

# Blended Deformable Models

(In *IEEE Trans. Pattern Analysis and Machine Intelligence*, April 1996, 18:4, pp. 443-448)

Douglas DeCarlo and Dimitri Metaxas \*

Department of Computer & Information Science

University of Pennsylvania

Philadelphia PA 19104-6389

dmd@gradient.cis.upenn.edu, dnm@central.cis.upenn.edu

## Abstract

This paper develops a new class of parameterized models based on the linear interpolation of two parameterized shapes along their main axes, using a blending function. This blending function specifies the relative contribution of each component shape on the resulting blended shape. The resulting blended shape can have aspects of each of the component shapes. Using a small number of additional parameters, blending extends the coverage of shape primitives while also providing abstraction of shape. In particular, it offers the ability to construct shapes whose genus can change. Blended models are incorporated into a physics-based shape estimation framework which uses dynamic deformable models. Finally, we present experiments involving the extraction of complex shapes from range data including examples of dynamic genus change.

**Keywords:** Shape Representation, Shape Blending, Shape Abstraction, Shape Estimation, Physics-Based Modeling

## 1 Introduction

Shape models incorporate trade-offs between conciseness of representation and descriptive power which affect their usefulness for different applications. For shape estimation, it is important that shape models cover a wide variety of shapes using a small number of intuitive parameters. Finding the right balance is a difficult and important problem. When the ultimate goal is recognition, abstraction of shape is also a significant issue.

There are many current shape representations that use a small number of parameters, such as generalized cylinders [3, 10, 14], superquadrics [1, 15, 18], hyperquadrics [6] and geons [2]. These are useful for recognition tasks, but lack the generality to represent a large class of shapes in a single model. Representations with many parameters, such as surfaces with free-form deformations [22] have a wide shape coverage, but have too many parameters to be useful in recognition tasks. Advancing front methods [9] and oriented particle systems [19] provide surface connectivity information and can model surfaces of arbitrary topology,

---

\*This work was supported by NSF grants IRI-9309917, MIP-94-20393 and ARPA grant DAAH-049510067

but do not provide a compact representation of shape. In fact, no existing model for shape estimation with a compact representation can represent objects of varying topology in a unified way—an abrupt change in the model (*both* geometric and representational) is required to perform the topological change. Making such a drastic decision during estimation is often difficult, and is not likely to be robust. Estimation using implicit polynomial based representations [20] has also been investigated. The degree and configuration of the algebraic surface to be used for fitting must be specified in advance, thus making smooth topological changes difficult. Models such as those used in solid modeling [7, 17] have flexible and intuitive representations, but they were not designed for shape estimation—they were designed for human use. For shape recovery applications using CAD models, compactness in representation is not often a major concern.

Systems which are applicable for both shape reconstruction and shape recognition have been presented [12, 16, 23]. In [16] the shape was specified by its deformation modes and extracted using a closed-form solution of modal analysis. Shape was represented in [23] using a wavelet basis and estimated by embedding it in a probabilistic framework. Both of these methods provide a collection of parameters ordered by level of detail. The models in [12, 21] incorporate global deformations which represent prominent shape features, and local deformations which capture surface detail.

Abstraction and compactness of representation are distinct concepts, but often both are required in recognition systems. Considering the issue of abstraction, the ability to combine together different shapes into a unified model is very important. Algebraic surface blends [7] provide this ability, but are not easily applied to shape estimation. Blobby models [13] can also combine shapes, but lack flexibility in the underlying combined shapes, resulting in large numbers of components.

We propose an extension to the shape representation of [12, 21] which we call *blended deformable models* to address the issue of combining shapes together into a single model. Given two shapes that can be defined parametrically on a common material coordinate space, blended shapes are constructed by the linear interpolation of two shapes using a blending function that specifies the relative contribution of each shape on the resulting blended shape. For example, a sphere and a cylinder blended together could produce a bullet shaped object (see figure 2). In addition, this parameterization is able to represent shapes of genus<sup>1</sup> 0 and 1: blending a sphere and torus together produces an object in which the presence of the hole depends

---

<sup>1</sup>the number of holes in a shape—a sphere has genus 0, a torus has genus 1

on the value of the blending function. In addition, a geometrically smooth transition from sphere to torus is achievable by smoothly changing the blending function.

Figure 3 shows a variety of shapes that we can create using blending. In a unified model, blended models compactly and intuitively represent a wide variety of shapes, including shapes of varying genus. An *abstraction* of shape is also provided—the above example blended shape is clearly composed of a sphere and cylinder, which are components of the representation. The global nature of these models allows an efficient approach to shape estimation and the ability to handle situations where range data are *incomplete or sparse*.

In this paper, we show how blended models can be incorporated into the previously developed physics-based estimation framework presented in [12, 21]. We conclude after demonstrating our technique through a series of experiments involving incomplete range data from various objects.

## 2 Geometry of blended models

### 2.1 Deformable model geometry

As in [12, 21], the models used in this paper are 3-D surface shape models. The position of a point on the model is given in world coordinates by  $\mathbf{x}$  which is the result of a translation and rotation of its position  $\mathbf{p}$ , with respect to a non-inertial reference frame. The material coordinates  $\mathbf{u} = (u, v)$  of these shapes are specified over a domain  $\Omega$ . The position of a point on the world model at time  $t$ , with material coordinates  $\mathbf{u}$ , with respect to an inertial frame of reference is

$$\mathbf{x}(\mathbf{u}, t) = \mathbf{c}(t) + \mathbf{R}(t)\mathbf{p}(\mathbf{u}, t), \tag{1}$$

where  $\mathbf{c}$  is the center of the inertial frame, and  $\mathbf{R}$  is a rotation matrix which specifies the relative orientation of the inertial frame to a fixed reference frame.

