# On 3D Shape Synthesis

Heung-yeung Shum   Martial Hebert  Katsushi Ikeuchi

November 1995

CMU-CS-95-213

School of Computer Science
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213



This research is partially sponsored by the Advanced Research Projects Agency under the Department of the Army, Army Research Office under grant number DAAH04-94-G-0006, partially supported by ONR under grant number N00014-95-1-0591, and partially supported by NSF under Contract IRI-9224521. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies or endorsements, either expressed or implied, of the Department of the Army, or the U.S. government.

**Keywords:** Object representation, 3D shape synthesis, morphing, curvature.

# Abstract

We present a novel approach to *3D shape synthesis* of closed surfaces. A curved or polyhedral 3D object of genus zero is represented by a curvature distribution on a spherical mesh that has nearly uniform distribution with known connectivity among mesh nodes. This curvature distribution, i.e., the result of forward mapping from shape space to curvature space, is used as the intrinsic shape representation because it is invariant to rigid transformation and scale factor. Furthermore, with regularity constraints on the mesh, the inverse mapping from curvature space to shape space always exists and can be recovered using an iterative method. Therefore, the task of synthesizing a new shape from two known objects becomes one of interpolating the two known curvature distributions, and then mapping the interpolated curvature distribution back to a 3D morph. Using the distance between two curvature distributions, we can quantitatively control the shape synthesis process to yeild smooth curvature migration. Experiments show that our method produces smooth and realistic shape morphs.

# Table of Content

# List of Figures

# 1 Introduction

A traditional approach to synthesizing 3D shape is to use volume metamorphosis (or volume morphing). Similar to image morphing [1], volume morphing [12] is achieved by first warping two original volumes to new volumes, and then blending both into an intermediate shape, i.e., a morph. A major challenge to the 3D morphing system is the automatic feature recognition and matching because it is crucial to defining the transformation of the two objects. Unlike image morphing, however, a simple morph map can not be easily established for volume morphing. What is to be morphed between two volumes is not clearly defined. Most 3D morphing systems [4][12] (except [7], which attempts to do automatic feature registration) resort to a good user interface which allows the user to specify feature segments in both objects. However, even when many feature segments are located, smooth shape transition between two shapes is not guaranteed because the curvature may not be interpolated properly. Computationally, the volume morphing is very expensive because of the huge amount of data used.

This paper provides a means of synthesizing a 3D shapes on 3D surface instead of on 3D volume. We restrict our study on shapes of genus zero. We prefer to synthesize shape on 3D surface rather than 3D volume because surface data are most common in computer vision applications, such as from a light stripe range finder, a laser range finder, or stereo. On the other hand, if volume data is available (e.g., from a CT scan), it can be converted to a surface model using techniques such as “marching cube” [14]. For the purpose of shape synthesis, volume data is perhaps redundant.

We define our 3D shape synthesis problem as follows. Given two 3D shapes A and B, we synthesize a sequence of intermediate shapes (or morphs) which should have the following desirable properties defined in [12]:

1. *Realism: the morphs should be realistic objects which have plausible 3D geometry and which retain the essential features of the originals.*
2. *Smoothness: the change in the sequence of morphs must show a smooth transition between two originals.*

To meet the above requirements, we propose to synthesize a shape in terms of its curvature distribution because curvature is independent of rigid transformation and scale. However, neither Gaussian curvature nor mean curvature fully captures the essence of 3D shape. People have long searched for proper spherical shape representations using curvatures [6][9][13] for closed shape. For example, from the Extended Gaussian Image, we can construct the original shape if the object is convex. However, to synthesize arbitrary non-convex shape, we have to search for proper representation of 3D shape.

In our work, we use a spherical mesh to represent an object shape, and store local curvature at each node as its intrinsic representation. From this intrinsic representation, we show that, provided that some regularity constraint is introduced, the original shape can be reconstructed up to a scale factor and a rigid transformation. The correspondence between two original shapes can be automatically obtained by minimizing the difference between two curvature distributions as we have shown previously in [5][16]. Then the task of synthesizing a new object from two known objects becomes one of interpolating the two curvature distributions, and then mapping the interpolated curvature distribution back to 3D shape.

We first illustrate our basic approach on shape synthesis in Section 2. After introducing our spherical shape representation of closed 3D surfaces, Section 3 describes the mapping from shape space to curvature space, while the inverse mapping from curvature space to shape space is discussed in Section 4. Section 5 describes our curvature-based shape synthesis in detail. Correspondence between two shapes is briefly discussed; details of this are given elsewhere [12]. Finally, we show the results of our shape synthesis. and close with a discussion and conclusion.

# 2 Basic Idea

Our ultimate goal is to generate a sequence of intermediate shapes from two given 3D shapes which can be either closed curved or polyhedral surfaces without holes. We represent our 3D shape using a special spherical coordinate system [5]. A semi-regularly tessellated sphere is deformed so that the meshes sit on the original data points, while the connectivity among the mesh nodes is preserved. After the deformation process, we obtain a spherical representation with local curvature[1] at each mesh node. The local curvature at each node is calculated by its relative position to its neighbors. By enforcing local regularity at each mesh node, we can reconstruct shape from its curvature distribution up to a scale factor and a rigid transformation.

Our approach to shape synthesis is straightforward: for each original shape, we build its curvature distribution as its intrinsic representation. The correspondence between them is either specified by the user or can be directly computed, e.g., by minimizing the difference between their curvature distributions [16]. The problem of synthesizing a new shape from two known shapes becomes one of interpolating two known curvature distributions. The new shape is reconstructed from the interpolated curvature distribution. This approach is summarized in Figure 1.

An advantage of our shape synthesis using curvatures is that the resulting shapes vary smoothly because the morphs are interpolated according to the original curvature distributions. Volume morphing, on the other hand, can hardly quantitatively specify this kind of shape change. Instead, it has to rely on careful feature selection which provides only qualitative shape change at best. By making explicit use of the curvature information, our method reduces the ambiguity of matching two shapes. This is especially important for smooth

1. We adopt the terms local curvature and curvature distribution here in a different way than the formal definition of curvature such as in [11]. Our local discrete curvature measure was first introduced in [5].

curved objects, where features are very difficult to identify manually. Compared with volume morphing, our method is more efficient.

