3D Triangular Mesh Parameterization with Semantic Features Based on Competitive Learning Methods

Shun MATSUI†, Kota AOKI(✉), and Hiroshi NAGAHASHI(✉), Members

SUMMARY In 3D computer graphics, mesh parameterization is a key technique for digital geometry processings such as morphing, shape blending, texture mapping, re-meshing and so on. Most of the previous approaches made use of an identical primitive domain to parameterize a mesh model. In recent works of mesh parameterization, more flexible and attractive methods that can create direct mappings between two meshes have been reported. These mappings are called “cross-parameterization” and typically preserve semantic feature correspondences between target meshes. This paper proposes a novel approach for parameterizing a mesh into another one directly. The main idea of our method is to combine a competitive learning and a least-square mesh techniques. It is enough to give some semantic feature correspondences between target meshes, even if they are in different shapes or in different poses.

key words: cross-parameterization, deformable mesh models, digital geometry processing

1. Introduction

Recent advance in Digital Geometry Processing (DGP) has led to the utilization of 3D objects in various fields. Users can not only display original objects but also generate a new object or animation by transforming or developing them. Mesh parameterization is a fundamental DGP algorithm for such an interactive geometry processing. It maps a triangular mesh of 3D object onto a certain domain and is useful for many geometry processing applications in computer graphics. For example, a lot of previous works for mesh parameterization focused on the morphing as a target application [1]. In the mesh morphing, a mapping is required to establish one-to-one correspondences between vertices of two meshes which have different sizes and different shapes each other. The mesh parameterization constructs such correspondences by mapping two meshes onto an identical domain or by mapping one mesh onto another directly. After constructing the correspondences, we can make a sequence of meshes by interpolating shapes and attributes of both meshes. Besides the morphing, mesh parameterization is an essential technique for many DGP applications such as texture mapping [2], re-meshing [3], 3D object retrieval [4], similarity estimation [5] and so on.

In this paper we propose a novel method for parameterizing a mesh into another one directly. Our parameterization method can generate not only a single correspondence between two meshes but also a unified correspondence among multiple meshes, while preserving their semantic relationships. As for inputs to our system, we only give mesh models and some semantic feature correspondences between them. Moreover, there is no need to make any common base mesh by user’s interaction. These properties enable us to apply the method to various DGP applications.

2. Related Works

Parameterization methods can be roughly classified into some categories according to their parameterization domains. In most previous techniques a plane is used for disk-like meshes [6], [7] and especially a sphere is the most popular domain for parameterizing genus-0 meshes [8], [9]. These techniques have a difficulty for matching semantic feature correspondences between two meshes in the morphing. Mesh parameterization methods that can achieve mappings to satisfy the constraint on semantic feature correspondences will be useful for several DGP applications [10]. A more general approach is to parameterize the meshes onto a common base mesh [11]–[14]. These techniques accelerate the convenience of the parameterization between meshes since the use of a common base mesh is suitable for creating a map from one mesh to another with semantic feature correspondences. But they work well only when the meshes have nearly identical shapes and postures. In addition, users are required to have a priori knowledge for constructing a base mesh to parameterize the both meshes robustly.

To create a direct mapping from one mesh to another is called “cross-parameterization” and is very attractive in this research area. The cross-parameterization does not need any intermediate domain and enables us to map one mesh onto another directly. Allen et al. [15] proposed a parameterization method based on an energy-minimization process for reconstructing multiple shapes with identical connectivity. However, it ensures the robustness of the parameterization only in cases that the meshes have nearly identical shape structures and postures. Kraevoy et al. [16] introduced similar approach using template mesh for recovering surfaces.

As one of more recent works, Kraevoy et al. [17] proposed a new parameterization method without requiring a common base mesh a priori. But their approach does not scale well with regard to the multiple meshes to be consistently parameterized. On the other hand, Schreiner et al. [18] have proposed a direct mapping between two meshes without any intermediate domains. Their method needs high

Copyright © 2008 The Institute of Electronics, Information and Communication Engineers
computational complexity for quality of the mapping and is limited to deal with the parameterization between two meshes. As shown later, our method can handle multiple mesh models.

