Appeared in The Visual Computer, (1994) 10:266-276.

# Computational Methods for Evaluating Swept Object Boundaries

*Laxmi Parida, S.P. Mudur*

National Centre for Software Technology, Gulmohar Cross Road No.9, Juhu, Bombay 400 049, India.

*Abstract*

*Many systems support the design of 2D regions and 3D volumes by sweeping a generator contour over a spine. Well established and widely used rendering systems accept high level boundary descriptions of regions and solids but not in the contour-spine form. Hence the swept object design specification must be such as to lend itself to efficient evaluation of a compact and high level boundary description. In this paper, we identify a sufficiently general class of swept objects and classify the sweep rules. We also present computational methods for directly evaluating the boundary representation which implicitly simulate the sweep process. The boundary is generated as a set of piecewise curves/surfaces in a compact form suitable for direct input to rendering systems like PostScript(R) in 2D and Renderman(R) in 3D.*

**Keywords**: Swept objects, sweep rules, boundary evaluation, volume modelling, region modelling, body modelling, pen-stroke modelling.

## 1. Introduction

Pen-stroke modelling in 2D[1,2] and body modelling in 3D[3] are typical examples of spine based modelling. Designing such regions/volumes as a spine with a generator contour sweeping over it has the advantages of not only providing a more natural way of modelling but also of effectively parameterising the essential geometric features of the model. For example, various arms may have the same spine and generator contour with different sweeping rules to indicate differing 'fatness' and shapes of the arm.

With advances in hardware, there are a number of systems which accept high level geometric descriptions and render them directly on paper or on the screen. For 2D we have page description languages which accept filled regions defined by their boundaries. Lines, conics and parametric cubics can be used for describing the boundary. For 3D we have

workstations with special rendering processors which also accept solids defined by their boundaries and some even support boolean operations. Once again, polygons, quadric surfaces, bi-cubics and NURBS may be used to describe the boundary surfaces of the solids. For example, PostScript[4] accepts regions bounded by lines and Bézier curves and also strokes of uniform thickness, but not a pen stroke specification. Similarly, Renderman[5] accepts quadrics, polyhedral solids and NURBS but not swept surfaces or swept solids. It, therefore, becomes imperative to convert the designs into boundary descriptions that can be accepted by these systems which are becoming *de facto* standards. Any modelling system based on the sweep object design technique will have to generate a description of the boundary of the solid modelled which can then be passed on to the rendering processor.

This paper primarily addresses the problem of computing a compact high level boundary description of the swept object which implicitly simulates the sweep operation on the spine

## 2. Other Related Work

The notion of swept objects has been extensively used in the fields of Computer Aided Design[6,7] and Computer Vision[8] where it goes by the name of generalised cylinders. The emphasis is primarily on geometrically modelling the filled region or solid.

There have been some attempts to specifically address the rendering of these models using techniques such as ray tracing[9]. However we are not aware of any work which deals with the extraction of high level boundary description of a swept object.

Yet another area of work has been surface reconstruction from cross-sectional data, like for example an organ construction from various sections[10,11]. Here, a fairly large number of closely spaced cross-sections are input to give a volume model of the object. The reconstruction process, generally, tries to minimise surface area during the volume construction.

## 3. The Swept Object Design Specification

In this paper, we shall consider the class of swept objects in which the generator contour is a linear segment in 2D and a planar contour in 3D.

Figure 1 shows typical swept object models in 2D and 3D.

A precise specification of the swept object must include the following:

1. **The Specification of the Spine**:

   The spine may be specified as linear, piecewise linear or curved. If curved, then the curve may be specified using analytic functions or by points to be interpolated. If interpolated, then the interpolation constraints must also be considered.

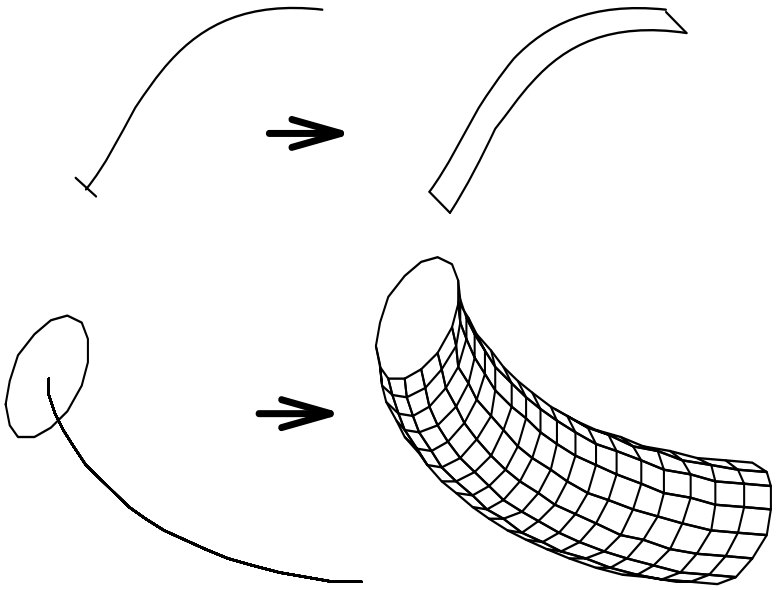

Figure 1. Typical swept objects in 2D and 3D.

2. **The Specification of the Generator Contour**

   The contour may be specified parametrically — in 2D a straight line given by length and orientation. In 3D a circle, ellipse, or any other closed curve given by geometric attributes such as centre, radius and orientation of the plane. Alternatively, it may be specified using analytic equations, functions or points to be interpolated linearly or smoothly.

3. **The Specification of the Sweep Rule**

   This could be specified in one of two ways. One would be to specify contour transformation functions that vary the contour as it is moved along the spine. For example, the radius of a circular contour could be transformed continuously along the spine. Another method would be to define the contours at distinct points along the spine to be interpolated and leave the sweep rule to be suitably inferred.