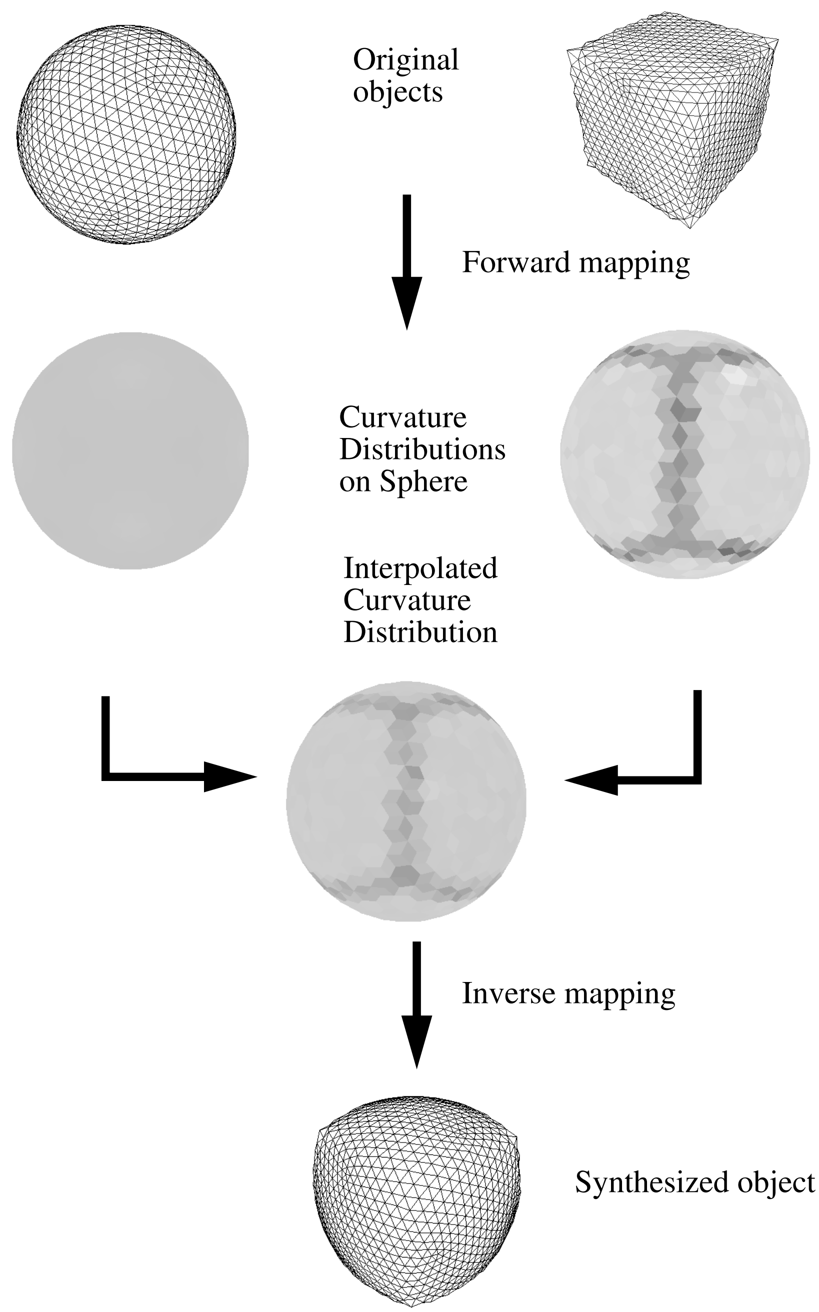


***Figure 1*** **Shape synthesis in curvature space: an example of a sphere and a hexahedron. The curvature has been color-coded so that the darker the bigger positive curvature and the lighter the bigger negative curvature.**

# 3 From Shape to Curvature Distribution: Forward Mapping

## 3.1 Spherical Representation of a Closed 3D Surface

To synthesize object shapes, one first has to find appropriate representations of those shapes. A natural discrete representation of a surface is a graph of nodes, or tessellation, such that each node is connected to each of its closest neighbors by an arc of the graph. For example, Szeliski and Tonnesen [17] used oriented particles to model arbitrary surface; Witkin and Heckbert [20] also used particles to sample surfaces. We use a special mesh each node of which has exactly three neighbors. Such a mesh can be constructed as the dual of a triangulation of the surface [5]. Our representation differs from particle systems [17][20] in that it restricts to the spherical topology and has a fixed neighborhood; but it leads to more efficient computation and robust local curvature approximation. To tessellate a unit sphere, we use a standard semi-regular triangulation of the unit sphere constructed by subdividing each triangular face of a 20-face icosahedron into $N^2$ smaller triangles. The final tessellation is built by taking the dual of the 20 $N^2$-face triangulation, yielding a tessellation with the same number of nodes.

In order to obtain a mesh representation for an arbitrary surface, we deform a tessellated surface until it is as close as possible to the object surface. The deformable surface algorithm drives the spherical mesh to converge to the correct object shape by combining forces between the data set and the mesh. Our algorithm originates from the idea of a 2D deformable surface [10][19] and is described in detail in [5]. The deformed surface can accurately represent concave as well as convex surfaces. Our deformable algorithm is not sensitive to deformation parameters such as the initial center and radius of the sphere. An example of a free-form object model created using the deformable surface and multiple view merging techniques [15] is shown in Figure 2. The deformation process is robust against data noise and moderate change of parameters such as initial sphere center and radius [15].

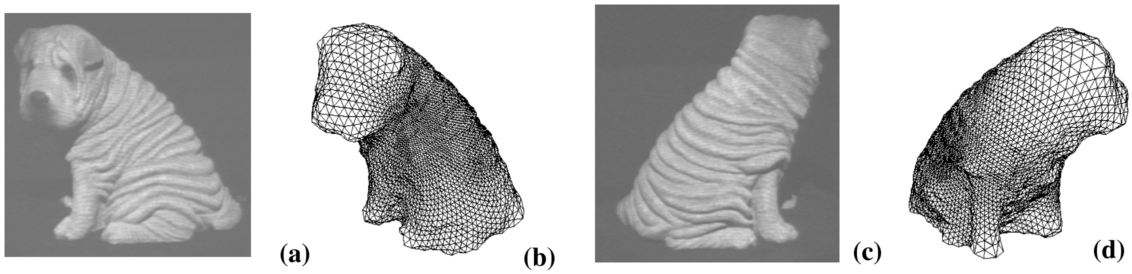


*Figure 2* **An example of a free-form object modeled from a deformable surface: (a) (c) Images of a sharpei; (b) (d) Deformable models of a sharpei**

The key idea of our spherical representation of a surface is to produce meshes in which the density of nodes on the object's surface is nearly uniform[1]. Although perfectly uniform distribution is impossible, a simple local regularity constraint can enforce a very high degree of uniformity across the mesh. We implement the local regularity constraint in the deformable surface algorithm such that each mesh has similar area as the others [5].

The local regularity constraint is a generalization to three dimensions of the regularity condition on two dimensional discrete curves; this condition simply states that all segments are of equal lengths. The difference between 2D and 3D is that it is always possible to create a uniform discrete curve in 2D, while only nearly uniform discrete surfaces can be generated in 3D. In practice, the variation of mesh nodes on the surface is on the order of 2% [5].

## 3.2 3D Local Curvature: An Approximation

After we obtain a nearly uniform surface mesh representation, the next step is to define a measure of curvature that can be computed from the surface representation. Conventional ways of estimating surface curvature, either by locally fitting a surface or by estimating first and second derivatives, are very sensitive to noise. This sensitivity is mainly due to the dis-

1. Koenderink warned that one has to be very careful of any method that uses the surface area of a polyhedral model (p.597 of [11]). Surface area depends on the way in which triangulations are done. In our previous work, we have shown how areas of different shapes are adjusted before comparison, in particular for partial views[5].

crete sampling and, possibly, to the noisy data. We introduced in [5] a robust measure of curvature computed at every node from the relative positions of its three neighbors. Our method is robust because all the nodes are at relatively stable positions after the deformation process. The deformable surface process serves as a smoothing operation over the possibly noisy original data. We call this measure of curvature the simplex angle.

