Chapter 5: Vein Attempts

. . . shooting eagerly aloft in youth's joyful prime, and towering serene and satisfied through countless years of calm and storm, the greatest of plants and all but immortal.
(John Muir, writing of the Giant Sequoia)

5.1 Summary

This chapter concerns the representation of cylindrical and branching-cylindrical forms, usually of circular cross-section. We present a comparison of parametric and implicit representations, emphasizing bulge-free implicit blends. The properties of convolution in general and as applied to branching structures are examined in depth, and related to algebraic blends. Several new techniques are developed, and suggestions for future research are given. In this chapter we lay the mathematical foundations for the hand described in chapter 6 and the leaf described in chapter 7.

5.2 Generalized Cylinders

As we have noted, [Wainwright 1988] argues for the generality and ubiquity of the cylinder as a structural element in plants and animals. [Thompson 1961] also finds the cylinder to be a common element in natural forms. The *generalized cylinder* was developed in [Agin 1972] for use in computer vision as a representation for objects in natural scenes. It differs from the straight cylinder in that a space curve may act as the central axis and the cross-section may vary in size or shape. Because of this variability, the generalized cylinder can represent a wide variety of forms. The generalized cylinder is one type of *swept surface* (or *sweep*). Other swept surfaces sweep a solid, rather than a cross-section [Klok 1986].

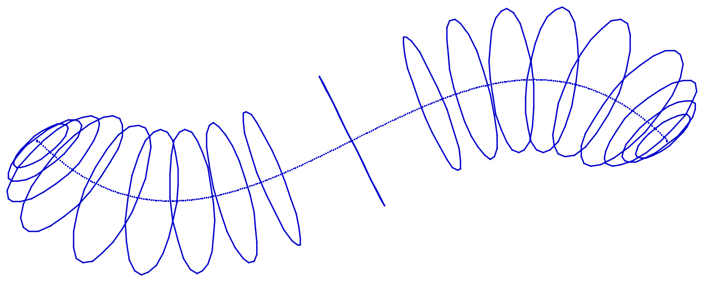

Figure 5.1 A Generalized Cylinder

When several cylinders are combined, we speak of their axes as a skeleton. In the following illustration we see that a curved skeleton improves organic appearance by eliminating the angularity of the straight skeleton. To some, however, the curved skeleton may appear too curved. When tapered cross-sections are added, however, the object is seen to have mass and the skeleton is seen as dissipating stress, rather than being unnecessarily curved. The skeleton below was generated by a jointly authored program [Bloomenthal and Glassner 1986].

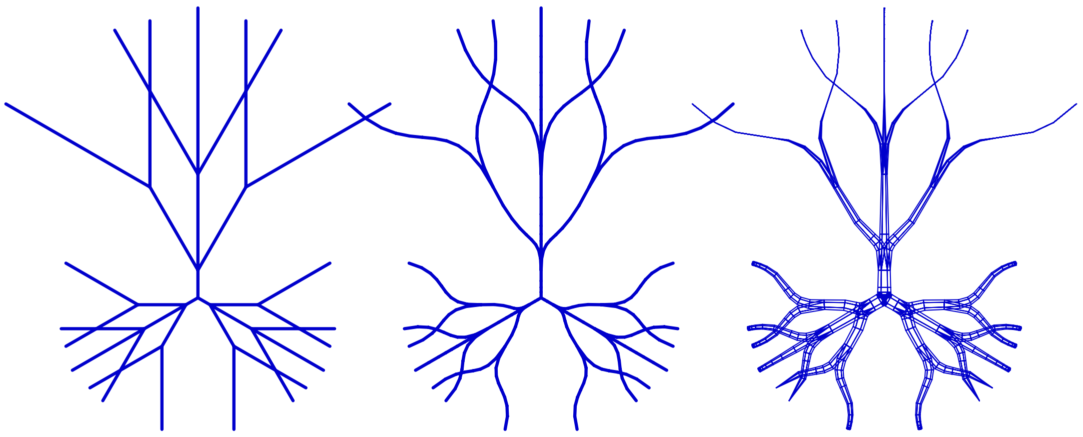

Figure 5.2 An Organic Progression: Straight, Curved, and Tapered

In [Bloomenthal 1985] the generalized cylinder was employed as a geometric model for tree limbs. In this chapter we review such use and demonstrate extensions that allow the generalized cylinder to represent particular branching patterns and shapes. We describe and compare methods that fall into the general categories of parametric and implicit. We first present parametric methods.

5.3 Parametric Methods

A *parametric generalized cylinder* consists of a set of connected, individually computed cross-sections. Before discussing cross-sections, however, we consider the reference frames that are required to stabilize them. We conclude with a discussion of branching, parametric generalized cylinders.

5.3.1 Reference Frames

To represent a parametric generalized cylinder, a series of individual cross-sections is established along the axis of the cylinder, as illustrated in the following figure.

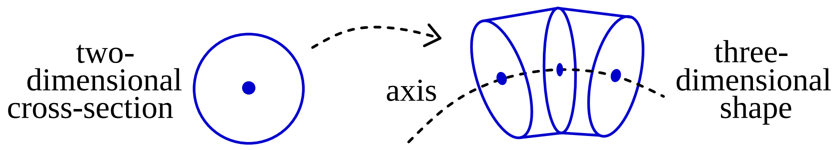

Figure 5.3 Transforming a Cross-Section

It is important that these cross-sections be stable with respect to each other. In the figure below, for example, the cross-sections twist with respect to their neighbors. Although the object retains a cylindrical shape, any texture associated with the surface will twist, an artifact usually considered undesirable.

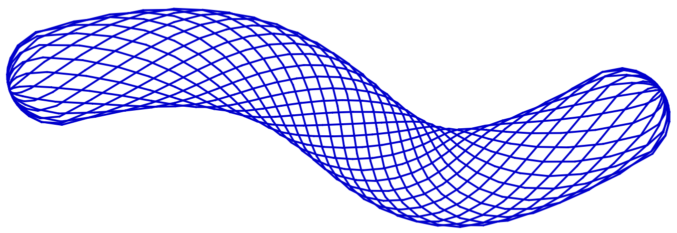

Figure 5.4 A Twisted Generalized Cylinder

Should the twist be severe, as in the next illustration, the cylinder will constrict. To avoid a distorted shape such as this, each cross-section must be aligned with respect to its neighbors. Usually this alignment is provided by a *reference frame*, a point and three orthonormal vectors that define position and orientation along the axis of the cylinder. We now examine several methods to construct reference frames along a space curve.

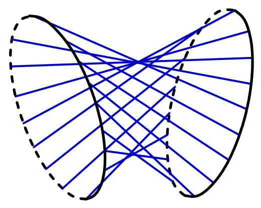

Figure 5.5 Severely Twisted Cross-Sections

The reference frame shown below is due to Frenet; it consists of a position ***p*** and three unit length vectors [Faux and Pratt 1979]:

v, the *velocity* of the curve,

n, the *principal normal*, usually defined in the direction of curvature, and

b, the *binormal*, usually computed as $\mathbf{v} \times \mathbf{n}$.

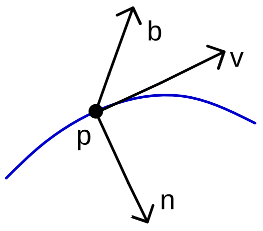

Figure 5.6 The Frenet Frame

The space curve typical of constructions found in computer graphics is parametrically defined by a cubic polynomial. Given a curve *C* defined by the three-dimensional vector coefficients $\mathbf{c}_1$, $\mathbf{c}_2$, $\mathbf{c}_3$, and $\mathbf{c}_4$, the point ***p*** at parametric value *t* along the curve is given as:

(5.1) $\mathbf{p}(t) = \mathbf{c}_1 t^3 + \mathbf{c}_2 t^2 + \mathbf{c}_3 t + \mathbf{c}_4$, where *t* typically ranges from 0 to 1.

The velocity (or *tangent*) is the derivative of position:

(5.2) $\boldsymbol{v}(t) = 3\boldsymbol{c}_1t^2+2\boldsymbol{c}_2t+\boldsymbol{c}_3$.

Curvature may be computed as:

(5.3) $\boldsymbol{k}(t) = \boldsymbol{v} \times \boldsymbol{a} \times \boldsymbol{v} \,/\, ||\boldsymbol{v}||^4$, where $\boldsymbol{a}(t) = 6\boldsymbol{c}_1t+2\boldsymbol{c}_2$.

Detailed descriptions of these vectors and other curve properties may be found in [Bartels *et al*. 1987]. When a space curve is utilized in this dissertation, it is a parametric, cubic curve.

The Frenet frame may be computed analytically at arbitrary points along the curve. It is, however, undefined wherever curvature vanishes, such as at points of inflection or along straight sections. Moreover, the direction of curvature suddenly reverses on either side of an inflection point, causing an extreme twist within a series of Frenet frames. This problem has been examined in [Shani and Ballard 1984], [Klok 1986], and [M. Bloomenthal 1988], where successive, ordered calculation of frames is proposed to minimize the rotation of a frame around the tangent to the curve. As noted in the latter two references, such minimization does not necessarily yield the desired result; in the case of a helix, for example, the Frenet frame appears more useful than a rotation-minimizing frame.

The rotation-minimizing method illustrated below, left, was used to compute reference frames for the tree limbs in [Bloomenthal 1985]. This method presumes an initial frame at t = 0 (the beginning of the curve) and steps along the curve, generating a new frame from the previous frame. In particular, a previous set of vectors $\{\boldsymbol{v}_1, \boldsymbol{n}_1, \boldsymbol{b}_1\}$, is rotated around the axis $\boldsymbol{v}_1 \times \boldsymbol{v}_2$ to yield a new set of vectors $\{\boldsymbol{v}_2, \boldsymbol{n}_2, \boldsymbol{b}_2\}$. An initial frame usually is given by $\{\boldsymbol{v}(0), \boldsymbol{k}(0), \boldsymbol{v}(0) \times \boldsymbol{k}(0)\}$; if curvature is undefined at t = 0, $\boldsymbol{n}$ can be any unit length vector perpendicular to $\boldsymbol{v}(0)$. Successive frames are generated, *in order*, by computing $\boldsymbol{p}$ and $\boldsymbol{v}$ at each new location on the curve and rotating the previous frame such that its $\boldsymbol{v}$ aligns with the new $\boldsymbol{v}$. Additional details are given in [Bloomenthal 1990].

A simpler, 'double-cross' successive method is proposed in [Sloan 1992] in which $\boldsymbol{n}_2 = \boldsymbol{b}_1 \times \boldsymbol{v}_2$ and $\boldsymbol{b}_2 = \boldsymbol{v}_2 \times \boldsymbol{n}_2$, as shown below, right. This is equivalent to rotation-minimization when $\boldsymbol{v}_2$ is in the plane of $\boldsymbol{v}_1$ and $\boldsymbol{b}_1$ or the plane of $\boldsymbol{v}_1$ and $\boldsymbol{n}_1$.

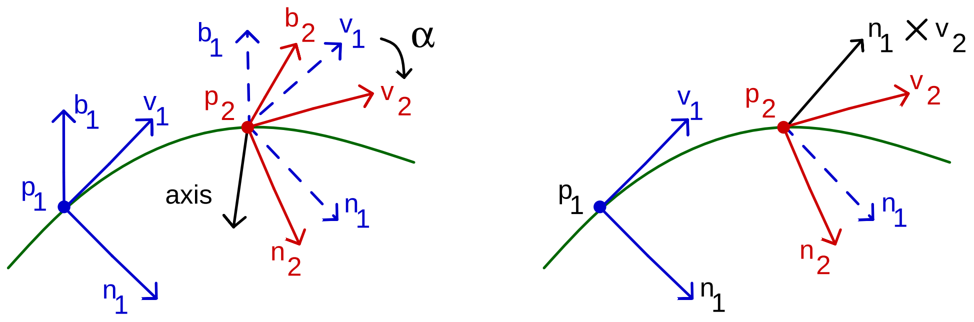

Figure 5.7 Computing a Reference Frame from the Previous Frame

left: minimizing rotation, right: double-cross

The three reference frame methods described above are visually compared below. Successive methods, which are not compatible with the Frenet frame, are immune to curvature degeneracies but do not permit analytical computation at arbitrary *t*. Thus, results depend on the distance between successive frames; when frames are closer, they follow the curve more accurately. It is appropriate to increase the number of frames at points of higher curvature, or to create intermediate reference frames in order to establish accurately a frame at a desired location. The rotation-minimizing frame at the end of a closed curve will not, in general, match that at the beginning; this is further discussed in [Shoemake 1994].

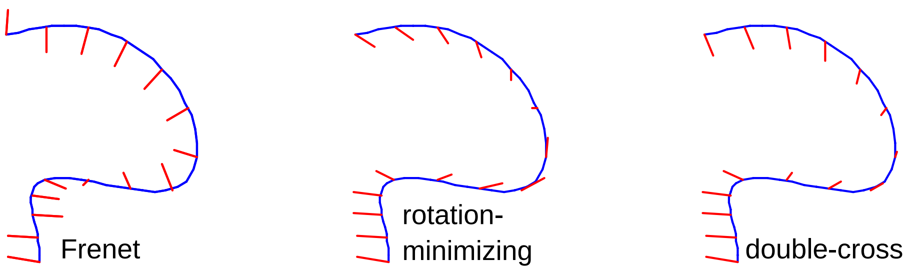

Figure 5.8 Normal Vector along a Non-Planar Curve

5.3.2 Cross-Sections

Although $\boldsymbol{v}$ is used to compute the reference frame, only the point $\boldsymbol{p}$, normal $\boldsymbol{n}$, and binormal $\boldsymbol{b}$ are needed to transform a two-dimensional cross-section into the plane defined by $\boldsymbol{p}$, $\boldsymbol{n}$ and $\boldsymbol{b}$. If $\boldsymbol{q}$ belongs to a two-dimensional cross-section, the corresponding surface point of the generalized cylinder is given by $\boldsymbol{p}+\boldsymbol{q}_x\boldsymbol{n}+\boldsymbol{q}_y\boldsymbol{b}$. This may be represented by a matrix multiplication of $\boldsymbol{q}_h$ (the homogeneous vector $[\boldsymbol{q}_x, \boldsymbol{q}_y, 1]$) with a 3 by 3 matrix whose rows consist of $\boldsymbol{n}$, $\boldsymbol{b}$, and $\boldsymbol{p}$:

$$(5.4) \qquad \boldsymbol{q}_{cylinder} = \boldsymbol{q}_h\,[\boldsymbol{n}\ \boldsymbol{b}\ \boldsymbol{p}]^{\mathrm{T}}.$$

When applied to a series of points that define a two-dimensional cross-section, the transformation yields a cross-section of the cylinder.

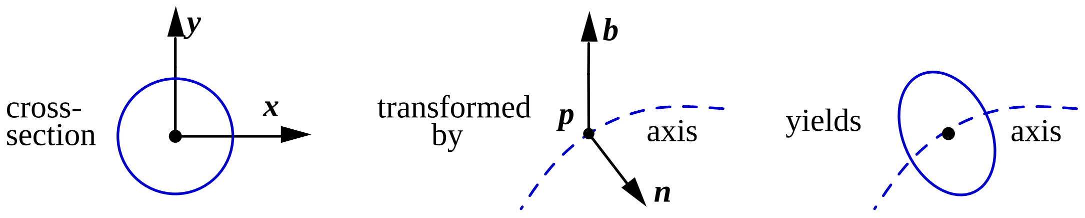

Figure 5.9 Position and Orientation of a Cross-Section

Cross-sections need not be circular, nor need they be similarly shaped; in the example below, they change as they progress along the axis. The number and placement of cross-sections can depend on changes in size or shape of the cross-sections as well as curvature of the axis.

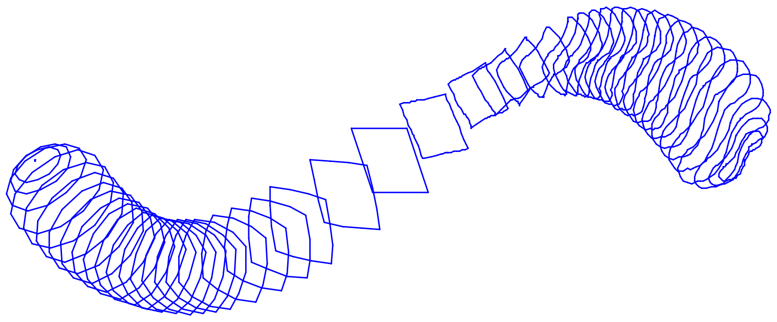

Figure 5.10 Varying Cross-Sections

We obtain a polygonal representation, shown below, by connecting corresponding edges of adjacent cross-sections to form quadrilaterals. Hemi-spherical cylinder ends are created by reducing the cross-sectional radius, r_c:

(5.5) $r_c = (r^2 - max(0, r-d)^2)^{½}$,

where d is distance from the cross-section center to the axis end, as shown below, right. In practice, the unit vectors $\mathbf{n}$ and $\mathbf{b}$ may be scaled by r_c.

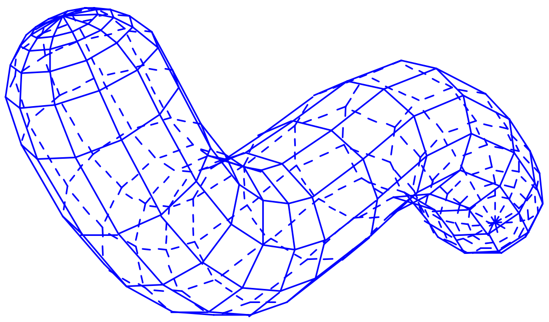
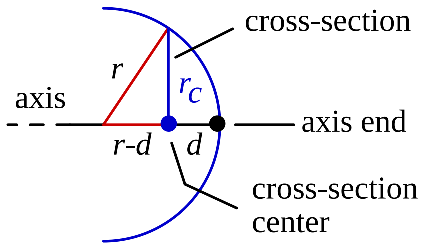

Figure 5.11 A Generalized Cylinder

left: polygonal form, right: calculation of cross-section radius

Although able to produce very general shapes, the parametric generalized cylinder can suffer from interpenetrating cross-sections. This problem is illustrated below.

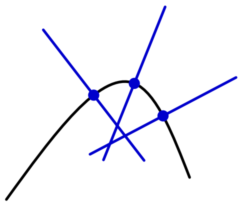

Figure 5.12 Intersecting Cross-Sections *(viewed edge-on)*

5.3.3 Ramification

The use of the generalized cylinder to represent simple organic shapes, such as individual tree limbs, is complicated by the possibility of self-intersection and the need for reference frames along the axis.[1] Nonetheless, the generalized cylinder is relatively simple to construct and yields many usable shapes.

The *ramiform*, that is, the branching or ramification of one cylinder into two or more, is a greater challenge. If we apply the previous techniques to each limb, an invalid polygonal form results. For example, in the figure below, one branch intersects the other. No valid boundary representation can be created from this set of polygons [Requicha 1980]. Also, the individual branches meet with tangent discontinuity, which is uncharacteristic of many natural forms.

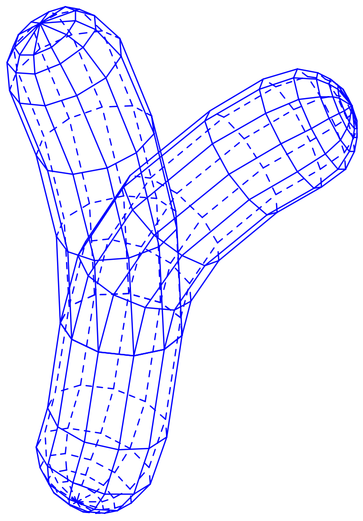

Figure 5.13 Branching Generalized Cylinder

5.4 Parametric Ramification

Methods based on ramification of the cross-section are often unusable or difficult to implement. For example, consider two such methods, illustrated below. At left, a series of contours is created by intersecting the surface with parallel planes; unfortunately, the intersection of surface and plane may be difficult to compute and becomes ill-defined for branches at right angles to the planes. At right, the contours are explicitly broken at a particular point; this method is not easily extended to *n*-way branching.

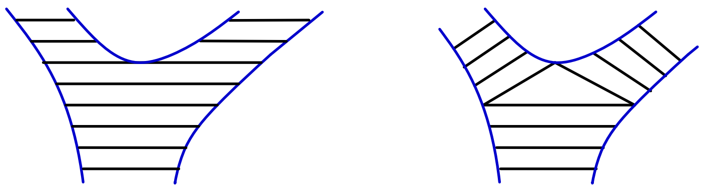

Figure 5.14 Ramification of Cross-Sections

More successful is an *ad hoc* technique given in [Bloomenthal 1985] that creates a *lofted surface*. In the illustration below, numbers refer to the following steps:

1. compute a *disk* at the end of the parent limb, given a reference frame
2. compute disks at the beginning of both child limbs
3. compute *silhouette curves* between parent and child disks
4. compute a *saddle curve* between the child disks
5. compute a *hoop curve* that connects the parent disk to the saddle curve
6. construct *lofting curves* that connect child disks through hoop curve
7. construct lofting lines from parent to child

Lines 6_i and 7_i, shown as white at right, may be converted to polygons.