3. Overview of Our System

In this section we give some basic definitions and an overview of our algorithm for cross-parameterization between meshes. The following notations are used in the description of our algorithm.

A mesh \( M \) is composed of a set \( V \) and an abstract simplicial complex \( K \), and is denoted by \( M = (V, K) \). The abstract simplicial complex \( K \) contains all topological information about \( M \), and is composed of three types of subsets of simplices \( K_v, K_e, \) and \( K_f \). These subsets are: a set of vertices, a set of edges, and a set of faces, respectively. On the other hand, the set \( V \) contains the position vectors of all vertices and it determines the geometry of the mesh. The position vector of a vertex \( i \in K_v \) is denoted as \( v_i \in V \) and has a 3D coordinates.

A mapping \( \Phi \) derived from a certain parameterization changes the positions of vertices in \( V \) while preserving its topological information in \( K \). This leads to the following relation:

\[
\Phi : M = (V, K) \mapsto M' = (V', K),
\]

where \( M' \) denotes a parameterized state of \( M \) mapped by the function \( \Phi \). The set \( V' \) is a set of new position vectors for each vertex of \( M \). The new positions are called “parameters”. When two meshes, that is, a source mesh \( M^S \) and a target mesh \( M^T \) are given, the function \( \Phi \) is defined to create a map \( M' \) from \( M^S \) to \( M^T \). It means that the resulting mesh \( M' \) in Eq. (1) has an identical connectivity of \( M^S \) and its geometry approximates the geometry of \( M^T \) as shown in Fig. 1.

Our approach assumes that some semantic feature correspondences are given in advance. We define a set \( F \) composed of multiple pairs of vertices in order to represent the correspondences between source and target meshes as follows:

\[
F = \{(s_1, t_1), \ldots, (s_m, t_m)\} \subseteq K^S_v \times K^T_v.
\]

A pair \( (s_i, t_i) \) indicates that the vertex \( s_i \) of \( M^S \) must be parameterized onto the vertex \( t_i \) of \( M^T \) semantically, and \( m = |F| \) denotes the number of such pairs. The term \(|\cdot|\) denotes the number of elements in the set.

Then our cross-parameterization algorithm is composed of the following three stages.

3.1 User Interaction

To begin with, a user specifies some vertices that characterize the same semantics in each mesh. For example, as shown in Fig. 2, the selected correspondences between two meshes of “gargoyle” and “bunny” are illustrated. In such meshes we can usually select easy semantic features such as eyes, nose, toes and so on. When two meshes are given, the number of correspondences to be specified generally depends on their complexities and differences between them. These pairs are used for computing quasi-optimal parameters in the following initialization stage.

3.2 Parameter Initialization

We modify the algorithm called least-squares meshes (LSM) proposed by Sorkine et al. ([19]) so that it adapts to the scheme of our parameter initialization. The LSM is very useful for reconstructing the geometry of an original mesh using a sparse set of feature vertices on the mesh. The algorithm is based on a mesh Laplacian expression. Thereby, the resulting mesh is visually smooth and fairly approximates the original geometry.

We use a set of semantic feature correspondences \( F \) to initialize the parameters of \( M^S \) onto \( M^T \). After the initialization stage, the positions of the feature vertices \( s_i \in K^S_v \) given in \( F \) are closely relocated in the positions of \( t_i \in K^T_v \), and the rest vertices are to be relocated in the positions that satisfy their Laplacian equations in a least-squares sense, respectively. The computed parameters are used as the inputs in the next stage.

3.3 Parameter Optimization

We have introduced a deformable model, called Self-organizing Deformable Model ([20]) (SDM) based on a com-
petitive learning method [21] and an external energy minimization. The algorithm maps a mesh onto a target surface by learning the distribution of all points (vertices) on the target surface and by changing the positions of the vertices on the mesh. Differing from other deformable models, our SDM has a distinctive characteristic that the vertices selected as semantic feature points on the SDM are controlled to move toward their corresponding points on the target surface.

