

Sweeping of Three-dimensional Objects*

R. R. Martin P. C. Stephenson
Department of Computing Mathematics,
University of Wales College of Cardiff

Submitted to CAD, August 11, 1989
Revised, September 27, 1989

1 Introduction

The basic problem which this paper addresses is the following: given an arbitrary three-dimensional object, which moves along an arbitrary path (possibly rotating as it does so), to compute the volume swept out by the solid object as it moves, or in other words, to find the new solid volume which represents all of the points in space which the object has occupied at some time during the motion.

The modelling of swept volumes is important for many areas of CAD and CAM. For example, it can provide a solution to many different types of interference problem [31], where it is necessary to tell whether a moving object will intersect or interfere with other objects as it moves. This is obviously very useful in mechanism design, for ensuring that all parts of a mechanism stay inside its casing as it operates, or that parts of the mechanism do not collide with each other. As an example, consider the problem of determining whether a car wheel collides with the wheel arch as it travels up and down on the suspension, and steers left and right. Another application of interference checking is in robot path planning [13,14,38], where a safe path is one that does not collide with any of the fixtures or other objects in the robot's environment. When machining a solid object, interference checking can be useful to make sure that the cutter does not gouge the table or clamps. Calculation of swept volumes may also be used for cutter path verification [43,44]: if the volume swept out by the moving cutter is subtracted from the original blank, the result should represent the same shape as the part to be machined. A final use of swept shape calculation is as a creative design tool, where the sweeping of an initial shape along a desired path is used to produce new and useful solid shapes [17,37]. For example, the frame of a chair made from tubular steel could be represented by the path swept out by a moving sphere.

*This work was supported by an SERC ACME Grant, Number GR/D 59434

It should be noted that other workers studying the interference problem have suggested that the swept volumes need not be computed explicitly. For example, if they are chosen carefully, several (many) successive three-dimensional models of the moving solid at discrete times can be used for collision detection, although there are problems of algorithm termination with this approach [13,38]. Alternatively, null object detection can be performed directly on four-dimensional (space and time) models, using the technique of S-Bounds to obtain greater efficiency, as advocated by Cameron [13,15]. Boyse [8] shows how interference checking can be performed using an implicit description of swept volumes, but his method is based on planar faced objects moving along straight or circular paths.

The method given here of explicitly computing the swept volume has two advantages over those mentioned above. Firstly, some of those methods mentioned above are specific to interference checking, and other uses of the modelling of moving objects may require an explicit description of the swept volume. Secondly, our approach is a general one, allowing an arbitrary solid object to be swept along an arbitrary path (possibly rotating as it goes).

Other previous workers have only considered restricted forms of sweeping. For example, Ghosh *et al.* [24] consider a two-dimensional version of the problem and show how it is of use in describing brush trajectories. The Build group [17] discuss sweeping a two-dimensional lamina along a straight or circular arc path to create prisms or solids of revolution for design purposes. Similar sweeping of two-dimensional outlines along three-dimensional paths is reported by the TIPS group [37], and Coquillart [18]. Wang and Wang [43,44] restrict their attention to the cylindrical and spherical geometries that describe the motion of a ball ended cutter during machining. De Pennington *et al.* [20] describe how to compute the swept volume for a moving object represented as a CSG union. However, their approach does not generalise to other CSG objects including the intersection and difference operators. While sweep and union commute, sweep does not commute with these other two operators.

Previous work by the current authors [31] discusses some of the basic ideas involved in computing swept envelopes, and their application to interference checking. This paper extends that work, and shows how it may be used in a solid modeller.

2 Envelopes

2.1 Envelope Theory

Initially, we will consider how to find the swept volume for a single moving surface. In later sections we will then show how this technique may be used for composite solid models bounded by many different surfaces.

The basic theory required is the theory of *envelopes* from classical differential

geometry [5,22,23,45]. An implicit equation of the form

$$f(x, y, z) = 0 \tag{1}$$

represents a surface in space. (Given instead a parametric surface, where $x, y,$ and z are known as functions of u and v , say, then at least in principle it is possible to *implicitize* [36] this description, by eliminating u and v between the three equations, justifying the use of the implicit surface form as a starting point.) On allowing this surface to move, a continuous family of surfaces is generated, each of which may be specified by a particular value of time, t . This family of surfaces

$$F(x, y, z, t) = 0 \tag{2}$$

may also be thought of as a hypersurface in four dimensions.

The envelope to this family is that surface which just encloses all of the members of the family. Thus, at any given instant of time, the corresponding member of the family of surfaces will just touch the envelope surface in some curve. This may also be expressed by saying that the envelope surface is that surface which is tangential to all members of the family of surfaces. Consider the two members of the family at times t and $t + dt$. The first has equation $F(x, y, z, t) = 0$, while the second is $F(x, y, z, t + dt) = 0$. Noting that the envelope must meet both of these surfaces, and thus satisfy both equations, it can be seen by expanding the latter in terms of dt , and letting dt tend to 0, that the envelope surface must simultaneously satisfy

$$F(x, y, z, t) = 0, \quad \frac{\partial F(x, y, z, t)}{\partial t} = 0. \tag{3}$$

The implicit form of the envelope surface can be obtained, again, in principle, by eliminating t from these two equations, giving

$$e(x, y, z) = 0. \tag{4}$$

One way of seeing that the envelope surface does indeed have the desired property of being tangent to each of the surfaces in the family is to show that it has the same surface normal as each member of the family where they meet. The normal vector to a given member of the family of surfaces is given by

$$\mathbf{N} = \left(\frac{\partial F}{\partial x}, \frac{\partial F}{\partial y}, \frac{\partial F}{\partial z} \right), \tag{5}$$

where t is thought of as having some definite value. The normal to the envelope may be written as

$$\mathbf{N}_e = \left(\frac{\partial F}{\partial x} + \frac{\partial F}{\partial t} \frac{\partial t}{\partial x}, \frac{\partial F}{\partial y} + \frac{\partial F}{\partial t} \frac{\partial t}{\partial y}, \frac{\partial F}{\partial z} + \frac{\partial F}{\partial t} \frac{\partial t}{\partial z} \right), \tag{6}$$

where t must now be thought of as being able to vary, but constrained to do so such that $F(x, y, z, t) = 0$. However, for the envelope, $\partial F/\partial t = 0$, and so the normal directions to the envelope and the individual members of the family are the same.

One point which will further be considered later is that this envelope does not start or finish at particular values of t , but assumes that members of the family exist for all values of t .

2.2 Degenerate Cases

Although the theory above is straightforward in most cases, a little more care is needed at times. Let us consider the special case of sweeping a plane, both because it will be needed by many applications, and because it illustrates degenerate envelopes. Let us take a general plane which starts out with equation

$$f(x, y, z) = ax + by + cz + d = 0. \quad (7)$$

If it is swept along a straight line with uniform velocity (V_x, V_y, V_z) , then at time t , it will have equation

$$F(x, y, z, t) = a(x - V_x t) + b(y - V_y t) + c(z - V_z t) + d = 0. \quad (8)$$

(How to construct the hypersurface from a description of the original surface and the motion will be discussed in general in the next section.) The envelope must also satisfy

$$\frac{\partial F}{\partial t} = -(aV_x + bV_y + cV_z) = 0. \quad (9)$$

Now, the latter expression is independent of all of the variables x, y, z and t , which means that there are two possibilities.

Firstly, the normal to the family of planes, $\mathbf{N} = (a, b, c)$ is not perpendicular to the velocity vector, in which case Equation 9 can not be satisfied, and no envelope surface exists: there is no surface which just touches all planes parallel to the initial plane.

Secondly, if the normal to the plane *is* perpendicular to the velocity vector, this means that the motion is a translation of the plane onto itself, *i.e.* the plane is moving within its own plane. Equation 9 is trivially satisfied, and so Equation 8 tells us that the original plane is also the envelope of the whole family, as t disappears from this equation. Indeed, it is not difficult to see that this will be the case whenever the plane is moving with any kind of motion within its own plane.

A second type of degenerate case may be illustrated by the case of a vertical plane rotating about the z -axis. This can be written as

$$F(x, y, z, t) = x \cos(t) + y \sin(t) = 0. \quad (10)$$

Differentiating, it is found that

$$\frac{\partial F}{\partial t} = -x \sin(t) + y \cos(t) = 0. \quad (11)$$

The solution to this pair of equations is $x = y = 0$, *i.e.* the z -axis. Here the envelope surface degenerates to a straight line, and has lower dimensionality than the general case.

These are only meant as illustrative examples, and many other degenerate possibilities exist, such as a sphere rotating so that it maps onto itself. It should furthermore be noted that the set of points satisfying Equations 3 for the envelope may include other loci as well the envelopes being sought, such as isolated points, and the loci swept out by any singular points and curves of the moving surface. Although in many cases the fact that these problem loci will correspond to repeated solutions of $\partial F/\partial t = 0$, or appear as a multiple factor (see Section 4.2) of the envelope surface $e(x, y, z) = 0$ may be used as a means of detecting them [21], care must be taken, as such a locus may also accidentally be the desired envelope locus. In general such loci must be compared to the original surface, and the description of the motion, before they are discarded.

