Chapter 6: Hand Crafting

To see a world in a grain of sand
And a heaven in a wild flower,
Hold infinity in the palm of your hand
And eternity in an hour.
(William Blake)

In this chapter we model a biological form, the human hand. It is the most complex body part, and will exercise and extend several of the methods we have discussed. Despite its complexity, the design of the hand required only a few hours, as explained in section 2.7. We utilized a skeleton derived directly from figure 2.4, right. It consists of a set of simple elements for the palm and fingers, with additional elements for tendons and veins. Although the resulting surface is smooth, we suggest methods for the addition of non-smooth features such as creases and fingernails.

The human hand is not a simple geometric union of its component fingers and palm. The skin that covers muscle and bone yields a blend of these components, which we believe can be represented by a convolution surface. To accommodate the relative flatness of the palm and the oblong cross-section of the fingers, we extend convolution from the one-dimensional skeletal elements described in the previous chapter to support two-dimensional skeletal elements, such as polygons.

6.1 Two Dimensional Skeletal Elements

As with one-dimensional skeletal elements, the convolution surface for two-dimensional skeletal elements is evaluated as the sum of independent primitives. We continue to assume the convolution filter is spherically symmetric so that the

convolution is independent of polygon orientation. And we assume the kernel is separable into the product of lower dimensional terms, namely, an 'integration filter' in the plane of the polygon and a 'distance filter' perpendicular to the plane. For a polygon lying in the *xy*-plane, we rewrite equation 5.25 as:

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

$$c - \int_{\Re^2} e^{-(\boldsymbol{p}_{xy}-\boldsymbol{u}_{xy})^2/2}\, d\boldsymbol{u}_{xy}\, e^{-\boldsymbol{p}_z{}^2/2}.$$

The distance filter is the filter kernel evaluated according to distance from $\boldsymbol{p}$ to the *xy*-plane; the integration filter is the two-dimensional convolution of the polygon in the *xy*-plane. Unlike the single, one-dimensional array required to integrate a line segment, a two-dimensional array is required for each polygon.

Figure 6.1 illustrates the convolution process. $\boldsymbol{p}_{xy}$ is the projection of $\boldsymbol{p}$ onto the plane of the polygon, and I is the image of the filtered polygon, *i.e*, the integration filter, which we will call the 'convolution image.' That is,

(6.2) $$f(\boldsymbol{p}) = c - h(d)\, I(\boldsymbol{p}_{xy}).$$

For several arbitrarily oriented polygons, each represented by image I_i, this is equivalent to:

(6.3) $$k\Sigma_i\, h(\boldsymbol{q}_z)\, I_i(\boldsymbol{q}_{xy})$$

where k is a constant (typically the reciprocal of the maximum image value) and $\boldsymbol{q}$ is $\boldsymbol{p}$ mapped into the coordinate system for I_i. Details concerning this mapping are given in the appendix. In practice, I is a bilinear interpolation of four samples; it is set to zero if the projection of $\boldsymbol{q}$ is beyond the bounds of the image. These bounds must accommodate the polygon plus the kernel support; otherwise, parts of the kernel domain are undefined during convolution beyond the edge but within the influence of the polygon.

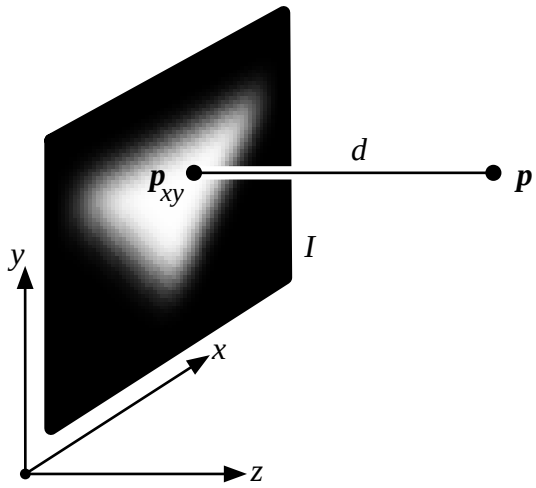

Figure 6.1 Computation of f(p)

Although there exist analytical methods for convolution of a two-dimensional polygon [Duff 1989], we employ conventional, numerical methods, such as those used for image processing. These work well and create computationally efficient, two-dimensional arrays. In [Bloomenthal and Shoemake 1991] the polygon was rendered on an image raster using a solid color, and then filtered with an appropriate two-dimensional kernel. Resulting values were stored as eight-bit integers. This quantization can introduce discretization artifacts that can be overcome by the use of a real-valued array.

The Gaussian kernel has infinite support, which complicates the production of a discrete representation. This was overcome in the one-dimensional case by use of a large, one-dimensional array.[1] For two-dimensional arrays, however, we must accept a lower resolution in view of the $O(n^2)$ demand on memory. In the examples presented in this chapter, the resolution provided for the filter support is scaled according to the desired object thickness and polygon raster resolution. For example, given a square of side length ½, a desired thickness of 1/10, and a target raster width of 200, the filter support should be 200(0.1/0.5) = 40 pixels.