The simplex angle $\varphi$ varies between $-\pi$ and $\pi$, and is negative if the surface is locally concave, positive if it is convex. Given a configuration of four points, the angle is invariant by rotation, translation, and scaling because it depends only on the relative positions of the points, not on their absolute positions.

The spherical representation can approximate not only free-form objects, but also polyhedral objects. For example, Figure 3 shows an example of a spherical polyhedral approximation of an octahedron with one concave face. Because of the regularity constraint, corners and edges are not represented perfectly. All plane surfaces, however, are well approximated even though the local regularity is enforced on all meshes. Different tessellation frequencies result in different approximations.

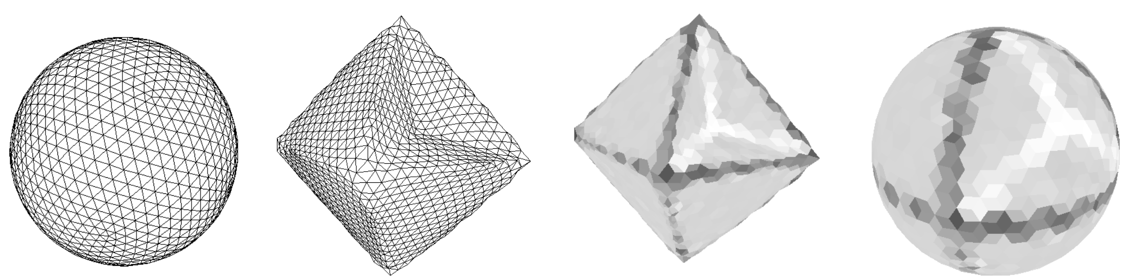

***Figure 3*** **(a) A spherical tessellation; (b) Deformable surface of an octahedron with a concave dent; (c) Local curvature on each mesh node; (d) Curvature distribution on spherical representation (The curvature on (c) and (d) is negative if it is light, positive if dark, zero if grey).**

# 4 From Curvature Distribution to Shape: Inverse Mapping

## 4.1 Shape Reconstruction

Now we know how to map object shape to its intrinsic representation defined above, i.e., a curvature distribution on a special spherical coordinate system. But does the inverse mapping exist? In other words, can we reconstruct the shape given its intrinsic representation?

We formulate this reconstruction as an optimization problem which minimizes the curvature difference between the known and reconstructed curvature distributions. The initial shape can be, for example, a sphere where constant curvature is at every mesh node. This minimization problem is, however, complicated because of the nonlinear and coupling nature of the local curvature computation. We devise an iterative method, similar to what has been used in deformable surface extraction, to solve this minimization problem. The motion of each vertex is modeled as a second order dynamics equation,

$$m\frac{d^2P_i}{dt^2} + k\frac{dP_i}{dt} = F_{int} \tag{EQ 1}$$

where $m$ is the mass unit of a node and $k$ is the damping factor. $F_{int}$ is the force which deforms the surface to have the assigned curvature distribution while remaining continuous. The above equation is integrated over time using finite differences. Starting from an initial shape, e.g., a semi-tessellated sphere, using Euler's method we can update each mesh node by

$$P_i^{(t)} = (1-k)\left(P_i^{(t-1)} - P_i^{(t-2)}\right) + F_{int}. \tag{EQ 2}$$

Notice that $F_{int}$ has to be updated at each iteration. The reader is referred to [3][5] for the details of how $F_{int}$ is computed. Figure 4 shows an example of the evolution of this reconstruction process. Given its curvature distribution, a toy sharpei is reconstructed from a sphere.

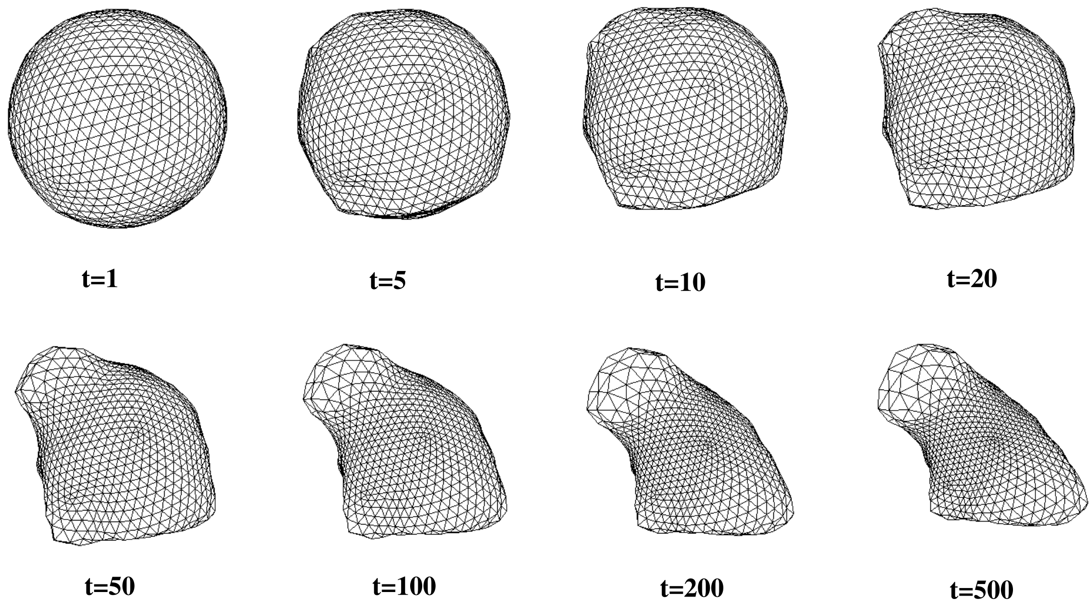


***Figure 4*** **An sequence of shapes at different steps of deformation. From sphere to sharpei.**

## 4.2 Regularization for the Underconstrained Minimization

The above minimization is unfortunately underconstrained. There are *3n* unknowns (3D coordinates at each mesh node of the unknown spherical mesh) but only *n* equations defined by local curvature at each mesh node. The object shape can not be determined without additional constraints. It is well-known that Gaussian curvature and mean curvature can not determine the shape. First fundamental forms are the necessary and sufficient condition for reconstructing shape [11].

### 4.2.1 Delingette's Constraint: metric parameters

One way to add more constraints is to record the relative positioning between each node and its three neighbors. Delingette [3] defined a so-called metric parameter set $\{\varepsilon_1, \varepsilon_2, \varepsilon_3\}$, such that

$$Q = \varepsilon_1 P_1 + \varepsilon_2 P_2 + \varepsilon_3 P_3, \tag{EQ 3}$$

$$\varepsilon_1 + \varepsilon_2 + \varepsilon_3 = 1, \tag{EQ 4}$$

where Q is the projection of $P$ on the plane $P_1P_2P_3$ as shown in Figure 5.