In the non-inertial (fixed) reference frame, the position of model points  $\mathbf{p}$ , is the sum of a reference shape  $\mathbf{s}$  and a local displacement  $\mathbf{d}$  so that

$$\mathbf{p}(\mathbf{u}, t) = \mathbf{s}(\mathbf{u}, t) + \mathbf{d}(\mathbf{u}, t). \tag{2}$$

These local displacements,  $\mathbf{d}$ , allow the representation of fine detail, while the reference shape,  $\mathbf{s}$ , captures salient shape features. The reference shape of the model,  $\mathbf{s}$ , is constructed by applying a global deformation

$\mathbf{T}$  (such as bending) with parameters  $\mathbf{q}_T$  to a shape primitive  $\mathbf{e}$  as follows:

$$\mathbf{s}(\mathbf{u}) = \mathbf{T}(\mathbf{e}; \mathbf{q}_T). \quad (3)$$

For a 3-D shape primitive (such as a superellipsoid [1]), we have  $\mathbf{e}(\mathbf{u}) : \Omega \rightarrow \mathbb{R}^3$ . To represent the geometry of the primitive, a mesh of nodes is used, where each node is assigned a unique point in  $\Omega$ . The edges connecting the nodes represent connectivity of the nodes in  $\Omega$  space. Nodes can be merged together to form a closed mesh where points in  $\Omega$  map to the same 3-D model location (such as for the poles of a sphere).

The primitives we will be considering have global shape parameters  $\mathbf{q}_e$  which specify the shape. Including these parameters, we represent the geometric primitive as

$$\mathbf{e}(\mathbf{u}; \mathbf{q}_e), \quad (4)$$

which is defined parametrically in  $\mathbf{u}$  over  $\Omega$  and has global shape parameters  $\mathbf{q}_e$ .

Even though our framework can be applied to any class of parameterized primitives, we will be using superellipsoid and supertoroid primitives [1] to create a blended model. We will now extend the above definition of the global shape  $\mathbf{s}$  to include blended models.

## 2.2 Shape blending

In a method analogous to the linear interpolation of two points, it is possible to blend two functions. Given two functions,  $f(x)$  and  $g(x)$ , we can blend them using a third function,  $\alpha(x)$  (with range  $[0, 1]$ ), so that

$$h(x) = f(x)\alpha(x) + g(x)(1 - \alpha(x)). \quad (5)$$

An example of this is shown in figure 1. Notice how  $h(x) = f(x)$  where  $\alpha(x) = 1$ ,  $h(x) = g(x)$  where  $\alpha(x) = 0$ , and how  $h(x)$  is between  $f(x)$  and  $g(x)$  everywhere.

Using this idea, we can blend parameterized shapes by the following formula:

$$\mathbf{s}(u, v) = \mathbf{s}_1(u, v)\alpha(u) + \mathbf{s}_2(u, v)(1 - \alpha(u)), \quad (6)$$

where  $\mathbf{s}_1$  and  $\mathbf{s}_2$  are two shapes parameterized over  $\Omega$ , as in figures 2(a) and (b). Figure 2(c) shows  $\mathbf{s}$ , the result of blending the shapes shown in figures 2(a) and (b). The blending function used to blend the shapes is shown in figure 2(d). The blending is performed along  $u$ , which corresponds to the  $z$ -axis in these shapes

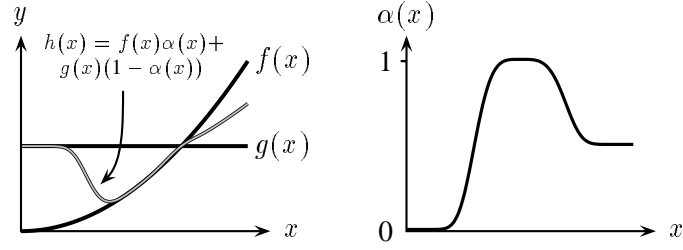


Figure 1: Blending of two functions  $f(x)$ ,  $g(x)$  given blending function  $\alpha(x)$

(from pole to pole). This particular blending function was chosen to illustrate how different parts of the component shapes are expressed in the resulting shape. Notice how the “top” of  $s$  looks like  $s_2$  (a cylinder)

since  $\alpha\left(\frac{\pi}{2}\right) = 0$ , and how the “bottom” of  $s$  looks like  $s_1$  (a sphere) since  $\alpha\left(-\frac{\pi}{2}\right) = 1$ .

