Blending Shapes by Using Subdivision Surfaces

Ryutarou Ohbuchi [a,*], Yoshiyuki Kokojima [b], Shigeo Takahashi [b]

[a] *Yamanashi University, 4-3-11 Takeda, Kofu, Yamanashi 400-8511, Japan*

[b] *Gunma University, 1-5-1 Tenjin-cho, Kiryu, Gunma 376-8515, Japan*

Abstract

This paper presents a shape-blending algorithm that interpolates between 2D and 3D polyhedrons. Shape blending, which is sometimes called shape metamorphosis or geometric morphing, has applications in such areas as entertainment and medical visualization.

Our algorithm directly interpolates vertices of polyhedral source shapes by using variationally optimized subdivision surfaces. To interpolate a pair of 3D polyhedrons, for example, a smooth 4D tetrahedral interpolator subdivision surface is created. Intersecting the 4D subdivision surface with another 4D surface produces a blended 3D mesh. Variational optimization of the interpolator surface ensures a smooth shape transition. At the same time, manipulable nature of the interpolator subdivision surface allows for feature correspondences, shape transition effects, and other controls over the shape blending.

Key words: Geometric modeling, Geometric morphing, Shape metamorphosis, Shape-interpolation control Tetrahedral mesh generation,

1 Introduction

The shape blending problem can be stated as follows; given two or more source shapes, construct a sequence of interpolated shapes so that blended shapes adjacent to each other in the sequence are geometrically close. Shape blending is sometimes called shape metamorphosis, geometric morphing, or shape interpolation. The technology can be applied to create stunning visual effects in movies and TV advertisements, for example.

Shape blending has been studied since around 1980s, and many algorithms have been published. An excellent review on 3D shape morphing by Lazarus and Verroust [1] notes that there are two major classes of approaches to 3D shape blending; the *boundary-representation based approach* and the *volume-based approach.*

1. Boundary representation based approach: This approach works in a parameter domain defined on the surface of objects. Polygonal mesh representation has been the boundary representation of choice for morphing. Most of the algorithms in this class create a common mesh, which is a common embedding of both of the source meshes. The common mesh is then geometrically deformed to produce morphed shapes. The common mesh is found typically on a spherical or a rectangular parametric domain.

The advantage of the boundary-representation based approach is its ability to control morphing. Feature correspondence can be established, for example, through vertex-to-vertex or mesh-to-mesh correspondence. A disadvantage of this approach is its difficulty in morphing between shapes having different surface topology (i.e., different genera

* Corresponding author. Tel.: +81-55-220-8570
Email adresses: ohbuchi@acm.org (Ryutarou Ohbuchi), kokojima@ail.cs.gunma-u.ac.jp (Yoshiyuki Kokojima), takahashis@acm.org (Shigeo Takahashi).
.

and/or connectivity), for the approach requires strict correspondence between vertices and edges of source shapes.

Kent, et al. [2] employed spherical embedding, while Kanai, et al. [3] employed embedding into a disk by using harmonic mapping. For feature correspondence, Gregory et al. [4] and Kanai et al. [5] employed coarse meshes overlaid on the source meshes to let users specify mesh-to-mesh feature correspondence. Lee, et al. [6] introduced multiresolution reparameterization of polygonal meshes to morphing. Their decomposition, an application of MAPS mesh reparameterization algorithm [7], is so that the feature lines and points specified in their original (i.e., highest resolution) meshes are preserved in the lowest resolution mesh. Consequently, feature correspondences can be established in the simplest, lowest resolution mesh.

As mentioned, this class of algorithm is not suited for interpolating shapes having different topology. A paper by DeCarlo and Gallier [8] is the only example we know of in this class that explicitly dealt with interpolation of shapes having different topologies.

2. Volume-based approach: This approach employs level sets of distance functions computed from the source shapes for morphing. These distance functions are interpolated to produce blended shapes.

A major advantage of this approach is its ability to morph easily between shapes having different surface topology. However, it lacks detailed control over morphing available in the boundary-representation based approach. Establishing feature correspondence and controlling shape transition are difficult. Another disadvantage of this approach is loss of quality, e.g., loss of sharp features, due to approximation of shapes using smooth distance functions.

Many algorithms in this class employed voxel representation of shapes, which can be considered as a 0-level set of a discretized distance function. Lerios et al. [9] extended feature-based 2D image morphing by Beier and Neely [10] to 3D. Hughes [11] brought the voxels into the Fourier domain, and He et al. [12] into the Wavelet-transformed domain, for morphing. Some algorithms employed signed distance functions in 3D to interpolate a set of scattered points in 3D and a set of 2D contours to construct 3D shapes [13,14]. Kaul and Rossginac [15] used weighted Micowski sum for morphing. They introduced a concept of influence shape to control shape transition, albeit indirectly. Payne and Toga [16] and Cohen-Or, et al. [17] also used signed distance function. Whitaker and Breen [18] introduced the idea of evolution equations to morphing. Turk and O'Brien [19] employed four-dimensional (4D) variational implicit functions to smoothly interpolate a pair of 3D distance functions computed from 3D source shapes. The 4D interpolator is then intersected with a plane to produce a 3D implicit function of the interpolated shape. Boundary (polyhedral) representation of the interpolated shape is then recovered by using an iso-surface extraction algorithm.

1.1 Our Approach

This paper presents a new shape blending algorithm for shapes defined as polyhedrons. The algorithm interpolates the polyhedrons, the "source" shapes, by using a subdivision surface having a dimension one higher than the source shapes. The subdivision surface is made to smoothly interpolate vertices of the source shapes by using variational optimization. Intersecting the interpolator subdivision surface with another surface produces a blended shape. Smooth shape transitions are achieved due to the use of the variational optimization. It should be noted that what is called a subdivision surface here is different from a typical subdivision surface. Of two components of a typical subdivision surface scheme, geometric smoothing and connectivity refinement, our method borrows the connectivity refinement only. Our scheme generate vertex coordinates using methods different from typical subdivision surface schemes.

Thanks to the variationally optimized subdivision surface used for the interpolation, the algorithm combines some of the advantages of the several algorithms in both of the two classes of approaches mentioned above. As in the case of algorithms that employ variationally optimized smooth implicit functions, our method is able to produce smooth shape transition. At the same time, similar to some of the boundary representation based methods, our method allows for various controls over the interpolation. This controllability is due to manipulability of the subdivision surfaces employed for the shape interpolation. For example, feature correspondences among vertices of source meshes can be established by using line geometric constraints. Our interpolation method also allows for other shape transition effects, such as spatially non-uniform shape transition and transiently exaggerated shapes. An additional advantage of our algorithm is its potential to blend source shapes having different surface topology. This is due mainly to the use of the interpolator surface whose dimension is one higher than that of the source shapes. While our current implementation limits such topology transcending shape blending be applicable only to 2D contours, future extensions should enable blending of 3D source shapes having different surface topology.

The following list summarizes important features of our approach.

(1) **Feature correspondence:** Direct point-to-point correspondence can be established between the source and target meshes by using constraints.
(2) **Smooth blending:** Shape transitions as well as blended shapes are smooth since source shapes are interpolated by using a smooth, variationally-optimized subdivision surface.
(3) **Sharp source features:** Sharp as well as smooth shape features can be represented in the source shapes.
(4) **Shape transition control:** Various shape transition effects can be incorporated. For example, spatially non-uniform shape transitions and exaggerated blended shapes can be achieved.

The rest of this paper is structured as follows. The shape blending algorithm is described in Section 2. While our emphasis in this paper is on the 3D shape blending algorithm, we also describe the 2D shape blending algorithm for it helps us to illustrate our approach. Section 2 also includes some of the results generated by our shape-blending algorithms. Additional results are presented in Section 3. We conclude the paper in Section 4 with the summary and future work.

2 The Shape Blending Algorithm

Our shape-blending algorithm interpolates source shapes defined as n-dimensional polyhedrons by using an $(n+1)$-dimensional subdivision surface [20–25] as an interpolator. The subdivision surface is globally optimized using a variational method so that it is smooth. If more than one source shape is given, our algorithm could interpolate them using a globally smooth subdivision surface. The smoothing subdivision rule of the standard subdivision surfaces may be modified, if necessary, to create sharp shape features in the source and blended shapes.

In the following of this paper, we call an n-dimensional source mesh *S-mesh* and an $(n+1)$-dimensional interpolator mesh *I-mesh*. We call the axis of interpolation *blending axis*, denoted by t, although it has little to do with time. The spatial axes in 2D are denoted by x and y, and, those in 3D are denoted by x, y, and z.

2.1 2D Shape Blending Algorithm

Assume that each source mesh (S-mesh) is a 2D polygonal contour defined on the x-y plane. Multiple S-meshes are located at different values of the blending axis t. An interpolator mesh (I-mesh) that interpolates S-meshes is a 3D triangular mesh. Blending of 2D shapes is achieved by following the steps below:

1. **Initial meshing:** Given S-meshes, each of which is defined as a 2D polygonal contour, the algorithm creates an initial 3D triangular I-mesh by connecting vertices of an adjacent pair of S-meshes on the blending axis t.
2. **I-mesh subdivision:** The initial triangular I-mesh is subdivided by applying a 1-to-4 subdivision rule, that is topologically identical to Loop's scheme [22]. The subdivision gives the I-mesh topological complexity enough for smooth interpolation of complex shapes.

3. **Variational optimization:** The subdivided triangular I-mesh is variationally optimized so that the I-mesh satisfies a given set of geometric constraints. Constraints usually include surface smoothness and end-point (that is, S-mesh) interpolation. In addition, vertex-to-vertex feature correspondence and other controls over the blending may also be given as geometric constraints to the optimization.
4. **Shape extraction:** The 3D I-mesh is intersected with a 3D surface to extract an interpolated (or a blended) 2D contour. The surface may be a plane at $t =$ const., or a curved surface defined as a function of x, y, and t.

When blending a pair of 2D polygonal contours, the initial mesh can be created automatically using a method described by Shinagawa et al. [26]. Given a pair of polygonal contours, the method finds pairs of closest vertices from the contours.

Figure 1 shows an example of 2D shape blending. Figure 1(a) shows a pair of source contours to be interpolated, outlines of letters "7" and "8", with a *topological key-shape* in the middle. Figure 1(b) is the initial interpolating triangular mesh that connects two 2D polygonal contours. Figure 1(a) is the interpolating subdivision surface produced as a result of variational optimization. Intersecting this smoothed mesh with a $t =$ const. surface will produces a blended shape.

The topological key-shape, an additional outline inserted in the middle of the blending axis, guides topological evolution of shapes between initial contours having different topology. A topological key-shape is inserted at each topological juncture in order to uniquely guide topological evolution of shapes. In the example of Figure 1, the letter "8" has two interior loops while the letter "7" has none. The key-shape in this example guides the topological evolution by relating the A1 with B1, A2 with B2, A3 with B3, and B3 with C3. Without the guidance, there are many equally feasible topological configurations of the I-mesh.

Please note that the key-shape is to guide topology only; its geometry (vertex coordinate) can be arbitrary. (It is true, however, that a reasonable key-shape geometry leads to a faster computation of the I-mesh.) Note also that the vertex topology of the source shapes and the key-shape need not correspond exactly. The differences in vertex topology are taken care of by the I-mesh connectivity subdivision, which increases degrees-of-freedom of the I-mesh.

2.2 3D Shape Blending Algorithm

In the case of 3D shape blending, each source shape (S-mesh) is a 3D polyhedral mesh. The S-meshes are interpolated by an I-mesh, which is a 4D tetrahedral mesh. While basic steps are similar, the 3D shape-blending algorithm is significantly more involved than its 2D counterpart. Much of the additional difficulty lies in the initial meshing stage. Unlike a 3D triangular mesh connecting 2D polygonal contours, it is not straightforward to create a tetrahedral mesh connecting a set of arbitrary polygonal meshes. Our approach to this difficulty of initial meshing is to "start simple and refine later".