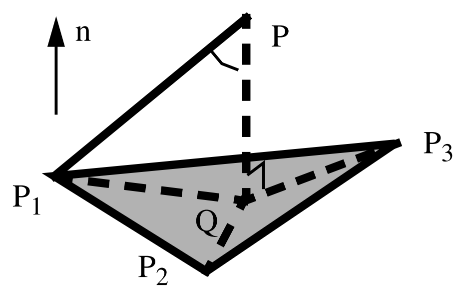


***Figure 5*** **Metric parameters relating a mesh node with its three neighbors.**

If the shape is known, i.e., if each mesh node $P$ and its three neighbors $P_1P_2P_3$ are known, the surface normal of the plane $P_1P_2P_3$ can be computed as

$$n = \frac{\overrightarrow{P_1P_2} \times \overrightarrow{P_1P_3}}{\left\| \overrightarrow{P_1P_2} \times \overrightarrow{P_1P_3} \right\|}. \tag{EQ 5}$$

From

$$\overrightarrow{QP} = \left\| \overrightarrow{QP} \right\| \vec{n} = \left( \overrightarrow{P_1P} \bullet \vec{n} \right) \vec{n} \text{ and } \overrightarrow{P_1Q} = \overrightarrow{P_1P} - \overrightarrow{QP} = \varepsilon_2 \overrightarrow{P_1P_2} + \varepsilon_3 \overrightarrow{P_1P_3},$$

we can easily solve the metric parameters from the following 2x2 linear equations

$$\begin{bmatrix} \overrightarrow{P_1P_2} \bullet \overrightarrow{P_1Q} \\ \overrightarrow{P_1P_3} \bullet \overrightarrow{P_1Q} \end{bmatrix} = \begin{bmatrix} \left\|\overrightarrow{P_1P_2}\right\|^2 & \overrightarrow{P_1P_2} \bullet \overrightarrow{P_1P_3} \\ \overrightarrow{P_1P_2} \bullet \overrightarrow{P_1P_3} & \left\|\overrightarrow{P_1P_3}\right\|^2 \end{bmatrix} \times \begin{bmatrix} \varepsilon_2 \\ \varepsilon_3 \end{bmatrix}. \quad \text{(EQ 6)}$$

The above equation provides us two equations at each mesh node. Thus we have *3n* sparse nonlinear equations for *3n* unknowns if we augment our intrinsic representation from $\{\varphi_i\}$ to $\{\varphi_i, \varepsilon_{2i}, \varepsilon_{3i}\}$ , $i = 0 \ldots n$ .

### 4.2.2 Regularity Constraint: a means of regularization

The metric parameter augmentation to our intrinsic shape representation may not be necessary. Instead, we introduce a regularity constraint in the shape reconstruction which forces each mesh node to be projected onto the center of its three neighbors. This regularity constraint implicitly gives a complete intrinsic description $\{\varphi_i, \frac{1}{3}, \frac{1}{3}\}$ , $i = 0 \ldots n$ . It shows that our intrinsic representation is sufficient to guarantee that the inverse mapping exists. This regularity constraint is, in spirit, similar to imposing a regularization term for ill-posed problems [18]. Given known local curvature, the expected location of a mesh node can be regularized through its three neighbors. As shown in Figure 6, we can then define $F_{int}$ as the scale distance between the current mesh node $P$ and its regularized position $P^*$

$$F_{int} = \alpha \cdot PP^*, \alpha = [0, 0.5]. \quad \text{(EQ 7)}$$

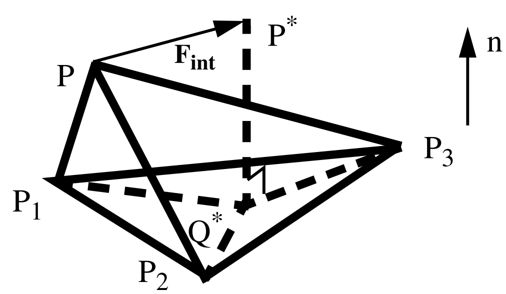


***Figure 6*** **Regularity and Internal force.** $\mathbf{Q^* = (P_1 + P_2 + P_3)/3}$**.**

Why does this regularity constraint ensure us the correct original shape? Obviously if we keep the intrinsic representation but pick an arbitrary set of metric parameters, we will reconstruct another shape which may be close to, but different from, the original shape. The reason is that this same regularity constraint has been enforced in the model extraction process when we construct an object model from range data [5][15]. At the end of the model extraction, the metric parameters converge to the expected value. Figure 7 shows the distribution of metric parameters of the sharpei model and their histograms. It clearly shows that the metric parameters are well regularized around its nominal values $\frac{1}{3}$.