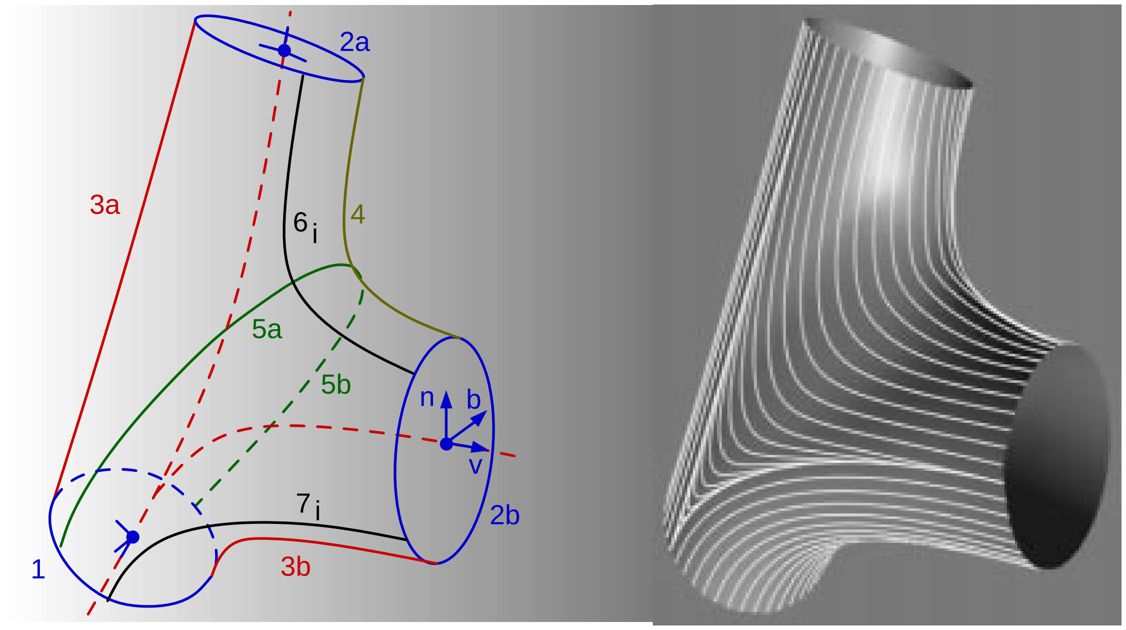

Figure 5.15 The Ramiform as a Lofted Surface

We briefly summarize a number of other non-implicit approaches that smoothly blend component surfaces, such as the branching limbs of a ramiform. The summary loosely follows those in [Rossignac and Requicha 1984] and [Woodwark 1986].

One approach is simply to annotate a non-smooth object without producing an actual geometric description; as such, the object is unsuitable for computer graphics. Alternatively, fillets and rounds can be applied to edges of polygonal models, but this requires significant interaction from the user [Chiyokura and Kimura 1983]. In [Rossignac and Requicha 1984] swept surfaces are suggested as useful in blending, but the ease with which cross sections can be varied in order to produce complex fillets is questioned. We have not seen examples of smooth ramiforms produced by sweeps. *Surface fitting* to scattered points, [Hoppe *et al*. 1992], is a hybrid combination of implicit and parametric techniques. It itself does not produce blends; points on fillets and on rounds must be identified before a satisfactory blend can be achieved.

Free-form (*i.e.*, ‘sculptured’) surfaces, such as parametric patches, permit extensive local control of the geometry of an object. Exercising this local control while maintaining geometric continuity has received considerable attention. In addition to significant user interaction, free-form surfaces may require non-rectangular surface patches to model fillets and rounds [Charrot and Gregory 1984]. Application of the *n-sided patch* to ramiforms appears difficult because the defining control mesh must provide geometric continuity at the patch junctions. Recent minimization techniques relax this requirement [Moreton and Sequin 1993], [Welch and Witkin 1993].

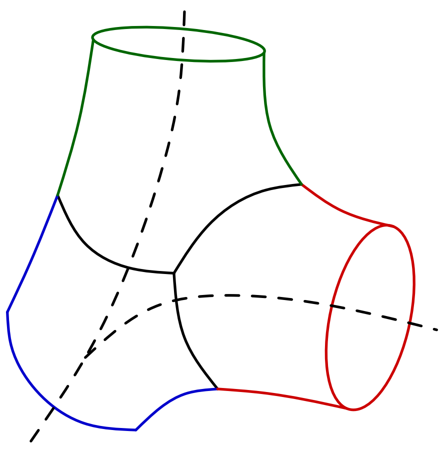

Figure 5.16 Ramiform as a Sculptured Surface

Rather than fit patches to a mesh, a smooth *subdivision surface* can be derived from a mesh [Catmull and Clark 1978], [Doo 1978], [Nasri 1987]. Because it is produced by recursive subdivision, the subdivison surface is a *limit surface*, not a parametric surface. Usually the subdivision surface does not interpolate its control mesh, but recent techniques provide for interpolation [Halstead *et al*. 1993].

Although free-form surfaces and limit surfaces are suitable for the representation of smooth branching forms, they require an intermediate data structure whose complexity depends on the topology of the surface, rather than the simpler topology of the skeleton. For example, the number of patches required for a ramiform is six, whereas the number of curves required is two. Although this complicates the extension of these methods to *n*-ramiforms, these alternative techniques suggest that complex, branching shapes are becoming more accessible to the designer.

This dissertation is devoted to the notion that skeletal design, rather than surface design, is more intuitive and simpler to implement. In particular, we have focused our energies upon implicit techniques and in this and subsequent chapters we will see the degree to which these methods succeed. We begin by investigating techniques for implicitly defining generalized cylinders and *n*-ramiforms.

5.5 Implicit Ramification

We have argued that implicit definitions permit surfaces to be described directly and intuitively from a skeleton, which is simpler than a surface to represent. This facilitates the implicit blend techniques we consider in the next several sections. Two additional advantages are specific to generalized cylinders: *a*) reference frames are not required for circular cross-sections, and, more importantly, *b*) the surface does not intersect itself when the cylinder is tightly curved. The latter advantage is generally true of implicit surfaces; for continuous functions in $\mathfrak{R}^3$, an implicit surface is, with few exceptions, a two-dimensional manifold.[2]

5.5.1 Distance Surfaces

One simple implicit definition for a generalized cylinder uses distance between the point $\boldsymbol{p}$ and the curve (or axis C) with associated radius r:

(5.6) $f(\boldsymbol{p}) = d(\boldsymbol{p}, C)/r-1 = 0$, where d is Euclidean distance.

We choose the form $d(\boldsymbol{p}, C)/r-1$ rather than $d(\boldsymbol{p}, C)-r$ because it normalizes $d(\boldsymbol{p}, C)$ against a ‘radius in isolation.’ Later we will see that this normalization is useful when considering the blend of multiple surfaces. We choose $d(\boldsymbol{p}, C)/r$ rather than $r/d(\boldsymbol{p}, C)$ so that $d(\boldsymbol{p}, C)$ may equal zero (*i.e.*, $\boldsymbol{p}$ lies on C). And finally, we choose $d(\boldsymbol{p}, C)/r-1$ rather than $1-d(\boldsymbol{p}, C)/r$ so that, per solid modeling convention, f is negative inside the volume defined by the surface, and positive outside.

During polygonization, $f(\boldsymbol{p})$ is repeatedly evaluated; therefore, the efficiency of f is important. We avoid the computationally expensive square root in computing Euclidean distance by using the following, equivalent surface definition:

(5.7) $f(\boldsymbol{p}) = d(\boldsymbol{p}, C)^2/r^2-1 = 0$.

The surface described by this function is a *distance surface* [Faux and Pratt 1979], also known as an *offset solid* [Requicha 1983]. The term ‘offset surface’ usually implies distance along a normal; for example, the offset surface to a square is a rectangular solid with sharp edges whereas the distance surface (or offset solid) to a square contains rounded edges. We will use the term ‘distance surface.’

We illustrate distance to a curve by evaluating a 300 by 200 set of points in the plane, with $x \in [-1, 1]$ and $y \in [-\frac{3}{4}, \frac{3}{4}]$. The curve C is restricted to the plane. The intensities shown below, left, represent the distance to the curve, which is highlighted in dashed white. Contours are emphasized, below right, by mapping $d(\boldsymbol{p}, C)$ through a sine function (in practice, this is achieved by modifying the color map of the display).

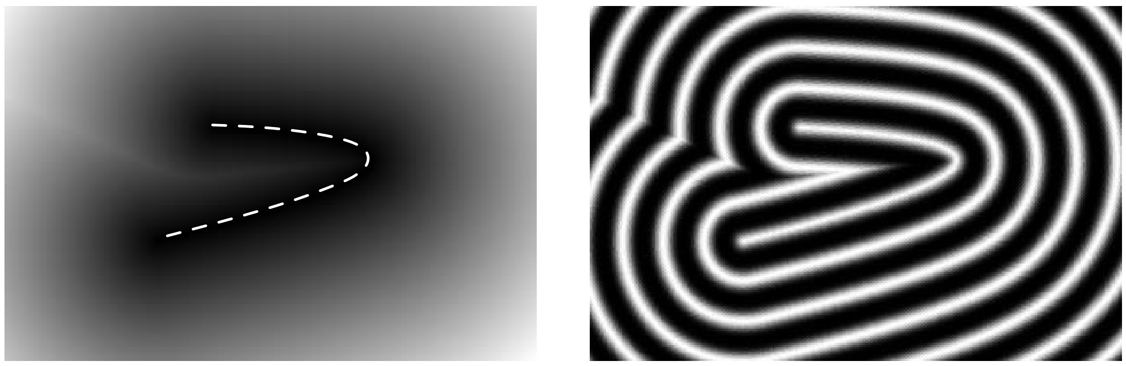

Figure 5.17 Distance to Curve, d(p, C)
left: without contours, right: with contours

We compute the distance between $\boldsymbol{p}$ and C as $\|\boldsymbol{p}-\boldsymbol{q}\|$ where $\boldsymbol{q}$ is the point on C closest to $\boldsymbol{p}$. Determining $\boldsymbol{q}$ requires finding the roots to a 5th degree uni-variate polynomial [Bloomenthal 1989], [Schneider 1990]. Given a point $\boldsymbol{p}$ and a cubic parametric curve defined by the vector coefficients $\boldsymbol{c}_1$, $\boldsymbol{c}_2$, $\boldsymbol{c}_3$, and $\boldsymbol{c}_4$, we find $t \in [0, 1]$ such that $\|\boldsymbol{p}-C(t)\|$ is minimal by solving $d\|\boldsymbol{p}-C(t))\|dt = 0$. With some algebra, it follows that:

$$(5.8)\quad d\|\boldsymbol{p}-C(t)\|^2/dt = 3\boldsymbol{c}_1\boldsymbol{\cdot c}_1t^5+5\boldsymbol{c}_1\boldsymbol{\cdot c}_2t^4+2(2\boldsymbol{c}_1\boldsymbol{\cdot c}_3+\boldsymbol{c}_2\boldsymbol{\cdot c}_2)t^3+ 3(\boldsymbol{c}_1\boldsymbol{\cdot e}+\boldsymbol{c}_2\boldsymbol{\cdot c}_3)t^2+(2\boldsymbol{c}_2\boldsymbol{\cdot e}+\boldsymbol{c}_3\boldsymbol{\cdot c}_3)t+\boldsymbol{c}_3\boldsymbol{\cdot e},$$

where $\boldsymbol{e} = \boldsymbol{c}_4\boldsymbol{\cdot p}$.

When there are multiple roots, such as at t_1 and t_2 shown below, each must be tested to determine the minimum distance from $\boldsymbol{p}$ to C.

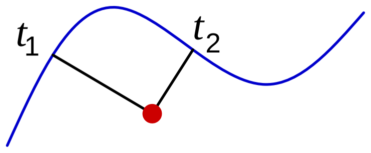

Figure 5.18 Multiple Roots

$d||\boldsymbol{p}-\boldsymbol{q}(t_1)||/dt = d||\boldsymbol{p}-\boldsymbol{q}(t_2)||/dt = 0$, *but* $||\boldsymbol{p}, \boldsymbol{q}(t_2)|| < ||\boldsymbol{p}, \boldsymbol{q}(t_1)||$

5.5.2 Piecewise Approximations

In [Abel 1881] the lack of a closed form solution for roots of a 5th degree polynomial is demonstrated (a translation of his proof appears in [Smith 1959]). As a consequence, numerical methods must be employed, which are computationally slow. Therefore, we now investigate piecewise approximations to the curve, which admit to efficient distance computations.

We can simplify the computational task of equation 5.8 by approximating the axis of the generalized cylinder with a series of line segments or a series of points, as shown below.

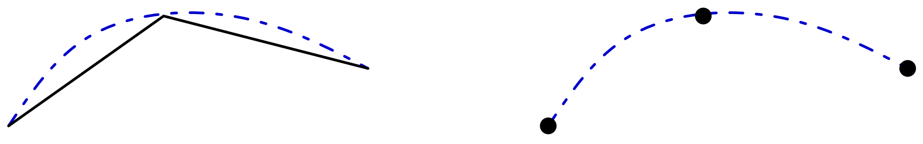

Figure 5.19 Linear Approximation by Segments (left) or Points (right)

Let us first examine the use of points. Given a cylinder with radius r and a skeleton S consisting of n point elements, an implicit definition is:

$$(5.9) \quad f(\boldsymbol{p}) = \min_{i}^{n} (||\boldsymbol{p}-\boldsymbol{point}_i||^2)/r^2-1 = 0.$$

We speak of $\boldsymbol{point}_i$ as an *element* of the skeleton, and $\|\boldsymbol{p}-\boldsymbol{point}_i\|/r$ as the *implicit primitive*$_i$ that contributes to f. Each primitive will be a sphere. In equation 5.9 only the minimally valued primitive (*i.e.*, the primitive of the closest point) is used. In other words, if $\boldsymbol{p}$ is within any individual primitive $volume_i$, then $\|\boldsymbol{p}, \boldsymbol{point}_i\|^2)/r^2 < 1$, $f(\boldsymbol{p}) < 0$, and $\boldsymbol{p}$ is interior to the solid model. Thus, *min* in equation 5.9 corresponds to the *union* of the individual volumes defined by the point primitives.

If the skeleton consists of n line segments, rather than n points, the implicit definition that corresponds with equation 5.9 is:

(5.10) $f(\boldsymbol{p}) = \min\limits_{i}^{n} (d(\boldsymbol{p}, segment_i)^2)/r^2-1 = 0.$

To compute the distance from a point $\boldsymbol{p}$ to a line $\boldsymbol{ab}$ we calculate $\|\boldsymbol{p}-\boldsymbol{q}\|$, where $\boldsymbol{q}$ is the projection of $\boldsymbol{p}$ onto the line, as illustrated below. For distance to the *segment* $\boldsymbol{ab}$, we restrict α to [0, 1].

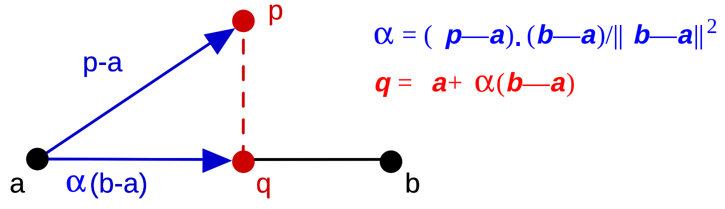
p
p-a
$\alpha = (\boldsymbol{p}-\boldsymbol{a}).(\boldsymbol{b}-\boldsymbol{a})/\|\boldsymbol{b}-\boldsymbol{a}\|^2$
$\boldsymbol{q} = \boldsymbol{a} + \alpha(\boldsymbol{b}-\boldsymbol{a})$
a
α(b-a)
q
b

Figure 5.20 Projecting a Point to a Line

It is efficient to pre-compute the following vectors for each segment:

$\boldsymbol{d}_i = \boldsymbol{b}_i-\boldsymbol{a}_i$, and

$\boldsymbol{e}_i = \boldsymbol{d}_i/\boldsymbol{d}_i\cdot\boldsymbol{d}_i$.

Thus, for a point $\boldsymbol{p}$ and a $segment_i$

(5.11) $\boldsymbol{q}_i = \boldsymbol{a}_i+max(0, min((\boldsymbol{p}-\boldsymbol{a}_i)\cdot\boldsymbol{e}_i, 1))\, \boldsymbol{d}_i$.

As the number of segments or points increases, the linear approximation converges to the curve. In [Ramer 1972] and [Douglas and Peucker 1973] a curve is recursively sub-divided at the point furthest from its approximating line segment. For the tests in the next section, the distance from the midpoint of a segment to the curve controls the subdivision, as shown below. The segment with the greatest deviation is subdivided by creating a new segment endpoint on the curve, half-way along the curve section as measured by arc-length.[3] For example, in the figure below, d_2 is less than d_1; thus, *segment*$_1$ would be subdivided before *segment*$_2$. This process is repeated until the desired number of segments is reached or a maximum allowed deviation is no longer exceeded.

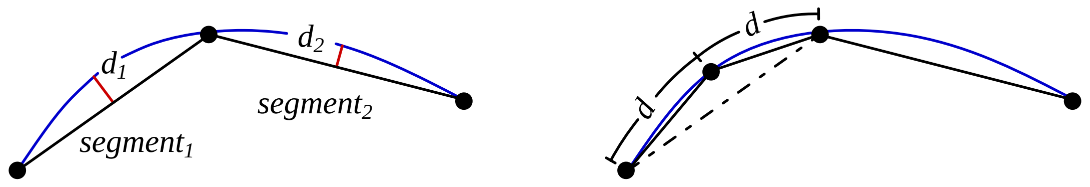

Figure 5.21 Piecewise Linear Approximation

left: measuring curve deviation

right: dividing a segment

5.5.3 Distance Surfaces, Conclusion

Distances to the nearest of 3, 5, and 9 segments and the nearest of 3, 5, and 9 points are shown below for the curve used in section 5.5.1. The segments have been placed according to the subdivision scheme described above; the points have been placed at equal arc-length intervals along the curve. The semi-circular (hemi-spherical in three dimensions) contours around the open segment endpoints are a consequence of the implicit definition.

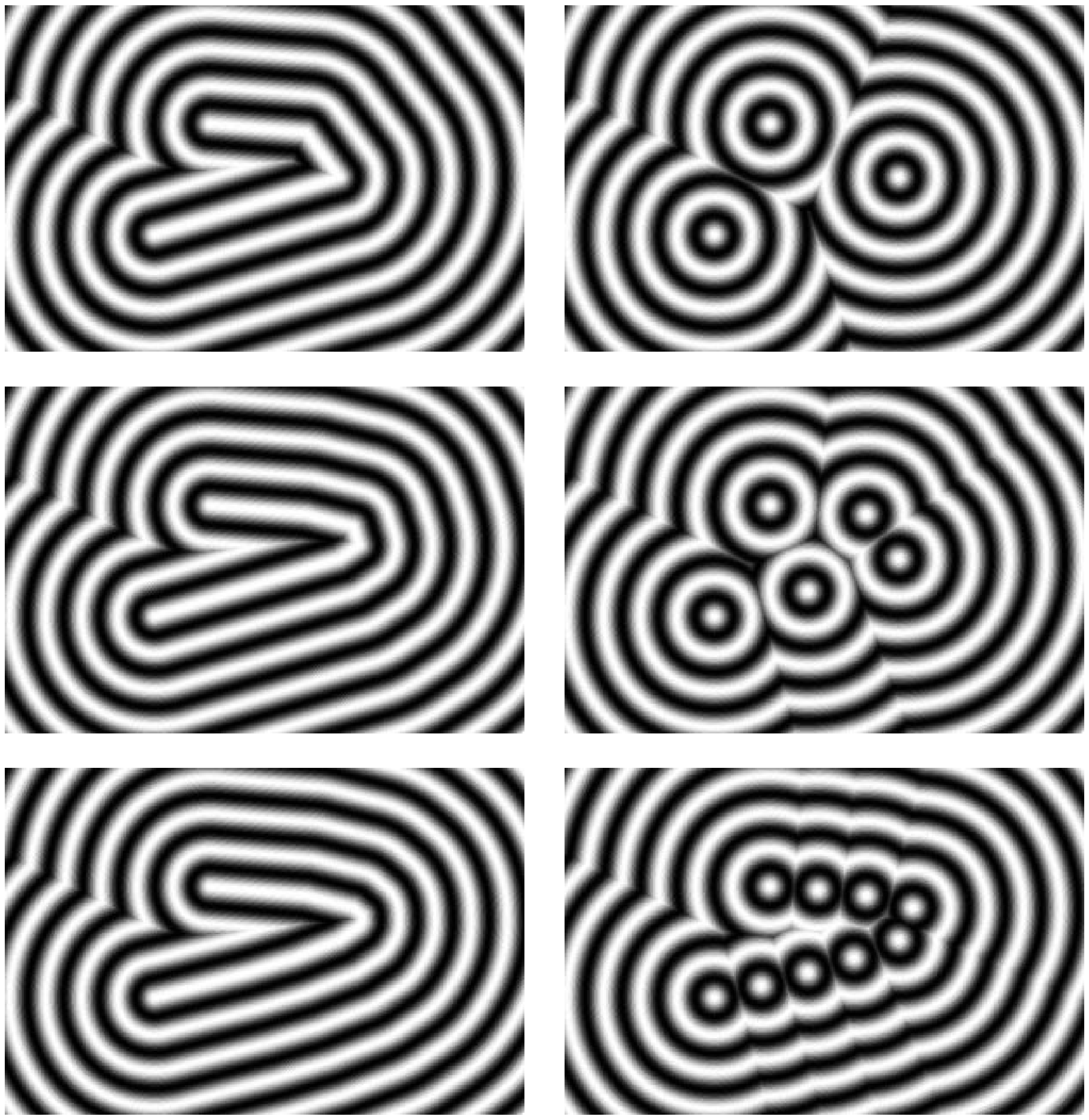

Figure 5.22 Minimum Linear Distance (contours added)

left: 3, 5, and 9 Segments, right: 3, 5, and 9 Points

To compute $\|\boldsymbol{p}-\boldsymbol{point}_i\|^2$ or $\|\boldsymbol{p}-\boldsymbol{q}_i\|^2$ requires 3 subtractions, 2 additions, and 3 multiplications. To compute $\boldsymbol{q}_i$ (equation 5.11), we must also execute 8 additions and 6 multiplications. Thus $d(\boldsymbol{p}, segment_i)^2$ requires 13 additions and 9 multiplications, approximately three times the computation required for

$\|\mathbf{p}-\mathbf{point}_i\|^2$. In the illustration above, the contours for nine points have artifacts that are more objectionable than the artifacts for three segments. Moreover, line segments provide an analytic, parametric representation for the skeleton, which simplifies not only skeletal design and articulation, but the application of surface texture as well. Therefore, we prefer the use of line segments to points, in most cases, and will develop the techniques of the next several sections using segments, not points. We will, however, return to points in sections 5.6.12 and 5.6.14.4.