Further useful discussions of degeneracy of envelopes are to be found in [9,29,39]. It is important that all types of degenerate cases are detected and handled appropriately by any software which uses envelopes.

3 Generating the Hypersurface

In the above section it was assumed that given a surface $f(x, y, z) = 0$ and some description of its motion, it was possible to find the corresponding hypersurface $F(x, y, z, t) = 0$. How to do this will now be considered.

Firstly, let us examine how the path may be specified. At each moment of time, there are six degrees of freedom to specify the relative position and orientation of the moving surface. Let us suppose that some point on or outside the original surface (before motion) was chosen as a reference point, and that a set of perpendicular axes was chosen through that point to describe the orientation of the surface. Firstly, during motion, the reference point will have moved to a new point. The vector linking the new reference point to the original one can be written as $T(t)$. Secondly, the object may also have rotated about the reference axes, where Euler angles may be used to specify the relation between the orientation of the old and new axes. This rotation may be described by a 3×3 orthogonal matrix, denoted by $R(t)$. Thus, overall, a point on the original surface $\mathbf{p}(0)$ corresponds to a point $\mathbf{p}(t)$ after the surface has moved, given by

$$\mathbf{p}(t) = R(t)\mathbf{p}(0) + T(t). \quad (12)$$

It should be noted that other methods of describing the rotation are also possible. One such possibility would be to define the orientation of the surface

relative to the Frenet frame [34] of the curve, formed by the curve tangent, normal, and binormal vectors, which may in some cases give a more “natural” basis for describing the orientation of the moving object than a fixed set of axes. One obvious case is when an object rotates rigidly in a circle about a fixed point, such that it keeps the same orientation relative to the normal to the circle, *i.e.* the same radial orientation. Klok [28] provides an useful discussion on the choice of frames when sweeping two-dimensional objects along a space curve, and shows how to handle problems which arise if the curve normal becomes ill-defined.

In practice, the original surface will be expressed in one of two forms. In a CSG modeller, it is most likely that the original surface will be expressed in implicit form as already given in Equation 1:

$$f(x, y, z) = 0. \tag{13}$$

This is a relationship between the x, y and z components of $\mathbf{p}(0)$. The corresponding equation between the x, y and z components of $\mathbf{p}(t)$ gives the implicit equation for the hypersurface, $F(x, y, z, t) = 0$. This can readily be found by inverting Equation 12 to make $\mathbf{p}(0)$ the subject, and substituting the components into Equation 13.

On the other hand, more particularly in boundary representation modellers, parametric surface descriptions of the form

$$\mathbf{r} = (x(u, v), y(u, v), z(u, v)) \tag{14}$$

are more likely to be used. In this case, a representation for a point on the hypersurface, in parametric terms, is simply given by

$$\mathbf{R} = (x(u, v, t), y(u, v, t), z(u, v, t), t) \tag{15}$$

where x, y and z are the components of $\mathbf{p}(t)$ found by substituting \mathbf{r} for $\mathbf{p}(0)$ in Equation 12, and the fourth component of \mathbf{R} is simply t .

In the latter case, although in principle u and v could be eliminated between the first three components to give an implicit form for the family of surfaces, it is also possible to work directly with the parametric form for the hypersurface, as will be seen in the next section.

4 Finding the Envelope

4.1 Computer Algebra Methods

The next task is to use the hypersurface found above to determine the envelope to the family of moving surfaces, by eliminating t between the two functions given in Equations 3. On the one hand, this can obviously be done in an *ad-hoc* manner for particular instances of sweeping, such as a range of sweep types that

one wishes to allow in a CAD system being constructed. The results can then be programmed individually for incorporation into the modeller. The disadvantages of doing this include the need to consider special cases, and the large amount of code resulting from the number of possible types of sweep. On the other hand, a much more flexible approach is to have a procedure within the CAD system itself capable of performing the necessary elimination manipulations as they are required.

In practice, in most CAD systems, surfaces of interest are described as polynomials, or rational polynomials (whether as implicit or parametric functions). If it is possible to express the hypersurfaces describing the moving shape in terms polynomial or rational polynomial functions, advantage can be taken of certain algorithmic procedures for eliminating variables between such equations, but if other more complex functions of the variables are used, such as exponentials and logarithms, this is not generally so. To perform the elimination, the polynomial equations involved will be manipulated *algebraically* (*i.e.* by multiplying them, differentiating them, and so on, to produce new polynomials) using the methods of Computer Algebra [19]. Here, it is advocated that the CAD system should include such computer algebra routines as are necessary to find envelope surfaces by carrying out the elimination calculations. Further discussions of the use of computer algebra techniques in geometric modelling are given in [6,7,32,42].

Although these methods are restricted to polynomial, or rational polynomial, equations, it may be possible to convert other forms to them in some instances. For example, $\sin(v)$ and $\cos(v)$ can be represented in rational polynomial form as $2u/(1+u^2)$ and $(1-u^2)/(1+u^2)$ by means of the substitution $\tan(v/2) = u$. Another possibility if a set of n equations includes $\sin(v)$ and $\cos(v)$ is to replace the latter pair with new variables s and c , and to include the extra equation $s^2 + c^2 = 1$, giving a new set of $n+1$ equations, containing one more unknown, but do not include explicit trigonometric functions. Such methods will not be of help, of course, if v appears both within the argument of trigonometric functions and in terms of powers of itself, as the trigonometric functions are transcendental.

One method of eliminating variables between algebraic equations is to use resultants. Sylvester's resultant has already been used by Sederberg [36], as mentioned above, for performing elimination in the context of the implicitization of parametric surfaces. The use of the more powerful multi-equational resultants for elimination in geometrical problems has been described by Bajaj *et al.* in [2]. In particular, Macaulay's form of the resultant has the advantage that the solution found does not contain any extraneous factors. Thus, Macaulay's resultant will produce $e(x, y, z) = 0$ for the envelope surface on eliminating t from Equations 3, while other resultant methods may produce solutions of the form $e(x, y, z)a(x, y, z) = 0$, where the factor $a(x, y, z) = 0$ is an unwanted artifact introduced by the method of computing the resultant. This is obviously an important consideration when interpreting the result, as well as

in keeping subsequent calculations as simple as possible.

A second method of performing elimination is to use Buchberger's Algorithm for computing Gröbner Bases [10,11,12]. This is rather more difficult to understand, and relies on deeper mathematical ideas, but again has the advantage of not introducing spurious factors. However, this method can in the worst case require a running time doubly exponential in the number of variables in the equations, while the resultant methods only need singly exponential time [2]. No detailed comparisons of the use of these two approaches to performing geometric manipulations appear to have yet been made.

4.2 Envelopes for Implicit Surfaces

The hypersurface representing the moving surface may either be given in implicit or parametric form. Taking the first of these possibilities, we have

$$F(x, y, z, t) = 0 \tag{16}$$

which can be symbolically differentiated with respect to t by algebra routines to obtain

$$\frac{\partial F(x, y, z, t)}{\partial t} = 0. \tag{17}$$

Straightforward application of either resultants or Gröbner Bases to eliminate t from this pair of equations will produce the envelope surface in the form

$$e(x, y, z) = 0. \tag{18}$$

However, this solution may not be as simple as it seems at first. For example, consider, in two dimensions for simplicity, sweeping the small circle initially described by

$$(x - 5)^2 + y^2 - 1 = 0 \tag{19}$$

around a circular path, so that its centre moves along the locus

$$x^2 + y^2 - 5^2 = 0, \tag{20}$$

as shown in Figure 1. It is easy to see that the geometric result is the pair of concentric circles given by

$$x^2 + y^2 - 4^2 = 0, \quad x^2 + y^2 - 6^2 = 0. \tag{21}$$

However, the result produced by the elimination methods above has the form

$$x^4 + y^4 + 2x^2y^2 - 52x^2 - 52y^2 + 576 = 0. \tag{22}$$

In general the expression resulting from elimination has to be factorised to separate the different geometric entities which make it up:

$$(x^2 + y^2 - 4^2)(x^2 + y^2 - 6^2) = 0. \tag{23}$$

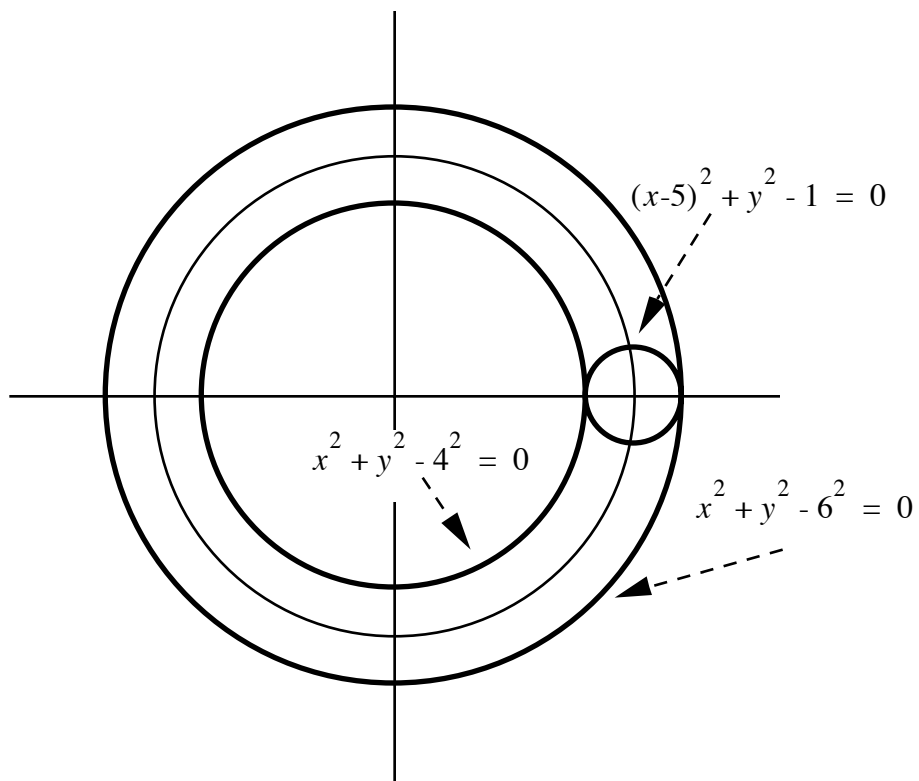


Figure 1: Sweeping a Circle, Resulting in a Pair of Curves

This concept is taken further by Neff [33], who shows how Gröbner Bases together with factorization can be used to decompose any algebraic set (an object defined by the common zeros of a set of polynomial equations, rather than the single equation considered here) into its irreducible components.

Although factorization routines are a feature of most current computer algebra systems, the problem of separating the components is in fact more difficult than suggested above. Again this is perhaps best illustrated by means of an example. Taking the pair of vertical lines two units from the y axis, $x = +2$ and $x = -2$, they will have the combined equation of $(x-2)(x+2) = 0$, or $x^2 - 4 = 0$, which can readily be factorised. However, a seemingly very similar case, the pair of lines parallel to and two units from the line $y = x$, have a combined equation of $x^2 + y^2 - 2xy - 2 = 0$, which factorises into $(x - y - \sqrt{2})(x - y + \sqrt{2}) = 0$. The crucial difference here is the presence of surds. Factorization routines usually used by most algebra systems are not capable of finding factors involving arbitrary roots. Nevertheless, algorithms are available for solving this question of “absolute irreducibility”, although they are quite complex [26,27].

It should be noted that although factorization provides a start to answering the question of how many pieces of surface a given implicit equation represents, it certainly does not give a complete answer. Again considering an example in two dimensions for simplicity, *elliptic curves* are those having equations of the form

$$y^2 - ax^3 - bx^2 - cx - d = 0. \quad (24)$$

In general, this expression cannot be factorised, but may have either one or two branches depending on whether the cubic in x has one or three real roots respectively. Examples of each type of curve are shown in Figure 2. (The two values of y which correspond to a single value of x always belong to the same branch, above and below the x -axis.)

Separating the different connected components of a surface presents a problem for other areas of computational geometry. For example, Woodward [48] describes how components other than the desired one may unexpectedly appear when using Liming-type methods for creating blending surfaces. Canny, in his thesis [16], shows how the existence of a suitable path for a robot can be reduced to the problem of deciding whether two points of an n -dimensional surface lie in the same or different components. Usefully, he goes further, and also gives an algorithm for solving this problem, which he calls the *Roadmap* Algorithm. It basically works by recursively reducing n -dimensional entities down to one-dimensional ones, such that each connected component of the n -dimensional entity eventually corresponds to a single piece of the one-dimensional one. Of course, it is then much easier to traverse the one-dimensional entity to determine its pieces. In passing, it should also be mentioned that his thesis contains much other material, for example on resultants and silhouettes, which is relevant to the current paper.

Finally, it is worth remarking that although in boundary modellers it will

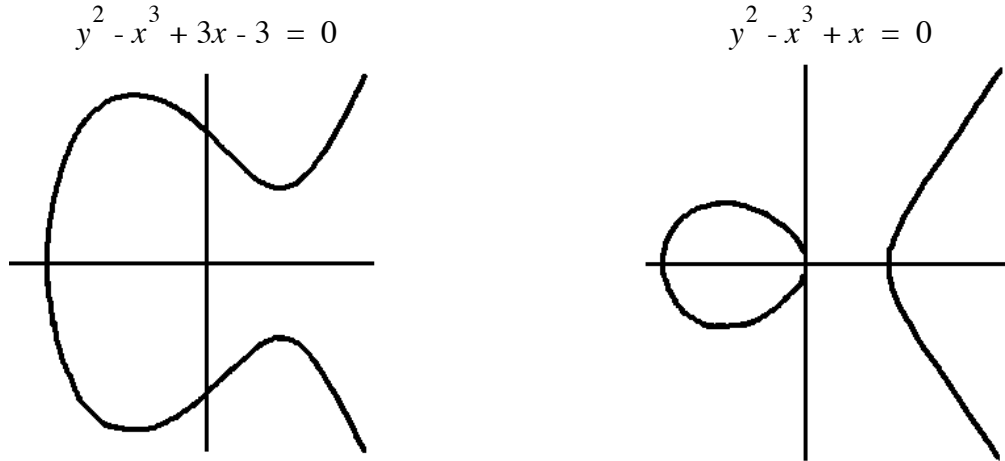


Figure 2: Elliptic Curves with One and Two Branches

be usually be necessary to know exactly how many connected pieces the swept envelope comprises (as they will be described as separate bodies), it may be possible in CSG modellers to avoid these complexities under certain circumstances. For example, if it is only desired to draw a ray-traced picture of a solid composed of several pieces, the ray tracing process itself will be sufficient to distinguish between the components.

4.3 Envelopes for Parametric Surfaces

In the case where the hypersurface describing the sweep has the parametric form $\mathbf{R} = (x(u, v, t), y(u, v, t), z(u, v, t), t)$, it is possible to find the envelope surface as follows. Let us suppose that this hypersurface can be put into the implicit form $F(x, y, z, t) = 0$ (it will be seen later that it is not necessary to actually carry out this step). Then the envelope surface satisfies both this equation, and $\partial F(x, y, z, t)/\partial t = 0$. Thus, taking the total derivative of the former equation, and substituting the latter equation into the result, it can be seen that

$$\frac{\partial F}{\partial x} \frac{\partial x}{\partial t} + \frac{\partial F}{\partial y} \frac{\partial y}{\partial t} + \frac{\partial F}{\partial z} \frac{\partial z}{\partial t} = 0. \quad (25)$$

However, the surface normal \mathbf{N} to a particular member of the family of moving surfaces can be expressed either, if the surface is written in implicit terms, as

$$\mathbf{N} = \left(\frac{\partial F}{\partial x}, \frac{\partial F}{\partial y}, \frac{\partial F}{\partial z} \right), \quad (26)$$

or as

$$\mathbf{N} = \frac{\mathbf{r}_u \times \mathbf{r}_v}{|\mathbf{r}_u \times \mathbf{r}_v|}, \quad (27)$$

if the surface is expressed parametrically, where the subscripts denote differentiation. Replacing the second of these forms for the first in Equation 25, which can be rewritten as $\mathbf{N} \cdot \mathbf{r}_t = 0$, it can be seen that the following determinant must be zero for points on the envelope surface:

$$\begin{vmatrix} \frac{\partial x}{\partial t} & \frac{\partial y}{\partial t} & \frac{\partial z}{\partial t} \\ \frac{\partial x}{\partial u} & \frac{\partial y}{\partial u} & \frac{\partial z}{\partial u} \\ \frac{\partial x}{\partial v} & \frac{\partial y}{\partial v} & \frac{\partial z}{\partial v} \end{vmatrix} = 0. \quad (28)$$

This is a relationship between u , v and t , which can be used to eliminate either u or v (whichever is simpler) from the expressions $x = x(u, v, t)$, $y = y(u, v, t)$, and $z = z(u, v, t)$. With luck, x, y and z will appear as linear factors in the result, in which case, it will be possible to express the envelope surface in parametric form as

$$e = (x(v, t), y(v, t), z(v, t)) \quad (29)$$

or

$$e = (x(u, t), y(u, t), z(u, t)). \quad (30)$$

It is much more likely, however, that any or all of x, y and z will appear in more complex combinations with the other two variables. In general, the surface form which will result will be

$$\begin{aligned} g(x, u, t) &= 0 \\ h(y, u, t) &= 0, \\ k(z, u, t) &= 0 \end{aligned} \quad (31)$$

assuming v was eliminated rather than u . To find points on the envelope surface corresponding to a particular choice of u and t , these three non-linear equations for x, y and z must be solved. Deciding which of the multiple solutions for x corresponds to which of those for y and z may be non-trivial, requiring backsubstitution into the original parametric form for the hypersurface, and checking the corresponding values for v .