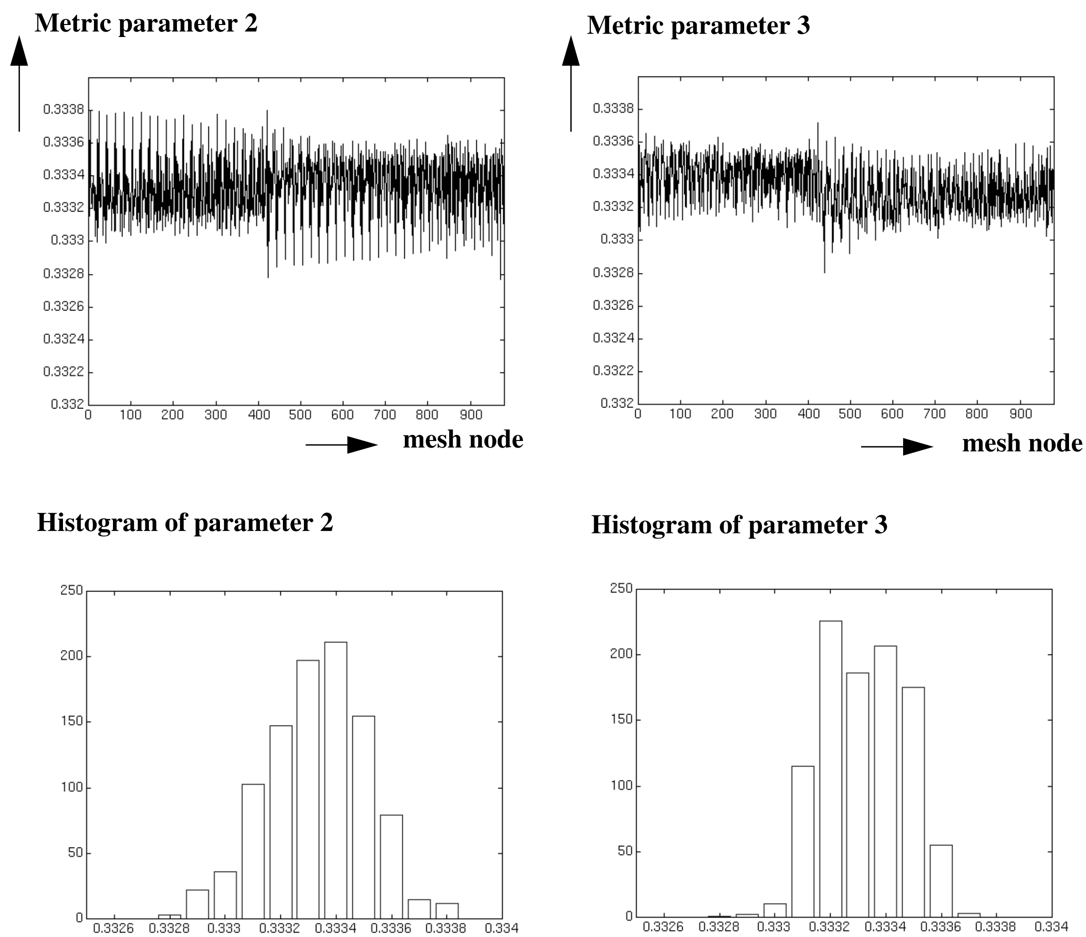


***Figure 7*** **Metric parameter distributions of a sharpei.**

## 4.3 Examples

The above analysis does not eliminate the local minima problem associated with our iterative process. Therefore the initial shape plays an important role for convergence to correct shape. To be fair, we start our reconstruction from a semi-tessellated sphere in all experiments. We show two examples of shape reconstruction: a polyhedral object and a free-form object in Figure 8 and Figure 9, respectively.

Both examples show very good reconstruction. The reconstructed shape has a very similar curvature distribution to that of the original one. The relative error in the SAI angle between the original and the reconstructed is less than 1% for each example. However, we do observe apparent shape discrepancy between the reconstructed and the original mainly due to the tessellation frequency, i.e., the discretization effect. Because the shape reconstruction is only up to an unknown rigid transformation (rotation and translation) and an unknown scale factor, constant area or constant volume [2] can be enforced during the reconstruction process.

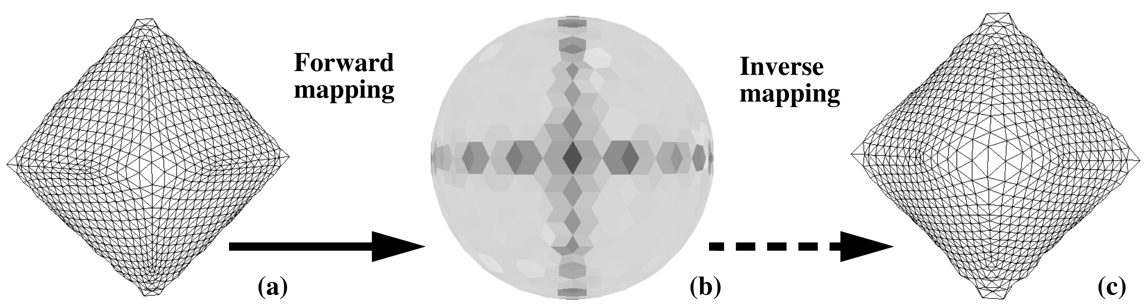


***Figure 8*** **An example of polyhedral object shape reconstruction (the solid arrow shows the forward mapping from shape to curvature, the dotted arrow shows the inverse mapping from curvature to shape): (a) Deformable surface of an octahedron; (b) Intrinsic representation or its curvature distribution (The curvature is negative if it is light, positive if dark, zero if grey). (c) The reconstructed shape from the curvature distribution (b).**

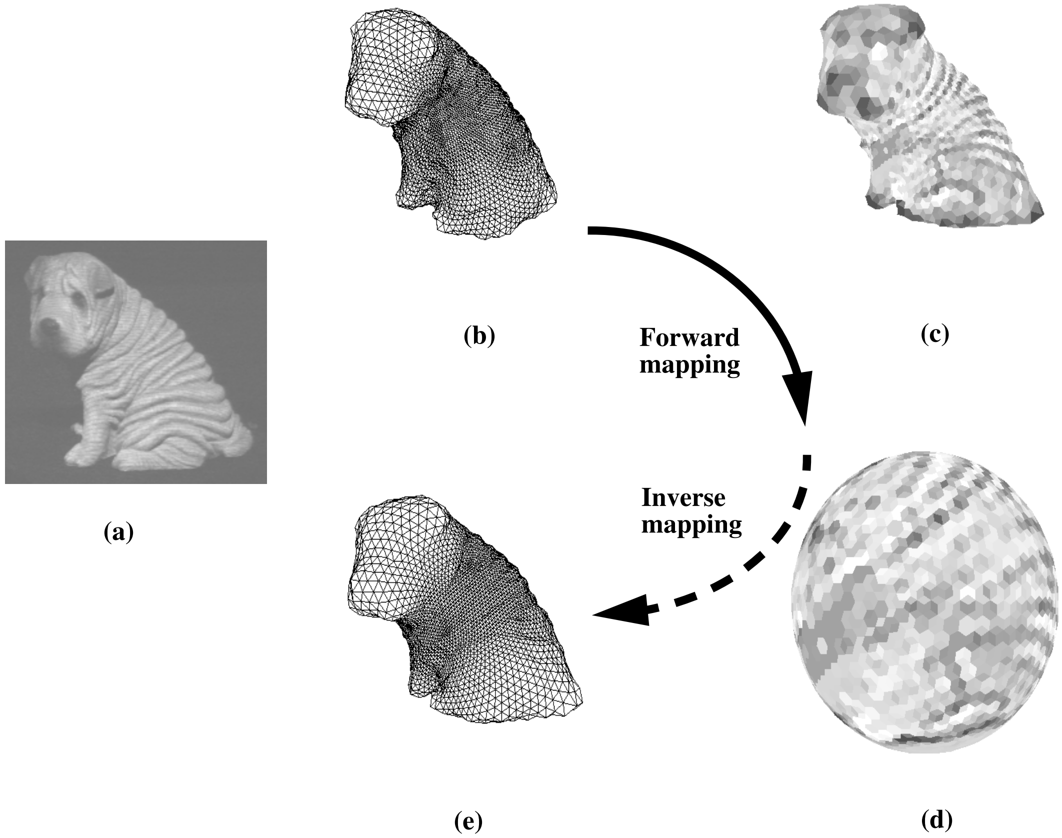


*Figure 9* **An example of free-form object shape reconstruction (the solid arrow shows the forward mapping from shape to curvature, the dotted arrow shows the inverse mapping from curvature to shape): (a) An image of a sharpei; (b) A sharpei model constructed from range data; (c) Local curvature distribution at each mesh node of a sharpei; (d) Intrinsic representation of a sharpei; (e) reconstructed sharpei model from its intrinsic representation.**

# 5 Shape Synthesis with Curvature Interpolation

## 5.1 Shape Correspondence

Given the intrinsic representations of two original shapes, we can compute correspondence between them based on two important properties of the semi-regularly tessellated sphere: that it has a fixed topology and that each mesh node has exactly three neighbors. We have observed [5] that the correspondence between two shapes is determined once three pairs of nodes are matched. In Figure 10, node *P* of the shape *A* corresponds *P'* of the shape *B*, and the two neighbors of *P* ($P_1$, and $P_2$) are put in correspondence with two of three neighbors ($P'_1$, $P'_2$ and $P'_3$) of *P'*, respectively. Figure 10 shows only *3* valid neighborhood matches, since each node has exactly three neighbors and the connectivity among them is always preserved. Moreover, the number of such correspondences is *3n* where *n* is the number of nodes of spherical tessellation [5].