Distance surfaces are easily computed, requiring only a single comparison for each of the skeletal elements (whether curves, segments, or points) and a single record in memory to store the smallest distance. The price for such simplicity is that the resulting surface encloses the simple union of the component primitive volumes. In other words, each point or segment is a skeletal element that defines a primitive volume, and the resulting composite volume is the union of the component volumes. In this context, *min* in equation 5.10 corresponds to the union operator in *CSG* modeling.

In general, distance surfaces produce rounded[4] surfaces where the skeleton is convex. Where the skeleton is concave, the surface is tangent discontinuous and exhibits a crease. For the purposes of representing smooth organic shapes, the creases along the concave (left) side of the above contours are undesirable. They are comparable to creases exhibited by a parametric generalized cylinder with self-intersections. To eliminate creases, the primitive volumes must form a *blend*, rather than a union.

5.5.4 Blends

The blending of primitives in the context of solid modeling has received considerable study, as shown by the survey in [Woodwark 1986] and a variety of recent work. From these sources we learn that blended surfaces (or *blends*) are used in mechanical design to reduce stress, improve air or water flow, simplify casting, and improve aesthetics. We employ blends because they mimic numerous biological shapes and are compatible with the skeletal design of implicit surfaces.

Implicit blends may be categorized as rolling-ball, volume-bounded, range-controlled, and global [*ibid*.]. The first three produce surfaces that ‘heel’ to parts of individual primitive surfaces when those parts are sufficiently distant from other primitives. The distance required is controlled by auxiliary surfaces in volume-bounded blends [Sabin 1978], [van Wijk 1986], by mutual proximity of the surfaces in range-controlled blends [Hoffmann and Hopcroft 1985], [Liming 1944], [Middleditch and Sears 1985], [Rockwood and Owen 1985], and by the radius of the ball in rolling-ball blends [Rossignac and Requicha 1984]. Global blends do not necessarily heel to a primitive surface, as in [Ricci 1973] and the example we provide in figure 5.25.

The blend we propose in section 5.6 is an extension of the global blend, and is most similar to those of [Blinn 1982], [Nishimura *et al*. 1985], and [Wyvill *et al*. 1986]. Other global blends are piecewise algebraic [Sederberg 1985], [Warren 1989], and [Bajaj 1992]. Before discussing our proposed blend, we briefly review those mentioned above.

5.5.4.1 The Rolling Ball Blend

The envelope of a ball rolling upon base surfaces produces a blend of the base surfaces, as shown below. Unfortunately, when expressed algebraically, this blend may be of high order, with hundreds of terms [Rockwood and Owen 1985].[5]

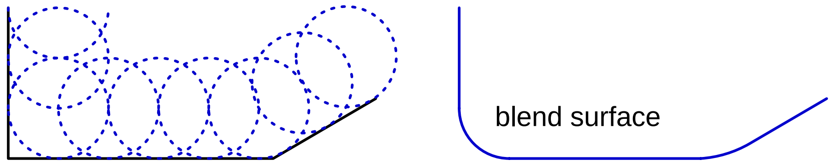

Figure 5.23 The Rolling Ball Blend

In [Rossignac and Requicha 1984] the rolling ball blend is approximated by applying offsets and shrinkages to an original shape, as demonstrated in the illustration below. This method produces rounds and fillets (approximated piecewise by cylinders and tori), whereas direct application of the method shown in figure 5.23 will not round a convex corner. The authors note that this method occasionally produces fillets at undesired locations, and in [Woodwark 1986] the method is suggested to be difficult to implement. In the following illustration, dashed lines represent the new contour; solid lines represent the contour from the previous step. The rolling ball method converts the solid line at the far left to the dashed line at the far right.

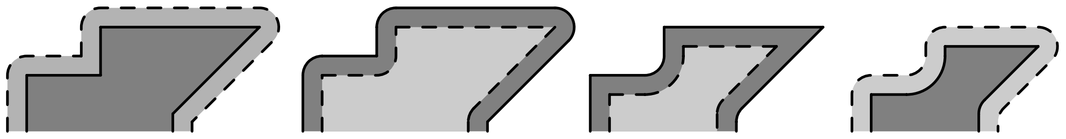

Figure 5.24 Approximation to the Rolling Ball Blend

left: initial shape is offset outward to form an outline with rounded convex corners

middle left: the outline shrinks to produce a surface with rounded concave corners

middle right: the surface shrinks again, retaining rounded concave corners

right: a second offset produces rounded concave and convex corners

(after [Rossignac and Requicha 1984])

5.5.4.2 Global Blends

A distinguishing characteristic of a blend is its degree, that is, the extent to which it is local or global. As described in [Warren 1989], algebraic surface blends have a well-defined form involving a sum of weighted polynomial primitives; the weighting determines the local and global qualities of the blend. An example of a simple blend, the average of three primitives, is shown below. Here we use the curve from figure 5.17. Although the creases are reduced, important local detail is also lost. A more sophisticated blend is required.

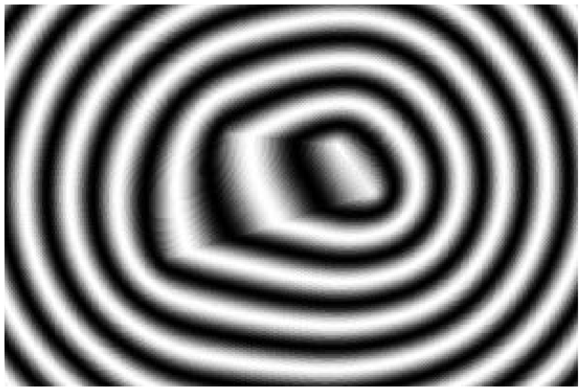

Figure 5.25 Average of Distances of 3-Segment Curve Approximation

5.5.4.3 Range-Controlled Blends

In [Rockwood and Owen 1985] is introduced a range-controlled blend that combines two three-dimensional primitives according to a two-dimensional 'blending function.' The range parameters used in the two-dimensional blend determine where the surface heels to the primitive surfaces. We give here the 'elliptic blend' of [Rockwood 1989], in which the amount of blending depends on the mutual proximity of the primitives P_1 and P_2:

$$(5.12)\quad B(P_1, P_2) = 1 - \left[1 - \frac{P_1(\mathbf{p})}{r_1}\right]_+^2 - \left[1 - \frac{P_2(\mathbf{p})}{r_2}\right]_+^2$$

where

P_1, P_2 are 'algebraic distances' (such as Euclidean) to elements 1, 2, and usually are presumed C^1 continuous.[6]

r_1 and r_2 are the ranges of influence for primitives P_1 and P_2, and

$[x]_+$ is *max* $(0, x)$.

The elliptic blend is so called because the graph of B is elliptical. In contrast to the average-of-distances blend, the elliptic blend heels to a primitive surface when beyond the range of other primitives. In the diagram below, we see an *added material blend* where the blend surface is exterior to the union of algebraic primitives (a *subtracted material blend* occurs if a blend surface is interior to the algebraic primitives). This blend occurs within the region of intersection of the primitive ranges. In the diagram below, the primitives are spherical, each of the form $P_i(\boldsymbol{p}) = \|\boldsymbol{p}-\boldsymbol{center}_i\|/radius_i-1$ (‘radius’ is distinct from ‘range’).

For $\boldsymbol{p}$ that is $radius_i$ distant from $\boldsymbol{center}_i$, P_i is 0. For a point closer to the center, that is, within the solid region of $primitive_i$, P_i is less than zero so that $[1-P_i(\boldsymbol{p})/r_i]_+^t > 1$ and $B < 0$. In other words, if a point is within a primitive volume, it is within the blend. If beyond the range of $primitive_i$, $P_i(\boldsymbol{p})/r_i > 1$ and $[1-P_i(\boldsymbol{p})/r_i]_+^2 = 0$; in other words, the primitive has no influence on the blend surface.

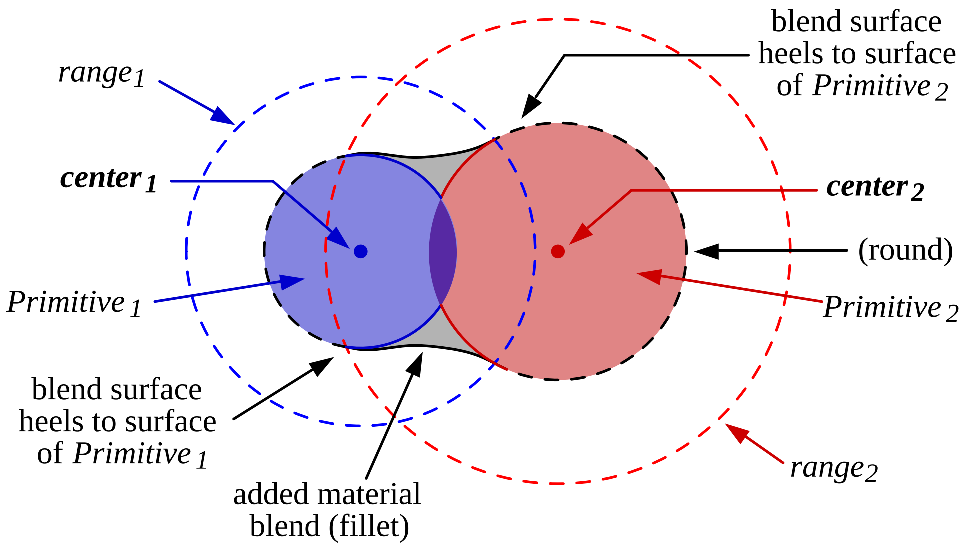
range 1
center 1
Primitive 1
blend surface heels to surface of Primitive 1
added material blend (fillet)
blend surface heels to surface of Primitive 2
center 2
(round)
Primitive 2
range 2

Figure 5.26 The Elliptic Blend

In [Rockwood 1989] the elliptic blend is extended to be *super-elliptic* by incorporating a real valued t, the ‘thumbweight:’

$$(5.13)\quad B(P_1, P_2) = 1 - \left[1-\frac{P_1(\boldsymbol{p})}{r_1}\right]_+^t - \left[1-\frac{P_2(\boldsymbol{p})}{r_2}\right]_+^t$$

Below are graphs of those points ($x = P_1(\boldsymbol{p})$, y $= P_2(\boldsymbol{p})$) for which $B(P_1, P_2) = 0$, for several values of t. For $t = \infty$, we obtain the union of the primitives.

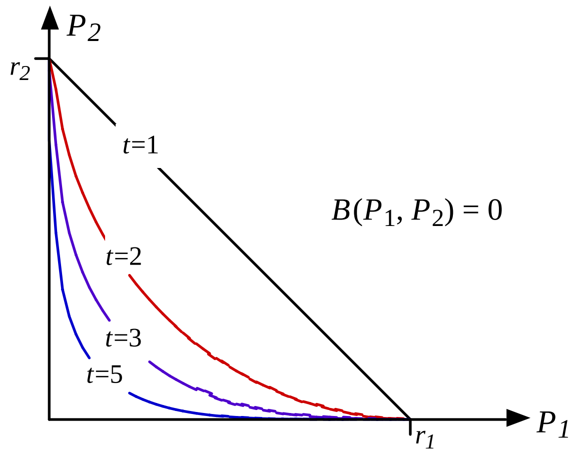

Figure 5.27 The Super-Elliptic Blend

(after Rockwood)

Consider an example from [Rockwood 1989] that blends the unit sphere (centered on the origin with radius 1) with a cylinder (centered on the x-axis with $x \in [0, 2]$ and radius .4). The sphere primitive is signed Euclidean distance from the sphere surface and the cylinder primitive is signed Euclidean distance from the cylinder surface; that is:

$$\text{sphere: } P_1(\mathbf{p}) = (\mathbf{p}_x^2+\mathbf{p}_y^2+\mathbf{p}_z^2)^{1/2}-1$$

$$\text{cylinder: } P_2(\mathbf{p}) = \begin{cases} (\mathbf{p}_x^2+\mathbf{p}_y^2+\mathbf{p}_z^2)^{1/2}-.4, & \mathbf{p}_x < 0 \\ (\mathbf{p}_y^2+\mathbf{p}_z^2)^{1/2}-.4, & 0 < \mathbf{p}_x < 2 \\ ((\mathbf{p}_x-2)^2+\mathbf{p}_y^2+\mathbf{p}_z^2)^{1/2}-.4, & \mathbf{p}_x > 2 \end{cases}$$

The blend $f(\mathbf{p}) = B(P_1(\mathbf{p}), P_2(\mathbf{p}))$, with $t = 3$, is displayed below, left, as a cross-section in the xy-plane. Although P_1 and P_2 are linear with respect to distance, B is not, which accounts for the unequal contour spacing. The rendered surface is shown below, right.

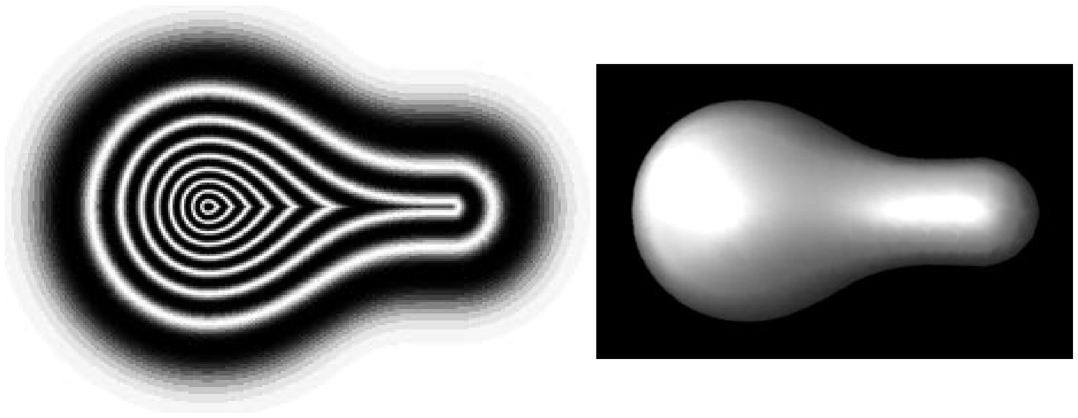

Figure 5.28 Super-Elliptic Blend of Sphere and Cylinder

left: $B(P_1(\mathbf{p}), P_2(\mathbf{p}))$ in xy-plane, right: rendered surface, $f(\mathbf{p}) = B(P_1(\mathbf{p}), P_2(\mathbf{p}))$

Similar blends, using a different formula for *B*, may be found in [Hoffmann and Hopcroft 1985] and [Middleditch and Sears 1985].

5.5.5 Bulges and the Combination Surface

We now consider two cylinders, one along the *x*-axis and one along the *z*-axis, as shown below, left. The super-elliptic blend of the cylinders, as shown to the right in the $z = 0$ plane, exhibits a bulge where the cylinders intersect.[7] Similar artifacts are visible in [Hoffmann and Hopcroft 1985] and [Middleditch and Sears 1985].

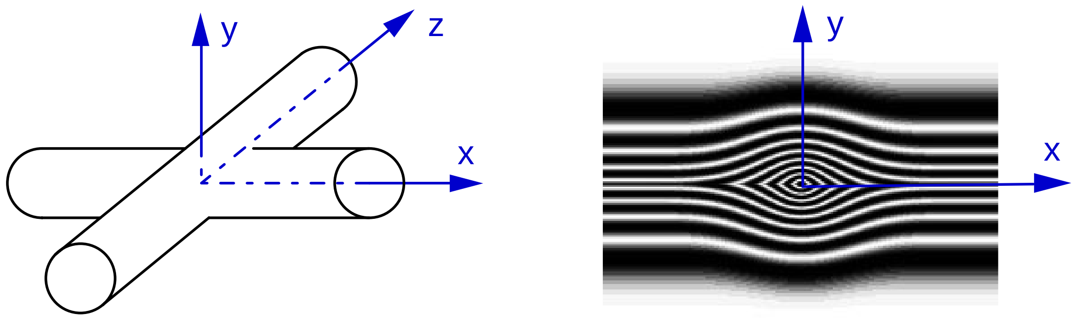

Figure 5.29 Super-Elliptic Blend (thumbweight = 3) of Two Cylinders

This bulge is to be expected, as those primitive values P_1 and P_2 that satisfy $B(P_1,$

P_2) = 0 do not sum to a constant. To compensate, a modification to the blend is discussed in [Rockwood 1989] whereby the range of a primitive is diminished according to the angle θ between the gradients of the two primitives at a point $\boldsymbol{p}$:

$$(5.14)\quad B(P_1, P_2) = 1-\left[1-\frac{P_1(\boldsymbol{p})}{r_1(1-\cos\theta))}\right]_+^t-\left[1-\frac{P_2(\boldsymbol{p})}{r_2(1-\cos\theta))}\right]_+^t.$$

As we have noted, offset surfaces round corners along convex regions of a skeleton. Blending is needed only along concave skeletal regions. In equation 5.14 the range is not diminished, and a blend occurs, for the fully concave condition of $\theta = 90^o$. For $\theta = 0$, the range is fully diminished and the simple union of the primitives results. We may express this relationship in general terms:

$$(5.15)\quad f(\boldsymbol{p}) = B(\boldsymbol{p})+ConvexityMeasure(\boldsymbol{p})\,(U(\boldsymbol{p})-B(\boldsymbol{p})),$$

where $B(\boldsymbol{p})$ is an arbitrary blend, $U(\boldsymbol{p})$ is the implicit definition for union, and *ConvexityMeasure*($\boldsymbol{p}$) is cos(θ), where θ is the angle between the vectors from $\boldsymbol{p}$ and the nearest points on the skeletal elements, as shown below. In [Rockwood 1989] cos(θ) is discussed in terms of a first order approximation to the co-tangency of two primitive surfaces; for many functions *ConvexityMeasure* is equivalent.

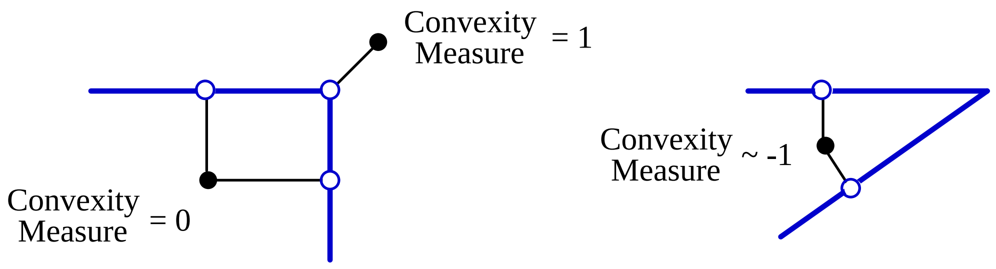

Figure 5.30 ConvexityMeasure

To further illustrate the combination of blend and union, we employ a somewhat simpler formula than equation 5.12:

$$(5.16)\quad B(P_1, P_2) = 1-(1-P_1)\,(1-P_2).$$

And we define

$$P_1 = h\,((d(\mathbf{p}, segment_1) - radius_1)/range_1)$$

$$P_2 = h\,((d(\mathbf{p}, segment_2) - radius_2)/range_2)$$

where $range_i$ is the radius of influence (> $radius_i$) of $segment_i$, and $h(x)$ is a decay function, shown below, such that $dh/dx = 0$ for $x < 0$ and for $h = 0$:

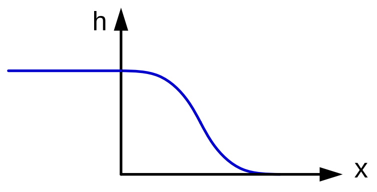

Figure 5.31 The Function h

As h is monotonically decreasing (rather than the increasing function in equations 5.9 and 5.10), we compute the union $U(P_1, P_2)$ as $max(P_1, P_2)$. Shown below are $U(P_1, P_2)$, $B(P_1, P_2)$, and the bulge-free combination, $f(\mathbf{p})$, computed with (5.15).