The surface form given above is really a hybrid form. It is neither explicitly a parametric form for x, y and z as functions of u and t , nor is it an implicit form for the surface. It is perhaps best described as an *implicit parametric* form, because there is an implicit equation for each of x, y and z . On the one hand, it can readily be converted into the implicit form for the surface, by elimination of the variables u and t between the three equations using the techniques discussed

earlier. On the other hand, the starting point for this discussion was originally a moving *parametric* surface, and it is quite probable that any solid modeller describing the original moving surface in parametric form would also require, or at least prefer, the final description of the envelope surface in parametric form rather than implicit form. Unfortunately, this can not in general be found. Although any polynomial parametric surface can be implicitized, the converse is not true, and it is not always possible to find rational polynomial parametric forms for implicit surfaces of degree greater than three [1,12], which in turn implies that Equations 31 cannot individually be put into parametric form as a first step towards finding a parametric form for x , y and z .

4.4 Self-Intersection of the Envelope

One problem which may arise is that the envelope surface may intersect itself. Loosely speaking, this is likely to happen if the object is moving along a curved path which locally has a small radius of curvature compared to the size of the object. It may also happen if the path passes close to earlier points on the path, close being interpreted as meaning a within a distance less than the size of the object.

This *may* be of no real consequence if a CSG modeller is being used, as for example a ray tracing procedure will find the outer boundary of a self-intersecting surface by virtue of the way ray-tracing works. On the other hand, it may be quite important to remove unwanted, inner, pieces of surface in the context of a boundary modeller which relies on all objects it is handling being manifold objects. In this case, an edge of self-intersection would be considered locally as having four distinct faces around it, and thus a surface containing such an edge would be a non-manifold object. Although recent progress has been made in the description of non-manifold objects by boundary modellers [46], it is probably still desirable to remove all pieces of the envelope surface which lie inside the envelope volume.

In fact, two types of self-intersection of the envelope surface may be distinguished, as shown in the two-dimensional example given in Figure 3, which represents a circle moving around a closed figure-of-eight path. On computing the envelope curve, it will be found to have two factors e_1 and e_2 such that

$$e_1 e_2 = 0. \tag{32}$$

Firstly, the different factors may in general intersect each other, as shown in this example at points A and B . Secondly, the individual factors may intersect themselves, as e_1 does at S_1 , and e_2 at S_2 . The overall swept shape desired is found in this case by taking each of the segments of e_1 and e_2 , limited by these intersections, and joining them together appropriately. Consideration of test points on each such part will show whether they are to be included in the final result.

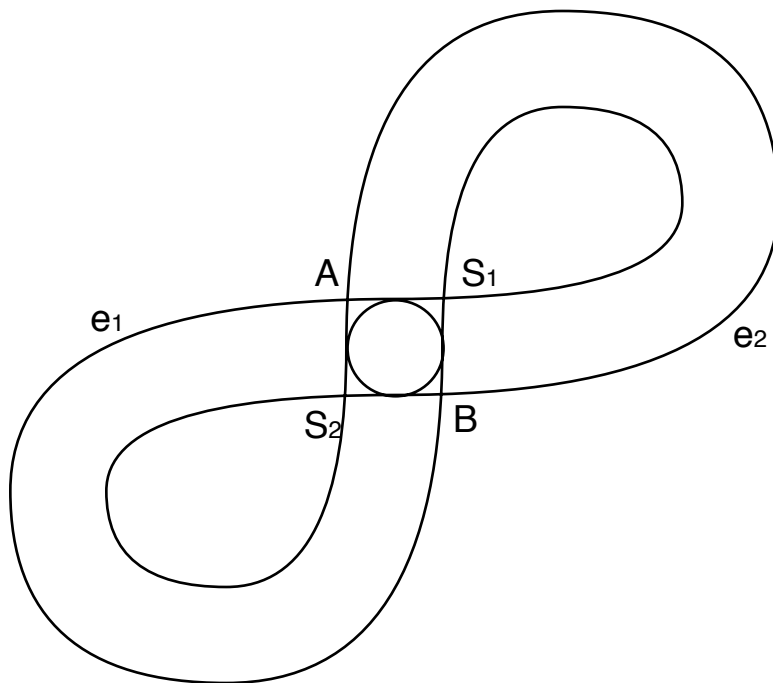


Figure 3: Self-Intersecting Envelope

Intersections between the different factors can be computed using the usual methods of surface intersection. Bajaj *et al.* give a useful discussion of reliable methods of performing surface intersections using algebraic techniques in [3].

Ideas from algebraic geometry again prove to be useful in finding and classifying the singular points and curves of a single surface [41]. In general, at singular points the surface normal becomes ill-defined, such as at the tip of a cone, or along an edge where a surface penetrates itself. Thus, the singular points of the surface $e(x, y, z) = 0$ can be found by simultaneously solving

$$e(x, y, z) = 0, \quad \frac{\partial e}{\partial x} = 0, \quad \frac{\partial e}{\partial y} = 0, \quad \frac{\partial e}{\partial z}, \quad (33)$$

the last three equations expressing the condition that the surface normal is indeterminate at the point. Another way of looking at this is to say that such self-intersections represent multiple points on the surface, and a value which is a multiple root of a polynomial equation must satisfy both the equation itself, and its derivative.

When the envelope surface is described in *implicit parametric* form, as in Equation 31, one approach to finding self-intersections is to implicitize the surface and proceed as above. Alternatively, a more numerical approach attempts to find corresponding values of u_1, u_2, t_1 and t_2 such that

$$\begin{aligned} g(x, u_1, t_1) &= g(x, u_2, t_2) \\ h(y, u_1, t_1) &= h(y, u_2, t_2), \\ k(z, u_1, t_1) &= k(z, u_2, t_2) \end{aligned} \quad (34)$$

in a manner reminiscent of that used for finding the intersection of two different parametrically defined surfaces.

Self-intersections are not the only type of surface singularity. Other possibilities include cuspidal edges, and isolated points, for example. Examination of the higher derivatives of the surface equation serve to distinguish between self intersections and other surface singularities [4] when unwanted pieces of surface must be discarded later. Further details of how to cope with degeneracies when interrogating surfaces can be found in [3].

5 Sweeping Solid Models

The result of sweeping a single surface was discussed in the previous section. However, solid models are usually bounded by many simple faces each with their own surface equations. The aim of this section is to show how the results of sweeping each of the faces in the original model may be combined to find the new overall swept volume. The main discussion which follows will be in the context of a boundary modeller, as the process to be described produces an explicit representation for the boundary of the swept volume. Note that because of the

limitations discussed above, although the original moving object may have its faces described either as implicit or parametric surfaces, the description of the faces of the swept volume will (usually) only be available in implicit, or implicit parametric form.

5.1 Boundary Modelling of Swept Shapes

In order to illustrate the algorithm to be described in this section with meaningful diagrams, the process will be described in one dimension lower, *viz.* a two-dimensional composite object will be swept along a two-dimensional path, giving a three-dimensional volume, which will then be collapsed back down into the desired two-dimensional swept area. This is justified in that the algorithm generalises in a straightforward manner to the higher-dimensional case which is really the aim of the work.

The basic process is as follows. Each of the boundary curves in the original model is formed into the corresponding hypercurve (surface) in three dimensions, by including its time dependence. A three-dimensional model is created by joining these (x, y, t) surfaces together. Assuming that the motion starts and stops at particular times, and does not follow a closed (repeating) path, the three-dimensional model must be restricted to a finite interval of time, and closed by using the initial and final two-dimensional models as its end faces, after limiting the other surfaces to that period of time which is of interest, *i.e.* between the starting and finishing times.

An example of this process is shown in Figure 4. The x and y axes are in the plane of the paper, with t going into the paper. The original two-dimensional shape has been swept along a straight line. Note that each edge in the original shape leads to corresponding surface in the three-dimensional volume. The extents of these faces, and the order in which they are joined to each other can readily be determined by consideration of the original two-dimensional shape. Note also that the initial and final positions of the two-dimensional shape give the end faces of the three-dimensional volume.

In greater detail, the edges of the three-dimensional volume are composed of the swept vertices of the original two-dimensional model, together with the edges belonging to the initial and final positions of the two-dimensional shape. Alternatively, the former edges of the three-dimensional volume may also be thought of as the intersections of the surfaces formed by sweeping the original edges.