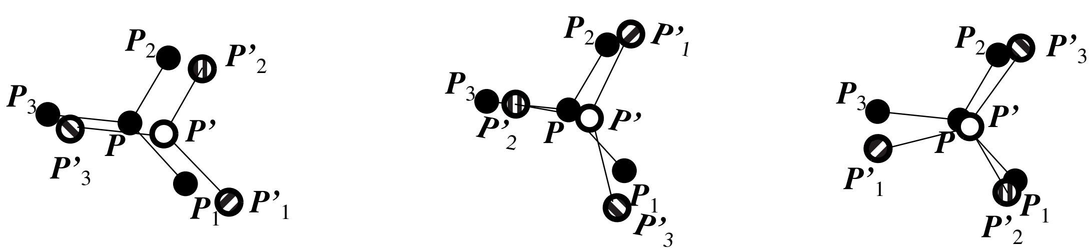


***Figure 10*** **Matching of neighbors from ($P_1$, $P_2$ $P_3$) to: (a) ($P'_1$, $P'_2$, $P'_3$); (b) ($P'_2$, $P'_3$, $P'_1$); (c) ($P'_3$, $P'_1$, $P'_2$) when P of shape *A* is matched to P' of shape *B*.**

Given the correspondence, the distance between two shapes is defined as the $L_P$ distance between two curvature distributions [16]. An efficient algorithm has been devised for comparing two shapes so that the minimum distance between two shapes, also defined as a shape similarity measure [16], represents the best matching.

## 5.2 Shape Similarity for Interpolation

It is unclear how to interpolate two shapes unless we know how to compare them. It is very difficult to compare two 3D shapes because of the unknown scale factor and rigid transformation. We have shown [16] that it is possible to quantitatively measure the distance between two shapes using the intrinsic representation. Thus we can also interpolate these two shapes to obtain a new curvature distribution from the intrinsic representations of two original shapes and their correspondence. An advantage of this approach is that it shows quantitatively how much the morph is different from the originals. For example, at each mesh node of the new mesh $C$, its curvature can be computed by a linear interpolation of its counterparts in the original shapes $A$ and $B$. More specifically,

$$\varphi_{C_i} = (1-t)\,\varphi_{A_i} + t\varphi_{B_i},\ t \in [0, 1],\ i = 1...\mathrm{n}, \qquad \text{(EQ 8)}$$

where $\varphi$ is the local curvature measure. Alternatively nonlinear cross-dissolve techniques can also be used. For instance, if we use the following interpolation function,

$$\varphi_{C_i} = \left(1 - 2t^2\right)\varphi_{A_i} + 2t^2\varphi_{B_i},\ t \in [0, 0.5],\ i = 1...\mathrm{n}, \qquad \text{(EQ 9)}$$

$$\varphi_{C_i} = 2(1-t)^2\varphi_{A_i} + \left(1 - 2(1-t)^2\right)\varphi_{B_i},\ t \in [0.5, 1],\ i = 1...\mathrm{n}, \qquad \text{(EQ 10)}$$

to blend curvatures, we can enforce desirable small shape change at initial steps. Figure 11(a) shows linear change of a shape similarity distance between each morph and the toy sharpei, while (b) shows the distance between each morph and the toy pig. Figure 12 shows the nonlinear change of a shape similarity distance. The distance between two shapes is defined as the $L_2$ norm of the distance between two curvature distributions, which has also been used as a measure of shape similarity in our previous work [16]. It is clear that our synthesis approach is not only consistent with, but also can be controlled by, our metric measure of shape similarity.

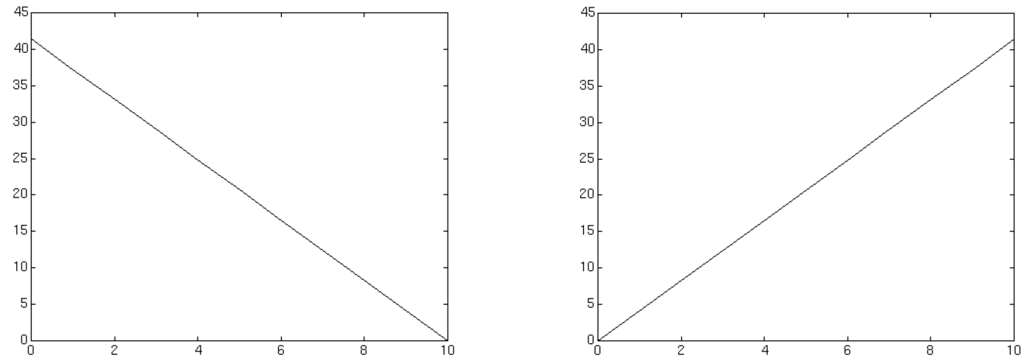


***Figure 11*** **Linear interpolation of shape distance between the morph sequence and two originals (a) sharpei; (b) pig.**

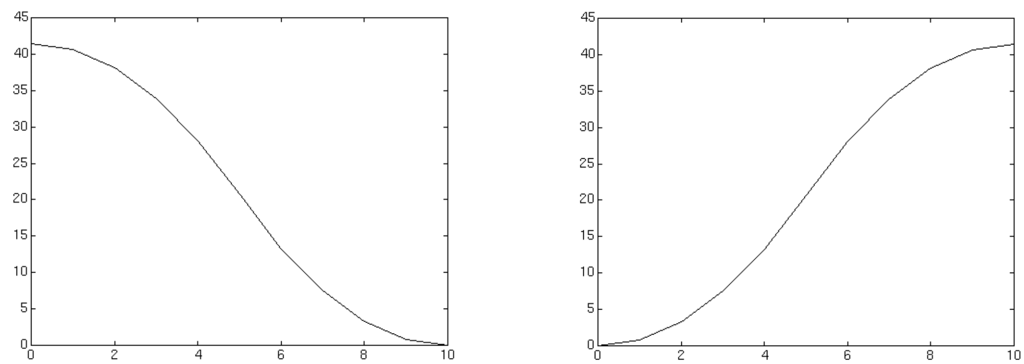


***Figure 12*** **Nonlinear interpolation of shape distance between the morph sequence and two originals (a) sharpei; (b) pig.**

## 5.3 Result and Discussion

Figure 13 shows a sequence of morphs which are synthesized from a toy sharpei and a toy pig, while Figure 14 shows a different view of the same sequence. The models of these free-form objects are constructed from real range images using methods described in [15]. The frequency of spherical tessellation is set to 13, which means that the total number of meshes is 3380. After the models and their curvature distributions are obtained, we use linear interpolation to generate the intermediate curvature distributions which are then inversely mapped to the morphs. These morphs clearly show a gradual and smooth shape transition between sharpei and pig.