The whole mesh are mapped onto the target surface so as to satisfy the given constraints. This is the key point of our algorithm in the semantic cross-parameterization. We optimize the initial parameters obtained in stage 2, by regarding \( M^3 \) and \( M^T \) as a SDM and a target surface, respectively. That is, the initial positions of vertices in SDM’s (stage 3) are supplied with results by the LSM algorithm (stage 2). Because SDM’s tend to be stuck in local minima especially when the shapes or poses of the source and the target are not similar, LSM-based initialization (a global optimization) should be required. The SDM algorithm can parameterize the source mesh onto the target by approximating the shape and pose of the target through the LSM algorithm.

4. LSM-Based Initialization

Sorkine et al. introduced a new mesh expression based on a mesh Laplacian, called least-squares meshes [19], which was previously developed by Mallet [22]. The LSM is a visually smooth and fair approximation of the given feature vertices. We basically use this framework for computing the initial parameters of vertices. We basically use this framework for computing the visually smooth and fair approximation of the given feature vertices.

\[ L_{ij} = \begin{cases} 1, & i = j \\ -1/d_i, & j \in N_i \\ 0, & \text{otherwise} \end{cases} \]  

(5)

Now, we define a new matrix \( A \) with \( L \) and \( F \) as:

\[ A = \begin{pmatrix} L \\ F \end{pmatrix}, \quad F_{ij} = \begin{cases} 1, & j = s_i, \ (s_i, t_i) \in F \\ 0, & \text{otherwise} \end{cases} \]  

(6)

Based on the above definition, we consider the following new linear systems:

\[ Ax = b^x, \ Ay = b^y, \ Az = b^z, \]  

(7)

where, the \((n + m) \times 1\) column vectors \( b^x, b^y, b^z \) are

\[ b^x_k = \begin{cases} 0, & k \leq n \\ x_{t_k-w}, & n < k \leq n + m \end{cases}, \]  

(8)

\[ b^y_k = \begin{cases} 0, & k \leq n \\ y_{t_k-w}, & n < k \leq n + m \end{cases}, \]  

(9)

\[ b^z_k = \begin{cases} 0, & k \leq n \\ z_{t_k-w}, & n < k \leq n + m \end{cases}. \]  

(10)

In Eqs. (8), (9) and (10), \( x_{t_k-w}, y_{t_k-w}, z_{t_k-w} \) are the coordinates of the vertex \( t_k \) on \( M^T \) corresponding to \( s_k \). The system reconstructs the \( x \)-coordinates of all vertices of \( M^S \) so as to minimize the following objective function:

\[ \|Ax - b^x\|^2 = \|Lx\|^2 + \sum_{(s_i, t_i) \in F} \|x_{s_i} - x_{t_i}\|^2. \]  

(11)

We can formulate the same functions about \( y \) and \( z \). Since \( A \) has full rank, the unique analytical solution of Eq. (11) is derived from:

\[ x = (A^T A)^{-1} A^T b. \]  

(12)

Then LSM-based initialization from \( M^3 \) to \( M^T \) is formulated by the minimization problem for finding \( M^{lam} \) and \( \Phi_{lam} \) which satisfy the following equations:

\[ M^{lam} = \Phi_{lam}(M^S, M^T), \]  

(13)

\[ \Phi_{lam}(M^S, M^T) = \arg \min_{s, y, z} E_{lam}(x, y, z), \]  

(14)

where, the actual objective function \( E_{lam}(x, y, z) \) is a linear combination given by

\[ E_{lam}(x, y, z) = \|Ax - b^x\|^2 + \|Ay - b^y\|^2 + \|Az - b^z\|^2. \]  

(15)

The minimization uniquely determines a new position of each vertex \( i \in K^S \). We call the new position a LSM-based parameter and represent the set of all LSM-based parameters as \( V^{lam} \). Then \( V^{lam} \) is denoted by \( (V^{lam}, K^{lam}) \), \( K^{lam} = K^S \) and is called a LSM-based mesh.