The tetrahedral mesh creation is relatively easy if the source meshes are simple enough. As an extreme example, a tetrahedral mesh between a pair of tetrahedra can be found quite easily. Our algorithm first simplifies the S-meshes using wavelet analysis. Prior to the wavelet analysis, source meshes are reparameterized, if necessary, so that the resulting mesh has 1-to-4 subdivision connectivity required for the wavelet analysis. The S-mesh reparameterized to have 1-to-4 subdivision connectivity is called a TS-mesh. Then, for each source mesh, the lowest resolution source mesh produced by the wavelet analysis is used to create a *base* or *level-0* tetrahedral I-mesh. An algorithm to create the base tetrahedral mesh will be explained in the next section.

The base tetrahedral I-mesh is topologically and geometrically refined into a fully featured tetrahedral I-mesh. This recursive refinement of vertex connectivity employs a 1-to-4 subdivision rule for the triangular S-meshes and a 1-to-8 subdivision rule for the tetrahedral I-mesh. The 1-to-4 subdivision rule our algorithm employed is topologically identical to Loop's scheme [22] for triangular meshes (Figure 6). The 1-to-8 subdivision rule for tetrahedral meshes is the "symmetric subdivision" rule of Moore [27] illustrated in Figure 7. At each subdivision step, detail coefficients from the wavelet analysis are used to provide geometric refinement.

A full-resolution initial I-mesh is created when the wavelet analysis performed on the source TS-meshes is fully reversed through the refinement. The refinement process using the wavelet coefficients reverts the source meshes to their original connectivities and geometries. The coordinates of vertices inserted to create the level-n I-mesh are computed recursively from those of the level-(n-1) I-mesh by using linear interpolation. Note that these coordinate values are initial values to be modified by the succeeding variational optimization step. The coordinates of vertices in the source meshes are to remain unchanged.

When the initial meshing is complete, subdivision of the I-mesh and variational optimization follows. Blended shape is extracted by intersecting the variationally optimized I-mesh with another surface.

Blending of 3D S-meshes is accomplished by following the steps below (see Figure 2):

1. **Initial meshing:** Create an initial 4D tetrahedral I-mesh from a given set of 3D polyhedral S-meshes. This is accomplished by (1) simplifying the S-meshes, (2) start a base tetrahedral I-mesh from the simplified S-meshes, and (3) systematically grow a complex I-mesh out of the base I-mesh.
   A. **Multiresolution analysis (MRA):** Each S-mesh is wavelet-analyzed to create a multiresolution (MR) representation of the S-mesh using the framework of Lounsbery et al. [28]. For each of the S-meshes, the analysis produces a base mesh and multiple levels of detail coefficients. Since the wavelet analysis assumes a triangular mesh with 1-to-4 subdivision connectivity as its input, the S-mesh may require reparameterization. To reparameterize, we employ *Multiresolution Adaptive Parameterization of Surfaces (MAPS)* algorithms by Lee et al. [7]. The S-mesh reparameterized to have 1-to-4 subdivision connectivity is called a TS-mesh.
   B. **Base I-mesh creation:** For each of the TS-meshes, the lowest resolution (i.e., the simplest) TS-mesh produced by the MR analysis above is used to create a base tetrahedral I-mesh that interpolates TS-meshes. The base tetrahedral I-mesh is created by connecting vertices of a pair of the lowest-resolution TS-meshes. (Details of the base tetrahedral I-mesh creation will be explained in Section 2.2.1.)
   C. **I-mesh refinement:** The base TS-meshes are recursively refined, both topologically and geometrically, by using multiple levels of detail coefficients produced by the MRA in the step 1A above. As the polygonal TS-meshes are refined, the tetrahedral I-mesh is also refined accordingly. The 1-to-4 subdivision connectivity of the TS-meshes enables systematic topological refinement of the tetrahedral I-mesh while maintaining the I-mesh's 1-to-8 subdivision connectivity by using the Moore's subdivision rule.
2. **I-mesh subdivision:** If necessary, both I-mesh and TS-meshes are subdivided to increase their topological complexity. Increased topological complexity gives the meshes increased degrees-of-freedom necessary to smoothly approximate complex shapes that could occur in the tetrahedral I-mesh.
3. **Variational optimization:** Refined and subdivided tetrahedral I-mesh is variationally optimized so that it satisfies various geometric constraints. Constraints typically include smoothness and end-point (i.e., source shape) interpolation. In addition, constraints used for various shape-blending effects may be included in the optimization. These shape-blending effects, including feature correspondence and spatially non-uniform shape transition, are among the major advantages of the method proposed in this paper.
4. **Shape extraction:** The I-mesh is intersected with a surface to extract an interpolated shape. The surface may be a $t =$ const. plane, or a curved surface defined as a function of x, y, z, and t.

The variational optimization step, whose details will be explained in Section 2.3, employs a framework similar to wavelet based multiresolution analysis and synthesis in order to optimize for different sets of geometric constraints at multiple resolution levels. Note that the multiresolution framework for the variational optimization of I-meshes is different from the one used for reparameterizing S-meshes.

Figure 3 shows an example of I-mesh creation. Figure 3(a) is the base I-mesh, created by tetrahedral-meshing a pair of level-0 TS-meshes. The level-0 TS-mesh of the star shape (an icosahedron) contains 12 vertices (20 triangular-faces, 30 edges), while the base mesh for the mannequin head contains 10 vertices (16 triangular-faces, 24 edges). A refinement step produces the I-mesh at the resolution level 1 shown in Figure 3(b), and three refinement steps create the I-mesh at resolution level 3 shown in Figure 3(c).

Note in Figure 3 that, to visualize a 4D tetrahedral mesh in 2D projections, one of the spatial axes z and the blending axis t are overlaid. Similar overlaying of coordinate axes will be employed throughout this paper to visualize 4D meshes in 2D projections.

2.2.1 *Creating Base Tetrahedral Mesh*

The base tetrahedral I-mesh is created from a pair of the lowest resolution TS-meshes by using the algorithm below. Let the two triangular S-meshes be M_s and M_t. If we assume that a graph G_s (a dual of the mesh M_s) and a graph G_t (isomorphic to G_s) can be embedded in the meshes M_s and M_t, respectively, then M_s and G_t can be meshed together by a set of tetrahedra. Figure 4 illustrates the relationships between M_s, M_t, G_s, and G_t.

1. Compute a graph G_s, which is a dual of the mesh M_s. (A face in M_s becomes a vertex in G_s, and vice versa.)
2. Establish a one-to-one mapping between vertices in G_s and a subset of vertices in M_t. Define a graph G_t having as its vertices the vertices of M_t in the above subset.
3. Add edges to G_t so that G_t is isomorphic to G_s.
4. Establish a correspondence between an edge in G_t and a *path* in M_t (that is, an edge in G_t), so that G_t is isomorphic to G_s. Here a path refers to a series of connected edges. Note that either one of the two paths does not share any vertex in M_t except at the ends.
5. Create a tetrahedron by applying either one of the three rules below (See Figure 5).
   (a) Generate a tetrahedron from a vertex in M_s and a face in M_t. (Figure 5(a))
   (b) Generate a tetrahedron from an edge in M_s and an edge in M_t. (Figure 5(b))
   (c) Generate a tetrahedron from a face in M_s and a vertex in M_t. (Figure 5(c))

In Step 2 above, if the number of vertices in G_s (that is, the number of faces in M_s) is larger than the one in M_t, we pick two adjacent faces in M_s and group them into one so that the number of vertices in G_s is identical to (or less than) M_t. The pair of faces to be grouped together is picked by their geometrical proximity, e.g., closeness of their centroids.

The same procedure can also be applied when we fail in Step 4 above. If we cannot find the correspondences between an edge in G_t and a path of M_t, we reduce the number of vertices in G_s by one and go back to the Step 1 again.

This algorithm always succeeds in establishing tetrahedral meshes. For example, if both of the two source meshes are topologically identical to spheres, the algorithm will terminate as the number of vertices in G_s (and hence G_t) goes down to 4. This is because, in such a case, the graph having only four vertices becomes a tetrahedron, and we can definitely embed a tetrahedron in a mesh topologically equivalent to a sphere. Actually, the algorithm finds tetrahedral meshes by merging at most a few faces. The same argument holds when the source meshes are of arbitrary topological type.

When wavelet-analyzing a pair of TS-meshes to create a pair of simplified TS-meshes, we reduce their vertex counts down to about 10 to 20. Meshes of this size are simple enough for the base tetrahedral I-mesh creation algorithm described above to terminate quickly on all the examples we tried.

2.3 Variational optimization

The variational optimization step of our algorithm employs a minor variation of Takahashi's subdivision surfaces with multiresolution constraints [29]. Compared to Gortler's [24] and Zorin's [25] method, Takahashi's subdivision surface is different, in that it allows an independent sets of geometric constraints to be attached at each of the multiple resolution levels. This feature allows us to specify a complex shape with a smaller number of constraints than are necessary with the other methods.

The variational optimization algorithms for the 3D triangular I-mesh and 4D tetrahedral I-mesh are almost identical. The most significant differences are the dimension of coordinates and the vertex topology of the meshes to be optimized. The following description employs terminology for the case of a 3D triangular I-mesh interpolating 2D polygonal contours. However, by replacing terms "mesh", "surface" and "polyhedron" with "tetrahedral mesh", "tetrahedral volume", and "4D polyhedron", respectively, the description can be understood as the variational optimization algorithm for a 4D tetrahedral subdivision surface.

2.3.1 Subdivision Surface

A subdivision surface is obtained by recursively refining the original control polyhedron $\Theta^{(0)}$ so that the sequence of refined polyhedra $\Theta^{(1)}, \Theta^{(2)}, \ldots$ converges to the limit surface $\Theta^{(\infty)}$. Lounsbery et al. [28] treated refinement levels of subdivision surfaces as the resolution levels in a wavelet-based framework of multiresolution analysis. In this context, the superscript (k) denotes the resolution level in the multiresolution analysis.

Let the number of vertices in the subdivision surface $\Theta^{(k)}$ be $m^{(k)}$. By using the vectors of basis functions $\varphi_\alpha^{(k)}(\alpha = 1, 2, \ldots, m^{(k)})$ and coefficients $c_\alpha^{(k)}(\alpha = 1, 2, \ldots, m^{(k)})$ corresponding to $\varphi_\alpha^{(k)}$, the shape of the subdivision surface $\Theta^{(k)}$ can be written as a vector inner product $C^{(k)} = \boldsymbol{\varphi}^{(k)^\top} \cdot \boldsymbol{c}^{(k)} = (\varphi_1^{(k)}, \ldots, \varphi_{m^{(k)}}^{(k)})^\top \cdot (c_1^{(k)}, \ldots, c_{m^k}^{(k)})$, where $C^{(k)}(\boldsymbol{x})(\in V^{(k)})$. In fact $\Theta^{(k)}$ belongs to a function space $V^{(k)}$ of dimension $m^{(k)}$. Here, the i-th element of $\boldsymbol{c}^{(k)}$ represents some coordinate (either x, y, or z, if it is of 3D) of the i-th vertex of $\Theta^{(k)}$. This set of basis functions $\varphi_1^{(k)}, \ldots, \varphi_{m^{(k)}}^{(k)}$ is called *scaling functions* at resolution level k.

Consider now a function space $W^{(k)}$ defined as a difference of two function spaces spanned by the sets of bases $\boldsymbol{\varphi}^{(k+1)}$ and $\boldsymbol{\varphi}^{(k)}$. The space $W^{(k)}$ is spanned by the vector of basis functions $\psi_\alpha^{(k)}$ and has the dimension $n^{(k)} = m^{(k+1)} - m^{(k)}$. By using a vector of coefficients corresponding to $\psi_\alpha^{(k)}(\boldsymbol{x})(\alpha = 1, 2, \ldots, m^{(k)})$, the difference space can be written as $D^{(k)}(\boldsymbol{x}) = \boldsymbol{\psi}^{(k)^\top} \cdot \boldsymbol{d}^{(k)}$, where $D^{(k)} \in W^{(k)}$. This set of basis functions $\psi_1^{(k)}, \ldots, \psi_{n^{(k)}}^{(k)}$ is called *wavelets* at the resolution level k.