## 4. Swept Objects and their Boundary Evaluation

The more straight forward but brute-force approach of simulating the sweeping process is to explicitly place the appropriate transformed/interpolated contours at closely spaced points on the spine and then to treat it as a piecewise linear/planar definition of the surface. This has the obvious disadvantages of not only being verbose but also causing the introduction of geometric discontinuity at higher scales. Figure 2 shows an example of a collection of regions defined as swept objects and the simulation of the sweeping process. We present a method that facilitates the computation of a compact boundary for the swept

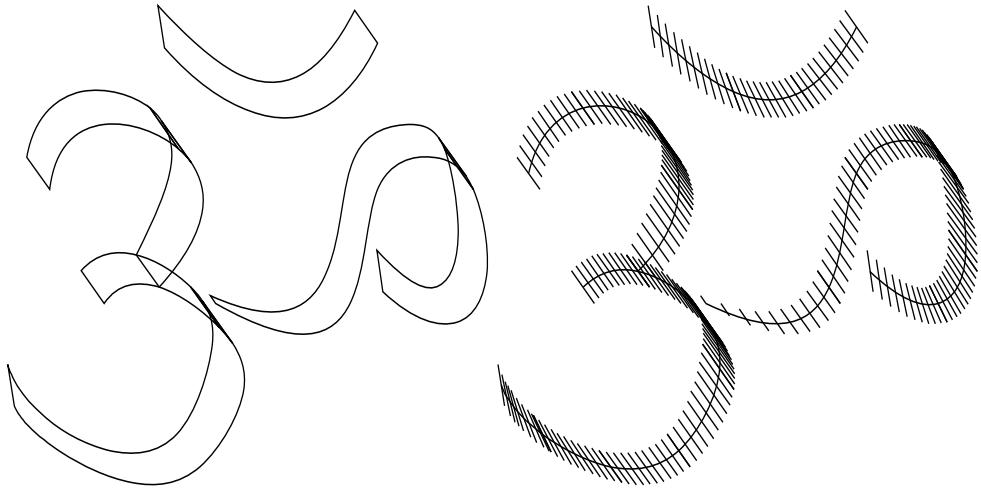

Figure 2. The swept object and the simulation of the sweeping process.

object and implicitly simulates the sweeping of the contour over the spine.

In our work we have assumed that the spine is defined by interpolating a sequence of points. The design problem addressed in the rest of this paper can be specified as follows: Obtain the boundary of a swept object model given:

- A sequence of points $P_0, P_1, ..., P_n$ which have to be smoothly interpolated to form the spine curve.
- A parameterised description of the contour[1] specified explicitly or implicitly at each of the points defining the spine.

### 4.1. Spine Definition

The spine is specified by a finite set of ordered points, $\mathbf{P}_1, \mathbf{P}_2, \ldots, \mathbf{P}_n$ with contours specified at each of these points. Any acceptable interpolation scheme may be used; however it must satisfy the following:

- each point on the spine must be definable as a *linear combination* of the given data points i.e., if $\mathbf{S}(u)$ is a point on the spine then there exist $\alpha_{ui}, i = 0 \ldots n$, such that

$$\mathbf{S}(u) = \Sigma_{i=0}^{n} \alpha_{ui} \mathbf{P}_i.$$

- points may be inserted efficiently into an interpolated set of points and coefficients reevaluated for an identical shape [2] i.e., if $m$ points, $\mathbf{Q}_1, \mathbf{Q}_2, \ldots, \mathbf{Q}_m$ lying on $\mathbf{S}(u)$

[1] In 2D the contour considered is a line segment given by its length, orientation and anchor point with the spine curve, whereas in 3D it is a planar curve suitably parameterised — we have considered circles, ellipses and smoothly interpolated piecewise polynomials.
[2] In the literature this is often termed as knot insertion.

are introduced as data point then there exist $\beta_i$s satisfying the following:

$$\mathbf{S}(u) = \Sigma_{i=0}^{n} \beta_{ui} \mathbf{P}_i + \Sigma_{i=n+1}^{m+n} \beta_{ui} \mathbf{Q}_i.$$

### 4.2. Contour Definition

As mentioned earlier the generator contour in 2D is a line segment. The point at which the spine passes through this contour, is termed the *anchor*. The two end points of the line segment are at distance $d_1$, $d_2$ from the *anchor*. Scaling the contour by $s$ implies that the points are now at $sd_1$ and $sd_2$ from the *anchor*. A variation of this is permitted, called *differential scaling*, where the scaling factors are $s_1$ and $s_2$.

In 3D, the boundary of the generator contour may either be open or closed. Once again this curve may be obtained by interpolating a given sequence of points. In our implementation, circles and ellipses specified by geometric parameters like radius or any other closed curve are also accepted but are internally approximated to piecewise parametric curves. Every point defining the boundary of the closed region, is at a distance, say $d_i$, from the *anchor*. Scaling the contour by $s$, implies that the point is now at distance $sd_i$ from the *anchor* With *differential scaling* the scaling factor is $s_i$. It is important to note that as the contour is regenerated by interpolating these new points, it is not necessarily just a scaled version of the generator contour.

#### 4.2.1. Sweep Rules

We shall classify the sweep rule variations considered in the paper into three classes.

**Class A**

Type A.1 *Translation Invariance* : The generator contour is swept over the spine without any transformation to itself.

Type A.2 *Symmetrical Scaling*: The contour is scaled continuously as it is swept over the spine. It is scaled by equal amounts about the anchor.

Type A.3 *Differential Scaling*: Different amounts of scaling to the distance between the end-points and the anchor is permitted.

**Class B**

Type B.1 *Normal Direction Invariance*: The contour is always normal to the spine i.e., the contour adjusts its orientation as it sweeps over the spine.

Type B.2 *Normal Direction Invariance with Scaling*: The contour apart from being normal to the spine curve can also be scaled continuously.

**Class C**

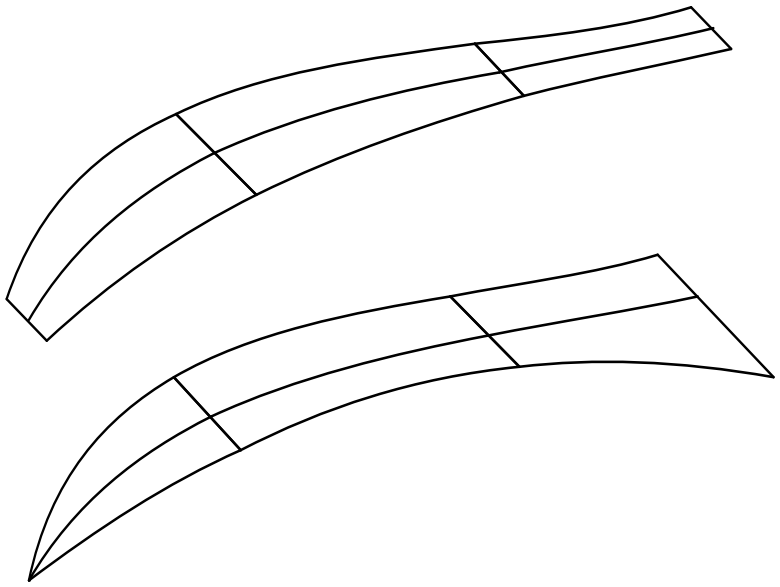

Figure 3. Symmetrical and Differential Scaling of the generator contour (Class A).