Figure 3 shows an example of LSM-based mesh obtained by regarding the “gargoyle” and the “bunny” as a source mesh and a target mesh, respectively. The left model is a LSM-based mesh and has the identical connectivity with \( M^3 \). The right image is a rendering result shaded by the normals of \( M^3 \) mapped on the LSM-based mesh. Figure 4 is also an example of initialization for multiple target meshes. These results show that our LSM-based initialization generates quasi-optimal parameters, but its geometrical shape is not yet a good approximation of \( M^T \).
The number of feature vertices and their positions depend on how much the user incorporates semantic knowledge about objects or how he/she wants to deform them. Our algorithm can possibly parameterize mesh models even if fewer vertices are supplied by the user, but the effect of semantic deformation becomes less.

5. Optimization Using Modified SDM’s

Our initialization stage described in Sect. 4 leads to a new mesh $\mathcal{M}^{lam}$ with quasi-optimal parameters. The LSM-based parameters are inputted to the next stage. In this section, we explain how the parameters are optimized onto a target mesh.

We have introduced a new deformable model [20], called Self-organizing Deformable Model (SDM), whose algorithm is based on a self-organizing map (SOM) [23] and an external energy minimization process. SOM is a well-known algorithm for computing a topology-preserving map, and it also works as an operator of vector mapping or quantization. Our SDM is a mesh and its topology is regarded as a SOM network. Each vertex in the SDM mesh has a 3D position vector which is associated with the reference vector of a unit in the network. On the other hand, each 3D vertex on a target surface is regarded as an input signal to the network. Hence, every vertex’s position vector of the SDM competitively learns the distribution of 3D points on the target surface, and the SDM comes to deform its shape to be looked like the target. In our method, the LSM-based mesh $\mathcal{M}^{lsm}$ and the target mesh $\mathcal{M}^T$ are assumed to be a SDM and a target surface.

Now we introduce a criterion to evaluate the similarity between two meshes. Although there are some measures considered for evaluating the similarity between surfaces, we use the following energy-function based distance:

$$D(M^1, M^2) = \frac{1}{2} \left[ D^0(M^1, M^2) + D^1(M^2, M^1) \right]$$

$$D^0(M^1, M^2) = \frac{1}{3} \sum_{i \in \Gamma^1_s} \sum_{f \in \Omega^1_i} \frac{\text{dist}(c, f)^2}{|\Gamma^1_s||\Omega^1_i|}$$

where two arguments $c$ and $f$ in $\text{dist}(c, f)$ represent a vertex and a face, respectively, and the function $\text{dist}(c, f)$ defines an Euclidean distance between $c$ and the center of $f$. In Eq. (17), $\Gamma^1_i$ is a set of faces which share the vertex $i \in \mathcal{K}^1_s$, and $\Omega^1_i$ is a set of vertices in $\mathcal{K}^1_c$ whose nearest face is $f \in \Gamma^1_i(\subset \mathcal{K}^1_c)$ in the sense of $\text{dist}(c, f)$. $|\Gamma^1_s|$ is the number of faces of $\Gamma^1_i$, and $|\Omega^1_i|$ is the number of vertices $\Omega^1_i$. Although our SDM is basically controlled by a SOM algorithm as mentioned before, it is finally defined as an energy minimization problem of the distance $D(\mathcal{M}^{lam}, \mathcal{M}^T)$.

Now we will enter into details about our modified SDM algorithm. Again, let the source and target meshes be $\mathcal{M}^S = (\mathcal{V}^S, \mathcal{K}^S)$ and $\mathcal{M}^T = (\mathcal{V}^T, \mathcal{K}^T)$, respectively, and the basic algorithm is as follows:

(1) Initialize the time parameter $\tau$ to $\tau = 0$.
(2) If $\tau$ is even, do the steps from (2a) to (2d). Else do the steps from (2a’) to (2d’).
(2a) Select a vertex $c$ randomly from the set $\mathcal{K}^S_c$ of $\mathcal{M}^S$.
(2b) Determine the corresponding vertex $w \in \mathcal{K}^T_c$ by the following equation:

$$w = \begin{cases} s_i, & \text{if } c = t_i \\ \arg\min_{v \in \mathcal{K}^T_c} \|v - p_c\|^2, & \text{otherwise} \end{cases}$$

where the $\| \cdot \|$ is the Euclidean norm and $p_c$ is the position vector of vertex $c$. When $n_w \cdot n_c < 0$ holds, the vertex $w$ is discarded to avoid back-facing errors, where $n_w$ and $n_c$ denote the unit normal vectors at vertices $w$ and $c$, respectively.