In the wavelet analysis framework, the difference $D^{(k)}(\boldsymbol{x})$ adds "details" to $C^{(k)}(\boldsymbol{x})$ to synthesize (or, reconstruct) a detail mesh $\Theta^{(k+1)}$. Wavelet synthesis can be written as

$$\boldsymbol{c}^{(k+1)} = \boldsymbol{P}^{(k)}\boldsymbol{c}^{(k)} + \boldsymbol{Q}^{(k)}\boldsymbol{d}^{(k)} \tag{1}$$

by using matrices $\boldsymbol{P}^{(k)}$ and $\boldsymbol{Q}^{(k)}$. Here, the pair of matrices $\boldsymbol{P}^{(k)}$ and $\boldsymbol{Q}^{(k)}$ is defined as

$$\boldsymbol{\varphi}^{(k)}(\boldsymbol{x})^\top = \boldsymbol{\varphi}^{(k+1)}(\boldsymbol{x})^\top \boldsymbol{P}^{(k)}, \quad \text{and} \tag{2}$$

$$\boldsymbol{\psi}^{(k)}(\boldsymbol{x})^\top = \boldsymbol{\psi}^{(k+1)}(\boldsymbol{x})^\top \boldsymbol{Q}^{(k)}. \tag{3}$$

These matrices are called *synthesis filters.*

In a typical subdivision surface framework, a coarser mesh $\Theta^{(k)}$ is topologically and geometrically refined to generate a finer mesh $\Theta^{(k+1)}$ without using the detail coefficients $\boldsymbol{d}^{(k)}$. The coordinate of each vertex after the topological refinement is computed by using a weighted sum of vertices around the vertex in question. Thus, the formula (1) above is simplified to

$$\boldsymbol{c}^{(k+1)} = \boldsymbol{P}^{(k)}\boldsymbol{c}^{(k)}. \tag{4}$$

Here the synthesis filter $\boldsymbol{P}^{(k)}$ is an $m^{(k+1)} \times m^{(k)}$, and it determines the coordinates of the refined mesh $\Theta^{(k+1)}$.

To construct a subdivision surface in 3D, we employ a triangular subdivision rule topologically identical to Loop's [22] (Figure 6). Our scheme differs from Loop's method in that we compute $\Theta^{(k+1)}$ from $\Theta^{(k)}$, plus a set of constraints at resolution level $(k+1)$ by using the local filtering method of Taubin et al. [30]. The vector of detail coefficients $\boldsymbol{d}^{(k)}$ is then obtained to satisfy formula (4) given the meshes $\Theta^{(k+1)}$ and $\Theta^{(k)}$. To construct a subdivision "surface" in 4D, that is, a "surface" consisting of tetrahedra, we employ a symmetric subdivision rule for a tetrahedron described by Moore [27] (Figure 7).

2.3.2 Geometric Constraints

To find subdivision surfaces that interpolate triangular source meshes (TS-meshes), the algorithm treats vertices of the shapes as geometric constraints to be interpolated. As explained, the interpolation algorithm first creates a coarse initial interpolator mesh (I-mesh) by connecting vertices of TS-meshes. Then, the initial interpolator mesh (I-mesh) is subdivided and variationally optimized to produce a smooth I-mesh.

We employ both *finite-dimensional* constraints and *transfinite-dimensional*constraints, as defined by Welch et al. [31], for the interpolation. The former refers to constraints defined on a discrete parametric domain, such as points, and the latter refers to constraints defined on a continuous parametric domain, such as lines and curves.

In the shape-blending algorithm, a finite-dimensional constraint is added to each vertex of S-mesh in order to make I-mesh interpolate the vertex. Finite-dimensional constraints can also be used to create various transitional effects, e.g., by pulling/pushing the surface by the constraints. Our algorithm [29] allows different sets of constraints to be added at each of the multiple resolution levels. This gives us control over the shape of interpolating subdivision surfaces that is more powerful than are possible with either Gortler's [24] or Zorin's [25] methods.

A finite-dimensional constraint that makes a surface $\Theta^{(k)}$ interpolate a point constraint θ_0 at a parameter value $\boldsymbol{x}_0$ can be written as

$$\Theta^{(k)}(\boldsymbol{x}_0) = \boldsymbol{\varphi}^{(k)}(\boldsymbol{x}_0)^\top \cdot \boldsymbol{c}^{(k)} = \theta_0 \tag{5}$$

To specify vertex-to-vertex correspondence, the algorithm connects a pair of vertices, one each from each of the adjacent (in blending axis) TS-meshes, using a transfinite-dimensional constraint such as a straight or a curved line. By specifying a path in 4D space the correspondence curve should follow, we could incorporate various shape transition effects.

The algorithm minimizes the least-mean-square error of an integral to approximately satisfy the transfinite-dimensional constraints [31]. Consider a coordinate of a transfinite-dimensional constraint $L(t) = L(\boldsymbol{l}(t))$ imposed on $\Theta^{(k)}(\boldsymbol{x})$, where $\Theta^{(k)}(\boldsymbol{x})$ denotes a coordinate of the subdivision surface $\Theta^{(k)}$ of the parameter $\boldsymbol{x}$, and $\boldsymbol{x} = \boldsymbol{l}(t)$ is the corresponding parametric path. To parameterize the subdivision surface, we employed a barycentric coordinate system. (Please refer [22] for the barycentric coordinates.) The integral of the squared difference between the surface and the curve constraint is

$$\int \Big(\Theta^{(k)}(\boldsymbol{l}(t)) - L(t)\Big)^2 dt. \tag{6}$$

To minimize this integral, we solve the following equation;

$$\Big(\int_{\boldsymbol{l}} \boldsymbol{\varphi}^{(k)}(\boldsymbol{l}) \cdot \boldsymbol{\varphi}^{(k)}(\boldsymbol{l})^\top\Big)\boldsymbol{c}^{(k)} = \int_{\boldsymbol{l}} L \cdot \boldsymbol{\varphi}^{(k)}(\boldsymbol{l}) dt. \tag{7}$$

Both finite- and transfinite-dimensional in the end result in a system of linear equations with respect to $\boldsymbol{c}^{(k)}$ as below;

$$\boldsymbol{M}^{(k)}\boldsymbol{c}^{(k)} = \boldsymbol{r}^{(k)}. \tag{8}$$

The number of linear constraints in this case is equivalent to the number of vertices contained in the faces where the transfinite-dimensional constraints traverse.

In our shape blending scheme, both finite- and transfinite-dimensional constraints can be imposed on the I-mesh at multiple resolution levels simultaneously. We call such constraints *multiresolution constraints*, which allow us to manipulate shape of an I-mesh effectively. Constraints added at a low resolution level would have global effects on I-mesh shape, while those added at a high resolution level would have localized effects. By imposing constraints simultaneously at multiple resolution levels, we could effectively manipulate shapes of I-meshes. Figure 12 and Figure 13 show examples of manipulated I-meshes and shape blending effects produced by the mesh by imposing constraints at different resolution levels. In the example, constraints controls blending of the letter “2” with the letter “3”. Please refer Section 2.4.3 for the explanation on the figure. Details of the algorithm for solving the multiresolution constraints via local smoothing, which was originally proposed in [29], will be presented in Appendix A.

2.4 *Controlling Shape Interpolation*

This section describes mechanisms available in our algorithm to impose various controls over shape blending.

2.4.1 *Feature Correspondence*

Feature correspondences are established by imposing transfinite-dimensional constraints on I-meshes. Figure 8 shows an example of feature correspondence. Figure 8(a) shows I-mesh without feature correspondence. Blending sequence resulted from this I-mesh is shown in Figure 8(d). In Figure 8(b), a pair of vertex-to-vertex feature correspondences are established by imposing a pair of straight-line transfinite-dimensional constraints on the I-mesh. The constraints connect two points of the star with a point of the triangle. (While not visible, there is another straight-line constraint connecting the top vertex of the star with a middle point of the upper edge of the triangle.) Solving for the constraints produced an I-mesh shown in Figure 8(c) and the blending sequence shown in Figure 8(e).

An example of feature correspondence in 3D shape blending is shown in Figure 9. A tip of the horn of the star shape is related by a curved line transfinite-dimensional constraint (shown as a thick black line) to the tip of the nose of the mannequin head model. Figure 9(a) shows the geodesic (shown as a gray line with dots) and the curved line constraint without the tetrahedral I-mesh, while Figure 9(b) includes the I-mesh. The constraint solver tries to match the geodesic on the surface of the I-mesh (shown as a gray line with dots) with the curved line constraint, producing a deformed I-mesh shown in Figure 9(c). Note that the geodesic, consisting of a sequence of edges of the I-mesh, is an approximation of a real geodesic. The geodesic on the I-mesh is created automatically by selecting a pair of vertices to be related, for manually specifying a sequence of edges on a 4D I-mesh not practical.

The blending sequence of Figure 10(a) is created without the feature correspondence, while that of Figure 10(b) is generated by using the feature correspondence based on the line constraint described above. The effect of feature correspondence between the horn of a star and the nose of the mannequin, with some exaggeration, is noticeable in Figure 10(b).

2.4.2 *Source Sharpness Control*

Certain shape interpolation requires that the sharp features of the source shapes to be preserved exactly, while shape transition be smooth and continuous. We realize such an effect by using special subdivision rules and transfinite-dimensional constraints. To control the sharpness of sources shapes, we attach a flag to each vertex indicating either the vertex is to remain sharp or be smoothed. Our algorithm applies different subdivision rules at the vertex depending on the flag.

Figure 11 demonstrates this source sharpness control feature of our algorithm. Source contours shown in Figure 11(a) are used to create an initial I-mesh in Figure 11(b). In the source meshes, vertices marked with circles are to remain sharp. The initial I-mesh is subdivided and smoothed to produce the I-mesh shown in Figure 11(c) and Figure 11(d). While the image in Figure 11(c) employed hidden-surface mesh rendering, that of Figure 11(d) employed Gouraud-shaded semi-opaque surface rendering. Note that, among the S-meshes, vertices of the star shape were not rounded while a subset of the vertices of the fan-like shape are smoothed, depending on the smoothing control flags attached. This kind of sharp features in a source shape are not easy to achieve if smooth an implicit-function is used to represent the source shapes.

2.4.3 Enhanced shape transition

Deformation of the I-mesh surface by using multiresolution constraints enables us to create interesting "enhanced" transitional shapes in shape-blending sequences. Both point and line geometric constraints can be used for such enhancements of shape transitions. For example, adding line constraints, the same approach as the one used for feature correspondence described in Section 2.4.1, can creates enhanced transitions between features of source shapes. Since constraints are of multiresolution nature, the locality of influence of a constraint can be selected by picking a mesh resolution level at which the constraint is applied.

Images in Figure 12 and Figure 13 compares, in 2D shape blending, exaggerated shape transitions effects generated by applying constraints at different resolution levels. Figures 12(a), (b), and (c) are smoothed I-meshes at the resolution levels 1, 2, and 3, respectively. Notice dots in each of the figure, which are the point geometric constraints added to deform the I-mesh for the exaggerated shape transition effects. Figures 12(a), (b), and (c) show constraints with their respective I-meshes at the resolution level which the constraints are applied. After solving for the constraints, I-meshes of Figures 12(d), (e), and (f) have resulted, which correspond, respectively, to Figures 12(a), (b), and (c). Deformations are global (Figure 12(d)) if the constraints are added at a low resolution level (Figure 12(a)). On the other hand, deformations are local (Figure 12(f)) if the constraints are added at a high resolution level (Figure 12(c)).