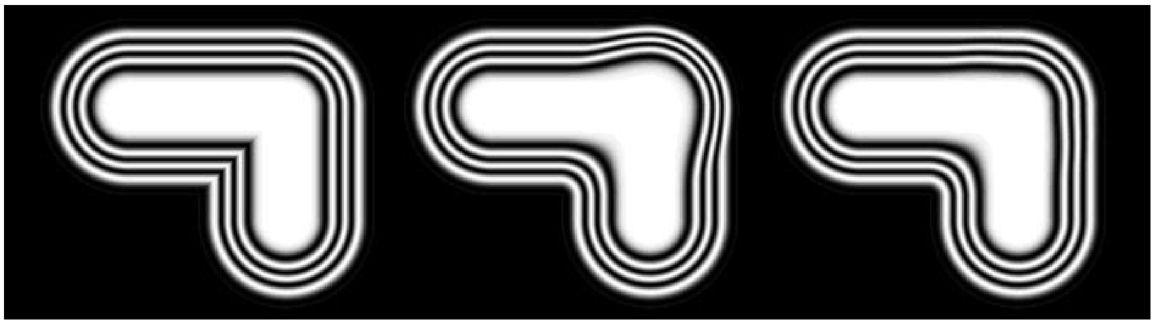

Figure 5.32 Two Perpendicular Segments

left: union, middle: bulging blend, right: combination

In the previous example, *ConvexityMeasure* is nonnegative. Should the two segments form an acute angle, however, cos(θ) could become as small as –1. Negative convexity values are undesirable; as can be seen below, a small protuberance results. A simple solution is to clamp cos(θ). Thus, our generic bulge-free *combination surface* becomes:

(5.17) $f(\boldsymbol{p}) = B(\boldsymbol{p}) + [\mathit{ConvexityMeasure}(\boldsymbol{p})]_{+}(U(\boldsymbol{p}) - B(\boldsymbol{p}))$.

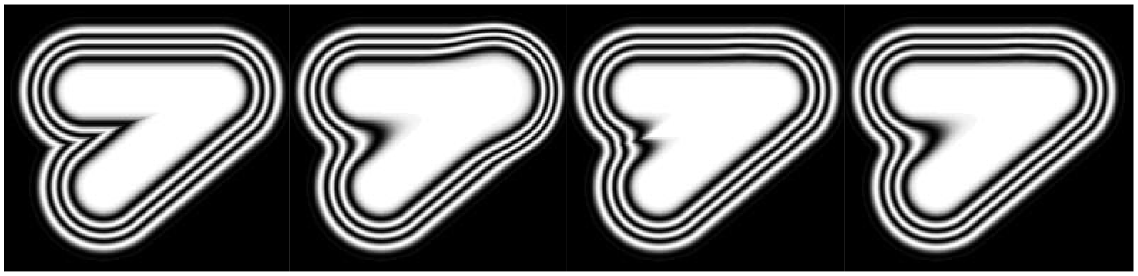

Figure 5.33 Two Acute Segments

left to right: union, bulging blend, unclamped combination, clamped combination

5.6 Convolution

ConvexityMeasure in equation 5.17 suggests that bulge-free blend methods require more information than distances to the skeletal elements. For example, in the figure below $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$ are distance d from each segment. The distance surface, dashed, is naturally rounded along the convex side of the skeleton; to produce a fillet along the concave portion of the skeleton, the implicit value at $\boldsymbol{p}_2$ must be negative, so that the actual surface, shown dotted, will pass through $\boldsymbol{p}_2'$. That is, the function must treat $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$ differently, depending on their relation to the skeleton. This is done in equations 5.14 and 5.17 by measuring the angle between the primitive gradients (*i.e.*, $\cos(\theta) = \nabla P_1 \cdot \nabla P_2$, for unit length ∇P_1 and ∇P_2).

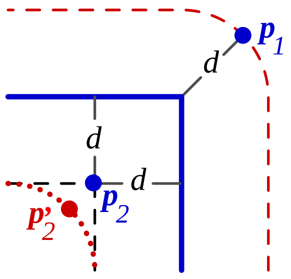

Figure 5.34 More than Distance is Required for a Desired Blend

In [Rockwood 1989] equation 5.13 is extended to k primitives:

(5.18) $$B_k = 1 - \sum_{i=1}^{k} \left[1 - \frac{P_i(\boldsymbol{p})}{r_i}\right]_+^t - c$$

The application of equation 5.18 to the non-bulge technique of equation 5.14 is problematic; in particular, what should θ measure? Equation 5.14 could be applied to successive primitives by functional composition, ordered according to the structure of a *CSG* tree. We prefer, however, an independent evaluation of primitives, in arbitrary order. Such independence simplifies implementation, promotes extensibility, and does not influence the design of the object.[8]

An independent evaluation of primitives is a feature of *convolution*, proposed as a bulge-free implicit blend technique in [Bloomenthal and Shoemake 1991]. This section is an expansion and elaboration on that material.

Although usually associated with signal processing, convolution has found application in the modeling of two and three-dimensional shapes. B-spline curves and surfaces, for example, can be defined as the convolution of the B-spline basis functions with control points [Farin 1988]. Indeed, convolution has been used previously as an implicit surface blend technique, although in a substantially different manner than presented here [Colburn 1990]. A discussion of multi-dimensional convolution can be found in [Dudgeon and Mersereau 1984].

Three-dimensional convolution treats a skeleton S as a set of points, each of which contributes to the implicit surface function according to its distance to $\boldsymbol{p}$. This is reminiscent of [Blinn 1982], in which an implicit surface is defined as the summation of terms, each based on the exponential (*i.e.*, *Gaussian*) decay of distance to a point:

(5.19) $f(\boldsymbol{p}) = c - \sum_i e^{-\|\boldsymbol{p}-\boldsymbol{s}_i\|^2/2}$, where a point on the skeleton is denoted by $\boldsymbol{s}_i$.

This summation is similar to the elliptic blend, although terms differ. Below, at left, is the elliptic blend (equation 5.12); at right is the Gaussian (equation 5.19).

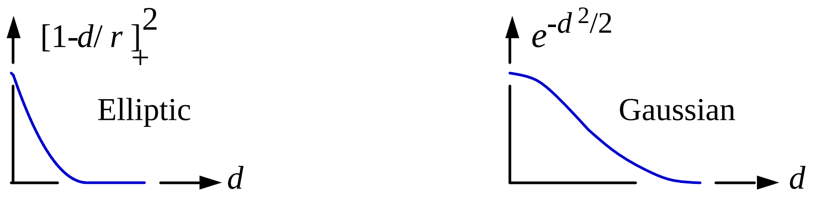

Figure 5.35 Terms for Two Blends ($d = ||\boldsymbol{s}_i - \boldsymbol{p}||$)

Although additional terms in equation 5.14 may be accommodated through composition (for example, according to the nodes of a *CSG* tree: $f(\boldsymbol{p}) = B(P_3, B(P_1, P_2))$ where P_3 belongs to the parent node of P_1 and P_2), additional terms in equation 5.19 are accommodated by summation.

Typically the skeleton S is approximated by a finite set of points. If, however, $S = \{\boldsymbol{s}_i\}$ is a set of infinitesimally spaced points, f can be expressed as an integral:

(5.20) $f(\boldsymbol{p}) = c - \int_S e^{-||\boldsymbol{p}-\boldsymbol{u}||^2/2}\, d\boldsymbol{u}$,

where $\boldsymbol{u}$ ranges over all points on the skeleton.

This is the convolution of a skeleton with a three-dimensional Gaussian *filter kernel*. It is a process whereby a *signal* is modified by a *filter*. Unlike algebraic blends, convolution achieves blending through integration.

We now consider a single kernel placed in two-dimensional space; any point $\boldsymbol{p}$ on the plane contributes to f according to the value of the kernel directly above $\boldsymbol{p}$. As illustrated, the kernel value is maximal for $\boldsymbol{p}$ at the kernel center and monotonically decreases as $\boldsymbol{p}$ moves away from the center.

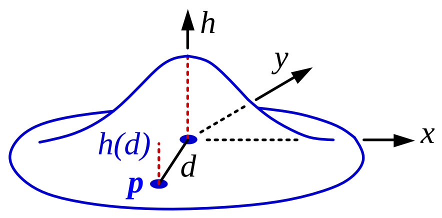

Figure 5.36 2D Filter Kernel, h(d)

Consider a set of evenly spaced kernels along a two-dimensional line segment *S*. As the number of kernels increases, adjacent kernels increasingly overlap and must be scaled so that their sum remains approximately constant (the kernels, thus, form a *partition of unity*). This is illustrated below for one to four kernels. Within each of the groups, the *d*-axis indicates the length of the segment as well as the domain of *h*. Although *d* is nonnegative, the function *h* is shown symmetrically around the *h*-axis to indicate distance to the left or to the right of the kernel center.

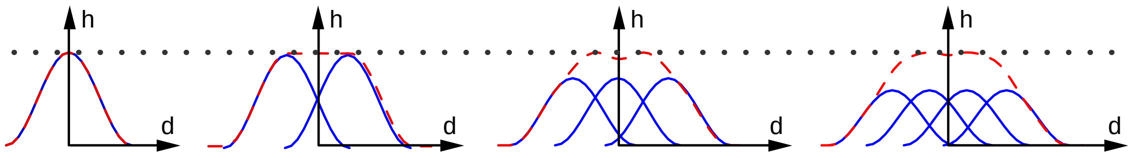

Figure 5.37 Sum of Weighted Kernels

To fully span the segment, the number of kernels must become infinite, the weight of each must become zero, and the summation must become *f*, the convolution of the kernel with the entire segment, as shown below. Mathematically, the convolution of *h* with a three-dimensional signal is given by $f = h \otimes s$. Before discussing the evaluation of *f*, we elaborate on its blend properties.

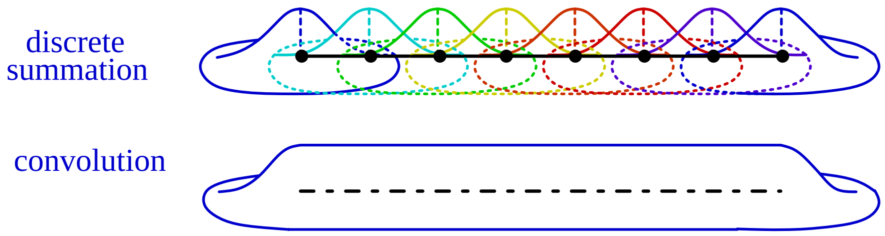

Figure 5.38 Infinite Sum is a Convolution

5.6.1 Convolution Blends

In general, the frequency components of the signal (or shape) are scaled by those of the filter. This can produce various results, but low-pass filtering, *i.e.*, the removal of high frequency components in the original shape, is the most relevant to blending.

When an *n*-dimensional signal is low-pass filtered, it loses detail and is said to be *smoothed*. For example, a two-dimensional image becomes blurred and objectionable creases, such as those in figure 5.22, are reduced or eliminated. The loss of high frequency energy is shown by the *Fourier transform*, which converts a signal (such as the kernel itself) into its frequency components. Interestingly, the Gaussian is its own Fourier transform; thus, the upper frequencies of the original shape are attenuated to a gently increasing degree. We call the zero set of the smoothed function a *convolution surface*.

Convolution is well-known as an elegant solution to a wide range of application problems; here its elegance lies in its ability to smooth a *non-branching* shape and not introduce bulges. This is due to its property of *superposition*. In subsequent sub-sections we discuss particular filters useful in constructing convolution surfaces, but first we discuss superposition and its relation to bulge-free blending.

In previous images we have highlighted creases by mapping *f* through sinusoids; in the following sections we do not, so that more subtle qualities may be examined.

5.6.2 Superposition

Because convolution is a linear operator, the sum of convolutions of any division of a skeleton is identically equal to the single convolution of the entire skeleton. This guarantees, for example, that two abutting, collinear segments, such as in the following figure, produce the same convolution as does the single segment that is their union. In other words, $h \otimes (s_1 + s_2) = (h \otimes s_1) + (h \otimes s_2)$. This property is known as superposition and it is essential to bulge-free blends.

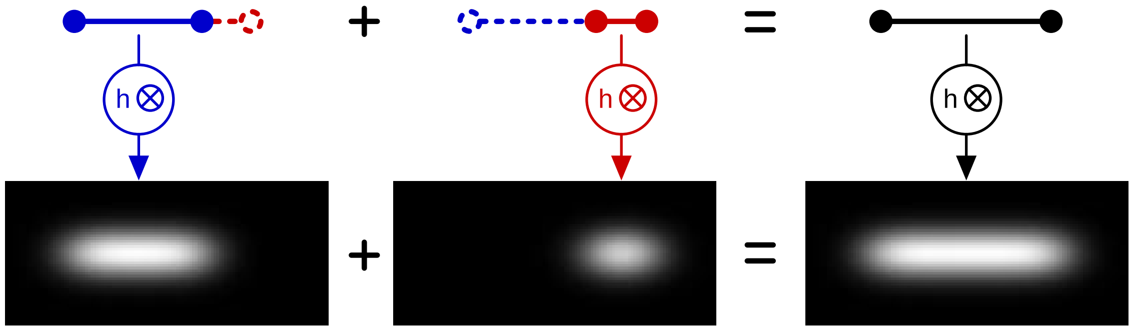

Figure 5.39 The Superposition Property of Convolution

That the convolution of a union of skeletal elements is identical to the sum of convolutions of the elements is in marked contrast with piecewise linear approximations in which the surface from a skeleton broken into pieces is not the same as that of the unbroken skeleton (compare with, *e.g.*, figures 5.17 and 5.25). This is because each skeleton generates a surface which is a union (using *max*) of components, whereas the convolution blend uses summation. If the skeleton is divided into infinitesimal pieces (*i.e.*, a set of points that continuously spans the skeleton), *max* (the union) becomes irrelevant because each piece is a set of one point. The result is a pure summation, or integral. Because of superposition, we may, in our implementation, convolve each skeletal element individually and sum the results.

5.6.3 Evaluation

Evaluation of the convolution integral, equation 5.20, requires ***u*** to range over the entire skeleton. The integral could be approximated by a summation if the skeleton were approximated by a set of points. For a sufficiently dense set, the summation could be a significant computational expense, which we wish to avoid.

Fortunately, the Gaussian filter is *separable*; it can be factored into a product of lower-dimensional Gaussians.[9] For a three-dimensional convolution, we can separate the *x*, *y*, and *z* terms of a vector ***d***:

(5.21) $h(\boldsymbol{d}) = e^{-\|\boldsymbol{d}\|^2/2} = e^{-(\boldsymbol{d}_x{}^2+\boldsymbol{d}_y{}^2+\boldsymbol{d}_z{}^2)/2} = e^{-\boldsymbol{d}_x{}^2/2}\, e^{-\boldsymbol{d}_y{}^2/2}\, e^{-\boldsymbol{d}_z{}^2/2}$.

Before we can apply this simplification, we must further consider the signal s, which is our one-dimensional skeleton S. In particular, S has infinitesimal thickness, and yet we wish $s(\boldsymbol{p})$ to integrate to 1 over a plane perpendicularly intersecting the skeleton. For continuous s, this is characteristic of the Dirac delta function, which is ∞ at the origin and 0 elsewhere, as shown below.

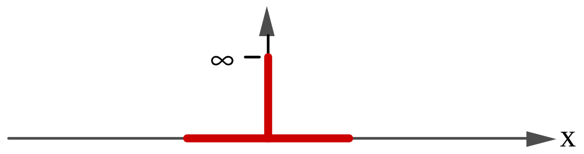

Figure 5.40 The Delta Function δ(x)

Thus, we may write s as:

(5.22) $$s(\boldsymbol{p}) = \begin{cases} \infty & \boldsymbol{p} \text{ on the skeleton } S \\ 0 & \boldsymbol{p} \text{ not on the skeleton } S. \end{cases}$$

If we require S to lie on the x-axis, between the origin and $(a, 0, 0)$, we may rewrite $s(\boldsymbol{p})$ as the product of one-dimensional delta functions, $k(\boldsymbol{p}_x)\ \delta(\boldsymbol{p}_y)\ \delta(\boldsymbol{p}_z)$, where

(5.23) $$k(x) = \begin{cases} 1 & \text{for } 0 \le x \le a \\ 0 & \text{otherwise.} \end{cases}$$

We observe that the convolution of a delta function with a filter is the filter itself. This *sifting property* of delta functions may be expressed as:

(5.24) $$(\delta \otimes f)(x) = \int_{-\infty}^{\infty} \delta(t)\, f(x-t)\, dt = f(x).$$

We can now express the convolution of the kernel h with the skeleton S as:

$$f(\boldsymbol{p}) = c - (h \otimes s)(\boldsymbol{p}) = c - \int_{\Re^3} h(\boldsymbol{p}-\boldsymbol{u})\, s(\boldsymbol{u})\, d\boldsymbol{u}.$$

Applying (5.21) yields:

$$f(\boldsymbol{p}) = c - \int_{-\infty}^{\infty} k(\boldsymbol{u}_x)e^{-(\boldsymbol{p}_x-\boldsymbol{u}_x)^2/2}d\boldsymbol{u}_x \int_{-\infty}^{\infty} \delta(\boldsymbol{u}_y)e^{-(\boldsymbol{p}_y-\boldsymbol{u}_y)^2/2}d\boldsymbol{u}_y \int_{-\infty}^{\infty} \delta(\boldsymbol{u}_z)e^{-(\boldsymbol{p}_z-\boldsymbol{u}_z)^2/2}d\boldsymbol{u}_z.$$

The latter two integrals disappear due to the sifting property, yielding:

$$c - (\int_0^a e^{-(\boldsymbol{p}_x-\boldsymbol{u}_x)^2/2}d\boldsymbol{u}_x)\ (e^{-\boldsymbol{p}_y^2/2}\ e^{-\boldsymbol{p}_z^2/2}) =$$

$$c - (\int_0^a e^{-(\boldsymbol{p}_x-\boldsymbol{u}_x)^2/2}d\boldsymbol{u}_x)\ e^{-(\boldsymbol{p}_y^2+\boldsymbol{p}_z^2)/2}. \tag{5.25}$$

We refer to the first term, $\int_0^a e^{-(\boldsymbol{p}_x-\boldsymbol{u}_x)^2/2}d\boldsymbol{u}_x$, as the *integration filter*, and to the second term, $e^{-(\boldsymbol{p}_y^2+\boldsymbol{p}_z^2)/2}$, as the *distance filter*. $\boldsymbol{p}_y^2+\boldsymbol{p}_z^2$ is the square of the distance from $\boldsymbol{p}$ to the *x*-axis (the line containing the segment), *not* to the segment itself.

We must still compute the one-dimensional integral of $e^{-(\boldsymbol{p}_x-\boldsymbol{u}_x)^2/2}$. Along the *x*-axis *s* can be treated as a *box function*; its convolution at a given point is simply the area of the kernel subtended by the box, when the kernel is centered at the given point. To illustrate, the areas of the kernel subtended by a segment at several points are shown below, assuming a kernel with *unit integral* (*i.e.*, the area under the curve of the kernel is one) and a box function with unit height.

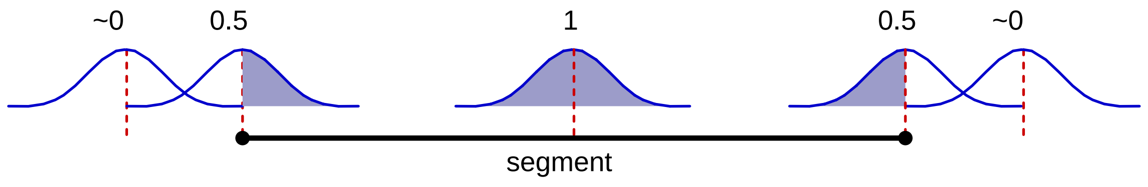

Figure 5.41 Integrating the Kernel at Various Points along a Segment

We can now see, though in one dimension only, the process of superposition: no matter where we break the above segment, the sum of the areas under the kernel subtended by the component segments will equal the area subtended by the union of the segments. Thus, as shown below, the actual point of contact between two collinear segments is immaterial.

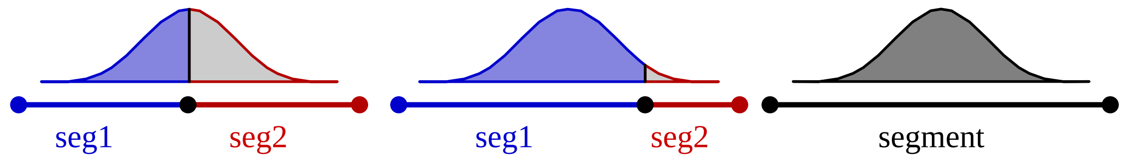

Figure 5.42 An Illustration of Superposition

The entire one-dimensional convolution is the graph of the above subtended value as a function of position; a smooth curve, reproduced below, results. The convolution approaches 0 beyond the segment endpoints, equals ½ at the segment endpoints (assuming *h* is symmetric), and plateaus at 1 in the middle of the segment. The shape and extent of the roll-offs to either side of the plateau depend on the filter kernel.