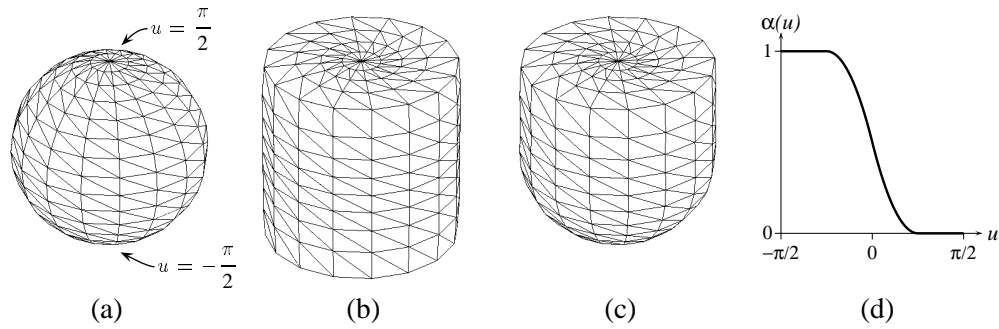


Figure 2: (a) Shape  $s_1$  (b) Shape  $s_2$  (c) Blended shape  $s$  (d) Blending function  $\alpha(u)$

The global parameters of  $s$  will include the global *shape* parameters of  $s_1$  and  $s_2$ , those that specify  $\alpha$  (see section 2.4), and the global *deformation* parameters  $\mathbf{q}_T$ . A common deformation  $\mathbf{T}$  is applied separately to each shape primitive so that  $s_1 = \mathbf{T}(\mathbf{e}_1, \mathbf{q}_T)$  and  $s_2 = \mathbf{T}(\mathbf{e}_2, \mathbf{q}_T)$ . These resulting deformed shapes are then blended together using (6).

When blending shapes, not all combinations of primitives will achieve desirable results. For example, a blend between two spheres where one is rotated 90 degrees from the other will produce an interpenetrating object. But since we are able to choose the models in advance for a vision application, we can simply choose compatible shapes, such as a superellipsoid and a supertoroid.

For the purposes of this paper, we will only have  $\alpha$  vary with  $u$  instead of both  $u$  and  $v$ . This limits the coverage to axially symmetric shapes. This restriction does not limit the applicability of blending to the process of shape abstraction. A variety of shapes produced using this restricted form of blending are shown in figure 3 by blending superellipsoids and supertoroids. While these shapes are expressible

using other representations [3, 17, 22], blending provides a compact and abstract representation. Algebraic surface blending [7] is a CAD method for connecting shapes together through the construction of blend surfaces which are placed adjacent to the component shapes. While similar in spirit, the underlying theory is very different from the blending presented here, since the smooth join between shapes is achieved by geometrically inserting blend surfaces, not by interpolation.

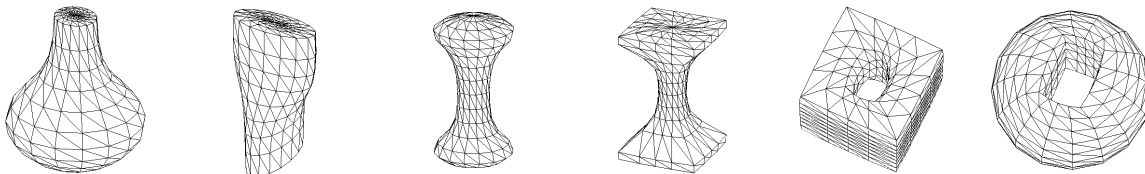


Figure 3: Examples of blended shapes

### 2.3 Supertoroid definition

In addition to the superellipsoid [1], we will be using the following definition for a supertoroid primitive:

$$\mathbf{e}_{\text{torus}}(u, v; a_1, a_2, a_3, a_4, a_5, \epsilon_1, \epsilon_2) = \begin{pmatrix} \frac{a_1}{a_4 + 1} (a_4 + C_{2u}^{\epsilon_1}) C_v^{\epsilon_2} \\ \frac{a_2}{a_5 + 1} (a_5 + C_{2u}^{\epsilon_1}) S_v^{\epsilon_2} \\ a_3 S_{2u}^{\epsilon_1} \end{pmatrix} \begin{matrix} u \in (-\pi/2, \pi/2] \\ v \in (-\pi, \pi] \end{matrix}, \quad (7)$$

where  $a_1, a_2, a_3, \epsilon_1, \epsilon_2 > 0$  and  $a_4, a_5 \geq 1$ .  $a_1, a_2$  and  $a_3$  are size parameters in the  $x, y$  and  $z$  directions respectively.  $\epsilon_1$  and  $\epsilon_2$  are squareness parameters as in a superellipsoid.  $a_4$  and  $a_5$  are hole size parameters in the  $x$  and  $y$  directions. The hole is closed when  $a_4 = a_5 = 1$ , and the hole opens for values greater than 1. As in a superellipsoid, we define  $C_\theta^\epsilon = \text{sgn}(\cos \theta) |\cos \theta|^\epsilon$  and  $S_\theta^\epsilon = \text{sgn}(\sin \theta) |\sin \theta|^\epsilon$ .

This definition is similar to the supertoroid given by Barr [1]. The addition of  $a_5$ , a second hole size parameter allows asymmetric holes. The presence of the scaling factors  $1/(a_4 + 1)$  and  $1/(a_5 + 1)$  separate the effects of the global size parameters ( $a_1, a_2$  and  $a_3$ ) from the hole size parameters ( $a_4$  and  $a_5$ ) to allow hole size changes that do not affect the global torus size.

## 2.4 Blending function parameterization

The blending function  $\alpha$  is implemented as a non-uniform quadratic B-spline function [5]. Given different

types of shape primitives, the domain of  $\alpha$  may vary. For a superellipsoid,  $\alpha$  maps  $[-\frac{\pi}{2}, \frac{\pi}{2}]$  to  $[0, 1]$ .