The shape shown below was defined by a skeleton consisting of a trapezoid and a

rectangle, each producing a convolution image.

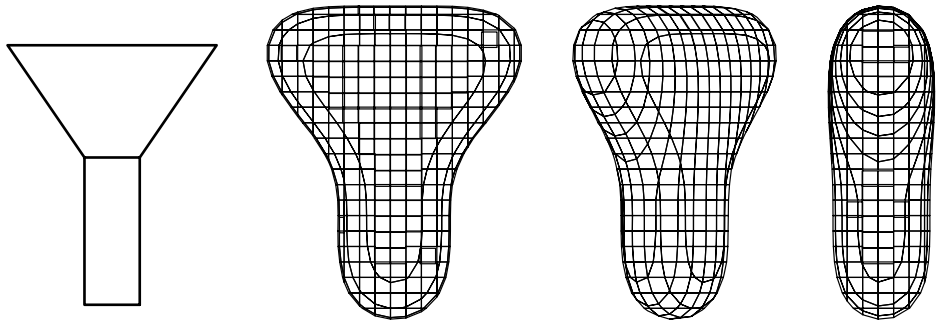

Figure 6.2 Skeleton and Resulting Shape (three orientations)

The image resolution used above was 80 by 80, which appears sufficient for this simple skeleton. In the following figure we degrade the image resolution. Serious discretization artifacts, such as scale changes and shifts in vertex locations, can be seen for resolutions less than 40 by 40.

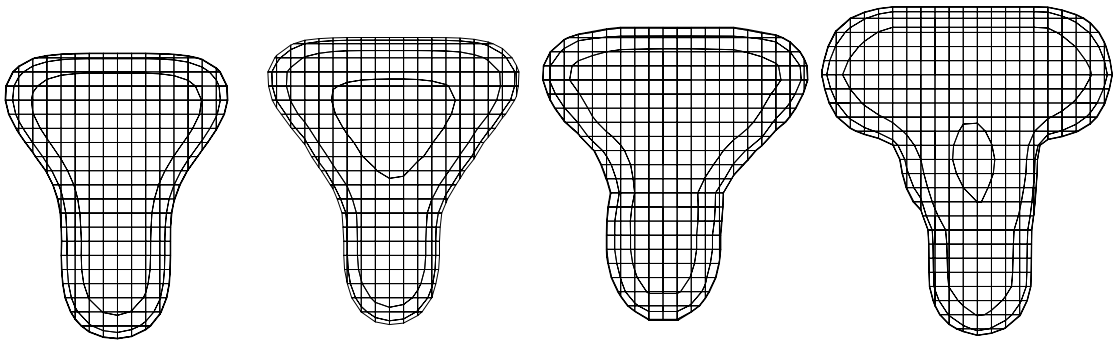

Figure 6.3 Effect of Image Resolution

left to right: 40 by 40, 20 by 20, 10 by 10, and 5 by 5 array size

Each evaluation of $f(\boldsymbol{p})$ requires approximately 35 floating point operations (approximately 6 for the filter, 11 for bilinear interpolation, 6 for distance, and 12 for projection), which compares favorably to the inside-polygon test required for distance surfaces. A two-dimensional convolution image requires significant computation, but need be calculated only once per skeletal polygon. The designer

is free to polygonize the implicit surface at different resolutions or change the orientation of the skeletal elements, without recomputing the convolution images.

6.2 Skeletal Contiguity, Cross-Sections, and Bulge and Crease Prevention

Towards the end of chapter 5 we expressed some dissatisfaction concerning the implementation complexity and geometric quality of the combination surface developed to prevent bulges that result from ramified skeletons. As observed in section 5.6.14.4, bulging is a consequence of increased skeletal density near the ramiform junction. In particular, the line segments of the starfish all meet at one point. If the skeleton were a simple cross, we could think of it not as four segments meeting at a point, but as two intersecting lines. Clearly, the skeletal density increases where the lines overlap. A comparable polygonal skeleton would consist of two overlapping polygons, as shown below, left. We speculate that a contiguous (*i.e.*, abutting and non-overlapping) skeleton, such as below, right, would alleviate the bulge. To prove this, and to explain the conditions under which it is true, we must examine the cross-section of a convolution surface derived from a polygon.

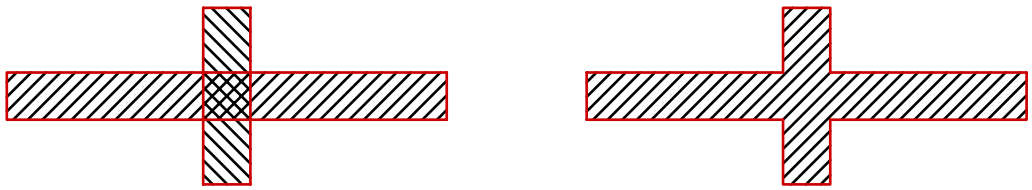

Figure 6.4 Overlapped (left) and Contiguous (right) Skeletons

As noted in section 5.6.11, the convolution surface of a line segment passes through its endpoints but does not produce a perfect hemisphere. Similarly, the surface passes through the edge of a polygonal skeletal element but, as shown below, does not yield a truly semi-circular contour. As shown below, different filter kernels require different polygon widths to produce equally thick surfaces. This is because, as defined in section 5.6.9, the domains differ and the integrals must be scaled to yield unit area over their domains. The Wyvill yields a near-

circular cross-section, which is an important design consideration.