Compared with volume morphing, our method is fast and simple to implement. While volume morphing takes more than a day [12] to render a sequence of morphs, our method takes less than an hour to do the shape synthesis. In our experiments, it takes 20 minutes to build deformable models and curvature distribution, and 20 minutes for cross-dissolve curvature interpolations and shape synthesis of a sequence of 10 morphs on a SUN Sparc 20. This does not account for the time taking images and rendering images for the final display. To speed up the iteration process, the morph from last step is used to start next morphing step.

One possible drawback of our approach is that the quality of approximation of a polyhedral or free-form surface depends on the number of patches chosen. For example, with frequency 7 semi-regular spherical tessellation, we have 980 surface patches; when the frequency is 13, we have 3380 patches. Obviously, the more surface patches we use, the better the approximation. Our shape synthesis approach generates good morphing sequence provided that a sufficient number of tessellations is adopted.

For the purpose of shape synthesis, we can also incorporate any user-specific force $F_{ext}$ in the (EQ 1) such that

$$m\frac{d^2P_i}{dt^2} + k\frac{dP_i}{dt} = F_{int} + F_{ext} \tag{EQ 11}$$

which is similar to deformable surface model extraction [5] where $F_{ext}$ is dominated by data force.

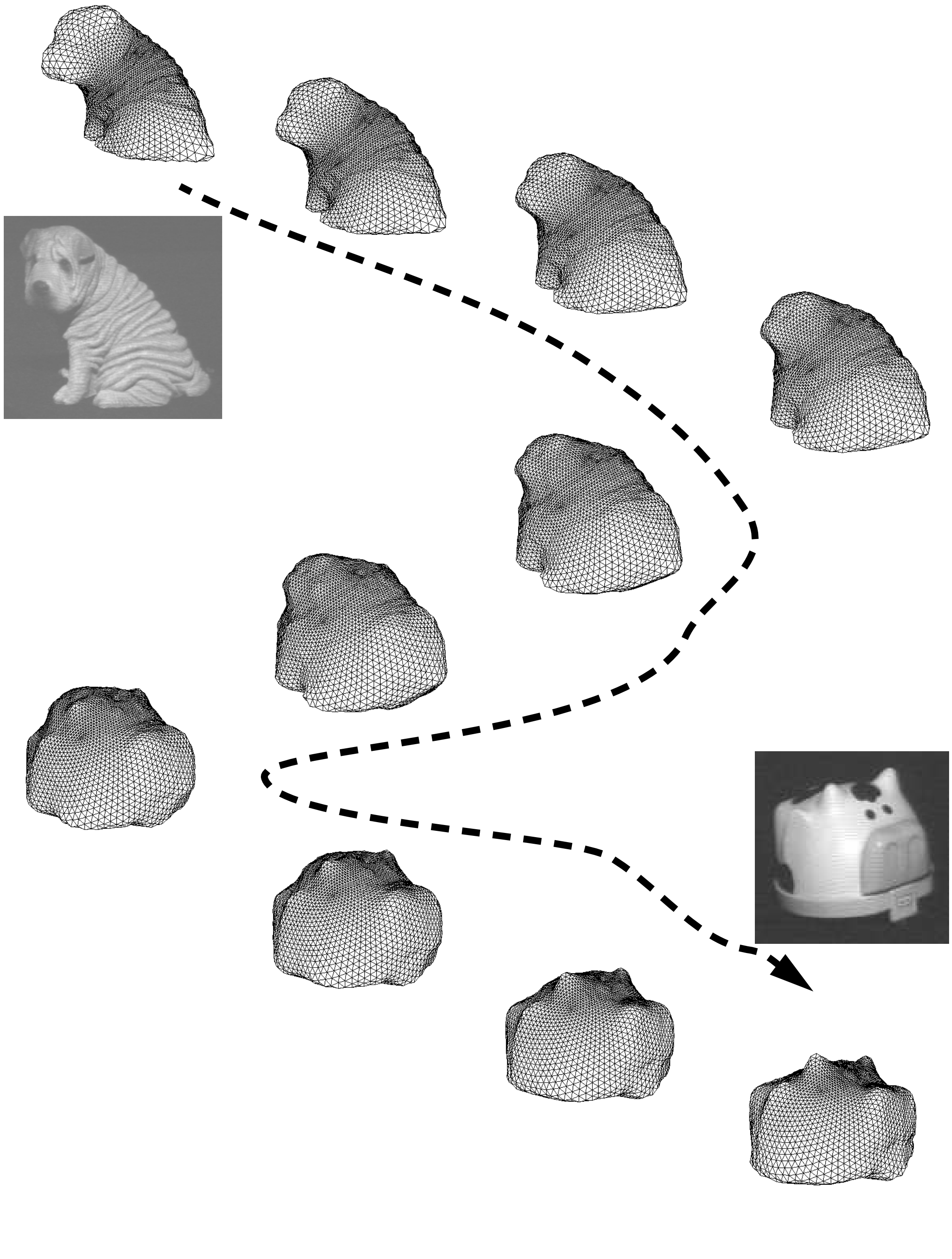

***Figure 13*** **A morphing sequence between a toy sharpei and a toy pig.**

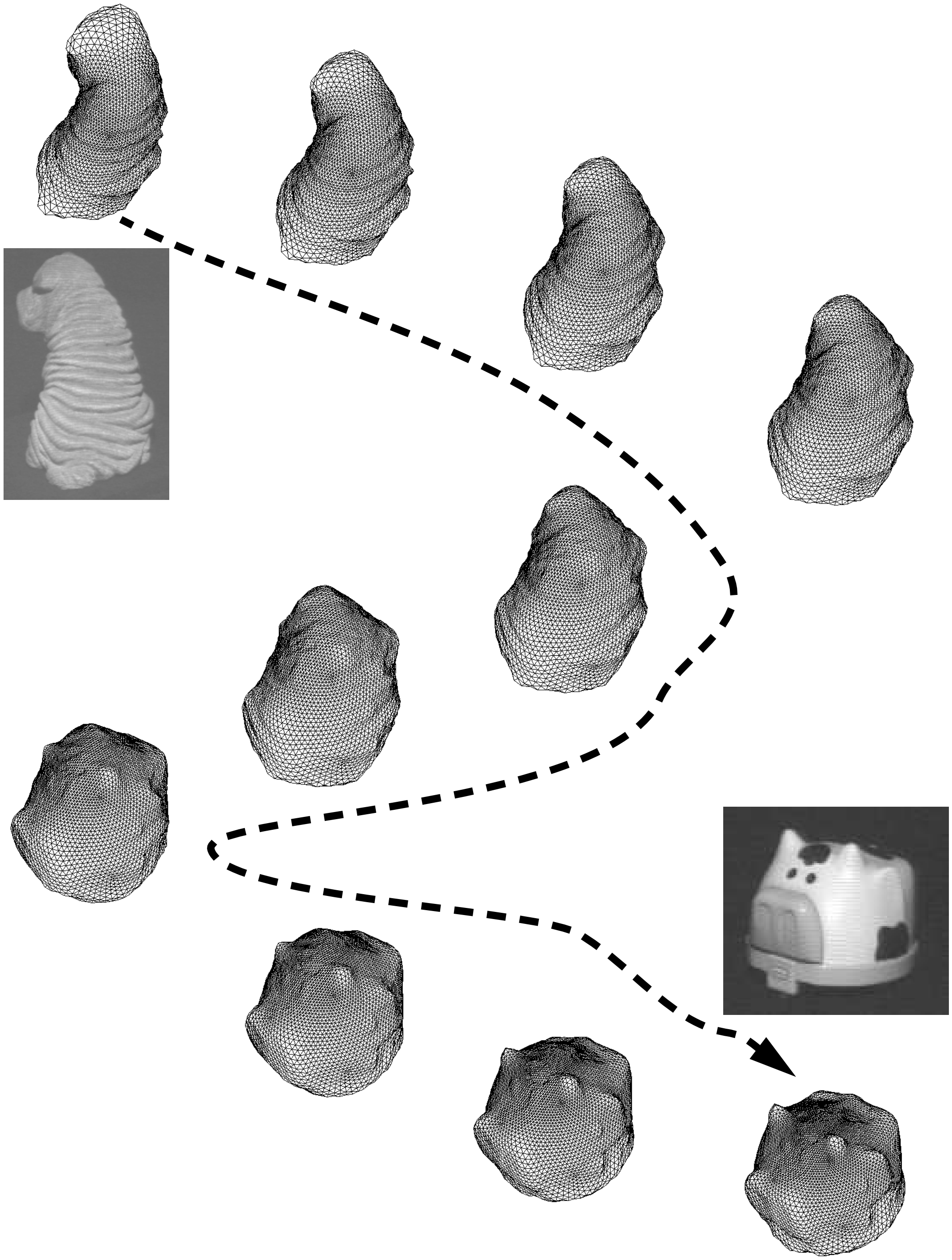

*Figure 14* **Another view of the morphing sequence between a toy sharpei and a toy pig.**

# 6 Conclusion

We have described a novel approach to synthesizing shapes using curvature distributions. To store an object shape of genus zero, either free form or polyhedral, we use a spherical mesh with known connectivity among its nodes. Our spherical mesh, which starts with a semi-tessellated sphere, is iteratively deformed to converge to the original object shape while maintaining nearly uniform distribution with known connectivity among mesh nodes. The local curvature computed at each node captures the averaged curvature information in its vicinity. This curvature distribution, i.e., the result of forward mapping from shape space to curvature space, is used as the intrinsic shape representation because it is invariant to rigid transformation and scale factor. Furthermore, with regularity constraints on the mesh, the inverse mapping from curvature space to shape space always exists and can be recovered using an iterative method. We have shown that the shape can be reconstructed, from the spherical curvature distribution only, up to a scale factor and a rigid transformation. Therefore, the task of synthesizing a morph from two original shapes is essentially one of interpolating two known curvature distributions generated from the deformed meshes generated from original shapes. This curvature-based interpolation yields smooth curvature migration along the morphing sequence. Experiments show that our shape synthesis creates realistic and intuitive shape morphs which show a gradual change between two originals.