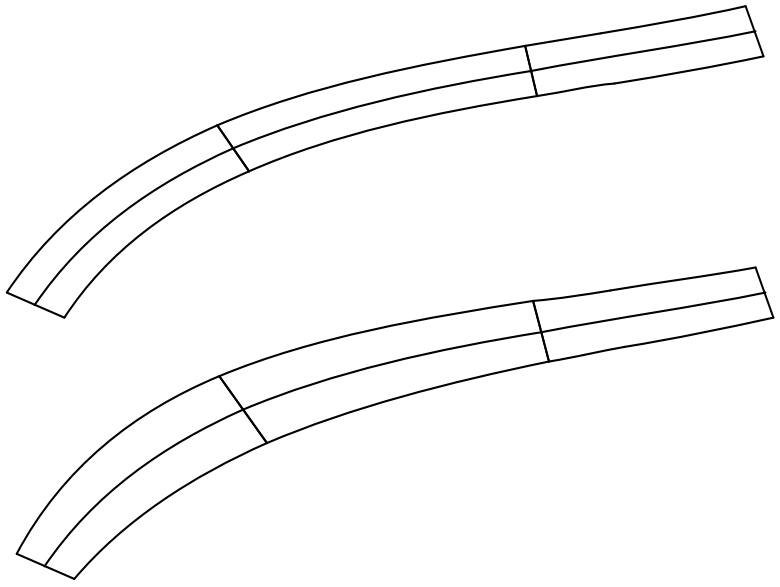

Figure 4. Normal Direction Invariance with scaling of the contour (Class B).

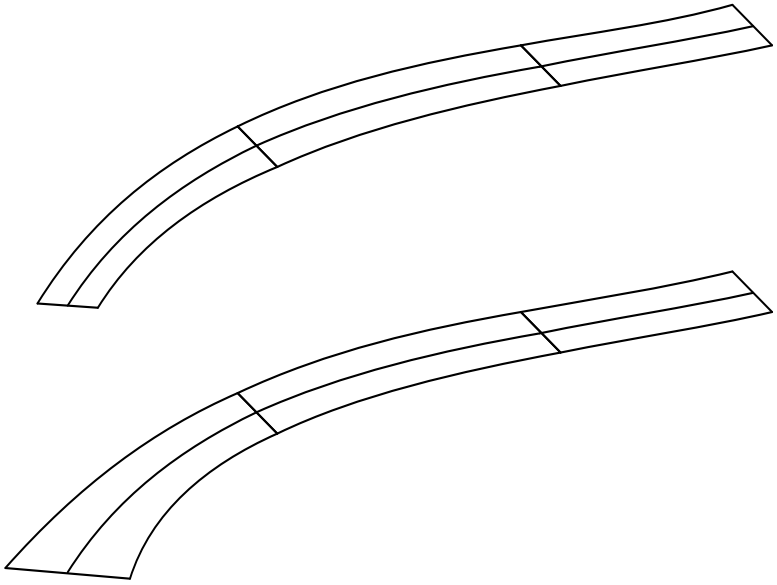

Figure 5. Rotation and Scaling of the contour at the leftmost point of the spine (Class C).

Type C.1 *Rotation*: The contour rotates continuously, as it sweeps over the spine.

Type C.2 *Scaling and Rotation*: The contour rotates and scales continuously as it sweeps over the spine.

(See Figure 3, Figure 4 and Figure 5.)

### 4.3. Boundary Tracing

In order to understand the boundary computation task let us consider a few examples of swept objects. We shall use only 2D examples as they are easier to illustrate and understand. The underlying concepts can be easily extended to 3D. Consider for example the 2D swept object of Figure 1. For this model the sweep rule is that the generator contour is swept over the spine without scaling or rotating. The boundary of this object consists of four edges: the line segments at the two end points of the spine and the two traces each of the two end points of the line segment. Intuitively it appears therefore a boundary description can be generated by first computing the curves/surfaces 'traced' by the boundary points of the contour and then computing the 'caps' at both ends of the spine. While this works for the example of Figure 1 it will fail for a small change in the shape of the spine as shown in the example of Figure 6. As can be seen the boundary description of the object of Figure 6 must include more than just the caps and the end point traces. Another fact to be observed is that the traces may intersect amongst themselves or with the caps.

The strategy that we have adopted is to carry out a minimal segmentation of the region/volume so that the boundary of each segment is well defined by just the caps and

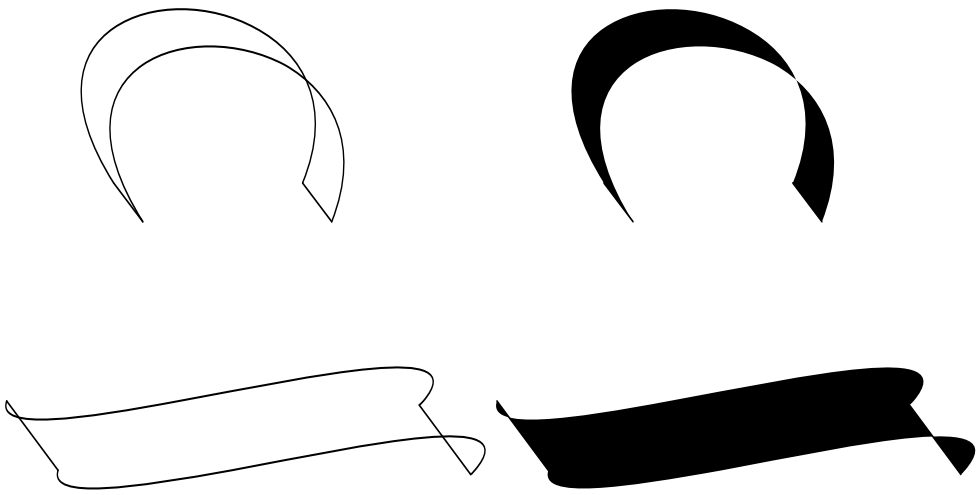

Figure 6. Outlines and the corresponding region obtained by tracing the end points of the contour in 2D.

the traces of some points on the contour. In this context, we state some definitions and theorems below. Further details and proofs of the theorems are presented in [18].

**Definition 1**
*Extraordinary Point*: Let $\mathbf{S}(u)$, $u_0 \leq u \leq u_1$, be the spine curve and let $\mathbf{C}_u$ denote the contour at $\mathbf{S}(u)$. $\mathbf{S}(u_e)$, $u_0 < u_e < u_1$, is an *extraordinary point* if there exists $\delta > 0$ such that $\forall 0 < u_\delta < \delta$, the contours $\mathbf{C}_{u-u_\delta}$ and $\mathbf{C}_{u+u_\delta}$ are in the same half-plane/half-space defined by $\mathbf{C}_u$.

**Definition 2**
*Dominant Point*: The two end points of a contour in a 2D swept object and all the points on the contour of a 3D swept object are defined as *dominant points* on the contour.

**Theorem 1**: Let $\mathbf{S}(u)$, $u_0 \leq u \leq u_1$, define the spine and let $\mathbf{C}_u$ be the contour at $\mathbf{S}(u)$. A point $\mathbf{P}$ can belong to the boundary of the swept object if

- $\mathbf{P} \in \mathbf{C}_{u_0}$ or $\mathbf{P} \in \mathbf{C}_{u_1}$.
- $\mathbf{P} \in \mathbf{C}_{u_e}$ $u_0 < u_e < u_1$, where $\mathbf{S}(u_e)$ is an *extraordinary point.*
- $\mathbf{P}$ is a dominant point of the contour.