Figures 13(a)-(d) compares the blending sequence produced by the enhanced meshes. Figure 13(a) is the base case without any enhancement. In the "enhanced" transition sequences of Figures 13(b)-(d), the middle horizontal "bar" of the letter "3" protrudes excessively during the transition. Applying constraints at the I-mesh resolution levels of 1, 2, and 3 produced, respectively, different versions of the enhanced shape transition sequences shown in Figures 13(b), (c), and (d).

Enhanced shape transition can also be created for 3D shape transitions by using constraints. The I-mesh manipulation shown in Figure 9 created the enhanced blending sequence of Figure 10(b). The blending sequence without enhancement is shown in Figure 10(a) for comparison. This effect is realized by specifying an arced line as a geometric constraint for feature correspondence.

2.4.4 Spatially non-uniform shape transition

A spatially non-uniform progress of the blending creates an interesting effect. For example, in Figure 14(d), by effectively increasing the rate of shape transition at the top, the letter "3" appears to blend into the letter "1" from top to bottom.

We achieve such non-uniform transition effect by warping an I-mesh by using spline functions before the I-mesh is intersected with another surface to extract a blended shape. We first compute a tight bounding box of the I-mesh and normalize its coordinates. We then employ a warp function to transform the normalized coordinate values of the surfaces. In the case of 2D shape blending, we first specify a cubic tensor-product Bézier patch

$$P(x, y, t) = \sum_{i,j=0}^{3} B_i(x) B_j(y) P_{ij}(t), \tag{9}$$

where $B_i(x)(i = 0, \ldots, 3)$ represents a Bernstein polynomial and t the blending axis. Each value of the sixteen control points $P_{ij}(t)(i, j = 0, \ldots, 3)$ for (9) is a function of t, which is calculated by specifying four values of $P_{ij}(t)$ at $0, \frac{1}{3}, \frac{2}{3}, 1$, and interpolating them with cardinal splines. Thus, the resulting function (9) is smooth and end-point interpolating. The function transforms the normalized coordinates of the vertices of the I-mesh into a warped surface, which is the I-mesh with spatially non-uniform progress of blending.

In the case of 3D shape blending, its I-mesh warping function is similar to that of 2D shape blending explained above except for its dimension. Intersecting the warped I-mesh surfaces with a sequence of planes perpendicular to the axis t at $t =$ const. produces a blended shape sequence with spatially non-uniform progress of blending.

Figure 14 shows, for 2D shape blending, an example of spatially non-uniform progress of shape blending. Figure 14(a) and Figure 14(b) show the I-meshes before and after the warp. Resulting spatially non-uniform blending sequence is shown in Figure 14(c), in which the letter "3" morphs into the letter "1" from top to bottom.

3 Results

This section presents some of the 3D shape blending results created by using our shape-blending method.

Figure 15(a) shows an example of blending the mannequin head with the tiger head. This example tries to preserve sharpness in the source models. Both of the source shapes shown in the figure are already reparameterized so that they have 1-to-4 subdivision connectivity. Source and blended meshes in this figure are at resolution level 4, that is, the base source mesh is refined 4 times by using the 1-to-4 subdivision rule. For example, the mannequin head at resolution level 0 consisted of 22 triangles, and its level 4 mesh consisted of 5632 triangles. Using our prototype implementation, computing a blended shape in this example took about 5 minutes on a PC (Intel Pentium III CPU running at 700 MHz, with 256 MByte memory). This time includes all the necessary steps for the shape blending, including reparameterization, initial meshing, variational optimization, and shape extraction. Figure 15(b) is another example that blended an octahedron with sharp edges and points with the mannequin head model.

An example of shape blending with feature correspondence is shown in Figure 16. In this example, the eyes and nose of the source meshes are related by vertex-to-vertex curve geometric constraints. Figure 16(a) and (b) show the source shapes with constraints relating ears and noses. In Figure 16(c), the I-mesh was deformed to satisfy given geometric constraints. Figure 16(d) shows the shape blending sequence resulted from the feature correspondence specified in Figure 16.

The blending sequence shown in Figure 17(a) is another example of feature correspondence, in which only one of the pair ears of the mannequin is related to its counterpart in the tiger head. Figure 17(b) is an example of spatially non-uniform shape transition in which the blending progresses from top to bottom. The mannequin head appears to turn into the tiger head from top down.

4 Conclusion

This paper presented a new geometric morphing algorithm for shapes defined using polyhedrons. The algorithm directly interpolates the polyhedrons (the "source" shapes) by using a subdivision surface having a dimension one higher than the source shapes. The vertices of the source shapes are treated as geometric constraints to be satisfied using a variational optimization method, producing a smooth interpolating surface.

The algorithm combines some of the advantages of the methods based on variational optimization of volume implicit function, e.g., that of Turk and O'Brien [19], and the methods based on a common mesh embedding of the source polyhedrons, e.g., that of Kanai, et al. [3]. As in the case of the former class of methods, our method produces smooth transition due to variational optimization. At the same time, our method allows for feature correspondence through geometric constraints, e.g., a vertex in a source shape can be coerced to become a vertex in another source shape through the morphing.

Our method also allows for various shape transition effects thanks mainly to manipulable nature of the subdivision surface. For example, deformations of the interpolating surface could produce spatially non-uniform shape transitions as well as exaggerated blended shape transitions.

The method we have presented still has much room for improvements. The foremost on the list of improvements is the issue of overly smooth blended shapes. As evidenced in the examples shown in Figure 17, the algorithm creates blended shapes that are too smooth. Detailed facial features of the tiger or the mannequin head have been smoothed out in the blended shapes, creating less appealing blending sequences. We intend to solve this problem by incorporating multiresolution framework to shape blending, an approach similar to Lee et al. [6]. In our proposed solution, we add back detailed features in the source TS-meshes on top of the smooth blended shapes by taking advantage of our multiresolution framework.

Almost equally important is the obvious extension of the 3D shape blending for source shapes having different surface topology. We must find a method to achieve topology transcending shape blending as well as an effective user interface to specify topological evolution.

Other issues of concern include performance improvement and better reparameterization algorithm. Our current implementation of the shape-blending algorithm is not particularly efficient. As mentioned in Section 3, current implementation took about 5 minutes to create a blended shape for a blended mesh having 25,000 triangles. We intend to improve both time- and space-complexities of our algorithm. We also would like find a mesh reparameterization algorithm that is better suited to our purpose.

Acknowledgments

The mesh data of the mannequin head is obtained from Stanford University, and that of the tiger head is obtained from the Avalon archive at Viewpoint Datalabs. We would like to thank Takashi Kanai for discussions. We also would like to thank Hiroshi Masuda, Kazunori Miyata and other members of the Visual Artifacts Study Group for comments. The first author is supported in part by grants from the Ministry of Education, Science, Sports and Culture (MESSC) of Japan (Grant No. 12680432) and from the Hayao Nakayama Foundation for Science & Technology and Culture. The third author is supported in part by grants from the MESSC of Japan (Grant No. 12780185) and from the Kayamori Foundation of Information Science Advancement.

Appendix A Solving Multiresolution Constraints

The multiresolution constraints are solved by using local smoothing filter as described by Taubin, et al. [30], which efficiently approximates global variational optimization originally used in Welch and Witkin's scheme [31]. By using Taubin's method, the variational optimization can be approximated with a computational cost linearly proportional to the number of vertices in the surface.

We apply smoothing one resolution level at a time. At each resolution level, we first add shape details as specified by constraints attached at the level, followed by local smoothing. This is repeated from the lowest resolution level 0 to the highest resolution level K.

Let $\widehat{\boldsymbol{c}}^{(k)} (k = 0, \ldots, K)$ be the detail added at the resolution level k. Then the vertex coordinate of the subdivision surface at level K, which is denoted as $\boldsymbol{c}^{(K)}$, can be written as below;

$$\boldsymbol{c}^{(K)} = \boldsymbol{P}^{(K-1)} \cdots \boldsymbol{P}^{(0)} \widehat{\boldsymbol{c}}^{(0)} + \boldsymbol{P}^{(K-1)} \cdots \boldsymbol{P}^{(1)} \widehat{\boldsymbol{c}}^{(1)} + \cdots + \widehat{\boldsymbol{c}}^{(K)} \quad \text{(A.1)}$$

That is, a shape at level K is defined as a combination of details defined at all resolution levels. We picked an exponential weighting function $\beta^{(k)} (k = 0, \ldots, K)$ to weight the contributions from multiple resolution levels.

$$\beta^{(k)} = \gamma^{k-K} \quad \text{(A.2)}$$

A normalized form below is used to weight resolution level k

$$\frac{\beta^{(k)}}{\beta^{\Sigma}} \quad \text{(A.3)}$$

where

$$\beta^{\Sigma} = \sum_{\lambda=0}^{K} \beta^{\lambda}. \quad \text{(A.4)}$$

The steps needed to determine the shape of a surface are as follows, assuming that we have a set of linear equations for given constraints as defined in (8).

Resolution level 0: Consider a "formal" detail coefficient vector $\widehat{\boldsymbol{c}}_f^0$ that satisfies a set of constraints at the level 0;

$$\boldsymbol{M}^{(0)}\widehat{\boldsymbol{c}}_f^{(0)} = \boldsymbol{r}^{(0)}. \tag{A.5}$$

Then, the actual detail $\widehat{\boldsymbol{c}}^{(0)}$ of the shape is computed as

$$\widehat{\boldsymbol{c}}^{(0)} = \frac{\beta^{(k)}}{\beta^{\Sigma}}\widehat{\boldsymbol{c}}_f^{(0)} \tag{A.6}$$

by using the normalized weight described above. From (A.5) and (A.6), the equation on $\widehat{\boldsymbol{c}}^{(0)}$ to be solved becomes;

$$\boldsymbol{M}^{(0)}\boldsymbol{c}^{(0)} = \frac{\beta^{(0)}}{\beta^{\Sigma}}\boldsymbol{r}^{(0)}. \tag{A.7}$$

We apply local smoothing to the surface so that this constraint equation (A.7) is satisfied. As we do, we simply use

$$\widehat{\boldsymbol{c}}^{(0)} = \boldsymbol{0} \tag{A.8}$$

as the initial value for the smoothing.

Resolution level k > 0: At resolution level $k > 0$, consider a vector $\widehat{\boldsymbol{c}}_f^{(k)}$ that satisfies the following equation;

$$\boldsymbol{M}^{(k)}\widehat{\boldsymbol{c}}_f^{(k)} = \boldsymbol{r}^{(k)}. \tag{A.9}$$

We want $\widehat{\boldsymbol{c}}^{(k)}$ to have only the contribution from the level k, so we subtract contributions from the levels 0 to $(k-1)$. The sum of contributions of details from the level 0 up to level $(k-1)$ is;

$$\sum_{\lambda=0}^{k-1} \boldsymbol{P}^{(k-1)} \cdots \boldsymbol{P}^{(\lambda)}\widehat{\boldsymbol{c}}^{(\lambda)}. \tag{A.10}$$

Thus, with weighting, $\widehat{\boldsymbol{c}}^{(k)}$ and $\widehat{\boldsymbol{c}}_f^{(k)}$ must satisfy the following equation;

$$\widehat{\boldsymbol{c}}^{(k)} = \frac{\beta^{(0)} + \cdots + \beta^{(k)}}{\beta^{\Sigma}} - \sum_{\lambda=0}^{k-1} \boldsymbol{P}^{(k-1)} \cdots \boldsymbol{P}^{(\lambda)}\boldsymbol{c}^{(\lambda)}. \tag{A.11}$$

From (A.10) and (A.11), we obtain the equation to solve for $\widehat{\boldsymbol{c}}^{(k)}$;

$$\boldsymbol{M}^{(k)}\widehat{\boldsymbol{c}}^{(k)} = \frac{\beta^{(0)} + \cdots + \beta^{(k)}}{\beta^{\Sigma}}\boldsymbol{r}^{(k)} - \boldsymbol{M}^{(k)}\Big(\sum_{\lambda=0}^{k-1} \boldsymbol{P}^{(k-1)} \cdots \boldsymbol{P}^{(\lambda)}\widehat{\boldsymbol{c}}^{(\lambda)}\Big). \tag{A.12}$$

As the initial value for the level k, we want the sum of detail contributions from the level 0 to $(k-1)$, i.e., the formula (A.10), but not that of k.