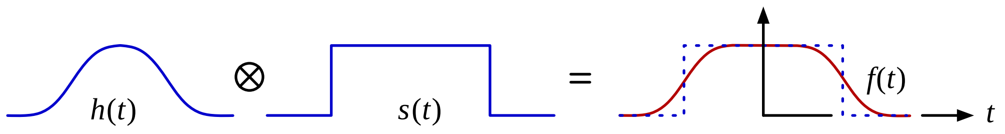

Figure 5.43 Convolution of a Box Function

5.6.4 Arbitrary Position and Orientation

We generalize equation 5.25 to line segments in arbitrary position and orientation using a series of steps, illustrated below. First, the point ***p*** is projected to the line containing the segment; we call the projection ***p'***. The distance from ***p'*** to the nearest segment endpoint is used to compute (or index) the integration filter of the segment along the axis of the segment; this is $area_{ds}$ in the figure. The distance filter, in the plane orthogonal to the segment, is simply the value of the filter given $d_{\boldsymbol{pp'}}$, the distance from ***p*** to ***p'***. The three-dimensional convolution at ***p*** is the product of the integration filter and distance filter.

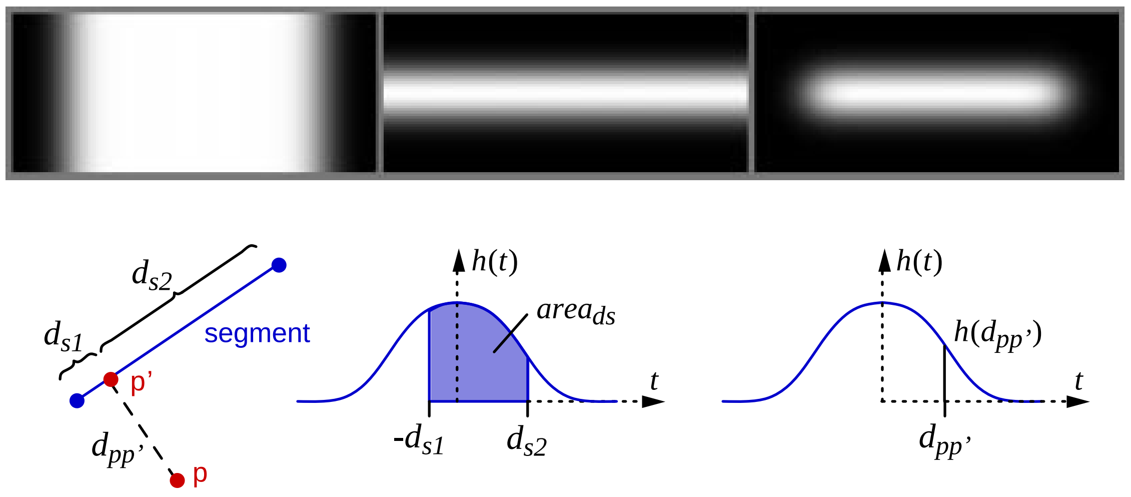

Figure 5.44 Lower-Dimensional Evaluation of the Convolution of a Segment

$f(\mathbf{p}) = area_{ds} \times h(d_{\mathbf{pp}'})$

top: left to right: integration filter, distance filter, convolution

*bottom left: geometry of **p** and the segment S*

bottom middle: integration filter along the axis of S

bottom right: distance filter in the plane orthogonal to S

The two-dimensional convolution requires a single evaluation of the Gaussian; the one-dimensional convolution requires its integration. The Gaussian is a symmetric function (*i.e.*, $f(t) = f(-t)$); in three dimensions it is spherically symmetric. As a consequence, our calculation of d need not consider segment orientation.

5.6.5 The Gaussian Integral

The convolution shown in figure 5.43 is the one-dimensional convolution of a Gaussian filter with a line segment, namely $\int h \, d\mathbf{u}_x$ evaluated from $-d_{s_1}$ to d_{s_2}, the distances illustrated in figure 5.44. In order to understand this integral, we use a description of the Gaussian that is more general than equation 5.20:

(5.26) $h(t) = e^{-\omega t^2}$,

where ω is the *width coefficient* and equals $1/2\sigma^2$. σ is the *standard deviation*, an attribute with widespread application to statistics. The integral of the Gaussian from $-\infty$ to $+\infty$ is $\sqrt{(\pi/\omega)}$. A Gaussian curve with width coefficient $\omega = 1$ has integral $\sqrt{\pi}$; a Gaussian with $\omega = \pi$ has unit integral.

Unit integral is an important property for a filter because it maintains the original energy of a signal; for example, if we convolve a unit integral kernel with a box function, the plateau amplitude (*i.e.*, *DC component*) will equal the amplitude of the box function.[10] The intensity of a segment that is long compared with the filter *support* (the half-width of the kernel) will be scaled by the integral of the kernel. Thus, a kernel with integral less than one would attenuate a signal; a kernel with integral greater than one would amplify the signal.

If we use a width coefficient other than π, but wish to maintain the overall energy of the signal, we must divide *f* by the integral of the Gaussian with the given width coefficient. We discuss this further in the next section, selecting ~.69314715 as our width coefficient. It is important that the same width coefficient be used for the integration filter as is used for the distance filter; otherwise the three-dimensional convolution would not be properly separated into lower order terms.

Unfortunately, the Gaussian integral (also known as the *error function*, often abbreviated ‘erf’) has no closed-form solution. Discrete values can be approximated numerically and stored for subsequent reference. In [Bloomenthal and Shoemake 1991] an array was created for each segment, which is not necessary. A single table of integration values, spanning the filter support, suffices.

Although the Gaussian has infinite support, it has very little energy for $t > 3$, with a width coefficient of .69314715. In practice we restrict the table domain to $[-10..0]$. -10 would seem a reasonable lower bound, as both $e^{-.69(10^2)}$ and the integral from $-\infty$ to -10 are zero to 30 decimal places. Because *h* is a symmetric function:

(5.27) $$\int_{-\infty}^{x} e^{-\omega t^2}\, dt \;=\; \int_{-\infty}^{\infty} e^{-\omega t^2}\, dt - \int_{-\infty}^{-x} e^{-\omega t^2}\, dt = \sqrt{(\pi/\omega)} - \int_{-\infty}^{-x} e^{-\omega t^2}\, dt.$$

Thus, our array need contain values of the integral from -10 to x, for negative x only, $x > -10$. Let $a(x) = \int_{-\infty}^{x} e^{-\omega t^2}\, dt$; then, for arbitrary x_1 and x_2:

(5.28) $$\int_{x_1}^{x_2} e^{-\omega t^2}\, dt \;=\; a(x_2) - a(x_1),$$

For positive x, $a(x) = \sqrt{(\pi/\omega)} - a(-x)$.

Rather than truncate the kernel, a kernel can be *windowed*, *i.e.*, smoothly attenuated to zero over a finite domain. We have not experimented with windowed kernels, but speculate there is no practical advantage in their use for line segments, and likely little advantage in their use for the polygonal skeletal elements discussed in the next chapter.

The integral table can be set using the trapezoidal rule or the power series. We implemented the former because of its simplicity, but, for completeness, we present the error function in terms of the latter [Abramowitz and Stegun 1965]:

(5.29) $$erf\,(x) \;=\; \frac{2}{\sqrt{\pi}} \sum_{i=0}^{\infty} \frac{(-1)^i\, x^{2i+1}}{i!\,(2i+1)} = \frac{2}{\sqrt{\pi}} \int_{0}^{x} e^{-t^2} dt.$$

The series can be terminated when terms fall below a given error threshold. Our function $a(x)$ could then be computed as $a(x) = \sqrt{(\pi/\omega)}\,(1+erf\,(x))\,/\,2$.

To determine an adequate size for the table, we consider the cubic B-spline, a function similarly shaped to the Gaussian. We evaluated the B-spline integral both analytically and numerically with a 1000 element table; the maximum discrepancy between the two was 0.000000954, which is adequately low for our purposes. In the examples in this chapter, the Gaussian integral has been approximated by a 10,000 element table (details may be found in the pseudo-code presented in the Appendix).

A pre-computed array is computationally efficient and places only moderate demands on memory for a sufficiently high resolution in a single dimension. For two-dimensional skeletons (*i.e.*, polygons), the demands on memory are substantially higher, and we must accept those discretization artifacts that may accompany a lower resolution, two-dimensional array. Analytical methods for two dimensions, when possible, are far more complex than those for one-dimensional integration [Duff 1989].

5.6.6 Normalizations

In order that the surface pass through segment endpoints, we choose ½ as an iso-surface contour level. Thus, equation 5.20 becomes:

(5.30) $f(\boldsymbol{p}) = ½ - (h \otimes s)(\boldsymbol{p})$.

We order the terms above so that f is, per convention, negative inside and positive outside the implicit volume.

As in equation 5.12, we may also normalize projected distance to the skeleton against a desired radius. Here we assume S lies on the x-axis, so that the projected distance, $d_{\boldsymbol{pp}'}$, equals $\sqrt{(\boldsymbol{p}_y^2+\boldsymbol{p}_z^2)}$. Thus, the kernel can be expressed as:

(5.31) $h(d_{\boldsymbol{pp}'}) = e^{-\omega(\boldsymbol{p}_y^2+\boldsymbol{p}_z^2)/r^2}$.

Consider a point on the surface of a cylinder defined by a line segment, and assume that the point is in the middle of the length of the cylinder and that the cylinder is sufficiently long so that the one-dimensional integral (discussed in section 5.6.3) equals 1. Thus, equation 5.30 becomes:

(5.32) $f(\boldsymbol{p}) = ½ - e^{-\omega(\boldsymbol{p}_y^2+\boldsymbol{p}_z^2)} = 0$, for $\boldsymbol{p}$ on the surface.

As indicated by equation 5.31, it is possible to associate a radius with the segment endpoints so as to change the diameter of the resulting cylinder, or, if the radii differ, to define a tapered cylinder. Now, if ***p*** is on the surface and its projection ***p***' lies within the segment, then its distance to the *x*-axis is *r*. If we normalize as in equation 5.31, $f(\boldsymbol{p}) = ½ - e^{-\omega(\boldsymbol{p}_y^2+\boldsymbol{p}_z^2)/r^2} = ½ - e^{-\omega(r/r)^2} = ½ - e^{-\omega} = 0$. Thus, ω = natural log of 2; *i.e.*, ω ≈ .69314715 allows *h*(1) = ½.

5.6.7 Convolution Surfaces

The following image displays a convolution surface; it uses a Gaussian kernel with ω = .69314715 and was computed as described in the previous sections. The zero set of *f* has been highlighted; the surface could be defined along any contour, however, and blending can occur with other objects anywhere within the filter support (*i.e.*, the non-black portion of the image).

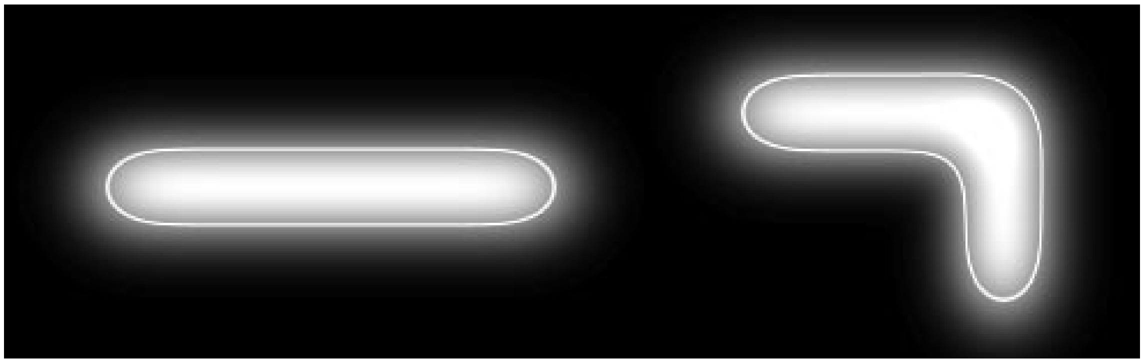

Figure 5.45 Two Segments Convolved with the Gaussian Kernel

To produce a smooth surface from a skeleton, we need only sum the convolutions of the skeletal elements, evaluating each convolution independently. For example, the contours below, resulting from the sum of two implicit primitives, are smooth regardless of the angle between the skeletal elements. Along convex portions of the skeleton the surface mimics the union operator, although convolution is not a union operation. Along concave portions, the surface remains smooth.

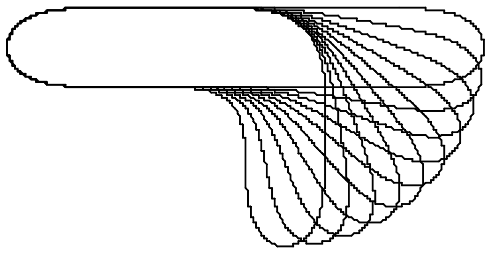

Figure 5.46 Rotated Segments

angles range from 0 to 90 degrees, in 10 degree increments

No bulges are produced, unlike the blend shown in figure 5.32, middle. Also, no creases are produced and the intermediate contours smoothly interpolate the extrema. Unfortunately, as we will see in section 5.6.14.2, convolution does produce a bulge when it is applied to a branching skeleton.

The following diagram is an interpolation between the distance surface and the convolution surface, both based on the curve in section 5.5.1. Not only is the convolution surface a smooth, bulge-free blend, but the interpolation to the distance surface is without anomaly. For isolated convex skeletons such as triangles, rectangles, or line segments, convolution surfaces have almost the same shape as distance surfaces. Now, however, concave skeletons yield smooth surfaces, and adjacent surfaces blend seamlessly.

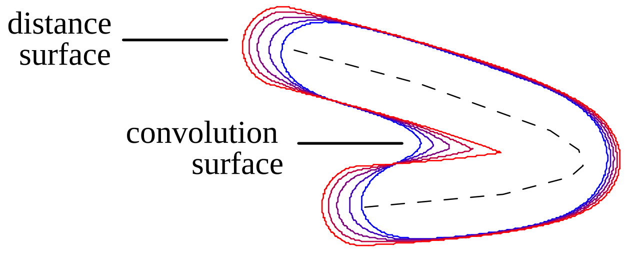

Figure 5.47 Seven Segments

inner contour: convolution surface, outer contour: distance surface

Although the Gaussian is a satisfactory kernel, we briefly examine the sinc, B-spline, and Wyvill kernels, which are prominent in the literature of computer graphics. We will not examine lower order kernels, such as the piecewise quadratic due to [Nishimura *et al.* 1985].

Different filters have different mathematical advantages. The Gaussian is separable, but its integral must be numerically approximated. The B-spline and Wyvill kernels can be analytically integrated, but they are not separable.[11] The sinc kernel is the ideal low-pass filter, but it is not separable and its integral must be numerically approximated. In the following sections, we discuss and compare these filters. Additional criteria for filters are given in [Kacic-Alesic 1991].

5.6.8 The Sinc Kernel

The sinc kernel is given by sin(πx)/πx. It has integral 1; it also has, unlike the other filters we discuss, negative lobes. Thus, elements in the neighborhood of the kernel center actually inhibit the signal (indeed, this behavior can be seen in the retina and other natural phenomena).

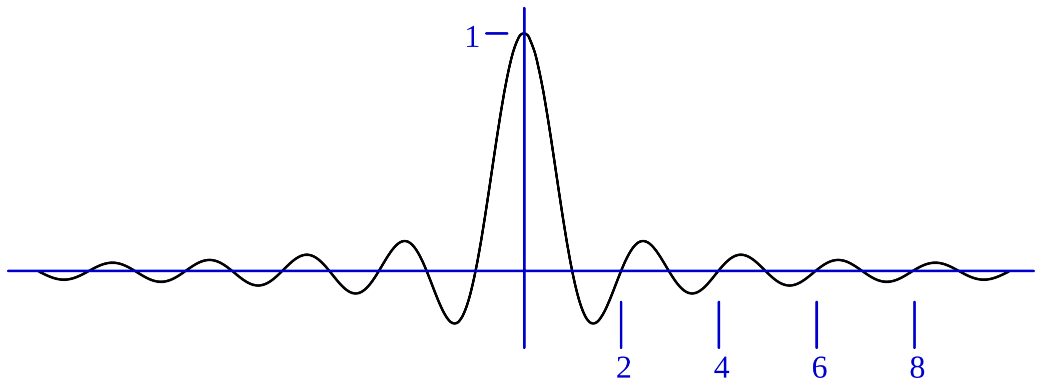

Figure 5.48 The Sinc Kernel

The sinc is the ideal low-pass filter; that is, its Fourier transform is the box function. No components of the original shape are attenuated below a certain frequency (the *cutoff frequency*); above this frequency, all components are removed. Ideal low-pass filtering is important to digital signal processing; in particular, the sampling theorem states that all frequencies above one-half the sampling frequency must be removed before the signal is sampled in order that the signal be reconstructed without aliasing. An ideal low-pass filter enables the most efficient, distortion-free discrete storage of a continuous signal.

The negative lobes raise a question concerning the iso-contour value. Although, as with the Gaussian, the maximum of the function is 1 and the function approaches 0 as *x* approaches ∞, the function is not monotonically decreasing and, indeed, has an infinite number of zero crossings. Thus, we must ask whether ½, or any value, is a meaningful iso-contour for the sinc kernel. Because the sinc is not separable, we approximate the skeleton with a sequence of points (we discuss such approximations in a later section). The image below compares a sequence of 100 point sources using the Gaussian and the sinc functions.

It would appear that monotonicity of the kernel is a necessary property for a satisfactory convolution surface; otherwise, the ringing of the filter may introduce a spurious contour, as in the image below. Because of ths artifact, we find the sinc unacceptable as a blending filter.

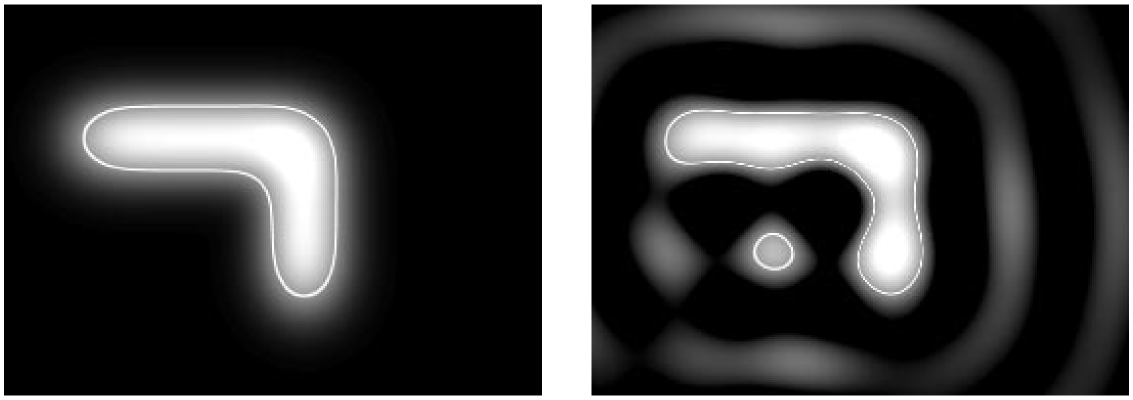

Figure 5.49 Convolution with Gaussian (left) and Sinc (right)

5.6.9 The B-spline and Wyvill Kernels

In this section we discuss the B-spline and Wyvill kernels, which are similarly shaped to the Gaussian. The B-spline kernel has prominent applications in computer graphics. That it has unit integral is important in the blending of curve or surface control points [Bartels *et al.* 1987]. It is a piecewise cubic filter, each piece defined parametrically in t as at^3+bt^2+ct+d. The function equals 2/3 for $t = 0$, and ½ for $t \approx .72235$.

The actual coefficients of the curve are given by the 4th order B-spline basis:

(5.31) $h_{B\text{-}spline} =$

$$\begin{array}{ll} (x+2)^3/6 & x \in (-2,-1) \\ (-3x^3-6x^2+4)/6 & x \in (-1,0) \\ (3x^3-6x^2+4)/6 & x \in (0,1) \\ (2-x)^3/6 & x \in (1,2) \end{array}$$

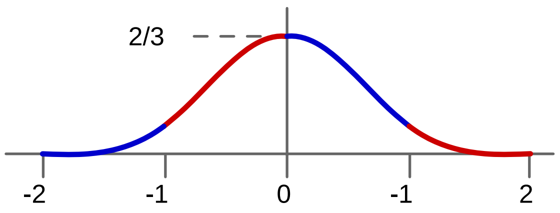

Figure 5.50 The Four Non-Zero Pieces of the Cubic B-Spline Kernel

The area under the B-spline curve is the following definite integral:

$$(5.33)\quad \int_{t_1}^{t_2} (at^3+bt^2+ct+d)\,dt = (at^4/4+bt^3/3+ct^2/2+dt)\Big|_{t_1}^{t_2}$$

The integral consists of four anti-derivatives, one for each non-zero kernel section:

$$\begin{array}{ll} (x+2)^4/24 & x \in (-2,-1) \\ (-3x^4-8x^3+16x)/24 & x \in (-1,0) \\ (3x^4-8x^3+16x)/24 & x \in (0,1) \\ -(2-x)^4/24 & x \in (1,2) \end{array}$$

We compute the integral $\int_{x_1}^{x_2} h$ as $(area(x_2)-area(x_1))/24$, where $area(x) = \int_{-\infty}^{x} h = areaNegX(x)$ for $x < 0$ else $24-areaNegX(-x)$, and

$$areaNegX(x) = \begin{cases} 0 & x < -2 \\ (x+2)^4 & x \in (-2,-1) \\ -3x^4-8x^3+16x+12 & x \in (-1,0) \end{cases}$$

In general, the near-trivial return of 1 (the kernel integral) is executed over the majority of a segment that is long compared with the filter support.