Further, no other point is a boundary point.

**Theorem 2**: The spine of a swept object with normal direction invariance (Class B) has no *extraordinary point.*

Three important points need to be noted:

1. The trace of the *dominant points* on the contour contributes to the boundary of the swept object.

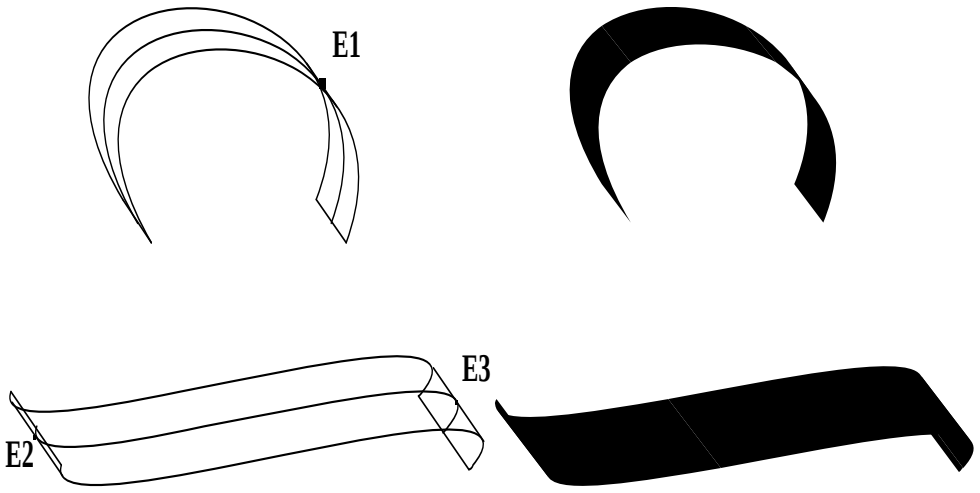


Figure 7. *Extraordinary points* marked on the spine and regions denoted by the swept model.

2. To compute the boundary description of the object the spine must be segmented at the *extraordinary points*. These are the points where the contour is tangential to the spine. Figure 7 shows the the marking of the *extraordinary points* and the region denoted by the swept model.

3. The actual shape of the swept region/volume is completely determined by the interpretation given to the sweep rule particularly in the case where the contour variation has to be inferred by say, interpolating a discrete number of contours specified at different positions in the spine. This can be viewed as the method used for 'fleshing' of the spine and is discussed in greater detail below.

### 4.4. 'Fleshing' of the spine

Any 'fleshing' method must define the region/volume reasonably well, i.e., should be a 'fair' representation of the contour being swept over the spine. A description of the boundary must therefore ideally satisfy the following constraints:

1. Compactness: The number of boundary segments is as small as possible.

2. Analytical Complexity: The analytic complexity of the algebraic functions describing the boundary segments is not higher than that of the spine or the contour. For example if the spine is a cubic then the boundary curve is at most a cubic.

3. Parametric Correspondence: The contour at any point on the spine is completely defined by the same parameter value as that of its anchor point on the spine. This is the mathematical equivalent of saying that the sweep process is simulated.

4. Fairness: The contour varies in as fair a manner as possible depending on the fairness of the spine.

In the theorem below, we present a method of 'fleshing' and then show the boundary constraints it satisfies. The fairness constraint is rather involved and is discussed separately.

**Theorem 3**: Let $\mathbf{P}_0, \mathbf{P}_1, \cdots, \mathbf{P}_n$ be the points on the spine at which the contours are defined. Let an interpolant of the points satisfying the *interpolant constraint* be given by

$$\mathbf{S}(u) = \Sigma_{i=0}^{n} \alpha_{ui} \mathbf{P}_i$$

Let $\mathbf{Q}_0, \mathbf{Q}_1, \cdots, \mathbf{Q}_n$ be the *dominant points* on the contour at $\mathbf{P}_0, \mathbf{P}_1, \cdots, \mathbf{P}_n$ respectively. Let

$$\delta\mathbf{P}_i = \mathbf{Q}_i - \mathbf{P}_i, i = 0, 1, \cdots n.$$

($\delta\mathbf{P}_i$'s are called the *displacement points.*)
If the boundary curve is defined as $\mathbf{B}(u) = \mathbf{S}(u) + \delta\mathbf{I}(u)$ where $\delta\mathbf{I}(u) = \Sigma_{i=0}^{n} \alpha_{ui} \delta\mathbf{P}_i$ then, the boundary curve satisfies the following constraints :

- compactness,
- analytic complexity and
- parametric correspondence.

**Proof**:
**Compactness and Analytic complexity**: It can be seen from the formulation that the boundary is as compact as the spine and the analytic complexity is that of the spine.

Consider the $k^{th}$ derivative for the spine curve -

$$\frac{\delta^k \mathbf{S}(u)}{\delta u} = \Sigma_{i=0}^{n} \frac{\delta^k}{\delta u} \alpha_{ui} \mathbf{P}_i$$

For the boundary curve

$$\frac{\delta^k \mathbf{B}(u)}{\delta u} = \Sigma_{i=0}^{n} \frac{\delta^k}{\delta u} \alpha_{ui} (\mathbf{P}_i + \delta\mathbf{P}_i)$$

Thus if the spine has $k^{th}$ order continuity, then so has the boundary curve.

**Parametric Correspondence**: The formulation as shown in the theorem for the boundary evaluation is a curve traced out on sweeping a point on the contour. This constraint requires that any point taken from such traced curve corresponding to a parameter value say $u_p$, must lie on the contour at $u_p$ of the spine.

*Class A*: If there are $n$ lines parallel to one another then a linear combination of $n$ points one from each of the line lies in a line parallel to them. Thus in 2D, as each contour at the data points is a line segment, it follows that $\mathbf{B}_1(u), \mathbf{B}_2(u)$ and $\mathbf{S}(u)$ lie on a straight line segment.

Analogously, if there are $n$ planes parallel to one another then a linear combination of $n$ points one from each of the plane lies in a plane parallel to them. Thus in 3D, as each contour at the data points is planar, it follows that $\mathbf{B}_1(u), \mathbf{B}_2(u), \cdots$ lie in a plane and $\mathbf{S}(u)$ also lies in this plane.