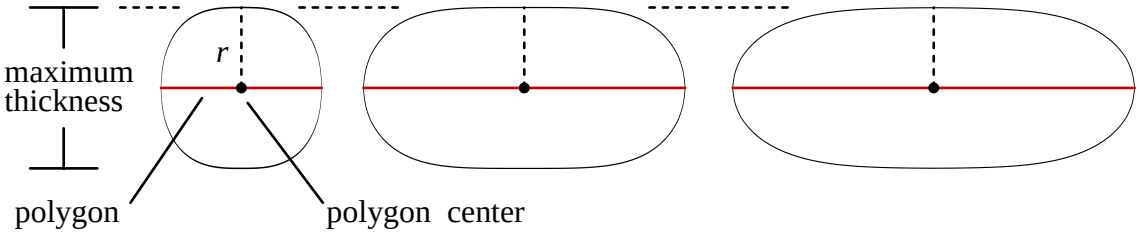

Figure 6.5 Surface Cross-Sections for Different Kernels

left: Wyvill (width = 2r), middle: B-Spline (4r), right: Gaussian (5r)

For each of the filters shown above, the polygon width equals the effective support of the filter. This means, at a point above the center of the polygon, the integration filter yields 1. A point *r* distant above the polygon center yields a distance filter value of ½; thus, the point is on the surface. Now, consider widening the polygon; this will not change the function value for points that were above the polygon center, but it will create a 'plateau' along the top and bottom of the surface, as shown below, left. If we narrow the polygon, however, there is no point over which the integration filter is 1. This reduces the thickness of the surface, so that the surface is less than *r* from the polygon, as shown below, right.

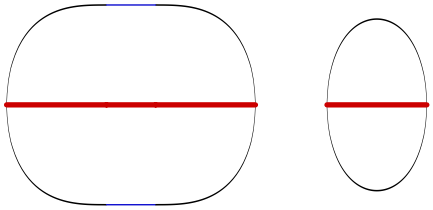

Figure 6.6 Varying Polygon Width (Wyvill Kernel)

left: wider polygon widens the surface, right: narrower polygon reduces thickness

Now, consider the contiguous skeletons, shown below. For a polygon whose width equals the support of the filter kernel, the entire filter projects onto the polygon, and the very centers of the polygons will yield an integration filter value of 1. For the wider polygon, the central region where the integration filter is 1 is larger,

creating a plateau. But for the narrower polygon, only at the junction center is there sufficient room for the integration filter to reach 1; elsewhere the kernel is clipped and the integration filter is less than 1. A bulge will occur at the center of the junction of the narrow polygons, but not with the wider polygons.

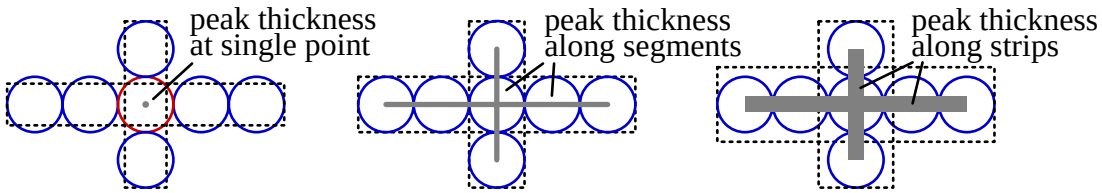

Figure 6.7 Affect of Varying Polygon Width on Integration Filter

left: polygon width < 2r, middle: polygon width = 2r, right: polygon width > 2r

Thus, for an appropriate choice of polygon width, a contiguous polygonal skeleton can yield ramiforms without bulge or crease. The cross-sections of surfaces are not perfectly circular, however, unlike those for line segments.[2] In the illustration below, the above skeleton is articulated.