The filter described in [Wyvill *et al.* 1986] has a unit integral with a support of one (as opposed to a support of two for the cubic B-spline) and is defined by:

(5.34) $h_{Wyvill}(x) = (9-4x^6+17x^4-22x^2)/9.$

This filter is not separable but is separable, which allows to compute the area under its curve from $-\infty$ to positive x as

$$areaForPosX(x) = \begin{cases} 1, & x > 1 \\ ½-((4/7)x^7+(17/5)x^5-(22/3)x^3)/9+x, & x < 1 \end{cases}$$

and the integral $\int_{x_1}^{x_2} h_{Wyvill}$ for arbitrary x as

$$area(x_2)\text{-}area(x_1), \text{ and } area(x) = \begin{cases} 1\text{-}areaForPosX(\text{-}x), & x < 0 \\ areaForPosX(x), & x > 0 \end{cases}$$

There have been trials with $h(x) = (1-x^2)^3$, $|x| \le 2$ [Wyvill 1994]. This and equation 5.34 are graphed below.

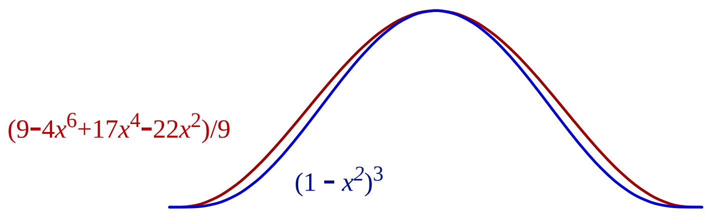

Figure 5.51 Rival Wyvill Filters

5.6.10 Separability, Revisited

Of the filters we have discussed, only the Gaussian is separable and spherically symmetric. This means that convolution with the B-spline or the Wyvill can only be approximated as a product of an integration filter and distance filter. Alternatively, the convolution can be given as the summation of terms due to point sources distributed along the skeleton.[12] We discuss the use of point sources in a subsequent section, but present here some results for the B-spline and Wyvill filters. The following images compare results for the summation of point source terms with results for the product of integration and distance filters.

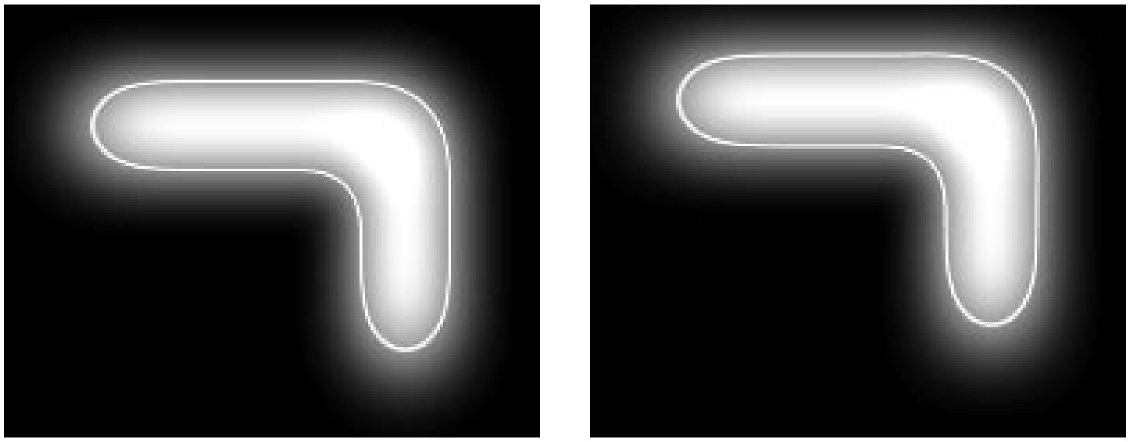

Figure 5.52 Approximations to B-Spline Convolution

left: summation of point sources, right: product of filters

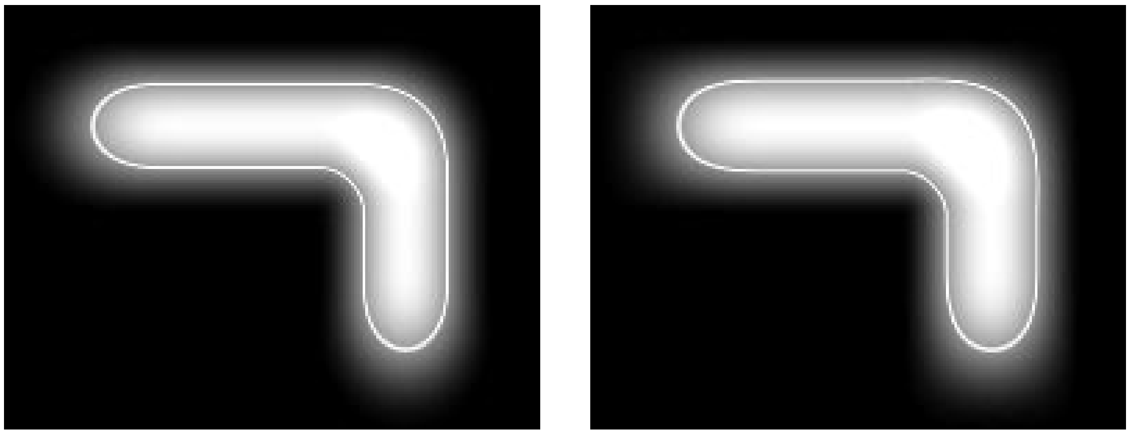

Figure 5.53 Approximations to Wyvill Convolution

left: summation of point sources, right: product of filters

There is remarkably little difference between the two methods for approximating the convolution. That we can treat these non-separable filters as separable and obtain reasonable results must be due to the similarity of these kernels with the Gaussian kernel. To illustrate this similarity, the three kernels (scaled so that $h(0) = 1$ and $h(1) = ½$) are graphed below. For all three the radius in isolation is 1 and the effective radius of influence is between 2 and 3. For now, we express no preference for a particular filter. Although in this chapter there are clear implementation differences, we do not observe any significant modeling differences until the next chapter.

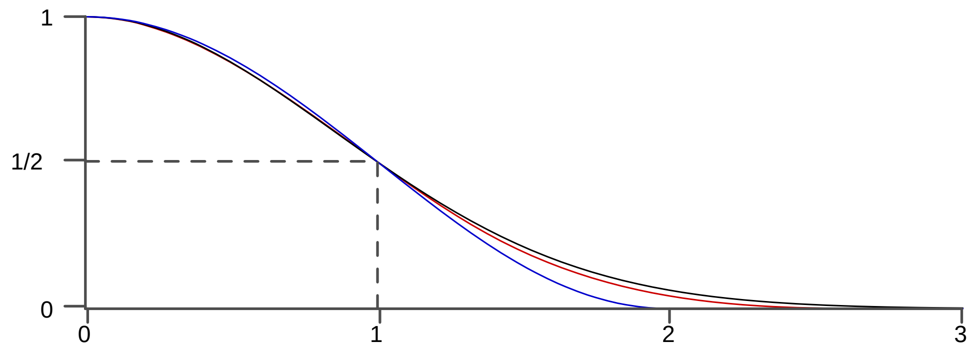

Figure 5.54 Comparison of Filter Kernels

upper (in black): the Gaussian kernel, $e^{-.69314715x^2}$

middle (in red): the B-spline kernel, $(1.5)h_{Bspline}(0.72235*x)$

lower (in blue): the Wyvill kernel, $h_{Wyvill}(0.5x)$

5.6.11 Blend Control

One might conclude from the previous section that separability, namely assuming that $h(\|\mathbf{p}\|) = h(\mathbf{p}_x)h(\mathbf{p}_y)h(\mathbf{p}_z)$ and treating the convolution as a product of a distance filter and an integration filter, can be applied freely, without regard for parameters such as the scales applied to the filter domains. But this is not so.

In section 5.6.3 the integration filter is normalized by the integral of the kernel so that the energy of the signal is neither attenuated nor amplified. Indeed, for sections of the surface where this integral is unity (*i.e.*, those points ***p*** whose projection onto the segment is within the segment and further than the filter support from either segment endpoint), the surface is not affected by the shape of the integral; it is only affected by the distance filter, which determines the thickness of the implicit volume. But towards the segment endpoints the shape of the integral is significant and must derive from the same kernel used for the distance filter; otherwise the separability of the Gaussian is not observed and the primitives will not blend properly.

For example, consider a scale applied to the domain of the integrating filter only. We can compensate for the change in the integral of the filter by dividing *f* by the scale. This change in scale will not affect the join of collinear segments; it does, however, affect the join of non-collinear segments as well as the shape of the free ends. For example, as the scale increases in the images below, the convex portion of the join as well as the free ends of the contour flatten. In this example, scales of 1.5 and 1.75 are used; the proper value is 1. The proper separation requires not only the same width coefficients for both the integration and distance filters, as observed in section 5.6.5, but also requires the same domain for the two filters.

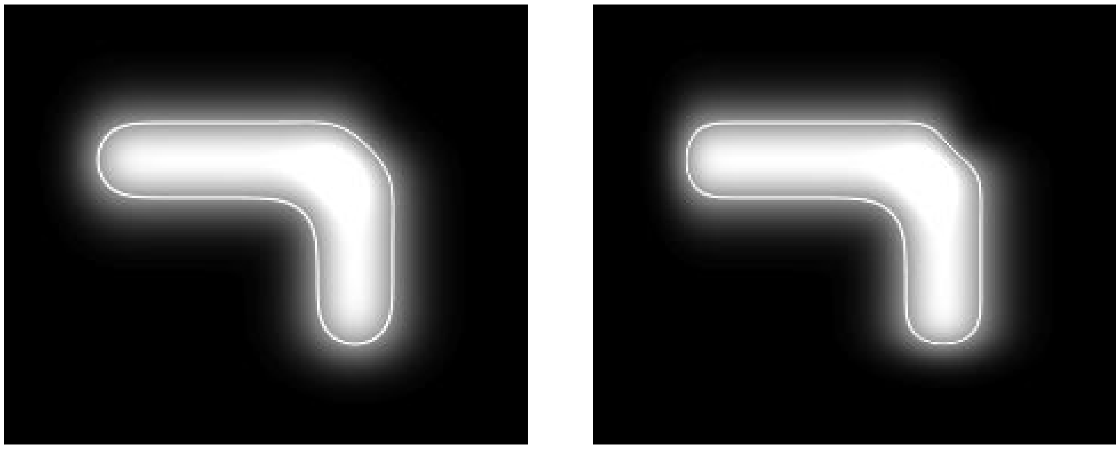

Figure 5.55 Domain of Integration Filter Scaled by 1.5 (left) and 1.75 (right)

Let us further consider the nature of the contour around the free end of a segment;

it is nearly semi-circular. In the diagram below, $\boldsymbol{p}_1$ is at the left endpoint of the segment. The one-dimensional integration filter will yield ½ at $\boldsymbol{p}_1$. Because $\boldsymbol{p}_1$ is on the segment, d = 0, and the distance filter for $\boldsymbol{p}_1$ will be 1; thus, $f(\boldsymbol{p}_1) = 0$, and $\boldsymbol{p}_1$ lies on the surface. The integration for $\boldsymbol{p}_2$ is from $-\infty$ to 1 (*i.e.*, $\boldsymbol{p}_2$ is r to the right of $\boldsymbol{p}_1$); this integral is approximately .88. Since $\boldsymbol{p}_2$ is r from the segment, d/r = 1 and h for $\boldsymbol{p}_2$ will be ½; thus $f(\boldsymbol{p}_2) \sim -.06$. If $\boldsymbol{p}_2$ were moved closer to the segment, so that its distance were .9r, then $h \sim .57$, and $f \approx 0$. Thus, the contour is slightly tapered compared with a semi-circle.

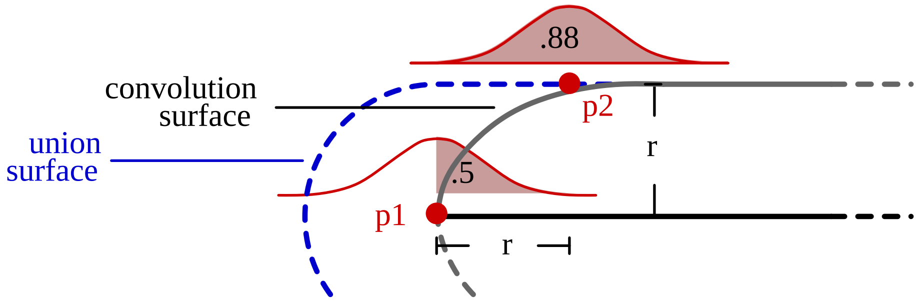

Figure 5.56 End Contours

5.6.12 Point Sources

In sections 5.6.7 and 5.6.9 we observed that the B-spline and Wyvill kernels, unlike the Gaussian kernel, are not separable. In section 5.6.10 we concluded that their similarity to the Gaussian permits these kernels to be treated as separable with reasonable results. To obtain a fully accurate convolution using the B-spline or Wyvill kernels, however, requires the summation of terms defined by individual points distributed along the skeleton, as suggested in section 5.6.3. In other words, the skeletal segments are treated as sets of closely packed points.

This use of point sources is attractive; it is uncomplicated and has been investigated by several researchers. In the literature, point sources are known as ‘atoms’ [Blinn 1982], ‘meta-objects’ [Nishimura *et al*. 1985], and ‘keys’ [Wyvill *et al*. 1986]; the surface that results is known as a ‘blobby molecule,’ ‘fusion cluster,’ or 'soft object,' respectively. We refer to the use of point sources as 'convolved point modeling' and to the resulting surface as a 'blend.' [13] Although a form of convolution, previous literature has not classified this use of point sources as a form of convolution.

The most widely reported filters used for convolved point modeling are the Gaussian and the Wyvill. Although the kernel used in [Nishimura *et al*. 1985] is not widely described in the literature, its use within a modeling system is marketed commercially by Meta Corporation as the 'Meta-Editor' [Graves 1993] Similar technology is marketed by SOFTIMAGE, Inc. and is called 'Meta-Clay'® [Roscoe 1993]. These commercial editors use point and/or ellipsoid sources. Some remarkably complex forms have been produced; unfortunately, we are not able to report here to what extent, if any, models produced by these systems organize their sources according to a skeleton. We believe the convolutional nature of these objects accounts for their versatility and aesthetic appeal.

These methods allow a user to distribute a large number of point sources, each with an associated 'weight,' that is, $f(\boldsymbol{p}) = c - \sum_i w_i h(\|\boldsymbol{p}-\boldsymbol{s}_i\|)$, where $\boldsymbol{s}_i$ are the individual point sources and w_i are the individual weights. These weights may all be the same, but, if we wish the point sources to approximate skeletal segments, it is important that the weight associated with a point source be attenuated by the area under the filter subtended by that part of the segment associated with the point. This technique has been applied to the point sources used to create the following images, which compare results for segments and points.

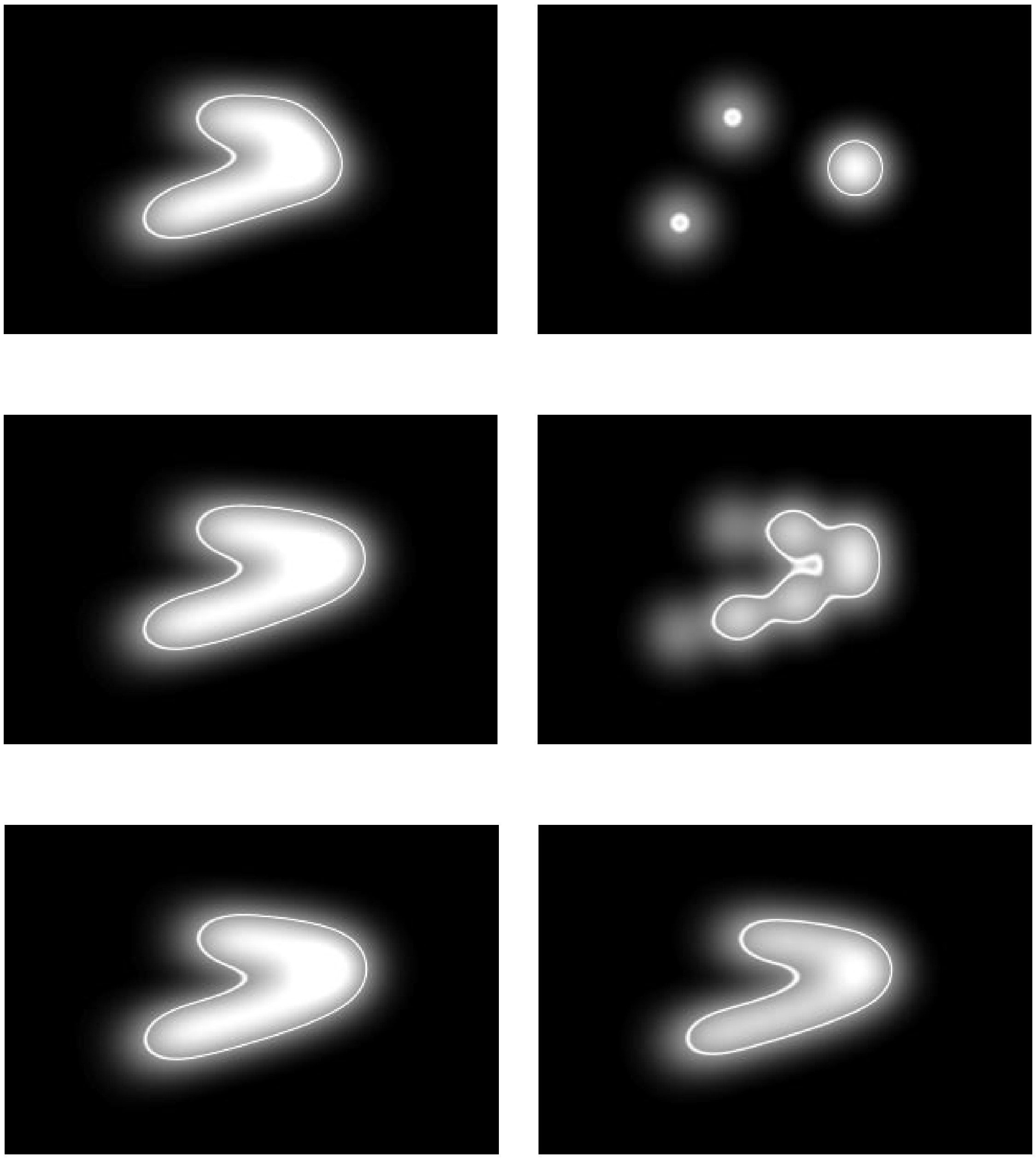

Figure 5.57 Gaussian Convolution for 3, 7, and 15 Segments (left) or Points (right)

5.6.13 Cross-Sections, Revisited

Although a circular cross-section can be achieved directly with the convolution kernel (that is, *h*(*d*/*r*) produces a cylinder of radius *r*), it could also be produced by a two-dimensional *cross-sectional image*. We make this speculation without development, and note that it is difficult to define functions that return distance to an arbitrary cross-section (illustrated below as '*d*'), such as the saucer shape below. We will return to this issue in chapter 7.

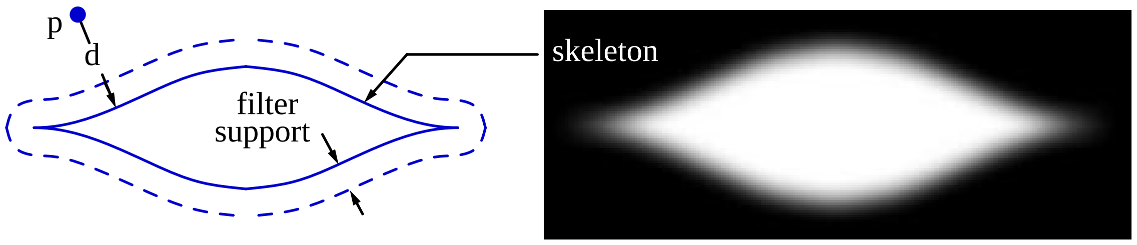

Figure 5.58 Cross-Section Defined by an Image

5.6.14 Ramification

In section 5.3.3 we presented a seven step parametric definition for a ramiform; the implicit counterpart is simpler because the branches constituting the ramiform may be evaluated independently. The disks, silhouette curves, saddle curve, and lofting curves used in the parametric process are not needed by the implicit definition. First, we consider a distance surface defined by a single limb as:

$$(5.35)\quad f_{limb}(\boldsymbol{p}, limb) = \frac{r_{limb}^{\;2}}{\|\boldsymbol{p}-\boldsymbol{q}\|^2},$$

where $\boldsymbol{q}$ is the point on the limb closest to $\boldsymbol{p}$.