(2c) Update the positions of $w$ and its topological neighborhoods $w'$ as follows:

$$v_w \leftarrow v_w + \epsilon(\tau)(p_c - v_w),$$

$$v_{w'} \leftarrow v_{w'} + \epsilon(\tau)\lambda(h, \tau)(p_c - v_{w'}).$$

where the topological neighborhoods of a vertex are the vertices that share edges (and also faces) with it.

(2d) Repeat the steps from (2a) to (2c) $|\mathcal{K}^T_c|$ times.

(2’a) Select a vertex $w$ randomly from the set $\mathcal{K}^S_c$ of $\mathcal{M}^S$. 

Fig. 3 \ LSM-based parameter initialization. After this stage, $\mathcal{M}^S$ obtains quasi-optimal parameters to $\mathcal{M}^T$. These parameters are used in the next optimization stage.

Fig. 4 \ Initialization for multiple target meshes. Using sparse feature correspondences, we can obtain quasi-optimal parameters for each target mesh. Our parameterization method can deal with multiple meshes.
(2’b) Determine the corresponding vertex \( c \in \mathcal{K}_\mathcal{F}^T \) by the following equation:

\[

c = \begin{cases} 
  t_i, & \text{if } w = s_i \\
  \arg \min_{i \in \mathcal{K}_\mathcal{F}^s} \| p_w - p_i \|^2, & \text{otherwise}
\end{cases}
\]

(2’c) Update the positions of \( w \) and its topological neighborhoods \( w' \) according to Eqs. (19) and (20).

(2’d) Repeat the steps from (2’a) to (2’c) \(|\mathcal{K}_\mathcal{F}^s|\) times.

(3) \( \tau \leftarrow \tau + 1 \).

(4) If \( \tau < T \), go to step (2). Otherwise, go to step (5).

(5) Minimize \( D(M_\mathcal{F}, M^T) \) directly by the steepest descent method (refer to the previous work [20]).

By repeating the steps from (2a) to (2d) and from (2’a) to (2’d) alternatively, our SDM is adapted to the target surface so as to keep the distance relationship between each vertex in \( \mathcal{K}_\mathcal{F}^s \) and its nearest control point in \( \mathcal{K}_\mathcal{F}^T \). This prevents the algorithm from generating undesirable large meshes. In the beginning of step (5), the mesh \( M^T \) considerably approximates to the target surface, but is not yet optimized in the sense of energy criterion defined in Eq. (16). Hence the energy minimization of step (5) is explicitly performed.

The topological distance between \( w \) and \( w' \) equals to the number of possible edges connecting them, and the neighborhood vertices whose distances from \( w \) are less than \( L_s \) are moved according to Eq. (20). In our experiments, \( L_s \) is kept to 4. In Eqs. (19) and (20), \( \epsilon(\tau) \) is a learning rate which defines the adaptation of vertex \( w \) or \( w' \) toward the vertex \( c \), and is defined by

\[
\epsilon(\tau) = \epsilon_0 \left( \frac{\epsilon_f}{\epsilon_i} \right)^{\tau/T}.
\]

The neighborhood function \( \lambda(h, \tau) \) means an adaptation rate for the movement of vertices and is a decreasing function defined as:

\[
\lambda(h, \tau) = \exp \left\{ -\frac{1}{2} \cdot \frac{h(w', w, c)^2}{\sigma(\tau)^2} \right\},
\]

where \( \sigma(\tau) \) and \( h(w', w, c) \) are given as follows:

\[
\sigma(\tau) = \sigma_f \left( \frac{\sigma_f}{\sigma_i} \right)^{\tau/T},
\]

\[
h(w', w, c) = \frac{\sqrt{\| p_c - v_w \|^2}}{\sqrt{\| p_c - v_{w'} \|^2}}.
\]

