show the 2-surface S and the silhouette curve corresponding to the offset curve in Figure 3(a). In $\mathscr{R}^4$ the curve is smooth without cusps or self-intersections, as we can see in Colour Plate 2(b) from a different viewing direction.

Alternatively, 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 above as the second phenomenon. In Colour Plates 2(c) and 2(d) the 2-surface S and the silhouette curve corresponding to the offset curve in Figure 3(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 3(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 $\partial h/\partial t = 0$ are two surfaces and their intersection is a curve. Note that $\partial h/\partial t = 0$ is equivalent to $\nabla h \cdot \mathbf{k} = 0$. 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 $\partial h/\partial x$ and $\partial h/\partial y$ cannot be zero simultaneously and so ∇h is always a nonzero vector. The dashed circle in Figure 3(b) is actually another branch of the silhouette curve as shown in Colour Plates 2(e) and 2(f).

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 tubular 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.

Collision detection and analysis

The advantage of using 4D geometry to deal with the collision detection problem has been explained in[5,10]. Briefly, we can extrude moving 3D objects into (x, y, z, t)-space. Two moving objects collide if and only if the intersection of their extrusions is nonempty.

3D objects are bounded by surfaces in 3-space. The extrusion of such a surface is a 3-surface in 4-space. The intersection of two 3-surfaces is a 2-surface and can be examined by our system. Assume that the 2-surface is nonempty. We will 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, the natural projections of the two normals at $\mathbf{p}$ into (x, y, z)-subspace are parallel. This condition often determines a 0-dimensional solution set on the 2-surface. It is usually too difficult to solve the nonlinear equations describing these points. By relaxing the condition, one may use numerical methods such as curve tracing. By Proposition 1(c), $\mathbf{p}$ must be on the silhouette curve of the 2-surface with respect to any orthographic projection with two centres both inside (x, y, z)-subspace.

Colour Plate 3 shows a moving cylinder's intersects with a moving sphere. The axis of the cylinder and the centre of the sphere pass through the origin at $t = 0$. The curves on the 2-surface are the silhouette curve branches with respect to the orthographic projection with two centres along the x-, y- or z-axes but now seen from different viewpoints. Colour Plate 3(a) shows the case where the cylinder and the sphere have the same radius. Since eye$_4$'s position is just a little off the t-axis, the 2-surface resembles the sweep of the intersecting curve in (x, y, z)-subspace. Colour Plate 3(b) shows the case where the cylinder has the larger radius. Note that the 2-surface has two separate components. Colour Plate 3(c) shows the case where the sphere has the larger radius. Note that although the 2-surface is connected, the silhouette curve branches can be separated.

ACKNOWLEDGEMENTS

The authors thank the referees for their valuable comments and suggestions.

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 (1987) pp 452–469

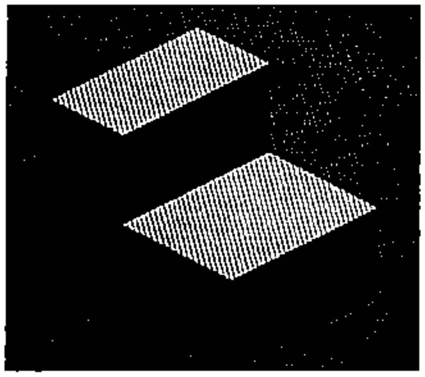
2 **Auslander, L and MacKenzie, R E** *Introduction to Differentiable Manifolds* McGraw-Hill Book Company, Inc. (1963)

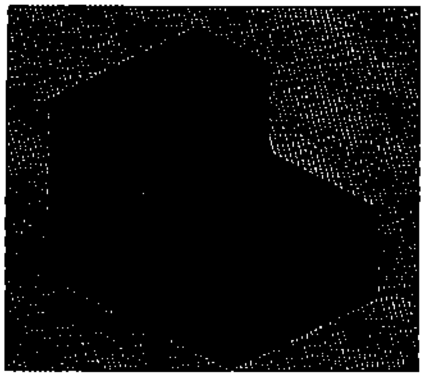
3 **Bajaj, C, Hoffmann, C, Hopcroft, J and Lynch, R** 'Tracing surface intersections', *Comput. Aided Geom. Des.* Vol 5 (1988) pp 285–307