The B-spline function is specified using  $L + 1$  control values  $\{c_i \mid i \in 0 \dots L\}$  and  $L$  knots  $\{u_i \mid i \in 1 \dots L\}$ , with  $u_1$  and  $u_L$  fixed to be the lower and upper bounds of the domain of  $\alpha$ . The function  $\alpha$  has the values  $\alpha(u_1) = c_0$  and  $\alpha(u_L) = c_L$  and has a continuous first derivative except where two knot values are equal.

The parameters used to construct the blending function are the  $L + 1$  control values and the  $L - 2$  movable knots ( $u_1$  and  $u_L$  are fixed), which yields  $2L - 1$  total parameters to specify  $\alpha$ . We concatenate all these parameters into the vector  $\mathbf{q}_b$ , so that

$$\mathbf{q}_b = (c_0, \dots, c_L, u_2, \dots, u_{L-1})^\top. \quad (8)$$

## 3 Genus changing

It is also possible to blend objects having genus 0 (a sphere) with objects having genus 1 (a torus). A hole will appear in the blended object as  $\alpha$  changes. There is no smooth transition between these two shapes because they are not homeomorphic<sup>2</sup>—no sequence of deformations will change a sphere into a torus. Yet it is possible to have a transition between the two where there is a single discontinuous event—when the object changes genus. This event affects only the topology of the object, not the geometry of the shape. An example transition is shown in figure 4.

Figure 4 is an illustrative sequence showing how a sphere can be transformed into a torus using a blended shape. The blended result is computed using (6), where  $\mathbf{s}_1$  is a torus and  $\mathbf{s}_2$  is a sphere. Initially, in figure 4(a),  $\alpha(u) = 0$  (for all values of  $u$ ) and the blended object has the geometry and topology of a sphere. The blended shapes in (b) and (c) show what happens if we slowly change  $\alpha(u)$  from 0 to 1. In (c), when  $\alpha(u) = 1$ , the shape is a pinched sphere [8]—the poles have dimpled inward until they touch. This has the same geometry as a torus (with the hole closed), but is topologically equivalent to a sphere.

At this time, at the location where the poles touch, we change the connectivity of the surface to be that

---

<sup>2</sup>topologically equivalent – a sphere and cylinder are homeomorphic, but a sphere and torus are not

of a torus. A discussion of how the node interconnections change is given in section 3.1. Once the pinched sphere is changed into a torus, the torus hole can now be opened by increasing the torus hole size parameters ( $a_4$  and  $a_5$ ), shown in (d) and (e) (shown from a slightly different viewpoint to make the hole visible).

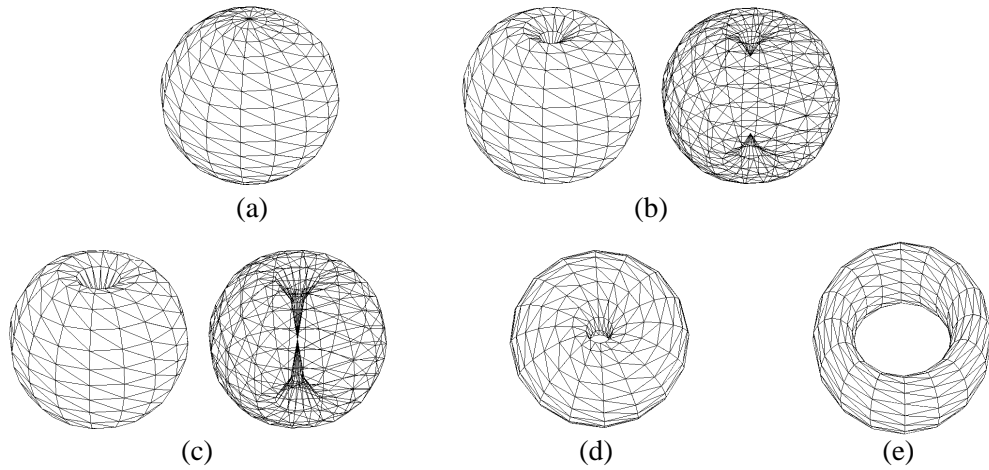


Figure 4: A blended shape changing from a sphere (a) to a torus (e)

There are two constraints on the parameters of a torus-sphere blend that must be enforced to insure the blended shape remains closed. The torus hole must remain closed when the object has genus 0. When the object has genus 1, the values  $\alpha\left(-\frac{\pi}{2}\right)$  and  $\alpha\left(\frac{\pi}{2}\right)$  must weight the torus so that the poles of the sphere are not

expressed in the blended shape. For figures 4(d) and (e), the constraint would be  $\alpha\left(-\frac{\pi}{2}\right) = \alpha\left(\frac{\pi}{2}\right) = 1$  since  $s_1$  is the torus. These constraints can be implemented in our framework by simply fixing the appropriate parameter values at the times in the estimation process when they are not permitted to change.

This entire process of genus change can be easily integrated into the physics-based estimation framework. For a hole to form, the object is deformed by the data forces into the configuration shown in (c). This point can be detected by examination of the blending function. At this point, the hole can *automatically* open due to forces from the data. Using this method, a hole can form in a physics-based way. The ideas presented can be applied to any shape primitives, although the actual steps involved may vary for different primitives.