As each resolution level λ is weighted by $\beta^{(\lambda)}$, the initial value for the actual detail vector $\widehat{\boldsymbol{c}}^{(k)}$ in the level k is given by;

$$\frac{\beta^{(k)}}{\beta^{(0)} + \cdots + \beta^{(k-1)}} \sum_{\lambda=0}^{k-1} \boldsymbol{P}^{(k-1)} \cdots \boldsymbol{P}^{(\lambda)}\widehat{\boldsymbol{c}}^{(\lambda)}. \tag{A.13}$$

References

[1] Lazarus F, Verroust A, Three-dimensional metamorphosis: A survey, The Visual Computer 1998;14:373–389.

[2] Kent JR, Carlson WE, Parent RE, Shape transformation for polyhedral objects, in: Computer Graphics (Proceedings Siggraph '92), 1992. p. 47–54.

[3] Kanai T, Suzuki H, Kimura F, 3D geometric metamorphosis based on hamonic maps, The Visual Computer 1998;14(4):166–176.

[4] Gregory A, State A, Lin M, Monocha D, Livingston M, Feature-based surface decomposition for correspondence and morphing between polyhedra, in: Computer Animation '98, IEEE Computer Society Press, 1998. p. 64–71.

[5] Kanai T, Suzuki H, Kimura F, Metamorphosis of arbitrary triangular meshes, IEEE Computer Graphics and Applications 2000;20(2):62–75.

[6] Lee AWF, Dobkin D, Sweldens W, Schröder P, Multiresolution mesh morphing, in: Computer Graphics (Proceedings Siggraph '99), 1999, p. 343–350.

[7] Lee AWF, Sweldens W, Schröder P, Cowsar L, Dobkin D, MAPS: Multiresolution adaptive parametrization of surfaces, in: Computer Graphics (Proceedings Siggraph '98), 1998. p. 95–104.

[8] DeCarlo D, Gallier J, Topological evolution of surfaces, in: Graphics Interface '96, 1996. p. 194–203.

[9] Lerios A, Garfinkle CD, Levoy M, Feature-based volume metamorphosis, in: Computer Graphics (Proceedings Siggraph '95), 1995. p. 449–456.

[10] Beier T, Neely S, Feature-based image metamorphosis, in: Computer Graphics (Proceedings Siggraph '92), 1992. p. 35–42.

[11] Hughes JF, Scheduled fourier volume morphing, in: Computer Graphics (Proceedings Siggraph '92), 1992. p. 43–46.

[12] He T, Wang S, Kaufman A, Wavelet-based volume morphing, in: IEEE Visualization '94, 1994. p. 85–92.

[13] Levin D, Multidimensional reconstruction by set-valued approximation, IMA Journal of Numerical Analysis 1986;6:173–184.

[14] Raya SP, Udupa JK, Shape-based interpolation of multidimensional objects, IEEE Transactions on Medical Imaging 1990;9(1):32–42.

[15] Kaul A, Rossignac J, Solid-interpolating deformations: Construction andanimation of pips, in: Eurographics '91, 1991. p. 493–505.

[16] Payne B, Toga AW, Distance field manipulation of surface models, IEEE Computer Graphics and Applications 1992;12(1):65–71.

[17] Cohen-Or D, Levin D, Solomovici A, Three-dimensional distance field metamorphosis, ACM Transactions on Graphics 1998;17(2):116–141.

[18] Whitaker R, Breen D, Level-set models for the deformation of solid objects, in: Proc. of Third International Workshop on Implicit Surfaces, 1998. p. 19–35.

[19] Turk G, O'Brien JF, Shape transformation using variational implicit functions, in: Computer Graphics (Proceedings Siggraph '99), 1999. p. 335–342.

[20] Catmull E, Clark J, Recursively generated b-spline surfaces on arbitrary topological meshes, Computer-Aided Design 1978;10(6):350–355.

[21] Doo D, Sabin M, Behaviour of recursive division surfaces near extraordinary points, Computer-Aided Design 1978;10(6):356–360.

[22] Loop C, Smooth subdivision surfaces based on triangles, Master's thesis, University of Utah, Department of Mathematics (1987).

[23] Dyn N, Levin D, Gregory JA, A butterfly subdivision scheme for surface interpolation with tension control, ACM Transactions on Graphics 1990;9(2):160–169.

[24] Gortler SJ, Cohen MF, Hierarchical and variational geometric modeling with wavelets, ACM Symposium on Interative 3D Graphics 1995. p. 35–42.

[25] Zorin D, Schröder P, Sweldens W, Interactive multiresolution mesh editing, in: Computer Graphics (Proceedings Siggraph '97), 1997. p. 259–268.

[26] Shinagawa Y, Kunii TL, The homotopy model: A generalized model for smooth surface generation from cross sectional data, The Visual Computer 1991;7(2-3):72–86.

[27] Moore D, Graphics Geoms III, Academic Press, 1992, Ch. Subdividing Simplices, p. 244–249.

[28] Lounsbery M, DeRose TD, Warren J, Multiresolution analysis for surfaces of arbitrary topological type, ACM Transactions on Graphics 1997;16(1):34–73.

[29] Takahashi S, Multiresolution constraints for designing subdivision surfaces via local smoothing, in: Computer Graphics and Applications (Proc. of Pacific Graphics '99), IEEE Computer Society Press, 1999. p. 168–178.

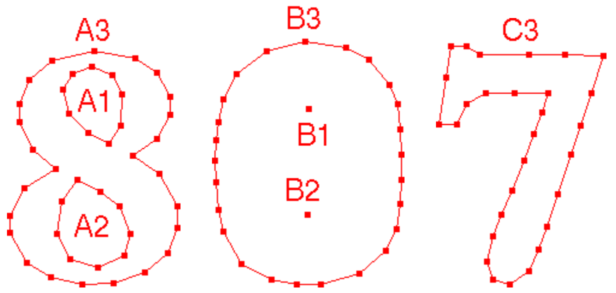
[30] Taubin G, A signal processing approach to fair surface design, in: Computer Graphics (Proceedings Siggraph '95), 1995. p. 351–358.

[31] Welch W, Witkin A, Variational surface modeling, in: Computer Graphics (Proceedings Siggraph '92), 1992, p. 157–166.

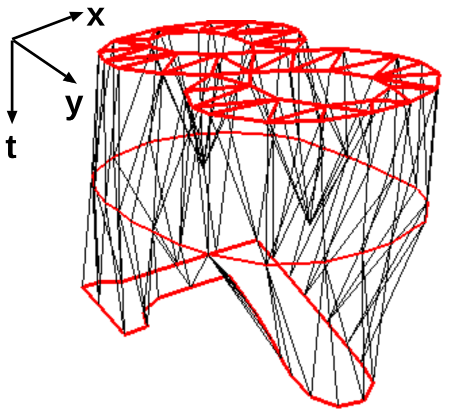
Ryutarou Ohbuchi is currently an associate professor of the Computer Science Department at Yamanashi University. He received his B. Sci. in Electrical and Electronic Engineering from the Sophia University, Tokyo, Japan, in 1981, his M. Sci in Computer Science from the University of Electro-Communications, Tokyo, Japan, in 1983, and his Ph. D. in Computer Science from the University of North Carolina at Chapel Hill, in 1994. He worked as a research staff at IBM Tokyo Research Laboratory from 1994 to 1999. His current research interest include computer graphics, human-computer interaction, and geometric modeling. He is a member of the ACM, IEEE, IPSJ (Information Society of Japan), and JSSST (Japan Society for Software Science and Technology).

Yoshiyuki Kokojima is currently a master course graduate student of the Department of Computer Science at Gunma University. He received his B.Sc., degree in computer science from Gunma University in 1999. His research interests include computer graphics and computer vision.

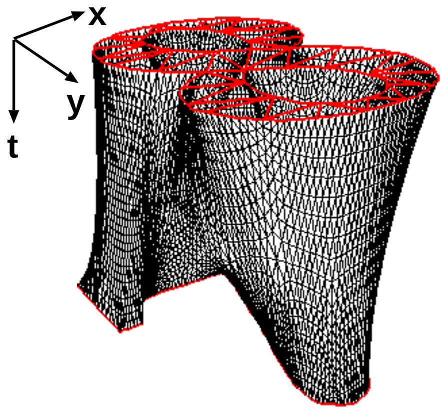
Shigeo Takahashi is currently an associate professor of the Computer Center at Gunma University. He received his B.Sc., M.Sc., and Ph.D degrees in computer science from the University of Tokyo in 1992, 1994, and 1997, respectively. His research interests include computer graphics and geometric modeling. He is a member of the ACM, IEEE Computer Society, IPSJ, and IEICE(The Institute of Electronics, Information and Communication Engineers).



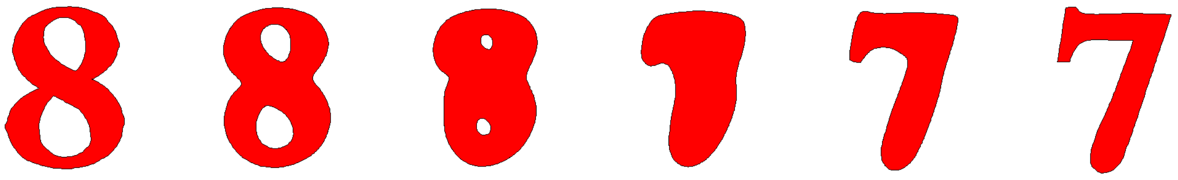
(a) Initial source contours (both sides) and a topological key-shape (in the middle).



(b) Initial triangular I-mesh.



(c) Subdivided and smoothed I-mesh.



(d) Shape blending sequence generated.

Fig. 1. Blending 2D contours of the letters ”8” and ”7”. Source contours in (a) are connected to create the initial I-mesh (b), which is subdivided and smoothed (c). The middle contour in (a), the topological key-shape, is inserted to guide the topological evolution of the I-mesh from ”8” to ”7”.