Currently our approach is restricted to genus zero shape topology. We can modify the mesh representation to accommodate the arbitrary topology as in [2]. With the appropriate user interface, we can also incorporate the user's specification for even more realistic and specific shape synthesis. As shown in the paper, this kind of user-specified constraint can be easily incorporated in our framework.

# Acknowledgement

We thank Marie Elm for proofreading different versions of this paper.


# References


[1] T. Beier and S. Neely. Feature-based Image Metamorphosis. Proc. SIGGRAPH'92, pp. 35-42, July, 1992.

[2] H. Delingette, H. Watanabe and Y. Suenaga. Simplex Based Animation. Tech report NTT Human Interface Lab. Japan, 1994.

[3] H. Delingette. Simplex Meshes: a General Representations for 3D Shape Reconstruction. INRIA report 2214, 1994.

[4] T. He, S. Wang and A. Kaufman. Wavelet-based Volume Morphing. Proc. Visualization'94, pp. 85-91. 1994.

[5] M. Hebert, K. Ikeuchi and H. Delingette. A Spherical Representation for Recognition of Free-form Surfaces. IEEE Trans. PAMI Vol. 17 No. 7, pp. 681-690, 1995.

[6] B.K.P. Horn. Extended Gaussian Image. Proc. of IEEE, Vol. 72, No. 12, pp. 1671-1686, December, 1984.

[7] J. Hughes. Scheduled Fourier Volume Morphing. Proc. SIGGRAPH, pp. 43-46, July, 1992.

[8] K. Ikeuchi and M. Hebert. From EGI to SAI. CMU-CS-95-197.

[9] S. Kang and K. Ikeuchi. The Complex EGI: New Representation for 3D Pose Determination. IEEE Trans. PAMI Vol. 15, No. 7, pp. 707-721, July, 1993.

[10] M. Kass, A. Witkin and D. Terzopoulos. Snakes: Active Contour Models. Int'l. J. Computer Vision. Vol. 1, No. 4, pp. 321-331, January, 1988.

[11] J. Koenderink. Solid Shape. The MIT Press, Cambridge, 1990.

[12] A. Lerios, C. Garfinkle, and M. Levoy. Feature-based Volume Metamorphosis. Proc. SIGGRAPH'95, pp.449-456, August, 1995.

[13] J. Little. Determining Object Attitude for Extended Gaussian Image. Proc. IJCAI, Los Angeles, California, pp. 960-963, August, 1985.

[14] W. Lorensen and H. Cline. Marching Cubes: A High Resolution 3D Surface Construction Algorithm. Proc. SIGGRAPH'87, pp. 163-169, July, 1987.

[15] H. Shum, M. Hebert, K. Ikeuchi and R. Reddy. An Integral Approach to Free-form Object Modeling. CMU-CS-95-135, May 1995.

[16] H. Shum, M. Hebert, and K. Ikeuchi. On 3D Shape Similarity. CMU-CS-95-212, November, 1995.

[17] R. Szeliski and D. Tonnesen. Surface Modeling with Oriented Particle Systems. Proc. SIGGRAPH'92, pp. 185-194, July, 1992.

[18] D. Terzopoulos. Regularization of Inverse Visual Problems Involving Discontinuities. IEEE Trans. PAMI, Vol. 8, No. 4, pp. 413-424, July, 1986.

[19] D. Terzopoulos, A. Witkin and M. Kass. Symmetry-Seeking 3D Object Recognition. Int. J. Computer Vision. Vol. 1, No. 1, pp.211-221, 1987.

[20] A. Witkin and P. Heckbert. Using Particles to Sample and Control Implicit Surfaces. Proc. SIGGRAPH'94 , pp. 269-278, July, 1994.