### 3.1 Node interconnections

When altering the topology of an object, the mesh of nodes must be reconnected to conform to the new topology. This is a straightforward but necessary part of the genus conversion process. Figure 5 shows how  $\Omega$  is “folded up” to produce a sphere or torus. The arrows in these diagrams indicate two nodes being “merged” together, since the material coordinates of the nodes map to the same 3-D model coordinates. For both the sphere and the torus, a tube is made first (the dotted lines). For the sphere in figure 5(a), the north and south poles are created by closing each end of the tube. For the torus in figure 5(b), the ends of the tube are connected together. When the genus changes, the node mesh first must be unfolded, and then re-folded to have the proper configuration.

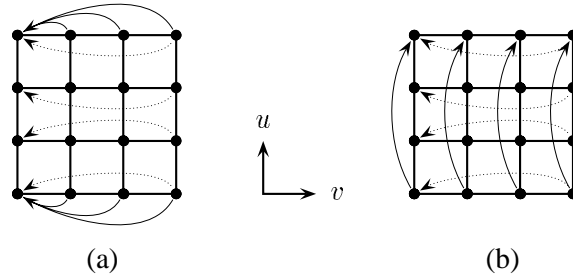


Figure 5: Node interconnection differences between a sphere (a) and torus (b)

## 4 Dynamics and generalized forces

The dynamics framework given in [12] can be used after several alterations. In this framework all the degrees of freedom needed to specify the shape (translation, rotation, global and local parameters) are collected together to form the generalized coordinates of the model,  $\mathbf{q}$ ,

$$\mathbf{q} = (\mathbf{q}_c^\top, \mathbf{q}_\theta^\top, \mathbf{q}_s^\top, \mathbf{q}_d^\top)^\top, \quad (9)$$

where  $\mathbf{q}_c = \mathbf{c}(t)$ ,  $\mathbf{q}_\theta$  is the quaternion used to specify  $\mathbf{R}(t)$ ,  $\mathbf{q}_d$  specifies the local deformations, and  $\mathbf{q}_s = (\mathbf{q}_{s_1}^\top, \mathbf{q}_{s_2}^\top, \mathbf{q}_b^\top, \mathbf{q}_T^\top)^\top$  are the global parameters ( $\mathbf{q}_{s_1}$  and  $\mathbf{q}_{s_2}$  are the parameters of each of the component shapes,  $\mathbf{q}_b$  are the parameters that specify the blending function  $\alpha$ , and  $\mathbf{q}_T$  are the parameters of the global parameterized deformations).

When fitting the model to data, the goal of shape reconstruction is to recover the parameters in  $\mathbf{q}$ . The approach used here performs the fitting in a physics-based way—the data apply forces to the surface of the

model, deforming it into the shape represented by the data [21]. The model can be made dynamic in  $\mathbf{q}$  by introducing mass, damping and stiffness and embedding it into a Lagrangian dynamics framework. The Lagrange equations of motion are second order differential equations [11]. In shape estimation applications, the mass is set to zero (so that the model has no inertia and comes to rest as soon as the applied forces equilibrate or vanish), resulting in the following simplified dynamic equation:

$$\mathbf{D}\dot{\mathbf{q}} + \mathbf{K}\mathbf{q} = \mathbf{f}_{\mathbf{q}} = (\mathbf{f}_{\mathbf{c}}^{\top}, \mathbf{f}_{\theta}^{\top}, \mathbf{f}_{\mathbf{s}}^{\top}, \mathbf{f}_{\mathbf{d}}^{\top})^{\top}, \quad (10)$$

where  $\mathbf{D}$  and  $\mathbf{K}$  are the damping and stiffness matrices respectively, and where  $\mathbf{f}_{\mathbf{q}}$  are the generalized forces [12]. These generalized forces can be further broken down into components each corresponding to a component of  $\mathbf{q}$  as given in (9) above.

Using (10),  $\dot{\mathbf{q}}$  can be computed, and an integration method can be used to update  $\mathbf{q}$ . Performing this process iteratively results in a model more closely representing the desired shape. Throughout the fitting process, parameter schedules are used [4, 14], as in other physics-based fitting frameworks. The fitting is performed initially using “coarse” parameters (translation, rotation, and major axis lengths), followed by the “fine” parameters (blending parameters, superquadric squareness values). This allows improvements in efficiency by initially reducing the dimension of the parameter space. By initially disabling the fine scale parameters, local minimum solutions also can be avoided.

We compute the generalized forces  $\mathbf{f}_{\mathbf{q}}$  from the 3-D applied forces. The computation of  $\mathbf{f}_{\mathbf{c}}$ ,  $\mathbf{f}_{\theta}$  and  $\mathbf{f}_{\mathbf{d}}$  are the same as described in [12]. The computation of  $\mathbf{f}_{\mathbf{s}}$  is given by

$$\mathbf{f}_{\mathbf{s}} = (\mathbf{R}\mathbf{J}_{\mathbf{s}})^{\top} \mathbf{f}_{\text{applied}}. \quad (11)$$

We compute  $\mathbf{J}_{\mathbf{s}}$ , the Jacobian for the global shape  $\mathbf{s}$ , as follows:

$$\mathbf{J}_{\mathbf{s}} = \partial\mathbf{s}/\partial\mathbf{q}_{\mathbf{s}}. \quad (12)$$

The Jacobian of the global shape,  $\mathbf{J}_{\mathbf{s}}$ , “converts” applied forces into generalized forces, which will deform the global shape. The addition of blending changes the computation of  $\mathbf{J}_{\mathbf{s}}$ . In particular, from (6) and (12):

$$\mathbf{J}_{\mathbf{s}} = [\alpha(u)\mathbf{J}_{\mathbf{s}_1} \mid (1 - \alpha(u))\mathbf{J}_{\mathbf{s}_2} \mid \mathbf{J}_b], \quad (13)$$

where  $\mathbf{J}_{\mathbf{s}_1} = \partial\mathbf{s}_1/\partial\mathbf{q}_{\mathbf{s}_1}$  is the Jacobian for the first shape,  $\mathbf{J}_{\mathbf{s}_2} = \partial\mathbf{s}_2/\partial\mathbf{q}_{\mathbf{s}_2}$  is the Jacobian for the second shape, and  $\mathbf{J}_b$  is the Jacobian for the parameters of the blending function, and is described below.