The definition of Eq. (25) leads to the quasi-conformality of the SDM mapping [24]. We set the parameters as follows:

\[
\sigma_i = 0.5, \quad \sigma_f = 0.01, \quad \epsilon_0 = 2.0, \quad \epsilon_f = 0.05, \quad T = 15.
\]

The SDM-based optimization from \( M_{\text{lsm}} \) to \( M^T \) is formulated by the minimization problem to find \( M_{\text{lsm}} \) (called SDM-based mesh) and \( \Phi_{\text{lsm}} \) which satisfy the following equations:

\[
M_{\text{lsm}} = \Phi_{\text{lsm}} \left( M_{\text{lsm}}, M^T \right),
\]

\[
\Phi_{\text{lsm}} \left( M_{\text{lsm}}, M^T \right) = \arg \min_{x, y, z} D(M_{\text{lsm}}, M^T).
\]

From these optimization steps, \( M^T \) is mapped onto \( M^T \). Consequently, the whole mapping \( \Phi \) in our algorithm is given as

\[
\Phi = \Phi_{\text{lsm}} \cdot \Phi_{\text{lsm}}.
\]

Figure 5 is an optimization result for the mesh \( M_{\text{lsm}} \) as shown in Fig. 3. In this example, the “gargoyle” and the “bunny” play the role of source and target meshes, respectively. The left mesh is a new one with the geometry of “bunny” and has the connectivity of “gargoyle”. The right model is also a rendering result shaded by the normals of “gargoyle”.

5.1 Extension of the Algorithm

When we want to construct a finer mesh model for a given object, the number of vertices and faces may rapidly increase according to the geometrical complexity of the object. This sometimes causes a problem like the shortage of memory or expensive computation time for matrix calculations. In order to cope with a huge mesh model, we introduce a hierarchical parameterization framework based on a progressive mesh representation [25]. The algorithm extended is as follows:

1. Represent the source mesh \( M^F \) in a progressive form and reconstruct a mesh \( M_{\text{prog}} = (V_{\text{prog}}, \mathcal{K}_{\text{prog}}) \) from the progressive representation, which has lower resolution than that of \( M^F \). The resolution of the new mesh generally depends on a computational environment.

2. If a vertex \( c \in \mathcal{K}_{\text{prog}}^s \) is not contained in \( \mathcal{K}_{\text{prog}}^T \), find a patch \( f \in \mathcal{K}_{\text{prog}}^T \) which is nearest to the vertex \( c \). Assuming the nearest patch is \( f = (i, j, k) \in \mathcal{K}_{\text{prog}}^T \), project the vertex \( c \) orthogonally onto the plane spanned by the patch \( f \). Let the projection of \( v_c \) on the plane be \( v'_c \), and calculate two values \( \alpha \) and \( \beta \) so that the following equation holds:

\[
v'_c = \alpha(v_k - v_i) + \beta(v_j - v_i) + v_i,
\]

where, \( v_i, v_j, v_k \) are positions of vertices \( i, j \) and \( k \), respectively.
3. Parameterize the mesh \( M^\text{prog} \) by
\[
\Phi_{\text{sdm}} \left( \Phi_{\text{lsm}} \left( M^\text{prog}, M^T \right), M^T \right)
\]  

4. Suppose that \( c \in \mathcal{K}^S \) is not contained in \( \mathcal{K}^\text{prog} \) and its nearest face of \( M^\text{prog} \) is \( f(=\{i,j,k\}) \in \mathcal{K}^\text{prog} \). Let the positions of vertices \( v_i, v_j, v_k \) be parameterized to \( v_i^{\text{new}}, v_j^{\text{new}}, v_k^{\text{new}} \), respectively, and also let the two values calculated by Eq. (29) for \( c \) be \( \alpha_c \) and \( \beta_c \). Then, a new parameter \( \theta^{\text{new}}_c \) for \( c \in \mathcal{K}^S \) is determined as follows:
\[
\theta^{\text{new}}_c = \alpha_c (v_i^{\text{new}} - v_c^{\text{new}}) + \beta_c (v_j^{\text{new}} - v_c^{\text{new}}) + v_c^{\text{new}}
\]

5. From the above steps, new parameters for every \( v_i \in \mathcal{V}^S, i \in \mathcal{K}^S \) can be obtained. We let the set of all new parameters be \( \mathcal{V}^{\text{new}} \) and a new mesh \( M^{\text{new}} \) be \( \left( \mathcal{V}^{\text{new}}, \mathcal{K}^S \right) \). This mesh may almost be parameterized to \( M^T \), but it can be re-optimized by \( \Phi_{\text{sdm}}(M^{\text{new}}, M^T) \), if necessary.