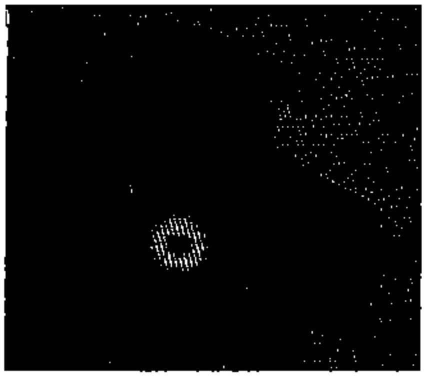
4 **Banchoff, T F** 'Visualizing two-dimensional phenomenon in four-dimensional space: a computer graphics

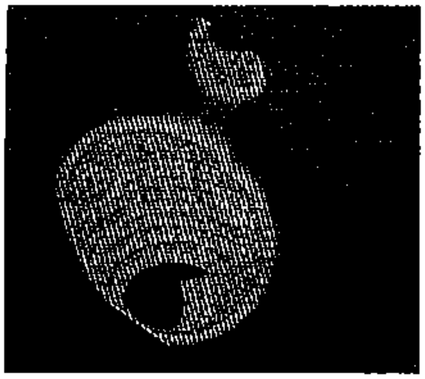
Some techniques for visualizing surfaces in four-dimensional space

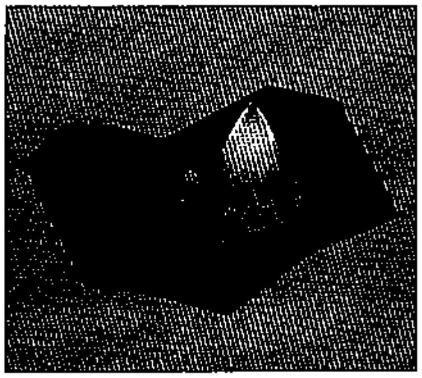
C M Hoffmann and J Zhou

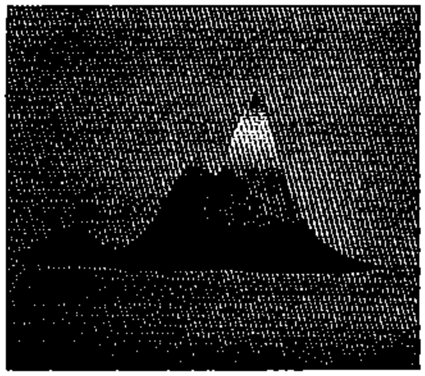

Colour Plate 1(a). Bracket viewed from $\theta = (0, 0, 0, 45, 60, 0)$ degrees and $r = (0, 0)$

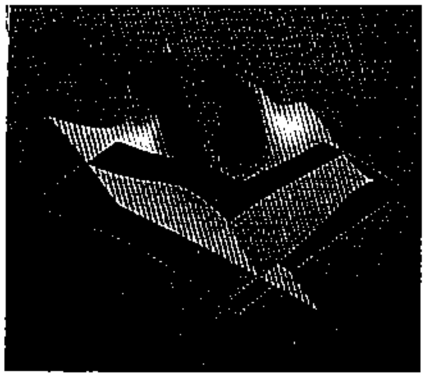

Colour Plate 1(b). Bracket viewed from same directions as (a) but shaded by colour scale representing w-values

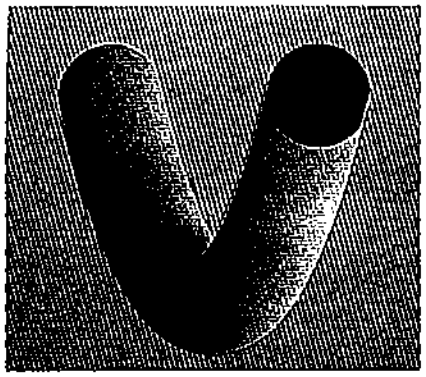

Colour Plate 1(c). Bracket viewed from $\theta = (45, 60, 90, 0, 0, 0)$ and $r = (0, 0)$; positions of eye_4 and eye_3 are exchanged from (b)