*Class B & Class C*: If there are $n$ lines in 2D (or planes in 3D) which are rotated and scaled versions of one another, then a linear combination of $n$ points one from each of these lines is also a rotated and scaled version. If we treat the *anchor point* as the point about which the scaling and rotation occur, then $\mathbf{B}_1(u), \mathbf{B}_2(u)$ and $\mathbf{S}(u)$ lie on a straight line segment in 2D and $\mathbf{B}_1(u), \mathbf{B}_2(u), \cdots$ and $\mathbf{S}(u)$ lie in a plane in 3D. $\triangleleft$

### 4.4.1. Fairness Constraint

For any interpolant a *fairness* criterion is difficult to define and quantify. In practice, different interpolants have been used for applications depending on the kind of need and user demands. As there is no precise mathematical definition for a *fair* curve, except may be for geometric conditions like position, first order, second order continuity etc., the fairness of the swept model too depends strongly on the fairness of the underlying spine curve.

We have used the formulation presented in this paper for an interpolant that does a piecewise cubic fit and also for an interpolating B-spline. It has worked well in both the cases. We believe that this method would work reasonably well for any other interpolant satisfying the *interpolant constraint*. Below we discuss the fairness constraints on a case by case basis.

**Scaling of the Contour** : In this we expect, the in between contour to scale but not rotate.

If points $\mathbf{P}_1$, $\mathbf{P}_2$, ..., $\mathbf{P}_n$ are linear in 2D, then if the interpolant satisfies the *interpolant constraint* then the interpolant is also linear. Similarly in 3D, if points $\mathbf{P}_1$, $\mathbf{P}_2$, ..., $\mathbf{P}_n$ are planar in 3D, then the interpolant is also planar.

Note that the *displacement points* at each data point, lie on a line in 2D and lie on a plane in 3D. By the above argument, the *displacement point* for every in between contour also lies on the same line in 2D and the same plane in 3D. Thus there is no rotation of the in between contour. Hence in the swept model, where the contour is only scaled, it does not

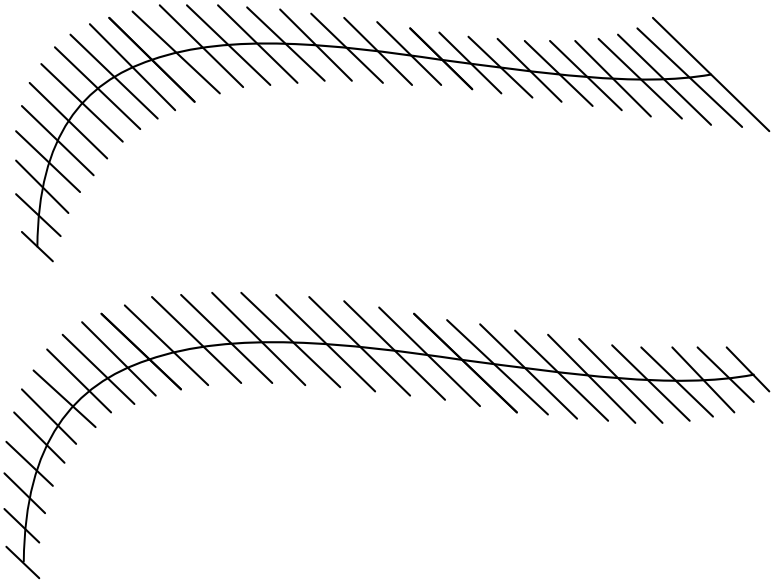

Figure 8. The contours shown for symmetrical and differential scaling.

rotate. (See Figure 8).

**Rotation of the Contour** : In this we expect, the the contour to rotate without scaling. Let us, for the sake of argument, include two more conditions as the *interpolant constraints*

- in 2D, if the points lie on a circle, the interpolating curve must be a circle
- in 3D, if the points lie on a sphere, the interpolating curve must lie on the sphere

The *displacement* points for a rotated, non-scaled contour lie on a circle in 2D. If the interpolant fits a circle when the data points lie on a circle, the in between contour rotates as expected between two defined contours. In 3D, the *displacement* points lie on a sphere, and the interpolant must fit a curve on this sphere. This ensures that the contour rotates without any scaling. Further, if the interpolant is constrained to be a geodesic of the *displacement* points then the contour rotates with minimum undulations between any two defined contours. (See Figure 9).

*Formulation of the Exact Boundary for Normal Invariance Sweep Rules*:
*The 2D Case*:
Given a sequence of points lying on a circle if the interpolant has got to be exactly a circle, then trigonometric or rational functions need to be used. A trigonometric formulation is as follows:

$x = l\cos(\theta), y = l\sin(\theta), \theta_1 \leq \theta \leq \theta_2$

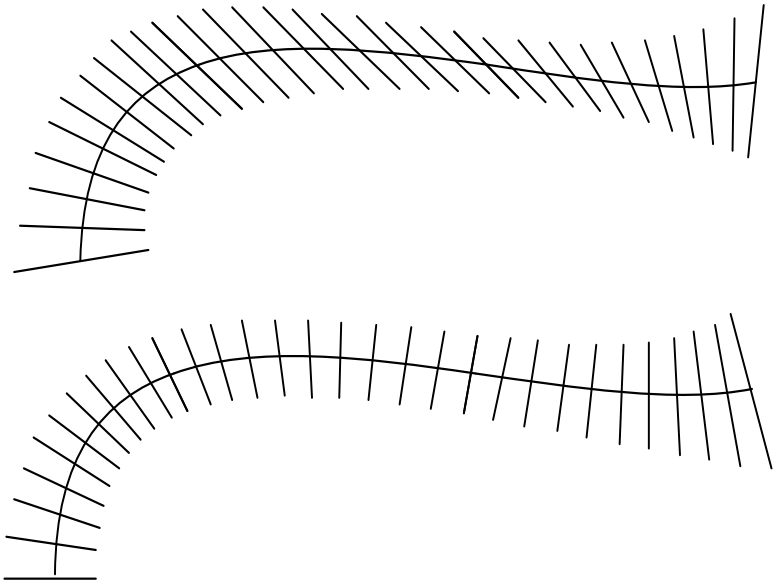

Figure 9. The contours shown for Class B and Class C sweep rules.

Let us look at the case of the normal direction invariance in 2D. Here the contour rotates and at every point is normal to the spine curve. If the spine curve is given by $(S_x(u), S_y(u))$ with a rotating contour, then the additional constraint gives rise to the following :

$$l\cos(\theta)S'_x(u) + l\sin(\theta)S'_y(u) = 0$$

Hence,

$$\theta = \tan^{-1}(-S'_x(u), S'_y(u)) = G(u)$$

Thus as $u$ varies from $u_0$ to $u_1$ in the spine curve, the contour variation function is as

$$(l\cos(G(u)), l\sin(G(u)))$$

*The 3D Case*:
For a given sequence of points in 3D lying on a sphere the interpolant too must lie on this sphere. Actually, it is the shortest path between the two points on the sphere, i.e., it is a geodesic on the sphere.