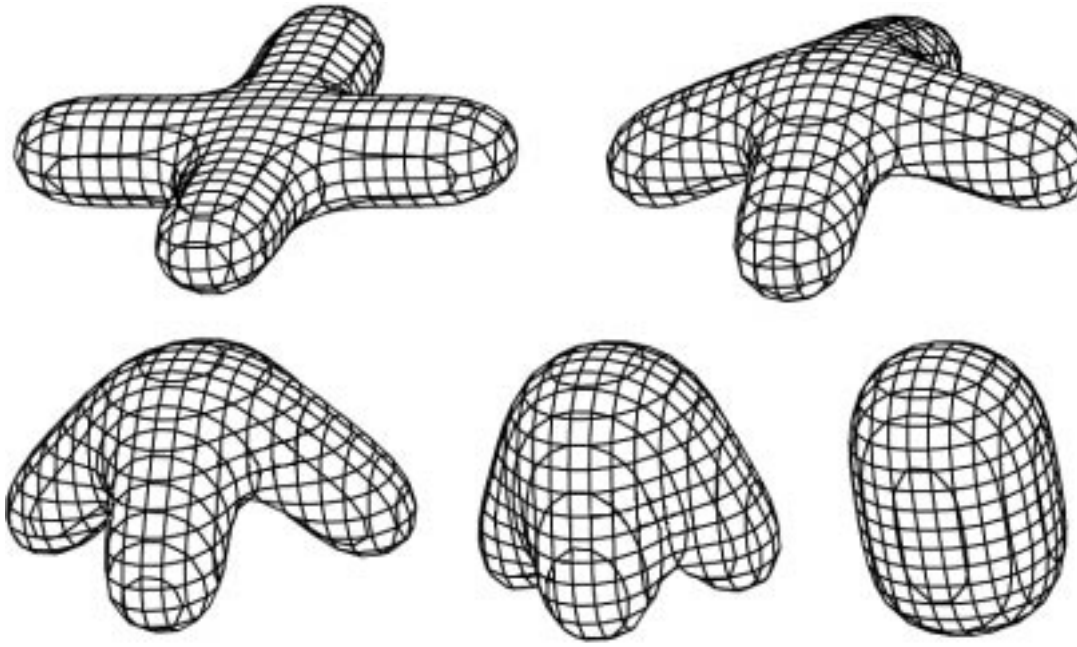

Figure 6.8 A Smoothly Folding, Bulge-Free Form

Although the above forms appear smooth, we speculate that improved fairness of the skeleton, as shown below, can increase the fairness for the surface. We have not, however, performed a detailed analysis of the fairness of convolution surfaces.[3]

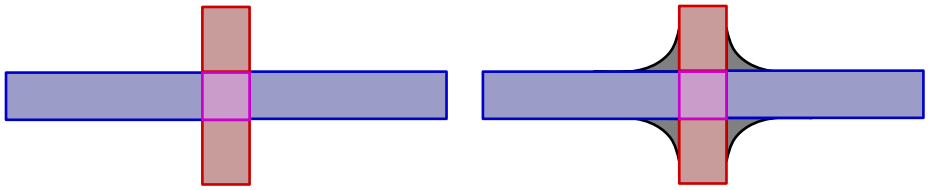

Figure 6.9 Improved Skeletal Fairness

left: contiguous skeleton, right: smooth contiguous skeleton

6.3 Articulation

In the previous section we discussed the 'plateau' that develops over wide polygons. Because of the superposition property of convolution, this plateau may develop from an arbitrary configuration of coplanar, abutting polygons, and will be seamless. This seamlessness provides flexibility in the definition of polygonal skeletons. The following articulation, in which an upper rectangular skeletal element rotates into alignment with a lower rectangle, demonstrates this seamless quality.

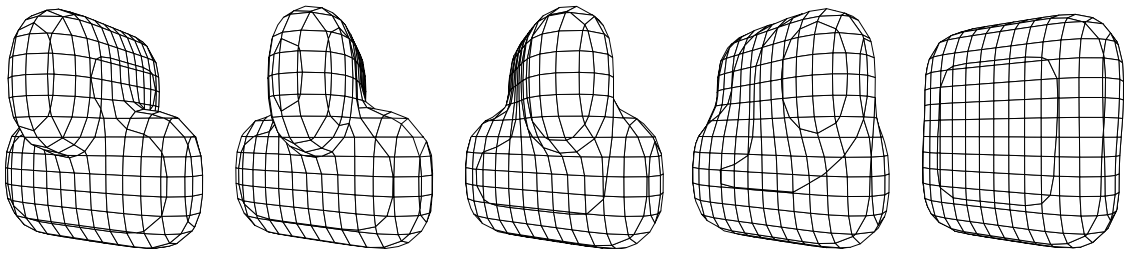

Figure 6.10 Seamlessness

6.4 Muscles and Arm

In [Bloomenthal and Shoemake 1991] convolution was applied to a polygonal skeleton to model the geometry of a human arm. We review this process here, providing additional details regarding the 'muscles' used in the model. Each muscle was defined by a planar polygon, as shown below. This simple representation was meant to facilitate interactive design. Much improved representations for biological muscle may be found in [Chen and Zeltzer 1992] [Smith 1990] and [Waters 1987].

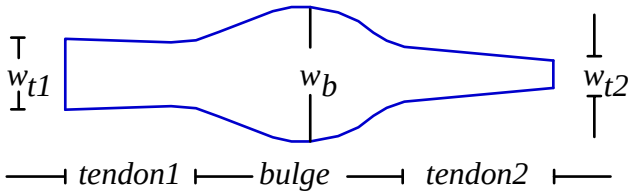

Figure 6.11 Individual Muscle and Parameters

As suggested in [Bloomenthal and Shoemake 1991], finer control over the surface can be obtained by a) the use of kernels of differing support, and b) the association of *weights* with the skeletal elements. In particular, a weight may be given to each polygon vertex before its initial rendering, thereby modifying the thickness of the resulting implicit primitive. Or, the entire polygon image can be scaled by a 'weight image.'

Weighting can provide for geometric taper. For example, the muscle tendons are smaller in cross-section than the muscle itself. Therefore, the tendons must have reduced thickness as well as reduced width. This was accomplished by assigning weights to the muscle polygon. The initial image and convolved image are shown below.