Intuitively, (13) means the Jacobians for the components of a blended shape have a greater or lesser effect at a particular location depending on the function  $\alpha$ . Considering the sphere/cylinder blending example in figure 2, if a force was applied to the “top” of the shape, only the parameters of the cylinder would be affected. Similarly, if a force was applied to the “bottom” of the shape, only the parameters of the sphere would change. Therefore, the blending function has the desirable effect of localizing the effect of a force to the appropriate shape component.

The Jacobian matrix  $\mathbf{J}_b$  reflects how the global shape  $\mathbf{s}$  changes with respect to the blending function parameters  $\mathbf{q}_b$ . Given (6) above,

$$\mathbf{J}_b = \frac{\partial \mathbf{s}(u, v)}{\partial \mathbf{q}_b} = (\mathbf{s}_1(u, v) - \mathbf{s}_2(u, v)) \frac{\partial \alpha(u)}{\partial \mathbf{q}_b}. \quad (14)$$

Given that  $\alpha$  is a B-spline, to compute  $\partial \alpha(u) / \partial \mathbf{q}_b$ , we apply the product rule to the de Boor algorithm [5]. The control value and knot constraints  $c_i \in [0, 1]$  for all  $0 \leq i \leq L$ , and  $u_i \leq u_j$  for all  $1 \leq i \leq j \leq L$ , are enforced to insure the components of  $\mathbf{q}_b$  have correctly bounded values. It is through  $\mathbf{J}_b$  that the blending function can change to reflect the shape of the data. Note that for blending to occur during shape estimation at a particular location on a shape, the underlying shapes must differ. If this was not the case, the difference of the two shapes ( $\mathbf{s}_1 - \mathbf{s}_2$ ) would be zero, making  $\mathbf{J}_b$  zero.

## 5 Experiments

In the following fitting experiments, we show the results of using blended shapes in our shape reconstruction system. Figure 6 shows information on each of the experiments including the number of data points, the resulting mean squared error (MSE), the size of the parameter set,  $L$  (the number of knots used to specify the blending function), the dimensions of the node mesh, and the number of iterations taken for the fit.

In each of the examples, the initial model configuration is shown. Initially, the model has all global shape parameters equal to 1, and is centered at the center of mass of the data. The blending function is initialized to  $\alpha(u) = 0$ . Initially, only 1/10 of the data are used (selected randomly). All of the models used are global in nature—no local deformations were used.

Figures 7 through 9 show the fitting results obtained for the five experiments. Each fitting example

Data	Source	Points	MSE	# Parm	$L$	Mesh	# Iter
light bulb	MSU (bulb1)	2024	1.27%	27	9	$17 \times 17$	150
sphere/cylinder	MSU (cylinder+sphere)	2015	3.64%	27	9	$17 \times 17$	223
torus	CAD generated	1503	0.62%	17	3	$16 \times 12$	147

MSU: Michigan State University PRIP database (special thanks to Anil Jain and Tim Newman)

Figure 6: Experiment data and statistics

starts with the initial configuration described above. After this, the first rough fit by varying only  $a_1$ ,  $a_2$  and  $a_3$  of  $\mathbf{s}_2$  is shown. The rest of the steps follow after this, and are described in detail below for each example.

Figure 7 shows the model in the process of fitting to light bulb data. A blend of two superellipsoids is used as the model. The initial model and range data are shown in (a), and the rough fit after the initial fit is shown in (b). The blending function changes in (c). In figure 7(d), all the data are used to complete the final step, where all the parameters are permitted to change. The final blending function (e) shows two distinct areas—where it is 0, and where it is 1, connected by a smooth transition.

Figure 8 shows the fitting of a sphere/cylinder object. Similar to the fitting process of the light bulb, (b) shows the initial fit, (c) shows the model after the blending function changes, and (d) shows the final fit using all the range data. With each step, the blending function is given to show how it changes during the fitting. Since this object has a corner where the sphere and cylinder meet, the blending function in (d) has developed a point where it is not differentiable.

Figure 9 shows the fitting of torus data using a blend of a superellipsoid and a supertoroid as the model. The initial range data are shown in (a), and the initialization is shown in (b). The rough initial fit is shown in (c). The poles are “pinched” together in (d), and the genus automatically changes to 1. The hole is pulled open in (e) and (f) (which are the same object from different viewpoints). A final fit using all data is shown in (g). Notice how the blending function (f) has  $\alpha\left(-\frac{\pi}{2}\right) = \alpha\left(\frac{\pi}{2}\right) = 1$ , since the hole is present. When fitting an object with a superellipsoid-supertoroid, it is necessary that there be some range data from the inside of the hole. Otherwise, the hole will not be able to be “pulled” through by data forces.