One ramiform definition, used to create the surface below, is simply the parent limb function or the summation of the child limb functions, whichever is greater:

$$(5.36)\quad f_{ramiform}(\boldsymbol{p}) = max(f_{limb}(\boldsymbol{p}, parent), \sum_{i}^{n} f_{limb}(\boldsymbol{p}, child_i),$$

where n is the number of children. To maintain continuity of radii, the radius of *parent* is scaled such that in the plane perpendicular to *parent* at its endpoint $f_{limb}(parent) = \Sigma f_{limb}(child_i)$.

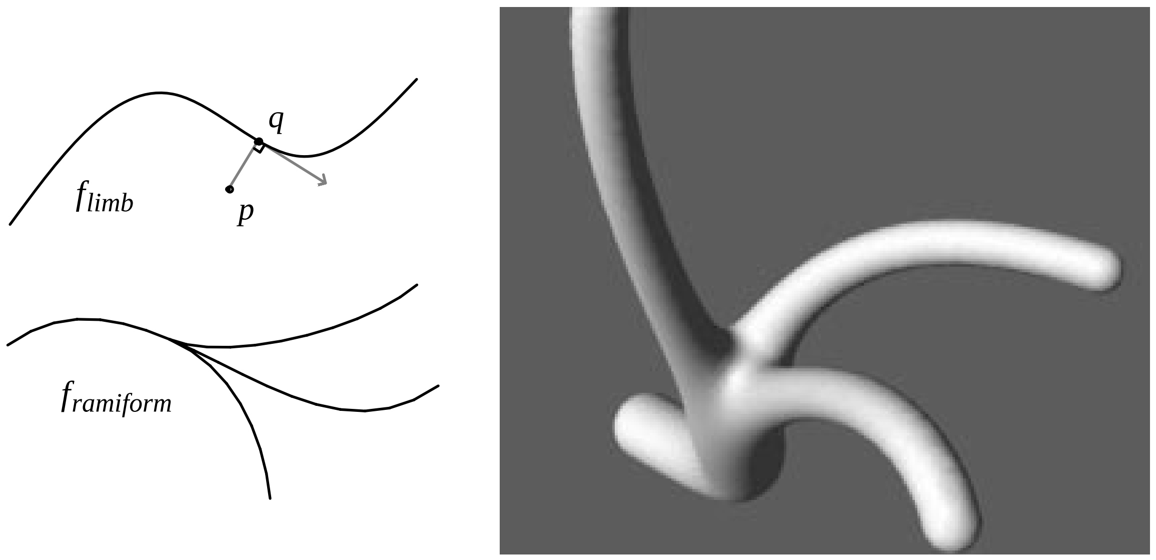

Figure 5.59 A Trifurcated Ramiform

left: skeletons, right: shaded surface

5.6.14.1 An Ad Hoc Method

The summation in the above definition implies that the parent branch will be thicker than the individual child branches, which is not necessarily undesirable. Many organic branching forms are so shaped; often a parent limb requires a larger diameter for greater strength and higher nutritional flow.

A designer may, however, prefer a ramiform with uniform radii along the limbs. In this case, we regard the shape in the above figure as another instance of bulging in implicit modeling. An *ad hoc* solution may be applied in some cases. For example, the two-ramiform below is defined according to the distance from ***p*** to the segment that joins the projections of ***p*** onto the two branches.14 Although this produces a smooth ramiform of constant radii, it does not readily extend to *n*-ramiforms.

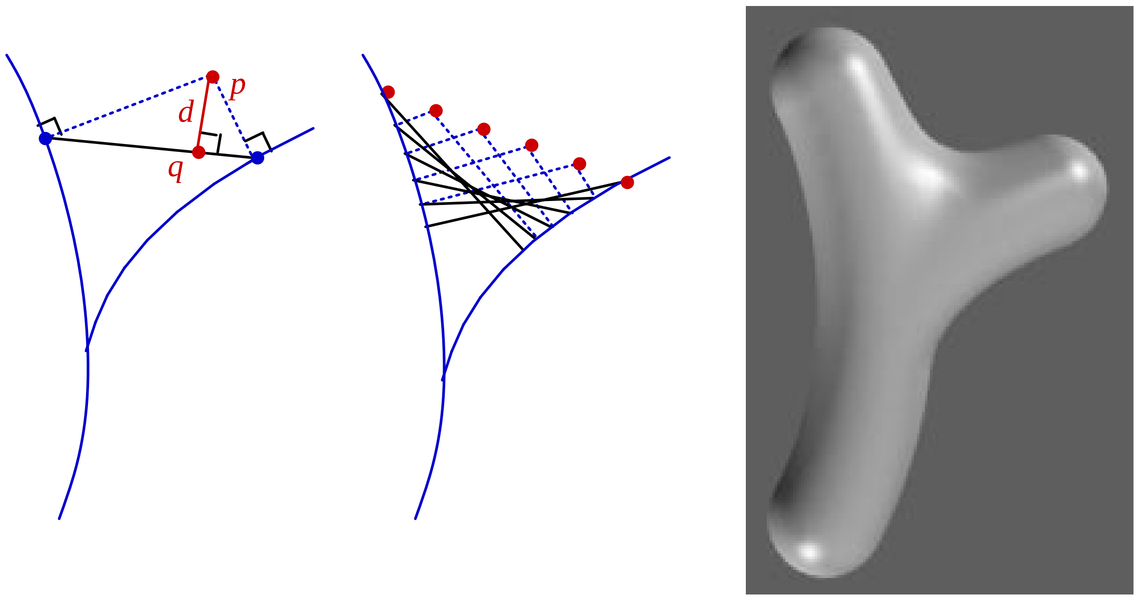

Figure 5.60 A Two-Ramiform with Constant Radii

*left: evaluation of f(**p**), middle: 'envelope' curve, right: shaded surface*

In this example, parent and child limbs are contiguous and non-overlapping. A different approach is the 'strand' model, in which all limbs originate at the root [Holton 1994]. The difference is shown below. Taken to its logical extension in the modeling of organic forms, the strand model produces highly realistic shapes [Greene 1991]. Numerous strands are required, however; if they are treated by convolution, efficient methods such as those presented in this chapter are relevant.

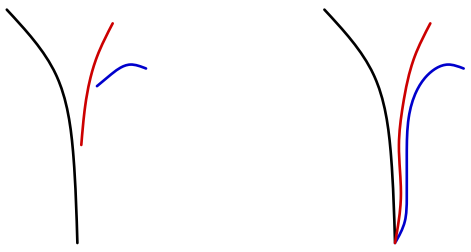

Figure 5.61 Two Ramiform Skeletons

left: contiguous non-overlapping skeleton (limbs shown separated for clarity)

right: strand skeleton

5.6.14.2 Bulges

We now consider whether convolution offers a solution to the problem of ramiform bulge. We first examine the simple ‘tee’ skeleton, below, which consists of two segments. Each cylinder has radius *r*.

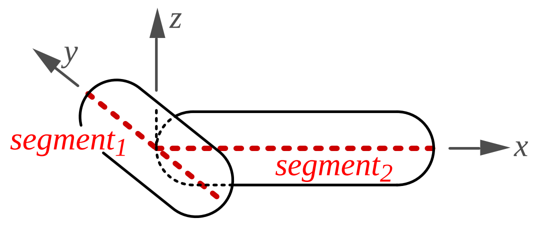

Figure 5.62 A ‘Tee’ Skeleton

Application of a Gaussian filter yields the following contours, shown in the *xy* and *xz* planes (dashed lines demarcate the isolated cylindrical primitives). Slight bulges at the segment junction are apparent in both planes. This problem was not considered in [Bloomenthal and Shoemake 1991]; in that study, attention focused on the use of polygons, not line segments, as skeletal elements. We will discuss such use in the next chapter. We mention now, however, that the lack of bulging in [*ibid.*] was a consequence of the ratio of polygon width to kernel support exceeding one. In this chapter we examine the bulge problem more closely than in the previous study.

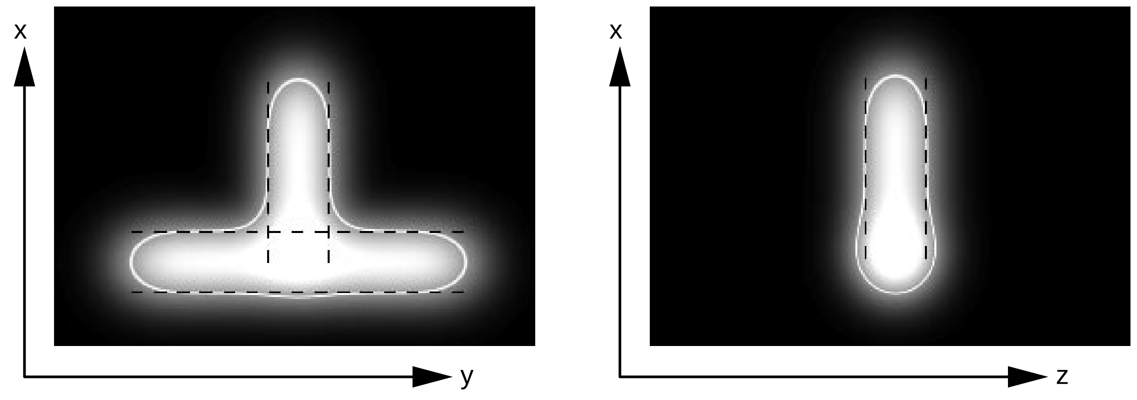

Figure 5.63 Contours for the ‘Tee’

To understand better the phenomenon, let us consider the behavior of $f(\boldsymbol{p})$ for points $\boldsymbol{p} = (x, 0, r)$, shown below, left. At $x = 0$, $\boldsymbol{p}$ is directly above the junction of the two segments and is r distant from both. Let h_1 and h_2 refer to the Gaussian distance filters applied to *segment*$_1$ and *segment*$_2$, respectively; let g_1 and g_2 refer to their respective integration filters. For *segment*$_1$, $h_1(r/r) = ½$ and $g_1 = 1$; for *segment*$_2$, $h_2(r/r) = ½$ and $g_2 = ½$; thus, $f(\boldsymbol{p}) = h_1g_1+h_2g_2–½ = (½)(1)+(½)(½)–½ = ¼$. As x increases and $\boldsymbol{p}$ moves along the $z = r$ line, g_1 remains 1 and h_2 remains ½, but g_2 and h_1 change. This is graphed in the figure below, right, and predicts the bulge at the junction of the two segments.

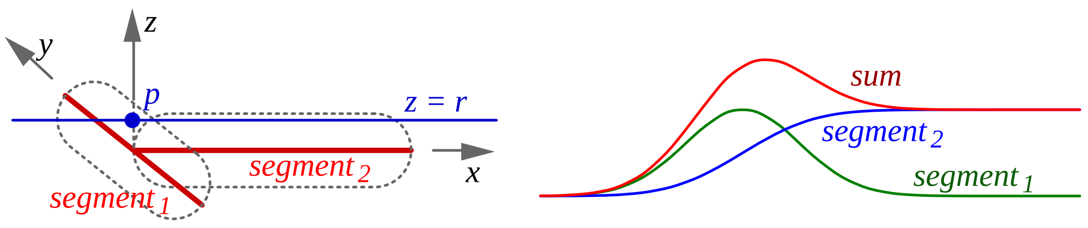

Figure 5.64 'Tee' Junction with Segment$_1$ and Segment$_2$ Touching

left: geometry, right: h_1g_1, h_2g_2 and $h_1g_1+h_2g_2$ along (x, 0, r)

5.6.14.3 Bulge Reduction

If we move the left endpoint of *segment*$_2$ to the right by r, as shown below, left, then at $(r, 0, r)$, $h_1 = h(\sqrt{2}) = ¼$, $g_1 = 1$, $h_2 = ½$, $g_2 = ½$, and $f(\boldsymbol{p})$ is, as desired, 0. Unfortunately, the summation along $(x, 0, r)$ is not constant, as shown below, right. A small bulge and a small dip occur near the junction. This is to be expected because h and g are not identically shaped; for example, the integral requires twice the filter support (*i.e.*, the full width of the kernel) to achieve its maximal value whereas the kernel itself achieves a maximal value within the filter support.

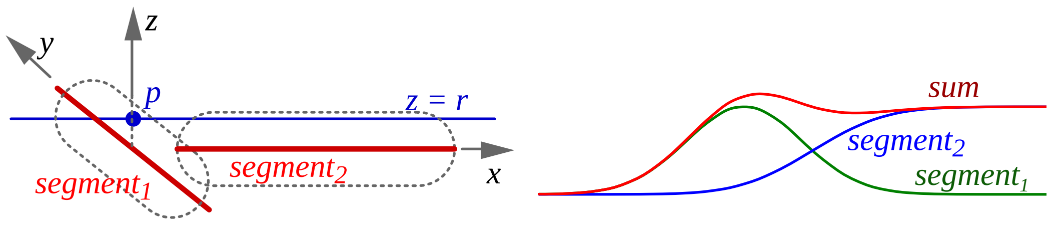

Figure 5.65 'Tee' Junction with Left Endpoint of Segment$_2$ Moved Amount r

left: geometry, right: h_1g_1, h_2g_2 and $h_1g_1+h_2g_2$ along (x, 0, r)

Shortening *segment$_2$* does, however, improve the appearance of the contours shown below. Remarkably, the infinite support of the Gaussian does not produce a significant bulge along the bottom edge of the horizontal segment; this is because points on this edge are $d \geq 2r$ from the lower segment and the Gaussian has very low energy for $d/r > 2$.

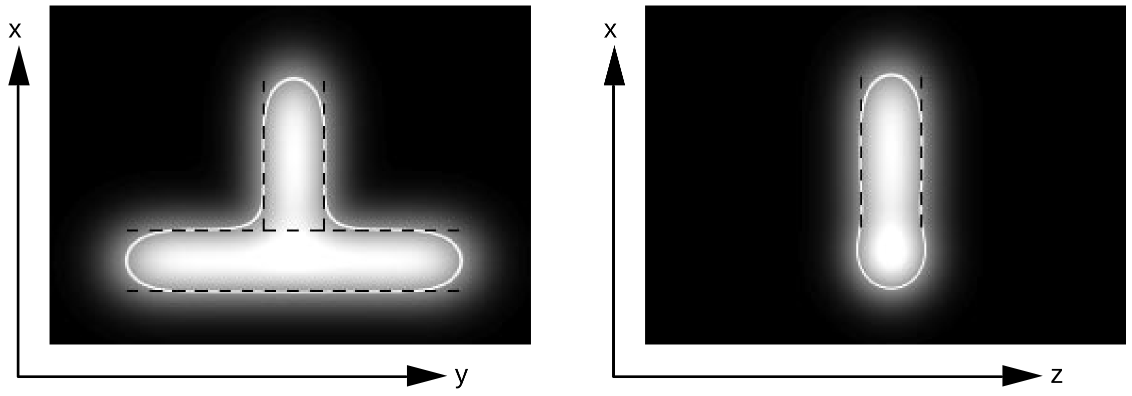

Figure 5.66 Contours for the 'Tee' with Shortened Segment$_2$

When sampled in three-dimensions, the convolution produces the surface shown below.

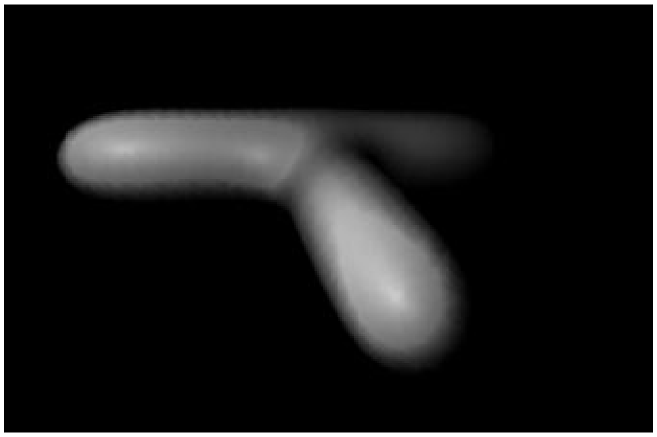

Figure 5.67 ‘Tee’ Surface with Segment$_2$ Shortened

The bulge and dip predicted by the mathematical model are visible, although sufficiently subtle for this surface to be reasonable in animation. In the following illustration we pass the vertical segment through the horizontal one. Separation of the surface into two isolated cylinders is evident in the last (lower right) frame.

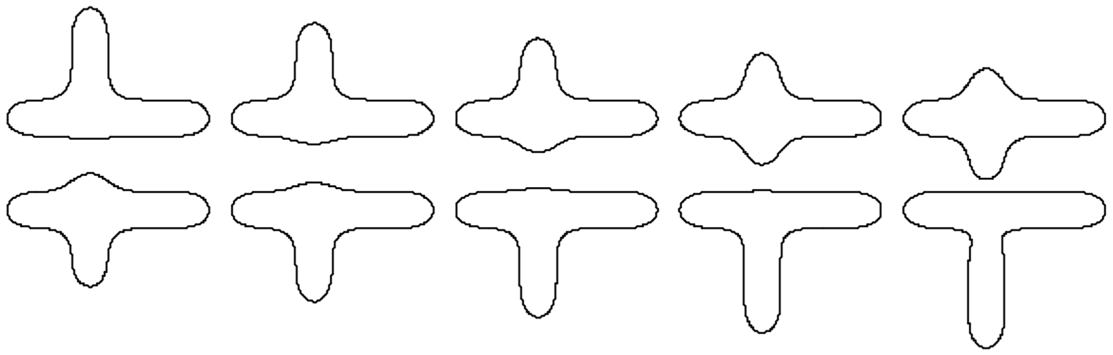

Figure 5.68 Animation from ‘Tee’ to Cross

5.6.14.4 Other Attempts

One approach to eliminate the bulge and dip is to taper the radius associated with the second segment by the inverse of *sum*, as is graphed in figure 5.6.5. Two taperings are illustrated below and are not unlike methods in structural engineering. Although this can eliminate the bulge in the *xz*-plane, it will introduce a pinch along the sides of the ‘tee,’ in the *xy*-plane.

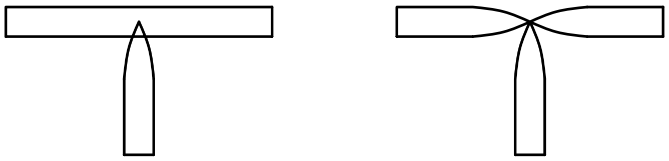

Figure 5.69 Two Methods of Tapered Junction

Tapering is an attempt to compensate for the increased skeletal density near the junction of the segments, as it is this increase that produces the bulge. We might speculate that a distribution of point sources would avoid this increased density and eliminate the bulge. As the following test shows, this speculation is upheld for a small number of points. A small number, however, yields a lumpy appearance. As the number of points increases, the lumpiness disappears but the resulting surface converges to that defined by segments and, once again, the bulge appears.

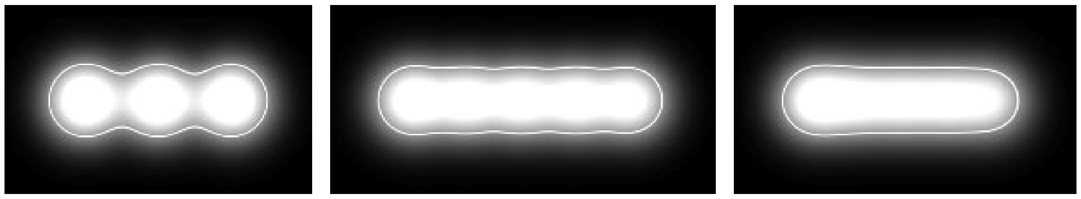

Figure 5.70 View of ‘Tee’ in xz-Plane

left: 7 points, middle: 9 points, right: 11 points

5.6.14.5 The Combination Surface

It would appear, therefore, that we must convolve when ***p*** is within the plane of the junction and avoid a bulge by not convolving when ***p*** is out of the plane. In effect, we enforce a union operation to the extent to which ***p*** is out of the plane of the ‘tee.’ This is reminiscent of equation 5.15 and is readily implemented for a skeleton; we simply associate an approximating plane with each skeletal joint. Convexity can be measured as the angle between the plane normal and the vector from the joint to the point ***p***. Thus, the implicit surface function becomes:

(5.37) $f_{combination} = convolutionValue + Convexity\,(unionValue - convolutionValue)$

Although we have used a ‘tee’ for demonstration in previous sections, we now employ a five-ramiform, which is more demanding with respect to bulge avoidance. The union surface and convolution surface are compared below.