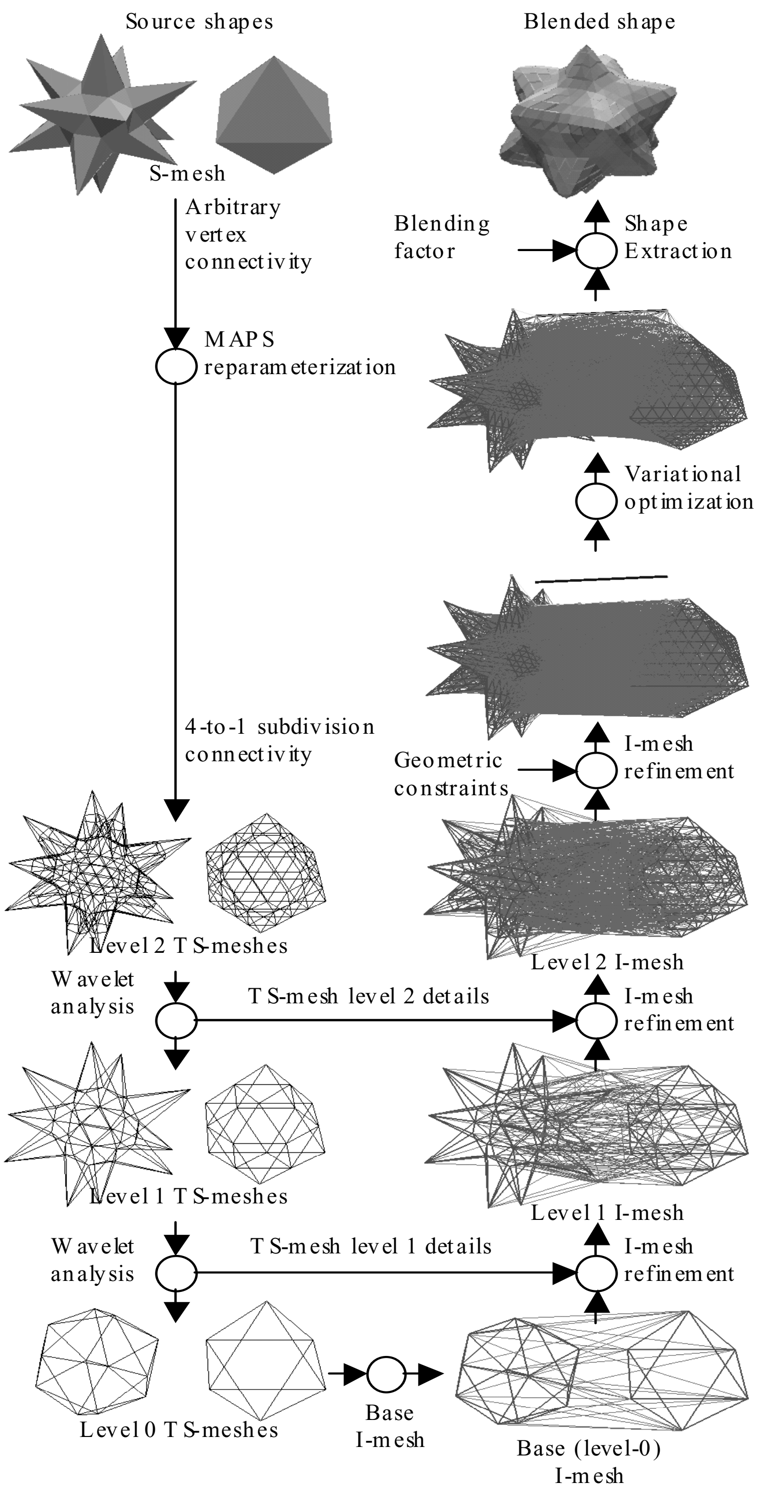
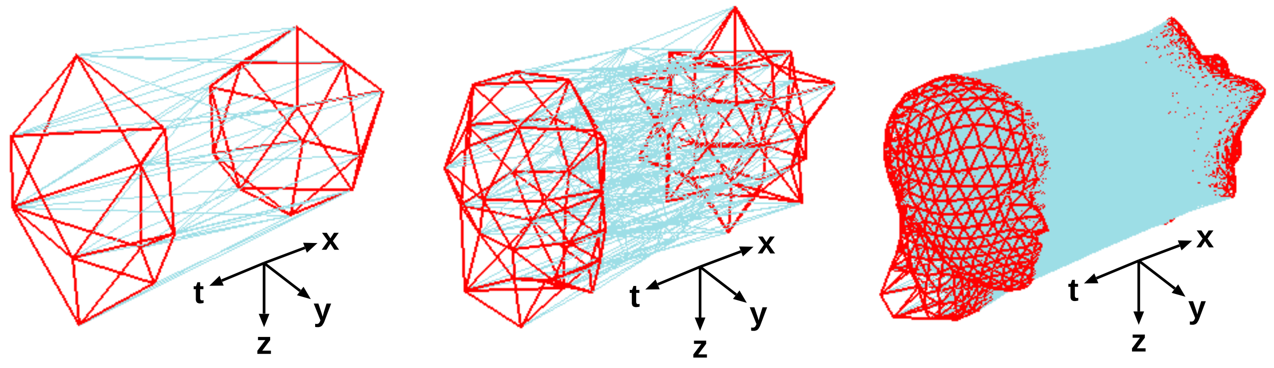
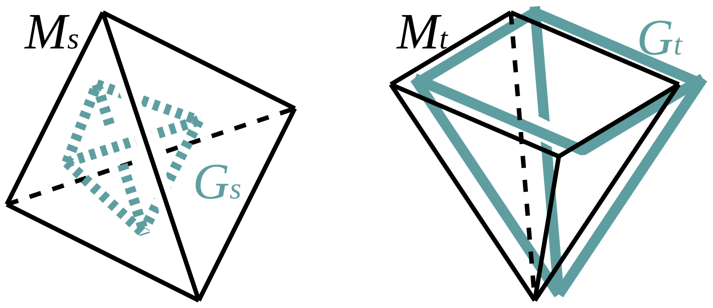


Fig. 2. The process flow of the 3D shape-blending algorithm by using subdivision surfaces. The figure is illustrated for the case of mesh refinement level of 2.



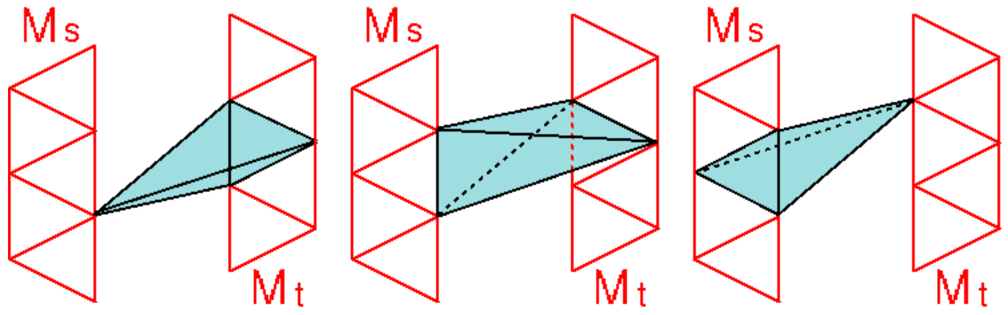
(a) Level-0 I-mesh (b) Level-1 I-mesh (c) Level-3 I-mesh

Fig. 3. The base I-mesh (a) created from the MR TS-meshes of the rounded star and the mannequin head models. The TS-meshes and the I-mesh are refined together by one level (b) and three levels (c).



$$\begin{tikzcd}
 & \text{source} & \text{target} \\
\text{meshes} & M_s \arrow[d, "\text{dual}"] & M_t \arrow[d, "\text{subgraph}"] \\
\text{graphs} & G_s \arrow[r, leftrightarrow, "\text{isomorphic}"'] & G_t
\end{tikzcd}$$

Fig. 4. Relationships between M_s, M_t, G_s, and G_t. A graph G_s is defined as the dual of a source mesh M_s, and a graph G_t is defined as a subgraph of a target mesh M_t so that G_t is isomorphic to G_s.



(a) Vertex-Face (b) Edge-Edge (c) Face-Vertex

Fig. 5. Three cases in which a tetrahedron is created between a pair of graphs.

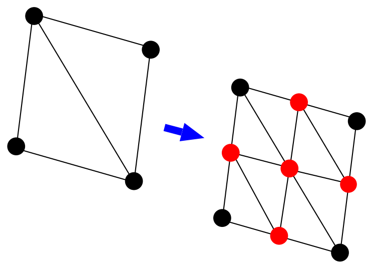


Fig. 6. The 1-to-4 subdivision rules for triangular meshes.

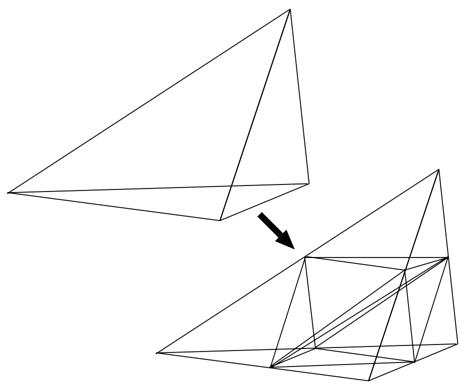
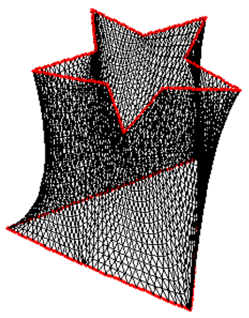
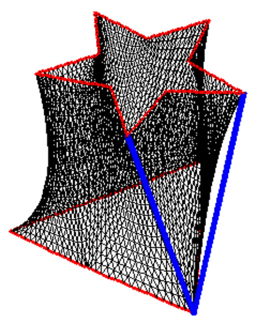
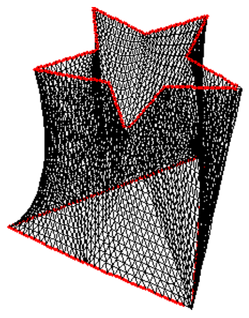
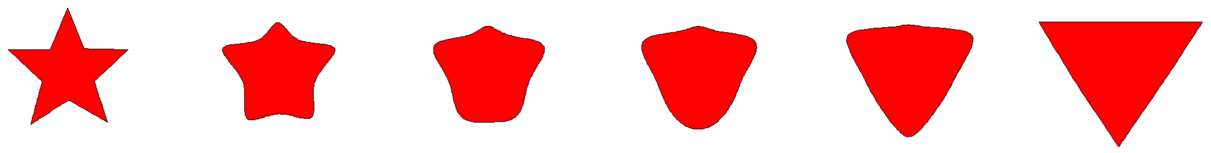


Fig. 7. The 1-to-8 “symmetrical” subdivision rules for tetrahedral meshes.


(a) I-mesh without feature correspondence.


(b) I-mesh with line constraints.


(c) I-mesh after solving for the constraints.


(d) The blending sequence without feature correspondence.

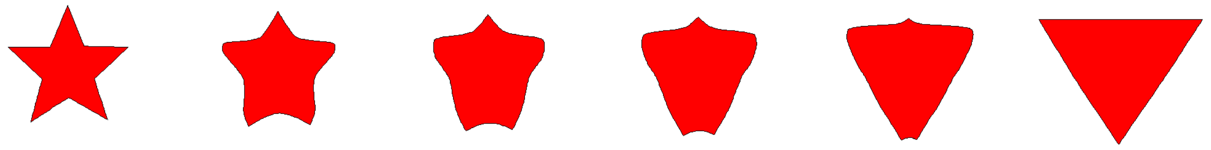
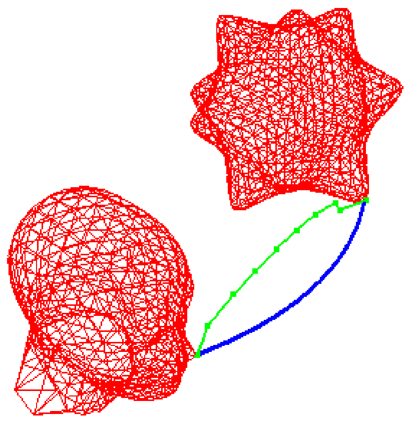
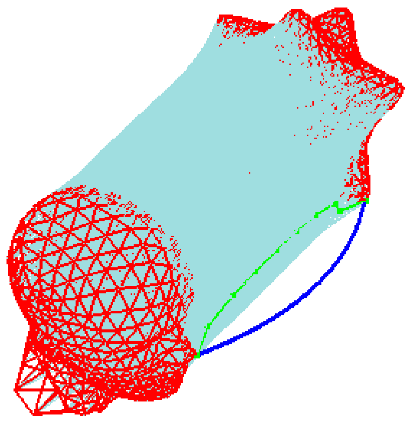

(e) The blending sequence with feature correspondence.

Fig. 8. Line constraints can be used to establish feature correspondence required for effective shape blending.


(a) The geodesic and the constraint shown without the I-mesh.


(b) The geodesic and the constraint shown with the I-mesh.