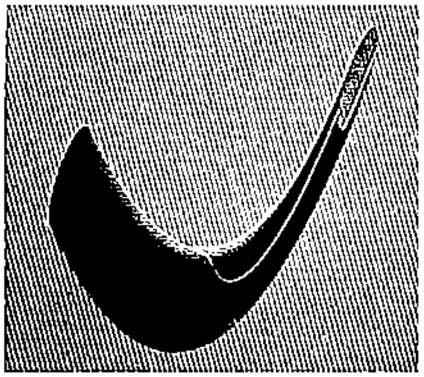

Colour Plate 1(d). Bracket in (c) cut by z-clipping plane to display isosurface $w = \text{constant}$

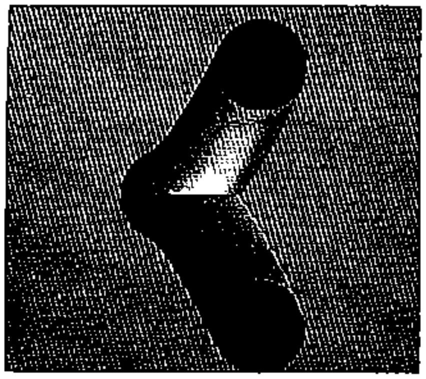

Colour Plate 1(e). 3D image implied by (c) seen from another direction, with $\theta = (45, 60, 90, -75, 45, 0)$ and $r = (0, 0)$

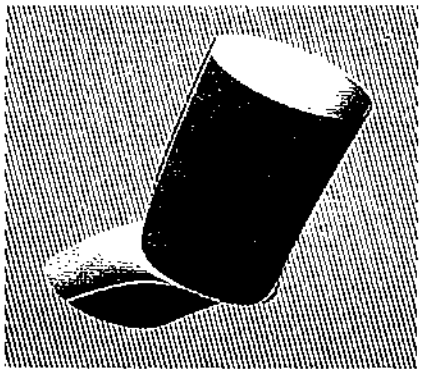

Colour Plate 1(f). 3D image implied by (c) is seen from $\theta = (45, 60, 90, -75, 90, 0)$ and $r = (0, 0)$

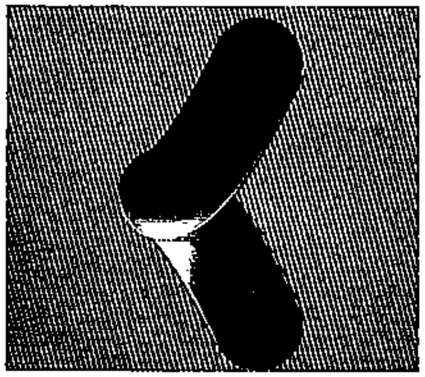

Colour Plate 1(g). Bracket viewed from $\theta = (45, 45, 45, 105, 90, 0)$ and $r = (0, 0)$. Only three surfaces in grid: $x = \text{constant}_1$, $y = \text{constant}_2$ and $z = \text{constant}_3$ are displayed. Red, orange, green and blue lines are projected x-, y-, z- and w-axes respectively

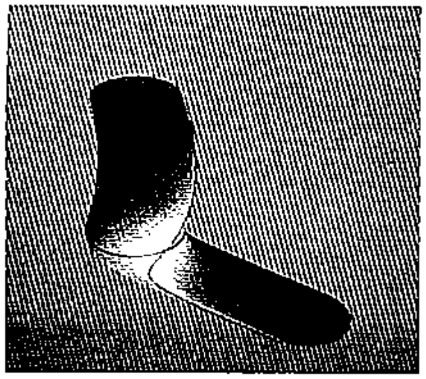

Colour Plate 2(a). Offset curve of $y - x^2 = 0$ traced in 4-space, and projected orthographically into (x, y)-plane by $\theta = (0, 0, 0, 0, 0, 0,)$ and $r = (0, 0)$