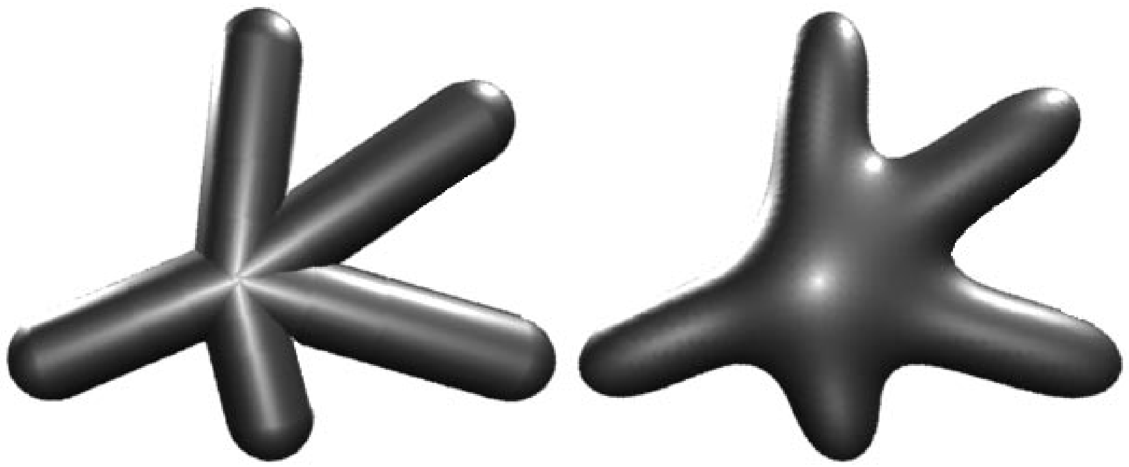

Figure 5.71 Union Surface (left) and Convolution Surface (right)

In the following figure we display the combination surface as a line drawing in order to demonstrate that the surface is free from bulges.

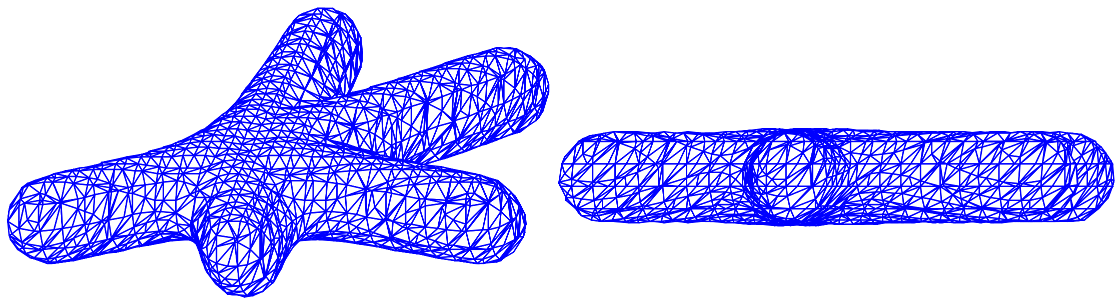

Figure 5.72 Line Drawings of Combination Surface

left: tilted view, right: profile

The webbing that blends together pairs of limbs will, necessarily, contain a small crease, as can be seen in the interpolated contours of figure 5.47. This does not appear objectionable in the following image, but may be objectionable on theoretical grounds. We return to the issue of bulges and creases in chapter 6. A pseudo-code implementation for the combination surface is given in the appendix.

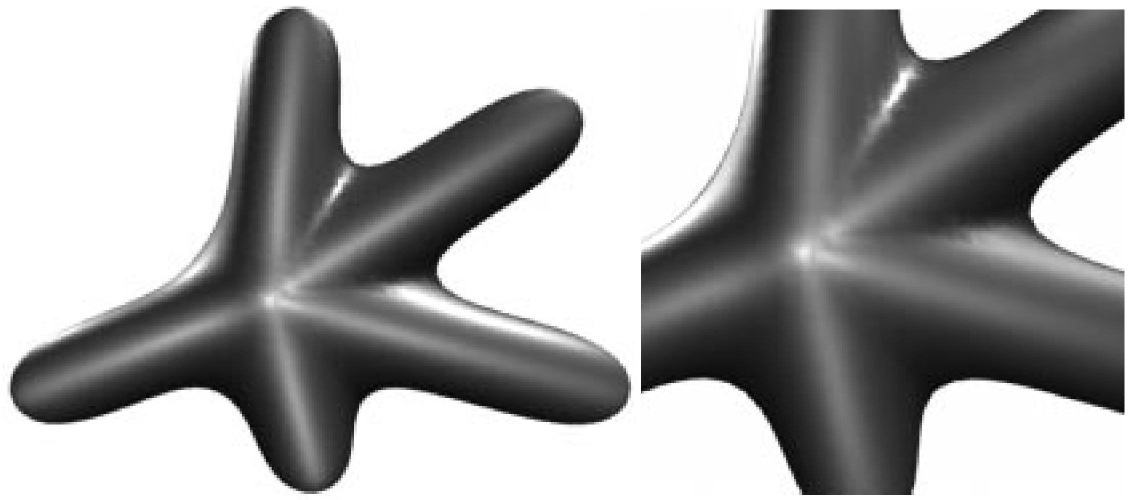

Figure 5.73 Combination Surface (left) and Closeup (right)

Many renderers capable of smooth (Gouraud or Phong) shading will estimate vertex normals if provided a polygonal description of the shape. Some renderers, however, produce improved images if given precise vertex normals. Special care is required when computing a vertex normal for the combination surface. It cannot simply be a (unit length) linear combination of the union vertex normal and the convolution vertex normal because the union vertex normal is discontinuous. Even a minor contribution from the union vertex normal can exaggerate the appearance of a crease in the surface. Conversely, if we utilize the convolution vertex normal only, the shaded image will evidence a bulge, even though the surface is free from bulges. Vertex normals for the combination surface were computed according to details provided in the appendix. They were used to render the shaded images of figure 5.73. The sensitivity of renderers to numerical instability of vertex normals accounts for the noisy highlights along the webbing blending the upper and upper-right limbs.

The combination surface can be modified in several ways; we may, for example, experiment with several of the functions involved in the model. By modifying the function that interpolates the union and convolution surfaces, we can alter the apparent 'hardness' of the model. By modifying the convexity function, upon which the interpolation function depends, we can further alter the characteristics of the blend. For example, rather than fitting a plane to the vertices of a joint, a free-form surface could be fit; thus, instead of the plane normal, the normal of the free-form surface at the point of proximity could be used to compute the convexity at ***p*** for points in the neighborhood of the joint.

The design system should provide the user with the option to associate with a joint the direction of convexity, *i.e.*, the direction in which no blending occurs. In figure 5.73 this direction is normal to the plane of the skeleton. Should the designer decline this option, the skeletal design system should compute reasonable default direction(s) for convexity. In particular, it would be reasonable if the default directions, shown as arrows in the following illustration, were automatically generated. In this scheme, convexity would be measured as $\max_i (\boldsymbol{v} \cdot \boldsymbol{d}_i)$ where ***v*** is the vector from the point ***p*** to the joint and $\boldsymbol{d}_i$ are the directions of convexity, either automatically generated or as specified by the designer.

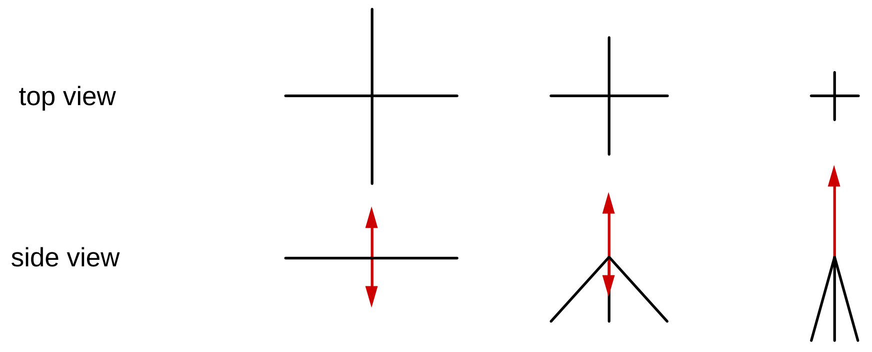

Figure 5.74 Convexity Directions for Various Joint Configurations

5.7 Texture Coordinates

In this section, we discuss the assignment of parametric (‘texture’) coordinates to vertices of the implicit surface. This form of parameterization is important in the generation of images with realistic surface texture in the absence of a geometric model of surface microstructure. *Bump mapping* and surface *texture mapping* both rely upon surface parameterization. For parametric surfaces, an obvious parameterization is trivial to construct, yet may not be the desired one. A ‘natural’ parameterization that follows the skeleton or the surface is usually desired, and is often difficult to construct for both implicit and parametric surfaces.

Surface coloring and *solid texture* are alternatives to texture coordinates. Surface coloring presumes a sufficiently dense number of vertices to establish surface texture by interpolating vertex color across a polygon, rather than indexing pixel color according to a texture map. This approach has been successfully applied in a texturing process called *reaction-diffusion* [Turk 1991], [Witkin and Kass 1991]. Solid texture is an approach that colors a surface vertex according to its position within three-space; this approach was demonstrated in [Perlin 1985] and [Peachey 1985], and first applied to implicit surfaces in [Wyvill *et al*. 1987].

Solid texture differs from surface texture in that the former produces an object that appears to be carved from a material and the latter produces an object that appears to be covered by a material. Natural, living forms generally appear to be covered, rather than carved; thus, we do not further consider solid texture. Although surface coloring appears applicable to many natural forms,[15] we choose in this chapter to emphasize texture mapping, as it is a widely applied concept in computer graphics and is suitable for a sparse set of vertices.

We assign texture coordinates according to the relation between surface vertices and the defining skeleton. One obvious relation is length along the skeletal curve; another relation derives from the circumferential location of a vertex with respect to a reference frame. Pursuing this approach with the ramiform illustrated below, we assign texture as a (*u*, *v*) ordered pair, with *u* indicating circumferential distance and *v* indicating axial distance with respect to the skeletal curves. With the initial *u*-coordinate assignments shown, the two branches and the parent agree in their *u*-values for 0 and ½. Remaining texture coordinates are assigned via a transformation of a surface vertex to the local *xy*-plane defined by the reference frame at the nearest point on the skeletal curves.

A difficulty arises along the longitudinal lines (shown as dashed-bold) connecting the parent and child disks at *u*-value of 0. In order that change in texture agree between parent and child, the lines of *u* = 0 must be local minima (the agreement in change in texture is indicated by four arrows indicating increasing *u*). Unfortunately, vertices on opposite sides of a *u* = 0 line may have the same *u*-coordinate, which would produce a 'stretch' in the apparent surface texture. This stretch is indeed visible in the shaded image below.

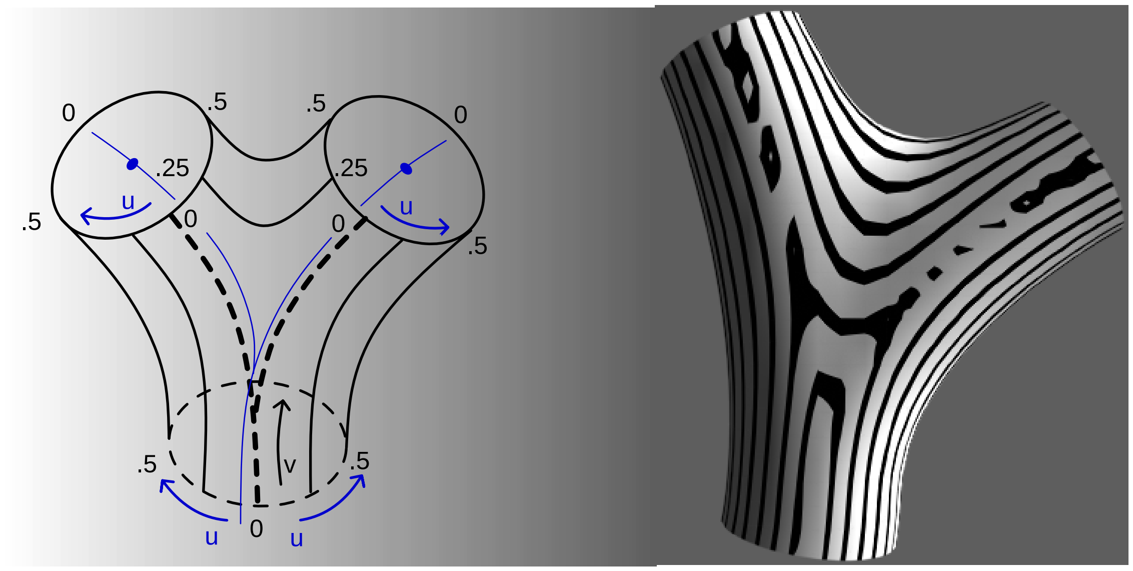

Figure 5.75 Texture Coordinates for an Implicitly Defined Ramiform

Stretching also occurs in the roughly triangular region in the center of the ramiform (between u = 0 and u = ½). If we think of this as an 'island' and the lines of constant u as topographic contours, then, intuitively, we wish to create a 'hill' on the island. This can be done by increasing the u-value with proximity to the center of the island. We define this center as a point midway along the u = ¼ line. The results of this compensation are shown in the image below.

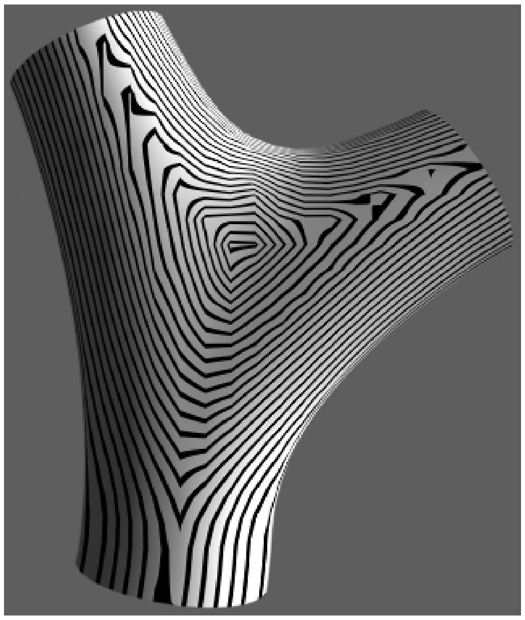

Figure 5.76 Modified Texture for an Implicitly Defined Ramiform

A criterion for judging this image is the evenness of texture spacing. In particular, we attempted to assign texture coordinates so that the texture distance between two vertices is proportional to their geometric distance. For non-developable surfaces [Faux and Pratt 1979] some variation in the proportion becomes necessary; for example, degeneracies occur at the poles of a sphere. We have not pursued this further, but refer the reader to related work [Gagalowicz 1985].

Although in this example we have been able to assign texture coordinates according to the skeleton that defines the surface, singularities are evident. The textured ramiform is a problem not yet solved, and surface coloring methods, such as reaction-diffusion, are promising.

5.8 Conclusions

In this chapter we have examined methods for constructing generalized cylinders and ramiforms, attempting to represent cylindrical or branching objects that are common to organic forms. We emphasized on implicit techniques and, in particular, focused on the convolution surface, an implicitly defined summation of primitives that each depend on an individual skeletal element.

Because convolution is an integration, it produces a material blend of primitive volumes; the resulting object appears pliable. The object is not amorphous, however, because blending occurs only where limbs are mutually close; the remainder of the surface closely follows the skeletal structure.

In comparison, the strict distance-to-curve implicit method produces an object that appears rigid and easily creased, much as would be produced using standard parametric sweep techniques. We noted that, unlike implicit techniques, parametric techniques can produce self-intersecting surfaces, require reference frames, and experience difficulty at branches. Reference frames do, however, allow cylindrical cross-sections to vary in shape and orientation; they can also play a role in the assignment of texture coordinates, which we examined for an implicitly defined ramiform.

We considered other approaches to cylinders and ramiforms, including patches and subdivision surfaces, but concluded that implicit surfaces were easier to implement, could more readily accommodate *n*-ramiforms, and were more conveniently defined by a skeleton. In the context of implicit surfaces, we considered piecewise approximations to curved skeletons.

In addition to convolution, we considered other implicit blends, including global blends, the rolling-ball blend, and range-controlled blends. We found convolution more attractive either in implementation or in extensibility to the *n*-ramiform. Additional study is needed to understand fully the relative advantages and disadvantages of convolution and other implicit blends. For example, the difficulty in extending the range-controlled blend to multiple primitives and the difficulty in preventing undesired fillets in the rolling-ball blend may be due to the reliance of these blends on a single distance measurement to each skeletal element. This differs from convolution, which involves an integration.

In discussing convolution, we considered its superposition property and the separability property of the Gaussian convolution kernel. We compared the Gaussian with other kernels, and developed techniques to calculate its integral and reduce the three-dimensional convolution of a curve to a product of lower-dimensional terms. These methods were developed with an interest in processing efficiency. This efficiency could be useful for representations such as the strand model, which may be biologically realistic but is computationally demanding.

We considered the problem of bulges in implicit surfaces, and concluded that a combination of distance surface and convolution surface yields a reasonable result. We suggested some user control over the bulge-free orientation may be desirable, and that the designer may wish to control the extent to which a blend is bulge-free.

Because there remain questions concerning non-bulge, *n*-ramiforms, we note that a slight bulge is not necessarily a serious, visually objectionable artifact. Indeed, many natural forms, such as the joining of veins in the hand or the leaf, do produce a slight bulge. There may be common biological processes that result in bulges at the junction of individual vascular and muscular elements. To the extent that this is true, the convolution method applied to skeletons mimics natural form. Overall, we believe that convolution of a skeleton remains an excellent method for modeling natural forms.

One promising direction for future research is the convolution of volumes, rather than the volume-less segments employed in this chapter and the volume-less polygons employed in the next chapter. This is precisely the approach taken in [Colburn 1990]. Unfortunately, the computational expense of this approach may be prohibitive within a design environment. One approximation to volumetric convolution is the strand application of convolution to a skeleton.

We have not considered combinations of creases and bulges. For example, a crease may form between two bulging shapes in mutual contact. Some interesting possibilities to mimic this natural phenomena may be found in [Gascuel 1993].

In the following two chapters, we will apply the concepts of this chapter to two organic forms: a human hand and a maple leaf.

5.9 Notes

1. The reference frame has an additional use, however, in the computation of surface texture coordinates, described in section 5.7.

2. One exception would be $f(\boldsymbol{p}) = x^2 - y^2 + z^2$, two cones that meet at the origin. In this example, the origin is an exceptional point; at the origin the surface is not homomorphic to a two-dimensional disk. Another exception would be the set of all points, $f(\boldsymbol{p}) = 0$.

3. We approximate the arc-length of a cubic curve by summation of the lengths of 100 segments approximating the curve.

4. Related terms are *bevel* (an inclined but flat surface) and *chamfer* (a furrow or bevel).

5. It would be interesting to consider implementing the rolling-ball blend as a procedural implicit surface function.

6. Throughout this chapter we use *P* to mean an implicit primitive, *i.e.*, an implicit surface function for a component of the complete surface; we use ***p*** to mean an arbitrary three-dimensional point that is argument to an implicit surface function.

7. We have not encountered a formal definition of 'bulge,' and so we propose the following. A 'surface bulge' has a cross-section that exhibits negative, then positive, then negative curvature with respect to the underlying skeleton.

8. As mentioned in chapter 3, however, a tree of Boolean set operation may provide a useful abstraction for a geometric shape.

9. For a discussion of properties of the Gaussian, see [Heckbert 1985].

10. This is also important for higher order primitives, such as the two-dimensional primitives used in the next chapter; the volume of the kernel must be normalized to one so as to maintain the overall energy in the two-dimensional image.

11. A three-dimensional B-spline could be constructed as a tensor product of three univariate B-spline basis functions. Although this is separable, it is not spherically symmetric, which is necessary for our purposes.

12. The Wyvill and Gaussian filters, which are the most reported in the literature, are usually applied to point sources (see [Blinn 1982] and [Wyvill *et al.* 1986]).

13. We dislike the popular term 'blobby' because it suggests something amorphous and without structure. The objective of this disseration, however, is to promote the design of structured shapes; their smoothness is not a lack of structure.

14. The family of segments in figure 5.60 produces an ‘envelope curve,’ a curve tangential to each of the segments [Faux and Pratt, 1979]. This curve is apparent when several positions of ***p*** are examined, as in figure 5.60, middle. The hyperbola, below left, is an envelope curve produced by lines connecting evenly spaced points along two axes. At right is a set of planar measurements evaluating the function described in section 5.6.14.1. As noted in section 3.7, a planar slice is a useful interactive technique. From the smooth gradations, one can imagine the shape gracefully expanding or contracting, depending on the iso-contour value.

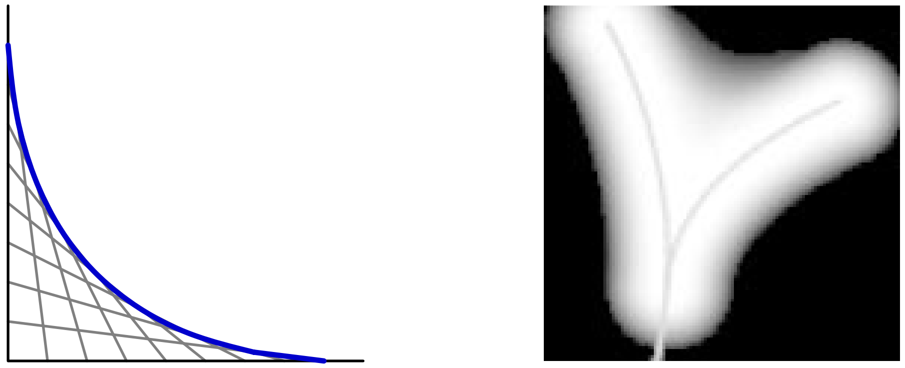

Figure 5.77 An Envelope Curve

15. Indeed, one of the first examinations of this subject concerned patterns found on the coats of animals [Turing 1952].

Woodman, spare that tree!

Touch not a single bough!

In youth it sheltered me,

And I’ll protect it now.

(George Pope Moris)