Let $\mathbf{P}_1$, $\mathbf{P}_2$ be the two points on the sphere which need to be connected by the shortest geodesic between them. Let $\mathbf{P_1} \times \mathbf{P_2} = \mathbf{N}$. Also let $\mathbf{N}$ be a unit vector. Then the shortest geodesic between the two points is given by

$$(G_x(\theta), G_y(\theta), G_z(\theta))$$

where

$$G_x(\theta) = l(\frac{N_x N_z}{\sqrt{1-N_z^2}}\cos(\theta) + \frac{N_y N_z}{\sqrt{1-N_z^2}})\ sin(\theta))$$

$$G_y(\theta) = l(\frac{-N_y}{\sqrt{1-N_z^2}}\cos(\theta) + \frac{N_x}{\sqrt{1-N_z^2}})\ sin(\theta))$$

$$G_z(\theta) = l(N_x\cos(\theta) + N_y\cos(\theta))$$

$$\theta_1 \leq \theta \leq \theta_2, \mathbf{G}(\theta_1) = \mathbf{P}_1, \mathbf{G}(\theta_2) = \mathbf{P}_2.$$

Let us look at the normal direction invariance in 3D. In addition to the interpolant being the shortest geodesic between the points it must also be normal to the spine. It is identical to the 2D case. The function for $\theta$ is

$$\theta = \tan^{-1}\frac{\frac{S'_y N_y - N_x N_z S'_x}{\sqrt{1-N_z^2}} - N_x S'_z}{\frac{S'_y N_x - N_y N_z S'_x}{\sqrt{1-N_z^2}} + N_y S'_z}$$

The function is as for the rotating contour.

The exact functions as can be seen are very complex. These involve trigonometric, square root and other such functions and as such are of much higher analytical complexity than the spine. As it is not possible to accept the exact functions, we have chosen an interpolant scheme that approximates these as closely as possible.

It may be noted here that the Class B.1 variation of contours amounts to computation of offset curves. There have been various attempts at computing offsets reported in literature [12,13,14]. It is well established that the computation of exact offset of lower order polynomial curves gives curves of higher order in the rational form with square root of the polynomial in the denominator. Also, a non self-intersecting spine may give rise to self-intersecting offset curve.

We have been able to tackle these problems to a reasonable extent by adopting the technique of stepwise refinement to handle the contour variations in the above mentioned classes. This implies that if a stretch of the spine does not give satisfactory boundaries, then the spine is broken into two or more pieces by suitably inserting points and the contour is then recomputed. This process is continued until satisfactory results are obtained. In practice, we have observed that the decision is best left to the user designing the model rather than an automatic refinement which may often result in unnecessarily verbose boundary descriptions.

Appeared in The Visual Computer, (1994) 10:266-276.

## 5. Algorithms for Swept Object Boundary Evaluation

We use the fleshing definition of Theorem 3 to define the swept object. This formulation results in fairly good designs and has high fidelity to the spine curve.

### 5.1. Algorithms for 2D Boundary Evaluation

The algorithm in 2D, has two main subtasks :

1. spine curve evaluation and
2. computation of the boundary curves of the region.

Let $\mathbf{P}_0, \mathbf{P}_1, \ldots, \mathbf{P}_n$ be the given data points. Let us assume, without loss of generality, that the spine is an open curve. The contour is defined by a length $h$ and inclination $\theta$. Represent the contour, in the contour coordinate system, as the line segment with the end points $(h\sin(\theta), h\cos(\theta))$ and $(-h\sin(\theta), -h\cos(\theta))$ where the anchor is $(0,0)$. Let $h\cos(\theta) = dx, h\sin(\theta) = dy$. Variations to the contour will appropriately change $dx, dy$. As already seen we need only to trace the two end points of the line segment. Let $\mathbf{l}_0, \mathbf{l}_1, \ldots, \mathbf{l}_n$ denote the first end point of the contour at the data points and $\mathbf{r}_0, \mathbf{r}_1, \ldots, \mathbf{r}_n$ denote the other, where $\mathbf{l}_i = (dx_i, dy_i), \mathbf{r}_i = (-dx_i, -dy_i)$.

#### 5.1.1. Spine Curve Evaluation

Step 1 Obtain the interpolant spline

$$\mathbf{S}(u) = \Sigma_{i=0}^{n} \alpha_{ui} \mathbf{P}_i$$

#### 5.1.2. Evaluating the Boundary of the region

Step 2 Obtain the interpolant spline of the difference vectors

$$\mathbf{I}_l(u) = \Sigma_{i=0}^{n} \alpha_{ui} \mathbf{l}_i$$

$$\mathbf{I}_r(u) = \Sigma_{i=0}^{n} \alpha_{ui} \mathbf{r}_i$$

Step 3 Let

$$\mathbf{B}_l(u) = \mathbf{S}(u) + \mathbf{I}_l(u)$$

$$\mathbf{B}_r(u) = \mathbf{S}(u) + \mathbf{I}_r(u)$$

Step 4 For Class B and Class C, the spine curve may have to be split with new points inserted as mentioned in the last section to fair the interpolated contours. If the point insertion is done, the corresponding factors $\beta$s are recomputed and step 3 is repeated.
Further, for Class A and Class C the spine needs to be split at the *extraordinary points*, say $u_e$. By the parametric correspondence design constraint of theorem 3, the boundary curves $\mathbf{B}_l$ and $\mathbf{B}_r$ also need to be split only at $u_e$.

Step 5 The bounding curves are obtained for each subregion by capping with the contours at the end points of the spine segment.

### 5.2. Algorithms for 3D Boundary Evaluation

The algorithms in 3D are extensions to the 2D case. Basically, the description is now a collection of faces instead of edges, with faces being defined by surface patches.

The subtasks are:

1. spine curve evaluation,
2. individual contour evaluation and
3. computation of the boundary surfaces of the volume.

#### 5.2.1. Spine Curve Evaluation

Step 1 Obtain the interpolant spline, $\mathbf{S}(u)$, and the linear coefficients $\alpha_i$, which define any point on the curve as

$$\mathbf{S}(u) = \Sigma\alpha_i(u)\mathbf{P}_i$$

#### 5.2.2. Individual Contour Evaluation

Step 2 Obtain the interpolant for the points of the planar contour.

Step 3 For every defined point on the spine, apply the desired variation in the contour and translate the contour so that the spine passes through the anchor point. This operation results in the mesh of the points $\mathbf{P}_{ij}$, $i = 1, 2, \ldots, n$, $j = 1, 2, \ldots, m$, where $m$ is the number of points on the contour and $n$ is the number of points on the spine. Obtain the linear coefficients $\beta_j$, which define any point on the contour as

$$\mathbf{C}_i = \Sigma\beta_j\mathbf{P}_i j$$

These coefficients will be used later while obtaining the tensor product definition of the bounding surface.