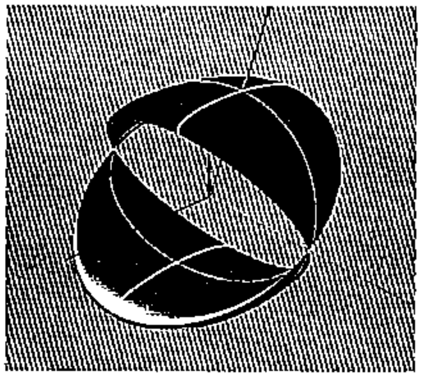

Colour Plate 2(b). Offset curve in (a) viewed from $\theta = (45, 105, 45, 75, 165, 0)$ and $r = (0, 0)$

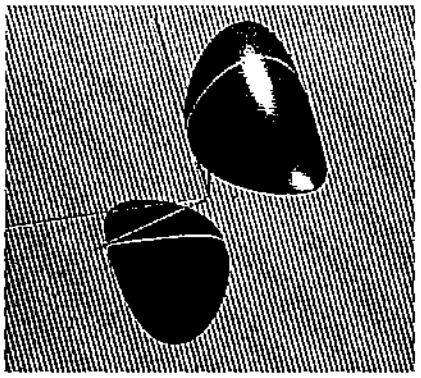

Colour Plate 2(c). Offset curve of $y^2 - x^3 = 0$ traced in 4-space, and projected orthographically into (x, y)-plane

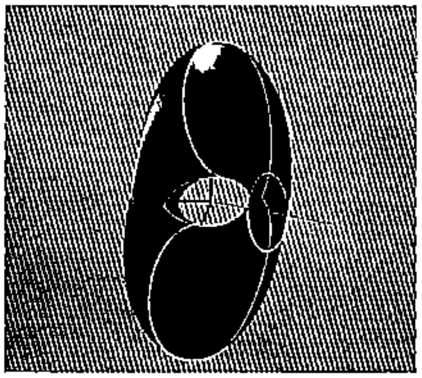

Colour Plate 2(d). Offset curve in (c) viewed from $\theta = (45, 40, 60, 105, 75, 0)$ and $r = (0, 0.04)$.


Colour Plate 2(e). Offset curve in (c) traced in 3-space, and projected orthographically into (x, y)-plane


Colour Plate 2(f). Offset curve in (e) viewed from $\theta = (0, 0, 0, -40, 60, 0)$ and $r = (0, 0.04)$


Colour Plate 3(a). Intersection of cylinder and moving sphere with the same radius, viewed from $\theta = (0, 18, 9, 120, 75, 0)$ and $r = (0, 0.1)$. Vertical blue line is the t-axis


Colour Plate 3(b). Cylinder has larger radius, viewed from $\theta = (0, 30, 105, -105, 30, 0)$ and $r = (0, 0.1)$


Colour Plate 3(c). Cylinder has smaller radius, viewed from $\theta = (0, -15, 123, 120, 90, 0)$ and $r = (0, 0.1)$

approach' *Statistical Image Processing and Graphics* **Wegman, E J and DePriest, D J (eds)** (1986) pp 187–202

5 **Cameron, S A** Modelling Solids in Motion, *Ph.D. Thesis*, University of Edinburgh (1984)

6 **Farouki, R T and Neff, C A** 'Algebraic properties of plane offset curves' *Comput. Aided Geom. Des.* Vol 7 (1990) pp 101–127

7 **Hoffmann, C M** *Geometric and Solid Modeling: An Introduction* Morgan Kaufmann Publishers, Inc (1989)

8 **Hoffmann, C M** 'A Dimensionality paradigm for surface interrogations' *Comput. Aided Geom. Des.* (to appear 1990)

9 **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 (1986) pp 282–308

10 **Rossignac, J R** 'Considerations on the interactive rendering of four-dimensional volumes' *CH Volume Visualization Workshop* (1989) pp 67–76

11 **Seidel, H P** 'Geometric constructions and knot insertion for β-splines of arbitrary degree' *Tech. Report*, Department of Computer Science, University of Waterloo (1990)

12 **Wittenburg, J** Dynamics of systems of rigid bodies, B G Teubner, Stuttgart (1977)

A relational graphical editing method for PCB design

Shi Kaijian and Sun Jian

PCB design is currently an essential part of the process of manufacturing electronic equipment, and many CAD systems have been developed to facilitate this design process[1,2,3,4]. For all these systems the PCB Editor is an important and inevitable component because automatic routers usually have a few simple control strategies for routing[2]. In many cases they cannot produce satisfactory PCB design and existing automatic routing methods cannot guarantee 100% connection in practice, although some of them are successful in theory.