Each iteration with a full data set takes (on average) 1/2 second on a 50 MHz SGI R4000 using data sets of this size. An adaptive Euler method is used to update the object state. Initially, iterations have  $O(n \log d)$  complexity (where  $n$  is the number of nodes,  $d$  is the number of data points) due to initial nearest-node computations (for force assignment). Once the shape acquires its rough general shape, the complexity

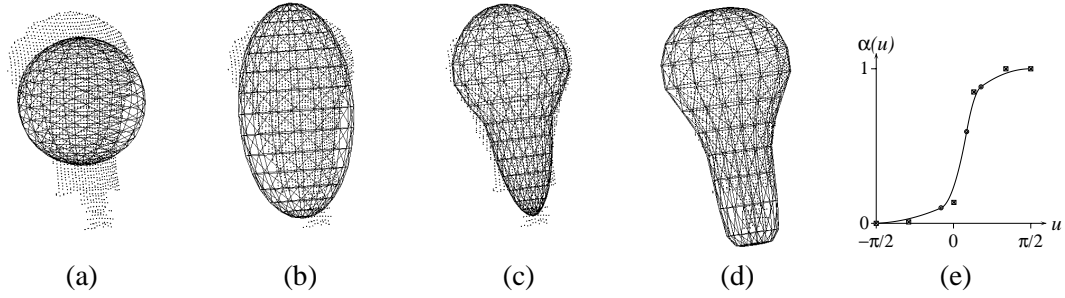


Figure 7: Fitting of light bulb data and blending function (e)

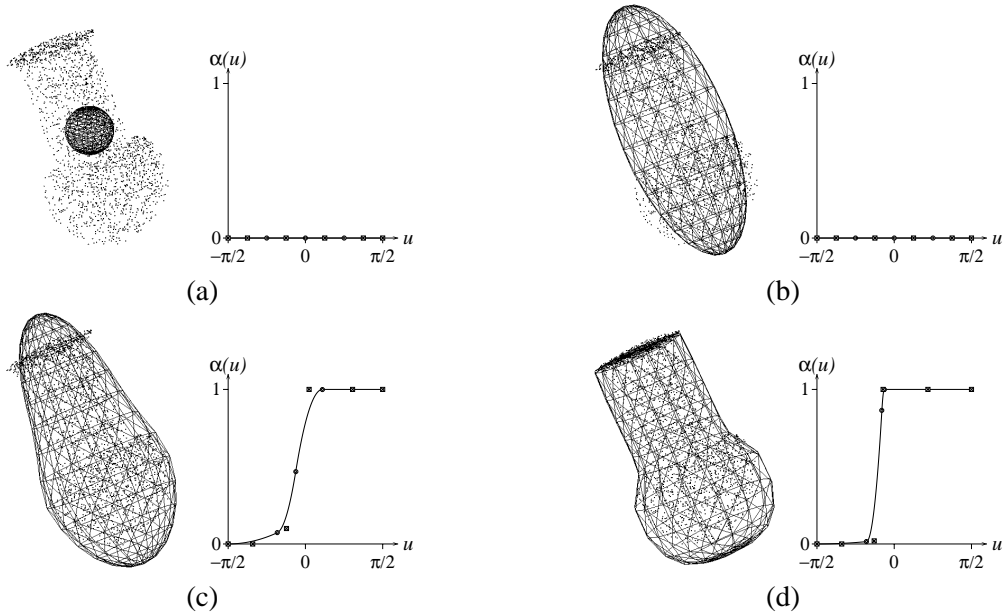


Figure 8: Fitting of sphere/cylinder data showing evolution of blending function

approaches  $O(n + d)$  since nearest-node information can often be carried across iterations. Since fewer range data can be used initially, this offers an additional constant factor speed increase. For the experiments presented here, this results in fits with durations ranging from 45 to 60 seconds each.

## 6 Conclusions and Future Work

We have developed and presented a new approach to shape modeling and estimation based on shape blending. These models we created can compactly and intuitively represent a large class of shapes in a single model, including shapes of varying genus. What we have presented here is also likely to be useful for recognition because blended shapes can be parameterized using a small number of intuitive global parameters. Blending provides a mechanism of changing topology without geometric discontinuity

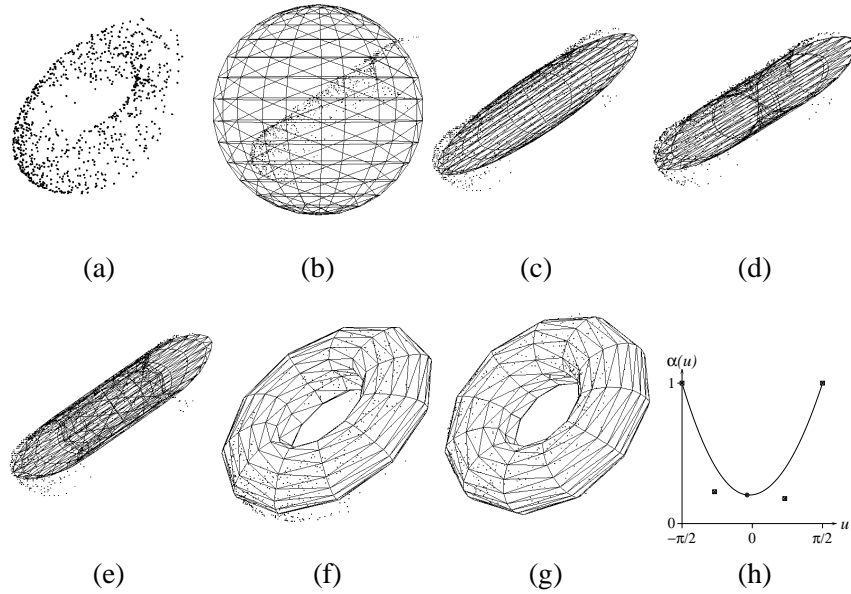


Figure 9: Fitting of torus data with genus change in (d)