### 5.2.3. Evaluation of the Boundary of the volume

Step 4 For Class B and Class C, the spine curve may have to be split with new points inserted as mentioned in the last section to fair the interpolated contours. If the point insertion is done, the corresponding factors $\beta$s are recomputed and step 3 is repeated.
Further, for Class A and Class C the spine needs to be split at the *extraordinary points*, say $u_e$. By the parametric correspondence design constraint of theorem 3, the boundary surfaces also need to be split only at $u_e$.

Step 5 The tensor product bi-polynomial surface that results is derived as follows: Consider one longitudinal isoparametric curve, i.e., a constant $u = u_0$ curve on the surface, assuming the order of the spine to be $\mu$, with $\mathbf{B}$ as the coefficients,

$$\sum_{j=0}^{\mu} \mathbf{B}_j^s(v)(\alpha_{j1}\mathbf{P}_{01} + \alpha_{j2}\mathbf{P}_{02} + \ldots + \alpha_{jm}\mathbf{P}_{0m}$$

$$\Rightarrow \sum_{j=0}^{\mu} \mathbf{B}_j^s(v)(\sum_m \alpha_{jm}\mathbf{P}_{0m})$$

More generally the longitudinal isoparametric curve at $u$ is given by

$$\sum_{i=0}^{\nu} \mathbf{B}_i^c(u)(\sum_n \beta_{in}\mathbf{P}_{nm})$$

Using this we obtain,

$$\sum_{j=0}^{\mu} \mathbf{B}_j^s(v)(\sum_m \alpha_{jm} \sum_{i=0}^{\nu} \mathbf{B}_i^c(u) \sum_n \beta_{in}\mathbf{P}_{nm})$$

Step 6 The bounding surfaces are obtained for each subvolume by capping with the contours at the end points of the spine segment.

In our implementation, we have used composite Bézier cubic curves with $G^1$ continuity for the spine and contour boundary curves. Hence we get composite Bézier bicubic patches with $G^1$ continuity for the boundary surface.

### 5.3. Size of the Boundary Definition

The boundary of the swept object is defined as the union of well defined subregions/subvolumes. with each having non-intersecting boundaries. (See Figure 10). Thus the size of the boundary definition depends on the number of such segments. Below, we give the upper limit on the number of these segments.

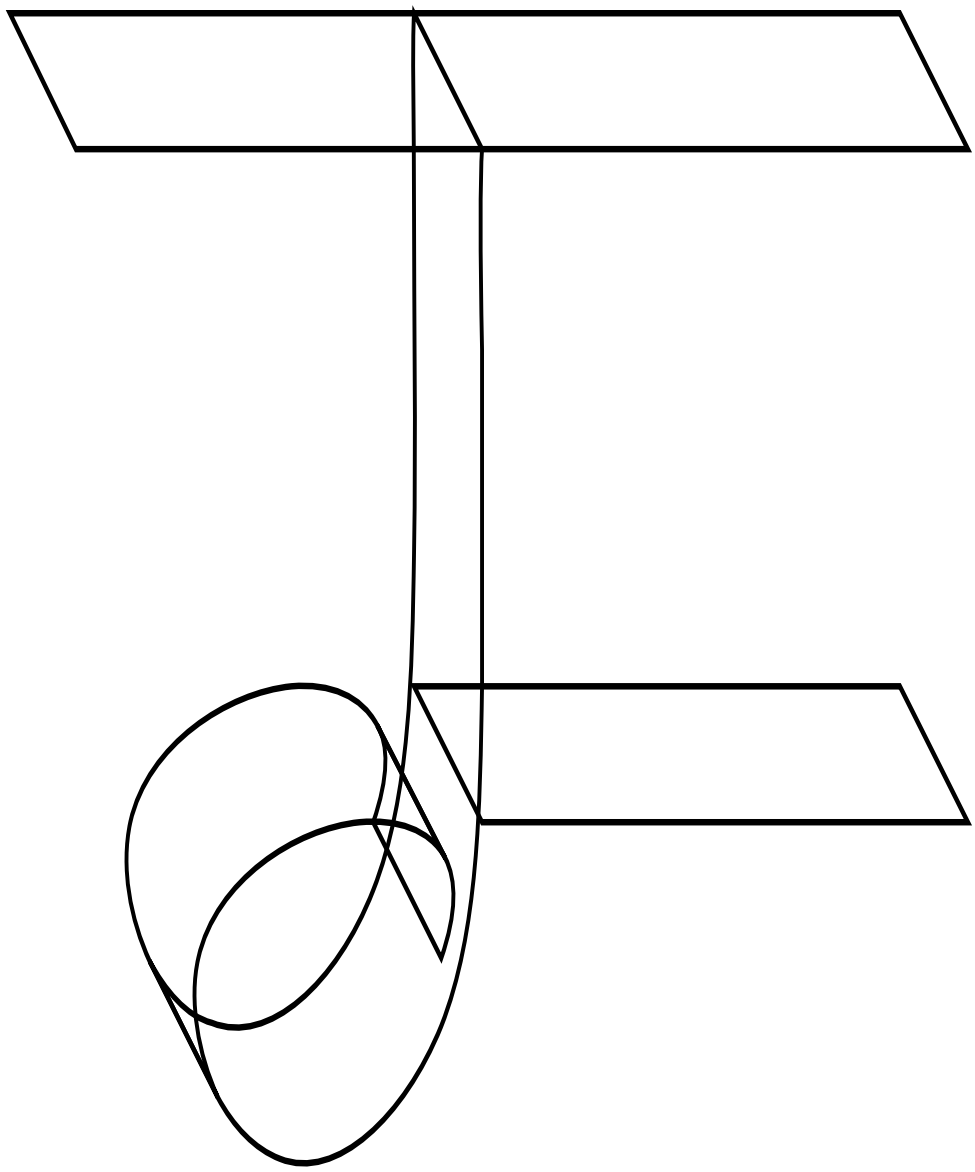

Figure 10. The swept object defined as a union of segments.

Note that the maximum number of *extraordinary points* on a spine of degree $m$ is $m-1$. Thus the total number of segments in a swept object with contour variation of Class A is at the most $(m-1)n$ where $m$ is the degree of the spine and $n$ is the number of pieces defining the spine.

The number of segments in a swept object with contour variation of Class B and Class C is $Mn$ where $M$ depends on the maximum number of segments per cubic (done in the stepwise refinement process) and $n$ is the number of pieces in the spine.

The above give an upper bound for the number of elements in the boundary description of the swept object. It is in most cases much smaller in number. A swept object with rotated contour on the other hand has generally more elements. In our implementation, during the use of the system, we have seen that in general this number is reasonably small.