From this three-dimensional volume, the two-dimensional swept shape is now to be computed. The edges of the two-dimensional swept shape will be of three types. These are, firstly, edges, or parts of edges, of the models of the original two-dimensional shape at its starting and finishing times; secondly, projections of the whole or parts of the other edges of the three-dimensional volume onto the x - y plane; and thirdly, the whole or parts of edges of the two-dimensional envelope curves obtained by sweeping the various original two-

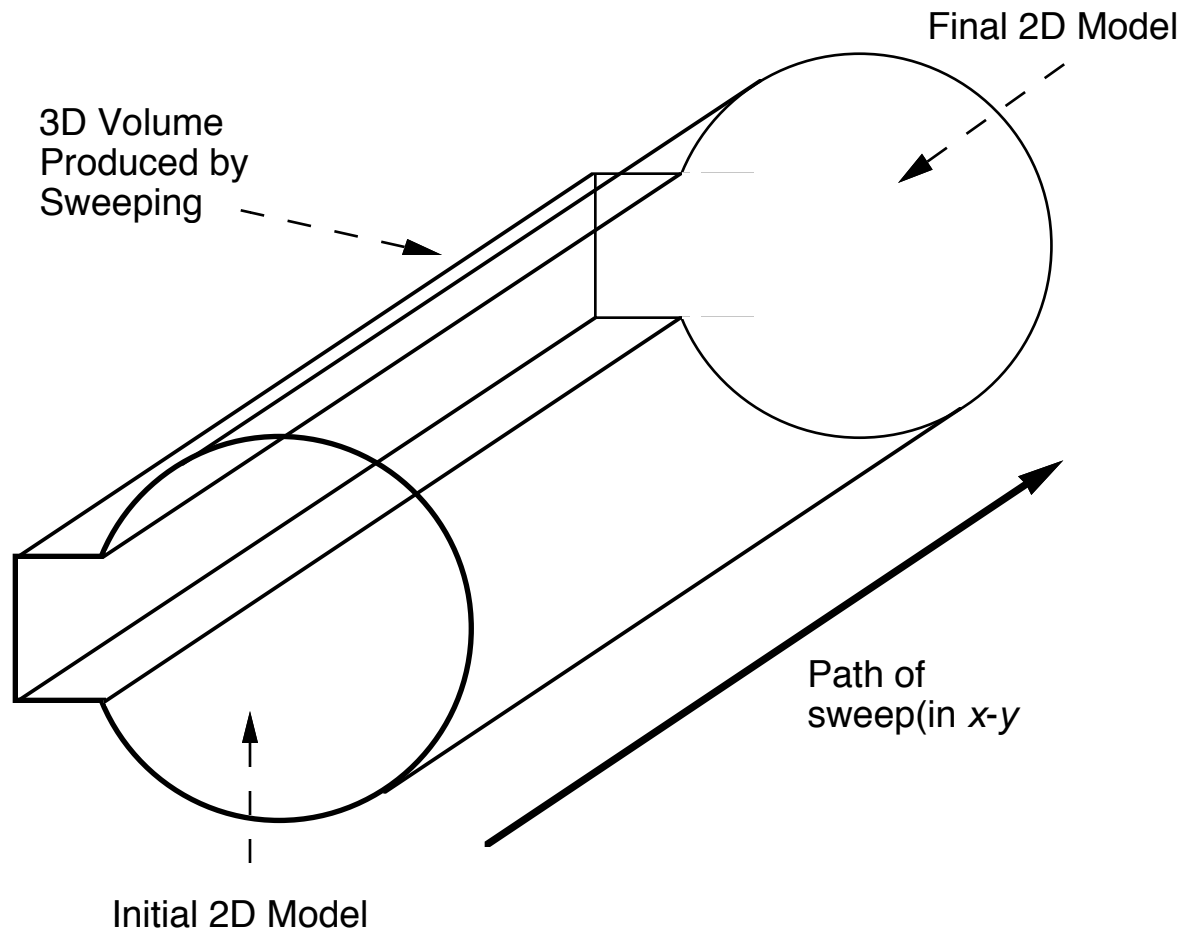


Figure 4: Sweeping a 2D Model to Produce a 3D Volume

dimensional edges. As an example, compare the final result as shown in Figure 8 with Figure 5, which shows edges of each of these three types that go towards making up the final swept shape.

Edges of the first type described above are immediately available. For the second type, if two adjacent edges of the original two-dimensional model, $f(x, y) = 0$ and $g(x, y) = 0$, have been swept into the three-dimensional surfaces $F(x, y, t) = 0$ and $G(x, y, t) = 0$ respectively, the implicit form for the equation of the projection of the edge between them onto the $t = 0$ plane can be found by eliminating t between these two equations. (If the original edges were described in parametric form, the simplest method seems to be to implicitize the parametric surface forms obtained after sweeping before attempting to eliminate t . Other approaches are possible, however, based on surface intersection techniques such as those mentioned earlier).

Edges of the third type required are envelope curves, and methods of finding their algebraic form have been discussed (at least, in the three-dimensional case) earlier.

Having generated these three types of edges, to find the overall swept shape, we only wish to keep those pieces of them which form the external boundary of the projection of the three-dimensional volume (and the boundaries of any holes within it). These pieces enclose the region of the x - y plane which our moving two-dimensional object has passed through.

One way of performing this restriction is as follows. After each edge of each type has been projected onto the x - y plane, all intersections between pairs of edges are found, as are self-intersections of single edges. Each edge is broken into segments lying between adjacent intersection points. It can now be seen that each segment will *either* contribute in its entirety to the boundary of the swept shape, *or* it will lie entirely within the shape (except possibly for its end points), and so can be discarded. This process is illustrated in Figure 6, where each segment has been marked.

A minor complication is that some of the segments may overlap in projection, in whole or in part. This will occur, for example, when a straight edge of the object ends up after the motion occupying the same place that the same or another straight edge occupied at the start of the motion. In such cases, duplicate segments should be removed at this stage to avoid problems with repeated edges when constructing the final model.

The next step is performed once more in three dimensions. A representative point (not an end point) is chosen on each of the segments. Firstly, the object is looked at from the front, *i.e.* in the $+t$ direction, and a ray is cast to each representative point. Those segments corresponding to points which cannot be seen, because another part of the three-dimensional volume is in the way, are discarded. As can be seen by the results shown for our example object in Figure 7, the pieces left are exactly those which form a hidden-line view of the three-dimensional shape as seen from the front.