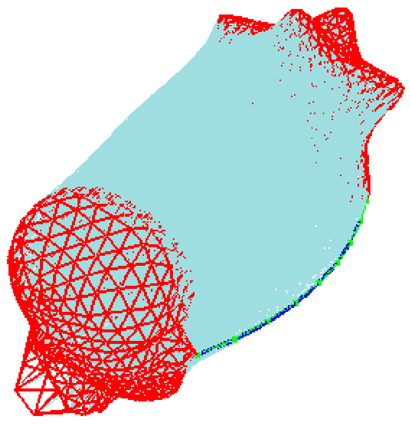
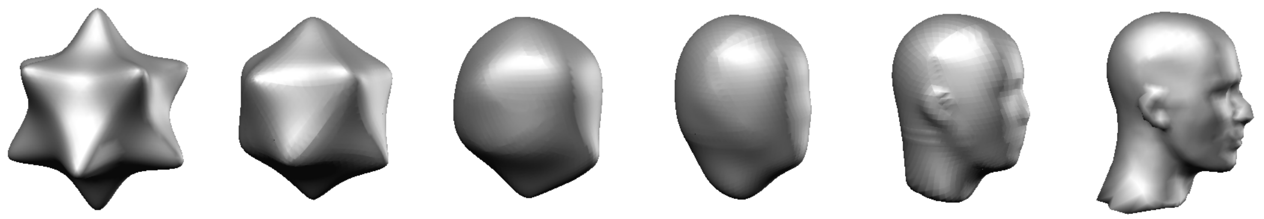
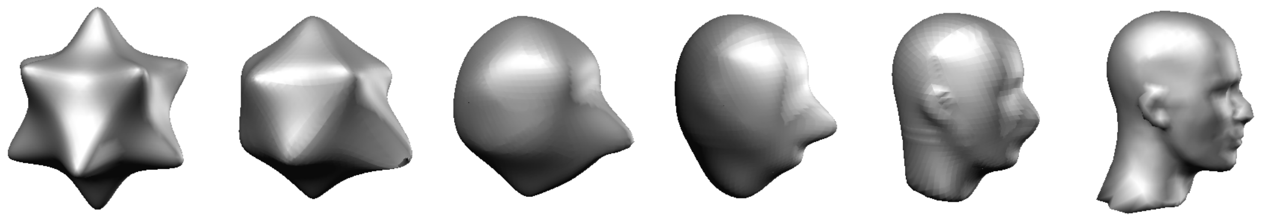

(c) After solving for the constraint.

Fig. 9. A feature correspondence is created as a line geometric constraint, which is shown as a thick black line in (a) (without I-mesh) and (b) (with I-mesh). Geodesic on the I-mesh (shown as a gray line with dots in (a) and (b)) connecting the feature vertices are pulled out along with the I-mesh nearby to satisfy the constraint in (c).



(a) Blending shapes without constraint.



(b) Blending shapes with correspondence that related the tip of the nose with the tips of the rounded star.

Fig. 10. Shape blending with (a) and without (b) feature correspondence, generated from the I-mesh shown in Figure 9. In (b), a tip of a point of the star shape and the nose of the mannequin's head are related by using a geometric constraint.

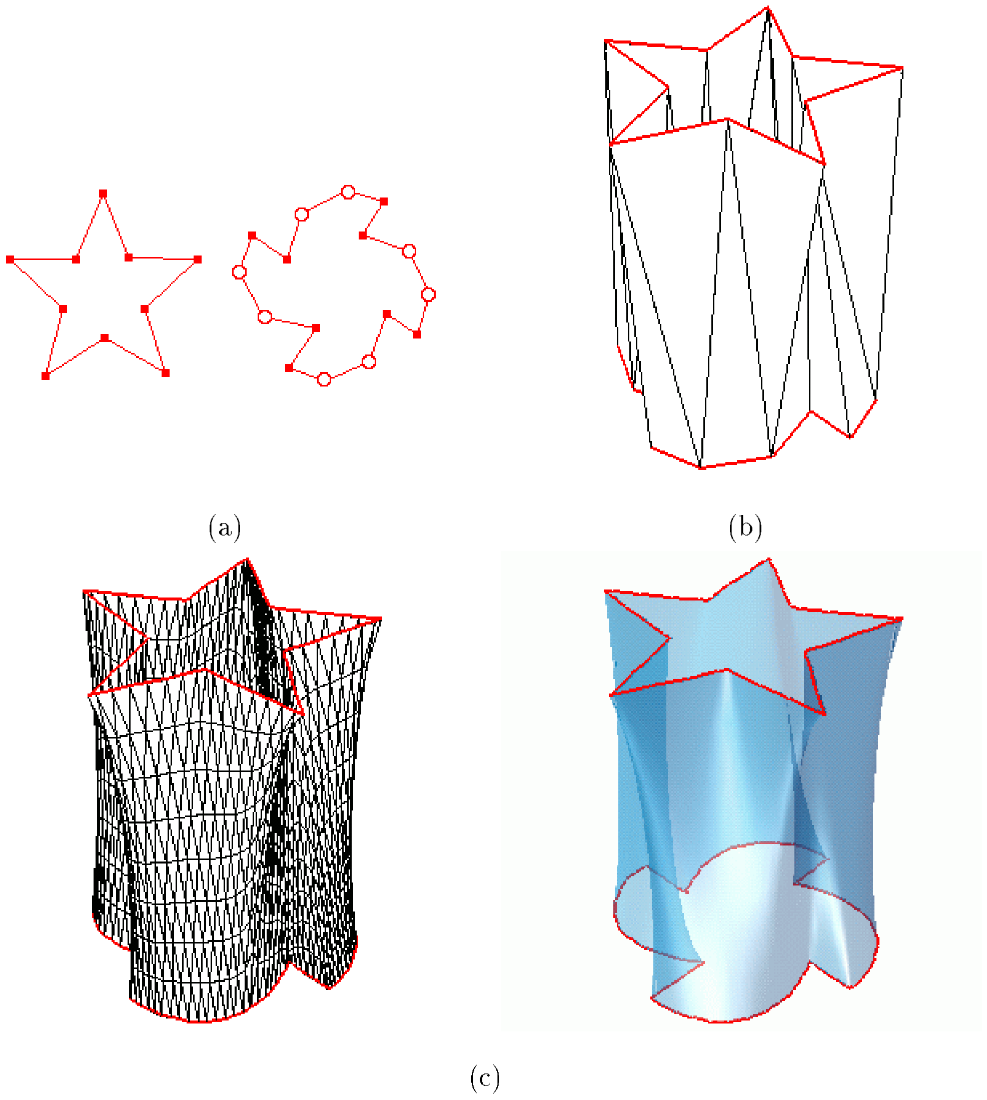
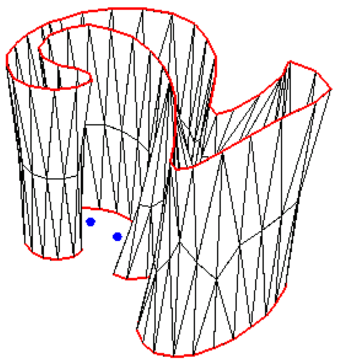
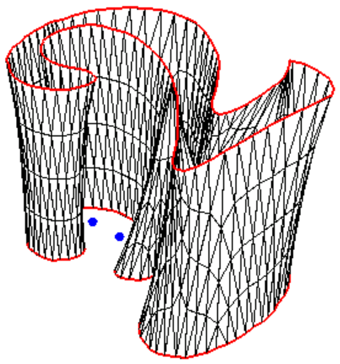


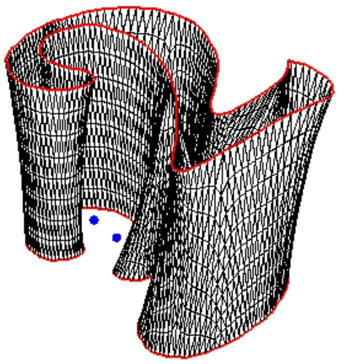
Fig. 11. An example of sharpness control. Source contours (a) and initial I-mesh (b) is subdivided and selectively smoothed (c). The I-mesh after smoothing is shown using two rendering methods to depict its shape.



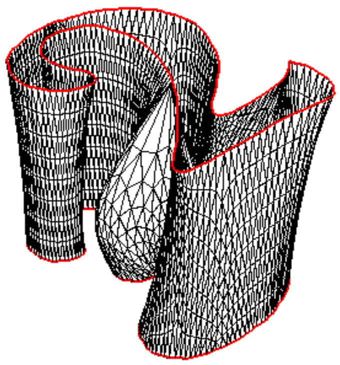
(a) Level 1 mesh and constraints.



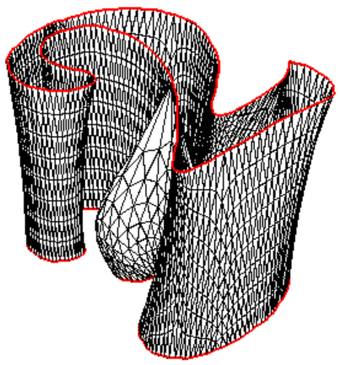
(b) Level 2 mesh and constraints.



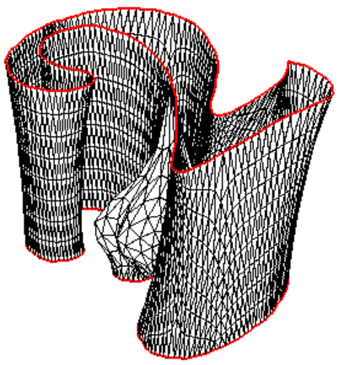
(c) Level 3 mesh and constraints.



(d) After solving for the constraints added at level 1.

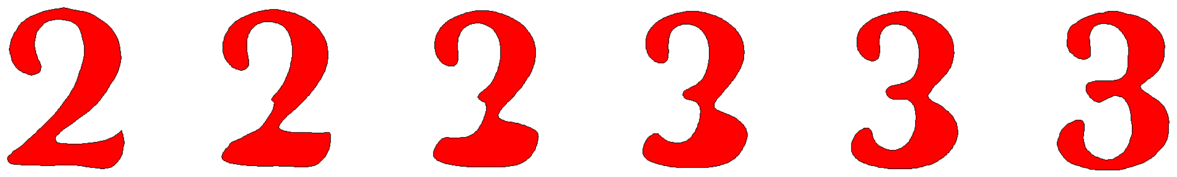


(e) After solving for the constraints added at level 2.

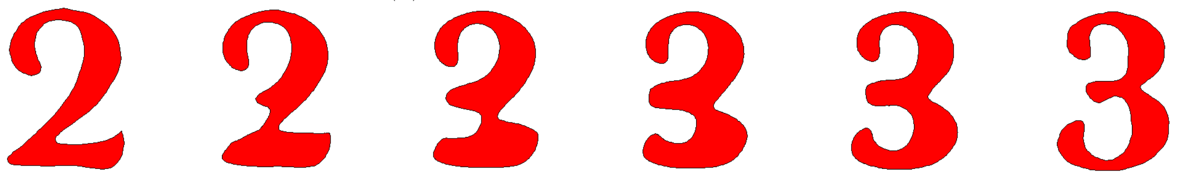


(f) After solving for the constraints added at level 3.

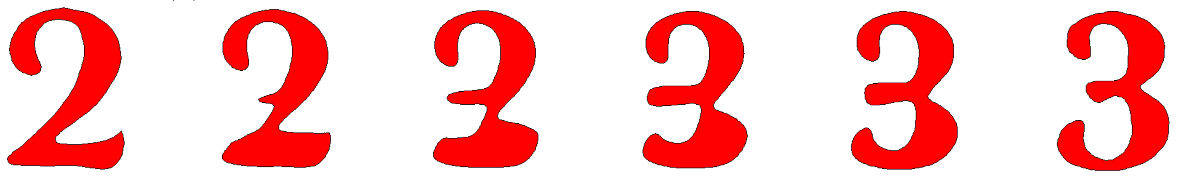
Fig. 12. Multiresolution constraints creates varying surface deformation for a transition effects in shape blending. Constraints attached at level 1 (a), level 2 (b), and level 3 (c) produced deformed meshes of (d), (e), and (f), respectively.