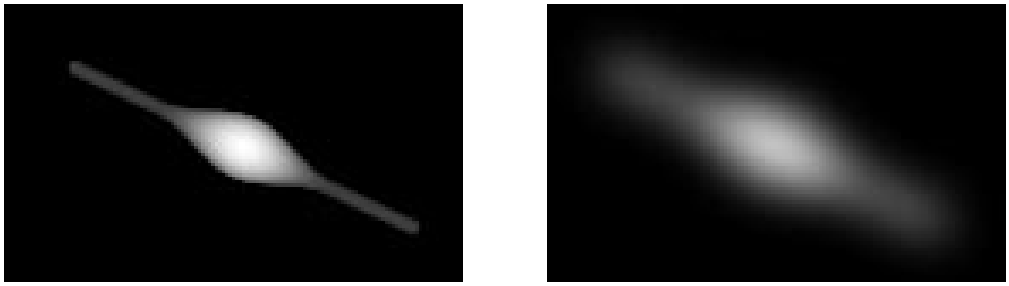

Figure 6.12 A Muscle

left: weighted skeletal element, right: convolution image

The five muscles were arranged as in the figure below, left, and their convolutions were summed to produce the surface shown below, right.[4]

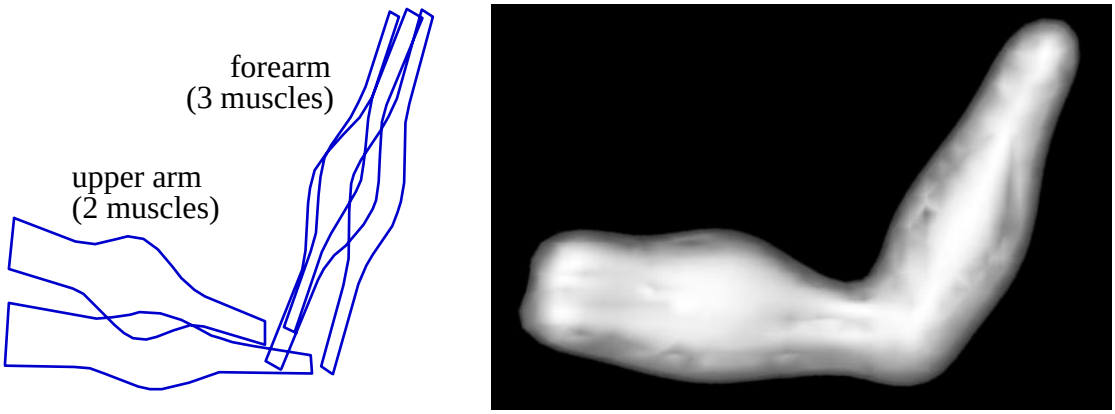

Figure 6.13 An Arm

left: arrangement of muscles, right: resulting surface

6.5 The Hand

The above arm is a collection of independent 'muscles.' We achieve greater definition in the hand by representing the bones of the palm and of the fingers with contiguous polygonal elements, as described in section 6.2. Additional one-dimensional elements represent the *adductor pollicis* muscle near the thumb [Gray 1973], the veins on the back of the hand, and the tendons to each of the fingers.

Modeling the human hand has interested several researchers. In [Thompson *et al*. 1988] mechanical aspects of the hand are simulated by manipulating the position and orientation of 'bones;' an overlying skin is not produced, however. In [Gourret *et al*. 1989] the hand is modeled as a deformable surface overlying a bone structure; the surface is parametrically defined and deformed according to forces acting upon the hand. In [Forsey 1991] a smooth, hierarchical B-spline surface is generated from articulated, contiguous skeletons, accounting for bulges and creases of the hand. And in [Landreth 1994] is developed a sophisticated combination of varying finger cross-sections and flexible inter-finger webbing, involving considerable user interaction with a commercial design system. All of these methods define a surface parametrically, requiring a predefined (or user defined) surface topology.

As we have argued in previous chapters, an implicit definition permits the designer to concentrate on the skeleton. There is at least one previous example of an implicitly defined hand provided in a tutorial for a commercial design package [Patterson 1994]. The tutorial is lengthy and yields an unconvincing model. The reasons are a) a hierarchical representation is not provided for the user, b) the implicit primitives are ellipsoids, rather than convolution surfaces, and c) there is no graphical interface for the user (in comparison, we regard the sketch in figure 2.4 as a graphical interface for the hand developed in this chapter).

The sketch in figure 2.4 was readily converted to the finger specifications outlined in section 2.7. As diagrammed below, the skeletal primitives of the hand include:

palm bones:	a contiguous set of 15 triangles,
finger bones:	a contiguous set of 48 triangles,
tendons:	five sets of connected line segments, 10 in total,
veins:	two sets of connected line segments, 11 in total,
muscle (adductor pollicis):	one line segment,

resulting in a total of 85 skeletal elements. The coloring in the shaded image

derived from colors assigned to the skeletal elements.