6. Results and Applications

In order to verify the effectiveness of our proposed method for cross-parameterization, we performed several experiments for some triangular meshes. All of the experiments were done on Pentium IV 2.8 GHz and 1 GB memory. In this section, we not only discuss the effectiveness and impact of our algorithm but also show several examples of DGP applications.

6.1 Cross-Parameterization

As shown in the above sections, Figs. 2, 3, and 5 represent the flow of our algorithm for cross-parameterization and show examples of two meshes \( M^S \) and \( M^T \) to be cross-parameterized, an LSM-based mesh \( M^{\text{lsm}} \), and an SDM-based mesh \( M^{\text{sdm}} \), respectively.

We show two additional examples in Fig. 6. In these results “armadillo” and “lion” are mapped onto “human” and “dinosaur”, respectively. In the figure, the rightmost two images show the parameterized meshes shaded by the normals of corresponding source meshes. These results show that our cross-parameterization algorithm is independent on the shapes or postures of the meshes and can parameterize without any intermediate domains, while preserving the semantic feature correspondences.

The details of the mesh data used for these exemplifications are given in Table 1. The calculation time and the approximation error for each result are also shown in Table 2. In Table 2, two columns indexed with “Init.” and “Opt.” give the total times (second) for numerical computations in the stages of initialization and optimization, respectively. On the other hand, the “Dist.” column indicates a kind of parameterization error which equals to the ratio of the distance between \( M^{\text{sdm}} \) and \( M^T \) calculated by Eq. (16) to the diagonal length of a bounding box of \( M^T \). The rightmost column indexed with “m” gives the number of feature vertices manually specified. In these experiments, each linear system in the initialization stage was solved directly, and this is a major reason for high computation cost. We will be able to save much computation time in this stage by using suitable numerical calculation method such as the LU factorization. As for parameterization errors, the example of “lion/dinosaur” has a big error compared with others. The error seems to be caused by the big difference in the numbers of meshes. When a source model has much less faces than a target one, the number of the faces is not enough to approximate the shape of the target. Then, our method may also need an adaptive subdivision scheme for improving the parameterization accuracy.

6.2 Shape Interpolation

Establishing one-to-one mappings between different meshes is the first step for mesh morphing. Figure 7 shows an example of the face morphing sequence between two meshes with textures. In this experiment, 43 feature correspondences were given manually to perform the cross-parameterization. Then, RGB colors of the target face were extracted and attached to the parameterized face. We generated interme-
Fig. 7  Morphing sequence between two facial meshes. Semantic feature correspondences enable us to make natural morphing sequence.

Fig. 8  Shape blending for two or more than two meshes. Our algorithm can parameterize multiple meshes with identical connectivity.

diate meshes from the source and its parameterized faces by linearly interpolating all vertices and their colors. This result demonstrates that our approach ensures semantic feature correspondences between meshes in the stage of parameterization, and we can make a natural animation with some correspondences such as eye to eye, nose to nose, mouth to mouth and so on.

Figure 8 shows two examples of the different application of our algorithm to a spatial shape blending. The upper example is a shape combination of lion’s upper body and horse’s lower body. This interpolation is very useful for mesh editing. The lower one is an example of multiple shape blending of three facial meshes. This example shows that our approach can parameterize the geometry of multiple meshes with identical connectivity and is useful for analyzing principal mesh components [15].

6.3 Texture Transfer

Several attributes such as normals, curvatures and textures can also be transferred by a map derived from the cross-parameterization. Especially when the texture domain is a 2D image, general approaches for texture transfer use a disk-like mesh as a target. Although the cross-parameterization can make a direct transfer from one mesh to another, the both meshes must have identical genus and sometimes they are required to have similar geometry. As our approach adopts two-step adaptation, it can cope with meshes that have considerably different shapes and can transfer textures between them. Figure 9 shows an example of texture transfer from “human” to “horse” based on our cross-parameterization.