(over time). While there is a representational change (clearly *some* change is necessary to alter topology when dealing with global shapes), we avoid the sudden geometric *and* representational changes that other compact shape estimations frameworks employ. Reducing the intensity of this decision should lead to greater robustness. We demonstrated the performance of our technique in a variety of shape estimation experiments involving the extraction of shapes with incomplete range data.

Currently, the blending function has a large number of degrees of freedom. If blending is to be used for abstraction, this number can be drastically reduced. Considering the blending functions shown in figures 7(e) and 8(d), the blending functions vary from 0 to 1, with a transition in between. A reduction in the number of parameters could be achieved by simply parameterizing the location and “character” of this transition. Blending functions with transitions such as these produce a blended shape which clearly shows parts of each component shape, and a transition region between the two shapes.

The abstractive power of blending is certainly the most useful characteristic. We are currently investigating how to extend the somewhat restricted form of blending presented here. By allowing blending to occur in arbitrary locations (not just axially), we hope to provide a general facility for combining together selected portions of different shapes (including the addition of holes at any location).

## References

- [1] A. Barr. Superquadrics and angle-preserving transformations. *IEEE Computer Graphics and Applications*, 1(1):11–23, 1981.
- [2] I. Biederman. Recognition-by-components: a theory of human image understanding. *Psychological Review*, 94:115–147, April 1987.
- [3] T. Binford. Visual perception by computer. In *IEEE Conference on Systems and Control*, December 1971.
- [4] D. DeCarlo and D. Metaxas. Blended deformable models. In *Proceedings CVPR '94*, pages 566–572, 1994.
- [5] G. Farin. *Curves and Surfaces for Computer Aided Geometric Design*. Academic Press, 1993.
- [6] A. J. Hanson. Hyperquadrics: smoothly deformable shapes with convex polyhedral bounds. *Computer Vision, Graphics, and Image Processing*, 44:191–210, 1988.
- [7] C. M. Hoffmann and J. Hopcroft. The geometry of projective blending surfaces. *Artificial Intelligence*, 37:357–376, 1988.
- [8] J. J. Koenderink. *Solid Shape*. MIT Press, 1990.
- [9] R. Malladi, J. A. Sethian, and B. C. Vemuri. Shape modeling with front propagation: A level set approach. *IEEE Pattern Analysis and Machine Intelligence*, 1994, to appear.
- [10] D. Marr and K. Nishihara. Representation and recognition of the spatial organization of three-dimensional shapes. *Proceedings Royal Society London*, 200:269–294, 1978.
- [11] D. Metaxas. *Physics-Based Modeling of Nonrigid Objects for Vision and Graphics*. PhD thesis, Department of Computer Science, University of Toronto, 1992.
- [12] D. Metaxas and D. Terzopoulos. Shape and nonrigid motion estimation through physics-based synthesis. *IEEE Pattern Analysis and Machine Intelligence*, 15(6):580–591, June 1993.
- [13] Shigeru Muraki. Volumetric shape description of range data using “blobby model”. In *Proceedings SIGGRAPH '91*, volume 25, pages 227–235, July 1991.
- [14] T. O’Donnell, T. Boult, X. Fang, and A. Gupta. The extruded generalized cylinder: A deformable model for object recovery. In *Proceedings CVPR '94*, pages 174–181, 1994.
- [15] A. Pentland. Perceptual organization and the representation of natural form. *Artificial Intelligence*, 28:293–331, 1986.
- [16] A. Pentland and S. Sclaroff. Closed-form solutions for physically based shape modeling and recognition. *IEEE Pattern Analysis and Machine Intelligence*, 13(7):715–729, 1991.
- [17] J. M. Snyder. *Generative Modeling for Computer Graphics and CAD*. Academic Press, 1992.
- [18] F. Solina and R. Bajcsy. Recovery of parametric models from range images: The case for superquadrics with global deformations. *IEEE Pattern Analysis and Machine Intelligence*, 12(2):131–147, 1990.
- [19] R. Szeliski, D. Tonnesen, and D. Terzopoulos. Modeling surfaces of arbitrary topology with dynamic particles. In *Proceedings CVPR '93*, pages 82–87, 1993.
- [20] G. Taubin. An improved algorithm for algebraic curve and surface fitting. In *Proceedings ICCV '93*, pages 658–665, 1993.
- [21] D. Terzopoulos and D. Metaxas. Dynamic 3D models with local and global deformations: Deformable superquadrics. *IEEE Pattern Analysis and Machine Intelligence*, 13(7):703–714, 1991.
- [22] D. Terzopoulos, A. Witkin, and M. Kass. Constraints on deformable models: Recovering 3D shape and nonrigid motion. *Artificial Intelligence*, 36(1):91–123, 1988.
- [23] B. C. Vemuri and A. Radisavljevic. Multiresolution stochastic hybrid shape models with fractal priors. *ACM Transactions on Graphics*, 13(2):177–207, 1994.