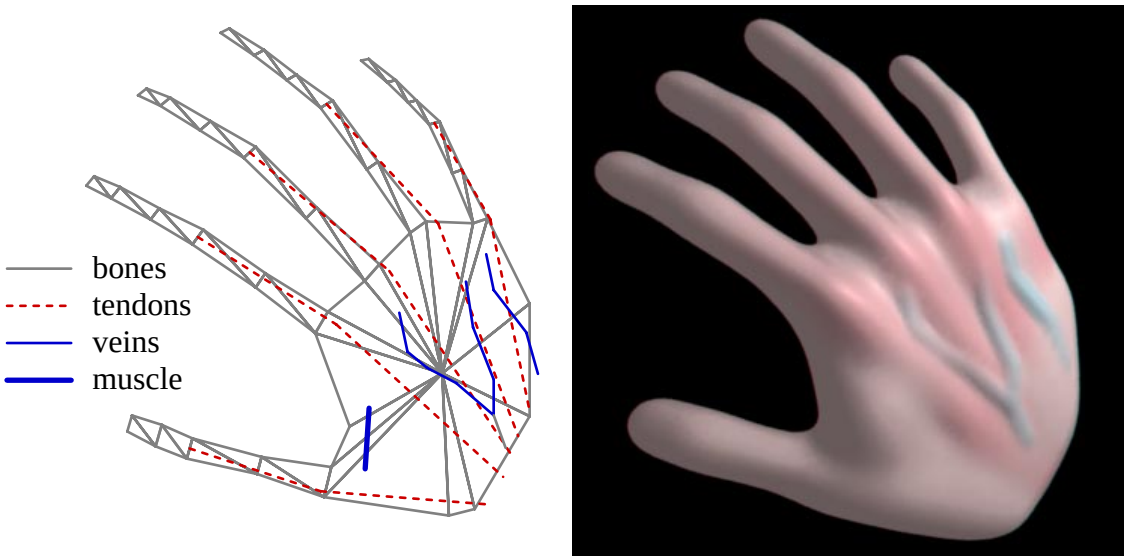

Figure 6.14 Skeleton and Hand

The following page displays individual and summed convolution surfaces that constitute the hand. At the upper left are the fingers in isolation; at the upper right are the tendons in isolation. The middle left shows the palm, the middle right shows the veins and the thumb muscle. At the lower left is the combination of palm and fingers, and at the lower right is the complete hand, consisting of fingers, palm, muscle, tendons, and veins. The images are derived completely from geometry, without any surface mapping. The geometry is free from seams and unnatural creases, which can be detected in several of the parametrically defined models.