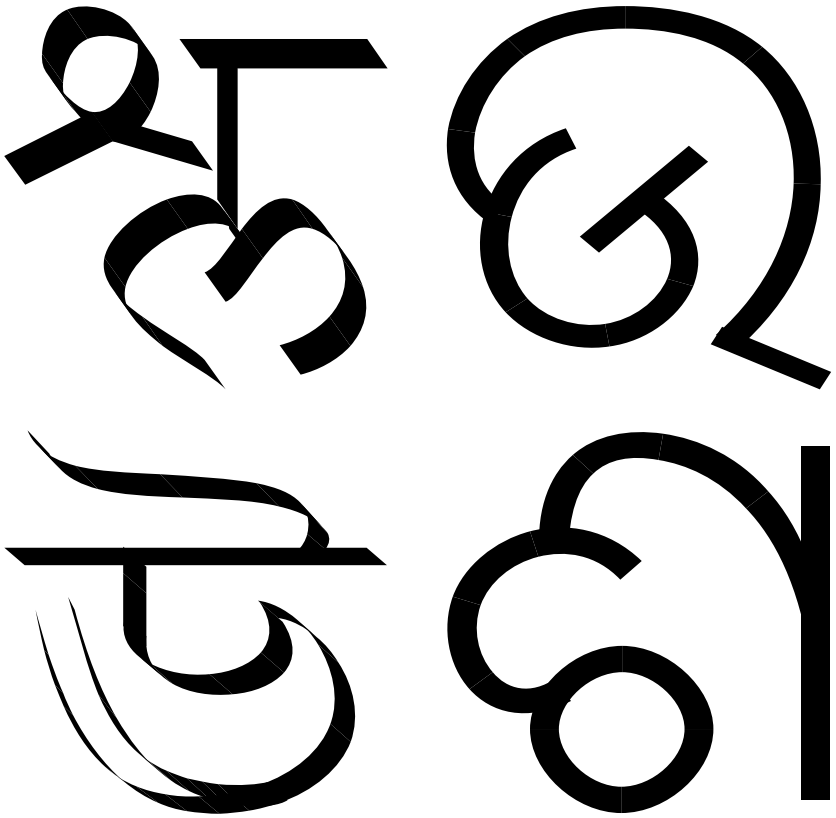

Figure 11. Samples of letterforms from Indian scripts designed using the swept object definition.

## 6. Implementation

The interpolation technique presented in [17] has been used in the implementation and gives composite Bézier cubics/bicubics with $G^1$ continuity.

The algorithms presented for 2D have been implemented and incorporated into an interactive font design tool, Vinyas [15, 16] which generates analytic font definitions in the Postscript language. This tool has been extensively used by various calligraphers to produce fonts in different Indian scripts (See Figure 11). The algorithms for 3D have also been implemented and tested but not as a part of any interactive design system. Their use has therefore been rather limited. Figure 12 shows some attempt at modelling an arm

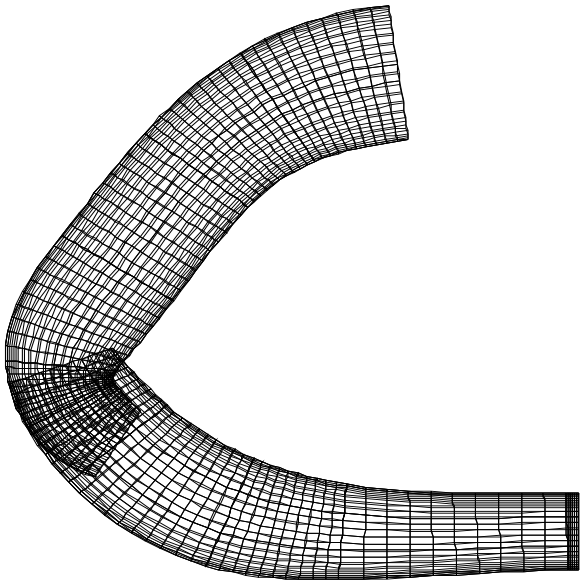

Figure 12. An arm modelled using the sweeping technique.

using the 3D implementation.

## 7. Acknowledgements

The authors are grateful to Dr. S. Ramani, Director, NCST, for his support and encouragement. Prof. R. K. Joshi and his Visual Design students of the Industrial Design Centre of the Indian Institute of Technology, Bombay, used the swept object technique to produce typefaces in various Indian languages, some of which are shown in Figure 11. The authors would also like to thank Ms. Mrudul Bapat for assistance in implementing the algorithms for 3D and generating the 3D picture for the paper.

## 8. References


1. D. Knuth, *Mathematical Typography*, Bulletin of the American Mathematical Society 1 (1979).

2. P.K. Ghosh, S. P. Mudur, *The brush-trajectory approach to figure specification: some algebraic solutions*, ACM Tr. on Computer Graphics 3 (1984).

3. N. Magnenat-Thalmann, D. Thalmann, *Construction and Animation of a synthetic actress*, Eurographics 88, september 1988.

4. Adobe Systems Incorporated, *Postscript Language: Reference Manual*, Addison-Wesley, Massachussetts (1985).

5. Steve Upstill (Pixar), *The Renderman Companion*, Addison-Wesley, New York (1990).

6. Jae-Woo Ahn, Myung-Soo Kim, Soon-Bum Lim, *Approximate General Sweep Boundary of 2D Object*, Visual Computing, T. L. Kunni (Ed.), Springer-Verlag, Tokyo, pp 547-566, 1992.

7. C. D. Woodward, *Cross-sectional design of B-spline surfaces*, Computers and Graphics, vol (11) no 2, 1987.

8. David Marr, *Vision: Computational Investigation into Human Representation and processing of visual information*, W. H. Freeman, San Francisco, (1982).

9. J. J. Wijk, *Ray Tracing objects defined by sweeping planar cubic splines*, ACM Tr. on Computer Graphics 3 (1984) 223-237.

10. J. D. Boissonnat, *Shape reconstruction from planar cross sections*, Compute Vision Graph Image Processing 41(1).

11. Y. Shinagawa and T. L. Kunni, *The homotopy model: a generalized model for smooth surface generation from cross sectional data*, The Visual Computer, (1991)7.

12. J. Hoschek and N. Wissel, *Optimal Approximate conversion of spline curves and spline approximation of offset curves*, Computer Aided Design, Vol 20, No 8, (October 1988).

13. R. T. Farouki, *The approximation of non-degenerate offset surfaces*, Computer Aided Geometric Design, Vol 3, No 1, (May 1986).

14. G. Elber and E. Cohen, *Error bound variable distance offset operator for free form curves and surfaces*, IJ of Computational Geometry and Applications, Vol 1, No 1, (March 1991).

15. Laxmi Parida, *Vinyas - User Reference Manual*, NCST Internal Report (March, 1990).

16. Laxmi Parida, *Vinyas: An Interactive Calligraphic Type Design System*, (under publication).

17. Hirokai Chiyokura, *Solid Modelling with Designbase: Theory and Implementation*, Addison-Wesley, England (1988).

18. Laxmi Parida, S. P. Mudur, *Computational Methods for Evaluating the Boundary of Cross Sectional Designs*, NCST Internal Report, 1991.