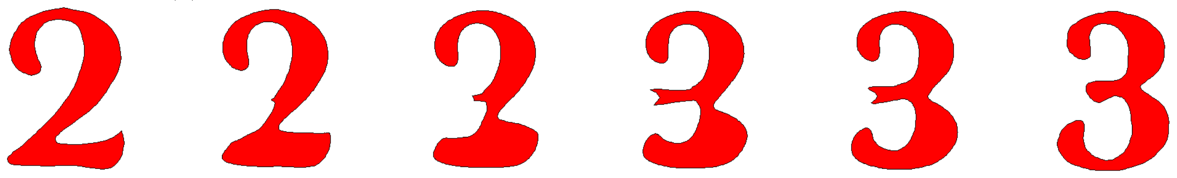
(a) Without the transition effect.



(b) Shape transition effect produced by constraints added at level 1.

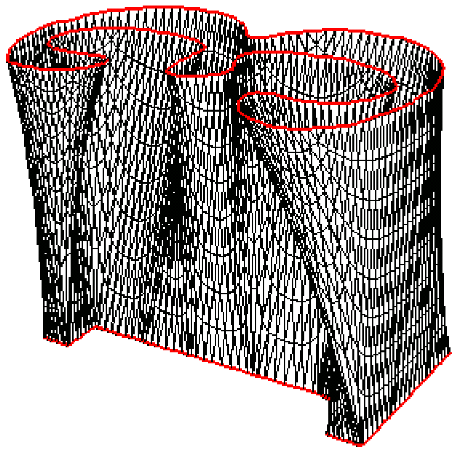


(c) Shape transition effect produced by constraints added at level 2.

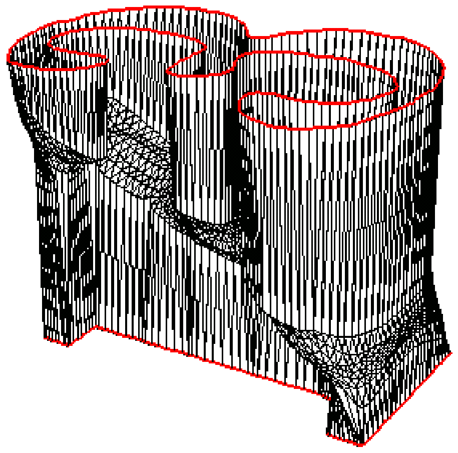


(d) Shape transition effect produced by constraints added at level 3.

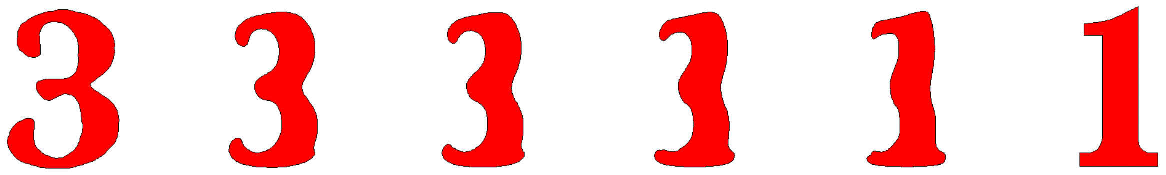
Fig. 13. Enhanced shape transitions sequences are created by the mesh deformations at level 1 (b), level 2 (c), and level 3 (d). The "nose" is pulled out from the letter 2 before it changes into the letter 3. For comparison, the first sequence (a) has no such enhancement.



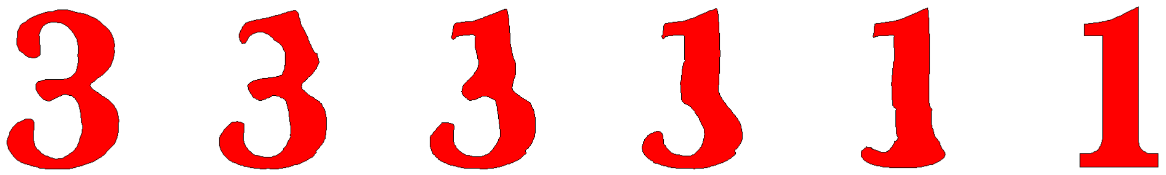
(a) Before warping I-mesh.



(b) After warping I-mesh.

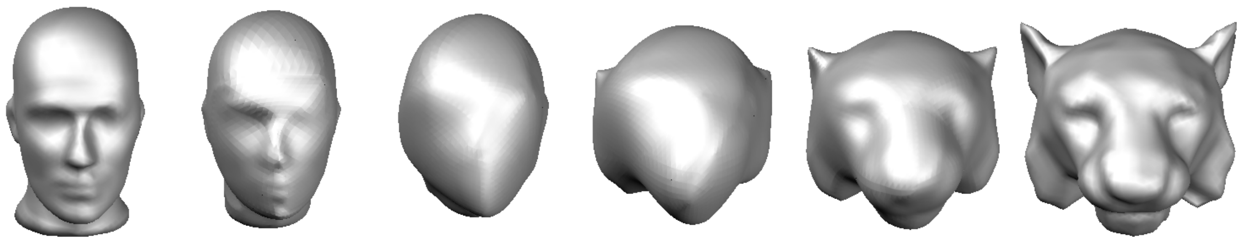


(c) Spatially uniform (original) shape-blending sequence.

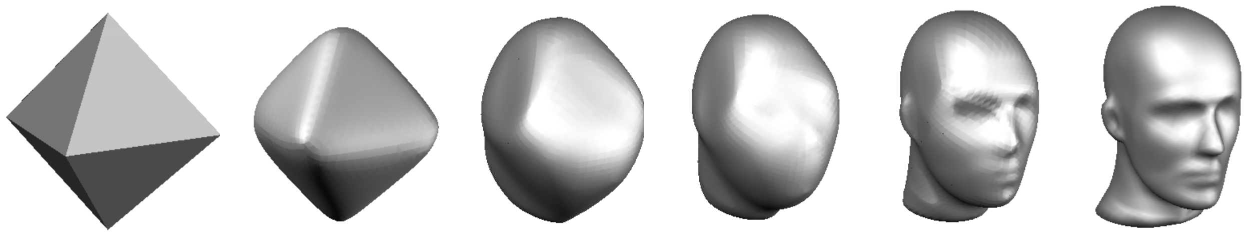


(d) Spatially non-uniform shape-blending sequence.

Fig. 14. Spatially non-uniform transition (d) is realized by warping the I-mesh.

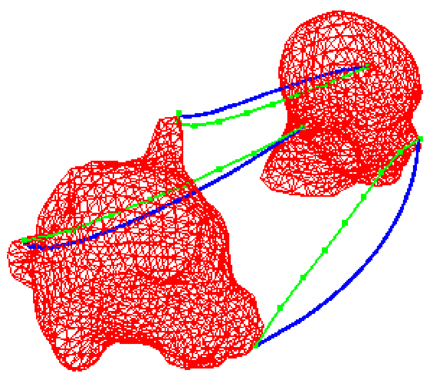


(a) Blending the mannequin head model with the tiger head model.

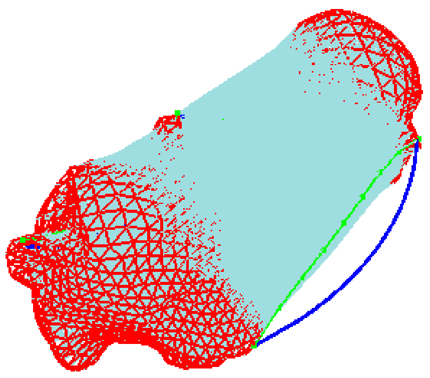


(c) Blending an octahedron with the mannequin head model. Sharp features, that are, vertices and edges, of the source shape are preserved.

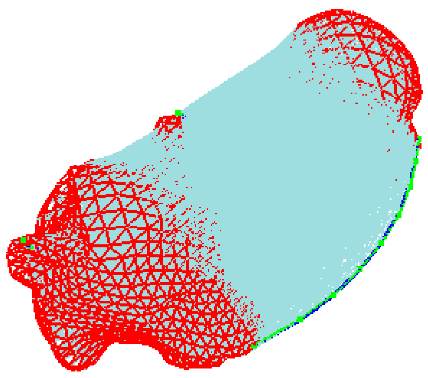
Fig. 15. 3D shape blending sequences created by using our algorithm.



(a)



(b)



(c)

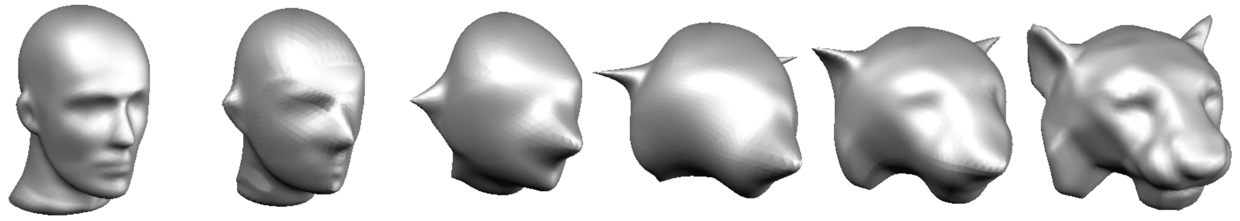
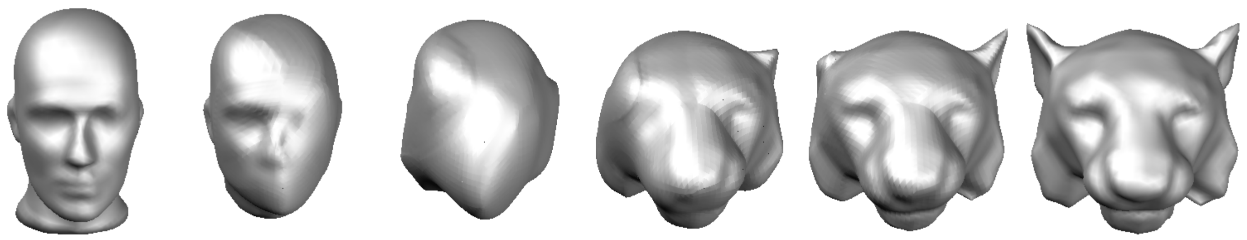
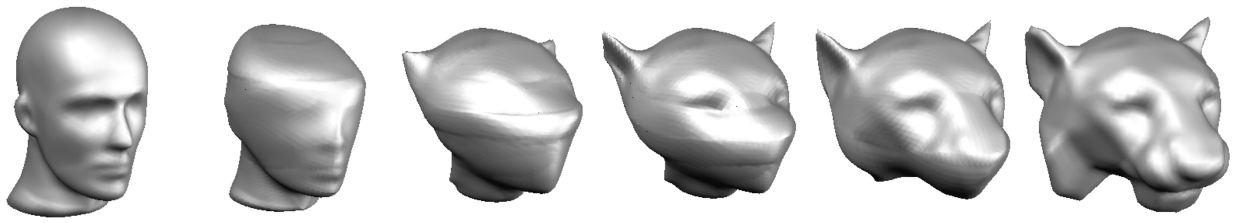


Fig. 16. Feature correspondence between ears and noses in the source meshes shown in (a), (b), and (c) created a shape blending sequence shown in (d), in which the ears and the nose of the mannequin and the tiger figures are related



(a) Only one of the ears of the mannequin is related to its counterpart in the tiger model by using linear geometric constraint for feature correspondence.



(b) Spatially non-uniform transition that propagates from from top to bottom.

Fig. 17. Examples of feature correspondence and spatially non-uniform shape transition.