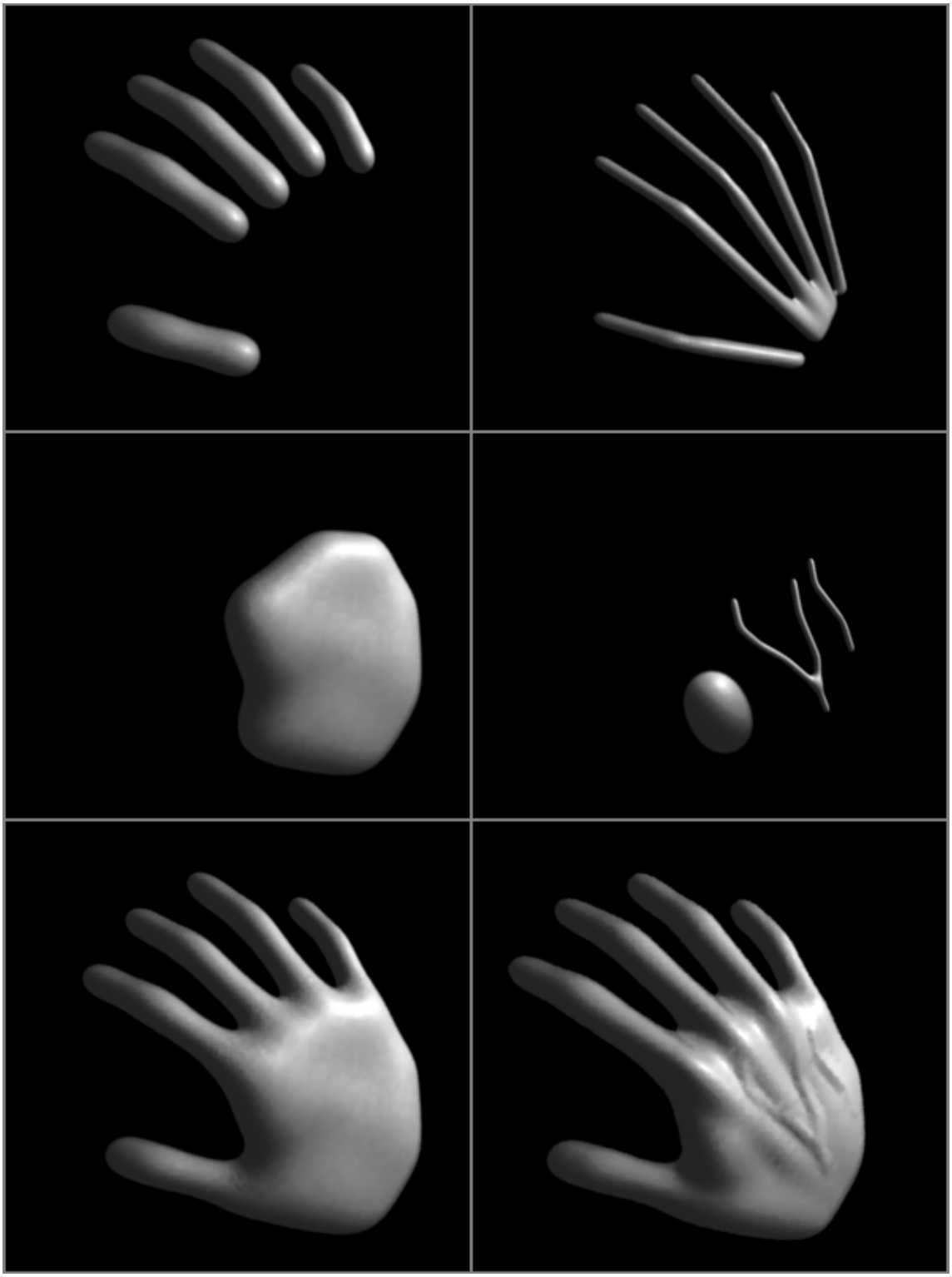

Figure 6.15 Individual and Combined Components

Convolution surfaces are a flexible and intuitive method for surface design based on skeletons. As illustrated below, the convolution surface follows a subset of a skeleton in a predictable manner. This affords the designer a stable environment within which to modify or complicate his or her design.

Figure 6.16 Palm and Tendons

6.6 Convolution of Volumes

We noted in the previous chapter that it is important that the convolution kernel be spherically symmetric so that the convolution surface is independent of skeletal orientation. For one-dimensional curves, this implies a circular cross-section. For planar skeletal elements, this implies a back-to-front (not bilateral) symmetry of each component surface. In sections 5.6.13 and 7.3 we discuss non-circular cross-sections for curves, and in section 6.4 a bi-directional 'weight image' could produce a non-circular cross-section for polygons. Alternatively, an asymmetric shape could be obtained by convolving an asymmetric *volumetric* skeleton.

Convolution of a volume was introduced in [Colburn 1990] in order to smooth previously defined solid objects. As shown below, just as convolution surfaces tend to interpolate the endpoints of a segment and the edges of a polygon, so they

tend to interpolate the surfaces of a volume. To accommodate volumetric skeletal elements, the discrete two-dimensional array discussed in this chapter could be extended to a three-dimensional array. We have not done so, however, because we prefer the design qualities of the one and two-dimensional elements. For example, these elements tend to inflate the skeleton, giving it body; convolution of a volume, however, tends to bevel and fillet, but not inflate, the skeleton.

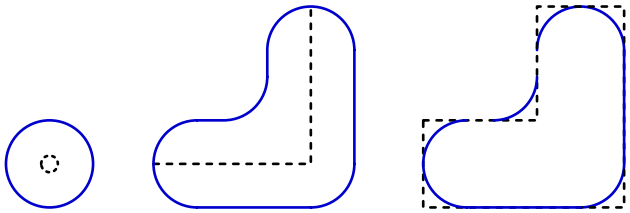

Figure 6.17 Cross-Sections of Differently Dimensioned Skeletal Elements

left: segment, middle: polygon, right: volume (skeletons are dashed)

6.7 Unwanted Blending

A common concern related to the implicit blend of volumes is the prevention of unwanted blends. One prevention technique is the judicious application of 'grouping' [Beier 1993], [Opalach 1993]. This is a technique whereby certain volumes are allowed to blend with some but not all other volumes. For example, in the illustration below, limb *a* is allowed to blend with limbs *b* and *e*, *b* is allowed to blend with *a*, *c*, and *e*, and *e* is allowed to blend with *a*, *b*, and *d*. But *c* is allowed to blend only with *b*, and *d* is allowed to blend only with *e*.

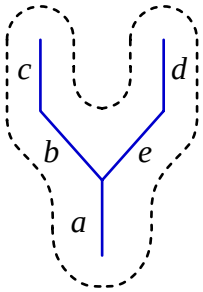

Figure 6.18 Skeletal Grouping

An alternative approach is to obtain finely detailed blending by employing very small primitives. This is reminiscent of the ‘strand’ model employed in [Greene 1991] and [Holton 1994]. The reduced influence of each skeletal element would reduce unwanted blending. It would be instructive to compare the ramiform blend of many small strands against that of a few large limbs.

It is also possible to provide implicit ‘separators’ between skeletal elements; these would be negatively valued functions that block the blending of adjacent primitives. This may, however, be an undue burden on the designer. Further, it isn’t necessarily reasonable to prevent all forms of unwanted blending. In particular, if an animator causes two primitives to actually intersect, it isn’t possible for the implicit surface function to maintain a separation. This problem can occur in the animation of non-implicit forms. Polyhedral primitives, for example, can intersect; ultimately the animator must prevent such an occurrence, or there must be limits on the articulation of a skeleton, for both parametric and implicit models.

Another alternative, proposed in [Gascuel 1993], is the compression of implicit primitives if their volumes intersect. Whichever combination of techniques is found suitable should operate in a natural manner and without discontinuities in the implicit primitives.