The final step is then to perform a similar computation looking from the back

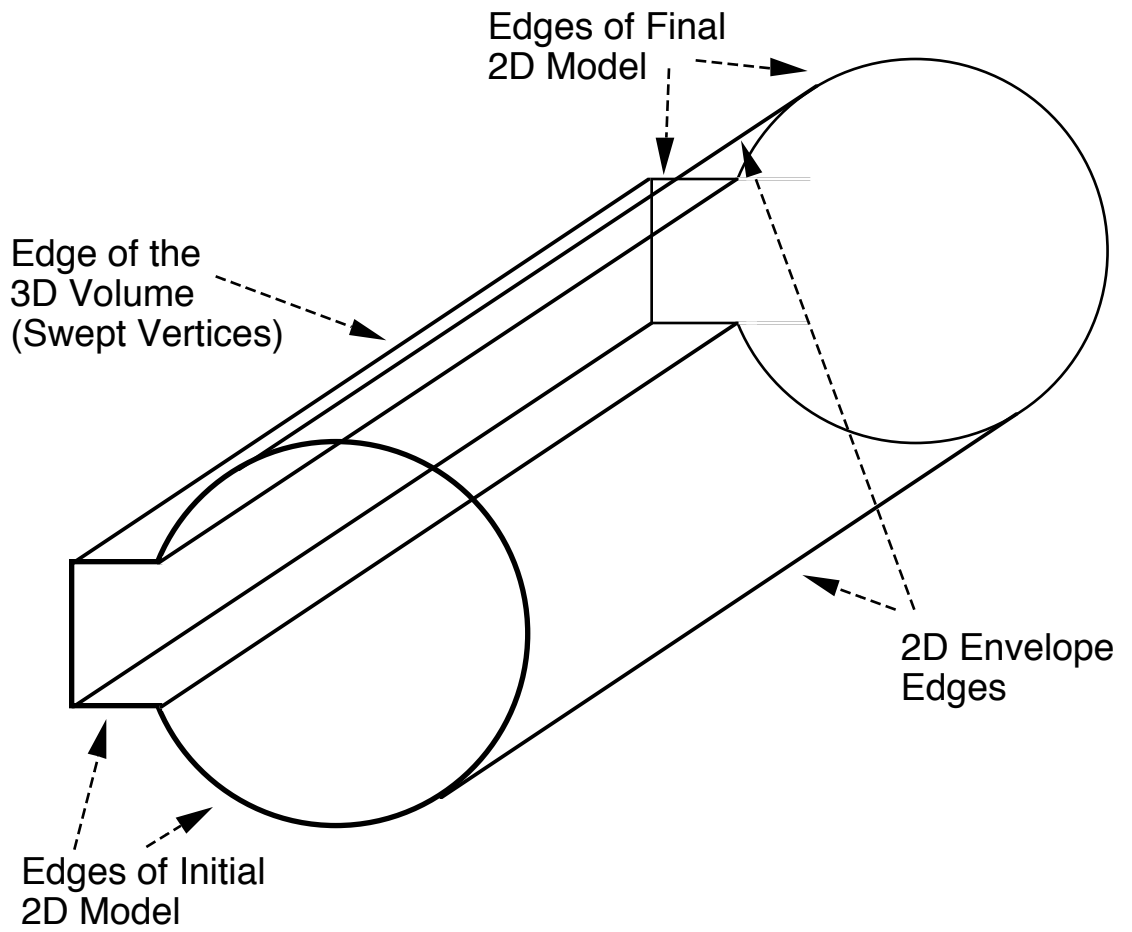


Figure 5: Types of Edges Making Up the Swept 2D Shape

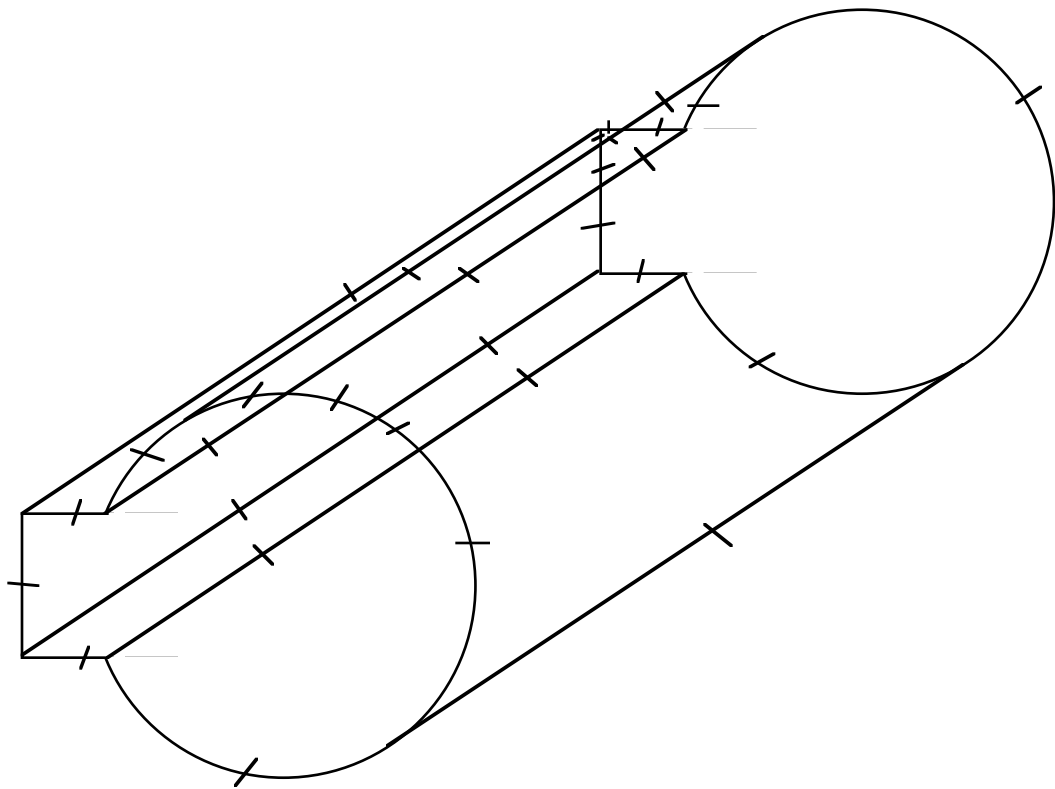


Figure 6: Projections of Edges Segmented and Marked

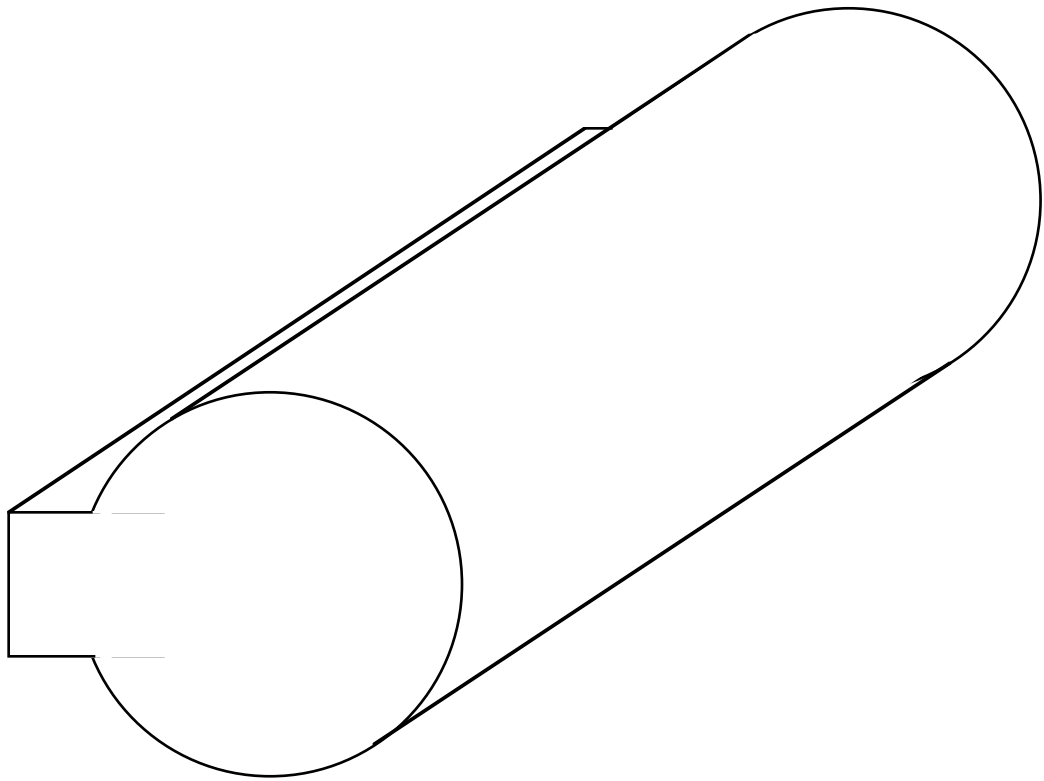


Figure 7: Elimination of Segments Not Visible from the Front

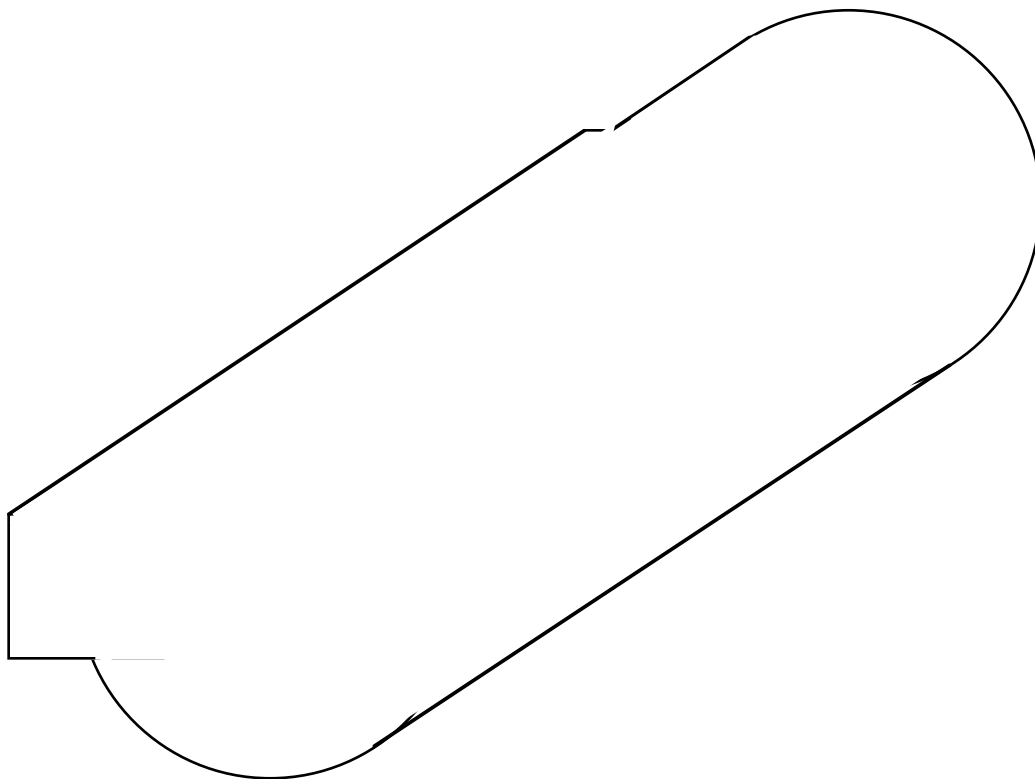


Figure 8: Segments Not Visible from the Rear also Eliminated

of the object, in the $-t$ direction, again discarding those segments which cannot be seen. The final result after both stages of discarding unwanted segments have been performed is exactly that set of segments which form the boundary of the two-dimensional swept object, as shown for the current example in Figure 8.

A simple examination of the segments which remain will allow them to be readily pieced together in the correct order to form a boundary model (in two dimensions) of the final swept area, as desired.