In this example, we do not consider the distortion and discontinuities of their textures. Some measures of texture error like texture stretch and deviation error proposed in [2], must be considered in order to obtain accurate textured mesh parameterization.

6.4 Deformation Transfer

The motion for mesh animation, which is generated from a sequence of mesh frames, is represented as a set of affine transformations of all vertices. Figure 10 is an example of deformation transfer between meshes [26]. By mapping a “lioness” mesh onto “cat” mesh, we obtain one-to-one or one-to-many mappings for each face on the meshes. In this example, the “lioness” at left side in the lower figure is cross-parameterized to the “cat” at left side in the upper figure. We compute affine transformations derived from the “cat” animation, and apply them to the corresponding “lioness” vertices. As a result, a new mesh at right side in the lower figure is created.
As shown in above examples, the proposed method has a high potentiality for many DGP applications in a unified framework. The independency on the shape and posture of the meshes may guide more interactive mesh processing.

7. Discussion

(1) Feature correspondences

Sumner and Popović [26] propose a method for deformation transfer, where the user supplies a set of marker points which are enforced as constraints. In their algorithm the source and target deformations are represented as affine transformations, while our method uses both mappings among meshes and affine transformations of the target to transfer deformations to the source. The method of Asirvatham et al. [27] also utilizes a set of user-specified feature points and parameterizes several genus-zero meshes onto a common spherical domain, but our approach does not require base domains and can be possibly used to parameterize genus-n meshes.

(2) Mesh genus

All the experiments in this paper are performed on genus-zero mesh models, however, our algorithm is not limited logically to the application to genus-zero meshes. Cross-parameterization between meshes with a same genus greater than zero will require more carefully specified parameters and appropriate feature correspondences between them.

(3) Fold-over

In our method, the fold-over also happens to occur. When it can be seen, we try to restart our algorithm after reconfiguring the parameters of the SOM algorithm, that is, the learning rate and the neighborhood function are reconfigured. More sophisticated or automatic approaches will be desired.

8. Conclusion

In this paper we have introduced a novel algorithm for mesh cross-parameterization which guarantees semantic feature correspondences between source and target meshes. While most of the previous works need to make a base mesh by user’s intervention or have some restrictions on the types of meshes, our algorithm can compute cross-parameterization without such restrictions and ensures the robustness of the parameterization even when the meshes have nearly the different shape structures and postures. The experimental results show the effectiveness for many DGP applications.

There are several avenues to improve the current algorithm for future work. First, the approach needs to make some feature correspondences for meshes. Therefore, we think that an automatic feature detection scheme and its evaluation will be important for robust cross-parameterizations. Second, new functions will be required to evaluate the parameters’ quality like angle distortion to a face. Finally, we will apply an adaptive subdivision scheme and multi-resolution technique to the present algorithm for getting a better approximation of geometry and dealing with a large mesh data directly.

We would like to develop new DGP applications based on the proposed cross-parameterization method.

Acknowledgment

We would like to thank Stanford university, University of Washington, Massachusetts Institute of technology and Cyberware for sharing the data sets of 3D triangular mesh models in our experiments.

References


Shun Matsui received the M.E. degree from Tokyo Institute of Technology in 2007. His research interests include 3D modeling, computer graphics and geometric modeling.

Kota Aoki received the BE degree and the PhD degree from Tokyo Institute of Technology in 2001 and in 2006, respectively. He is currently an assistant professor in Imaging Science and Engineering Laboratory at Tokyo Institute of Technology. His research interests include computer vision, computer graphics, and statistical modeling.

Hiroshi Nagahashi received his B.E. and Ph.D. degrees from Tokyo Institute of Technology in 1975 and 1980, respectively. Since 1990 he has been with Imaging Science and Engineering Laboratory, Tokyo Institute of Technology, where he is currently a professor. His research interests include image processing, computer vision, computer graphics and geometric modeling.