PCB design, graphical edit

As the PCB edit process is so important in PCB design, which itself is a fundamental part of electronic equipment design, an improvement in this area would be valuable. For this reason, an investigation of PCB edit process has been undertaken which leads to a new PCB edit method – Relational Graphics PCB Edit method. In the paper, the principle of this method is described; the algorithmic description is provided; and some considerations about editing with the new method are discussed. The method is capable of performing PCB edit operations automatically wherever necessary to keep the PCB layout correct, and of avoiding edit errors by verifying connection completeness and detecting net conflict. The PCB edit process with this method is more convenient and less error prone than conventional ones, since the designer is relieved of the burden of many tedious operations for editing a set of relative lines since there is a single line edit. These editing operations, which constitute a good part of PCB edit effort, are automatically taken care of by the proposed method.

PRINCIPLE OF RELATIONAL GRAPHICS PCB EDIT METHOD

To help the edit process, PCB design systems usually provide various graphical edit facilities, by which users can insert, delete, drag, or extend lines to meet their requirements. However, all these facilities (which are called conventional methods in the following part of the paper) are not yet satisfactorily in use and the PCB edit process is often tiresome and subject to editing errors.

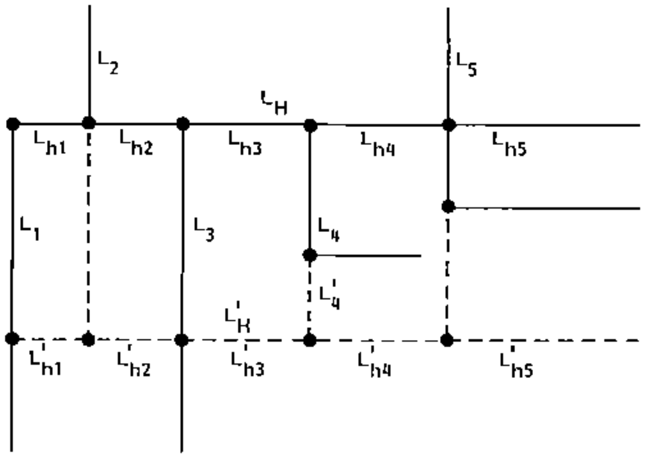


Figure 1. PCB edit case

The authors' research shows that the main cause of the inconvenience in PCB edit process with the conventional methods is that they treat PCB lines as individual lines or lines having connections at their ends only[4].

Consider the situation of Figure 1.

(The solid lines represent a part of PCB before L_H is edited. The broken lines show the situation after L_H is dragged down to the position of L'_H, and the relative lines (L_1–L_5) are edited accordingly.)

Suppose L_H needs to be dragged down to L'_H. To keep the connection correct, L_1 and L_3 should be cut downward. L_2 and L_5 should be extended downward, and L_4 should be erased while L'_4 should be generated. This is achieved in two ways using conventional methods. One way is to drag L_H down first and then, one by one, to edit all the relevant lines by extending or cutting them according to their situations. The other way is to treat L_H as a set of individual segments $\{L_{hi}\}$ and to drag them by their ends one after another until every part of L_H has been edited[4]. Neither way is convenient because first, a geometric change of a line demands additional edit operations from the user in order to maintain correct connection, and second these additional edit operations are tiresome and increase the probability of edit errors.

PCB lines are connected with each other, grouping into different connection sets with certain relationships such as equality of electrical potentials[5], and therefore, they would be better represented as relational networks. Based on this means of representation a sophisticated graphics PCB editor could be built which would allow the user to perform only an initial edit operation and leave all consequent operations for modifying relevant

Electronic Engineering Laboratories, The University of Kent at Canterbury, Kent CT2 7NZ, UK
Paper received: 10 April 1989. Revised: 14 March 1990