Although this solves the problem, let us now reconsider the issue of generating the envelope curves required. If a three-dimensional object which includes curved faces is represented as a wireframe drawing, then not only are the real edges drawn in the projection, but also virtual, *silhouette*, edges must be added, at places where a curving face bends so that it changes from pointing towards the viewer to pointing away from the viewer, as shown in Figure 9. This is of direct interest, because the two-dimensional envelope curves of the original moving curves *are exactly* the silhouette curves of the three-dimensional volume when its projection is drawn on the $x-y$ plane, as can be seen by looking at the

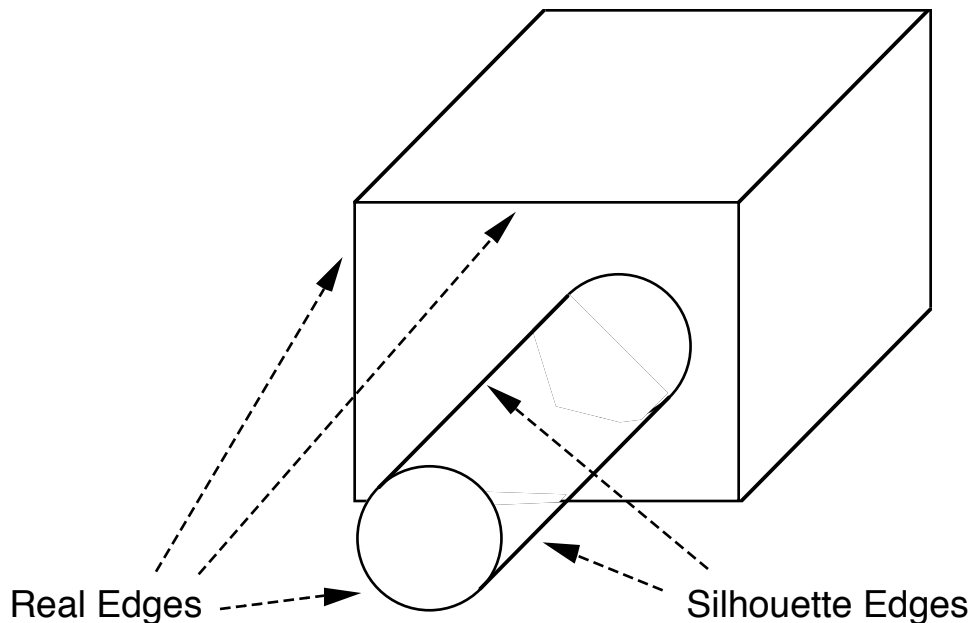


Figure 9: Silhouette Edges

example in Figure 4. This assertion can readily be justified. If some surface of the three-dimensional volume has implicit form

$$f(x, y, t) = 0, \quad (35)$$

then the normal vector to this surface is given by

$$\mathbf{N} = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial t} \right). \quad (36)$$

On projecting this surface onto the x - y plane, the silhouette curve comprises the boundary between when the t component of the surface normal changes from pointing towards the viewer (negative) to pointing away from the viewer (positive), *i.e.* those points where the t component of the surface normal is zero: $\partial f / \partial t = 0$. The silhouette curve simultaneously satisfies this equation, and $f = 0$, as it lies on the surface, which is exactly the pair of equations that the envelope curve satisfies.

The usefulness of this observation that envelope curve generation in two dimensions is the same problem as silhouette curve generation for drawings of three-dimensional objects, is that the silhouette curve problem is a well known one in computer graphics. Thus, algorithms developed for silhouette curve generation can be used in a rather different context to generate envelope curves.

Indeed, as already pointed out, the steps described above for finding the swept outline are exactly the same as those needed to compute a hidden-line view with silhouette curves of a three dimensional object, except, of course, that ray-firing has taken place from both the front and the back, whereas the production of a hidden-line picture would require only examination from the front. Thus, any of the ideas already explored in the literature for the generation of such pictures could be used to produce two-dimensional swept shapes from corresponding three-dimensional volumes. Although this might seem to be a promising avenue for seeking other approaches to the straightforward one discussed above (for example, using a preprocessing step to eliminate some curves in their entirety *before* intersecting them), the literature on producing such pictures seems surprisingly poor. Many papers only discuss how to proceed with parametric surfaces, or they start by faceting their model first [25,30,35,47].

So far, the discussion has covered the simplified task of constructing the swept area of a moving *two*-dimensional shape. Whilst this problem is of interest in its own right in computer graphics, as it describes the trail of ink left behind by a moving pen or brush [24], we will now reconsider the original problem of finding the volume swept out by a moving solid. Each of the steps described above can be straightforwardly generalised to one dimension higher. Thus, the three-dimensional surfaces are swept to form a four-dimensional model, which is terminated by the initial and final three-dimensional solids. The other boundary (volume) elements of the four-dimensional model are the swept edges of the original three-dimensional model. These, on projection back into three dimensions, together with envelope surfaces of the moving three-dimensional surfaces, will give rise to the faces of the new three swept volume to be formed. Simple consideration of the three-dimensional models will readily permit decisions as to how the pieces of the four-dimensional model fit together. Intersections of the projections of these various surface elements are then computed, again resulting in segments which can be tested for inclusion or exclusion in the boundary of the final model by ray-firing as before. Again, the ideas in Bajaj *et al.* [3] will be of use. Techniques such as those proposed by Canny [16] may be used to keep track of the various surface pieces. Both when computing and using the envelope surfaces, care must be taken to detect and correctly handle degenerate cases such as those discussed in Section 2.2, for example if no envelope surface exists, or if it has reduced dimensionality.

A final point which has not been considered so far, but can be easily handled, is how to cope with a motion that is not continuous, but is described piecewise in terms of several simpler motions one after the other, such as sweeping a given distance in the x direction immediately followed by sweeping a distance in the y direction. The most straightforward approach is to evaluate the sweeps corresponding to the different parts of the motion separately, and then use the usual set union operator to combine the results together. The correctness of this method follows directly from the definition of a sweep as being the continuous union of all instances of the moving object at every moment of time during the

motion.

An alternative approach involves constructing an overall four-dimensional model for the whole motion, which now consists of the four-dimensional models for each of the separate parts of the motion, joined together on faces corresponding to the three-dimensional models of the object at the times when the motion changes over from one part of the piecewise description to the next. These changeovers play a similar role to the inclusion of the initial and final models for a simple motion.

5.2 CSG Modelling of Swept Shapes

We will now briefly consider how the above ideas might be used in a set-theoretic context. Firstly, it should be noted that the computation of the four-dimensional model from a description of the path and the three-dimensional model is simple, at least in principle. This is because the four-dimensional model resulting from sweeping a set-theoretic combination of primitive solids along a given path is equivalent to sweeping each of the primitives individually, and then combining them using the same set-theoretic tree, as shown by Cameron [13]. (Note the difference between this result, and the one mentioned earlier, which is that when this four-dimensional volume is projected back into three dimensions to find the three-dimensional swept volume, the intersection and difference operators *do not* commute with the combined operation of sweeping into four dimensions followed by projection back into three dimensions.)

The four-dimensional model created above now has to be projected back onto the $t = 0$ hyperplane. As discussed above, this process is rather akin to creating a drawing. Apart from the process of ray-tracing which produces a primitive kind of picture, *i.e.* a set of pixels, methods of creating pictures of set-theoretic models rely on converting them to a suitable boundary representation model first. Thus, it would seem that any description which we produce of the swept volume will be in boundary representation terms. However, the further use of this volume within a set-theoretic modeller would require its conversion into set-theoretic terms. Unfortunately, conversion of boundary models into set-theoretic ones is a long-standing problem of geometric modelling, with no well understood solution at present. One method relies on recursively taking convex hulls of the object and subtracting what is left, but has problems with curved objects, and algorithm termination. A positive piece of work in this area by Vossler [40] shows a method of converting boundary representations of two-dimensional objects consisting of circular arcs and straight lines into a set-theoretic description.

6 Conclusions

This paper has presented a theoretical basis for computing the volumes swept out by solid objects as they move. This work brings together ideas from current solid modelling technology, differential geometry, computer algebra and computer graphics algorithms. Although preliminary experiments have been carried out, it is fair to say that a fully working system has not yet been put into practice, mainly because of administrative problems with the research grant rather than because of practical difficulties.

It should be pointed out that working systems including each of the major components discussed do exist, and implementing a single system which ties them together should present no unforeseen difficulties. (General four-dimensional modelling will not be required, but only a restricted set of operations). On the other hand, careful consideration *will* be required when taking into account some of the detailed points discussed, such as the correct detection and handling of degenerate envelope surfaces.

As the discussion in the Introduction shows, there are many applications in Computer Aided Engineering which could usefully use the computation of swept volumes. An obvious question to ask is whether this work is likely to be able to produce practically useful results, particularly in a realistic length of time.

The two main components of the method presented are the algebraic manipulation stages, and the algorithm for combining the results of sweeping the individual faces of the object. The latter should take a similar amount of time (in order of magnitude terms) to producing a hidden-line removed picture of the original object, as each two-dimensional face in the solid model leads to a single three-dimensional hypersurface in the four-dimensional model. As such pictures are produced as a matter of routine by current systems, this gives no cause for concern.