6.8 Details, Details

We have presented techniques whereby individually defined implicit primitives are combined to produce a smooth, macro-structured hand. Highly realistic models also require details such as fingernails and creases. The implicit definition of fingernails would, presumably, require a non-manifold definition, as described in chapter 4.

Creases tend to appear at points of maximum skin compression. Conversely, maximum smoothing seems to occur at points where skin is stretched, since this suggests an inward, smoothing pressure on muscles and tissue. Thus, the location

of creases and (intentional) bulges can be linked to skeletal articulation. In particular, we wish to mimic the relationship between surface and skeleton illustrated below. We speculate that negative functions can implement creases near concave portions of a joint, and positive functions can implement bulges near convex portions of a joint. Alternatively, as mentioned in section 3.6, creases can be added to an otherwise smooth surface subsequent to its conversion to a concrete representation.

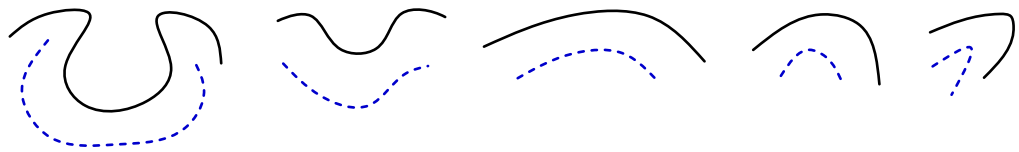

Figure 6.19 Creased and Stretched Surfaces

(skeleton is dashed)

Although we have included veins as a primitive in the construction of the macro-structured hand, it may be appropriate to regard them as detail placed upon the macro-structure. Indeed, vein placement was the most difficult task in the design of the hand. In section 3.6 we considered navigation along a concrete surface as a means to distribute surface detail. An interesting use of this technique would be to place the vein skeletal elements after an initial polygonization of the palm, fingers, and tendons. Once the vein skeletal elements are in place, the hand could be polygonized again, with the additional detail. In [Jevans *et al.* 1988] are provided techniques to improve the performance of polygonization when local changes have been made to the defining function of a previously polygonized surface.

6.9 Conclusions

We have implemented a model for the human hand that consists of skeletal elements representing the palm of the hand, its fingers, tendons, and veins, and the prominent adductor muscle for the thumb. We have not implemented details such as skin creases or fingernails, nor have we implemented grouping to avoid

unwanted blending. We have yet to animate the hand to observe its articulated behavior. Because the skeleton is expressed in terms of joint angles for the base of the fingers and for the knuckles, articulation is readily specified. Animation may, however, indicate a need for additional skeletal elements representing, for example, the webbing between fingers.

A design environment utilizing convolution surfaces enables the creation of complex, well-behaved shapes through the specification of the position, orientation, scale, and weights of individual polygonal skeletal elements. In using convolution, however, the designer loses explicit control over fillets and chamfers. This is more than compensated by the generality of convolution blending. Convolution is an integral, not a summation of discrete pieces. That is, the planar skeletal elements are fragmented into an infinitude of points, each scaling a replica of the filter kernel. The superposition of these kernels is the three-dimensional convolution of the polygon with the filter. Resulting surfaces are quite smooth and are bulge-free provided the skeletal elements are contiguous and at least as wide as the full filter support.

Although in section 6.1 we suggested a method to determine filter support in pixels given polygon size and desired volumetric thickness, the determination of an acceptable resolution for the rasterized polygon given the filter width, the size and shape of the polygon, and the sample size used during polygonization, remains a subject for continued investigation. Another interesting area of study is the modification of primitive thickness, as described in section 6.4.

6.10 Notes

1. In this chapter, a three-dimensional, separable cubic B-spline filter approximates the Gaussian. The B-spline filter has finite support and approximate spherical symmetry.

2. The choice of ½ as an iso-contour value was made to force the surface through

the endpoints of a segment or the edges of a polygon. If we wished for greater inflation of the surface, the iso-value can be lowered or the implicit primitives scaled. This, in fact, was done for the arm (section 6.4) and hand (section 6.5). As observed in section 5.6.11, the normalization by *r* for the distance filter (equation 5.31) must occur for the integration filter as well. This means that changing *r* in order to inflate the surface beyond the edges of the polygon will fail because the polygon will need to increase in width in order to span the increased domain of the integration filter.

3. We speculate that the continuity of a surface is G^{s+k}, where the kernel has C^k parametric continuity and the skeleton has G^s geometric continuity. There is a step function across both one-dimensional and two-dimensional skeletons, so that their geometric continuity in three-space is G^0. The geometric continuity of a 'density skeleton,' *i.e.*, a volumetric skeleton consisting of a density distribution, would be greater than zero. Thus, for the skeletons used in this dissertation, the geometric continuity of the convolution surface is fixed by the convolution kernel. The fairness, however, also depends on skeletal fairness.

4. This image was generated before the correction of errors in the surface normal computation, which produced the mottled appearance of the surface.

I try to testify in my prints
that we live in a beautiful, orderly world,
and not in a formless chaos, as it so often seems.
(M.C. Escher)