The more straightforward computer algebra manipulations can be carried out rapidly, in milliseconds or less. However, elimination calculations can take several minutes, tens of minutes, or more, using current workstation technology, even for fairly simple surfaces moving along fairly simple paths. It is thus fair to observe that perhaps some of the more complicated sweeps of current engineering interest can not be computed in a realistic amount of time with current hardware technology and computer algebra methods. Furthermore, this observation is perhaps unlikely to change greatly with time, because of the exponential behaviour of the algorithms involved, and the fact that the resulting algebraic surfaces will necessarily be complicated objects. For example, even a simple bicubic parametric patch, when expressed in implicit form, is represented by a degree 18 equation with 1330 coefficients [36], and this is even before its motion has been considered! Further work is needed to investigate what types of surface and motion can be realistically handled these methods, before the true practical usefulness of the approach advocated here can be finally assessed.

7 Acknowledgements

We would like to thank Adrian Bowyer, James Davenport and Philip Milne of the University of Bath, Alan Middleditch of Brunel University, Stephen Cameron of Oxford University, and Chanderjit Bajaj of Purdue University for many helpful discussions and insightful comments during the course of this work. We would also like to thank the SERC ACME Directorate for funding this work under Grant Number GR/D 59434.

References

- [1] S. S. Abhyankar, C. Bajaj *Automatic Parametrization of Rational Curves and Surfaces II: cubics and cuboids* Computer Aided Design, 19 (9) 499-502, 1987.
- [2] C. Bajaj, T. Garrity, J. Warren *On the Applications of Multi-Equational Resultants* Technical Report CSD-TR-826, Computer Sciences Department, Purdue University, 1988.
- [3] C. Bajaj, C. M. Hoffman, J. E. Hopcroft, R. E. Lynch *Tracing Surface Intersections* Technical Report CSD-TR-728, Computer Sciences Department, Purdue University, 1987.
- [4] R. J. T. Bell *Coordinate Geometry of Three Dimensions (Second Edition)* Macmillan, 1912.
- [5] V. C. Boltyanskii *Envelopes* Pergamon, 1964.
- [6] A. Bowyer, J. H. Davenport, P. S. Milne, J. Padget, A. Wallis *A Geometric Algebra System* Proc. 1986 Conference on Geometric Reasoning, IBM UK Scientific Centre, Winchester, England, 1986.
- [7] A. Bowyer, A. F. Wallis, P. S. Milne *Symbolic Ray Tracing* Proc. Computer Graphics 87, Online Conference Publications, London, 1987.
- [8] J. W. Boyse *Interference Detection Among Solids and Surfaces* Communications of the ACM, 22, 1979.
- [9] J. W. Bruce, P. J. Giblin *Curves and Singularities* Cambridge University Press, 1984.
- [10] B. Buchberger *Gröbner Bases: An Algorithmic Method in Polynomial Ideal Theory* in Recent Trends in Multidimensional Systems, Ed. N. K. Bose, D. Reidel, 1983.

- [11] B. Buchberger *Applications of Gröbner Bases in Non-Linear Computational Geometry* Proc. Workshop on Scientific Software, Minneapolis, 1987. IMA Volumes in Mathematics and its Applications 14, Springer Verlag 1987.
- [12] B. Buchberger *Algebraic Methods for Geometric Reasoning* Ann. Rev. Comput. Sci., 3, 85-119, 1988.
- [13] S. A. Cameron *Modelling Solids in Motion* Ph. D. Thesis, University of Edinburgh, 1984.
- [14] S. A. Cameron *A Study of the Clash Detection Problem in Robotics* IEEE Int. Conf. Robotics and Automation 1985, St. Louis, 488-493, 1985.
- [15] S. A. Cameron *Efficient Intersection Tests for Objects Defined Constructively* International Journal of Robotics Research, 8 (1), 3-25, 1989.
- [16] J. F. Canny *The Complexity of Robot Motion Planning* MIT Press, 1988.
- [17] C. A. G. Cary *BUILD User's Guide* C. A. D. Group Document No. 102, Cambridge University Computer Laboratory, 1979.
- [18] S. Coquillart *A Control-Point-Based Sweeping Technique* IEEE Computer Graphics and its Applications, 7 (11), 36-45, 1987.
- [19] J. H. Davenport, Y. Siret, E. Tournier *Computer Algebra* Academic Press, 1988.
- [20] A. de Pennington, M. S. Bloor, M. A. Balila *Geometric Modelling: A Contribution Towards Intelligent Robots* Proc. 13th International Symposium on Industrial Robots, Chicago, 1983.
- [21] J. Edwards *The Differential Calculus (3rd. Edition)* Macmillan & Co., 1896.
- [22] L. P. Eisenhart *Differential Geometry* Ginn and Company, Boston, 1909.
- [23] I. D. Faux, M. J. Pratt *Computational Geometry for Design and Manufacture* Ellis Horwood, 1979.
- [24] P. K. Ghosh, S. P. Mudur *The Brush-Trajectory Approach to Figure Specification: Some Algebraic Solutions* ACM Transactions on Graphics 3 (2), 110-134, 1984.
- [25] J. G. Griffiths *A Surface Display Algorithm* Computer Aided Design, 10 (1) 65-73, 1979.
- [26] J. Heintz, M. Sieveking *Absolute Primality of Polynomials is Decidable in Random Polynomial Time in the Number of Variables* Proc. ICALP 81, Springer Lecture Notes in Computer Science 115, 16-28, 1981.

- [27] E. Kaltofen *Fast Parallel Irreducibility Testing* J. Symbolic Computation, 1, 57-67, 1985.
- [28] F. Klok *Two Moving Coordinate Frames For Sweeping Along a 3D Trajectory* Computer Aided Geometric Design, 3, 217-229, 1986.
- [29] E. Kreyszig *Differential Geometry* Oxford University Press, 1959.
- [30] J. Levin *A Parametric Algorithm for Drawing Pictures of Solid Objects Composed of Quadric Surfaces* Communications of the ACM, 19 (10) 555-563, 1976.
- [31] R. R. Martin *Aspects of the Interference Problem* Proc. 4th Anglo-Hungarian Seminar on Computer Aided Design, Computer and Automation Institute, Hungarian Academy of Sciences, Budapest, 1985.
- [32] R. R. Martin *Geometric Reasoning for Computer Aided Design* Artificial Intelligence in Design, Ed. D. T. Pham, IFS Publications, 1989.
- [33] C. A. Neff *Decomposing Algebraic Sets using Gröbner bases* Computer Aided geometric Design, 6 (3), 249-263, 1989.
- [34] A. W. Nutbourne, R. R. Martin *Differential Geometry Applied to Curve and Surface Design Volume 1: Foundations* Ellis Horwood, Chichester, 1988.
- [35] Y. Ohno *A Hidden Line Elimination Method for Curved Surfaces* Computer Aided Design 15 (7) 1983.
- [36] T. W. Sederberg, D. C. Anderson, R. N. Goldman *Implicit Representation of Parametric Curves and Surfaces* Computer Vision, Graphics and Image Processing 28, 72-84, 1984.
- [37] Y. Shiroma, N. Okino, Y. Kakazu *Research on 3-D Geometric Modelling by Sweep Primitives* Proc. CAD '82 Conference, Butterworths, 1982.
- [38] S. Stifter *A Medley of Solutions to the Robot Collision Problem in Two and Three Dimensions* Ph.D. Thesis, Johannes Kepler University of Linz, 1988.
- [39] J. J. Stoker *Differential Geometry* Wiley-Interscience, 1969.
- [40] D. L. Vossler *Sweep to CSG Conversion Using Pattern Recognition Techniques* IEEE Computer Graphics and Applications, 5 (8) 61-68, 1985.
- [41] R. J. Walker *Algebraic Curves* Princeton University Press, 1950.
- [42] A. F. Wallis, A. Bowyer, J. H. Davenport, P. S. Milne, J. Padget *The Use of Symbolic Computation in Geometric Modelling* Mathematics of Surfaces III, Ed. D. C. Handscomb, Oxford University Press, 1989.

- [43] W. P. Wang, K. K. Wang *Real-Time Verification of Multiaxis NC Programs with Raster Graphics* Proc. IEEE Int. Conf. on Robotics and Automation, San Francisco, April 1986.
- [44] W. P. Wang, K. K. Wang *Geometric Modelling for Swept Volume of Moving Solids* IEEE Computer Graphics and Applications, 6 (12) 8-17, 1986.
- [45] C. E. Weatherburn *Differential Geometry of Three Dimensions* Cambridge University Press, 1927.
- [46] K. Weiler *Topological Structures for Geometric Modelling* Ph. D. Thesis, Rensselaer Polytechnic Institute, 1986.
- [47] R. Weiss *BE VISION, A Package of IBM 7090 FORTRAN Programs to Draw Orthographic Views of Combinations of Plane and Quadric Surfaces* Journal of the ACM, 13 (2) 194-204, 1966.
- [48] J. R. Woodwark *Blends in Geometric Modelling* The Mathematics of Surfaces II, Ed. R. R. Martin, Oxford University Press, 1987.